

Thadomal Shahani Engineering College
Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss SAIKARTHIK IYER
of IT Department, Semester V with
Roll No. 41 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 2024 - 2025


Teacher In-Charge

Head of the Department

Date 18/10/24

Principal

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	To understand the benefits of cloud infrastructure and setup AWS Cloud 9 IDE, launch Cloud 9 and perform collaboration demonstration	1 - 7	16/7/24	
2	To build your app using CodeBuild and deploy on S3 using Code Pipeline	8 - 15	23/7/24	
3	To understand Kubernetes Cluster architecture, install and spinup a cluster on Linux machine platforms.	16 - 20	30/7/24	
4	Install Kubernetes & execute command to manage cluster & deploy your first Kubernetes application.	21-25	6/8/24	✓ Yes
5	To understand terraform lifecycle & install it on a Linux machine.	26 - 29	13/8/24	
6	To build & destroy AWS using Terraform	30 - 35	3/9/24	
7	To understand Static analysis SAST process & learn to integrate SonarQube SAST to sonarqube.	36 - 40	11/10/24	
8	To create Lambda function which will log "an image has been added" once you add an object to bucket in S3.	41 - 50	24/9/24	
9	To understand continuous monitoring & installation & configuration of Logstash on Linux machine.	51 - 55	8/10/24	
10	To understand Lambda workflow & create your first Lambda function	56 - 60	16/7/24	
11	Written Assignment - 01	61 - 63	3/10/24	
12	Written Assignment - 02	64 - 65	3/10/24	

Name : Saikarthik Iyer

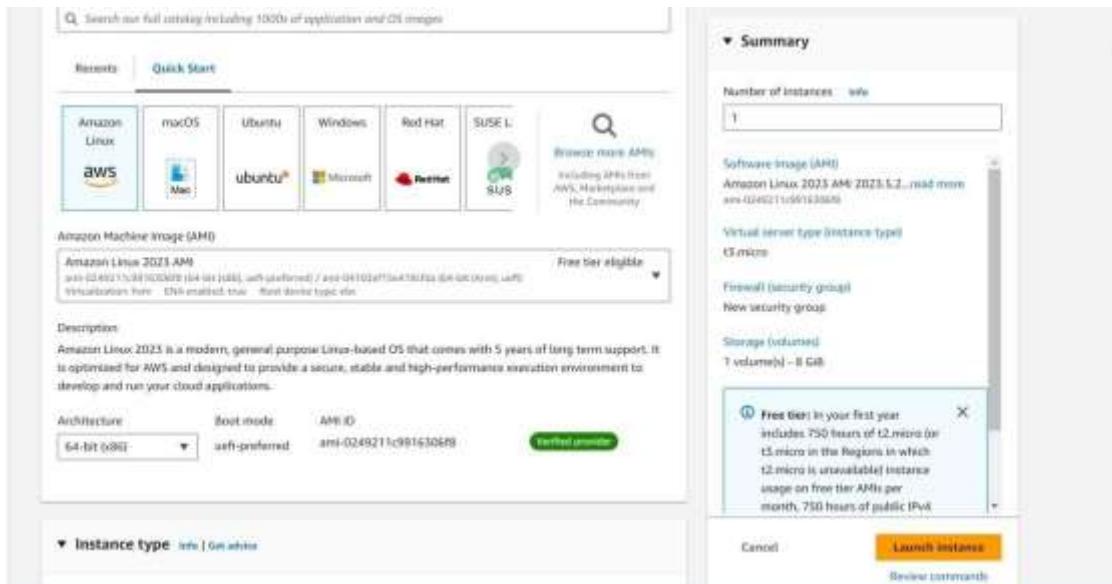
Roll No : T1341

Subject : Advance Devops Lab 1

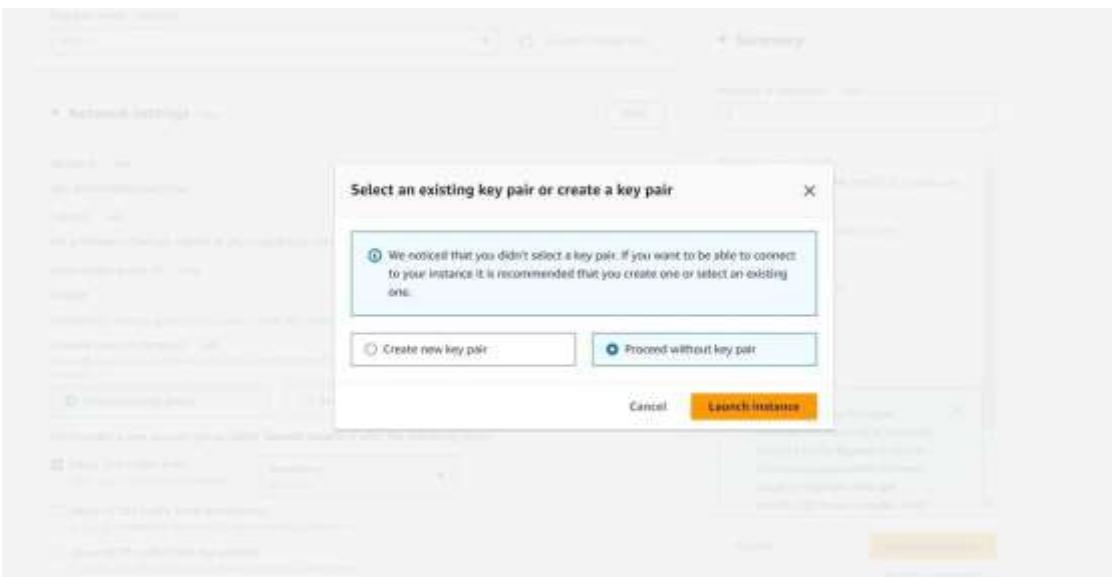
Aim : To implement Ec2 and Cloud9 Service on AWS servers.

Creating an Instance using Ec2 :

Step – 1 : Launch an Instance.



Step – 2 : Without key pair.



Step – 3 : Instance creation successful.

The screenshot shows the AWS EC2 Instances page with a green success banner at the top stating "Successfully initiated launch of instance i-00ca6ebbe00f65747". Below the banner, there's a "Launch log" link. A "Next Steps" section contains several options: "Create billing and free tier usage alerts", "Connect to your instance", "Connect an RDS database", "Create EBS snapshot policy", "Manage detailed monitoring", "Create Load Balancer", "Create AWS budget", and "Manage CloudWatch alarms". A navigation bar at the bottom includes links for "1", "2", "3", "4", "5", "6", and "7".

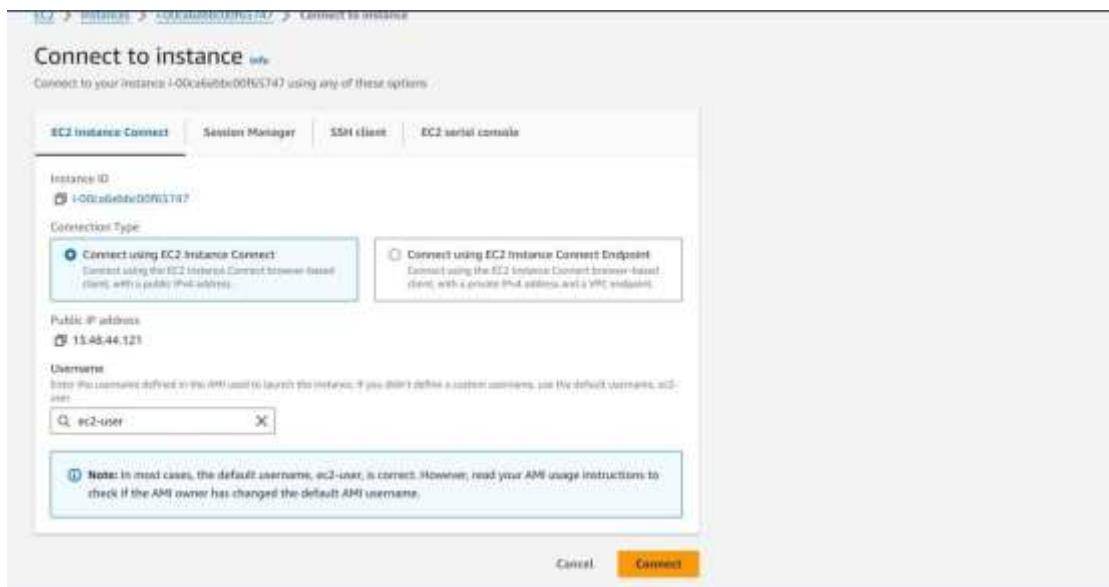
Step – 4 : Check the created instance.

The screenshot shows the AWS EC2 Instances Details page for the instance i-00ca6ebbe00f65747. The instance summary table includes the following details:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
i-00ca6ebbe00f65747	i-00ca6ebbe00f65747	Running	t1.micro	initializing	View alarms +	eu-north-1b

Below the table, the "Details" tab of the instance summary is selected, showing the Public IPv4 address (15.48.44.121), Private IPv4 address (172.31.33.180), Public DNS (ec2-15-48-44-121.eu-north-1.compute.amazonaws.com), and the instance state (Running).

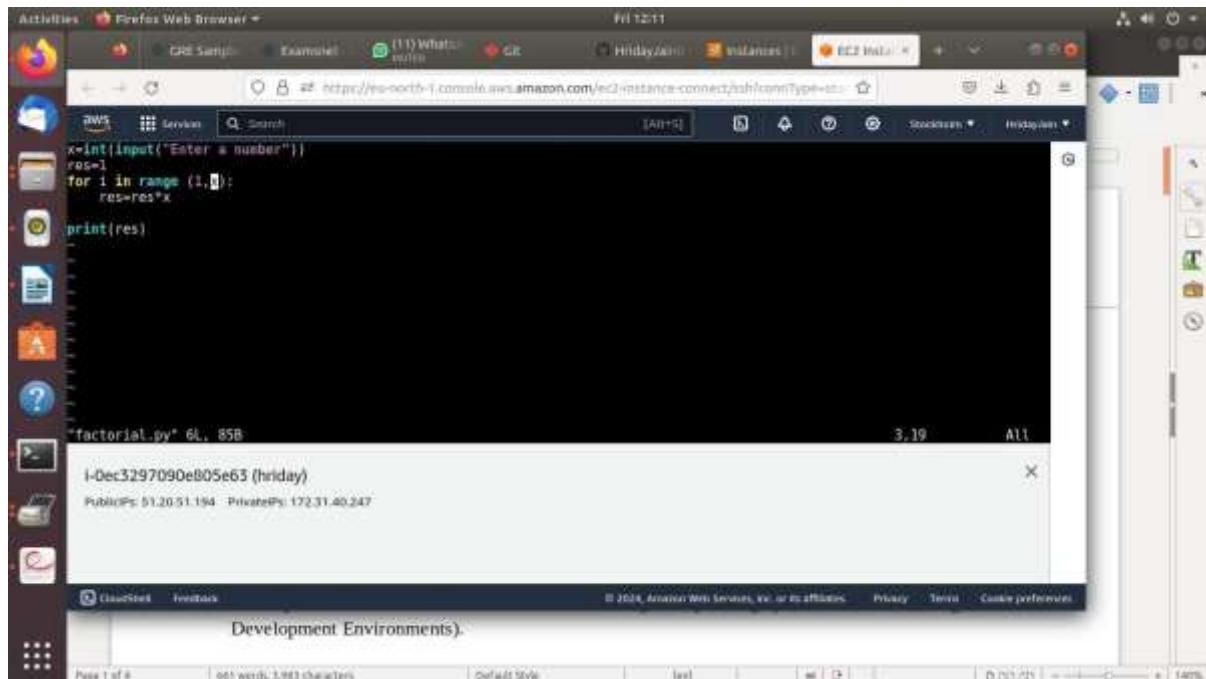
Step – 5 : Connect the created instance.



Step – 6 : Linux Instance created successfully.



Step 7 – Write a program using the Aws command line



A screenshot of a Linux desktop environment. A terminal window is open in a web browser titled "Firefox Web Browser". The URL is <https://eu-north-1.console.aws.amazon.com/ec2/instance-connect/shell?connType=st>. The terminal shows the following Python script:

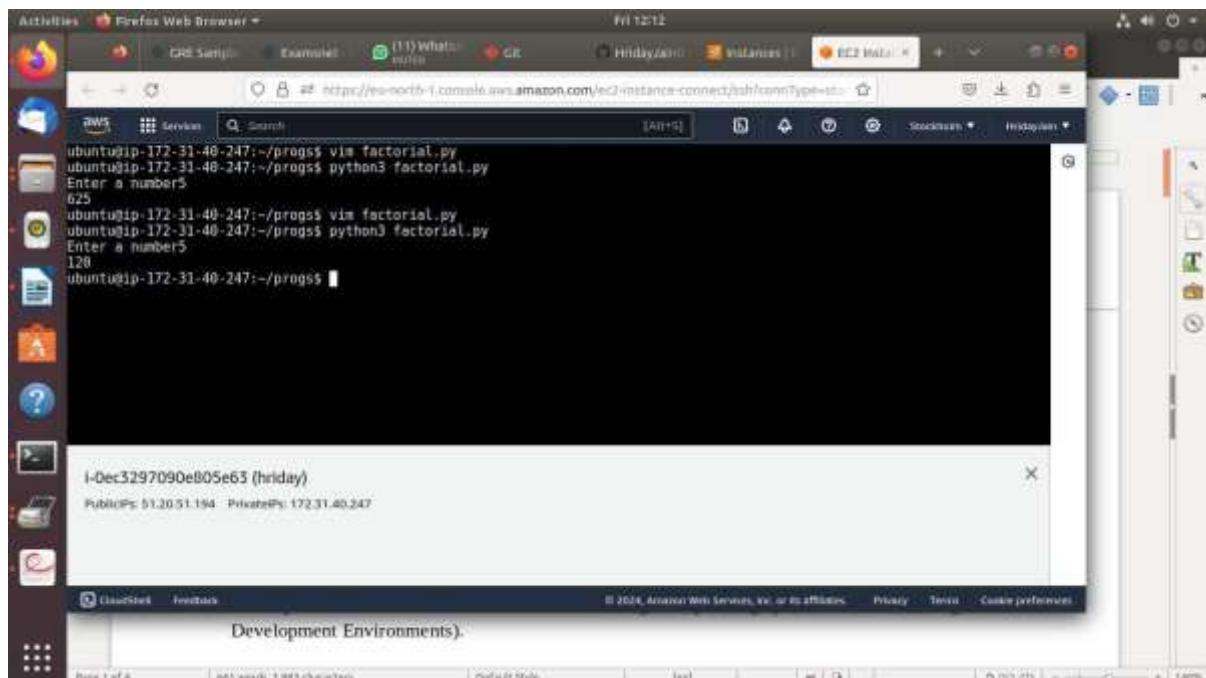
```
x=int(input("Enter a number"))
res=1
for i in range (1,x):
    res=res*x
print(res)
```

The output of the script is displayed below the code:

```
factorial.py 6L 85B
i-Dec3297090e805e63 (hriday)
PublicIPs: 51.20.51.194 PrivateIPs: 172.31.40.247
```

The terminal window has a dark background and white text. The browser interface includes tabs, a search bar, and a status bar at the bottom.

Step 8 – Execute the program using the Aws command line

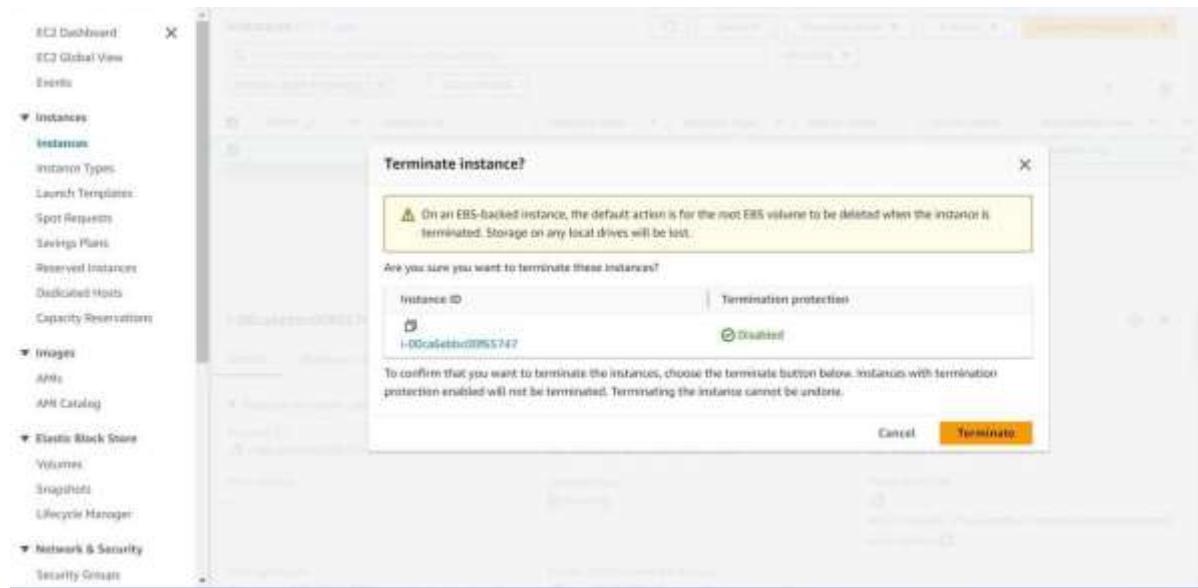


A screenshot of a Linux desktop environment. A terminal window is open in a web browser titled "Firefox Web Browser". The URL is <https://eu-north-1.console.aws.amazon.com/ec2/instance-connect/shell?connType=st>. The terminal shows the execution of the factorial.py script:

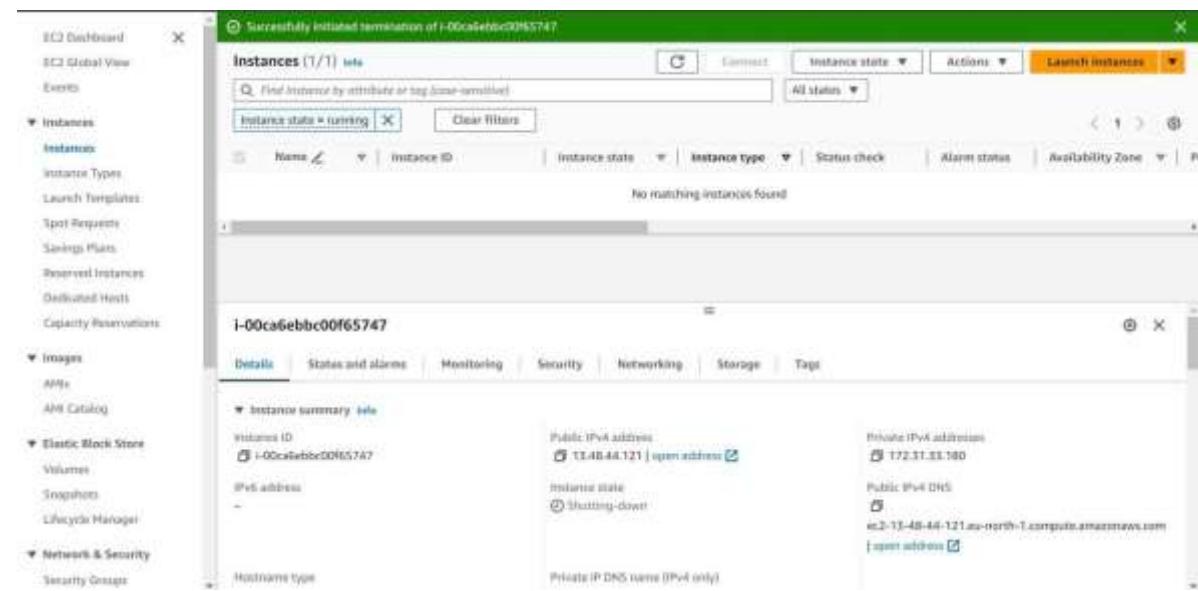
```
ubuntu@ip-172-31-40-247:~/progs$ vim factorial.py
ubuntu@ip-172-31-40-247:~/progs$ python3 factorial.py
Enter a number
625
ubuntu@ip-172-31-40-247:~/progs$ vim factorial.py
ubuntu@ip-172-31-40-247:~/progs$ python3 factorial.py
Enter a number5
120
ubuntu@ip-172-31-40-247:~/progs$
```

The terminal window has a dark background and white text. The browser interface includes tabs, a search bar, and a status bar at the bottom.

Step – 9 : Terminate Instance.



Step – 10 : Instance Terminated Successfully.



Step – 11 : Check for running instance.

The screenshot shows the AWS EC2 Instances page. The left sidebar includes options like EC2 Dashboard, Global View, Events, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (Security Groups). The main pane displays a table titled 'Instances (3)'. The columns are Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. All three instances listed are in the 'Terminated' state, with instance types 't1.micro' and availability zones 'eu-north-1b'. A modal window titled 'Select an Instance' is open at the bottom, showing the same three terminated instances.

Step – 12 : Delete the security group.

The screenshot shows the AWS EC2 Security Groups page. The left sidebar includes options like EC2 Dashboard, Global View, Events, Instances (selected), Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images (AMIs, AMI Catalog), Elastic Block Store (Volumes, Snapshots, Lifecycle Manager), and Network & Security (Security Groups selected). A modal window titled 'Delete security groups' is open in the center. It asks 'Are you sure that you want to delete this security group?' and lists 'sg-023b48f345b4298 - launch-wizard-1'. It has 'Cancel' and 'Delete' buttons.

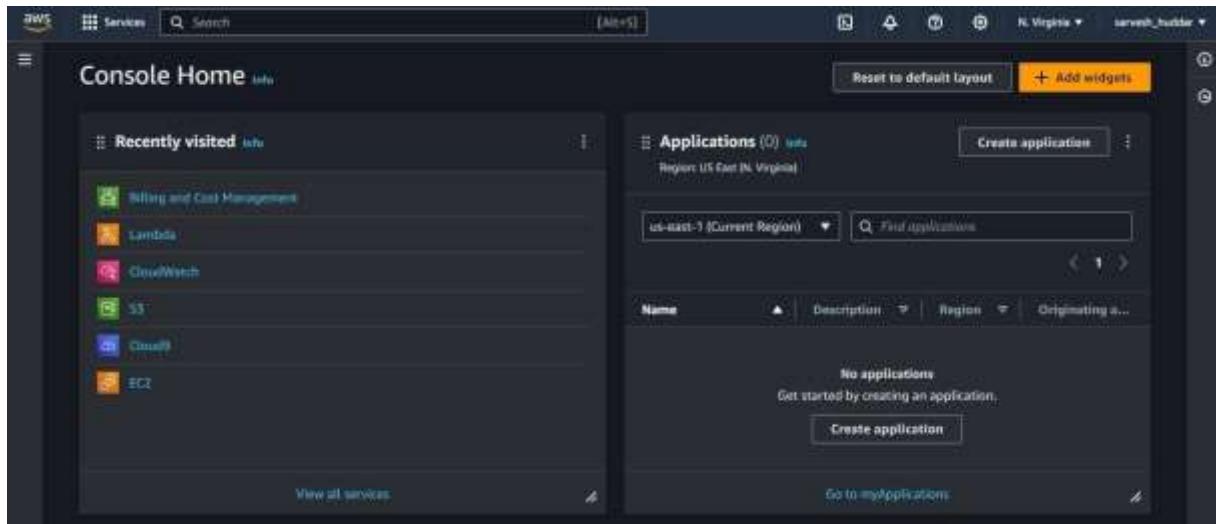
Step – 13 : Security group deleted successfully.

