Sanskruti Adap

T11-1

LAB ASSIGNMENT 1

AIM: To launch an EC2 instance in AWS.

THEORY :

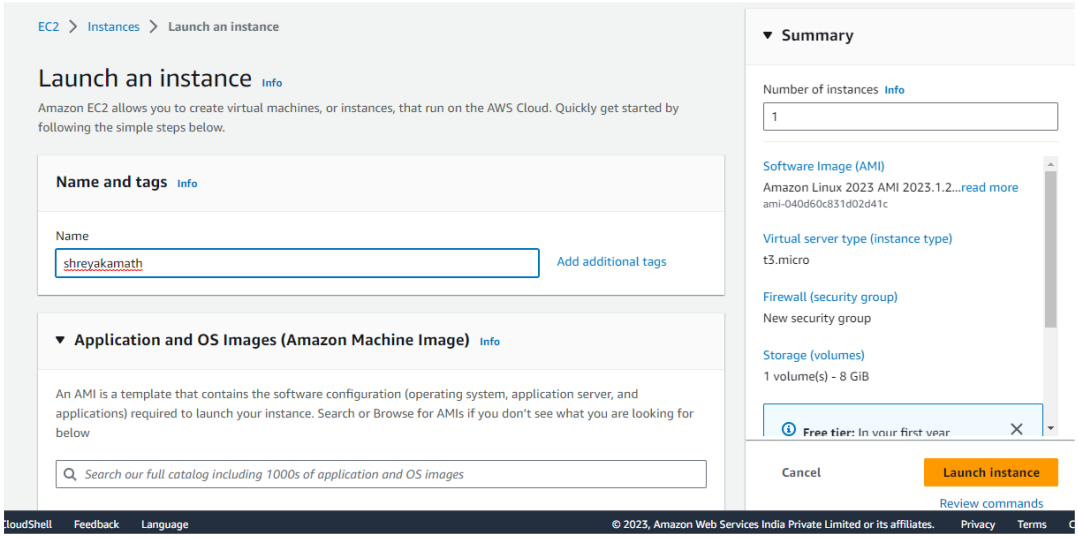
Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 reduces hardware costs so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage. You can add capacity (scale up) to handle compute-heavy tasks, such as monthly or yearly processes, or spikes in website traffic. When usage decreases, you can reduce capacity (scale down) again.

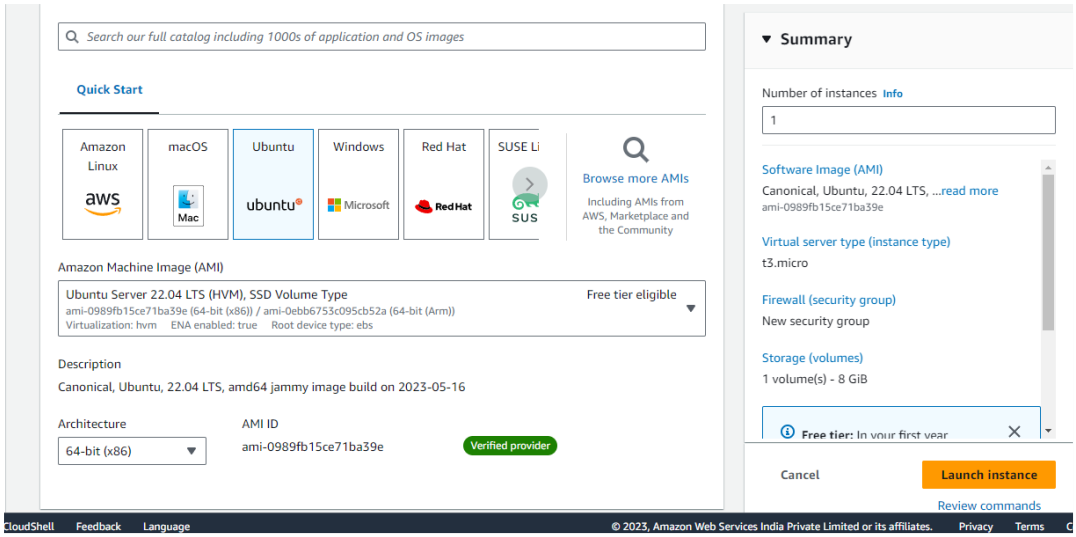
STEPS :

1. Create a free tier account on AWS.

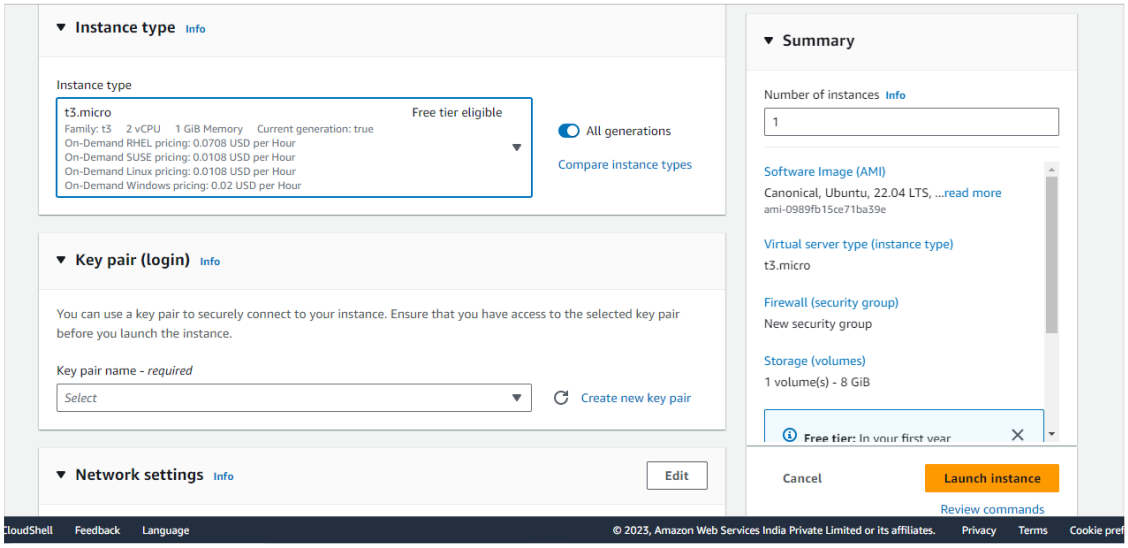
2. Go to EC2 Services.

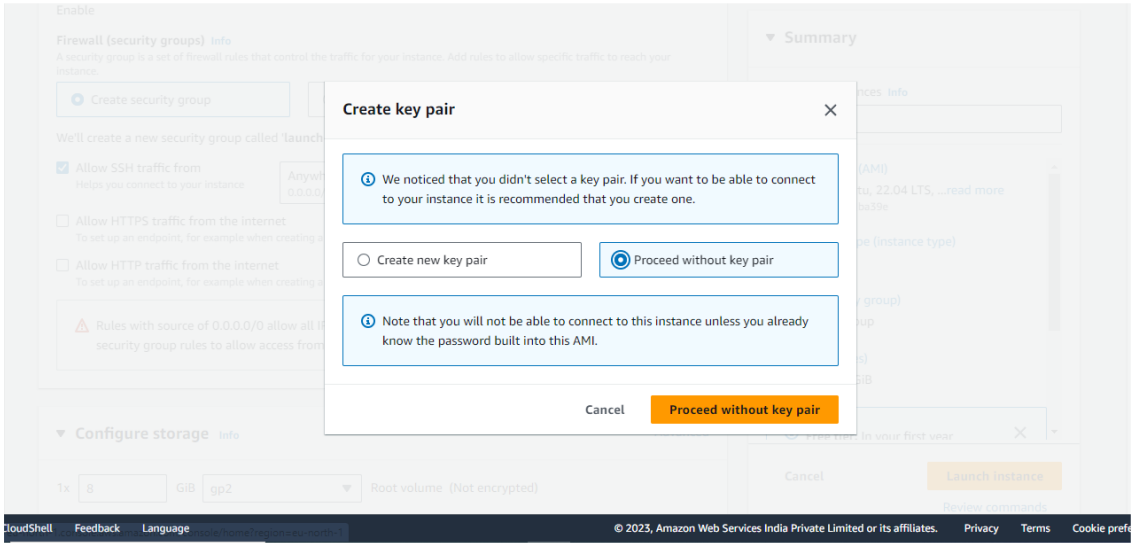
3. Launch an instance by giving it a name and selecting an AMI.



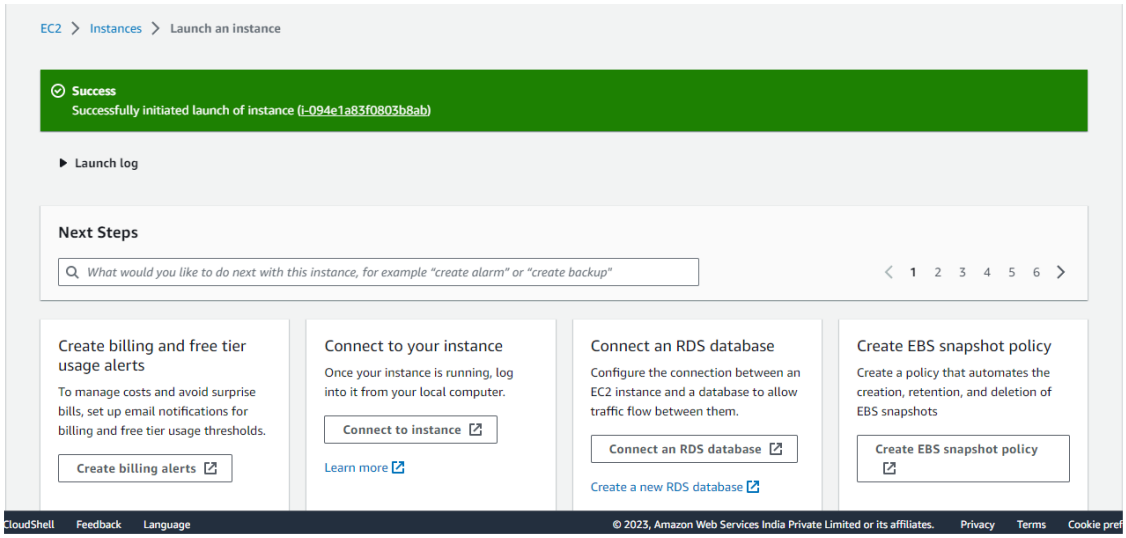


4. Proceed without Key Pair

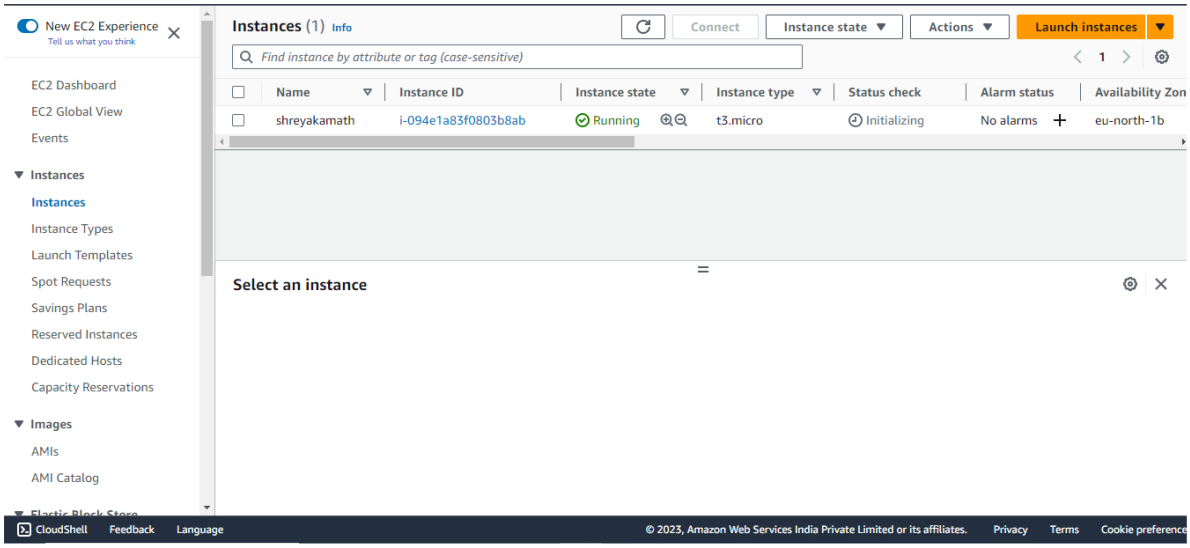




5. After the instance is launched, connect using EC2 instance Connect.

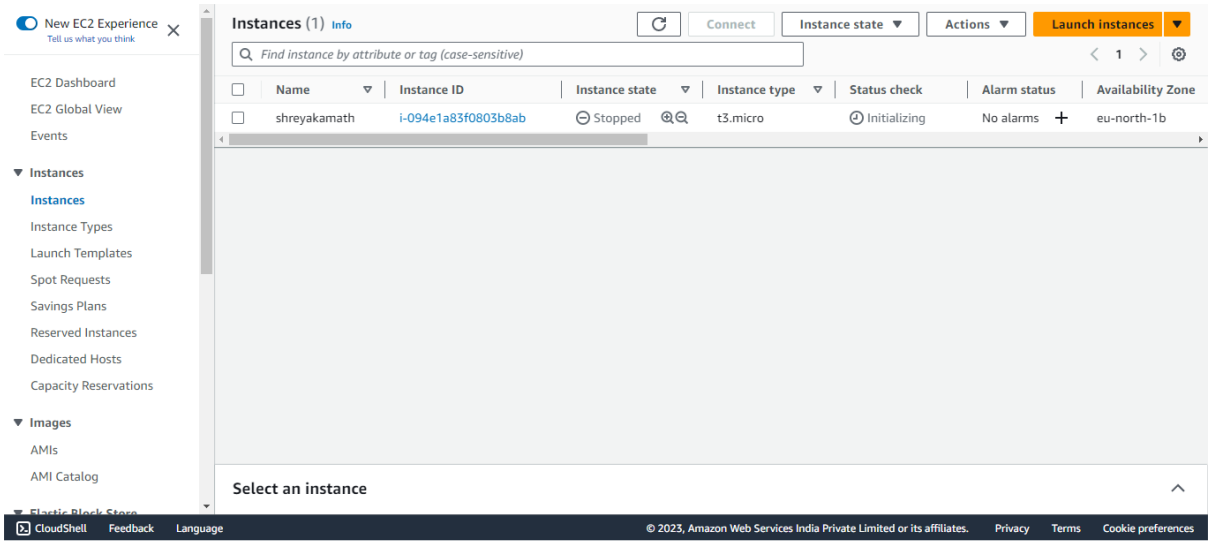


6. Service has started.



7. Security groups are created.

8. Terminate the instance.



9. Delete the security groups.

LAB OUTCOME:

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

Sanskruti Adap

T11-1

LAB ASSIGNMENT 1B

Aim : To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and perform collaboration demonstration.

Theory :

AWS Cloud9 allows you to write, run, and debug your code with just a browser. With AWS Cloud9, you have immediate access to a rich code editor, integrated debugger, and built-in terminal with preconfigured AWS CLI. You can get started in minutes and no longer have to spend the time to install local applications or configure your development machine.

Create an AWS Cloud9 development environment on a new Amazon EC2 instance or connect it to your own Linux server through SSH. Once you've created an AWS Cloud9 environment, you will have immediate access to a rich code editor, integrated debugger, and built-in terminal with pre-configured AWS CLI – all within your browser.

Using the AWS Cloud9 dashboard, you can create and switch between many different AWS Cloud9 environments, each one containing the custom tools, runtimes, and files for a specific project.

Steps :

1. Create an AWS Cloud9 development environment on a new Amazon EC2 instance

2. Open the AWS Cloud9 environment.

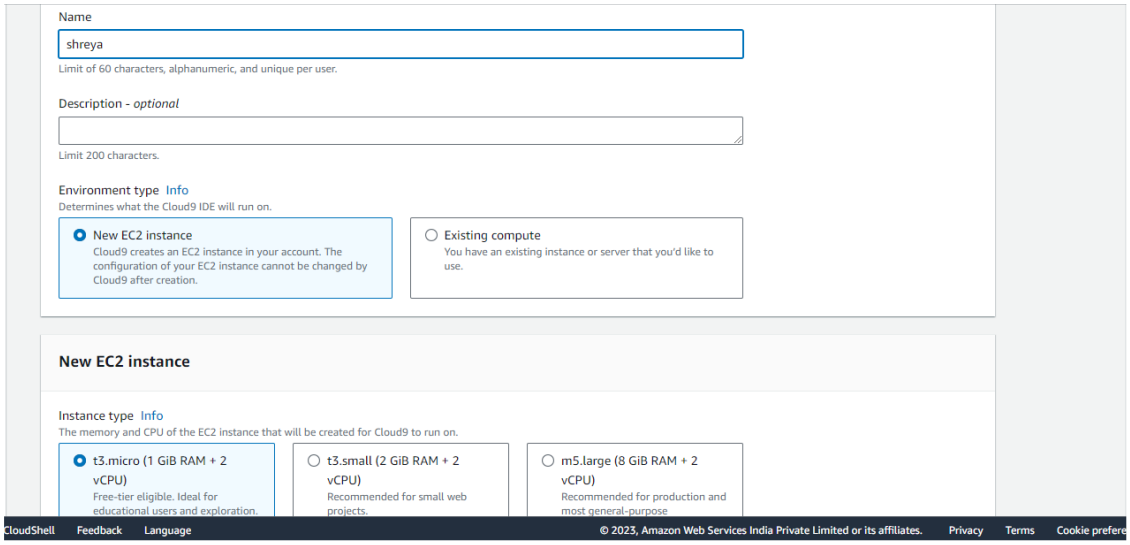
3. You will have immediate access to a rich code editor, integrated debugger, and built-in

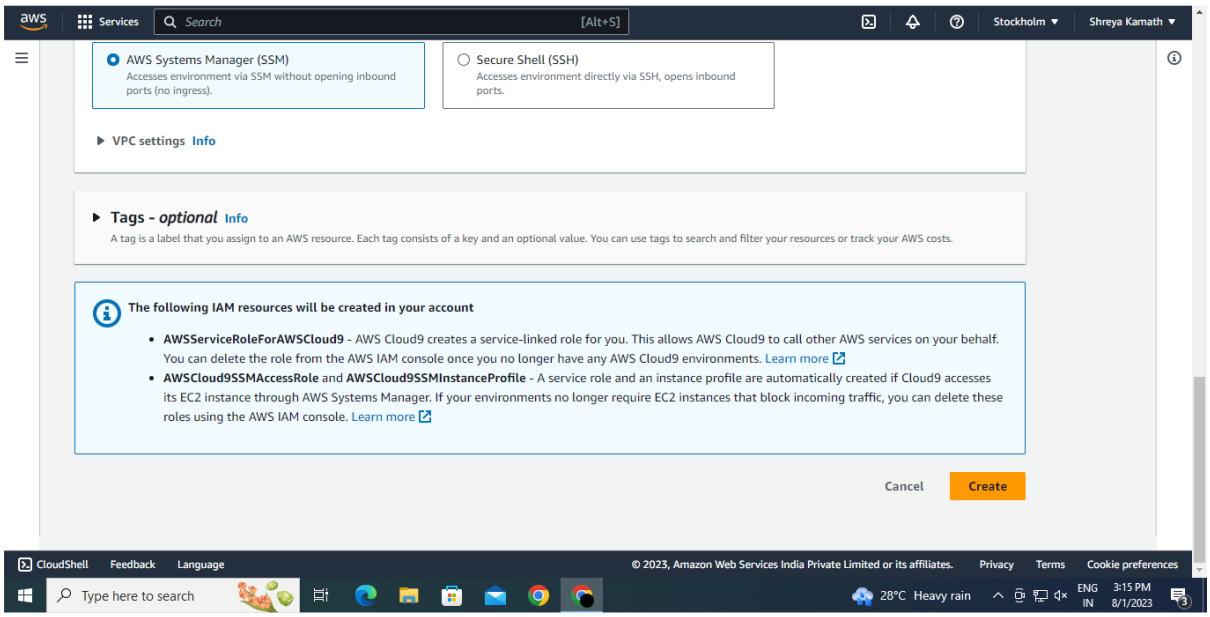
terminal with pre-configured AWS CLI – all within your browser.

