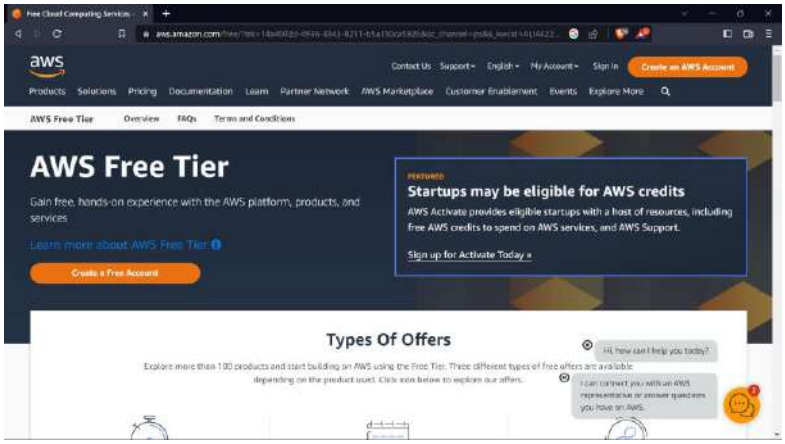
ASSIGNMENT 1

Problem Statement: Create an account in AWS and configure a budget.

Procedure: (a) Create an account in AWS

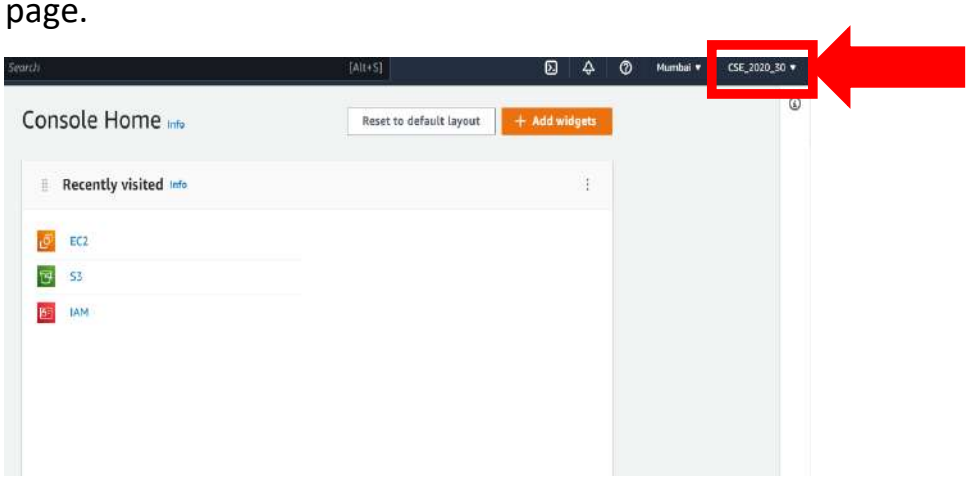


1. Open the Amazon Web Services home page .
2. Choose **Create an AWS account**. Make sure to create the account as a **root user** to unlock full functionality of the newly created AWS account.
3. Enter your account information, and then choose **Continue**. Be sure that you enter your account information correctly, especially your email address.
4. Choose **personal** account.
5. Enter your personal information.
6. Read and accept the **AWS Customer Agreement**.
7. Choose Create Account and Continue.
8. On the Payment Information page, enter the information about your **payment method**, and then choose **Verify and Add**. You can't proceed with the sign-up process until you add a valid payment method.
9. Next, you must **verify** your **phone number**. Choose your country or region code from the list, and enter a phone number where you can be reached in the next few minutes.
10. Enter the code displayed in the CAPTCHA, and then submit.
11. When the automated system contacts you, enter the PIN you receive and then choose Continue.
12. On the **Select a Support Plan** page, **choose the free tier**.
13. Finally, wait for your new account to be **activated**. This usually takes a few minutes but can take up to **24 hours**.
14. When your account is **fully activated**, you receive a **confirmation email message**. Check your email and spam folder for the confirmation message. **After you receive this email message, you have full access to all AWS services**.

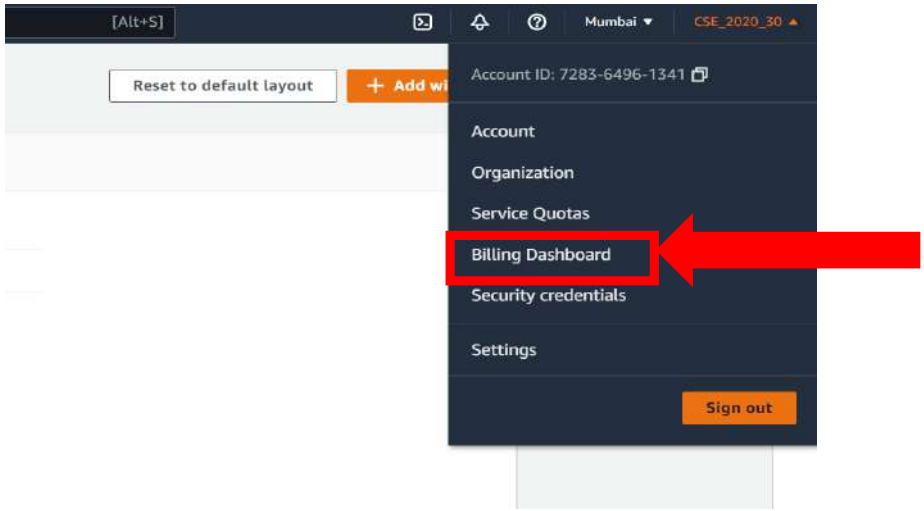
(b)Configure a Budget

1. Sign in to the console using your credentials for your newly created account.

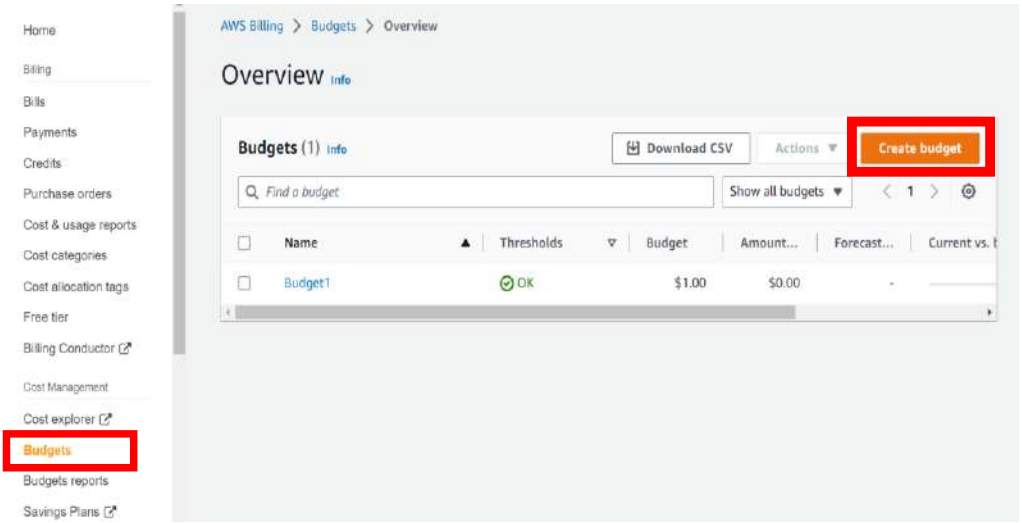
2. Now after successfully signing in we will arrive at the home page. Then click on down arrow beside your account name on the top right corner of the home page.



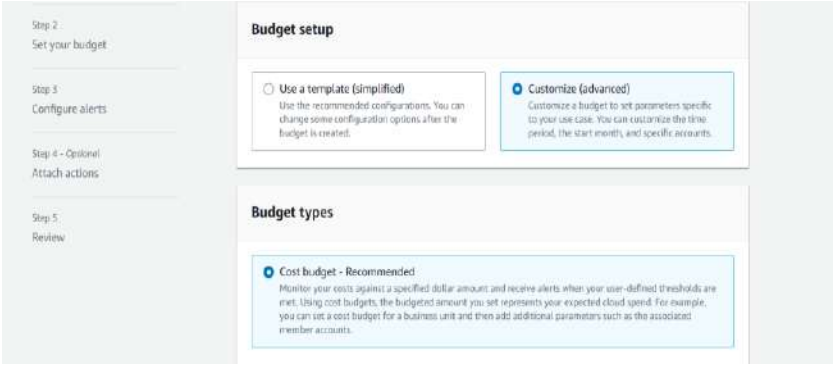
3. A drop down will appear and select **Billing Dashboard** from the list.



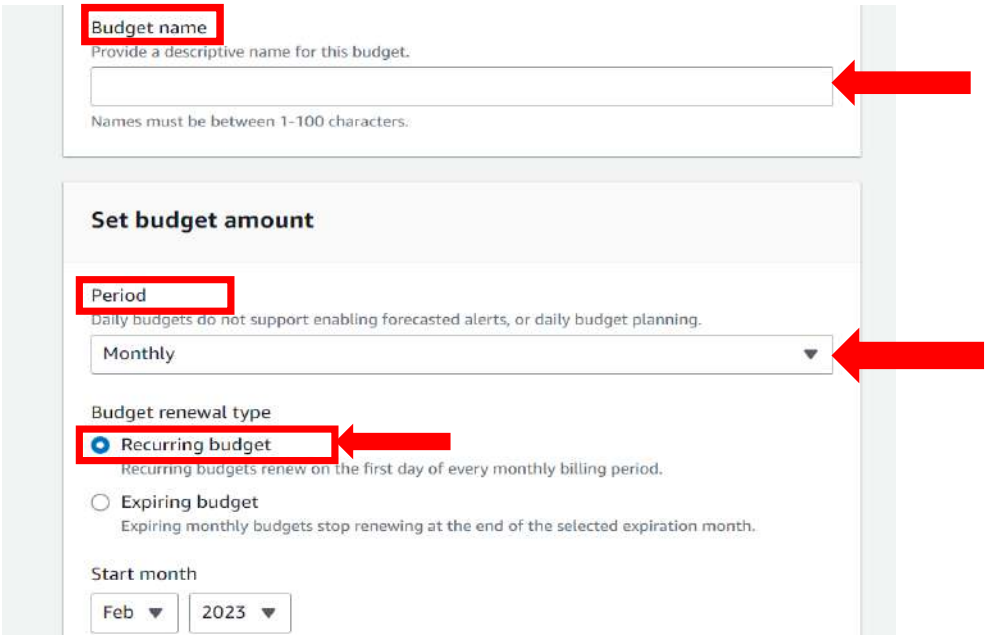
4. After arriving in Billing Dashboard, go to the **Budgets** section on the left side panel under cost management. Click on the **Create Budget** button in the **overview page** to start creating a new budget for your AWS account.



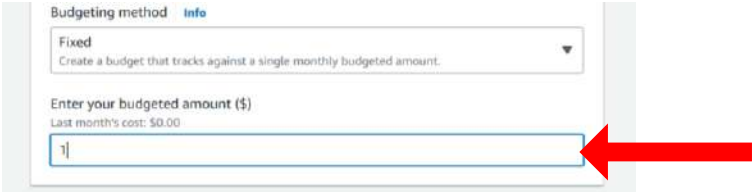
5. Select **Customize** in Budget Setup section and **Cost Budget** in the Budget Types section. Then click on **Next.**



6. Enter **Budget Name.** Then set your **Budget Period**, preferably **Monthly** and set **Renewal Type** to **Recurring.**

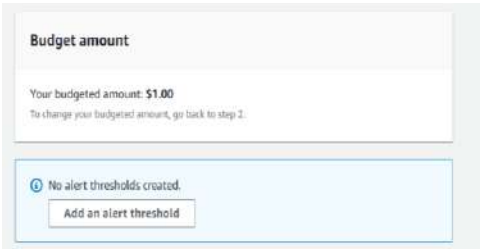


7. Scroll down and Enter your Budgeted amount. For best free usage enter 1 in the box. (All amounts are automatically considered in dollars “$”)

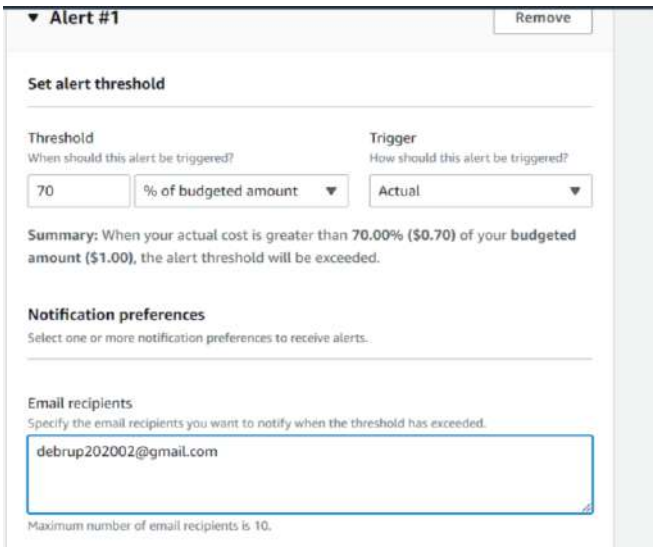


8. No need to change any other options in the page. So, now we click on **Next**.

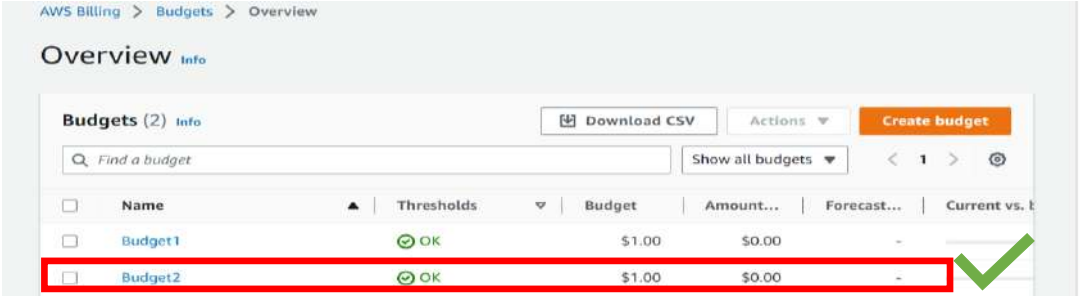
9. Next, we move on to Configure Alerts. This is required to let AWS know when to show us a warning when we reach a certain limit or threshold of our budgeted amount in our stipulated time period. Here, we select **Add an Alert Threshold** button.



10. Now set Alert #1. Give a suitable threshold value in percentage of the actual budgeted amount. If that threshold is reached an alert will be send by AWS. Also, mention the email address(/es) where AWS should send you the alerts.



11. Click on Next.
12. Again, click on Next after reviewing yours alert summary.
13. Now, we are finally at the review or summary page of the whole budget. Click on **Create Budget** button to finally create your Budget!
14. After this step you will be redirected to the **overview page** where all your budgets will be shown. You can see your newly created budget in the table format with various information related to it. Our Budget creation is complete!

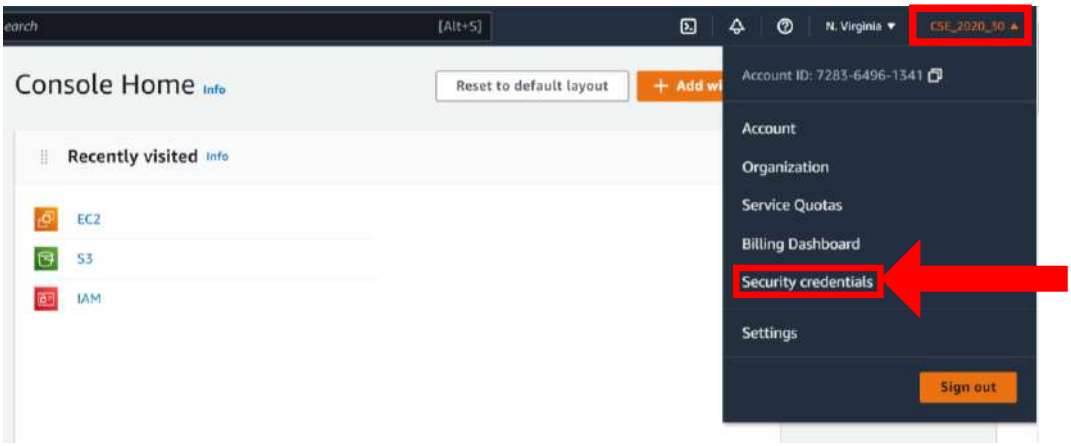


ASSIGNMENT 2

Problem Statement: Create Multi Factor Authentication(MFA) for account authentication.

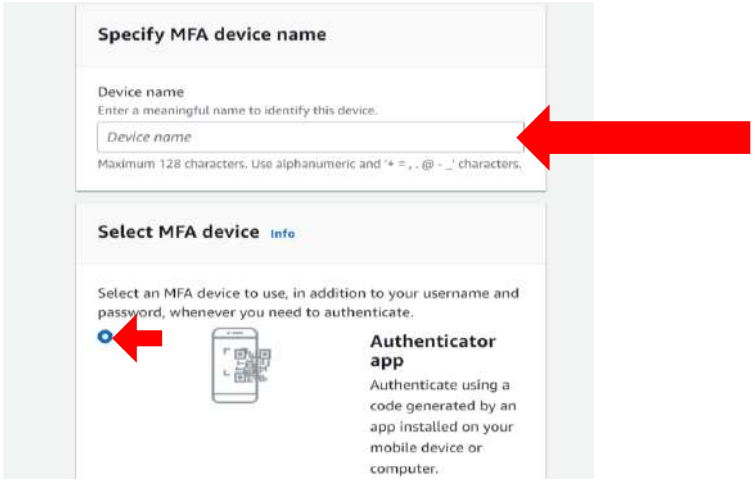
Procedure:

1. Sign-in to your AWS console. Then select the **down arrow** beside your **account name** in the top right side of the page.
2. Now select **Security Credentials** option in the drop-down menu.



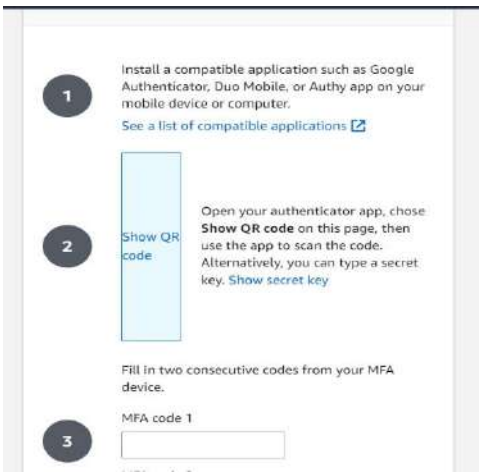
3. Now after arriving in My Security Credentials page, now scroll down to **Multi-Factor Authentication (MFA)** section. Click on the **Assign MFA device** button.

4. Next Assign a **unique device name** (very important and needs to be unique). This name is actually the name of the account that shows up in your authenticator app with the security codes and hence is essential for identification later. Select **Authenticator app** in **Select MFA device** section.

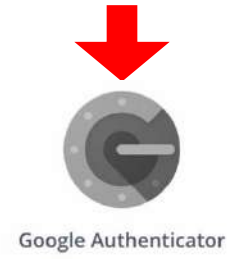


5. Click on **Next.**
6. After that you have to make sure you download an authenticator app from app store in your android/iOS device. Google/Microsoft authenticator is preferred. After installation, click on the **Show QR code box** here in the

website. Scan the QR code with your authenticator app. Your device name given will show up in your authenticator main page.

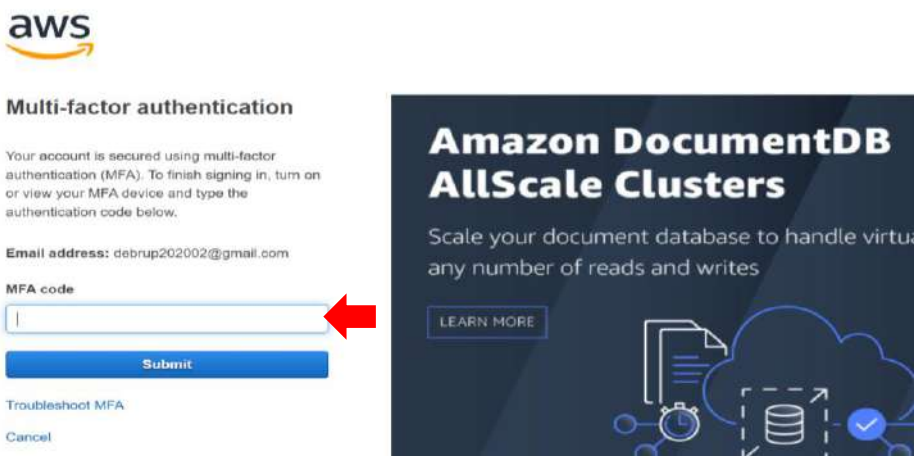


Download some authenticator app from an App Store in your mobile phone (Android/iOS). Then continue with steps 2 and 3 of the section as seen in the snapshot.



7. We will see a certain unique combination appearing against our given device name for our AWS account and stays only for 30-60 seconds. Enter 2 consecutive codes appearing in the given box in the website to authenticate your MFA.
8. After successfully verifying your MFA, click on the Add MFA button.
9. You will be redirected to the security credentials page and see your newly added MFA method with your given device name for your account.

10. Hence, we have successfully added an MFA device.
11. Now sign-out and try to re-login to the console.
12. Now after providing user email and password, from now on you have to enter the MFA code which is given by the authenticator app in your phone. Be mindful that the code changes every 30-60 seconds and we have to enter the current or existing one which has not expired (or in this case stopped showing).

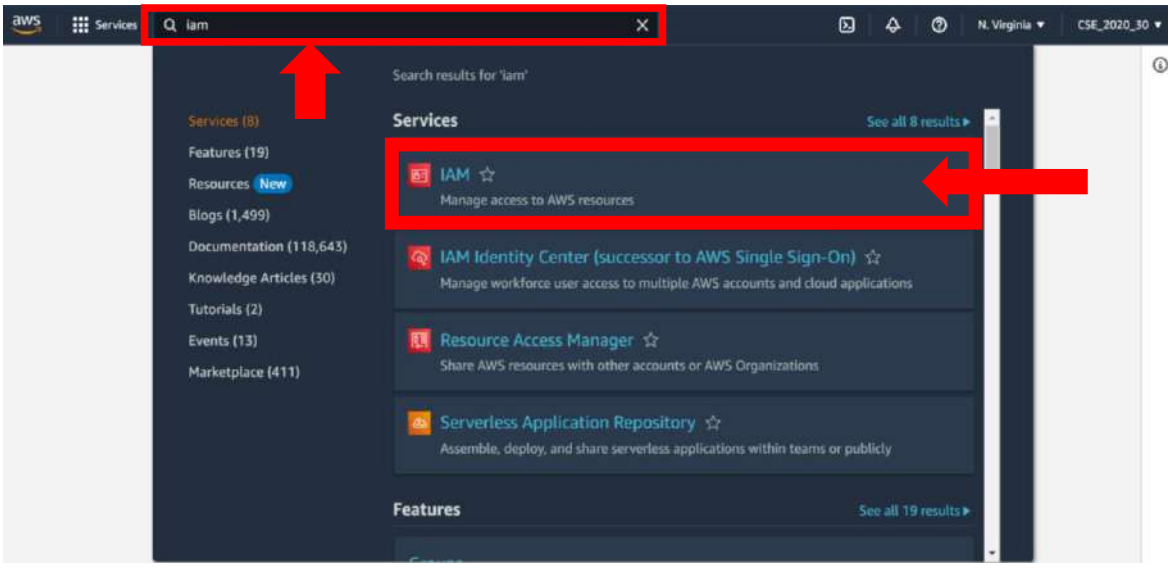


ASSIGNMENT 3

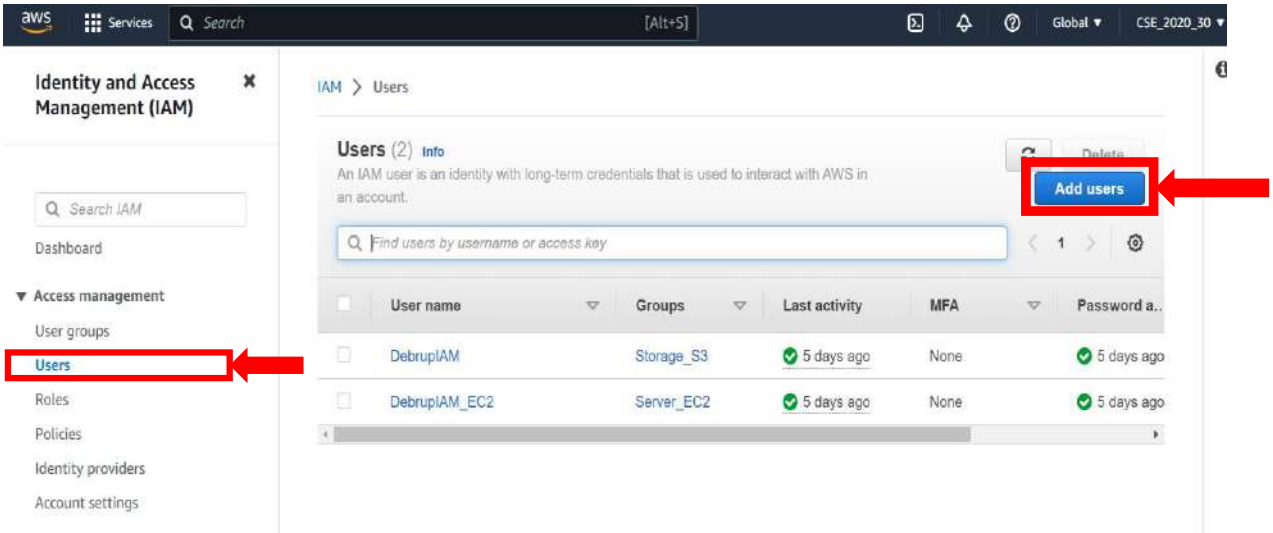
Problem Statement: Create IAM resource giving full access of S3(storage).

