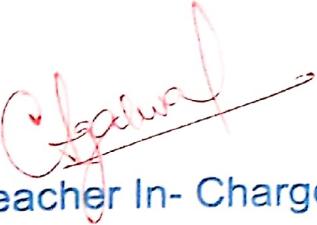


# Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

## ∞ CERTIFICATE ∞

Certify that Mr./Miss ANIMESH NARAYAN PARAB  
of I-T Department, Semester V with  
Roll No. 88 has completed a course of the necessary  
experiments in the subject Advance Devops Lab under my  
supervision in the **Thadomal Shahani Engineering College**  
Laboratory in the year 20 23 - 20 24

  
Teacher In-Charge

Head of the Department

Date 21/10/23

Principal

## CONTENTS

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

## **CONTENTS**

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
10	To understand AWS Lambda it's works flow various functions and create function	02-9-23		<i>Chatur 10/23</i>
11	Assignment 1	26-7-23		<i>Chatur 10/23</i>
12	Assignment 2	11-12-23		

## **Assignment 1**

**Aim:-** Study and create an AWS EC2 instance.

**LoP mapped:-**

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

**Theory:-**

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable computing capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Here's a detailed explanation of an AWS EC2 instance:

1. Virtual Server:

An EC2 instance is a virtualized computing environment that mimics a physical server. It runs an operating system of your choice and supports a wide range of applications, databases, and services.

2. Scalability:

One of the key benefits of EC2 is its scalability. You can easily scale your instances horizontally (adding more instances) or vertically (resizing an instance) to meet changing workloads. This elasticity helps you optimize performance while controlling costs.

3. Variety of Instance Types:

EC2 provides a variety of instance types optimized for different use cases. These instance types vary in terms of CPU, memory, storage, and network capabilities, allowing you to choose the best fit for your application's requirements.

4. Amazon Machine Image (AMI):

An AMI is a pre-configured template that contains the information required to launch an instance. It includes the operating system, application software, and any additional configuration you've applied. You can choose from a wide range of public AMIs or create your own custom AMIs.

5. Networking and Security:

Each EC2 instance is associated with a security group, which acts as a virtual firewall controlling inbound and outbound traffic. You can also assign Elastic IP addresses for consistent IP assignments, create Virtual Private Clouds (VPCs) for isolated networking, and configure network settings like subnets, routing tables, and network access control lists.

6. Cost Flexibility:

EC2 instances are available in different pricing models, including On-Demand (pay-as-you-go), Reserved Instances (upfront payment for long-term usage), and Spot Instances (bid-based pricing). This allows you to choose the most cost-effective option for your workload.

**Output:-**

LOGIN TO AWS ACCOUNT,

THEN SEARCH EC2.

The screenshot shows the AWS EC2 Dashboard for the US East (N. Virginia) Region. The left sidebar includes links for EC2 Global View, Events, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Capacity Reservations), and Images (with sub-links for AMIs and AMI Catalog). A 'New EC2 Experience' feedback link is also present. The main content area has three main sections: 'Resources' (listing 0 instances running, 0 dedicated hosts, 0 instances, 0 load balancers, 2 security groups, and 0 volumes), 'Launch instance' (with a 'Launch instance' button), and 'Service health' (with a 'AWS Health Dashboard' link). On the right, there's a 'Account attributes' section showing the Default VPC (vpc-03d739f7fb1496d54) and a 'Settings' section with links for Data protection and security, Zones, EC2 Serial Console, Default credit specification, and Console experiments. An 'Explore AWS' box highlights Graviton-based instances for better price performance compared to x86 instances.

Resources

Instances (running)	0	Auto Scaling Groups	0
Dedicated Hosts	0	Elastic IPs	0
Instances	0	Key pairs	0
Load balancers	0	Placement groups	0
Security groups	2	Snapshots	0
Volumes	0		

Launch instance

Service health

Account attributes

Default VPC vpc-03d739f7fb1496d54

Settings

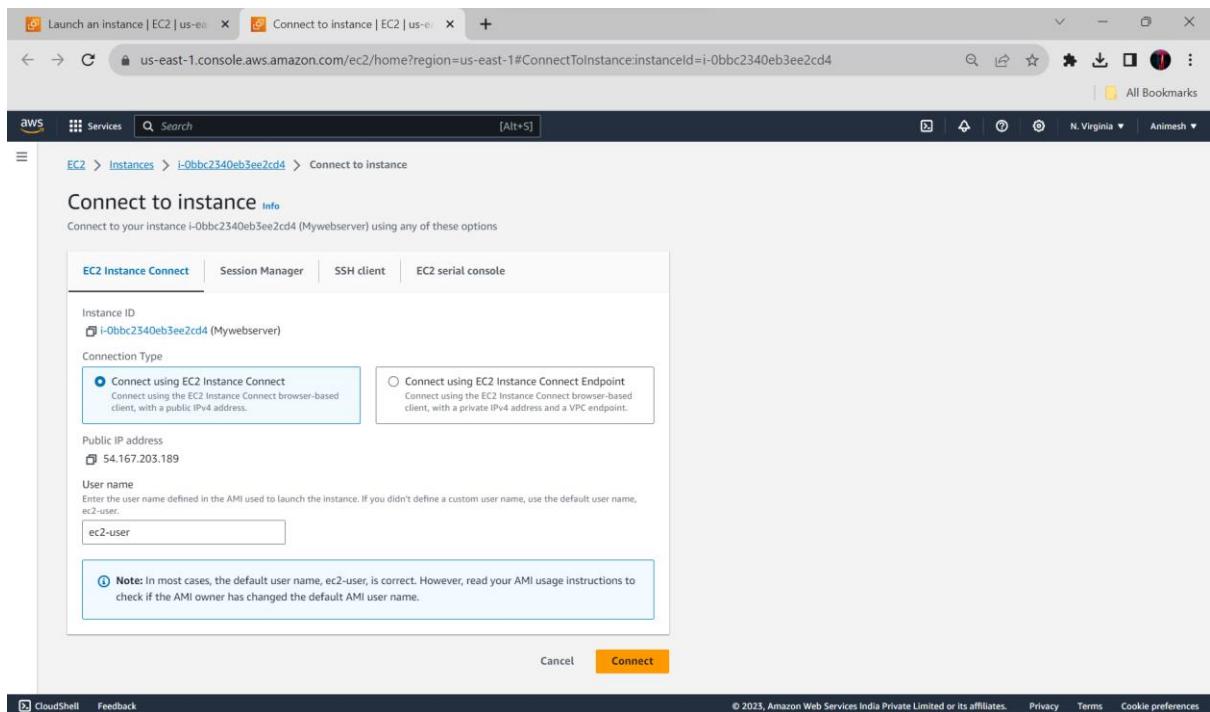
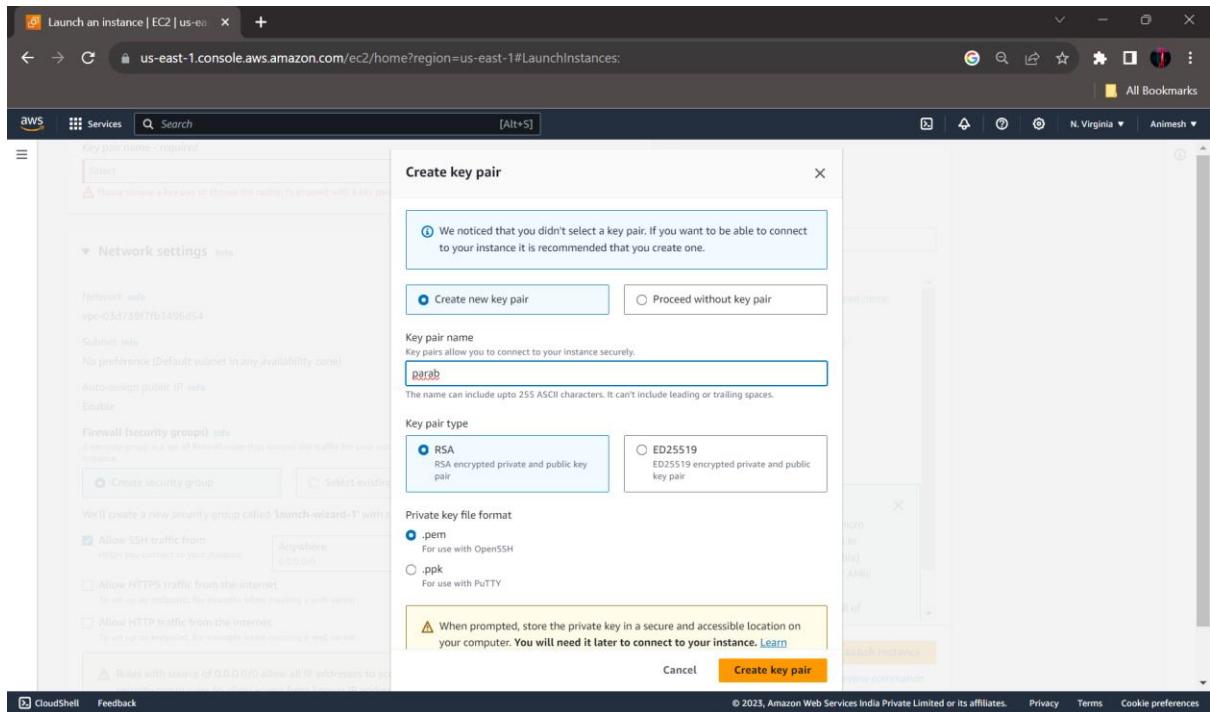
Explore AWS

NOW CLICK ON LAUNCH / CREATE NEW INSTANCES.

The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The current step is 'Name and tags'. In the 'Name' field, 'Mywebserver' is entered. Below the form, there's a section titled 'Application and OS Images (Amazon Machine Image)'. A note says, 'An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.' A search bar is present with the placeholder 'Search our full catalog including 1000s of application and OS images'. On the right side, the 'Summary' panel shows the configuration: 1 instance, Amazon Linux 2023 AMI 2023.2.2..., t2.micro instance type, New security group, and 1 volume(s) - 8 GiB storage. At the bottom right is a large orange 'Launch instance' button.

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

The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The current step is 'Application and OS Images (Amazon Machine Image)'. A note says, 'An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below.' A search bar is present with the placeholder 'Search our full catalog including 1000s of application and OS images'. Below the search bar, there's a 'Quick Start' section with icons for various operating systems: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE, and a link to 'Browse more AMIs'. A callout box highlights the 'Free tier' information: 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million I/Os, 1 GB of snapshots, and 100 GiB of bandwidth to the internet.' At the bottom left, a table shows the selected AMI details: Ubuntu Server 22.04 LTS (HVM), SSD Volume Type, ami-053b0d5c279eac90 (64-bit Arm) / ami-053b0d5c279eac90 (64-bit Arm). At the bottom right is a large orange 'Launch instance' button.



**Instances (1/1) Info**

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
mywebserver	i-0189ca15b75b55fc	Running	t2.micro	Initializing	No alarms	us-east-1b	ec2-54-174-70-237.co...	54.174.70.237	-

**Instance: i-0189ca15b75b55fc (mywebserver)**

**Details** | Security | Networking | Storage | Status checks | Monitoring | Tags

**Instance summary**

Instance ID	i-0189ca15b75b55fc (mywebserver)	Public IPv4 address	54.174.70.237 [open address]
IPv6 address	-	Private IPv4 addresses	172.31.33.254
Hostname type	IP name: ip-172-31-33-254.ec2.internal	Public IPv4 DNS	ec2-54-174-70-237.compute-1.amazonaws.com [open address]
Answer private resource DNS name	ipv4 (A)	Instance type	t2.micro
Auto-assigned IP address	54.174.70.237 [Public IP]	VPC ID	vpc-03d739f7fb1496d54

Waiting for us-east-1.console.aws.amazon.com...

```
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

 System information as of Sat Oct 7 17:41:14 UTC 2023

 system load: 0.84423829125  Processes: 99
 Usage: 20.6% of 7.57GB  Users logged in: 0
 Memory usage: 4%  IPv4 address for eth0: 172.31.33.254
 Swap usage: 0%  Swap usage: 0

 Expanded Security Maintenance for Applications is not enabled.

 0 updates can be applied immediately.

 Enable ESM Apps to receive additional future security updates.
 See https://ubuntu.com/esm or run: sudo pro status

 The list of available updates is more than a week old.
 To check for new updates run: sudo apt update

 The programs included with the Ubuntu system are free software;
 the exact distribution terms for each program are described in the
 individual files in /usr/share/doc//copyright.

 Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
 applicable law.

 To run a command as administrator (user "root"), use "sudo <command>".
 See "man sudo_root" for details.

ubuntu@ip-172-31-33-254:~$
```

i-0189ca15b75b55fc (mywebserver)  
PublicIPs: 54.174.70.237 PrivateIPs: 172.31.33.254

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

## ASSIGNMENT 2 CLOUD9

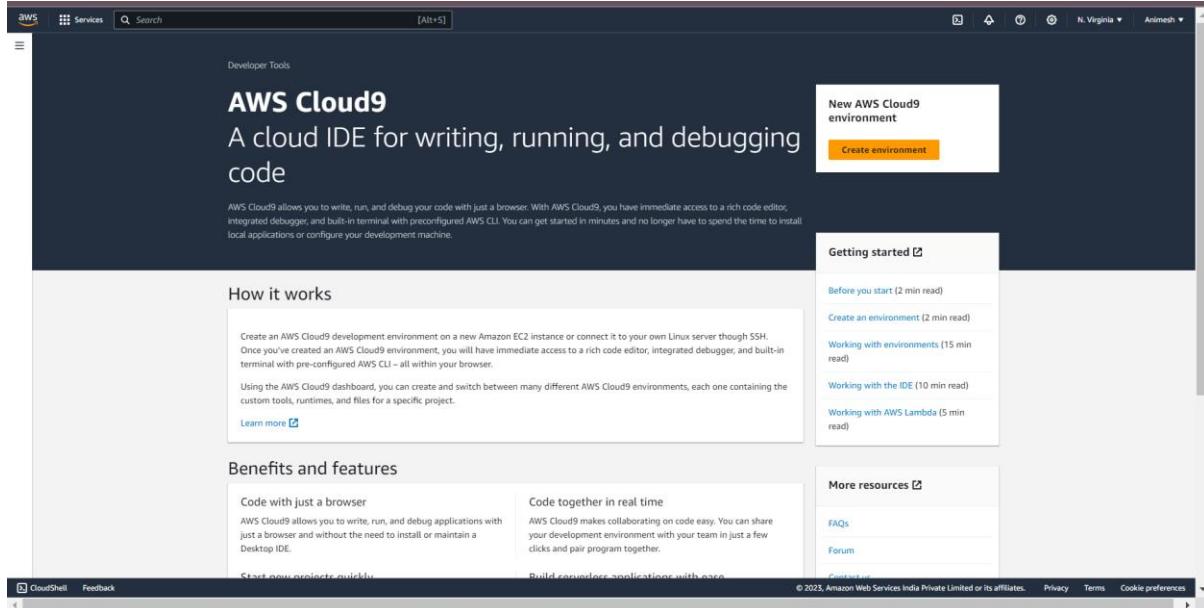
**AIM-** To create aws Cloud9 IDE account and execute python application development.

**Theory-**Cloud9 IDE is an Online IDE, published as open source from version 2.0, until version 3.0. It supports multiple programming languages, including C, C++, PHP, Ruby, Perl, Python, JavaScript with Node.js, and Go. It is written almost entirely in JavaScript, and uses Node.js on the back-end.

### **STEPS-**

LOG IN TO YOUR AWS ACCOUNT,

SEARCH FOR CLOUD 9 IN THE SEARCH BAR



CLICK ON CREATE ENVIRONMENT,

NAME THE ENVIRONMENT

**Create environment** [Info](#)

### Details

Name  Limit of 60 characters, alphanumeric, and unique per user.

Description - optional  Limit 200 characters.

**Environment type** [Info](#)  
Determines what the Cloud9 IDE will run on.

New EC2 instance  
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.

Existing compute  
You have an existing instance or server that you'd like to use.

### New EC2 instance

Instance type [Info](#)  
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.

t2.micro (1 GiB RAM + 1 vCPU)  
Free-tier eligible. Ideal for educational users and exploration.

t3.small (2 GiB RAM + 2 vCPU)  
Recommended for small web projects.

m5.large (8 GiB RAM + 2 vCPU)  
Recommended for production and most general-purpose development.

Additional instance types  
Explore additional instances to fit your need.

[CloudShell](#) [Feedback](#) © 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

**Platform** [Info](#)  
This will be installed on your EC2 instance. We recommend Amazon Linux 2.

Amazon Linux 2

Timeout  
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.  
30 minutes

### Network settings

Connection  
How your environment is accessed.

AWS Systems Manager (SSM)  
Accesses environment via SSM without opening inbound ports (no ingress).

Secure Shell (SSH)  
Accesses environment directly via SSH, opens inbound ports.

[VPC settings](#) [Info](#)

**Tags - optional** [Info](#)  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

**I** The following IAM resources will be created in your account

- [AWSServiceRoleForAWSCloud9](#) - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. [Learn more](#)
- [AWSCloud9SSMAccessRole](#) and [AWSCloud9SSMInstanceProfile](#) - A service role and an instance profile are automatically created if Cloud9 accesses its EC2 instance through AWS Systems Manager. If your environments no longer require EC2 instances that block incoming traffic, you can delete these roles using the AWS IAM console. [Learn more](#)

[Cancel](#) [Create](#)

[CloudShell](#) [Feedback](#) © 2023, Amazon Web Services India Private Limited or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

The screenshot shows the AWS Cloud9 interface. On the left, there's a sidebar with options like 'My environments', 'Shared with me', and 'All account environments'. The main area is titled 'Environments (1)' and lists a single environment named 'Animesh Parab'. The environment details are as follows:

Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
Animesh Parab	Open	EC2 instance	AWS Systems Manager (SSM)	Owner	arn:aws:iam::368680180531:root

At the bottom of the page, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services India Private Limited or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

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

The screenshot shows the AWS Cloud9 IDE interface. The top navigation bar includes 'File', 'Edit', 'Find', 'View', 'Go', 'Run', 'Tools', 'Window', 'Support', and 'Preview'. The main area has tabs for 'Welcome' and 'Developer Tools'. The 'Welcome' tab displays the 'AWS Cloud9' logo and the message 'Welcome to your development environment'. Below this, it says: 'AWS Cloud9 allows you to write, run, and debug your code with just a browser. You can [view the IDE](#), write code for AWS Lambda and Amazon API Gateway, share your IDE with others in real time, and much more.' The 'Getting started' section contains buttons for 'Create File', 'Upload Files...', and 'Clone from GitHub'. The 'Configure AWS Cloud9' section allows users to set the Main Theme (jett-dark), Editor Theme (Jett), and Keyboard Mode (Default). The bottom part of the interface shows a terminal window with the command 'bash -lp-172-31-40-96 x' and the prompt 'ec2-user:~/environment \$'. The status bar at the bottom indicates 'CodeMirror' and 'AWS profile:default'.

