

## Assignment-3

Name-Suraj Kumar Choudhary

PNR No.-240840127041

Q.1 Create a Lambda function which will display “This is server” message if the code sent is 001. “This is node1” if code sent is 002 and “This is a router” if the code sent is 003. For any other code it will display “Invalid code” message.

Create a test event with code as 002. and display the message.

*Following are the steps to implement the Lambda function.*

### Step 1: Create a Lambda Function in AWS

#### 1. Open the AWS Lambda Console:

- Go to the AWS Lambda Console.

#### 2. Create a Lambda Function:

- Click Create function.
- Select Author from scratch.
- Set the function name as CodeMessageLambda.
- Select Python 3.x (latest version) as the runtime.
- Under Permissions, choose Use an existing role and select LambdaExecutionRole (or create a new role if you haven't already).
- Click Create function.

#### 3. Add the Python Code:

*In the inline editor, replace the default code with the following:*

```

import json

def lambda_handler(event, context):
    # Extract the 'code' from the query string
    parameters = event.get('queryStringParameters', {})
    code = parameters.get('code', '')

    # Determine the message based on the code
    if code == "001":
        message = "This is server"
    elif code == "002":
        message = "This is node1"
    elif code == "003":
        message = "This is a router"
    else:
        message = "Invalid code"

    # Return the response as JSON
    return {
        'statusCode': 200,
        'body': json.dumps({
            'message': message
        }),
        'headers': {
            'Content-Type': 'application/json'
        }
    }

```

*This code checks the code query parameter and returns a corresponding message.*

#### 4. Deploy the Function:

- Click Deploy to save and deploy the function.

## Step 2: Set Up an API Gateway to Call the Lambda Function

### 1. Open the API Gateway Console:

- Go to the API Gateway Console.

## 2. Create a New HTTP API:

- Click Create API.
- Choose HTTP API and click Build.

## 3. Configure the API:

- Set the API name to CodeMessageAPI.
- Click Next.

## 4. Set Up Lambda Integration:

- In Integrations, select Lambda.
- Choose the Lambda function you created (CodeMessageLambda).

## 5. Create a Route:

- Under Routes, click Add route.
- Set the Method to ANY (this allows you to call the function with any HTTP method like GET or POST).
- Set the Resource path to / (root path).
- Click Create.

## 6. Deploy the API:

- Click on Deployments in the left sidebar.
- Click Create and enter a stage name, such as prod.
- Click Deploy.

## 7. Get the API URL:

*After deploying, note the Invoke URL. This URL will be similar to:*

`https://abcd1234.execute-api.us-east-1.amazonaws.com`

`https://abcd1234.execute-api.us-east-1.amazonaws.com`

## Step 3: Test the API

### 1. Access the URL in Your Browser:

Open your browser and enter the API URL with the query parameter code:

<https://abcd1234.execute-api.us-east-1.amazonaws.com/?course=hpcsa>

<http-api.us-east-1.amazonaws.com/?course=hpcsa>

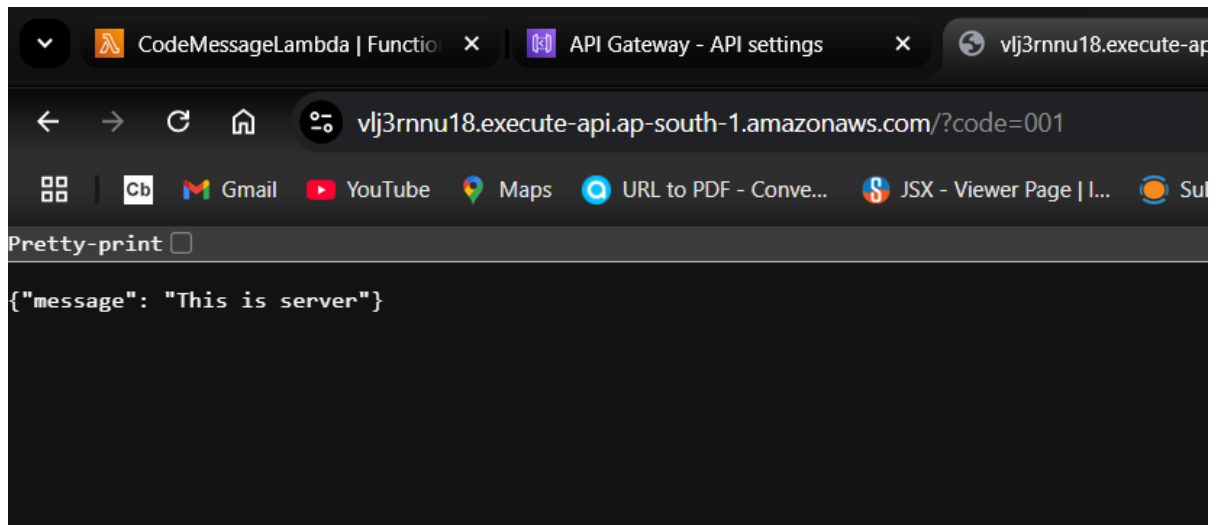
## 2. Expected Output:

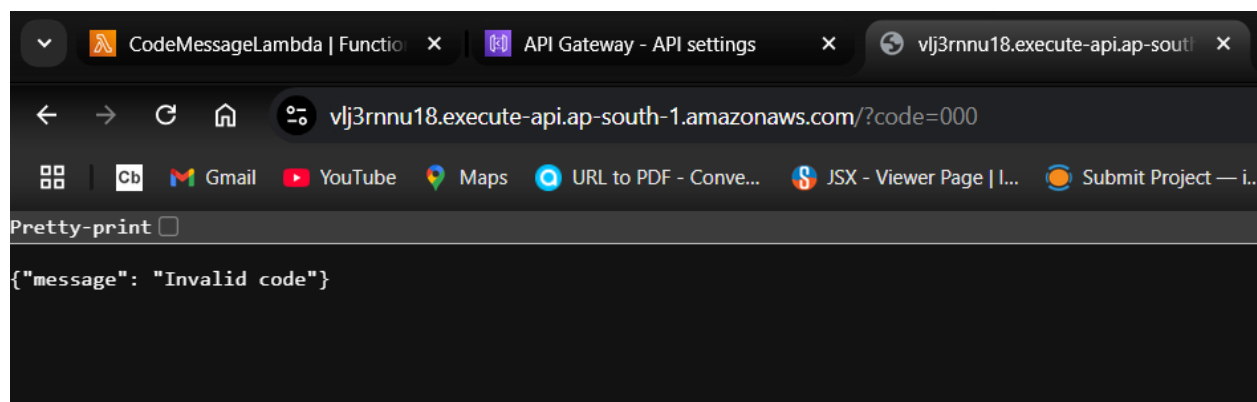
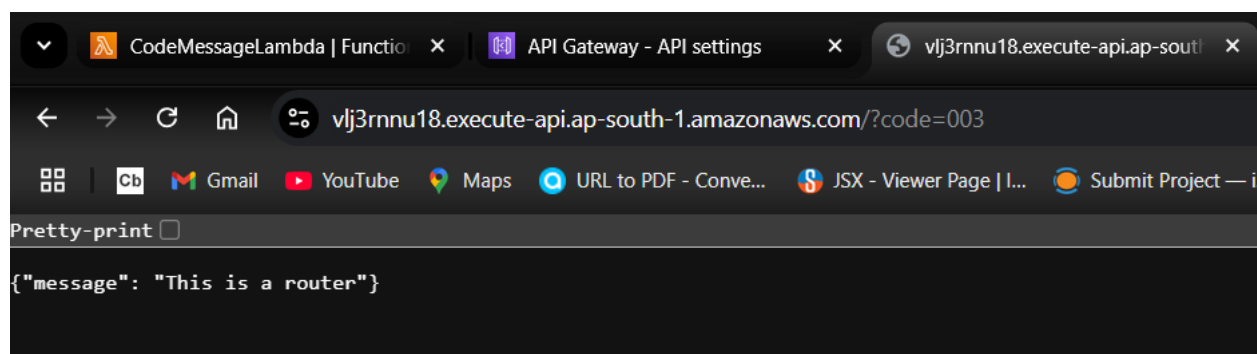
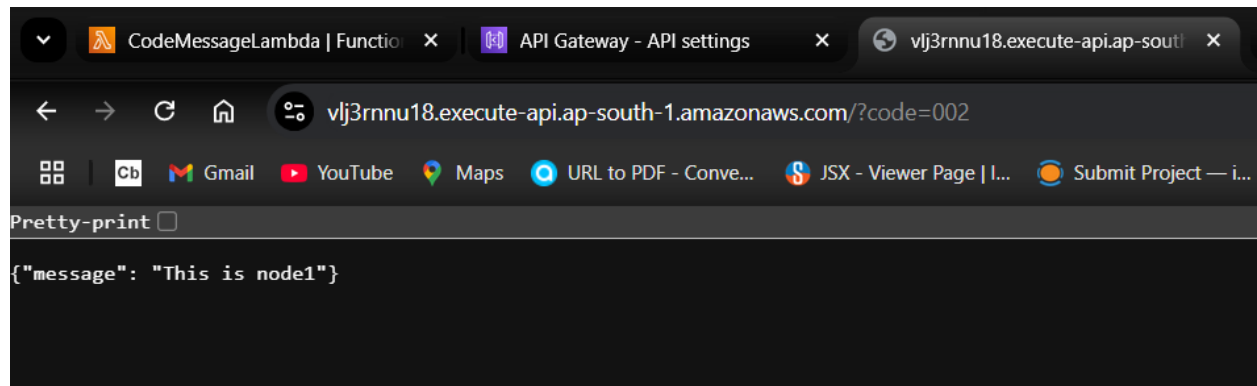
You should see a JSON response similar to

```
{  
  "Welcome to HPCSA course"  
}
```

```
{  
  "Welcome to HPCSA course"  
}
```

