Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss Parth Sawant

of IT Department, Semester V with

Roll No. 112 has completed a course of the necessary

experiments in the subject Adv Dev Ops under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Agarwal

Teacher In- Charge

Head of the Department

Date 20/10/23

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	DATE PAGE NO.	DATE	TEACHERS SIGN.
1)	To study and perform the setup of AWS EC2 service and launch an EC2 instance	18/7/23	18/7/23	
2)	To study & perform the setup of AWS Cloud9 service & launch a python program in cloud 9	25/7/23		
3)	To study AWS S3 service & Create a bucket for hosting static web application	1/8/23		Agarwal 20/10/23
4)	To study codepipeline & deploy a web application using codepipeline	2/8/23		
5)	To study the kubernetes cluster architecture	12/9/23		
6)	To understand terraform lifecycle & to build change and destroy AWS infrastructure using Terraform	22/8/23		
7)	To perform Static analysis on python programs using Sonar Qube SAST process	29/8/23		
8)	To understand Continous monitoring using Nagios	26/8/23		
9)	To understand AWS lambda Function & Create a lambda Function using python to log "An image has been added"	31/8/23		Agarwal 20/10/23
10)	To create AWS lambda Function using Python for adding data to DynamoDB database	5/9/23		
11)	Written Assignment 1	23/9/23		
12)	Written Assignment 2	3/10/23		

ASSIGNMENT #1

AIM: To create an EC2 machine/instance using AWS
LO: LO1

THEORY: Amazon Elastic Compute Cloud is a part of Amazon.com's cloud-computing platform, Amazon Web Services, that allows users to rent virtual computers on which to run their own computer applications. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction.

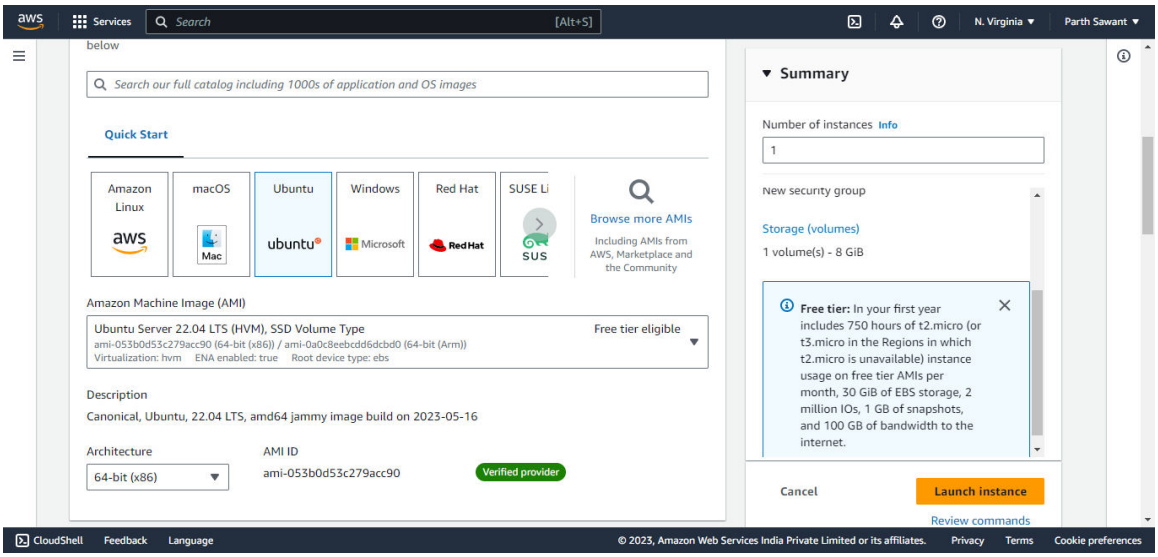
The following are the features of AWS:

- Flexibility
- Cost-effective
- Scalable and elastic
- Secure
- Experienced

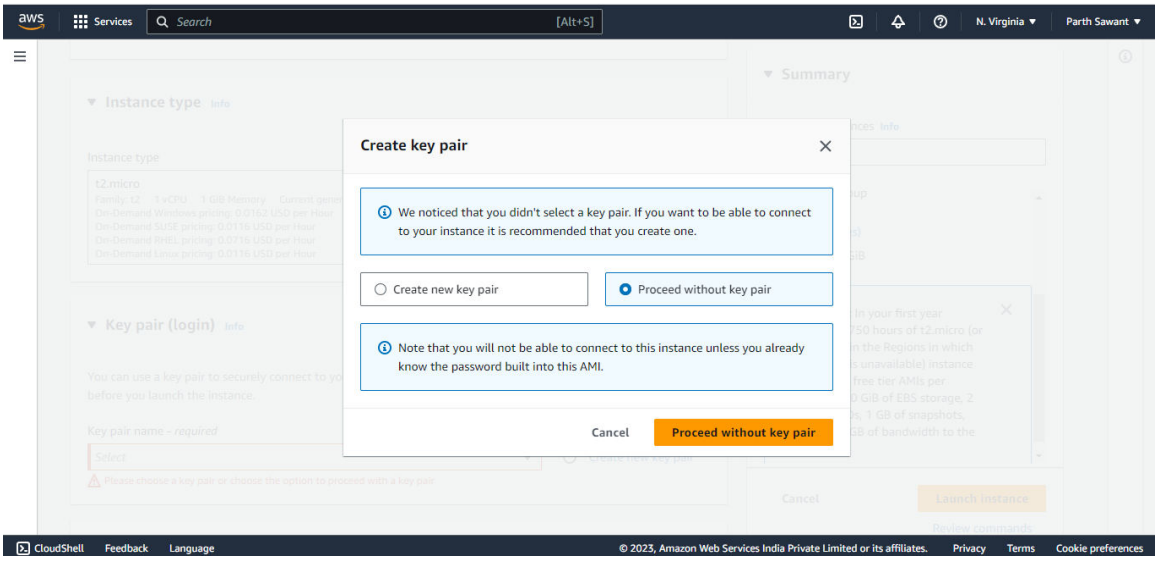
STEPS:

1: Select the cloud computing platform you want to work on

Parth Sawant 112

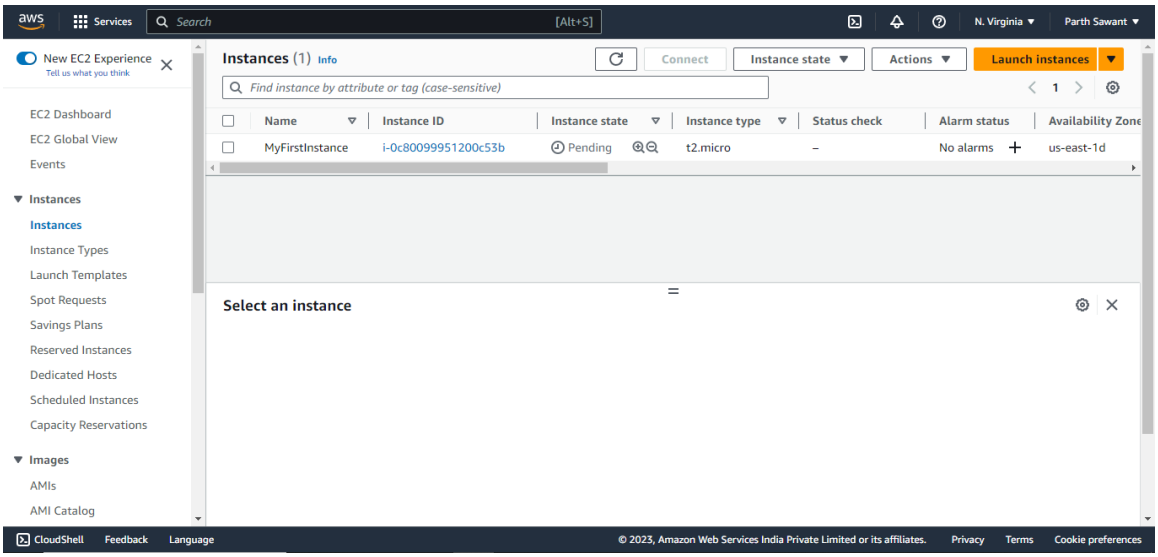


2: Select Proceed withpoud key pair to continue

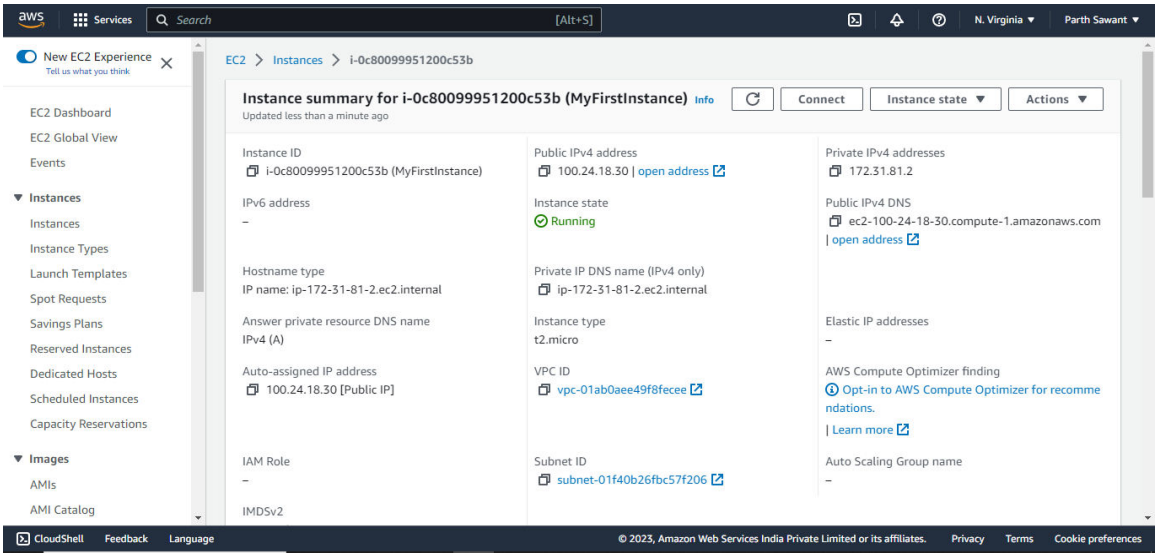


3: The Instance is created

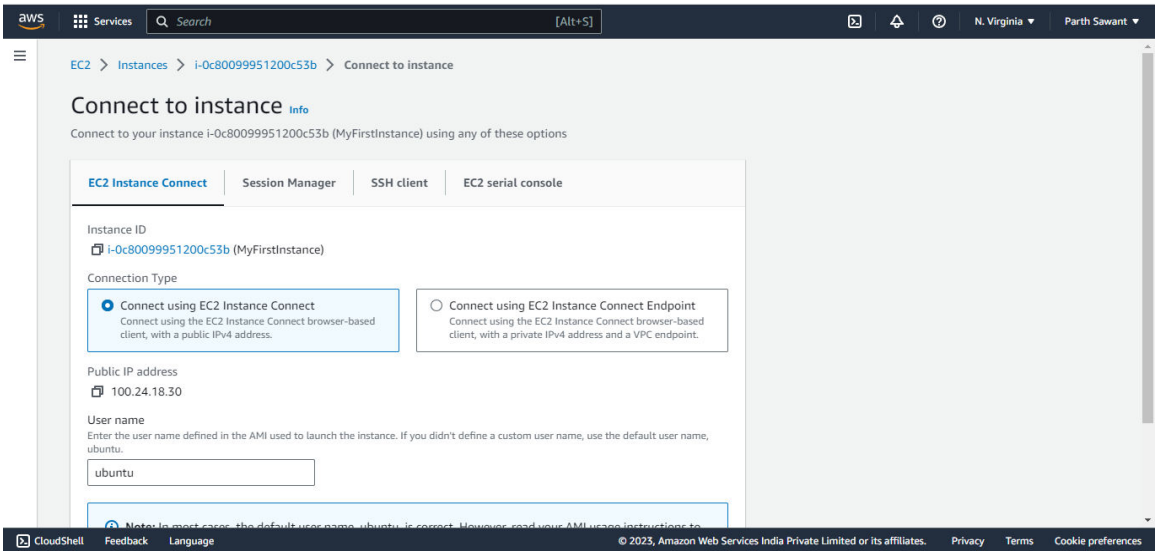
Parth Sawant 112



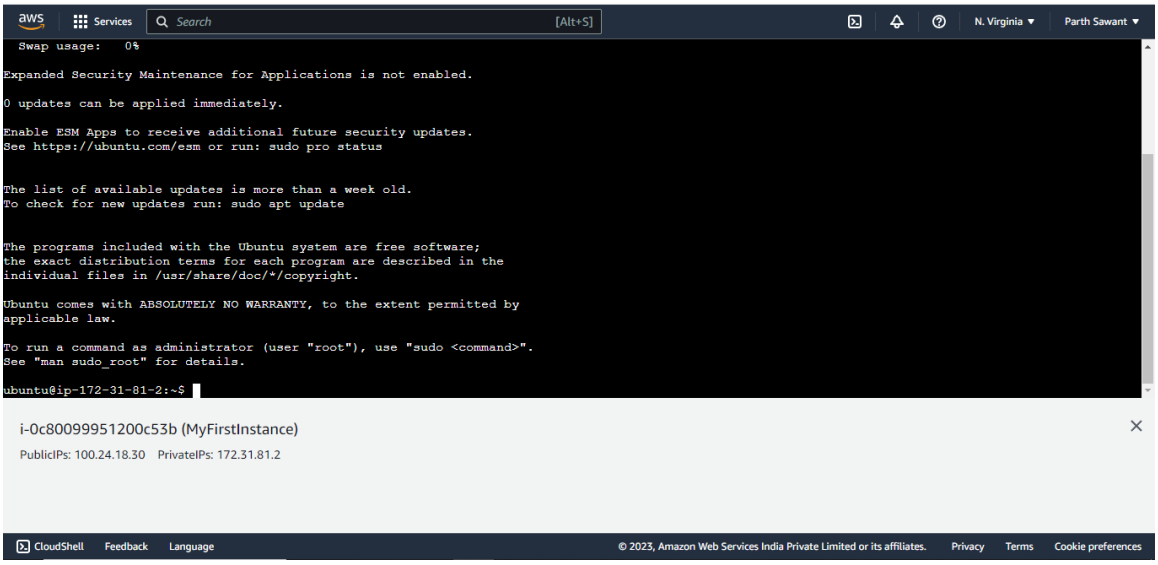
4: We have the instance Id and the current state