Procedure:

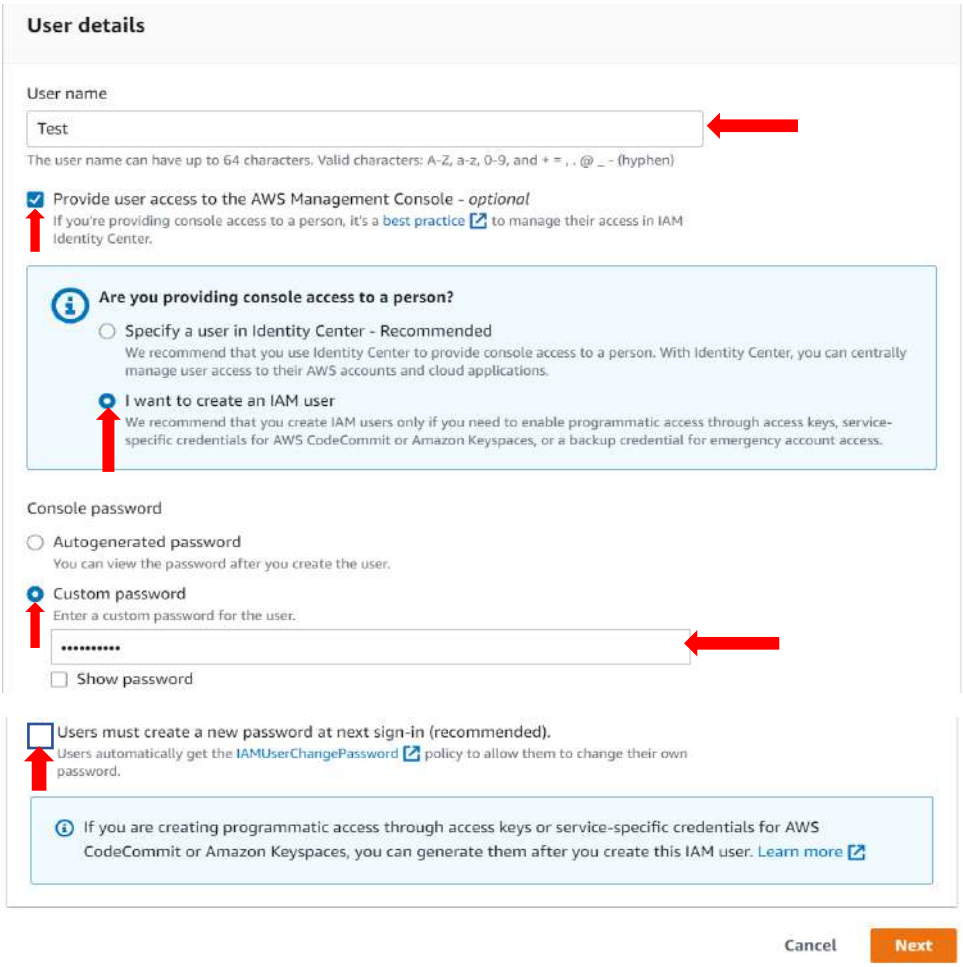
1. Sign in to your console (as root user).
2. On the top side of the page go to the **Search bar** and type “IAM”.
3. Click on the first result showing “IAM”.



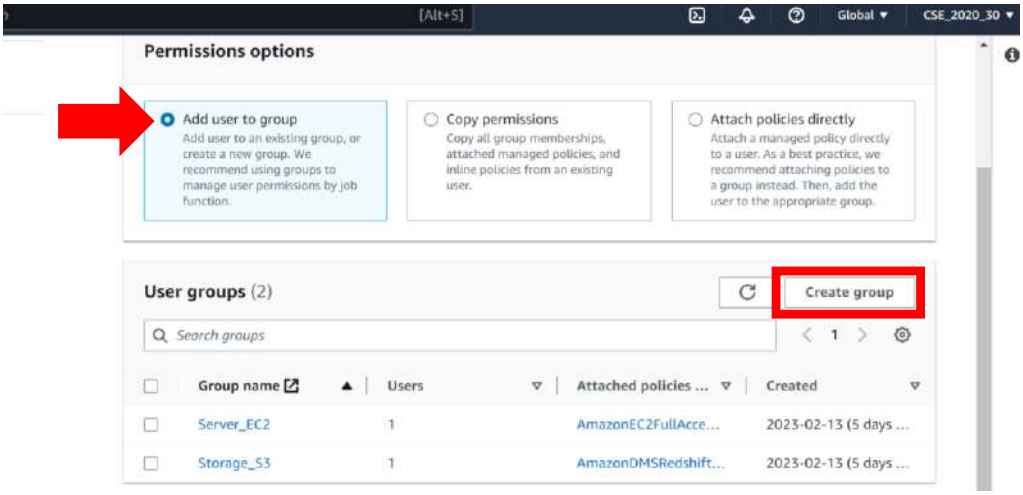
4. We are then redirected to the Identity and Access Management (IAM) dashboard. We then have to select the **user** option in the left side panel under **Access Management.**
5. Next click on **Add Users** button in the **Users** page.



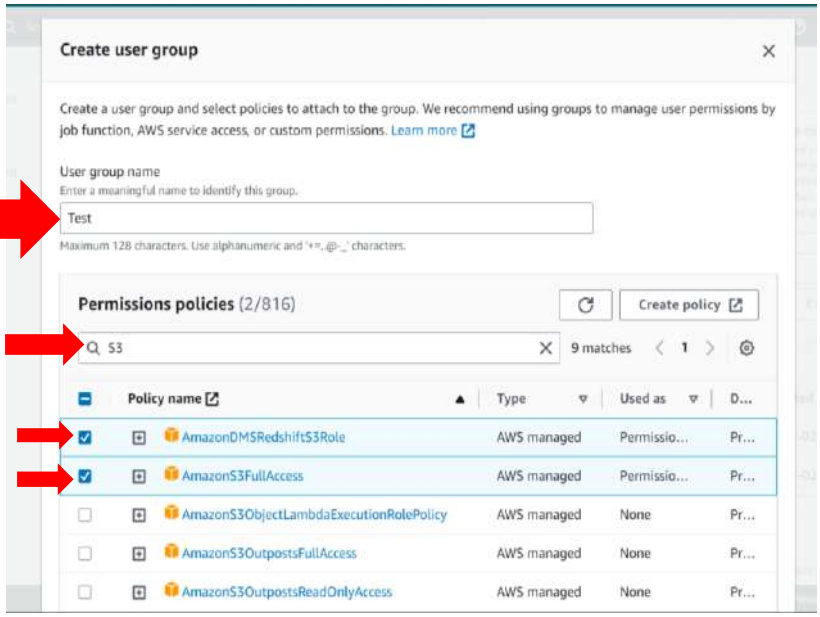
6. After that you have to create a user and specify the details.
    a. Specify the name of the user
    b. **Check** the “Provide user access to the AWS Management Console” box
    c. **Select** the option “I want to create an IAM user”.
    d. Select custom password and enter it.
    e. **Uncheck** the “Users must create a new password at next sign-in” box.
    f. Then click on next



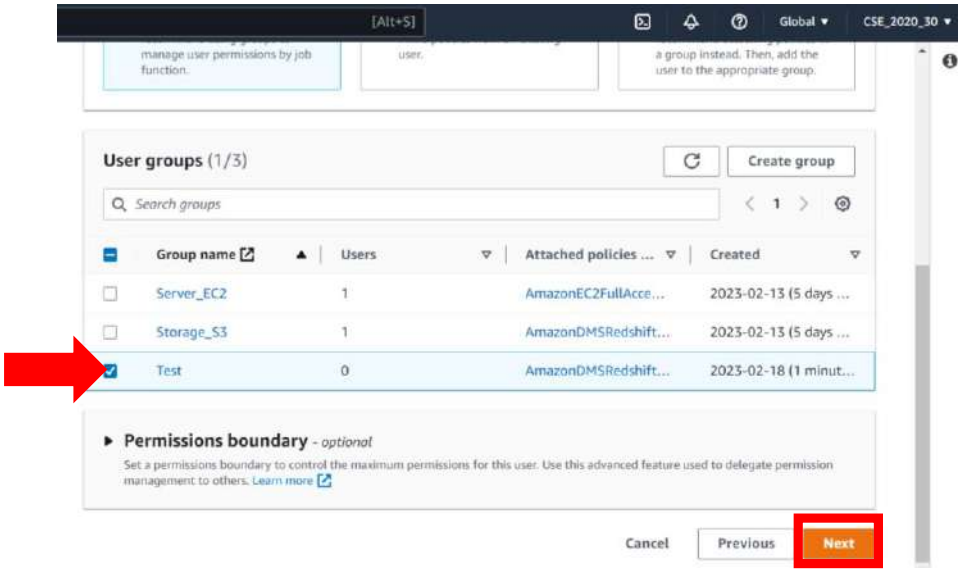
7. Now under **Permissions Options**, select **Add user to Group** option**.**
8. Under **User Groups** click on **Create Group** button.



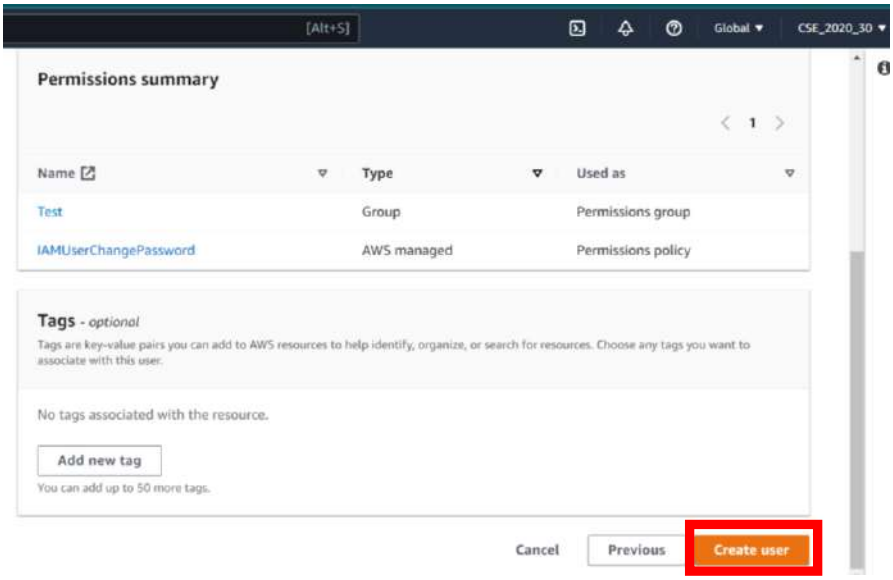
9. A pop-up will appear where you have to specify the new group name and edit the policies/permissions associated with it
    a. Enter the **User Group Name**
    b. Next in the find policies search bar type **S3** as we have to give permission only for S3.
    c. Select the **first two** options
    d. Then click on **Create User Group**



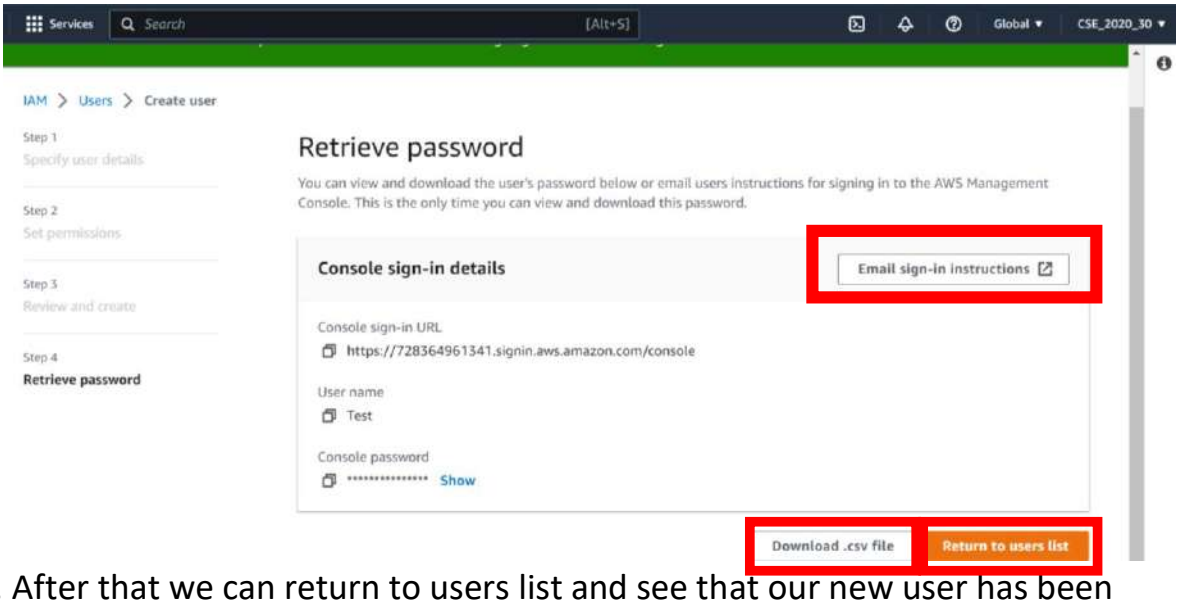
10. Now the pop-up closes and under the **User Groups** section our newly created group is visible in a table format. Select the group.
11. Then click on **Next.**



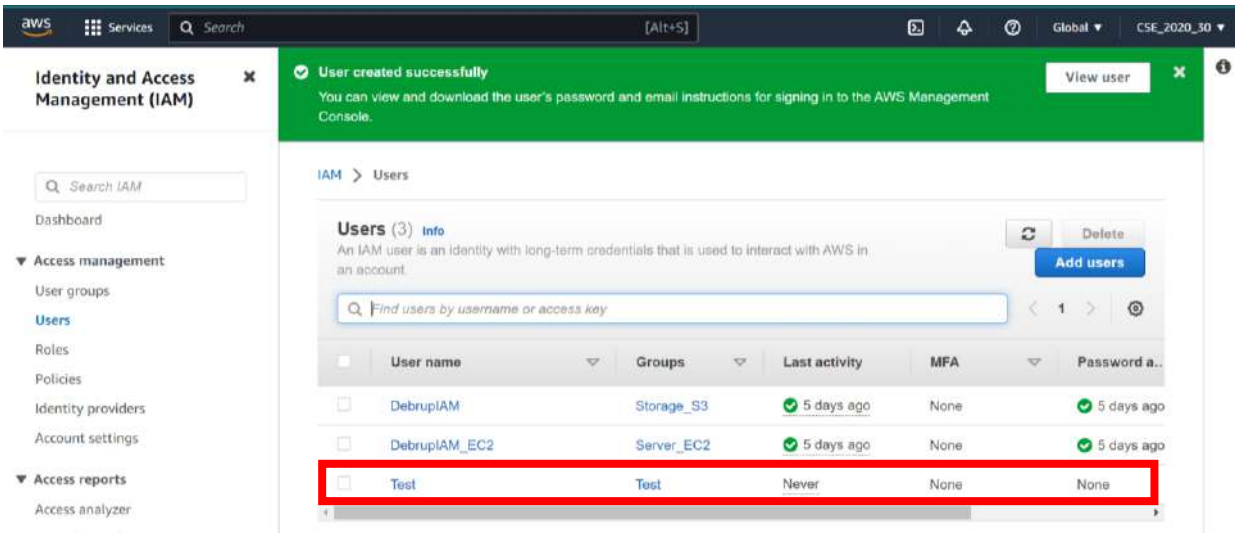
12. We arrive at the **Review and Create** page. After reviewing click on the **Create User** button.



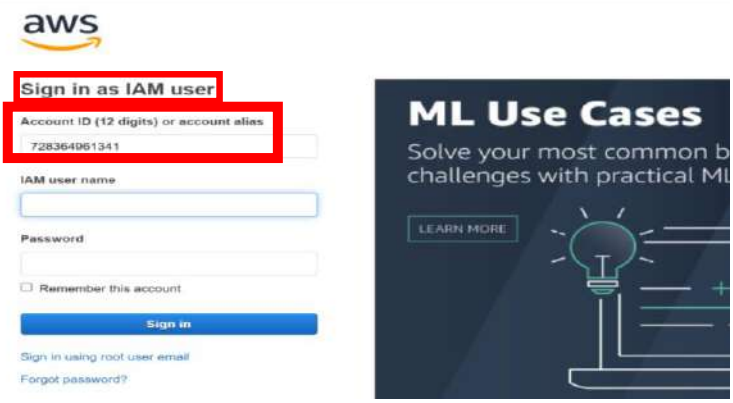
13. Next, we arrive at the **Retrieve Password** page where we can download a **.csv file** or **email the sign-in details of the newly created IAM user**.



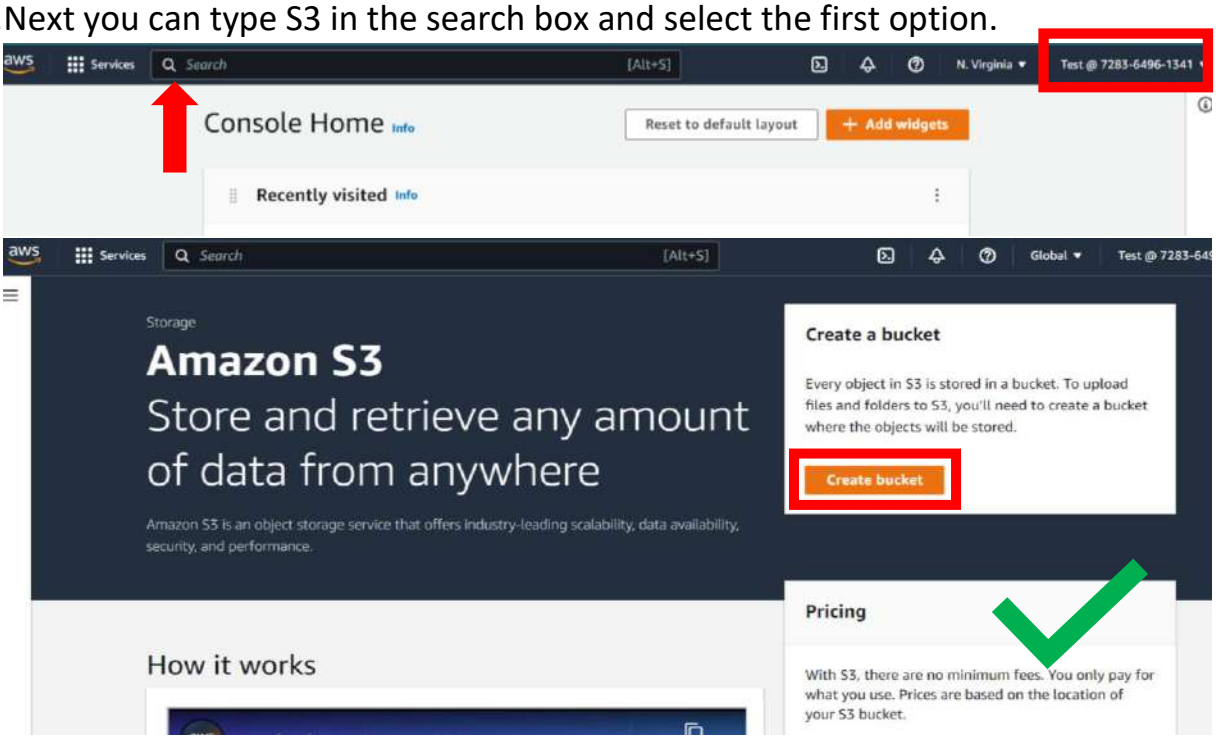
14. After that we can return to users list and see that our new user has been added to the users' table.



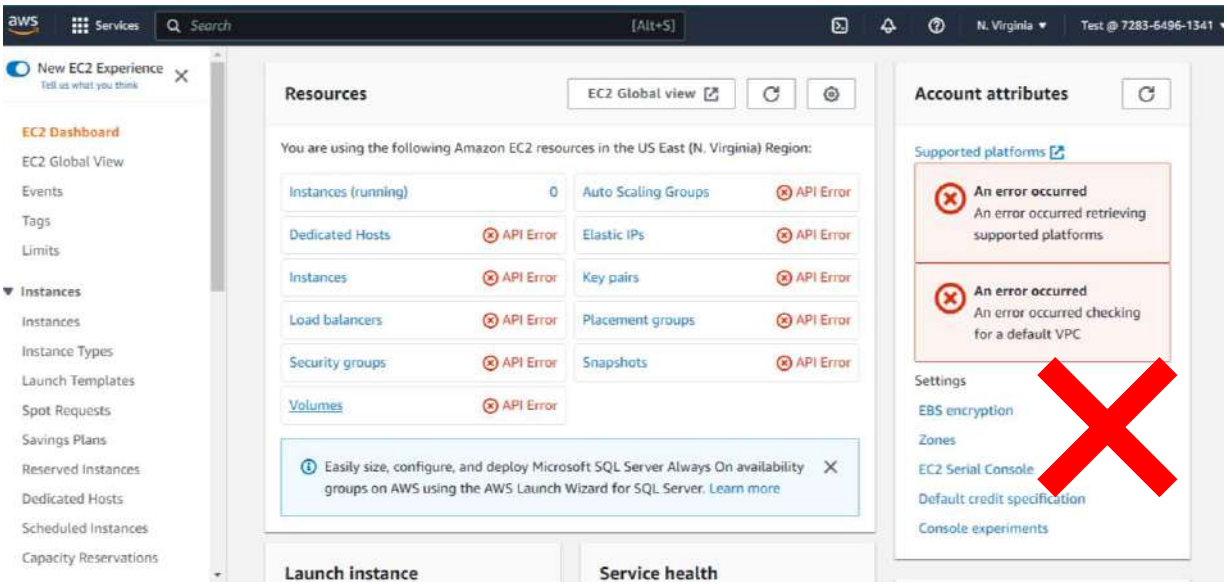
15. Now we logout of our console.
16. Next, we again try to login to the console. But now we select **IAM user login.**
17. Here we have to enter **Account ID** of the root user. We can get that in the drop-down menu after logging in our root user account.
    Alternatively, we can use **the link** in our **downloaded .csv file** or our **email** which if used in our **browser** will redirect use to the login page with the Account ID already entered!



18. Enter the credentials.
19. Note the username in the top right corner. Also, you cannot access your account page as it is controlled only by your root user.
20. Next you can type S3 in the search box and select the first option.



21. Here we get to Create Bucket. Hence we have full access of S3.
22. Now to check our limits let us search EC2 in search bar. Select the first choice.



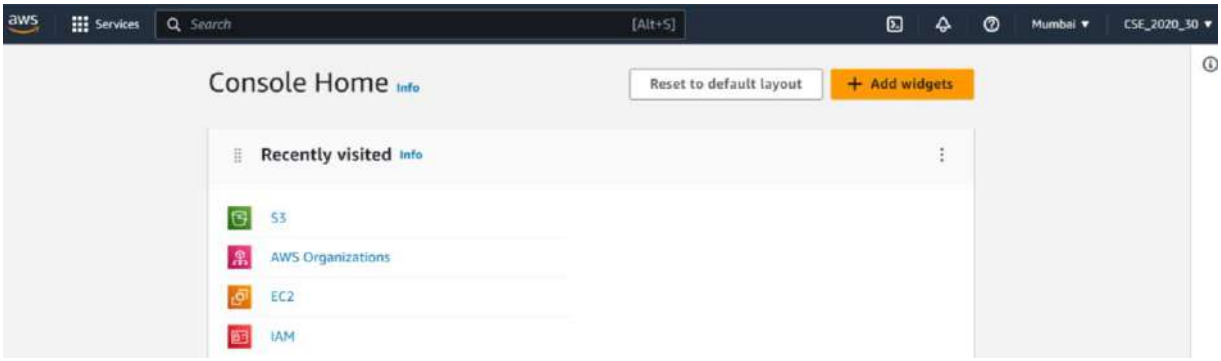
23. Here, we encounter API error. This is proof that we do not have access to EC2. Hence, we have successfully restricted access to our IAM user.
24. Thus, we have successfully created an IAM user and given it only S3 access.
25. Now, we can logout.

ASSIGNMENT 4

Problem Statement: Create a private bucket in AWS. Upload a file and check that through pre-signed URL whether you can access the file or not.