## Final Output Screenshots





**Q2. Create a Lambda function with URL. In the URL user will define course=. If the course=hpcsa then display “Welcome to HPCSA course” message. Similarly it should display messages for ditiss, dbda course.**

**Access function URL and display the output for all courses.**

**All the implementation steps are same as Assig-1**

**-----Only the code part is change to this-----**

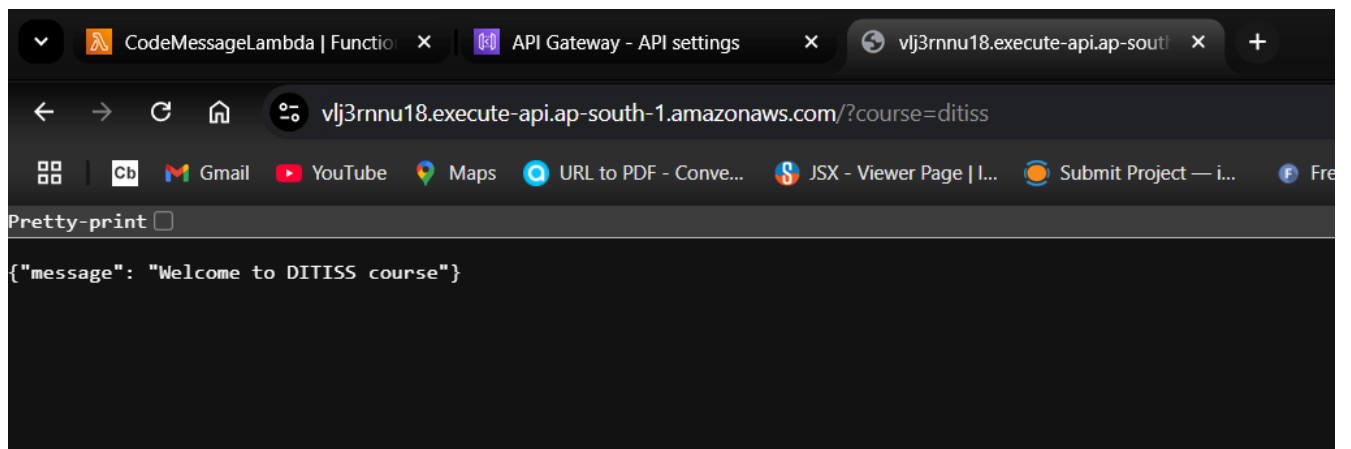
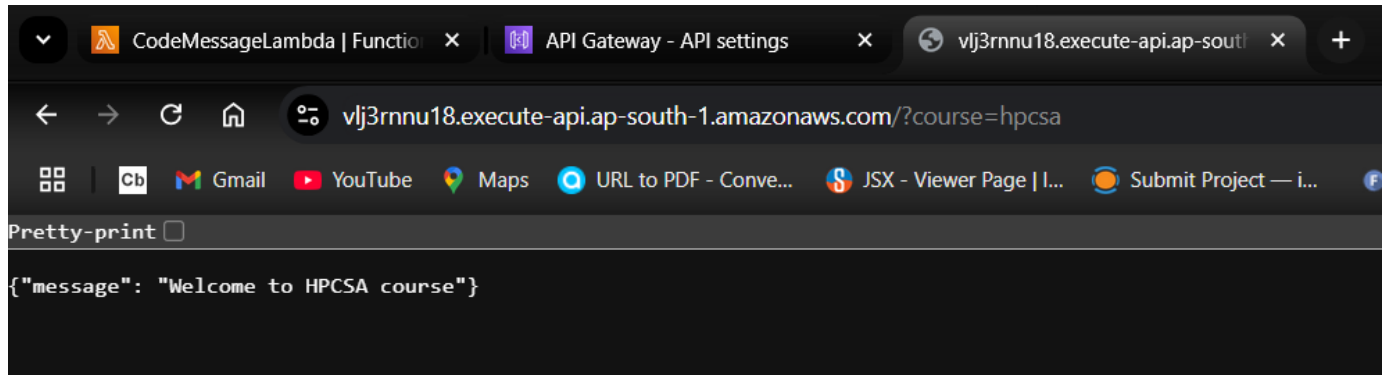
```
import json

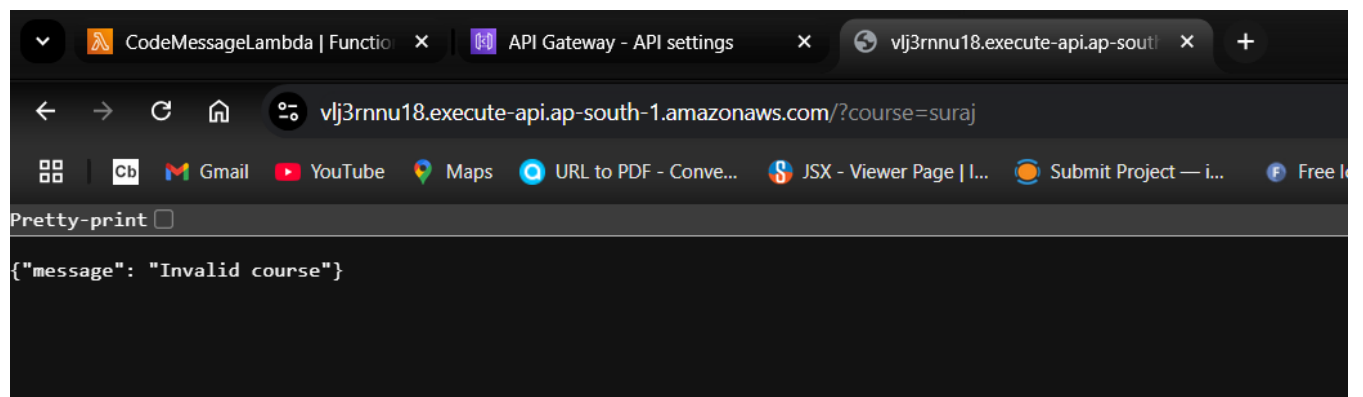
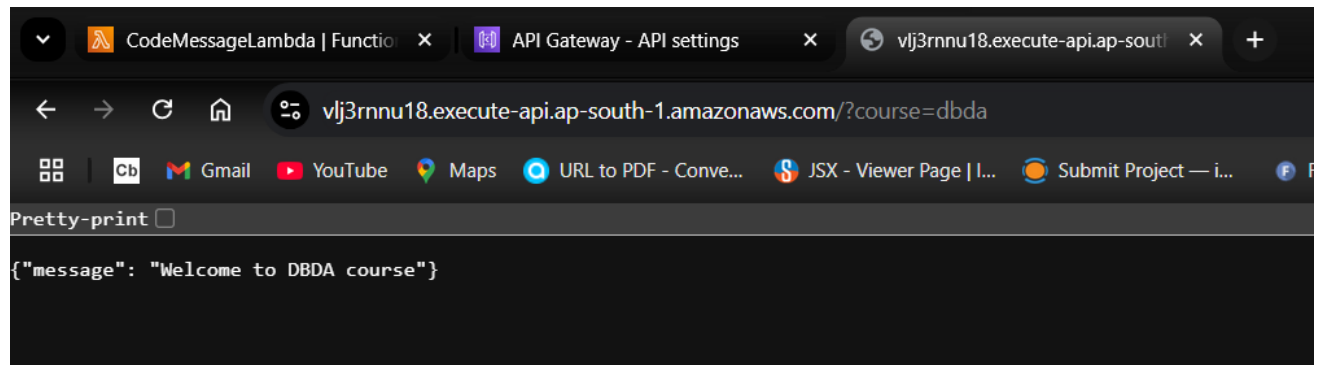
def lambda_handler(event, context):
    # Get the 'course' parameter from the query string
    course = event.get('queryStringParameters',
    {}).get('course', '')

    # Define messages based on the course parameter
    if course == "hpcsa":
        message = "Welcome to HPCSA course"
    elif course == "ditiss":
        message = "Welcome to DITIIS course"
    elif course == "dbda":
        message = "Welcome to DBDA course"
    else:
        message = "Invalid course"

    # Return the response as JSON
    return {
        'statusCode': 200,
        'body': json.dumps({
            'message': message
        }),
        'headers': {
            'Content-Type': 'application/json'
        }
    }
```

