**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

code = event.get('queryStringParameters', {}).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"*

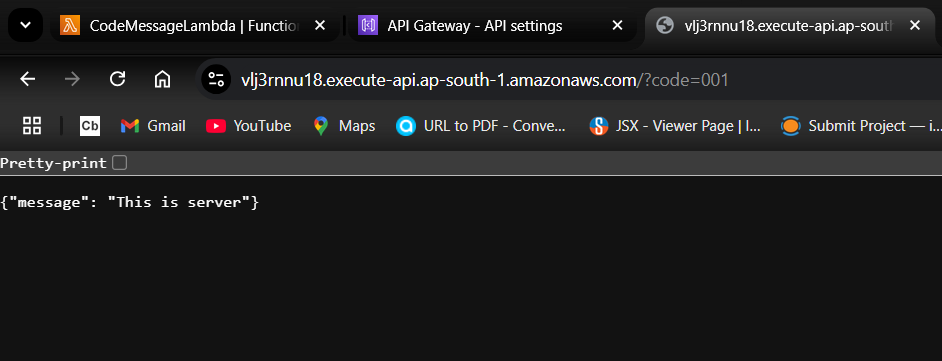
*}*

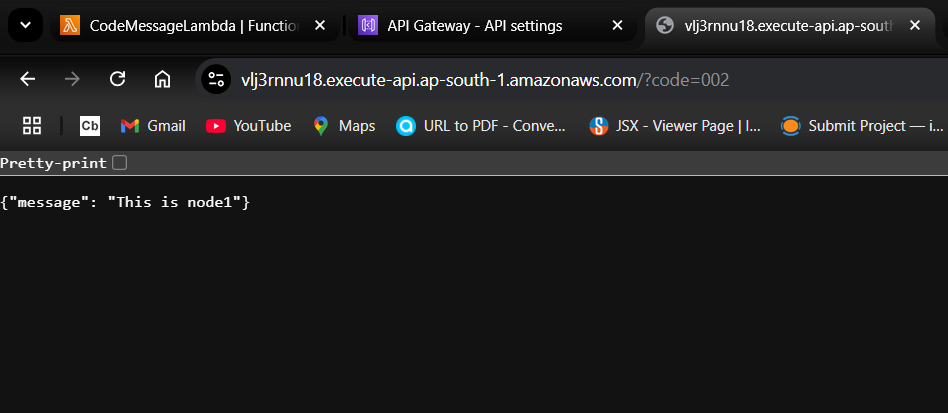
{

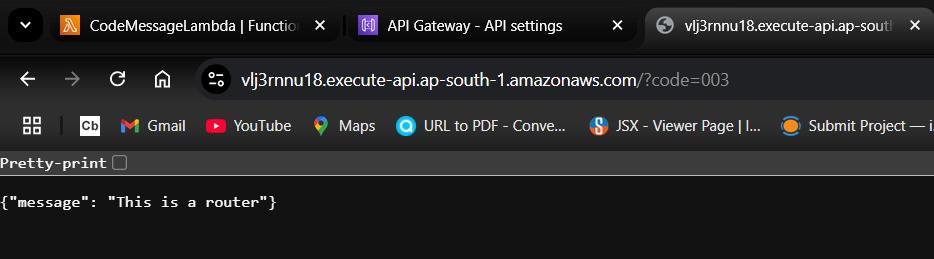
"Welcome to HPCSA course"

