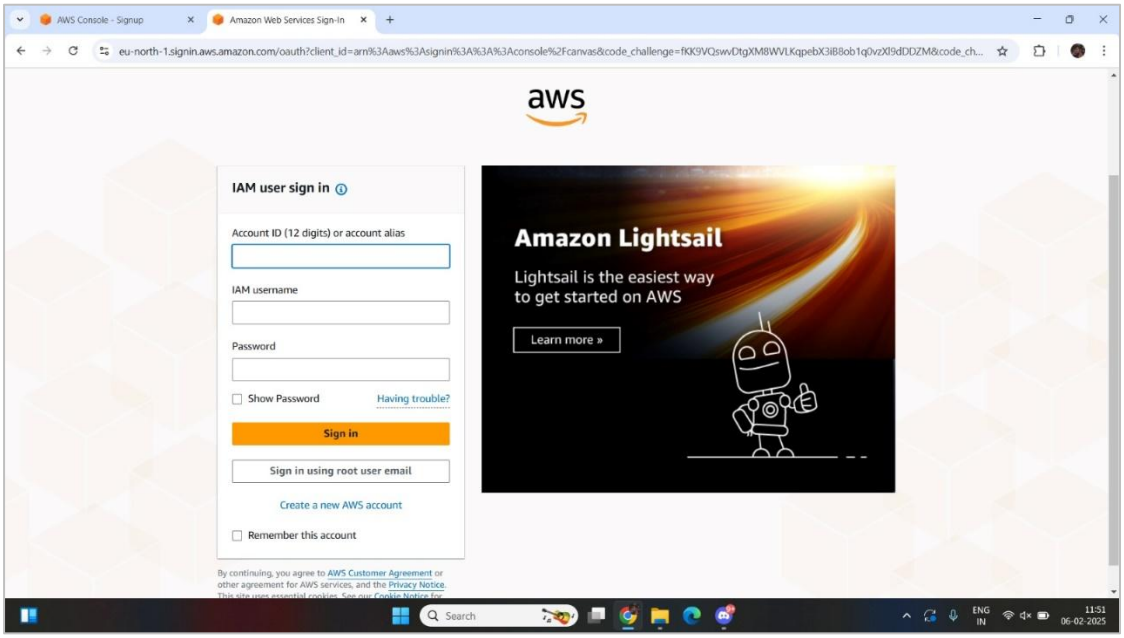
ASSIGNMENT 1:

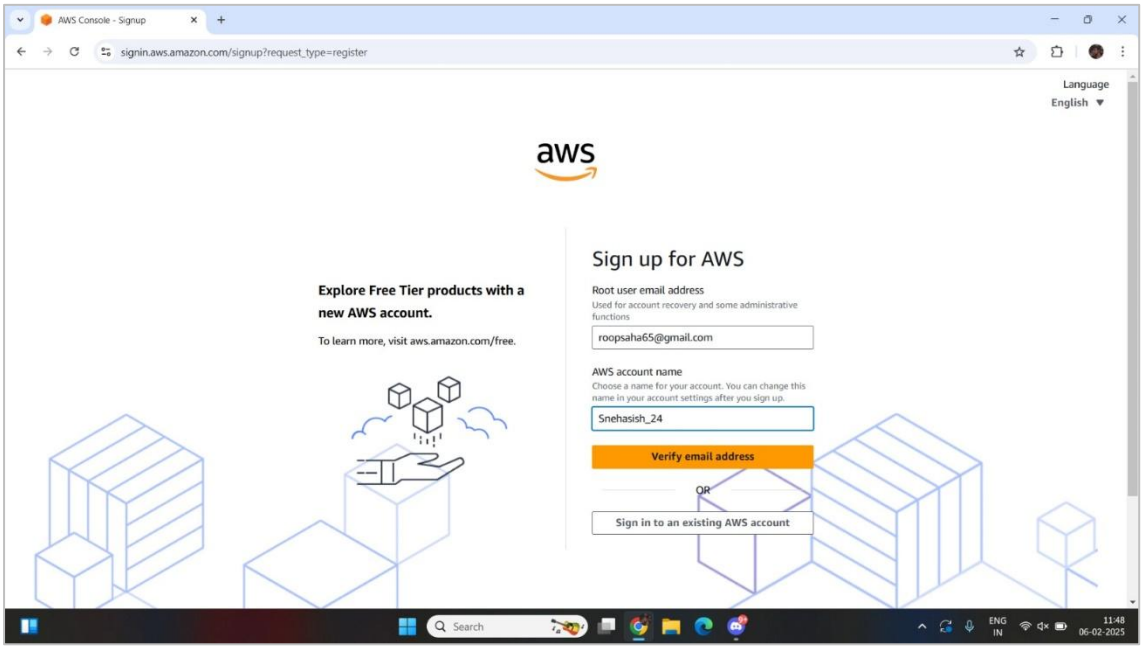
Problem Definition: Create an account in AWS and configure a budget

Account Creation:

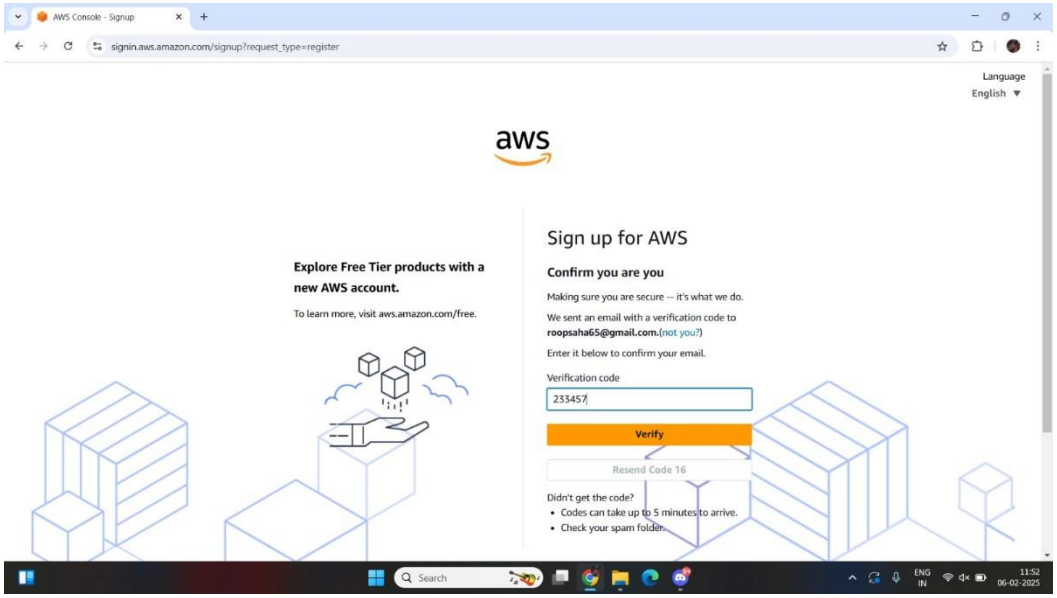
1. Go to ***AWS Management Console*** and click on "***Sign Up***".



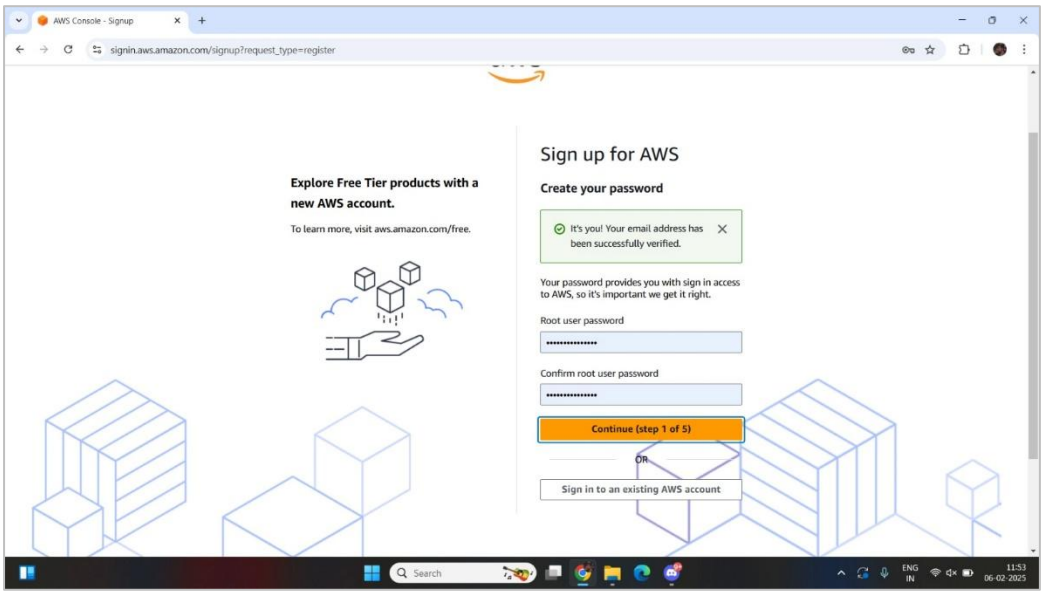
2. Enter *root user email address* and *AWS account name*.



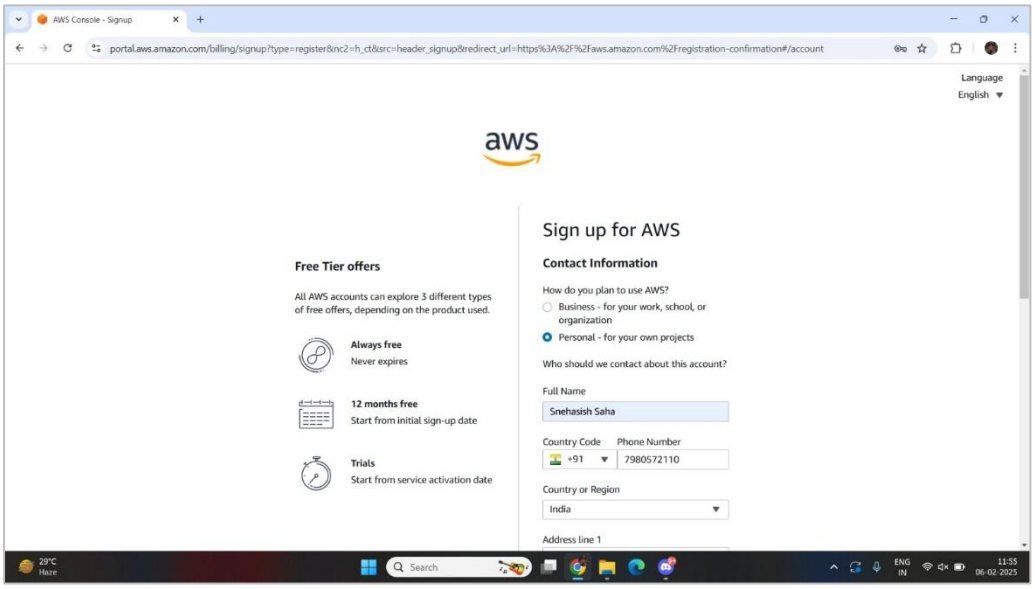
3. Verify your email address.



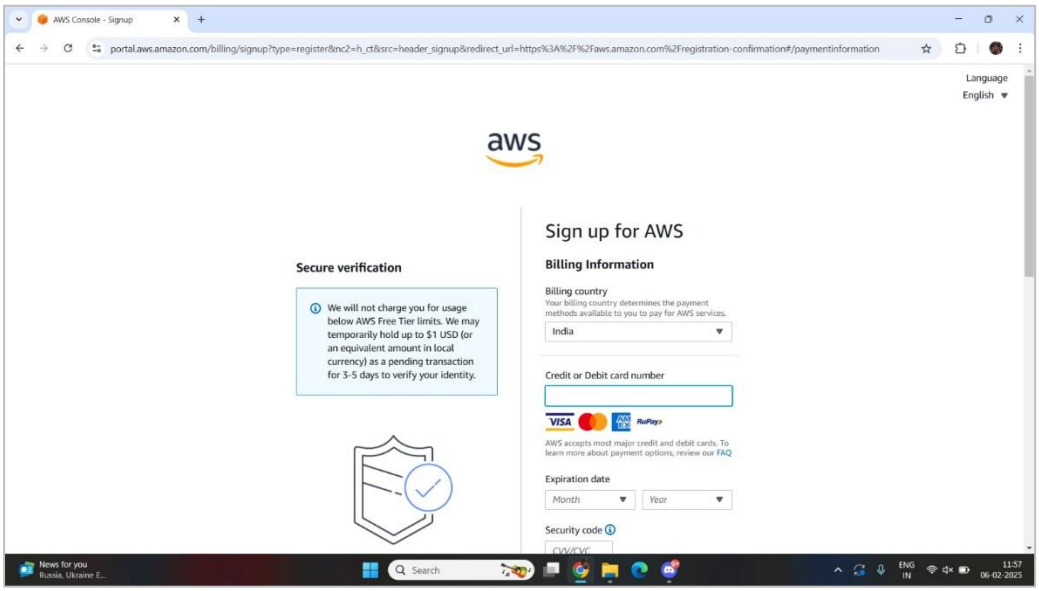
4. Set your password, then click on "***Continue (step 1 of 5)***".



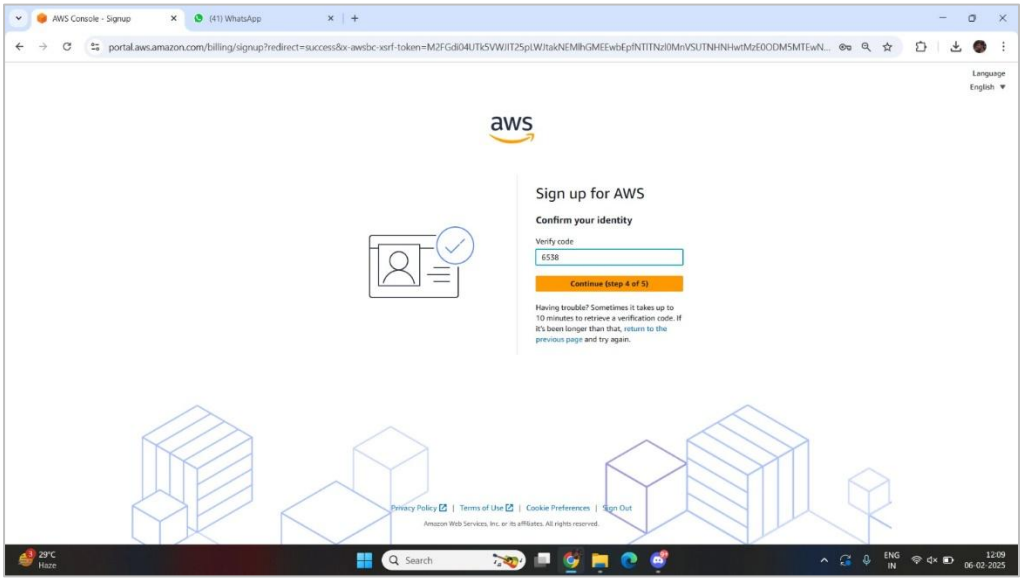
5. Fill the required details, tick off the check box and then click on ***"Continue (step 2 of 5)"***.



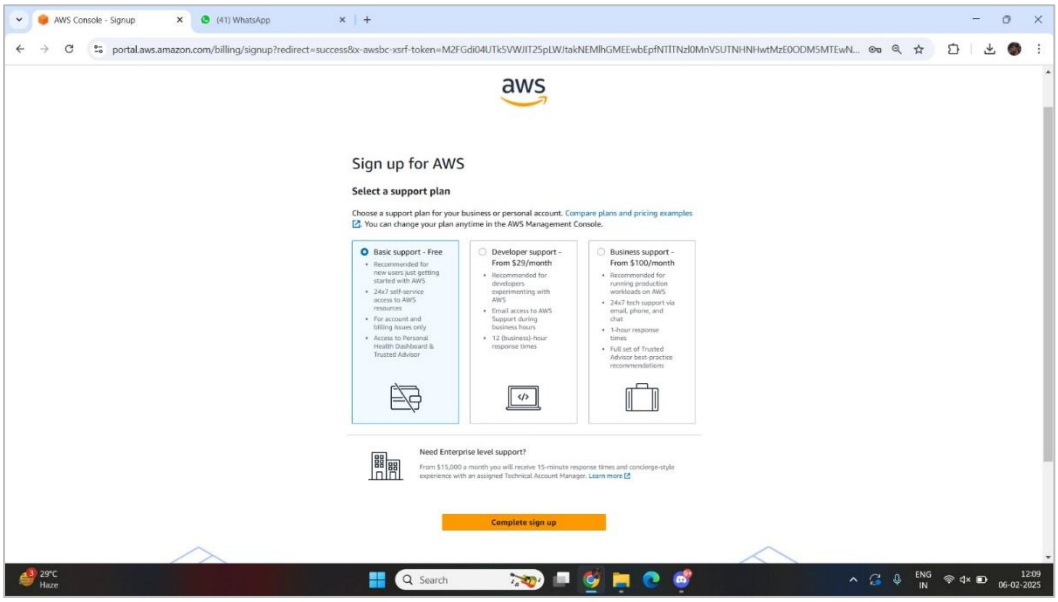
6. Enter billing information and complete payment, then click on ***"Continue (step 3 of 5)"***.



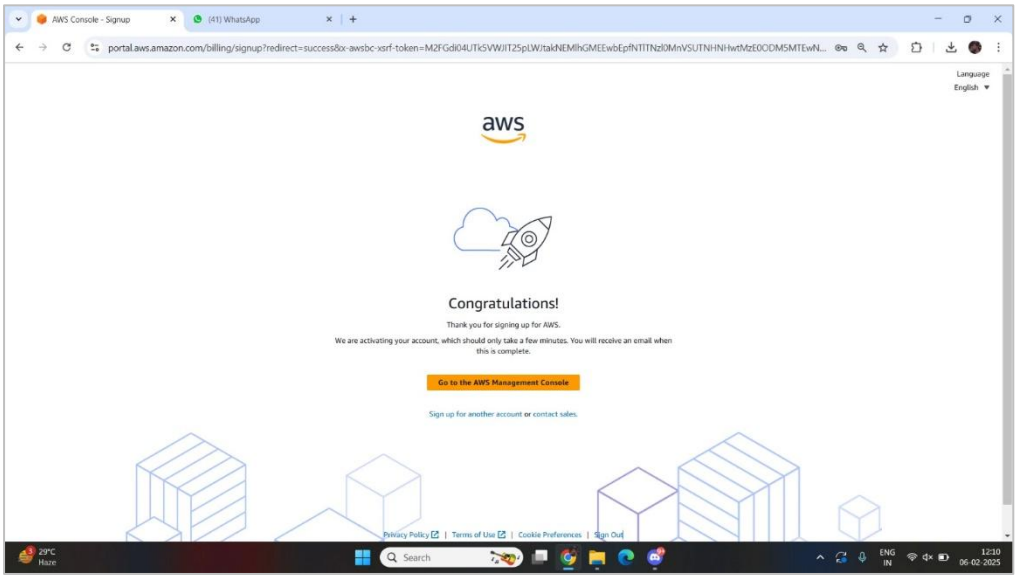
7. Verify your phone number then click ***"Continue (step 4 of 5)"***.



8. Select a *Support Plan* then click on **"Complete sign up"**.

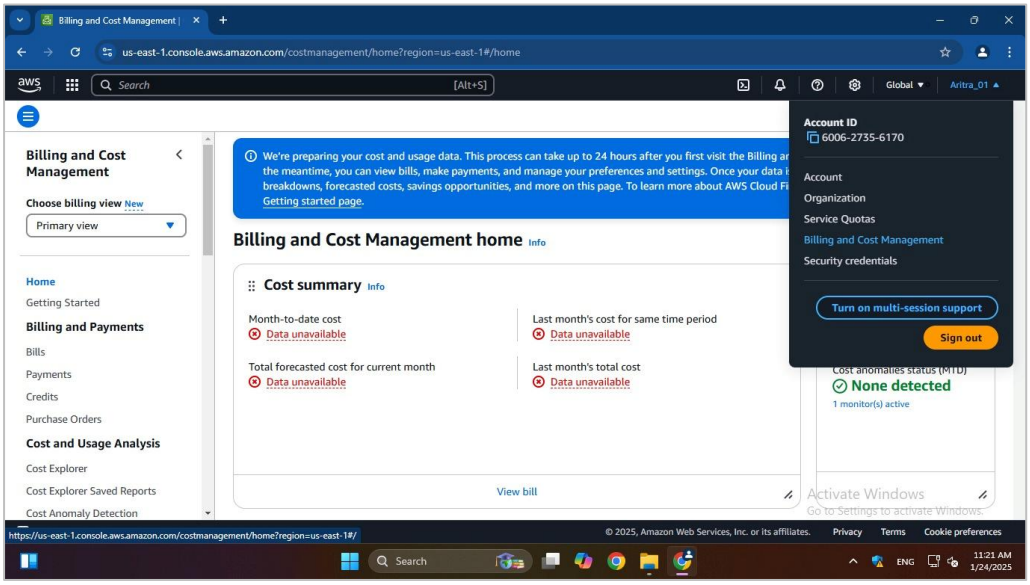


9. Congratulation page will be shown if account is created successfully.

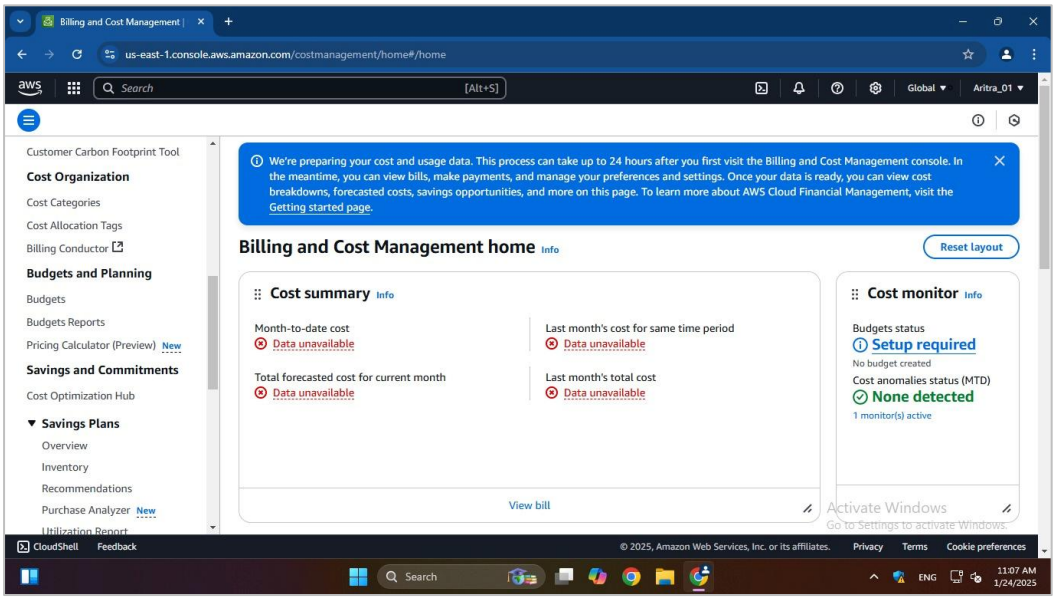


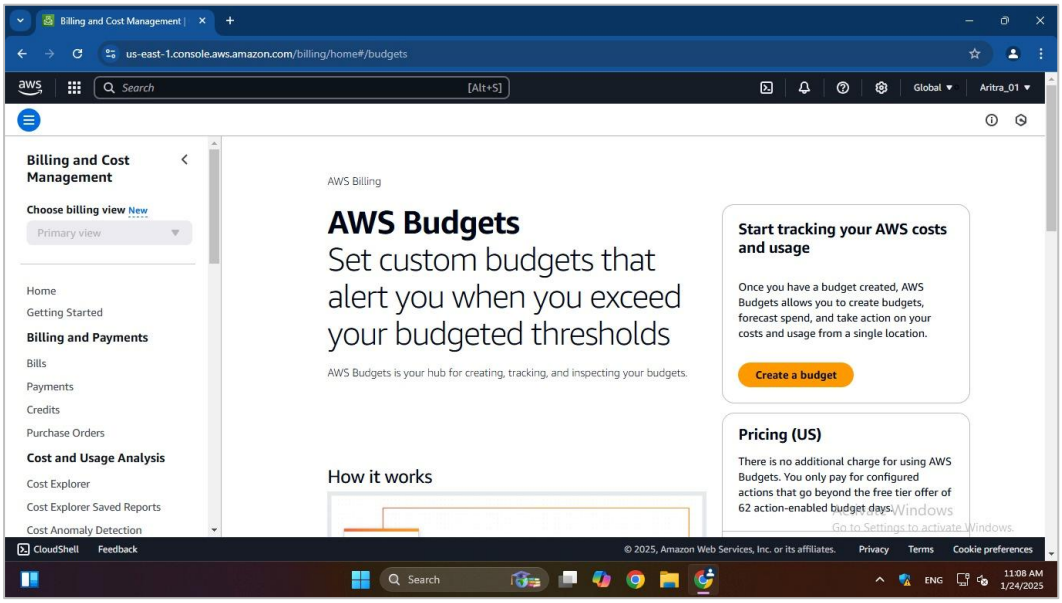
Budget Configuration:

1. Login to ***AWS Management Console***, click on your username at the top right corner and click on ***"Billing and Cost Management"***.

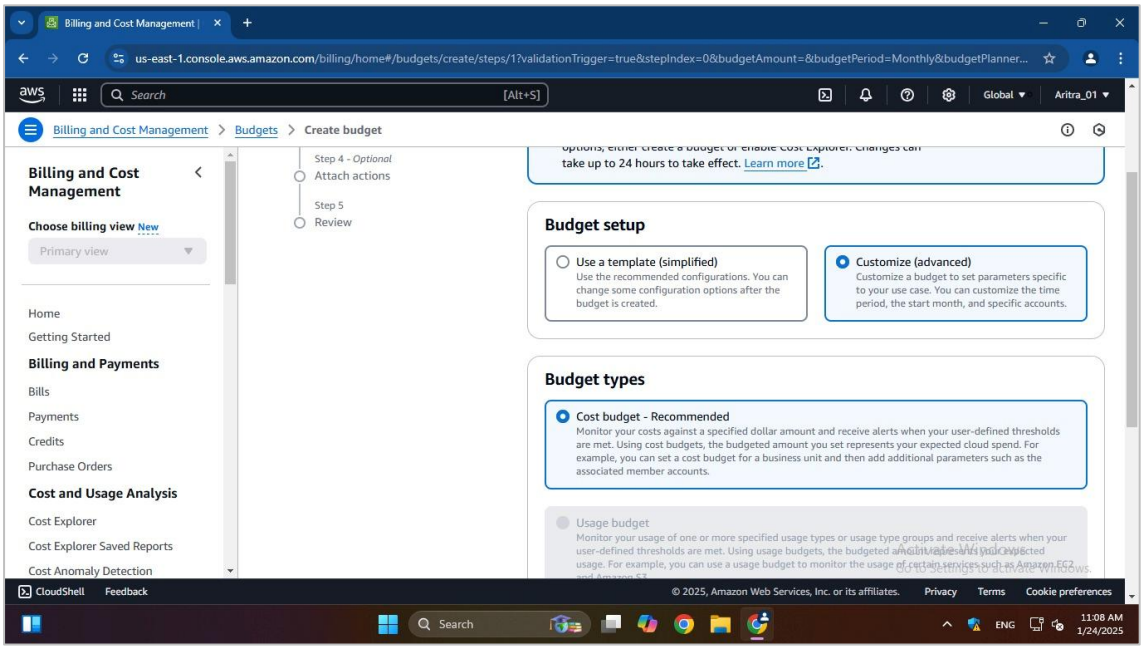


2. Under ***"Budgets and Planning"***, click on ***"Budgets"*** and then click on ***"Click on budget".***

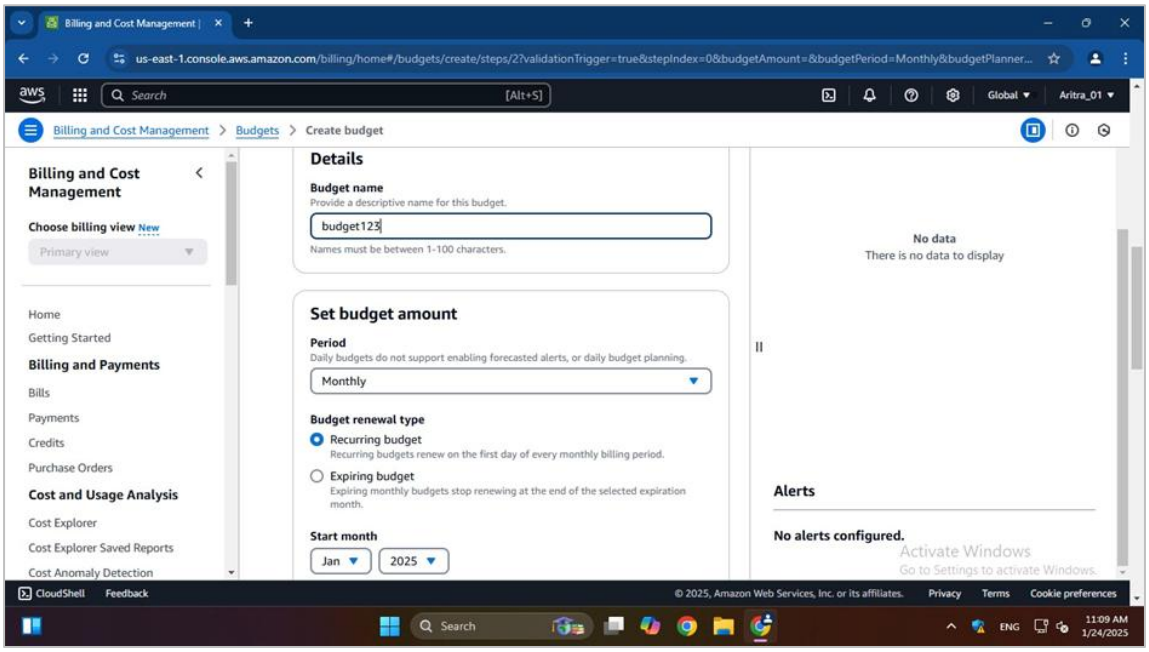




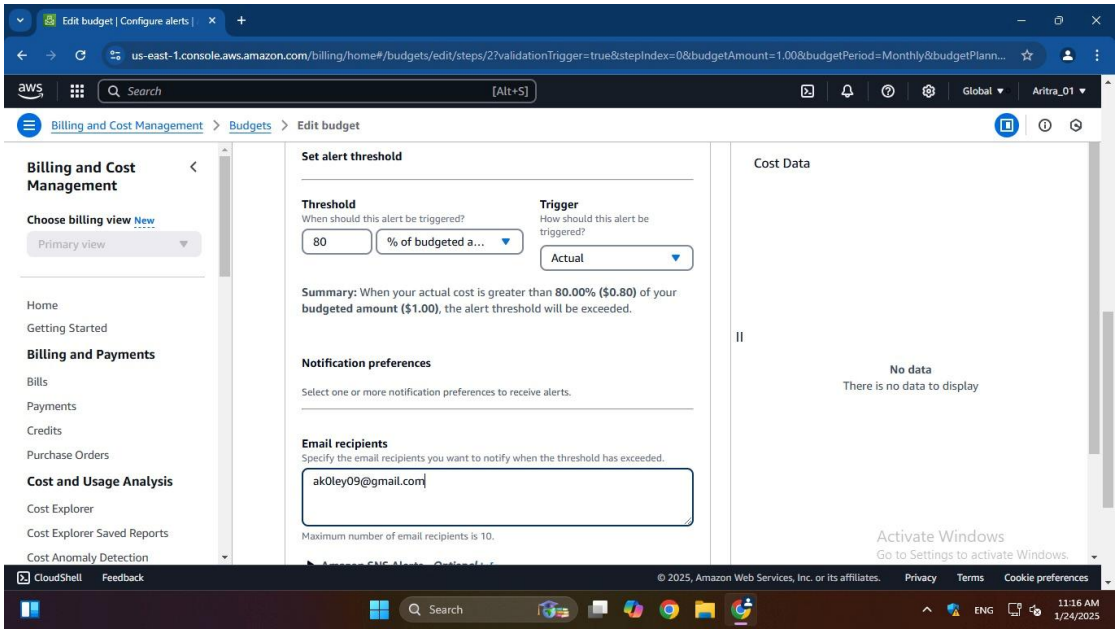
3. Configure budget setup by selecting ***"Customize (advanced)"*** then click on ***"Next"*.**



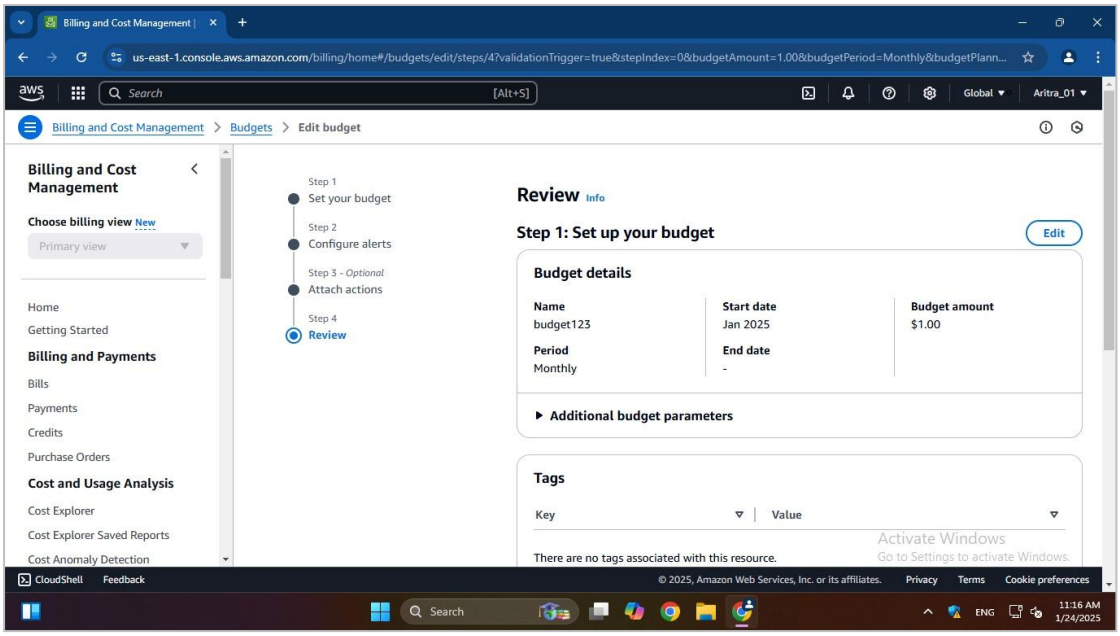
4. Add budget details.