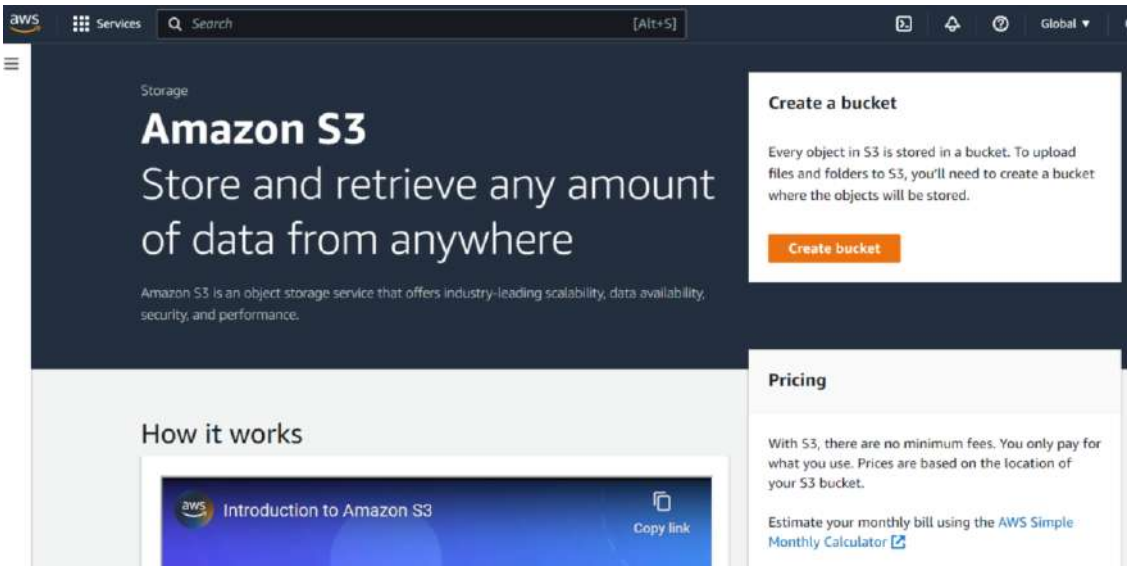
Procedure:

1. Sign in to your **AWS account** as root user.

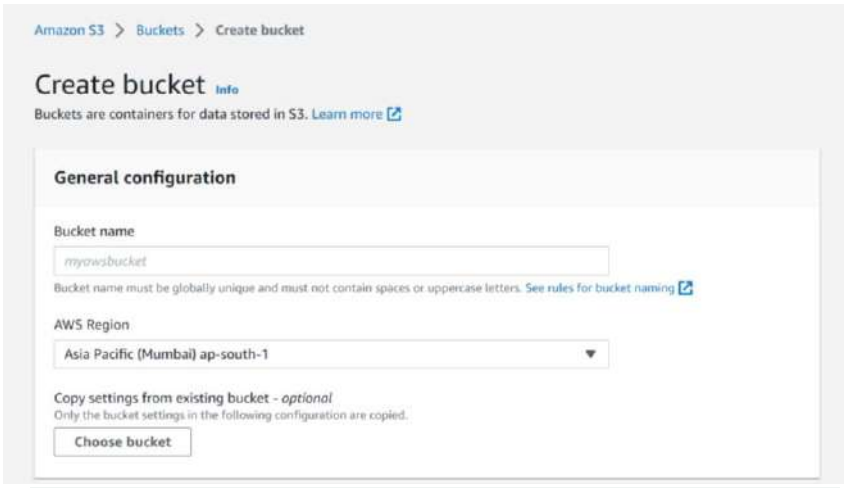


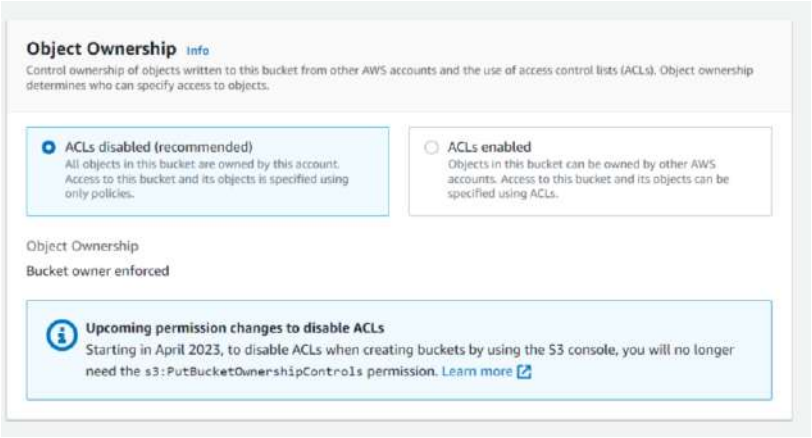
2. Now in the **homepage** search for **S3** in the **search box** and then select the first option displayed.

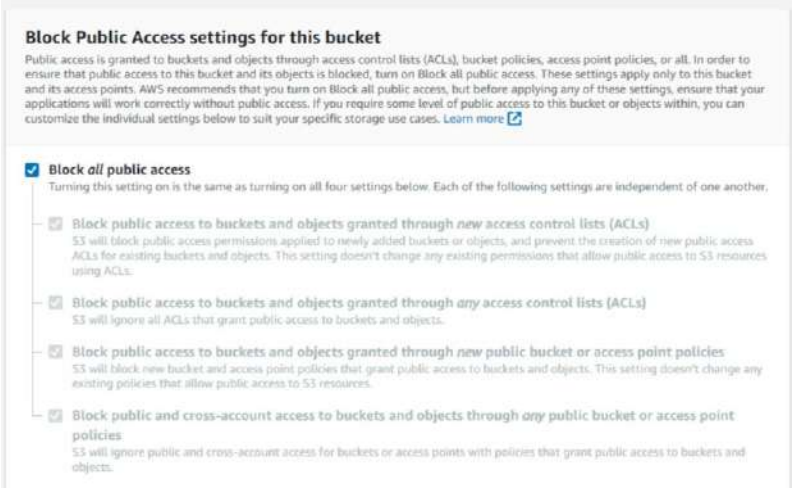
3. After clicking on it, you will be redirected to the **Amazon S3** homepage. There we have to click on the **create bucket** button on the right hand side.

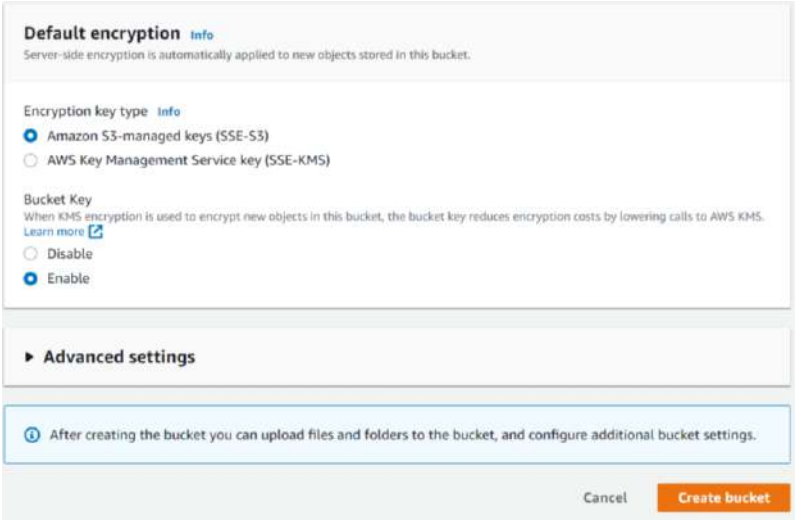


4. Next you will go to the **Create bucket screen** where you have to configure your bucket before creating it.
    a. Choose a globally unique name for your bucket. It should NOT contain any spaces or any uppercase letters.
    b. Select the **AWS Region** as **Asia Pacific (Mumbai) ap-south-1**. **Remember** you can avail other options but each server region has **different pricing** associated with it. Since, we are **living in India**, we are choosing the one **closest to us** to remain fairly priced.
    c. Next we go to Object Ownership section where we keep ACLs disabled option checked (as it is).
    d. Next, we keep all public access blocked (as it is).
    e. Everything else remains unchanged.
    f. Now click on the Create bucket button.

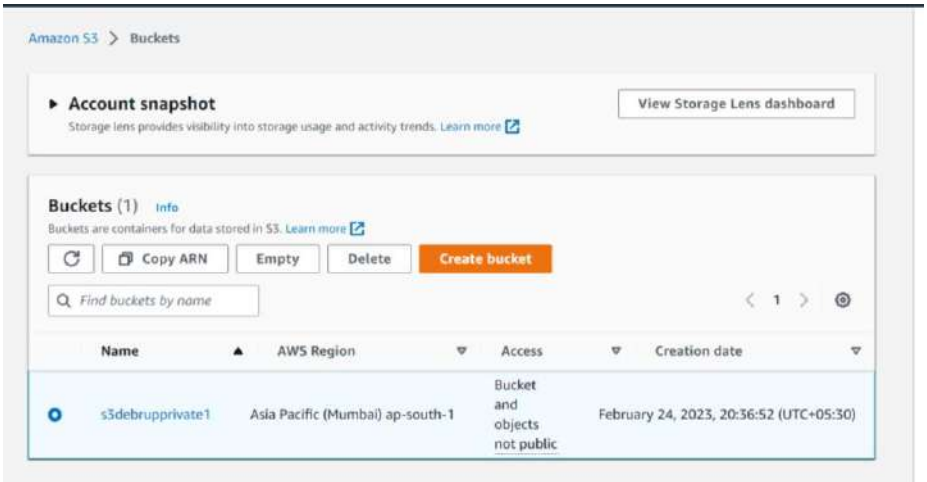




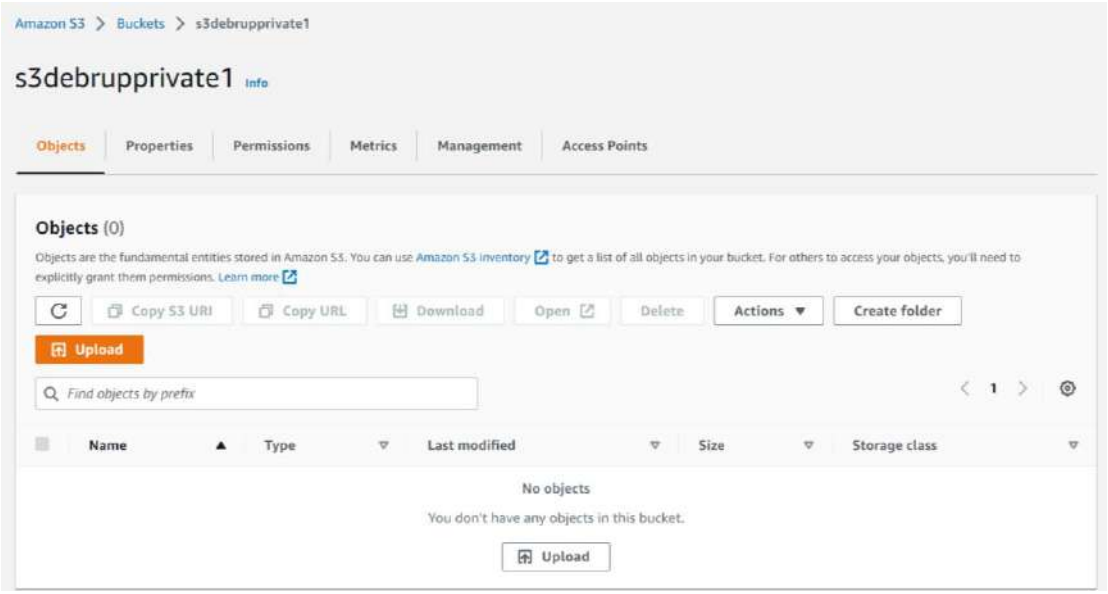




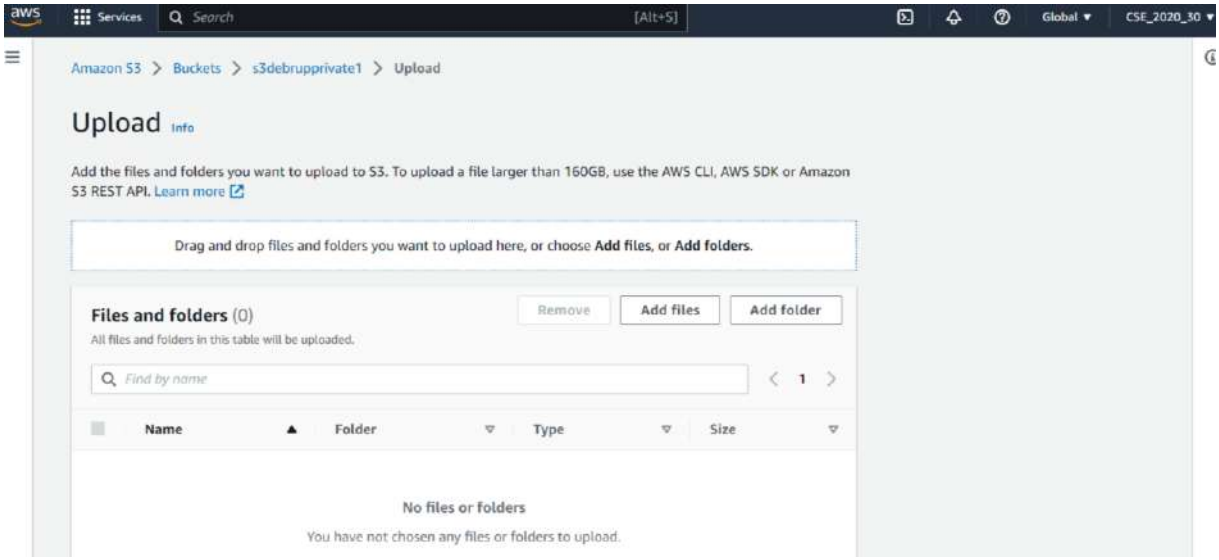
5. After that we are redirected to the buckets page where we can see all our buckets in a table format.



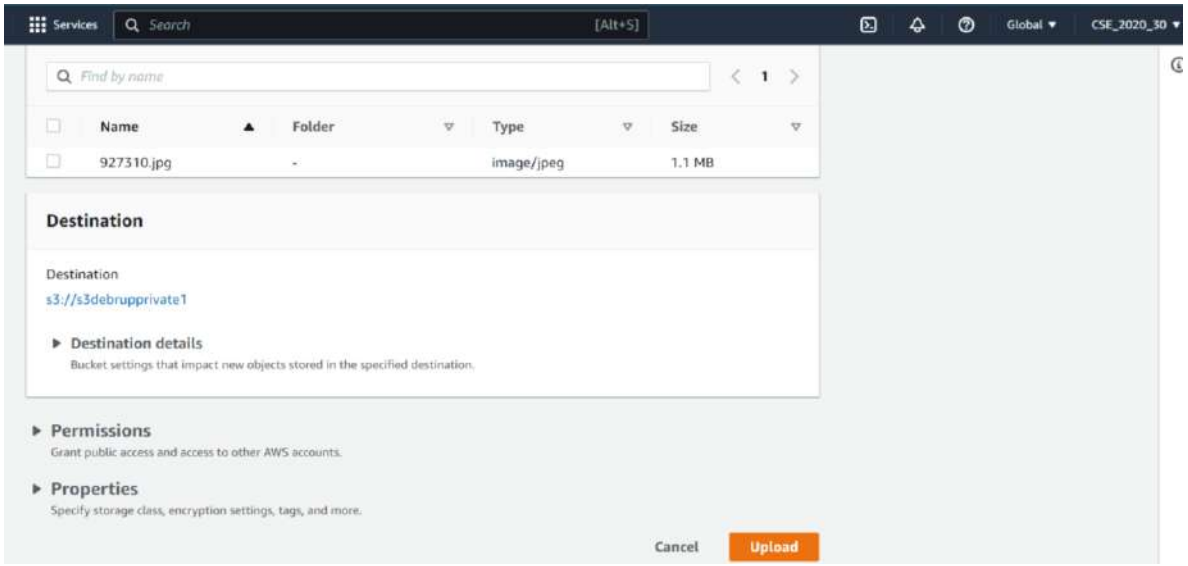
6. Now we click on our newly selected bucket (on the name).
7. Now we have successfully entered into our newly created bucket.



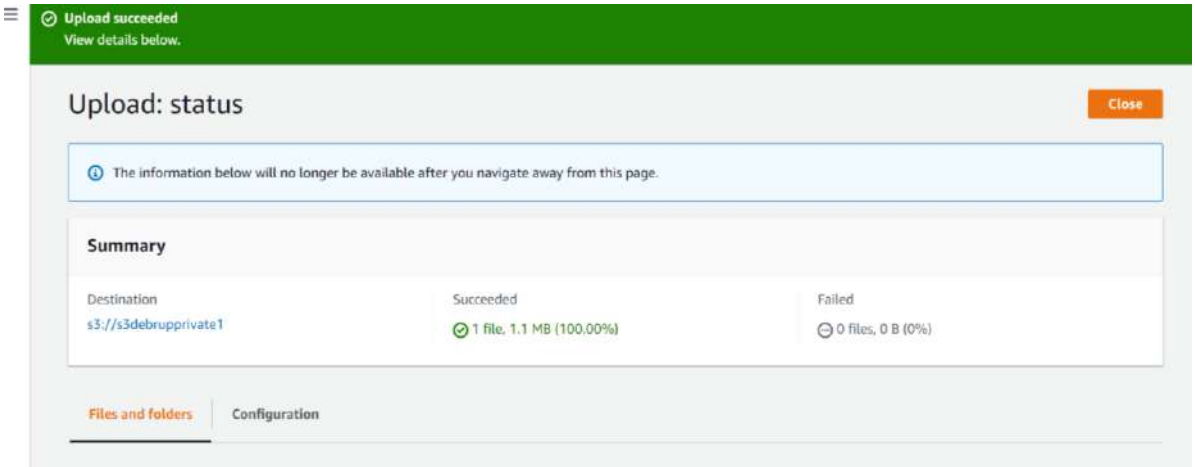
8. Click the Upload button to upload a file in our bucket.
9. After clicking you will be redirected to the Upload page. Click on Add files button to add a file.



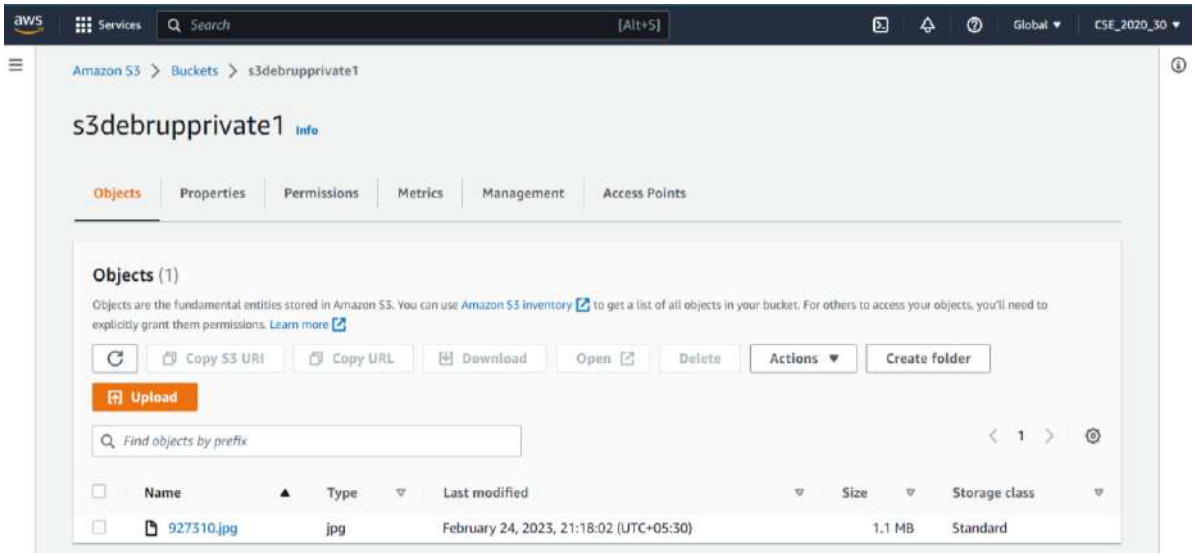
10. You will open a pop up to browse from your pc to upload a file. After selection click on upload button.



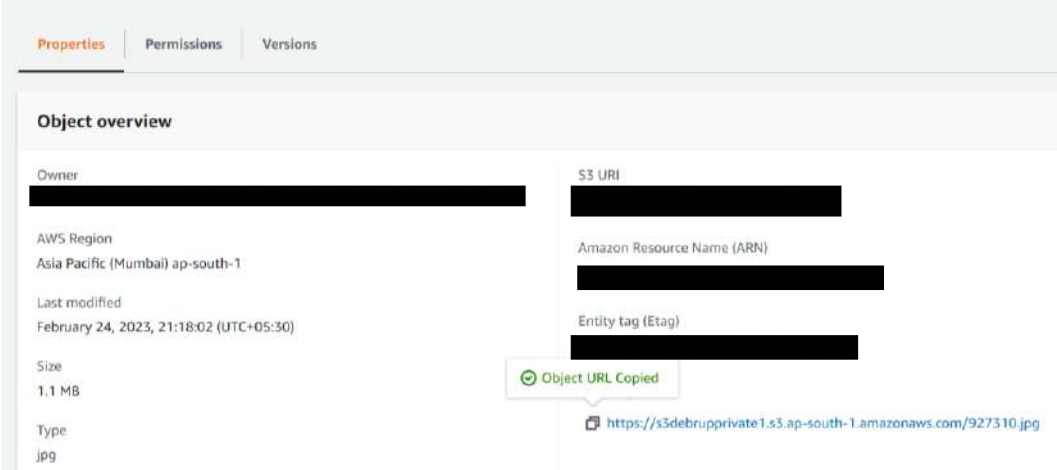
11. You will then be redirected to the upload status page where a status bar will be present showing the progress of your upload. After completion it will look like this.



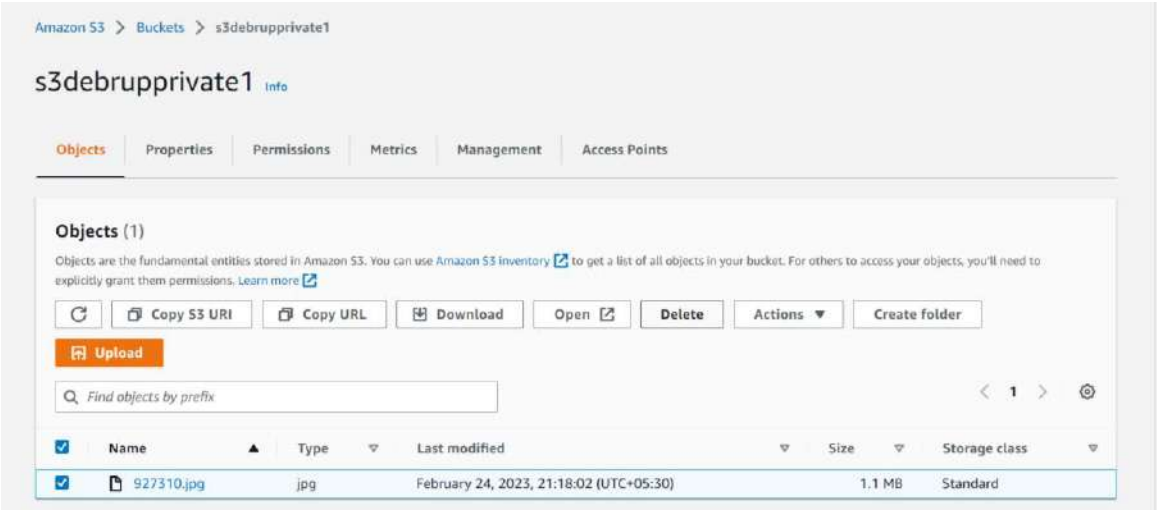
12. Close your status page. Now in the bucket page you will see the file you have uploaded in the objects section.



13. Now click on the file.
14. Scroll down and copy the Object URL.

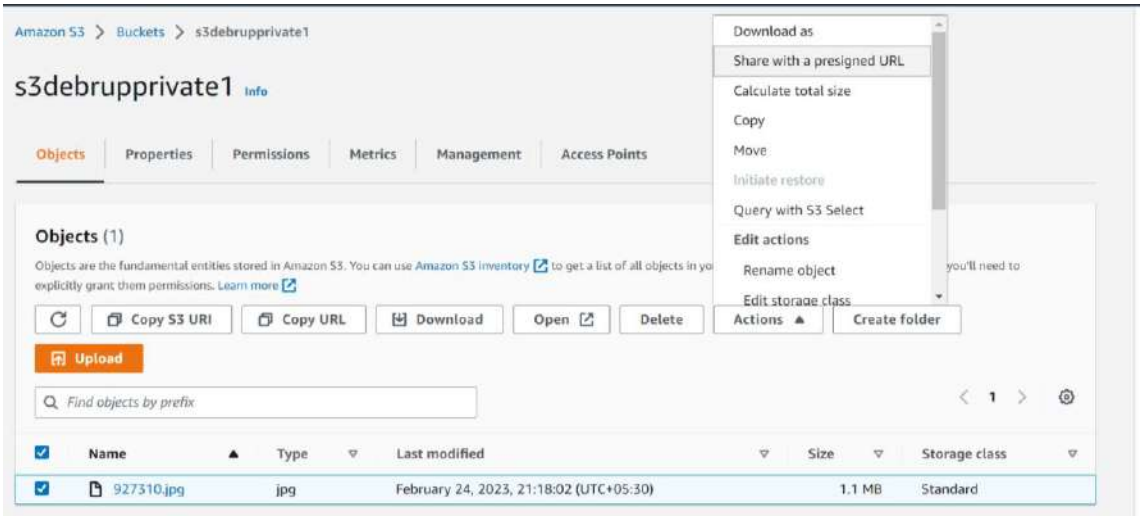


15. Paste it in another browser.
16. IT WILL SHOW ERROR.
    This is because your uploaded file is in a private bucket. Hence, it cannot be accessed by anyone other than you. Now, to let others access, you can only send them a pre-signed URL which remains active for a specific duration.
17. NOW WE WILL GENERATE A PRESIGNED URL
18. Go back to the previous page and select your file.



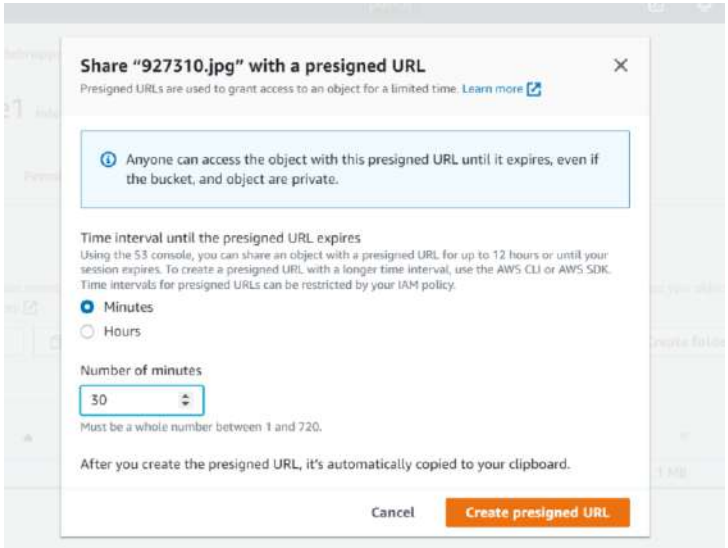
19. Next click on the Actions button as shown above.

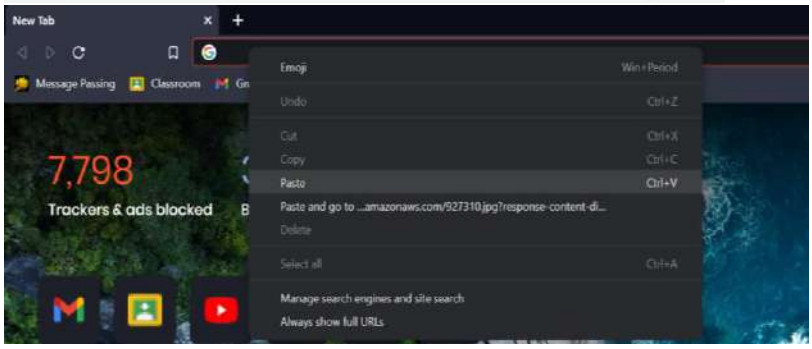
20. Select the “Share with presigned URL” option.