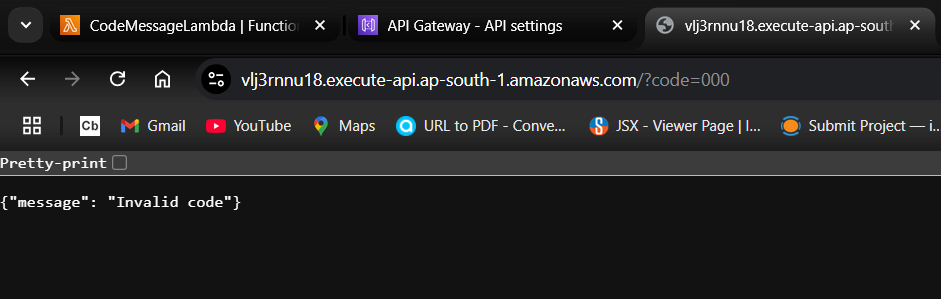
}

**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 DITISS 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

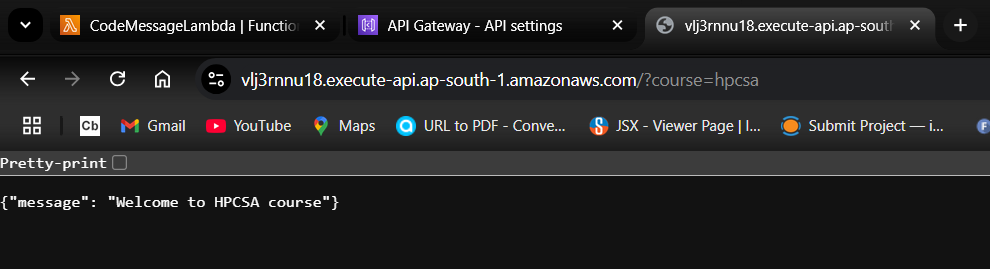
}),

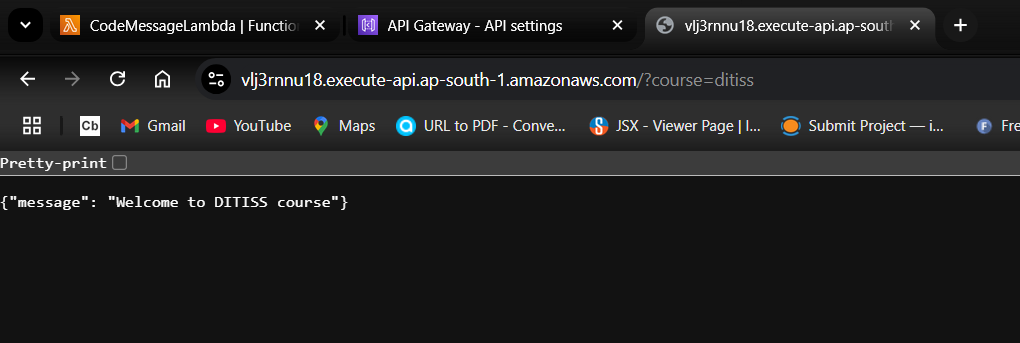
'headers': {

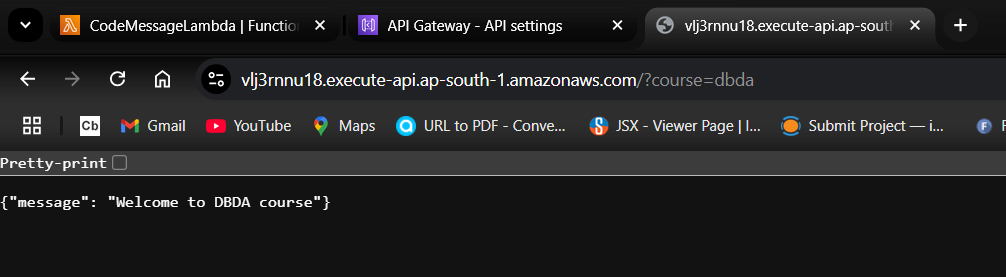
'Content-Type': 'application/json'

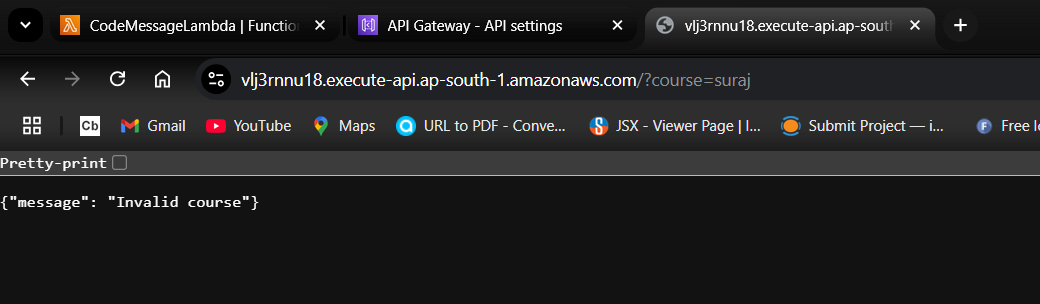
}

**Final Output Screenshots**









**Q3. Create a Lambda function with URL. In the URL user will define course=. If the course=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.**

***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)

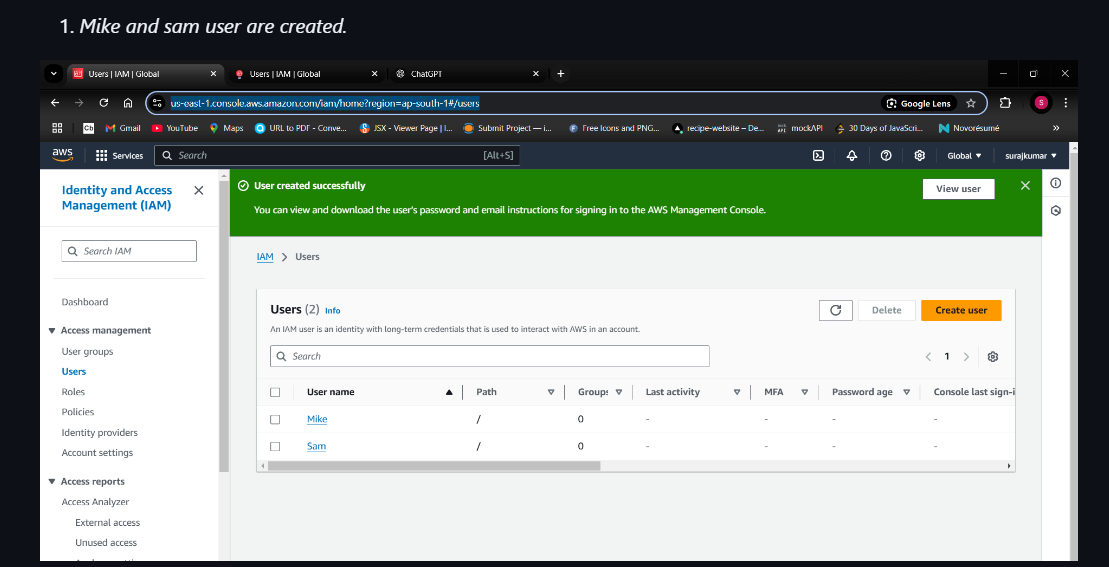
* Login as Sam:
* Use the login URL for Sam (from the .csv file).
* Check Bucket Access:
* Once logged in, navigate to the S3 Console.
* Verify Sam can see and access the bucket mike-sam-test-bucket-12345.
* Sam can view and download files, but will not see the Upload button and cannot upload files due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