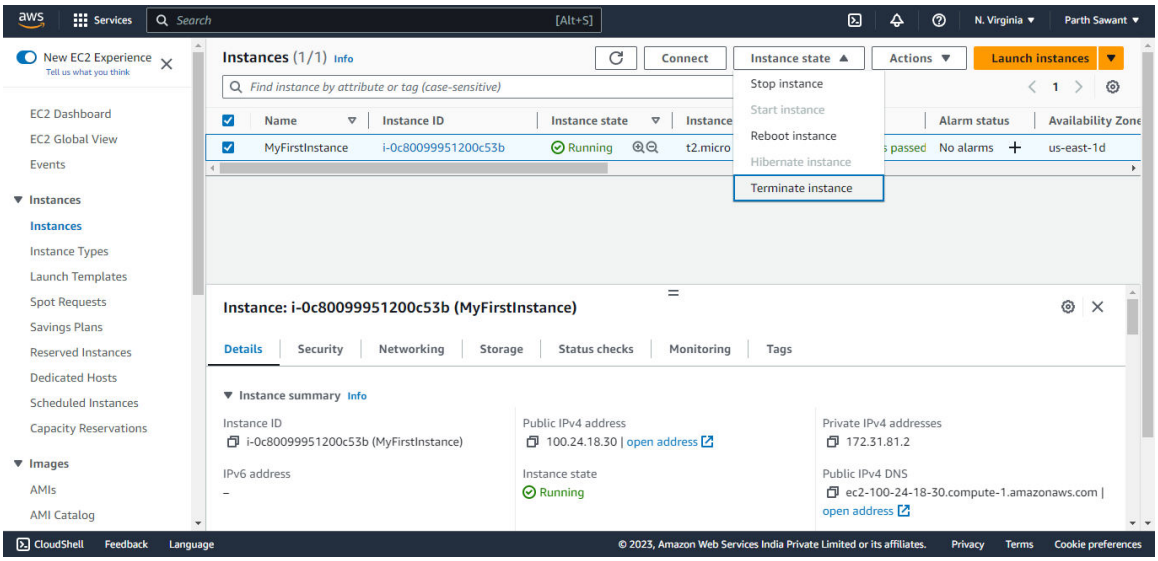
Parth Sawant 112



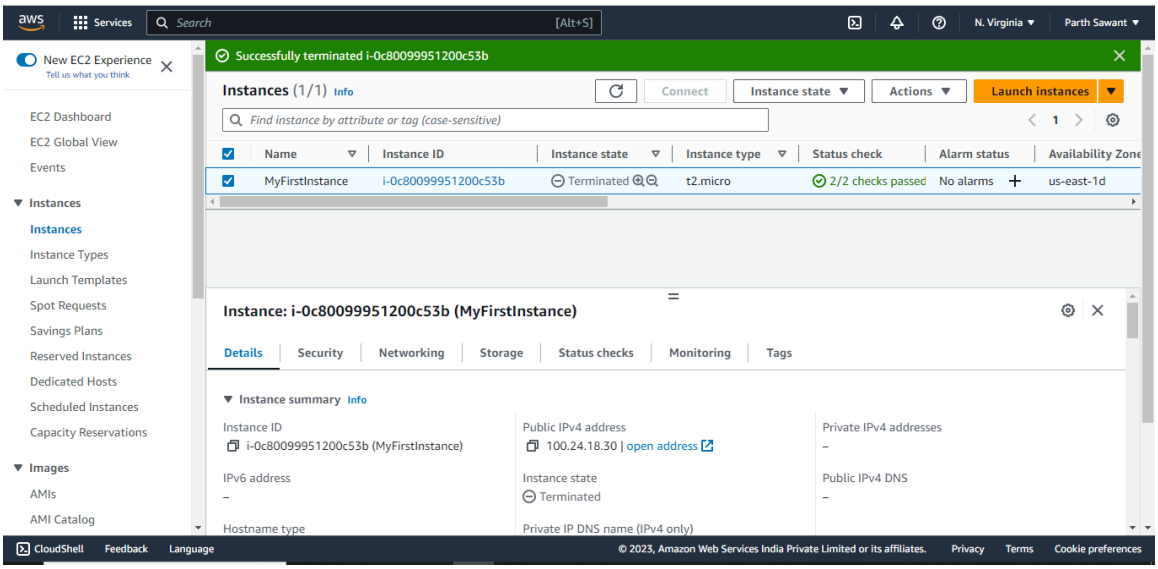
5: After connecting to instance succesfully the ubuntu portal is ready



6: To terminate the Instance



7: Termination successful



CONCLUSION: Understanding the concept of EC2 and to create an EC2 machine/instance using AWS was performed successfully.

ASSIGNMENT #2

AIM: To understand the benefits of Cloud infrastructure and setup AWS Cloud9 IDE and Perform Collaboration demonstration.

LO: LO1

THEORY:

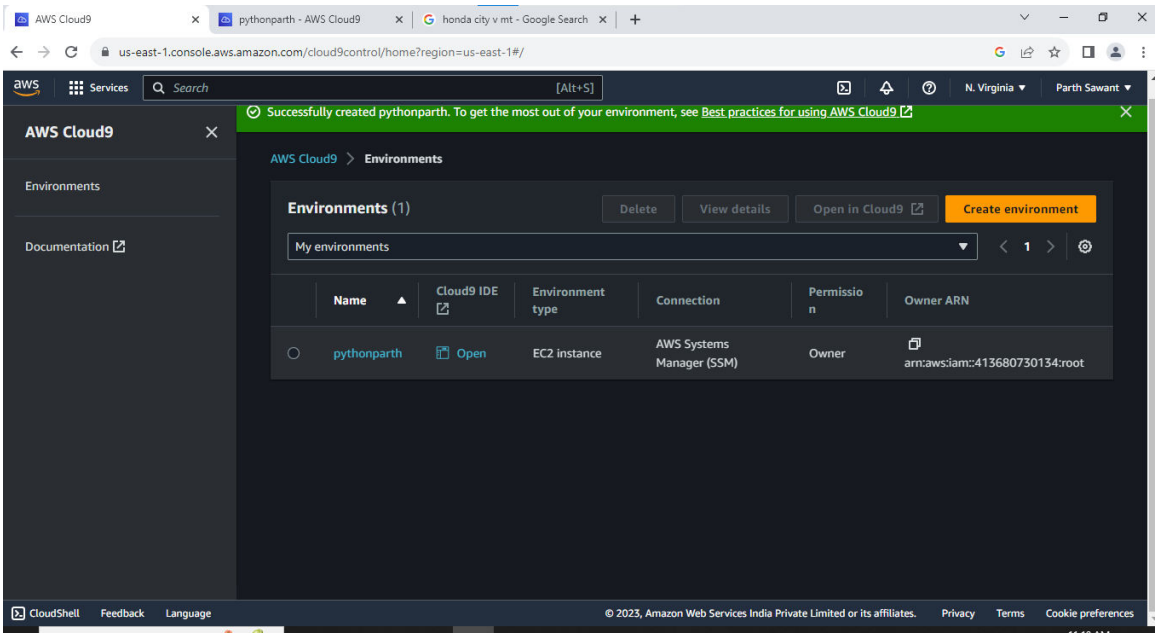
Cloud9 IDE is an Online IDE (integrated development environment), published as open source from version 2.0, until version 3.0. It supports multiple programming languages, including C, C++, PHP, Ruby, Perl, Python, JavaScript with Node.js, and Go.

It is written almost entirely in JavaScript, and uses Node.js on the back-end. The editor component uses Ace.

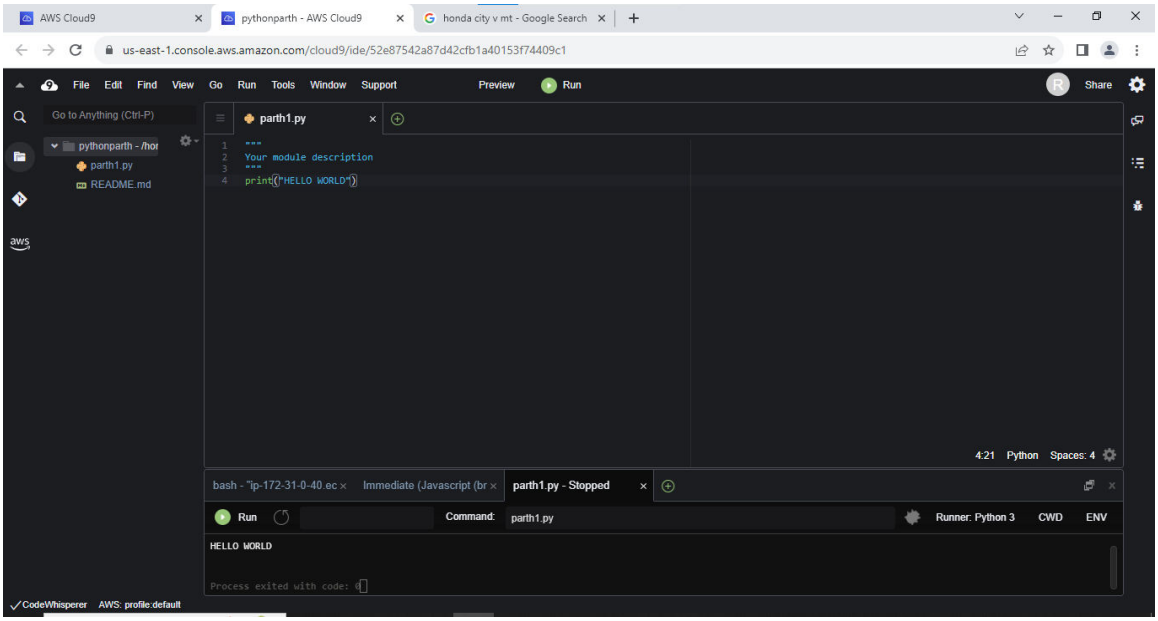
Cloud9 was acquired by Amazon in July 2016[4] and became a part of Amazon Web Services (AWS). New users may only use the Cloud9 service through an AWS account.[5]

From March 2018, existing accounts on Cloud9's original website could be used, but new accounts could not be created. On April 2, 2019, Cloud9 announced that users would not be able to create new and use old workspaces on c9.io (aka original version, not Amazon Cloud9) after June 30, 2019.

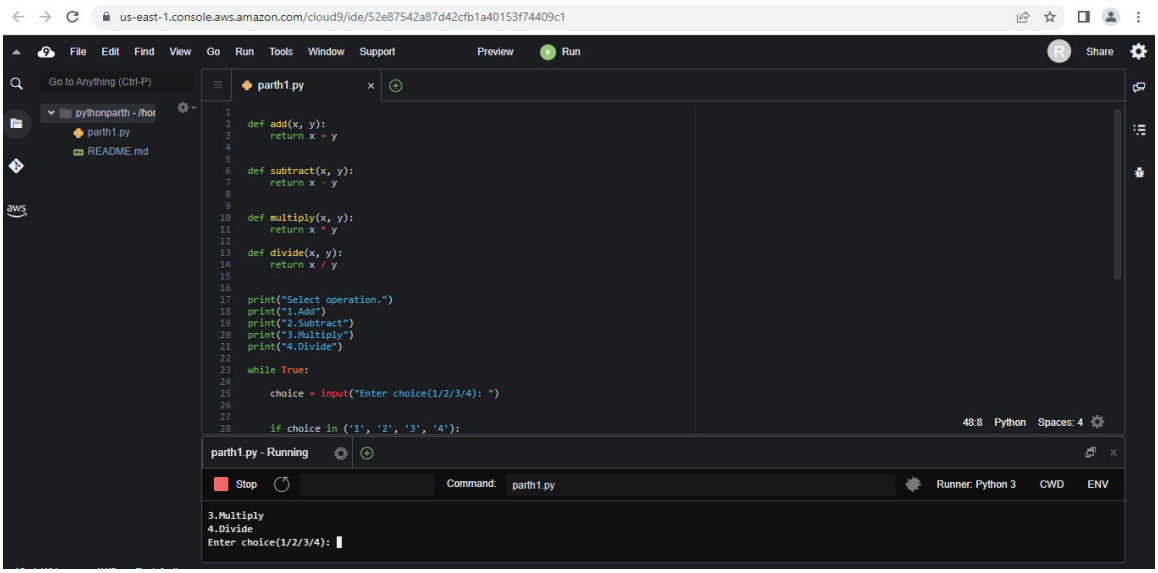
1: Creating cloud9



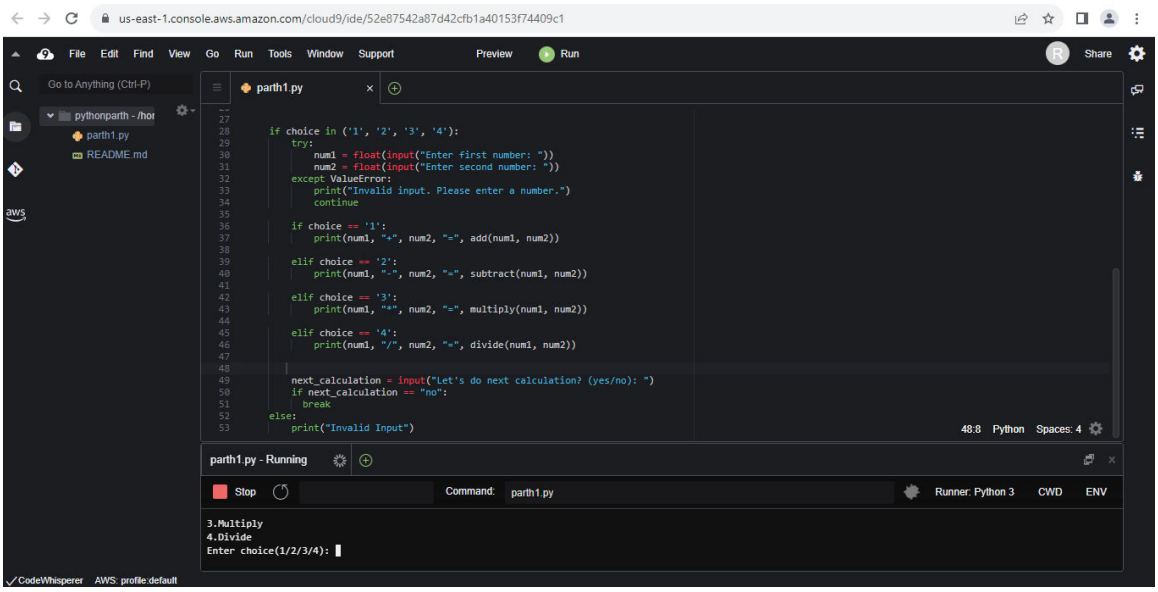
2: Testing demo Python code



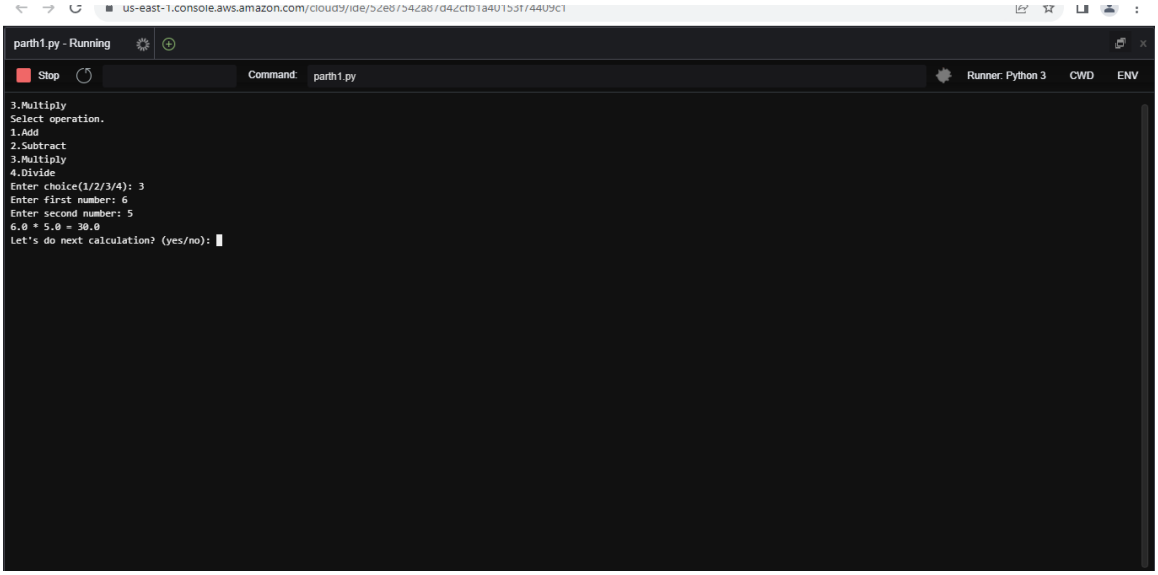
3: Calculator code part 1



4: Calculator code part 2



5: Output



CONCLUSION: The benefits of Cloud infrastructure and setup AWS Cloud9 IDE and Perform Collaboration demonstration was successfully executed.