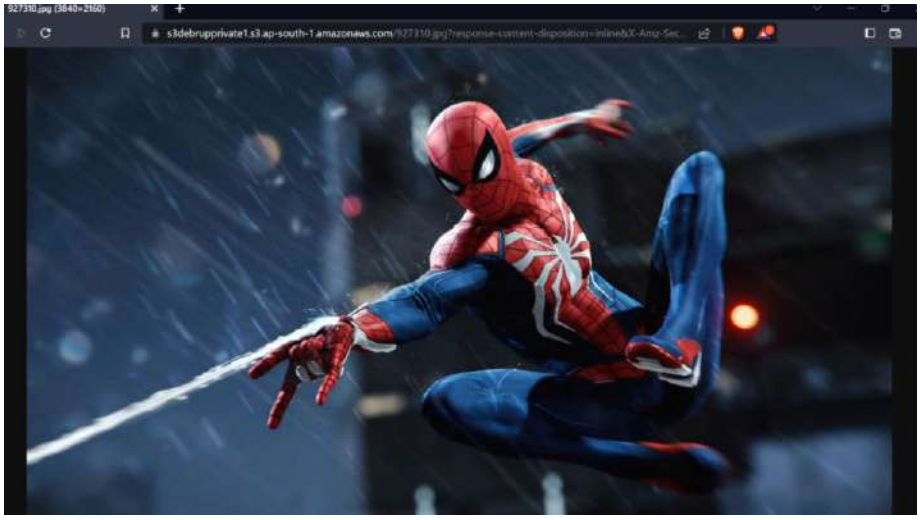
21. Now a pop-up will appear as shown below. You have to specify the duration for which the link remains active. Next click on Create presigned URL.

    Note that after creation the URL link automatically gets copied (in your clipboard). So you do not have to manually copy it. Just right click and paste it in another browser(Or use Ctrl+V shortcut in the browser search box)





22. After pasting the link in the bar, press Enter key. Now we can access our file using the presigned URL.



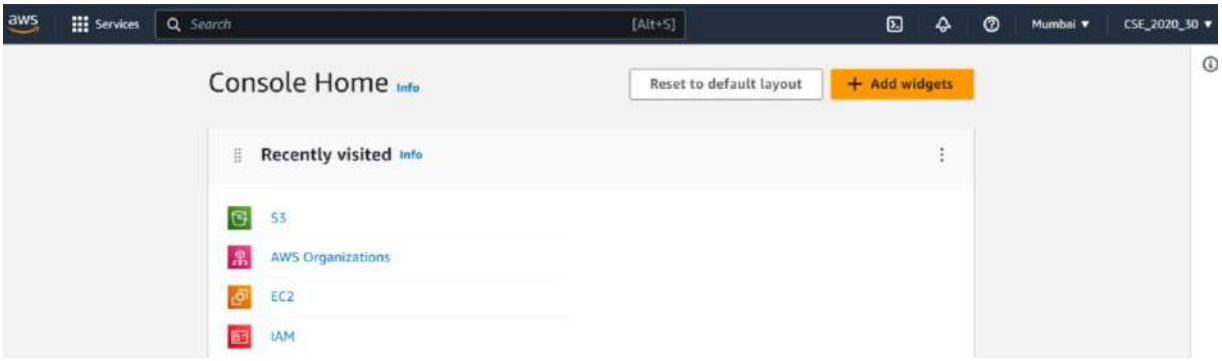
So, our file is private and can only be accessed by those with the pre-signed URL link for a limited duration.

ASSIGNMENT 5

Problem Statement: Create a public bucket in AWS. Upload a file and give the necessary permission to check the file URL is working or not.

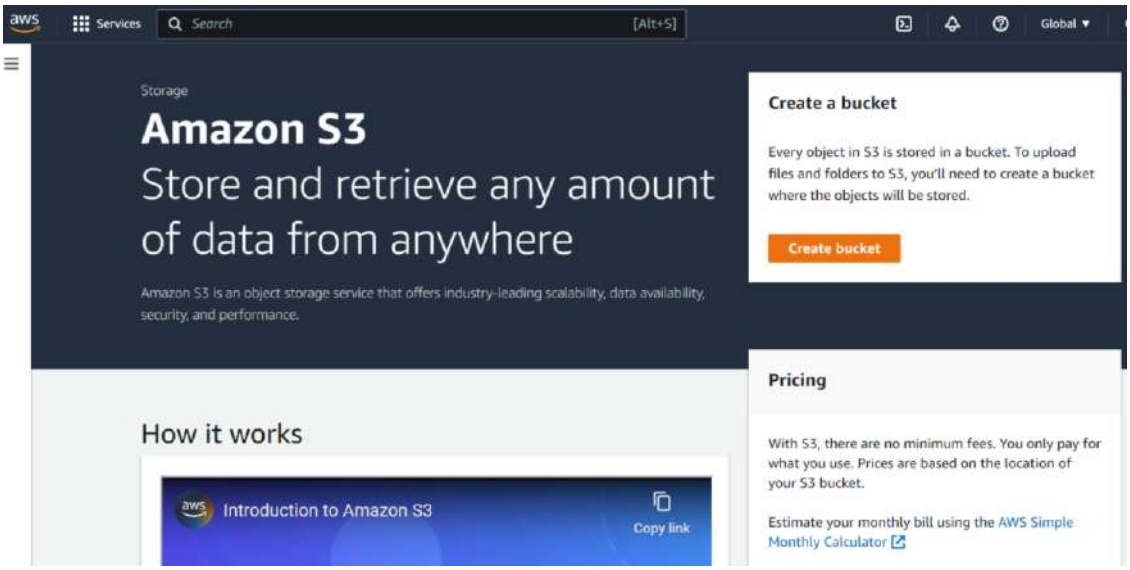
Procedure:

1. Sign in to your **AWS account** as root user.

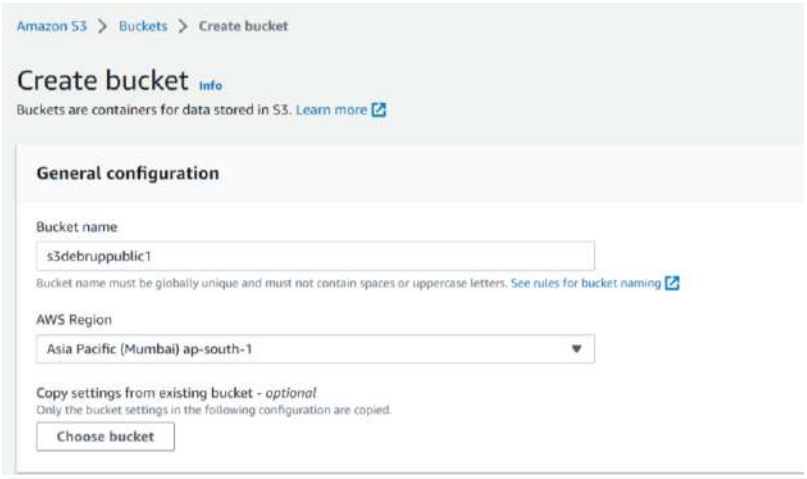


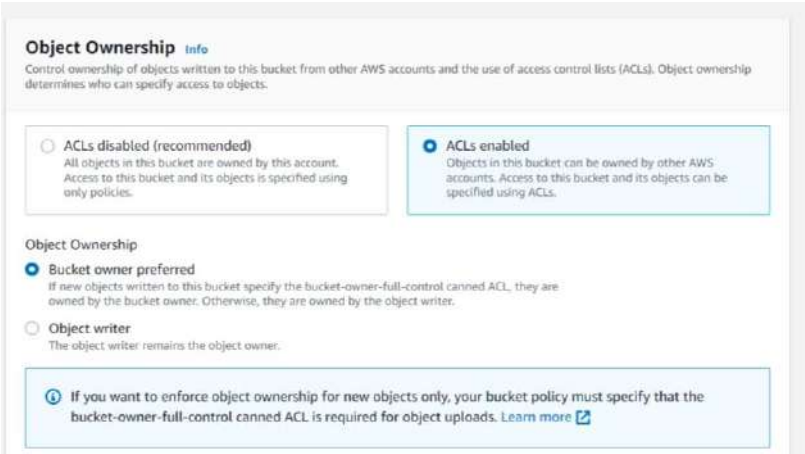
2. Now in the **homepage** search for **S3** in the **search box** and then select the first option displayed.

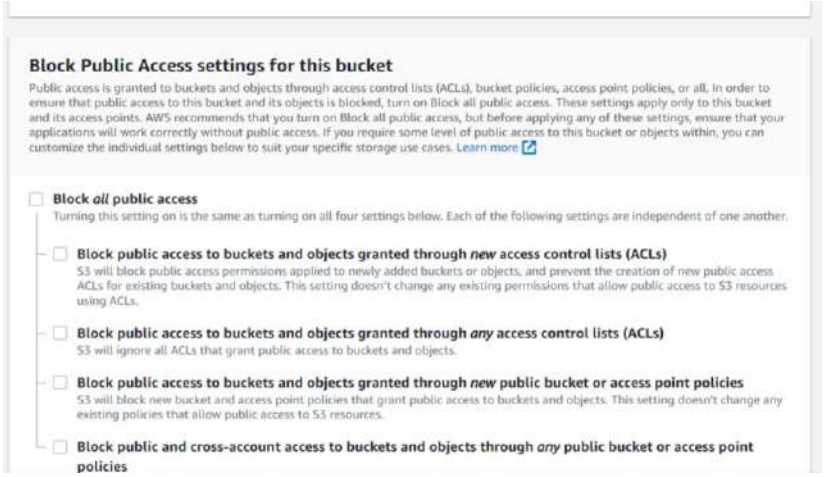
3. After clicking on it, you will be redirected to the **Amazon S3** homepage. There we have to click on the **create bucket** button on the right-hand side.

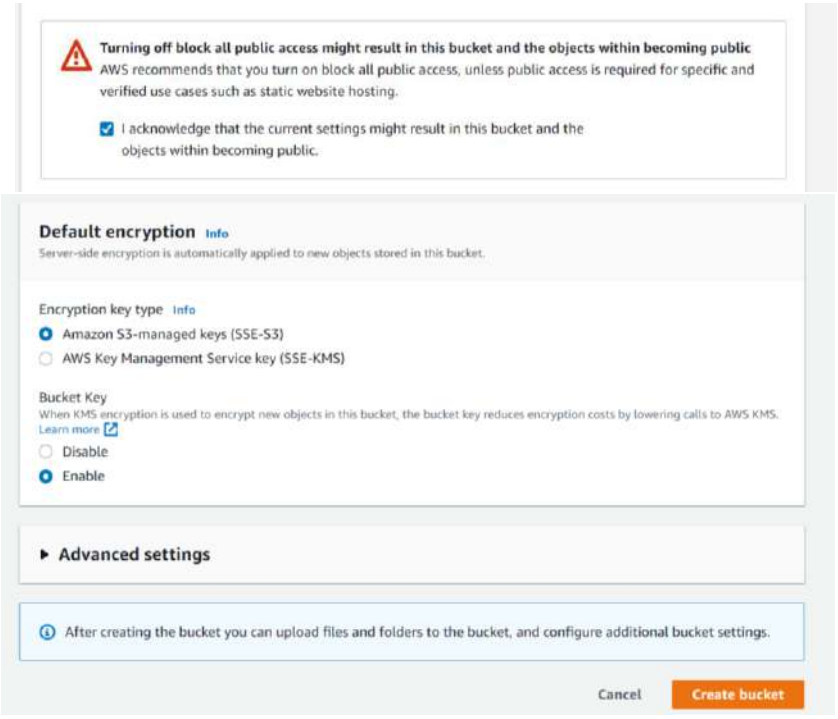


4. Next you will go to the **Create bucket screen** where you have to configure your bucket before creating it.
    a. Choose a globally unique name for your bucket. It should NOT contain any spaces or any uppercase letters.
    b. Select the **AWS Region** as **Asia Pacific (Mumbai) ap-south-1**. **Remember** you can avail other options but each server region has **different pricing** associated with it. Since, we are **living in India**, we are choosing the one **closest to us** to remain fairly priced.
    c. Next, we go to Object Ownership section where we keep ACLs enabled option checked. We also keep “Bucket owner preferred” option checked in the object ownership choice
    d. Next, we UNCHECK the Block all public access option.
    e. Remember to check the “I acknowledge that the current settings might result in this bucket and the objects within becoming public” option just below.
    f. Everything else remains unchanged.
    g. Now click on the Create bucket button.

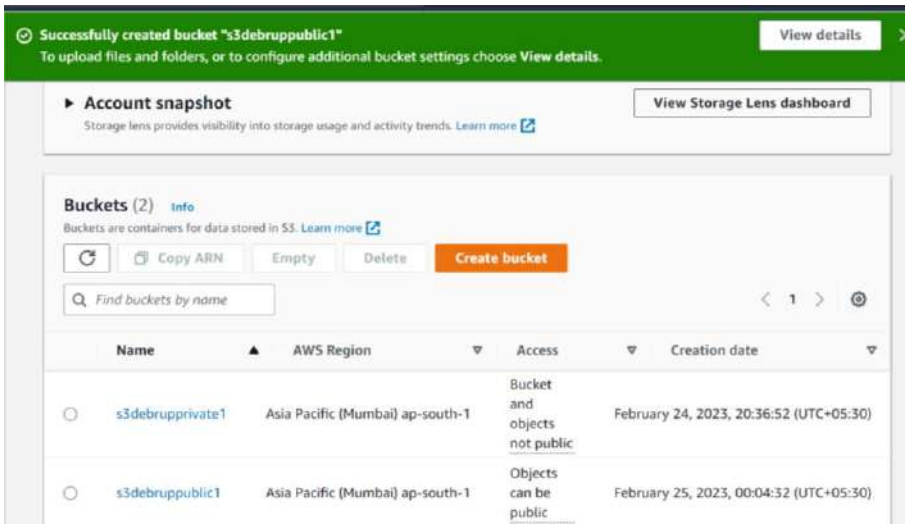




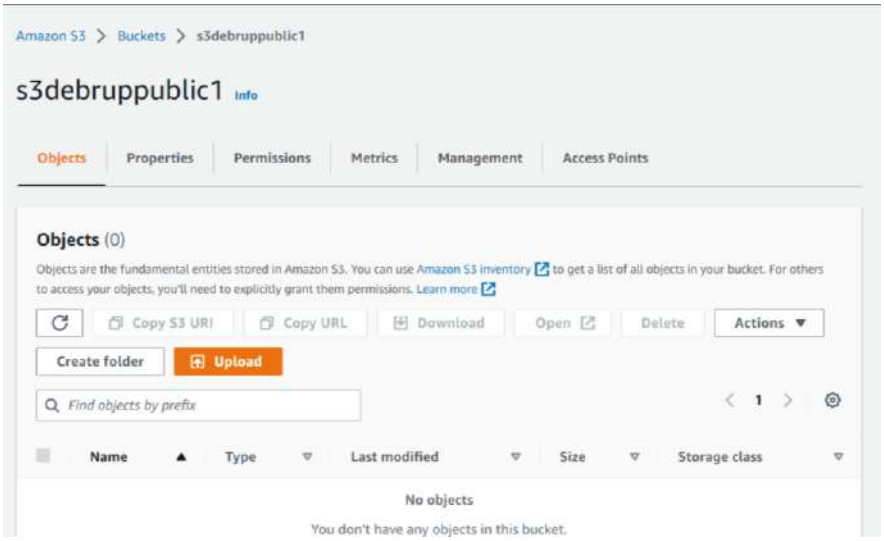




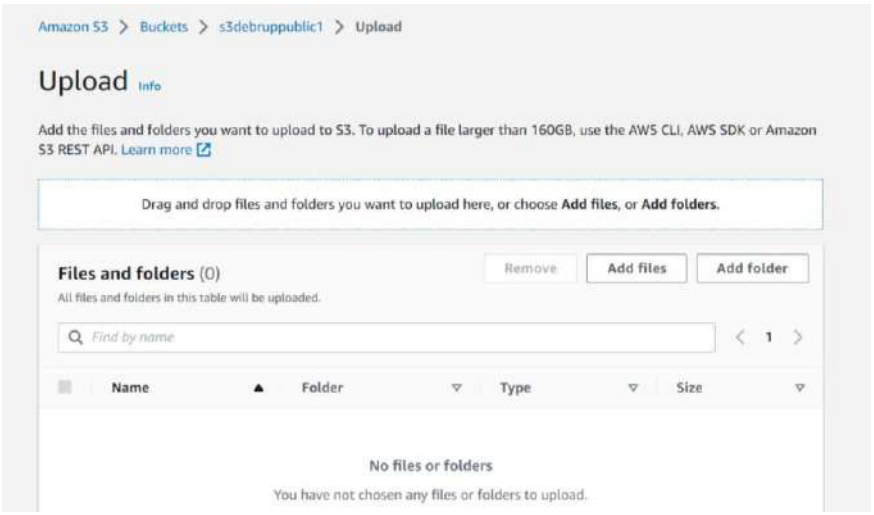
5. After that we are redirected to the buckets page where we can see all our buckets in a table format.



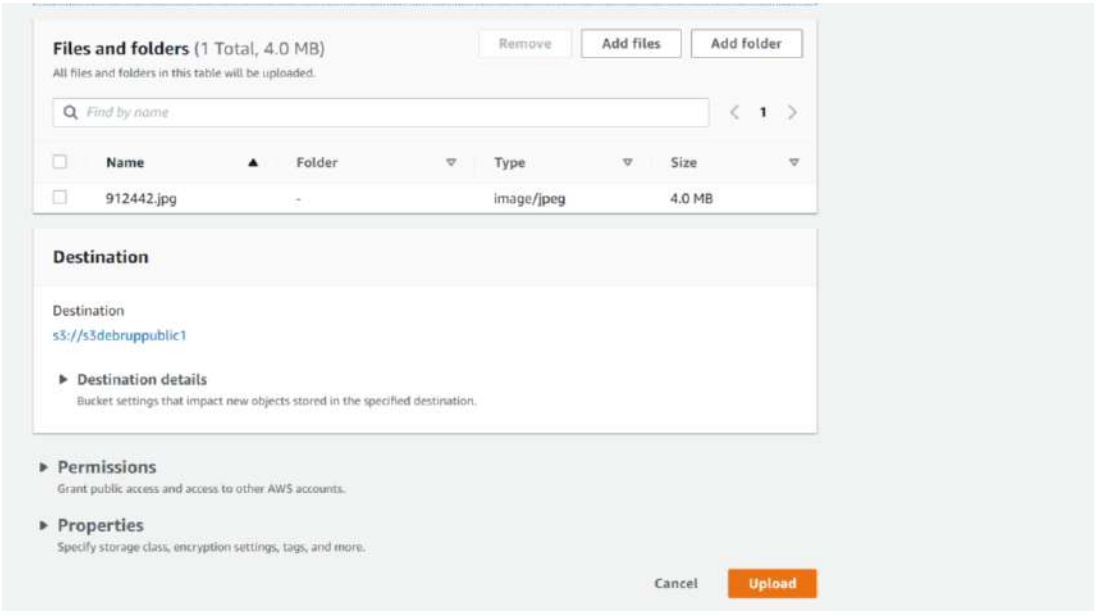
6. Now we click on our newly selected bucket (on the name).
7. Now we have successfully entered into our newly created bucket.
8. Click the Upload button to upload a file in our bucket.



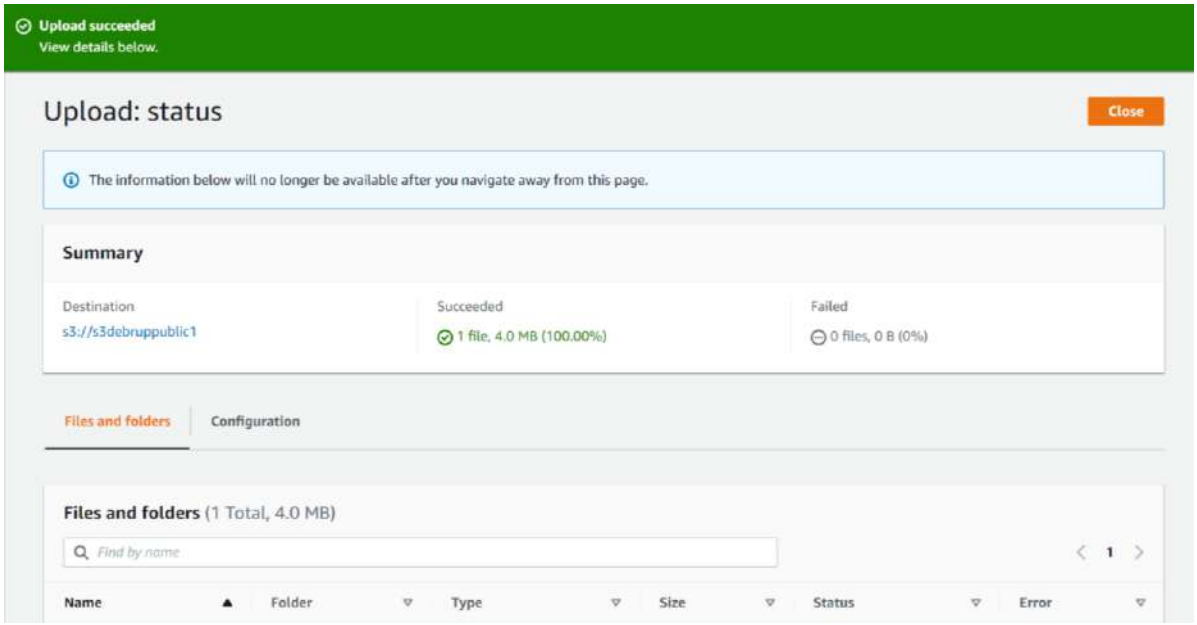
9. After clicking you will be redirected to the Upload page. Click on Add files button to add a file.



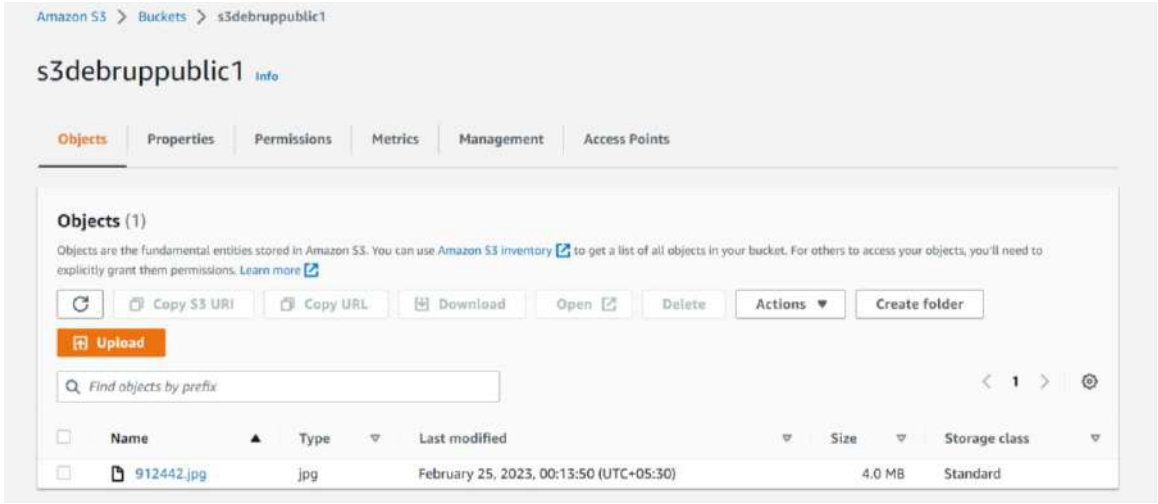
10. You will be shown a box to browse from your pc to upload a file. After selection click on upload button.



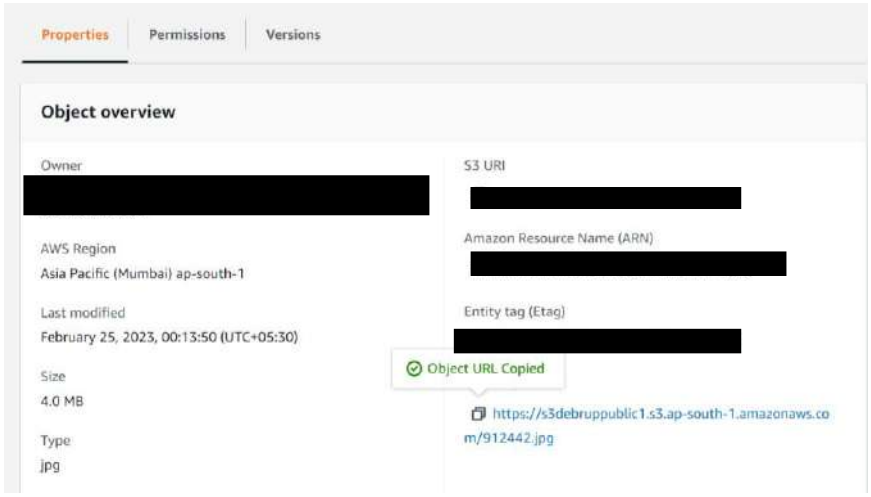
11. You will then be redirected to the upload status page where a status bar will be present showing the progress of your upload. After completion it will look like the following.