A screenshot of the Cloud9 IDE interface. The top navigation bar includes File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, and Run. The preview tab is active. The left sidebar shows a file tree for 'Animesh Parab -' containing p1.cpp, p2.c, p2.o, and README.md. The main workspace has three tabs: 'Welcome' (active), 'p1.cpp', and 'p2.c'. The 'p2.c' tab contains the following C code:

```
#include <stdio.h>
int main()
{
    // prints "Hello, world!" displays the string inside quotation
    printf("Hello, world!");
    return 0;
}
```

The bottom terminal window shows the output of running the program:

```
bash -tp-172-31-40-96.e ✘ Immediate ✘ p2.c - Stopped ✘ p2.c
Running /home/ec2-user/environment/p2.c
Hello, world!
Process exited with code 0
```

A screenshot of the Cloud9 IDE interface. The top navigation bar includes File, Edit, Find, View, Go, Run, Tools, Window, Support, Preview, and Run. The preview tab is active. The left sidebar shows a file tree for 'Animesh Parab -' containing Main.java, p1.cpp, p2.c, p2.o, and README.md. The main workspace has four tabs: 'Welcome' (active), 'p1.cpp', 'p2.c', and 'Main.java'. The 'Main.java' tab contains the following Java code:

```
public class Main {
    public static void main(String[] args) {
        int num = 29;
        boolean flag = false;
        for (int i = 2; i <= num / 2; ++i) {
            if (num % i == 0) {
                flag = true;
                break;
            }
        }
        if (!flag)
            System.out.println(num + " is a prime number.");
        else
            System.out.println(num + " is not a prime number.");
    }
}
```

The bottom terminal window shows the output of running the Java program:

```
bash -tp-172-31-40-96.e ✘ Immediate ✘ p2.c - Stopped ✘ p3.java - Stopped ✘ Main.java - Stopped ✘
Building Main.java and running Main
29 is a prime number.
Process exited with code 0
```

**Conclusion –** Hence learned and implemented steps to Create an Cloud9 environment

## Animesh Parab T2-T21 88

### ASSIGNMENT-3

**AIM-** To study AWS S3 service and create a bucket for housing static web application.

#### **THEORY-**

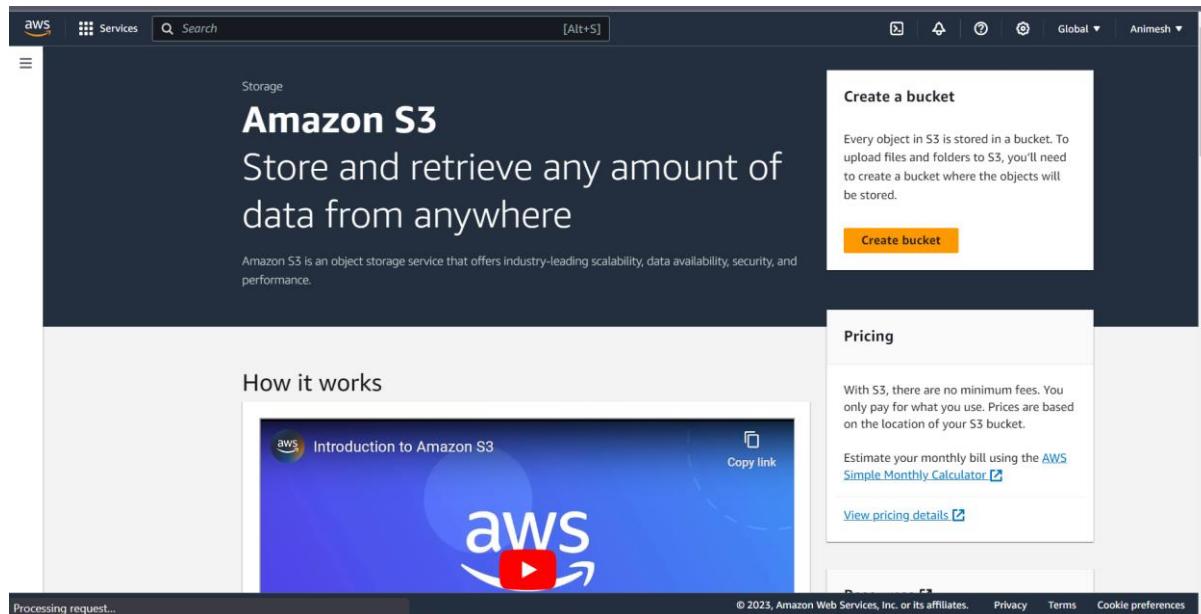
AWS Simple Storage Service (S3) from the aforementioned list, S3, is the object storage service provided by AWS. It is probably the most commonly used, go-to storage service for AWS users given the features like extremely high availability, security, and simple connection to other AWS Services.

An Amazon S3 bucket can be set up to operate similarly to a website. This section illustrates how to host a website using Amazon S3. There are mainly 7 steps to hosting a static website using Amazon Web Service(AWS) S3.

#### **STEPS:**

##### **Step 1: Creating a Bucket**

1. First, we have to launch our S3 instance. Follow these steps for creating a Bucket
2. Open the Amazon S3 console by logging into the AWS Management Console at



##### **Step 2: Block Public Access settings for the bucket**

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

The screenshot shows the 'Create bucket' page in the AWS S3 console. In the 'General configuration' section, the 'Bucket name' is set to 'myanimesh'. The 'AWS Region' is set to 'US East (N. Virginia) us-east-1'. Under 'Object Ownership', it says '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.' There are two options: 'ACLs disabled (recommended)' and 'ACLs enabled'. 'ACLs enabled' is selected, and a note says: 'We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.' At the bottom, 'Bucket owner preferred' is selected under 'Object Ownership'.

#### 4. Choose Bucket Name – Bucket Name Should be Unique

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

This screenshot shows the 'Object Ownership' settings for a bucket. It highlights the 'ACLs enabled' option, which is recommended for public access. A note below it states: 'We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.' Below this, the 'Bucket owner preferred' setting is selected, and a note says: 'If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.' The 'Object writer' setting is also shown.

#### Step 2: Block Public Access settings for the bucket

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

**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 policies that grant public access to buckets and objects.

**⚠️ Turning off block all public access might result in this bucket and the objects within becoming public**  
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

## 2. Now click on create bucket

Select Bucket and Click your Bucket Name.

3. Bucket is created

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

[View Storage Lens dashboard](#)

Buckets (1) <a href="#">Info</a>			
Buckets are containers for data stored in S3. <a href="#">Learn more</a>			
Name	AWS Region	Access	Creation date
myanimesh	US East (N. Virginia) us-east-1	Objects can be public	October 20, 2023, 22:55:36 (UTC+05:30)

## Step 3: Now upload code files

Select Bucket and Click your Bucket Name.

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

The screenshot shows the AWS S3 console with a green header bar indicating "Upload succeeded". Below it, a summary table shows one file uploaded successfully to the destination "s3://myanimesh". The "Files and folders" tab is selected, displaying a single item: "Files and folders (1 Total, 1.7 KB)".

Destination	Succeeded	Failed
s3://myanimesh	1 file, 1.7 KB (100.00%)	0 files, 0 B (0%)

**Summary**

**Files and folders** Configuration

Find by name < 1 >

Name Folder Type Size Status Error

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

- Block public access
- Object Ownership
- Make public Object

The screenshot shows the "Block public access (bucket settings)" page. The "Block all public access" checkbox is checked. Below it, four sub-options are listed, each with a corresponding checkbox:

- Block public access to buckets and objects granted through new access control lists (ACLs)
- Block public access to buckets and objects granted through any access control lists (ACLs)
- Block public access to buckets and objects granted through new public bucket or access point policies
- Block public and cross-account access to buckets and objects through any public bucket or access point policies

At the bottom, there are "Cancel" and "Save changes" buttons.

**Block public access (bucket settings)**

Block all public access

Block public access to buckets and objects granted through new access control lists (ACLs)

Block public access to buckets and objects granted through any access control lists (ACLs)

Block public access to buckets and objects granted through new public bucket or access point policies

Block public and cross-account access to buckets and objects through any public bucket or access point policies

Cancel Save changes

© 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various global settings. Below it, the path 'Amazon S3 > Buckets > myanimmesh' is shown. The main area is titled 'myanimmesh' with a 'Info' link. A horizontal menu bar includes 'Objects' (which is selected), 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. Under the 'Objects' section, a sub-section titled 'Objects (1)' is displayed. It contains a message about objects being fundamental entities stored in S3, a 'Learn more' link, and a toolbar with buttons for 'Copy', 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and 'Upload' (which is highlighted). Below the toolbar is a search bar with the placeholder 'Find objects by prefix' and a 'Show versions' toggle. A table lists the single object: 'home.html' (Type: html, Last modified: October 20, 2023, 22:56:30 (UTC+05:30), Size: 1.7 KB, Storage class: Standard). The table has columns for Name, Type, Last modified, Size, and Storage class. At the bottom of the page, there are links for 'CloudShell', 'Feedback', and copyright information: '© 2023, Amazon Web Services, Inc. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

## Step 5: Copy your Object URL

Now, click on your HTML File Object Name.

Copy the Object URL.

## Step 6

**Check out your Website!**

### Profile

- [Home](#)
- [Skills](#)
- [Contact](#)

Hello,

**Hi Animesh Parab**

Software engineer



### About me

Animesh is a pre final year student at Thadomal Shahani engineering college.

He has worked with HSBC as an analyst, where he played the role of a data engineer.

He worked on software like data visualization.

He holds a degree in Information Technology.

Copyright © Designed by Animesh

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

# **Animesh Parab T2-T21 88**

## **ASSIGNMENT-4**

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

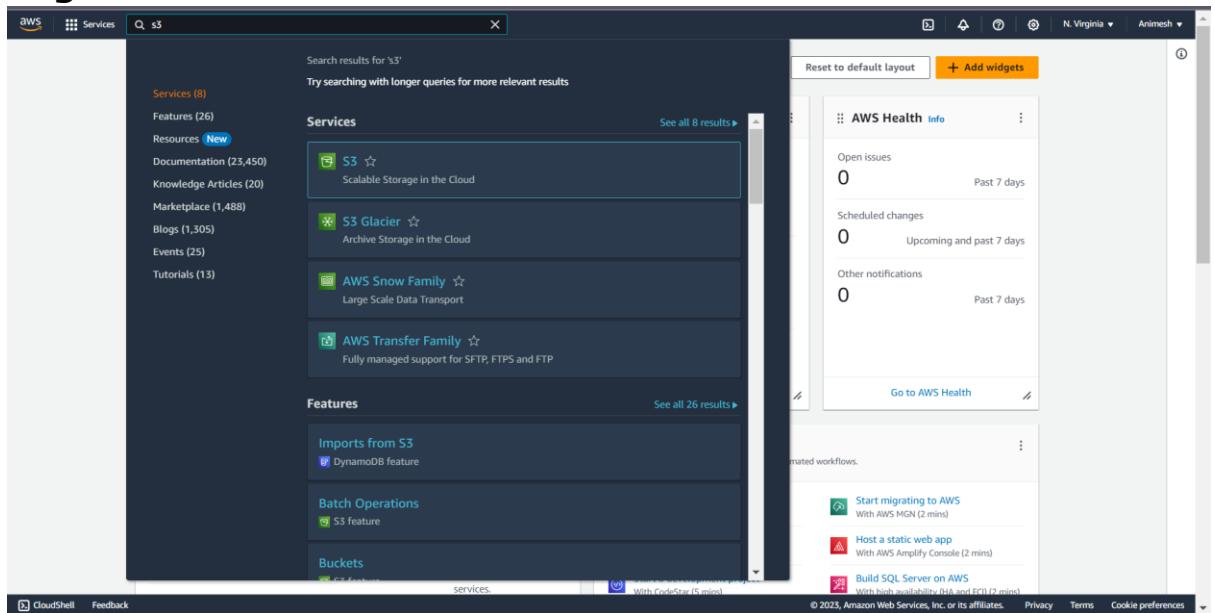
**LO MAPPED: LO1 , LO2**

### **THEORY:**

Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

### **STEPS:**

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



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

The screenshot shows the Amazon S3 landing page. At the top right, there is a call-to-action box titled "Create a bucket". It contains text explaining that every object in S3 is stored in a bucket and needs one to upload files and folders. Below this is a large orange "Create bucket" button. To the left of the main content area, there is a section titled "How it works" featuring a video thumbnail titled "Introduction to Amazon S3".

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

The screenshot shows the "Create bucket" configuration page. The "General configuration" section is visible, where the "Bucket name" field is filled with "myparab". The "AWS Region" dropdown is set to "US East (N. Virginia) us-east-1". Below this, there is a "Copy settings from existing bucket - optional" section with a "Choose bucket" button. The "Object Ownership" section is also partially visible at the bottom.

The screenshot shows the AWS S3 Buckets page. At the top, a green banner indicates that a bucket named "myparab" has been successfully created. Below the banner, the "Account snapshot" section is visible, followed by the "Buckets (1) info" section. A table lists the single bucket "myparab" with details such as AWS Region (US East (N. Virginia) us-east-1), Access (Bucket and objects not public), and Creation date (October 21, 2023, 08:10:05 (UTC+05:30)).

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

The screenshot shows the AWS S3 Upload status page. A green banner at the top indicates an upload was successful. The main area displays a summary of the upload: Destination (s3://myparab), Status (Succeeded, 21 files, 150.3 KB (100.00%)), and Failed (0 files, 0 B (0%)). Below this, a table titled "Files and folders (21 Total, 150.3 KB)" lists the uploaded files and folders. The table includes columns for Name, Folder, Type, Size, Status, Error, and a search bar.

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

The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, Services, a search bar, and global settings. Below the navigation, the breadcrumb path indicates the location: Amazon S3 > Buckets > myparab > IP-main/ > index.html. On the right side, there are buttons for 'Copy S3 URI', 'Download', 'Open', and 'Object actions'. The main content area is titled 'index.html' with an 'Info' link. Below this, there are tabs for 'Properties', 'Permissions', and 'Versions'. The 'Properties' tab is selected, displaying detailed information about the object:

Object overview	
Owner	192109animeshparab
AWS Region	US East (N. Virginia) us-east-1
Last modified	October 21, 2023, 08:11:48 (UTC+05:30)
Size	5.4 KB
Type	html
Key	IP-main/index.html
S3 URI	s3://myparab/IP-main/index.html
Amazon Resource Name (ARN)	arn:aws:s3:::myparab/IP-main/index.html
Entity tag (Etag)	58ca75083bfb582eb3db063a45b1cd5
Object URL	<a href="https://myparab.s3.amazonaws.com/IP-main/index.html">https://myparab.s3.amazonaws.com/IP-main/index.html</a>

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

The screenshot shows the AWS S3 console interface. The top navigation bar includes the AWS logo, Services, a search bar, and global settings. Below the navigation, the breadcrumb path indicates the location: Amazon S3 > Buckets > myparab. On the right side, there are buttons for 'CloudShell' and 'Feedback'. The main content area is titled 'myparab' with an 'Info' link. Below this, there are tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Permissions' tab is selected, displaying the 'Permissions overview' section:

Permissions overview	
Access	Bucket and objects not public

Below this, there is a section titled 'Block public access (bucket settings)'. It contains a note about public access being granted through ACLs, policies, and access point policies. It also notes that turning on 'Block all public access' will block public access to all buckets and objects. A 'Learn more' link is provided. There is an 'Edit' button. Under 'Block all public access', there is a radio button labeled 'On' which is selected, and a link to 'Individual Block Public Access settings for this bucket'.

## 6. Change the public access permission.

The screenshot shows the 'Edit Block public access (bucket settings)' page for the 'mypyab' bucket. It includes a detailed description of what each setting does and how they interact. The 'Block all public access' checkbox is checked, and the 'Block public access to buckets and objects granted through new access control lists (ACLs)' checkbox is also checked. Other options like 'any access control lists', 'new public bucket or access point policies', and 'cross-account access' are listed but not selected. A note at the bottom states that turning on 'Block all public access' is equivalent to turning on all four settings.

The screenshot shows the 'Permissions' tab for the 'mypyab' bucket. A green success message at the top indicates that the Block Public Access settings were successfully edited. Below it, the 'Permissions overview' section shows that 'Bucket and objects not public'. The 'Block public access (bucket settings)' section shows that 'Block all public access' is turned off. The AWS navigation bar at the bottom includes CloudShell, Feedback, and links to Privacy, Terms, and Cookie preferences.

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

**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.

**ACLs disabled (recommended)**  
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

**ACLs enabled**  
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

**⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.**

**⚠ Enabling ACLs turns off the bucket owner enforced setting for Object Ownership**  
Once the bucket owner enforced setting is turned off, access control lists (ACLs) and their associated permissions are restored. Access to objects that you do not own will be based on ACLs and not the bucket policy.

I acknowledge that ACLs will be restored.

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

**Specified objects**

Find objects by name

Name	Type	Last modified	Size
IP-main/	Folder	-	-

Cancel **Make public**

aws Services Search [Alt+S] Global ▾ Animesh ▾

Successfully edited public access  
View details below.

### Make public: status

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

#### Summary

Source	Successfully edited public access	Failed to edit public access
s3://myparab	21 objects, 150.3 KB	0 objects

[Failed to edit public access](#) Configuration

Failed to edit public access (0)

Name	Folder	Type	Last modified	Size	Error

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Porfile Home Skills Contact Us

Hello,  
**Hii Animesh Parab**  
Software Developer

### About me

Animesh is a pre final year student at Thadomal shahani engineering college. He has worked with HSBC as an analyst, where he played the role of a data engineer. He worked on a software like data visualization. He holds a degree in Information Technology.