## Final Output Screenshots







**Q3. Create a Lambda function with URL. In the URL user will define course=. If the course=hpcs then display "Welcome to HPCS course" message. Similarly it should display messages for ditiss, dbda course.**

**Access function URL and display the output for all courses.**

***Following are the steps to implement the Lambda function***

## **Step 1: Create IAM Users**

### **1. Create User Mike (S3 Full Access):**

- Go to IAM Console.
- Click Users on the left sidebar and then Add user.
- Username: Mike. -Access Type: Select Programmatic access and AWS Management Console access (optional).
- To set a password: Choose Auto-generated password or set a custom password.
- If setting a custom password, ensure the "User must create a new password at next sign-in" box is selected (recommended).
- In the Permissions section, select AmazonS3FullAccess.
- Click Next: Tags, then Next: Review, and finally Create user.
- Save the credentials for Mike.

### **2. Create User Sam (S3 Read-Only Access):**

- Go to IAM Console > Users > Add user.
- Username: Sam
- Access Type: Select Programmatic access and AWS Management Console access (optional).
- Set a password similarly as described for Mike.
- In the Permissions section, select AmazonS3ReadOnlyAccess.
- Click Next: Tags, then Next: Review, and finally Create user.
- Save the credentials for Sam.

## **Step 2: Create an S3 Bucket**

- Create S3 Bucket:
- Go to the S3 Console.

- Click Create bucket.
- Set a unique bucket name (e.g., mike-sam-test-bucket-12345).
- Choose a region (e.g., us-east-1).
- Leave other settings as default and click Create bucket.