12. Close your status page. Now in the bucket page you will see the file you have uploaded in the objects section.



13. Now click on the file.
14. Scroll down and copy the Object URL.



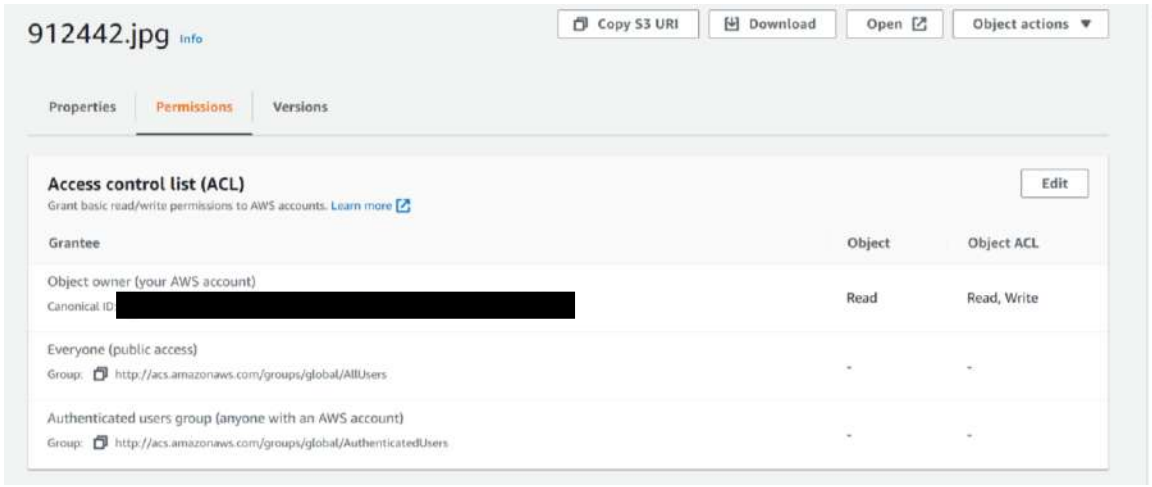
15. Paste it in another browser.
16. IT WILL SHOW ERROR.

    This is because your uploaded file, though present in a public bucket, has to be given specific permission so that others can access it. Hence, it cannot

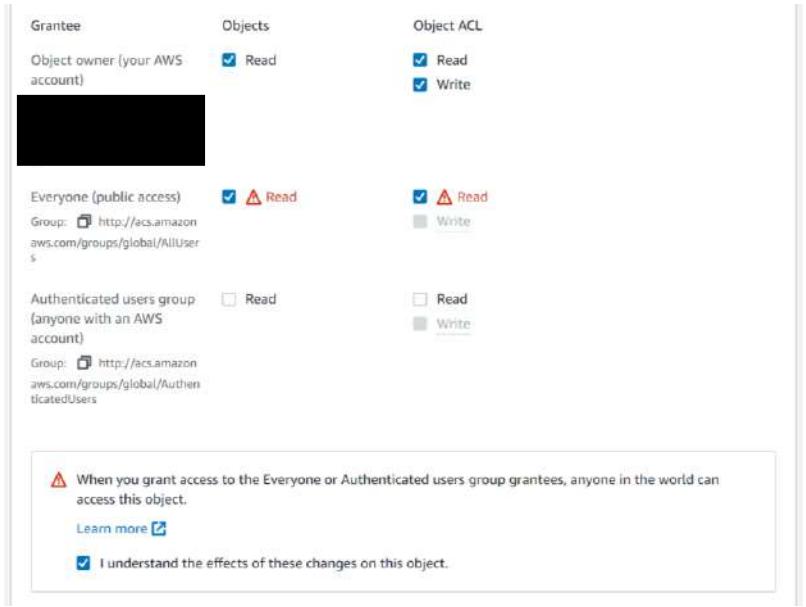
be accessed by anyone other than you. Now, to let others access, you can change the permissions associated with your file using the ACL.

17. NOW WE WILL GIVE PERMISSIONS…….
18. Scroll to the top and Click on the permissions bar on the top below the filename and beside the properties bar(in orange font).

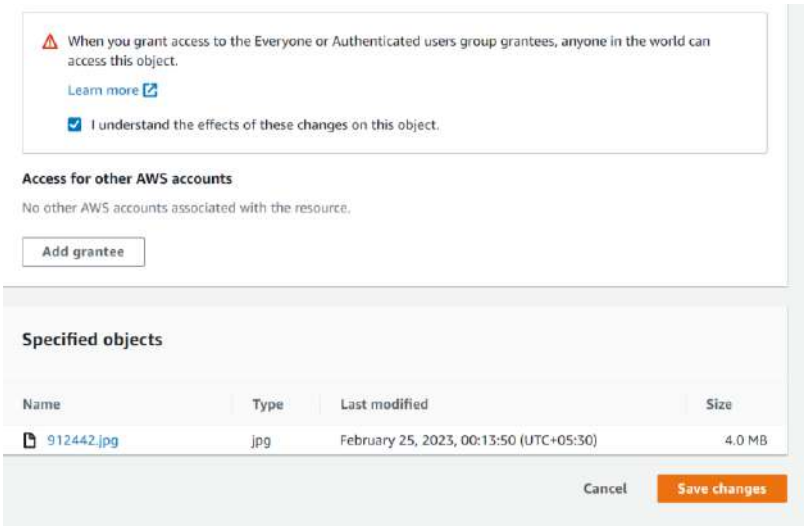
19. You will arrive in the permissions section of your file. You will clearly see the first section as the ACL or Access Control List of your file. Now click on the Edit button on the right hand side of the ACL section.



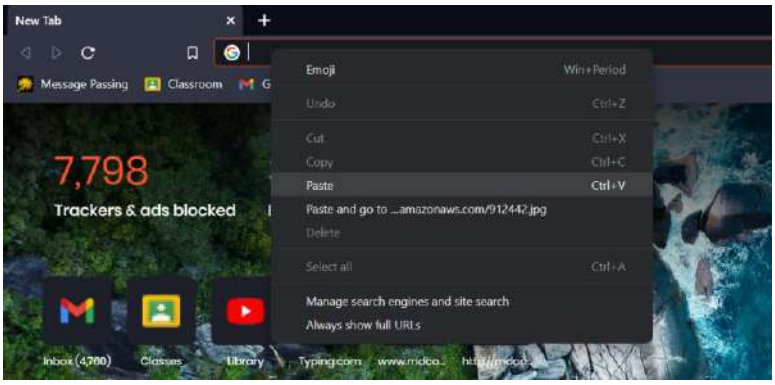
20. Now select CHECK both “Read”(Object & Object ACL) option for **Everyone(public access)**
    Also, remember to check the I understand the effects of these changes on this object option, to further proceed.



21. Now, scroll down and click the save changes button.



22. Now, again copy the object URL and paste it in a different browser.



23. After pasting the link in the bar, press Enter key. Now we can access our file using the object URL directly.

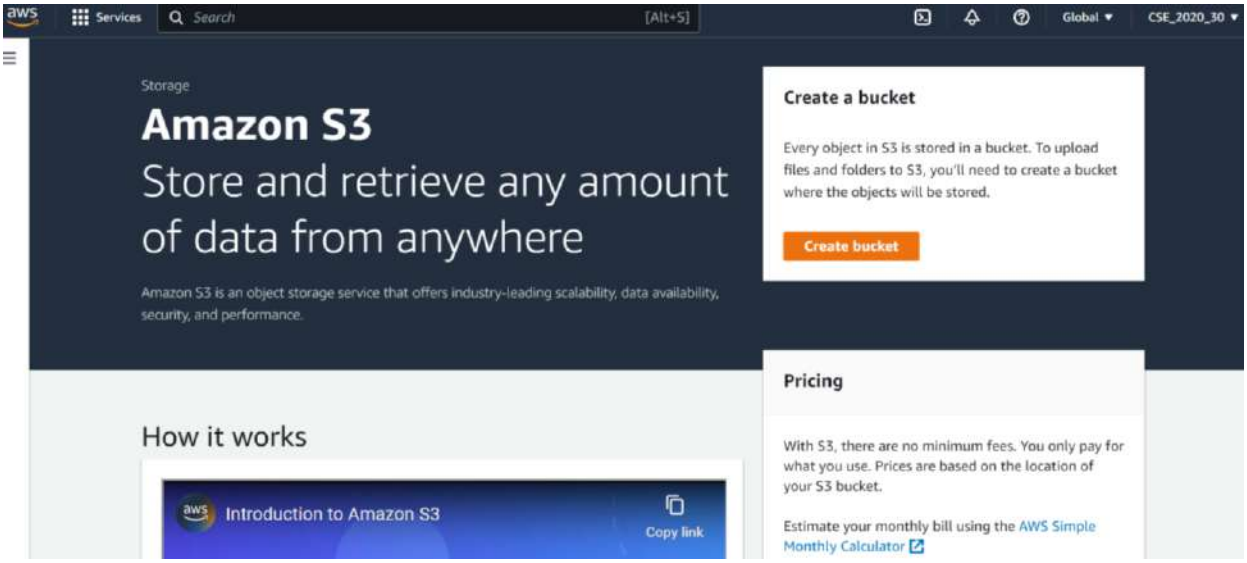
So, our file is public and can be accessed by those with the object URL(link) anytime anywhere. The URL is working perfectly as intended.

ASSIGNMENT 6

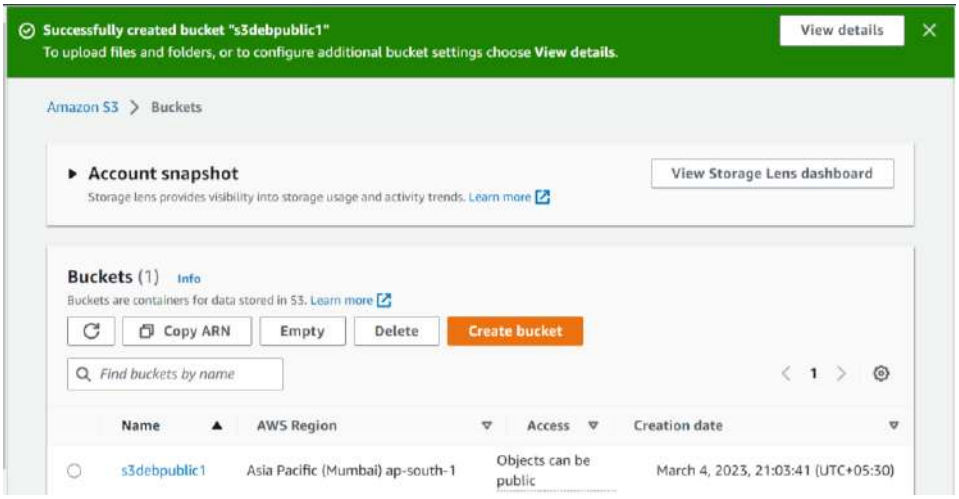
Problem Statement: Upload a static website on S3.

Procedure:

1. Sign in to your AWS account as a root user.



2. Then create a **public S3 bucket**. (Refer to Assignment 5 for full procedure)



3. Now create three html files anywhere in your computer (preferably in a folder in the Desktop).
    a. The first one is named Home.html
    b. The second one is named About.html
    c. The third one is named next.html

    Remember, you can give any name to the files but you have to modify the steps shown further accordingly. We are going to use the given file names to proceed.

The contents of the files are shown in the given snapshots……

Home.html

```
<html>
<body bgcolor="yellow">
<h1>Welcome to my page</h1>
<a href="next.html"><h2>Go to Next</h2></a>
<a href="About.html"><h2>Go to About</h2></a>
</body>
</html>
```

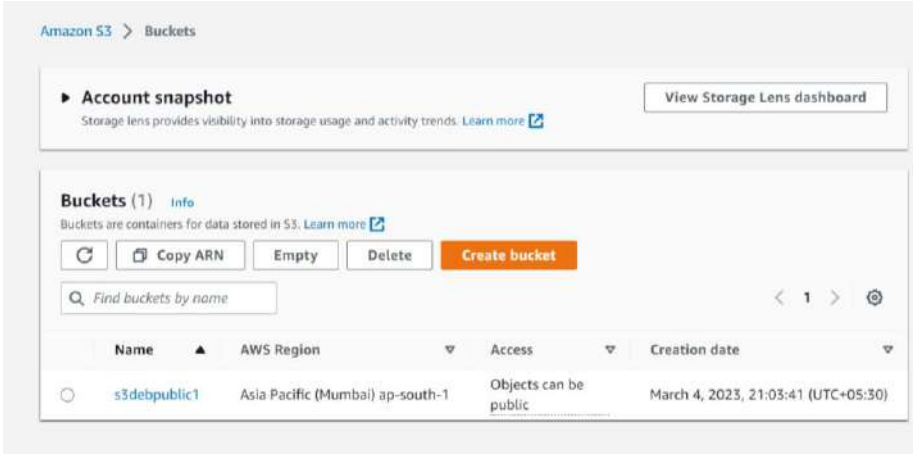
About.html

```
<html>
<body bgcolor="pink">
<h1>Hello World</h1>
<a href="Home.html"><h2>Go to Home</h2></a>
<a href="next.html"><h2>Go to Next</h2></a>
</body>
</html>
```

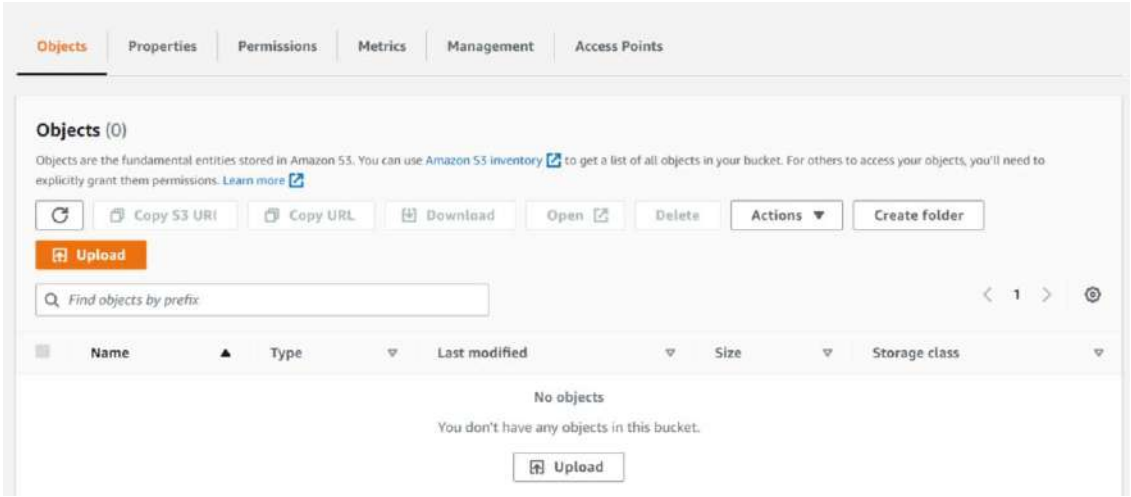
next.html

```
<html>
<body bgcolor="aqua">
<h1>Welcome to my page</h2>
<a href="Home.html"><h2>Go to Home</h2></a>
<a href="About.html"><h2>Go to About</h2></a>
</body>
</html>
```

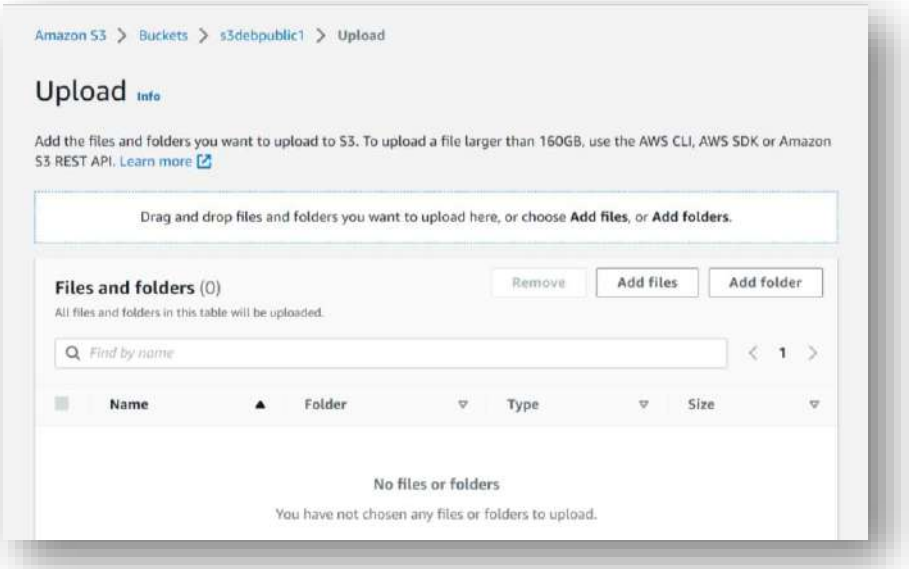
4. Now let us go back to AWS. Select the newly created public bucket.

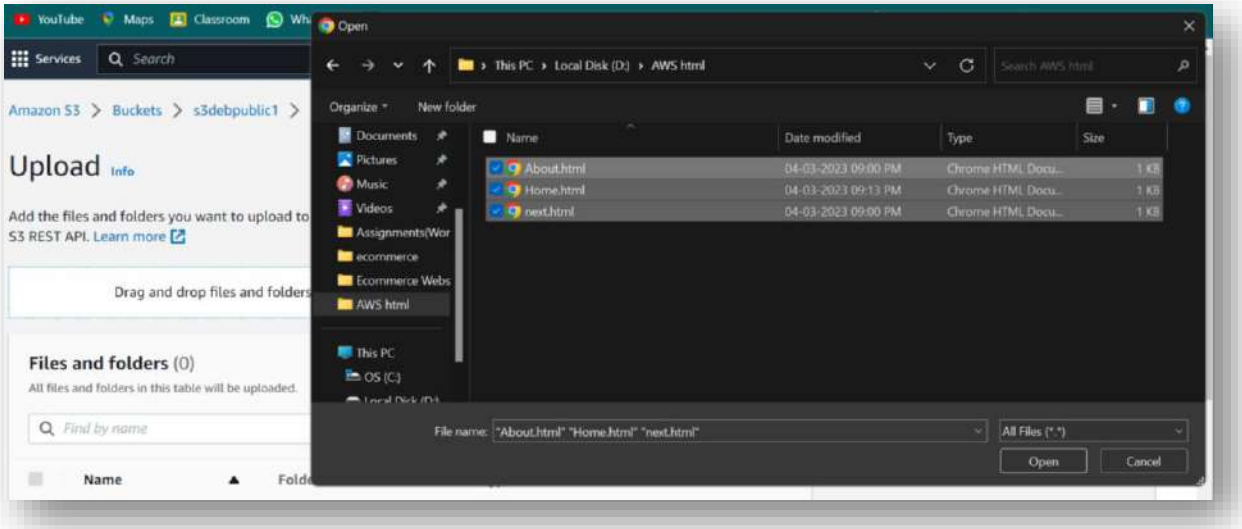


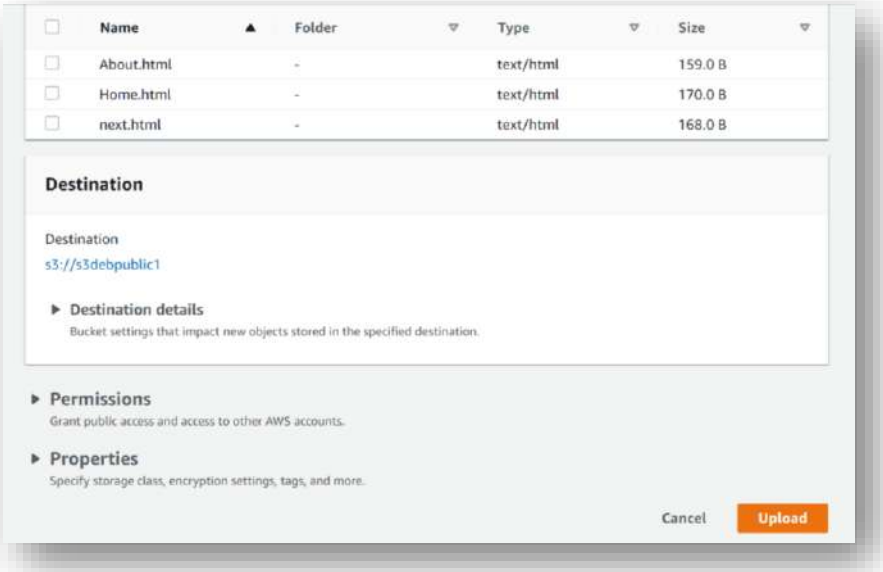
5. Click the upload button.



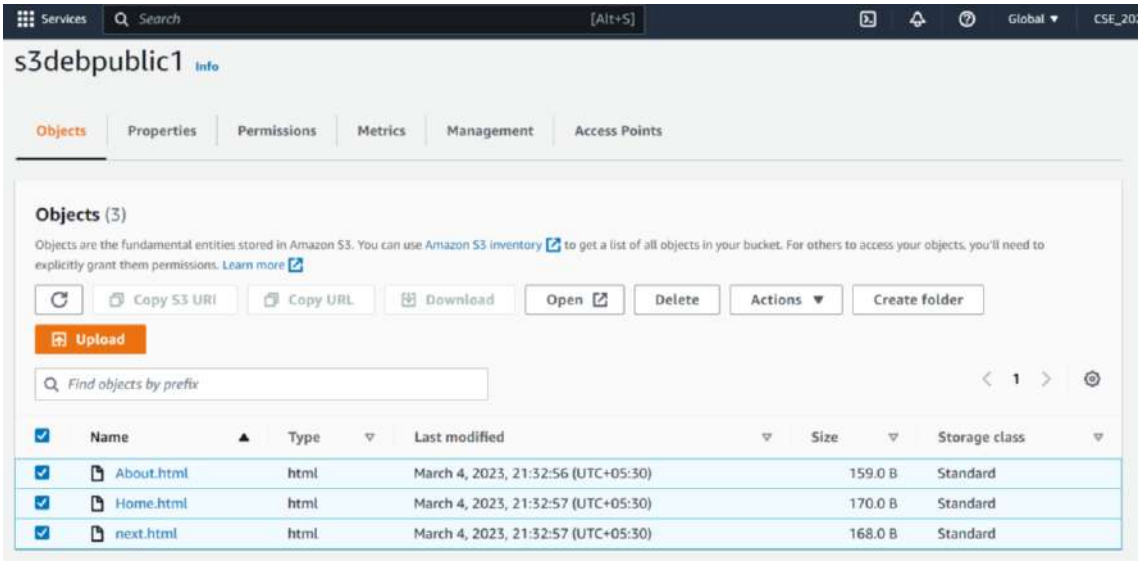
6. Next click on the add files button. Select all the three html files and upload them by pressing the upload button.



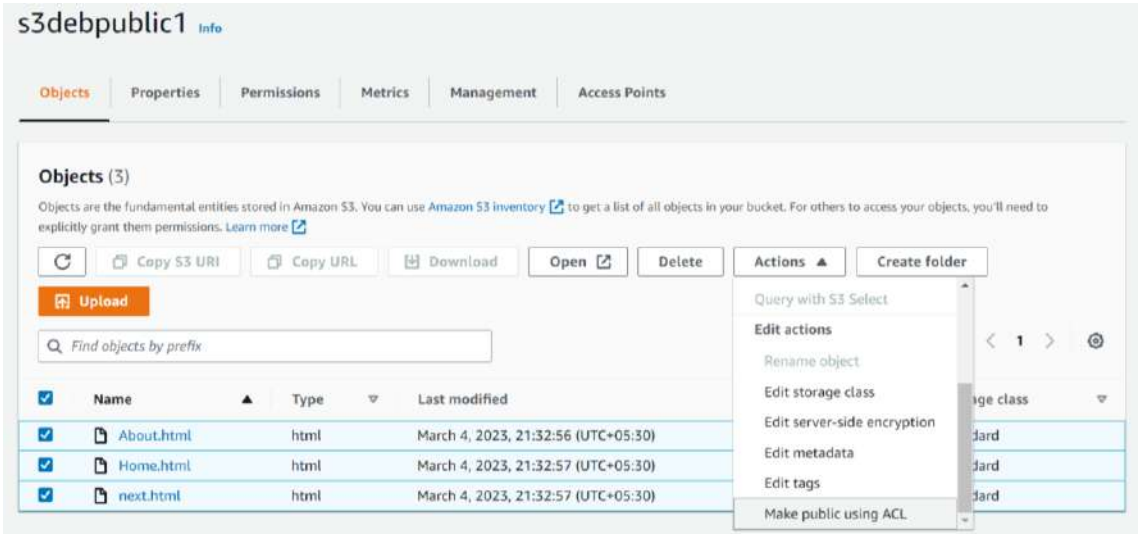




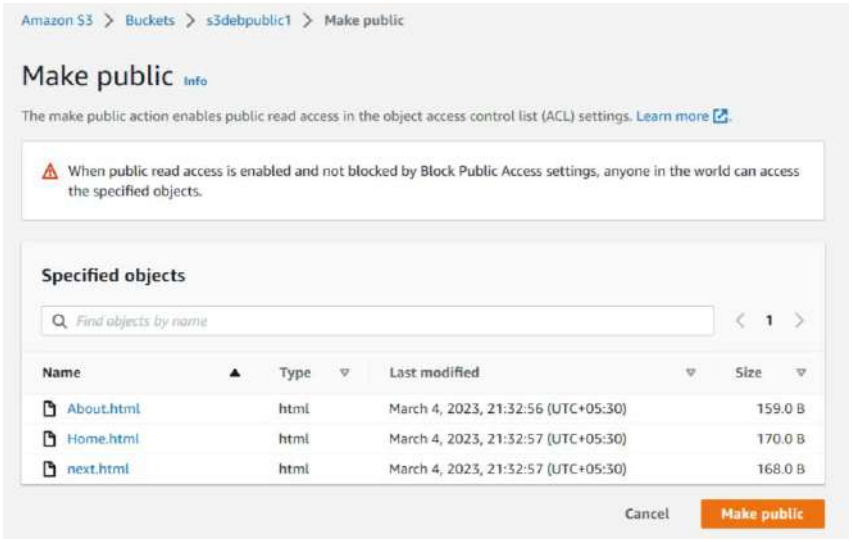
7. Now, after closing the upload status page after all files are updated, select all the files shown in the object table in the public bucket.