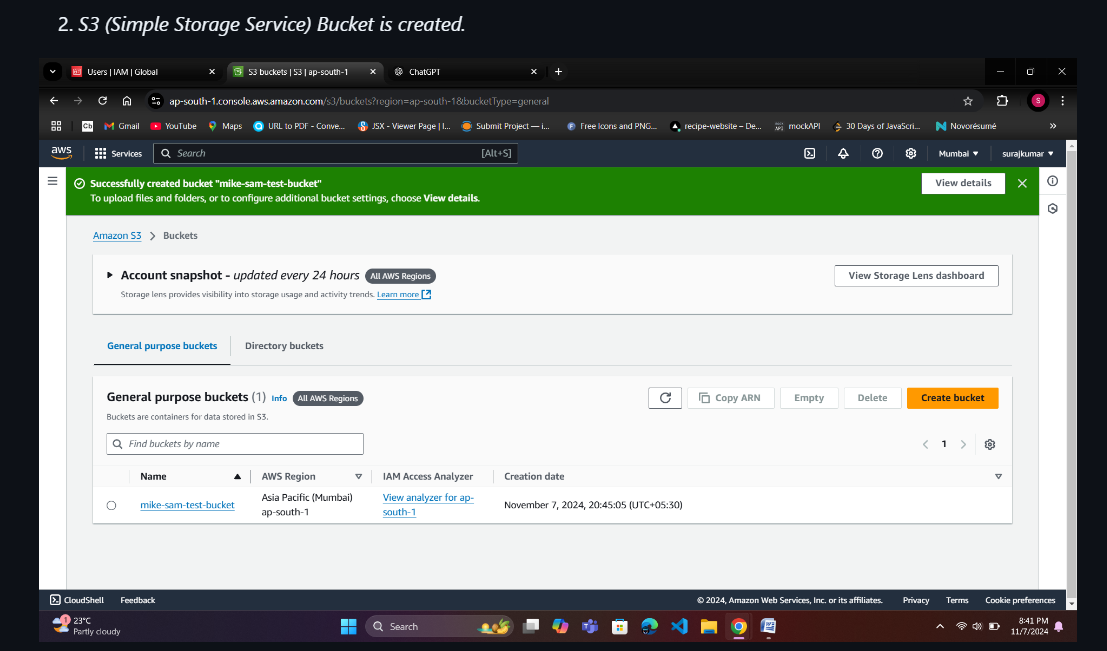
### Step 5: Verify Permissions

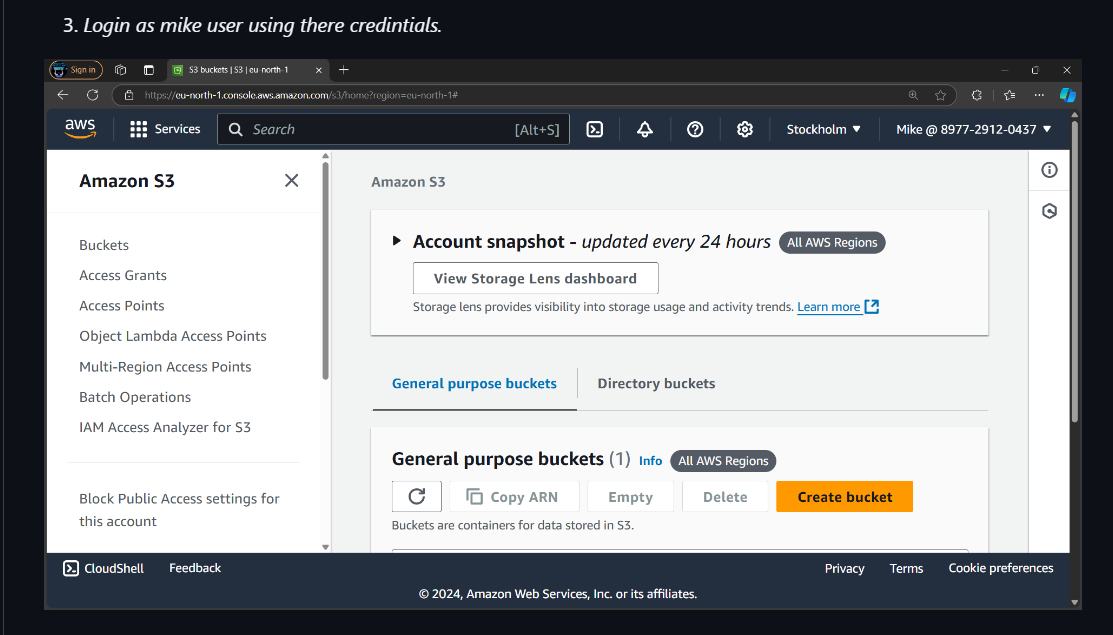
* Mike (Full Access) can upload, delete, and view files in the bucket.
* Sam (Read-Only) can only view and download files, but cannot upload files.

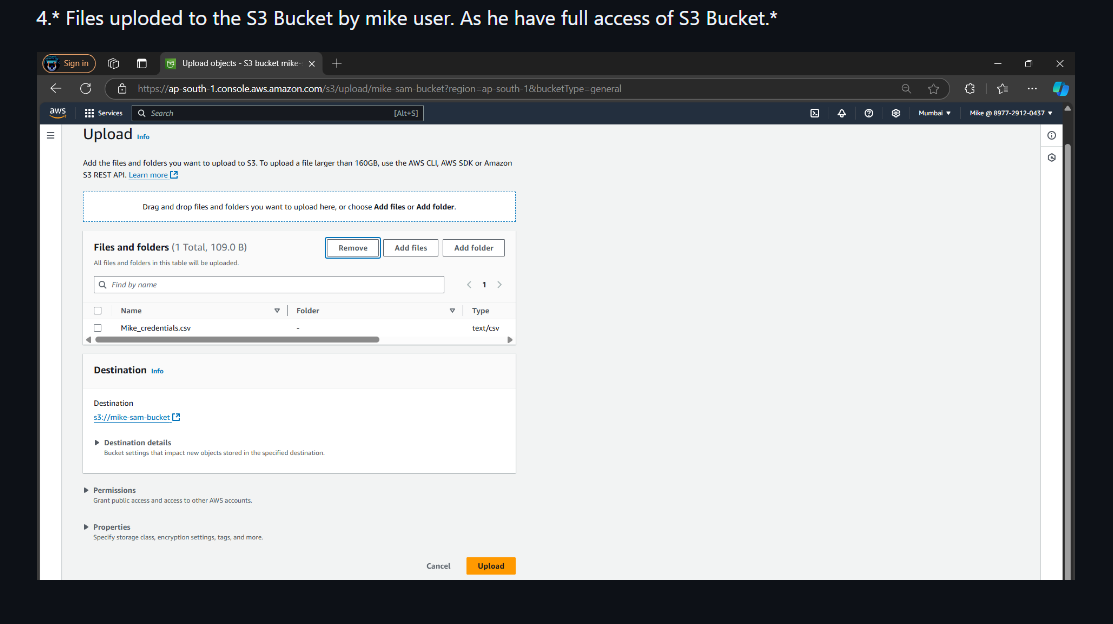
## Summary:

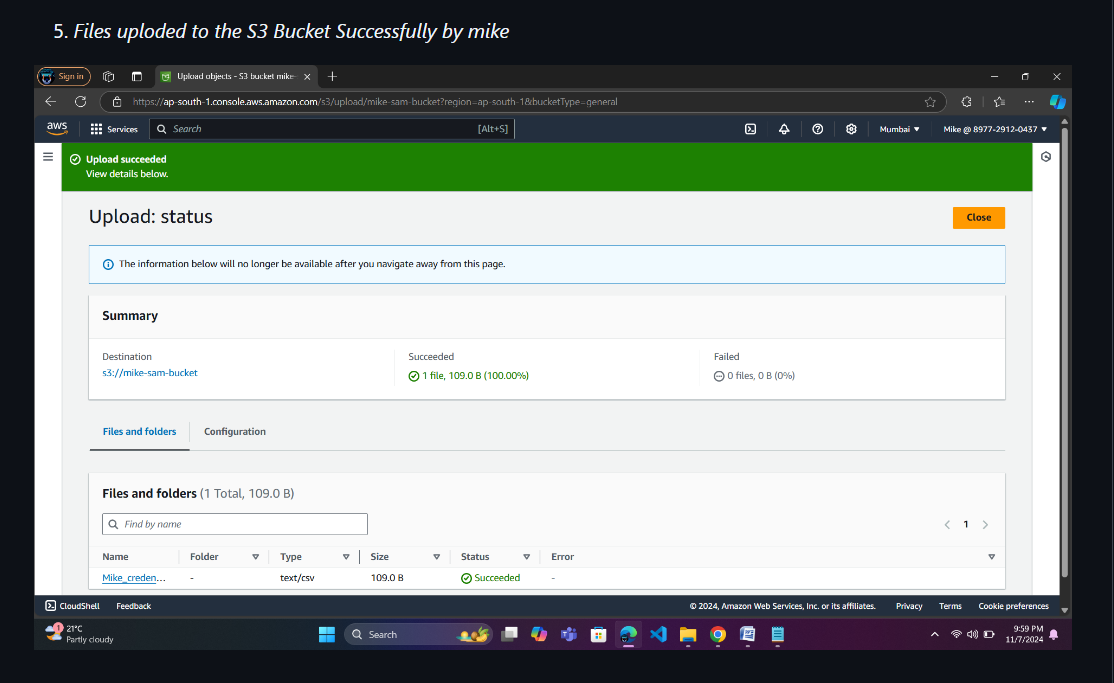
* Mike can upload files (Full Access).
* Sam can only view and download files (Read-Only).
* The user can only access the services and resources for which permissions have been granted.
* In this example, the user has access to only S3 but not to EC2, Lambda or other services unless explicitly allowed.