The screenshot shows the AWS EC2 Dashboard with the 'Security Groups' section selected. A success message at the top left states: "Security group sg-022b40f5345b84298 (launch-wizard-1) successfully deleted". The main table displays one security group entry:

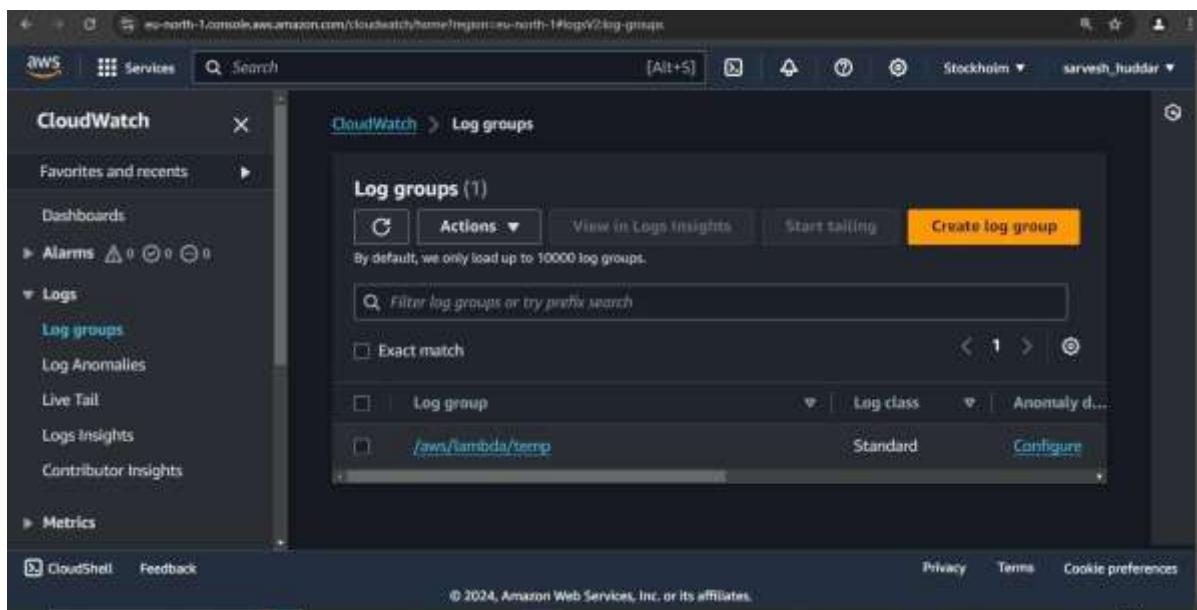
Name	Security group ID	Security group name	VPC ID
-	sg-022b40f5345b84298	default	vpc-07947b0b5c5b4275a

Creating an Instance using Cloud9 :

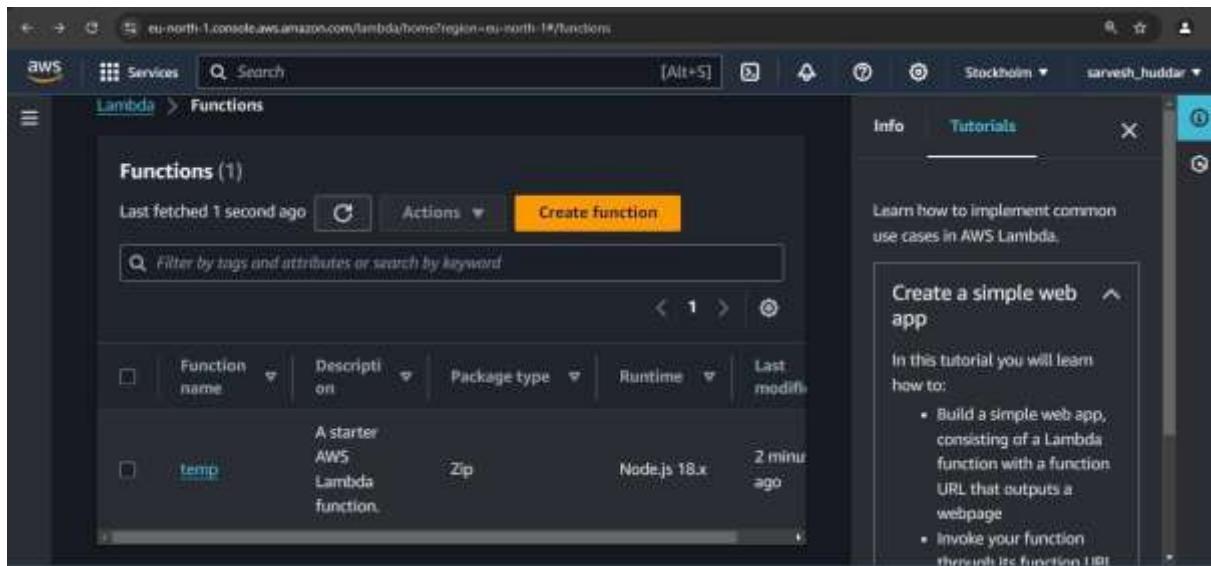
Step 1 - Create and configure the environment in Cloud9



Step 2 – Create a new bucket in S3 and make a log group in Cloudwatch

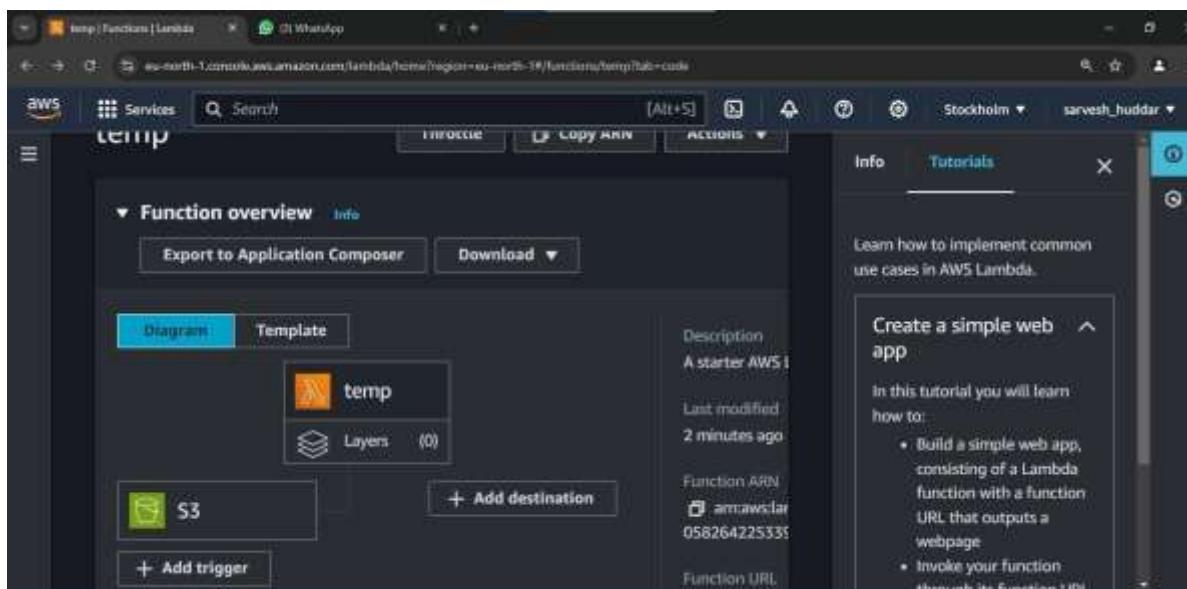


Step 3 – Open the Lambda Console and create a new function



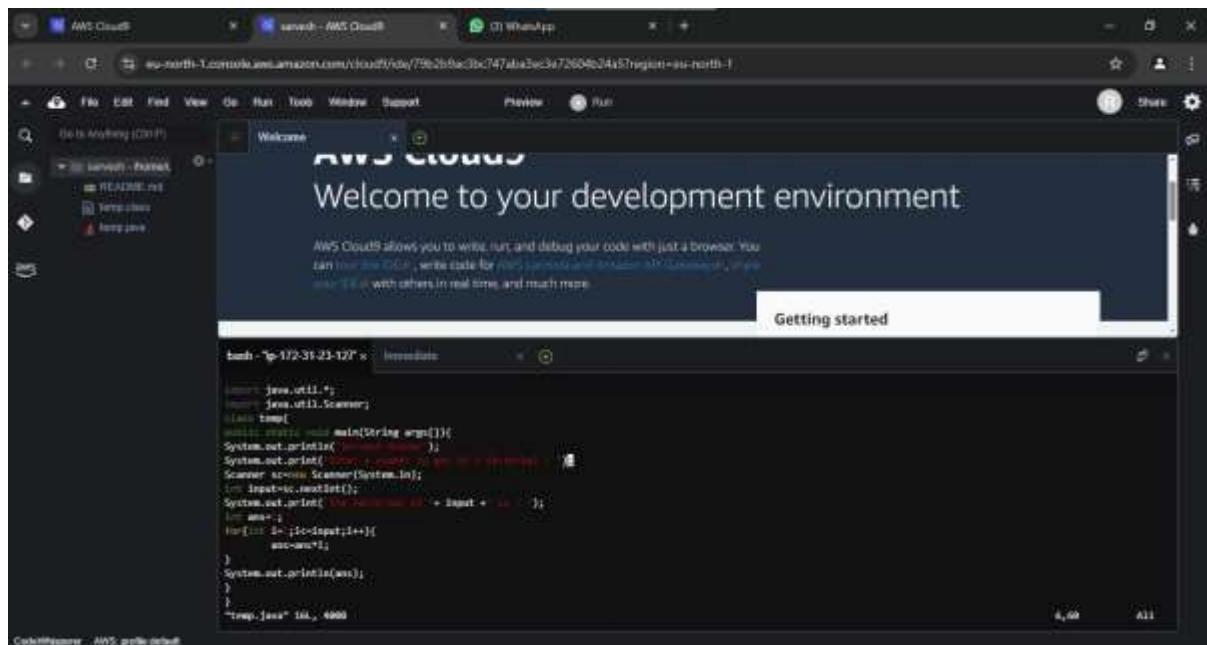
The screenshot shows the AWS Lambda Functions console. At the top, there's a search bar and a 'Create function' button. Below that is a table with columns: Function name, Description, Package type, Runtime, and Last modified. One row is visible for a function named 'temp'. To the right of the table, there's a sidebar titled 'Create a simple web app' with a list of steps: 'Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage' and 'Invoke your function through its function URL'.

Step 4 - Add S3 Trigger and configure it with Cloudwatch logs for Lambda



The screenshot shows the 'Function overview' page for the 'temp' function. It displays the function's ARN and URL. On the left, there are tabs for 'Diagram' and 'Template', and a section for triggers with an 'S3' icon and a '+ Add trigger' button. A sidebar on the right provides a tutorial on creating a simple web app.

Step 5 – Deploy the code from Cloud9

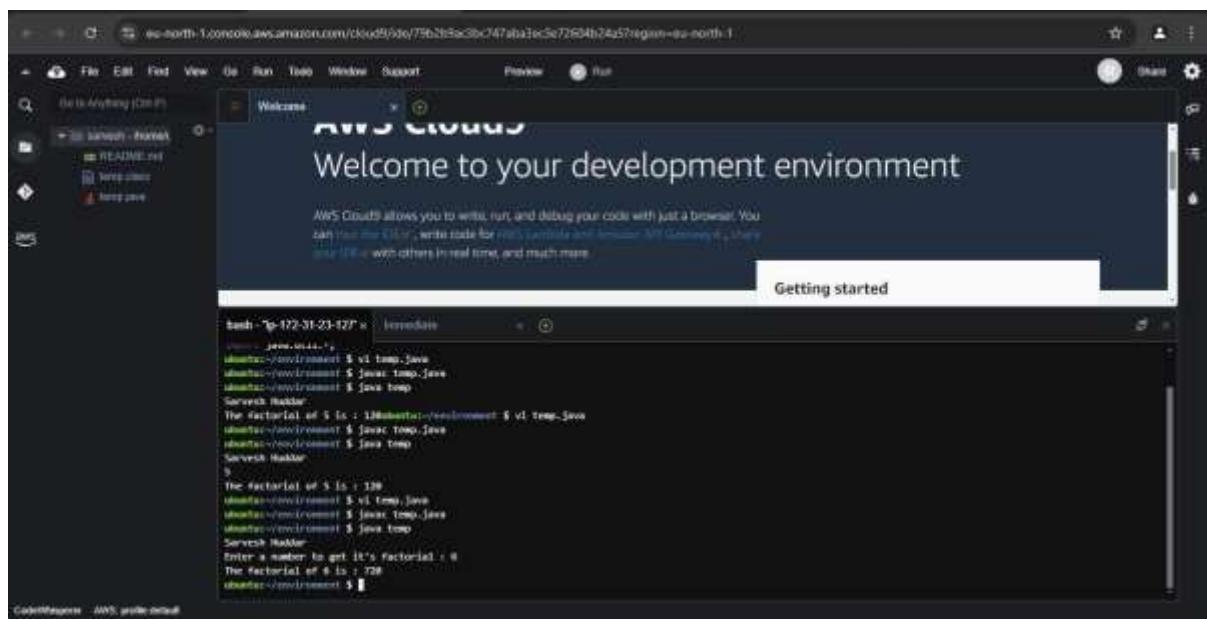


The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a file tree with files like `README.md`, `Temp.java`, and `temp.pwd`. The main area displays a Java program:

```
import java.util.*;
import java.util.Scanner;
class Temp{
    public static void main(String args[]){
        System.out.println("Enter a number");
        System.out.print("Enter a number : ");
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.print("The factorial of " + n + " is : ");
        int ans=1;
        for(int i=1;i<=n;i++){
            ans*=i;
        }
        System.out.println(ans);
    }
}
```

The code is run in an immediate window, and the output shows the factorial of 5 being calculated.

Step 6 – Use the command line to execute the program



The screenshot shows the AWS Cloud9 terminal window. It displays the same Java program as above, but the output shows the results of running it:

```
sh-4.2$ vi temp.java
sh-4.2$ javac temp.java
sh-4.2$ java temp
The factorial of 5 is : 120
sh-4.2$ java temp
The factorial of 5 is : 120
sh-4.2$ java temp
The factorial of 5 is : 120
sh-4.2$ java temp
The factorial of 5 is : 120
sh-4.2$
```

Conclusion : We have learnt the creation of instances in AWS EC2 and Cloud9. This knowledge enhances the ability to efficiently develop, test, and deploy applications using AWS's scalable infrastructure.

Name : Saikarthik Iyer

Roll no : 41

batch T13

Assignment 03

- Key Features of Amazon S3:

1. Scalability: S3 can handle massive amounts of data, scaling automatically as the volume of stored objects increases.

2. Durability and Availability: S3 is designed for 99.999999999% durability and 99.99% availability of objects over a given year. Data is redundantly stored across multiple devices in multiple

facilities.

3. Security: S3 provides several security features including encryption of data at rest and in transit, access control mechanisms (such as IAM policies, bucket policies, and ACLs), and logging and

monitoring capabilities.

4. Cost-effectiveness: S3 provides different storage classes, allowing users to optimize costs by choosing the appropriate class based on access frequency and performance needs.

5. Data Management: S3 offers lifecycle management policies that can automatically transition objects between different

storage

classes or delete them after a specified period.

Aim: To implement lambda function Theory:

Amazon S3 (Simple Storage Service): Amazon S3 is a scalable object storage service provided by Amazon Web Services (AWS). It is designed to store and retrieve any amount of data, at any time, from anywhere on the web. S3 is primarily used for

storing files, such as images, videos, backups, and log files, but it can also store data in other formats like JSON, XML, or plain text.

AWS Lambda: AWS Lambda is a serverless compute service that automatically manages the underlying compute resources. It allows you to run code without

provisioning or managing servers. You only pay for the compute time consumed by your code, which makes it cost-effective for many applications.

- Key Features of AWS Lambda:

1. Serverless Computing: AWS Lambda abstracts the infrastructure,
2. Event-driven Execution: Lambda functions can be triggered by various AWS services, such as S3, DynamoDB, SNS, and CloudWatch. They can also be invoked directly via HTTP

requests

uhsaifnngyAdomluazeor nfuAncPtIioGnaitnewreasyponse to incoming requests. It can o

4. Pthayo-upsearn-udseoPf rciocinncgu:rYreonut eaxrecbuitliloednso.nly for the number of requests served and the compute time required to run your code. This makes it very cost-efficient for infrequent tasks.
5. Support for Multiple Languages: AWS Lambda supports several

Integration of S3 with Lambda: One of the most common use cases of AWS Lambda is to process files stored in Amazon S3. When a new object is uploaded to an S3 bucket, it can trigger a Lambda function that processes the object.

This

integration allows for automated workflows, such as data processing, file validation, image resizing, and real-time notifications.

- How S3 Triggers Lambda:

1. S3 Event Notifications: An S3 bucket can be configured to send event notifications to AWS Lambda when certain actions occur

(e.g., object creation, deletion, or modification). This notification contains metadata about the event, such as the object key,

2. Lnbautimficc,kabeted

noam Faatniodun envc tretinotngg et yErspxe eth. cution: When the event e specified Lambda function. The function

receives the event metadata as input, and it can then perform operations such as reading the object from S3, processing the

and storing the results in another S3 bucket or a different AWS service.

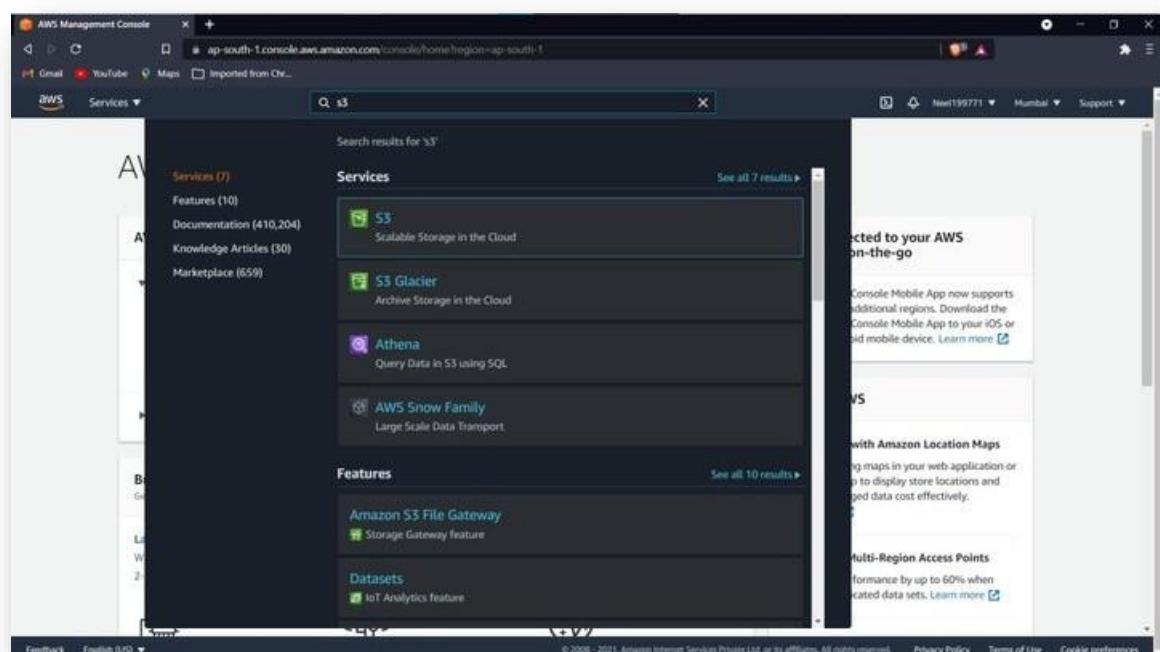
- Example Use Case: Imagine an application where users upload images to an S3 bucket. An S3 event can trigger a Lambda function that automatically resizes the image to different dimensions, compresses it, and saves the processed images back to the S3 bucket. The entire workflow is automated, and no servers are involved in the process.

Advantages of Using S3 with Lambda:

1. Automation: The integration enables seamless automation of workflows without requiring human intervention.
2. Scalability: Both S3 and Lambda scale automatically with the amount of data handled.
3. Cost Efficiency: By using Lambda, you only pay for the compute time used, and by using S3, you only pay for the storage you consume.
4. Ease of Use: The combination of S3 and Lambda simplifies complex tasks like data processing, logging, and notifications.

Steps:

1. Login to AWS account and search S3



2. Create an S3 bucket by giving it a name

The image consists of three vertically stacked screenshots of the AWS S3 'Create bucket' wizard.

Screenshot 1: General configuration

AWS Region: Asia Pacific (Mumbai) ap-south-1
Bucket name: shrutikbuc
Copy settings from existing bucket - optional

Screenshot 2: Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Screenshot 3: Default encryption

Encryption type: Server-side encryption with Amazon S3 managed keys (SSE-S3)
Bucket Key: Disable

Screenshot 4: Summary and Confirmation

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Screenshot 5: Bucket List

Buckets (1)

Name	AWS Region	IAM Access Analyzer	Creation date
shrutikbuc	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	August 14, 2024, 11:41:01 (UTC+05:30)

3. Click on upload and add any .py or .java files after S3 bucket is created in the object section

The screenshot shows the AWS S3 console interface. At the top, a green banner displays "Upload succeeded" and "View details below". Below the banner, a message states: "The information below will no longer be available after you navigate away from this page." The main area is titled "Summary" and shows the destination "s3://shrutikbu" with two succeeded files (2 files, 150.0 B (100.00%)) and zero failed files. There are tabs for "Files and folders" and "Configuration", with "Files and folders" selected. A table lists the uploaded files: "Shruti.java" (106.0 B, Succeeded) and "shruti.py" (24.0 B, Succeeded). The URL in the address bar is <https://ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1>.

4. Search lambda

The screenshot shows the AWS Lambda search results. The search term "lambda" is entered in the search bar. The results are categorized under "Services" and "Features". Under "Services", "Lambda" is listed as "Run Code without Thinking about Servers". Under "Features", "Local processing" (IoT Core feature) and "Target groups" (EC2 feature) are listed. On the right side of the screen, there is a sidebar with options like "Create new function", "Storage class", and "Standard". The URL in the address bar is <https://s3.console.aws.amazon.com/s3/buckets/neel-patel-121182/region-ap-south-1/objects>.