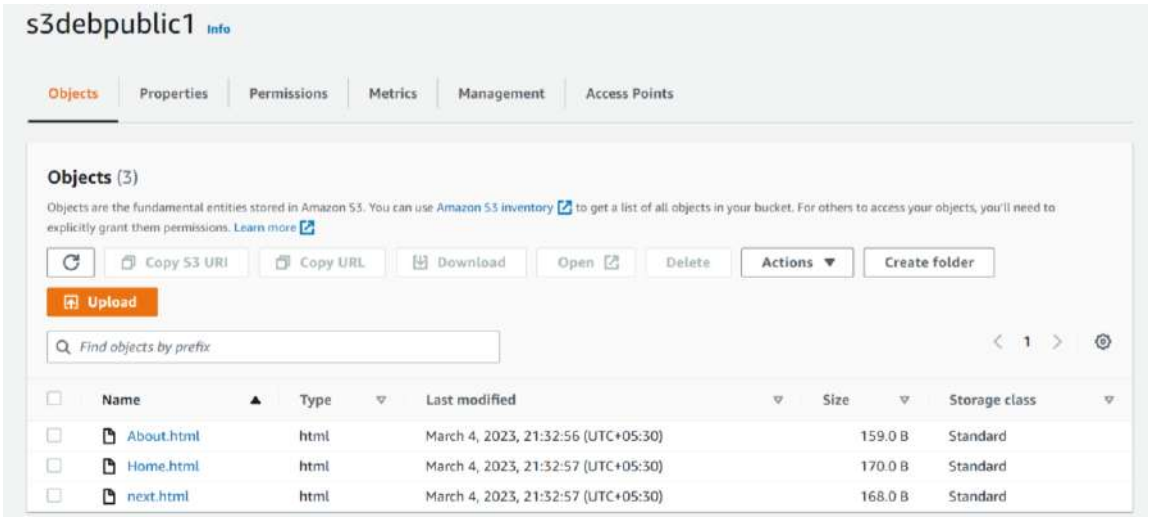
8. Click on the actions button and from the subsequent dropdown menu select Make public using ACL.



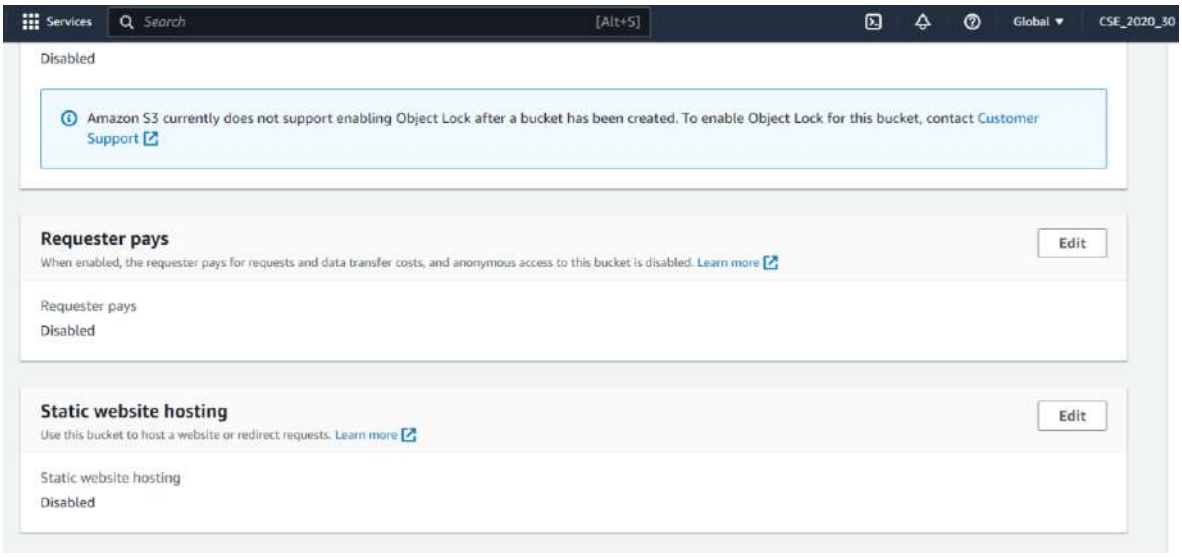
9. Click on Make public button to make all the selected files public.



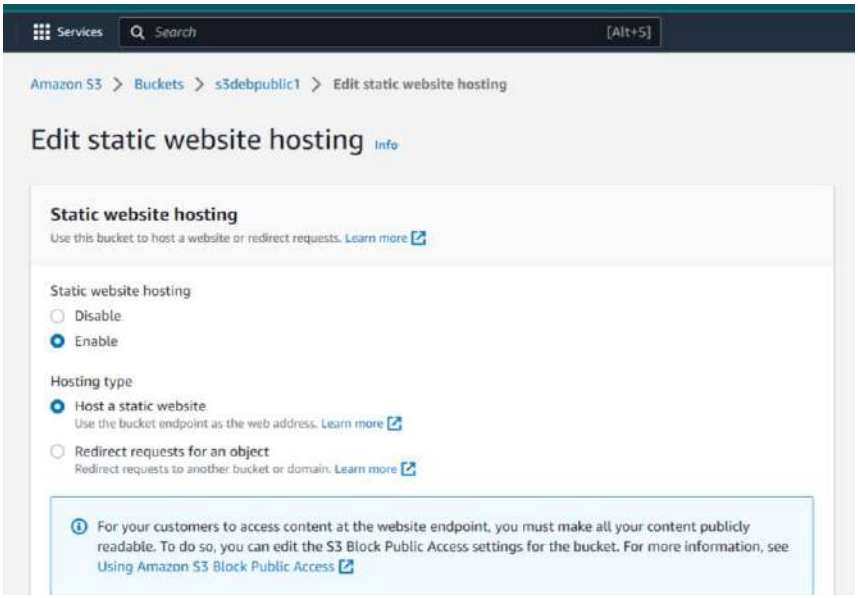
10. Now, we after making public we close the status page and we are redirected to our public bucket page. Now we select the properties tab of the bucket which is located in the right side of the objects tab (in orange font).



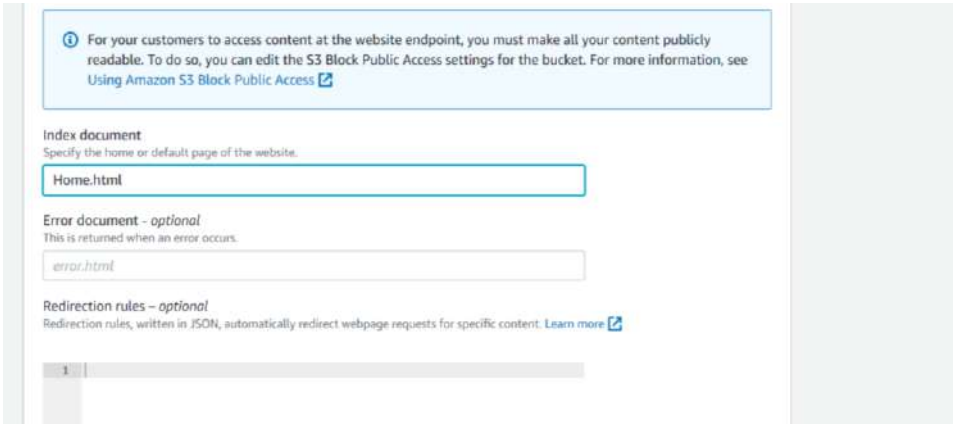
11. After arriving in the properties tab. Scroll down all the way to the bottom.
    We will focus only on the Static website hosting section.
    By default, it will show disabled.
    We have to enable it. In order to do so click on the Edit button on the right-hand side of the section.



12. Now we will choose the enable option. After choosing it multiple options will appear. Just follow the snapshot provided below and make the same changes.

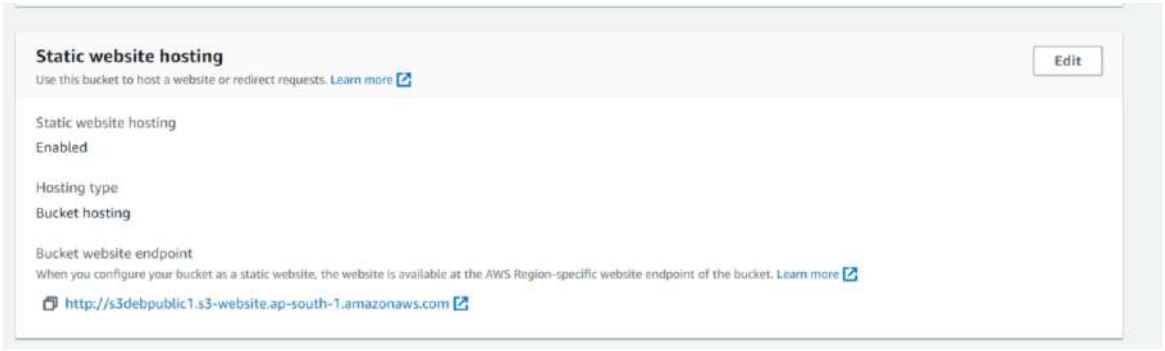


13. Next scroll down. Now we have to mention the html document our link will show. This is the one which anyone can access using the bucket link. In our case we will mention Home.html as our index or main html file. You can choose according to your wish and design.



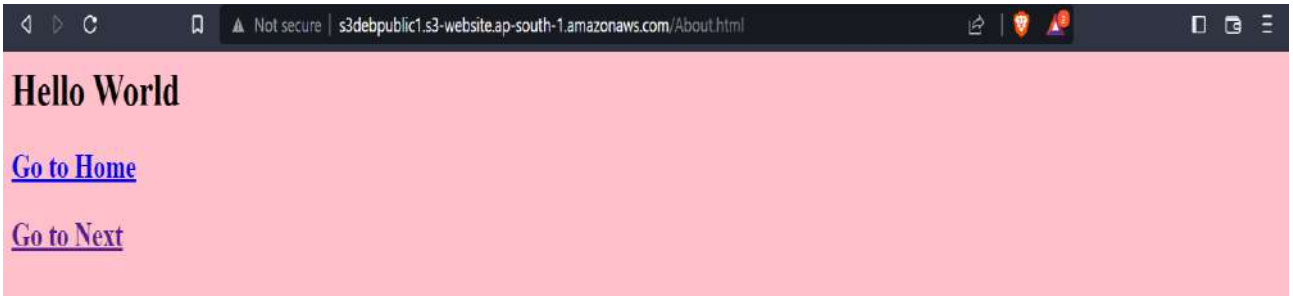
14. Next scroll down and click on save changes button.

15. Now, you will again arrive in the bucket's properties tab. Scroll down to the static website hosting area. Now you can see a link has arrived.



16. Copy the link and paste it in another browser.

17. We are now viewing the Home.html page. And as scripted, we can click on the Go to Next to arrive at the next.html page. Similarly, we can click on the Go to About page to go the About.html page.



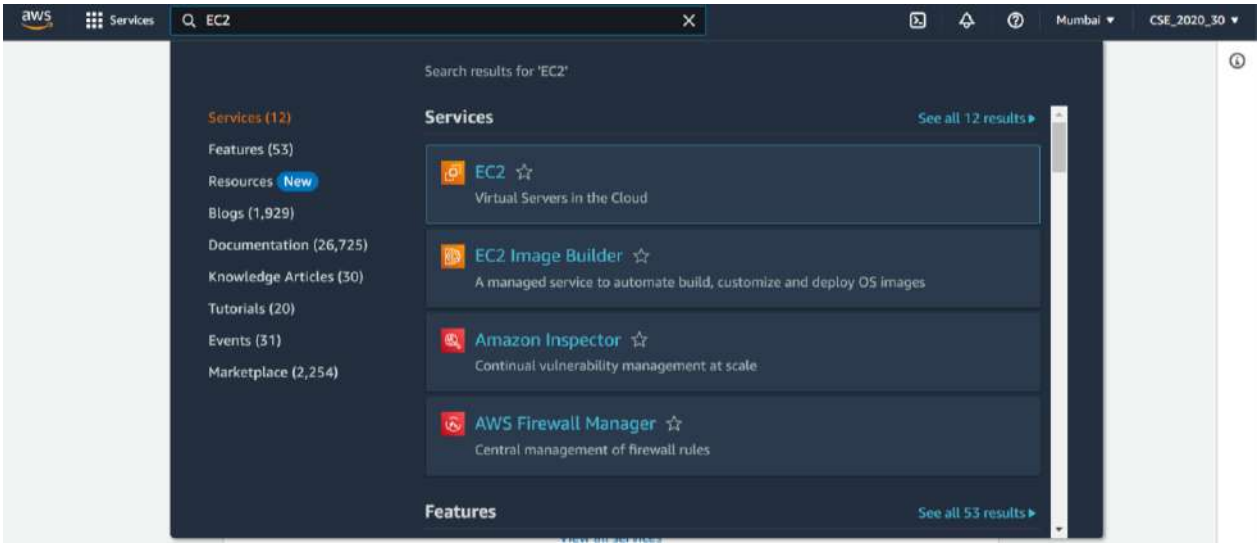
So, we have successfully hosted a static website in amazon S3.

ASSIGNMENT-7

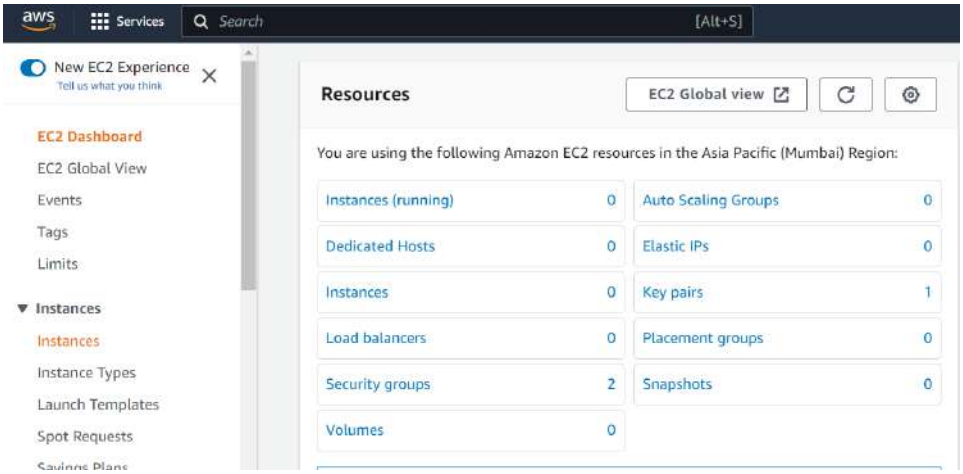
Problem Statement: Upload a static website on EC2.

Procedure:

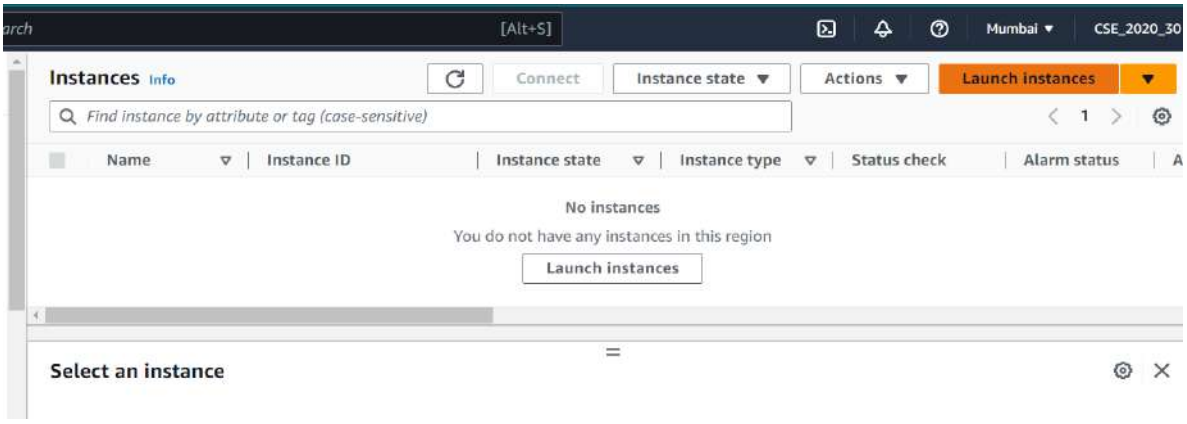
1. Login to your AWS account as root user. Then search “EC2” in the search box. Click on the first result that appears.



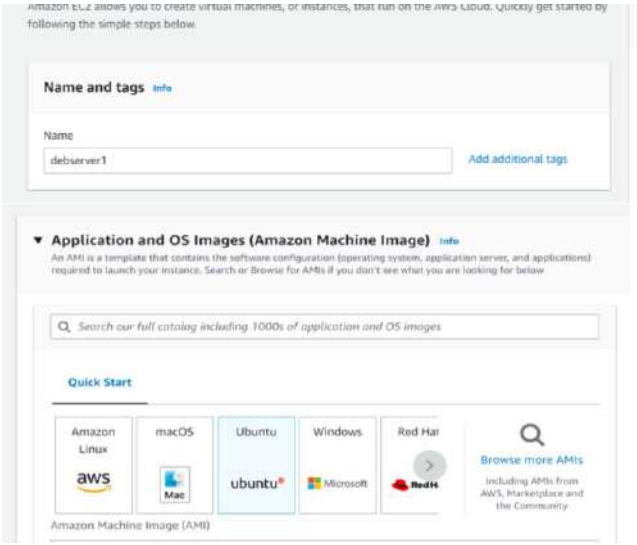
2. Click on Instances dropdown menu on the left sidebar. Then again click on instances.

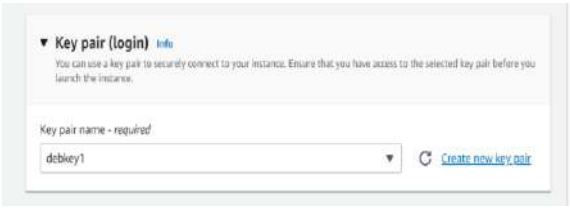


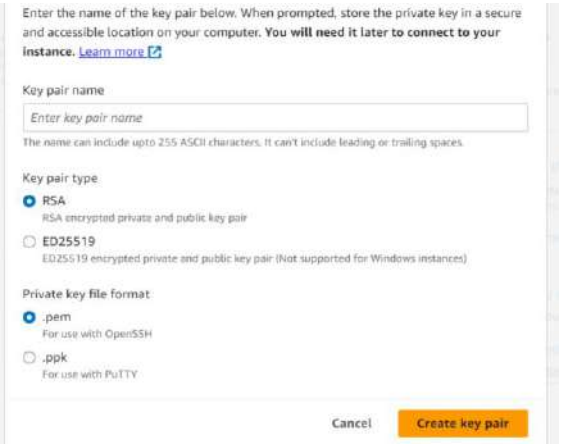
3. Next click on Launch instances button.

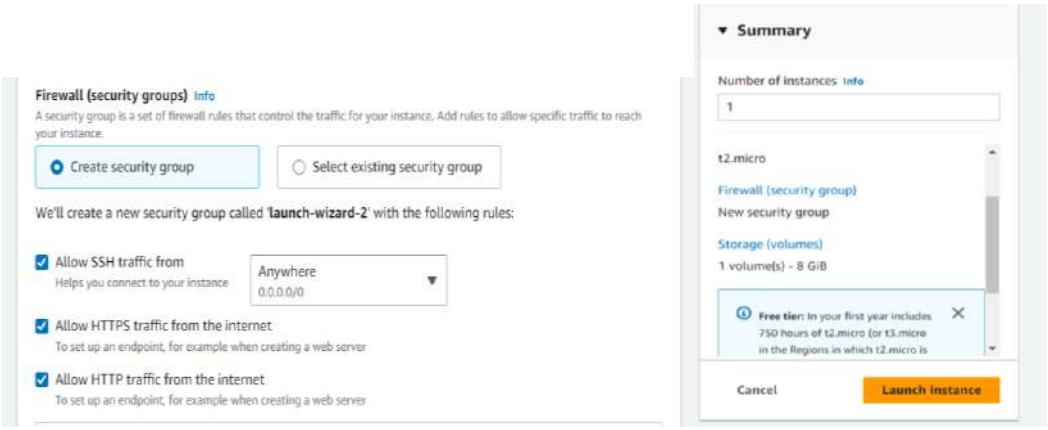


4. Now customize the instance you want to launch.
    a. Set the unique instance name.
    b. Select Ubuntu as OS.
    c. Next go to key pair(login) section.
        i. Click on create new key pair
        ii. Enter the name of key pair.
        iii. Select RSA as Key pair type.
        iv. Select ". pem" as file format.
        v. Create the key pair.
        vi. Save the automatically downloaded file. It will be required later.
    d. Now select the newly created key pair from the dropdown selection.
    e. Go at the bottom of the network settings section and check the
        i. Allow HTTP traffic box.
        ii. Allow HTTPS traffic box.
    f. Next Click on Launch Instance button on the right side.

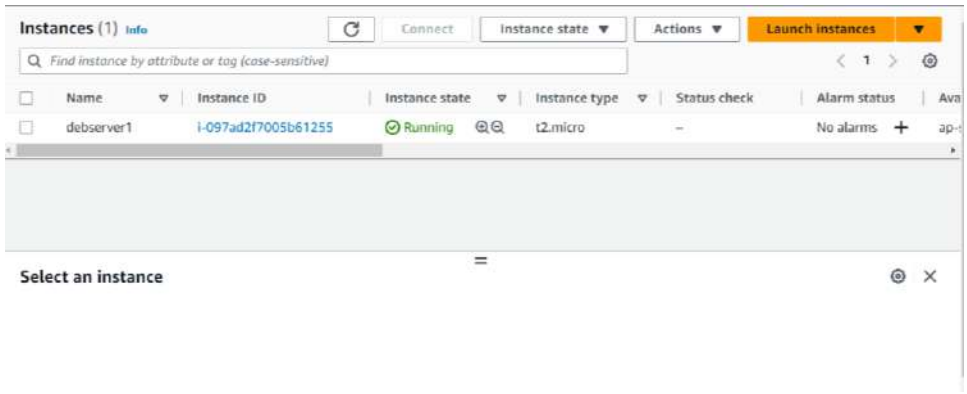








5. Now check whether your newly created instance is running or not in the instances page. Note it will take a few seconds to show the running status. (From Pending)

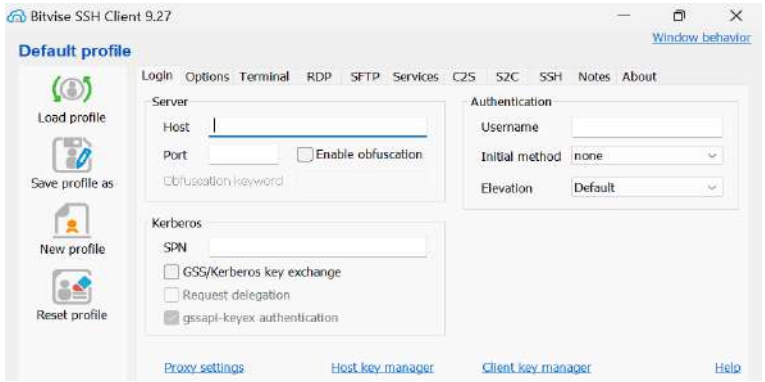


6. Now click on the Instance ID of the server.

7. Copy the Public IPv4 address.

8. Now for the next steps we require **Bitvise SSH client**. Download it and install in your local pc.

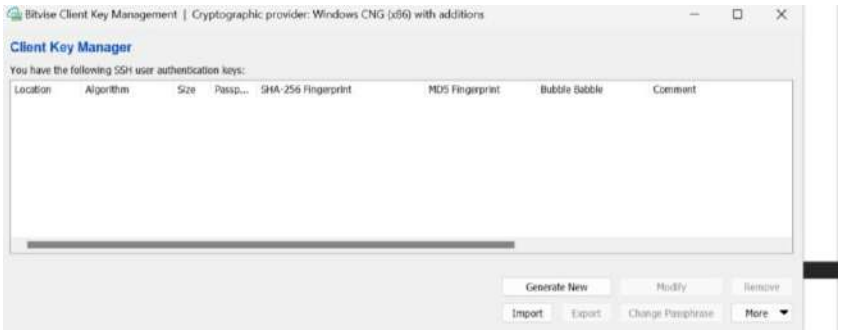
9. Now open the Bitvise SSH Client.



10. Paste the copied IPv4 address in the Host section.

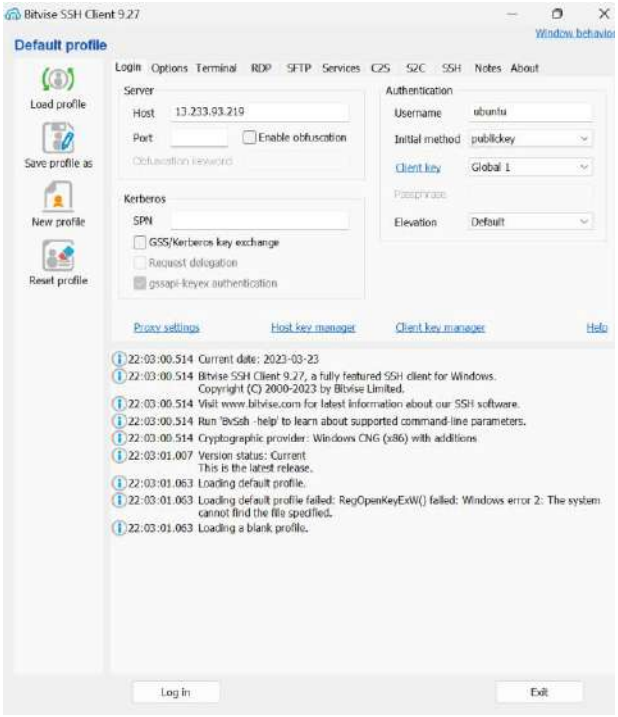
11. Set user name to ubuntu.

12. Click on the client key manager link below the authentication section.
It will open another pop-up window. There click on import button.
Select the previously downloaded .pem file. Click on import. Then close the Client key manager window.