### Skills

## Contact Me

+91 9326211146  
animesh.nparab@gmail.com

First Name

Last Name

Email

feedback

Copyright © Designed by Animesh Parab

<https://myparab.s3.amazonaws.com/lP-main/contact.html>

SUCCESSFULLY EMPTIED BUCKET "myparab"

View details below. If you want to delete this bucket, use the [delete bucket configuration](#).

### Empty bucket: status

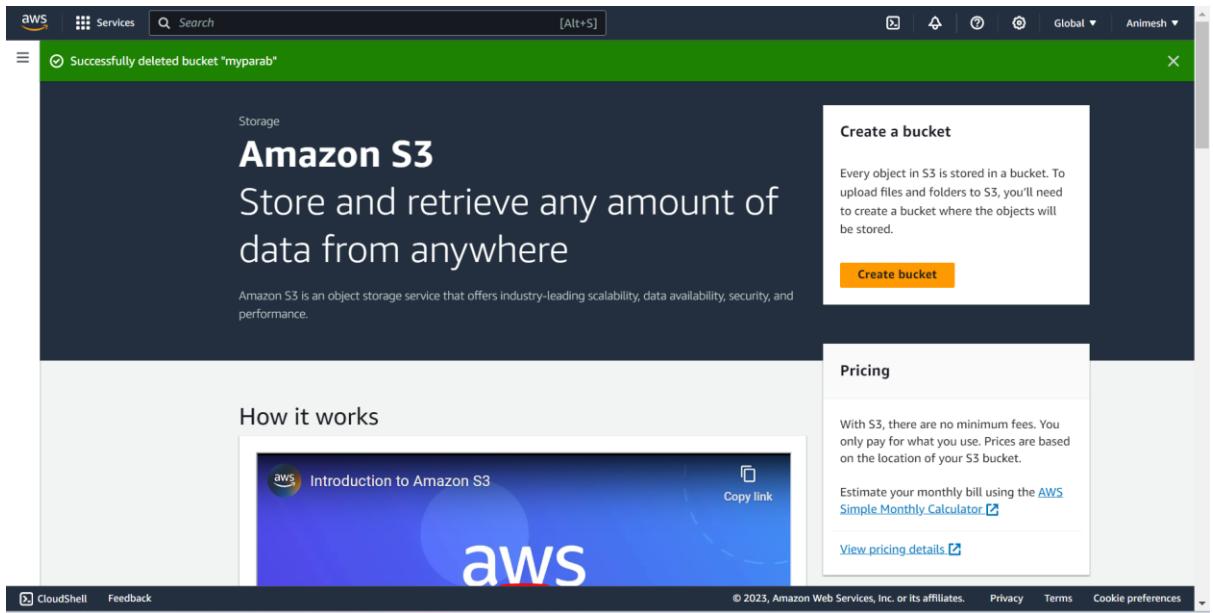
The details below are no longer available after you navigate away from this page.

Summary		
Source	Successfully deleted	Failed to delete
s3://myparab	21 objects, 150.3 KB	0 objects

### Failed to delete (0)

Name	Prefix	Version ID	Type	Last modified	Size	Error
No failed object deletions						

CloudShell Feedback © 2023, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



## CONCLUSION:-

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

## **ASSIGNMENT-5**

**AIM:** To understand the Kubernetes Cluster Architecture.

**LO MAPPED:** LO1 , LO3

### **THEORY:**

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

In a Kubernetes cluster, there are several essential services that run on nodes. These services are critical for the proper functioning of the cluster. Below, I'll describe the role of each service in detail:

#### **kubelet:**

The kubelet is responsible for managing containers on a node. It ensures that the containers in a Pod are running and healthy. It communicates with the control plane to receive Pod specifications and takes actions to make sure the containers match the desired state. For example, if a Pod specification indicates that it should run three containers, the kubelet ensures that those containers are up and running. If a container fails, the kubelet restarts it.

**Example:** Let's say you have a Pod with three containers, and one of them crashes due to a software issue. The kubelet will detect the failure and restart the failed container to maintain the desired state.

#### **kube-proxy:**

Kube-proxy is responsible for managing network connectivity to and from Pods. It maintains network rules on the host to enable communication between Pods and external networks. It sets up routes, handles load balancing, and ensures that network traffic is properly directed to the correct Pod. **Example:** If you have a service in your cluster that needs to load balance traffic to a set of Pods, kube-proxy manages this load balancing by configuring network rules and routes, directing traffic to the appropriate Pods.

#### **Container Runtime:**

The container runtime is responsible for running containers within Pods. Kubernetes supports various container runtimes, such as Docker, containerd, and CRI-O. These runtimes are responsible for pulling container images, creating containers, and managing their lifecycle.

**Example:** If you define a Pod that runs a Docker container with a specific image, the container runtime (e.g., Docker) pulls the image from a container registry and runs the container as specified.

#### **cAdvisor (Container Advisor):**

cAdvisor is responsible for collecting and exposing resource usage and performance data for containers. It provides valuable information about CPU, memory, network, and disk usage of running containers.

**Example:** You can use cAdvisor to monitor the resource consumption of your containers. For instance, it can help you identify a container that is consuming an unusually high amount of CPU or memory, indicating a potential performance issue.

#### **Node Problem Detector (Optional):**

The Node Problem Detector is responsible for detecting and reporting hardware and system failures on the node. It helps in identifying and isolating issues with nodes, such as hardware errors, kernel panics, or out-of-memory conditions.

**Example:** If a node experiences a hardware issue, such as a failing disk drive, the Node Problem Detector can detect this problem and report it, allowing administrators to take action and potentially drain the node to prevent further issues.

#### **Device Plugins (Optional):**

Device plugins are used to expose and manage specialized hardware resources on the node, such as GPUs, FPGAs, or hardware accelerators. They enable Pods to use these resources when required.

**Example:** If you have GPUs on your nodes and want to run machine learning workloads that require GPU acceleration, you can use a GPU device plugin to expose these GPUs to your Pods. Pods that need GPU resources can request them in their specifications.

#### **OS Services (e.g., SSH, NTP):**

These services, including SSH for remote access and NTP for time synchronization, are essential for maintaining the health and reliability of the node. SSH provides administrative access for troubleshooting and maintenance, while NTP ensures the node's clock is synchronized with the cluster, preventing time-related issues.

**Example:** You can use SSH to log in to a node for troubleshooting or updates. NTP ensures that all nodes in the cluster have synchronized clocks, which is crucial for maintaining consistency in distributed systems.

#### **Kubelet Container:**

The kubelet itself runs in its own container on the node. It is responsible for interacting with the container runtime, managing container logs, and performing garbage collection to reclaim disk space from unused container images.

**Example:** If you examine a running node, you'll find the kubelet running in its own container. It manages container-related tasks on the node, such as cleaning up old container images to free up storage space. These roles collectively ensure the smooth operation of a Kubernetes node and the containers within it, making it possible to run and manage containerized applications in a distributed environment.

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

A **Pod Disruption Budget (PDB)** is a Kubernetes resource that allows you to control the disruption or eviction of Pods during voluntary disruptions (e.g., maintenance) and involuntary disruptions (e.g., hardware failures). PDBs define the minimum availability requirements for Pods in a set of related Pods, such as those belonging to a Deployment or StatefulSet. They ensure that a certain number of Pods are available at all times, helping to maintain the stability and availability of your applications in a Kubernetes cluster.

Here's a detailed explanation of PDBs and an example to illustrate their use:

Components of a Pod Disruption Budget (PDB):

**minAvailable:** This field specifies the minimum number of Pods that must be kept running in the group (e.g., a Deployment or StatefulSet). This ensures that a minimum number of replicas remain available during disruptions.

**maxUnavailable:** This field specifies the maximum number of Pods that can be unavailable during disruptions. It's complementary to minAvailable. You can choose to define one or the other, but not both. It provides a way to limit the maximum unavailability of Pods.

**selector:** PDBs are associated with Pods using label selectors. The selector is used to match Pods in the group that the PDB applies to.

### Use Cases for PDBs:

**Rolling Updates:** When performing rolling updates of applications using Deployments or StatefulSets, PDBs can be used to ensure that a certain number of Pods remain available during the update process, preventing unintended disruptions to the application.

**Node Drains:** When nodes need maintenance or are being drained, PDBs can prevent the simultaneous eviction of too many Pods, ensuring that the application maintains its desired level of availability.

**High Availability:** PDBs can be used to enforce high availability requirements for critical components of an application. For example, a database cluster may require a certain number of replicas to be available at all times to prevent data loss.

### Example of a Pod Disruption Budget:

Let's say you have a Deployment managing a web application with a replica count of 5. You want to ensure that at least 3 replicas of the web application are available at all times during updates or node maintenance. Here's how you would define a PDB for this use case:

```
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
  name: web-app-pdb

spec:
  minAvailable: 3
  selector:
```

matchLabels:

app: web-app

In this example:

minAvailable: 3 specifies that at least 3 replicas of Pods with the label

app: web-app must remain available during disruptions.

selector specifies that this PDB applies to Pods with the label app: web-app.

With this PDB in place, if you perform a rolling update of the Deployment or if nodes need maintenance, Kubernetes will ensure that at least 3 Pods of the web-app Deployment remain operational during these events, thereby meeting the specified availability requirements.

PDBs are a powerful tool for maintaining application stability and availability in Kubernetes clusters, particularly when handling planned or unplanned disruptions to your workloads.

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

In Kubernetes, a Load Balancer is a critical component that helps distribute network traffic evenly across a set of Pods or Services. Load balancing is essential for ensuring high availability, scaling applications, and maintaining stable network connections. Here's a detailed explanation of the role of Load Balancers in Kubernetes, along with an example:

#### **Role of Load Balancers in Kubernetes:**

**Distributing Traffic:** Load Balancers evenly distribute incoming network traffic across multiple Pods or Services. This ensures that no single Pod or Service becomes overwhelmed, improving the responsiveness and availability of the application.

**High Availability:** Load Balancers are typically configured with health checks to monitor the status of Pods or Services. If a Pod or Service becomes unhealthy or unresponsive, the Load Balancer can automatically route traffic away from it, ensuring the application remains available.

**Scaling:** As your application grows and you need to add more instances (Pods) to handle increased traffic, Load Balancers can seamlessly adapt to include these new instances in the traffic distribution. This makes it easier to scale your application horizontally.

**Session Persistence:** Some Load Balancers support session persistence or sticky sessions, which ensure that requests from the same client are consistently routed to the same backend Pod. This is useful for stateful applications that rely on session data.

**External Access:** Load Balancers often act as a point of entry for external traffic into your cluster. They can route traffic to the appropriate Services within the cluster based on the configuration and rules you define.

#### **Types of Load Balancers in Kubernetes:**

**Service Type:** LoadBalancer: Kubernetes provides a native LoadBalancer Service type. When you define a Service of type LoadBalancer, the Kubernetes cluster provisions an external Load Balancer, typically

provided by the cloud provider, to distribute traffic to the Service. For example:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
    type: LoadBalancer
```

**Ingress Controllers:** Ingress controllers, such as Nginx Ingress or HAProxy, are used to manage external access to Services within a cluster. Ingress controllers

provide more advanced routing and traffic management capabilities than the basic LoadBalancer Service type.

### **Example of Load Balancer in Kubernetes:**

Let's say you have a web application deployed as a set of Pods and you want to make it accessible to external users. You can create a LoadBalancer Service to achieve this:

```
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
    type: LoadBalancer
```

In this example:

`metadata.name` is the name of the Service.

`spec.selector` specifies the Pods to which the traffic should be load balanced.

`spec.ports` define the ports to which the Load Balancer should forward traffic.

`type: LoadBalancer` indicates that you want to provision an external Load Balancer for this Service.

Once this configuration is applied, Kubernetes (or the cloud provider) will provision an external Load Balancer and assign it an IP address. Users can then access your web application using this IP address, and the Load

Balancer will distribute incoming requests across the Pods running your web application.

Load Balancers are a fundamental component for ensuring the availability, scalability, and external accessibility of applications in a Kubernetes cluster. They play a crucial role in maintaining a stable and responsive environment for your applications.

**CONCLUSION:** Hence, In this assignment we learned what is the Kubernetes Cluster Architecture.

## **Animesh Parab T2-T21 88**

### **ASSIGNMENT - 06**

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

#### **LO MAPPED – LO1,LO3**

#### **THEORY-**

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.

The key features of Terraform are:

- **Infrastructure as Code:** Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.
- **Execution Plans:** Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.

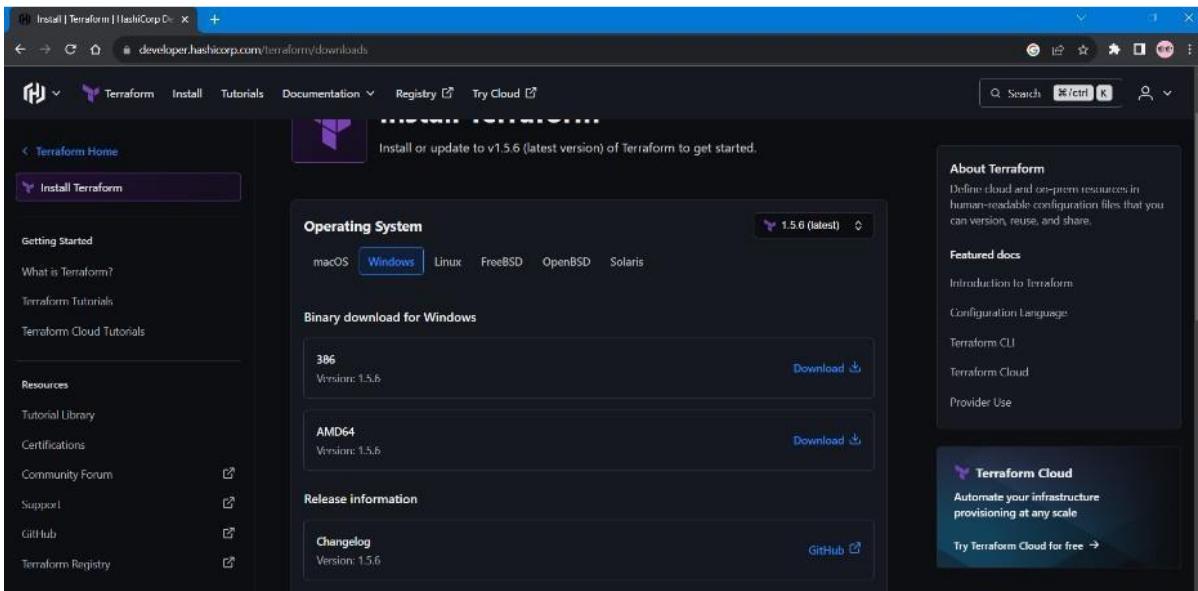
- **Resource Graph:** Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform

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

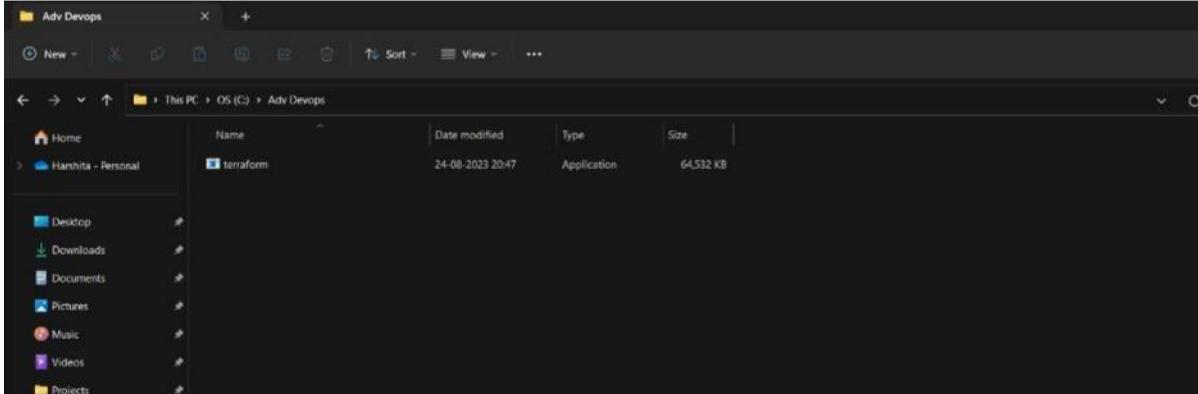
- **Change Automation:** Complex changesets can be applied to your infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, you know exactly what Terraform will change and in what order, avoiding many possible human errors.

#### **STEPS-**

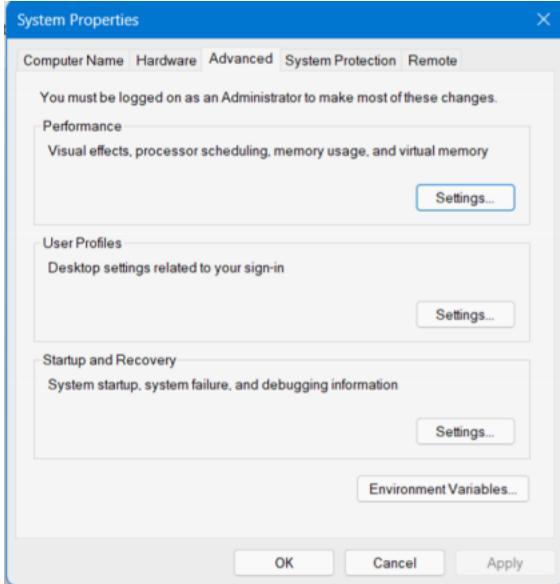
**Google Search – Terraform DOWNLOAD**



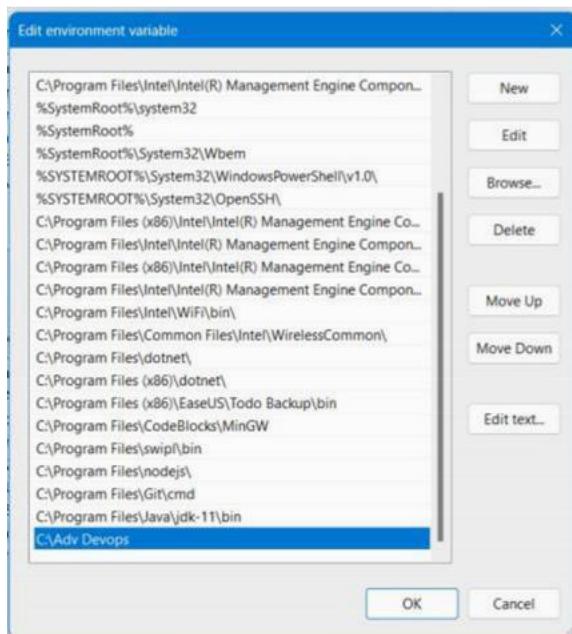
**Create a Folder in Your C drive Named AdvDevops  
Then Extract The downloaded file in The C drive Folder Named AdvDevops  
Shown Below**



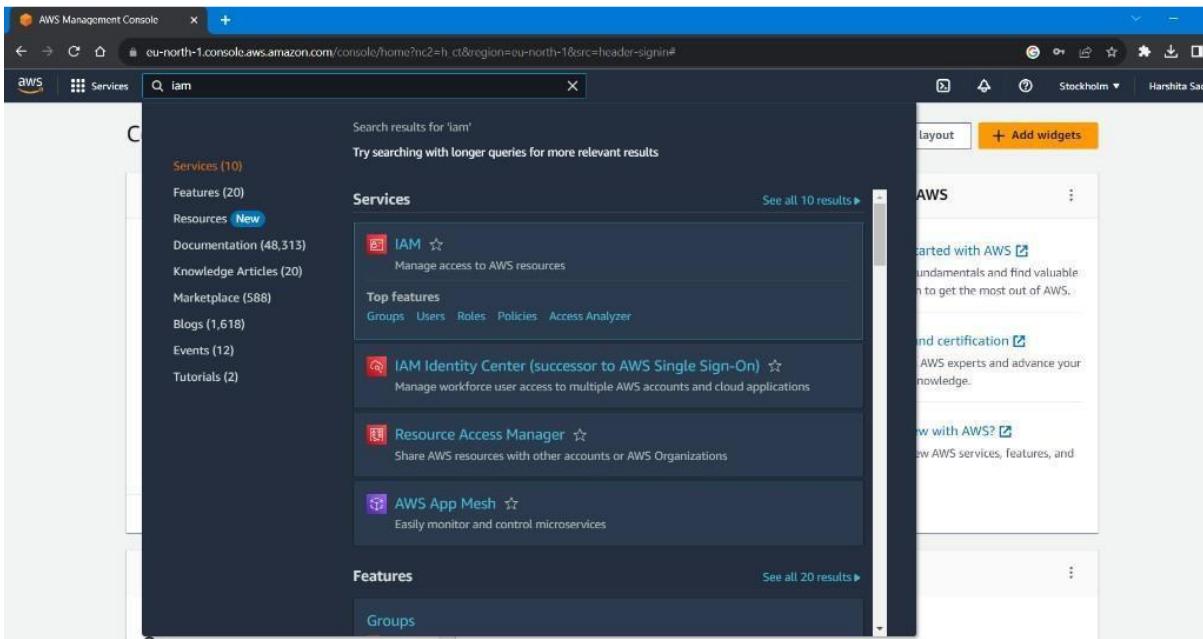
**Now search EDIT THE SYSTEM ENVIRONMENT VARIABLES in your windows se**



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



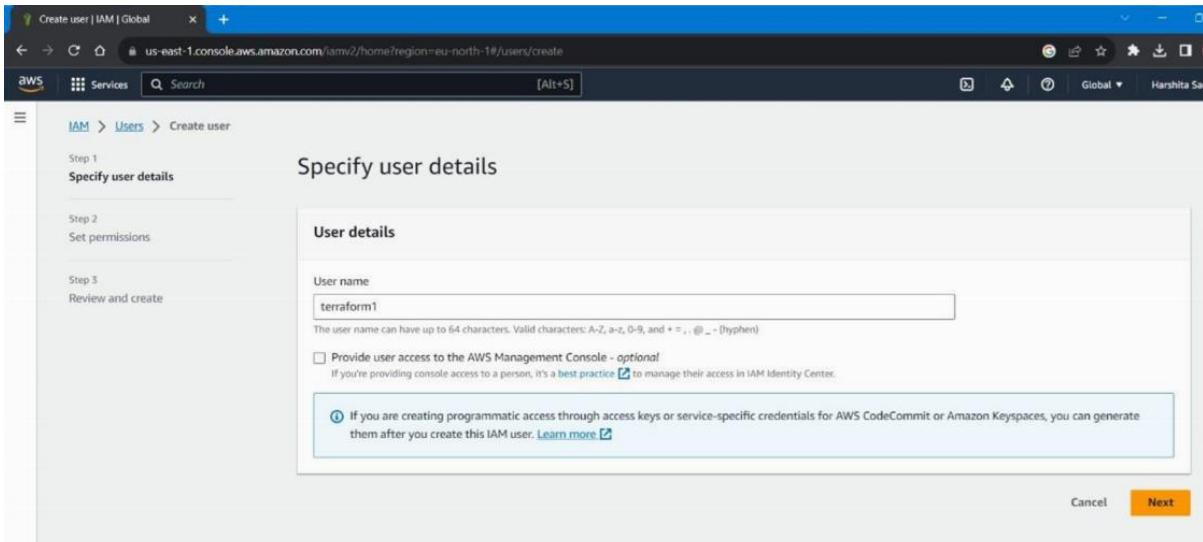
**Repeat same procedure for system variables.**



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

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

name



Screenshot of the AWS IAM User Details page for 'terraform1'.

**Summary**

ARN	arn:aws:iam::374819197411:user/terraform1	Console access	Disabled	Access key 1	Create access key
Created	August 24, 2023, 20:57 (UTC+05:30)	Last console sign-in	-		

**Permissions** | Groups | Tags | Security credentials | Access Advisor

**Permissions policies (0)**  
Permissions are defined by policies attached to the user directly or through groups.

Filter by Type  
Search | All types

Policy name	Type	Attached via
No resources to display		

Screenshot of the 'Create access key' wizard step 2.

**Access key best practices & alternatives**

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

**Use case**

**Command Line Interface (CLI)**  
You plan to use this access key to enable the AWS CLI to access your AWS account.

**Local code**  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

**Application running on an AWS compute service**  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

**Third-party service**  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

**Application running outside AWS**  
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

**Other**  
Your use case is not listed here.

**Alternatives recommended**

Create access key | IAM | Global X +

us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/users/details/terraform1/create-access-key

Services Search [Alt+S]

IAM > Users > terraform1 > Create access key

Step 1 Access key best practices & alternatives

Step 2 - optional Set description tag

Step 3 Retrieve access keys

**Set description tag - optional** Info

The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value  
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ . / = + - @

Cancel Previous Create access key

Create access key | IAM | Global X +

us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/users/details/terraform1/create-access-key

Services Search [Alt+S]

IAM > Users > terraform1 > Create access key

Access key created

This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 1 Access key best practices & alternatives

Step 2 - optional Set description tag

Step 3 Retrieve access keys

**Retrieve access keys** Info

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
<input type="text" value="AKIAVORHXCHRRXZVS47N"/>	<input type="text" value="*****"/> Show

**Access key best practices**

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

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

Download .csv file Done

```
File Edit Selection View Go Run Terminal Help ← → ↻

main.tf ×

Adv Devops > main.tf > ...
provider "aws" {
  access_key = "AKIAVORHXCHR25KBMXPS"
  secret_key = "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"
  region     = "eu-north-1"
}

resource "aws_instance" "terra"{
  ami      = "ami-08766f81ab52792ce"
  instance_type = "t3.micro"
}
```

```
File Edit Selection View Go Run Terminal Help ← → Untitled (Workspace)

main.tf ×

Adv Devops > main.tf > resource "aws_instance" "terra"
provider "aws" {
  access_key = "AKIAVORHXCHRVC66AB3T"
  secret_key = "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"
  region     = "us-east-2"

OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE COMMENTS

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

C:\Adv Devops>terraform init
Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.1...
- Installed hashicorp/aws v5.13.1 (signed by Hashicorp)

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

Terraform has been successfully initialized!

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

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

C:\Adv Devops>]
```

```
resource "aws_instance" "Ubuntu"{
  ami           = "ami-0989fb15ce71ba39e"
  instance_type = "t3.micro"
}

C:\Adv Devops>terraform plan

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

Terraform will perform the following actions:

# aws_instance.Ubuntu will be created
+ resource "aws_instance" "Ubuntu" {
    + ami
    + 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
    + host_id
    + host_resource_group_arn
    + iam_instance_profile
    + id
    + instance_initiated_shutdown_behavior
    + instance_lifecycle
    + instance_state
    + instance_type
    + ipv4_address_count
    + ipv4_addresses
    + key_name
    + monitoring
    + outpost_arn
    + password_data
    + placement_group
    + placement_partition_number
}
```

The screenshot shows the Visual Studio Code interface with a dark theme. The top bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help' menus. A search bar on the right says 'Untitled (Workspace)'. On the far right, there are icons for minimizing, maximizing, and closing the window, along with a timestamp '08'.

The left sidebar contains icons for file operations like Open, Save, Find, Replace, and Git integration.

The main editor area has a title bar 'main.tf' with a close button. Below it is a breadcrumb trail: 'Adv Devops > main.tf > resource "aws\_instance" "terra"'. The code itself is:

```
provider "aws" {
  access_key = "AKIAVORHXCHR25KBW0PS"
  secret_key = "SdKnf3vnth10a8D+3IKU6XFzjj2T+61SLyyRCUJm"
  region     = "eu-north-1"
}

resource "aws_instance" "terra" {
```

Below the code, there are tabs for 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'SQL CONSOLE', and 'COMMENTS'. The 'TERMINAL' tab is selected, showing the output of a Terraform apply command:

```
+ 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)
}
```

The terminal also displays the plan summary:

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

And the confirmation message:

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

The user enters 'yes' at the prompt:

```
Enter a value: yes
```

The terminal shows the progress of the instance creation:

```
aws_instance.terra: Creating...
aws_instance.terra: Still creating... [10s elapsed]
aws_instance.terra: Creation complete after 15s [id=i-09d1e4dec65500aee]
```

Finally, the terminal outputs:

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

The screenshot shows the AWS EC2 Instances page. There are two instances listed:

- Myfirstinstance**: Instance ID i-02631b2a75a6e47d5, Stopped, t3.micro, No alarms, eu-north-1b, Public IPv4 DNS ec2-16-171-134-
- : Instance ID i-09d1e4dec65500aee, Running, t3.micro, Initializing, No alarms, eu-north-1b, Public IPv4 DNS ec2-16-171-134-

```

main.tf
provider "aws" {
  access_key = "AKIAVORHXCHR25KBMXPS"
  secret_key = "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"
  region = "eu-north-1"
}

resource "aws_instance" "terra" {
  root_block_device {
    delete_on_termination = true
    device_name           = "/dev/sda1"
    encrypted              = false
    iops                  = 100
    tags                  = {}
    throughput             = 0
    volume_id              = "vol-03e9fe83f6e5cc0ad"
    volume_size            = 8
    volume_type            = "gp2"
  }
}

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.terra: Destroying... [id:i-09d1e4dec65500aee]
aws_instance.terra: Still destroying... [id:i-09d1e4dec65500aee, 10s elapsed]
aws_instance.terra: Still destroying... [id:i-09d1e4dec65500aee, 20s elapsed]
aws_instance.terra: Still destroying... [id:i-09d1e4dec65500aee, 30s elapsed]
aws_instance.terra: Still destroying... [id:i-09d1e4dec65500aee, 40s elapsed]
aws_instance.terra: Still destroying... [id:i-09d1e4dec65500aee, 50s elapsed]
aws_instance.terra: Still destroying... [id:i-09d1e4dec65500aee, 1m0s elapsed]
aws_instance.terra: Destruction complete after 1m2s

Destroy complete! Resources: 1 destroyed.

```

The screenshot shows the AWS EC2 Instances page. The instance named '-' has been terminated.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
Myfirstinstance	i-02631b2a75a6e47d5	Stopped	t3.micro	-	No alarms	eu-north-1b	-
-	i-09d1e4dec65500aee	Terminated	t3.micro	-	No alarms	eu-north-1b	-

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

## **CONCLUSION –**

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



## **Animesh Parab T2-T21 88**

### **ASSIGNMENT-7**

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

#### **LO MAPPED: LO4**

#### **THEORY:**

SonarQube is a universal tool for static code analysis that has become more or less the industry standard. Keeping code clean, simple, and easy to read is also a lot easier with SonarQube.

What is SonarQube?

SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications. It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.

Benefits of SonarQube

Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications. Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code

- Quality code - Code quality control is an inseparable part of the process of software development.
- Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
- Increase consistency - Determines where the code criteria are breached and enhances the quality
- Business scaling - No restriction on the number of projects to be evaluated
- Enhance developer skills - Regular feedback on quality problems helps developers to improve their coding skills

Why SonarQube?

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

Additionally, they tend to leave unused variables, methods, etc. In this scenario, the code would work in the desired way.

To avoid these issues in code, developers should always follow the good coding practice, but sometimes it is not possible to follow the rules and maintain the good quality as there may be many reasons.

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

## STEPS:

### Download SonarQube and Sonar Scanner

The screenshot shows the SonarQube download page for version 9.1. It features a navigation bar at the top with links for Product, What's New, Documentation, and Community, along with a Download button. Below the navigation is a banner stating "SonarQube 9.1 is here! Project PDFs, JS AWS Lambda taint analysis, Kotlin coroutine rules and more! →". The main content area is titled "Download SonarQube" and describes it as "The leading product for Code Quality and Security HELPING DEVs SINCE 2008". Below this, there are four sections representing different editions:

- Community EDITION**: Used and loved by 200,000+ companies. **FREE & OPEN SOURCE**. Download for free.
- Developer EDITION**: Built for developers by developers. Download.
- Enterprise EDITION**: Designed to meet Enterprise Requirements. Download.
- Data Center EDITION**: Designed for High Availability. Download.

Each section lists additional features available in that edition, such as static code analysis for 15 languages, C/C++ support, and various reporting options.

The screenshot shows the SonarScanner documentation page for version 4.6.2. The left sidebar includes a search bar and links for Try Out SonarQube, Requirements, Setup and Upgrade, Analyzing Source Code (Overview, Scanners), Analysis Parameters, Languages, Test Coverage & Execution, Importing External Issues, and Background Tasks. The main content area is titled "SonarScanner" and includes links for By SonarSource, GNU LGPL 3, and Issue Tracker. It shows the release date (2021-05-07) and provides links for Linux 64-bit, Windows 64-bit, Mac OS X 64-bit, Docker, and Any (Requires a pre-installed JVM). A note states: "The SonarScanner is the scanner to use when there is no specific scanner for your build system." Below this is a section titled "Configuring your project" which explains how to create a configuration file named `sonar-project.properties` and provides a sample configuration snippet:

```
# must be unique in a given SonarQube instance
sonar.projectKey=my:project

# --- optional properties ---

# defaults to project key
sonar.projectName=My project
# defaults to 'not provided'
sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Defaults to .
sonar.sources=.
```

The right sidebar contains a "On this page" menu with links to various configuration topics like Configuring your project, Running SonarScanner from the zip file, and Scanning C/C++ or ObjectiveC Projects.

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

Open command prompt. Run commands:

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

Open another command prompt. Run command:

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

```
C:\ Command Prompt
C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin>sonar-scanner
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)
INFO: Windows 10 10.0 amd64
INFO: User cache: C:\Users\Priyansi\.sonar\cache
INFO: Scanner configuration file: C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default locale: "en_IN", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: -----
INFO: EXECUTION FAILURE
INFO: -----
INFO: Total time: 3.958s
INFO: Final Memory: 5M/20M
INFO: -----
ERROR: Error during SonarScanner execution
ERROR: Not authorized. Analyzing this project requires authentication. Please provide a user token in sonar.login or other credentials in sonar.login and sonar.password.
ERROR:
ERROR: Re-run SonarScanner using the -X switch to enable full debug logging.