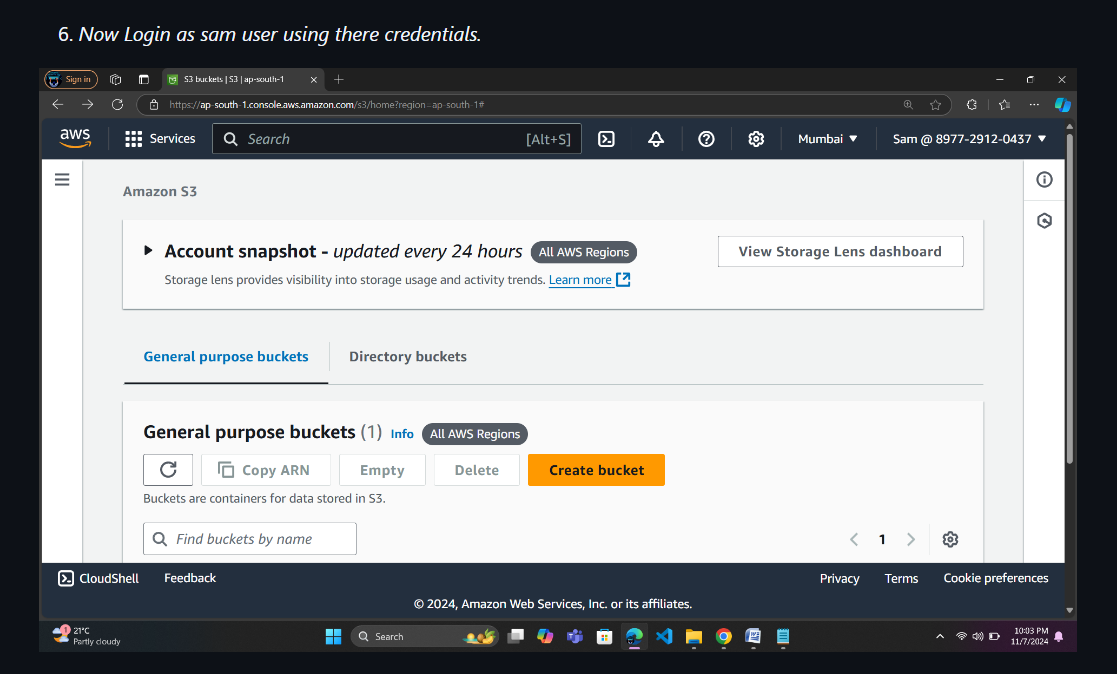


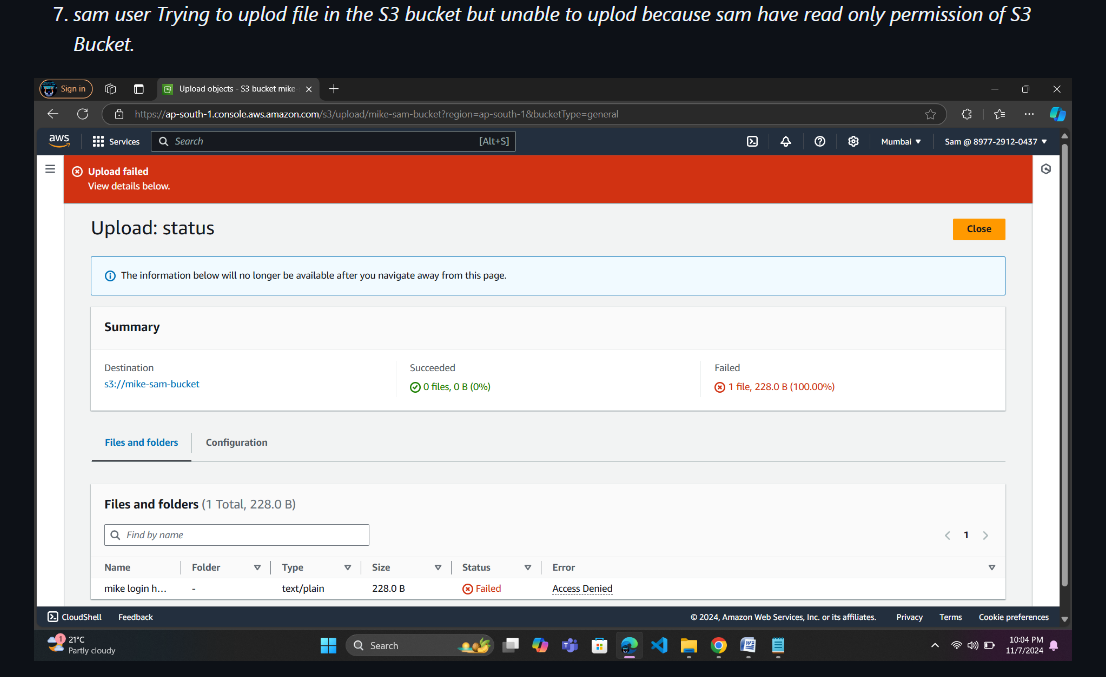


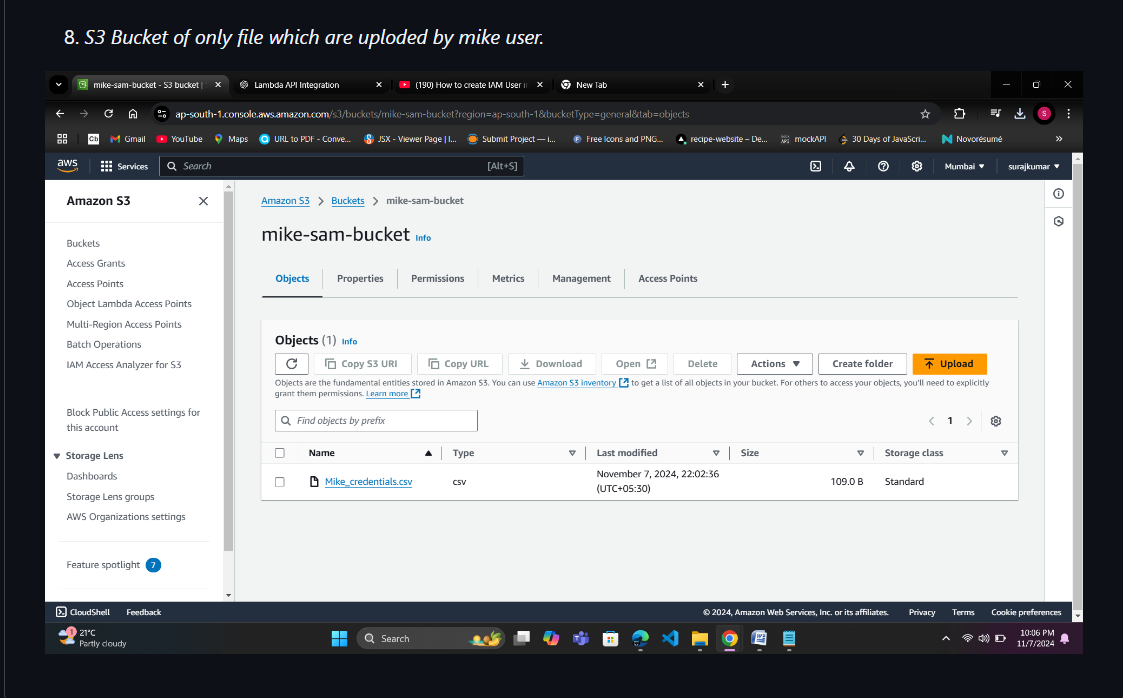










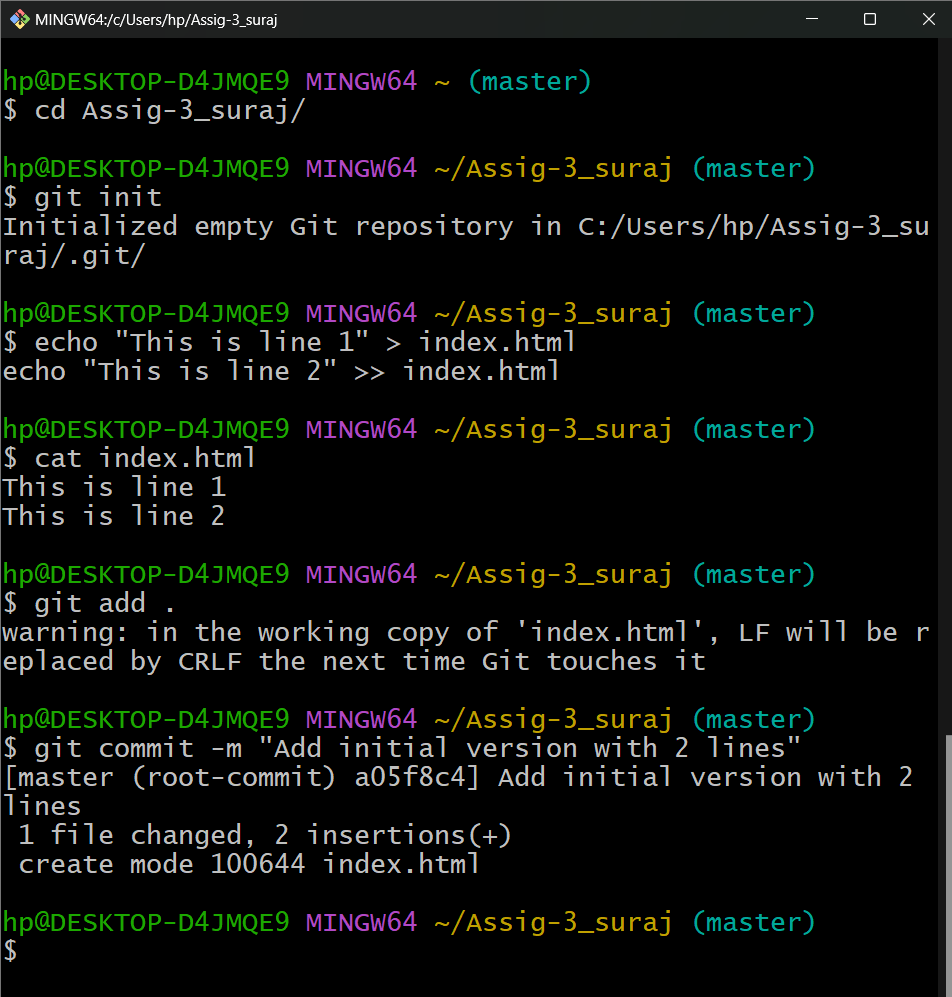


**Q4. Create a local git repository. Create a file by name index.html. Type 2 lines in it. Perform commit. Modify index.html file