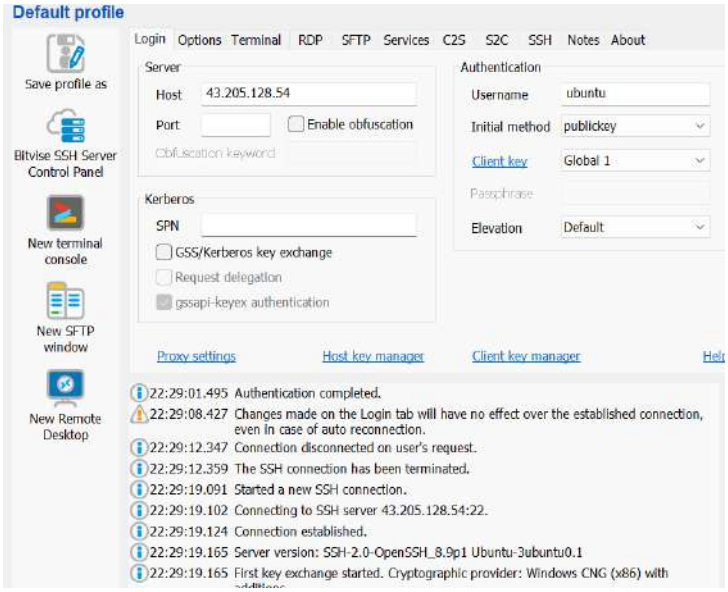
13. Now set initial method to public key.

14. Set Client Key to Global 1.



15. Now click on the Log In button at the bottom of the Window.
Click on Accept and Save button on the pop-up.
One of many ways in which you can know that whether you have successfully logged in is if your Log In button has changed to Log Out.

16. Now newly created options will arise on the left sidebar on successful login. Click on the new terminal console to open terminal of our server.



17. Enter the following commands:

a. **sudo apt-get update**

b. **sudo apt-get upgrade**

(Remember to press Y and then Enter when prompted)

(After the process is completed a new box/window appears. But just press Enter to continue.)

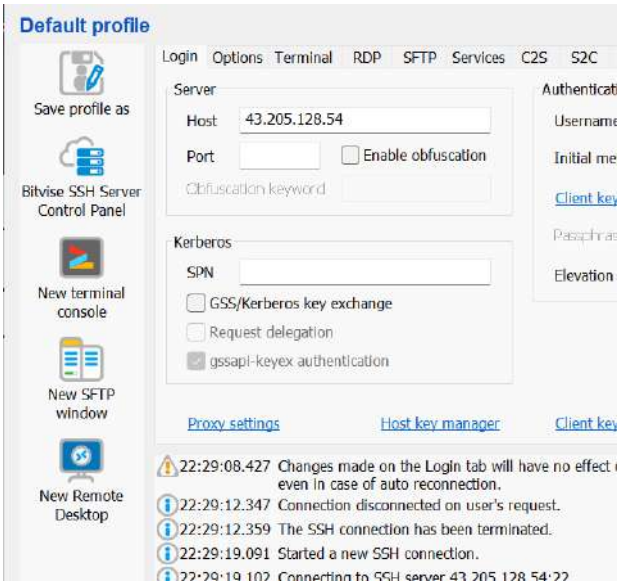
c. **sudo apt-get install nginx**

(Remember to press Y and then Enter when prompted)

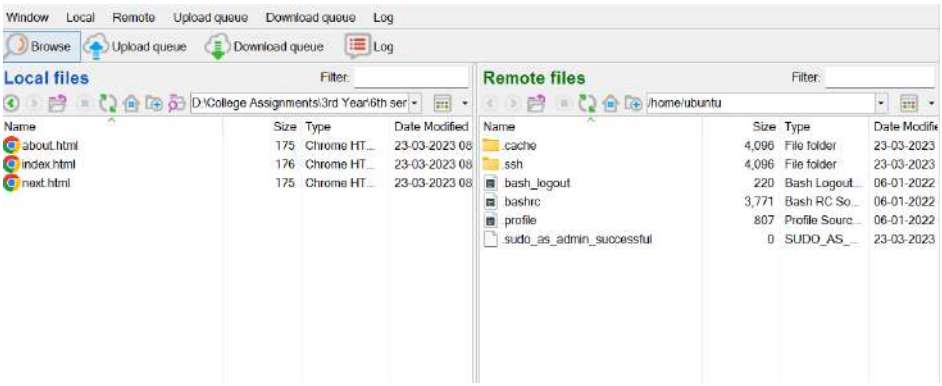
(After the process is completed a new box/window appears. But just press Enter to continue.)

18. Now minimize the console.

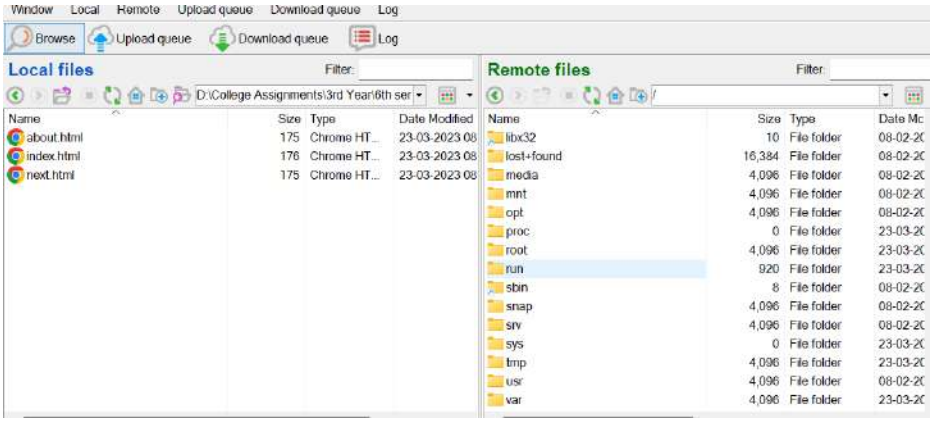
19. Click on the new SFTP window icon on the left sidebar.



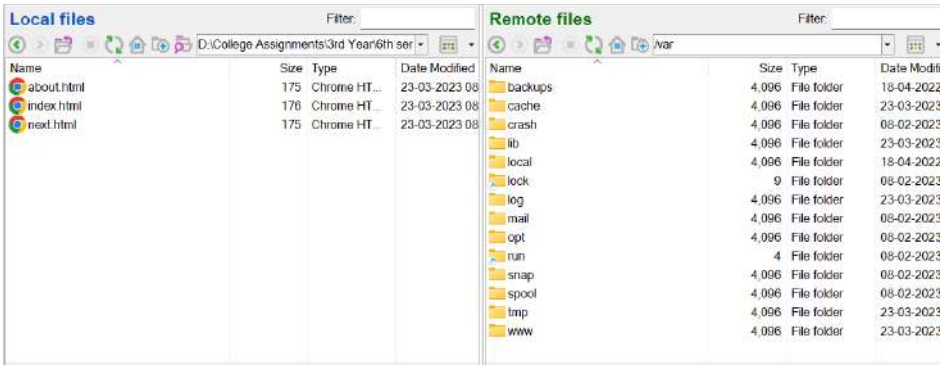
20. Select the folder where you have kept HTML files of your website on the local files section. Just keep it open.



21. Now click the Up button (2 times) on the Remote Files section. You will be able to see a bunch of folders. Scroll down and open the last folder named “var”.



22. Now again open the last folder in it named “www”.



23. Open the only folder named “html” and keep it open. You will see a default html already present.

You can check whether nginx is working by pasting our previously copied IPv4 address of our server instance in a different browser. It will show something like this.

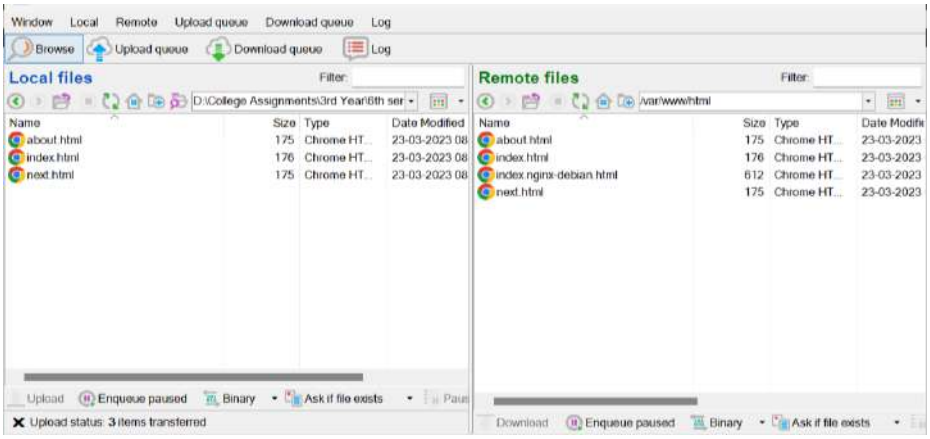
24. We actually need to transfer our local html file here in this open folder of the remote server. However, we do not have such permissions for this folder. To give such permission we need to go back to the terminal console and give the required permissions to the folder.

25. Now type the following commands in the terminal.

a. **cd /**
b. **cd var/www/**
c. **sudo chmod 777 html**

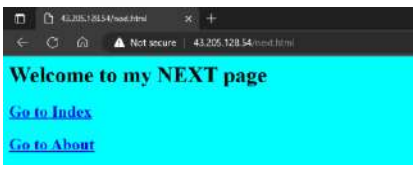
Now the permission (Read, Write, Execute) of the folder is successfully granted.

26. Now drag and drop all the files from local to remote.



Remember you must have the opening html named “index.html” in order to show the opening html page by the web server.

27. Finally open the website from any browser or device by using the public IPv4 address that you copied.



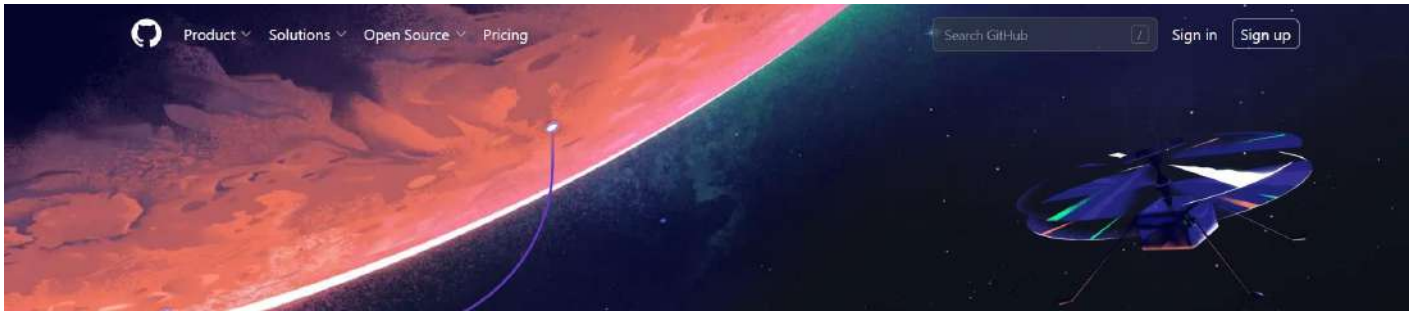
We now have successfully hosted a static website on an AWS EC2 sever.

ASSIGNMENT-8

Problem Statement: Deploy a project from Local Machine to GitHub and vice-versa.

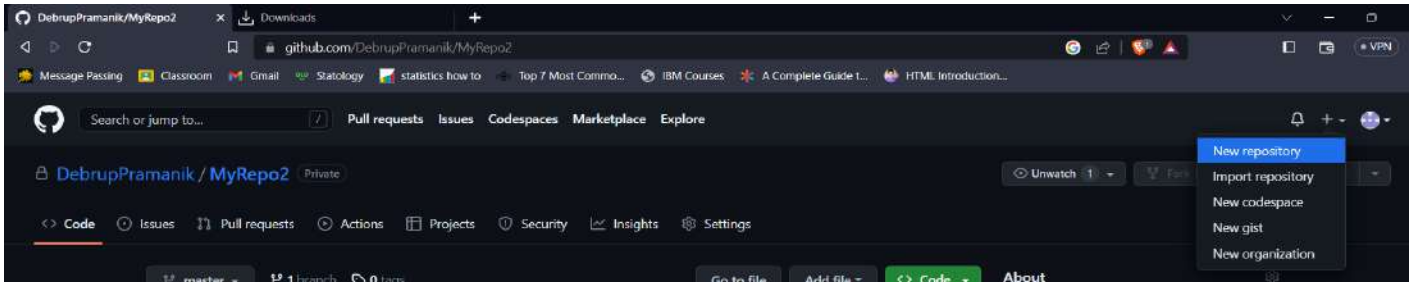
Procedure:

1. Make sure you have installed Git for Windows Application if you are using a Windows machine. If not then download the same from https://gitforwindows.org/ website. Install and make sure you have integrated it with the Windows Shell. (It is checked on by default so no need to change anything.)
2. Now go to GitHub Website https://github.com/ and Sign In to your account.
   If you do not have an account then click on Sign Up button to Create an account for yourself. Then just follow the on-screen instructions to complete your registration. Now finally you will Sign-In to your account.

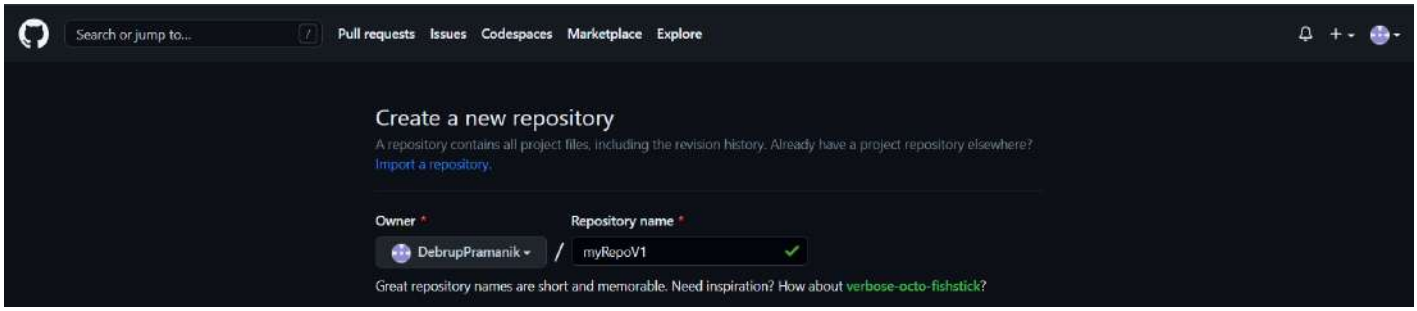


3. After successfully signing in GitHub, click on the + button situated on the top right corner of the website beside your profile picture.

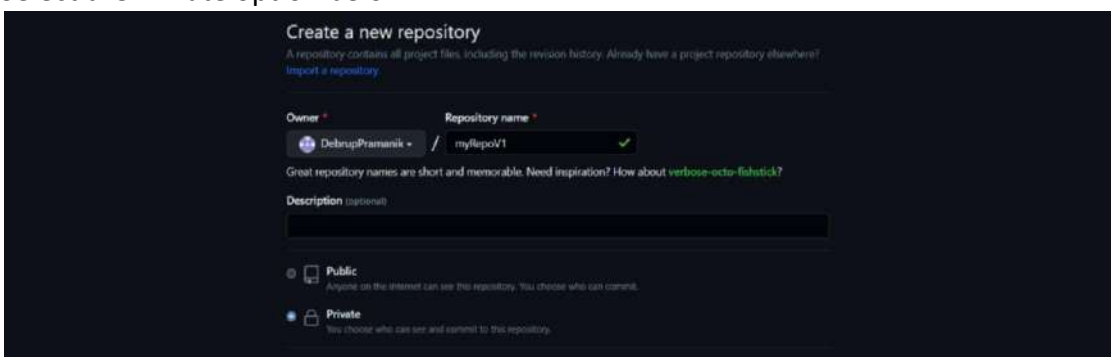
4. Now after clicking a menu will appear. Click on New Repository. This will create a New Repository where you can upload/deploy your project folders and files.



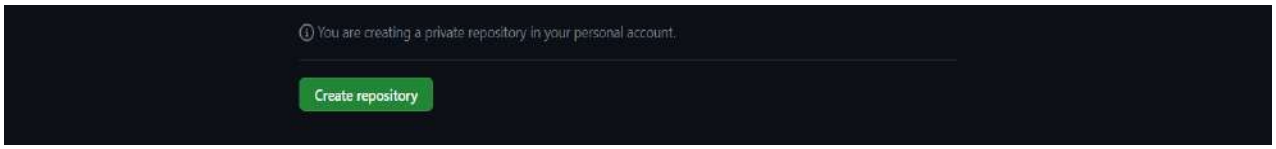
5. Now Enter the name of the Repository.



6. Select the Private option below.



7. Next scroll-down and click on Create Repository.

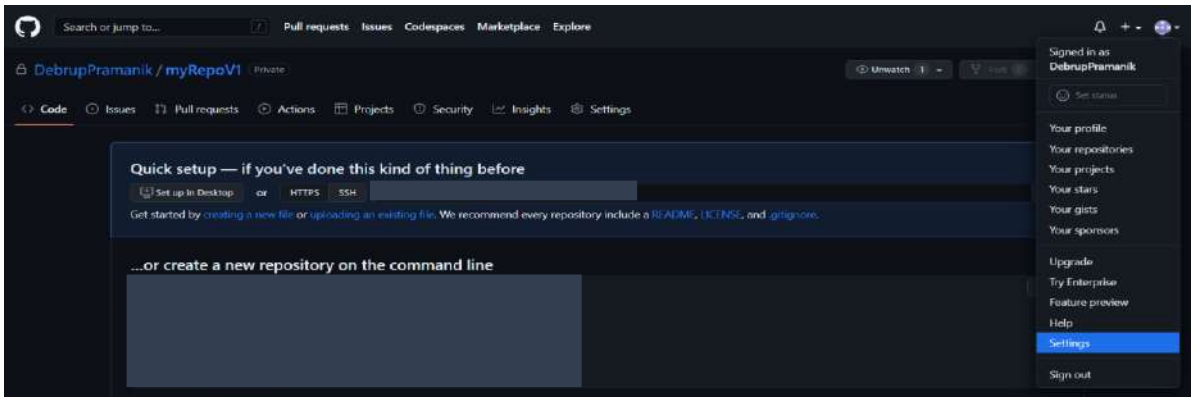


8. You will be redirected to the Repository code page.
Now while adding your project to your GitHub repository, you need to provide credentials of your account and sign-in every time you are doing this, through the Git bash terminal. Or there is another way, and that is by using Tokens generated for your account.

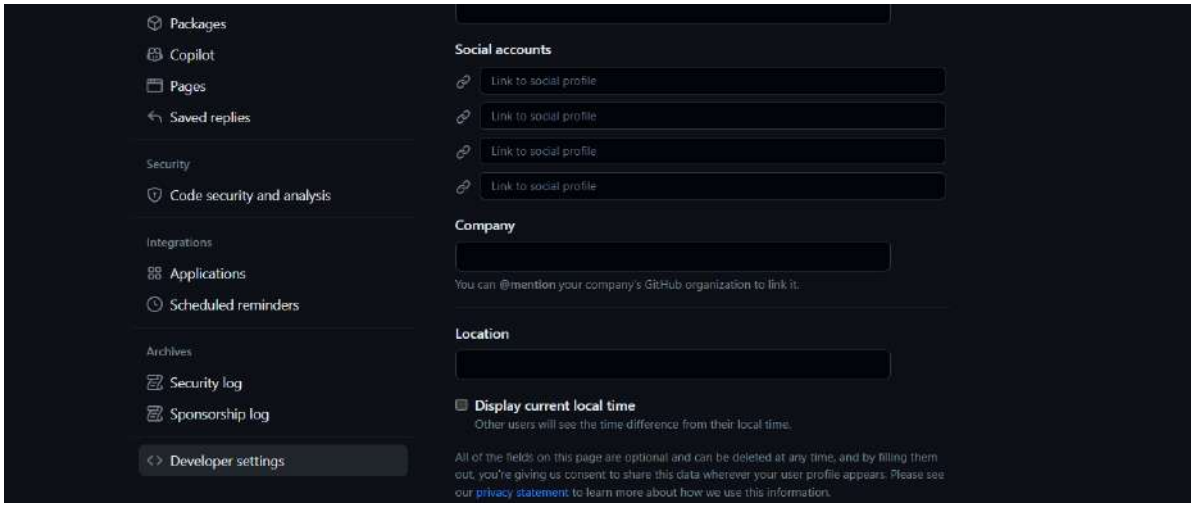
9. So, for generating token for your account follow the steps:

a. Click on your profile on the top right corner of the web page.

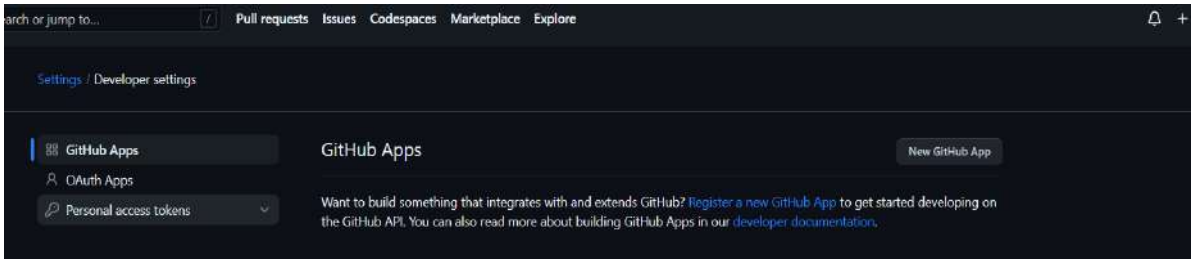
b. A drop-down menu will appear. Click on the Settings option.



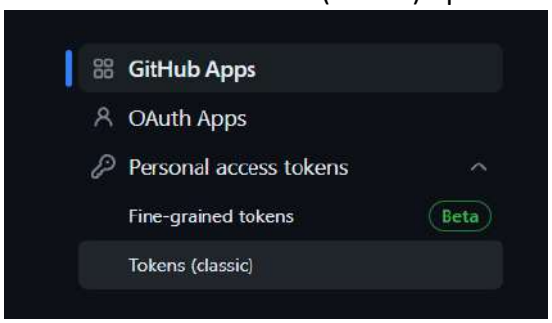
c. Now scroll down until you see a Developer Options button on the left Nav bar. Click on it.



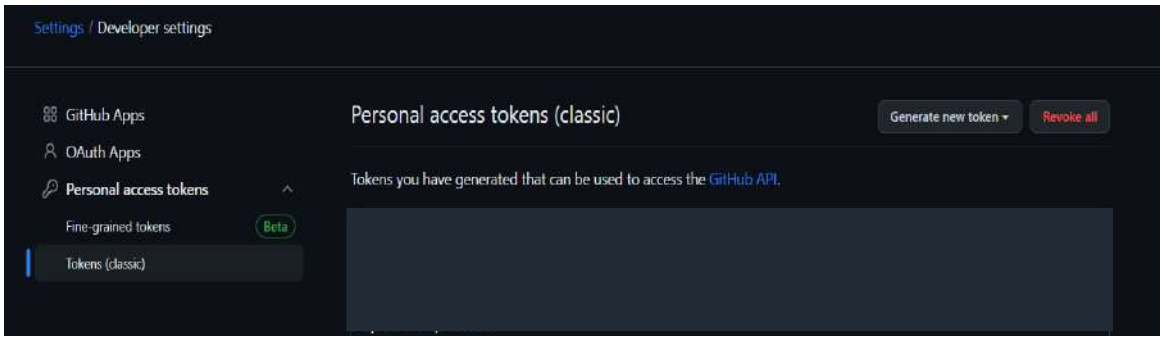
d. Now click on the Down arrow beside Personal Access Tokens in the left Nav bar.



e. Next click on the Tokens(classic) option.

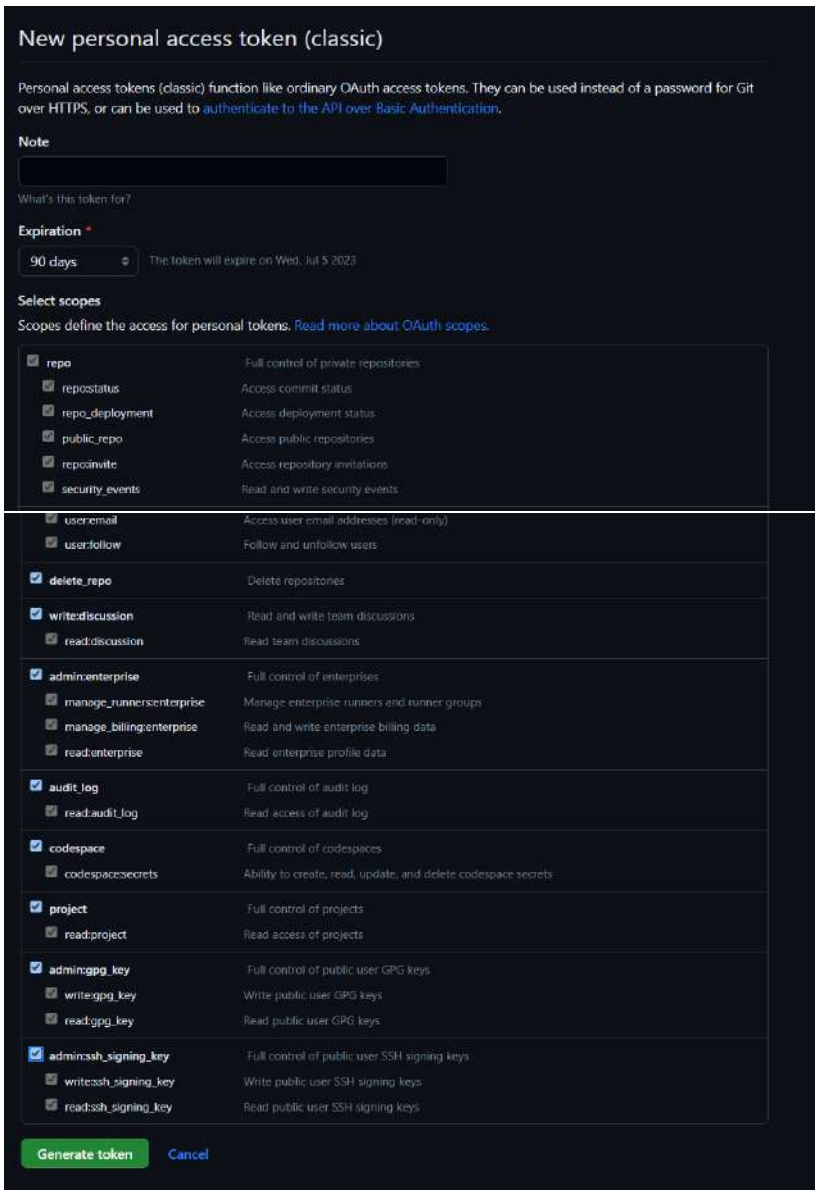


f. Next click on Generate New Token button



g. Again, click on Generate New Token(classic).