C:\Users\Priyansi\Downloads\sonar-scanner-4.6.2.2472-windows\bin>
```

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

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



sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration Search for projects... A

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration.

**From Azure DevOps** **From Bitbucket** **From GitHub** **From GitLab**

**Set up global configuration** **Set up global configuration** **Set up global configuration** **Set up global configuration**

Are you just testing or have an advanced use-case? Create a project manually.

**Manually**

**!** Embedded database should be used for evaluation purposes only

Click on Create a project **Manually**.

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration Search for projects... A

Create a project

All fields marked with \* are required

**Project display name \***  
sonarPythonProgram1 ✓  
Up to 255 characters. Some scanners might override the value you provide.

**Project key \***  
sonarPythonProgram1 ✓  
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '.' (underscore), ';' (period) and ':' (colon), with at least one non-digit.

**Set Up**

**!** Embedded database should be used for evaluation purposes only  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

Give any Project display name.

sonarPythonProgram1 master

How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.

- With Jenkins
- With GitHub Actions
- With Bitbucket Pipelines
- With GitLab CI
- With Azure Pipelines
- Other CI

Are you just testing or have an advanced use-case? Analyze your project locally.

Locally

Embedded database should be used for evaluation purposes only  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

## Click on Locally

sonarPythonProgram1 master

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

① Provide a token

Generate a token

Generate

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your user account.

② Run analysis on your project

Embedded database should be used for evaluation purposes only  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.

SonarQube™ technology is powered by SonarSource SA  
Community Edition - Version 9.1 (build 47736) - LGPLv3 - Community - Documentation - Plugins - Web API - About

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

**Click on Continue.**

**Save a Python program in a folder. class Solution(object):**

```
def romanToInt(self, s)
roman =
{'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000,'IV':4,'IX':9,'XL':40,'XC':
90,'CD':400,'CM':900}

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

Open command prompt in this folder and Run program using copied command. "sonar-scanner.bat -

D"sonar.projectKey=sonarPythonProgram1" -D"sonar.sources=." -  
D"sonar.host.url=http://localhost:9000" -  
D"sonar.login=41740dddf269d68dfda1ec55f28cd250be46d48f"

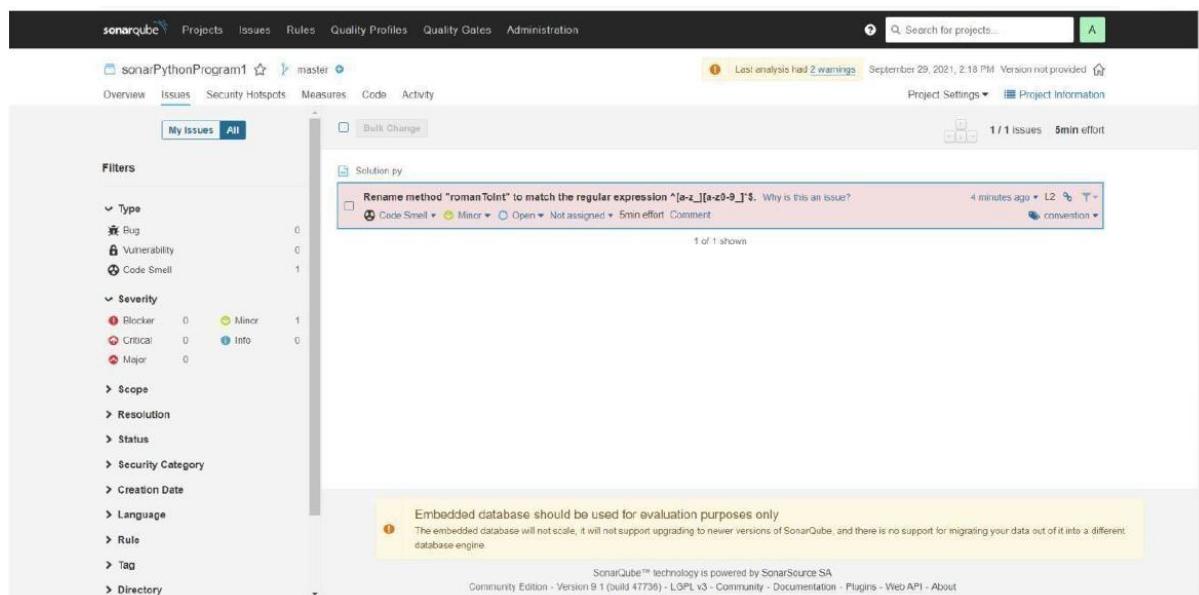
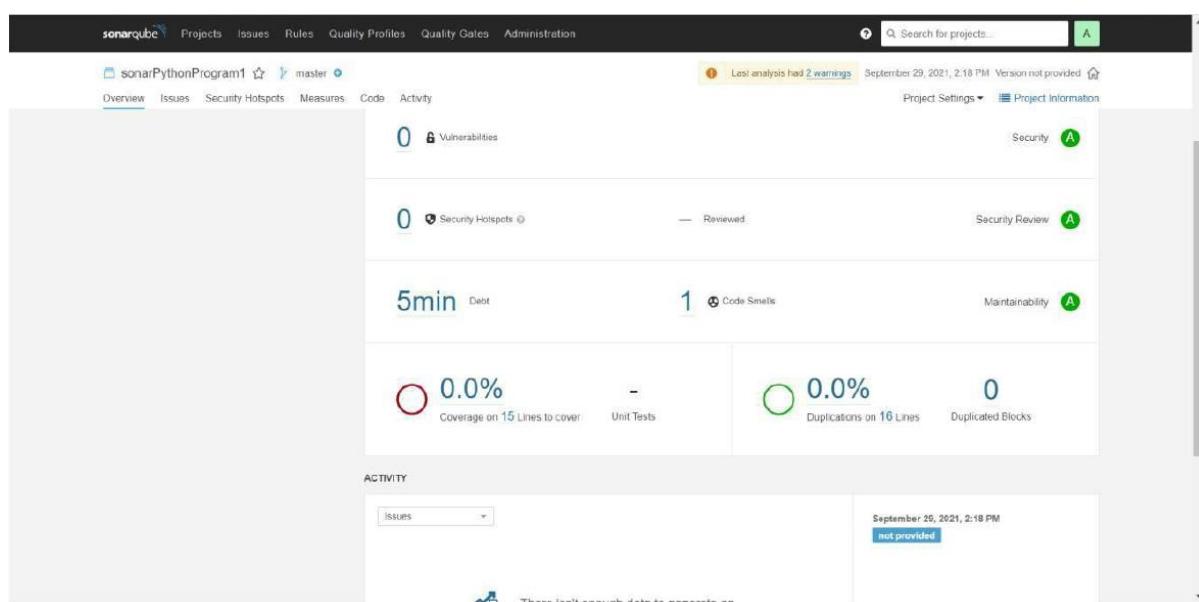
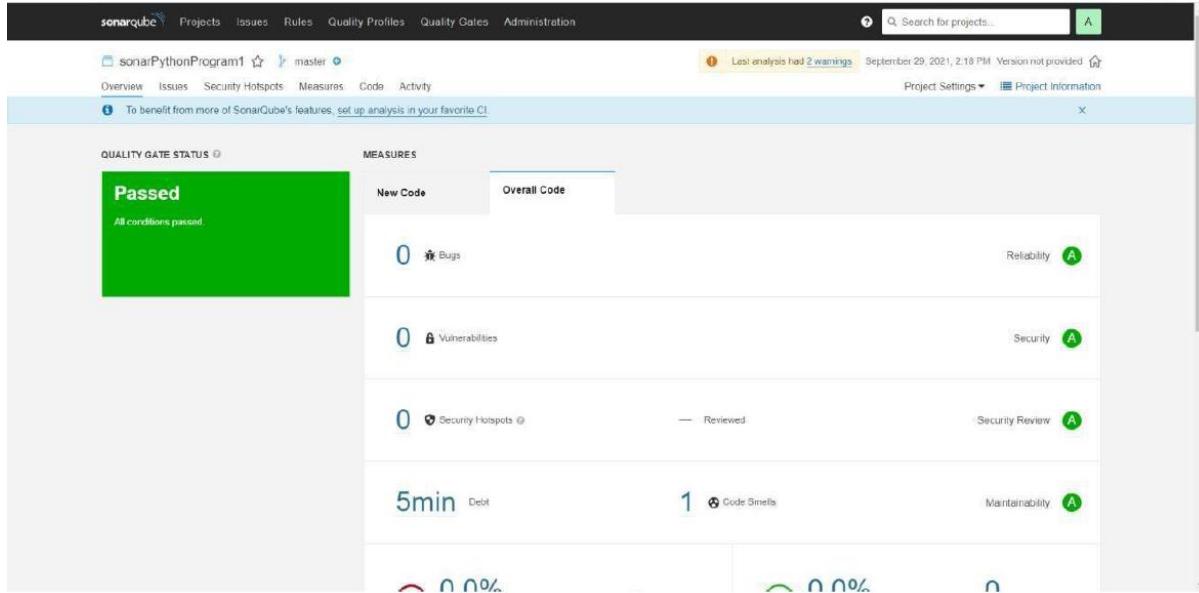
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Priyanshi\Documents\SonarExp\sonan-scanner.bat -D"sonar.projectKey=sonarPythonProgram1" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=41740dddf269d68dfda1ec55f28cd250be46d48f"

INFO: Scanner configuration file: C:\Users\Priyanshi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: SonarScanner 4.6.2.2472
INFO: Java 11.0.11 AdoptOpenJDK (64-bit)
INFO: Windows 10 10.0 amd64
INFO: User cache: C:\Users\Priyanshi\sonar\cache
INFO: Scanner configuration file: C:\Users\Priyanshi\Downloads\sonar-scanner-4.6.2.2472-windows\bin..\conf\sonar-scanner.properties
INFO: Project root configuration file: NONE
INFO: Analyzing on SonarQube server 9.1.0
INFO: Default encoding: "UTF-8", source code encoding: "windows-1252" (analysis is platform dependent)
INFO: Load global settings
INFO: Load global settings (done) | time=20ms
INFO: Server id: BF41AF2-ABwghP94c91b2ce2L5Cu
INFO: User cache: C:\Users\Priyanshi\sonar\cache
INFO: Load/download plugins
INFO: Load plugins index
INFO: Load plugins index (done) | time=102ms
INFO: Load/download plugins (done) | time=1674ms
INFO: Process project properties
INFO: Process project properties (done) | time=20ms
INFO: Execute project builders
INFO: Execute project builders (done) | time=2ms
INFO: Project key: sonarPythonProgram1
INFO: Base dir: C:\Users\Priyanshi\Documents\SonarExp
INFO: Working dir: C:\Users\Priyanshi\Documents\SonarExp\sonan-work
INFO: Load project settings for component key: 'sonarPythonProgram1'
INFO: Load project settings for component key: 'sonarPythonProgram1' (done) | time=40ms
INFO: Load quality profiles
INFO: Load quality profiles (done) | time=20ms
INFO: Load active rules
INFO: Load active rules (done) | time=4452ms
WARNING: SCM provider autodetection failed. Please use "sonar.scm.provider" to define SCM of your project, or disable the SCM Sensor in the project settings.
INFO: Indexing files...
INFO: Project configuration:
INFO: 1 file indexed
INFO: Quality profile for my: Sonar Way
INFO: Load metrics on module sonarPythonProgram1
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=37ms
INFO: Sensor Python Sensor [python]
WARNING: Your code is analyzed as compatible with python 2 and 3 by default. This will prevent the detection of issues specific to python 2 or python 3. You can get a more precise analysis by setting a python version in your configuration via the parameter "sonar.python.version"
INFO: Starting global symbols computation
INFO: 1 source file to be analyzed
INFO: Load project repositories
```