Parth_112

AdvDevOps Assignment 5

Aim: To understand Kubernetes Cluster Architecture

LO: LO1, LO2

Theory:

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

In a Kubernetes cluster, several services and components run on each node to manage various aspects of container orchestration, networking, and communication. Here are some of the key services and their roles:

Kubelet: The Kubelet is an agent that runs on each node and communicates with the Kubernetes Control Plane (Master) to ensure that containers are running in a Pod. It starts and stops containers, monitors their health, and reports the current node's status to the control plane.

Kube Proxy: Kube Proxy is responsible for network connectivity and load balancing. It maintains network rules on the node and forwards traffic to the appropriate container. It helps implement services of type NodePort and LoadBalancer.

Container Runtime: This is not a Kubernetes-specific service, but it's a crucial component. It's the software responsible for running containers. Docker, containerd, and CRI-O are common container runtimes used with Kubernetes.

Node Controller: The Node Controller monitors the health of nodes and manages their lifecycle. If a node becomes unresponsive or fails, it is the Node Controller's responsibility to handle the situation.

Kube DNS: Kube DNS is a cluster DNS service that provides DNS for services within the cluster. It ensures that services and Pods can communicate with each other using their DNS names.

cAdvisor: Container Advisor (cAdvisor) is a tool that collects, aggregates, processes, and exports information about running containers. It is primarily used for monitoring and performance analysis.

Ingress Controller: An Ingress Controller manages external access to services in the cluster. It is responsible for routing external traffic to the appropriate services based on Ingress resources defined in the cluster.

Heapster/Metrics Server: Heapster was used for cluster-wide monitoring and resource usage analysis. In newer versions of Kubernetes, Heapster has been deprecated in favor of the Metrics Server, which provides resource usage metrics to various parts of Kubernetes.

Container Network Interface (CNI): The CNI is not a single service but a framework that allows different networking solutions to be used with Kubernetes. Services like Flannel, Calico, and Weave provide network plugins to implement container networking.

Kubelet Garbage Collection: This service helps clean up old and unused resources, such as unused images and containers, to reclaim disk space.

Node Problem Detector: This service detects and reports hardware and operating system problems on nodes, helping to improve cluster stability.

Kubelet Plug-ins: Kubelet supports dynamic configuration through plug-ins. These can be used for device plugins, resource management, and custom extensions to the Kubelet's functionality.

2. What is Pod Disruption Budget?

A Pod Disruption Budget (PDB) is a Kubernetes resource that allows you to define policies for how disruptions to your Pods can be controlled during events like voluntary evictions (e.g., due to scaling down) or involuntary disruptions (e.g., node failures). PDBs provide a way to ensure the availability of your applications during maintenance or scaling operations while also allowing for safe disruptions.

Here are some key aspects of Pod Disruption Budgets:

Availability Control: PDBs allow you to set constraints on how many Pods of a particular application can be disrupted simultaneously. This helps maintain a minimum level of availability for your application during various operational procedures, such as rolling updates.

Selector-Based Policy: PDBs are associated with one or more Pods through label selectors. You can specify which Pods the PDB applies to by selecting them based on labels.

MaxUnavailable: The most common constraint specified in a PDB is maxUnavailable, which defines the maximum number or percentage of Pods that are allowed to be unavailable during disruptions. For example, you might specify maxUnavailable: 1 to ensure that at least one instance of your application is available at all times.

MinAvailable: Alternatively, you can use minAvailable to specify the minimum number or percentage of Pods that must remain available. This can be useful in scenarios where you want to ensure a certain level of redundancy.

Scope: PDBs can be scoped at the namespace level, allowing you to set different disruption budgets for different applications or components within a cluster.

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

In Kubernetes, load balancing plays a crucial role in distributing network traffic across multiple Pods or replicas of an application to ensure high availability, optimal resource utilization, and the efficient use of the cluster's resources. The primary role of load balancing in Kubernetes is to:

Ensure High Availability: Load balancing helps maintain the availability of your applications by distributing incoming traffic across multiple instances of your application (Pods) running on different nodes. If one of the Pods or nodes fails, the load balancer can route traffic to healthy instances.

Optimize Resource Utilization: By evenly distributing traffic, load balancing helps prevent overloading of specific Pods or nodes. This results in better resource utilization, improved performance, and prevents any single point of failure.

Scaling and Horizontal Pod Autoscaling: Load balancing works seamlessly with scaling mechanisms like Horizontal Pod Autoscaling (HPA). When your application experiences increased traffic, HPA can dynamically create more replicas of your Pods, and the load balancer will automatically start routing traffic to the new instances.

Service Discovery: Load balancers often integrate with DNS, making it easier for clients to discover services. Kubernetes provides a built-in DNS service that resolves service names to their associated Pod IP addresses.

Traffic Management: Kubernetes offers different types of services, such as ClusterIP, NodePort, and LoadBalancer services. LoadBalancer services expose applications to external traffic, and the cloud provider's load balancer, in this case, performs the load balancing.

Ingress: Ingress controllers provide a way to manage external access to services within the cluster. They often include load balancing capabilities to distribute incoming traffic to different services based on hostnames or paths.

Session Affinity: Some load balancers can be configured to use session affinity (sticky sessions), which ensures that a client's requests are consistently directed to the same backend Pod. This can be useful for stateful applications.

Security: Load balancers can also enhance security by providing features like SSL termination, which allows them to handle encryption and decryption of traffic.

Global Load Balancing: In multi-region or multi-cloud setups, global load balancing can distribute traffic across different clusters or data centers, ensuring low-latency access for users.

Conclusion: In this assignment we understood the Kubernetes Cluster Architecture.

Assignment #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 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.

Some of the benefits of AWS S3 are:

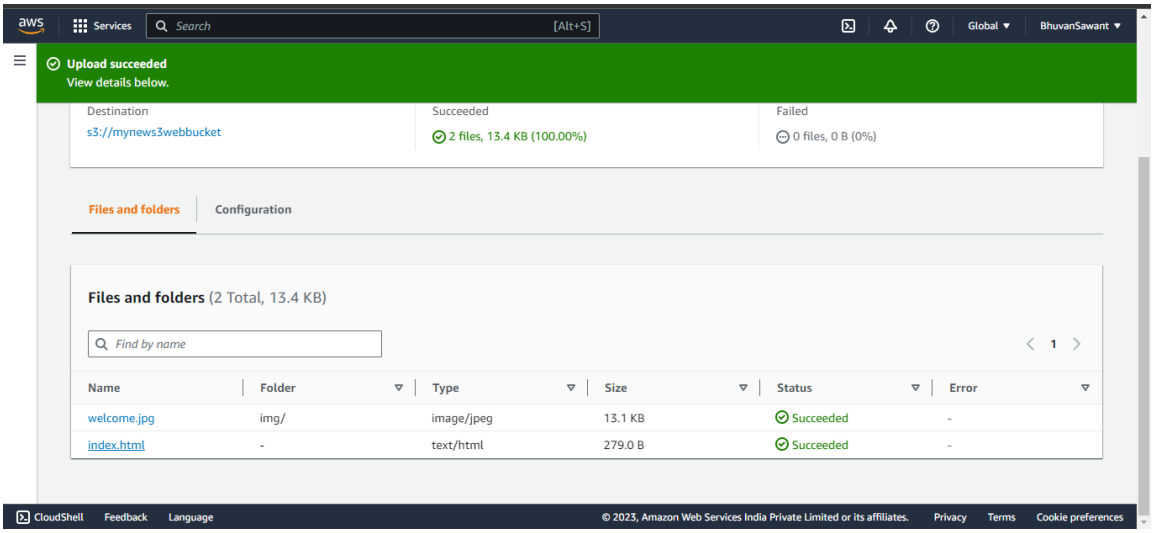
- Durability: S3 provides 99.999999999 percent durability.
- Low cost: S3 lets you store data in a range of "storage classes." These classes are based on the frequency and immediacy you require in accessing files.
- Scalability: S3 charges you only for what resources you actually use, and there are no hidden fees or overage charges. You can scale your storage resources to easily

meet your organization’s ever-changing demands.

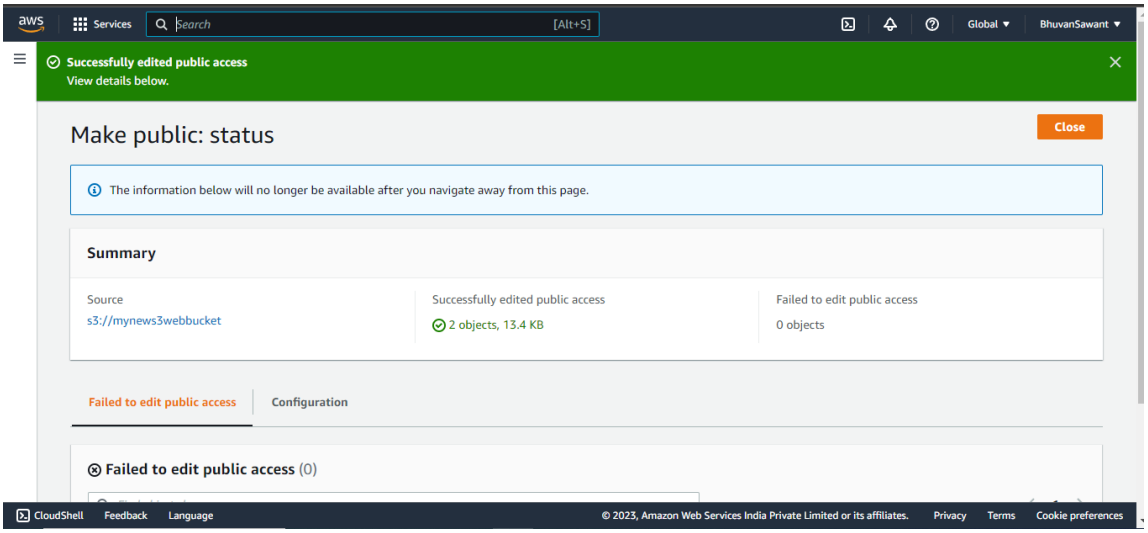
- Availability: S3 offers 99.99 percent availability of objects
- Security: S3 offers an impressive range of access management tools and encryption features that provide top-notch security.
- Flexibility: S3 is ideal for a wide range of uses like data storage, data backup, software delivery, data archiving, 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.

Parth 112

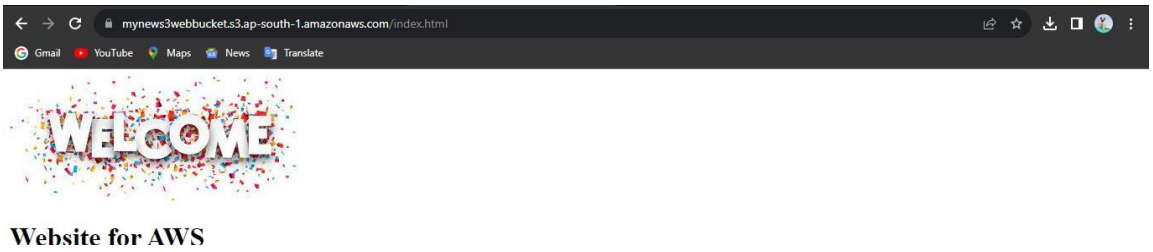
First login to your AWS account.



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 theS3 bucket. We create a two-stage pipeline that uses a versioned S3 bucket and CodeDeploy to release a sample application.

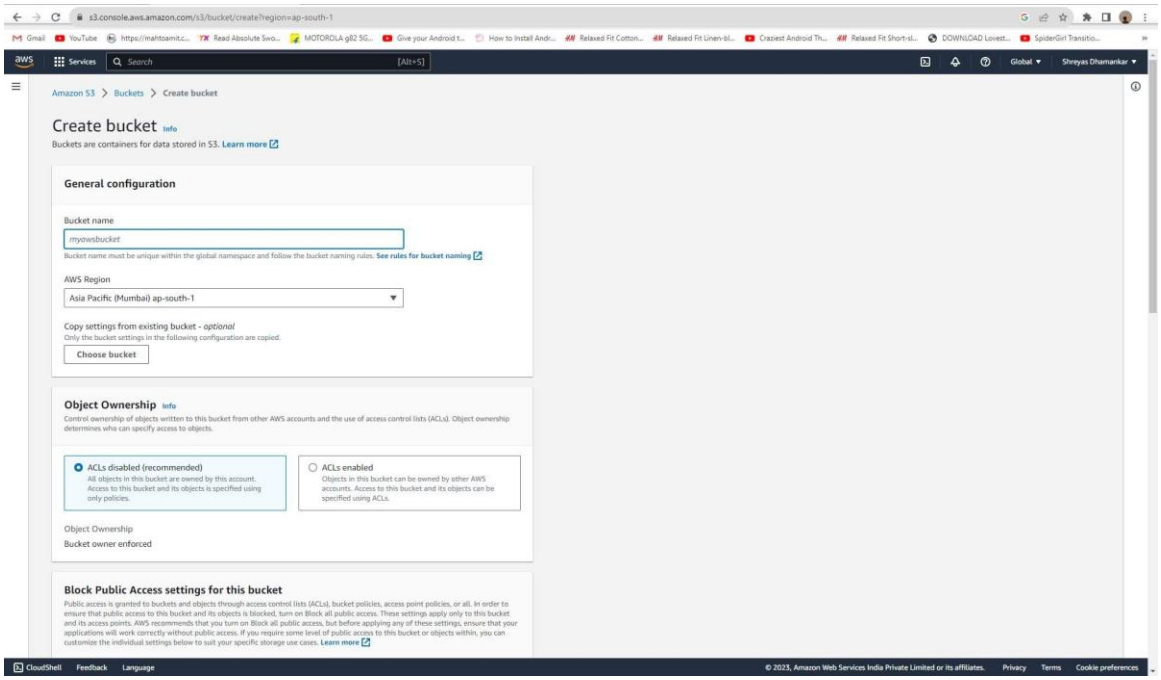
ASSIGNMENT 9

AIM : TO create a lambda function which will log “An image has been added” once you add an object to specific bucket in S3

LO: LO1 and LO5

THEORY :

Create bucket:

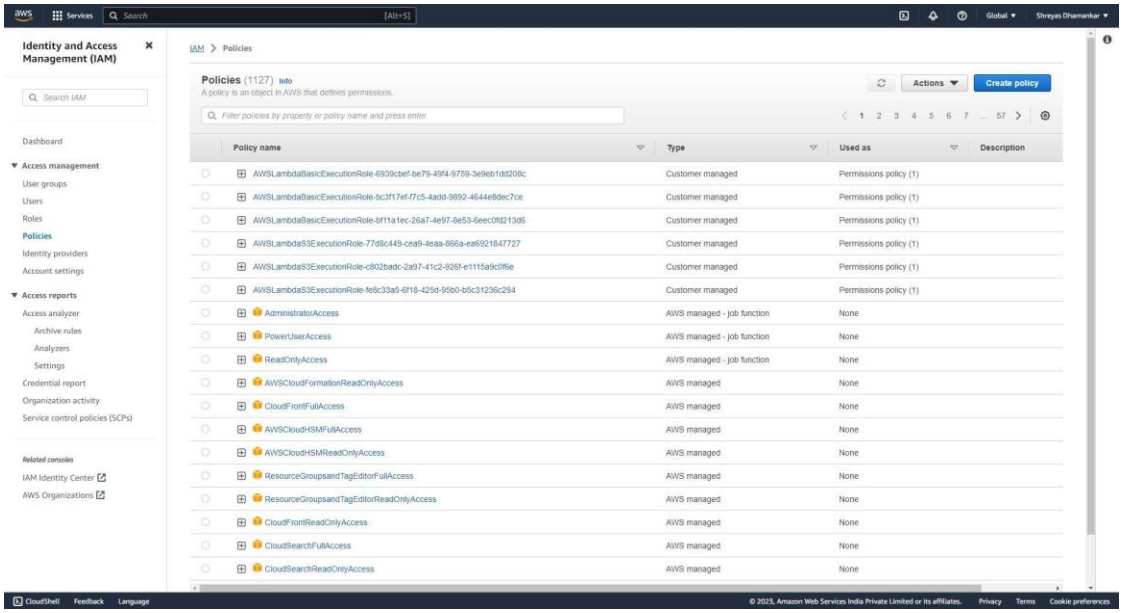


create a new policy from iam dashboard;

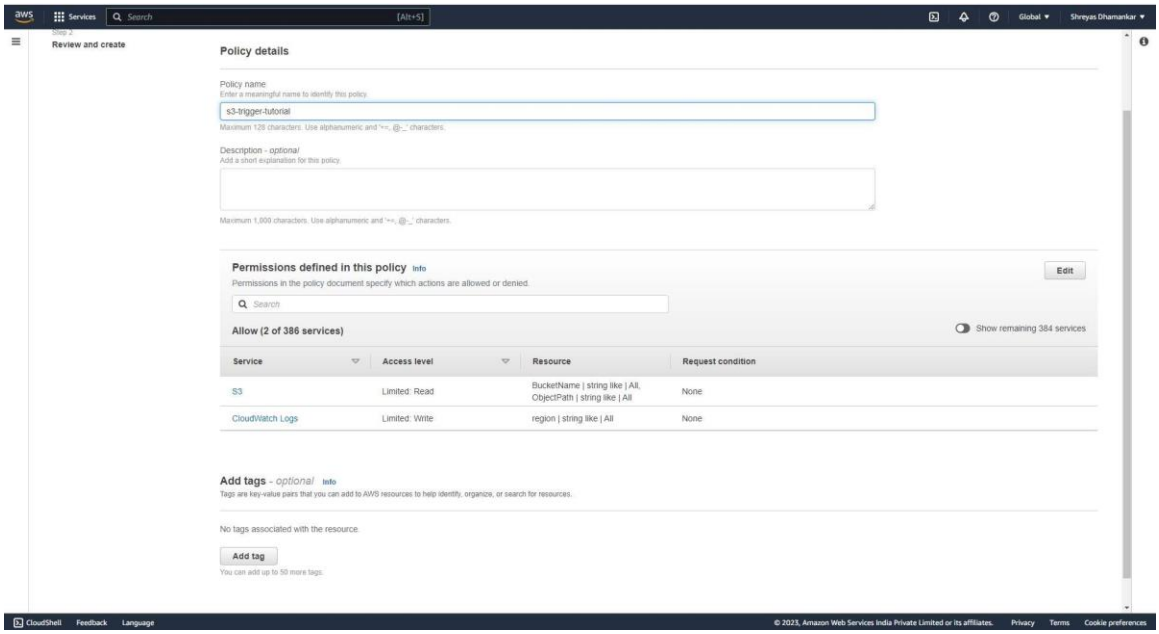
while creating policy select json tab and paste the following code:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
```

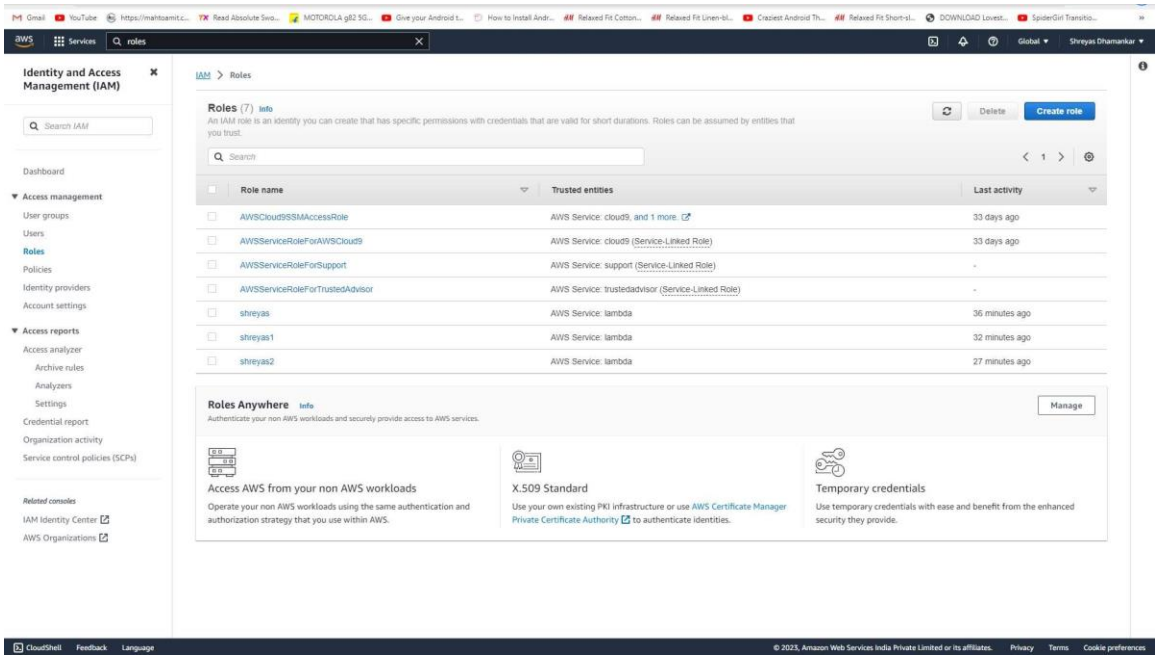
```
            "logs:PutLogEvents",
            "logs:CreateLogGroup",
            "logs:CreateLogStream"
        ],
        "Resource": "arn:aws:logs:*:*:*"
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:GetObject"
        ],
        "Resource": "arn:aws:s3:::*/*"
    }
  ]
}
```



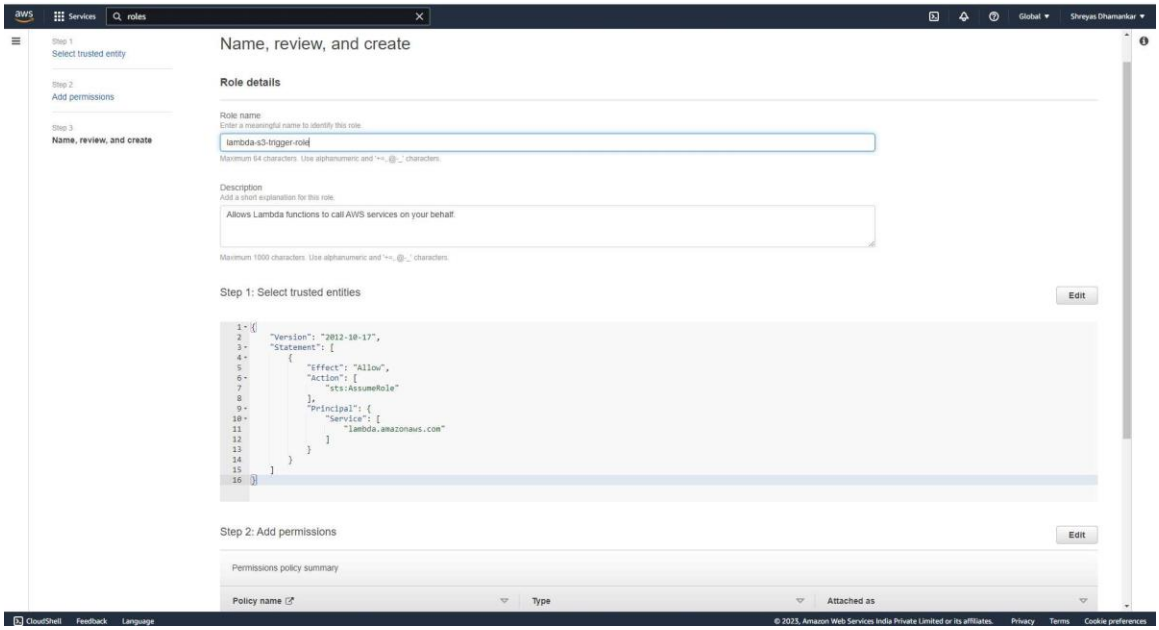
create policy and name it:



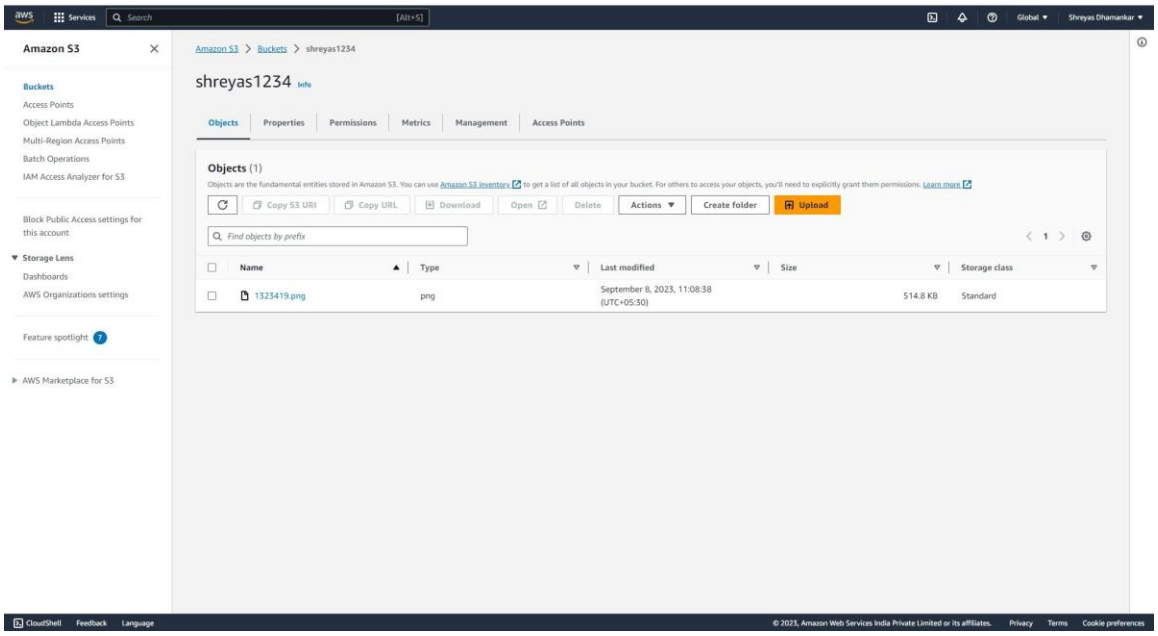
go to roles page and select create a new role



under policies for role select the policy that you have created and click next. Then name the role as follows:



Upload an image file in the S3 bucket.



Go to lamda dashboard in aws and create a new function named s3-trigger tutorial. select change execution code and choose the option use existing role. Select lambda-s3-trigger-role.

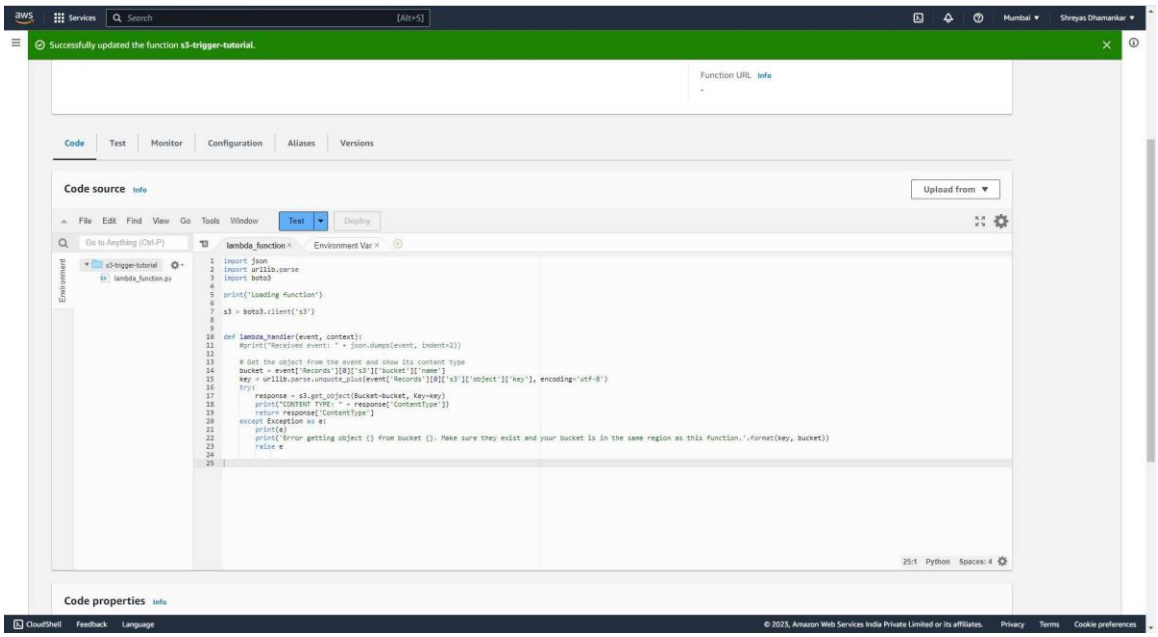
Once function is created go to code panel and paste the following code.

import json

import urllib.parse

```
import boto3
print('Loading function')
s3 = boto3.client('s3')
def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))
    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("An object : "+response['ContentType']+" has been added to S3 Bucket")
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

then select deploy changes.



then click on test and create a new custom event named MyTestEvent.

For Template, choose S3 Put.