5. Create a function and configure as shown

The screenshot shows the AWS Lambda 'Create function' wizard. The top navigation bar includes 'Services', 'Search', and 'Mumbai'. The main title is 'Create function' with a 'info' link. Below it, a sub-header says 'Choose one of the following options to create your function.' with three radio button options:

- Author from scratch: Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases. This option is selected.
- Container image: Select a container image to deploy for your function.

The 'Basic information' section contains the following fields:

- Blueprint name:** Get S3 object (selected)
- Runtime:** python3.10
- Function name:** shrutikfunction
- Architecture:** x86_64

The 'Trigger configuration' section shows the URL: <https://ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1>. It includes a dropdown for selecting an S3 bucket and a list of triggers:

- All object create events

Below this, there are optional prefix and suffix fields:

- Prefix - optional:** e.g. images/
- Suffix - optional:** e.g. jpg

The 'Recursive invocation' section contains a note about writing objects to the same S3 bucket and a checkbox acknowledgement:

If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

At the bottom right are 'Cancel' and 'Create function' buttons.

6. Select the bucket created & create trigger, click on create function

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

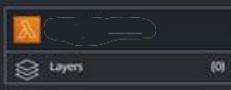
Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.

CloudShell Feedback AWS Services Search [Alt+S] © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Mumbai ShrutiK ⓘ

Lambda > Functions >  Throttle Copy ARN Actions ⓘ

Function overview Info

Diagram Template


Layers (0)

+ Add trigger + Add destination

Description
An Amazon S3 trigger that retrieves metadata for the object that has been updated.

Last modified
7 seconds ago

Function ARN
arn:aws:lambda:ap-south-1:009160046510:function:shrutikfunction

Function URL Info

Code Test Monitor Configuration Aliases Versions

CloudShell Feedback AWS Services Search [Alt+S] © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Mumbai ShrutiK ⓘ

Lambda > Add triggers Add trigger

Trigger configuration Info

S3 aws asynchronous storage

Bucket
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.
 Bucket region: ap-south-1

Event types
Select the events that you want to trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

All object create events

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Mumbai ShrutiK ⓘ

7. Check if the given trigger is created

The screenshot shows the AWS Lambda Functions console. In the top navigation bar, 'Lambda' is selected under 'Services'. Below it, 'Functions' is selected under 'shrutikfunction'. A search bar at the top right contains the text 'bhatfunction'. On the left, there's a sidebar with a tree view showing 'bhatfunction' expanded, with 'S3' listed as a child node. The main content area has a green success message: 'The trigger shrutikbucket was successfully added to function shrutikfunction. The function is now receiving events from the trigger.' To the right, there's a 'Function overview' section with tabs for 'Diagram' (selected) and 'Template'. It shows a diagram with a box labeled 'bhatfunction' connected to a box labeled 'S3'. Buttons for '+ Add destination' and '+ Add trigger' are visible. On the far right, there's a 'Description' panel with details like 'An Amazon S3 trigger that retrieves metadata for the object that has been updated.', 'Last modified 45 seconds ago', and a 'Function ARN' field containing 'arn:aws:lambda:ap-south-1:009160046510:function:shrutikfunction'. A 'Function URL' link is also present.

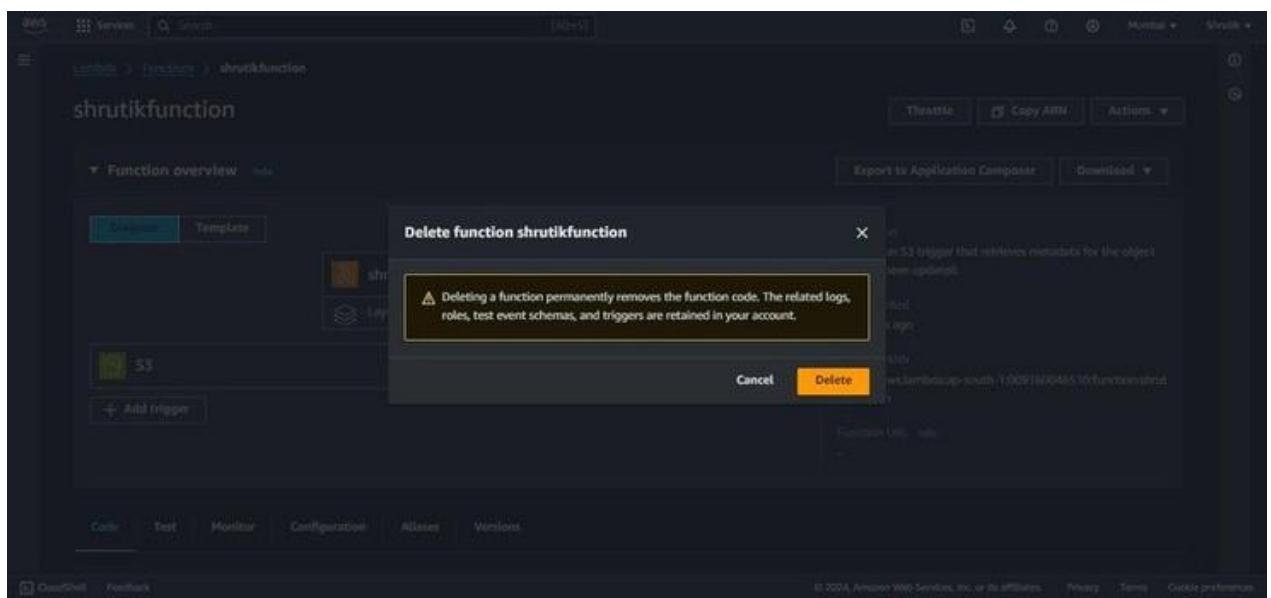
8. Click on test

The screenshot shows the 'Test' tab of the AWS Lambda function configuration. At the top, there's a search bar with 'SS' and a '+ Add trigger' button. Below the tabs 'Code', 'Test' (selected), 'Monitor', 'Configuration', 'Aliases', and 'Versions', the 'Code source' tab is active. It displays the code for 'lambda_function.py':

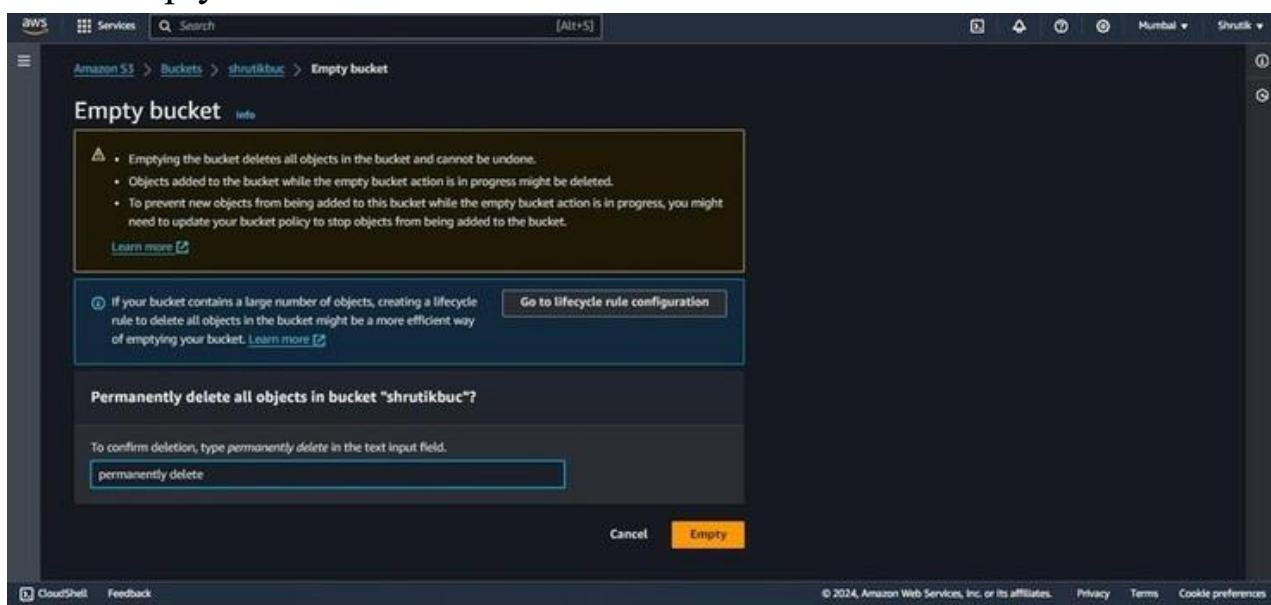
```
1 import json
2 import urllib.parse
3 import boto3
4
5 print('Loading function')
6
7 s3 = boto3.client('s3')
8
9
10 def lambda_handler(event, context):
11     print("Received event: " + json.dumps(event, indent=2))
12
13     # Get the object from the event and show its content type
14     bucket = event['Records'][0]['s3']['bucket']['name']
15     key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
16     try:
```

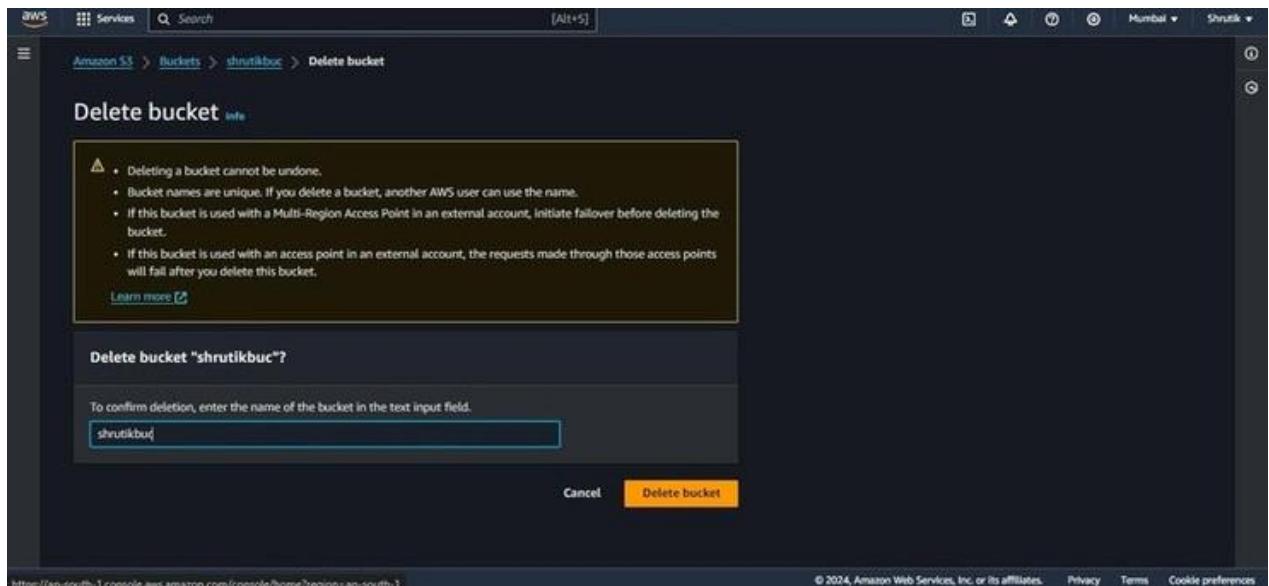
At the bottom, there are buttons for 'Upload from' and file operations like 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', and 'Deploy'. The status bar at the bottom right shows 'E1 Python Spaces:4'.

9. Delete the function



10. Empty and delete the bucket





Conclusion: Hence learned & implemented S3 buckets and lambda function

Name : Saikarthik Iyer

Batch : T13

Roll no : 41

AdvDevops Assignment 3

Aim : To install and understand Kubernetes

Theory and Outcome:

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available. The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation results from counting the eight letters between the "K" and the "s". Google opensourced the Kubernetes project in 2014. Kubernetes combines over 15 years of Google's experience running production workloads at scale with best-of-breed ideas and practices from the community.

Go to kubernetes website for installation.

The screenshot shows the official Kubernetes documentation website. At the top, there is a navigation bar with links to Documentation, Kubernetes Blog, Training, Partners, Community, Case Studies, Versions (dropdown), and English (dropdown). On the left, there is a sidebar with a search bar and a navigation menu. The main content area is titled 'Install Tools' and specifically focuses on 'kubectl'. It includes a brief description of what kubectl is, its availability across multiple platforms, and a list of installation steps for Linux, macOS, and Windows. To the right of the main content, there is a sidebar with options to edit the page, create a child page, create a documentation issue, and print the entire section. Below the sidebar, there is a list of related terms: kubectl, kind, minikube, and kubeadm.

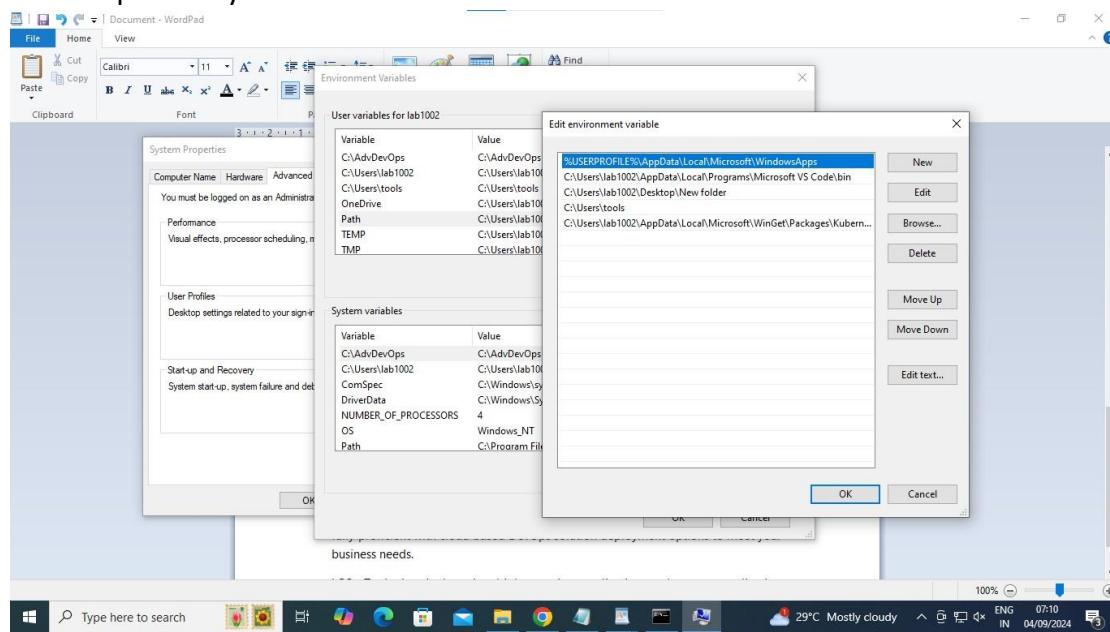
Select your appropriate OS.

v1.31.0	windows	386	kubectl-convert.exe	dl.k8s.io/v1.31.0/bin/vconvert.exe (checksum
v1.31.0	windows	386	kubectl.exe	dl.k8s.io/v1.31.0/bin/v(checksum signature
v1.31.0	windows	amd64	kube-log-runner.exe	dl.k8s.io/v1.31.0/bin/vrunner.exe (checksum
v1.31.0	windows	amd64	kube-proxy.exe	dl.k8s.io/v1.31.0/bin/vproxy.exe (checksum s
v1.31.0	windows	amd64	kubeadm.exe	dl.k8s.io/v1.31.0/bin/win(checksum signature

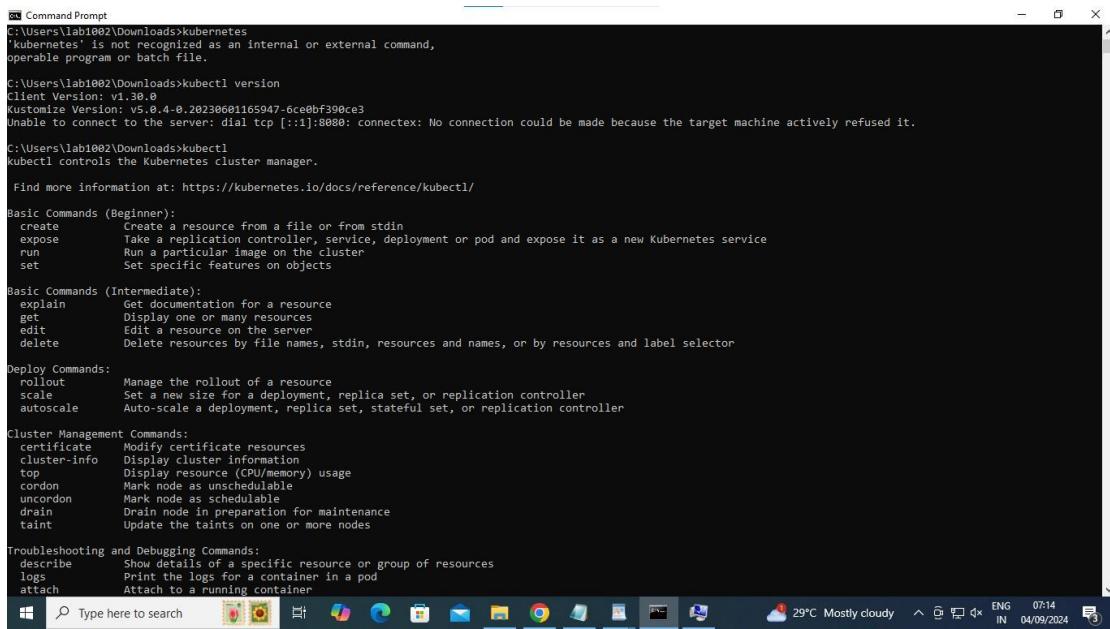
Install the Kubernetes .



Edit the path in systems environment variables



Open cmd and run and kubernetes command to check for successful installation



A screenshot of a Windows Command Prompt window titled "Command Prompt". The window displays the output of the command "kubectl help". The output provides detailed information about various Kubernetes commands, categorized into Basic Commands (Beginner), Basic Commands (Intermediate), Deploy Commands, Cluster Management Commands, and Troubleshooting and Debugging Commands. The window also shows the Windows taskbar at the bottom with icons for File Explorer, Edge, and other applications.

```
C:\Users\lab1002\Downloads>kubernetes
'kubernetes' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\lab1002\Downloads>kubectl version
Client Version: v1.38.8
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Unable to connect to the server: dial tcp [::1]:8080: connectex: No connection could be made because the target machine actively refused it.

C:\Users\lab1002\Downloads>kubectl
kubectl controls the Kubernetes cluster manager.

Find more information at: https://kubernetes.io/docs/reference/kubectl/

Basic Commands (Beginner):
  create      Create a resource from a file or from stdin
  expose     Take a replication controller, service, deployment or pod and expose it as a new Kubernetes service
  run        Run a particular image on the cluster
  set        Set specific features on objects

Basic Commands (Intermediate):
  explain    Get documentation for a resource
  get       Display one or many resources
  edit      Edit a resource on the server
  delete    Delete resources by file names, stdin, resources and names, or by resources and label selector

Deploy Commands:
  rollout   Manage the rollout of a resource
  scale     Set a new size for a deployment, replica set, or replication controller
  autoscale Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:
  certificate Modify certificate resources
  cluster-info Display cluster information
  top        Display resource (CPU/memory) usage
  cordon    Mark node as unschedulable
  uncordon  Mark node as schedulable
  drain     Drain node in preparation for maintenance
  taint    Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
  describe  Show details of a specific resource or group of resources
  logs      Print the logs for a container in a pod
  attach    Attach to a running container
```

Kubernetes is successfully installed.

Conclusion : LO1 - To understand the fundamentals of cloud computing and be fully proficient with cloud based DevOps solution deployment options to meet your business needs. LO2 - To deploy single and multiple container applications and manage application

Saikarthik Iyer

T13

41

AIM: To understand the Kubernetes Cluster Architecture.

THEORY:

What are the various Kubernetes services running on nodes? Describe the role of each service. Kubernetes relies on various services running on cluster nodes to ensure smooth and efficient orchestration of containers. Here are some key services running on Kubernetes nodes and their roles:

1. Kubelet:

- **Role:** Kubelet is an essential component that runs on each node and communicates with the Kubernetes Control Plane.
- **Responsibilities:** It ensures that containers are running in a Pod as expected. Kubelet takes PodSpecs (defined by the higher-level orchestration system) and ensures that the containers described in those PodSpecs are running and healthy.

2. Kube Proxy (kube-proxy):

- Role: Kube Proxy is responsible for network communication within the cluster.
- Responsibilities: It maintains network rules on nodes and enables network communication to and from the Pods. Kube Proxy handles tasks such as load balancing, routing, and network security.

3. Container Runtime (e.g., Docker, container):

- Role: The container runtime is responsible for running and managing containers.
- Responsibilities: It interacts with the underlying Linux kernel to create, run, and manage containers. Common container runtimes used with Kubernetes include Docker and containers.

4. Node OS:

- Role: The host operating system on which Kubernetes nodes run.
- Responsibilities: The node OS provides the foundational environment for containers and system-level resources required by Kubernetes services.

5. CNI (Container Network Interface):

- Role: CNI is a set of specifications and libraries for network plugins in Kubernetes.
- Responsibilities: It allows different networking plugins to work with Kubernetes nodes. CNI plugins enable the

creation and management of network interfaces for Pods, facilitating network connectivity.

6. Cloud Provider Components (if applicable):

- Role: Some Kubernetes clusters run on cloud providers like AWS, GCP, or Azure.
- Responsibilities: Cloud provider components, such as AWS cloud controller manager or Azure cloud controller manager, manage interactions between the cluster and the underlying cloud infrastructure. They enable features like load balancing, storage provisioning, and auto-scaling in a cloud-specific manner.

7. Monitoring and Logging Agents (optional):

- Role: These agents collect and send monitoring and logging data to external systems.

- Responsibilities: Agents like Prometheus, Fluentd, and others gather metrics and logs from containers and Pods, aiding in monitoring, troubleshooting, and performance analysis.

8. Kubelet Garbage Collection (optional):

- Role: The kubelet performs garbage collection to remove old resources.
- Responsibilities: It helps maintain cluster cleanliness by removing unused images, volumes, and Pods to free up resources.

These services work together to ensure the proper functioning of Kubernetes nodes and the containers running on them. Each service plays a specific role in managing containers, networking, and node-level operations within the cluster.

What is the Pod Disruption Budget (PDB)?

A **Pod Disruption Budget (PDB)** is a policy in Kubernetes that helps define the maximum allowable disruption to a set of Pods during voluntary disruptions, such as scaling down or draining nodes. It provides a way to control the impact of actions like node maintenance or scaling events on the availability of workloads running in a Kubernetes cluster.

Key characteristics and concepts of Pod Disruption Budgets include:

1. Selector: A PDB specifies a set of Pods using label selectors. These are the Pods to which the budget applies.
2. Min Available: The PDB defines a `minAvailable` parameter, indicating the minimum number of Pods that must remain available. This helps ensure a level of availability for the application during disruptions.
3. Max Unavailable: Another parameter, `maxUnavailable`, sets the maximum number of Pods that can be unavailable at any given time.
4. Scope: PDBs can be defined at various levels, such as at the Namespace level or for specific Deployments, StatefulSets, or ReplicaSets.
5. Enforcement: Kubernetes controllers, such as the Deployment controller, ensure that the specified `minAvailable` and `maxUnavailable` values are adhered to during actions like scaling down or draining nodes. The controller won't proceed with actions that would violate the PDB.