and add 2 more lines. Perform commit. Modify the file again and add 2 more line. Restore the earlier verison of the file with 4 lines. Modify file again and 3 lines. Perform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

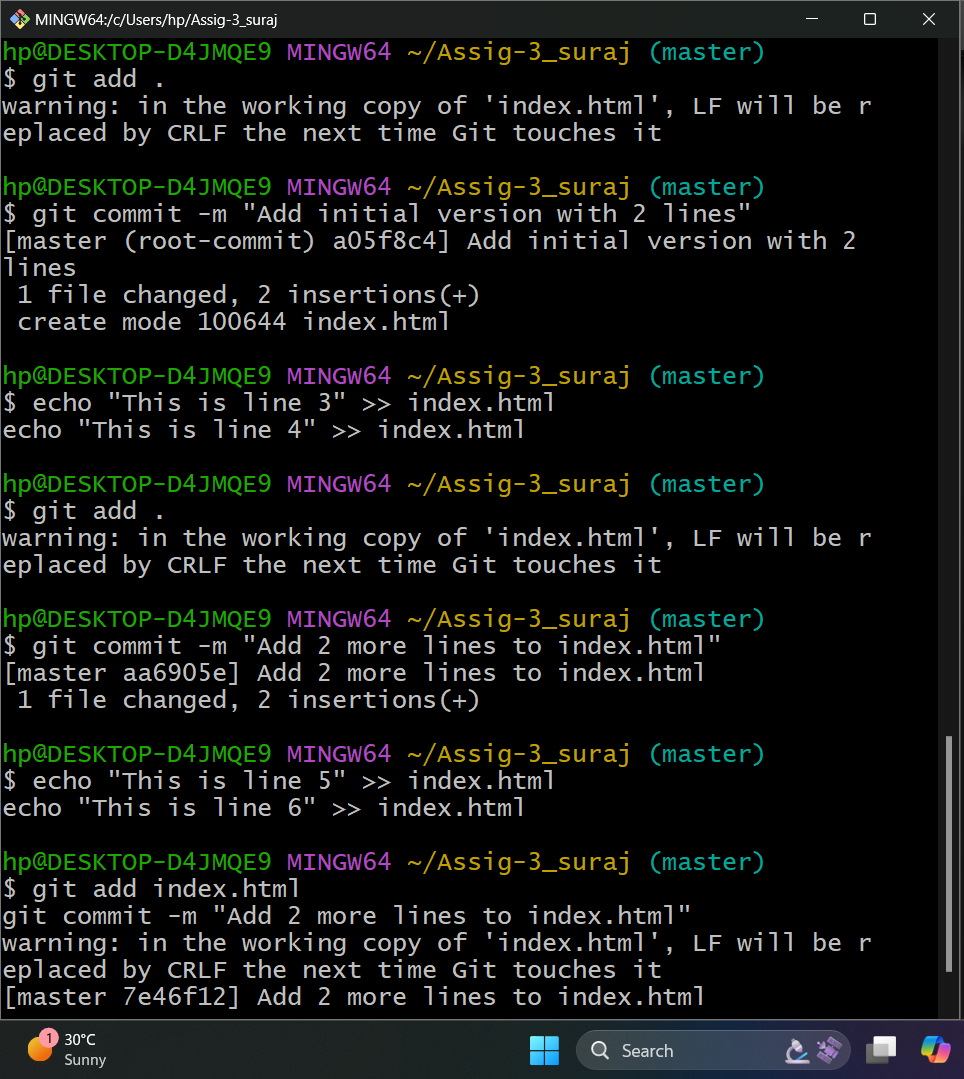
### Step 1: Create a Local Git Repository