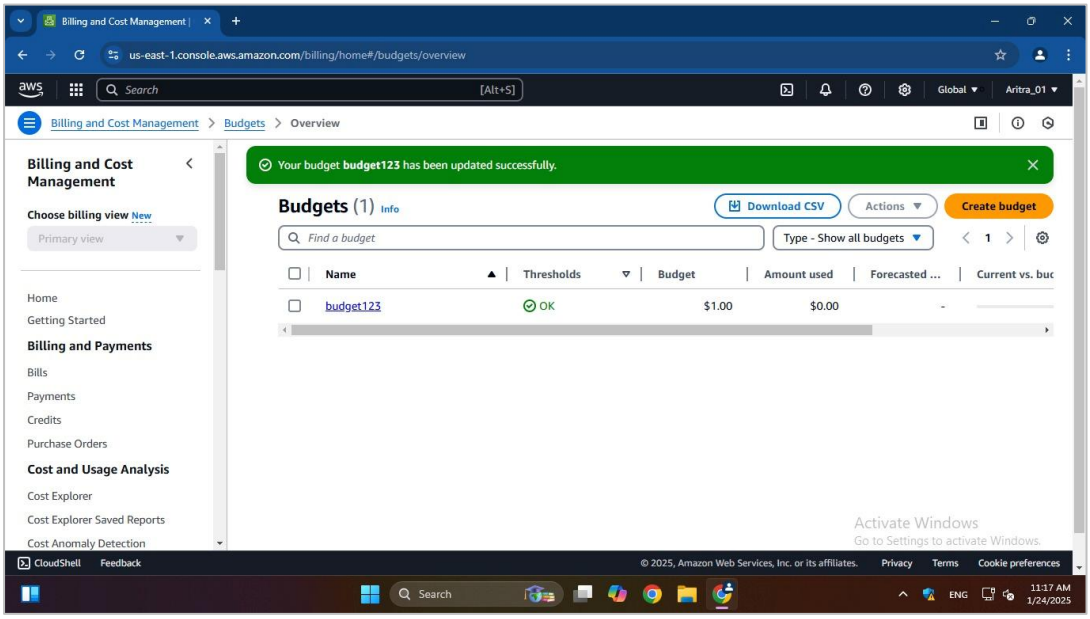
5. After selecting ***"Add threshold"***, fill-in the necessary details and then click on ***"Next"***.



6. Review the budget details and click on ***"Save"*** to finalize the budget.



7. Budget is successfully created.

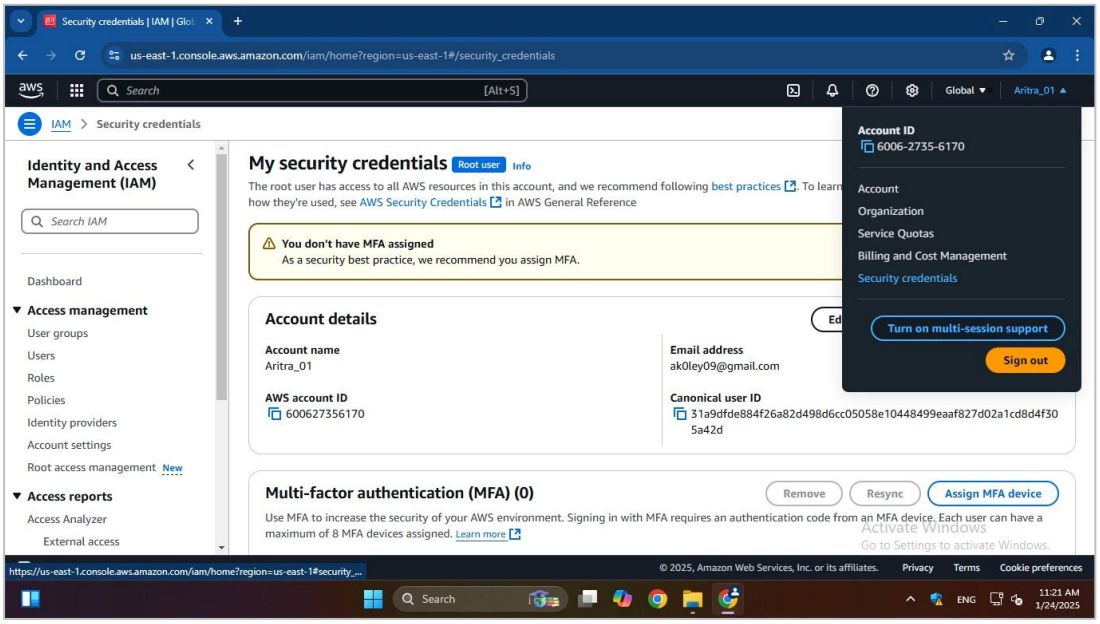


ASSIGNMENT 2:

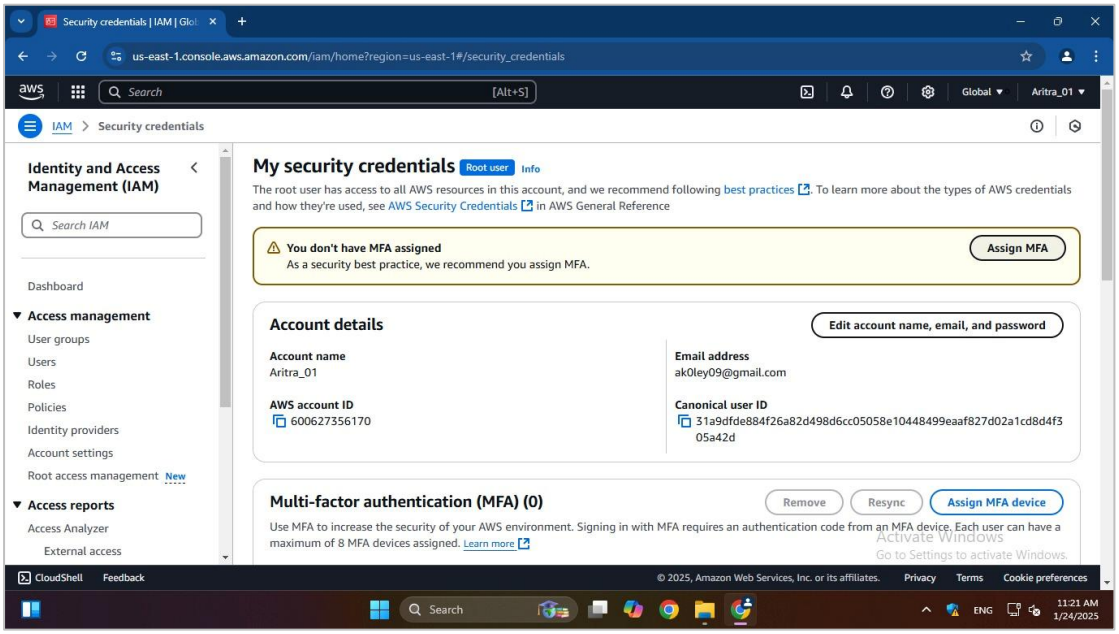
Problem Definition: Create MFA (Multi-Factor Authentication) for authentication.

Instructions:

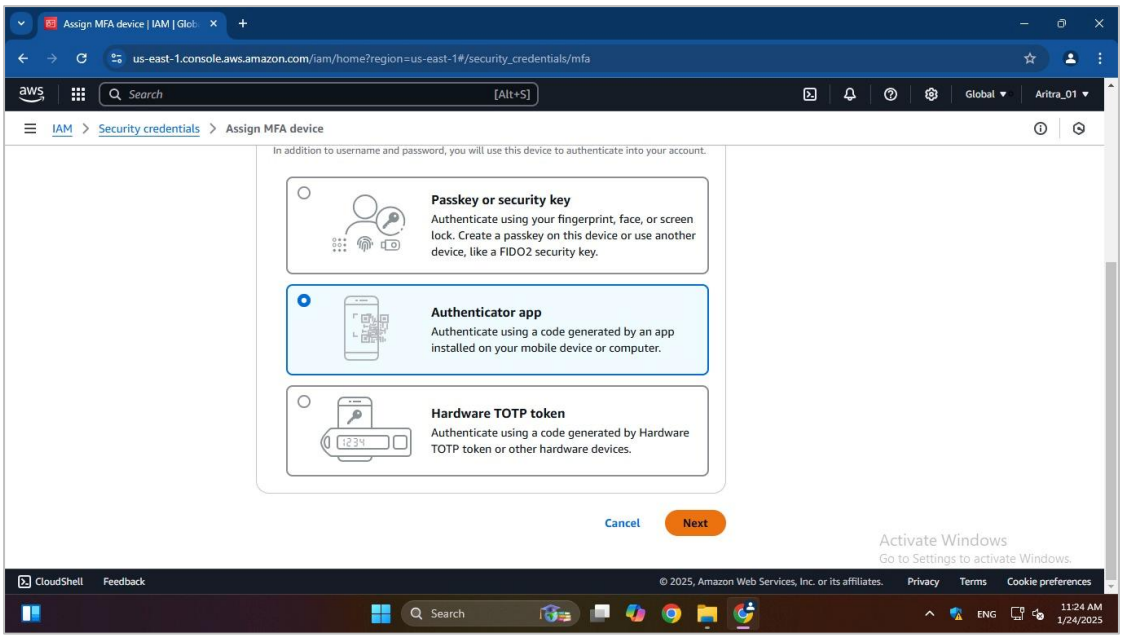
1. Log into AWS account and go to ***"Security credentials".***



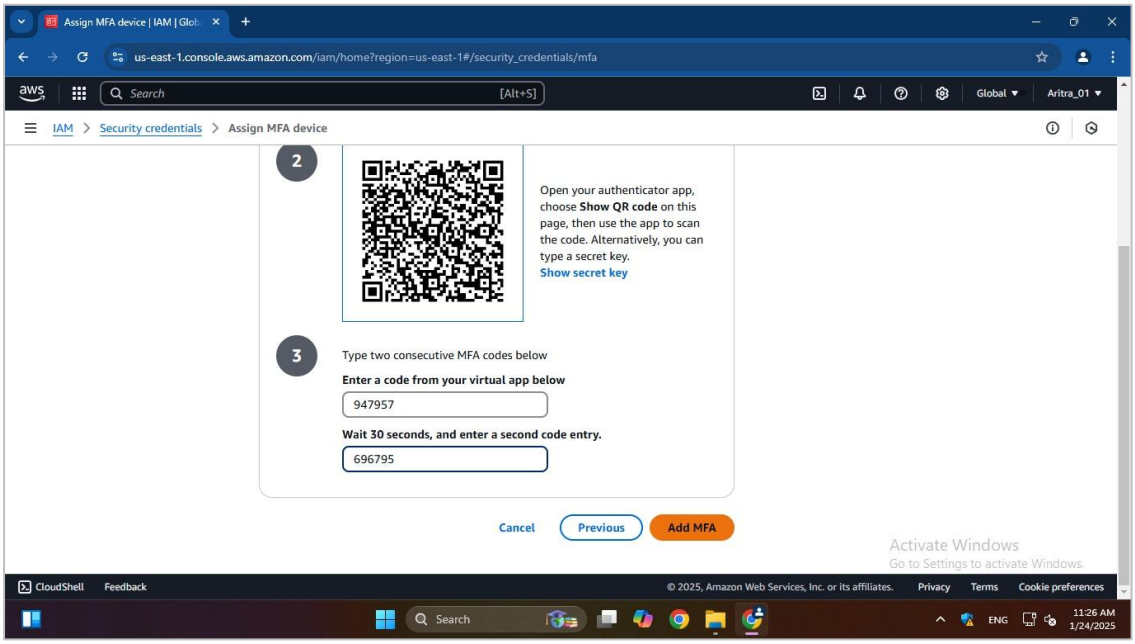
2. Add MFA device by clicking on ***"Assign MFA device"***.



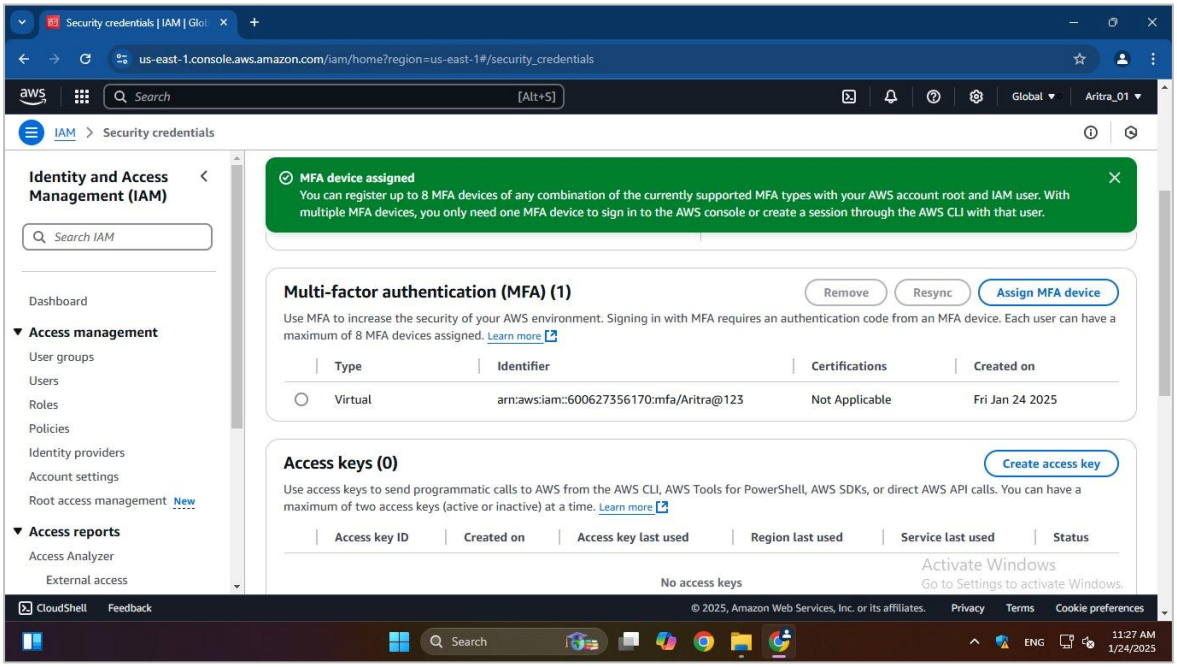
3. Enter a desired device name, MFA device and then click on ***"Next"***.



4. Enter code using authenticator app and click on ***"Add MFA"*** to complete the process.



5. MFA device assigned successfully.

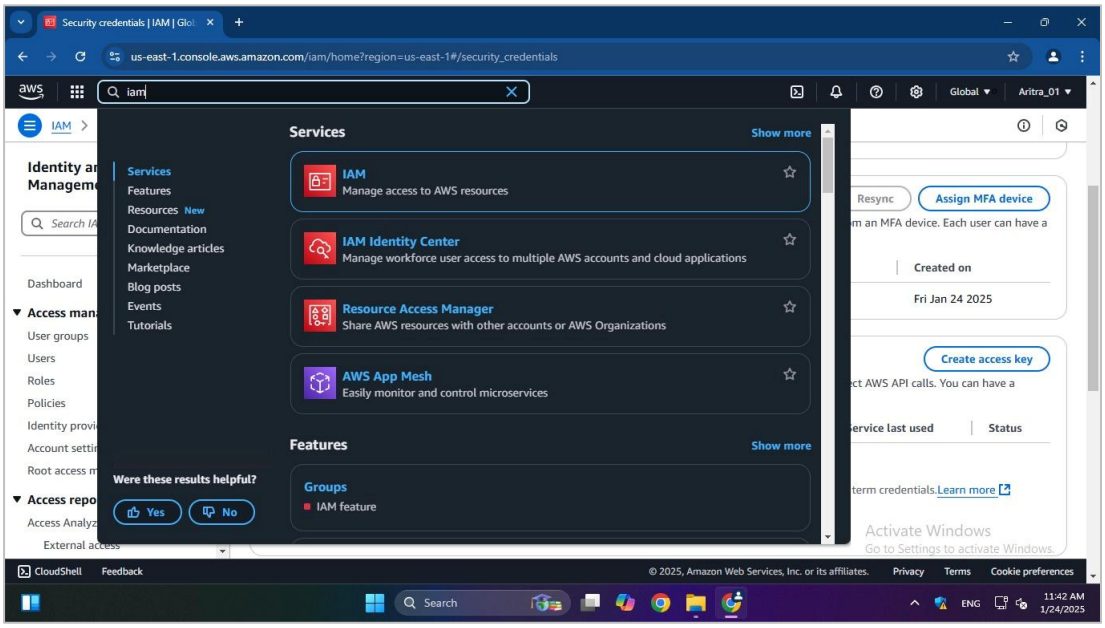


ASSIGNMENT 3:

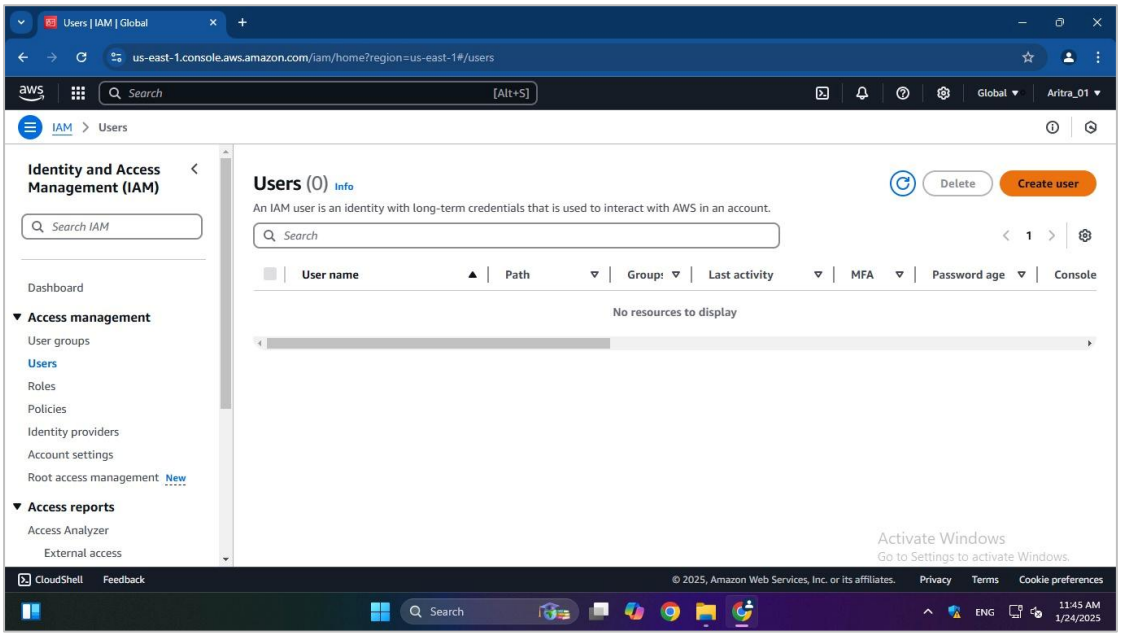
Problem Definition: Create IAM user and give full access to S3.

Instructions:

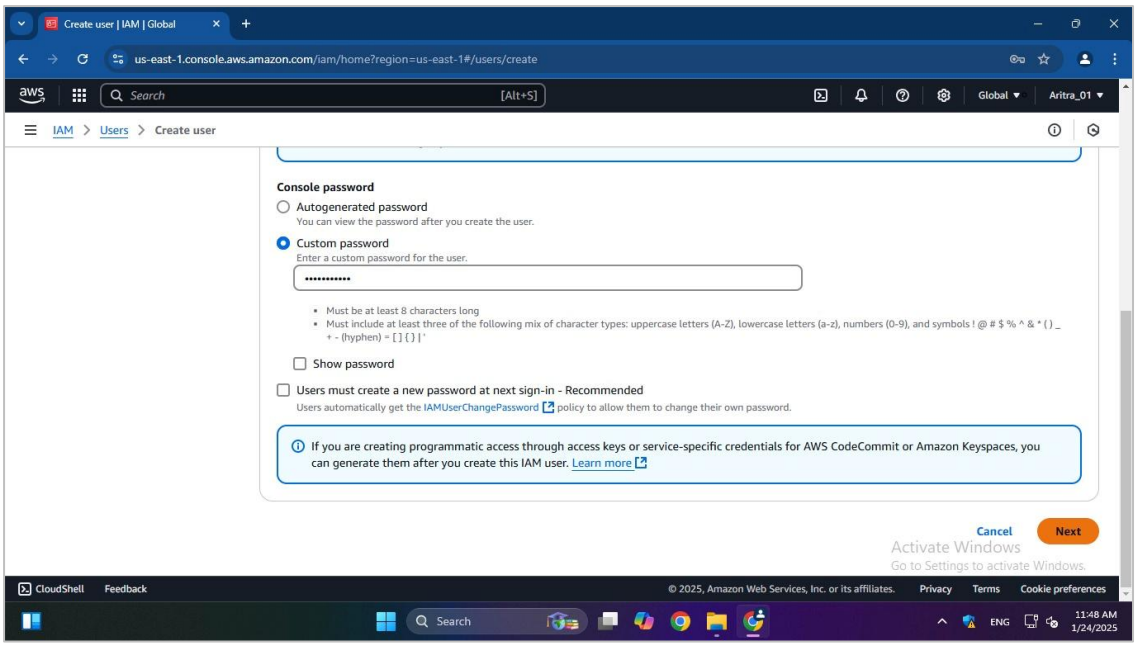
1. From AWS console, search for ***"IAM"*** and click on it.



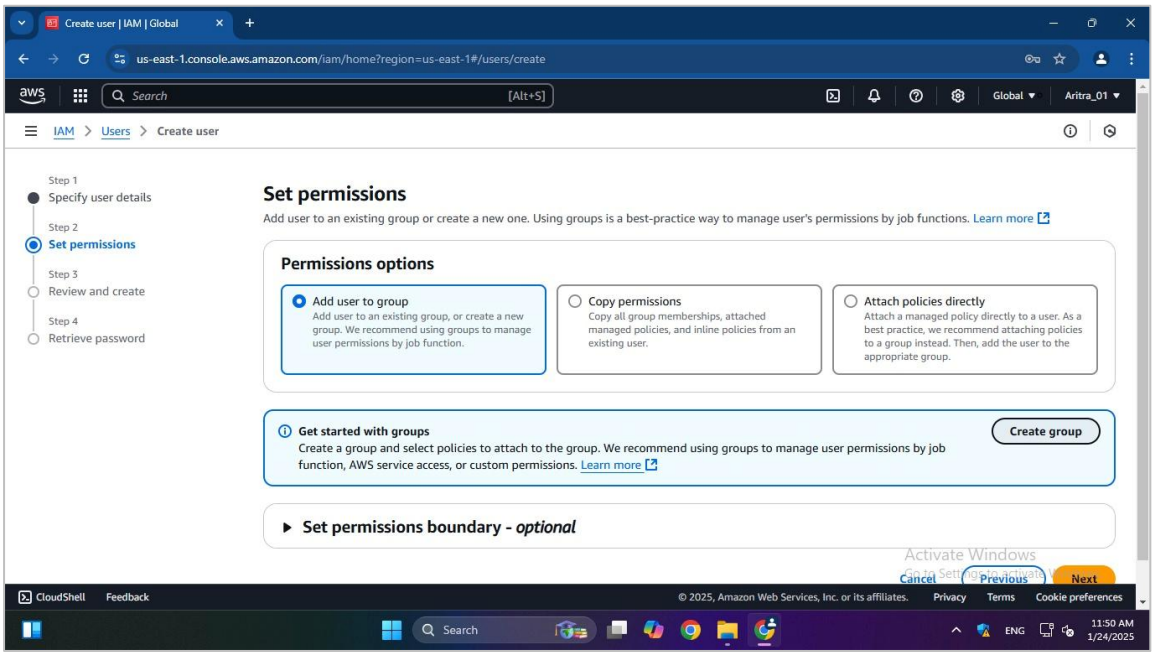
2. Under "***IAM resources"*** click on ***"Users"***.



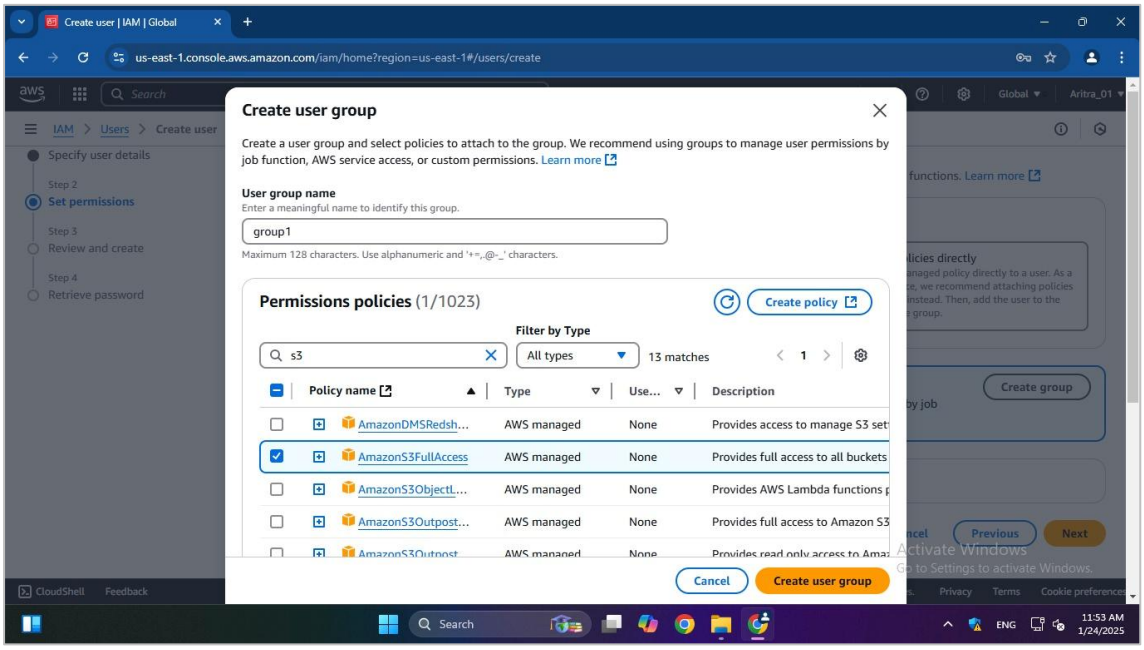
3. Click on ***"Create User"***. Provide User details (name, password, etc.) and click on ***"Next"*** to proceed.



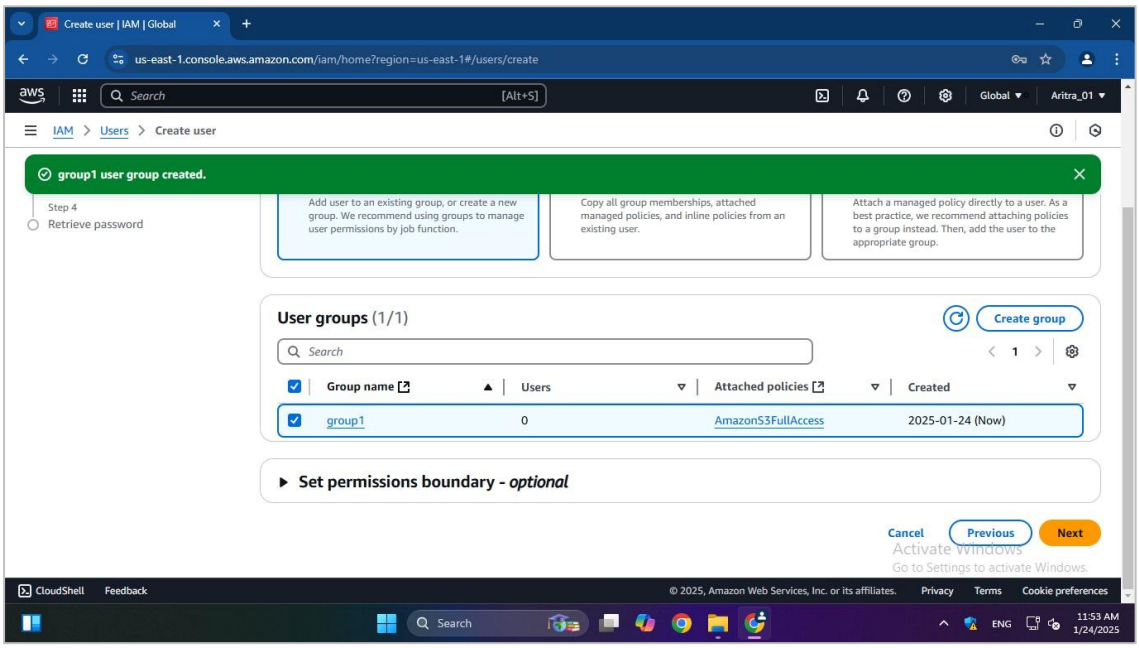
4. Under ***"Permissions options"***, click on ***"Add user to group"*** and go to ***"Create group"***.



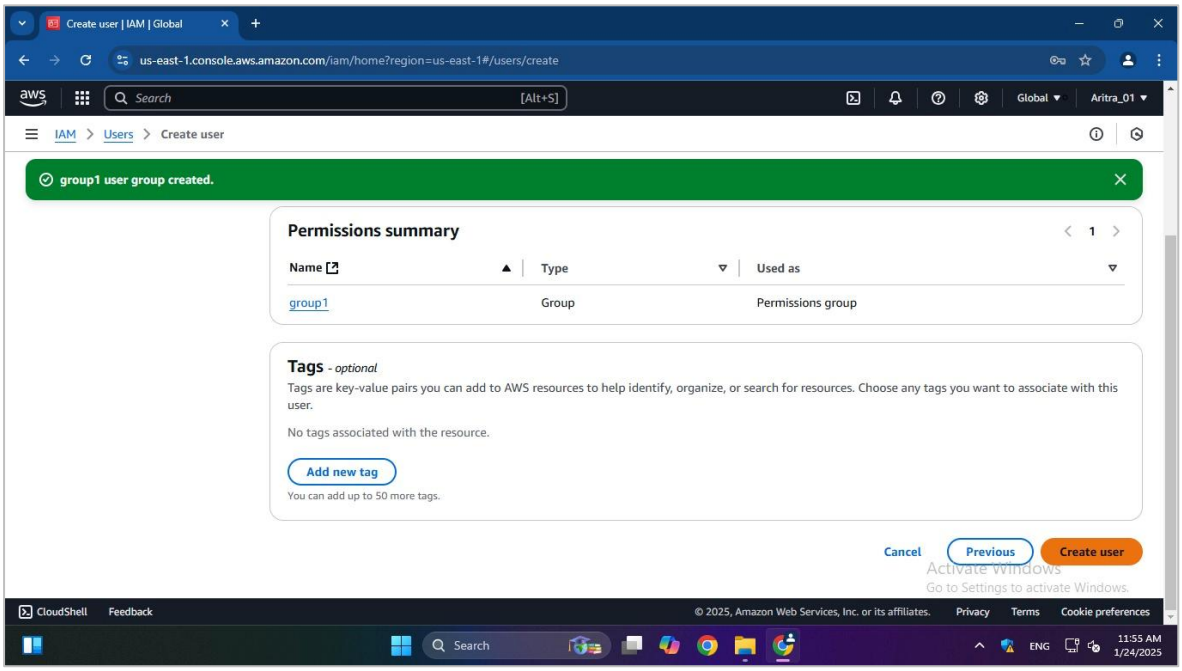
5. To create user group, first provide ***"User group name"***, then from "***Permission policies"*** search for ***"S3"*** and select full access of the particular permission we want to give to that group. Then click on ***"Create User Group"***.



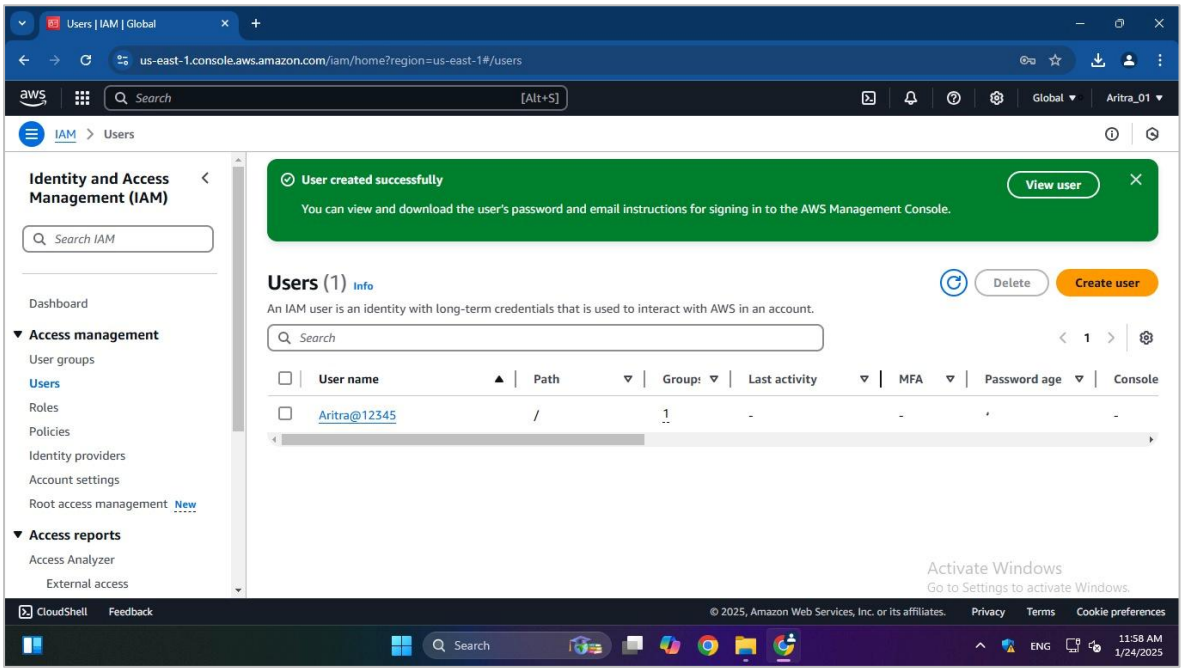
6. The group is successfully created. Tick the checkbox of the created group in ***"User groups***", so that our IAM user is connected to this group and gets the required permission. Then click on ***"Next"***.



7.Under ***"Review and create"***, click on ***"Create user"*** to create our IAM user.