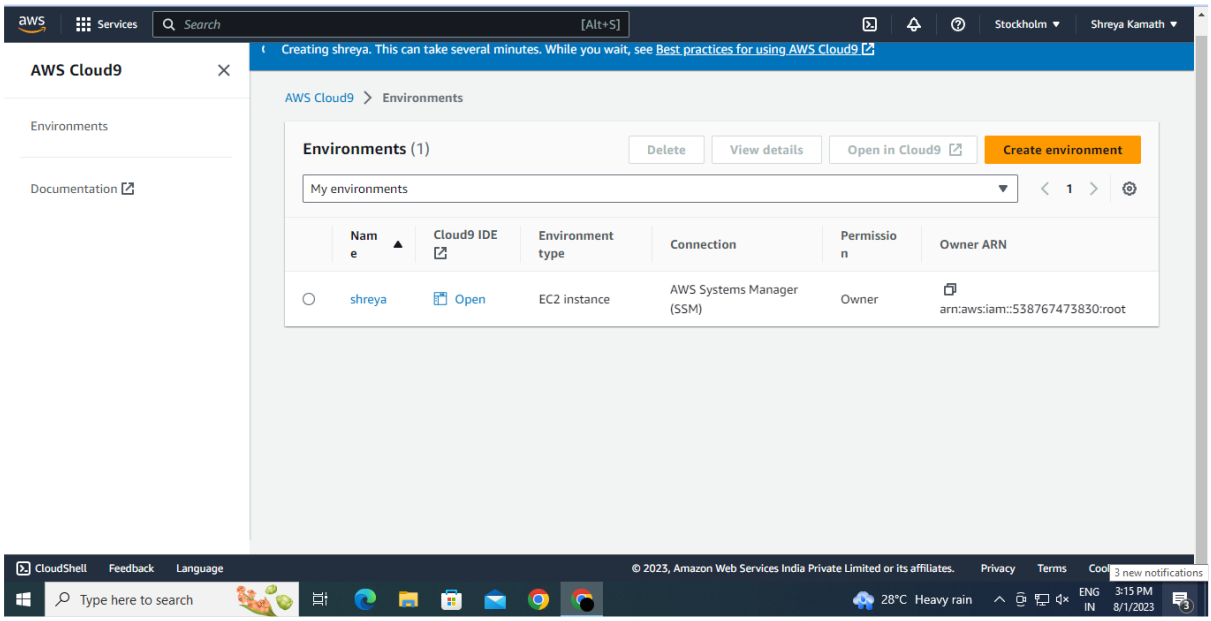
4. Write a code in any language and save it.

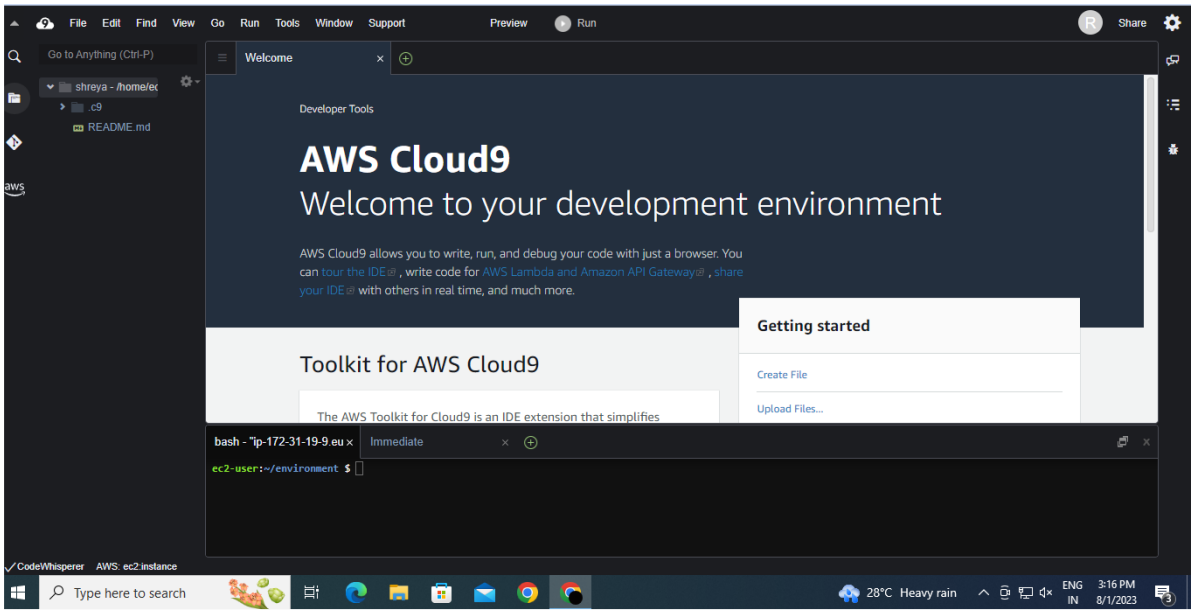
5. Run the file.

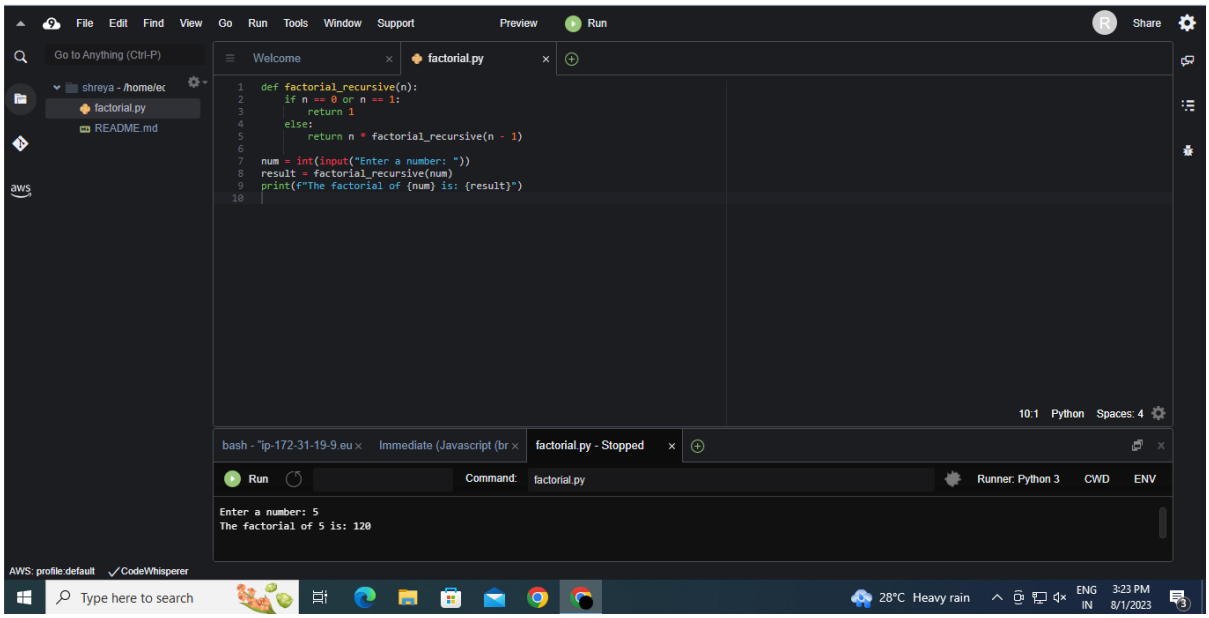
6. Delete the AWS Cloud9 environment.

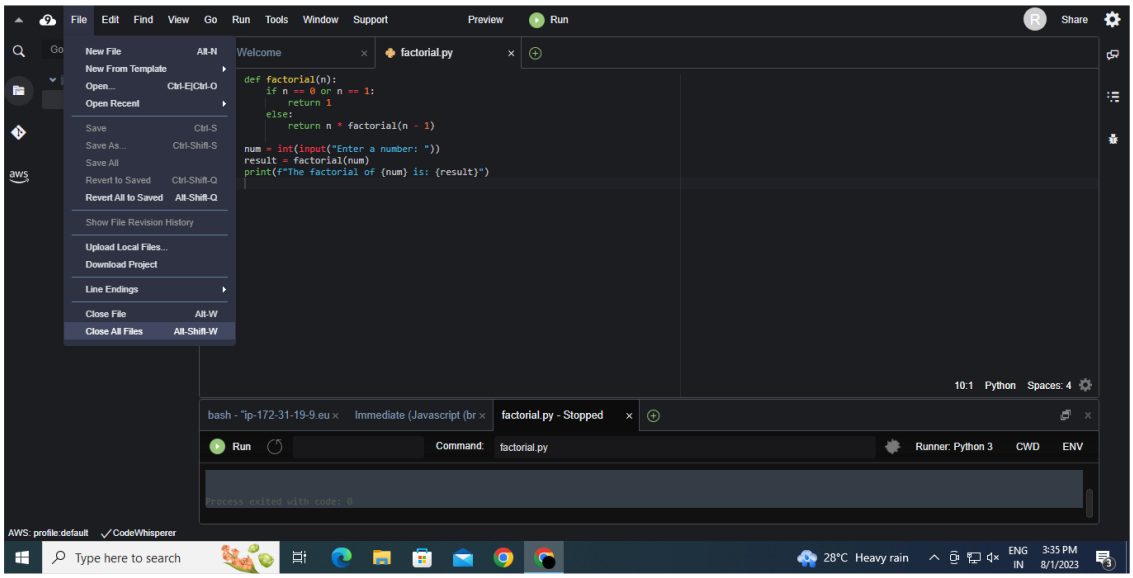

Name
shreya
Limit of 60 characters, alphanumeric, and unique per user.
Description - optional
Limit 200 characters.
Environment type Info
Determines what the Cloud9 IDE will run on.
New EC2 instance
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.
Existing compute
You have an existing instance or server that you'd like to use.
New EC2 instance
Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.
t3.micro (1 GiB RAM + 2 vCPU)
Free-tier eligible. Ideal for educational users and exploration.
t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.
m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie prefere

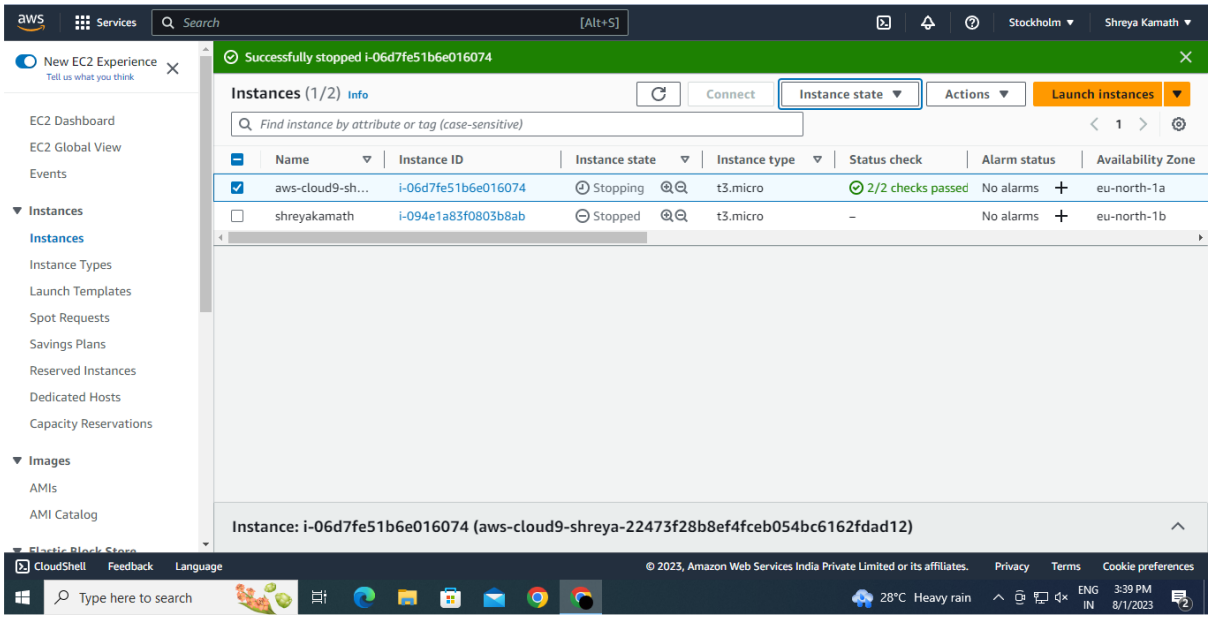

aws Services Search [Alt+S] Stockholm Shreya Kamath
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Tags - optional Info
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
The following IAM resources will be created in your account
AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more
AWSCloud9SSMAccessRole and AWSCloud9SSMInstanceProfile - A service role and an instance profile are automatically created if Cloud9 accesses its EC2 instance through AWS Systems Manager. If your environments no longer require EC2 instances that block incoming traffic, you can delete these roles using the AWS IAM console. Learn more
Cancel
Create
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences
Type here to search
28°C Heavy rain
ENG IN
3:15 PM 8/1/2023


aws Services Search [Alt+S] Stockholm Shreya Kamath
AWS Cloud9
Environments
Documentation
Creating shreya. This can take several minutes. While you wait, see Best practices for using AWS Cloud9
AWS Cloud9 > Environments
Environments (1)
Delete
View details
Open in Cloud9
Create environment
My environments
Name
Cloud9 IDE
Environment type
Connection
Permission
Owner ARN
shreya
Open
EC2 instance
AWS Systems Manager (SSM)
Owner
arn:aws:iam::538767473830:root
CloudShell Feedback Language
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms
3 new notifications
Type here to search
28°C Heavy rain
ENG IN
3:15 PM 8/1/2023


File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
shreya - /home/ec
.c9
README.md
Welcome
Developer Tools
AWS Cloud9
Welcome to your development environment
AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can tour the IDE, write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.
Getting started
Create File
Upload Files...
Toolkit for AWS Cloud9
The AWS Toolkit for Cloud9 is an IDE extension that simplifies
bash - "ip-172-31-19-9.eu
Immediate
ec2-user:~/environment $
CodeWhisperer AWS: ec2:instance
Type here to search
28°C Heavy rain
ENG IN 3:16 PM 8/1/2023


File Edit Find View Go Run Tools Window Support Preview Run Share
Go to Anything (Ctrl-P)
shreya - /home/ec
factorial.py
README.md
Welcome
factorial.py
def factorial_recursive(n):
if n == 0 or n == 1:
return 1
else:
return n * factorial_recursive(n - 1)
num = int(input("Enter a number: "))
result = factorial_recursive(num)
print(f"The factorial of {num} is: {result}")
10:1 Python Spaces: 4
bash - "ip-172-31-19-9.eu
Immediate (Javascript (br
factorial.py - Stopped
Run
Command: factorial.py
Runner: Python 3 CWD ENV
Enter a number: 5
The factorial of 5 is: 120
AWS: profile:default CodeWhisperer
Type here to search
28°C Heavy rain
ENG IN 3:23 PM 8/1/2023


File Edit Find View Go Run Tools Window Support Preview Run Share
New File Alt-N
New From Template
Open... Ctrl-E|Ctrl-O
Open Recent
Save Ctrl-S
Save As... Ctrl-Shift-S
Save All
Revert to Saved Ctrl-Shift-Q
Revert All to Saved Alt-Shift-Q
Show File Revision History
Upload Local Files...
Download Project
Line Endings
Close File Alt-W
Close All Files Alt-Shift-W
Welcome
factorial.py
def factorial(n):
if n == 0 or n == 1:
return 1
else:
return n * factorial(n - 1)
num = int(input("Enter a number: "))
result = factorial(num)
print(f"The factorial of {num} is: {result}")
10:1 Python Spaces: 4
bash - "ip-172-31-19-9.eu
Immediate (Javascript (br
factorial.py - Stopped
Run
Command: factorial.py
Runner: Python 3 CWD ENV
AWS: profile:default CodeWhisperer
Type here to search
28°C Heavy rain
ENG IN 3:35 PM 8/1/2023



Lab Outcome :

LO1 is mapped.

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

Sanskruti Adap

T11-1

Assignment 2

AIM: To study AWS S3 service and create a bucket for hosting static web applications.

THEORY:

AWS Simple Storage Service (S3) from the aforementioned list, S3, is the object storage service provided by AWS. It is probably the most commonly used, go-to storage service for AWS users given the features like extremely high availability, security, and simple connection to other AWS Services. An Amazon S3 bucket can be set up to operate similarly to a website. This section illustrates how to host a website using Amazon S3. There are mainly 7 steps to hosting a static website using Amazon Web Service(AWS) S3.

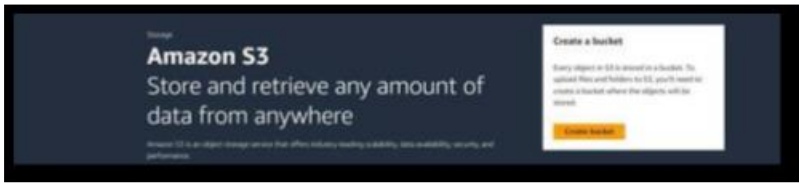
Step 1: Creating a Bucket

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

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

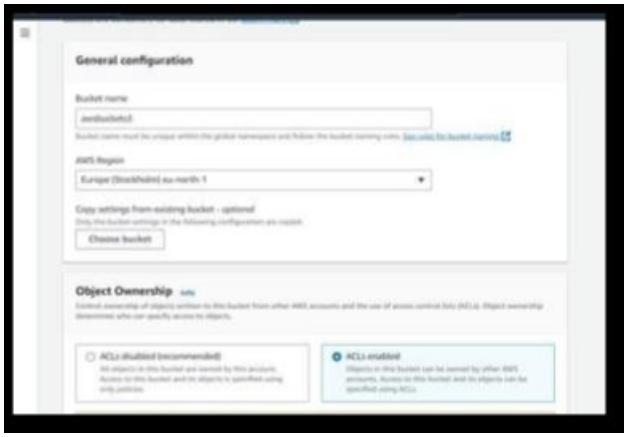
https://console.aws.amazon.com/s3/.

3. Click on Create Bucket.



4. Choose Bucket Name – Bucket Name Should be Unique

5. Object Ownership – Enable for making Public, Otherwise disable



Step 2: Block Public Access settings for the bucket

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

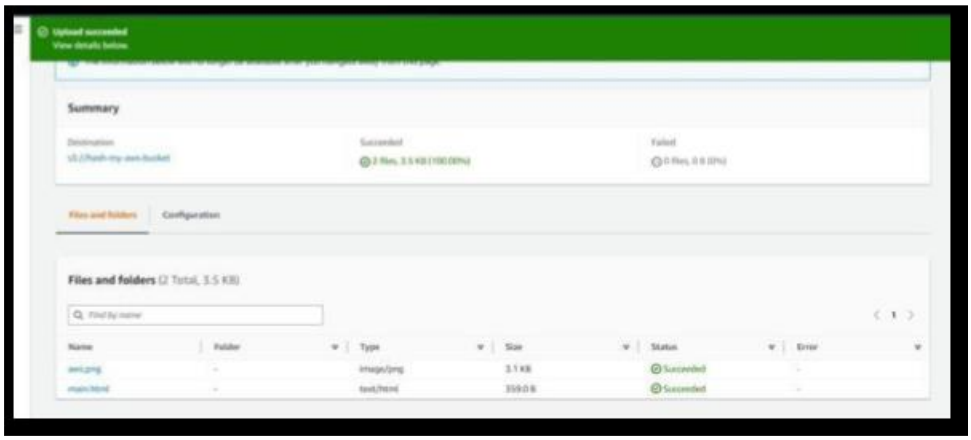
2. Now click on create bucket

3. Bucket is created

Step 3: Now upload code files

Select Bucket and Click your Bucket Name.

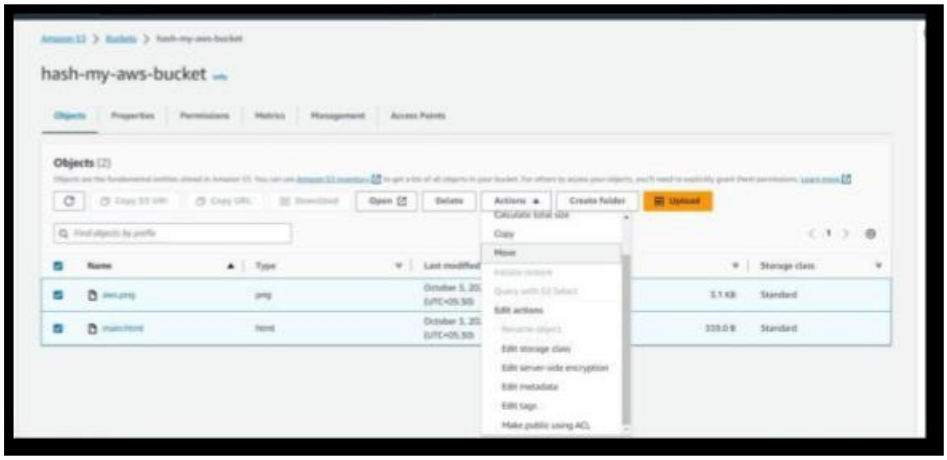
Now, click on upload (then click add File/folder) and select your HTML code file from your PC/Laptop.



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

- Block public access
- Object Ownership

- Make public Object



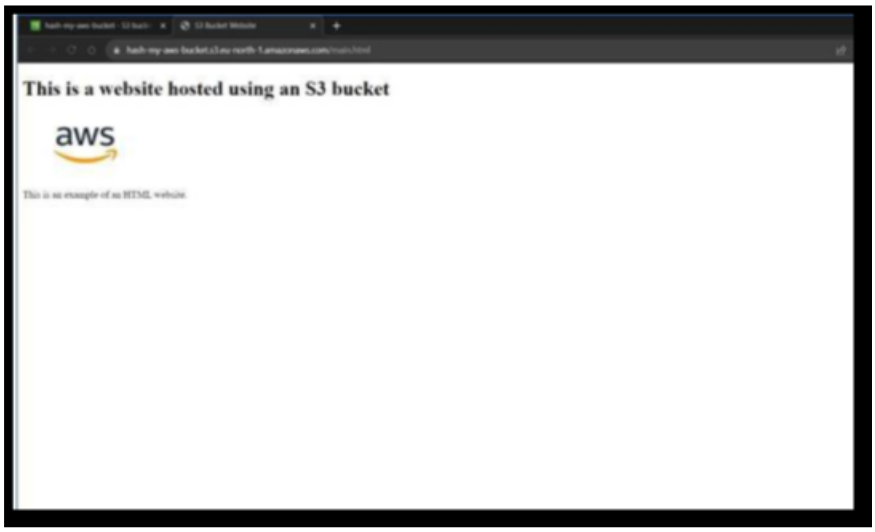
Step 5: Copy your Object URL

Now, click on your HTML File Object Name. Copy

the Object URL.