Pod Disruption Budgets are particularly useful when you need to ensure that critical applications maintain a certain level of availability during planned maintenance activities or when scaling down resources. They help prevent the excessive disruption of Pods and maintain the reliability of the applications running in a Kubernetes cluster.

What is the role of Load Balance in Kubernetes?

In Kubernetes, a **Load Balancer** plays a crucial role in distributing network traffic across a set of Pods or Services, ensuring high availability, reliability, and efficient resource utilisation. The primary role of a Load Balancer in Kubernetes is as follows:

1. Traffic Distribution: Load Balancers evenly distribute incoming network traffic across a group of Pods or Services. This ensures that no single Pod or Service is overwhelmed with traffic, leading to better performance and fault tolerance.
2. High Availability: Load Balancers enhance the availability of applications by ensuring that traffic is directed to healthy Pods or Services. If one Pod becomes unhealthy or is taken down for maintenance, the Load Balancer automatically redirects traffic to healthy instances, minimising downtime.
3. Scalability: As the demand for a service or application grows, Load Balancers can distribute traffic to multiple Pods or replicas of a Service. This horizontal scaling ensures that the system can handle increased load effectively.
4. Service Discovery: Load Balancers often act as a single entry point for external clients to access a Kubernetes Service. This abstraction simplifies the process of discovering and accessing Services running in the cluster.
5. Network Path Optimization: Load Balancers can optimise the network path for incoming traffic. They can route

traffic to the nearest or least congested Pods or data centres, improving network performance.

6. Security: Load Balancers can be configured to provide security features, such as SSL termination and DDoS protection, to safeguard the applications running in the cluster.
7. Public-Facing Services: For applications or Services that need to be publicly accessible, Load Balancers provide a secure and controlled way to expose the Services to the internet.

Kubernetes supports different types of Load Balancers, including cloud provider-specific Load Balancers (e.g., AWS ELB, GCP Load Balancers) and Kubernetes-native Load Balancers like kube-proxy and kube-router. The choice of Load Balancer type depends on your cluster setup, requirements, and infrastructure.

In summary, Load Balancers in Kubernetes are instrumental in ensuring that network traffic is distributed efficiently and reliably across Pods and Services, enhancing application availability and performance while simplifying the process of accessing Services from external clients.

CONCLUSION:

Hence, I have understood the Kubernetes Cluster Architecture.

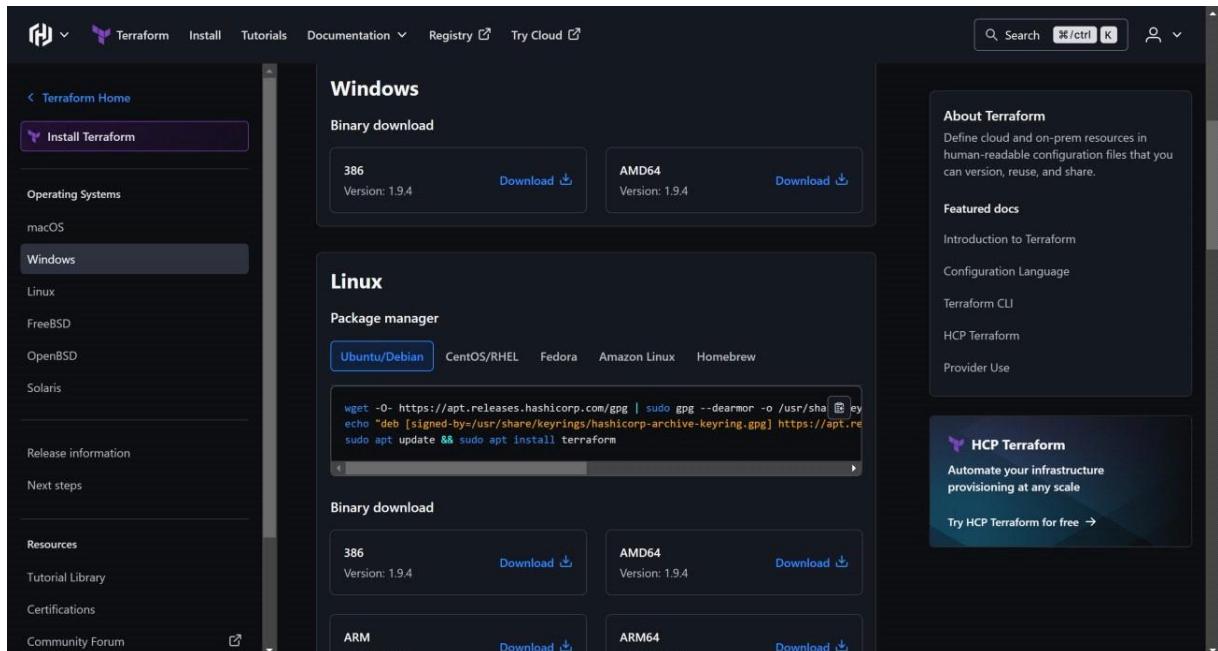
Name : Saikarthik Iyer

Roll No.: T1341

Subject : Advanced DevOps Lab

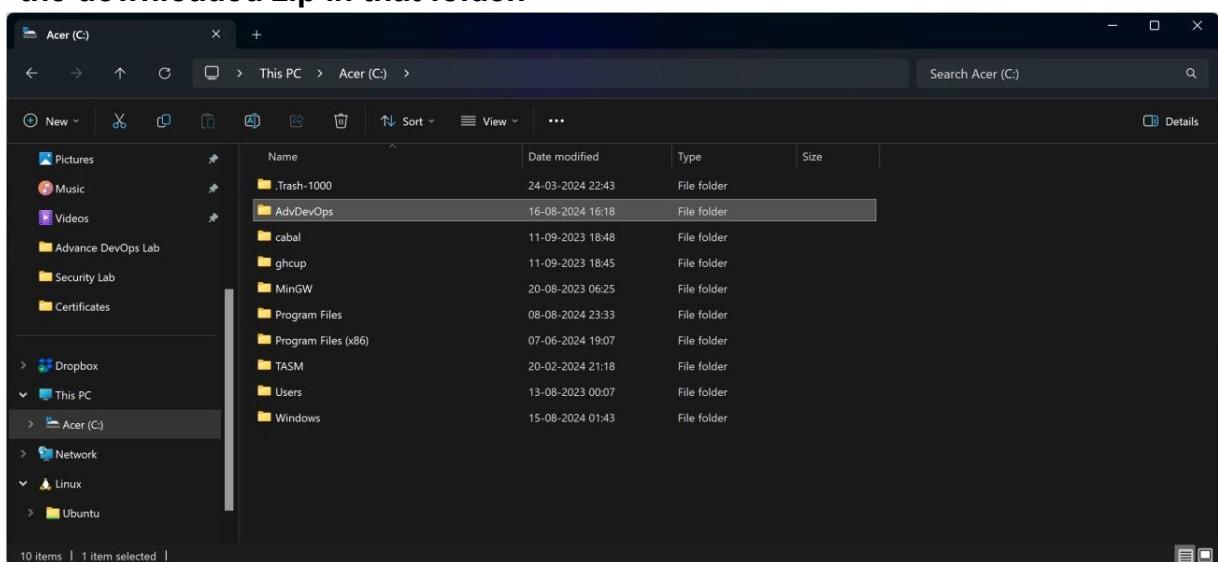
Assignment – 5

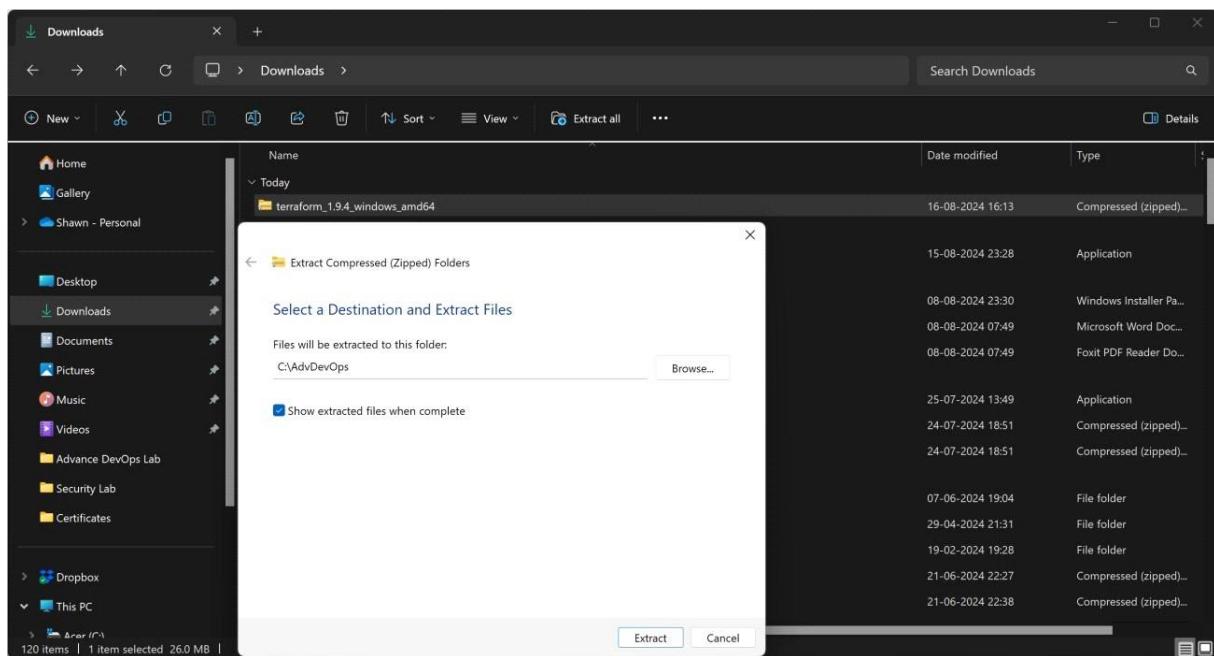
Aim : To understand the terraform lifecycle, core concepts/terminologies and install it.



Step – 1 : Browse for installing Terraform on Windows.

Step – 2 : Create a folder name AdvDevOps in C-Drive and extract the files of the downloaded zip in that folder.





Step – 3 : Log in to AWS and search for IAM.

The screenshot shows the AWS IAM Dashboard. The left sidebar has a 'Search IAM' bar and navigation links for IAM, Dashboard, Access management, Policies, and Access reports. The main area has a 'Security recommendations' section with two items: 'Add MFA for root user' (with a link to 'Add MFA') and 'Root user has no active access keys' (with a link to 'Using access keys'). Below is an 'IAM resources' summary with counts for User groups, Users, Roles, Policies, and Identity providers. A 'What's new' section is at the bottom. To the right, there's an 'AWS Account' summary with Account ID (058264240223), Account Alias (Create), and a sign-in URL (https://058264240223.signin.aws.amazon.com/console). There's also a 'Quick Links' section for 'My security credentials'.

Step – 4 : Create a user and user group.

IAM > Users

Users (1) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

User name	Path	Group	Last activity	MFA	Password age
demoUser	/	0	-	-	-

Create user

Search IAM

Identity and Access Management (IAM)

- Dashboard
- Access management**
 - User groups
 - Users**
 - Roles
 - Policies
 - Identity providers
 - Account settings
- Access reports**
 - Access Analyzer
 - External access
 - Unused access

IAM > Policies

Policies (1226) Info
A policy is an object in AWS that defines permissions.

Policy name	Type	Used as	Description
AccessAnalyzerSer...	AWS managed	None	-
AdministratorAccess	AWS managed - job fu...	Permissions policy (1)	-
AdministratorAcce...	AWS managed	None	-
AdministratorAcce...	AWS managed	None	-
AlexaForBusinessD...	AWS managed	None	-
AlexaForBusinessF...	AWS managed	None	-
AlexaForBusinessG...	AWS managed	None	-

Create policy

Search IAM

Identity and Access Management (IAM)

- Dashboard
- Access management**
 - User groups
 - Users**
 - Roles
 - Policies**
 - Identity providers
 - Account settings
- Access reports**
 - Access Analyzer
 - External access
 - Unused access

Step – 5 : User created successfully.

The screenshot shows the AWS IAM 'Create access key' wizard at step 5. The top navigation bar includes 'CloudShell', 'Feedback', 'Services', 'Search', and 'Global'. The left sidebar shows the path: IAM > Users > Shawn > Create access key. Step 1 is 'Access key best practices & alternatives'. Step 2 is 'optional' with 'Set description tag'. Step 3 is 'Retrieve access keys'. The main content area is titled 'Access key best practices & alternatives' with a 'Info' link. It advises against long-term credentials like access keys. Below is a 'Use case' section with several options:

- Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
- Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
- Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
- Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
- Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.
- Other
Your use case is not listed here.

A callout box says: "It's okay to use an access key for this use case, but follow the best practices:"

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access keys when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Buttons at the bottom right are 'Cancel' and 'Next'.

The screenshot shows the AWS IAM 'Create access key' wizard at step 6. The top navigation bar includes 'CloudShell', 'Feedback', 'Services', 'Search', and 'Global'. The left sidebar shows the path: IAM > Users > Shawn > Create access key. Step 1 is 'Access key best practices & alternatives'. Step 2 is 'optional' with 'Set description tag'. Step 3 is 'Retrieve access keys'. The main content area is titled 'Retrieve access keys' with a 'Info' link. It states: "This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time." Below is a table showing the access key information:

Access key	Secret access key
AKIAQ3EGRYBPVDV2VKUT	***** Show

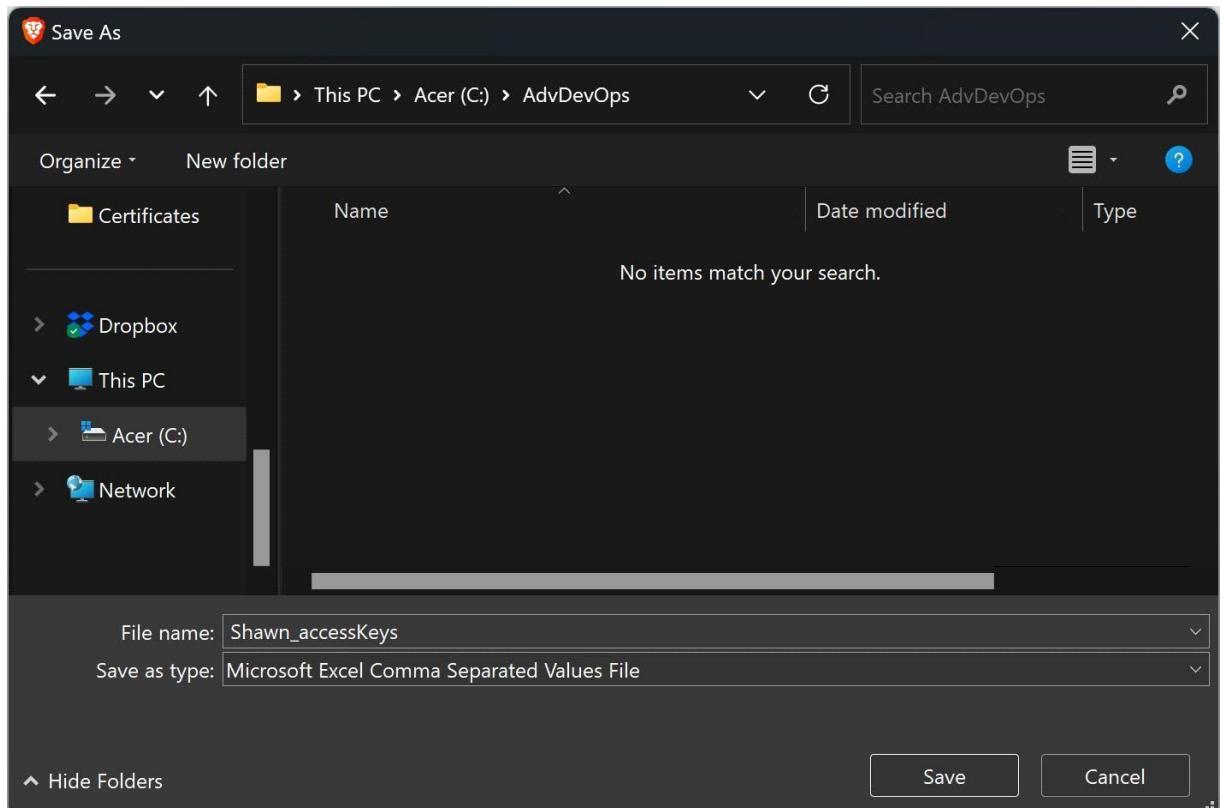
A callout box says: "Access key best practices"

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [best practices for managing AWS access keys](#).

Buttons at the bottom right are 'Download .csv file' and 'Done'.

Step – 6 : Create access key.



Step – 7 : Download CSV File.

Step – 8 : Go to EC2 in AWS.

Service Status

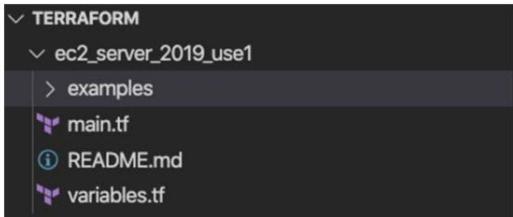
THINK  STACK

Solutions ▾ Consulting ▾ About ▾ Resources ▾ Contact Us

This module will do a few things:

1. Create an EC2 Instance
2. Automatically look up the latest Windows Server 2019 AMI for the EC2 instance.
3. Create and attach an additional drive.
4. Create a Cloudwatch Alarm Metric to monitor CPU.

The folder structure looks like this:



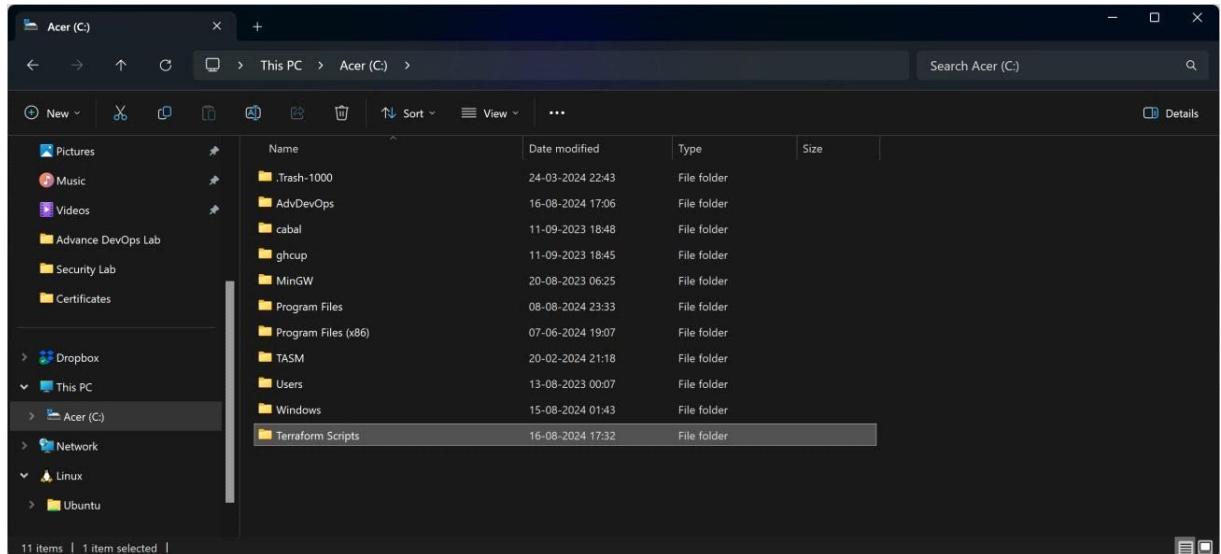
```

    TERRAFORM
      ec2_server_2019_use1
        examples
        main.tf
        README.md
        variables.tf
  
```

First things first... I created the main.tf file which contains all of my configuration except for the variables and outputs.

Step – 9 : Browse for Creating an EC2 instance using AWS.

Step – 10 : Create a folder named ‘Terraform Scripts’ in the C-Drive where the ‘AdvDevOps’



folder is located.

Step – 11 : Write the below script in Notepad and save it the recently created folder in the above step.