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

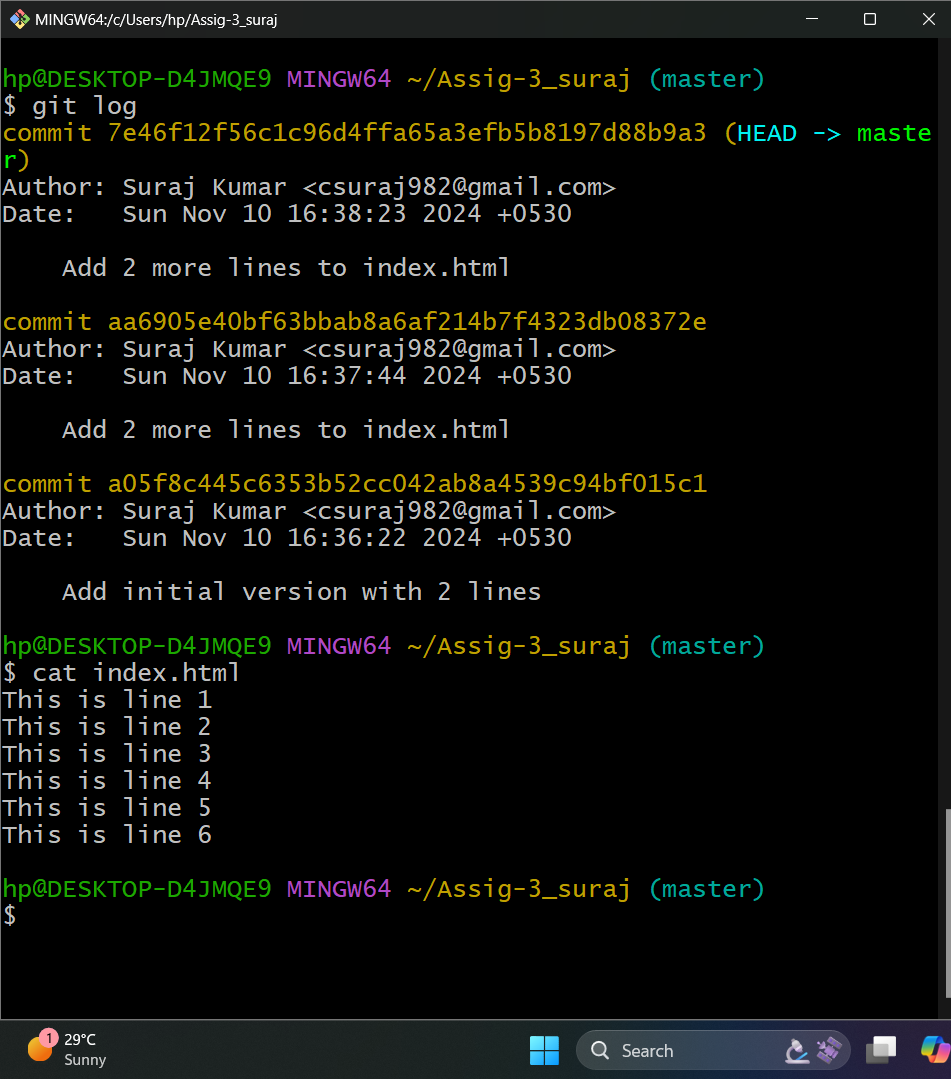
**Step 3: Stage and Commit the Initial Version**