```
C:\Windows\System32\cmd.exe
INFO: Sensor HTML [Web] (done) | time=2ms
INFO: Sensor VB.NET Project Type Information [vbnet]
INFO: Sensor VB.NET Project Type Information [vbnet] (done) | time=1ms
INFO: Sensor VB.NET Analysis Log [vbnet]
INFO: Sensor VB.NET Analysis Log [vbnet] (done) | time=12ms
INFO: Sensor VB.NET Properties [vbnet]
INFO: Sensor VB.NET Properties [vbnet] (done) | time=0ms
INFO: Sensor VB.NET Properties [vbnet] (done) | time=0ms
INFO: Sensor VB.NET Properties [vbnet] (done) | time=0ms
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=12ms
INFO: SCM Publisher No SCM system was detected. You can use the 'sonar.scm.provider' property to explicitly specify it.
INFO: CPD Executor Calculating CPD for 1 file
INFO: CPD Executor CPD calculation finished (done) | time=10ms
INFO: Analysis report generated in 59ms, dir size=103.9 KB
INFO: Analysis report uploaded in 19ms, zip size=14.7 KB
INFO: Analysis report uploaded (done)
INFO: ANALYSIS SUCCESSFUL. You can browse http://localhost:9000/dashboard?id=sonarPythonProgram1
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
INFO: More about the report processing at http://localhost:9000/api/ce/task?id=AxwwwVJhw9lb8xezLKH
INFO: Analysis total time: 7.592 s
INFO:
INFO: EXECUTION SUCCESS
INFO:
INFO: Total time: 10.897s
INFO: Final Memory: 7M/34M
INFO:
```

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



Press “**Ctrl + C**” to stop the server.

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

## **Animesh Parab T2-T21 88**

### **ASSIGNMENT-8**

**AIM:-**To understand continuous monitoring using Nagios

**LO MAPPED:- LO1, LO5**

#### **THEORY:-**

Nagios is an open-source monitoring system that provides monitoring of services, applications, and network resources. It is designed to alert system administrators about potential issues before they become critical problems. Nagios allows you to monitor the entire IT infrastructure, including servers, switches, applications, and services. It provides a comprehensive monitoring solution for both small and large organizations. Some key features and capabilities of Nagios include:

**Monitoring Capabilities:** Nagios can monitor a wide variety of network services including SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH, and many more.

**Alerting and Notification:** It provides alerting and notification functionalities to notify system administrators when something goes wrong. Nagios can send alerts via email, SMS, or other methods to ensure that the right people are notified in real-time.

**Plugin Architecture:** Nagios has a modular architecture that allows users to develop their plugins and addons to monitor specific devices and services that are not covered by default.

**Customizable Dashboards and Reports:** Nagios offers customizable dashboards and reporting capabilities that provide insights into the performance and health of the monitored resources.

**Scalability and Flexibility:** Nagios can scale to monitor complex, large-scale IT infrastructures. It is highly flexible and can be customized to meet specific monitoring and alerting requirements.

**Extensibility:** Nagios can be extended through various addons and plugins, allowing it to integrate with other tools and

services, and enabling the monitoring of a wide range of devices and applications.

**Historical Monitoring and Trend Analysis:** Nagios can store historical data and provide trend analysis, allowing system administrators to identify patterns and plan for future infrastructure needs.

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

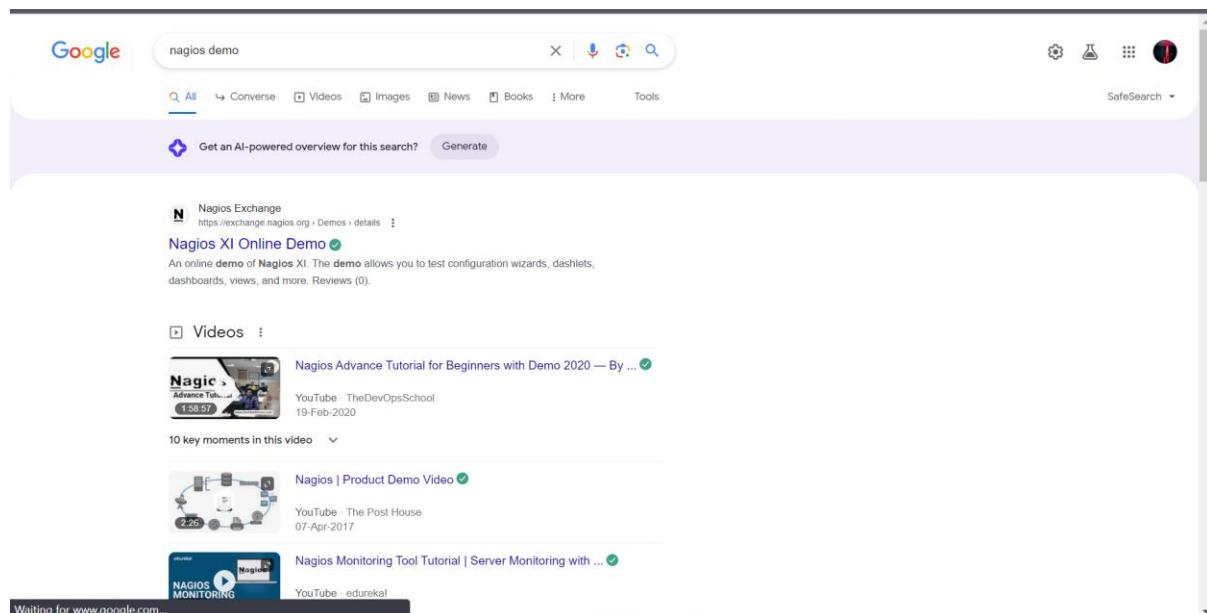
**Centralized Monitoring:** Nagios provides a centralized view of the entire IT infrastructure, allowing administrators to have a comprehensive overview of the health and performance of all monitored resources from a single location.

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

### **STEPS:-**

**1) Go to google.com, Search Nagios Demo**

**Click on the first link shown below**



**2) Now click on the website-**

**Nagios®**

Home | Directory | About

Home | Directory | Demos | Nagios XI Online Demo

Directory Tree

### Nagios XI Online Demo

Submit review | Recommend | Print | Visit | Claim |

Rating ★★★★☆ 0 votes Favoured: 0

Owner [egalstad](#)

Website [nagiosxi.demos.nagios.com](http://nagiosxi.demos.nagios.com)

Hits 150421

An online demo of Nagios XI. The demo allows you to test configuration wizards, dashlets, Waiting for www.google.com...

Search Exchange search... Search Advanced Search

Search All Sites Go

Nagios Live Webinars Let our experts show you how Nagios can help your organization. Register Now

Contact Us Phone: 1-888-NAGIOS-1 Email: sales@nagios.com

Login

### 3) Now click on login as administrator

**Nagios® XI** Login

**Login**

Username  
Password

[Forgot your password?](#)

Select Language:

**Nagios XI Demo System**

**Demo Account Options**

You can access the demo with different accounts to get a different view of the monitoring system.

- Administrator Access** - An account with administrative privileges.  
 Username: nagiosadmin  
Password: nagiosadmin
- Read-Only User Access** - A user account that can view all hosts and services, but not make any changes or issue commands.  
 Username: readonly  
Password: readonly
- Advanced User Access** - An advanced user account that can see and control (schedule downtime, edit, etc) all hosts and services.  
 Username: advanced  
Password: advanced
- Normal User Access** - A sample "normal" user account that has rights to view only a defined subset of all hosts and services.

**Home Dashboard**

**Getting Started Guide**

- Common Tasks:
  - Change your account settings
  - Change your notifications settings
  - Configure your monitoring setup
- Getting Started:
  - Learn about XI
  - Signup for XI news

**Host Status Summary**

Up	Down	Unreachable	Pending
136	105	1	0
Unhandled		Problems	All
24	106	242	

Last Updated: 2023-10-20 14:07:07

**Service Status Summary**

Ok	Warning	Unknown	Critical	Pending
1267	30	101	452	0
Unhandled		Problems	All	
200	583	1850		

Last Updated: 2023-10-20 14:07:08

We're Here To Help!

**Host Status**

All hosts

Showing 1-242 of 242 total records

Host	Status	Duration	Attempt	Last Check	Status Information
www.google.com	Up	319d 10h 20m 52s	1/5	2023-10-20 14:08:06	TCP OK - 0.002 second response time on www.google.com port 80
9a34-181-162-10-91.sa.ngrok.io	Up	128d 20h 22m 7s	1/5	2023-10-20 14:08:06	TCP OK - 0.152 second response time on 9a34-181-162-10-91.sa.ngrok.io port 443
dns.google	Up	142d 3h 57m 5s	1/5	2023-10-20 14:08:06	OK - 8.8.8.8 rta 1.114ms lost 0%
10.0.0.0	Down	295d 0h 3m 9s	5/5	2023-10-20 14:08:06	check_icmp: Failed to resolve 10.0.0.0: Name or service not known
Router	Up	319d 15h 32m 39s	1/10	2023-10-20 14:08:05	OK - 127.0.1.1 rta 0.182ms lost 0%
localhost:22222	Up	319d 10h 20m 55s	1/5	2023-10-20 14:08:05	OK - localhost rta 0.080ms lost 0%
102.68.102.38	Up	123d 2h 19m 20s	1/5	2023-10-20 14:08:05	OK - 102.68.102.38 rta 216.841ms lost 0%
one.one.one.one	Up	319d 10h 20m 57s	1/5	2023-10-20 14:08:05	OK - 1.1.1.1 rta 1.764ms lost 0%
datatys.com	Up	128d 1h 39m 54s	1/5	2023-10-20 14:08:05	HTTP OK: HTTP/1.1 200 OK - 1410 bytes in 0.465 second response time
kredim.com.tr	Up	220d 7h 15m 56s	1/5	2023-10-20 14:08:05	HTTP OK: HTTP/1.1 301 Moved Permanently - 276 bytes in 0.055 second response time
219.92.71.54	Up	125d 4h 19m 56s	1/5	2023-10-20 14:08:05	OK - 219.92.71.54 rta 239.908ms lost 0%
centos5.ngios.local	Up	302d 16h 58m 32s	1/5	2023-10-20 14:08:04	OK: No data received yet
www.itas-ksa.com	Up	122d 2h 31m 23s	1/2	2023-10-20 14:08:04	HTTP OK: HTTP/1.1 200 OK - 442 bytes in 0.188 second response time
siga.uem.mz	Up	1m 12s	1/5	2023-10-20 14:08:04	HTTP OK: HTTP/1.1 301 Moved Permanently - 459 bytes in 0.475 second response time
monster01	Up	128d 20h 21m 14s	1/5	2023-10-20 14:08:04	TCP OK - 0.150 second response time on e0a3-181-162-10-91.sa.ngrok.io port 80

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

## **5) Now click on Host Group Status.**

Nagios® XI

Home Views Dashboards Reports Configure Tools Help Admin

Quick View

- Home Dashboard
- Tactical Overview
- Birdseye
- Operations Center
- Operations Screen
- Open Service Problems
- Open Host Problems
- All Service Problems
- All Host Problems
- Network Outages

Details

- Service Status
- Host Status
- Hostgroup Summary
- Hostgroup Overview
- Hostgroup Grid
- Servicegroup Summary
- Servicegroup Overview
- Servicegroup Grid
- BP1
- Metrics

Graphs

- Performance Graphs
- Graph Explorer

Maps

- World Map
- Bonmap
- Hypermap
- Minimap
- NavVis
- Network Status Map

Incident Management

## Host Group Status

Summary View

Host Status Summary

Up	Down	Unreachable	Pending
53	2	0	6
Unhandled	Problems	All	
2	3	5	

Last Updated: 2021-10-05 05:09:55

## Service Status Summary

Ok	Warning	Unknown	Critical	Pending
661	99	5	36	7
Unhandled	Problems	All		
239	239	1037		

Last Updated: 2021-10-05 05:09:55

### Status Summary For All Host Groups

Host Group	Hosts	Services
Monitoring Servers (Monitoring Servers)	6 UP	93 OK 14 Warning 2 Critical
Hostgroup Two (hg2)	3 UP 1 Down	11 OK 4 Warning 2 Unknown 2 Critical
Some Other Hostgroup (hg3)	2 UP	11 OK 1 Warning 2 Critical
Linux Servers (linux-servers)	31 UP	238 OK 27 Warning 2 Unknown 10 Critical
Network Devices (network-devices)	7 UP	213 OK 35 Warning 62 Critical

Nagios XI 5.7.2 • Check for Updates

## **6) Now we click on BBMap**

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



## CONCLUSION:-

Continuous monitoring with Nagios enables proactive detection of system issues, ensuring minimal downtime and enhanced operational efficiency. Through customizable alerts and comprehensive reporting, Nagios empowers administrators to maintain optimal performance across diverse IT environments. Its scalable architecture and robust community support make it an invaluable tool for streamlined and centralized monitoring.

## **Animesh Parab T2-T21 88**

### **ASSIGNMENT-9**

#### **AIM:- To understand AWS Lambda functions**

#### **LO MAPPED: LO1, LO5**

#### **THEORY:**

AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS) that enables you to run code in response to various events without the need to manage servers. It's a key component of AWS's serverless computing offerings. To understand the theory behind AWS Lambda functions, let's break it down into its core concepts:

#### **Serverless Computing:**

Serverless computing is a cloud computing model where cloud providers (like AWS) manage the infrastructure for you, allowing you to focus solely on your code.

#### **Lambda Function:**

A Lambda function is the fundamental unit of execution in AWS Lambda. It is a piece of code that can be executed in response to events such as HTTP requests, file uploads, database changes, etc. Lambda supports multiple programming languages, including Node.js, Python, Java, and more.

#### **Event Sources:**

Lambda functions are triggered by events. These events can come from various AWS services, like Amazon S3, Amazon DynamoDB, Amazon API Gateway, or custom events from your applications. Lambda listens to these event sources and automatically executes the code you've configured.

#### **Stateless Execution:**

Lambda functions are stateless, meaning they don't maintain any server-specific state between invocations. Each invocation of a Lambda function is independent and isolated from the others.

#### **Scaling and Concurrency:**

AWS Lambda automatically scales based on the number of incoming events. If there are more events, AWS will create more instances of your Lambda function to handle the load, and if there are fewer events, it will scale down accordingly. You pay only for the compute time your code consumes.

#### **Execution Environment:**

Each Lambda function runs in an execution environment provided by AWS. You can't control or manage this environment, but you can specify its configuration, including the amount of memory allocated to the function.

### **Function Versioning and Aliases:**

You can create multiple versions of a Lambda function. This is useful for deploying and managing different versions of your code. You can also create aliases to point to specific versions, allowing you to easily switch between them.

### **IAM Roles:**

Lambda functions can assume AWS Identity and Access Management (IAM) roles. These roles define what AWS services and resources the function can interact with, ensuring proper security and access control.

### **Logging and Monitoring:**

AWS Lambda provides built-in logging to capture function execution details. You can also integrate it with AWS CloudWatch for monitoring and creating custom metrics.

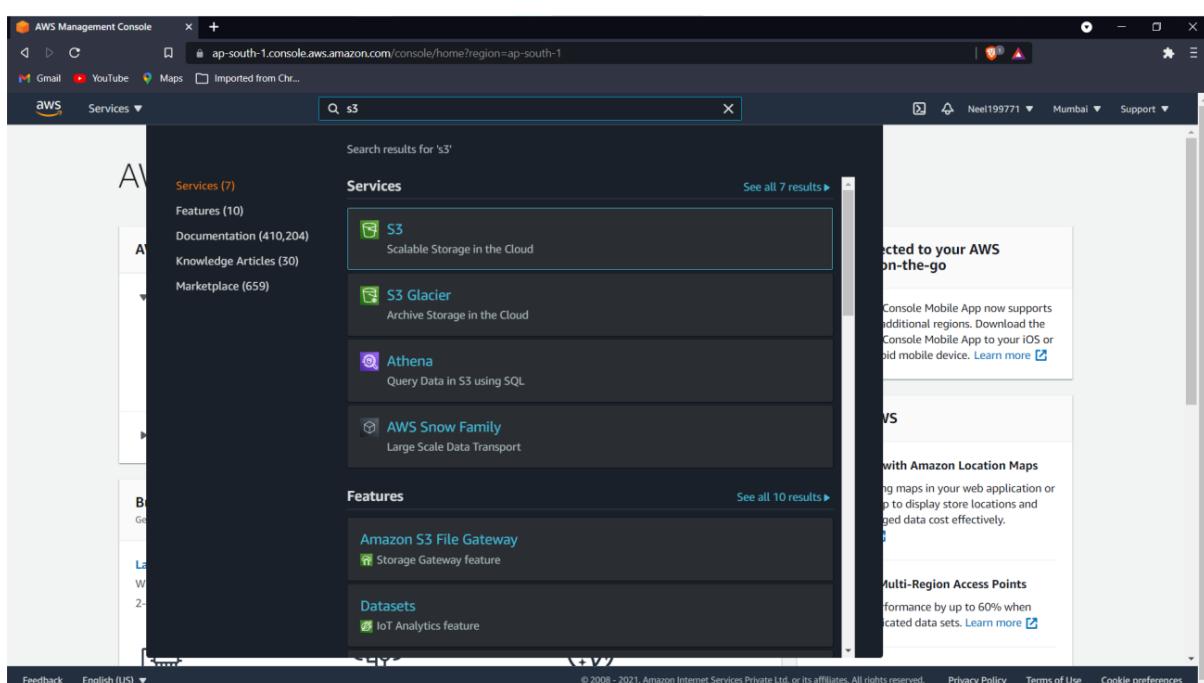
### **Triggers and Destinations:**

Lambda can be triggered by various event sources and can send the results to destinations such as other AWS services, like S3, DynamoDB, SNS, and more.

## **STEPS:**

### **Login to Aws account-**

**Search S3 ,click on the option below shown-**



**Create an S3 bucket by giving it a name**

The screenshot shows the 'Create bucket' wizard on the AWS S3 service. The 'General configuration' section is active, displaying fields for 'Bucket name' (set to 'myawsbucket') and 'AWS Region' (set to 'Asia Pacific (Mumbai) ap-south-1'). Below these are optional 'Copy settings from existing bucket' and 'Choose bucket' options. The 'Block Public Access settings for this bucket' section is expanded, showing the 'Block all public access' checkbox is checked. The 'Tags (0) - optional' section indicates no tags are associated with the bucket. The 'Default encryption' section shows 'Server-side encryption' set to 'Disable'. The 'Advanced settings' section is collapsed. A note at the bottom states: 'After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.' The navigation bar at the bottom includes 'Feedback', language selection ('English (US)'), copyright information ('© 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.'), and links to 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

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

The screenshot shows the 'Upload' page in the Amazon S3 console. At the top, the navigation path is 'Amazon S3 > neel-patel-t21-82 > Upload'. Below the title 'Upload' is a note: 'Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more'.

A central area contains a dashed blue border with the text 'Drag and drop files and folders you want to upload here, or choose Add files, or Add folders.' Below this is a table titled 'Files and folders (0)' with columns: Name, Folder, Type, and Size. A search bar 'Find by name' is above the table. The message 'No files or folders' is displayed, followed by 'You have not chosen any files or folders to upload.'

Below the table is a section titled 'Destination' with a single row labeled 'Destination'.

At the bottom, there are links for 'Feedback', 'English (US) ▾', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

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

The screenshot shows the 'Objects' tab in the Amazon S3 console. The top navigation bar includes tabs for 'Objects' (which is selected), 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'.

The main area displays a table titled 'Objects (2)'. The table has columns: Name, Type, Last modified, Size, and Storage class. Two files are listed:

Name	Type	Last modified	Size	Storage class
Sum1.py	py	September 7, 2021, 15:16:21 (UTC+05:30)	150.0 B	Standard
T11.jpg	jpg	September 7, 2021, 15:31:45 (UTC+05:30)	130.1 KB	Standard

At the bottom, there are links for 'Feedback', 'English (US) ▾', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

**Now search Lamda**

The screenshot shows the AWS Lambda search results page. The search term 'lambda' has been entered into the search bar at the top right. The results are displayed under the 'Services' section, with 'Lambda' being the top result. Other services listed include CodeBuild, AWS Signer, and Amazon Lex. Below the services, there are sections for 'Features' such as Local processing and Target groups.

## Click create function

The screenshot shows the AWS Lambda Functions page. The 'Functions' tab is selected in the navigation bar. A single function is listed with the name 'Functions (1)'. There is a 'Create function' button prominently displayed at the top right of the list.

## Click on below options and click on configure

The screenshot shows the 'Create function' blueprint selection page. The 'Use a blueprint' option is selected. Below this, a grid of blueprint cards is shown, each with a title, description, and a small preview image. Some visible blueprint titles include 'kinesis-firehose-syslog-to-json', 's3-get-object-python', 'config-rule-change-triggered', 'lex-book-trip-python', 'dynamodb-process-stream', 'microservice-http-endpoint', 'kinesis-analytics-output', and 'node-exec'.

Lambda > Functions > Create function > Configure blueprint s3-get-object-python

**Basic information** [Info](#)

Function name  
Neelt21

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

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

ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Role name  
Enter a name for your new role.  
myRoleName

Use only letters, numbers, hyphens, or underscores with no spaces.

Policy templates - optional. [Info](#)  
Choose one or more policy templates.

Amazon S3 object read-only permissions [X](#)

Feedback English (US) ▾ © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

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

**S3 trigger** [Remove](#)

Bucket  
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.  
neel-patel-t21-82

Event type  
Select the events that you want to have 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.  
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

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

**Lambda function code**  
Code is preconfigured by the chosen blueprint. You can configure it after you create the function. [Learn more](#) about deploying Lambda functions.

Feedback English (US) ▾ © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#) [Cookie preferences](#)

**Check the given trigger is created  
Click on**

```

6 s3 = boto3.client('s3')
7
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
17     try:
18         response = s3.get_object(Bucket=bucket, Key=key)
19         print("CONTENT TYPE: " + response['ContentType'])
20     except Exception as e:

```

**Click on the orange test button-**