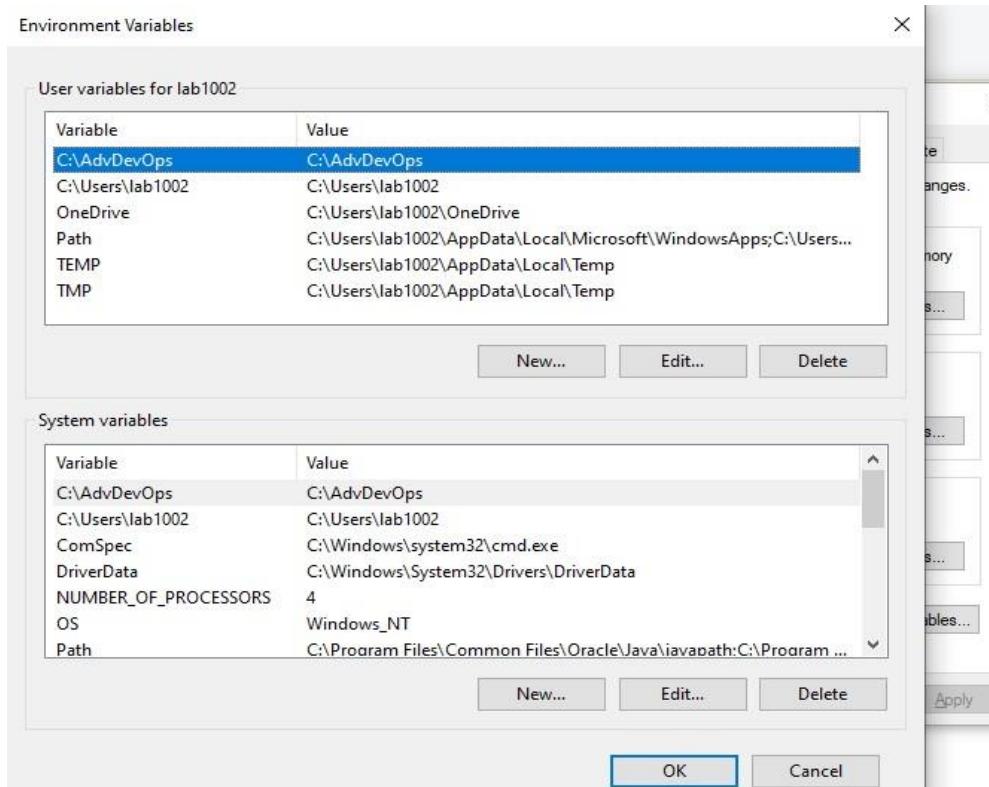
A screenshot of a code editor window titled "test.tf". The window shows a Terraform configuration file with the following content:

```
provider "aws" {  
  access_key="AKIAQ3EGRYBPVDV2VKUT"  
  secret_key="DDWfuju05A3fqUgv8sgBviPECd1wQDWN4yX3cvh"  
  region="eu-north-1"  
}  
  
resource "aws_instance" "Ubuntu" {  
  ami="ami-08200e1ffcab6af77"  
  instance_type="t4g.micro"  
}
```

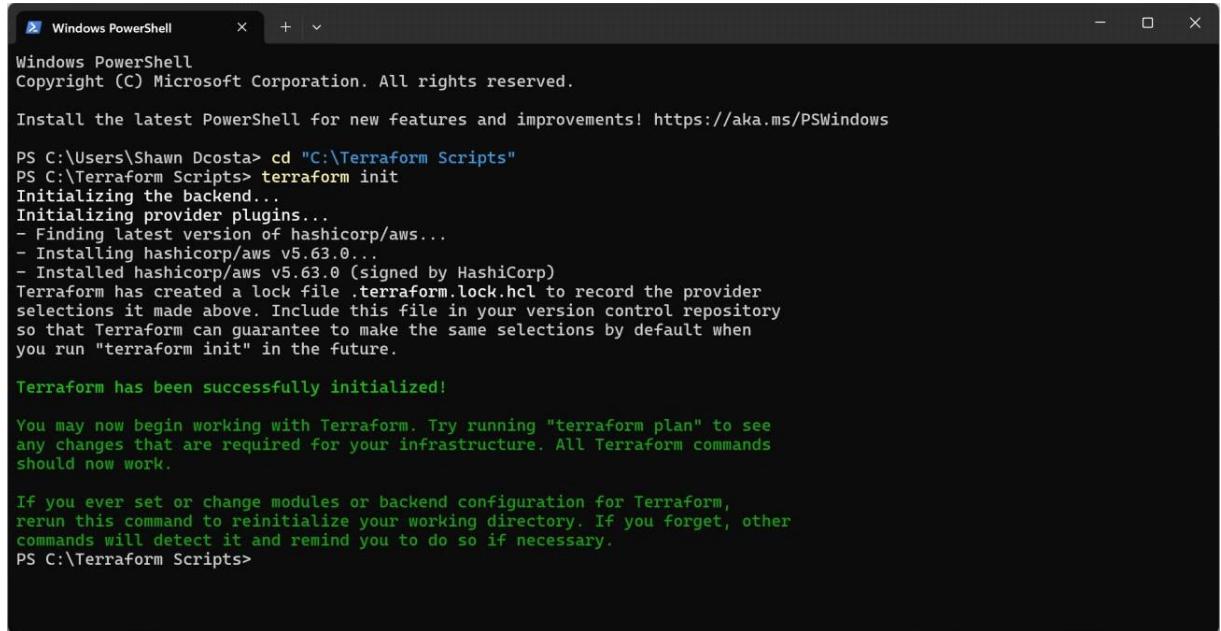
The status bar at the bottom indicates "Ln 8, Col 28 218 characters" and "Windows (CRLF) UTF-8".

Note : a) Change Access Key and Secret Key to the one in the downloaded CSV file.

Step – 12 : Add the path of ‘AdvDevOps’ folder to the PATH in system variables and user variables.



Step – 13 : Run the following command on Command Prompt/PowerShell.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Shawn Dcosta> cd "C:\Terraform Scripts"
PS C:\Terraform Scripts> terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.63.0...
- Installed hashicorp/aws v5.63.0 (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.
PS C:\Terraform Scripts>
```

Note: Make sure you enter the right name of the folder while specifying the path.

```
Windows PowerShell
PS C:\Terraform Scripts> 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-00399ec92321828f5"
    + arn                                = "(known after apply)"
    + associate_public_ip_address        = "(known after apply)"
    + availability_zone                  = "(known after apply)"
    + cpu_core_count                     = "(known after apply)"
    + cpu_threads_per_core              = "(known after apply)"
    + disable_api_stop                 = "(known after apply)"
    + disable_api_termination          = "(known after apply)"
    + ebs_optimized                      = "(known after apply)"
    + get_password_data                = false
    + host_id                            = "(known after apply)"
    + host_resource_group_arn          = "(known after apply)"
    + iam_instance_profile             = "(known after apply)"
    + id                                 = "(known after apply)"
    + instance_initiated_shutdown_behavior = "(known after apply)"
    + instance_lifecycle               = "(known after apply)"
    + instance_state                   = "(known after apply)"
    + instance_type                     = "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)"

    + capacity_reservation_specification (known after apply)
    + cpu_options (known after apply)
    + ebs_block_device (known after apply)
    + enclave_options (known after apply)
    + ephemeral_block_device (known after apply)
    + instance_market_options (known after apply)
    + maintenance_options (known after apply)
    + metadata_options (known after apply)
    + network_interface (known after apply)
    + private_dns_name_options (known after apply)
    + root_block_device (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.
```

```

PS C:\Terraform Scripts> terraform apply

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-08200e1ffcab6af77"
    + arn                               = (known after apply)
    + associate_public_ip_address       = (known after apply)
    + availability_zone                = (known after apply)
    + cpu_core_count                  = (known after apply)
    + cpu_threads_per_core            = (known after apply)
    + disable_api_stop                = (known after apply)
    + disable_api_termination         = (known after apply)
    + ebs_optimized                   = (known after apply)
    + get_password_data               = false
    + host_id                          = (known after apply)
    + host_resource_group_arn          = (known after apply)
    + iam_instance_profile             = (known after apply)
    + id                               = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance.lifecycle              = (known after apply)
    + instance.state                  = (known after apply)
    + instance_type                   = "t4g.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)

    + capacity_reservation_specification (known after apply)
    + cpu_options (known after apply)
    + ebs_block_device (known after apply)
    + enclave_options (known after apply)
    + ephemeral_block_device (known after apply)
    + instance_market_options (known after apply)
    + maintenance_options (known after apply)
    + metadata_options (known after apply)
    + network_interface (known after apply)
    + private_dns_name_options (known after apply)
    + root_block_device (known after apply)
}

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

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

Enter a value: yes

aws_instance.Ubuntu: Creating...
aws_instance.Ubuntu: Still creating... [10s elapsed]
aws_instance.Ubuntu: Creation complete after 15s [id=i-0f4c3a86f0a51bcf3]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Terraform Scripts>

```

Step – 14 : Check if the instance has been created and is in the ‘Running’ state in the EC2 Dashboard.

EC2 Dashboard X

Instances (1) info

Find Instance by attribute or tag (case-sensitive)

Instance state: All states

Launch instances

Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
	i-0f4c3a86f0a51bcf3	Running	t4g.micro	2/2 checks passed	View alarms	eu-north-1b

Select an instance

EC2 Dashboard X

Instances

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances

Dedicated Hosts

Capacity

Reservations New

Images

AMIs

AMI Catalog

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

The screenshot shows the AWS EC2 Instances page with a single instance listed. The instance is named 'i-0f4c3a86f0a51bcf3', is in the 'Running' state, is a 't4g.micro' type, and has passed all 2/2 health checks. It is located in the 'eu-north-1b' availability zone. A modal window titled 'Select an instance' is overlaid on the main page, indicating that an action is being performed on the selected instance.

Step – 15 : Terminate the instance by using the following command and check it in EC2 Dashboard.

```

Windows PowerShell
PS C:\Terraform Scripts> terraform destroy
aws_instance.Ubuntu: Refreshing state... [id=i-0f4c3a86f0a51bcf3]
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
  - arn
  - associate_public_ip_address
  - availability_zone
  - cpu_core_count
  - cpu_threads_per_core
  - disable_api_stop
  - disable_api_termination
  - ebs_optimized
  - get_password_data
  - hibernation
  - id
  - instance_initiated_shutdown_behavior
  - instance_state
  - instance_type
  - ipv6_address_count
  - ipv6_addresses
  - monitoring
  - placement_partition_number
  - primary_network_interface_id
  - private_dns
  - private_ip
  - public_dns
  - public_ip
  - secondary_private_ips
  - security_groups
    - "default"
  - source_dest_check
  - subnet_id
  - tags
  - tags_all
  - tenancy
  - user_data_replace_on_change
  - vpc_security_group_ids
    - "sg-a1991a8635bae32",
  ] -> null
  # (8 unchanged attributes hidden)

- capacity_reservation_specification {
  - capacity_reservation_preference = "open" -> null
}

- cpu_options {
  - core_count = 2 -> null
  - threads_per_core = 1 -> null
  - # (1 unchanged attribute hidden)
}

- credit_specification {
  - cpu_credits = "unlimited" -> null
}

- enclave_options {
  - enabled = false -> null
}

- maintenance_options {
  - auto_recovery = "default" -> null
}

- metadata_options {
  - http_endpoint = "enabled" -> null
  - http_protocol_ipv6 = "disabled" -> null
  - http_put_response_hop_limit = 1 -> null
  - http_tokens = "optional" -> null
  - instance_metadata_tags = "disabled" -> null
}

- private_dns_name_options {
  - enable_resource_name_dns_a_record = false -> null
  - enable_resource_name_dns_aaaa_record = false -> null
  - hostname_type = "ip-name" -> null
}

- root_block_device {
  - delete_on_termination = true -> null
  - device_name = "/dev/sda1" -> null
  - encrypted
  - iops
  - tags
  - tags_all
  - throughput
  - volume_id = "vol-0a7ab1949a9318d7f" -> null
  - volume_size = 8 -> null
  - volume_type = "gp2" -> null
  - # (1 unchanged attribute hidden)
}

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

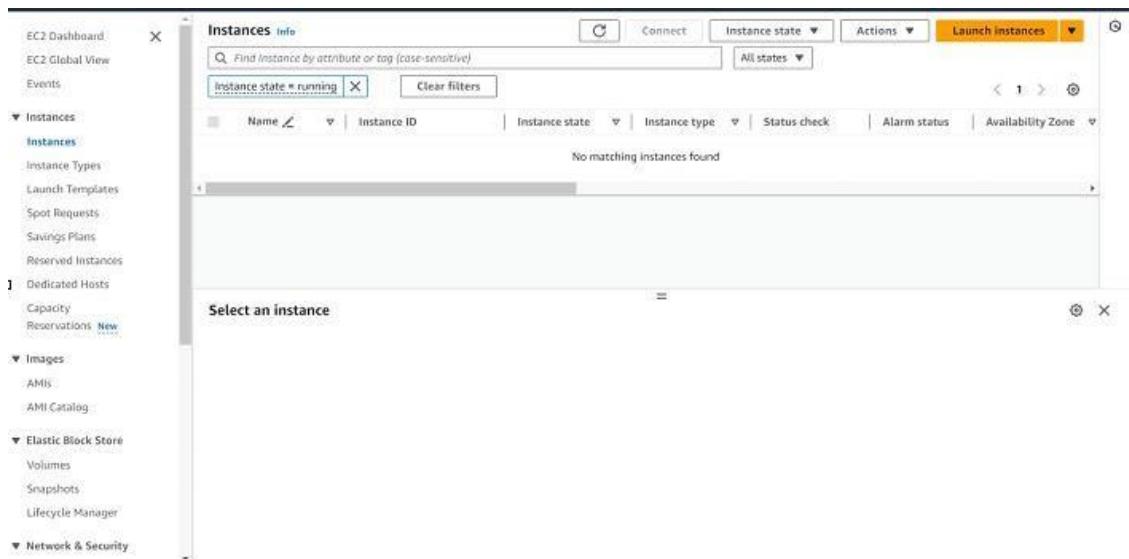
Do you really want to destroy all resources?
Terraform will destroy all your managed infrastructure, as shown above.
There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_instance.Ubuntu: Destroying... [id=i-0f4c3a86f0a51bcf3]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 20s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 30s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 40s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 50s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 1m0s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0f4c3a86f0a51bcf3, 1m10s elapsed]
aws_instance.Ubuntu: Destruction complete after 1m13s

Destroy complete! Resources: 1 destroyed.
PS C:\Terraform Scripts>

```



Conclusion : Successfully installed Terraform and ran an EC2 instance using it on Windows.

Name: Saikarthik Iyer

Roll no : 41

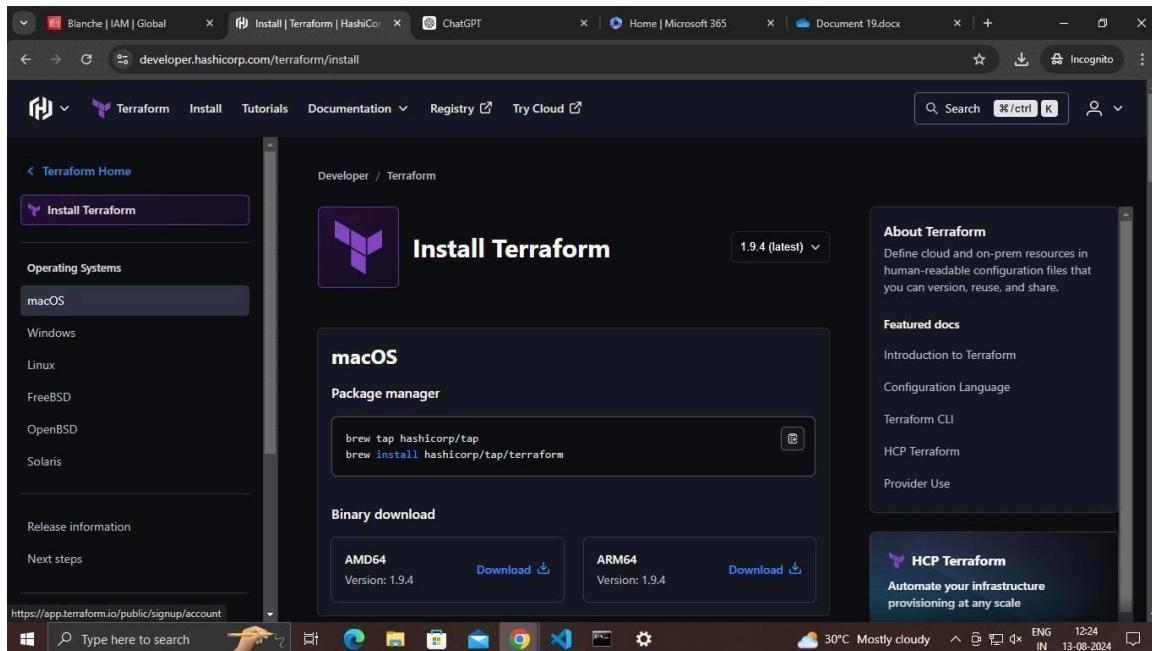
Batch : T13

Assignment 5 & 6: Terraform

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

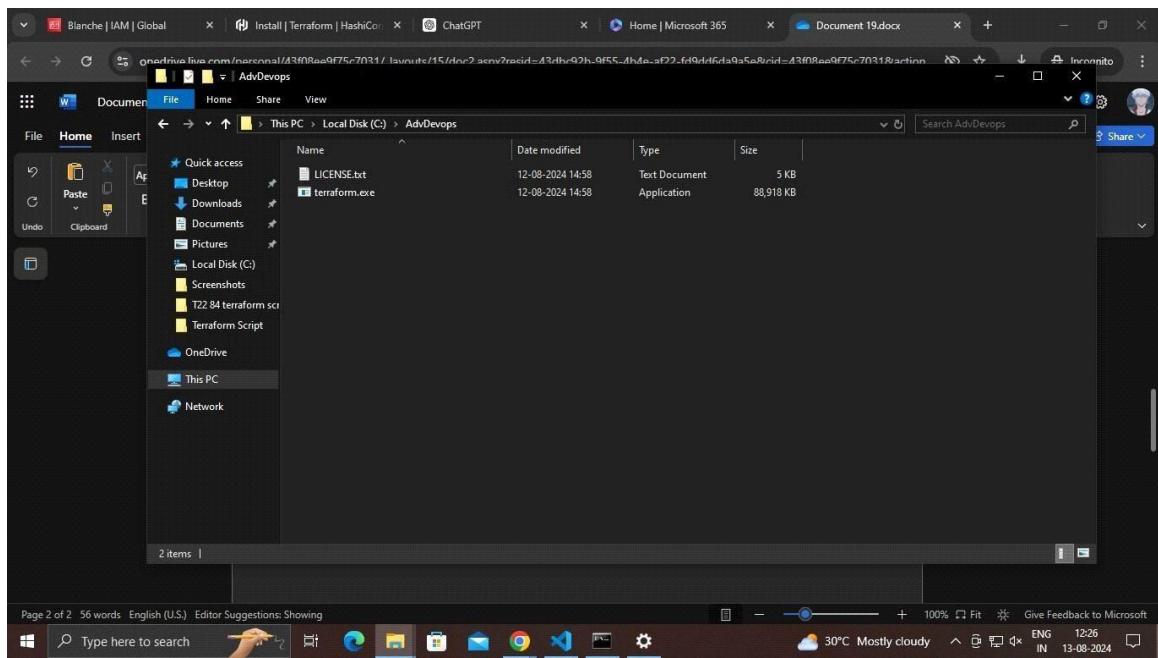
STEPS For Installation-

Google Search - Terraform DOWNLOAD



Create a folder in your c drive named advdevops

Then extract the downloaded file in the c drive folder named advdevops

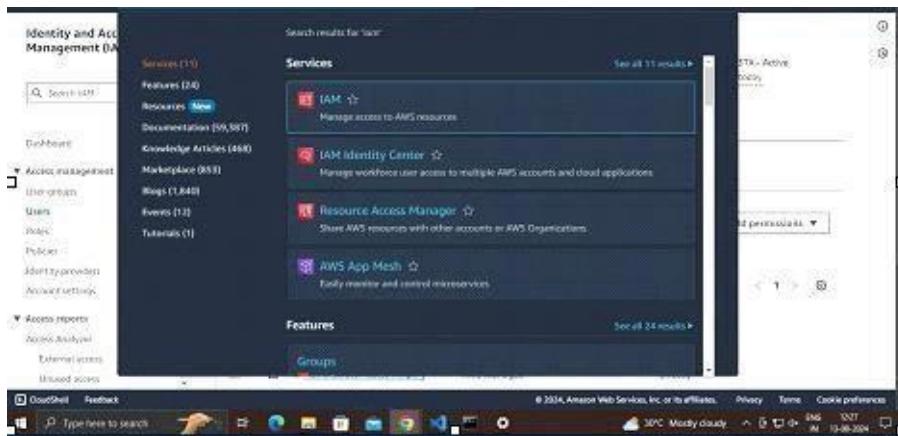


Till that time open

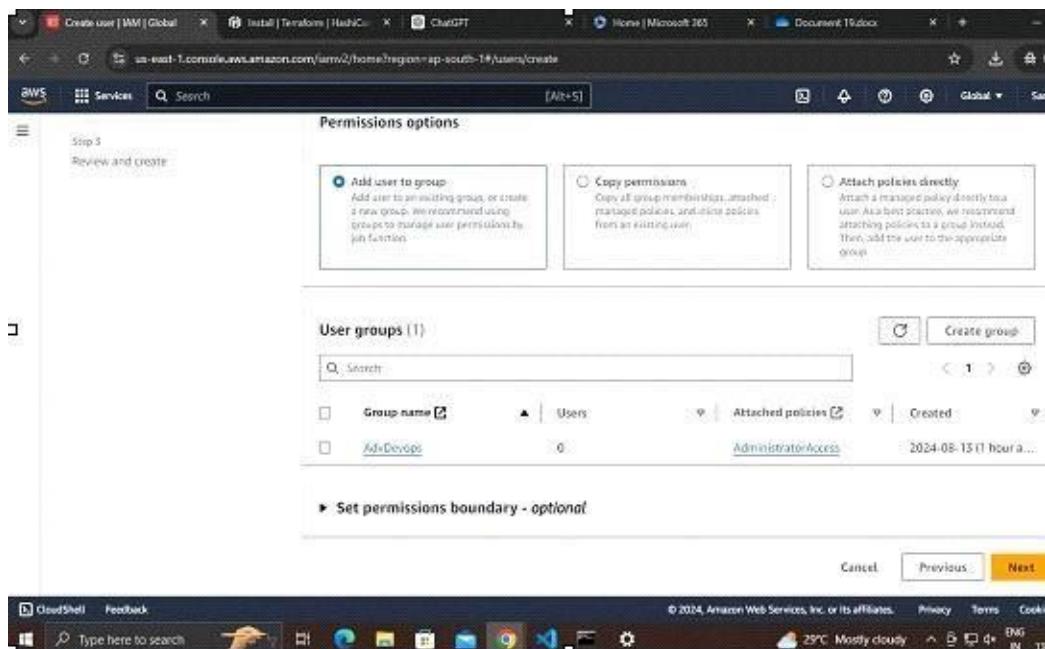
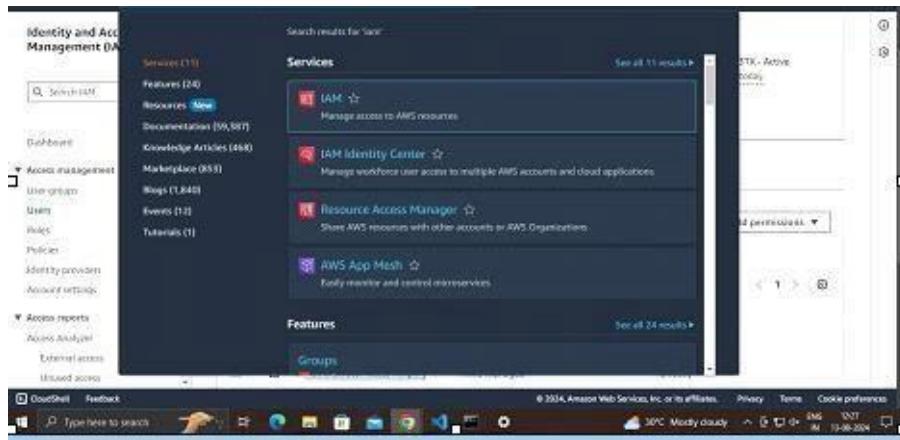
and login aws

consoleSearch IAM

and click on it



Click on add users in the user section



Give

user

nam

e

Cre
ate

grou

p

Select policies

The screenshot shows the AWS IAM 'Add user' wizard, step 2: Create group. The dialog title is 'Create group'. It displays a list of policies for the group 'AdDevops'. The table has columns: Policy name, Type, Used as, and Description. One policy, 'AdministratorAccess', is selected (indicated by a checked checkbox). Other policies listed include 'AdministratorAccess-Amplify', 'AdministratorAccess-AWSelastic...', 'AlexaForBusinessDeviceSetup', and 'AlexaForBusinessFullAccess'. At the bottom right of the dialog are 'Cancel' and 'Create group' buttons.

	Policy name	Type	Used as	Description
<input checked="" type="checkbox"/>	AdministratorAccess	Job function	None	Provides full access to AWS services and resources.
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permissions while explicitly allowing direct acce...
<input type="checkbox"/>	AdministratorAccess-AWSelastic...	AWS managed	None	Grants account administrative permissions. Explicitly allows developers and a...
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaForBusiness services
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness resources and access to related AWS...

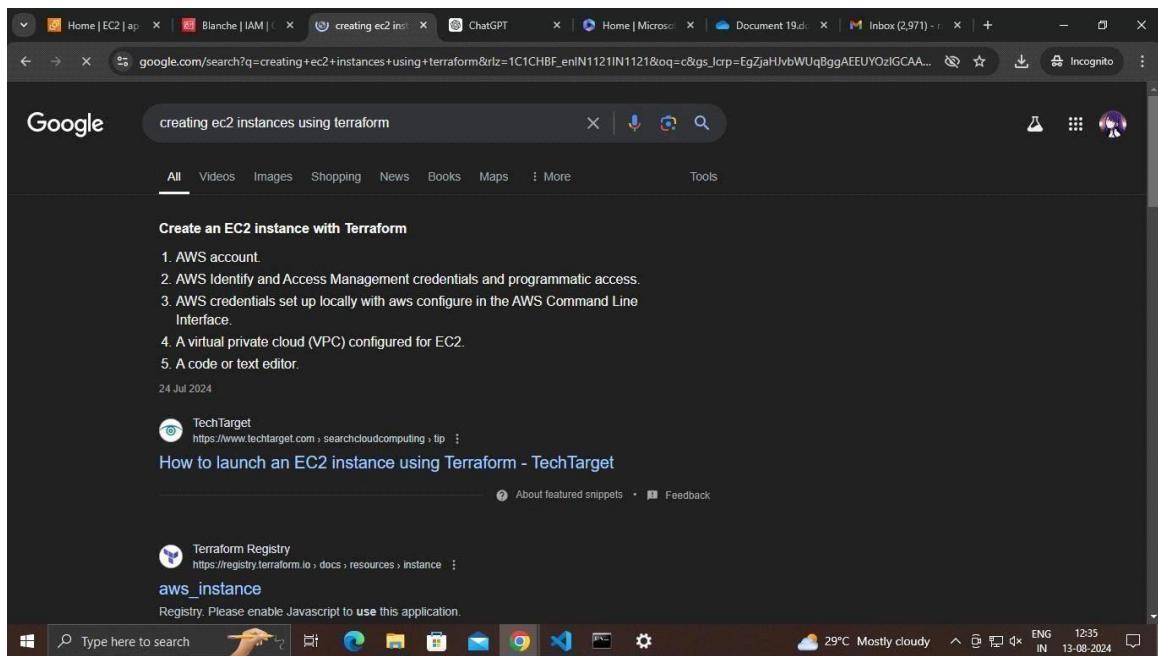
Download .csv file and get acess key

The screenshot shows the AWS Identity and Access Management (IAM) service in the AWS Management Console. A new user named 'user1' has been created. The 'Summary' tab is selected, displaying details such as the ARN, access key status, and creation date. Other tabs include 'Permissions', 'Groups', 'Tags', 'Security credentials', and 'Access Advisor'. Below the summary, there's a 'Console sign-in' section with a 'Enable console access' button. The left sidebar lists various IAM management options like Dashboard, Access management, User groups, Users, Roles, Policies, Identity providers, Account settings, and Access reports.

Go to ec2

The screenshot shows the AWS search interface with the query 'ec2' entered. The results are categorized into 'Services', 'Features', and 'Documentation'. The top result is 'EC2' under Services, described as 'Virtual Servers in the Cloud'. Other visible results include 'EC2 Image Builder', 'Recycle Bin', and 'Amazon Inspector'. The bottom section shows 'Features' and a 'Dashboard' link. The interface includes standard AWS navigation elements like CloudShell, Feedback, and a search bar.

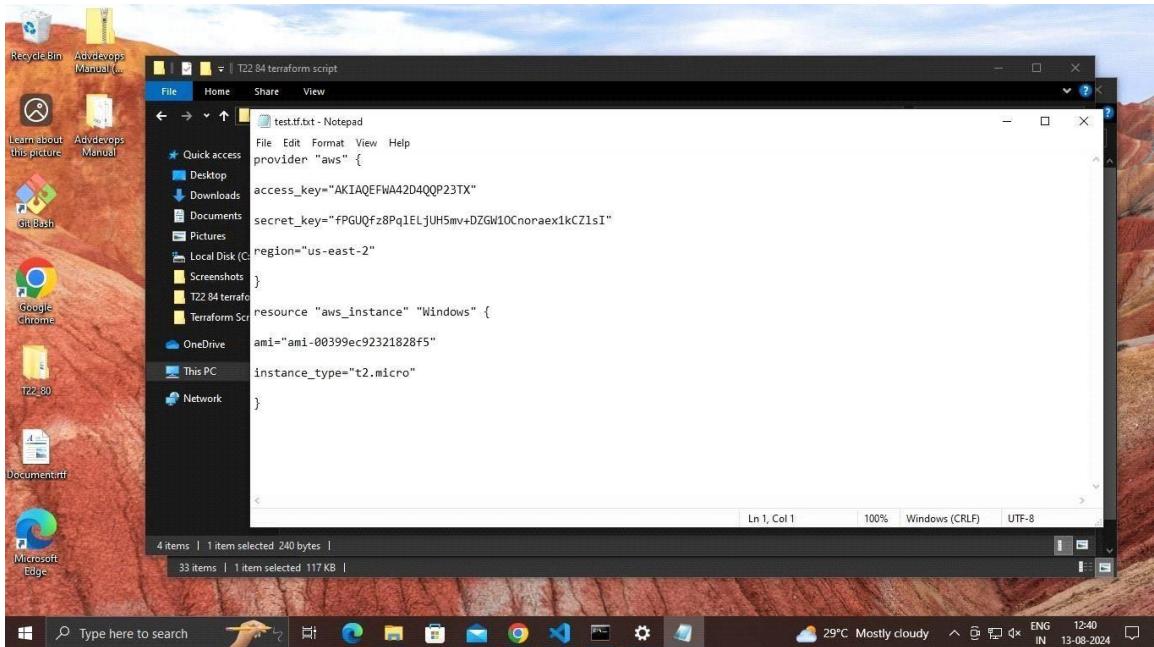
Search the following on google



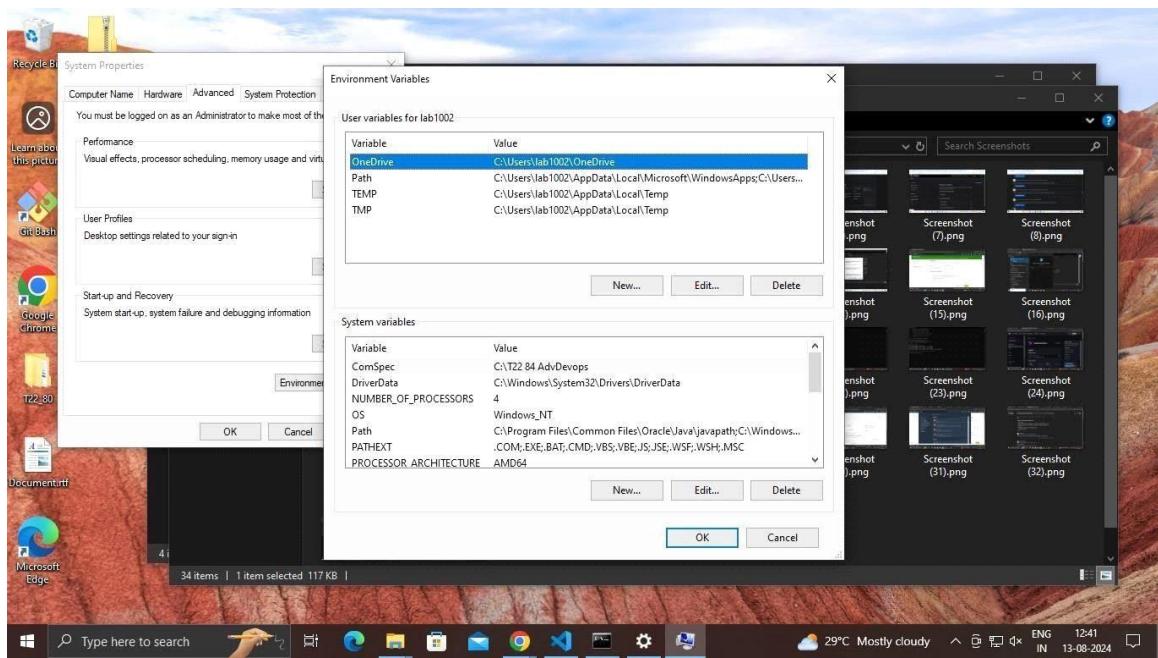
Create a folder names terraform script

go to notepad and type the details properly change access key and secret key in .csv file

Set region same as below



Now search for system environment variables and and select the path of both as the folderof advdevops



Open command prompt and paste the path of terraform script

Eg cd Terraform Script as shown below

Now type terraform init command