Step 6: Check out your Website!

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



CONCLUSION:-

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

Sanskruti Adap

T11-1

Assignment No. 3

Aim: To complete study of Kuberenetes.

Theory:

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

Kubernetes is a complex container orchestration platform, and it relies on several services and components that run on nodes to manage and control containerized applications. These services work together to provide the functionality required for deploying, scaling, and maintaining applications. Here are some of the key Kubernetes services running on nodes, along with their roles:

Kubelet:

Role: The Kubelet is an essential component that runs on each node in the cluster. It is responsible for ensuring that containers are running in a Pod. It takes Pod specifications (from the API server) and makes sure the containers described in the Pod are running and healthy on the node.

Container Runtime (e.g., Docker, containerd, or CRI-O):

Role: The container runtime is responsible for pulling container images, running containers, and managing their lifecycle. It's the software that actually runs containerson the node.

Kube Proxy:

Role: Kube Proxy is responsible for network connectivity in the cluster. It maintains network rules on each node to route traffic to the appropriate pods based on services and labels. It helps provide load balancing and service discovery.

CNI Plugins (Container Network Interface):

Role: CNI plugins are responsible for setting up and managing the network connectivity for containers in the pod. These plugins configure the network interfaces, IP addresses, and routes for containers so they can communicate with each other and with external networks.Node Status and Heartbeat (NodeStatus and NodeHeartbeat):

Role: These services are responsible for sending the current status and health of the node to the control plane. They provide information about the node's available resources, conditions, and any issues it might be facing.

Docker Daemon (if using Docker):

Role: If you're using Docker as the container runtime, the Docker Daemon manages container images, storage, and the execution of containers.

Containerd (if using Containerd):

Role: Containerd is an alternative container runtime that performs similar functions toDocker. It is used when you opt for a more lightweight runtime compared to Docker.

Kubelet Registrar and Garbage Collector:

Role: These components help with registering nodes with the control plane and ensuring that nodes and their resources are cleaned up and properly managed.

Node OS (Operating System):

Role: The underlying node OS is responsible for providing the necessary environment for running containers. It includes the kernel, system libraries, and utilities required for containers to function.

Systemd or Init System:

Role: The init system (e.g., systemd) on the node is responsible for managing system processes and ensuring that the Kubernetes services are started at boot time and restarted if they fail.

- What is Pod Distribution Budget?

A Pod Distribution Budget is a concept in Kubernetes that allows you to specify constraints and requirements on how pods should be distributed across nodes in a cluster. It is used to control the distribution of pods to ensure that they are evenly spread across nodes or follow specific rules for placement. This is particularly useful for enhancing high availability and fault tolerance, improving resource utilization, and ensuring compliance with policies or regulations.

Pod Distribution Budgets are part of the Pod Topology Spread Constraints feature,which was introduced in Kubernetes to provide more control over how pods are scheduled on nodes.

Here are the key components of a Pod Distribution Budget:

Topology Key: A topology key is a label or field key that is used to determine the distribution of pods across nodes. For example, it could be the availability zone or region of a node.

Max Skew: Max Skew is a numerical value that represents the maximum allowed imbalance in the distribution of pods. For example, if the Max Skew is set to 1, it means that the difference in the number of pods across nodes should not exceed 1.

Topology Spread Constraints: These are rules that specify how pods should be distributed based on the topology key. For instance, you can define constraints to ensure that pods are distributed evenly across nodes in different availability zones.

Pod Distribution Budgets are created by defining a CustomResourceDefinition (CRD) object in Kubernetes. You can then reference this object when creating or updating Deployments, StatefulSets, or other types of workload controllers. The Pod Distribution Budget constraints are checked during the scheduling process, ensuring that pods are placed on nodes in compliance with the defined rules.

By using Pod Distribution Budgets, you can improve the resilience and stability of your applications by avoiding overloading specific nodes and ensuring that your workloads are evenly distributed across your cluster, which is essential for achieving a balanced and efficient Kubernetes cluster.

- What is the role of loadbalance in Kubernetes?

Load balancing in Kubernetes plays a crucial role in ensuring the availability, scalability, and reliability of applications and services running within the cluster. Here'san
overview of the role of load balancing in Kubernetes:

Service Accessibility: Kubernetes abstracts the underlying infrastructure, which means that pods and services can be distributed across different nodes in the cluster. Load

balancing is used to ensure that these services remain accessible to clients regardless of the specific node or pod where they are running.

Service Discovery: Kubernetes Services allow you to define a stable DNS name or IP address for a set of pods. Load balancers route traffic to the appropriate pods associated with a service based on selectors, labels, and other criteria, making it easy for clients to discover and connect to services without needing to know the specific details of pod locations.

Scaling: Kubernetes allows for automatic scaling of pods based on CPU or other resource usage metrics. Load balancers distribute traffic across these dynamically changing instances, ensuring that traffic is evenly distributed even as pods are added or removed in response to demand.

High Availability: Load balancers provide high availability by distributing traffic across multiple replicas of an application. If one pod or node fails, the load balancer can route traffic to healthy instances, minimizing downtime.

Security: Load balancers can serve as a point of entry for traffic into the cluster, providing a layer of security by controlling which traffic is allowed and which is not. They can also be used to terminate SSL/TLS encryption, ofloading the decryption process from the application.

Traffic Splitting: Load balancers support advanced traffic management features like A/B testing and canary deployments. They allow you to route a percentage of traffic to one version of an application and a different percentage to another version for testing or
gradual rollouts.

Global Load Balancing: For multi-cloud or multi-region deployments, Kubernetes can utilize global load balancers to route traffic to the nearest or most available data center or cloud region. This improves latency and redundancy.

Request Routing and Path-Based Routing: Load balancers can route traffic based on different criteria, such as the path in the URL. This enables microservices architectures where different paths are handled by different services.

Kubernetes itself does not provide a load balancer but integrates with various load balancing solutions, such as external cloud load balancers, NodePort services, ClusterIP services, and Ingress controllers. The choice of load balancer depends on the specific requirements and the underlying infrastructure used in your Kubernetes cluster.

Conclusion:

In this assignment we learnt what exactly are kubernetes.

Sanskruti Adap

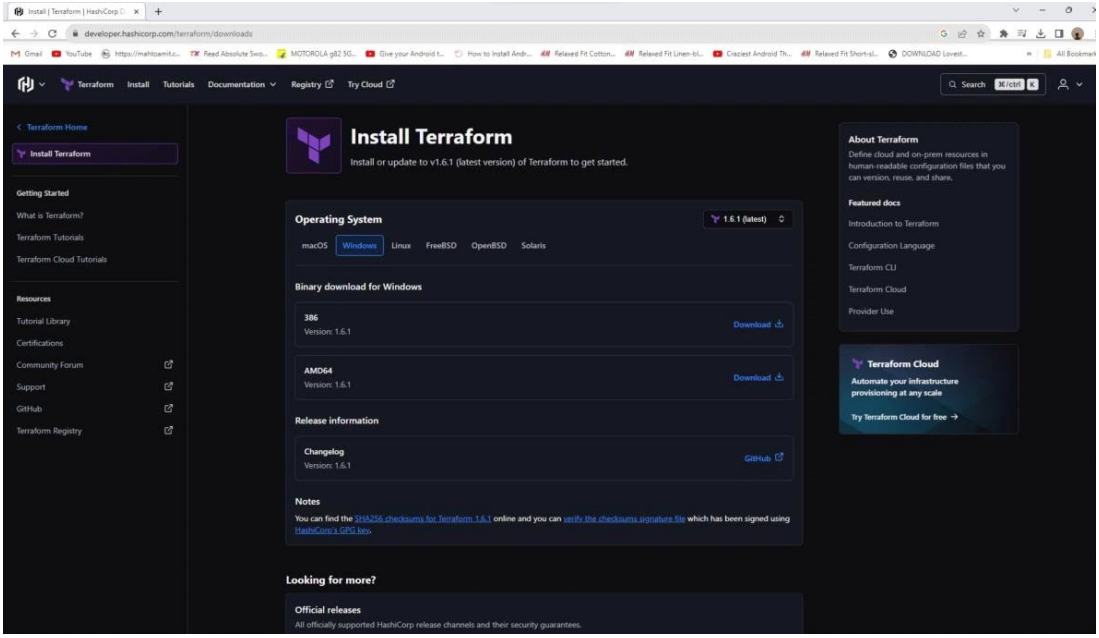
T11-1

Assignment No. 4

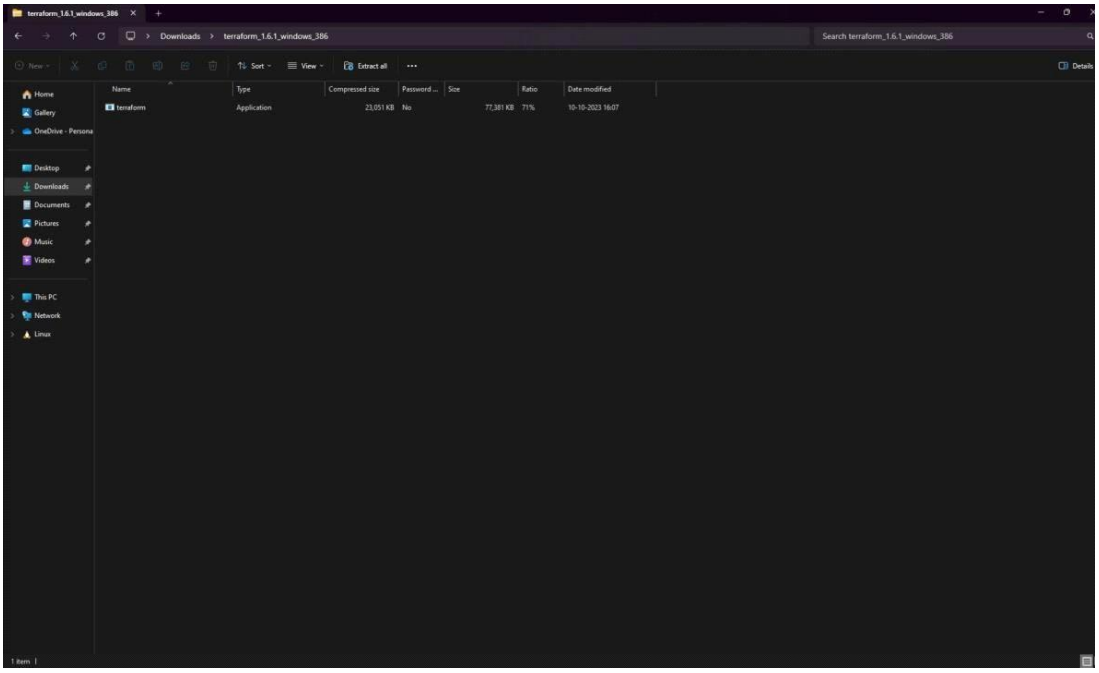
Aim: To understand the installation process of terraform.

Theory:

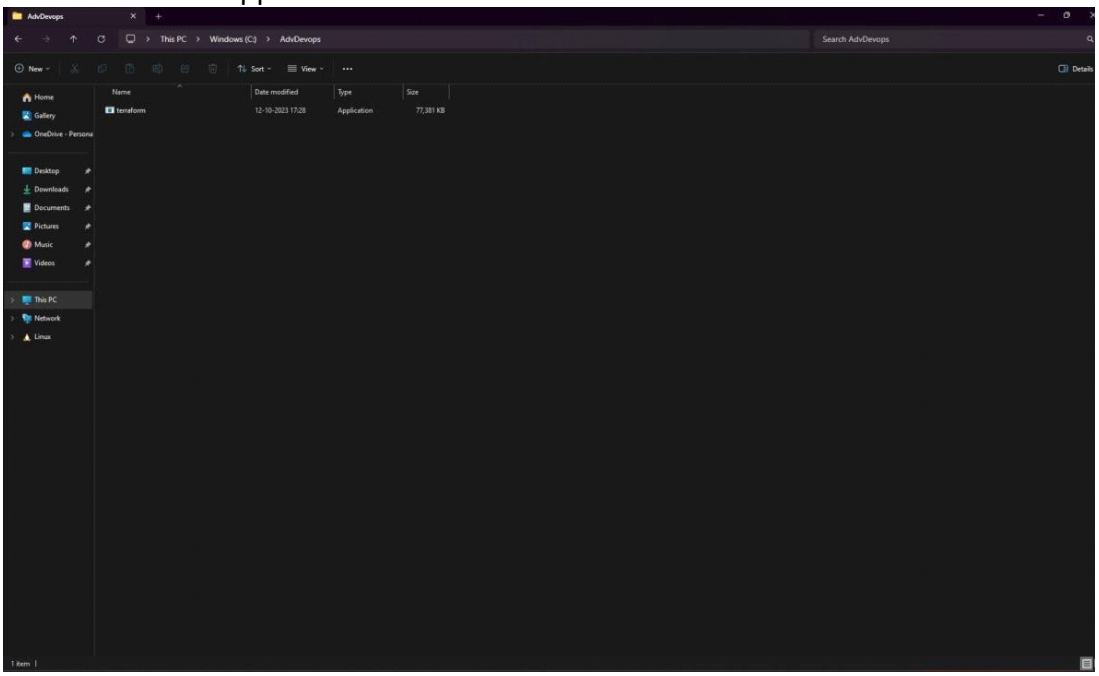
Go to the browser and search terraform download. Go to the first website "developer.hashicorp.com/terraform/downloads". Select the operating system ofyour choice.



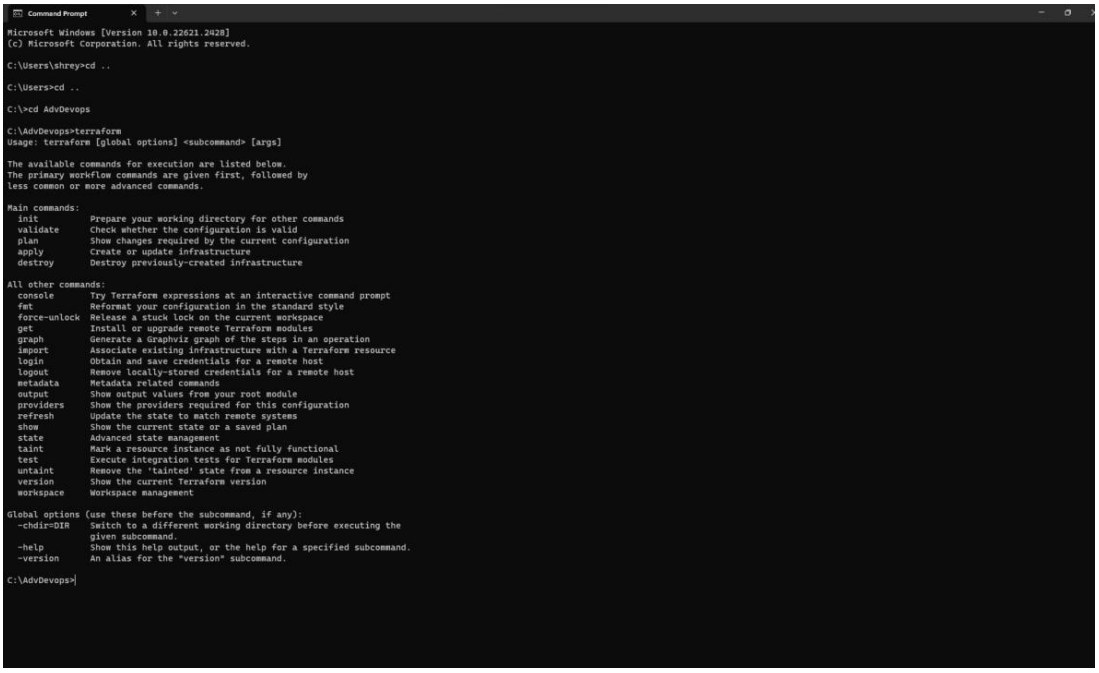
Download any version that is suitable with your operating system. Once you finish downloading go to the Downloads folder and open the zip file.



Store this application in a folder in C drive.



Once it is done go to system libraries and add this application path to system variables so that you can use terraform scripts in your system.



We can see that terraform path has been set up. This completes ou rinstallation ofterraform.

Conclusion:

In this assignment we learnt how to install terraform and use it torun scripts.

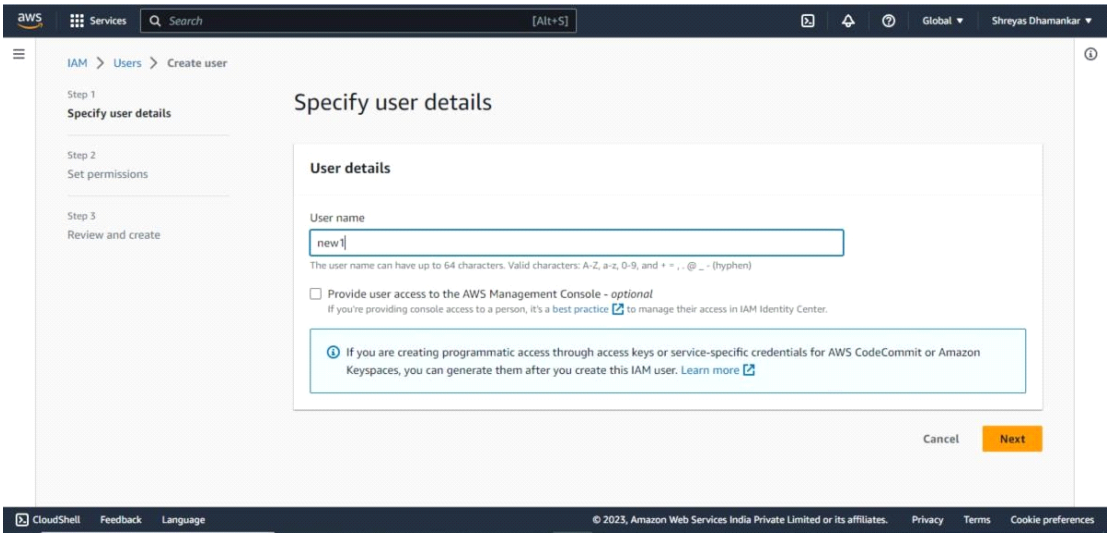
Sanskruti Adap

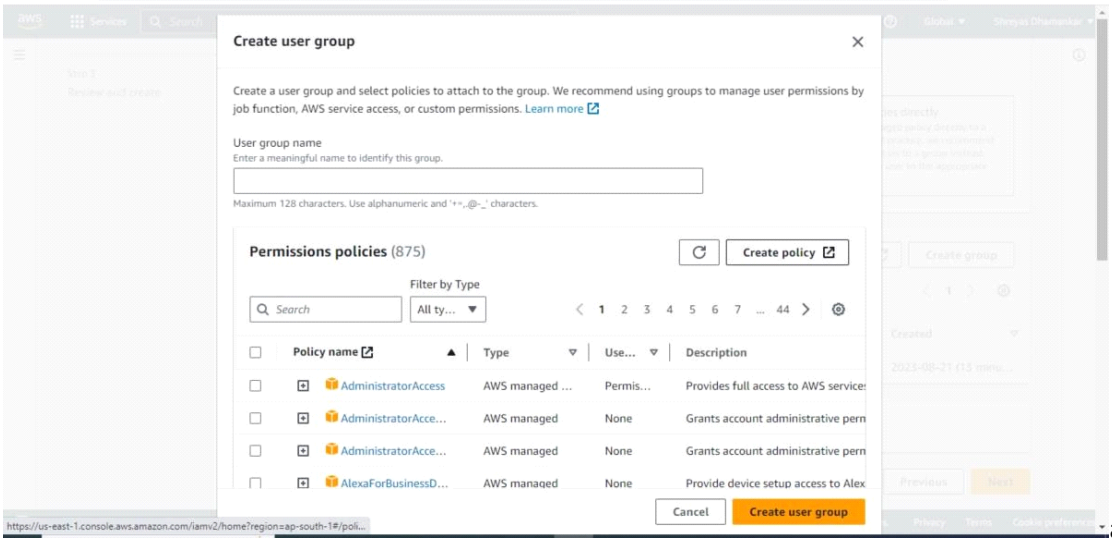
T11-1

Assignment No. 5

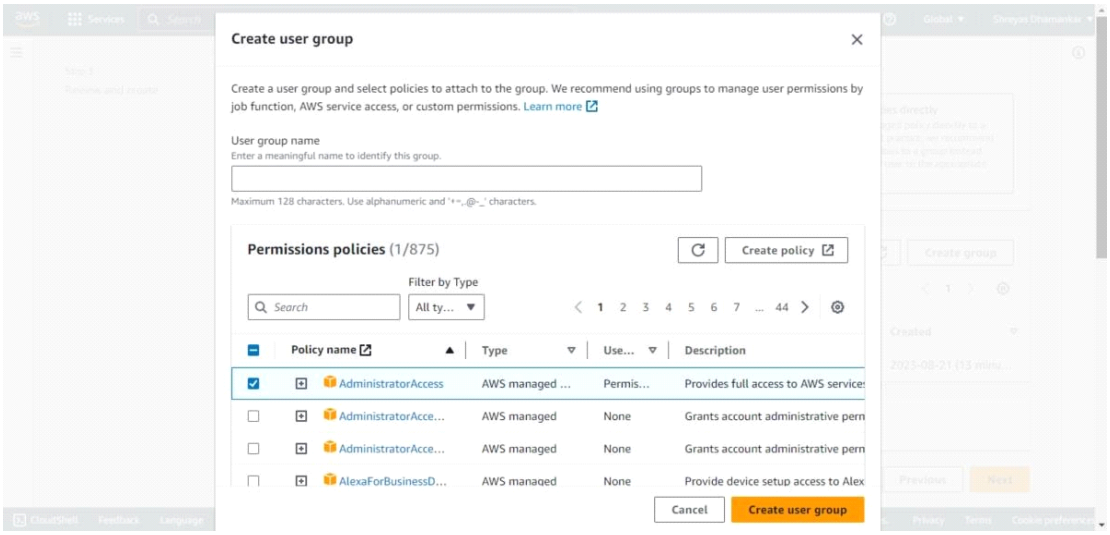
Aim: To understand the concept of terraform and how to use it to create an instance.