In the Event JSON, replace the following values:

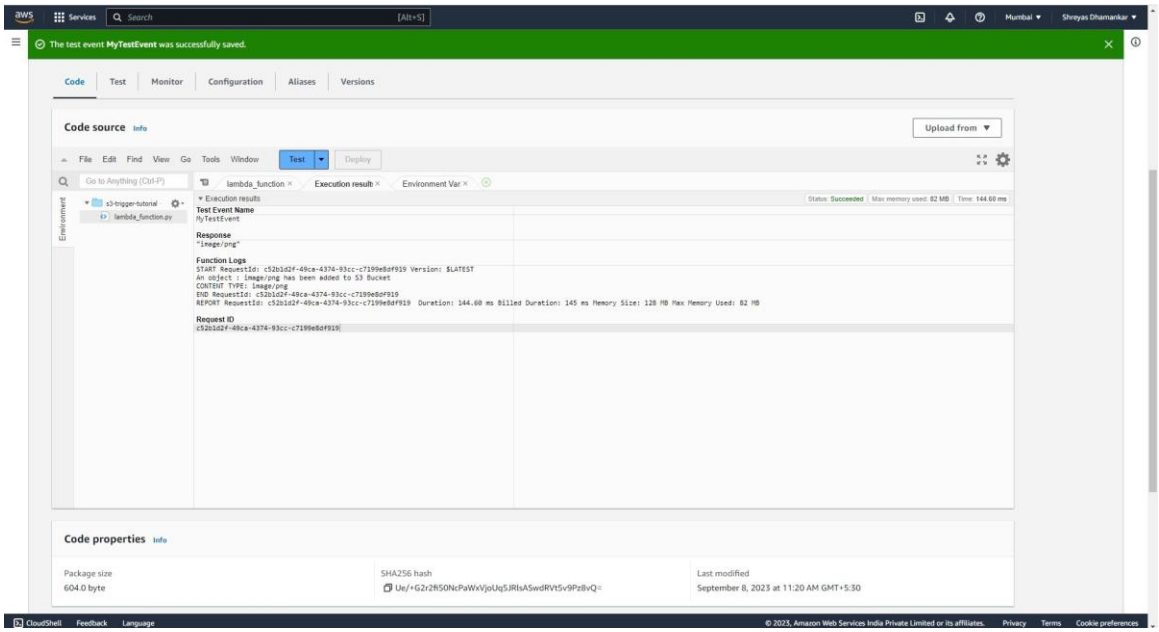
Replace us-east-1 with the region you created your Amazon S3 bucket in.

Replace both instances of example-bucket with the name of your own Amazon S3 bucket.

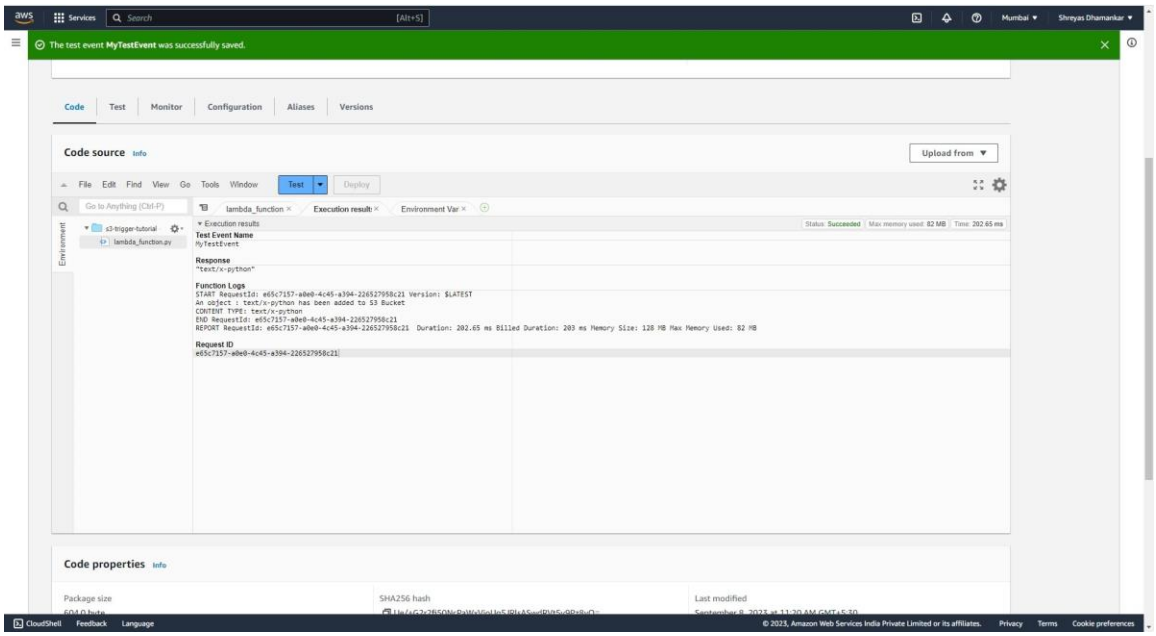
Replace test%2FKey with the name of the test object you uploaded to your bucket earlier (for example, factorial.py).

Click on save and press Test. You will see the results in Execution tab.

Check execution Tab for the CONTENT TYPE. If you have uploaded the file properly you will get the file type that you have uploaded in the S3 Bucket. Check Function logs where the line has been printed that "An Image has been Upladed to S3 Bucket".



Next I have uploaded a python file too and this is the output we get in the excution tab.



CONCLUSION : In this assignment we learnt how to use Lambda function to log what

type of object has been uploaded to S3 Bucket.

Assignment No. 6

Aim: To understand terraform lifecycle and to Build, change, and destroy AWS infrastructureusing 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

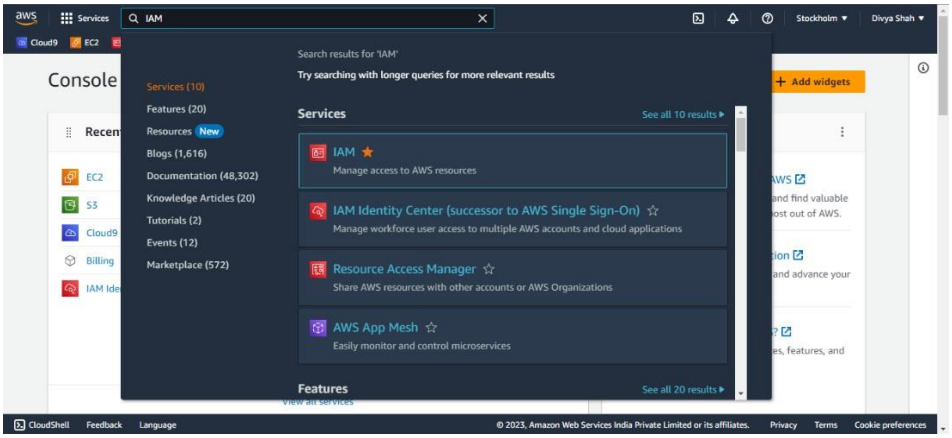


Fig 1

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:

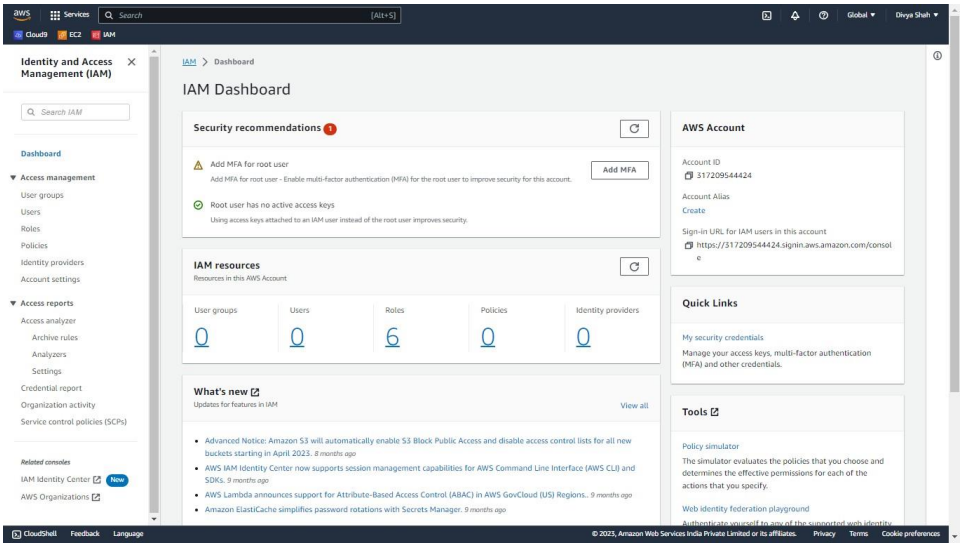


Fig. 2

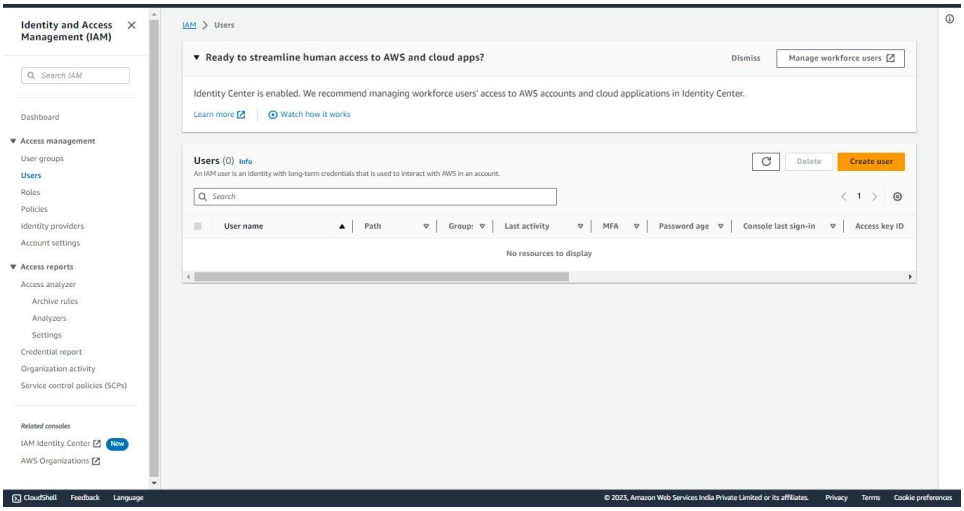


Fig.3

III. Provide a “Username” and click next:

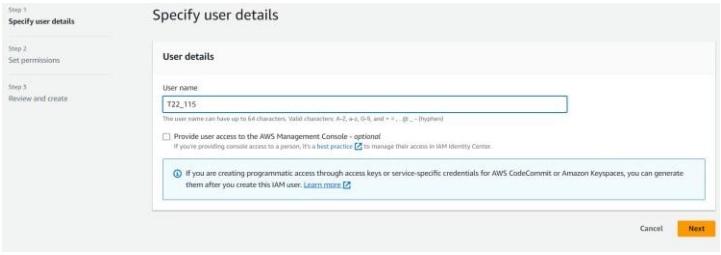


Fig. 4

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

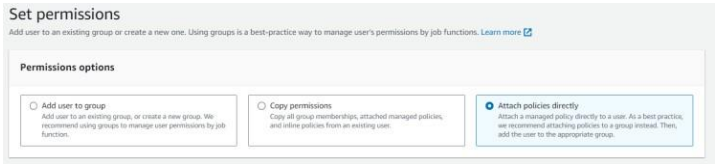


Fig. 5

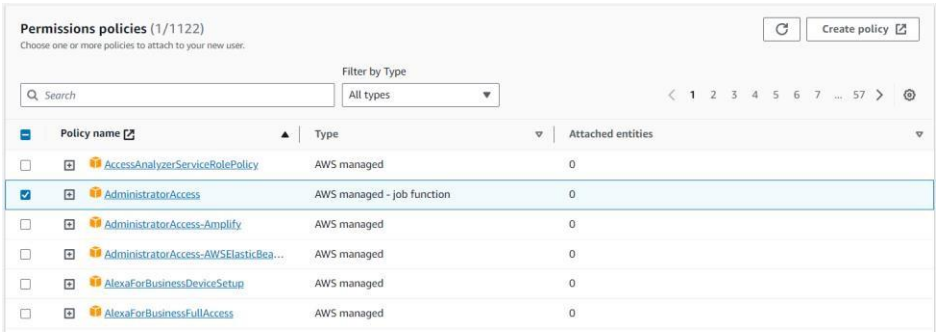


Fig. 6

V. Now click on “Create user”:

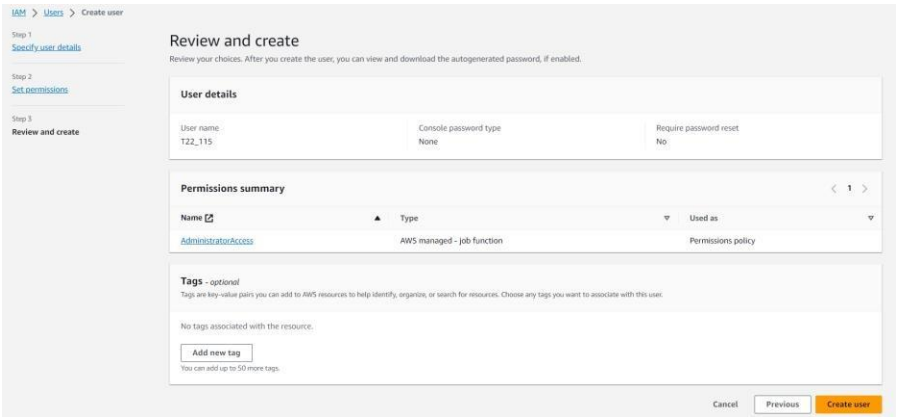


Fig. 7

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

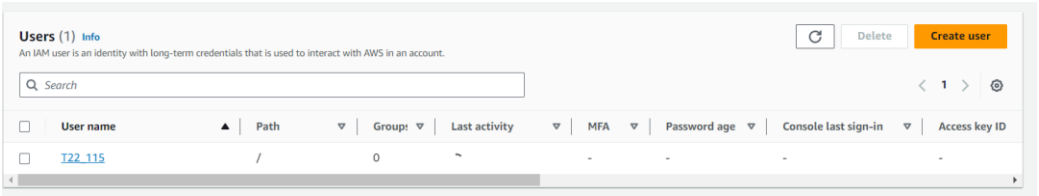


Fig.8

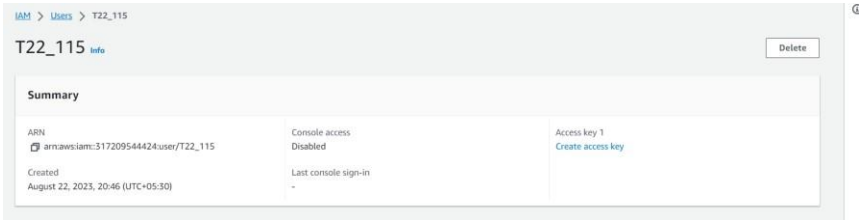


Fig.9

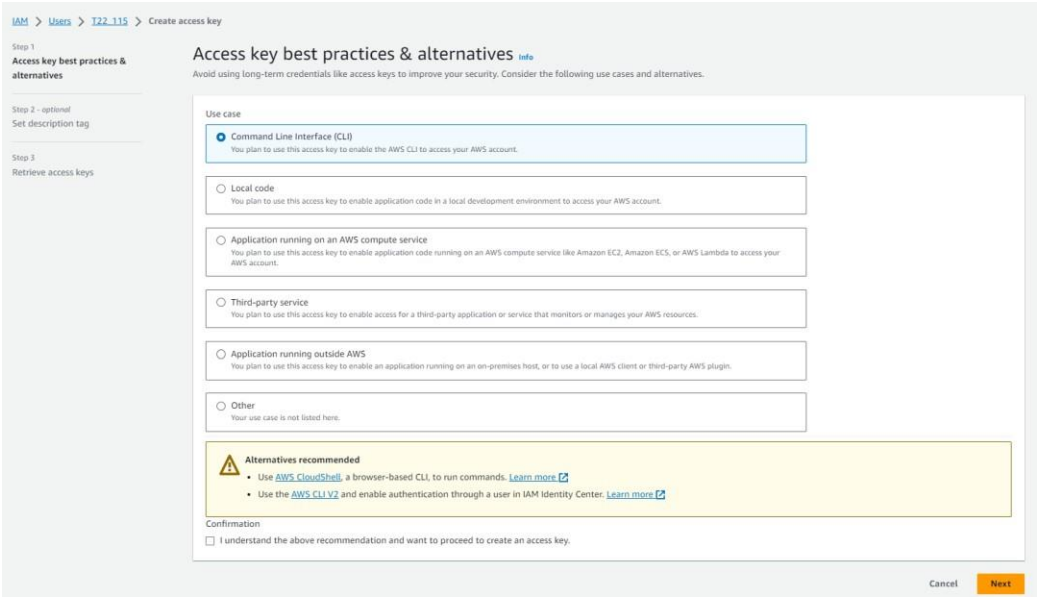


Fig.10

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

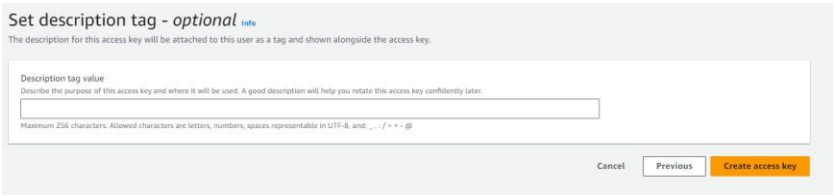


Fig.11

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

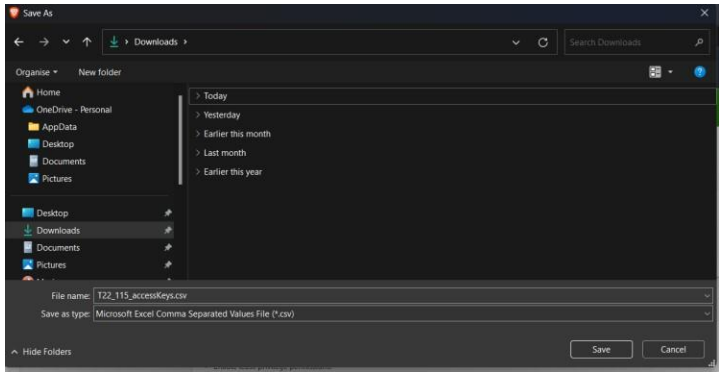


Fig. 12

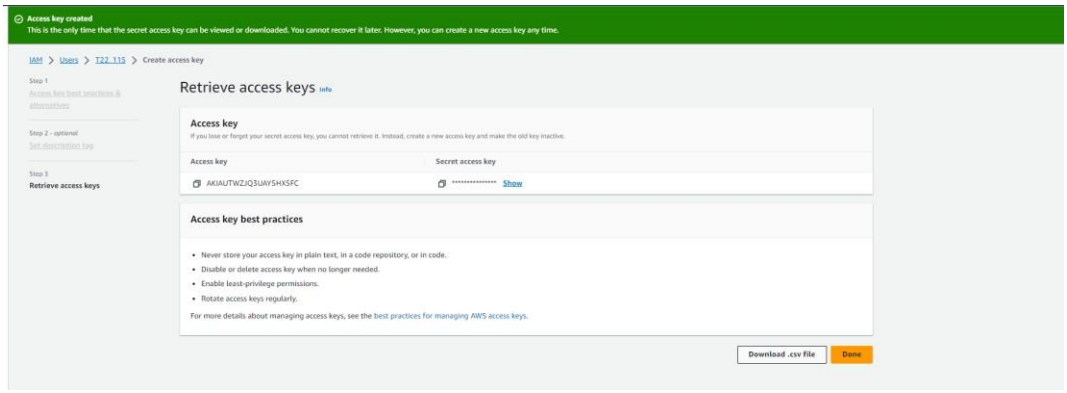


Fig.13

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.

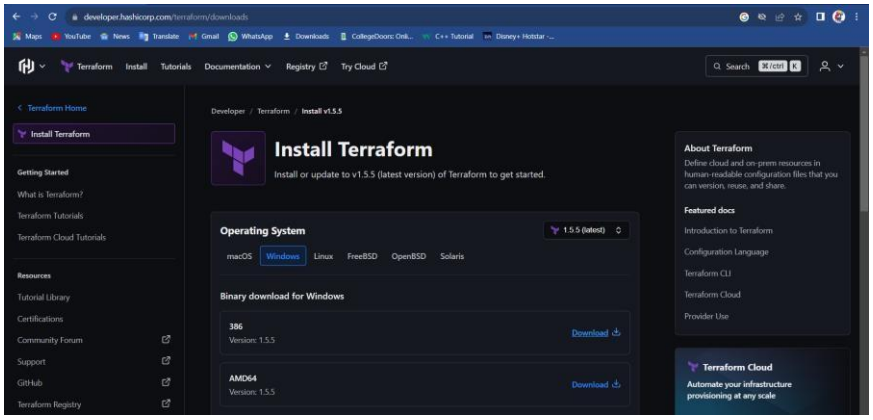


Fig.15

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

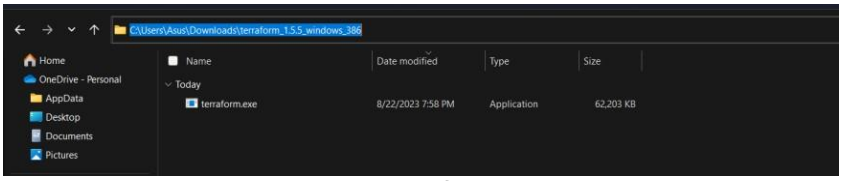


Fig.16

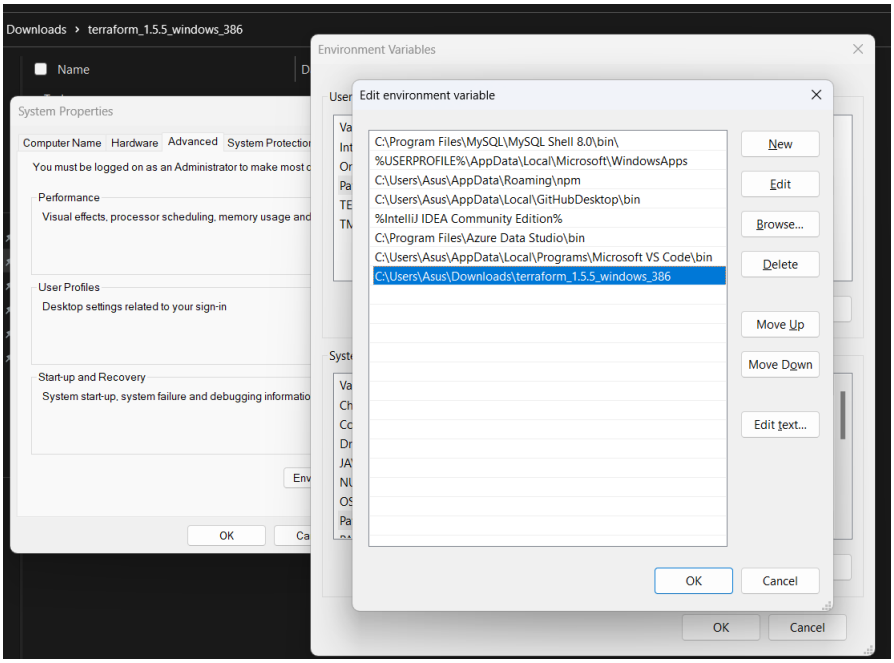


Fig.17

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

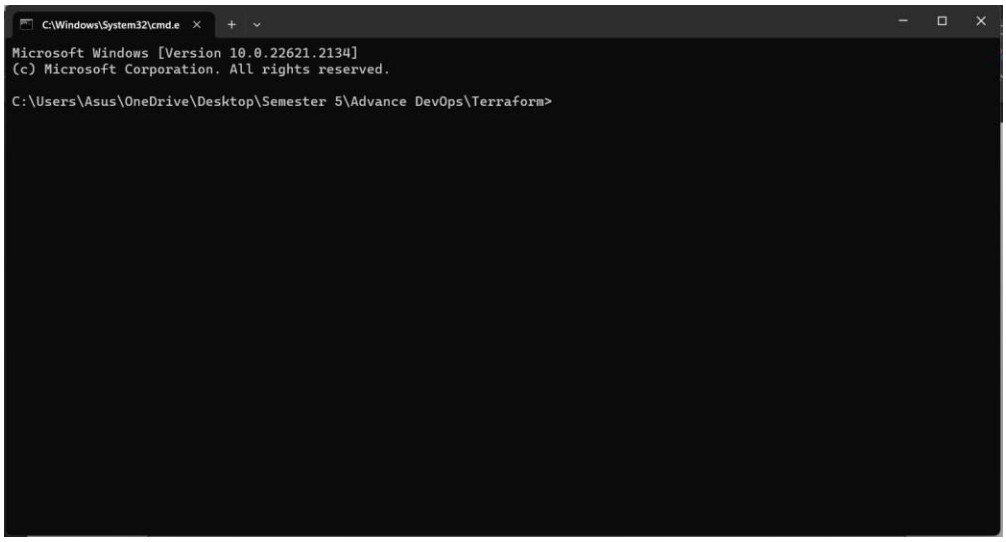


Fig. 18

XI. Now run command:

$$command: terraform\ init$$

```
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.

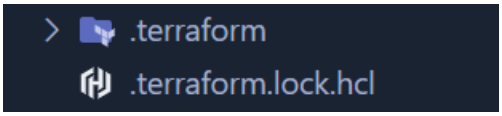


Fig. 21

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

```
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:

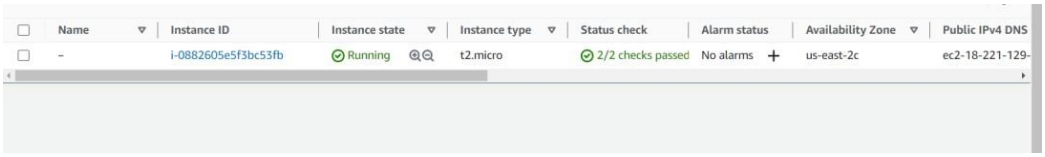


Fig. 24

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

$$command: terraform\ destroy$$

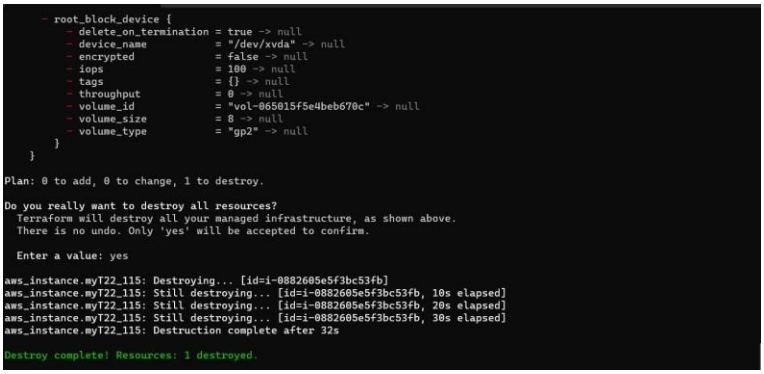


Fig. 25

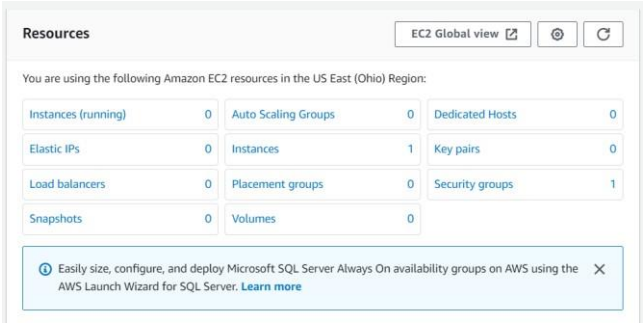


Fig. 26

Conclusion: In this experiment we understood the terraform lifecycle and to build/change AWS infrastructure using terraform.

Aim- To understand continous monitoring and installation and configuration of nagios on linux machine

LO: LO1, 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-

```
      4. Rerun the configure script.

NOTE: If you can't get the configure script to recognize the GD libs
      on your system, get over it and move on to other things.  The
      CGIs that use the GD libs are just a small part of the entire
      Nagios package.  Get everything else working first and then
      revisit the problem.  Make sure to check the nagios-users
      mailing list archives for possible solutions to GD library
      problems when you resume your troubleshooting.

*****************************************************************


checking ltdl.h usability... no
checking ltdl.h presence... no
checking for ltdl.h... no
checking dlfcn.h usability... yes
checking dlfcn.h presence... yes
checking for dlfcn.h... yes
checking for dlopen in -ldl... yes
checking for extra flags needed to export symbols...  -Wl,-export-dynamic
checking for linker flags for loadable modules... -shared
checking for traceroute... no
checking for type va_list... yes
checking for perl... /usr/bin/perl
checking for unzip... no
```

Now let us install plugins by

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

then

sudo tar zxf nagios-plugins.tar.gz

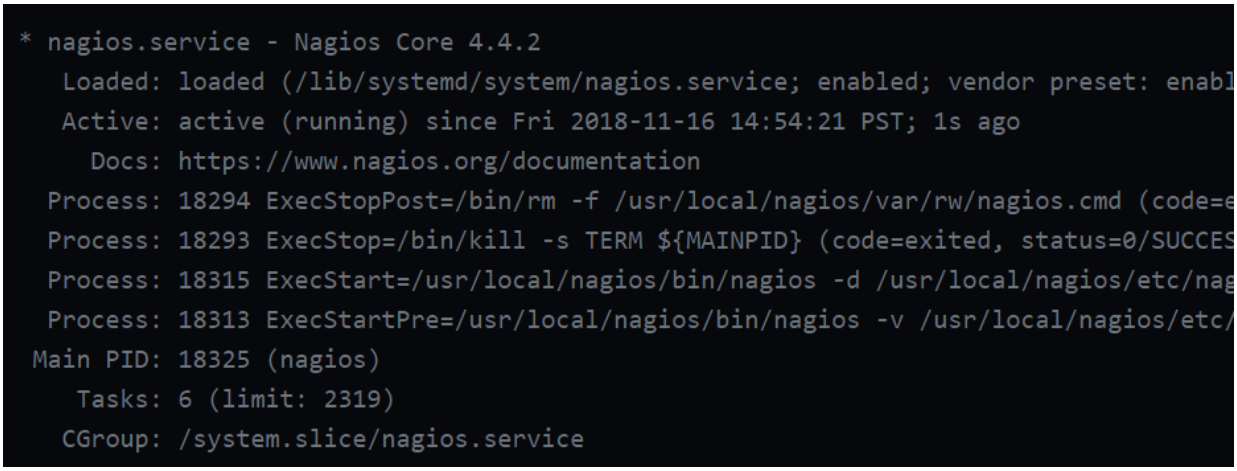
then

cd nagios-plugins-release-2.2.1

finally start and then check status of nagi os

sudo systemctl start nagios

sudo systemctl status nagios



CONCLUSION: **continous monitoring and installation and configuration of nagios on linux machine was succesfullyy implemented and output was observed.**

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 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.