8. IAM user is created successfully.

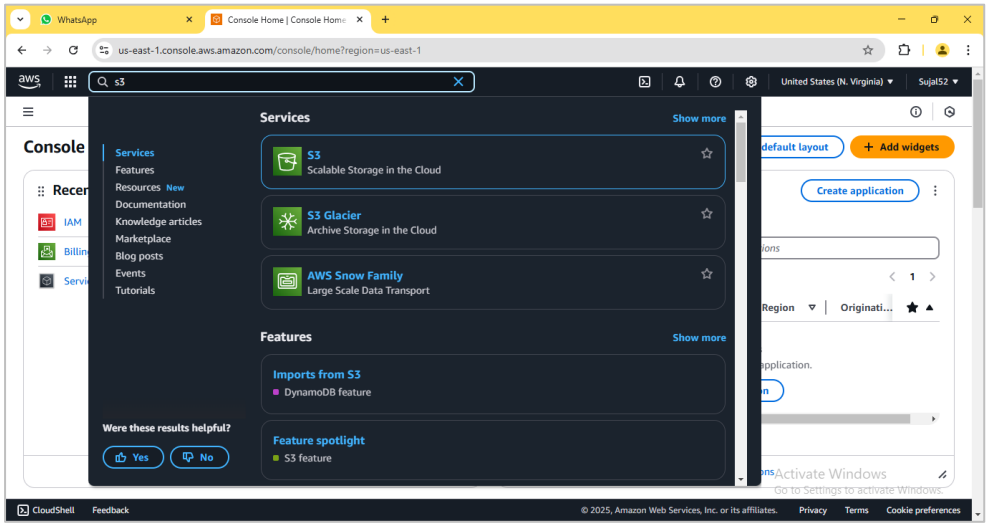


ASSIGNMENT 4:

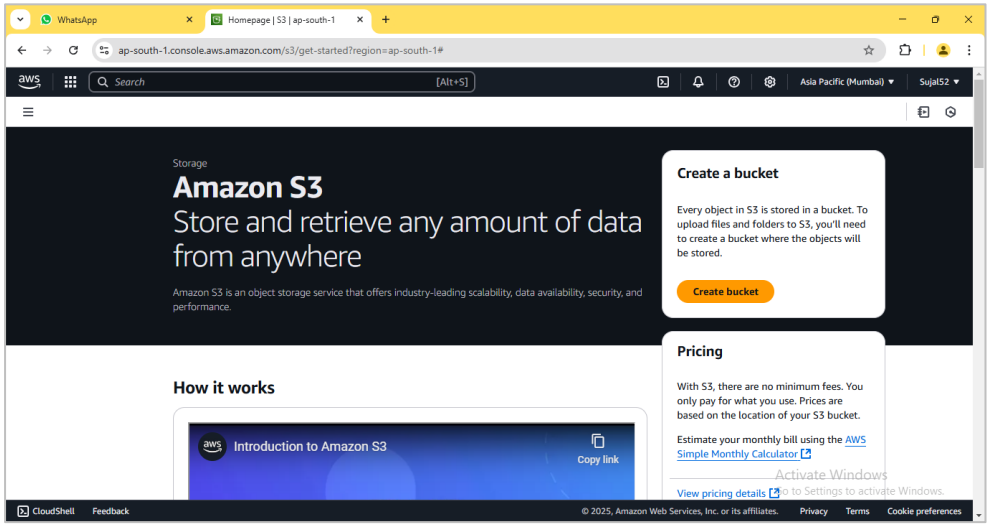
Problem Definition: Create a private bucket in AWS. Upload a file and check by reassigned URL whether you can access the file or not

Instructions:

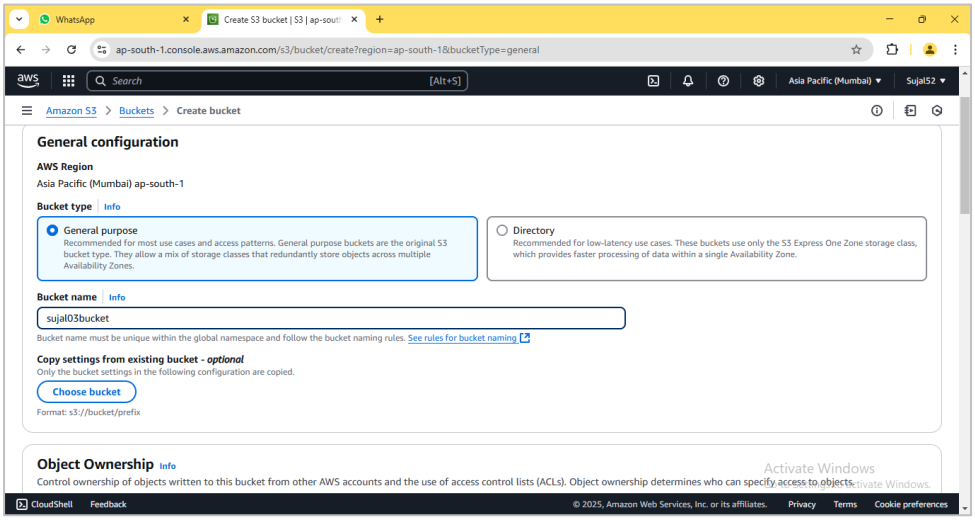
1. Access the ***AWS console***, search for ***S3***, and select the top option from the search results.

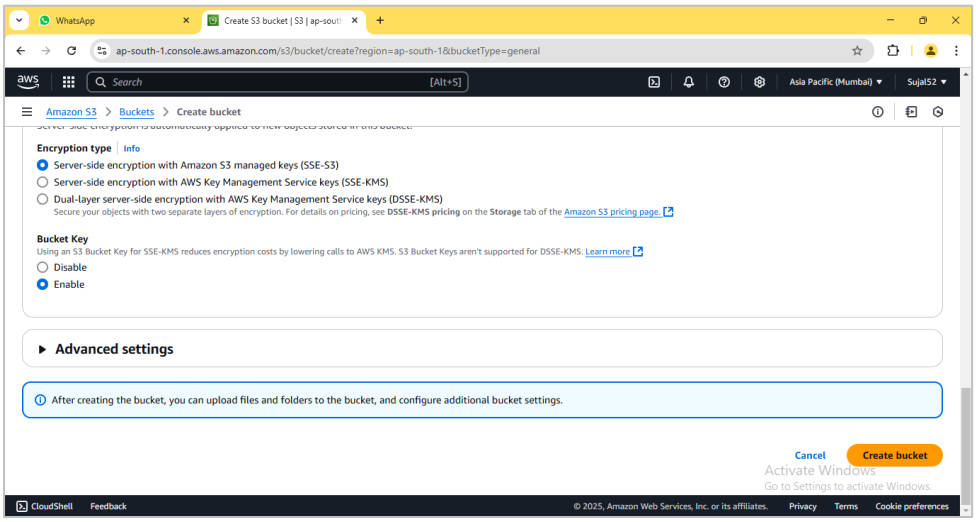


2. Click on ***"Create bucket"***.

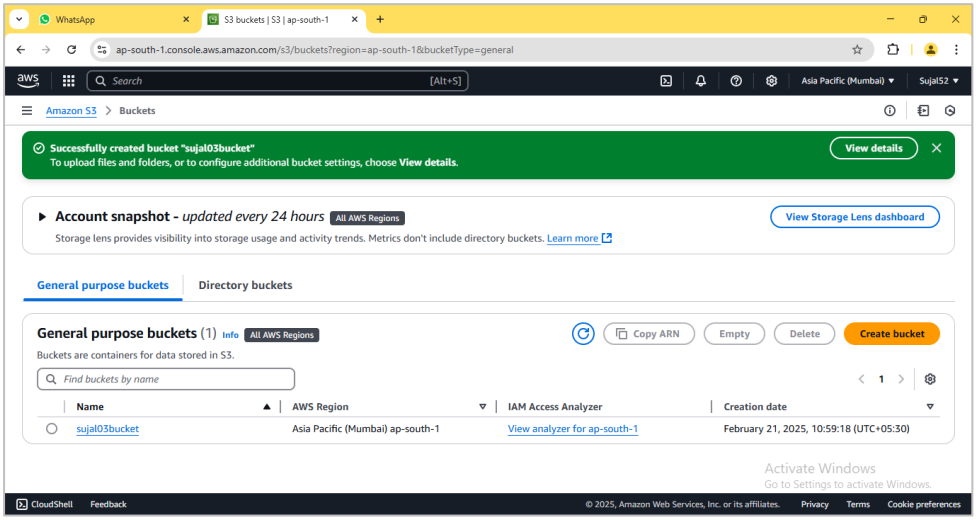


3. Enter bucket name. Leave the other options set to default as we are creating a private bucket. Then click on ***"Create bucket"***.

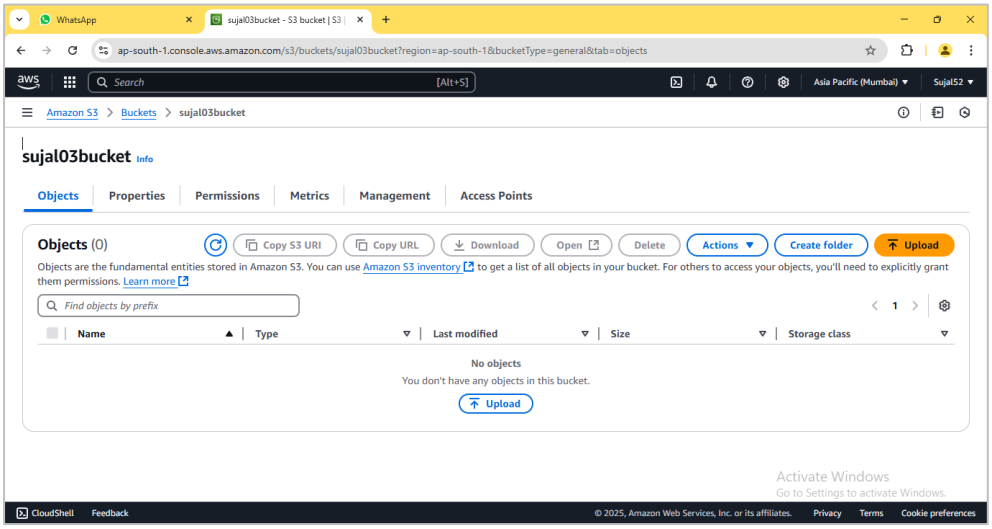




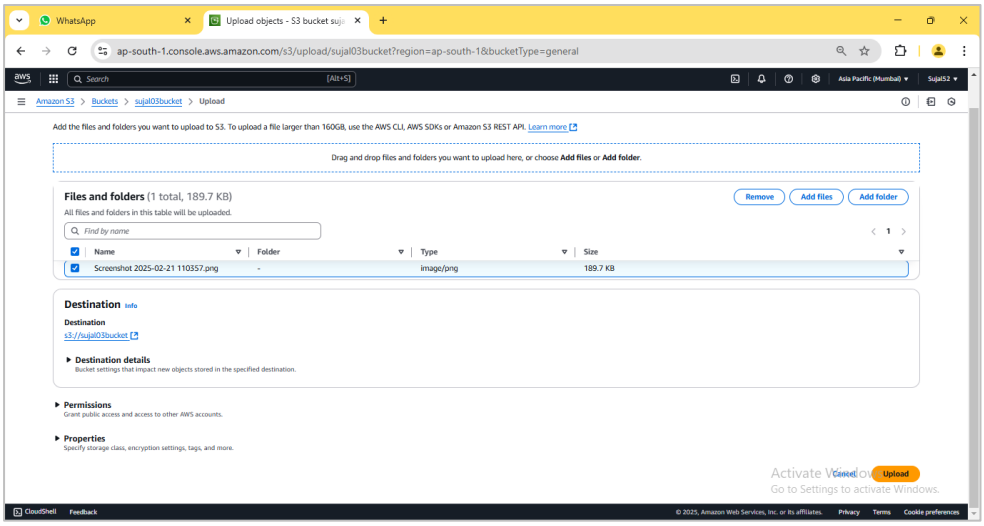
4. A green coloured pop-up dialogue box shall confirm the creation of our bucket. It shall also be displayed on the bucket list.



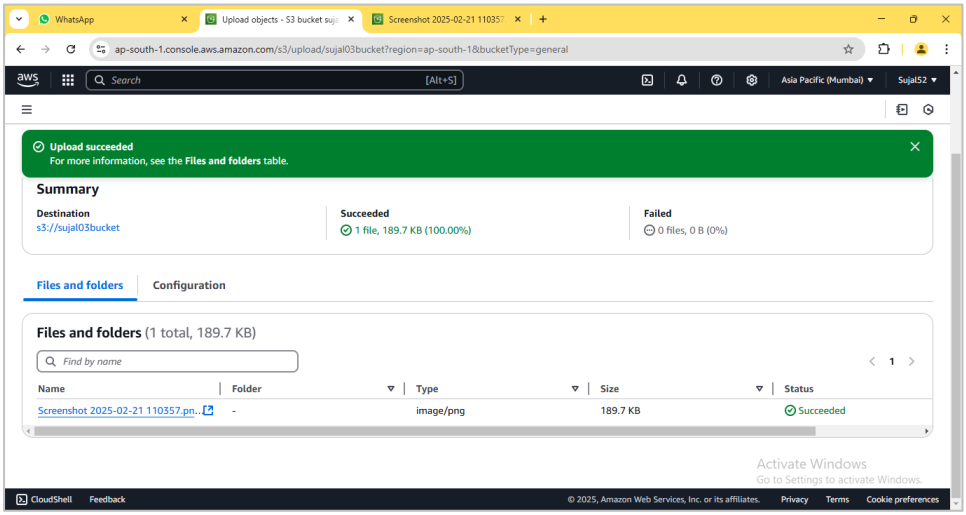
5. Click on the name of our newly created bucket. On the bucket detail page, click on ***"Upload"***.



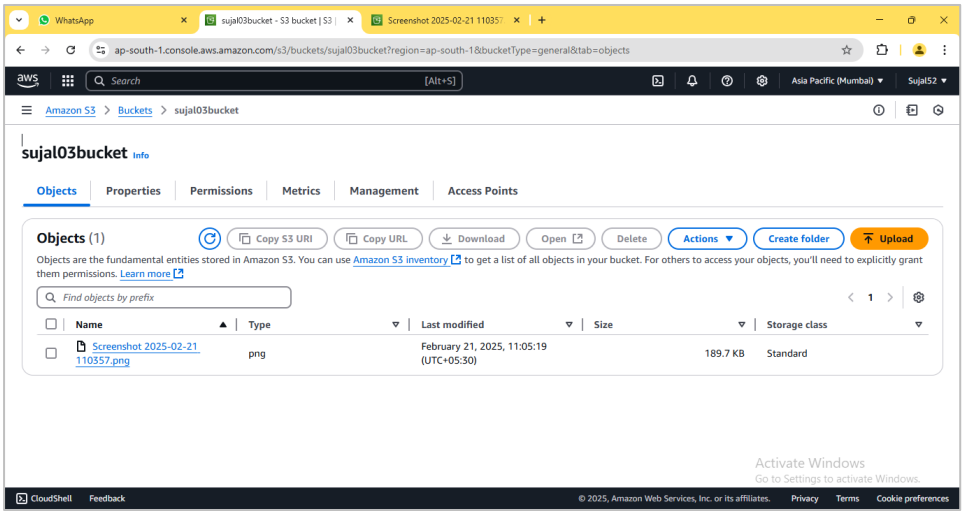
6. Click on ***"Add files"*** then select the file we want to upload to our bucket. Select our file from the list and click on ***"Upload"*** button.

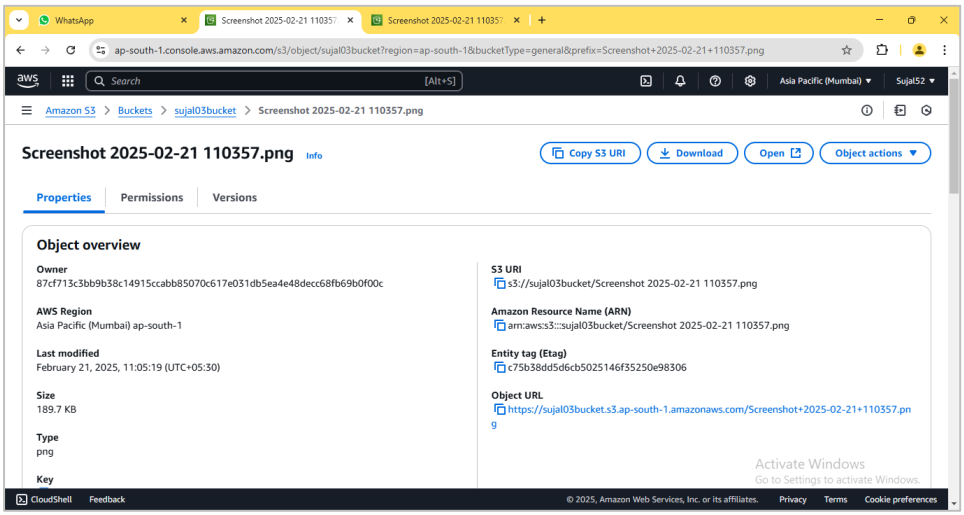


7. The notification confirms that the file has been successfully uploaded. Click on ***"Close"*** to return to our bucket details page.

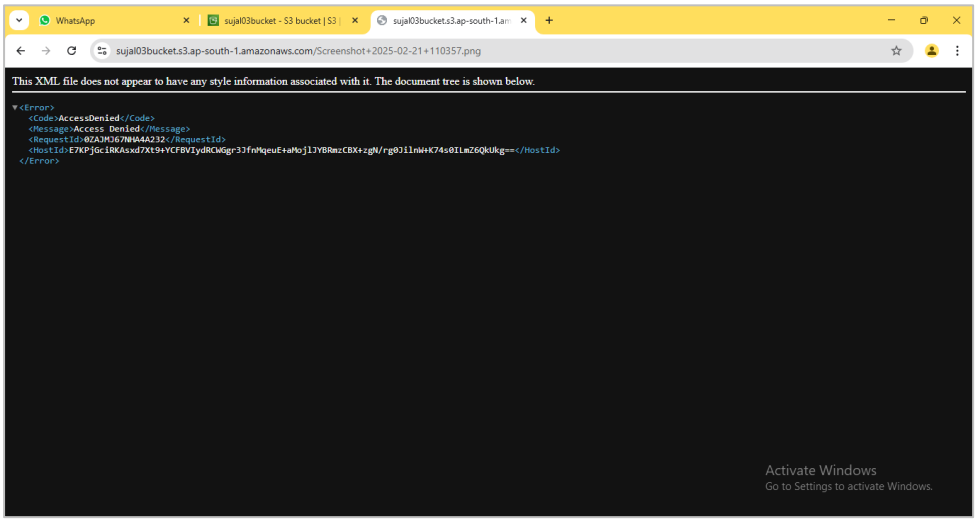


8. Click on our previously uploaded file from the list. Copy the ***"Object URL"*** from the file details page that appears.

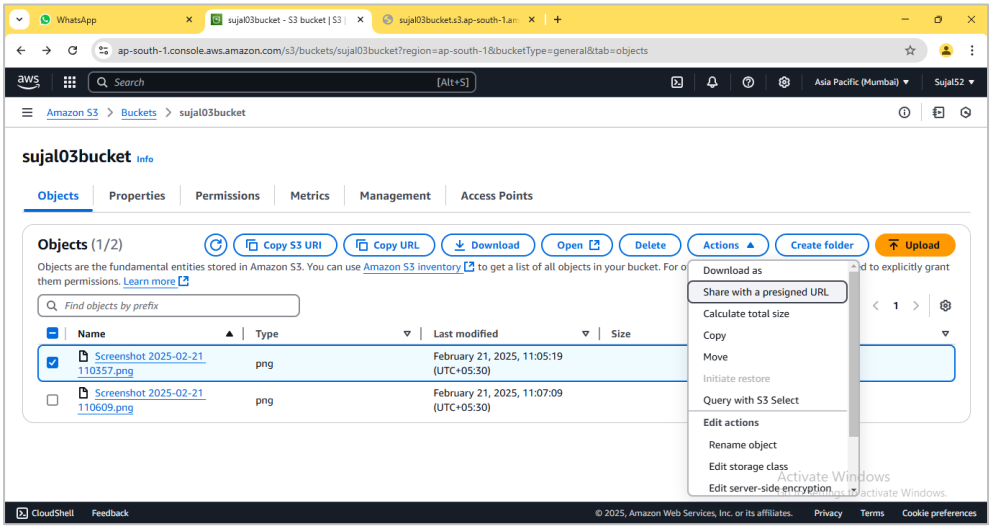




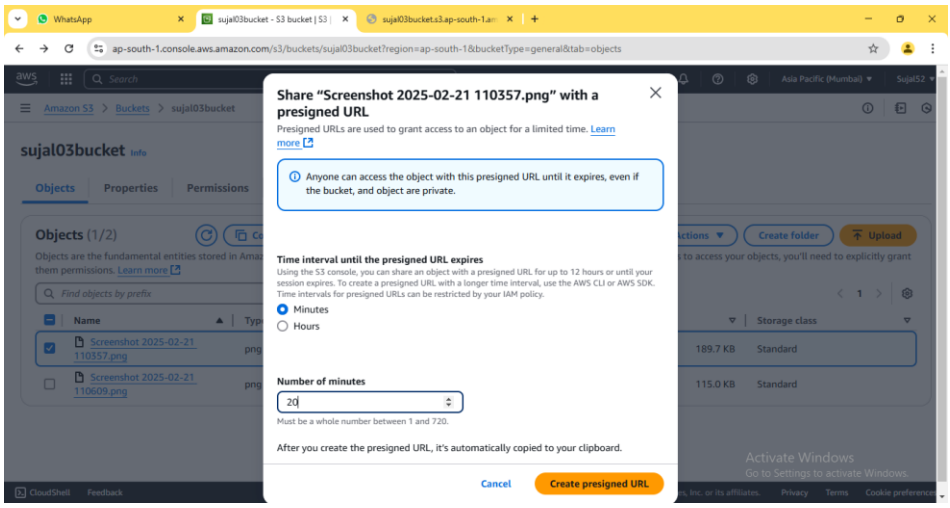
9. Open the copied URL in a new browser window. An error appears indicating that access to the contents is restricted since this is a private bucket.



10. Go back to bucket details page, select our file from the list, click on ***"Actions"*** and select ***"Share with a presigned URL"*** from the dropdown.



11. Select the desired duration until you want the presigned URL to expire then click on ***"Create presigned URL"***.



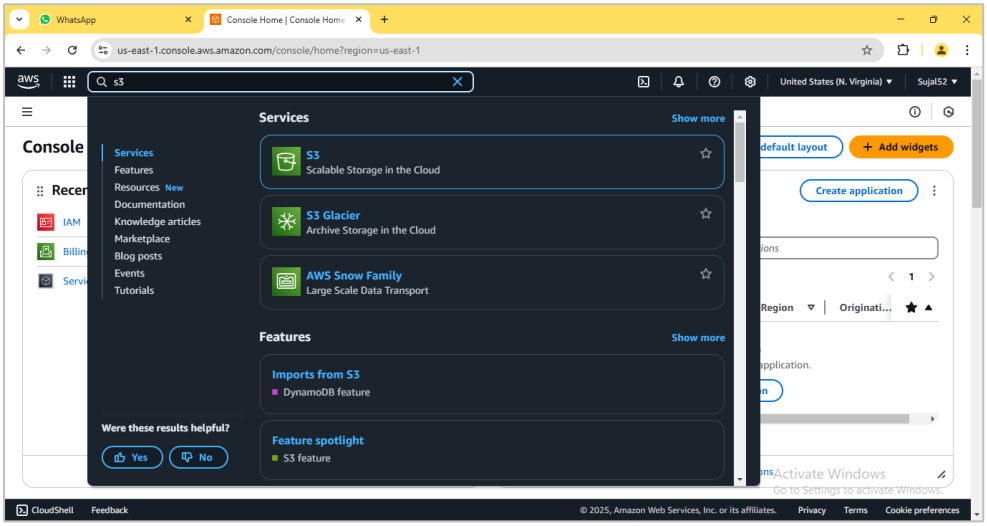
12. Copy the pre-assigned URL and paste it on a new window. Now, you can view the uploaded file.

ASSIGNMENT 5:

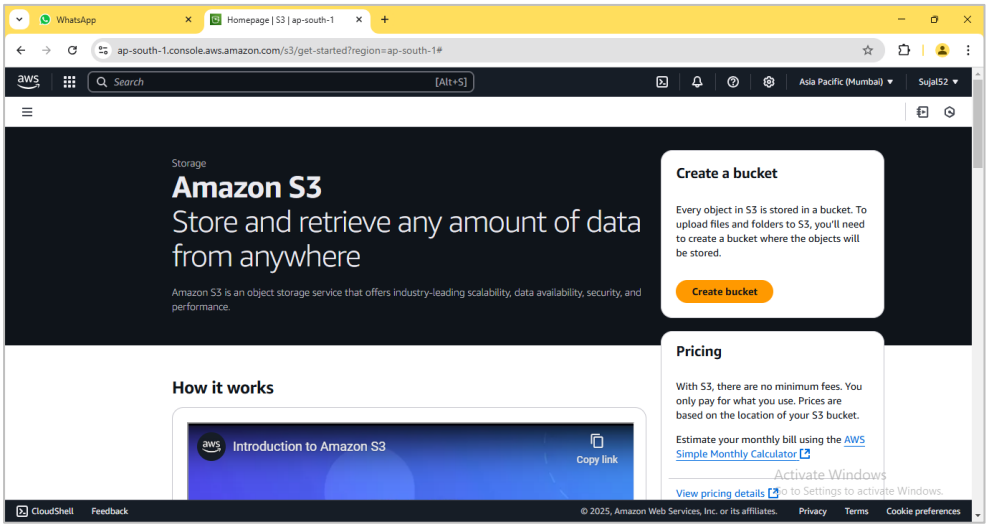
Problem Definition: Create a public Bucket in AWS. Upload a file and give the necessary permission to check whether file the URL is working.

Instructions:

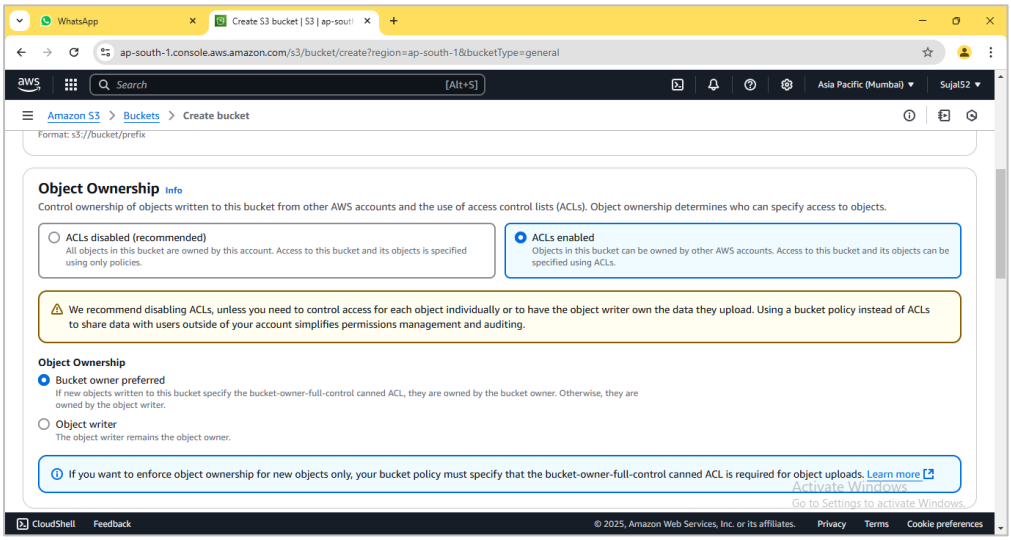
1. Access the ***AWS console***, search for ***S3***, and select the top option from the search results.



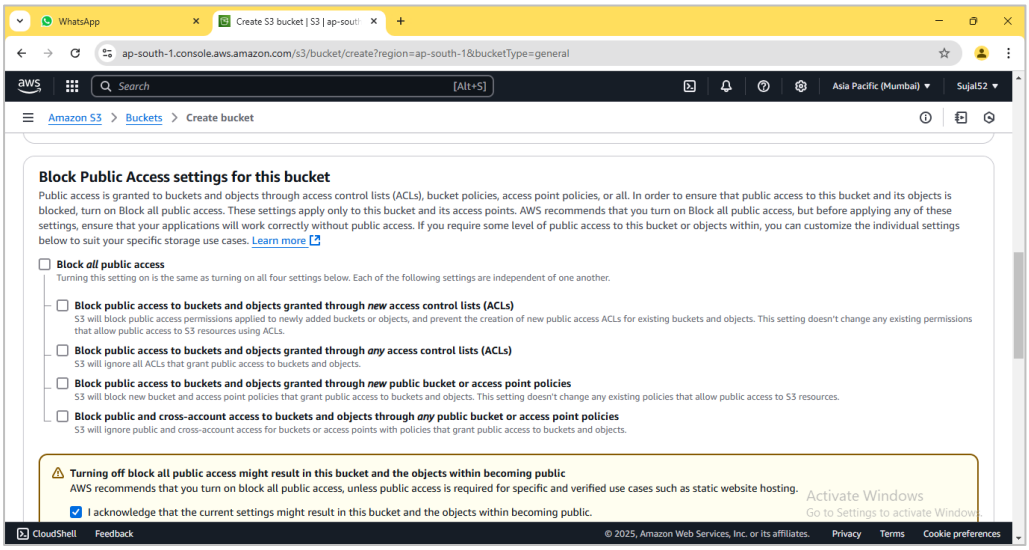
2. Click on ***"Create bucket"***.



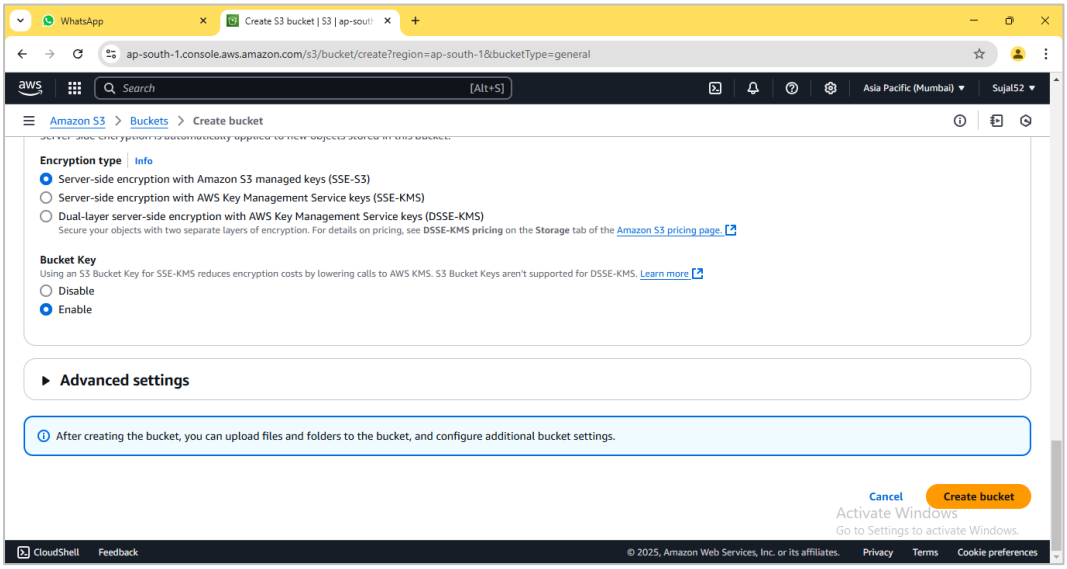
3. Enter bucket name. Select ***"ACLs enabled"*** since we are creating a public bucket.



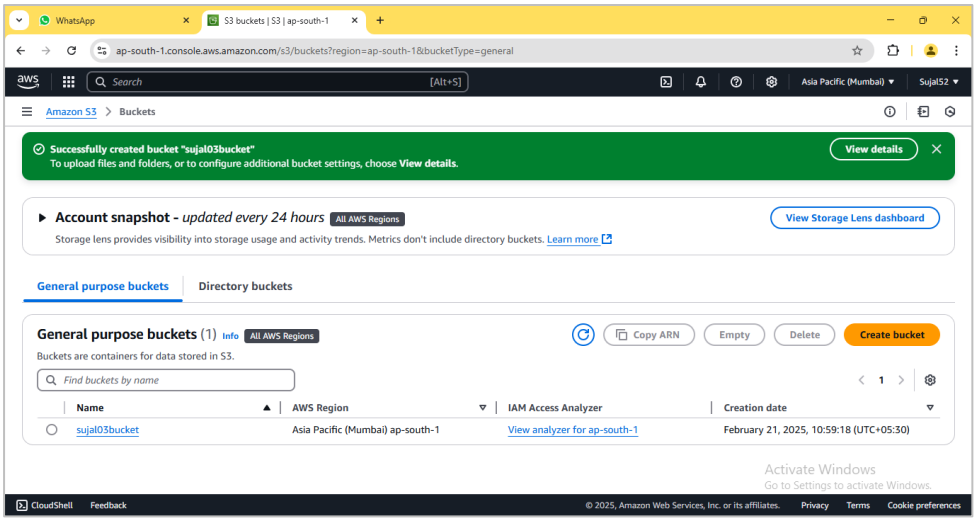
4. Now, uncheck the box for ***"Block all public access"***. At the bottom of the window, check the box confirming that the current setting changes may result in a public bucket.



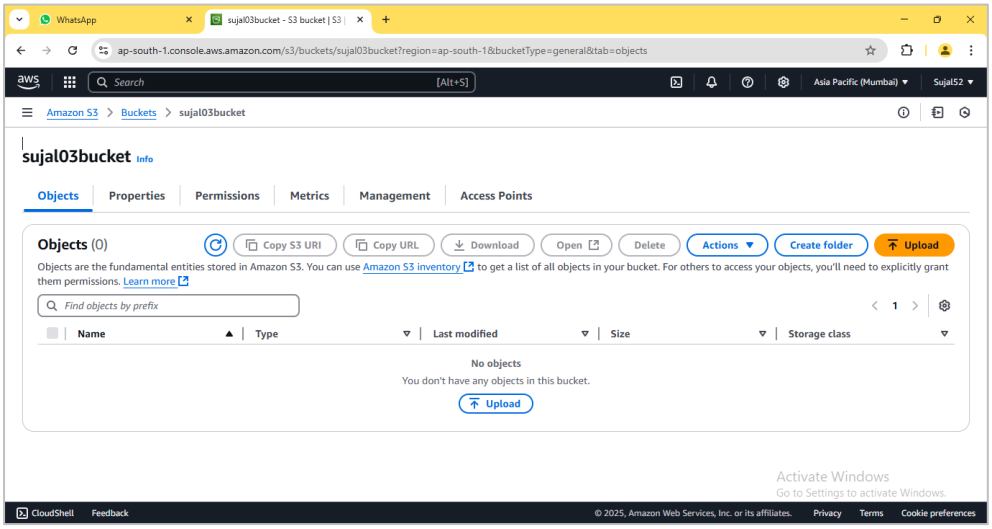
5. Scroll down without any further modifications and proceed to click on ***"Create Bucket"***.