```

6 s3 = boto3.client('s3')
7
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
17     try:
18         response = s3.get_object(Bucket=bucket, Key=key)
19         print("CONTENT TYPE: " + response['ContentType'])
20     except Exception as e:

```

**Now,**

**Check the logs of the test-**

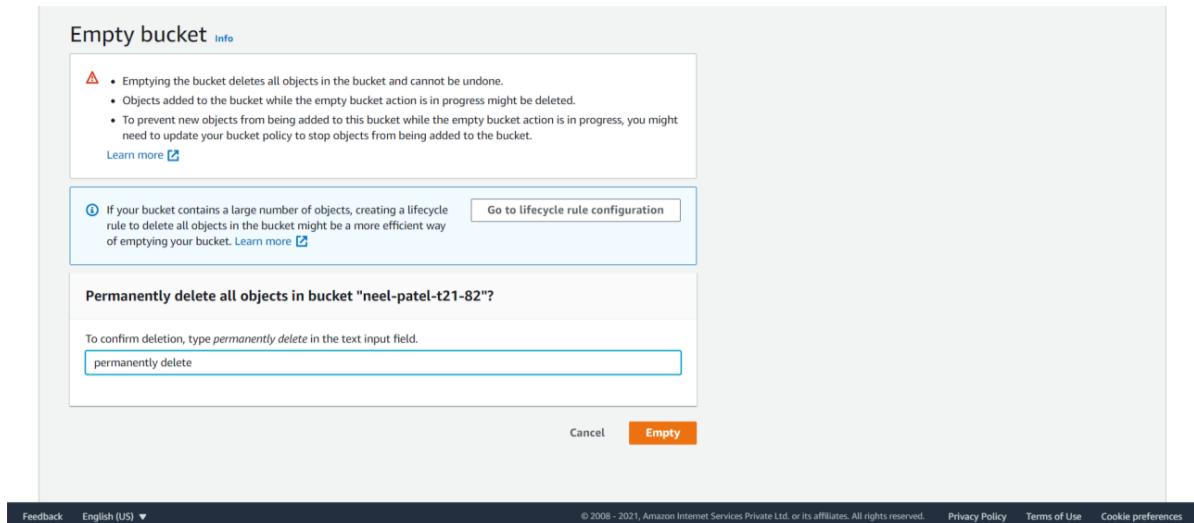
The screenshot shows the AWS CloudWatch Log Groups interface. On the left, a navigation sidebar includes links for New menu experience, Favorites, Dashboards, Alarms, Logs (selected), Log groups (highlighted in orange), Logs Insights, Metrics, Events, Application monitoring (New), Insights, Settings, and Getting Started. The main content area displays the log group details for '/aws/lambda/neel-t21'. It shows retention settings (Never expire), creation time (21 minutes ago), stored bytes (~), ARN (arn:aws:logs:ap-south-1:996163440903:log-group:/aws/lambda/neel-t21:1), KMS key ID (~), metric filters (0), subscription filters (0), and contributor insights rules (~). Below this, the 'Log streams' tab is selected, showing two log streams with their last event times: 2021-09-07/[\$LATEST]e51215ab14be44f855e7bff287da1d (2021-09-07 15:47:48 UTC+05:30) and 2021-09-07/[\$LATEST]842026dddeba34f8baea274d87a7b9793 (2021-09-07 15:32:47 UTC+05:30). A search bar at the top of the log stream list allows filtering by prefix.

**Now terminate-  
Click on delete function.**

The screenshot shows the AWS Lambda function overview for 'neel-t21'. The 'Code' tab is selected. The function has an S3 trigger. A context menu is open over the function name, showing options: Publish new version, Create alias, Export function, and Delete function. The 'Delete function' option is highlighted. The 'Description' field indicates it's an Amazon S3 trigger that retrieves metadata for updated objects. The 'Last modified' field shows it was last modified 31 minutes ago. The 'Function ARN' is listed as arn:aws:lambda:ap-south-1:996163440903:function:neel-t21. Below the code tab, the 'Test' tab is active, showing deployment status: 'Changes deployed'. The browser address bar shows the URL ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/neel-t21?tab=code. The bottom of the screen shows the AWS Lambda service navigation and a search bar.

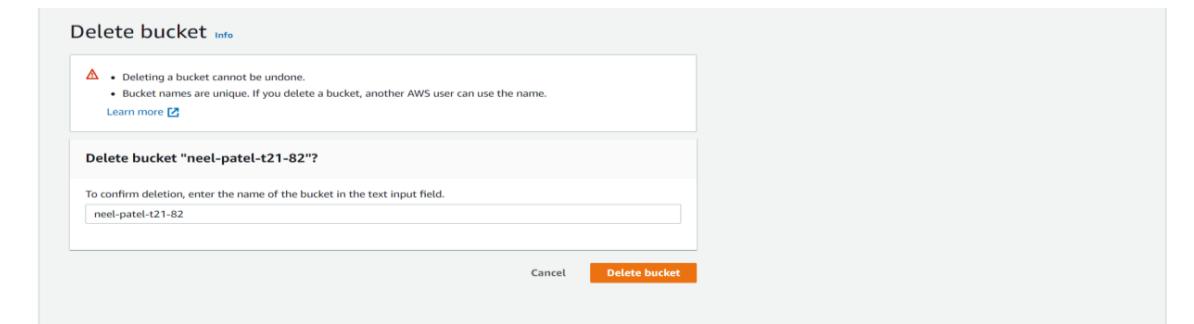
The second part of the screenshot shows the Lambda function details page for 'neel-t21'. A modal dialog titled 'Delete function neel-t21' is open, containing a warning message: 'Deleting a function permanently removes the function code. The related logs and roles are retained in your account.' It has 'Cancel' and 'Delete' buttons. The 'Delete' button is highlighted in orange.

**Empty bucket**



Feedback English (US) ▾ © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

## Delete bucket-



## CONCLUSION:

In this assignment, we covered the core concepts and terminologies of AWS Lambda, explored its practical applications, and gained an understanding of how it integrates with various AWS services.

## **Animesh Parab T2-t21 88**

### **ASSIGNMENT-10**

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

**LO MAPPED: LO1, LO5**

**THEORY:-**

#### **DYNAMO DB**

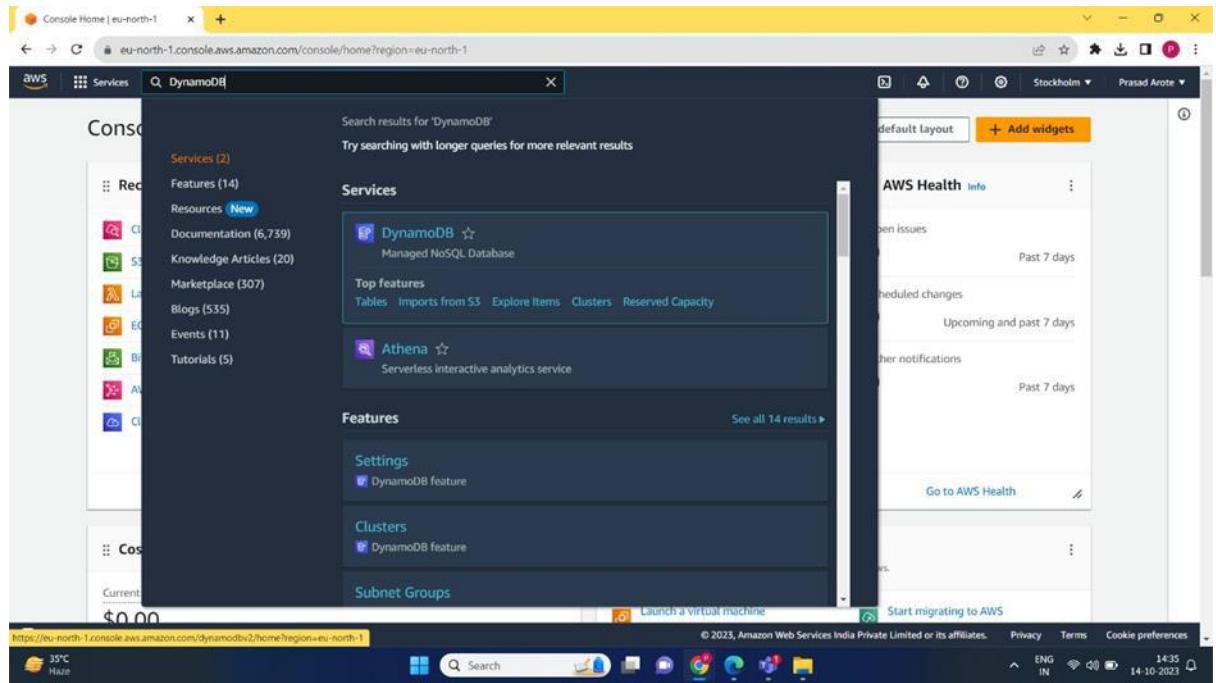
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic. You can scale up or scale down your tables' throughput capacity without downtime or performance degradation. You can use the AWS Management Console to monitor resource utilization and performance metrics

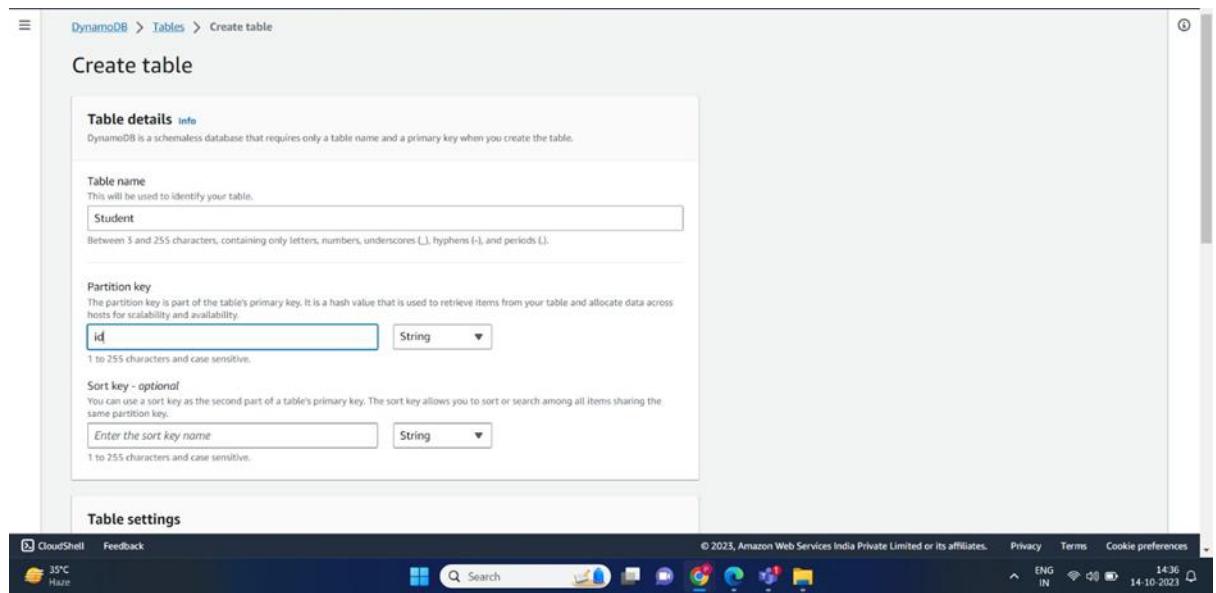
DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

**Steps :-**

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



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



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

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Select trusted entity

**Trusted entity type**

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

**Use case**  
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case  
Lambda

Choose a use case for the specified service.

CloudShell Feedback 35°C Haze © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:40 14-10-2023

IAM > Roles > Create role

Step 1  
Select trusted entity

Step 2  
Add permissions

Step 3  
Name, review, and create

### Add permissions

**Permissions policies (1/887) info**  
Choose one or more policies to attach to your new role.

Policy name	Type	Description
<input checked="" type="checkbox"/>  AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB.
<input type="checkbox"/>  AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon DynamoDB.
<input type="checkbox"/>  AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to DynamoDB...
<input type="checkbox"/>  AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Stream...

▶ Set permissions boundary - optional

Cancel Previous Next

CloudShell Feedback 35°C Haze © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:41 14-10-2023

IAM > Roles > Create role

Step 2  
Add permissions

Step 3  
Name, review, and create

### Role details

**Role name**  
Enter a meaningful name to identify this role.  
prasad\_admin

Maximum 64 characters. Use alphanumeric and '-' characters.

**Description**  
Add a short explanation for this role.  
Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '-' characters.

CloudShell Feedback 35°C Haze © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:41 14-10-2023

Step 1: Select trusted entities

### Trust policy

```

1+ [
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "sts:AssumeRole"
8       ],
9       "Principal": [
10      {
11        "Service": [
12          "lambda.amazonaws.com"
13        ]
14      }
15    ]
16  }
17]

```

CloudShell Feedback 35°C Haze © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:43 14-10-2023

The screenshot shows the AWS Identity and Access Management (IAM) service. On the left, there's a sidebar with options like Dashboard, Access management, Access reports, and CloudShell. The main area is titled 'Roles (9)' and shows a table of roles. The table includes columns for Role name, Trusted entities, and Last activity. The roles listed are:

Role name	Trusted entities	Last activity
AWSCloud9SSMAccessRole	AWS Service: ec2, and 1 more...	75 days ago
AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application	-
AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked)	75 days ago
AWSServiceRoleForSupport	AWS Service: support (Service-Linked)	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service)	-
lambdafunc1-role-11cSlf6u	AWS Service: lambda	1 hour ago
prasad_admin	AWS Service: lambda	-
PyRole	AWS Service: lambda	68 days ago
Runpython	AWS Service: lambda	68 days ago

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

The screenshot shows the AWS Lambda service. At the top, there's a search bar with 'Search results for 'lambda'' and a link to 'Try searching with longer queries for more relevant results'. Below the search bar, there's a 'Services' section with links to Documentation, Knowledge Articles, Marketplace, Blogs, Events, Tutorials, and AWS Signer. To the right of the search bar, there's an 'AWS Health' info panel with sections for Open issues, Scheduled changes, and Other notifications.

The main area is titled 'Create function' and has a sub-section 'Basic information'. It includes fields for 'Function name' (set to 'addstudentdata'), 'Runtime' (set to 'Python 3.9'), 'Architecture' (set to 'x86\_64'), and 'Permissions'. There are also tabs for 'Author from scratch', 'Use a blueprint', 'Container image', and 'Browse serverless app repository'.

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 when writing your function. Note that the console code editor supports only Node.js, Python, and Ruby.  
 Python 3.9

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

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

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:46 14-10-2023

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](#).  
 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.  
 View the [prasad\\_admin role](#) on the IAM console.

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:46 14-10-2023

▼ Advanced settings

Enable Code signing [Info](#)  
Use code signing configurations to ensure that the code has been signed by an approved source and has not been altered since signing.

Enable function URL [Info](#)  
Use function URLs to assign HTTP(S) endpoints to your Lambda function.

Enable tags [Info](#)  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources, track your AWS costs, and enforce attribute-based access control.

Enable VPC [Info](#)

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:46 14-10-2023

## 4) Write the following code in code source

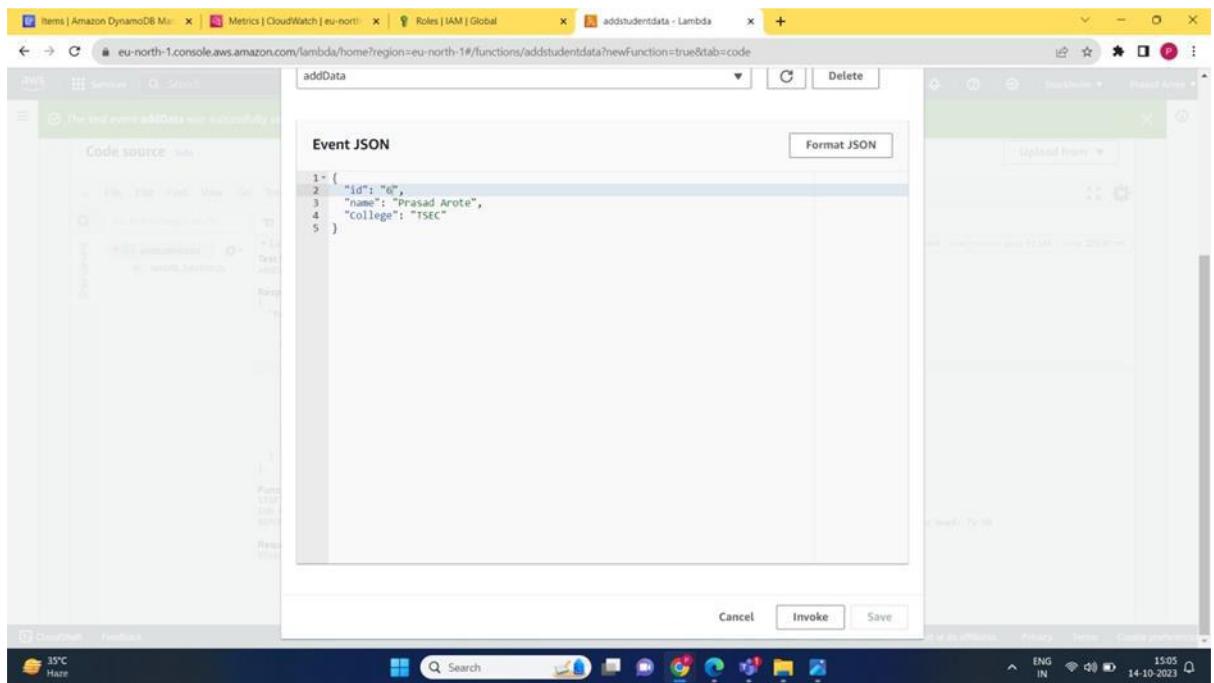
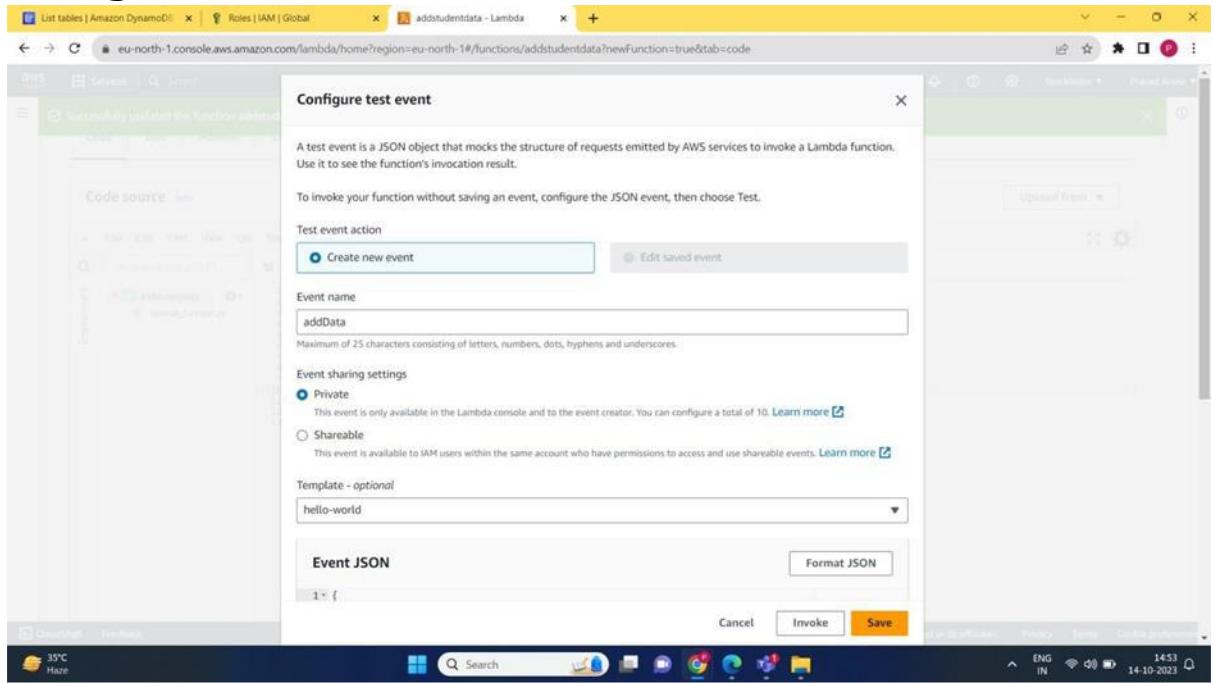
```

Code source Info Upload from ▾
File Edit Find View Go Tools Window Test Deploy
lambda_function Environment Var
lambda_function
Environment addstudentdata lambda_function.py
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     # TODO implement
6     client = boto3.resource('dynamodb')
7     table = client.Table('Student')
8
9     response = table.put_item(Item=event)
10    return response
11
12
13