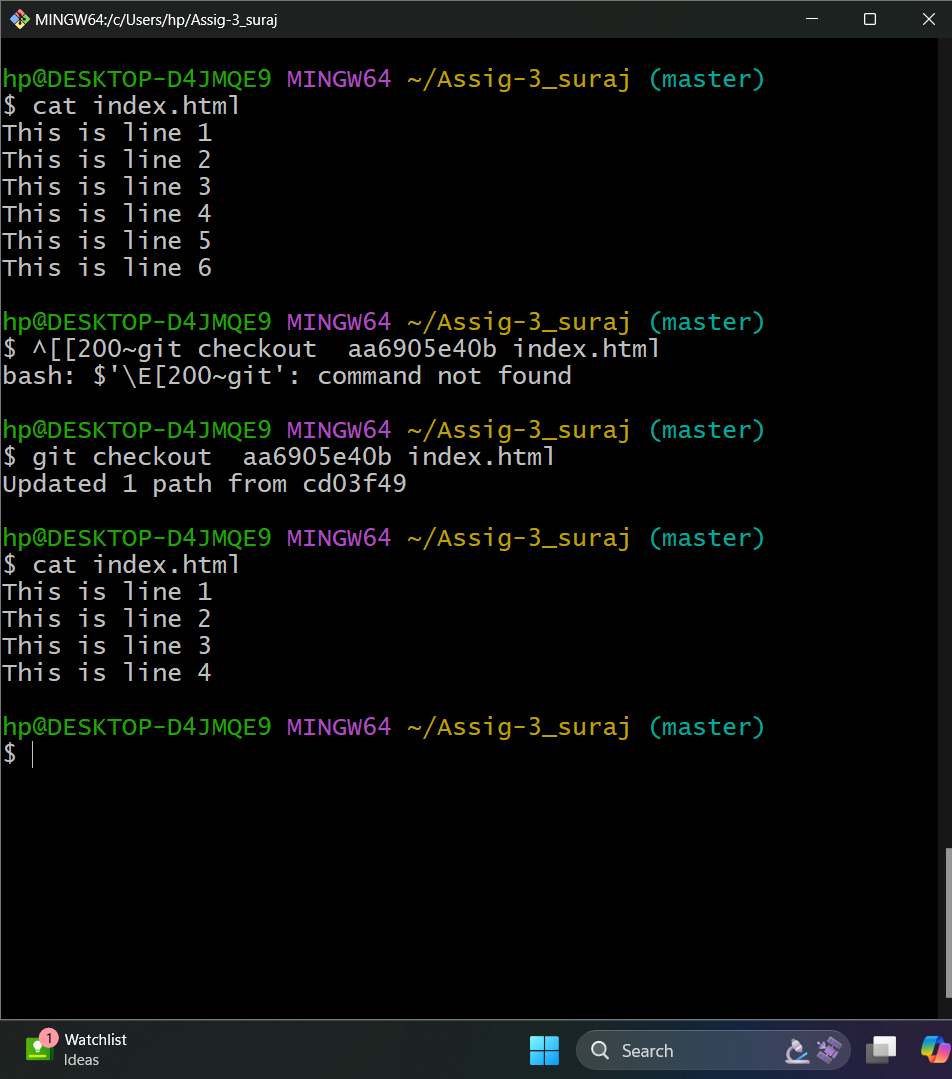
**Step 4: Modify index.html and Add 2 More Lines. Stage and commit this version**

**.** 

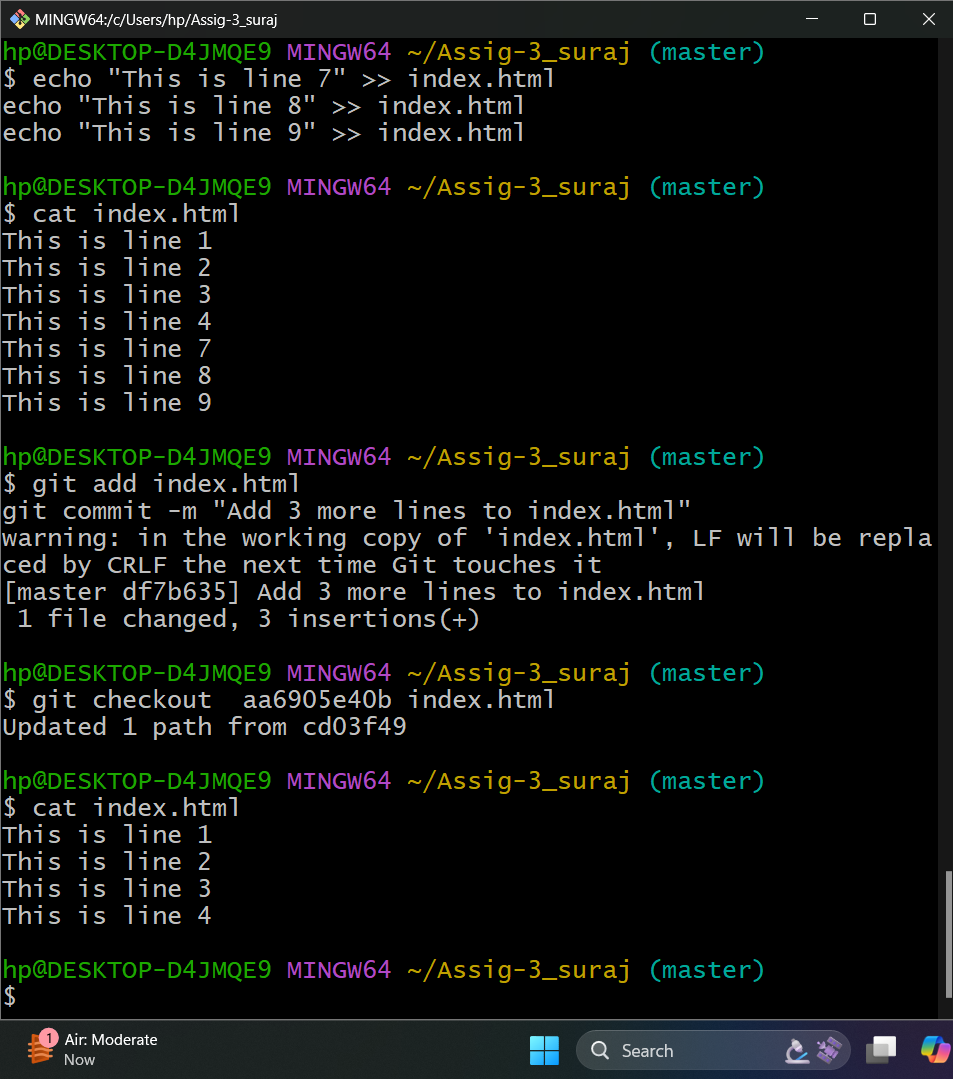
**Step 5: Modify index.html Again and Add 2 More Lines**



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



**Stage and commit this restored version.**



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