### Step 3: Test Access for Mike (S3 Full Access)

- Login as Mike:
- Use the login URL for Mike (from the .csv file).
- Check Bucket Access:
- Once logged in, navigate to the S3 Console.
- Verify Mike can see and access the bucket mike-sam-test-bucket-12345.
- Mike should be able to upload files to the bucket since he has full S3 access.

### Step 4: Test Access for Sam (S3 Read-Only Access)

- Login as Sam:
- Use the login URL for Sam (from the .csv file).
- Check Bucket Access:
- Once logged in, navigate to the S3 Console.
- Verify Sam can see and access the bucket mike-sam-test-bucket-12345.
- Sam can view and download files, but will not see the Upload button and cannot upload files due to read-only permissions.

### Step 5: Verify Permissions

- Mike (Full Access) can upload, delete, and view files in the bucket.
- Sam (Read-Only) can only view and download files, but cannot upload files.

### Summary:

- Mike can upload files (Full Access).
- Sam can only view and download files (Read-Only).
- *The user can only access the services and resources for which permissions have been granted.*
- *In this example, the user has access to only S3 but not to EC2, Lambda or other services unless explicitly allowed.*

## 1. Mike and sam user are created.

The screenshot shows the AWS IAM console interface. A green notification banner at the top states "User created successfully" with a "View user" button. The left sidebar shows the "Identity and Access Management (IAM)" menu. The main content area displays the "Users (2) info" section, which includes a search bar and a table of users.

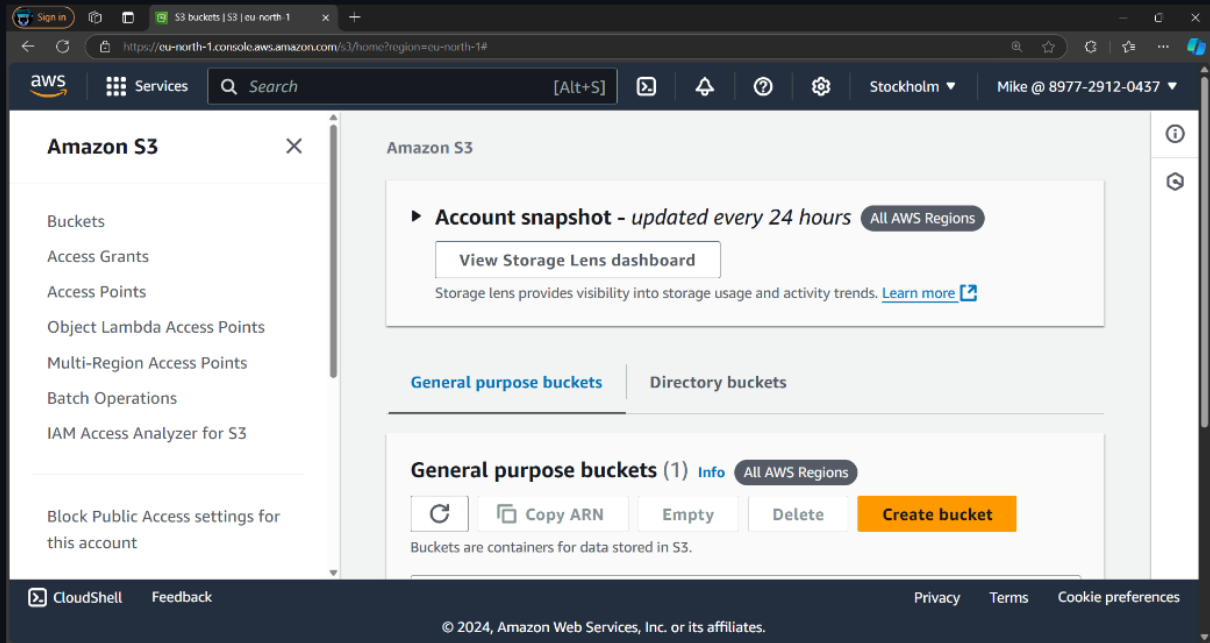
	User name	Path	Groups	Last activity	MFA	Password age	Console last sign-in
<input type="checkbox"/>	Mike	/	0	-	-	-	-
<input type="checkbox"/>	Sam	/	0	-	-	-	-