Assignment Number 10

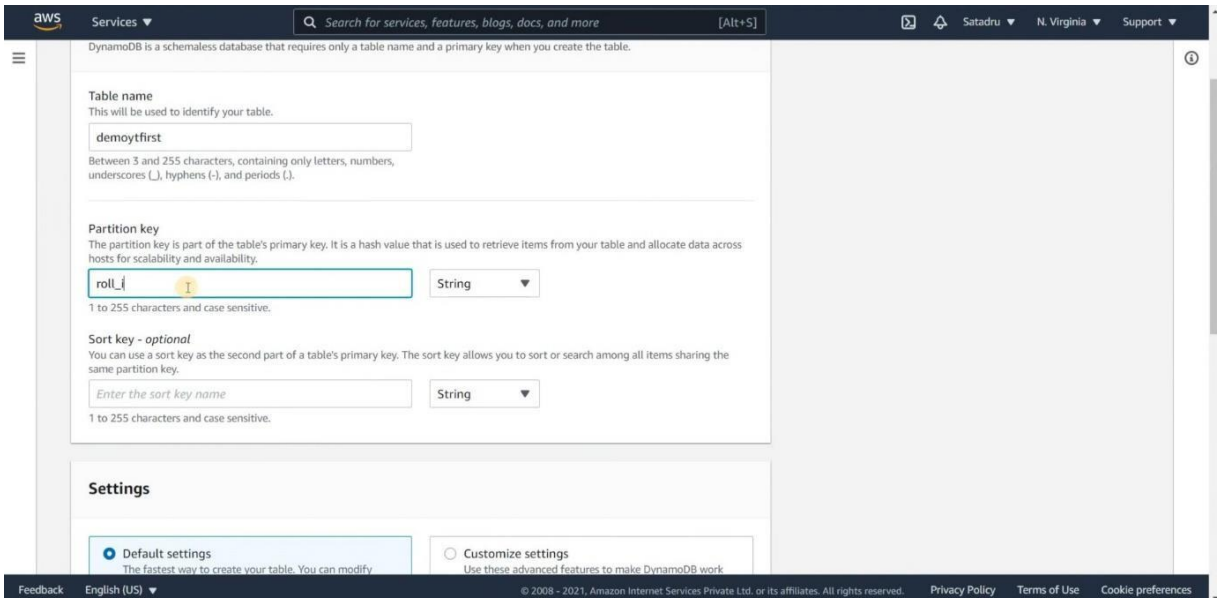
Aim: To create a Lambda function using Python for adding data to Dynamo DB database.

LO mapped: LO6

Theory:

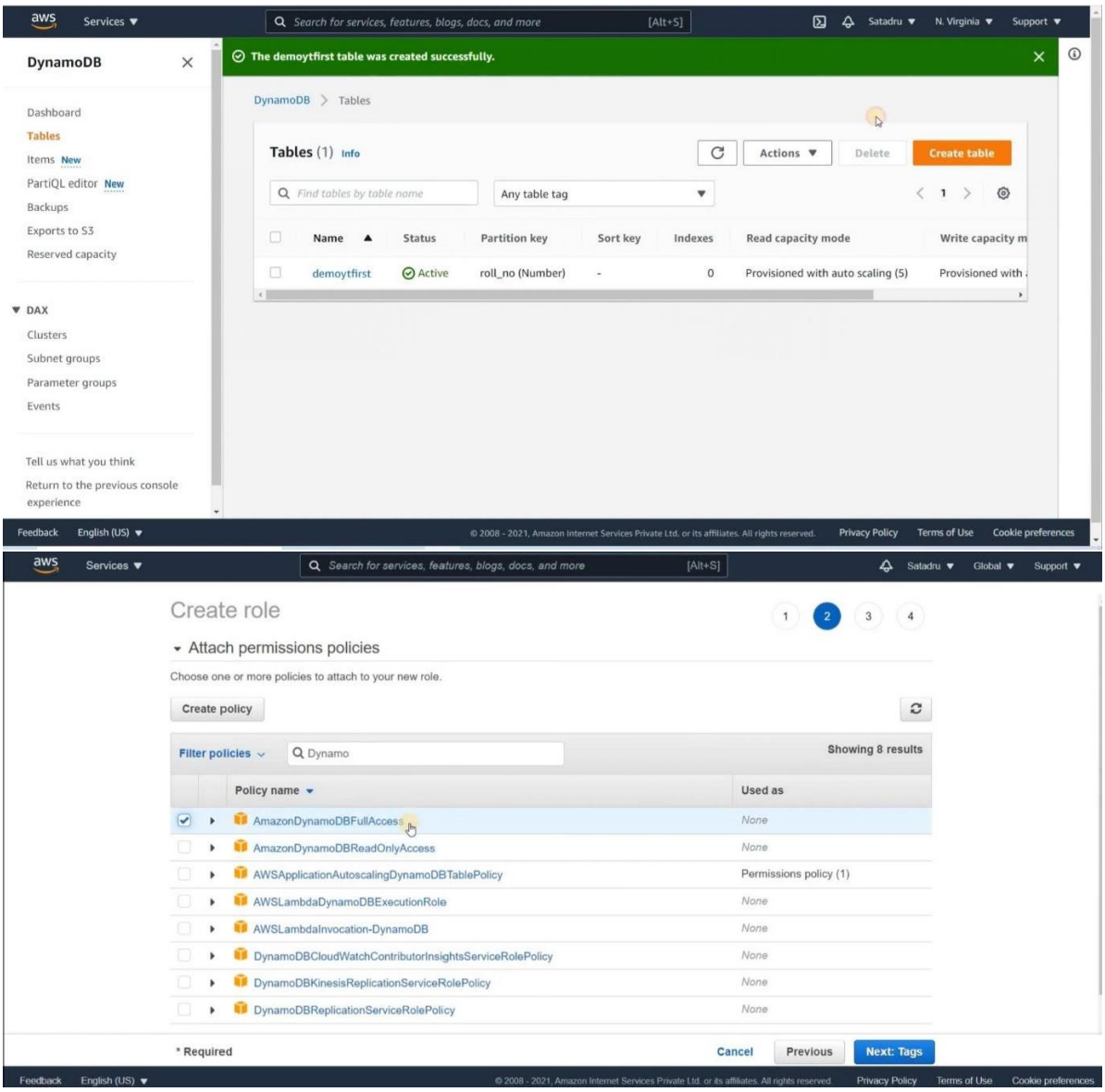
Step 1: Set Up AWS Environment

1. **Create an AWS Account:** If you don't have an AWS account, sign up for one at AWS Console.
2. **Access AWS Lambda Console:**



- Go to the AWS Lambda Console.
- Click on "Create function."

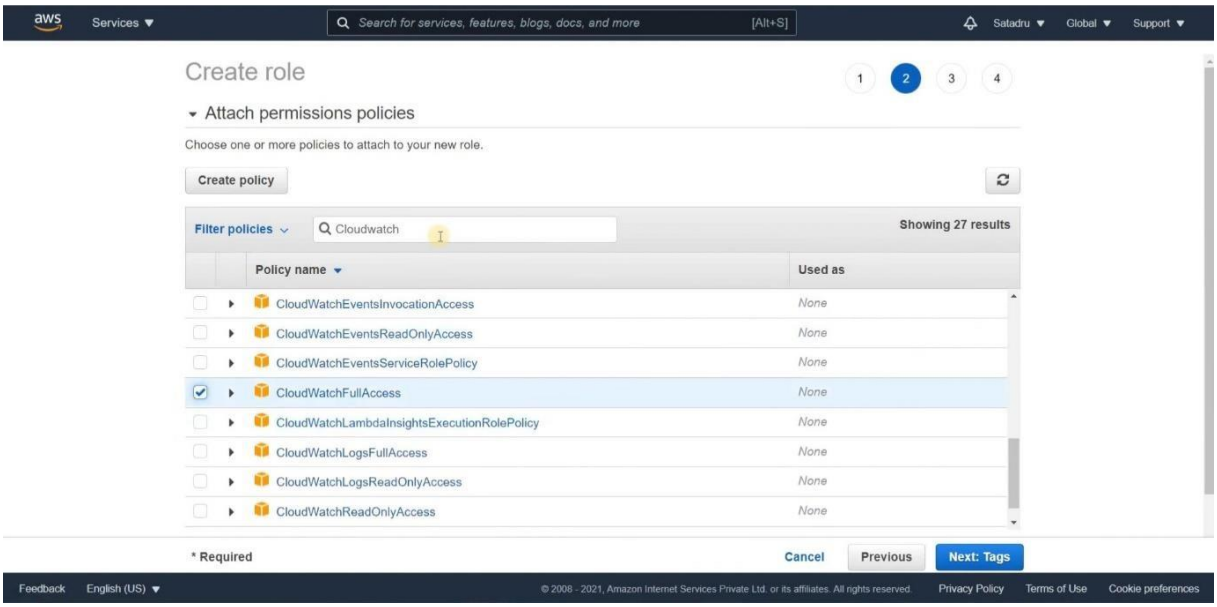
3. Create a New Lambda Function



:

- Choose "Author from scratch."
- Enter a name for your function (e.g., **AddToDynamoDBFunction**).
- Choose "Python" as the runtime.

4. **Configure Execution Role**

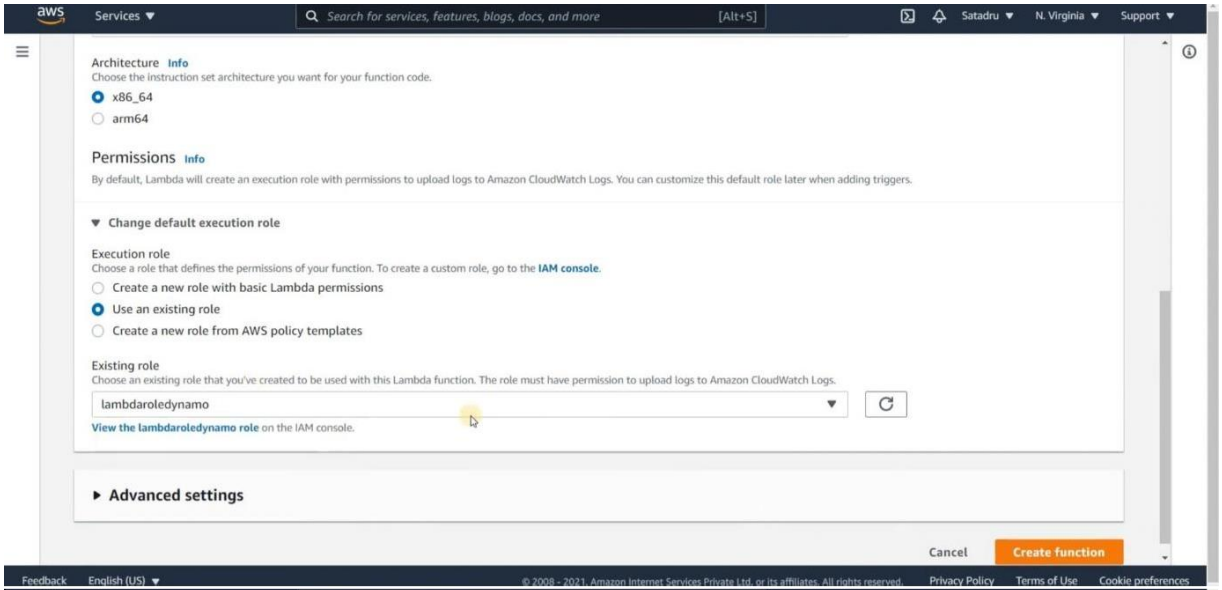


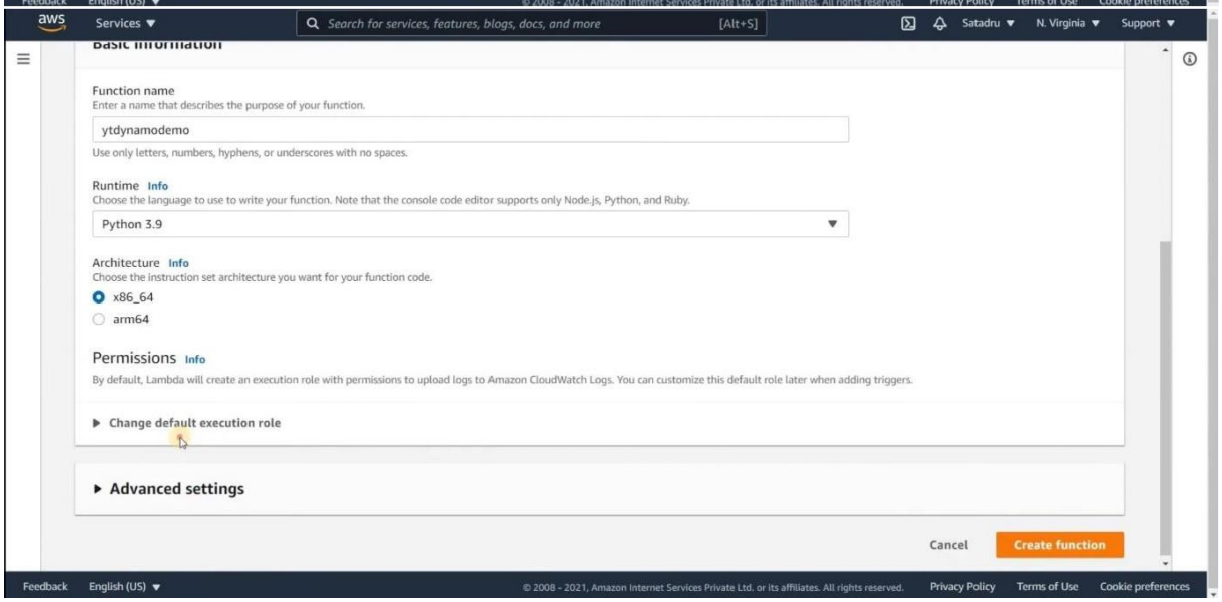
:

- Create a new role with basic Lambda permissions and additional permissions to interact with DynamoDB.
- Attach the role to your Lambda function.

Step 2: Set Up DynamoDB

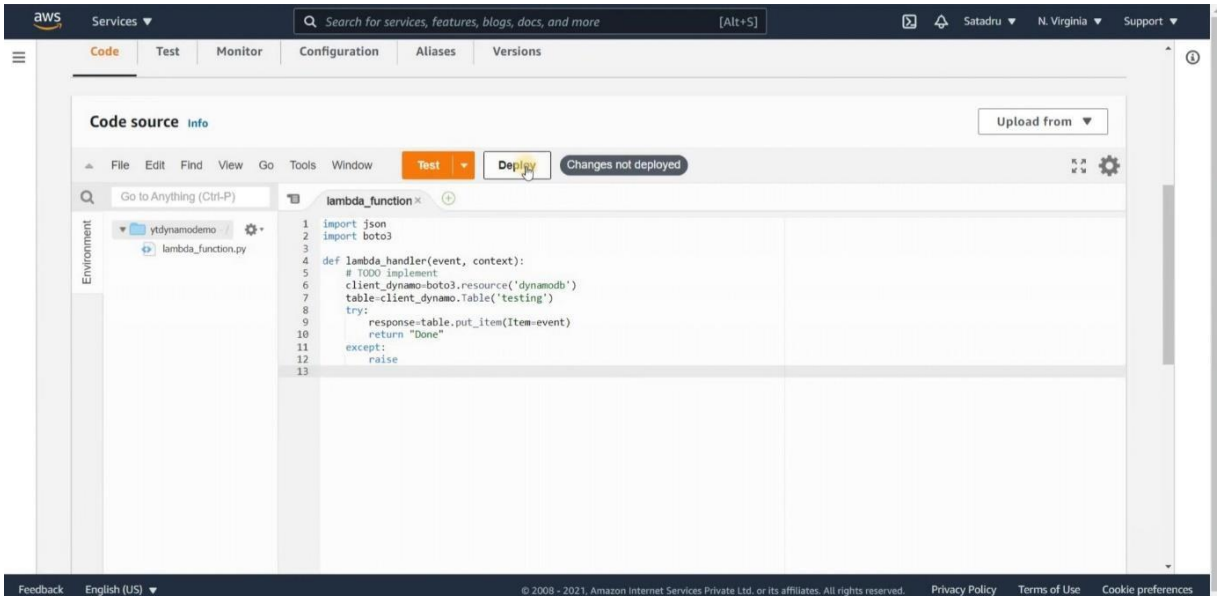
1. **Access DynamoDB Console:**
   - Go to the AWS DynamoDB Console.