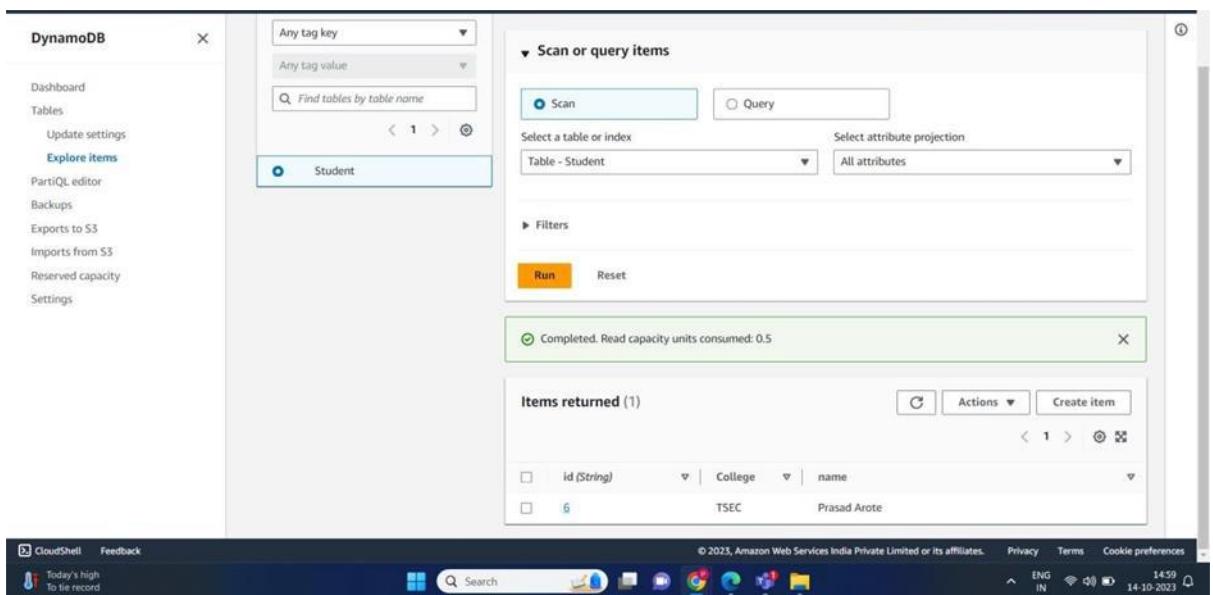
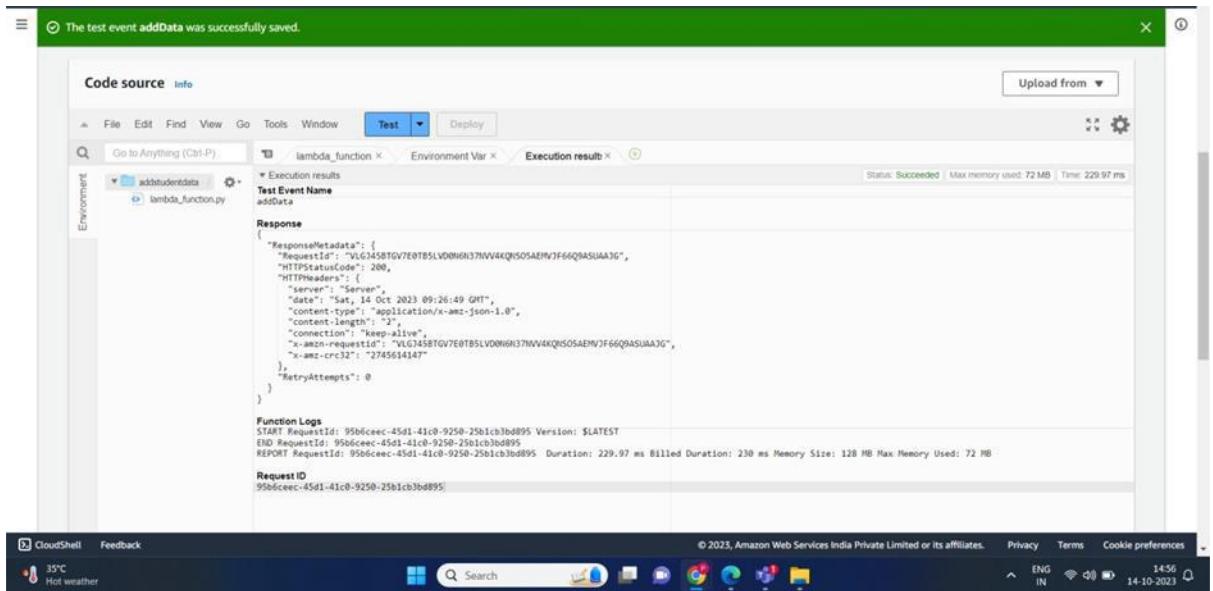
```

CloudShell Feedback © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences ENG IN 14:52 14-10-2023

## 5) Configure the test event and save



## 6) Run the test and afterwards go to the DynamoDB>Explore items> Student where you can see the record inserted using lambda function.



## CONCLUSION:-

Learnt about Amazon DynamoDB database service and inserted data into DynamoDB database by creating a new user , granting him permissions and then using a lambda function

**Adv. DevOps Written Assignment : 01**

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

1. Role-Based Access Control (RBAC): RBAC restricts who can perform actions within a Kubernetes cluster. It defines roles and role bindings to specify what resources and operations users or service accounts can access. This prevents unauthorized access and actions within the cluster.
2. Regular Updates: Keeping Kubernetes and its components up to date is crucial. New releases often include security patches. Regular updates help mitigate known vulnerabilities and ensure your cluster remains secure.
3. Network Policies: Network policies allow you to define rules for communication between pods. By specifying which pods can communicate with each other, you can limit the attack surface and prevent unauthorized access.
4. Container Security Tools: Employ container security tools like vulnerability scanners to assess the security of container images. These tools can identify and remediate vulnerabilities in the containerized applications before they are deployed.
5. Monitoring and Audit: Implement monitoring and auditing solutions to track cluster activity. This helps detect and respond to suspicious or unauthorized behavior. Tools like Prometheus and Grafana can be used for monitoring, while audit logs provide insights into cluster activity.
6. Secrets Management: Sensitive data like API keys, passwords, and certificates should be stored securely using Kubernetes secrets or external vaults. This prevents sensitive information from being exposed within containers or configuration files.
7. PodSecurityPolicies (PSP): PSP is a Kubernetes feature that enforces security policies at the pod level. It allows you to define restrictions on privilege escalation, host access, and other security-sensitive configurations for pods.

8. Namespaces: Use Kubernetes namespaces to logically isolate workloads. This provides a level of separation between different applications or teams, reducing the risk of unauthorized access or interference between them.
9. Admission Controllers: Admission controllers are webhook plugins that intercept and validate requests to the Kubernetes API server. You can use them to enforce custom policies and ensure that only compliant resources are admitted to the cluster.
10. Container Runtime Security: Implement container runtime security solutions like Docker Security Scanning or container runtime protection tools. These tools monitor containers at runtime for abnormal behavior, helping to detect and respond to potential threats.

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

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

Three security techniques commonly used to protect data are:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
number: 80
```

```
```
```

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

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

```
```bash
```

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

```
```
```

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

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

#### **4. Which service protocols does Kubernetes ingress expose?**

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

- 1. HTTP:** Ingress is commonly used to expose HTTP services. You can define routing rules based on URL paths, hostnames, and other HTTP attributes.
- 2. HTTPS:** Secure HTTP services can be exposed through Ingress by configuring TLS certificates. This allows you to terminate SSL/TLS encryption at the Ingress controller and route decrypted traffic to your services.
- 3. TCP:** Some Ingress controllers, like Nginx Ingress, support TCP services. This enables you to expose non-HTTP services such as databases or custom protocols. TCP-based routing typically relies on port numbers.
- 4. UDP:** While less common, some Ingress controllers support UDP services. UDP is a connectionless protocol used for various purposes, including DNS and VoIP. Exposing UDP services may require specific controller support.
- 5. gRPC:** If your services use the gRPC protocol, you can configure Ingress resources to handle gRPC traffic. gRPC is a high-performance RPC (Remote Procedure Call) framework often used for communication between microservices.
- 6. WebSocket:** Ingress controllers can be configured to support WebSocket connections. WebSocket is a protocol that enables full-duplex communication over a single TCP connection and is used for real-time applications.

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

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

## **Animesh Parab T2-T21 88**

### **WRITTEN ASSIGNMENT-2**

#### **Q.1 How to deploy Lambda function on AWS?**

=>Deploying a Lambda function on AWS involves several steps. Here's a detailed guide on how to deploy a Lambda function:

##### **Prerequisites:**

An AWS account.

The AWS Command Line Interface (CLI) installed and configured with appropriate permissions.

Your Lambda function code packaged as a ZIP archive or uploaded to an Amazon S3 bucket.

Familiarity with the programming language and runtime you're using for your Lambda function (e.g., Node.js, Python, Java, etc.).

##### **Step-by-Step Guide to Deploying a Lambda Function:**

Create or Prepare Your Lambda Function Code:

Write your Lambda function code or prepare it if you haven't already.

Ensure it follows the AWS Lambda function structure, including the handler function.

Package Your Code:

If your function code consists of multiple files or dependencies, package it as a ZIP archive. Make sure that the primary function handler is at the top level of the archive.

Create an Execution Role (if needed):

Lambda functions often need permissions to interact with other AWS services. Create an AWS Identity and Access Management (IAM) role that grants the necessary permissions to your Lambda function. The role should have policies attached that allow access to AWS resources, like S3, DynamoDB, or others.

Upload Code to Amazon S3 (if needed):

If your deployment package is larger than 3 MB, you will need to upload it to an Amazon S3 bucket. Make sure your Lambda function has permissions to access this bucket.

##### **Deploy the Lambda Function:**

You can deploy a Lambda function using the AWS Management Console, AWS CLI, AWS SDKs, or AWS CloudFormation. Here's how to deploy using the AWS CLI:

Open a terminal and run the following AWS CLI command to create your Lambda function:

```
aws lambda create-function --function-name MyFunctionName --runtime nodejs14.x --role arn:aws:iam::123456789012:role/MyRole --handler index.handler --zip-file fileb://function.zip
```

Replace MyFunctionName with your desired function name.

Specify the correct runtime for your function (e.g., nodejs14.x for Node.js 14).

Use the --role option with the ARN of the IAM role you created. Specify the --handler option with the name of your handler function. Use --zip-file to reference your deployment package.

### **Test Your Lambda Function:**

After creating your Lambda function, you can test it using the AWS Management Console, AWS CLI, or an SDK. Make sure it works as expected.

### **Configure Event Sources (if needed):**

If your Lambda function is triggered by events from other AWS services (e.g., S3, SNS, API Gateway), configure these event sources in the AWS Management Console.

### **Set Up Environment Variables (if needed):**

If your function relies on environment variables, configure these in the Lambda function's configuration.

### **Deploy Updates (if needed):**

If you make changes to your function code or configuration, you can update the Lambda function by uploading a new deployment package, and the changes will be applied.

### **Monitor and Troubleshoot:**

Use AWS CloudWatch and other monitoring tools to track the performance and behavior of your Lambda function. You can also view logs and error messages in CloudWatch Logs to troubleshoot issues.

### **Scale and Manage Your Function:**

AWS Lambda automatically scales your function to handle incoming requests. You can configure concurrency limits, timeout settings, and other properties to manage its behavior.

### **Cost Management:**

Keep an eye on the costs associated with your Lambda function, as you'll be billed based on the number of requests and duration of execution.

Utilize cost management tools and set up billing alerts to avoid unexpected charges.

Deploying a Lambda function on AWS is a straightforward process, but it's important to pay attention to configuration, permissions, and monitoring to ensure your function operates as expected in a production environment.

## **Q.2 What are the deployment options for AWS Lambda?**

AWS Lambda offers several deployment options, each suited for different use cases and development workflows. Here are the primary deployment options for AWS Lambda, explained in detail:

### **Upload Deployment Package:**

This is the simplest deployment option. You create a ZIP archive that contains your function code and dependencies. Then, you manually upload it when creating or updating your Lambda function. This approach is suitable for small functions or when you need full control over your deployment process.

Steps:

Create a ZIP archive containing your function code and dependencies. Use the AWS Management Console, AWS CLI, or AWS SDKs to create or update your Lambda function, providing the ZIP archive as the deployment package.

#### S3 Bucket Deployment:

For larger deployment packages or for separating the deployment process from the Lambda function creation or update, you can store your deployment package in an Amazon S3 bucket. When you create or update a Lambda function, you specify the S3 bucket location for your deployment package.

#### Steps:

Upload your deployment package to an S3 bucket.

Use the AWS Management Console, AWS CLI, or AWS SDKs to create or update your Lambda function, specifying the S3 bucket location.

#### AWS Serverless Application Model (SAM):

SAM is an open-source framework for building serverless applications. It extends AWS CloudFormation to provide a simplified way to define the Amazon API Gateway APIs, AWS Lambda functions, and Amazon DynamoDB tables needed by your serverless application. You can define your Lambda function configurations and deployment details in a template.yaml file, which SAM uses to create and deploy the function.

#### Steps:

Create a template.yaml file that defines your Lambda function and its dependencies.

Use the AWS SAM CLI to package and deploy your serverless application to AWS. This CLI tool simplifies packaging and deployment, including the creation of Amazon S3 buckets for deployment.

#### Lambda Layers:

Lambda Layers allow you to separate your function code from its dependencies. You can create a custom Layer with your dependencies and then reference it in your

Lambda function. When updating the dependencies, you only need to update the Layer, reducing the size and complexity of the deployment package.

#### Steps:

Create a Lambda Layer containing your dependencies.

Reference the Layer in your Lambda function configuration.

When updating dependencies, update the Layer without changing your function code.

#### Container Images (AWS Lambda for Containers):

AWS Lambda added support for running functions in container images. With this option, you build a container image with your function code, dependencies, and any runtime environment you need. You can use your preferred container registry to store the image. Lambda manages the container execution, scaling, and resource allocation.

#### Steps:

Build a Docker container image with your function code and dependencies.

Push the image to a container registry (e.g., Amazon ECR, Docker Hub). Create a Lambda function using the image from your container registry.

#### Continuous Deployment Tools:

You can integrate AWS Lambda deployment into your continuous deployment pipelines using tools like AWS CodePipeline, AWS CodeBuild, Jenkins, or any other CI/CD solution. This approach automates the deployment process, allowing you to push changes to your function code in a version-controlled manner.

#### Steps:

Set up a CI/CD pipeline that monitors your code repository.

Configure the pipeline to build and deploy your Lambda function when changes are detected.

Each of these deployment options has its own advantages and is suitable for different scenarios. The choice depends on factors such as the size and complexity of your function, your development workflow, and whether you require more granular control over your deployments.

### **Q.3 What are the 3 full deployment modes that can be used for AWS?**

In the context of AWS, there are three primary full deployment modes, each offering a distinct approach to deploying and managing applications. These modes are designed to accommodate different use cases and operational requirements. Let's explore each of them in detail:

#### **EC2-Based Deployment:**

##### **Description:**

EC2-based deployment is the traditional deployment mode in AWS where you provision and manage virtual machines (EC2 instances) to run your applications. In this mode,

you have full control over the underlying infrastructure, including the choice of instance types, operating systems, and configuration. This mode is ideal for applications that require a high degree of customization or when you have legacy systems that need to run in a traditional virtualized environment.

##### **Use Cases:**

Running legacy applications or software that can't be containerized.

Applications with complex network configurations or specific hardware requirements.

When you need to manage the entire stack from the operating system up.

##### **Key Components:**

Amazon Elastic Compute Cloud (EC2) instances.

Amazon Virtual Private Cloud (VPC) for network isolation.

Elastic Load Balancers for distributing traffic.

Amazon RDS or other database services for data storage.

##### **Benefits:**

Complete control over infrastructure.

Compatibility with a wide range of software.  
Ability to run non-containerized or legacy applications.

**Challenges:**

Manual scaling and management of EC2 instances.  
More operational overhead compared to serverless or container-based solutions.  
Limited automation compared to other deployment modes.

**Serverless Deployment:**

**Description:**

Serverless deployment is a modern cloud computing paradigm in which you focus solely on writing code (usually in the form of functions) and let the cloud provider manage all the underlying infrastructure. AWS Lambda is a key component of this approach, allowing you to run code in response to events without provisioning or managing servers. Serverless computing is highly scalable and event-driven.

**Use Cases:**

Building scalable, event-driven applications.  
Microservices architecture.  
Real-time data processing and analysis.  
Reducing operational overhead by offloading infrastructure management.

**Key Components:**

AWS Lambda for running code in response to events.  
Amazon API Gateway for exposing APIs.  
Various AWS services for data storage and processing.  
Amazon EventBridge or Amazon S3 event triggers for event-driven applications.

**Benefits:**

Auto-scaling and high availability.  
Minimal operational overhead.  
Pay-per-use pricing.  
Easy integration with other AWS services.

**Challenges:**

Stateless execution, which may require workarounds for stateful applications.  
Limited runtime options compared to EC2 instances.  
Function duration and resource limits.

**Container-Based Deployment:**

**Description:**

Container-based deployment leverages containerization technology to package applications and their dependencies into a consistent and portable format. AWS offers Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) for managing containers in a scalable, automated, and highly available manner. This deployment mode is ideal for containerized applications, microservices, and orchestrating container workloads.

**Use Cases:**

Microservices architecture.

Porting and running containerized applications.

Managing applications that need to scale and have dependencies isolated in containers.

**Key Components:**

Amazon ECS or Amazon EKS for managing containers.

Amazon ECR for container registry.

Docker for building and running containers.

Kubernetes for container orchestration (EKS).

**Benefits:**

Scalability and flexibility of containerization.

Portability and consistency of containers.

Advanced orchestration and management features in Kubernetes.

**Challenges:**

Container management complexity.

Learning curve for orchestrators like Kubernetes.

Ongoing operational overhead.

These three full deployment modes represent different approaches to deploying and managing applications in AWS. Your choice should be based on your specific requirements, including the nature of your applications, your scalability needs, and your desired level of operational control. It's not uncommon for organizations to use a combination of these deployment modes, depending on their application portfolio and use cases.

#### **Q.4 What are the 3 components of AWS Lambda?**

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows you to run code in response to events without the need to manage servers. AWS Lambda has three core components that work together to enable serverless compute capabilities:

**Lambda Function:**

**Description:** A Lambda function is the core unit of execution in AWS Lambda. It represents your code, which can be written in various programming languages such as Node.js, Python, Java, Go, and more. A Lambda function is a small, self-contained piece of code that can perform a specific task when triggered by an event. It can be as simple as a few lines of code or more complex, and it typically follows a specific structure, including a handler function that AWS Lambda invokes when an event occurs.

**Use Cases:** Lambda functions are used for a wide range of purposes, including data processing, automation, real-time file processing, API endpoints, and more. They are particularly well-suited for building serverless applications and microservices.

**Key Characteristics:**

Small, single-purpose code.

Stateless (no persistent storage of data between invocations).

Event-driven execution.

Automatic scaling and resource allocation.

**Event Source:**

**Description:** Event sources are triggers that initiate the execution of a Lambda function. These sources can be various AWS services or external systems that generate events. When an event occurs, AWS Lambda is automatically invoked, and the event data is passed to the Lambda function for processing. AWS Lambda supports a wide range of event sources, including AWS services like Amazon S3, Amazon DynamoDB, AWS SNS, and custom event sources using AWS Step Functions.

**Use Cases:** Event sources enable Lambda functions to respond to changes in data, incoming requests, system events, and more. This makes them suitable for building event-driven applications and automating workflows.

**Key Characteristics:**

Diverse sources, including AWS services and custom events.

Real-time event triggering. Integration with various AWS services.

**Execution Environment:**

**Description:** The execution environment is the runtime environment where Lambda functions run. AWS Lambda manages and provisions these environments dynamically as needed, and it abstracts the underlying infrastructure from developers. The execution environment includes the compute resources (CPU, memory) and the network configuration necessary for the function's execution. The environment automatically scales with incoming event load.

**Use Cases:** The execution environment is responsible for ensuring that Lambda functions can run in a scalable and highly available manner without the need for manual provisioning or management. It enables the on-demand execution of code.

**Key Characteristics:**

Automatic provisioning and scaling.

Abstracts infrastructure management.

Resource allocation (memory and CPU) defined per function.

Isolation between concurrent executions.

Together, these three components form the foundation of AWS Lambda. A Lambda function processes events from various event sources within an execution environment provided by AWS Lambda. The serverless nature of Lambda, where you focus on code rather than infrastructure, allows you to build applications that are highly scalable and responsive to real-time events with minimal operational overhead. This serverless model is particularly valuable for organizations looking to optimize resource utilization and reduce infrastructure management complexities.