Theory:

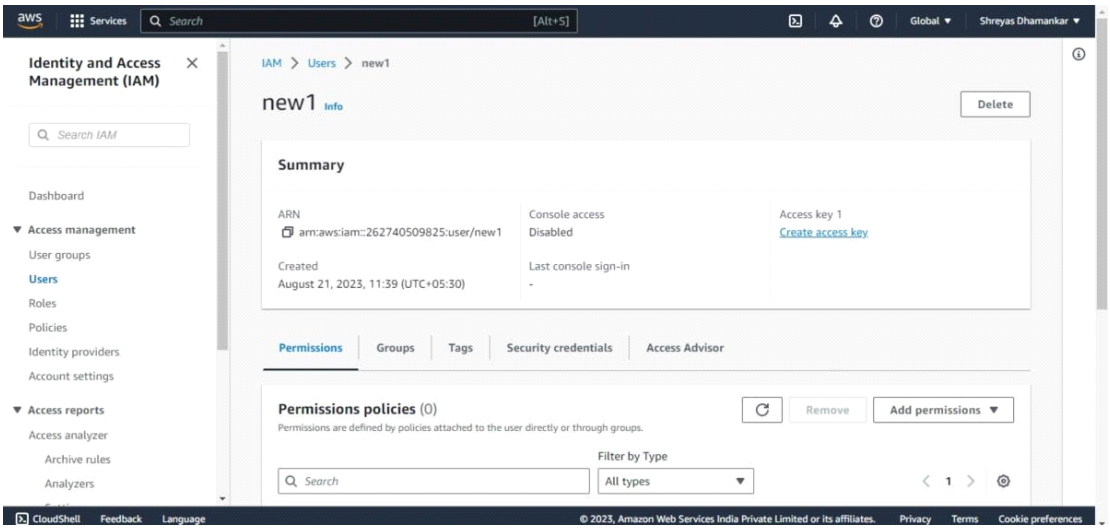




add a new user group with permission to have administrator access



then go to user and select create new access key. after selecting select create CLI and dowload the csv.



write this script and save it in Terraform Script folder. use the acces keys and secret key presentin the csv files.

open cmd and choose the path where you have stored the script and run the following commands init,plan,apply and destroy.

Command Prompt

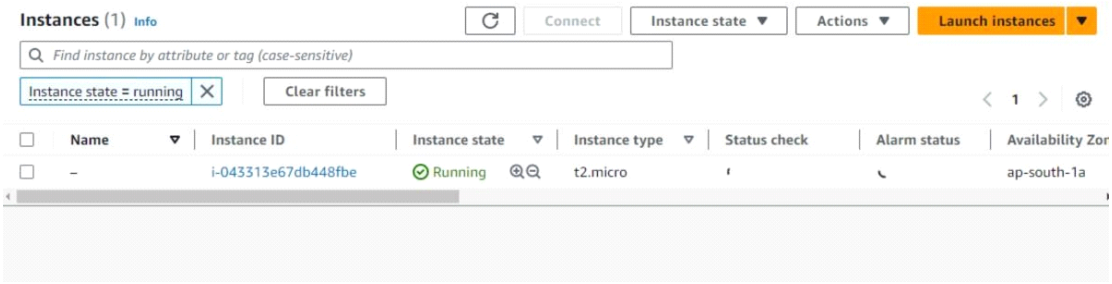
```
      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/sda1" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-0798195950b794d7d" -> 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.Ubuntu: Destroying... [id=i-09e2dcb327124c5bf]
aws_instance.Ubuntu: Still destroying... [id=i-09e2dcb327124c5bf, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-09e2dcb327124c5bf, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-09e2dcb327124c5bf, 30s elapsed]
aws_instance.Ubuntu: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.

C:\Terraform Script>
```

```
C:\Terraform Script>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.0...
- Installed hashicorp/aws v5.13.0 (signed by HashiCorp)

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

Terraform has been successfully initialized!

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

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

```
C:\Terraform Script>terraform plan

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

Terraform will perform the following actions:

  # aws_instance.Ubuntu will be created
  + resource "aws_instance" "Ubuntu" {
      + ami                                  = "ami-0f5ee92e2d63afc18"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_stop                     = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
      + host_resource_group_arn              = (known after apply)
      + iam_instance_profile                 = (known after apply)
      + id                                   = (known after apply)
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
```

```
      + instance_initiated_shutdown_behavior = (known after apply)
      + instance_lifecycle                   = (known after apply)
      + instance_state                       = (known after apply)
      + instance_type                        = "t2.micro"
      + ipv6_address_count                   = (known after apply)
      + ipv6_addresses                       = (known after apply)
      + key_name                             = (known after apply)
      + monitoring                           = (known after apply)
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + 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.
```

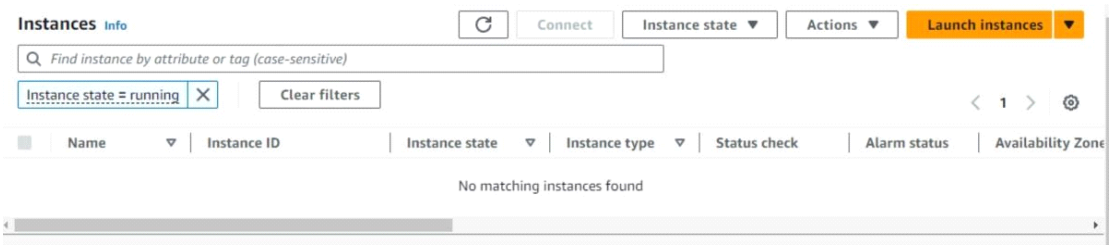
instance has been created.



```
C:\Terraform Script>terraform destroy
aws_instance.Ubuntu: Refreshing state... [id=i-043313e67db448fbe]

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

Terraform will perform the following actions:

  # aws_instance.Ubuntu will be destroyed
  - resource "aws_instance" "Ubuntu" {
      - ami                                  = "ami-0f5ee92e2d63afc18" -> null
      - arn                                  = "arn:aws:ec2:ap-south-1:262740509825:instance/i-043313e67db448fbe" -> null
      - associate_public_ip_address          = true -> null
      - availability_zone                    = "ap-south-1a" -> null
      - cpu_core_count                       = 1 -> null
      - cpu_threads_per_core                 = 1 -> null
      - disable_api_stop                     = false -> null
      - disable_api_termination              = false -> null
      - ebs_optimized                        = false -> null
      - get_password_data                    = false -> null
      - hibernation                          = false -> null
      - id                                   = "i-043313e67db448fbe" -> null
      - instance_initiated_shutdown_behavior = "stop" -> null
      - instance_state                       = "running" -> null
      - instance_type                        = "t2.micro" -> null
      - ipv6_address_count                   = 0 -> null
      - ipv6_addresses                       = [] -> null
      - monitoring                           = false -> null
      - placement_partition_number           = 0 -> null
      - primary_network_interface_id         = "eni-060f15a4b86e45f2f" -> null
      - private_dns                          = "ip-172-31-37-169.ap-south-1.compute.internal" -> null
      - private_ip                           = "172.31.37.169" -> null
      - public_dns                           = "ec2-3-111-147-14.ap-south-1.compute.amazonaws.com" -> null
      - public_ip                            = "3.111.147.14" -> 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.Ubuntu: Destroying... [id=i-043313e67db448fbe]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-043313e67db448fbe, 30s elapsed]
aws_instance.Ubuntu: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.
```

instance has been detroyed



once terraform destroy command has been run and completed you can delete all security keysand csv files and delete users and user groups.

Conclusion:

In this assignment we learnt how to install terraform and use it torun scripts.

Sanskruti Adap

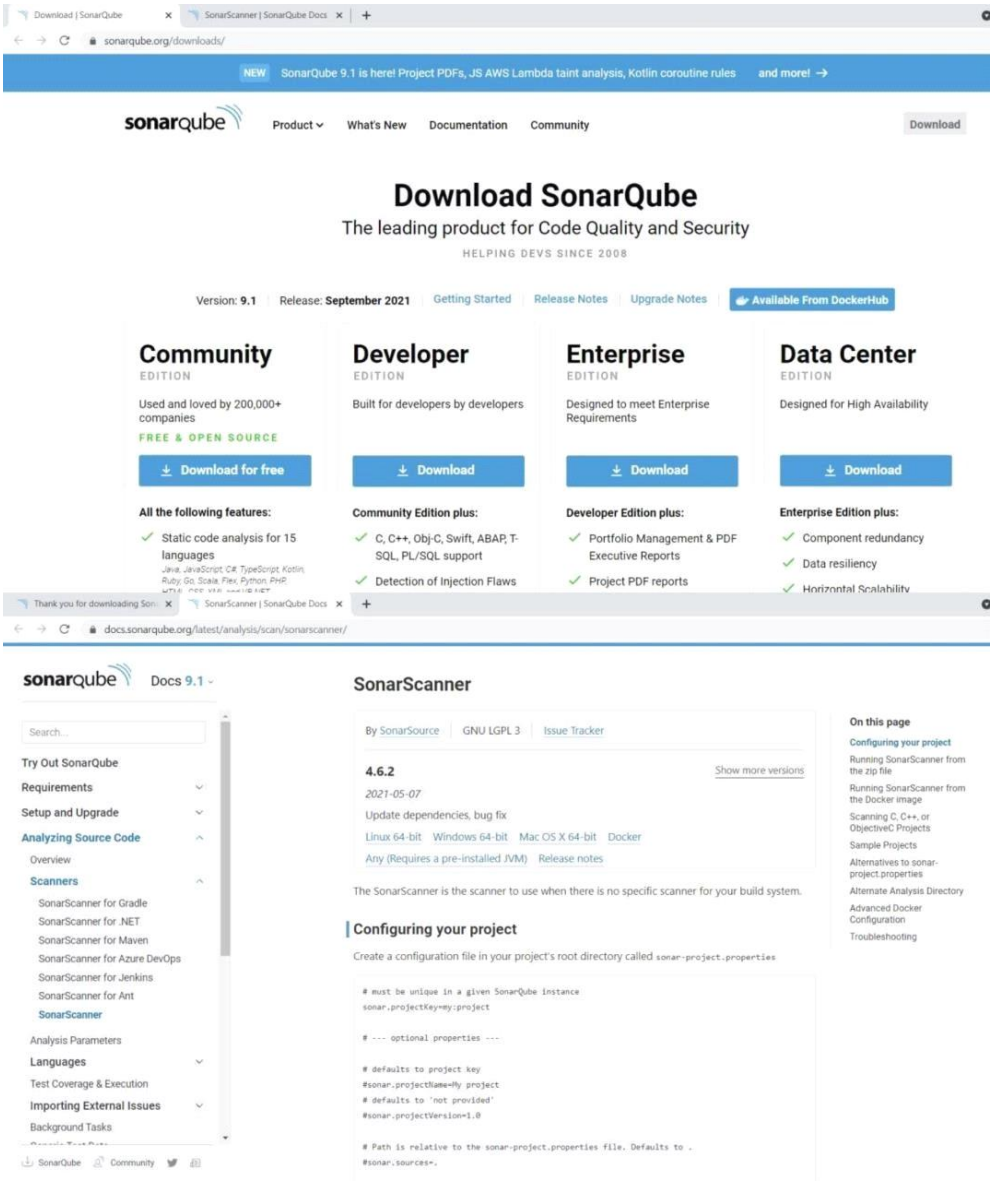
T11-1

Assignment no. 6

Aim: To understand the concept of sonarqube and sonar Scanner.

Theory:

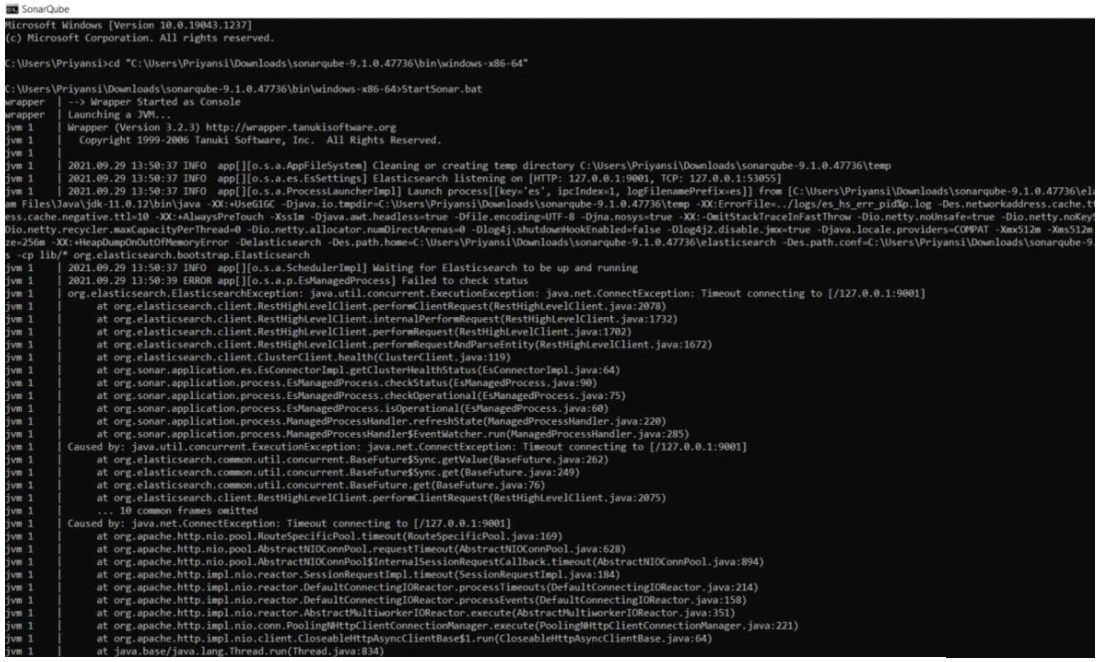
Download SonarQube and Sonar Scanner

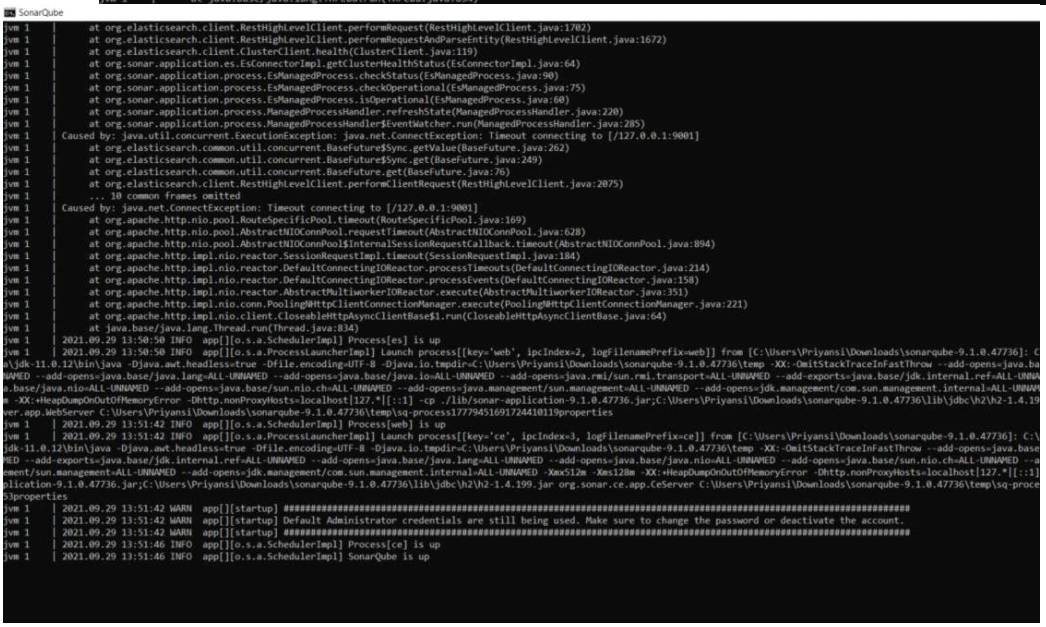


After downloading, set Environment Variables. Add “sonarqube-9.1.0.47736\bin” toPath.

Open command prompt. Run commands:

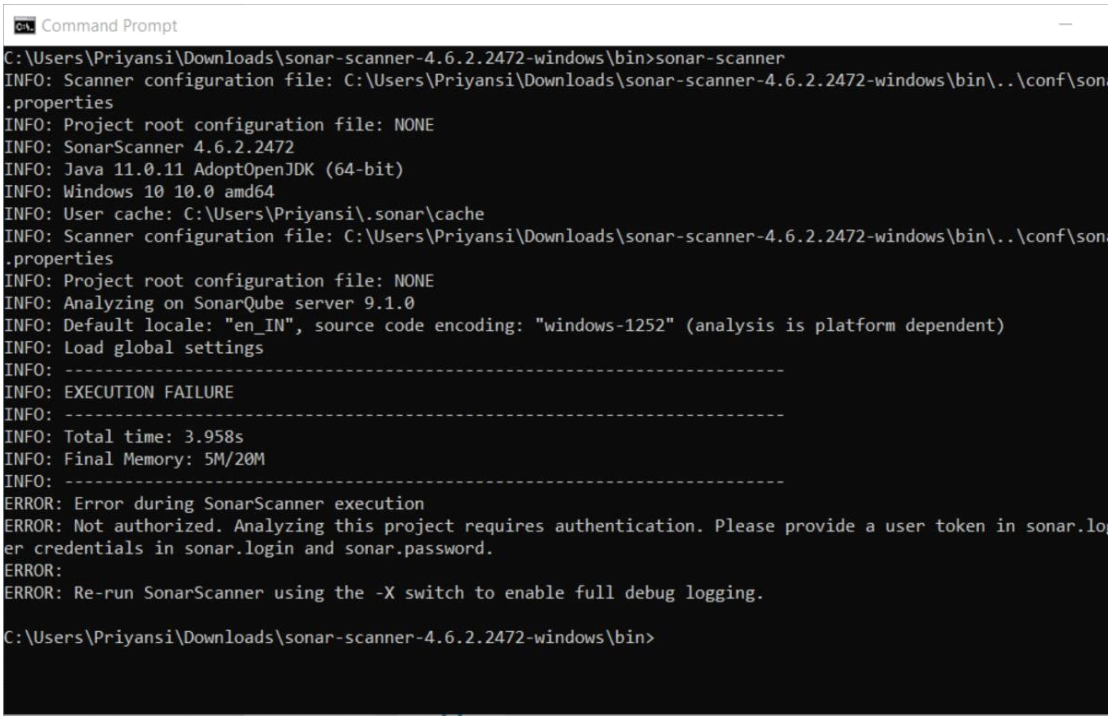
- cd “sonarqube-9.1.0.47736\bin\windows-x86-64”
- StartSonar.bat





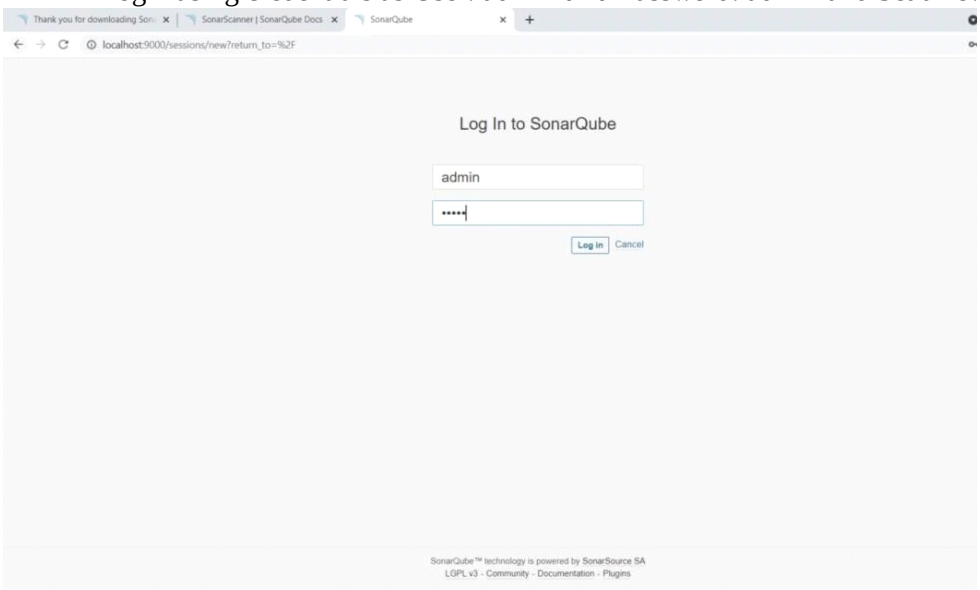
Open another command prompt. Run command:

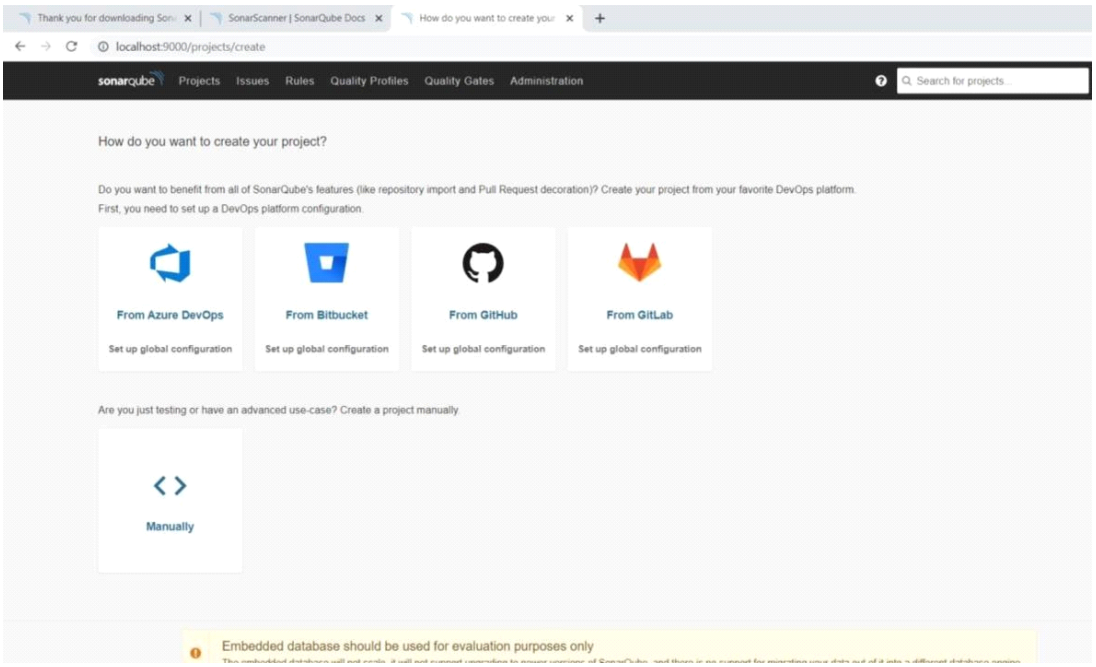
- cd “sonar-scanner-4.6.2.2472-windows\bin”
- sonar-scanner



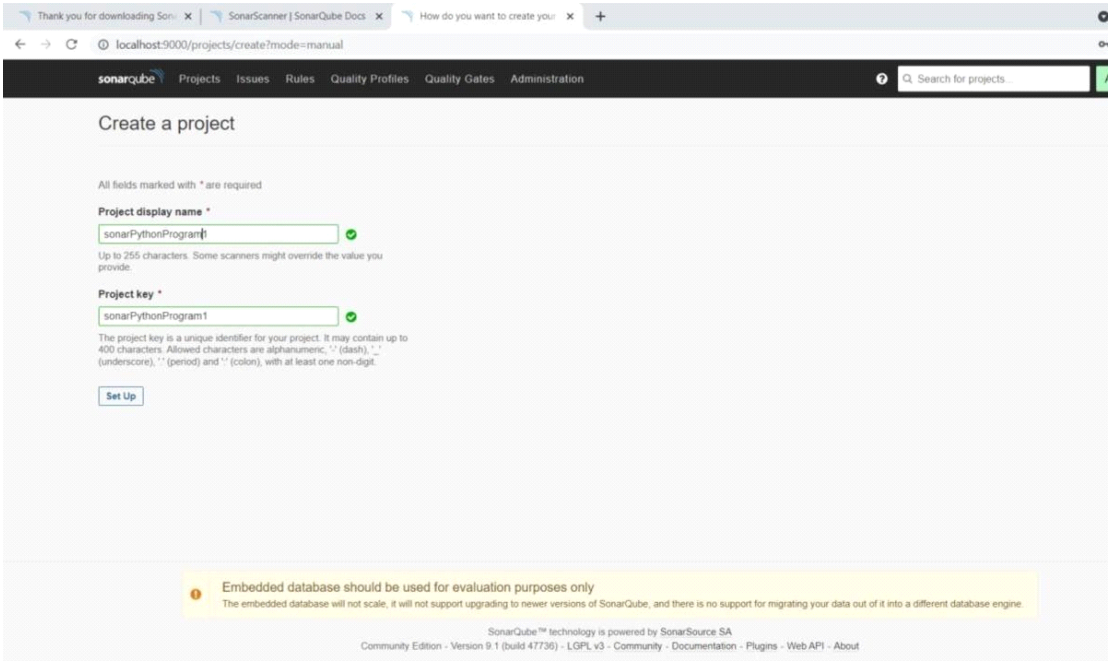
Server up and running on **localhost:9000**

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

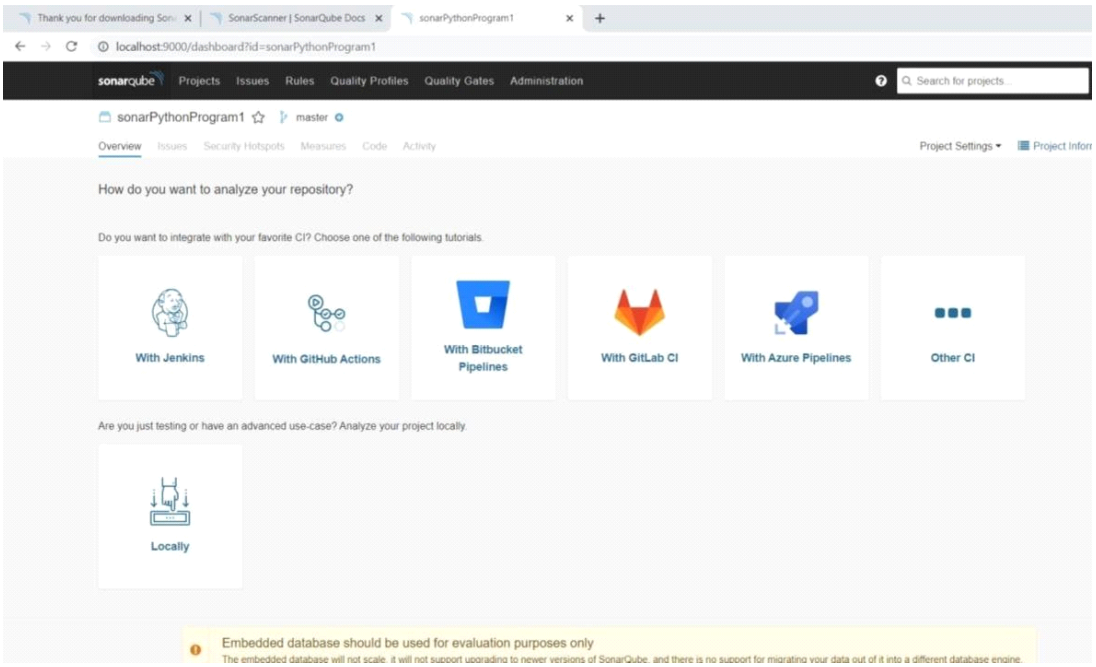




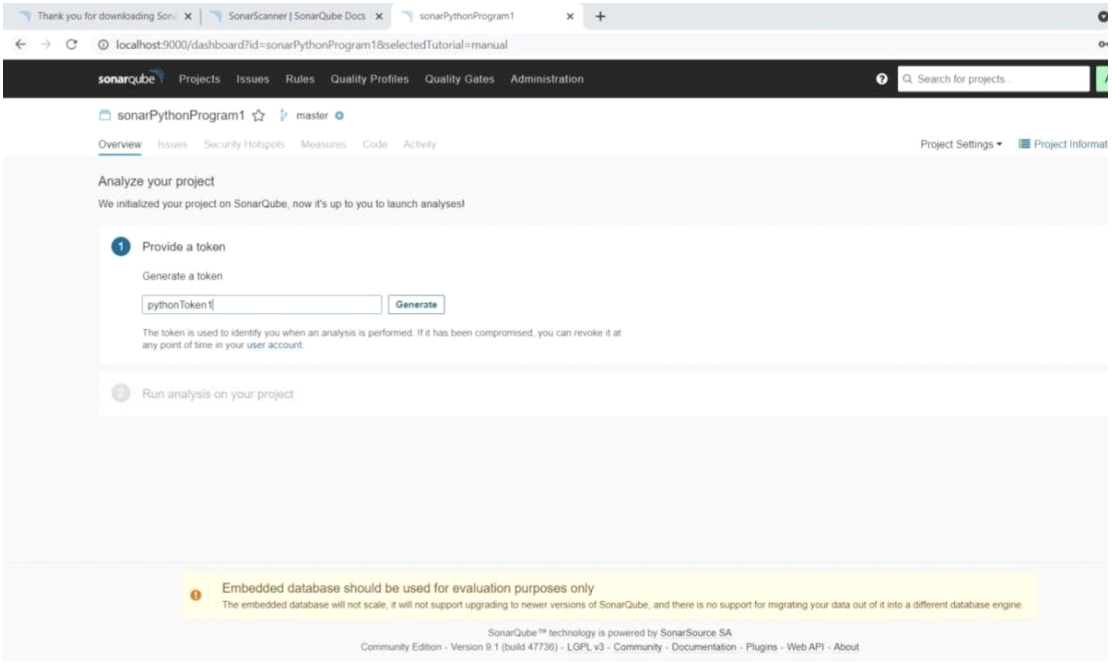
Click on Create a project **Manually.**



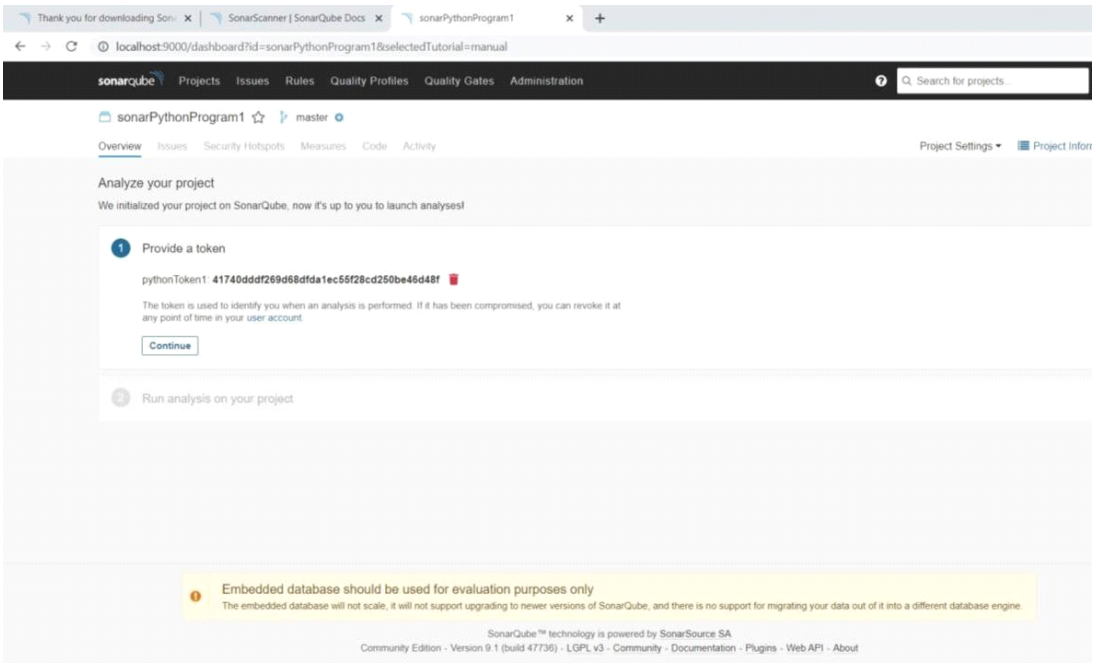
Give any Project display name.



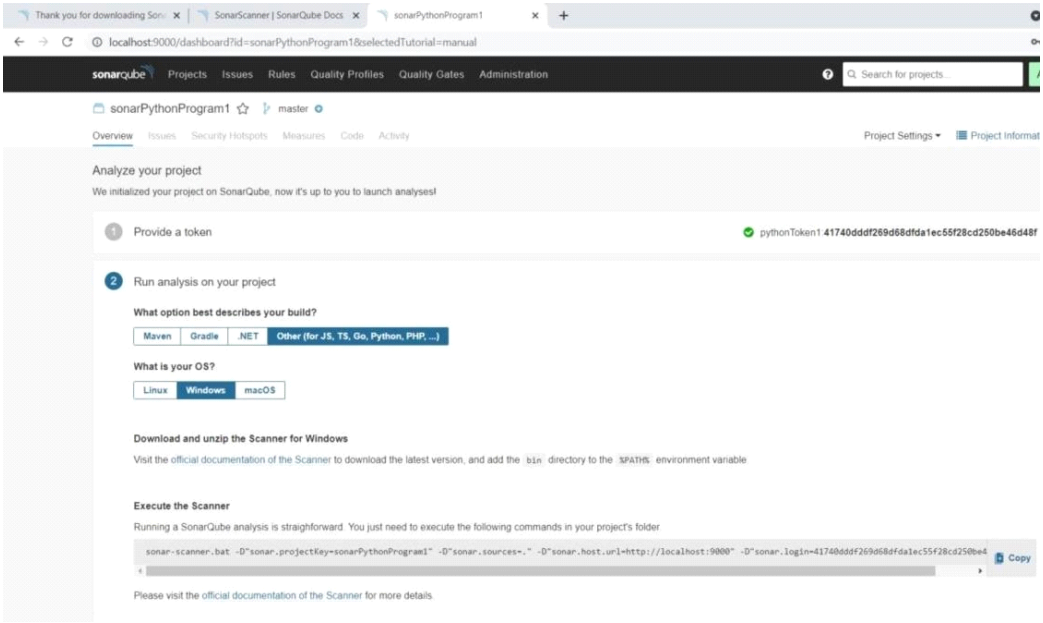
Click on **Locally.**



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



Click on **Continue.**



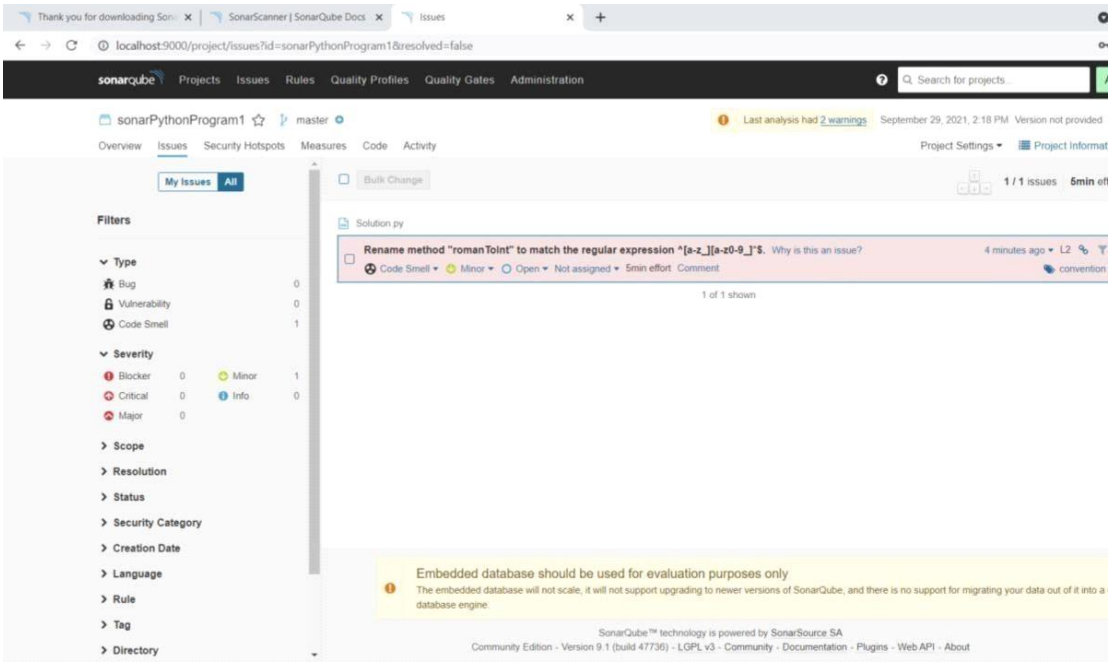
Save a Python program in a folder.

```
def checkIfRomanNumeral(numeral):
```

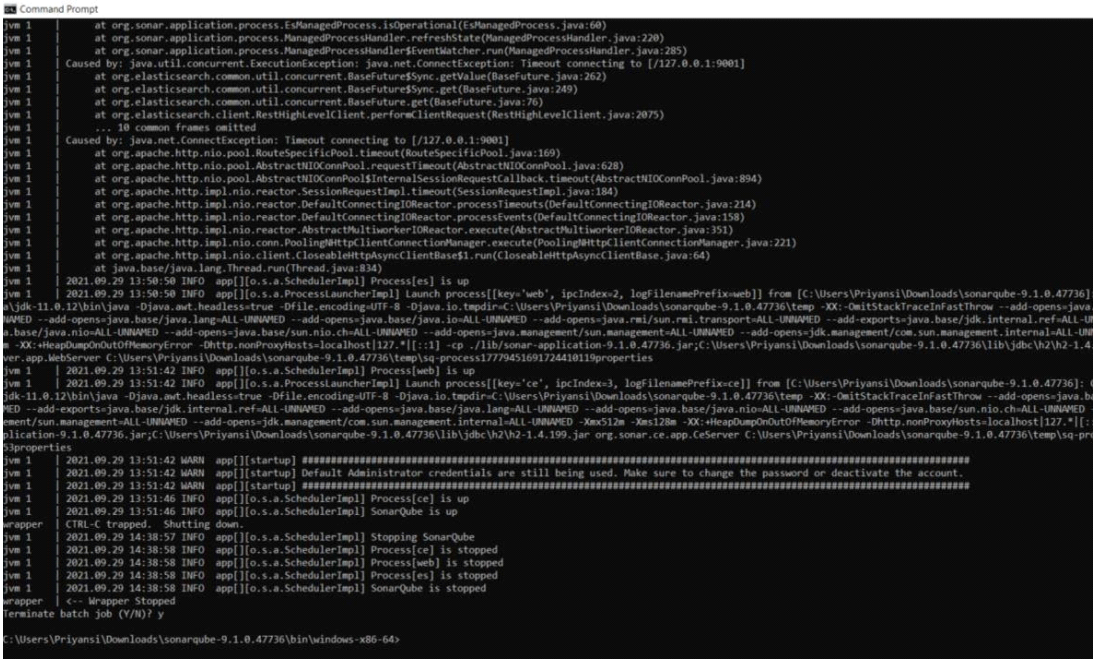
```
"""Controls that
the userinput
only contains
valid roman
numerals"""

    numeral
=
numeral.up
per()


validRomanNumerals

= ["M", "D", "C", "L",

"X", "V", "I", "(", ")"]

    for
letters in
numeral:

        if letters not
in
validRomanNumer
als:

            print("
Sorrythat is not
a valid roman
numeral")

            return
        Trueelif
        letters in

validRomanNumerals:


romanToInt(num
            eral)
            break
```

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.



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



Conclusion:

In this assignment we learnt how to use sonarqube and sonarscanner to find out errors in our code and how too debug them.

Sanskruti Adap

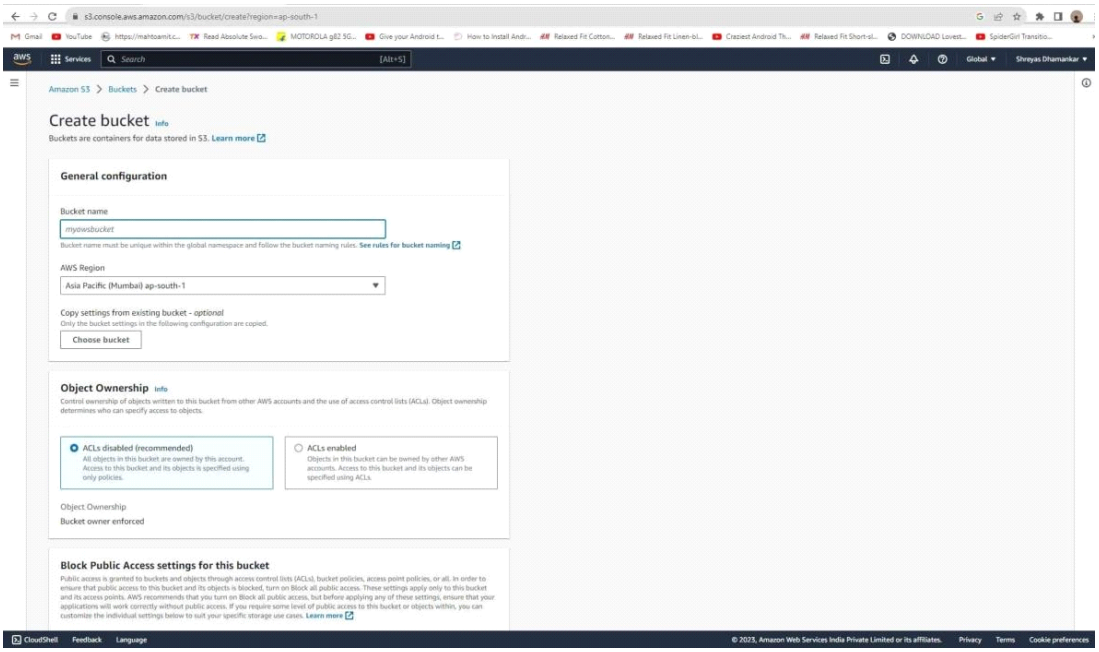
T11-1

Assignment No. 7

Aim: To learn how to use Lamda in order to run a simple program from S3 Bucket.

Theory:

Create bucket:

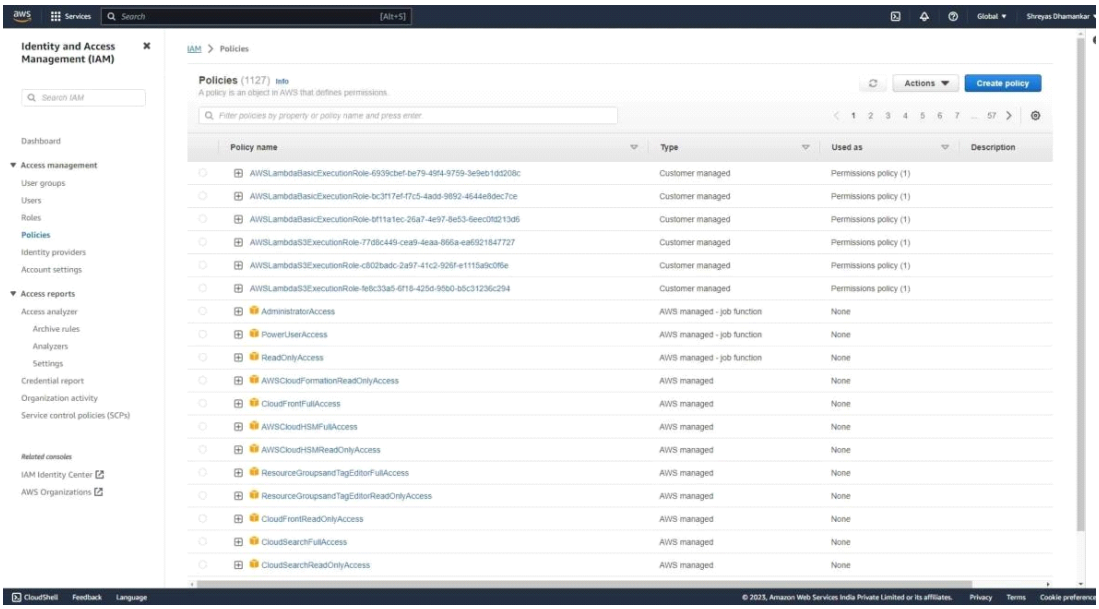


create a new policy from iam dashboard;

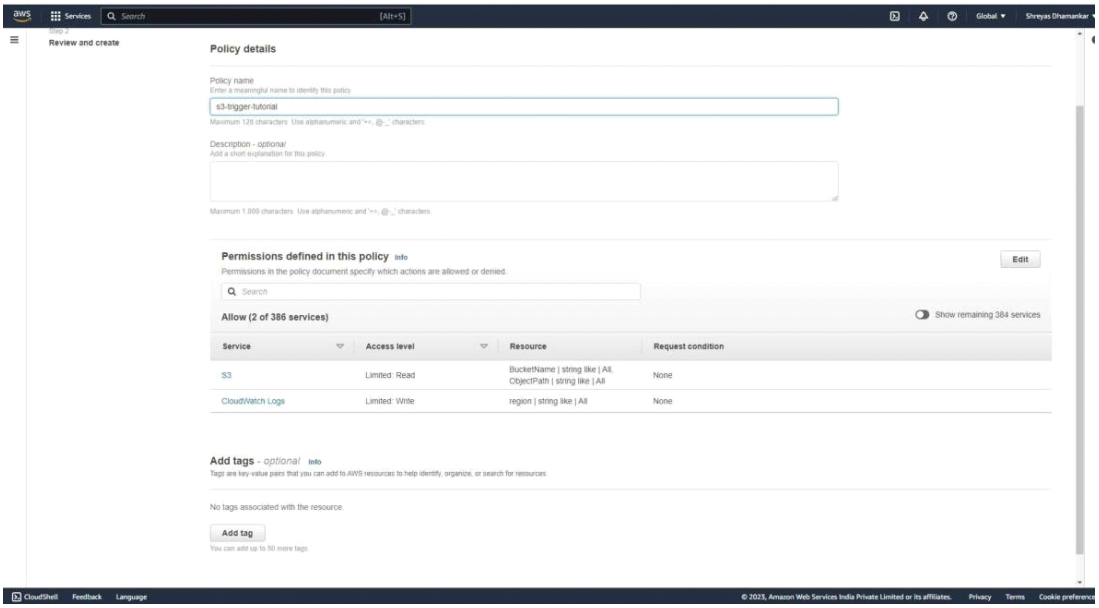
while creating policy select json tab and paste the following code:

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

      {
```

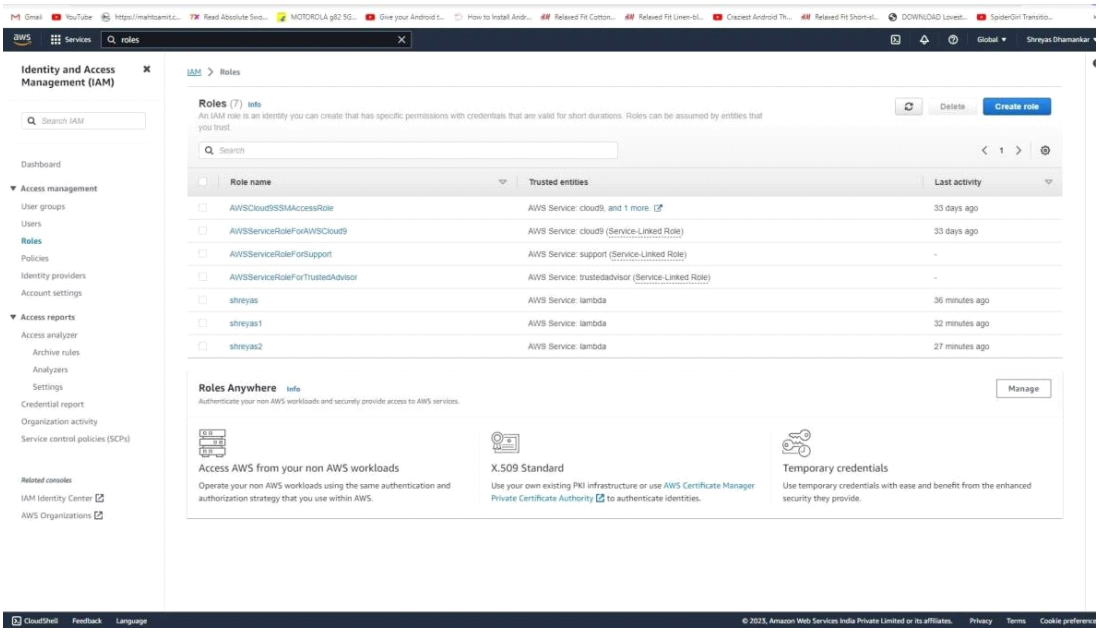
```
            "Effect":
            "Allow",
            "Action": [
                "logs:PutLog
                        Even
                        ts",
                "logs:CreateLo
                gGroup",
                "logs:CreateLog
                Stream"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Effect":
            "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::*/*"
        }
    ]
}
```



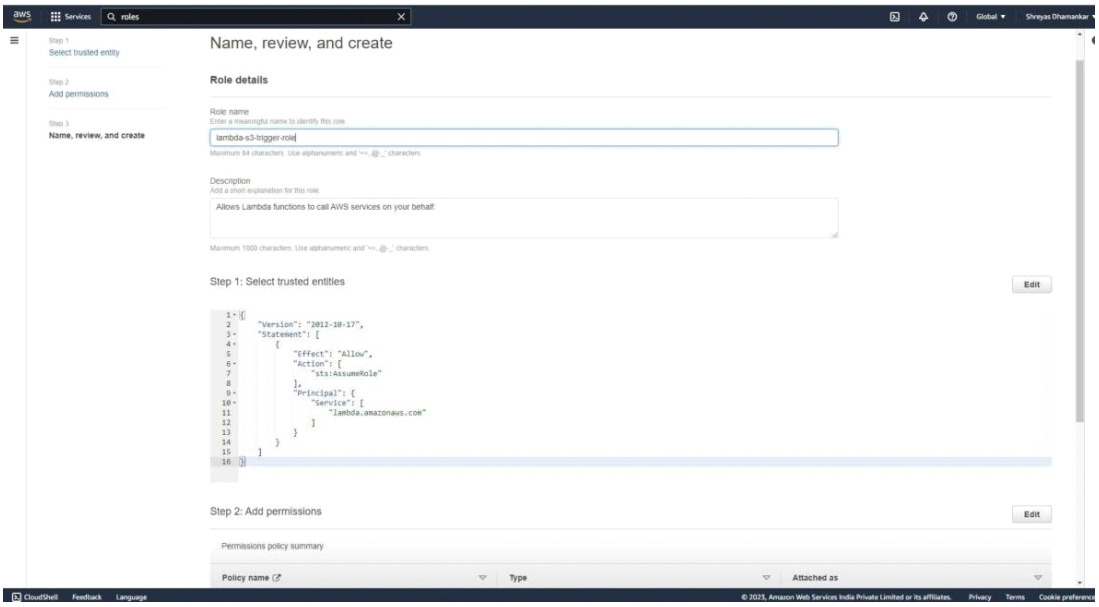
create policy and name it:



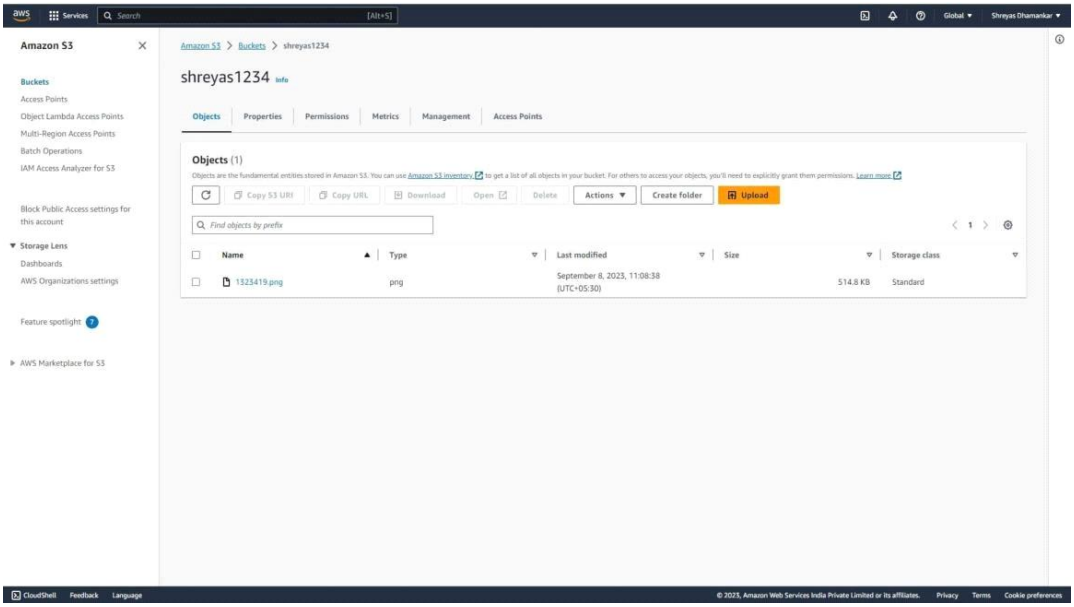
go to roles page and select create a new role



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

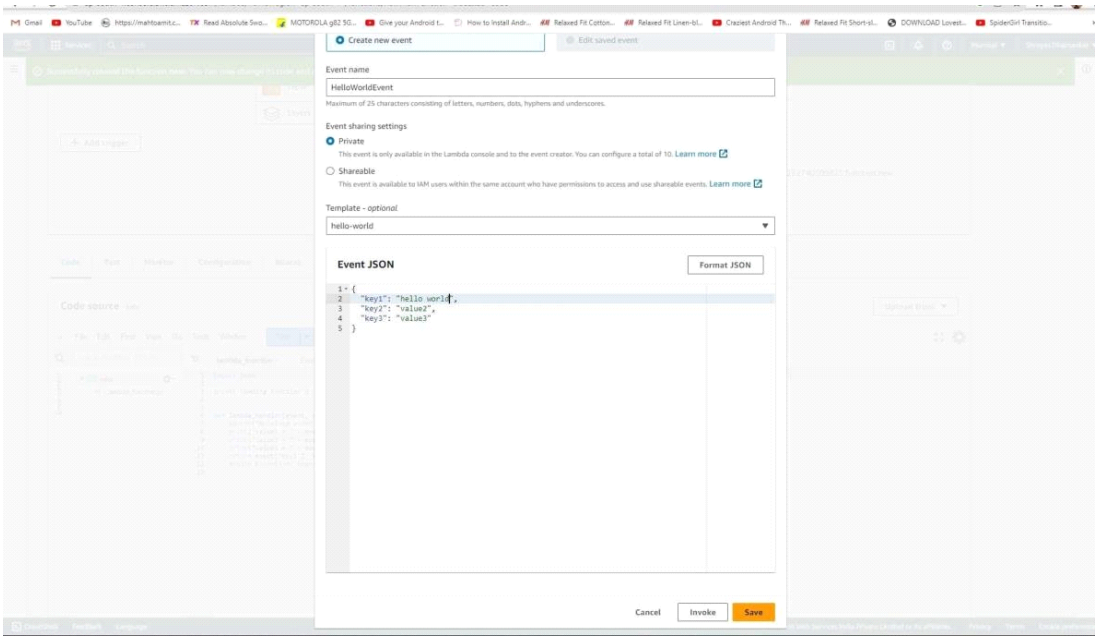


Upload an image file in the S3 bucket.



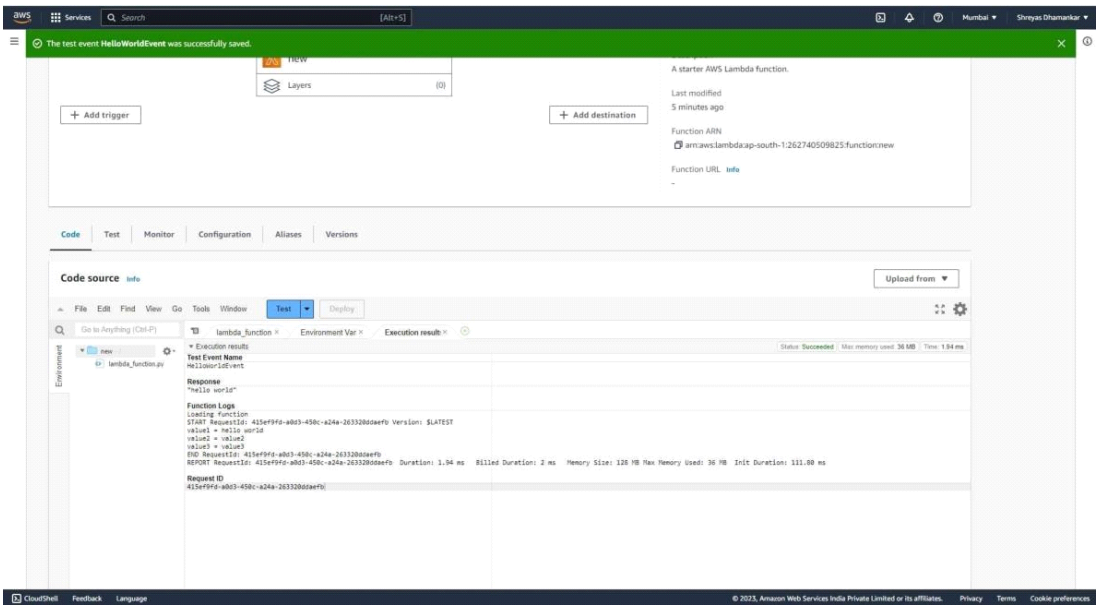
Go to lamda dashboard in aws and create a new function named s3-trigger tutorial. selectuse exisiting blueprint and choose 'hello world python 3.7 blueprint'.

CLick on create function. Once function is created go to test and create new Test event.



Change Key 1 to Hello World.

CLick on test and obtain the results in Execution tab.



Conclusion:

In this assignment we learnt how to use Lambda function to run abasic Hello World program in Python.

Sanskruti Adap

T11-1

Assignment No. 8

Aim: Understand and InstaII Nagios , Monitor host using Nagios

Objectives

- Understand Nagios
- Understand installation process for Nagios on Ubuntu
- Monitor Localhost using different parameters.

Theory:

What is Nagios?

Nagios is a free to use open source software tool for continuous monitoring. It helps you to monitor system, network, and infrastructure. It is used for continuous monitoring of
systems,applications, service and business process in a DevOps culture.

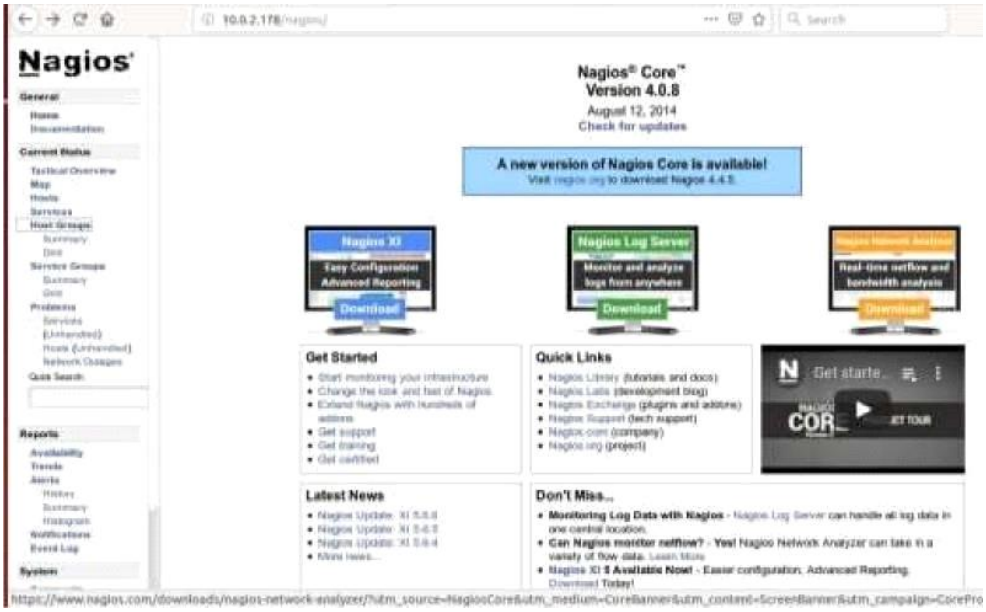
Nagios runs plugins stored on the same server. It plugin's connects with a host or another serve on your network or the lnternet. Therefore, in the case of failure Nagios core can alert the technical staff about the issues. So that, your technical team performs the recovery process before outage in the business processes.

Why We NeedNagios?