```
C:\T22 84 terraform script>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.62.0...
- Installed hashicorp/aws v5.62.0 (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.
```

Now terraform plan

```
Command Prompt
C:\T22 84 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.Windows will be created
+ resource "aws_instance" "Windows" {
    + ami                               = "ami-00399ec92321828f5"
    + arm                               = (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               = (known after apply)
    + host_id                          = (known after apply)
    + host_resource_group_arn          = (known after apply)
    + iam_instance_profile             = (known after apply)
    + id                               = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle               = (known after apply)
    + instance_state                  = (known after apply)
    + instance_type                   = "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
}

Type here to search 29°C Mostly cloudy ENG 12:43 IN 13-08-2024
```

Now terraform apply

```
Command Prompt
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:\T22 84 terraform script>terraform apply

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.Windows will be created
+ resource "aws_instance" "Windows" {
    + ami                               = "ami-00399ec92321828f5"
    + arm                               = (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               = (known after apply)
    + host_id                          = (known after apply)
    + host_resource_group_arn          = (known after apply)
    + iam_instance_profile             = (known after apply)
    + id                               = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle               = (known after apply)
    + instance_state                  = (known after apply)
    + instance_type                   = "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)
}

Type here to search 29°C Mostly cloudy ENG 12:44 IN 13-08-2024
```

Type yes

```
Command Prompt
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.Windows: Creating...
aws_instance.Windows: Still creating... [10s elapsed]
aws_instance.Windows: Still creating... [20s elapsed]
aws_instance.Windows: Still creating... [30s elapsed]
aws_instance.Windows: Creation complete after 37s [id=i-087cdcf903de26f85]

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

C:\V22 84 terraform script>
```

Now go to ec2 and check thaT THERE IS AN INSTANCE created by the name of WINDOWSand is in running status and then come back to the command prompt and terminate the instance by typing

Terraform destroy

```
Command Prompt - terraform destroy
    # (1 unchanged attribute hidden)
}

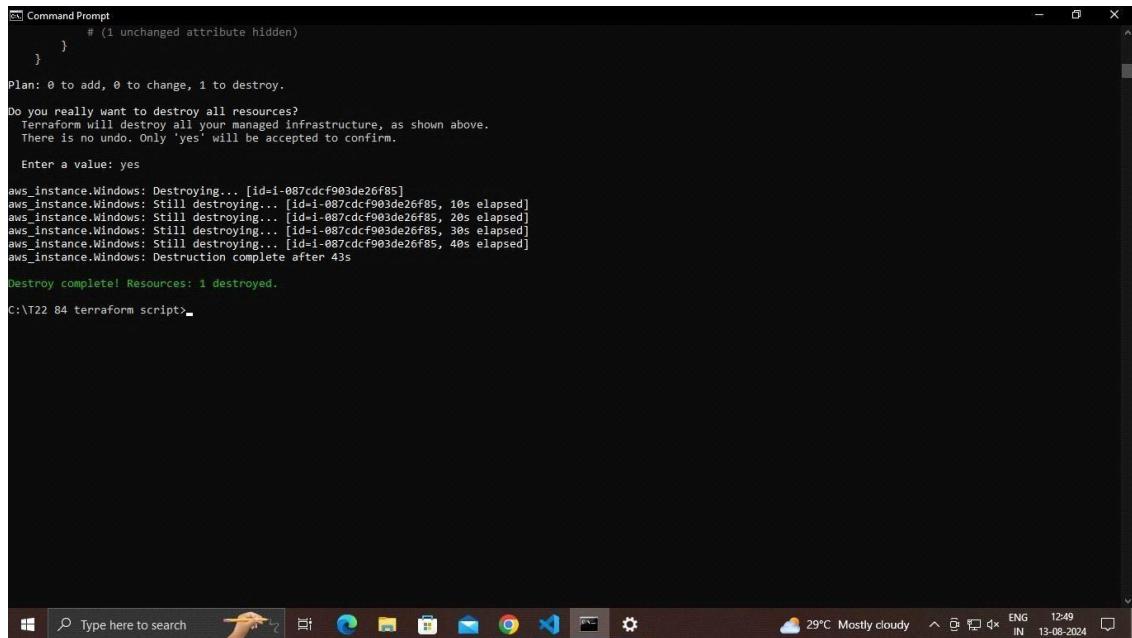
Plan: 0 to add, 0 to change, 1 to destroy.

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: -
```

Type yes

The resources will be destroyed



```
PS C:\T22 84 terraform script> terraform destroy
      # (1 unchanged attribute hidden)
    }

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.Windows: Destroying... [id=i-087cdcf903de26f85]
aws_instance.Windows: Still destroying... [id=i-087cdcf903de26f85, 10s elapsed]
aws_instance.Windows: Still destroying... [id=i-087cdcf903de26f85, 20s elapsed]
aws_instance.Windows: Still destroying... [id=i-087cdcf903de26f85, 30s elapsed]
aws_instance.Windows: Still destroying... [id=i-087cdcf903de26f85, 40s elapsed]
aws_instance.Windows: Destruction complete after 43s

Destroy complete! Resources: 1 destroyed.

C:\T22 84 terraform script>
```

Now go back to EC2 check if the instance is terminated , if yes
then logout of the awsconsole

And close the command prompt

Conclusion: Terraform performed and LO3 achieved

Saikarthik Iyer

T13

41

ASSIGNMENT 8

AIM: To understand AWS Lambda Functions and create a Lambda function using Python to log “An Image has been added” message, once a file is added to an S3 Bucket.

THEORY:

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). It allows you to run code in response to events without having to provision or manage servers. AWS Lambda functions are small, self-contained units of code that can be executed in the cloud, triggered by various AWS services or external events.

Key features of AWS Lambda functions include:

1. Event-Driven: Lambda functions are designed to be event-driven. They can be triggered by various events such as changes to data in Amazon S3, updates to a DynamoDB table, HTTP requests through Amazon API Gateway, and more.

2. Scaling: AWS Lambda automatically scales your functions in response to incoming traffic. You don't need to worry about server provisioning or capacity planning.
3. Pay-as-You-Go: With Lambda, you pay only for the compute time consumed during function execution. There are no upfront costs or infrastructure management fees.
4. Support for Multiple Languages: You can write Lambda functions in various programming languages, including Node.js, Python, Java, C#, Ruby, Go, and custom runtimes.
5. Stateless and Isolated: Each Lambda function is stateless and runs in isolation, ensuring that one function's execution doesn't affect another. Data can be stored and retrieved using other AWS services like S3 or DynamoDB.
6. Easy Integration: Lambda functions can be easily integrated with other AWS services, creating serverless architectures and event-driven workflows.
7. Custom Runtimes: You can use custom runtimes to execute functions written in languages not officially supported by AWS.

AWS Lambda is a powerful tool for building serverless applications, automating tasks, and responding to events in a highly scalable and cost-effective manner.

Amazon S3

Buckets

Access Points

Object Lambda Access Points

Multi-Region Access Points

Batch Operations

IAM Access Analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

AWS Organizations settings

Feature spotlight

Account snapshot

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (1) Info

Buckets are containers for data stored in S3. [Learn more](#)

Name	AWS Region	Access	Creation date
codepipeline-eu-north-1-530481351972	Europe (Stockholm) eu-north-1	Bucket and objects not public	August 8, 2023, 16:19:22 (UTC+05:30)

View Storage Lens dashboard

Copy ARN

Empty

Delete

Create bucket

Find buckets by name

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 3:06 PM IN 8/29/2023

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name: triggerimagebucket53

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region: Europe (Stockholm) eu-north-1

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Choose bucket

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 3:10 PM IN 8/29/2023

The screenshot shows the 'Default encryption' step of the AWS S3 Bucket creation wizard. It includes options for encryption type (SSE-S3 selected), bucket key (disabled), and advanced settings. A note indicates that files can be uploaded after creation.

Encryption type [Info](#)

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service keys (SSE-KMS)
- Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

- Disable
- Enable

▶ Advanced settings

ⓘ After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel [Create bucket](#)



The screenshot shows the 'Buckets' list in the AWS S3 console. It displays two buckets: 'codepipeline-eu-north-1-530481351972' and 'triggerimagebucket53'. Both were created on August 8, 2023, at 16:19:22 (UTC+05:30) and are located in Europe (Stockholm) eu-north-1.

Successfully created bucket "triggerimagebucket53". To upload files and folders, or to configure additional bucket settings choose [View details](#).

Amazon S3 > Buckets

▶ Account snapshot

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

Buckets (2) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

Name	AWS Region	Access	Creation date
codepipeline-eu-north-1-530481351972	Europe (Stockholm) eu-north-1	Bucket and objects not public	August 8, 2023, 16:19:22 (UTC+05:30)
triggerimagebucket53	Europe (Stockholm) eu-north-1	Bucket and objects not public	August 29, 2023, 15:10:31 (UTC+05:30)

Copy ARN Empty Delete Create bucket



Goint to IAM

Screenshot of the AWS IAM Dashboard:

The dashboard shows the following information:

- Security recommendations:**
 - Add MFA for root user
 - Root user has no active access keys
- AWS Account:** Account ID: 538767473830, Account Alias: Create, Sign-in URL: https://538767473830.signin.aws.amazon.com/console
- IAM resources:** Resources in this AWS Account
 - User groups: 2
 - Users: 1
 - Roles: 8
 - Policies: 2
 - Identity providers: 0
- Quick Links:** My security credentials, Manage your access keys, multi-factor authentication (MFA) and other

Bottom navigation bar: https://us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/roles

Screenshot of the 'Create role' wizard - Step 1: Select trusted entity:

The page shows the following steps:

- Step 1: Select trusted entity
- Step 2: Add permissions
- Step 3: Name, review, and create

Trusted entity type:

- AWS service: Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case:
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
Common use cases:
 EC2: Allows EC2 instances to call AWS services on your behalf.

Bottom navigation bar: https://us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/roles

Screenshot of the AWS IAM User Pools console showing the "Step 3: Name, review, and create" configuration screen.

The "Use case" section is expanded, showing the following options:

- Allow AWS services like EC2, Lambda, or others to perform actions in this account.**
- Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.**
- Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.**
- SAML 2.0 federation**
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy**
Create a custom trust policy to enable others to perform actions in this account.

The "Use case" dropdown shows "Choose a service to view use case".

At the bottom right are "Cancel" and "Next" buttons.

Screenshot of the AWS IAM console showing the search results for "S3".

Step 3: Name, review, and create.

Search bar: "S3" (highlighted)

Filter: "S3" (highlighted)

Clear filters

	Policy name	Type	Description
<input checked="" type="checkbox"/>	AmazonS3FullAccess	AWS m...	Provides full access to all buckets via the AWS Management Console.
<input type="checkbox"/>	AmazonS3ReadOnly...	AWS m...	Provides read only access to all buckets via the AWS Management Console.
<input type="checkbox"/>	AmazonDMSRedsh...	AWS m...	Provides access to manage S3 settings for Redshift endpoints for DMS.
<input type="checkbox"/>	QuickSightAccessF...	AWS m...	Policy used by QuickSight team to access customer data produced by S3 ...
<input type="checkbox"/>	AmazonS3Outposts...	AWS m...	Provides full access to Amazon S3 on Outposts via the AWS Management...
<input type="checkbox"/>	AmazonS3Outposts...	AWS m...	Provides read only access to Amazon S3 on Outposts via the AWS Manag...
<input type="checkbox"/>	AmazonS3ObjectLa...	AWS m...	Provides AWS Lambda functions permissions to interact with Amazon S3 ...
<input type="checkbox"/>	AWSBackupService...	AWS m...	Policy containing permissions necessary for AWS Backup to restore a S3 ...
<input type="checkbox"/>	AWSBackupService...	AWS m...	Policy containing permissions necessary for AWS Backup to backup data i...

Set permissions boundary - optional Info

Select a permissions boundary to control the maximum permissions this role can have. This is not a common setting, but you can use it to delegate permission.

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG IN 8/29/2023

Screenshot of the AWS IAM console showing the search results for "cloudwatch".

Step 3: Name, review, and create.

Search bar: "cloudwatch" (highlighted)

Clear filters

	Policy name	Type	Description
<input checked="" type="checkbox"/>	CloudWatchFullAcc...	AWS m...	Provides full access to CloudWatch.
<input type="checkbox"/>	CloudWatchReadO...	AWS m...	Provides read only access to CloudWatch.
<input type="checkbox"/>	CloudWatchLogsFul...	AWS m...	Provides full access to CloudWatch Logs
<input type="checkbox"/>	CloudWatchLogsRe...	AWS m...	Provides read only access to CloudWatch Logs
<input type="checkbox"/>	CloudWatchActions...	AWS m...	Provides read-only access to CloudWatch alarms and metrics as well as E...
<input type="checkbox"/>	AmazonAPIGatwea...	AWS m...	Allows API Gateway to push logs to user's account.
<input type="checkbox"/>	AmazonDMSCloud...	AWS m...	Provides access to upload DMS replication logs to cloudwatch logs in cust...
<input type="checkbox"/>	CloudWatchEvents...	AWS m...	Provides read only access to Amazon CloudWatch Events.
<input type="checkbox"/>	CloudWatchEvents...	AWS m...	Allows built-in targets in Amazon CloudWatch Events to perform EC2 actio...
<input type="checkbox"/>	CloudWatchEventsI...	AWS m...	Allows Amazon CloudWatch Events to relay events to the streams in AWS...

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG IN 8/29/2023

AWS Services Search [Alt+S] Global Shreya Kamath

Permissions policy summary

Policy name	Type	Attached as
CloudWatchFullAccess	AWS managed	Permissions policy
AmazonS3FullAccess	AWS managed	Permissions policy

Tags

Add tags - optional Info Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag You can add up to 50 more tags.

Cancel Previous Create role

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 3:22 PM IN 8/29/2023

AWS Services Search [Alt+S] Global Shreya Kamath

Identity and Access Management (IAM)

Role triggerrole53 created. View role

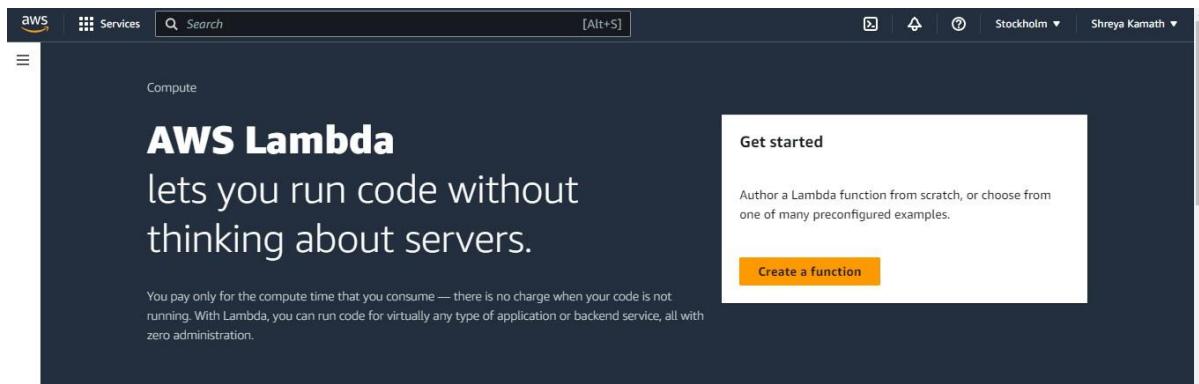
IAM > Roles

Roles (9) Info An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

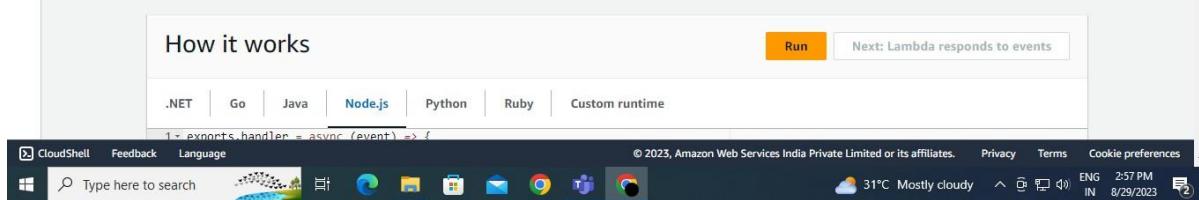
Role name	Trusted entities	Last activity
AWSCloud9SSMAccessRole	AWS Service: cloud9, and 1 more.	27 days ago
AWSCodePipelineServiceRole-eu-north-1-webapppipeline1	AWS Service: codepipeline	20 days ago
AWSServiceRoleForAmazonSSM	AWS Service: ssm (Service-Linked Role)	31 minutes ago
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	28 days ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-
cwe-role-eu-north-1-webapppipeline1	AWS Service: events	-
deployrole	AWS Service: codedeploy	-
triggerrole53	AWS Service: lambda	-

Roles Anywhere Manage

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 3:23 PM IN 8/29/2023



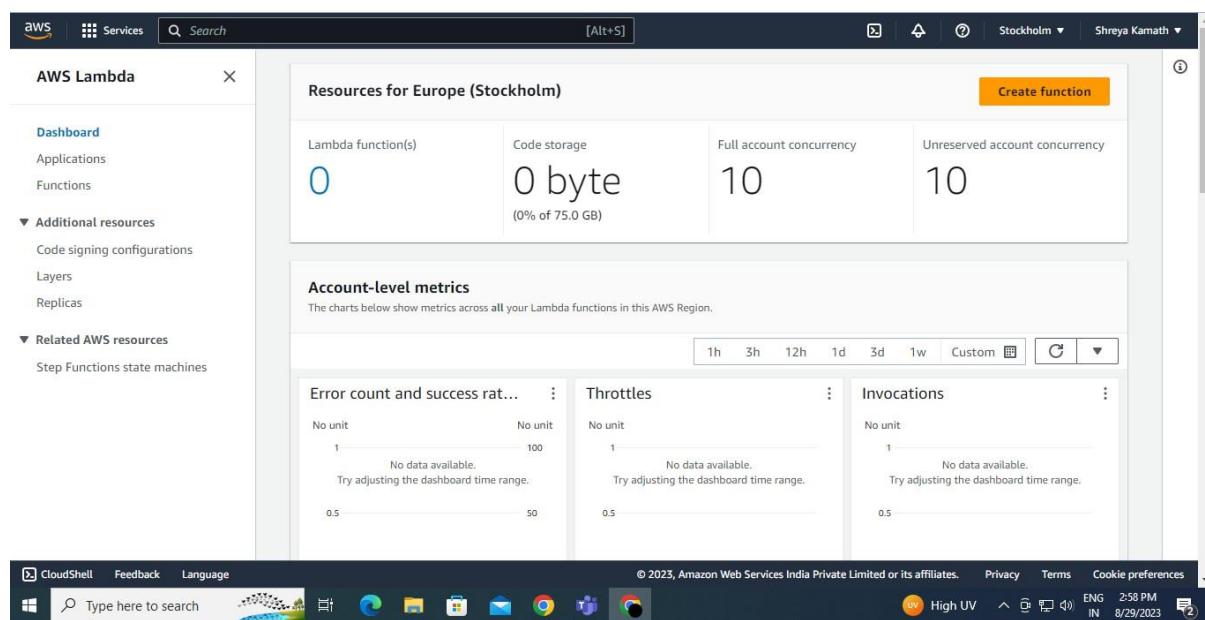
The screenshot shows the AWS Lambda landing page. At the top, there's a search bar and navigation links for 'Services' and 'Compute'. The main heading is 'AWS Lambda' with the tagline 'lets you run code without thinking about servers.' Below this, a paragraph explains the cost model: 'You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.' To the right, a 'Get started' section includes a 'Create a function' button.



This screenshot shows the 'How it works' page for AWS Lambda. It features a navigation bar with tabs for '.NET', 'Go', 'Java', 'Node.js' (which is selected), 'Python', 'Ruby', and 'Custom runtime'. Below the tabs, there's a code snippet example:

```
1 exports.handler = async (event) => {
```

The page also includes a 'Run' button and a link to 'Next: Lambda responds to events'. The Windows taskbar at the bottom shows various open applications like File Explorer, Edge, and Google Chrome.



The screenshot displays the AWS Lambda dashboard for the Europe (Stockholm) region. On the left, a sidebar lists 'AWS Lambda' under 'Compute', with sections for 'Dashboard', 'Applications', 'Functions', 'Additional resources', and 'Related AWS resources'. The main area shows 'Resources for Europe (Stockholm)' with metrics: 0 Lambda function(s), 0 byte of code storage (0% of 75.0 GB), 10 units of full account concurrency, and 10 units of unreserved account concurrency. Below this is an 'Account-level metrics' section with three charts: 'Error count and success rate', 'Throttles', and 'Invocations', each showing a single data point of 1 with a note that 'No data available. Try adjusting the dashboard time range.' The Windows taskbar at the bottom shows the same set of open applications as the previous screenshot.

Screenshot of the AWS Lambda 'Create function' wizard.

The top navigation bar shows 'Services' and a search bar. The location bar indicates 'Lambda > Functions > Create function'. The title is 'Create function' with an 'Info' link. Below it, a note says 'Choose one of the following options to create your function.'

Four options are listed:

- Author from scratch: Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.
- Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

A 'Basic information' section follows:

Function name: Enter a name that describes the purpose of your function. The input field contains 'triggerimagelambda'. A note below says 'Use only letters, numbers, hyphens, or underscores with no spaces.'

Runtime: Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. The dropdown menu shows 'Node.js 18.x'.

Architecture: Choose the instruction set architecture you want for your function code. The dropdown menu shows 'Node.js 18.x'.

The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, Language, and a search bar. It also displays weather information (31°C, Mostly cloudy), time (3:23 PM IN 8/29/2023), and user details (Shreya Kamath).

Screenshot of the AWS Lambda 'Create function' wizard, showing a different configuration.

The top navigation bar shows 'Services' and a search bar. The location bar indicates 'Lambda > Functions > Create function'. The title is 'Create function' with an 'Info' link. Below it, a note says 'Choose one of the following options to create your function.'

Four options are listed:

- Author from scratch: Start with a simple Hello World example.
- Use a blueprint: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image: Select a container image to deploy for your function.
- Browse serverless app repository: Deploy a sample Lambda application from the AWS Serverless Application Repository.

A 'Basic information' section follows:

Function name: Enter a name that describes the purpose of your function. The input field contains 'triggerimagelambda'. A note below says 'Use only letters, numbers, hyphens, or underscores with no spaces.'

Runtime: Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. The dropdown menu shows 'Python 3.10'.

Architecture: Choose the instruction set architecture you want for your function code. The dropdown menu shows 'x86_64'.

Permissions: By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Advanced settings: A collapsed section.

The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, Language, and a search bar. It also displays weather information (31°C, Mostly cloudy), time (3:24 PM IN 8/29/2023), and user details (Shreya Kamath).

AWS Services Search [Alt+S] Stockholm Shreya Kamath

x86_64 arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console [IAM console](#).

Create a new role with basic Lambda permissions
 Use an existing role
 Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

triggerrole53

View the triggerrole53 role [on the IAM console](#).

► Advanced settings

Create function

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy 3:24 PM IN 8/29/2023

AWS Services Search [Alt+S] Stockholm Shreya Kamath

Successfully created the function triggerimagelambda. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > triggerimagelambda

triggerimagelambda

Throttle Copy ARN Actions ▾

▼ Function overview Info

triggerimagelambda

Layers (0)

+ Add trigger + Add destination

Description

Last modified 13 seconds ago

Function ARN [arn:aws:lambda:eu-north-1:538767473830:function:triggerimagelambda](#)

Function URL [Info](#)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy 3:25 PM IN 8/29/2023

AWS Services Search [Alt+S]

key.

POST X

Prefix - optional
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.
e.g. images/

Suffix - optional
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters.
e.g. .jpg

Recursive invocation
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)

I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

Lambda will add the necessary permissions for AWS S3 to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Add



AWS Services Search [Alt+S]

Lambda > Functions > triggerimagelambda

triggerimagelambda

The trigger triggerimagebucket53 was successfully added to function triggerimagelambda. The function is now receiving events from the trigger.

Function overview Info

triggerimagelambda

S3

Add destination

Add trigger

Description
-

Last modified
2 minutes ago

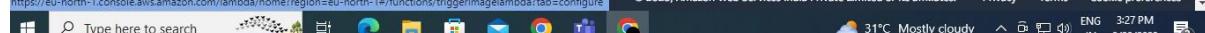
Function ARN
`arn:aws:lambda:eu-north-1:538767473830:function:triggerimagelambda`

Function URL Info

https://eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#functions/triggerimagelambda?tab=configure

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

CloudShell Feedback Language Type here to search 31°C Mostly cloudy 3:26 PM IN 8/29/2023



The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes 'Services', a search bar, and user information for 'Stockholm' and 'Shreya Kamath'. The main area has tabs for 'Code source' and 'Info', with 'Test' and 'Deploy' buttons. A status message 'Changes not deployed' is visible. The code editor displays a file named 'lambda_function.py' with the following content:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('An Image has been added!')
8     }
```

The screenshot shows the AWS Lambda function editor interface after a successful update. A green notification bar at the top states 'Successfully updated the function triggerimagelambda.'. The rest of the interface is identical to the previous screenshot, showing the 'lambda_function.py' code.

Go back to s3

AWS Services Search [Alt+S] Global Shreya Kamath

Files and folders (1 Total, 840.9 KB)

All files and folders in this table will be uploaded.

Find by name < 1 >

Name Folder Type Size

Destination

Destination s3://triggerimagebucket53

▶ Destination details Bucket settings that impact new objects stored in the specified destination.

▶ Permissions Grant public access and access to other AWS accounts.

▶ Properties Specify storage class, encryption settings, tags, and more.

Cancel Upload

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 3:35 PM IN 8/29/2023

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 4:00 PM IN 8/29/2023

Upload succeeded View details below.

Upload: status Close

The information below will no longer be available after you navigate away from this page.

Summary

Destination	Succeeded	Failed
s3://triggerimagebucket53	1 file, 840.9 KB (100.00%)	0 files, 0 B (0%)

Files and folders Configuration

Files and folders (1 Total, 840.9 KB)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 4:00 PM IN 8/29/2023

Screenshot of the AWS IAM Permissions page for the role 'triggerrole53'.

The left sidebar shows the IAM navigation menu. The 'Permissions' tab is selected in the top navigation bar.

Permissions policies (2)

You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

Policy name	Type	Description
CloudWatchFullAccess	AWS managed	Provides full access to CloudWatch
AmazonS3FullAccess	AWS managed	Provides full access to all bucke

Permissions boundary - (not set)

Set a permissions boundary to control the maximum permissions this role can have.

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 4:01 PM IN 8/29/2023

Screenshot of the 'Add permissions' dialog for the role 'triggerrole53'.

The URL in the browser is [IAM > Roles > triggerrole53 > Add permissions](#).

Attach policy to triggerrole53

Current permissions policies (2)

Other permissions policies (Selected 1/874)

Filter policies by property or policy name and press enter. 1 match

"amazonseventbridgefullaccess"

Policy name	Type	Description
AmazonEventBridgeFullAccess	AWS managed	Provides full access to A

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Mostly cloudy ENG 4:01 PM IN 8/29/2023

AWS Services Search [Alt+S] Global Shreya Kamath

Last activity None Maximum session duration 1 hour

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analyzers

Permissions policies (3) Info You can attach up to 10 managed policies.

Filter policies by property or policy name and press enter.

Policy name Type Description

Policy name	Type	Description
CloudWatchFullAccess	AWS managed	Provides full access to CloudWatch
AmazonS3FullAccess	AWS managed	Provides full access to all buckets
AmazonEventBridgeFullAccess	AWS managed	Provides full access to Amazon Event

Permissions boundary - (not set) Info

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Type here to search 31°C Mostly cloudy 4:02 PM 8/29/2023

The screenshot shows the AWS Identity and Access Management (IAM) service interface. On the left, there's a navigation sidebar with links for Dashboard, Access management, Access reports, and other IAM features like Policies and Roles. The main content area is titled 'Permissions policies' and shows three managed policies: CloudWatchFullAccess, AmazonS3FullAccess, and AmazonEventBridgeFullAccess. Each policy has a small icon, the name, the type (AWS managed), and a brief description. At the bottom of the page, there's a 'Permissions boundary' section which is currently set to '(not set)'. The browser's address bar and various system icons are visible at the very bottom of the screen.

AWS Services Search [Alt+S] Stockholm Shreya Kamath

Your changes have been saved.

Lambda > Functions > triggerimagelambda

triggerimagelambda

Throttle Copy ARN Actions

Function overview Info

triggerimagelambda

Layers (0)

S3 Amazon EventBridge

+ Add trigger + Add destination

Description -

Last modified 33 minutes ago

Function ARN arn:aws:lambda:eu-north-1:538767473830:function:triggerimagelambda

Function URL Info -

This screenshot shows the AWS Lambda function configuration page for 'triggerimagelambda'. It displays the function's name, a description field, and a trigger section. The trigger section shows an S3 icon and an Amazon EventBridge icon, with '+ Add trigger' and '+ Add destination' buttons. On the right, there are buttons for Throttle, Copy ARN, and Actions. Below the main configuration area is a detailed view of the function's ARN and URL.

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Type here to search 31°C Mostly cloudy 4:04 PM ENG IN 8/29/2023

aws Services Search [Alt+S] Global Shreya Kamath

Files and folders (1 Total, 840.9 KB) Remove Add files Add folder

All files and folders in this table will be uploaded.

Find by name < 1 >

Name Folder Type Size

Destination

Destination s3://triggerimagebucket53

Destination details Bucket settings that impact new objects stored in the specified destination.

Permissions Grant public access and access to other AWS accounts.

Properties Specify storage class, encryption settings, tags, and more.

Cancel Upload

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Type here to search 31°C Mostly cloudy 4:04 PM ENG IN 8/29/2023

This screenshot shows the AWS S3 upload interface. It displays a list of files (1 total, 840.9 KB) with buttons for Remove, Add files, and Add folder. Below the list is a search bar and a sorting header for Name, Folder, Type, and Size. A 'Destination' section is present, showing the target bucket 's3://triggerimagebucket53'. It includes a 'Destination details' section with a note about bucket settings and sections for Permissions and Properties. At the bottom are 'Cancel' and 'Upload' buttons.

The screenshot shows the AWS Lambda 'Upload: status' page. At the top, a green banner displays the message 'Upload succeeded' with a link to 'View details below.' Below the banner, the title 'Upload: status' is shown with a 'Close' button. A note indicates that the information will no longer be available after navigating away. The 'Summary' section shows the destination 's3://triggerimagebucket53' with a status of 'Succeeded' and '1 file, 840.9 KB (100.0%)'. The 'Failed' section shows '0 files, 0 B (0%)'. Below the summary, tabs for 'Files and folders' (selected) and 'Configuration' are visible. The 'Files and folders' section shows '(1 Total, 840.9 KB)' and lists a single file. The AWS navigation bar at the bottom includes CloudShell, Feedback, Language, a search bar, and system status.

The screenshot shows the AWS Lambda function monitoring interface. The top navigation bar includes CloudShell, Feedback, Language, a search bar, and system status. The main area is titled 'Function URL' with an 'Info' link. Below it, tabs for 'Code', 'Test', 'Monitor' (selected), 'Configuration', 'Aliases', and 'Versions' are shown. Under the 'Monitor' tab, sub-tabs for 'Metrics', 'Logs' (selected), and 'Traces' are visible, along with links to 'View CloudWatch logs', 'View X-Ray traces', 'View Lambda Insights', and 'View CodeGuru profiles'. A 'CloudWatch Logs' section follows, with an 'Info' link. It states that Lambda logs all requests and automatically stores logs generated by your code through Amazon CloudWatch Logs. It includes a table for recent and expensive function invocations and a link to the 'Monitor' section. A time range selector from '1h' to 'Custom' is present. The bottom navigation bar includes CloudShell, Feedback, Language, a search bar, and system status.

CONCLUSION:

Hence, I have understood AWS Lambda Functions and created a Lambda function using Python to log “An

“Image has been added” message, once a file is added to an S3 Bucket.

Name : Saikarthik Iyer

Batch : T13

Roll no : 41

Assignment 9

Aim : Nagios - Features and Applications / USE

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://an-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://code.load.github.com/NagiosEnterprises/nagioscore/tar.gz/nagios-4.4.2 [following]
--2021-10-05 10:24:05- https://code.load.github.com/NagiosEnterprises/nagioscore/tar.gz/nagios-4.4.2
Resolving code.load.github.com (code.load.github.com)... 13.233.43.20
Connecting to code.load.github.com (code.load.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.gz
```

then

```
sudo tar zxf nagios-plugins.tar.gz
```

then

```
cd nagios-plugins-release-2.2.1 finally
```

start and then check status of nagi os