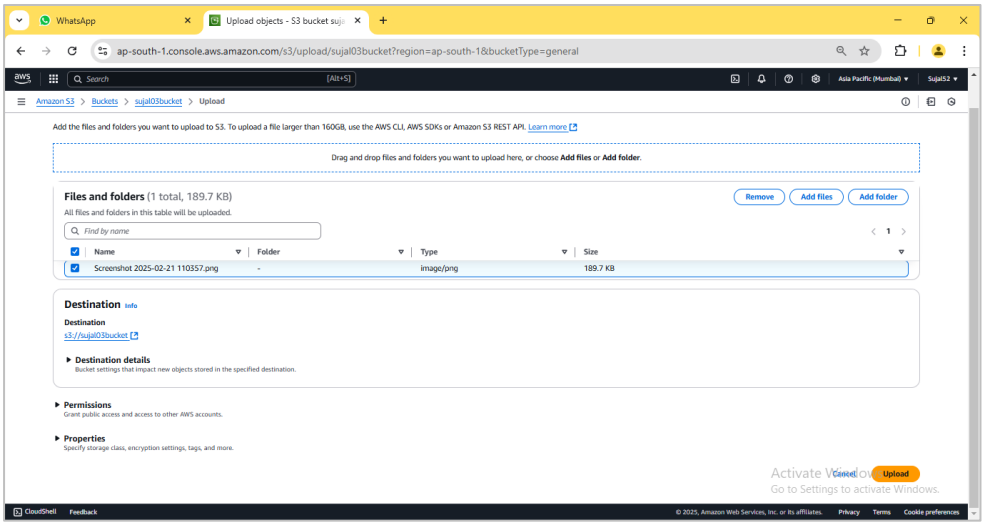
6. A green coloured pop-up dialogue box shall confirm the creation of our bucket. It shall also be displayed on the bucket list.



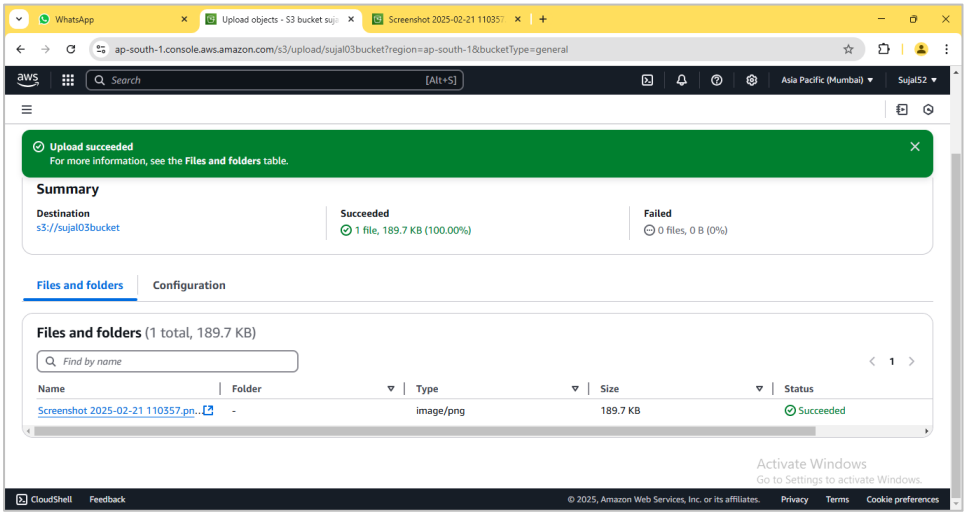
7. Click on the name of our newly created bucket. On the bucket detail page, click on ***"Upload"***.



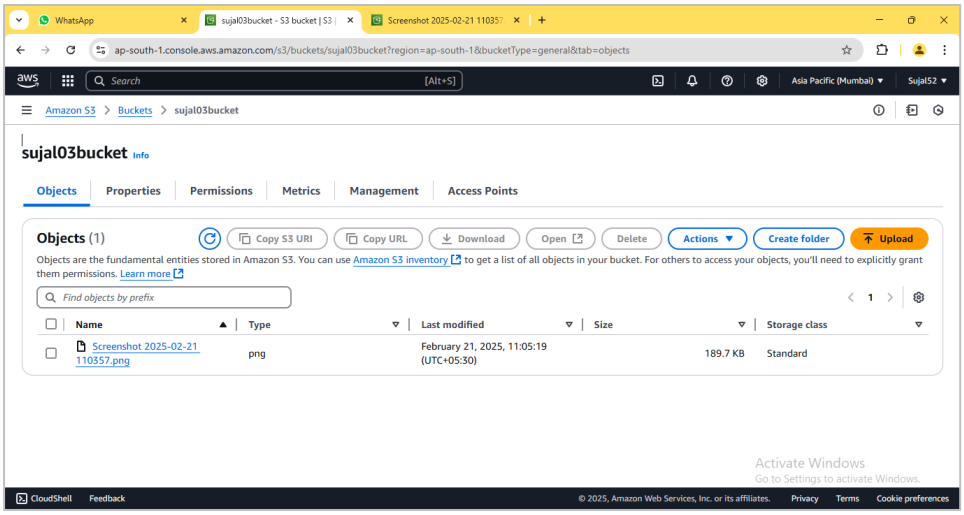
8. Click on ***"Add files"*** then select the file we want to upload to our bucket. Select our file from the list and click on ***"Upload"*** button.

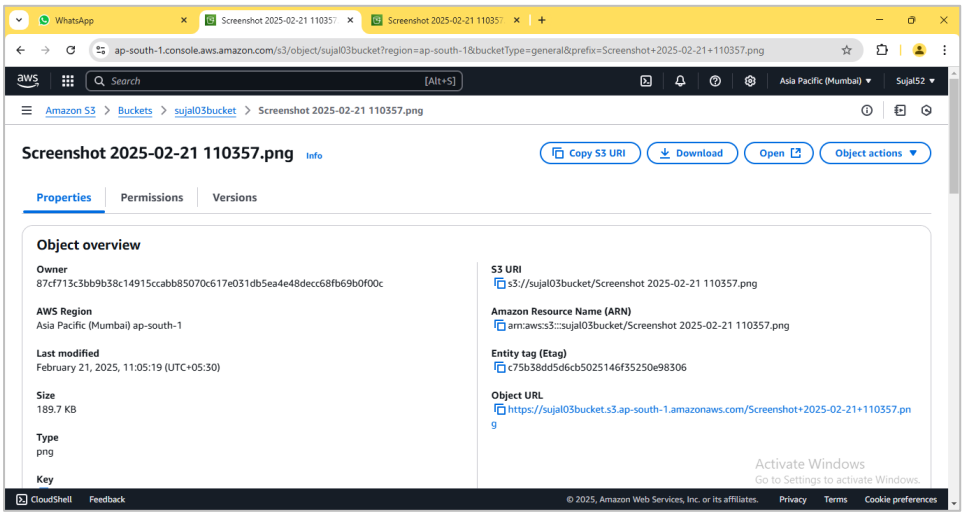


9. The notification confirms that the file has been successfully uploaded. Click on ***"Close"*** to return to our bucket details page.

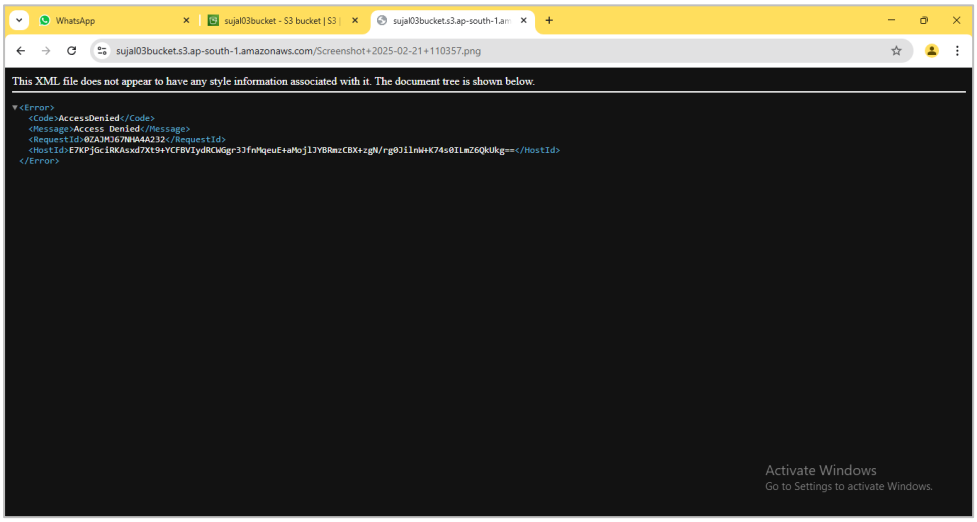


10. Click on our previously uploaded file from the list. Copy the ***"Object URL"*** from the file details page that opens.

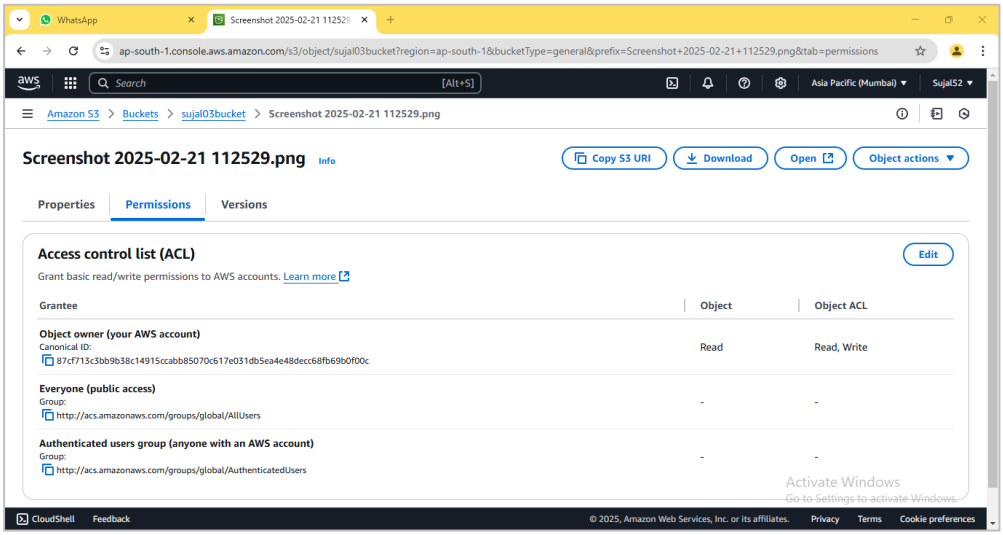




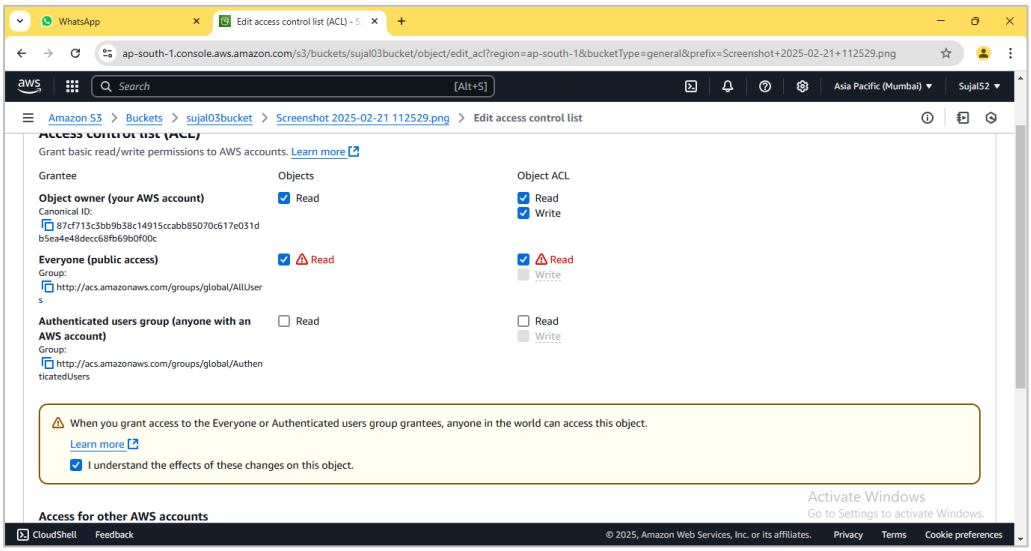
11. Open the copied URL in a new browser window. An error appears indicating that access to the contents is restricted since this is a private bucket.



12. Go back to file details page and click on ***"Permissions"*** tab. Then click on ***"Edit"*** button.



13. Edit the permissions as follow then click on ***"Save changes"***.



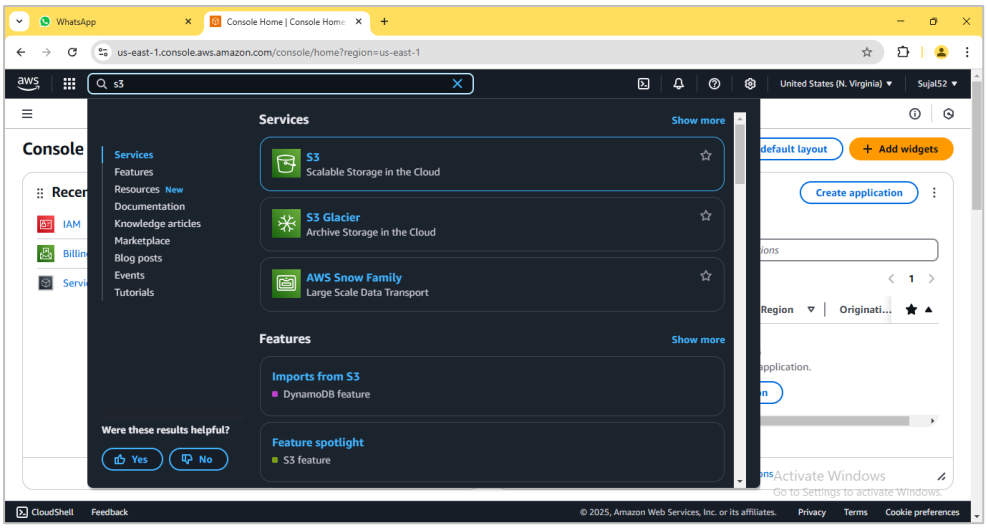
14. Copy the URL as mentioned previously and open it on new window.

ASSIGNMENT 6:

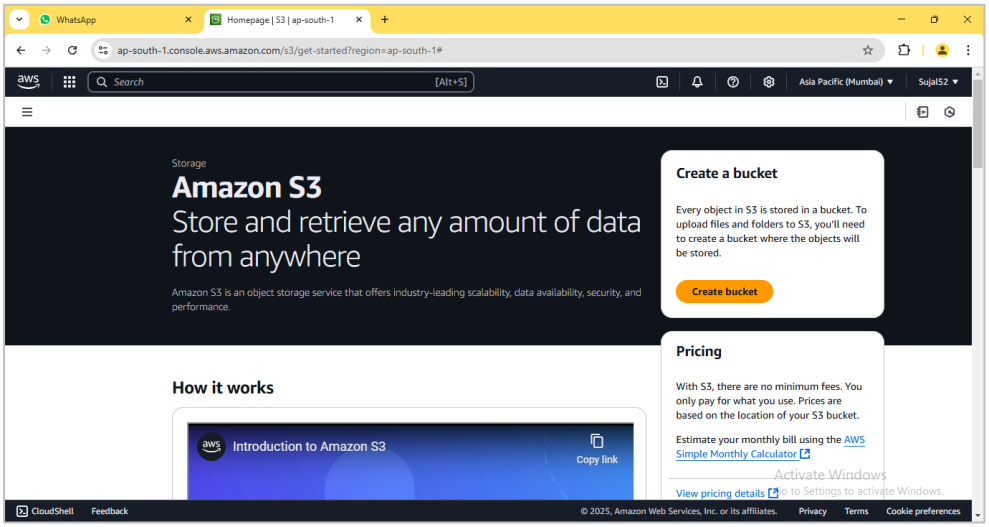
Problem Definition: Upload a static website on S3.

Instructions:

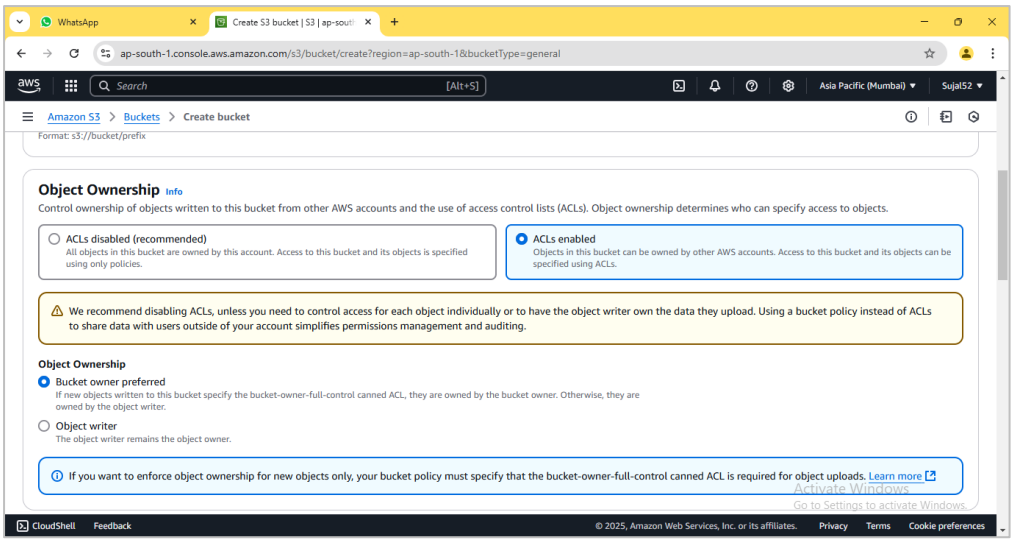
1. Access the ***AWS console***, search for ***S3***, and select the top option from the search results.

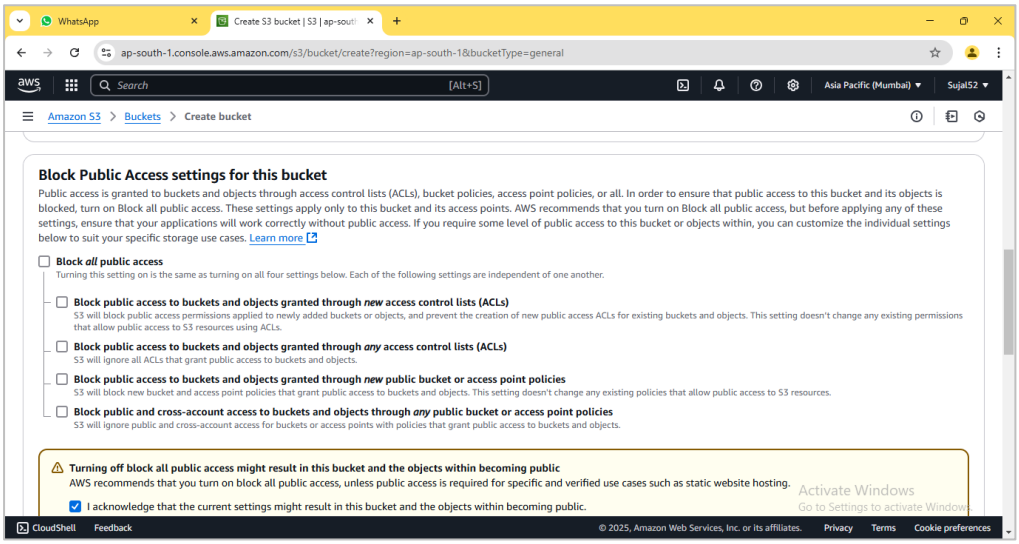


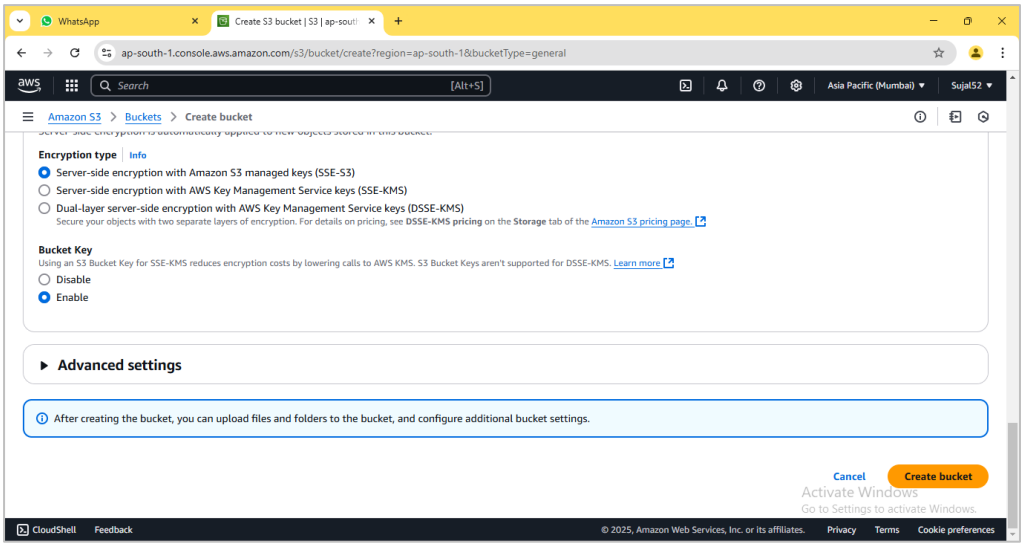
2. Click on ***"Create bucket"***.



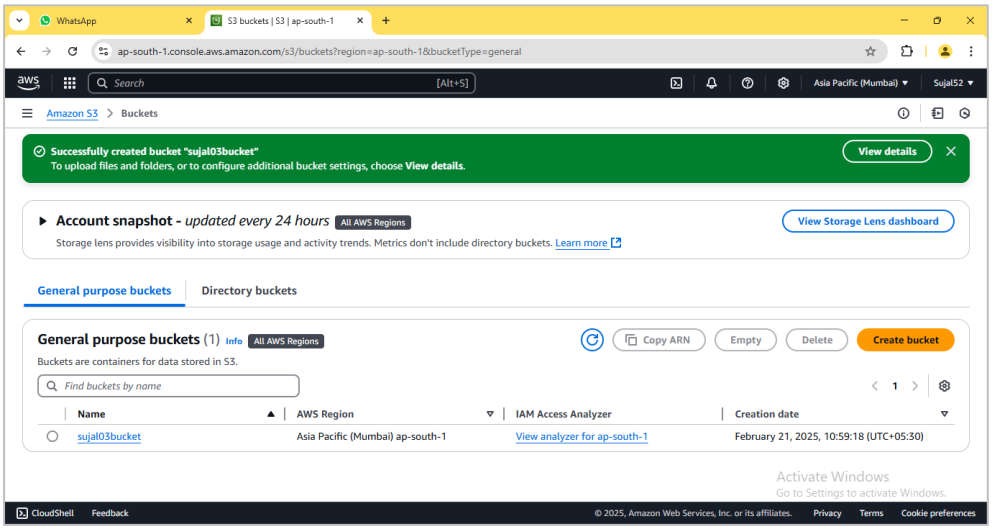
3. Enter bucket name. Select ***"ACLs enabled"*** since we are creating a public bucket. Uncheck the box for ***"Block all public access".*** Leave other options as default and click on ***"Create bucket".***



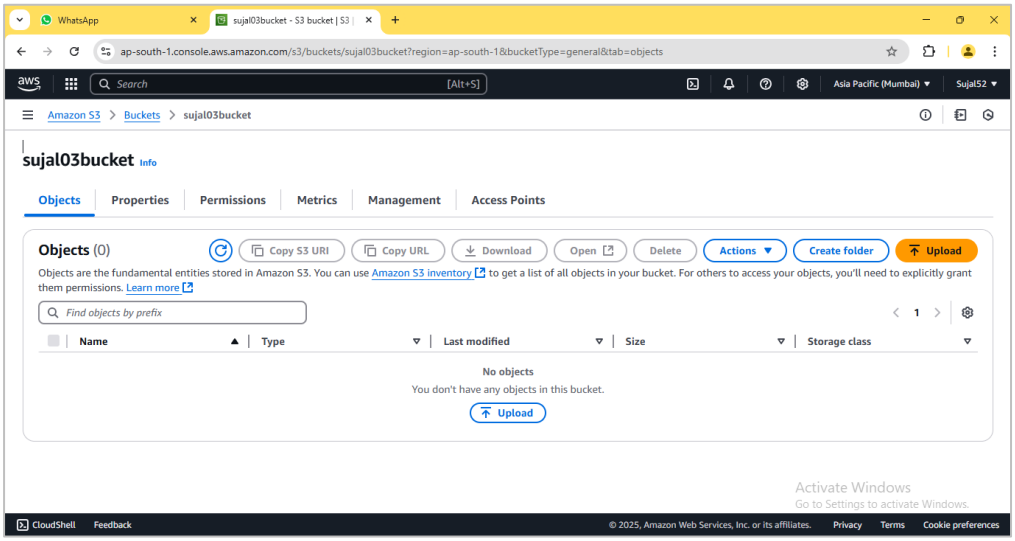




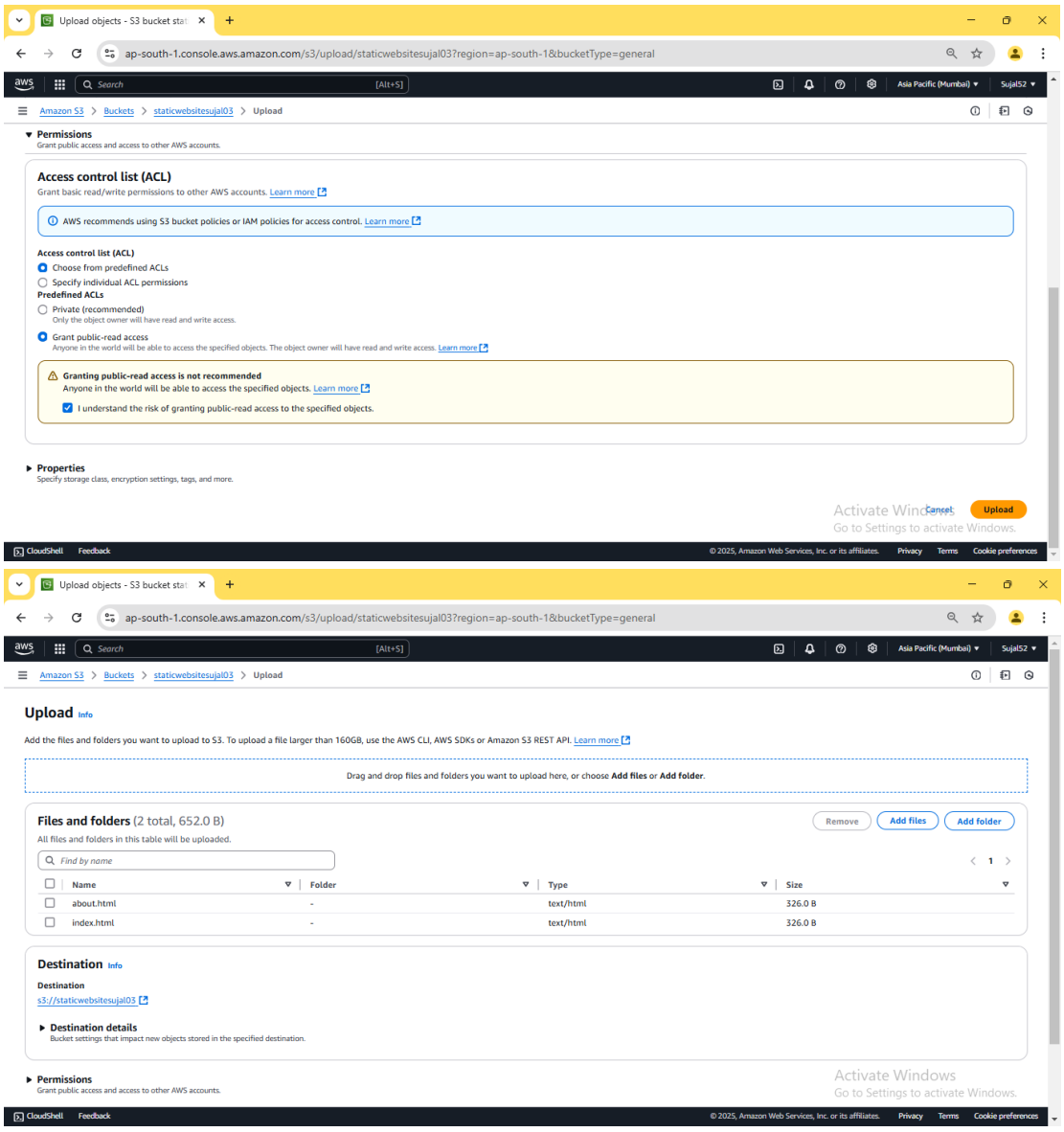
4. A green coloured pop-up dialogue box shall confirm the creation of our bucket. It shall also be displayed on the bucket list.



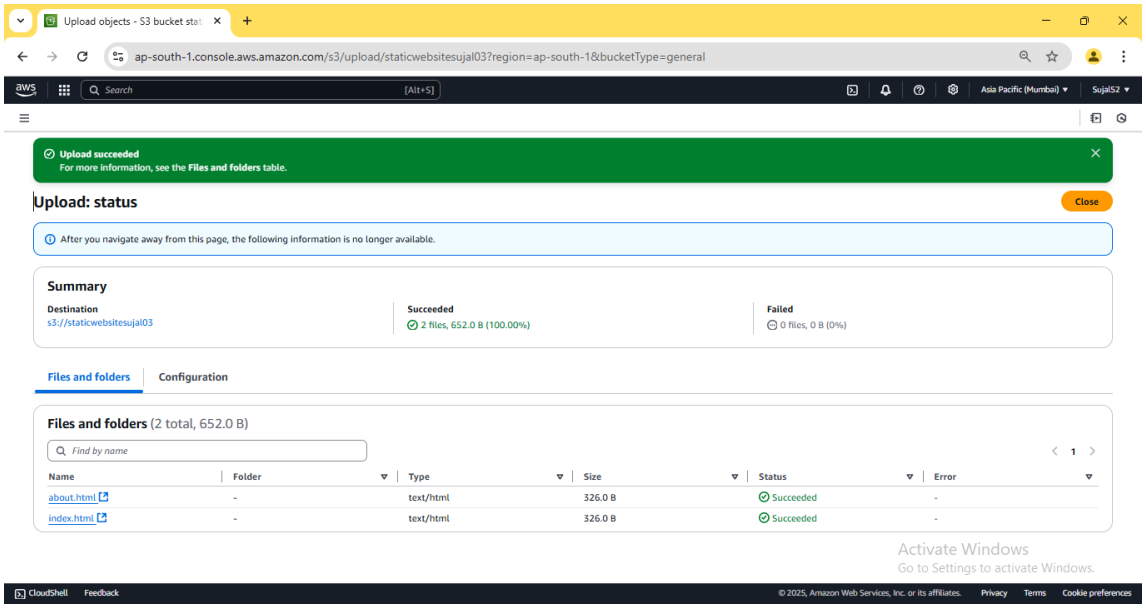
5. Click on the name of our newly created bucket. On the bucket detail page, click on ***"Upload"***.



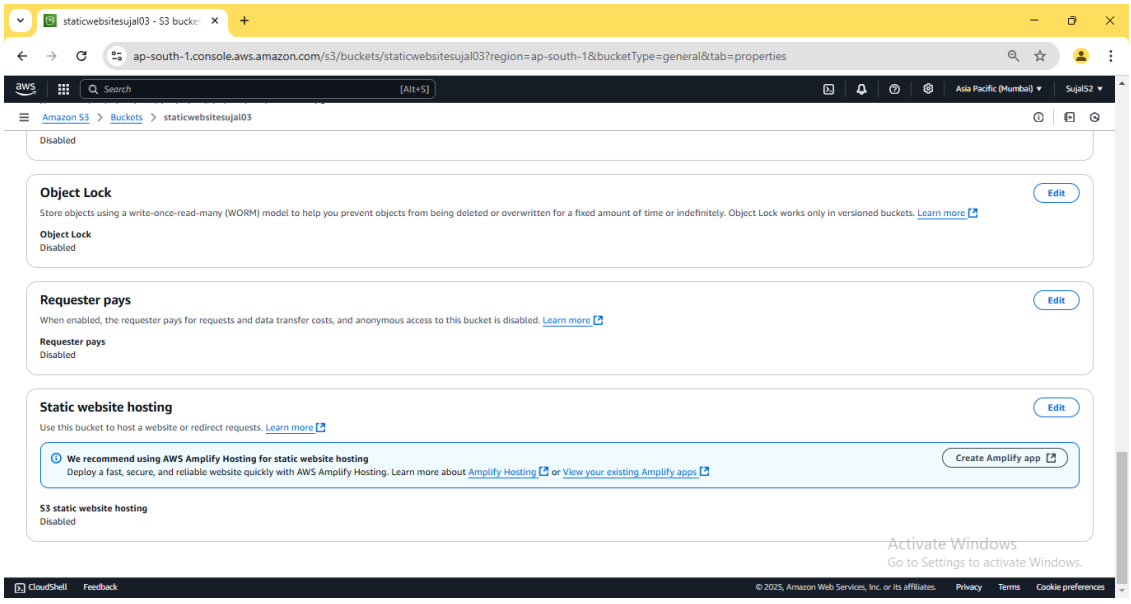
6. Click on ***"Add files"***. Select our html files from the list. Expand ***"Permissions"*** tab, select ***"Grant public read access"*** and check the confirmation popup. Finally, click on ***"Upload"*** button.