Here, are Important reasons to use Nagios monitoring tool are:

- Detects all types of network or server issues
- Helps you to find the root cause of the problem which allows you to get the permanent solution to the problem
- Active monitoring of your entire infrastructure and business processes

- Allows you to monitors and troubleshoot server performance issues
- Helps you to plan for infrastructure upgrades before outdated systems create failures
- You can maintain the security and availability of the service
- Automatically fix problems in a panic situation web interface



Click on Host Groups

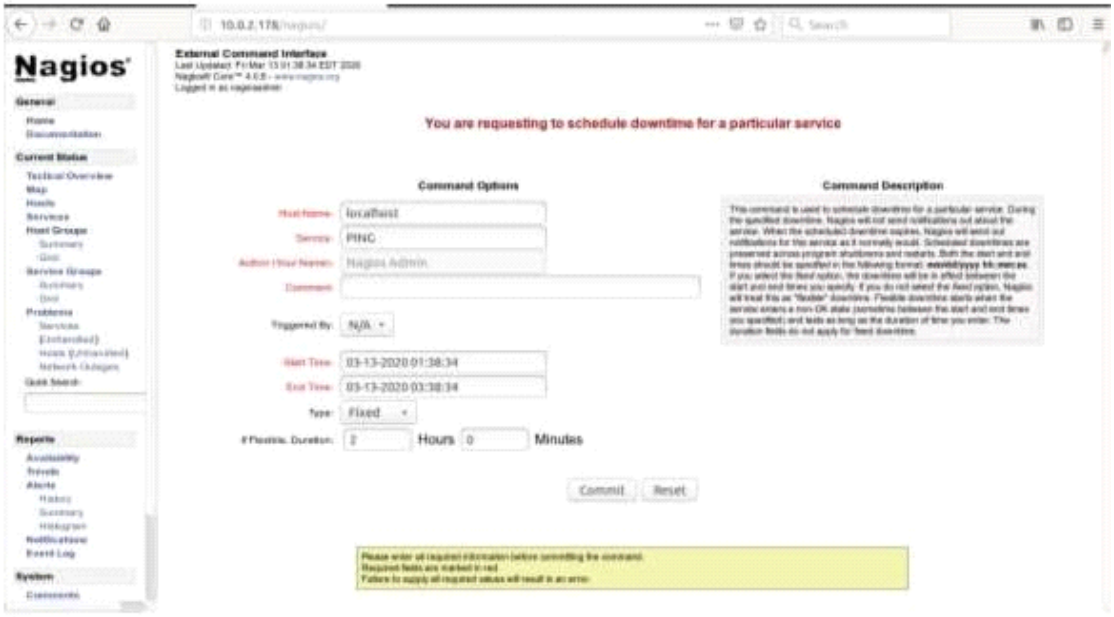
Click on localhost

you can click on a particular entry to view more details about it. For example, here is the “Swap

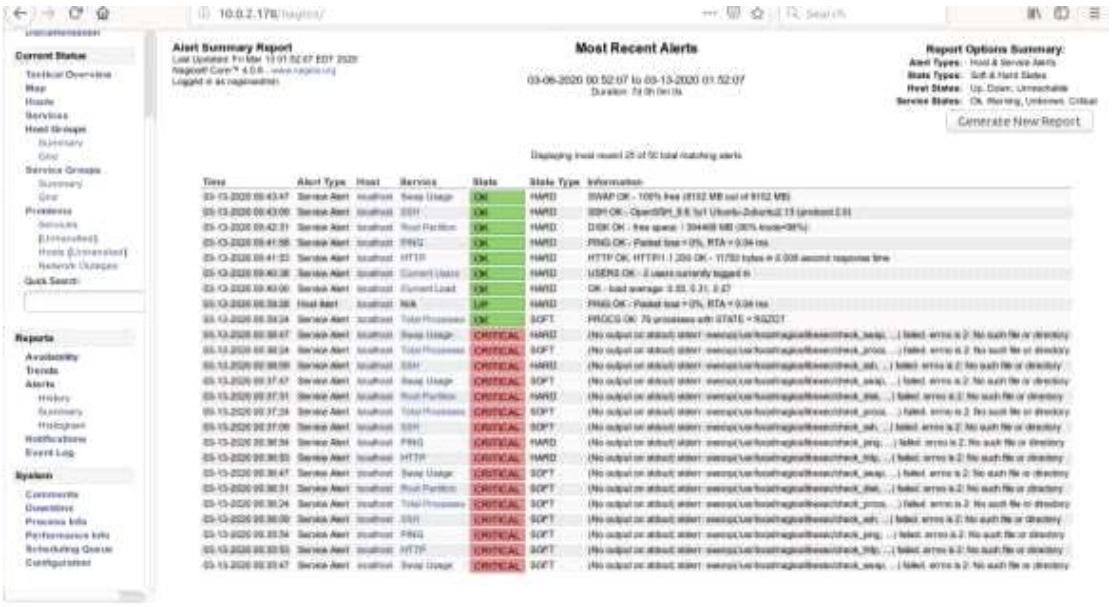
Usage” information page for the current node. You can also issue a number of service commands

from the right, such as disabling the check.

Reschedule downtime of ping service:



Summary report of localhost



CONCLUSION: In this assignment we studied about NagiOs and how to study its different features.

Sanskruti Adap

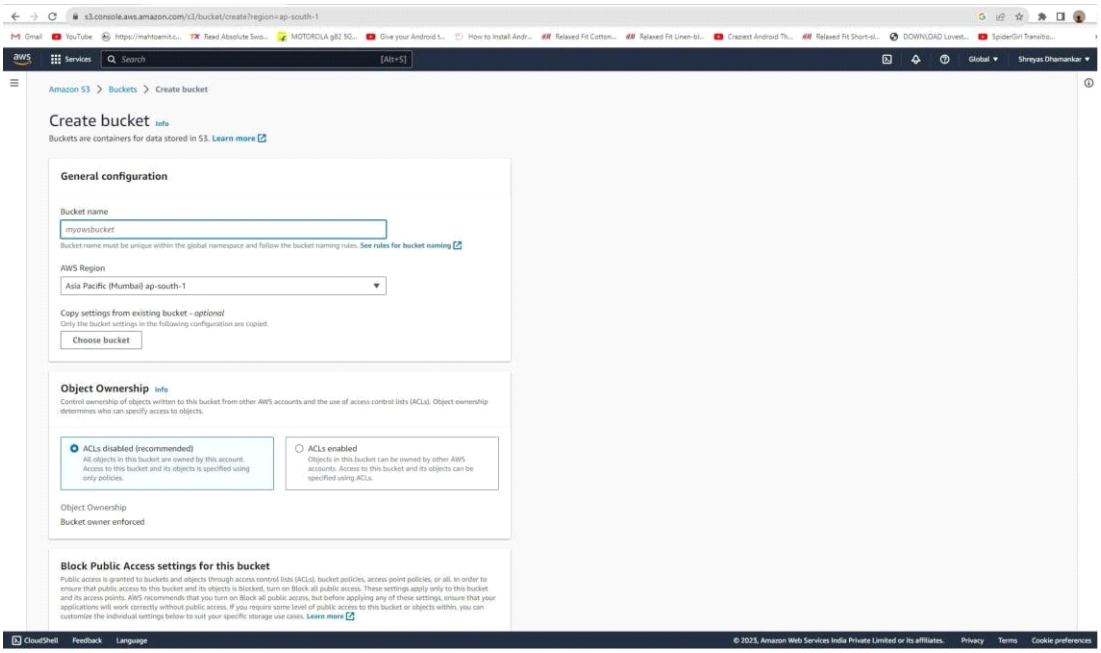
T11-1

Assignment No. 9

Aim: To learn how to use Lamda in order to find the ContentType of Object uploaded in S3 Bucket.

Theory:

Create bucket:

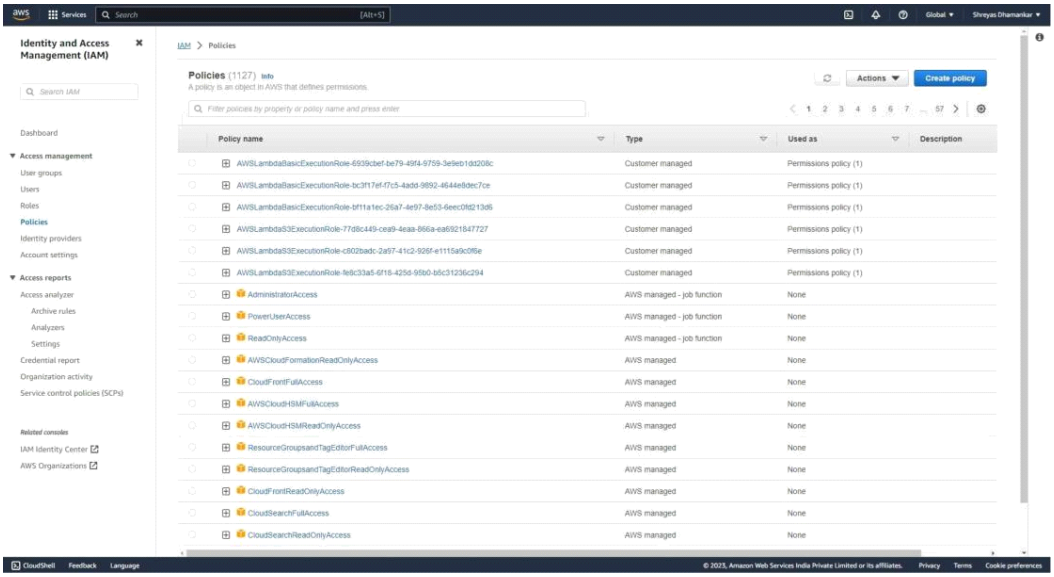


create a new policy from iam dashboard;

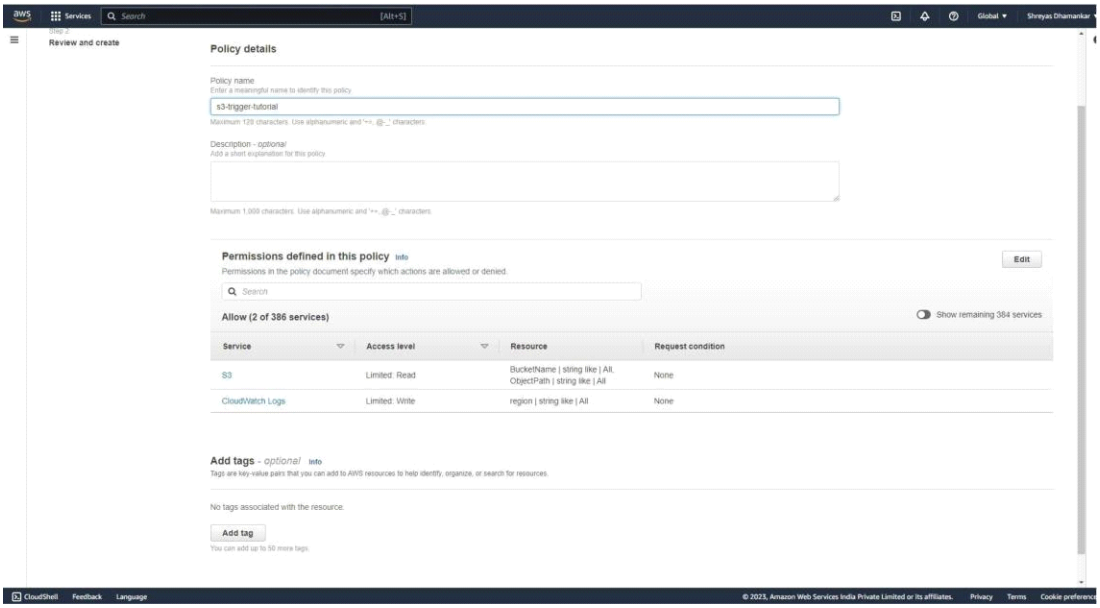
while creating policy select json tab and paste the following code:

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

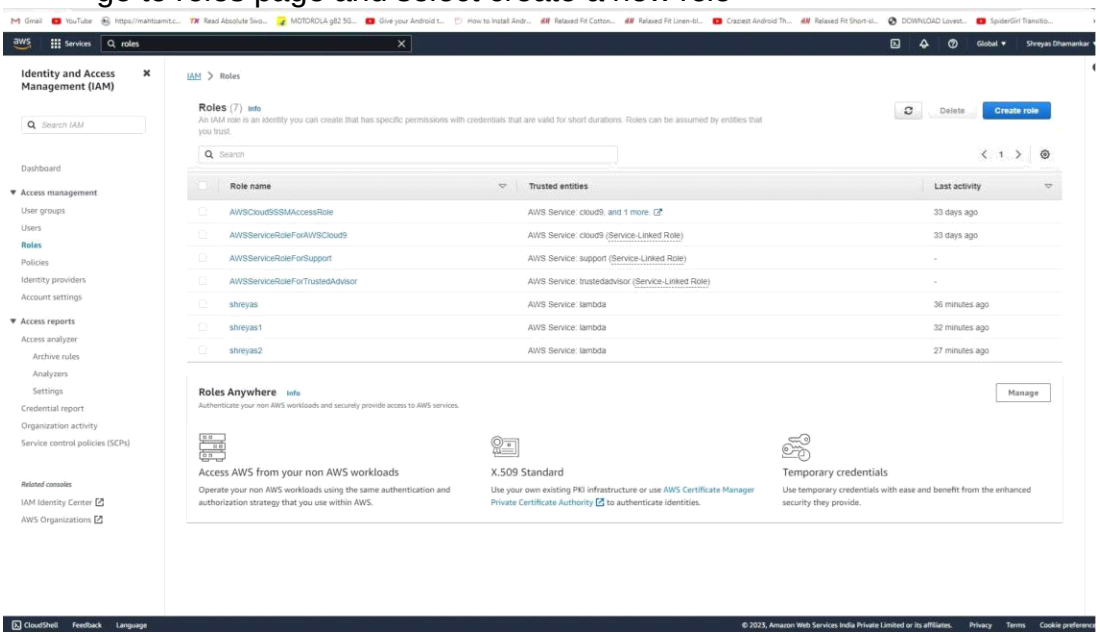
```
            "Effect": "Allow",
            "Action": [
            "logs:PutLogEvents",
                "logs:CreateLogGroup",
                "logs:CreateLogStream"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Effect":
            "Allow",
      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



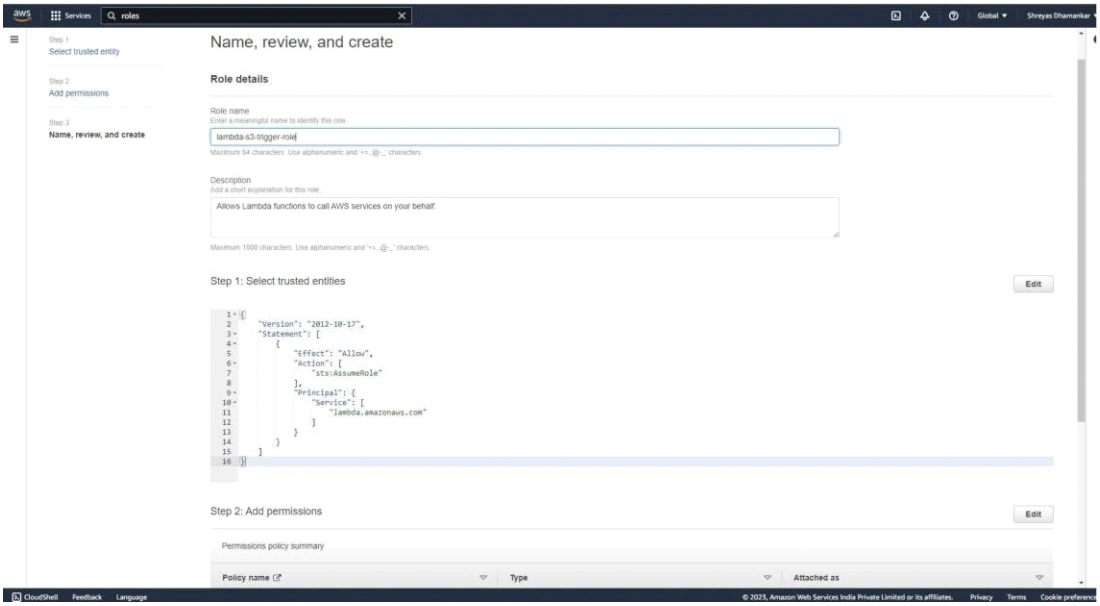
create policy and name it:



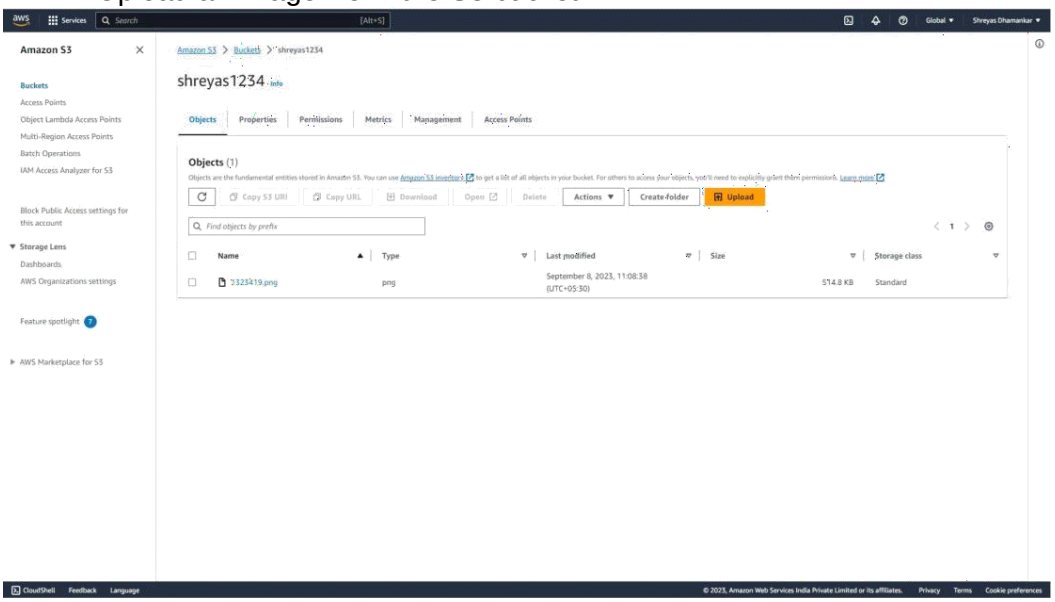
go to roles page and select create a new role



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 jso

import

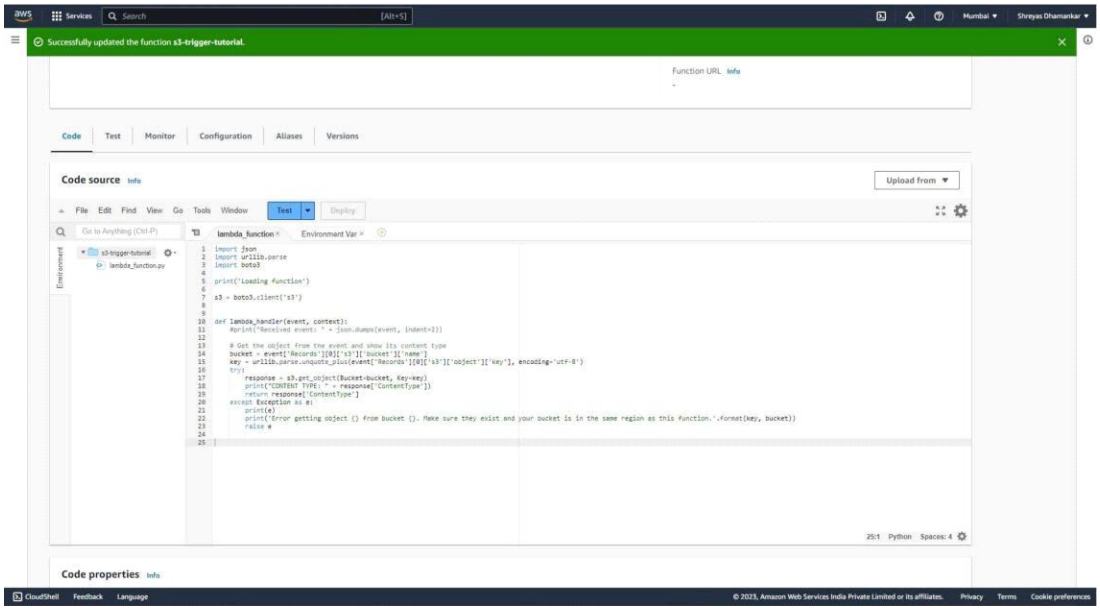
urllib.parse

import boto3

print('Loadin

```
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

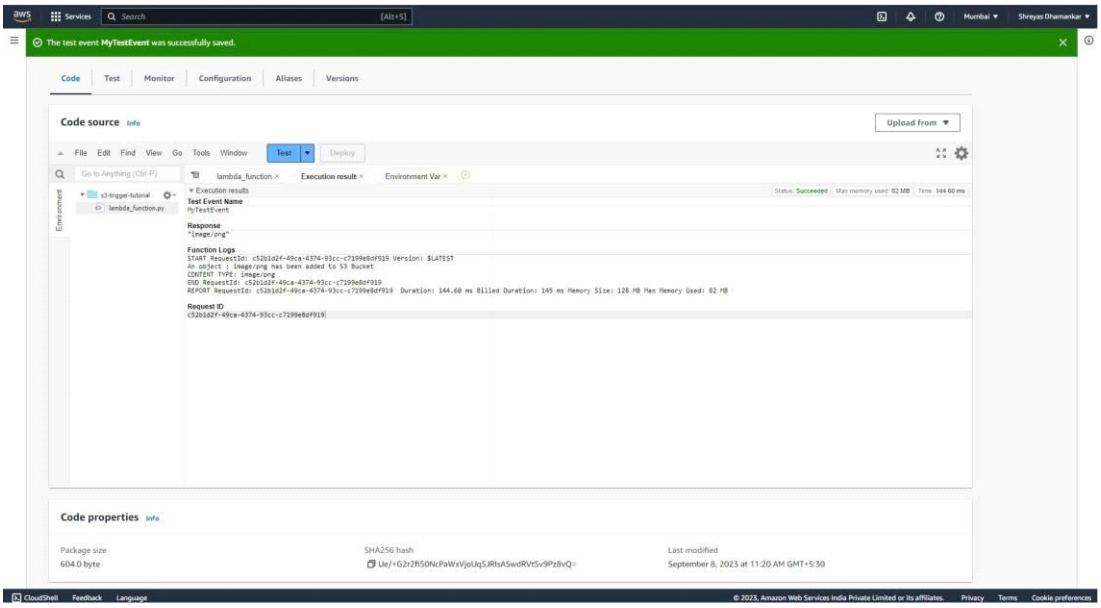
Amazon S3 bucket. Replace test%2FKey with the name of the test object you

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".