## 2. S3 (Simple Storage Service) Bucket is created.

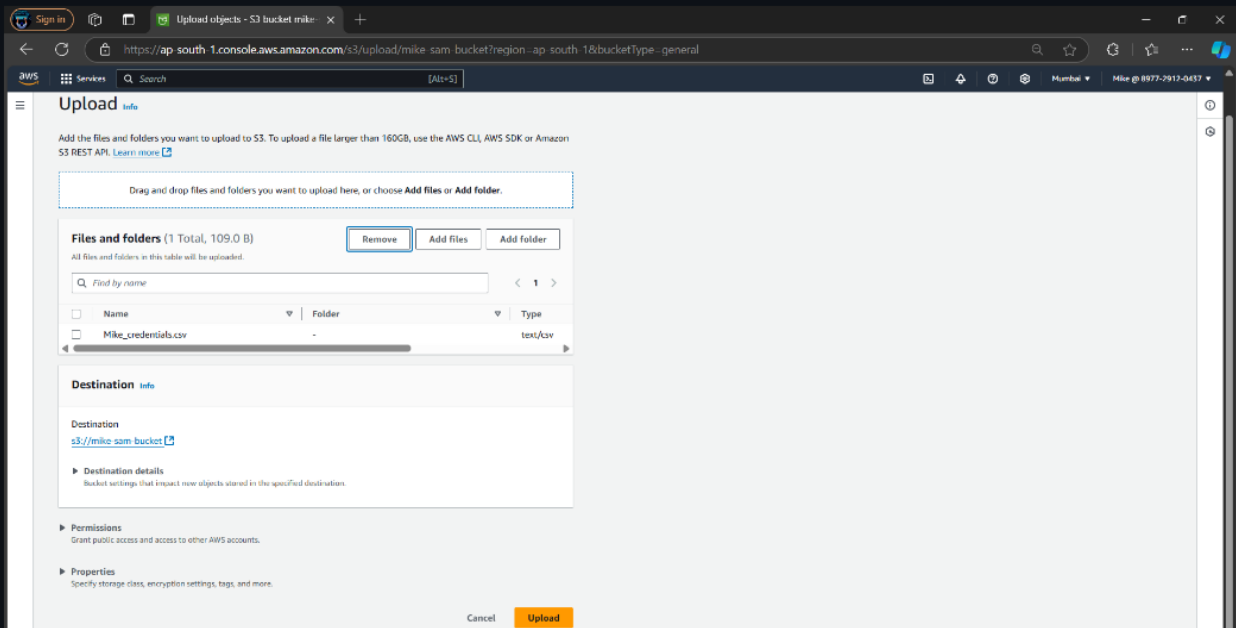
The screenshot shows the AWS S3 console interface. A green notification banner at the top states "Successfully created bucket 'mike-sam-test-bucket'" with a "View details" button. The left sidebar shows the "Amazon S3" menu. The main content area displays the "General purpose buckets (1) info" section, which includes a search bar and a table of buckets.

	Name	AWS Region	IAM Access Analyzer	Creation date
<input type="radio"/>	mike-sam-test-bucket	Asia Pacific (Mumbai) ap-south-1	<a href="#">View analyzer for ap-south-1</a>	November 7, 2024, 20:45:05 (UTC+05:30)

### 3. Login as mike user using there credintials.



### 4.\* Files uploaded to the S3 Bucket by mike user. As he have full access of S3 Bucket.\*



### 5. Files uploaded to the S3 Bucket Successfully by mike

The screenshot shows the AWS Management Console interface. At the top, a green banner indicates 'Upload succeeded' with a link to 'View details below.' Below this, the 'Upload: status' page is displayed. A light blue box contains a warning: 'The information below will no longer be available after you navigate away from this page.' The 'Summary' section shows the upload details: Destination is 's3://mike-sam-bucket', Succeeded is '1 file, 109.0 B (100.00%)', and Failed is '0 files, 0 B (0%)'. Below the summary, there are tabs for 'Files and folders' and 'Configuration'. The 'Files and folders' tab is active, showing a list of files. The list has a header with columns: Name, Folder, Type, Size, Status, and Error. One file is listed: 'Mike\_creden...' with a size of 109.0 B and a status of 'Succeeded'. The bottom of the screen shows the Windows taskbar with the date 11/7/2024 and time 9:59 PM.

Upload: status

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

**Summary**

Destination	Succeeded	Failed
s3://mike-sam-bucket	1 file, 109.0 B (100.00%)	0 files, 0 B (0%)

**Files and folders** (1 Total, 109.0 B)

Name	Folder	Type	Size	Status	Error
Mike_creden...	-	text/csv	109.0 B	Succeeded	-

### 6. Now Login as sam user using there credentials.

The screenshot shows the AWS Management Console interface for the 'sam' user. The top navigation bar shows the user is logged in as 'Sam @ 8977-2912-0437'. The main content area is titled 'Amazon S3'. It features a section for 'Account snapshot - updated every 24 hours' with a link to 'View Storage Lens dashboard'. Below this, there are tabs for 'General purpose buckets' and 'Directory buckets'. The 'General purpose buckets' tab is active, showing a list of buckets. The list has a header with columns: Name, Folder, Type, Size, Status, and Error. One bucket is listed: 'Mike\_creden...' with a size of 109.0 B and a status of 'Succeeded'. The bottom of the screen shows the Windows taskbar with the date 11/7/2024 and time 10:03 PM.

Amazon S3

**Account snapshot - updated every 24 hours** All AWS Regions View Storage Lens dashboard

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

**General purpose buckets** Directory buckets

**General purpose buckets (1)** Info All AWS Regions

[Refresh](#) [Copy ARN](#) [Empty](#) [Delete](#) [Create bucket](#)

Buckets are containers for data stored in S3.