```
sudo systemctl start nagios sudo
```

```
systemctl status nagios
```

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

What is Nagios? State three features with brief explanation and the applications of Nagios?

Nagios is an open source monitoring system for computer systems. Nagios software runs periodic checks on critical parameters of application, network and server resources. For example, Nagios can monitor memory usage, disk usage, microprocessor load, the number of currently running processes and log files.

The three features of Nagios are as follows:

1. Host and Service Monitoring: Nagios allows you to monitor both hosts (e.g., servers, network devices) and services (e.g., HTTP, FTP, disk usage). It provides real-time visibility into the health and performance of your IT infrastructure.
2. Flexible Configuration: Nagios features a highly configurable setup that allows you to define monitoring checks, schedules, and thresholds through text-based configuration files. This flexibility enables you to tailor monitoring to your specific needs.
3. Distributed Monitoring: Nagios supports distributed monitoring setups, allowing you to monitor multiple sites or networks from a central Nagios server. This is useful for large or geographically dispersed environments.

The applications of Nagios are:

1. Network Monitoring: Nagios can monitor network devices such as routers, switches, and firewalls. It checks for network connectivity, bandwidth usage, and the status of various network services.

2. Server Monitoring: It is widely used to monitor servers, including their hardware health, operating system performance, and running services. This includes checking CPU load, memory usage, disk space, and more.
3. Application Monitoring: Nagios can be configured to monitor the performance and availability of applications and services, such as web servers (Apache, Nginx), databases (MySQL, PostgreSQL), and email servers.
4. Performance Monitoring: By integrating with performance graphing tools, Nagios can track and visualize key performance metrics over time. This helps in capacity planning and identifying performance trends.
5. Security Monitoring: Nagios can monitor security-related aspects of your infrastructure, such as unauthorized access attempts or changes in critical system files, helping to ensure the integrity and security of your systems.

Why are we using Nagios?

Nagios is used primarily for monitoring IT infrastructure to ensure that systems and services are operating correctly. Here are several reasons why organizations choose Nagios:

1. Comprehensive Monitoring: Nagios provides detailed monitoring of a wide range of IT components, including servers, network devices, applications, and services. This comprehensive approach helps ensure that all critical aspects of your infrastructure are under watch.
2. Real-Time Alerts and Notifications: It offers real-time alerts and notifications for issues like downtime, performance degradation, or configuration changes. This allows IT teams to quickly respond to and address problems before they impact users.
3. Customizability: Nagios is highly configurable, allowing users to tailor the monitoring setup to their specific needs. You can define custom checks, thresholds, and notification rules, and create a monitoring solution that fits your environment precisely.
4. Extensive Plugin Ecosystem: With a large repository of plugins and add-ons, Nagios can be extended to monitor a wide variety of applications, services, and hardware. This flexibility helps in adapting the tool to diverse and evolving monitoring needs.
5. Scalability: Nagios can scale from monitoring a single server to a large, distributed network. Its architecture supports distributed and redundant monitoring setups, making it suitable for both small and large environments.

Nagios is used to ensure the reliability, performance, and security of IT systems, providing peace of mind and operational efficiency for organizations.

Conclusion : LO1 - To understand the fundamentals of cloud computing and be fully proficient with cloud based DevOps solution deployment options to meet your business needs.

Name : Saikarthik Iyer

Batch : T13

Roll no : 41

ASSIGNMENT 10

Aim : To implement lambda function

Theory :

AWS Lambda is a serverless compute service that enables you to execute code in response to events without the need to manage servers. This service is designed to be event-driven, with functions triggered by various events and services such as changes in Amazon S3, incoming HTTP requests via Amazon API Gateway, or events from AWS services like SNS or CloudWatch. Lambda functions are small, focused units of code that execute quickly and can be written in various programming languages. With a pay-as-you-go pricing model, you only pay for the compute time your functions consume. It integrates seamlessly with other AWS services, making it a versatile solution for building serverless applications and automating workflows.

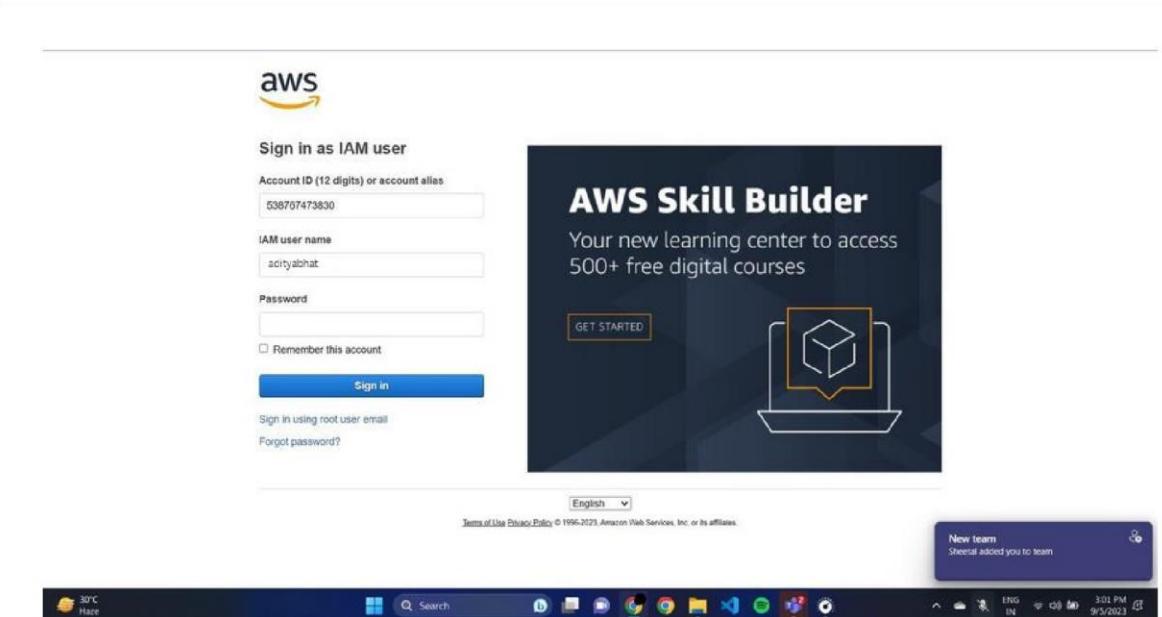
Advantages of Using S3 with Lambda:

Automation: The integration enables seamless automation of workflows without the need for human intervention.

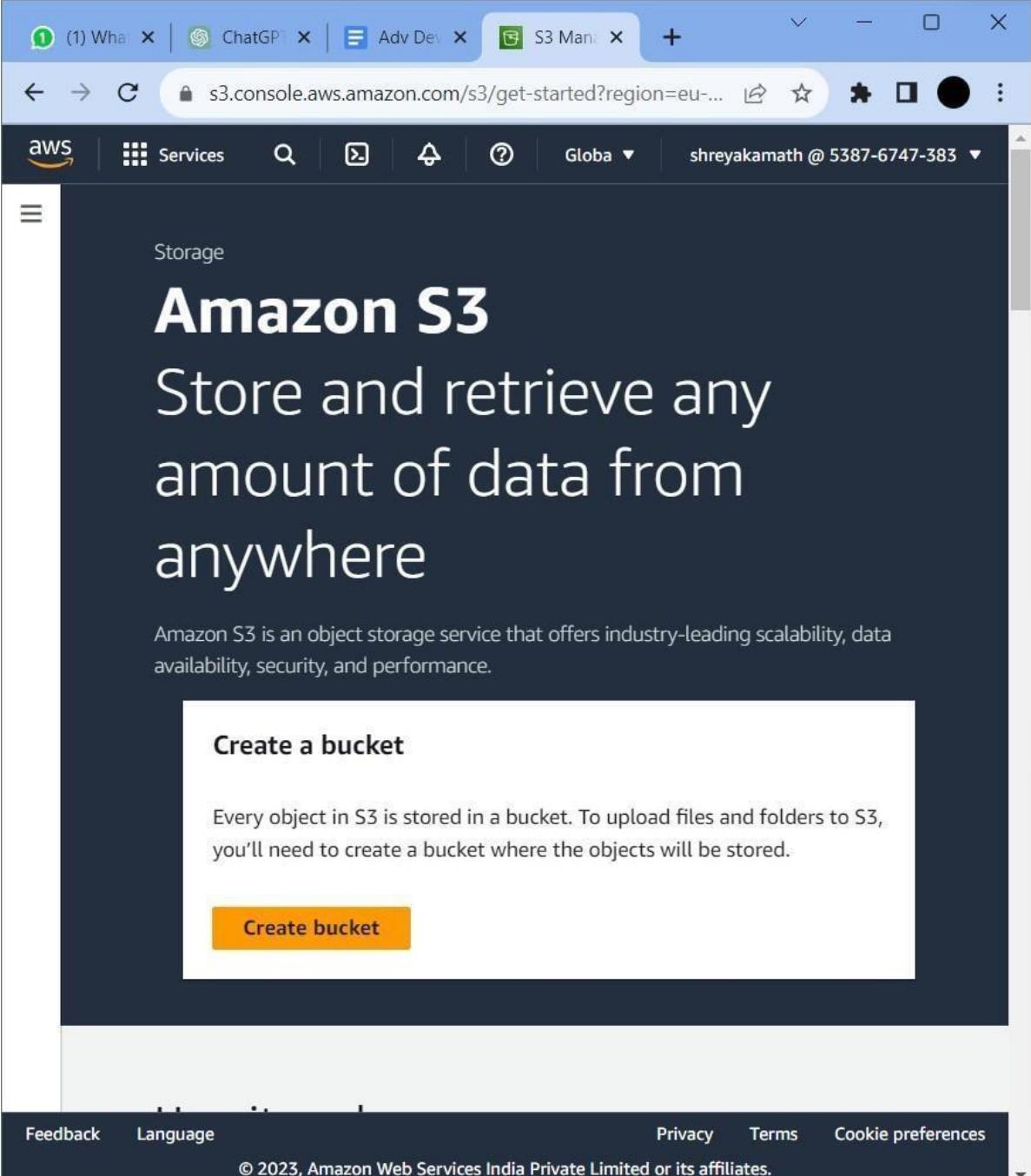
Scalability: Both S3 and Lambda scale automatically with the amount of data and number of requests.

Cost Efficiency: By using Lambda, you only pay for the compute time used, and by using S3, you only pay for the storage you consume.

Ease of Use: The combination of S3 and Lambda simplifies complex tasks like data processing, logging, and notifications.



The screenshot displays the AWS Console Home page. It includes a 'Recently visited' section with links to CodePipeline, CodeDeploy, CodeCommit, Cloud9, and EC2. A 'Welcome to AWS' section offers links to 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. Below these are sections for 'AWS Health' (showing 0 open issues) and 'Cost and usage'. The bottom navigation bar includes CloudShell, Feedback, Language, and links for privacy, terms, and cookie preferences. The system tray at the very bottom shows the date and time as 9/5/2023 3:01 PM.



The screenshot shows a web browser window with multiple tabs open at the top. The active tab is titled "S3 Man..." and displays the URL s3.console.aws.amazon.com/s3/get-started?region=eu-central-1. The browser interface includes standard controls like back, forward, and search, along with a user profile icon for "shreyakamath @ 5387-6747-383".

The main content area is titled "Amazon S3" in large, bold letters. Below the title, a large heading reads "Store and retrieve any amount of data from anywhere". A descriptive paragraph follows: "Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance." At the bottom of this section is a call-to-action button labeled "Create bucket".

At the very bottom of the page, there is a dark footer bar containing links for "Feedback", "Language", "Privacy", "Terms", and "Cookie preferences", along with a copyright notice: "© 2023, Amazon Web Services India Private Limited or its affiliates."

AWS Services Global shreyakamath @ 5387-6747-383

Amazon S3 > Buckets > Create bucket

Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.
[Choose bucket](#)

Object Ownership

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Feedback Language Privacy Terms Cookie preferences
© 2023, Amazon Web Services India Private Limited or its affiliates.

aws | Services | Search | Global | shreyakamath @ 5387-6747-383 ▾

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with

aws | Services | Search | [Alt+S] | Global | shreyakamath @ 5387-6747-383 ▾

Amazon S3 > Buckets > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name:
Bucket name must be unique within the global namespace and follow the bucket naming rules. See [rules for bucket naming](#)

AWS Region:

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

Object Ownership [Info](#)
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

ACLs disabled (recommended) ACLs enabled

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 30°C Haze 9/5/2023 3:10 PM

Successfully created bucket "webdatabucket53". To upload files and folders, or to configure additional bucket settings choose [View details](#).

Amazon S3 > Buckets

▼ Account snapshot

Total storage: Pending | Object count: Pending | Average object size: Pending | You can enable advanced metrics in the "default-account-dashboard" configuration.

Buckets (1) info

Buckets are containers for data stored in S3. [Learn more](#)

[Create bucket](#)

Name	AWS Region	Access	Creation date
webdatabucket53	Europe (Stockholm) eu-north-1	Bucket and objects not public	September 5, 2023, 15:10:35 (UTC+05:30)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 3:10 PM 9/5/2023 ENG IN

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 3:10 PM 9/5/2023 ENG IN

aws Services Search [Alt+S] shreyakamath @ 5387-6747-3830

Amazon S3 > Buckets > webdatabucket53 > Edit bucket policy

Edit bucket policy [Info](#)

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Bucket ARN: arn:aws:s3:::webdatabucket53

Policy

1 | Edit statement

Select a statement
Select an existing statement in the policy or add a new statement.

+ Add new statement

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 3:12 PM 9/5/2023 ENG IN

AWS Services Search [Alt+S] Global shreyakamath @ 5387-6747-3830 ⓘ

Amazon S3 > Buckets > webdatabasebucket53 > Edit static website hosting

Edit static website hosting [Info](#)

Static website hosting
Use this bucket to host a website or redirect requests. [Learn more](#) ⓘ

Static website hosting
 Disable
 Enable

Hosting type
 Host a static website
Use the bucket endpoint as the web address. [Learn more](#) ⓘ
 Redirect requests for an object
Redirect requests to another bucket or domain. [Learn more](#) ⓘ

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#) ⓘ

Index document
Specify the home or default page of the website.

Error document - optional
This is returned when an error occurs.

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 3:13 PM ENG IN 9/5/2023 ⓘ

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 3:13 PM ENG IN 9/5/2023 ⓘ

Successfully edited static website hosting. ⓘ

You can use bucket tags to track storage costs and organize buckets. [Learn more](#) ⓘ

Key	Value
No tags associated with this resource.	

Default encryption [Info](#)
Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)
Server-side encryption with Amazon S3 managed keys (SSE-S3)

Bucket Key
When KMS encryption is used to encrypt new objects in this bucket, the bucket key reduces encryption costs by lowering calls to AWS KMS. [Learn more](#) ⓘ
Enabled

Intelligent-Tiering Archive configurations (0)
Enable objects stored in the Intelligent-Tiering storage class to tier-down to the Archive Access tier or the Deep Archive Access tier which are optimized for objects that will be rarely accessed for long periods of time. [Learn more](#) ⓘ

[View details](#) [Edit](#) [Delete](#) [Create configuration](#)

Find Intelligent-Tiering Archive configurations

Name	Status	Scope	Days until transition to Archive Access tier	Days until transition to Deep Archive Access tier
© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 3:17 PM ENG IN 9/5/2023 ⓘ				

Screenshot of the AWS IAM console showing the "Select trusted entity" step. The "Trusted entity type" section is displayed, with the "AWS service" option selected. Other options include "AWS account", "Web identity", "SAML 2.0 federation", and "Custom trust policy". The "Use case" section shows "EC2" selected as a common use case. A dropdown menu for "Use cases for other AWS services" is open, showing "Choose a service to view use case".

Screenshot of the AWS IAM console showing the "Step 2: Add permissions" step. The "Permissions policy summary" table lists three managed policies: "AmazonS3FullAccess", "AmazonDynamoDBFullAccess", and "CloudWatchFullAccess", all attached as permissions policies. The "Tags" section shows no tags associated with the resource, with an "Add tag" button available.

aws Services Search [Alt+S] Stockholm shreyakamat @ 5387-6747-3830 ⓘ

DynamoDB > Tables > Create table

Create table

Table details ⓘ

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (_), hyphens (-), and periods (.)

Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.

st_id	Number
-------	--------

1 to 255 characters and case sensitive.

Sort key - optional

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.

Enter the sort key name	String
-------------------------	--------

1 to 255 characters and case sensitive.

Table settings

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 323 PM ENG IN 9/5/2023 ⓘ

Haze 31°C

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 323 PM ENG IN 9/5/2023 ⓘ

aws Services Search [Alt+S] Stockholm shreyakamat @ 5387-6747-3830 ⓘ

DynamoDB ⓘ The studentdata table was created successfully.

DynamoDB > Tables

Tables (1) Info

Name	Status	Partition key	Sort key	Indexes	Deletion protection	Read capacity mode	Write capacity mode
studentdata	Active	st_id (N)	-	0	Off	Provisioned with auto scaling (5)	Provisioned with auto scaling (5)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 325 PM ENG IN 9/5/2023 ⓘ

Haze 31°C

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 325 PM ENG IN 9/5/2023 ⓘ

aws Services Search [Alt+S] Stockholm shreyakamath @ 5387-6747-3830

DynamoDB > Explore items: studentdata > Create item

Create item

You can add, remove, or edit the attributes of an item. You can nest attributes inside other attributes up to 32 levels deep. Learn more [?](#)

[Form](#) [JSON view](#)

Attributes

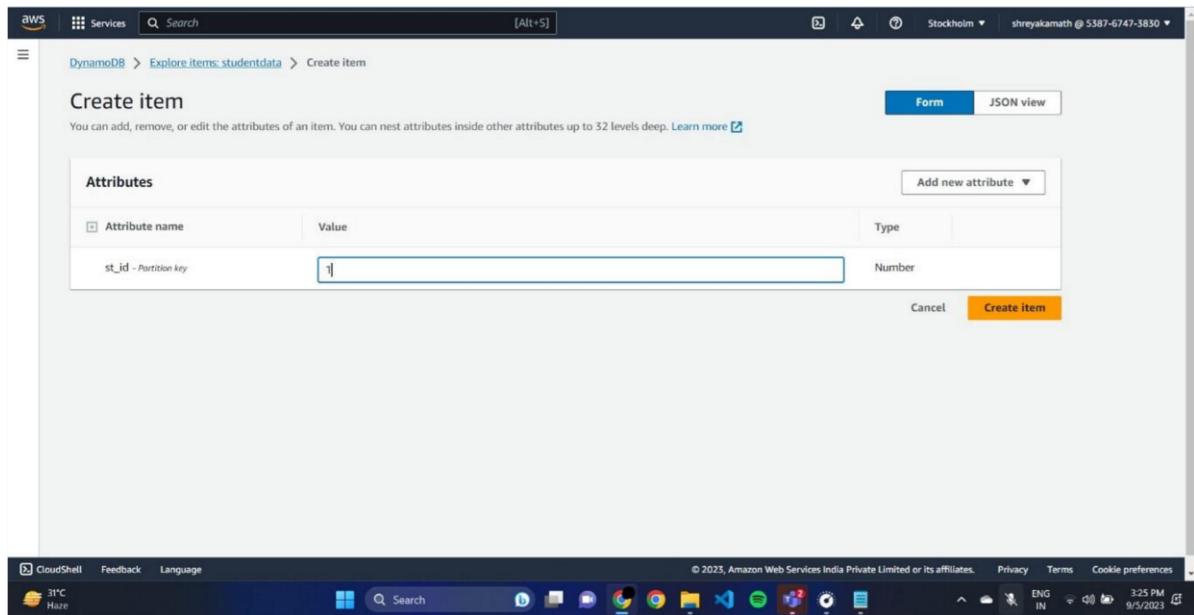
Attribute name	Value	Type
st_id - Partition key	1	Number

Add new attribute ▾

Cancel [Create item](#)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

31°C Haze 3:25 PM 9/3/2023



The screenshot shows the AWS Lambda landing page. At the top, there's a navigation bar with 'Services' and a search bar. The main heading is 'AWS Lambda' with the tagline 'lets you run code without thinking about servers.' Below this, a paragraph explains that users pay only for compute time consumed. To the right, there's a 'Get started' section with a 'Create a function' button. The bottom half features a 'How it works' section with tabs for '.NET', 'Go', 'Java', 'Node.js', 'Python', 'Ruby', and 'Custom runtime'. A code editor window shows a simple Node.js handler:

```
1 exports.handler = async (event) => {
2     console.log(event);
3     return 'Hello from Lambda!';
4 };
```

aws Services Search [Alt+S] Stockholm shreyakamat @ 5387-6747-3830 ⓘ

Basic information

Function name
Enter a name that describes the purpose of your function.
 Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
 ⚙️

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▶ Change default execution role

▶ Advanced settings

Cancel [Create function](#)

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Haze 3:41 PM 9/5/2023 aws Services Search [Alt+S] Stockholm shreyakamat @ 5387-6747-3830 ⓘ

Successfully created the function **webdatafunction**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Code source [Info](#)

File Edit Find View Go Tools Window Test Deploy Changes not deployed

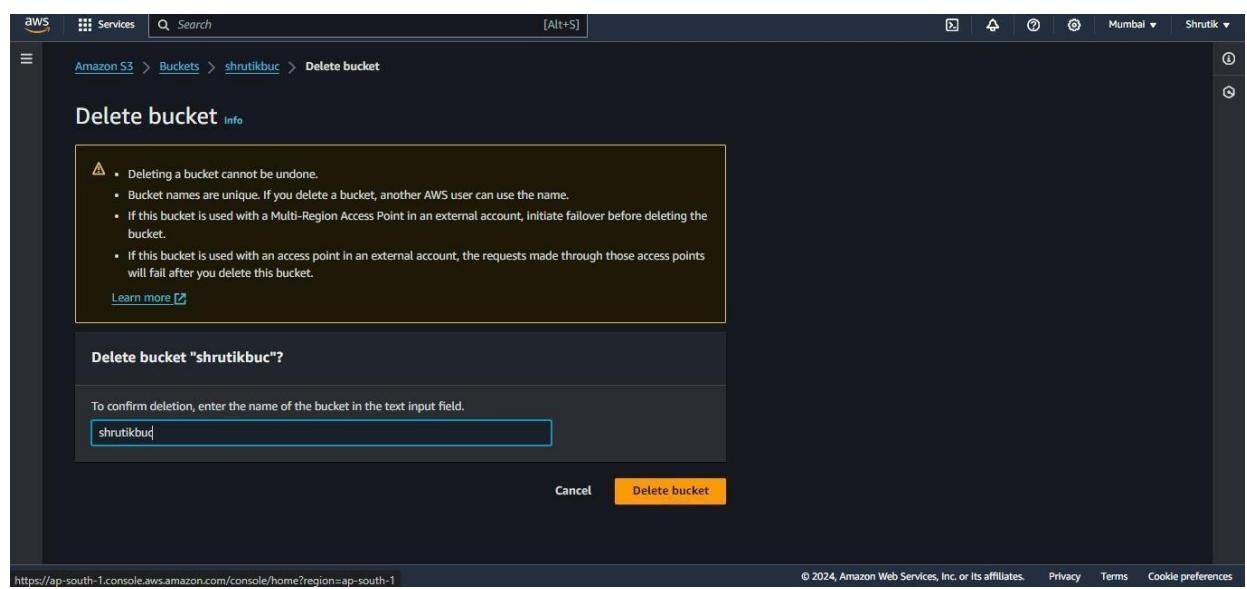
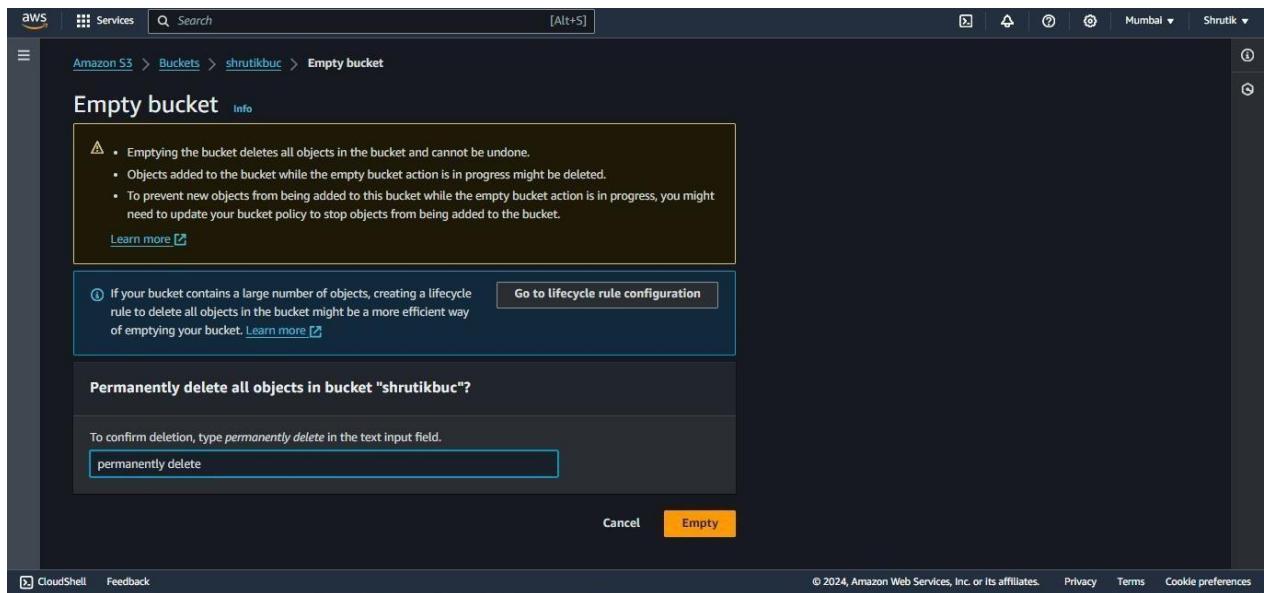
Go Anything (Ctrl+P)

lambda_function Environment Var

```
lambda_function
1 import json
2 import boto3
3
4 dynamo = boto3.client('dynamodb')
5
6 def lambda_handler(event, context):
7     sid = event['sid']
8     sname = event['sname']
9     marks = event['marks']
10    dynamo.put_item(TableName="studentdata"),
11        Item={
12            'st_id': {'S': sid},
13            'marks': {'S': marks},
14            'st_name': {'S': sname},
15        })
```

Upload from ⚙️

CloudShell Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences 31°C Haze 3:46 PM 9/5/2023 aws Services Search [Alt+S] Stockholm shreyakamat @ 5387-6747-3830 ⓘ



Conclusion : Hence, learned & implemented S3 buckets and lambda function

Name : Saikarthik Iyer

Roll no : 41

Batch : T13

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 security techniques that can be used to protect data?

Ans)

1. Encryption:
 - 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.
 - 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.
2. Access Control and Identity Management:
 - 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.
 - 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.
3. 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.

Name : Saikarthik Iyer

Roll no : 41

Batch : T13

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 AmazonAPI 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.

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 APIGateway 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.

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 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:

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

2. Lambda Function Code:

- This is the actual code or script that you want to execute in response to the event triggered by the event source.
- Lambda functions can be written in several programming languages, including Node.js, Python, Java, C#, Ruby, and more.
- 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.

3. Execution Role:

- 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.

- 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.
- 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.