Conclusion:

In this assignment we learnt how to use Lambda function to log what type of object has been uploaded to S3 Bucket.

Sanskruti Adap

T11-1

Assignment No. 10

Aim: To complete study of Kuberenetes.

Theory:

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

Kubernetes is a complex container orchestration platform, and it relies on several services and components that run on nodes to manage and control containerized applications. These services work together to provide the functionality required for deploying, scaling, and maintaining applications. Here are some of the key Kubernetes services running on nodes, along with their roles:

Kubelet:

Role: The Kubelet is an essential component that runs on each node in the cluster. It is responsible for ensuring that containers are running in a Pod. It takes Pod specifications (from the API server) and makes sure the containers described in the Pod are running and healthy on the node.

Container Runtime (e.g., Docker, containerd, or CRI-O):

Role: The container runtime is responsible for pulling container images, running containers, and managing their lifecycle. It's the software that actually runs containerson the node.

Kube Proxy:

Role: Kube Proxy is responsible for network connectivity in the cluster. It maintains network rules on each node to route traffic to the appropriate pods based on services and labels. It helps provide load balancing and service discovery.

CNI Plugins (Container Network Interface):

Role: CNI plugins are responsible for setting up and managing the network connectivity for containers in the pod. These plugins configure the network interfaces, IP addresses, and routes for containers so they can communicate with each other and with external networks.Node Status and Heartbeat (NodeStatus and NodeHeartbeat):

Role: These services are responsible for sending the current status and health of the node to the control plane. They provide information about the node's available resources, conditions, and any issues it might be facing.

Docker Daemon (if using Docker):

Role: If you're using Docker as the container runtime, the Docker Daemon manages container images, storage, and the execution of containers.

Containerd (if using Containerd):

Role: Containerd is an alternative container runtime that performs similar functions toDocker. It is used when you opt for a more lightweight runtime compared to Docker.

Kubelet Registrar and Garbage Collector:

Role: These components help with registering nodes with the control plane and ensuring that nodes and their resources are cleaned up and properly managed.

Node OS (Operating System):

Role: The underlying node OS is responsible for providing the necessary environment for running containers. It includes the kernel, system libraries, and utilities required for containers to function.

Systemd or Init System:

Role: The init system (e.g., systemd) on the node is responsible for managing system processes and ensuring that the Kubernetes services are started at boot time and restarted if they fail.

- What is Pod Distribution Budget?

A Pod Distribution Budget is a concept in Kubernetes that allows you to specify constraints and requirements on how pods should be distributed across nodes in a cluster. It is used to control the distribution of pods to ensure that they are evenly spread across nodes or follow specific rules for placement. This is particularly useful for enhancing high availability and fault tolerance, improving resource utilization, and ensuring compliance with policies or regulations.

Pod Distribution Budgets are part of the Pod Topology Spread Constraints feature,which was introduced in Kubernetes to provide more control over how pods are scheduled on nodes.

Here are the key components of a Pod Distribution Budget:

Topology Key: A topology key is a label or field key that is used to determine the distribution of pods across nodes. For example, it could be the availability zone or region of a node.

Max Skew: Max Skew is a numerical value that represents the maximum allowed imbalance in the distribution of pods. For example, if the Max Skew is set to 1, it means that the difference in the number of pods across nodes should not exceed 1.

Topology Spread Constraints: These are rules that specify how pods should be distributed based on the topology key. For instance, you can define constraints to ensure that pods are distributed evenly across nodes in different availability zones.

Pod Distribution Budgets are created by defining a CustomResourceDefinition (CRD) object in Kubernetes. You can then reference this object when creating or updating Deployments, StatefulSets, or other types of workload controllers. The Pod Distribution Budget constraints are checked during the scheduling process, ensuring that pods are placed on nodes in compliance with the defined rules.

By using Pod Distribution Budgets, you can improve the resilience and stability of your applications by avoiding overloading specific nodes and ensuring that your workloads are evenly distributed across your cluster, which is essential for achieving a balanced and efficient Kubernetes cluster.

- What is the role of loadbalance in Kubernetes?

Load balancing in Kubernetes plays a crucial role in ensuring the availability, scalability, and reliability of applications and services running within the cluster. Here'san
overview of the role of load balancing in Kubernetes:

Service Accessibility: Kubernetes abstracts the underlying infrastructure, which means that pods and services can be distributed across different nodes in the cluster. Load

balancing is used to ensure that these services remain accessible to clients regardless of the specific node or pod where they are running.

Service Discovery: Kubernetes Services allow you to define a stable DNS name or IP address for a set of pods. Load balancers route traffic to the appropriate pods associated with a service based on selectors, labels, and other criteria, making it easy for clients to discover and connect to services without needing to know the specific details of pod locations.

Scaling: Kubernetes allows for automatic scaling of pods based on CPU or other resource usage metrics. Load balancers distribute traffic across these dynamically changing instances, ensuring that traffic is evenly distributed even as pods are added or removed in response to demand.

High Availability: Load balancers provide high availability by distributing traffic across multiple replicas of an application. If one pod or node fails, the load balancer can route traffic to healthy instances, minimizing downtime.

Security: Load balancers can serve as a point of entry for traffic into the cluster, providing a layer of security by controlling which traffic is allowed and which is not. They can also be used to terminate SSL/TLS encryption, ofloading the decryption process from the application.

Traffic Splitting: Load balancers support advanced traffic management features like A/B testing and canary deployments. They allow you to route a percentage of traffic to one version of an application and a different percentage to another version for testing or
gradual rollouts.

Global Load Balancing: For multi-cloud or multi-region deployments, Kubernetes can utilize global load balancers to route traffic to the nearest or most available data center or cloud region. This improves latency and redundancy.

Request Routing and Path-Based Routing: Load balancers can route traffic based on different criteria, such as the path in the URL. This enables microservices architectures where different paths are handled by different services.

Kubernetes itself does not provide a load balancer but integrates with various load balancing solutions, such as external cloud load balancers, NodePort services, ClusterIP services, and Ingress controllers. The choice of load balancer depends on the specific requirements and the underlying infrastructure used in your Kubernetes cluster.

Conclusion:

In this assignment we learnt what exactly are kubernetes.

Sanskruti Adap
T11-1

Written Assignment No. 1

Q1) What are the best security measures that you can take while using Kubernetes?

Ans) A product follows industry-specific regulations by adhering to the laws, standards, and guidelines set by the relevant regulatory authorities. Here are the general steps a company might take:

- Research: Identify the specific regulations that apply to your industry and product. This may involve local, state, national, or international laws.

- Compliance Assessment: Evaluate your product to determine if it meets the regulatory requirements. This may involve testing, documentation, and risk assessments.

- Design and Development: Integrate compliance considerations into the product design and development process. This may include using materials and manufacturing processes that meet regulatory standards.

- Documentation: Create detailed records and documentation to prove compliance. This may include test results, safety assessments, and labeling requirements.

- Testing and Certification: If necessary, send your product to accredited testing laboratories for certification. This is often required for products in highly regulated industries like healthcare or aerospace.

- Quality Control: Implement quality control measures to ensure that products consistently meet regulatory standards during manufacturing.

- Labeling and Documentation: Ensure that the product is labeled appropriately with required information, such as safety warnings, ingredients, and certifications.

- Registration and Reporting: Register the product with relevant authorities and regularly report data if required. This is common in industries like pharmaceuticals.

- Stay Informed: Keep up-to-date with changes in regulations, as they can evolve over time. Non-compliance can lead to fines, recalls, or legal issues.

- Consult Experts: Sometimes, it's beneficial to seek legal or regulatory affairs experts to ensure full compliance with complex regulations.

Remember, the specific steps and requirements can vary significantly depending on the industry and location. It's crucial to work closely with experts and regulatory bodies to navigate the complex regulatory landscape effectively.

Q.2 What are three 3 security techniques that can be used to protect data?

Ans)

1 Encryption:

2 Data in Transit*: Use encryption protocols like TLS/SSL to protect data as it moves between different components of the DevOps pipeline, such as from development to testing or production.

3 Data at Rest: Encrypt data stored in databases, configuration files, and other repositories to safeguard it from unauthorized access. Use tools like database encryption or file-level encryption.

1 Access Control and Identity Management:

2 Implement robust access controls to limit who can access and modify data throughout the DevOps process. This involves using tools like role-based access control (RBAC) and ensuring that only authorized personnel have access to critical data.

3 Integrate identity and access management (IAM) solutions to manage user identities, permissions, and authentication, ensuring that only authorized individuals can access the DevOps tools and data.

Vulnerability Scanning and Automated Testing: Integrate security into the DevOps pipeline by employing automated vulnerability scanning and testing tools. These tools can identify security issues in code, configurations, and dependencies.

Implement Continuous Integration and Continuous Deployment (CI/CD) security checks to automatically scan and assess code for vulnerabilities at various stages of the development pipeline. This helps catch security issues early in the development process.

By incorporating these security techniques into the DevOps workflow, you can better protect sensitive data and ensure that security is an integral part of the software development and delivery process.

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

Ans) Exposing a service using Ingress in Kubernetes involves several steps. Ingress is a Kubernetes resource that manages external access to services within a cluster. Here's a high-level overview of the process:

1. Set Up Kubernetes Ingress Controller: You first need to have an Ingress controller running in your Kubernetes cluster. Common controllers include Nginx Ingress, Traefik, and HAProxy Ingress. You can deploy one of these controllers based on your requirements.

For example, you can deploy the Nginx Ingress controller using a command like: shell

kubectl apply -f

https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.46.0/deploy

/static/provider/cloud/deploy.yaml

2. Define Ingress Resource: Create an Ingress resource that defines how you want to expose your service. This resource specifies the rules for routing external traffic to internal services.

Here's an example Ingress resource definition:

yaml

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

metadata:

name: my-ingress spec:

rules:

- host: example.com http:

paths:

- path: /app pathType: Prefix backend:

service:

name: my-service port:

number: 80

In this example, incoming traffic to example.com/app will be forwarded to the my-service service on port 80.

3. Deploy and Expose Your Service: Ensure that your service (e.g., my-service in the example above) is deployed and running in your cluster. You should expose your service as a ClusterIP or NodePort service, depending on your requirements.

4. DNS Configuration: Ensure that the DNS records for the host specified in the Ingress resource (e.g., example.com) point to the IP address of your Kubernetes cluster or the Load Balancer, if you are using one.

5. Test and Access: After the Ingress resource is created and DNS is configured, you should be able to access your service externally by visiting the specified host and path (e.g., http://example.com/app).

6. TLS/SSL Configuration (Optional): If you want to secure your Ingress using TLS/SSL, you can configure a TLS secret and add it to your Ingress resource.

Here's an example TLS configuration in your Ingress resource:

yaml spec:

tls:

- hosts:

- example.com secretName: my-tls-secret

The my-tls-secret should be a Kubernetes Secret containing your TLS certificate and private key.

These steps outline the process of exposing a service using Ingress in Kubernetes. Keep in mind that specifics may vary depending on the Ingress controller and cloud provider you are using, so refer to their documentation for detailed configuration options.

Q.4 Which service protocols does Kubernetes ingress expose?

Ans) Kubernetes Ingress exposes services using HTTP and HTTPS protocols. Ingress is primarily designed for routing external HTTP and HTTPS traffic to services within a Kubernetes cluster based on rules defined in the Ingress resource.

The key features of Ingress include host and path-based routing, SSL termination, and other capabilities related to HTTP and HTTPS traffic. Ingress controllers like Nginx Ingress, Traefik, and others handle the configuration and management of these rules, making it easier to manage and secure external access to services in a Kubernetes cluster.

While Ingress primarily focuses on HTTP and HTTPS, if you need to expose services using other protocols, you might consider different solutions, such as NodePort or LoadBalancer services for protocols like TCP or UDP. These services can provide low-level network access to your pods without the advanced HTTP routing features provided by Ingress.

Sanskruti Adap
T11-1

Written Assignment No. 2

Q1) How to deploy Lambda function on AWS?

Ans) Deploying a Lambda function on AWS involves several steps. Lambda is a serverless computing service that allows you to run code in response to events, and it can be triggered by various AWS services or HTTP requests. Here's a general guide on how to deploy a Lambda function:

Create a Lambda Function:

1. Sign in to the AWS Management Console: Go to the AWS Management Console and log in to your AWS account.

2. Open Lambda Service: Navigate to the Lambda service by searching for "Lambda" in the AWS Console.

3. Create Function: Click on the "Create function" button.

4. Select Author from Scratch: Choose to create a new function from scratch.

5. Basic Information: Provide a name for your function, choose the runtime (e.g., Python, Node.js, Java), and select an execution role with necessary permissions.

6. Function Code: You can either upload a .zip file with your code or edit it in the Lambda editor.

7. Handler: Specify the handler for your function. It's in the format filename.handler, where filename is the name of your script and handler is the name of the function that will be called when your Lambda is triggered.

8. Basic Settings: Configure memory, timeout, and VPC settings for your function.

9. Environment Variables (Optional): Set environment variables if your Lambda function relies on them.

10. Tags (Optional): Add tags to your Lambda function for better organization.

11. Execution role: Ensure the execution role has permissions to access any AWS services or resources your Lambda function needs.

12. Advanced settings (Optional): Configure settings like concurrency, reserved concurrency, and error handling.

13. Triggers: Add triggers to your Lambda function if you want it to be invoked by specific events (e.g., S3 uploads, API Gateway requests).

14. Review: Review your function configuration and click "Create function."

Test Your Function:

After creating the function, you can test it within the Lambda Console by configuring test events. This is useful for verifying that your function works as expected.

Set Up API Gateway (Optional):

If you want to expose your Lambda function as an HTTP API, you can set up Amazon API Gateway to create RESTful APIs or HTTP endpoints.

Deploy Your Function:

To deploy your Lambda function, you don't need to perform a separate deployment step like traditional application deployments. AWS Lambda automatically manages the deployment and scaling of your function.

Monitoring and Logging:

Configure CloudWatch Logs to monitor and log the execution of your Lambda function. You can set up custom CloudWatch Alarms and metrics to track performance and troubleshoot issues.

Versioning and Aliases (Optional)

You can create versions and aliases for your Lambda functions to maintain different versions and promote them to production as needed.

Security and Access Control:

Use AWS Identity and Access Management (IAM) to manage permissions and security settings for your Lambda function. This ensures that your function has the right level of access to AWS resources.

Scaling and Optimization:

AWS Lambda automatically scales your function based on the number of incoming requests. You can fine-tune your function's performance and optimize costs using settings like memory allocation and concurrency.

Monitoring and Error Handling:

Regularly monitor your Lambda function's performance, set up alarms, and implement error handling to deal with exceptions and issues that may arise during execution.

Integration with Other AWS Services:

Integrate your Lambda function with other AWS services to build powerful serverless applications. For example, you can connect it to an S3 bucket, an SNS topic, or a DynamoDB table to perform specific tasks.

Once your Lambda function is deployed and properly configured, it's ready to be triggered by events or HTTP requests as needed. AWS Lambda takes care of the underlying infrastructure, ensuring that your code is executed reliably and efficiently. 3

Q2) What are the deployment options for AWS Lambda?

Ans) AWS Lambda offers several deployment options to manage the deployment of your serverless functions and applications. These options are designed to accommodate different development and deployment scenarios. Here are the main deployment options for AWS Lambda:

Direct Deployment from AWS Management Console:

You can create, configure, and deploy Lambda functions directly from the AWS Management Console using the Lambda service. This is suitable for simple use cases and quick prototypes.

AWS Command Line Interface (CLI):

The AWS CLI allows you to create, package, and deploy Lambda functions from your local development environment. You can use the aws lambda create-function and aws lambda update-function-code commands to deploy your functions.

AWS Serverless Application Model (AWS SAM):

AWS SAM is an open-source framework for building serverless applications. It extends AWS CloudFormation to provide a simplified way to define the Amazon API Gateway APIs, AWS Lambda functions, and Amazon DynamoDB tables needed by your serverless application.

You can package and deploy your serverless application using the sam deploy command.

AWS CloudFormation:

AWS CloudFormation is a service that allows you to define your infrastructure as code. You can use CloudFormation templates to specify your Lambda functions and their associated resources. This is particularly useful for managing more complex deployments and infrastructure.

Serverless Framework:

The Serverless Framework is an open-source framework for building serverless applications. It simplifies the deployment and management of Lambda functions, APIs, and other AWS resources. You define your serverless application in a serverless.yml file and use the Serverless Framework CLI to deploy your application.

Continuous Integration/Continuous Deployment (CI/CD):

Many organizations integrate AWS Lambda deployments into their CI/CD pipelines. This involves automatically building and deploying Lambda functions using CI/CD tools like Jenkins, Travis CI, CircleCI, and AWS CodePipeline.

Amazon CodeDeploy:

Amazon CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, and Lambda functions. You can use CodeDeploy to manage Lambda function deployments and coordinate updates across multiple functions in a release. 4

Third-Party Deployment Tools:

There are third-party tools and services designed to simplify Lambda function deployment and management. Some popular options include the Serverless Framework, Zappa (for Python), and Claudia.js (for Node.js).

The choice of deployment option depends on your specific use case, preferred development workflow, and whether you need to integrate Lambda function deployments into a broader infrastructure-as-code strategy. For simple tasks, direct deployment from the AWS Management Console or the AWS CLI may be sufficient, but for more complex applications, using AWS SAM, CloudFormation, or a serverless framework can streamline the deployment process and provide better management and automation capabilities.

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

Ans) In AWS, there are three primary full deployment modes or strategies used for deploying and managing applications and infrastructure:

Blue-Green Deployment:

Blue-Green deployment is a strategy where you maintain two separate environments, often referred to as the "blue" and "green" environments. At any given time, only one of these environments is in production, while the other is a clone of the production environment, ready for updates and testing.

To deploy updates or changes, you switch traffic from the current "blue" environment to the "green" environment. This minimizes downtime and allows for easy rollback in case of issues.

AWS Elastic Beanstalk, AWS CodeDeploy, and AWS CodePipeline support blue-green deployments.

Canary Deployment:

Canary deployment is a deployment strategy that involves rolling out changes or updates to a small subset of users or instances before making them available to the entire user base. This approach allows you to monitor the effects of the changes on a limited scale before committing to a full rollout.

AWS Elastic Beanstalk, AWS CodeDeploy, and AWS CodePipeline can be configured to perform canary deployments.

Rolling Deployment:

Rolling deployment is a strategy where updates are gradually applied to a subset of instances or resources while others continue to run the previous version. The process is typically automated, and it progresses iteratively until all instances or resources have been updated.

AWS Elastic Beanstalk, AWS CodeDeploy, and AWS CodePipeline support rolling deployments.

These deployment modes provide flexibility and control over how updates and changes are applied to applications and infrastructure in AWS. The choice of deployment mode depends on your specific requirements, such as the level of risk tolerance, the need for rapid updates, and the ability to monitor changes in a controlled manner. Each of these modes can be configured and managed using various AWS services, making it possible to align your deployment strategy 5

with the needs of your application and organization.

Q4) What are the 3 components of AWS Lambda?

Ans) AWS Lambda is a serverless compute service that allows you to run code in response to events without the need to manage servers. Lambda functions consist of three main components:

1 Event Source:

2 An event source is the trigger that invokes your Lambda function. It could be various AWS services, custom applications, or external systems that send events to Lambda.

3 Common event sources include Amazon S3 (e.g., object uploads), Amazon SNS (e.g., notifications), Amazon DynamoDB (e.g., database changes), Amazon API Gateway (e.g., HTTP requests), and more.

4 You can configure multiple event sources for a single Lambda function, allowing it to respond to different types of events.

1 Lambda Function Code:

2 This is the actual code or script that you want to execute in response to the events triggered by the event source.

3 Lambda functions can be written in several programming languages, including Node.js, Python, Java, C#, Ruby, and more.

4 You package your code along with any required dependencies and libraries and upload it to AWS Lambda. You also specify the handler function that Lambda should invoke when the event occurs.

1 Execution Role:

2 The execution role is an AWS Identity and Access Management (IAM) role that defines what AWS resources your Lambda function can access and what it can do.

3 You attach a role to your Lambda function that provides permissions to access other AWS services, such as reading from an S3 bucket, writing to a DynamoDB table, or publishing to an SNS topic.

4 The role's permissions should be defined based on the principle of least privilege, ensuring that your Lambda function only has the permissions necessary to perform its specific tasks.

These three components work together to create a serverless execution environment where your code responds to events from various sources without you needing to manage server provisioning or scaling. AWS Lambda automatically handles the scaling, availability, and execution of your code in response to the events generated by the event source.