[Find buckets by name](#)

7. sam user Trying to uplod file in the S3 bucket but unable to uplod because sam have read only permission of S3 Bucket.

The screenshot shows the AWS Management Console interface for the 'mike-sam-bucket'. A red banner at the top indicates 'Upload failed' with a message to 'View details below.' Below this, the 'Upload: status' section shows a summary of the upload attempt. The destination is 's3://mike-sam-bucket'. The summary shows 0 files succeeded and 1 file failed (228.0 B, 100.00%). The 'Files and folders' section shows a table with one entry: 'mike login h...' of type 'text/plain' and size '228.0 B', with a status of 'Failed' and an error message 'Access Denied'.

Name	Folder	Type	Size	Status	Error
mike login h...	-	text/plain	228.0 B	Failed	Access Denied

8. S3 Bucket of only file which are uplod by mike user.

The screenshot shows the AWS Management Console interface for the 'mike-sam-bucket'. The 'Objects' tab is selected, showing a list of objects. There is one object named 'Mike\_credentials.csv' of type 'csv', last modified on November 7, 2024, at 22:02:36 (UTC+05:30), with a size of 109.0 B and a storage class of 'Standard'.

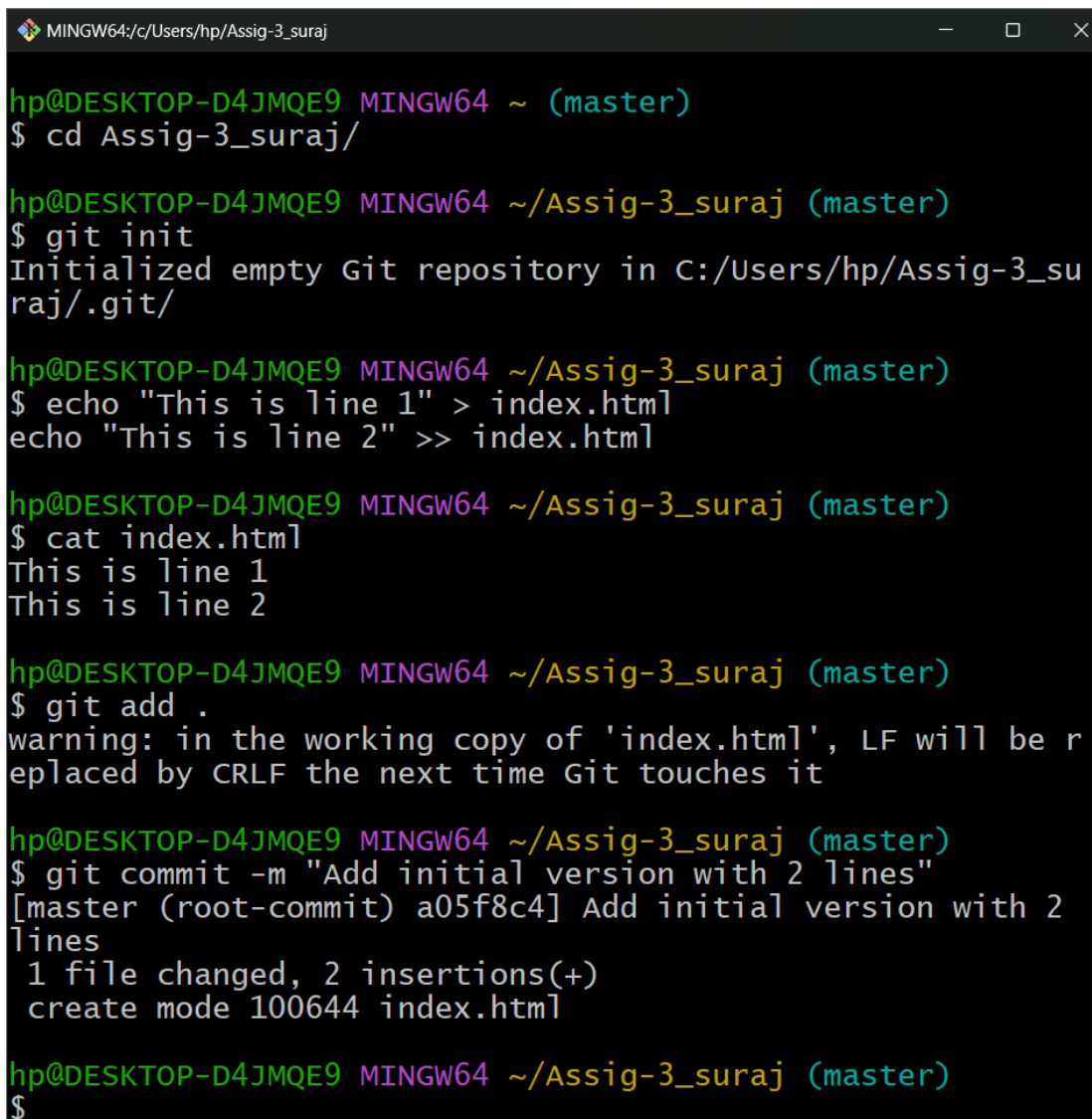
Name	Type	Last modified	Size	Storage class
Mike_credentials.csv	csv	November 7, 2024, 22:02:36 (UTC+05:30)	109.0 B	Standard