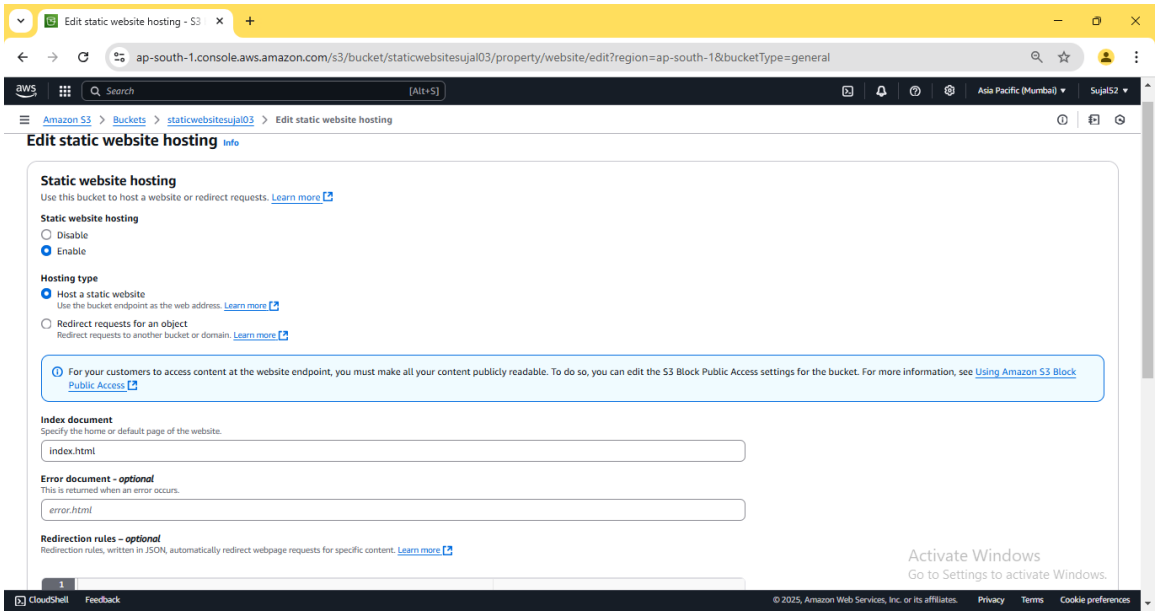
7. The notification confirms that the file has been successfully uploaded. Click on ***"Close"*** to return to our bucket details page.



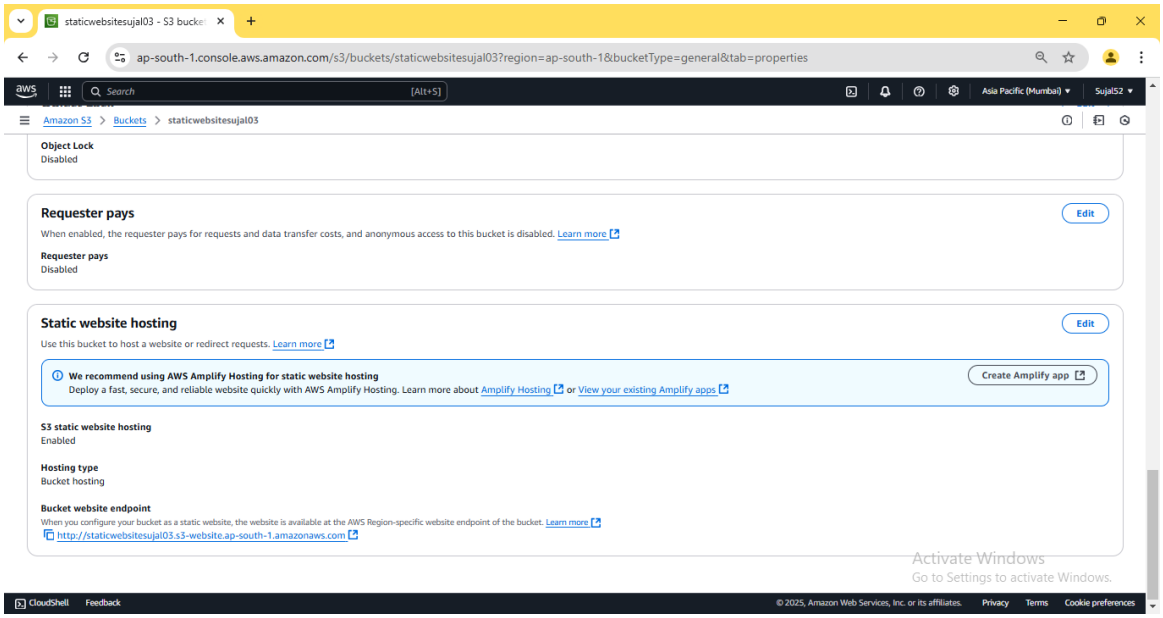
8. From the bucket details page, click on ***"Properties"*** tab, scroll down and click on ***"Edit"*** next to ***"Static website hosting"***.



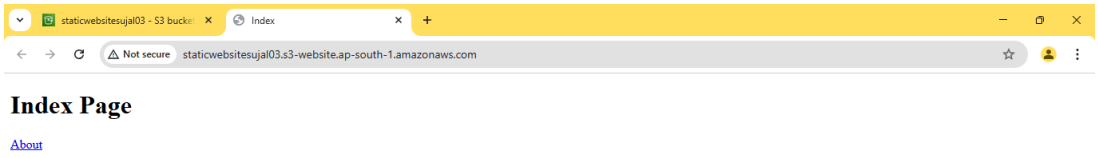
9. Under ***"Static website hosting"***, click on ***"Enable"***, then fill in the name of index document and click on "***Save changes"***.



10. Once again, scroll down to ***"Static website hosting"*** and copy the URL under ***"Bucket website endpoint"***.



11. Open the copied URL in a new browser window. Our website should be accessible now.

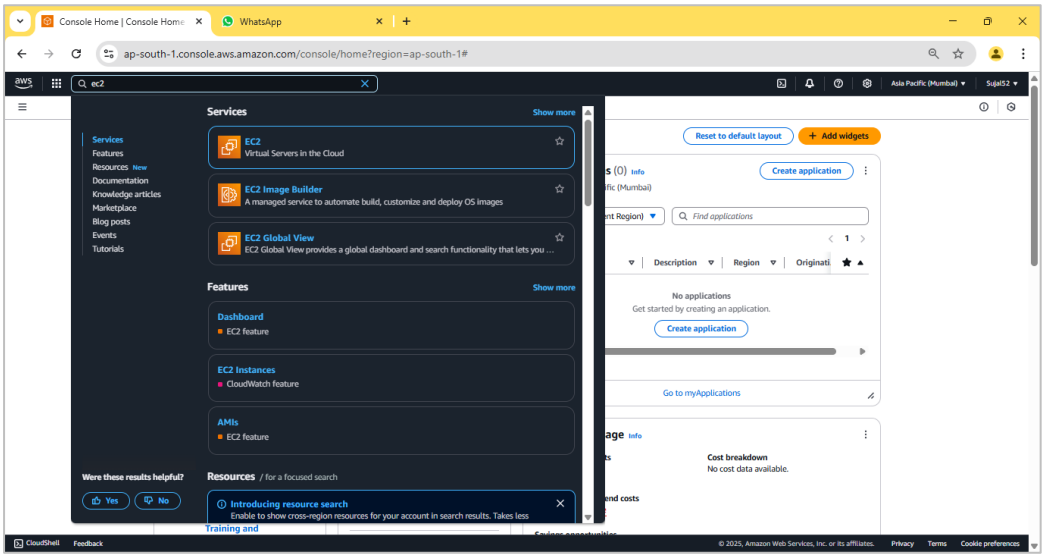


ASSIGNMENT 7:

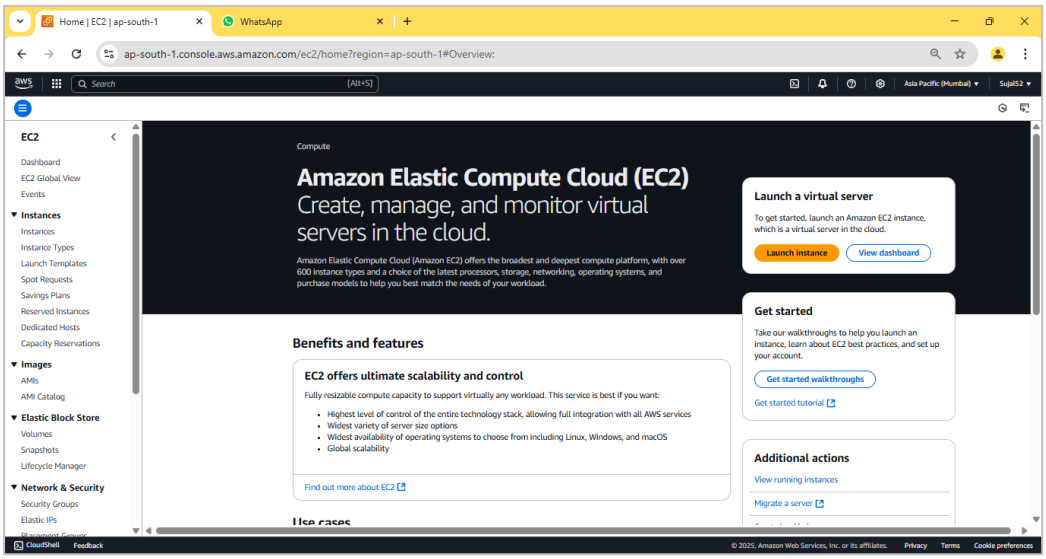
Problem Definition: Hosting a website on EC2.

Instructions:

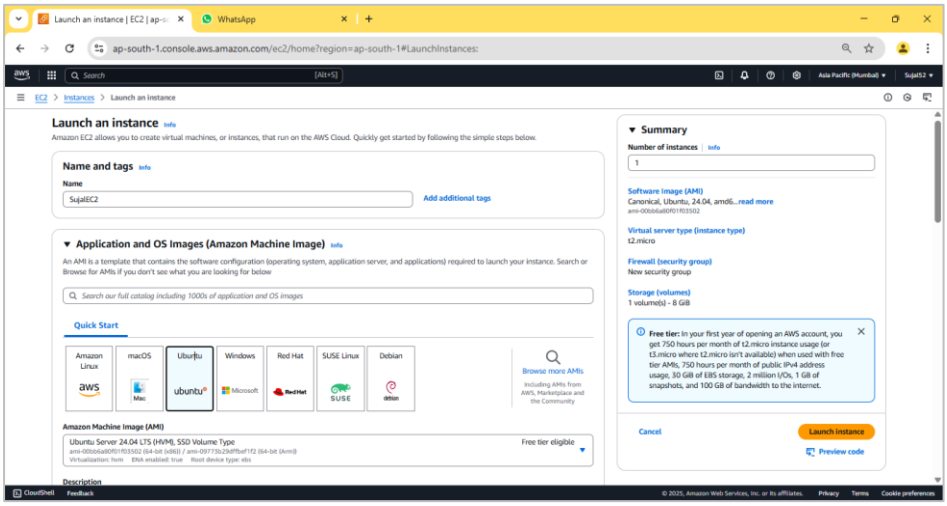
1. Access the ***AWS console***, search for ***EC2***, and select the top option from the search results.

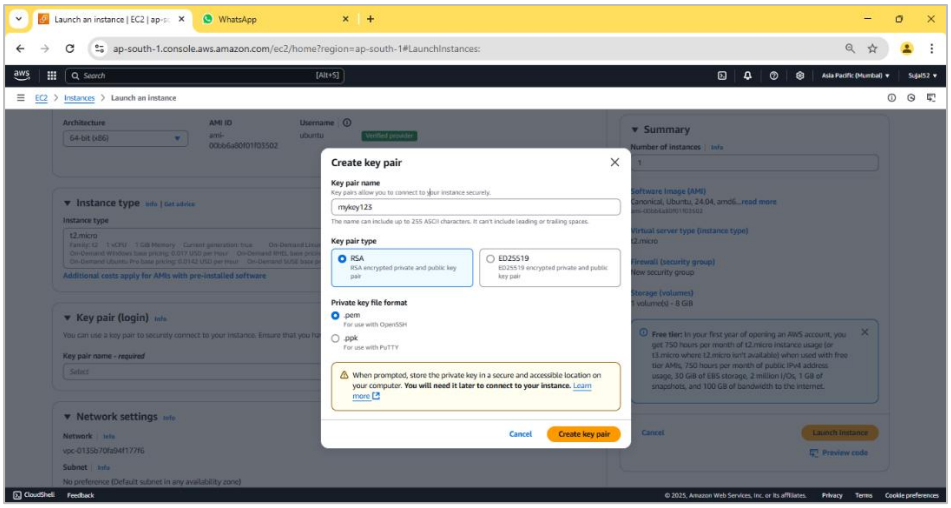


2. Click on ***"Launch instance"***.

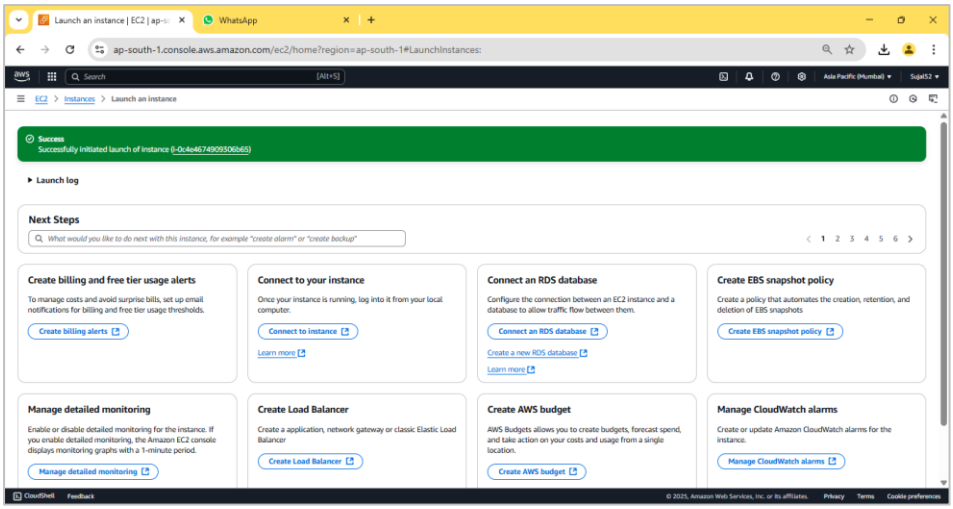


3. Enter instance name and select ***Ubuntu*** under ***"Application and OS Images".*** Under ***"Key Pair (login)"*** click on ***"Create new key pair"***, enter key pair name then click on ***"Create key pair"*** button, a *.pem file will be downloaded. Then click on ***"Launch instance"*** button

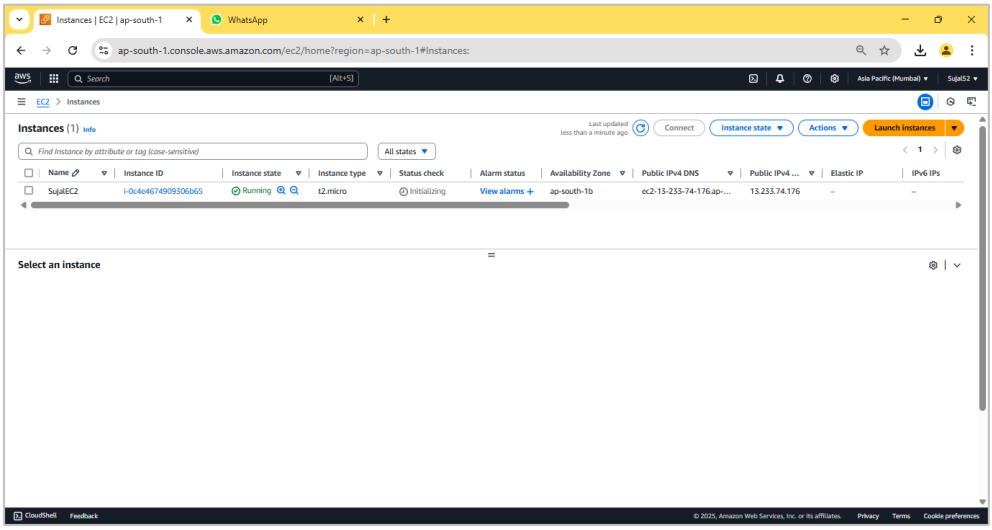




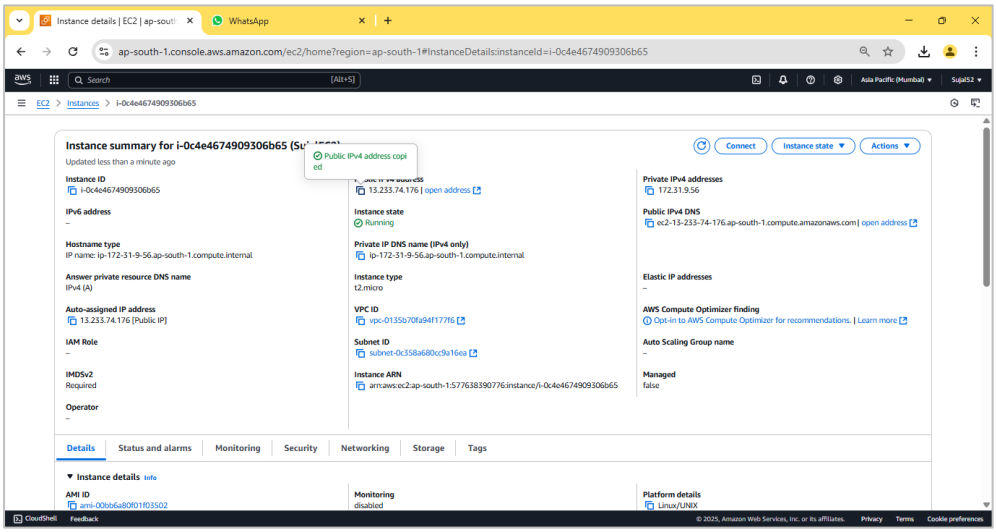
4. Our instance is created successfully.



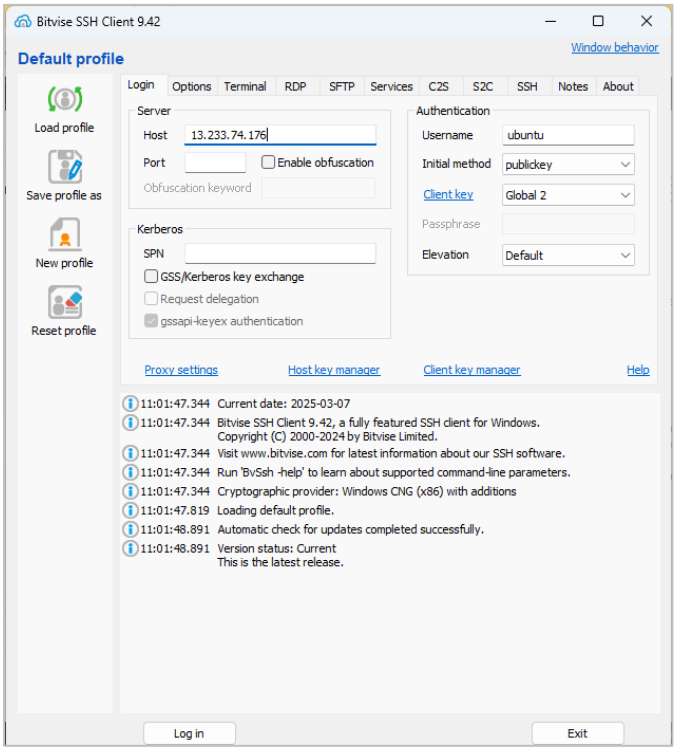
5. Go to ***"Instance"*** from the sidebar, then click on ***Instance ID*** of our newly created instance.



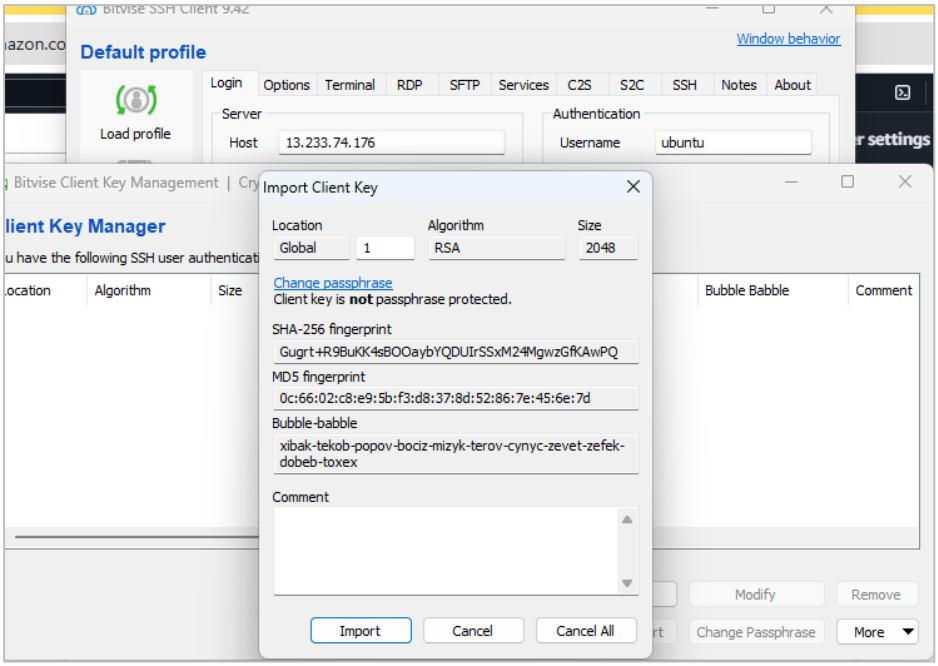
6. Copy ***"Public IPV4 address"***.



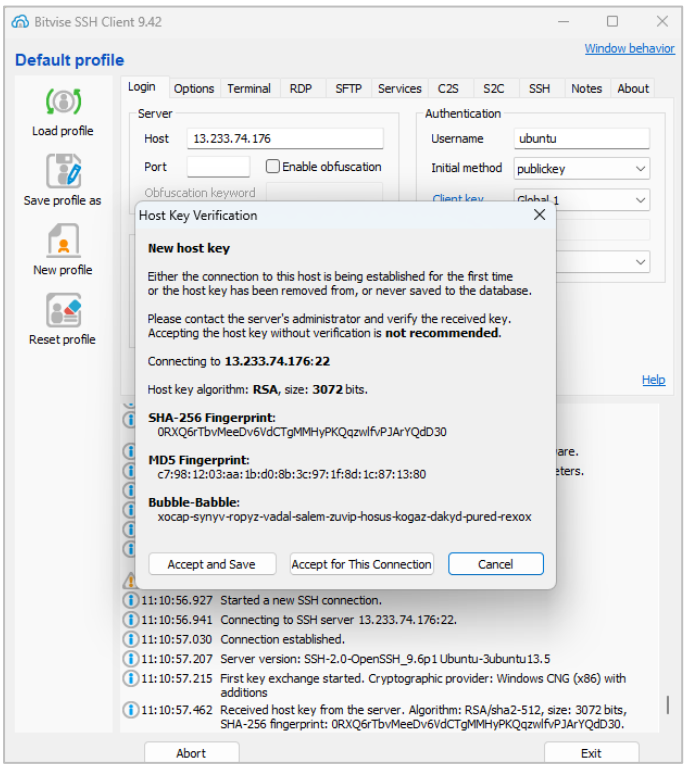
7. Open ***Bitvise SSH Client,*** paste the previously copied address in host field.



8. Click ***"Client key manager",*** click on ***"Import"*** and select our previously downloaded key pair file. Click on ***"Import"*** button. Our key pair should be imported successfully.



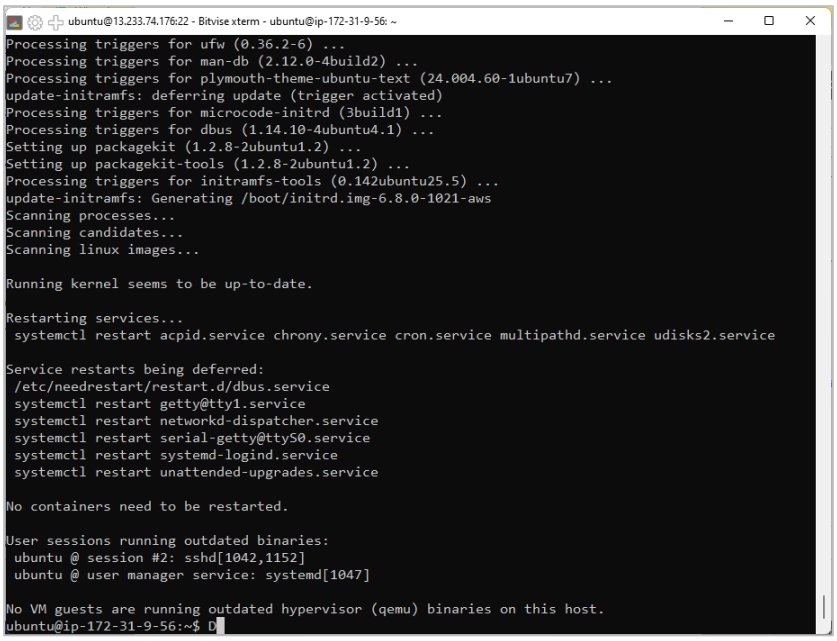
9. In ***"Bitvise SSH Client"***, under ***"Authentication"*** give the username as ***"ubuntu"***, select ***"Global 1"*** under ***"Client Key"*** then click on ***"Log in"*** & ***"Accept and save"***.



10. Click on ***"New terminal console"*** from the sidebar. In the terminal enter the following commands:

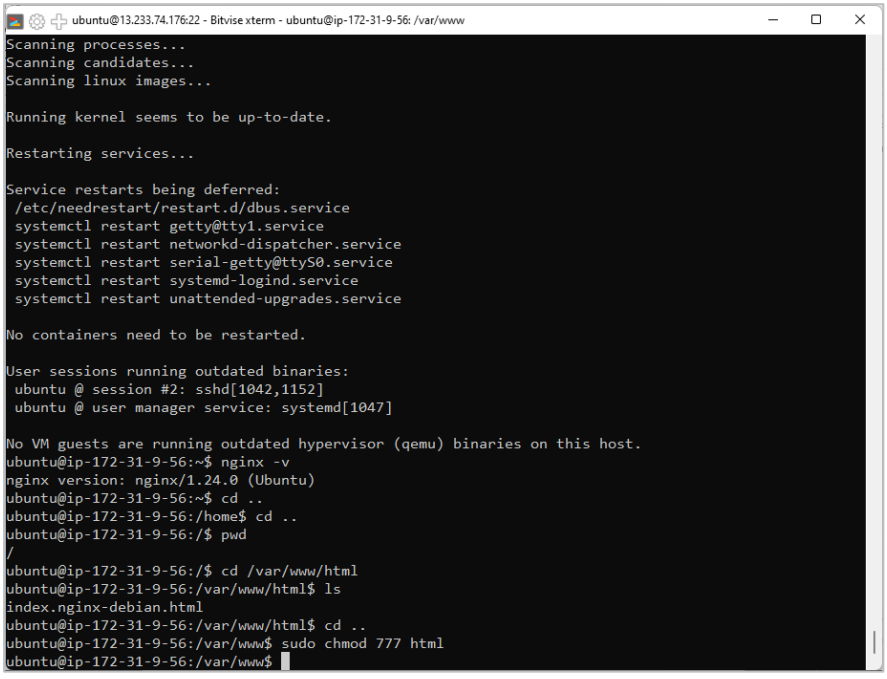
```
sudo apt-get update && sudo apt-get upgrade
```

sudo apt-get install nginx



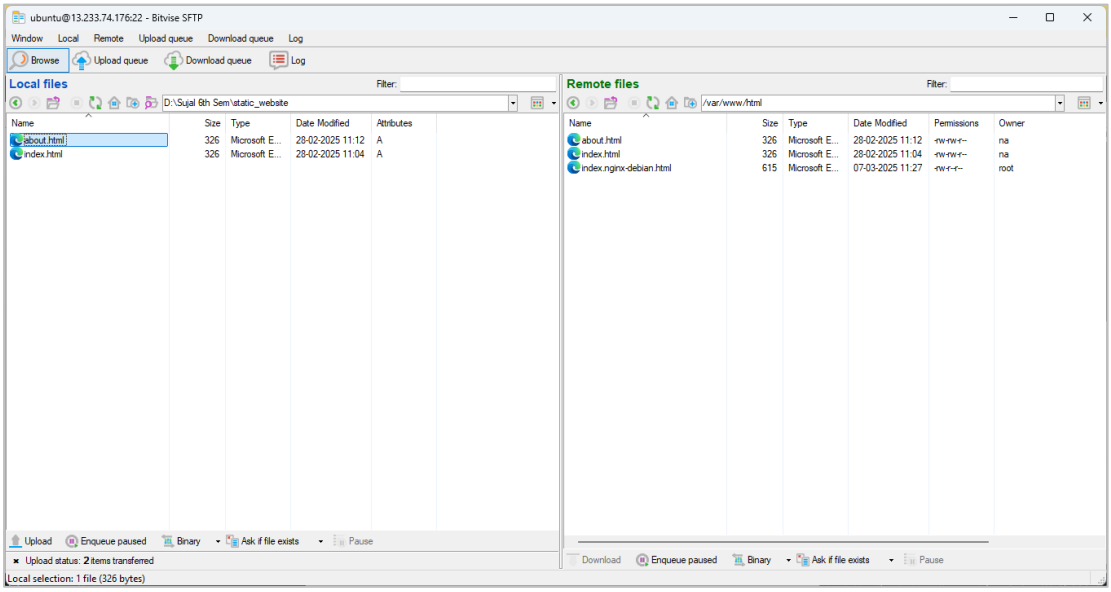
11. We need to move our html files into var/www/html directory to host it, but first we need to give write permission to do so. To change file permission enter the following command:
cd var/www/

sudo chmod *777* html

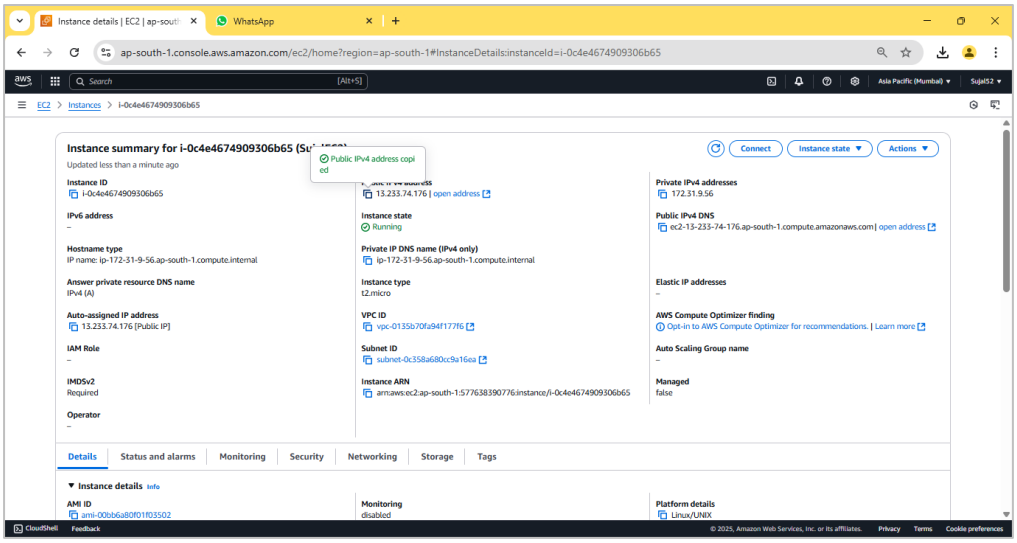


12. Go to SFTP window from the sidebar, under ***Local file*** open the

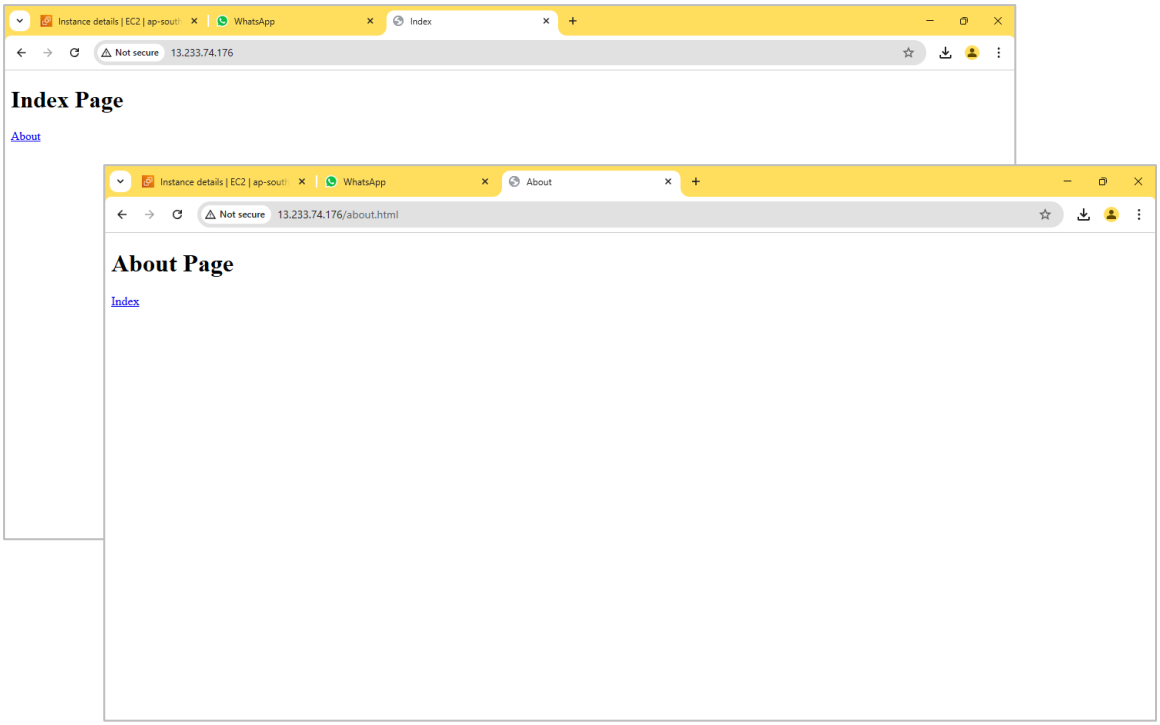
folder containing html files and under ***Remote file*** open /var/www/html. Drag and drop the HTML files from local to remote.