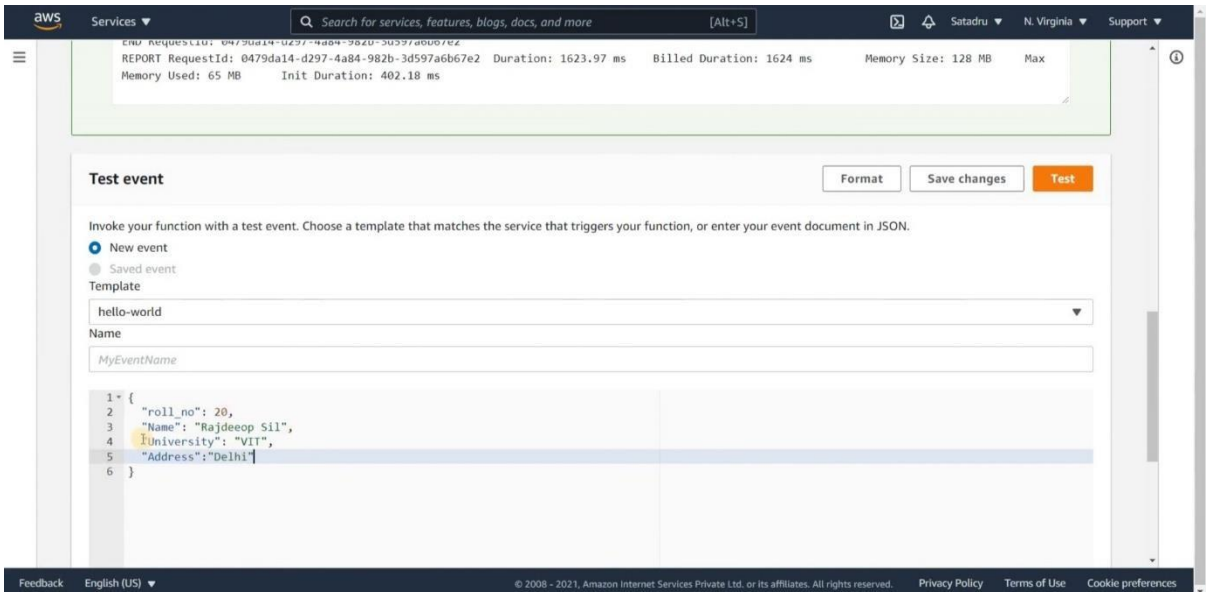


2. Create a DynamoDB Table:

- Click on "Create table."
- Enter a table name and a primary key (e.g., **ID**).
- Configure other settings as needed and create the table.



Step 3: Write Lambda Function Code

1. 

Ensure to replace **'YourDynamoDBTableName'** with the actual name of your DynamoDB table.

2. **Configure Lambda Handler:**

   - In the Lambda function configuration, set the handler to **filename.lambda_handler** (e.g., **yourfilename.lambda_handler**).

Step 4: Configure Environment Variables

1. **Add DynamoDB Table Name:**

   - In the Lambda function configuration, add an environment variable (e.g., **DYNAMODB_TABLE**) with the value set to your DynamoDB table name.

Step 5: Test Locally

1. **Test Lambda Function Locally:**
    - Create a test event with sample data.
    - Test the Lambda function using the "Test" button in the Lambda console.

Step 6: Deploy Lambda Function

1. **Deploy the Lambda Function:**
    - Click on the "Deploy" button in the Lambda console.

Step 7: Test in AWS Lambda Console

1. **Test in AWS Lambda Console:**
    - Use the Lambda console to test the function by creating a test event with sample data.
    - Verify that the function executes successfully.

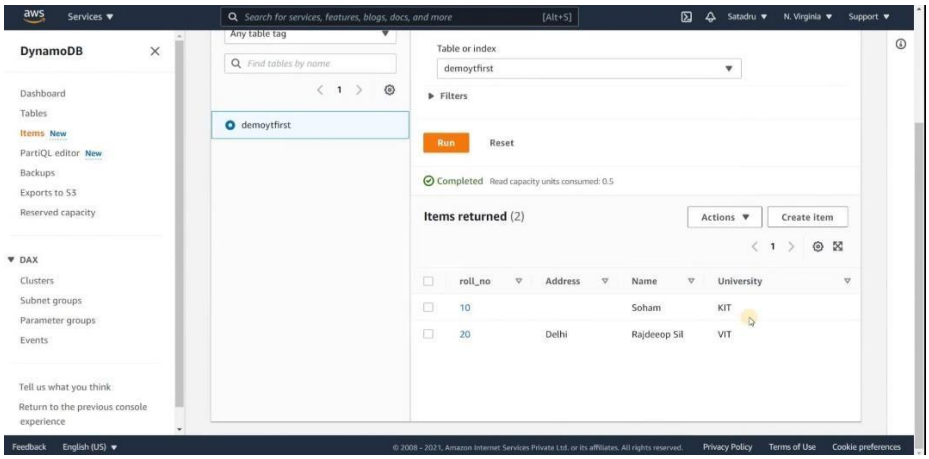
Step 8: Monitor and Troubleshoot

1. **Check CloudWatch Logs:**
    - Use CloudWatch Logs to check logs generated by your Lambda function for troubleshooting.

Step 9: Documentation

1. **Create Documentation:**
    - Document the purpose, input parameters, and expected output of your Lambda function.
    - Explain any challenges faced during development and how they were resolved.

By following these steps, you can create a Lambda function using Python to add data to a DynamoDB database in the AWS environment.



Conclusion: By this assignment we learn how to create a Lambda function using Python for adding data to Dynamo DB database.

Assignment #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 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.

Some of the benefits of AWS S3 are:

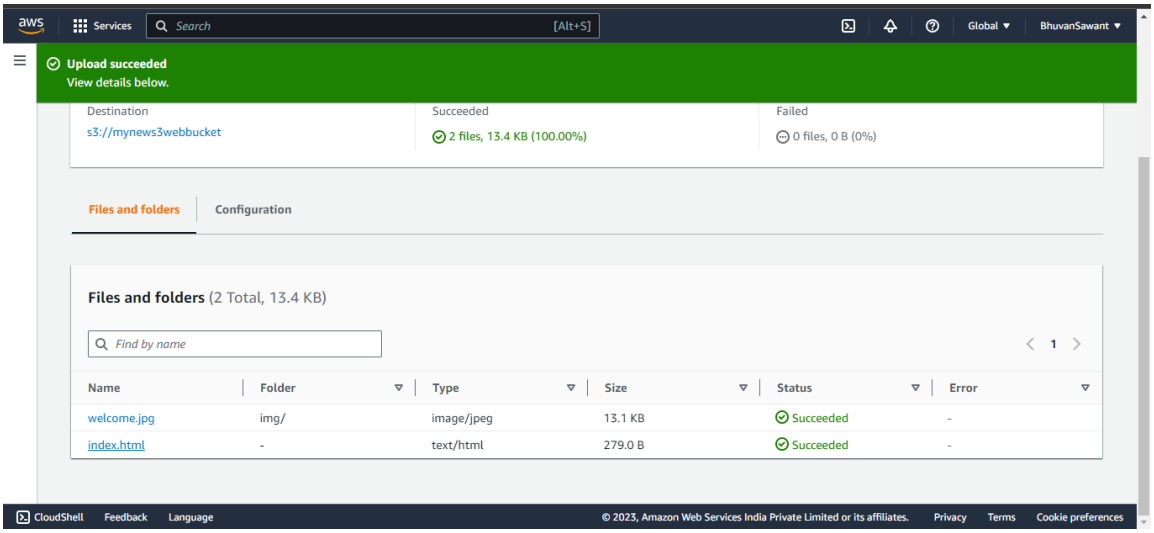
- Durability: S3 provides 99.999999999 percent durability.
- Low cost: S3 lets you store data in a range of "storage classes." These classes are based on the frequency and immediacy you require in accessing files.
- Scalability: S3 charges you only for what resources you actually use, and there are no hidden fees or overage charges. You can scale your storage resources to easily

meet your organization’s ever-changing demands.

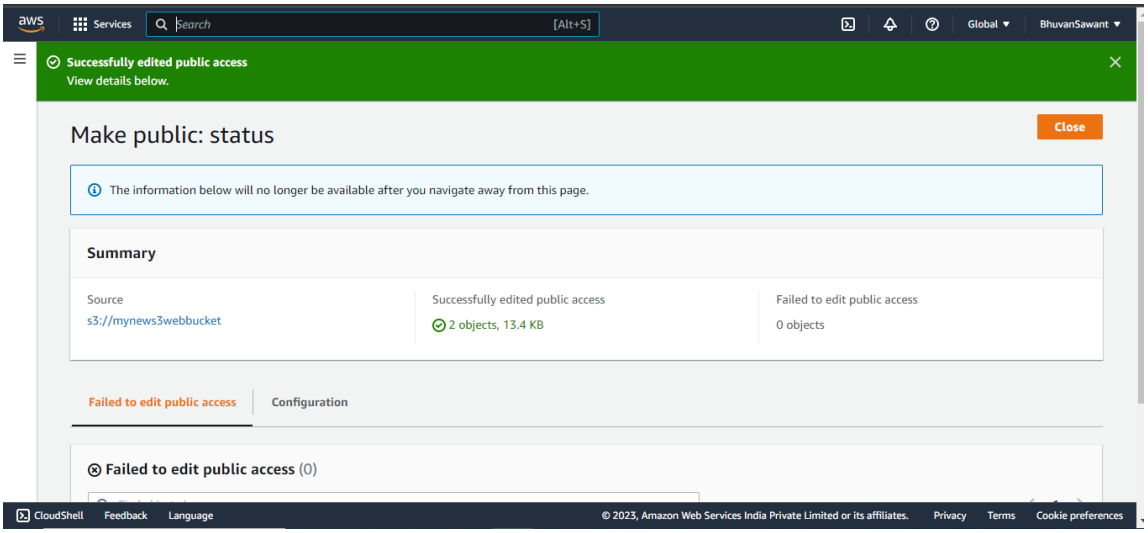
- Availability: S3 offers 99.99 percent availability of objects
- Security: S3 offers an impressive range of access management tools and encryption features that provide top-notch security.
- Flexibility: S3 is ideal for a wide range of uses like data storage, data backup, software delivery, data archiving, 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.

Parth 112

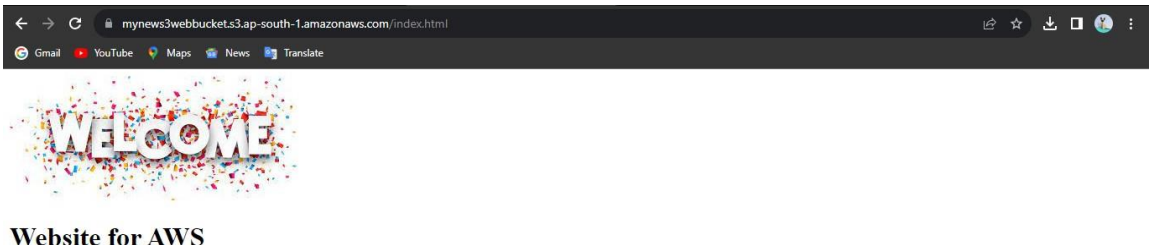
First login to your AWS account.



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 theS3 bucket. We create a two-stage pipeline that uses a versioned S3 bucket and CodeDeploy to release a sample application.

Assignment Title: AWS Lambda Deployment and Components

Name: Parth Sawant

Roll no: 112

Q)1: How to deploy Lambda function on AWS?

To deploy a Lambda function on AWS, follow these steps:

Step 1: Login to AWS Console

- Go to the AWS Management Console (https://aws.amazon.com/).
- Sign in with your AWS account.

Step 2: Access AWS Lambda

- Click on "Services" in the top left corner.
- Under "Compute," select "Lambda."

Step 3: Create a Lambda Function

- Click the "Create function" button.
- Choose "Author from scratch" or select a blueprint or application as needed.
- Configure the function's name, runtime, role, and other settings.

Step 4: Configure Triggers (if needed)

- You can configure event sources like S3, API Gateway, etc., to trigger your Lambda function.

Step 5: Write or Upload Code

- In the "Function code" section, write or upload your code.

- You can also set environment variables if required.

Step 6: Configure Function

- Set memory, timeout, VPC, and other settings under "Function Configuration."

Step 7: Review and Create

- Review your function's configuration.
- Click "Create function" to deploy it.

Step 8: Testing

- Test your Lambda function using the "Test" button.

Step 9: Monitor and Troubleshoot

- Monitor your function's execution and set up logging for troubleshooting.

Step 10: Integration

- Integrate your Lambda function with other AWS services or external applications.

Q)2: What are the deployment options for AWS Lambda?

AWS Lambda provides several deployment options:

1. **Create a Lambda Function from Scratch:** You can create a new Lambda function and write the code from scratch.
2. **Use Blueprints:** AWS offers pre-built blueprints for common use cases. You can select a blueprint and customize it as needed.
3. **Container Images:** You can package your code as a container image and deploy it on Lambda, providing more flexibility and customizability.
4. **Serverless Application Model (SAM):** SAM is a framework for defining serverless applications. You can use SAM templates to deploy and manage Lambda functions and related resources.

5. **AWS Serverless Application Repository:** Explore and deploy pre-built serverless applications and Lambda functions from the AWS Serverless Application Repository.

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

AWS offers three primary deployment modes for Lambda functions:

1. **Single Lambda Function:** In this mode, a single Lambda function is deployed to perform a specific task. It's suitable for simple use cases where a single function can handle the workload.
2. **Microservices:** Deploying Lambda functions as microservices involves breaking down a larger application into smaller, independent functions. Each function serves a specific purpose, and they work together to create a complete application.
3. **Serverless Applications:** This mode involves building complex, multi-component applications using Lambda functions and other AWS services. Serverless applications combine Lambda functions, API Gateway, DynamoDB, S3, and other services to create scalable and highly available applications.

Q)4: What are the 3 components of AWS Lambda?

AWS Lambda comprises the following three key components:

1. **Function Code:** This is the actual code that is executed when the Lambda function is triggered. You can write your code in languages such as Python, Node.js, Java, or use custom runtimes. You can also include any required libraries or dependencies.
2. **Execution Environment:** AWS Lambda provides a runtime environment for your function code to run. It manages resources like memory allocation and CPU power. You can configure these settings based on your function's requirements.
3. **Event Source:** Event sources trigger Lambda functions. These sources can include AWS services (e.g., S3, DynamoDB, SNS), custom events, or HTTP requests via API Gateway. When an event occurs, Lambda is invoked, and the function code is executed in response to the event.

These components work together to enable the serverless execution of code in response to events, making AWS Lambda a powerful and flexible compute service.