**Q4. Create a local git repository. Create a file by name index.html. Type 2 lines in it. Perform commit. Modify index.html file and add 2 more lines. Perform commit. Modify the file again and add 2 more line. Restore the earlier version of the file with 4 lines. Modify file again and 3 lines. Perform commit. Restore 4 line version of the file.**

**Step 1: Create a Local Git Repository**

**Step 2: Create index.html File and Add Initial Content**

**Step 3: Stage and Commit the Initial Version**



```
MINGW64: c:/Users/hp/Assig-3_suraj

hp@DESKTOP-D4JMQE9 MINGW64 ~ (master)
$ cd Assig-3_suraj/

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git init
Initialized empty Git repository in C:/Users/hp/Assig-3_suraj/.git/

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ echo "This is line 1" > index.html
echo "This is line 2" >> index.html

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git add .
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git commit -m "Add initial version with 2 lines"
[master (root-commit) a05f8c4] Add initial version with 2 lines
1 file changed, 2 insertions(+)
create mode 100644 index.html

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$
```

#### Step 4: Modify index.html and Add 2 More Lines. Stage and commit this version

```
MINGW64~/c/Users/hp/Assig-3_suraj
hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git add .
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git commit -m "Add initial version with 2 lines"
[master (root-commit) a05f8c4] Add initial version with 2 lines
1 file changed, 2 insertions(+)
create mode 100644 index.html

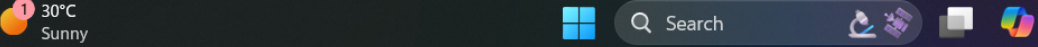
hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ echo "This is line 3" >> index.html
echo "This is line 4" >> index.html

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git add .
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git commit -m "Add 2 more lines to index.html"
[master aa6905e] Add 2 more lines to index.html
1 file changed, 2 insertions(+)

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ echo "This is line 5" >> index.html
echo "This is line 6" >> index.html

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git add index.html
git commit -m "Add 2 more lines to index.html"
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
[master 7e46f12] Add 2 more lines to index.html
```





## Step 5: Modify index.html Again and Add 2 More Lines

```
MINGW64:/c/Users/hp/Assig-3_suraj

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git log
commit 7e46f12f56c1c96d4ffa65a3efb5b8197d88b9a3 (HEAD -> master)
Author: Suraj Kumar <csuraj982@gmail.com>
Date:   Sun Nov 10 16:38:23 2024 +0530

    Add 2 more lines to index.html

commit aa6905e40bf63bbab8a6af214b7f4323db08372e
Author: Suraj Kumar <csuraj982@gmail.com>
Date:   Sun Nov 10 16:37:44 2024 +0530

    Add 2 more lines to index.html

commit a05f8c445c6353b52cc042ab8a4539c94bf015c1
Author: Suraj Kumar <csuraj982@gmail.com>
Date:   Sun Nov 10 16:36:22 2024 +0530

    Add initial version with 2 lines

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$
```

## Step 6: Restore the Earlier Version (4-Line Version)

```
MINGW64: c/Users/hp/Assig-3_suraj
hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ ^[[200~git checkout aa6905e40b index.html
bash: $'\E[200~git': command not found

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git checkout aa6905e40b index.html
Updated 1 path from cd03f49

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ |
```

Stage and commit this restored version.

```
MINGW64: c:/Users/hp/Assig-3_suraj
hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ echo "This is line 7" >> index.html
echo "This is line 8" >> index.html
echo "This is line 9" >> index.html

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4
This is line 7
This is line 8
This is line 9

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git add index.html
git commit -m "Add 3 more lines to index.html"
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
[master df7b635] Add 3 more lines to index.html
1 file changed, 3 insertions(+)

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git checkout aa6905e40b index.html
Updated 1 path from cd03f49

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$
```

## Step 7: Modify the File Again and Add 3 New Lines

```
MINGW64; c:/Users/hp/Assig-3_suraj

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4
This is line 7
This is line 8
This is line 9

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git add index.html
git commit -m "Add 3 more lines to index.html"
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
[master df7b635] Add 3 more lines to index.html
1 file changed, 3 insertions(+)


hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git checkout aa6905e40b index.html
Updated 1 path from cd03f49

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ git checkout commit_hash index.html
git add index.html
git commit -m "Restore 4-line version of index.html again"
error: pathspec 'commit_hash' did not match any file(s) known to git
[master 202d725] Restore 4-line version of index.html again
1 file changed, 3 deletions(-)

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ cat index.html
This is line 1
This is line 2
This is line 3
This is line 4

hp@DESKTOP-D4JMQE9 MINGW64 ~/Assig-3_suraj (master)
$ |
```

 1 Air: Moderate Now