13. Now go back to the AWS window and copy the ***"Public IPV4 address"***.



14. Open the copied address in a new browser window. Our website should be accessible now.

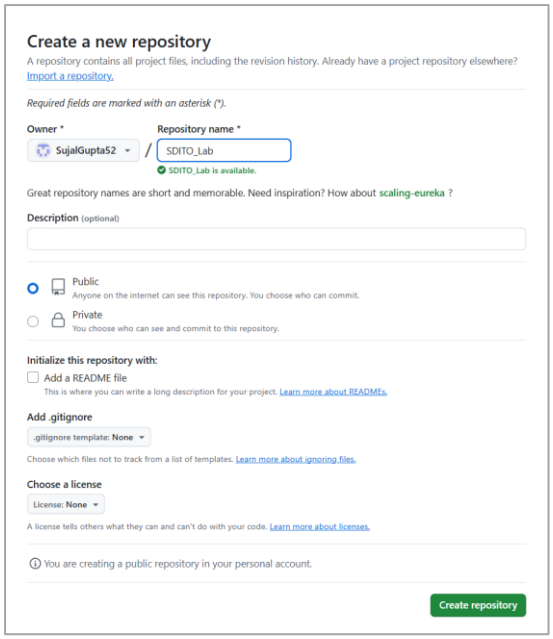


ASSIGNMENT 8:

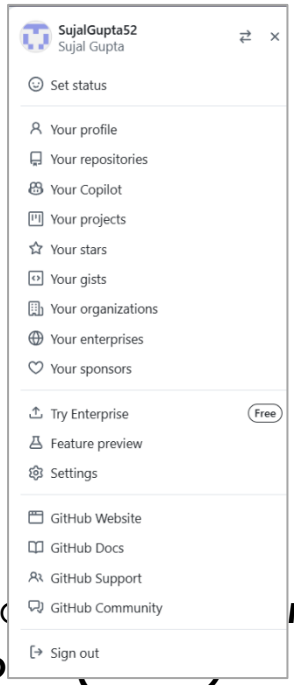
Problem Definition: Deploy a project from a local machine to GitHub and vice versa.

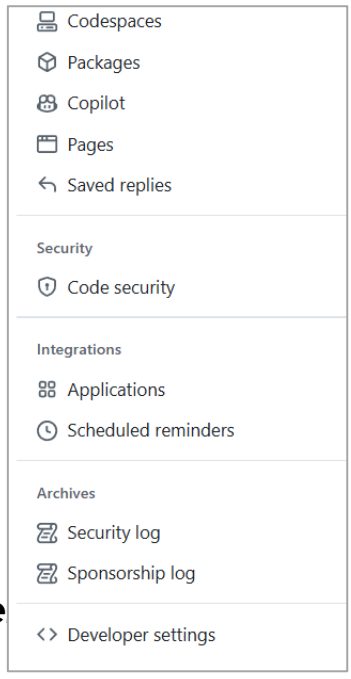
Instructions:

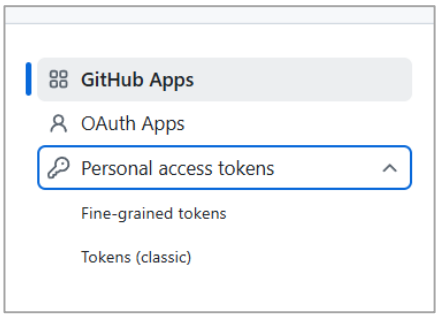
1. Create a new public Github repository. Click on ***"Create repository"*** to proceed.



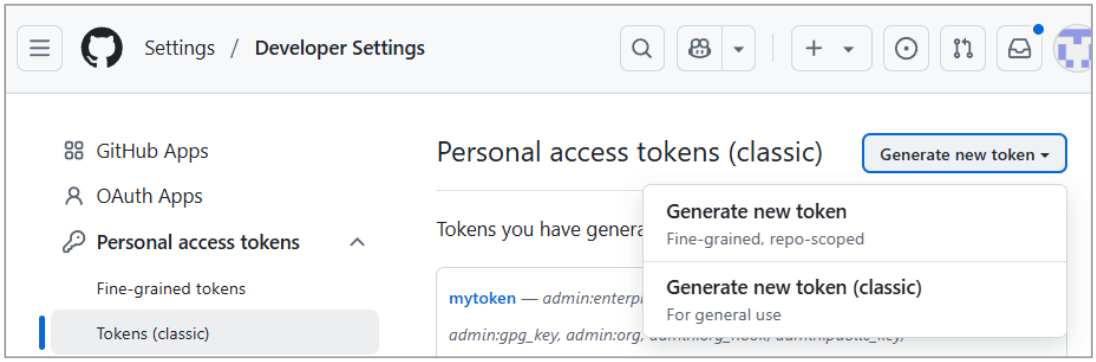
2. Go to ***Settings*** then ***Developer settings***. Now click on ***"Personal access tokens"*** dropdown and then click on ***"Tokens(classic)"***.



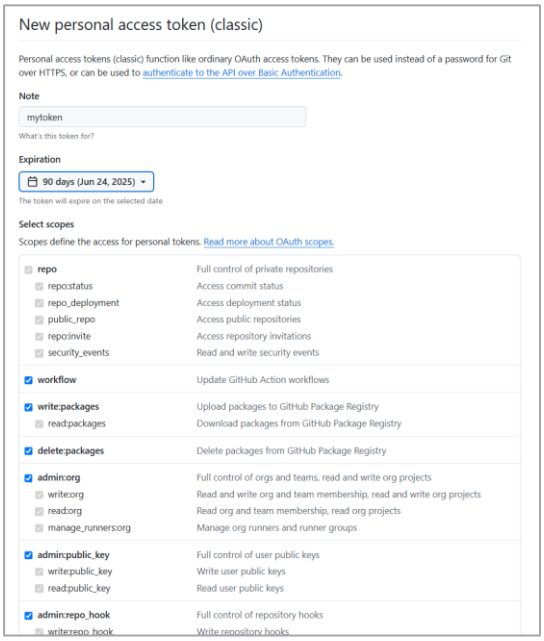


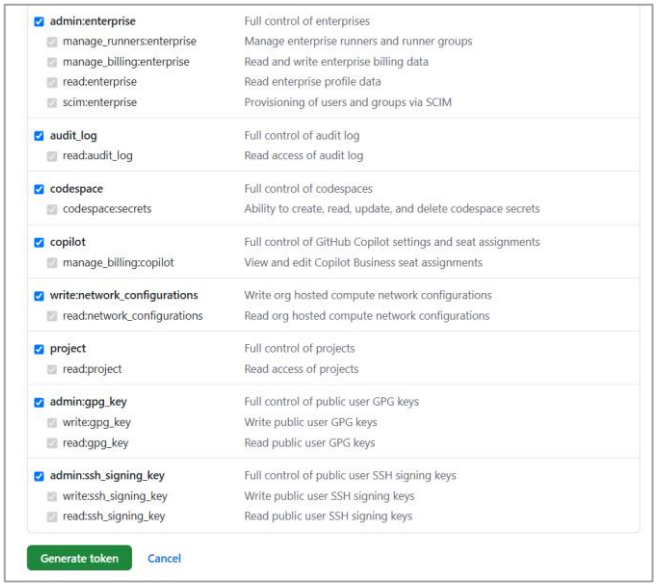


3. N**ne**nd then ***"Generate new to***

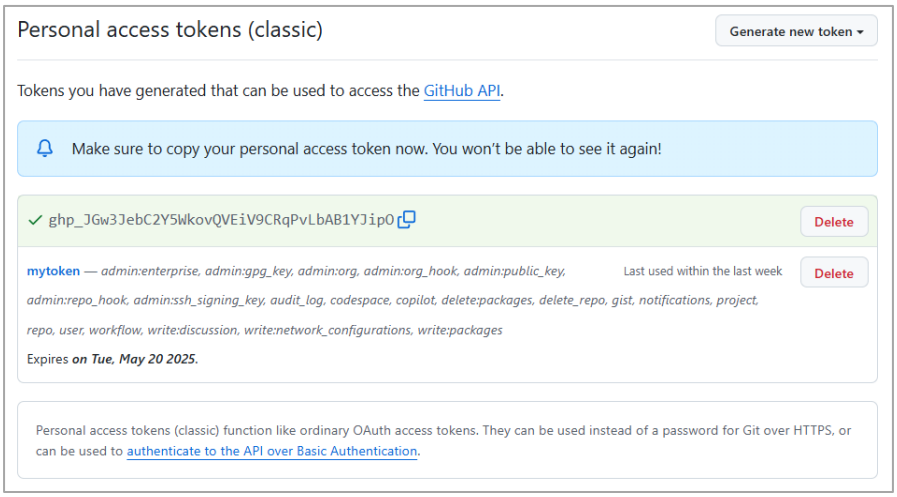


4. Set expiration date for as long as possible and select all scopes. Then click on ***"Generate token"***.

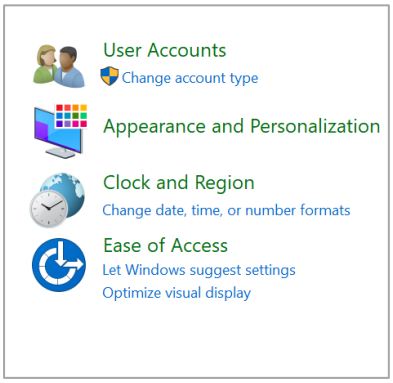


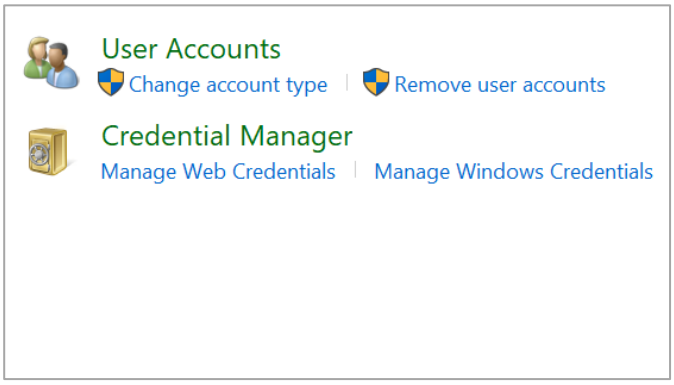


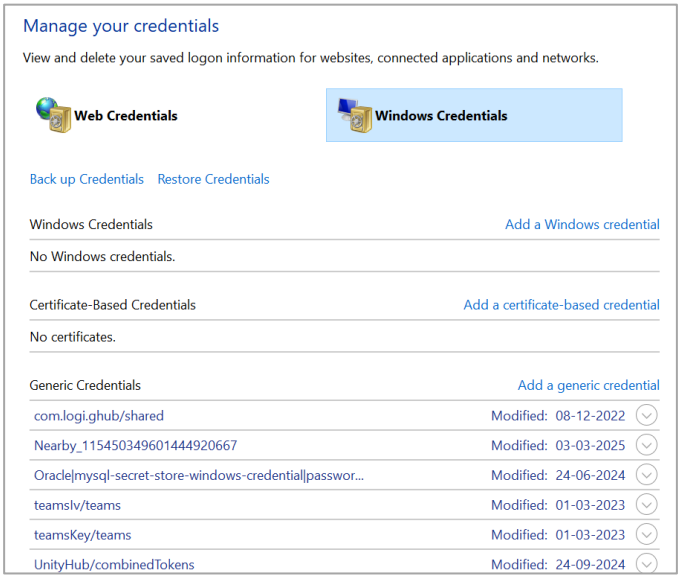
5. After generating the token, click on copy icon and copy the token somewhere secure.



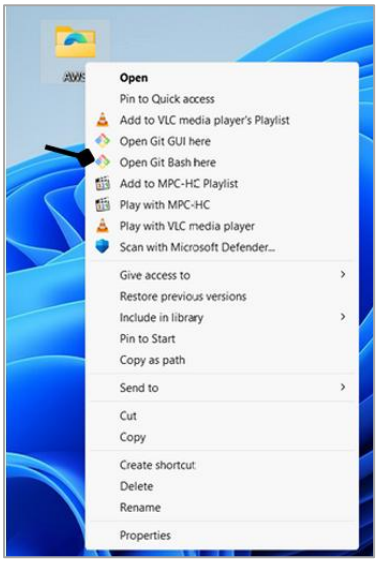
6. Now go to ***Control Panel***, click ***User's account***, then go to ***Credential Manager*** and click on ***Window Credentials***. From ***Generic Credentials*** remove any existing GitHub account credentials.







7. Go to the folder containing your project files. Right Click, and select ***"Open Git Bash Here"*** from the pop-up menu.



8. Run the following commands:

git init

git add .

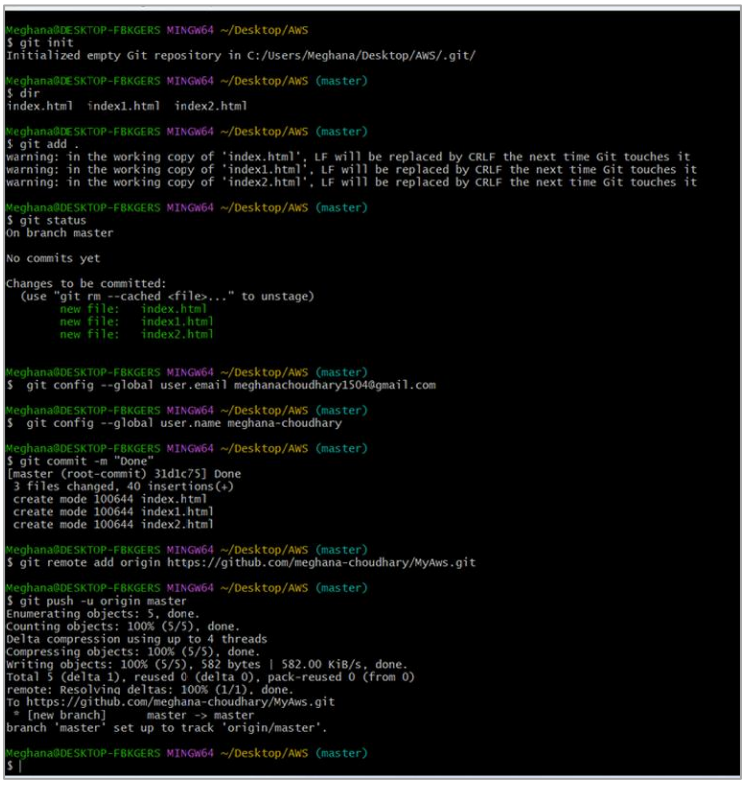
git status

git config --global user.email <user email address>

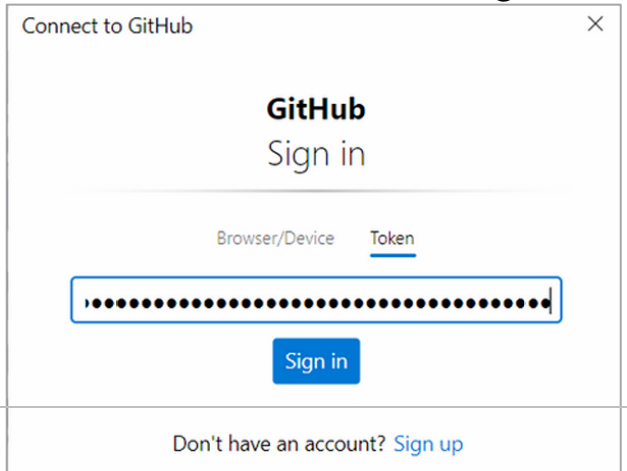
git config --global user.name <username>

git commit -m “<message>”

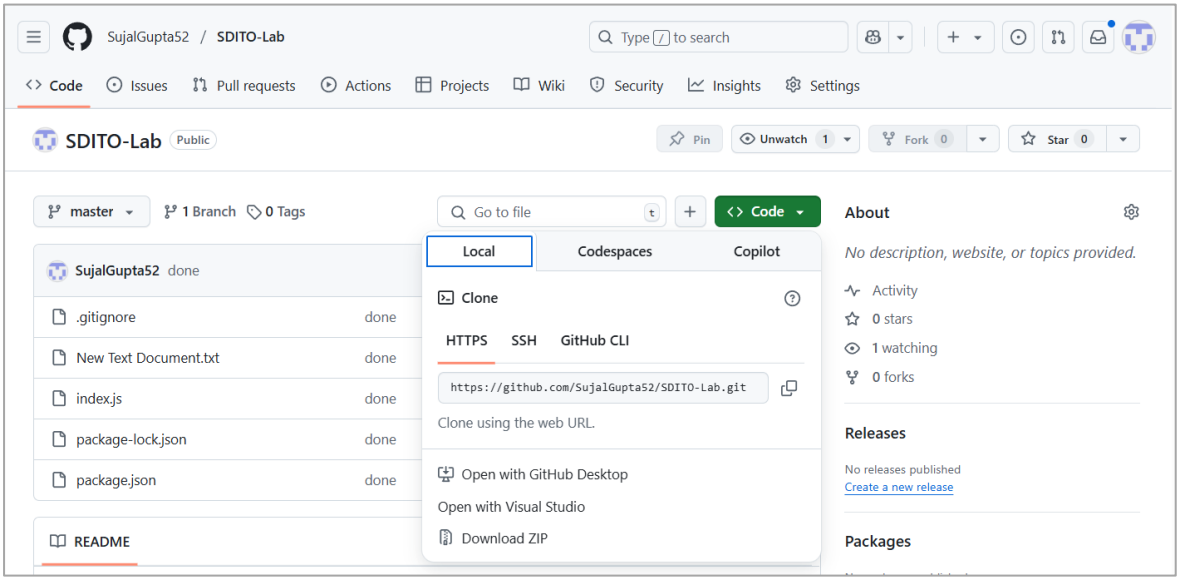
git push -u origin master



9. After the last command, a new ***“Connect to GitHub”*** window should open. Go to ***“Token”*** tab and paste the previously generated token into the text field. Click on ***“Sign in”*** button.



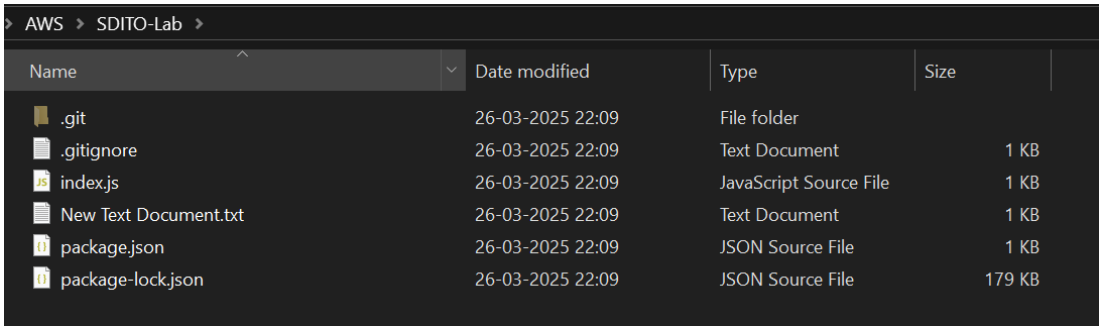
10. Open the repository in **Github**. Our project files should now be visible. Now click on ***"Code"*** button and copy the HTTPS URL.



11. Now make a new folder and open the newly created folder in git bash terminal. Run the following command to clone our repo:
    `git clone <Repository address>`

```
PS C:\Users\sujal\OneDrive\Desktop\AWS> git clone git@github.com:SujalGupta52/SDITO-Lab.git
Cloning into 'SDITO-Lab'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 7 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (7/7), 48.22 KiB | 137.00 KiB/s, done.
```

12. Our repo files is successfully cloned locally.

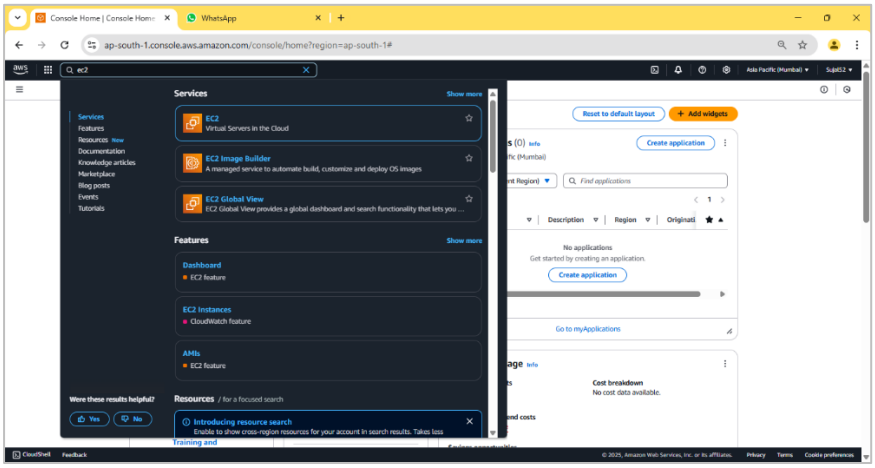


ASSIGNMENT 9:

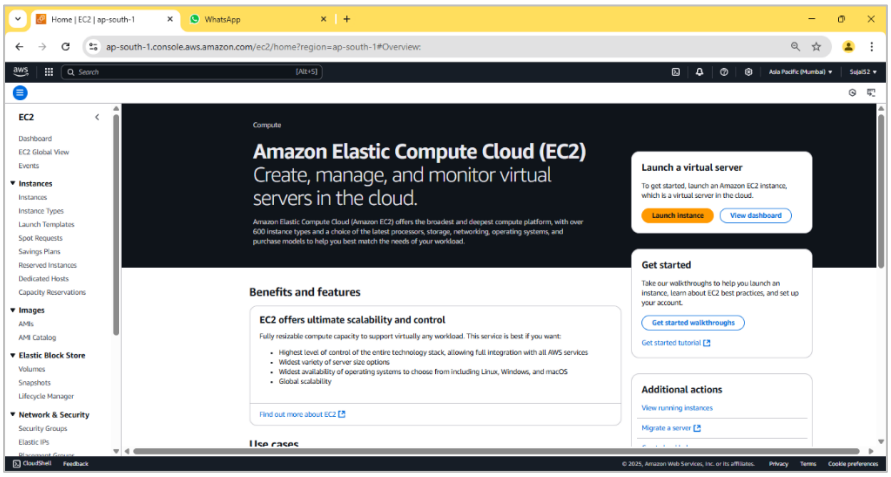
Problem Definition: Deploy a project from GitHub to EC2.

Instructions:

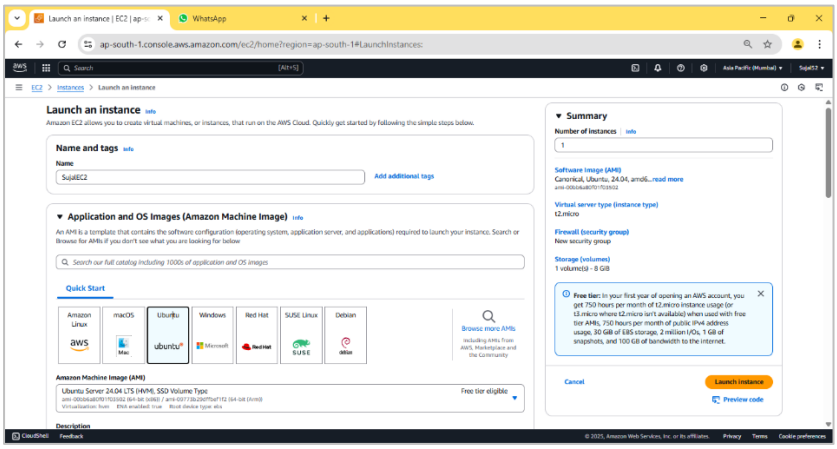
1. Access the ***AWS console***, search for ***EC2***, and select the top option from the search results.

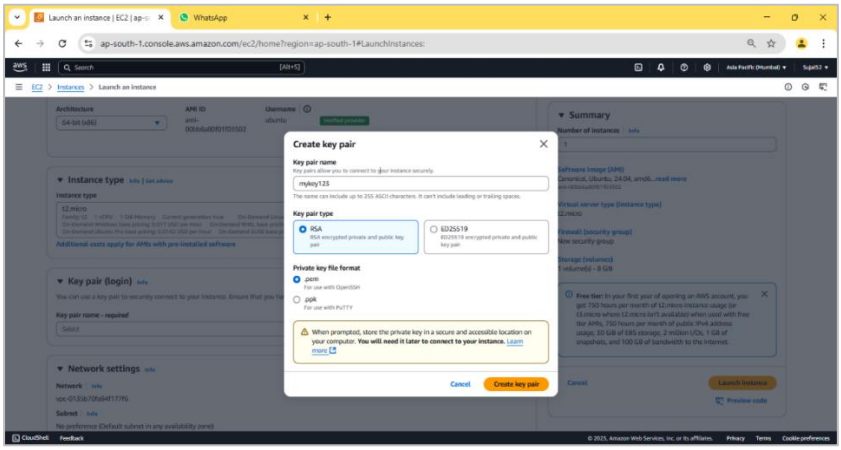


2. Click on ***"Launch instance"***.

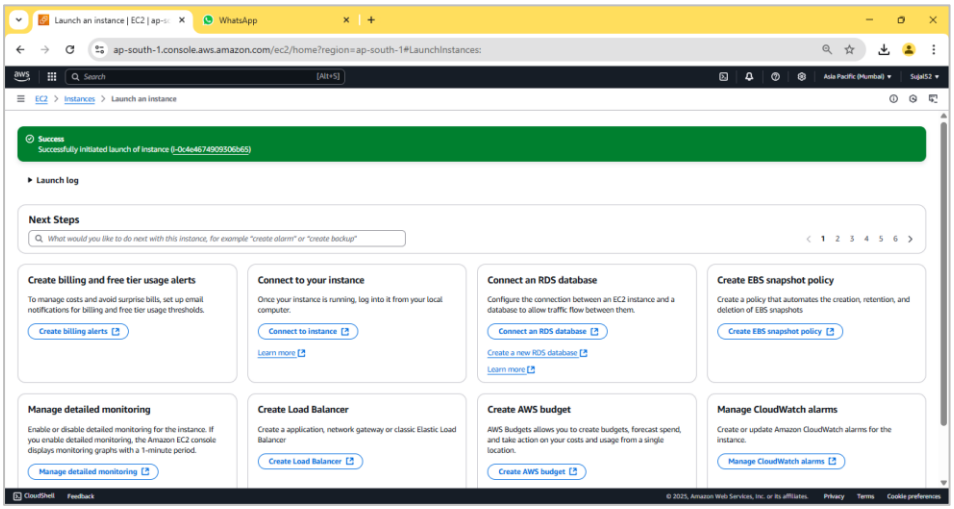


3. Enter instance name and select ***Ubuntu*** under ***"Application and OS Images".*** Under ***"Key Pair (login)"*** click on ***"Create new key pair"***, enter key pair name then click on ***"Create key pair"*** button, a *.pem file will be downloaded. Then click on ***"Launch instance"*** button

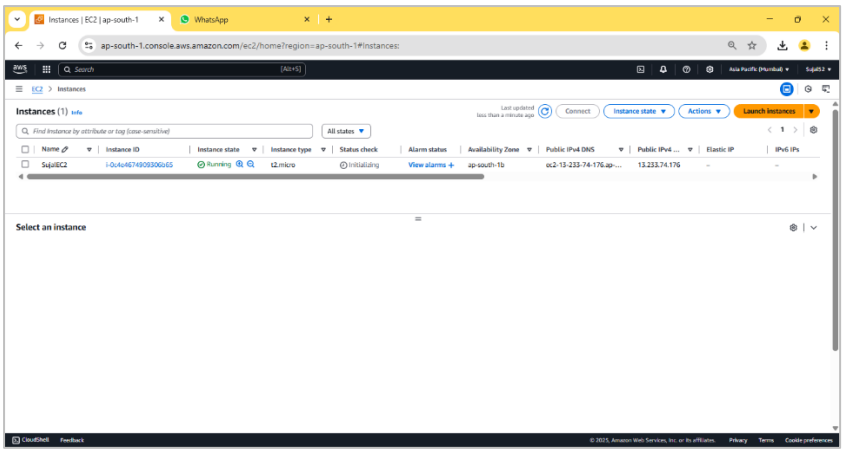




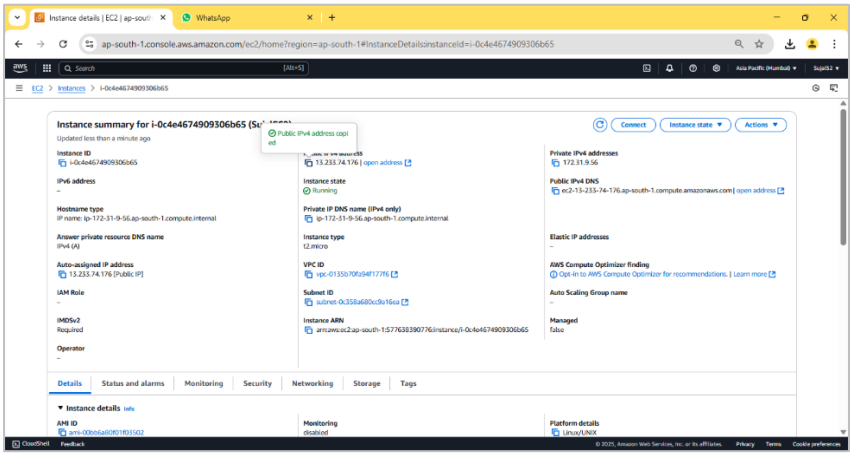
4. Our instance is created successfully.



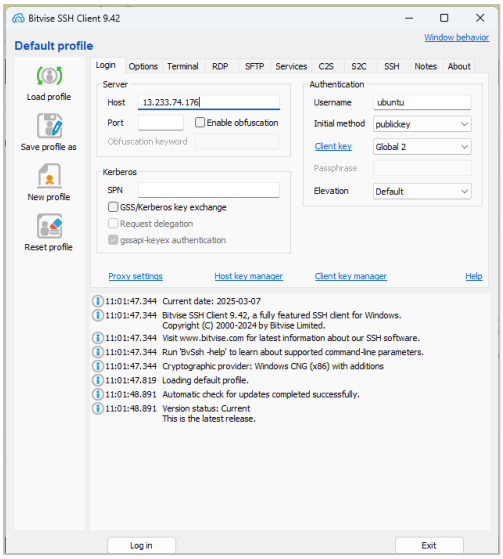
5. Go to ***"Instance"*** from the sidebar, then click on ***Instance ID*** of our newly created instance.



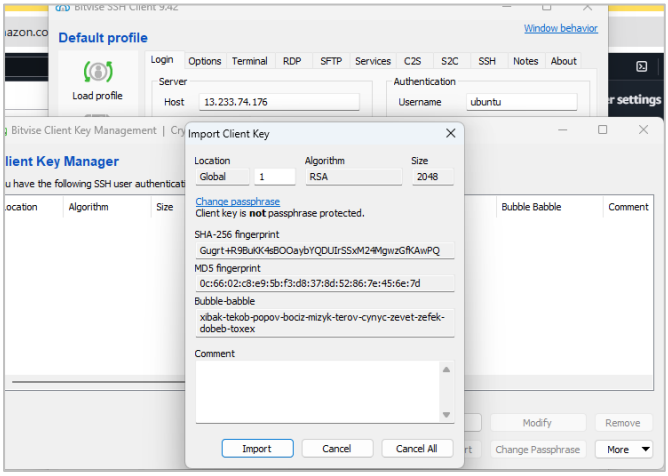
6. Copy ***"Public IPV4 address"***.



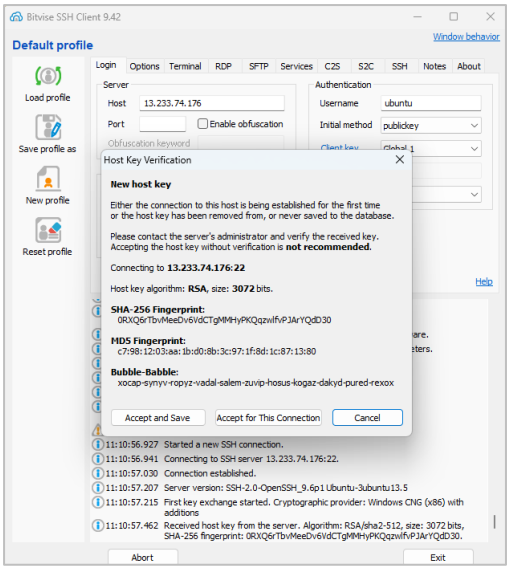
7. Open ***Bitvise SSH Client,*** paste the previously copied address in host field.



8. Click ***"Client key manager",*** click on ***"Import"*** and select our previously downloaded key pair file. Click on ***"Import"*** button. Our key pair should be imported successfully.



9. In ***"Bitvise SSH Client"***, under ***"Authentication"*** give the username as ***"ubuntu"***, select ***"Global 1"*** under ***"Client Key"*** then click on ***"Log in"*** & ***"Accept and save"***.

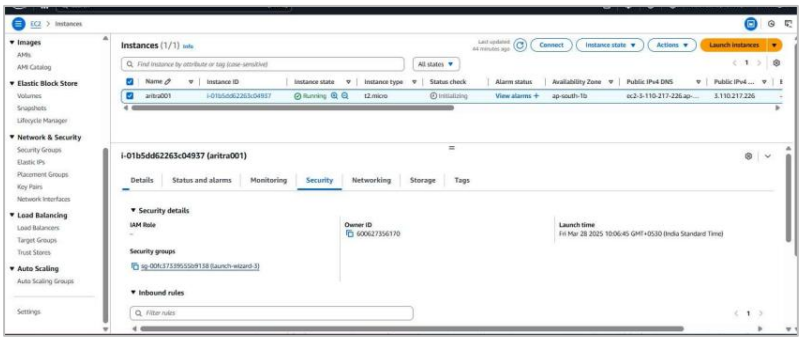


10. Click on ***"New terminal console"*** from the sidebar. In the terminal enter the following commands to clone our repo and run our Node server:

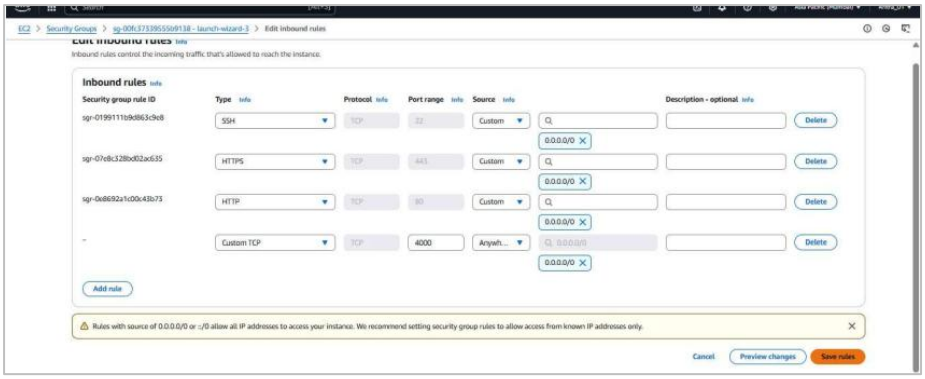
```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash –
sudo apt install nodejs
git clone <Github repo link>
cd <Github repo name>
npm install
node index.js
```

```
ubuntu@ip-172-31-13-131:~$ ls
Aritra_01
ubuntu@ip-172-31-13-131:~$ cd Aritra_01
ubuntu@ip-172-31-13-131:~/Aritra_01$ ls
index.js  package-lock.json  package.json
ubuntu@ip-172-31-13-131:~/Aritra_01$ npm install
npm warn deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math
.random() in certain circumstances, which is known to be problematic.  See https://v8.dev/blog/math-
random for details.

added 258 packages, and audited 259 packages in 8s

18 packages are looking for funding
  run `npm fund` for details

23 vulnerabilities (3 low, 2 moderate, 16 high, 2 critical)

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.2.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.2.0
npm notice To update run: npm install -g npm@11.2.0
npm notice
ubuntu@ip-172-31-13-131:~/Aritra_01$ npm -v
10.8.2
ubuntu@ip-172-31-13-131:~/Aritra_01$ node index.js
Started server
```

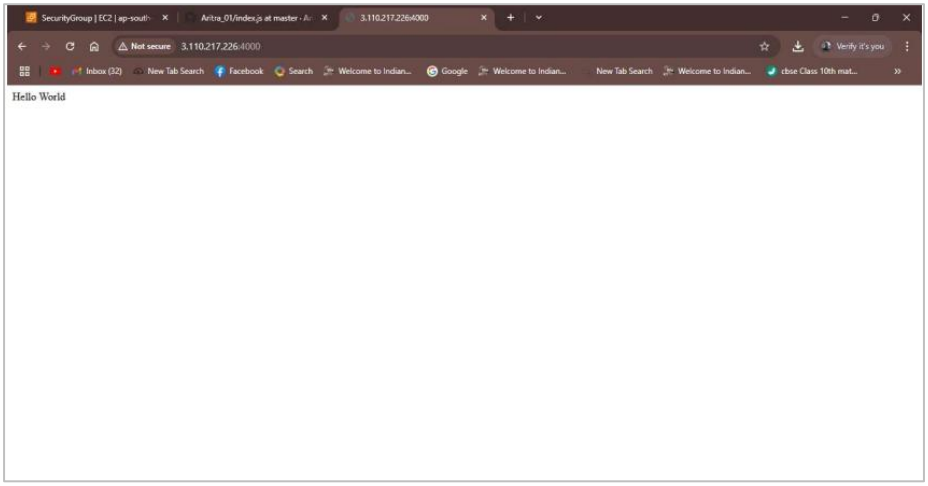
11. Go to ***"Instance Id",*** and click on our instance. Then go to 'Security' and click on link under 'Security groups' to get the edit option.



12. Then in ***"Inbound rules"***, click on ***"Edit inbound rules"***. Click on **"Add rule"** and set the ***"Port range"*** to **"4000"**, in ***"Source"*** set ***"0.0.0.0/0"*** & click on ***"Save rules"***. A confirmation is shown stating our rules are saved.



13. Now copy the ***"Public IPv4 address"*** & paste it on a new tab. Add ***":4000"*** at the end and then press enter. The project has been successfully deployed from GitHub to EC2.

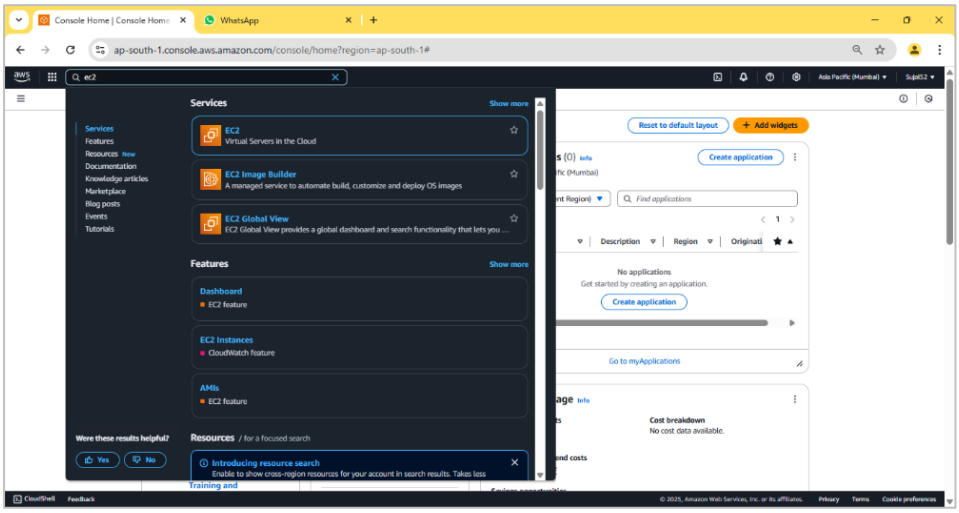


ASSIGNMENT 10:

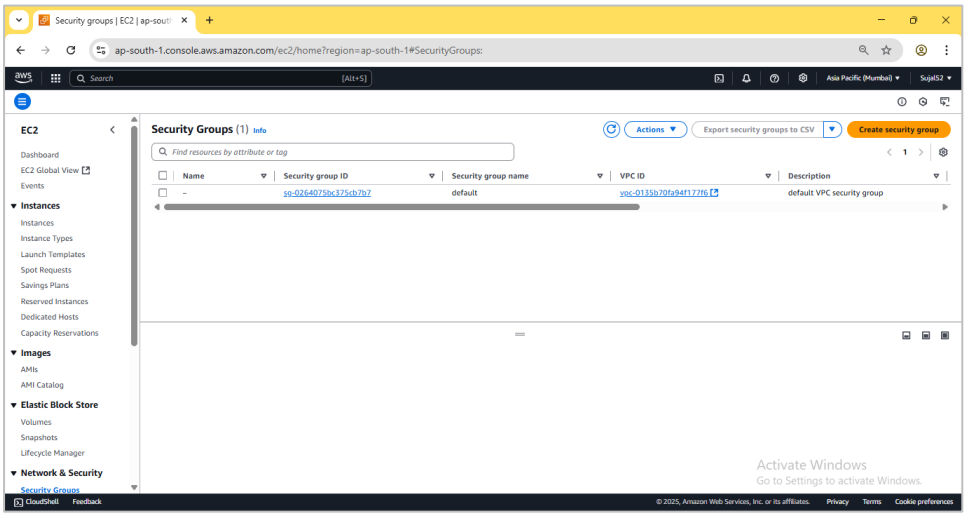
Problem Definition: Deploy a project from GitHub to EC2 by creating a new security group and user data.

Instructions:

1. Access the ***AWS console***, search for ***EC2***, and select the top option from the search results.

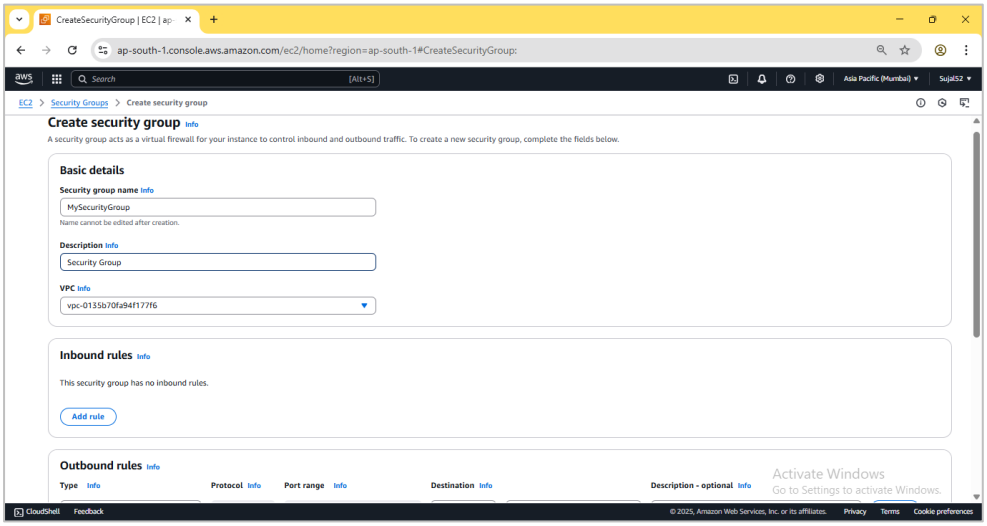


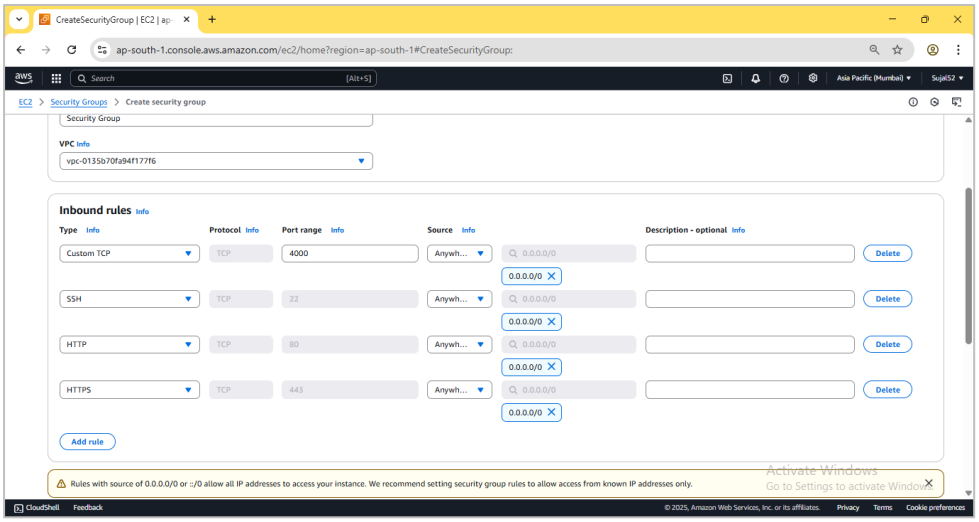
2. Select ***Security Groups*** from the side bar and click on ***"Create security group"***.

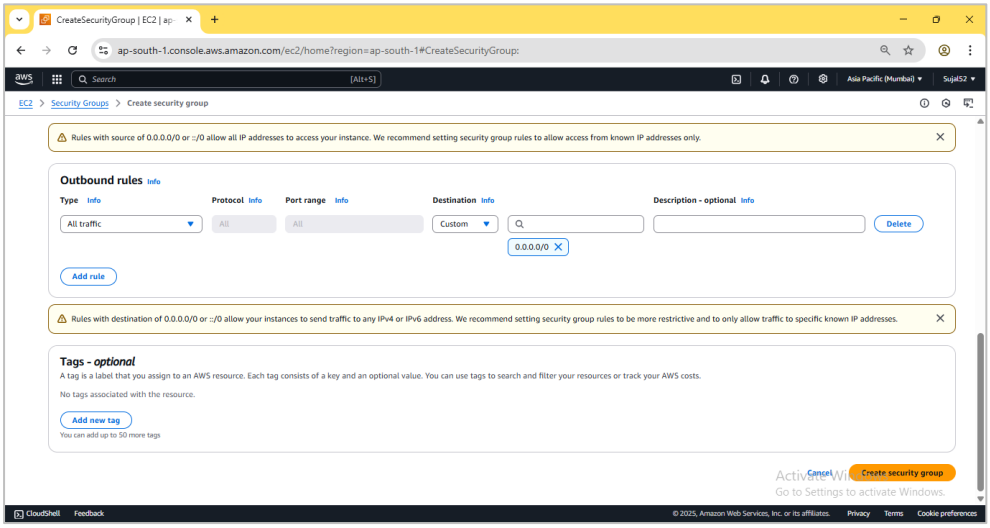


3. Add name and description. Under ***"Inbound Rules"***, add the ***"Add rule"*** to add the following rules: SSH, HTTP, HTTPS and Custom TCP with port 4000. Select 0.0.0.0/0 under source for all rules. Finally click

on ***"Create security group".*** Pop up is shown stating our rules are created successfully.

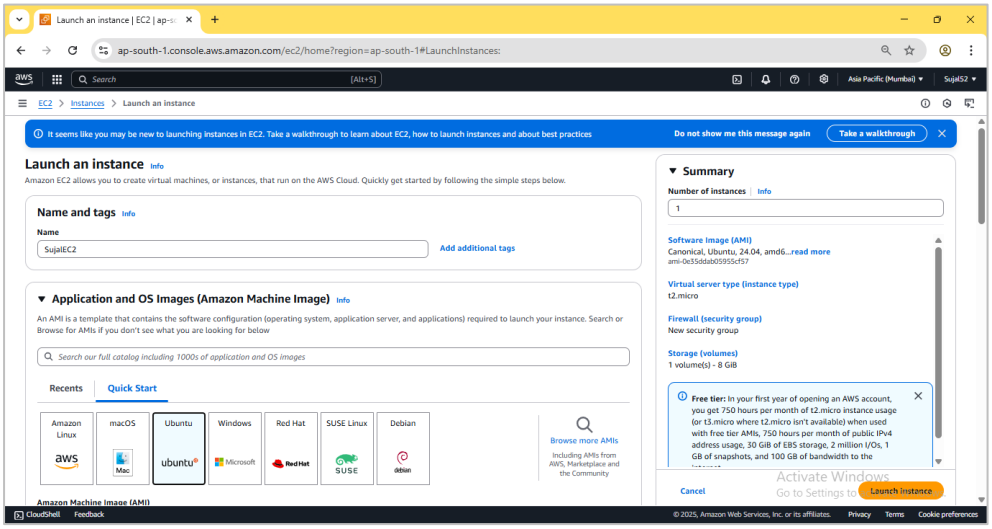


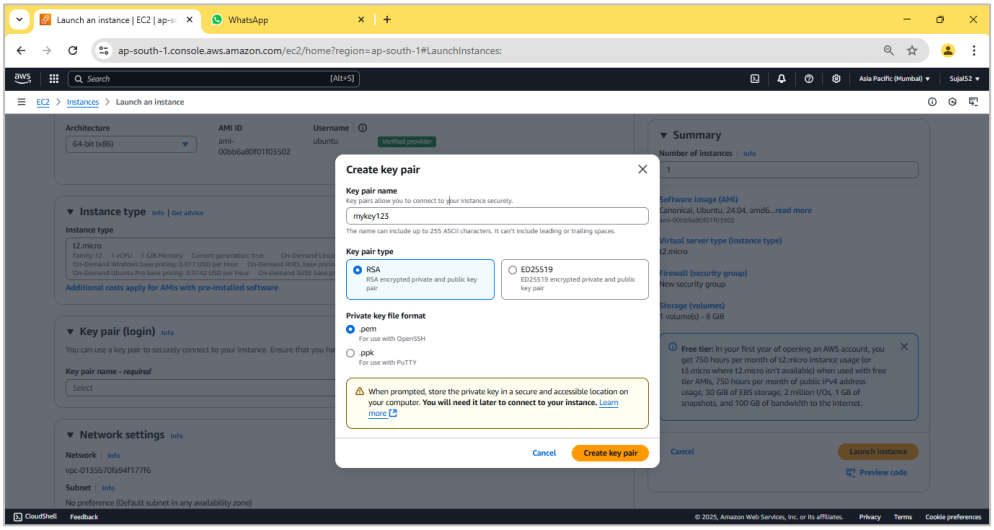


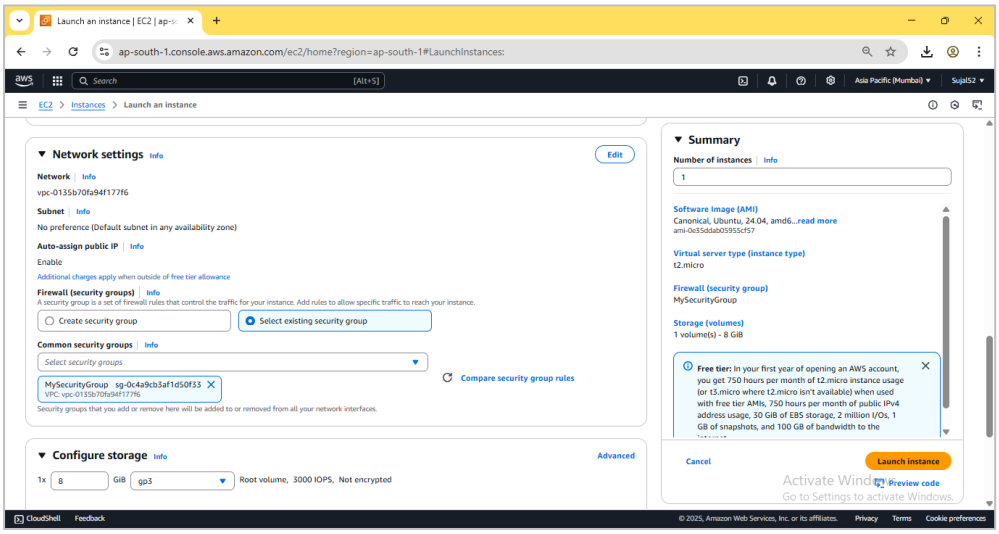


4. Go to ***Instances*** from the sidebar and click on ***"Launch Instance"***. Enter instance name, select Ubuntu as operating system, select an existing key pair or create a new one. Under ***"Network***

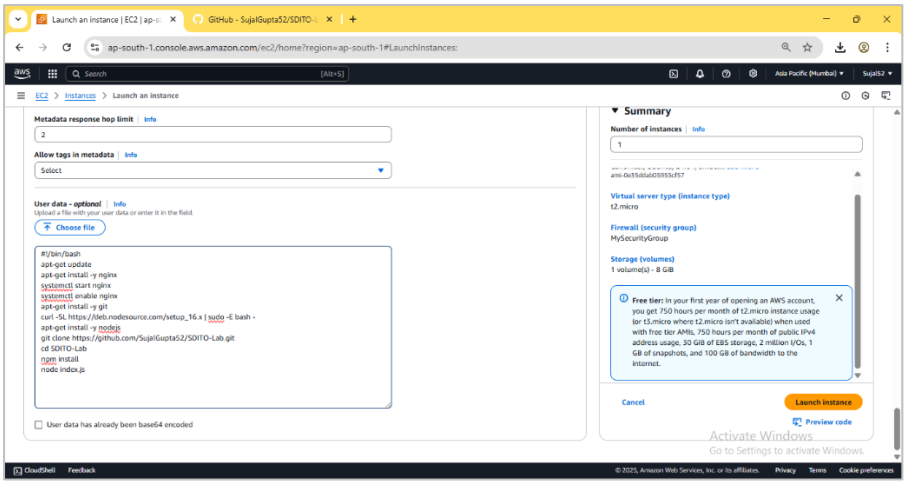
settings", click on ***"Select existing security group"*** and select our previously created security group.



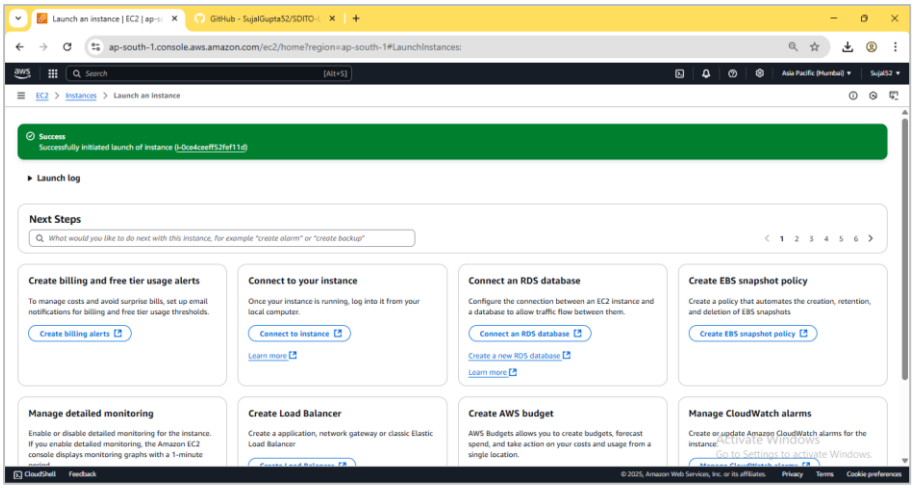




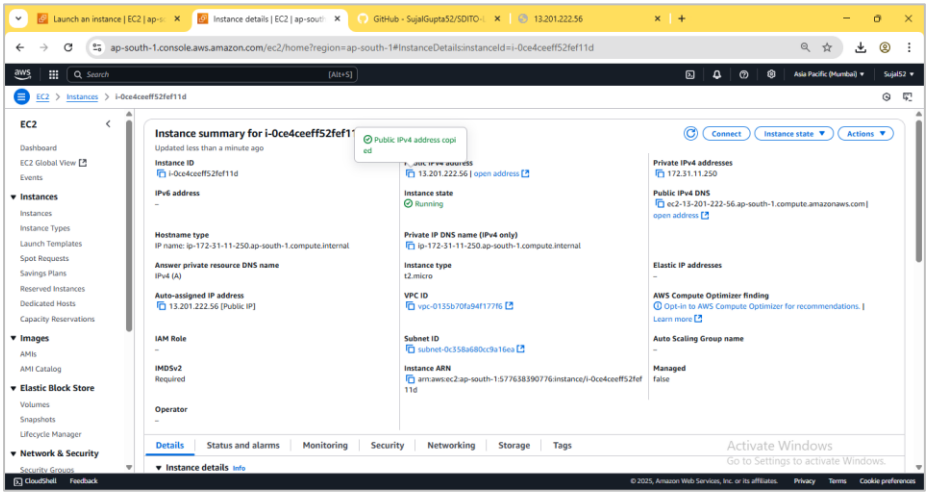
5. Expand ***"Advanced details"***, scroll down to ***"User data"*** and put the following bash script. Then click on ***"Launch Instance"***.



6. Our instance is created successfully

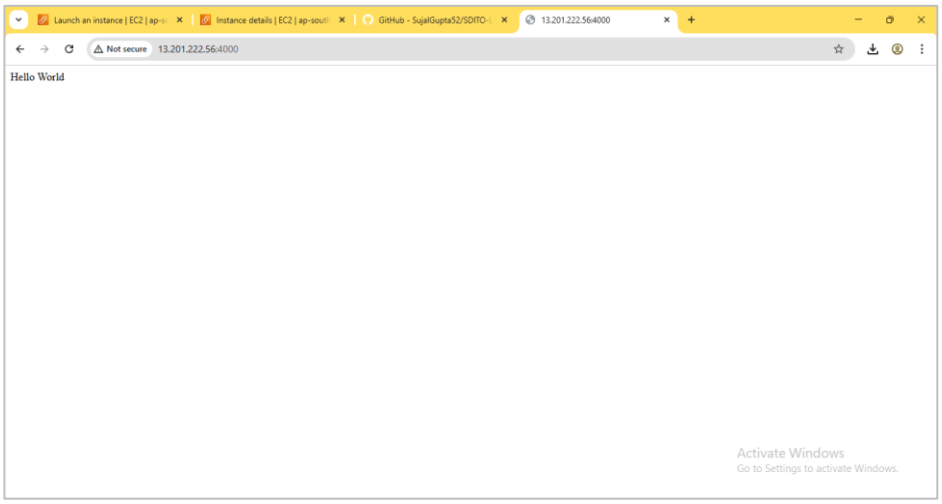


7. Go to ***Instance***, click on ***"Instance ID"*** of our newly created instance. Copy the ***IPV4 address***.



8. Paste the previously copied address into a new browser tab and suffix ":4000" to indicate our port. Our NodeJS server is successfully

deployed.

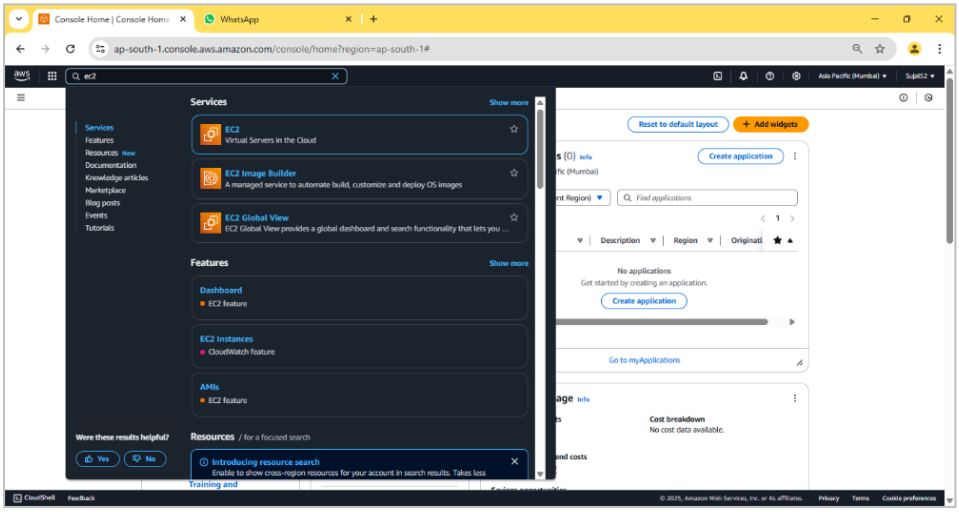


ASSIGNMENT 11:

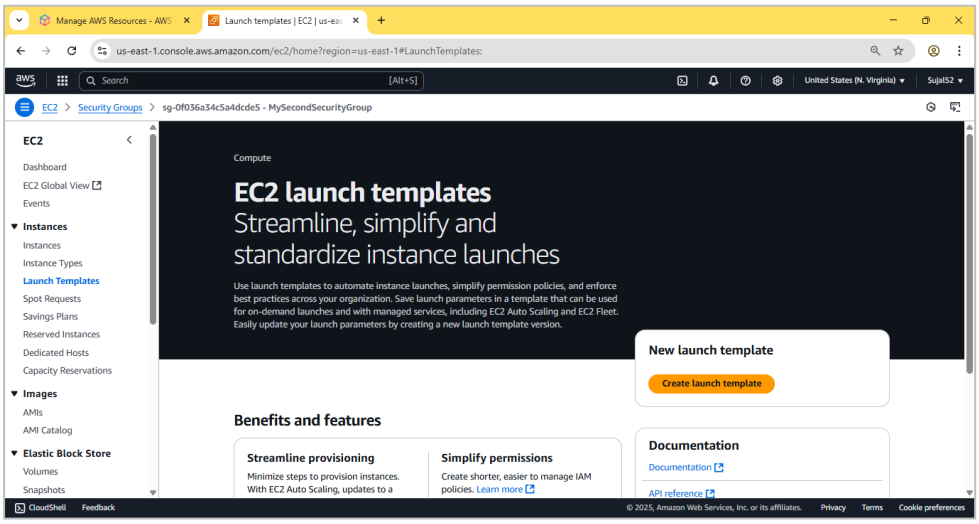
Problem Definition: Build scaling plans in AWS that balance load on different EC2 instances.

Instructions:

1. Access the ***AWS console***, search for ***EC2***, and select the top option from the search results.

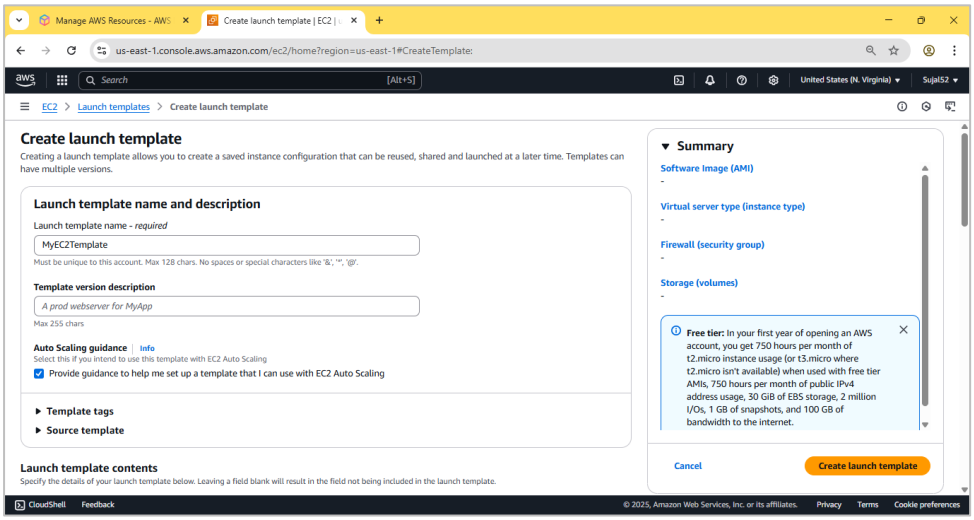


2. Click on ***"Launch Templates"*** from the sidebar then click on ***"Create launch template"***.

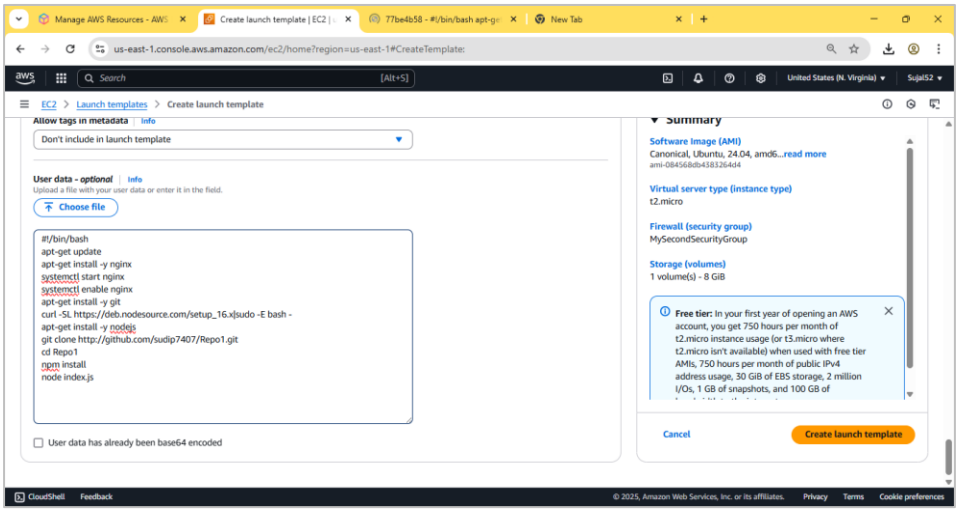


3. Enter name and description. Check the box under ***"Auto Scaling***

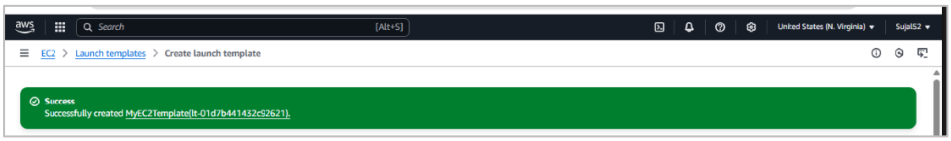
guidance". Select ***Ubuntu*** as OS Image, instance type as ***t2.micro***. Create keypair and select an existing security group where port 4000 is open for incoming traffic.



Under ***"Advance details"***, scroll down to ***"User data"*** and add the following bash script and then click on create instance

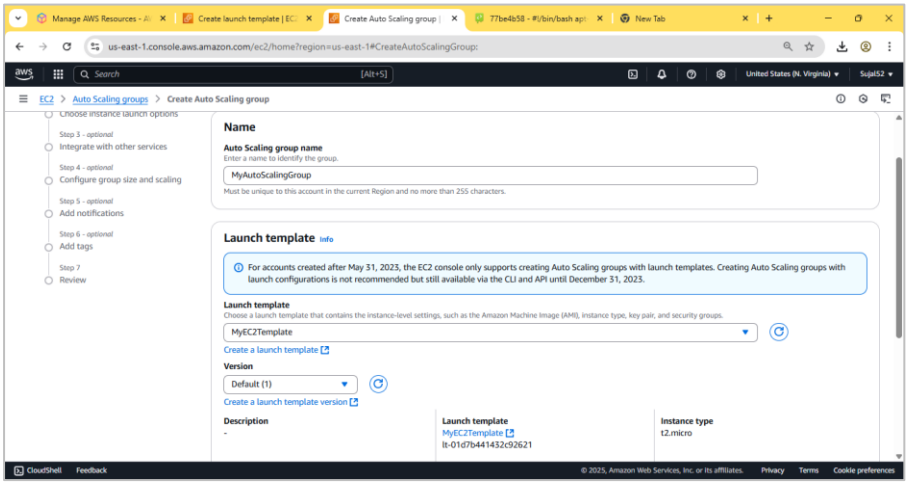


Our launch template is created successfully.

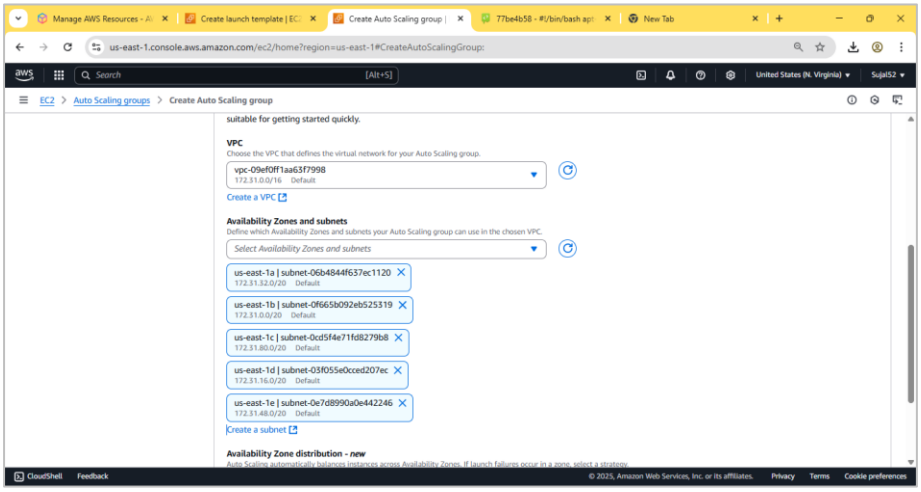


4. From sidebar, scroll down and select ***"Auto Scaling Groups"***. Enter

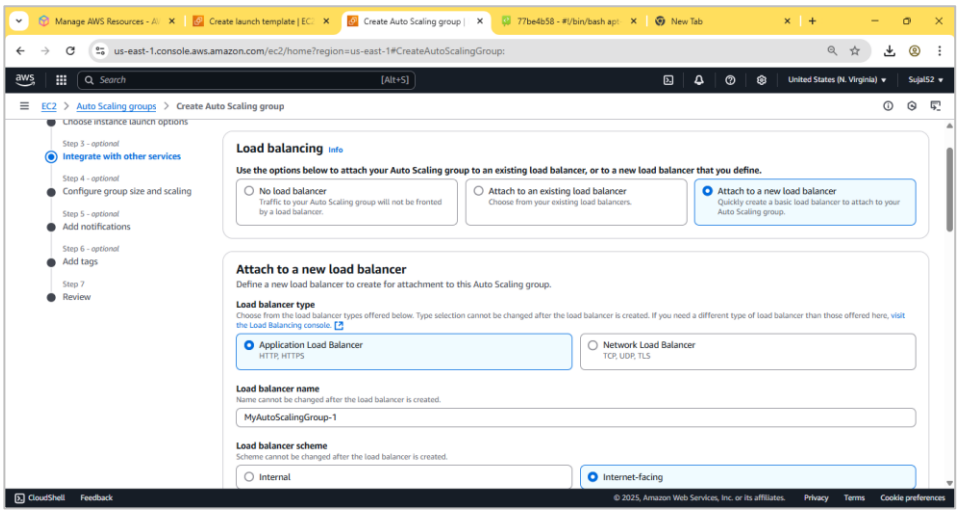
name and select our previously created launch template from the dropdown. Then click on next.



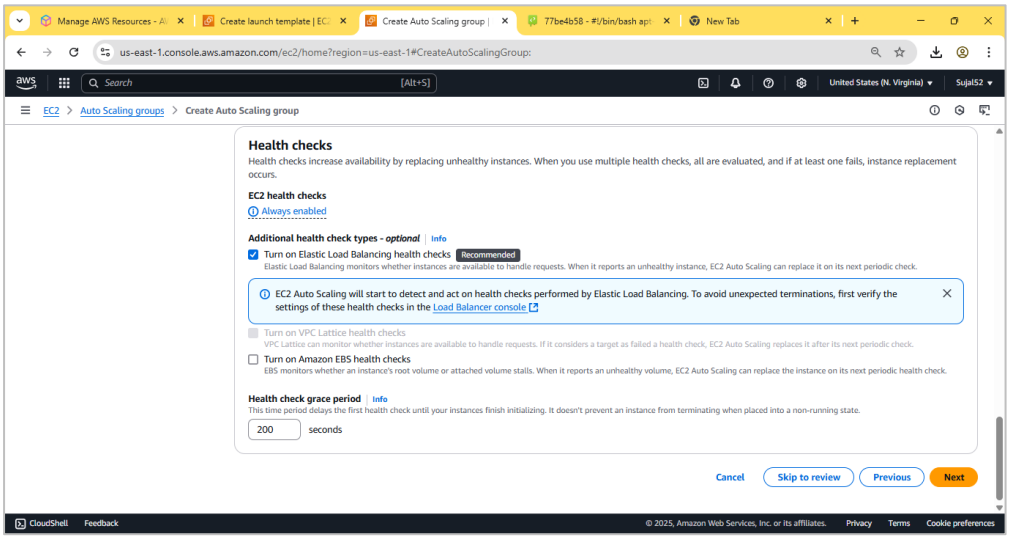
Choose all ***"Availability Zones"*** available then click on next.



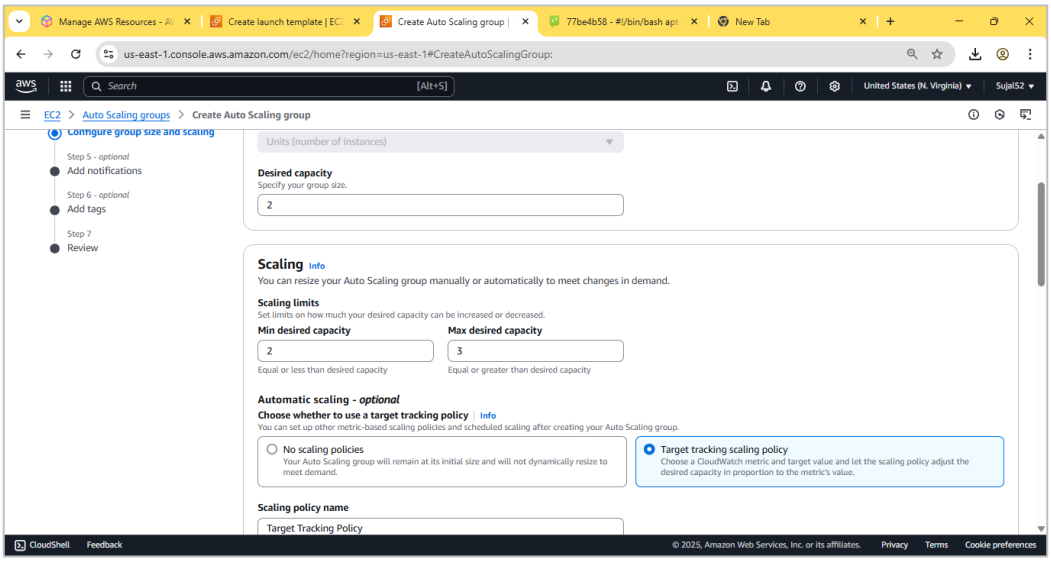
Under ***"Load balancing"***, select ***"Attach a new load balancer"***, under type select ***"Application load balancer"*** and ***"Internet-facing"***



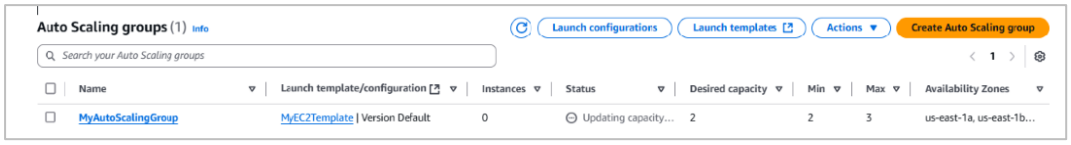
Under ***"Health checks"*** tick the box on ***"Turn on Elastic Load Balancing Health Check"*** and enter the ***"Health check grace period"*** as 240 seconds. Click on next.



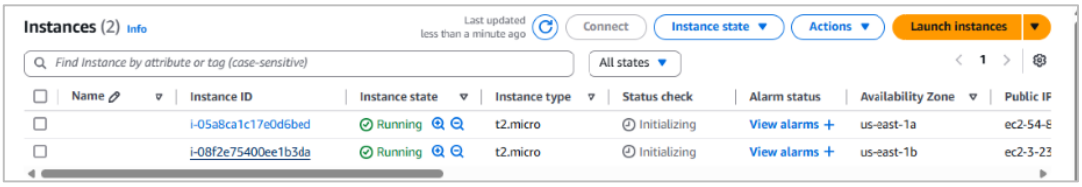
In ***Group size*** set ***Desired capacity*** as 2. In ***Scaling*** set ***Min desired capacity*** as 2 and ***Max desired capacity*** as 3. In ***Automatic scaling*** choose ***target scaling policy*** and set the ***instance warmup*** as 240 second. Keeping everything else as default, click on Next.

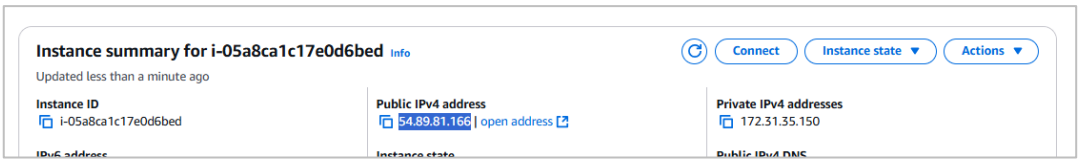


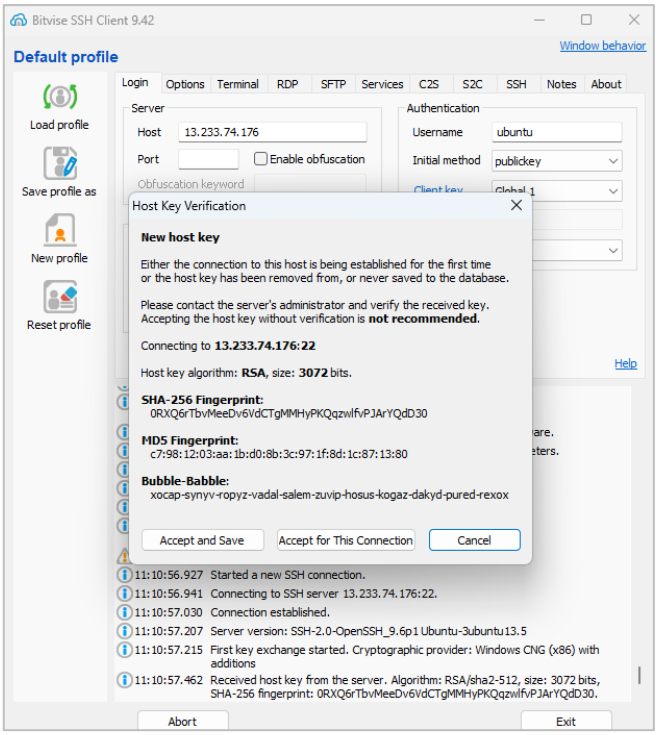
Skip last two steps and click on ***Create Auto Scaling group***. Our auto scaling group is successfully created.



5. Now go to ***Instances*** from the sidebar. We can see two instances are created. Copy the ***IPv4 address*** of any of the two instance and log into ***Bitwise SSH Client*** using key pair.







6. Open terminal console and enter the following commands:

```
vim file.sh
```

Now paste the following code:

```
#!/bin/bash

while(true)

do

    echo “Inside Loop“
```

done

Now make the script executable and run it enter the following commands:

sudo chmod 777 file.sh

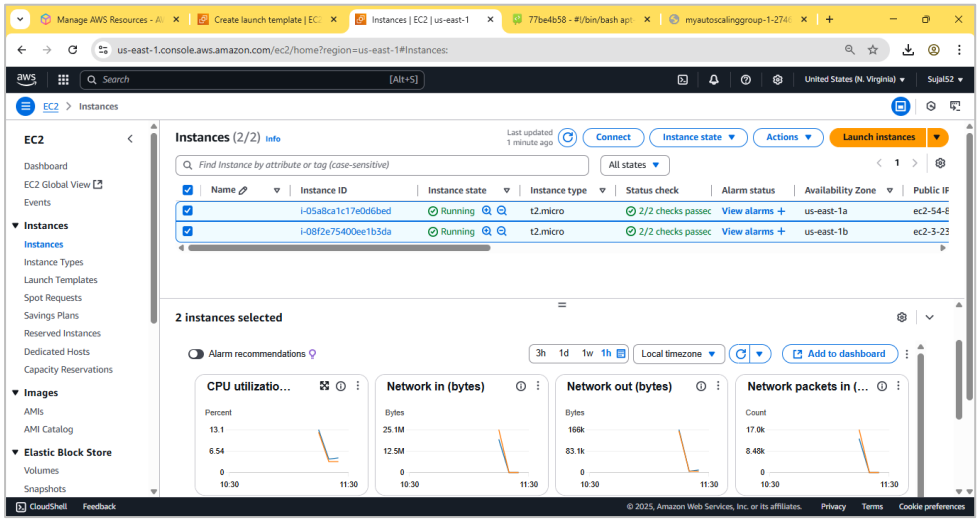
sh file.sh

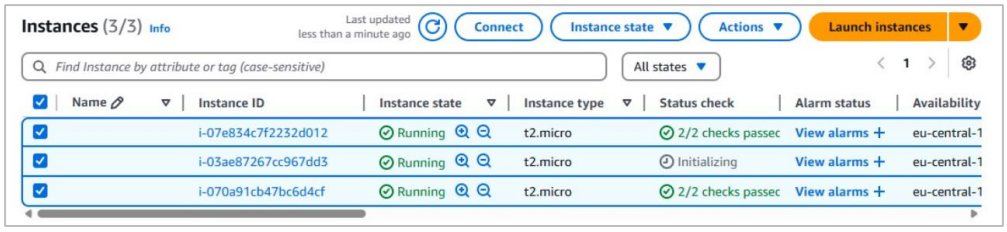
```
ubuntu@ip-172-31-35-150:~$ sudo nano file.sh
ubuntu@ip-172-31-35-150:~$ sudo chmod +x file.sh
ubuntu@ip-172-31-35-150:~$
```

An infinite loop will start to run, stressing the cpu of the instance.

```
Inside Loop
Inside Loop
Inside Loop
Inside Loop
Inside Loop
Inside Loop
Inside Loop
Inside Loop
```

7. Select the two instances to see their utilization graph. After a while, when it surpasses 50% threshold, we can see that auto scaler spins up another instance to balance the load.



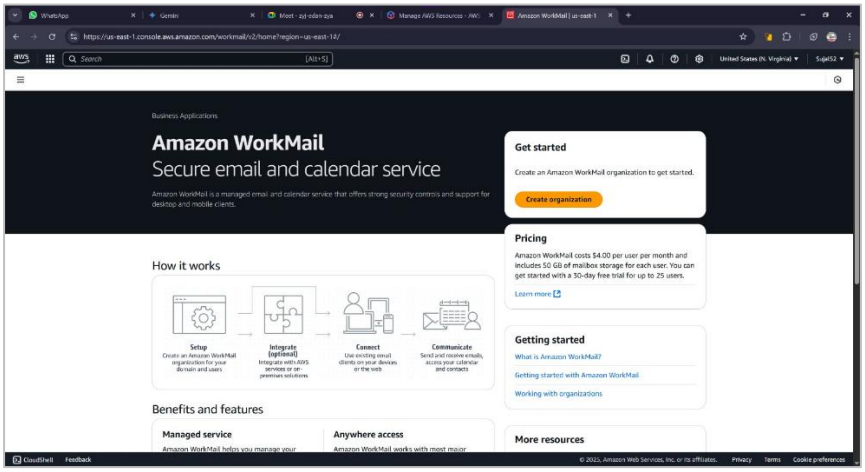


ASSIGNMENT 13:

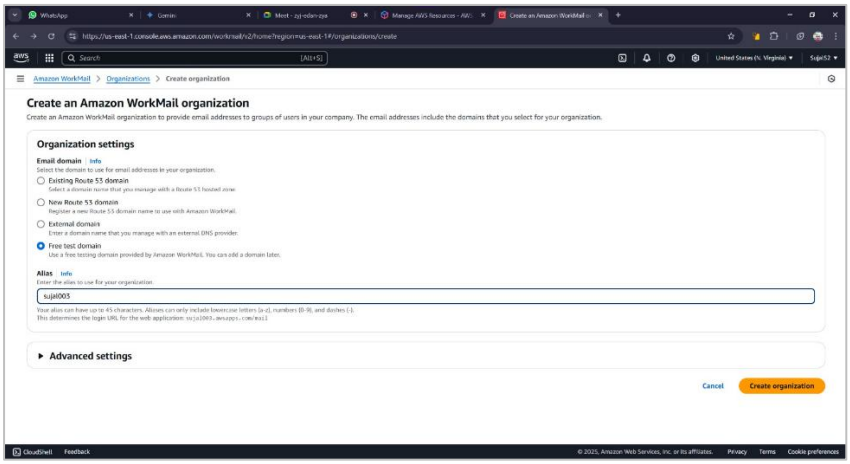
Problem Definition: Create a workmail for your organization.

Instructions:

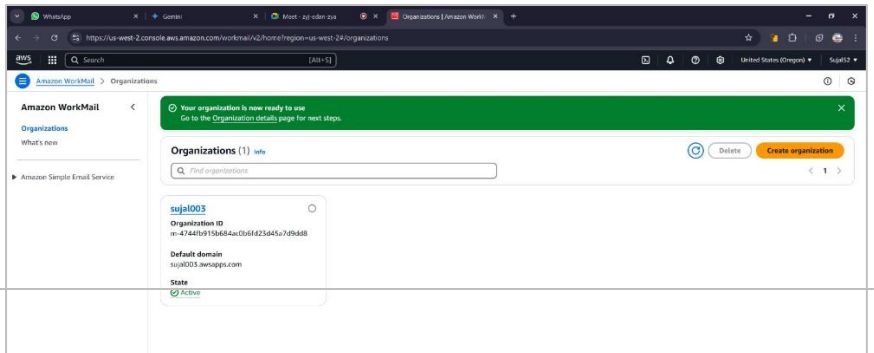
1. Access the ***AWS console***, search for ***Workmail***, and select the top option from the search results. Click on ***Create Organization***.



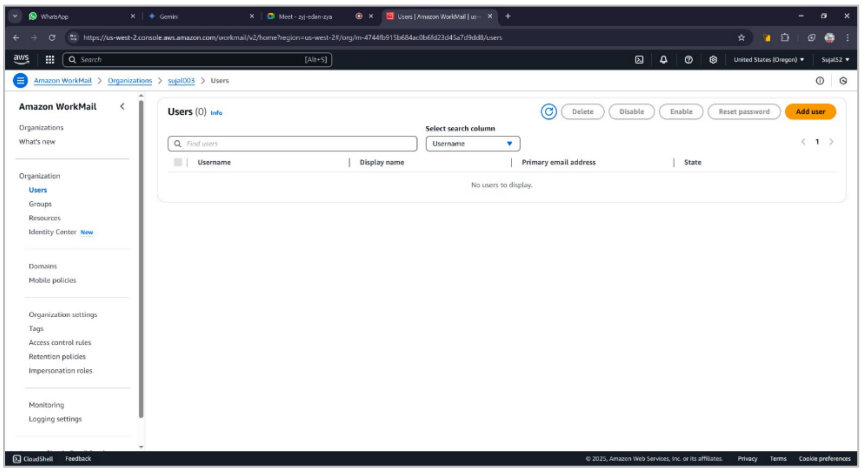
2. Select ***Free Test Domain*** under ***Organization settings***. Enter an alias then click on ***Create Organization***.



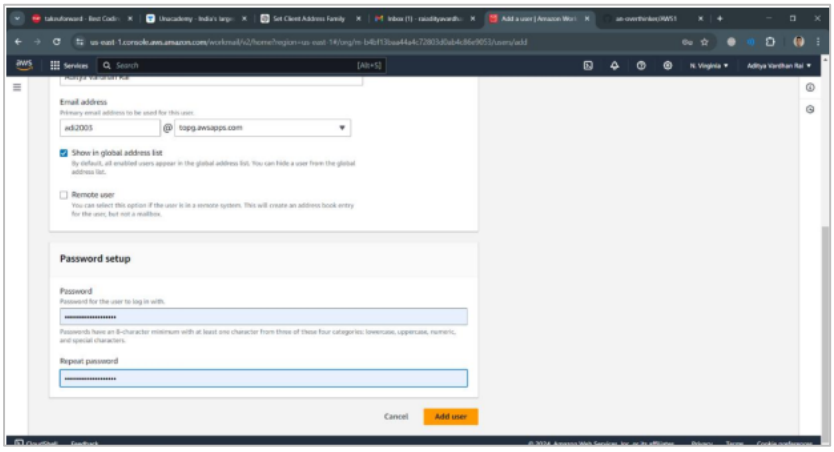
Our organization is created successfully.



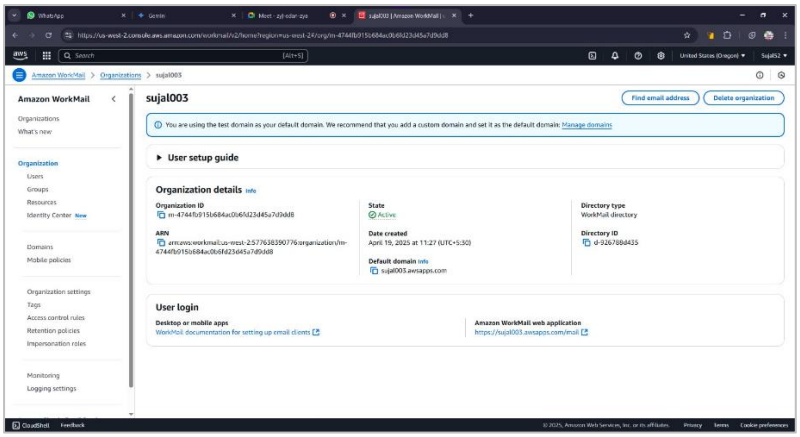
3. Click on Organisation name. Go to ***Users*** and click on ***Add User*** button.



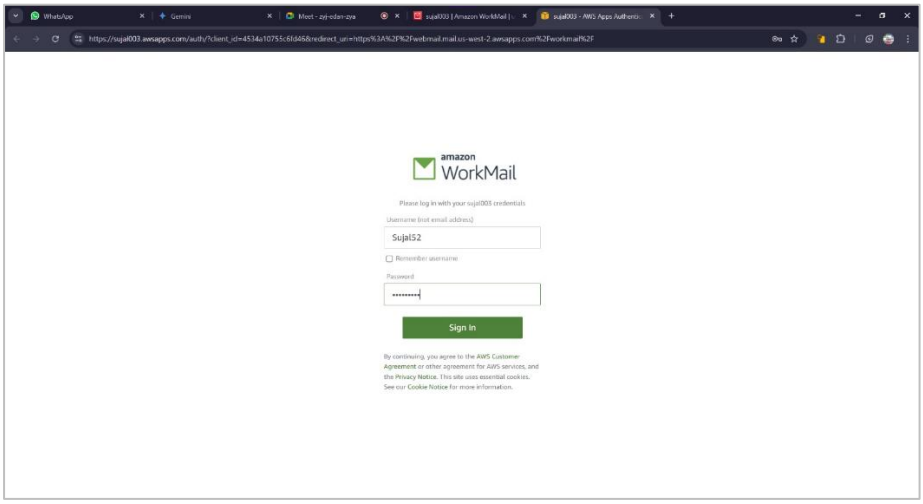
4. Fill username, display name and password. Then click on ***Add User***. A green banner will be displayed indicating our user is created successfully.

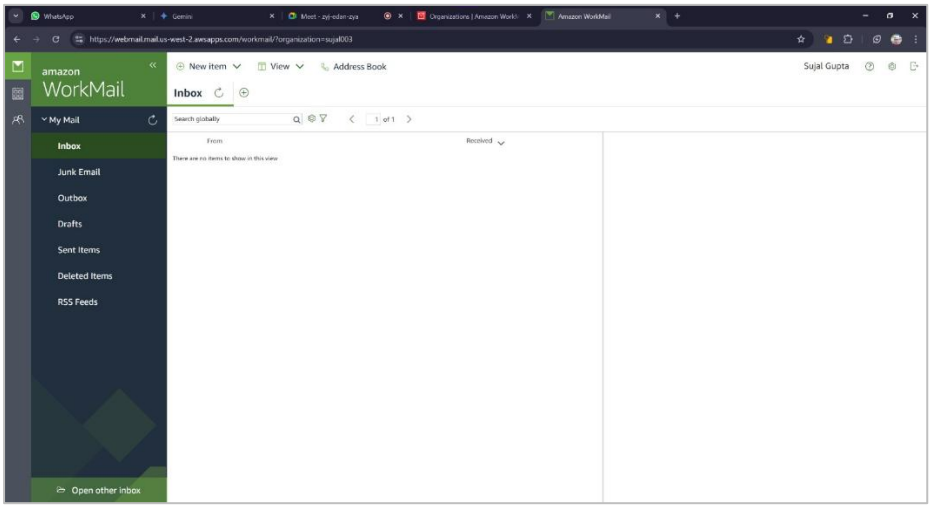


5. Go to Organisation Details page again and click on the ***Amazon WorkMail web application***.



6. Enter our previously created username and password. The inbox should open indicating our workmail is created successfully.



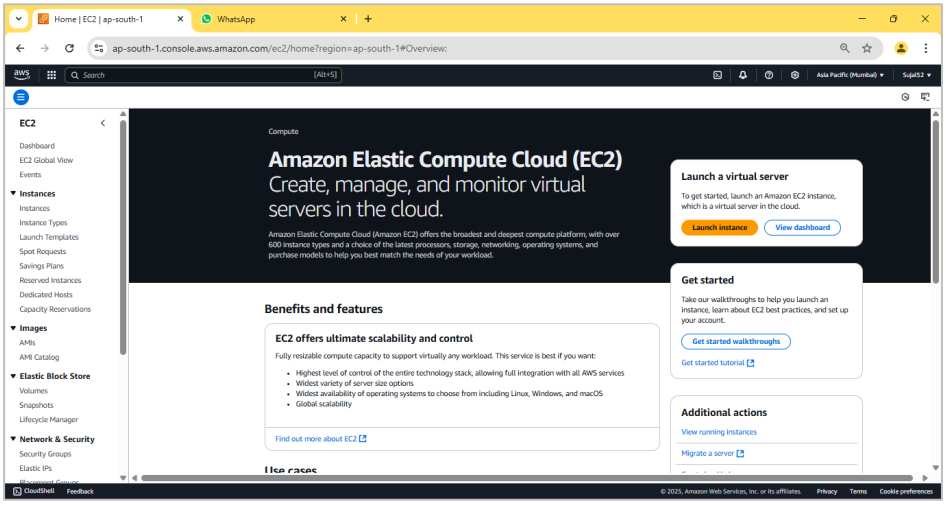


ASSIGNMENT 14:

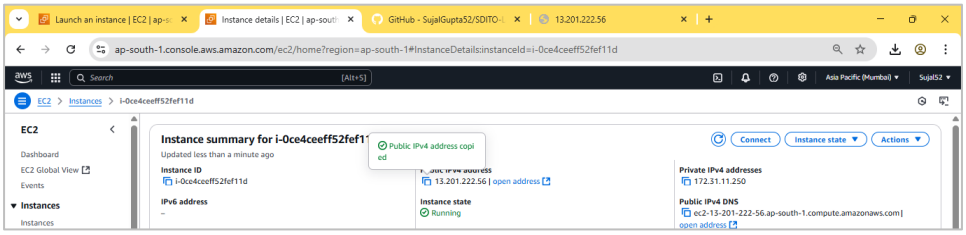
Problem Definition: Create an Elastic IP for an Instance.

Instructions:

1. Access the ***AWS console***, search for ***EC2***, and select the top option from the search results. Then go to ***Instances*** from the sidebar.



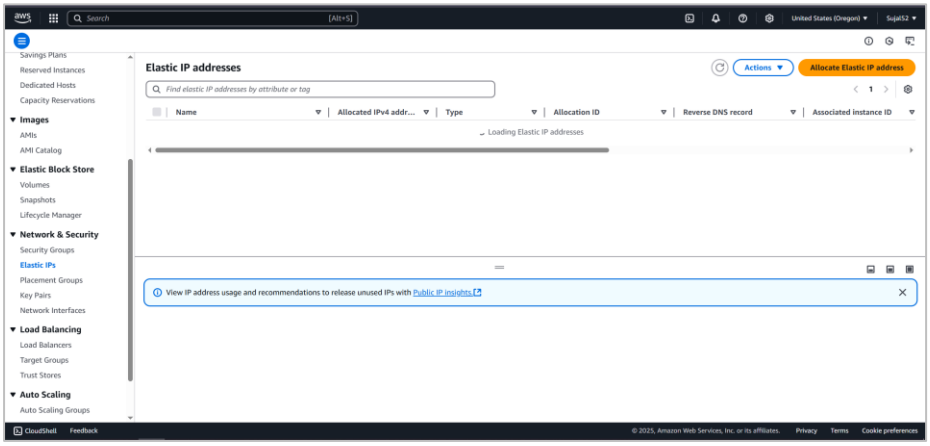
2. Launch a EC2 instance and copy its Public IPv4 address for future comparison.



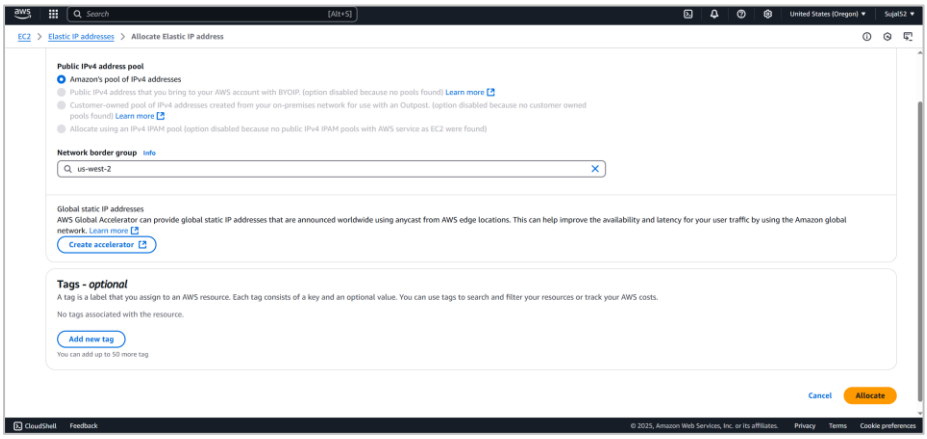
3. Now restart the instance and compare its Public IPv4 address with

the one we previously saved. We can see it changes every time a instance is launched.

4. From the ***EC2 Dashboard***, go to ***Elastic IPs*** and click on ***Allocate Elastic IP***

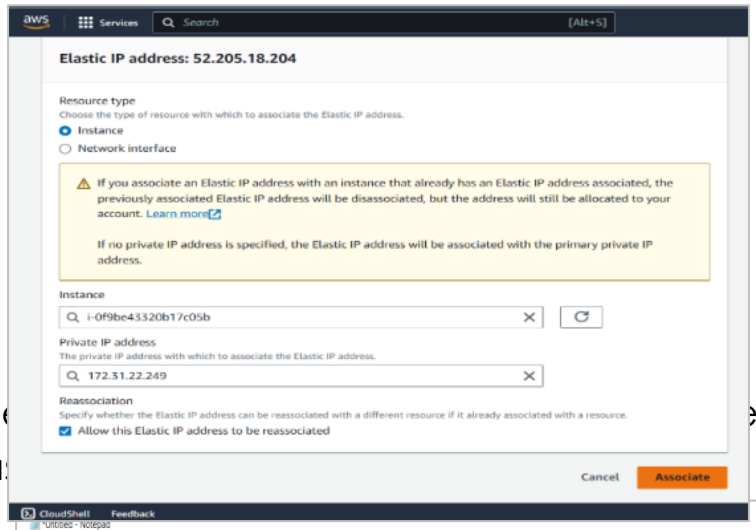


5. Select ***Amazon's pool of IPv4 Addresses*** and click on ***Allocate.*** A green banner will be shown indicating our Elastic IP has been created.

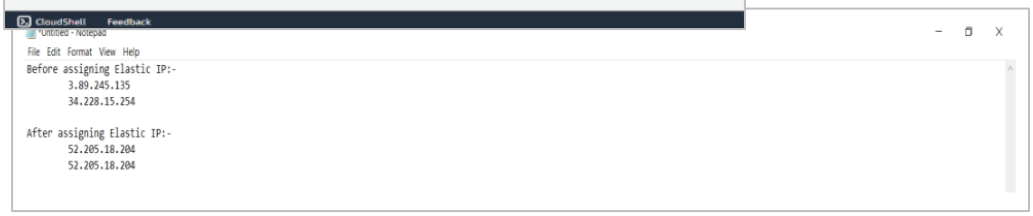


6. Click on ***Associate Elastic IP address***. Select resource type as ***Instance***, select name of our instance, its private IP address and

then click on ***Associate***.



7. R e our Elastic IP is a

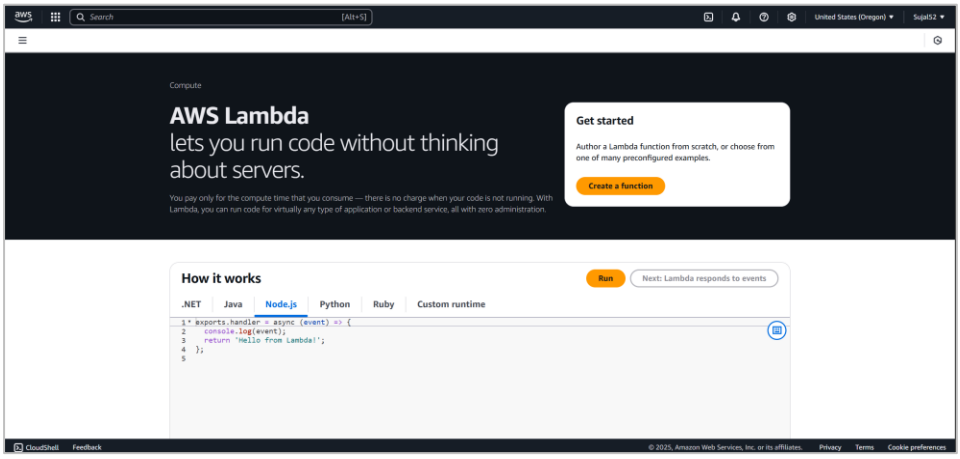


ASSIGNMENT 15:

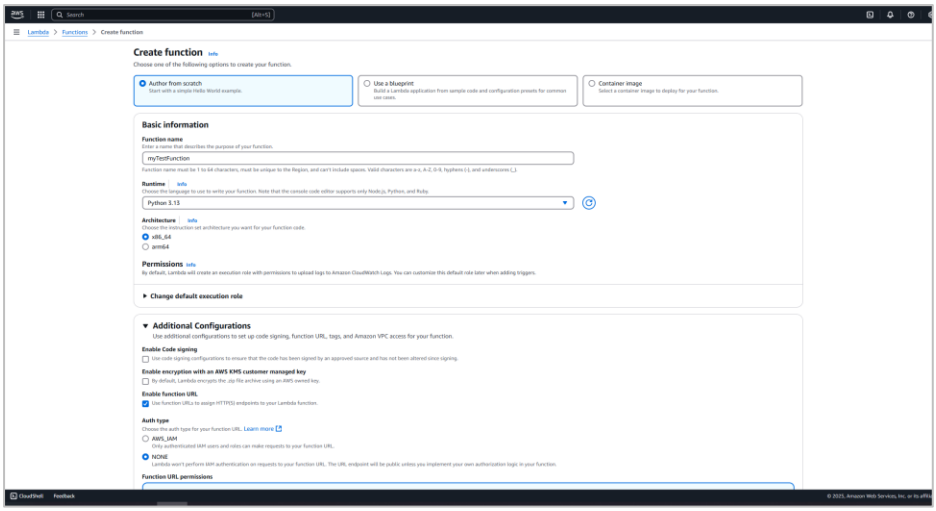
Problem Definition: Create a serverless computing service

Instructions:

1. Access the ***AWS console***, search for ***Lambda***, and select the top option from the search results.



2. Click on ***Create a function***. Enter name and select ***Python*** as ***Runtime***. Under ***Advanced Configuration***, check ***Enable function URL*** and select ***Auth type*** as ***None***. Click on ***Create Function***.



3. Our function is created successfully. Now copy the function URL and open it in a new tab.