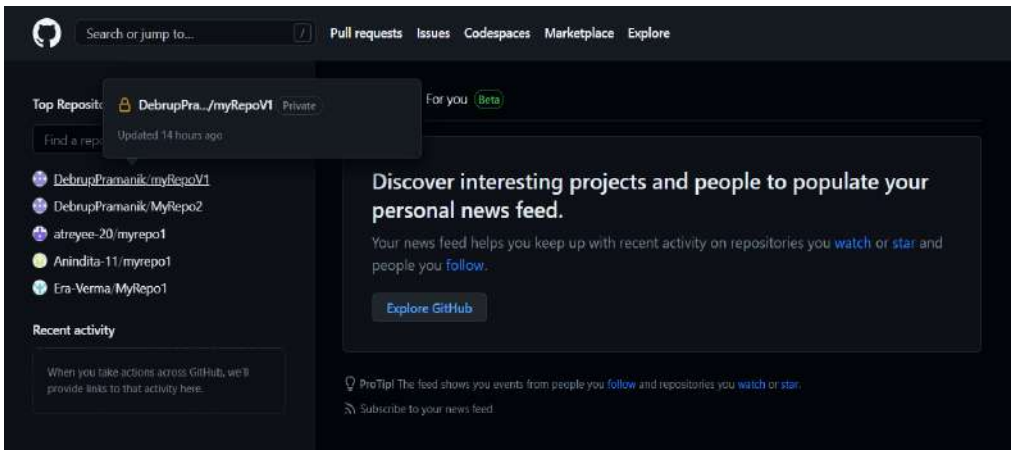
h. Give the Token Name.
Select the Expiration time (Preferably 90 Days).
Now check all the parent boxes below.
Then Click on Generate Token Button.



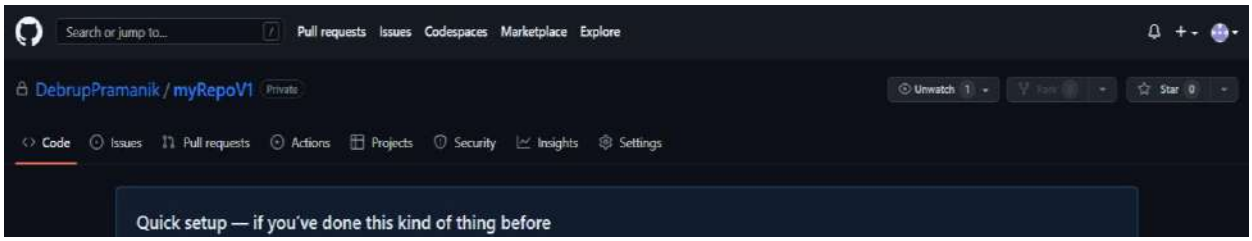
i. Now a Token will be generated. Copy it and save it in a text file. Keep it somewhere safe as we will need it later.

10. Now scroll-up and click on the icon on the top left corner of the web page. This will redirect you to your home page.

11. Now on the left nav bar you will find your Top Repositories You are currently working/contributing to. You can also locate your newly created Repository here. Click on it.

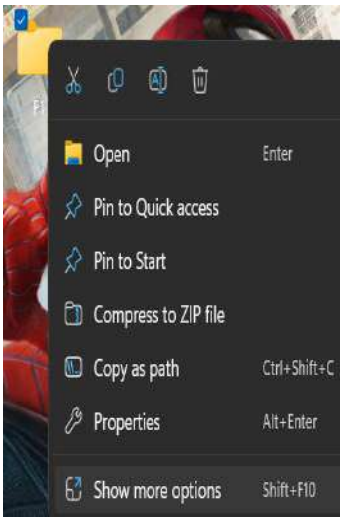


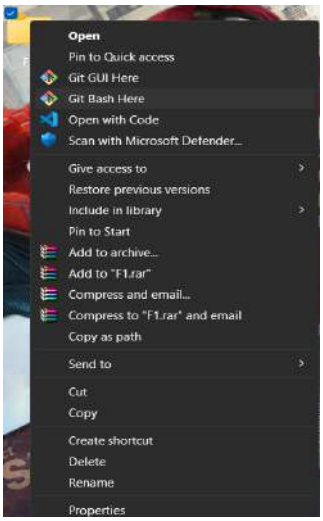
12. Go to the code section and keep it open. Now minimize your browser.



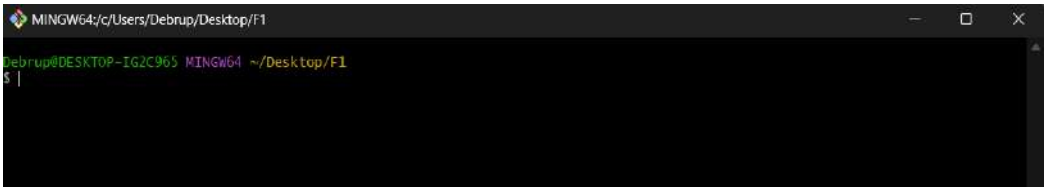
13. Now create a folder anywhere in your computer. Give it a name. Now Right Click on it and select Git Bash here.

If you are in Win 11 then after Right click go to Show more Options and then select Git Bash here.





14. It will open the Git Bash Terminal

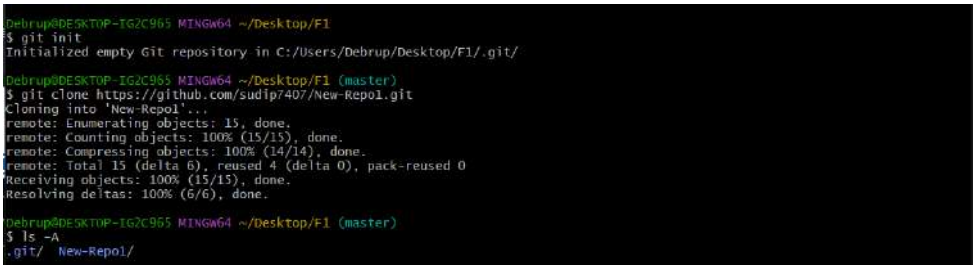


15. Now type the following to **clone** the project provided/required (Link will be provided or specified or can be obtained from visiting the required project's GitHub page and copying it):

➔ **git init**

➔ **git clone** https://github.com/sudip7407/New-Repo1.git

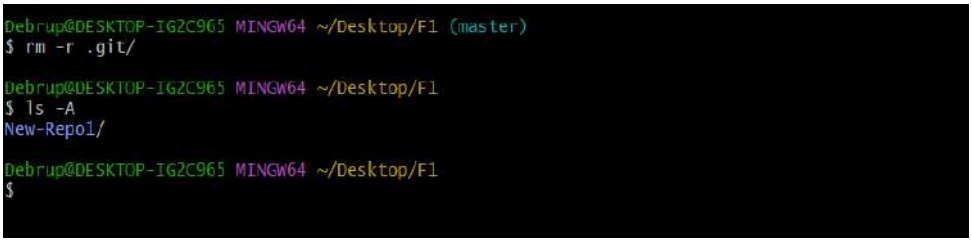
➔ **ls -A**



(We see all the files downloaded from the repository we just cloned. Now our job is to remove/delete the .git/ files from all the folders, including nested ones. Also, we have to remove. gitignore files from wherever it is present.)

➔ **rm -r .git/**

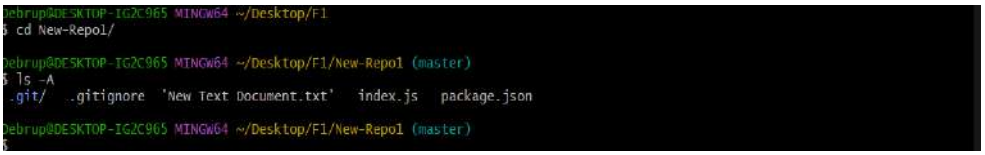
➔ **ls -A**



(We have successfully removed the .git/ file from the main directory. However we have to check the New-Repo1/ directory to see if it has no unwanted files.)

➔ **cd New-Repo1/**

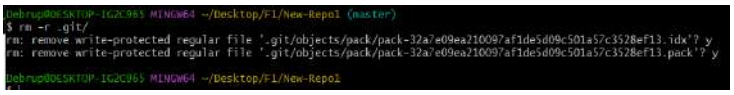
➔ **ls -A**



(We now see we have to remove the .git/ and .gitignore files. Then we need to go back to the main directory)

(Sometimes permissions are asked when removing some files. Just type y when prompted and press enter and repeat it when asked every time.)

➔ **rm -r .git/**



➔ **rm -r .gitignore**

➔ **ls -A**

(Our work here is done. Now we need to return to our main directory)

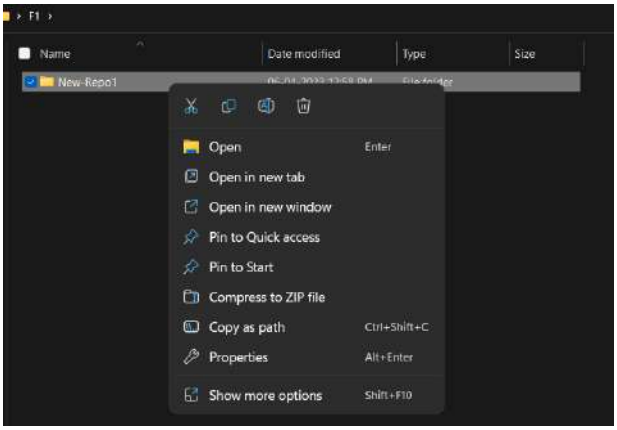
➔ **cd ..**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1
$ cd ..

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1
$ ls -A
New-Repo1/
```

We now have successfully cloned our required project.

16. Now close the terminal and open the folder in which you just cloned the project.

17. Single Click on the New-Repo1 file and right click and just like explained above again select Git bash here option.



18. Again, a Git bash terminal will open.

19. Again, type the following commands but now we will now upload this cloned project to our created repository on our GitHub:

➔ **git init**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1
$ git init
Initialized empty Git repository in C:/Users/Debrup/Desktop/F1/New-Repo1/.git/

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$
```

➔ **ls -A**

➔ **git status**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1
$ git init
Initialized empty Git repository in C:/Users/Debrup/Desktop/F1/New-Repo1/.git/

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ ls -A
 .git/  'New Text Document.txt'   index.js   package.json

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        New Text Document.txt
        index.js
        package.json

nothing added to commit but untracked files present (use "git add" to track)
```

➔ **git config --global user.email "Your email here"**

➔ **git config --global user.name "Your GitHub account username here"**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git config --global user.email "debrup202002@gmail.com"

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git config --global user.name "DebrupPramanik"
```

➔ **git config user.name**

➔ **git config user.email**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git config user.name
DebrupPramanik

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git config user.email
debrup202002@gmail.com

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
```

(this is required to set as it will commit the files in your repository using the set username and email.)

➔ **git add .**

➔ **git status**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git add .
warning: in the working copy of 'cls', LF will be replaced by CRLF the next time
 Git touches it

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   New Text Document.txt
        new file:   cls
        new file:   index.js
        new file:   package.json


Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$
```

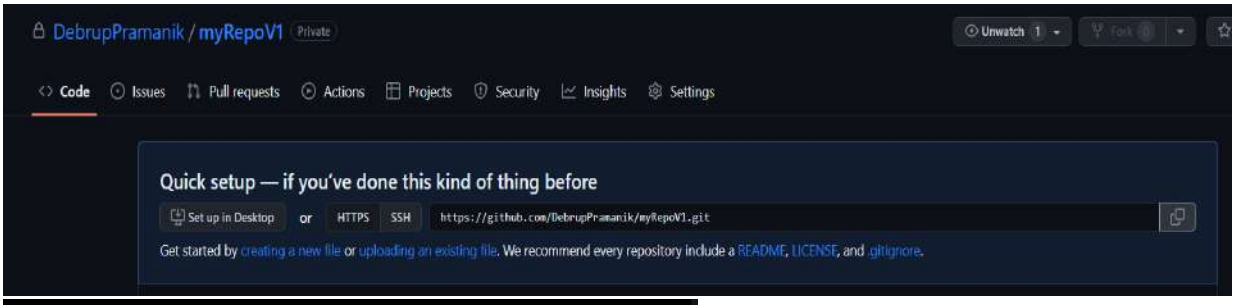
(Notice all the files that was colored red previously is now green, indicating that they have been added to your local repository. We now have to commit the changes we made and then push it to our GitHub repository)

➔ **git commit -m "type your own message here"**

```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git commit -m "Hello World. My first ever commit"
[master (root-commit) 36046dc] Hello World. My first ever commit
 4 files changed, 49 insertions(+)
 create mode 100644 New Text Document.txt
 create mode 100644 cls
 create mode 100644 index.js
 create mode 100644 package.json

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$
```

➔ **git remote add origin https://github.com/yourusername/Yourrepositoryname.git**

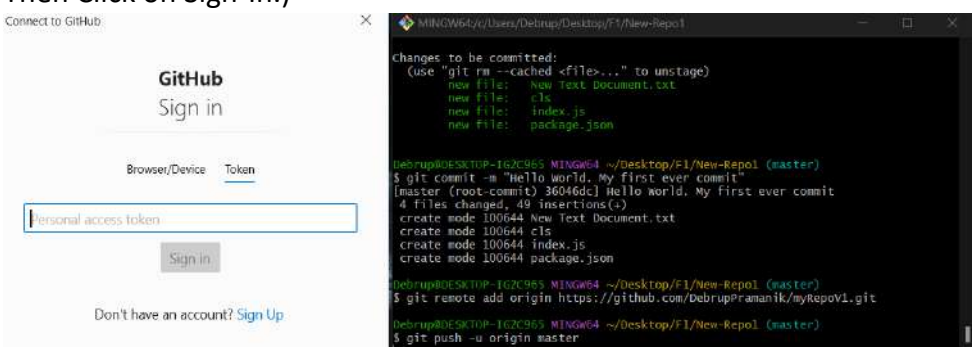
(The https address is the address of your repository. To obtain it maximize your browser where your GitHub repository is open and there copy the https address)




```
Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git commit -m "Hello World. My first ever commit"
[master (root-commit) 36046dc] Hello World. My first ever commit
 4 files changed, 49 insertions(+)
 create mode 100644 New Text Document.txt
 create mode 100644 cls
 create mode 100644 index.js
 create mode 100644 package.json

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git remote add origin https://github.com/DebrupPramanik/myRepoV1.git

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$
```

➔ **git push -u origin master**

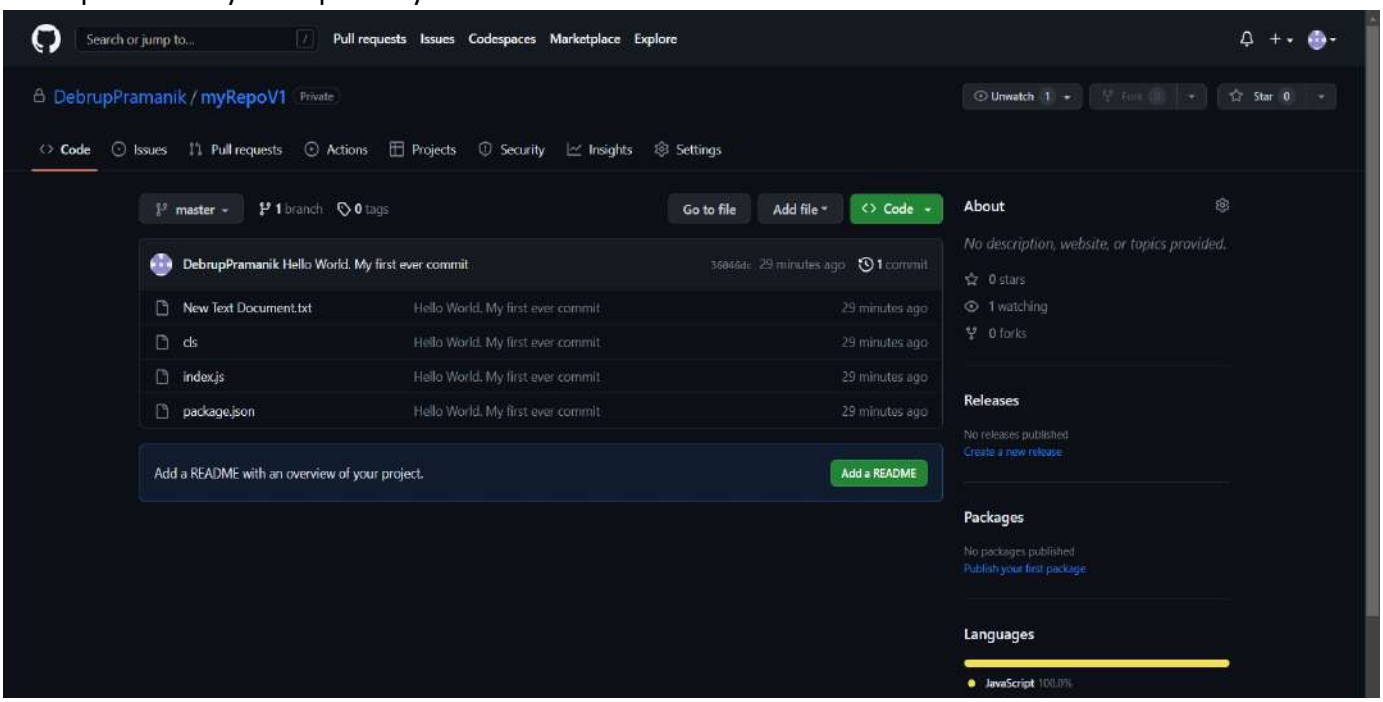
(This is the final command. A pop-up window will open named Connect to GitHub. You will have several options to Sign-In. However, we will be using our Generated token for your account to Sign-In to our account. Click on Token option beside Browser/Device. Paste your token in the placeholder. Then Click on Sign-In.)



(After Successful sign-in the following will show up in the terminal. The Pop-up Window will close automatically)

```
[master (root-commit) 36046dc] Hello World. My first ever commit
 4 files changed, 49 insertions(+)
 create mode 100644 New Text Document.txt
 create mode 100644 cls
 create mode 100644 index.js
 create mode 100644 package.json

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git remote add origin https://github.com/DebrupPramanik/myRepoV1.git

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 6 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.05 KiB | 1.05 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DebrupPramanik/myRepoV1.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Debrup@DESKTOP-IG2C965 MINGW64 ~/Desktop/F1/New-Repo1 (master)
$
```

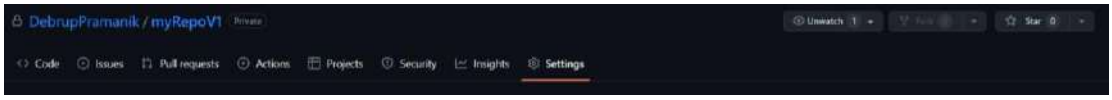
20. Now go to your browser where your GitHub repository is open. Refresh the page. Now you will see the files uploaded in your repository!



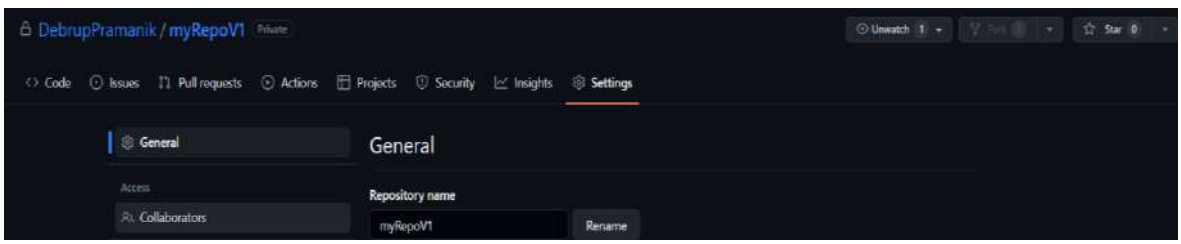
We have successfully Cloned and Uploaded our project to GitHub using Git and Git Bash terminal.

21. You can also add collaborators who can access your private repository and can contribute to it.

a. Go to the Setting section of your repository (It is in the same nav bar as the code section)



b. Then select the collaborators option in the left nav bar.
It will ask you to enter your password.
Enter it.



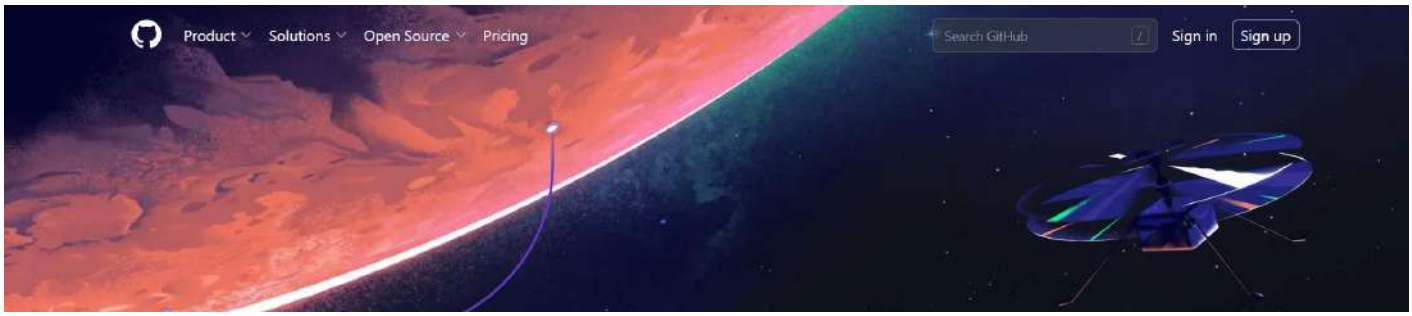
c. Now you can click on add people button to add others as collaborators of your repository. You have to search by their Username or Email.

ASSIGNMENT-9

Problem Statement: Deploy a project from GitHub to EC2.

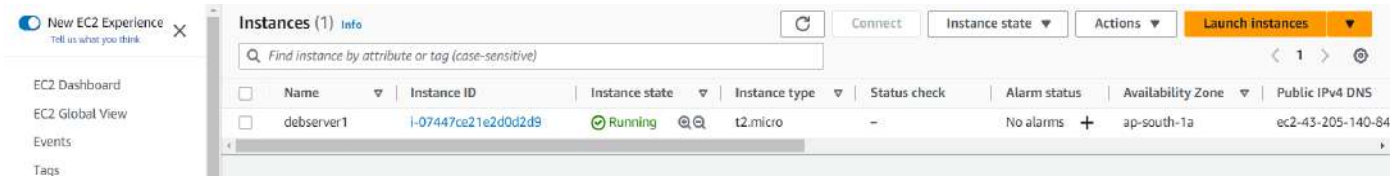
Procedure:

1. Go to GitHub Website https://github.com/ and Sign In to your account.

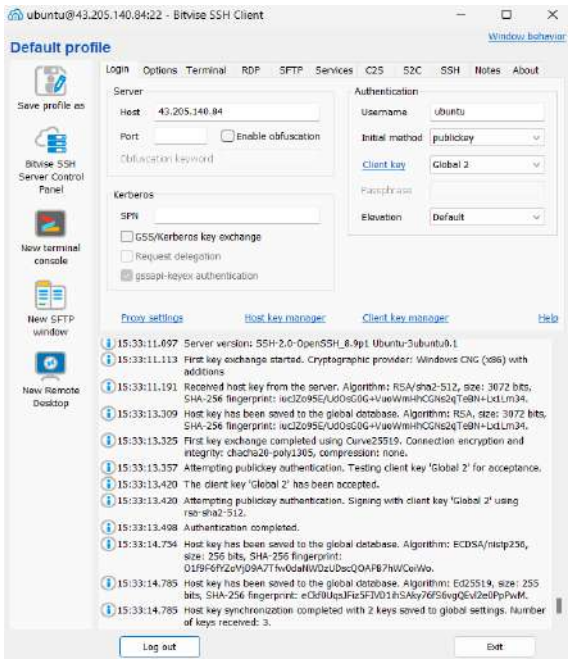


2. Also, Sign-In to your AWS account.

3. Create an EC2 instance (Refer to Ass7)



4. Connect the to the instance using the Bitvise SSH Client (Refer to Ass7)



5. Now Click on New Terminal Console option in the Left Sidebar of the Bitvise Client.

6. A terminal window will open and in it type the following commands:-

- ➔ **sudo apt-get update && sudo apt-get upgrade**
  (After few steps of progression, in case of any prompts asking (Y/N) press 'y' button and then press 'Enter' to continue. At the last stages if a UI appears on the screen, just press 'Enter' to continue. After the whole process is complete enter the next command as mentioned below)
- ➔ **sudo apt-get install nginx**
  (After few steps of progression, in case of any prompts asking (Y/N) press 'y' button and then press 'Enter' to continue. At the last stages if a UI appears on the screen, just press 'Enter' to continue. After the whole process is complete enter the next command as mentioned below)
- ➔ **nginx -v**

```
ubuntu@ip-172-31-38-127:~$ nginx -v
nginx version: nginx/1.18.0 (Ubuntu)
```

(This command displays the nginx version installed in the server system)

➔ **curl -Sl https://deb.nodesource.com/setup_18.x | sudo -E bash -**

(This command downloads NodeJS files with all dependencies in our server system)

➔ **sudo apt install nodejs**

(Press 'Enter' to continue when any UI appears on screen)

(This command installs NodeJS in our server system)

➔ **node -v**

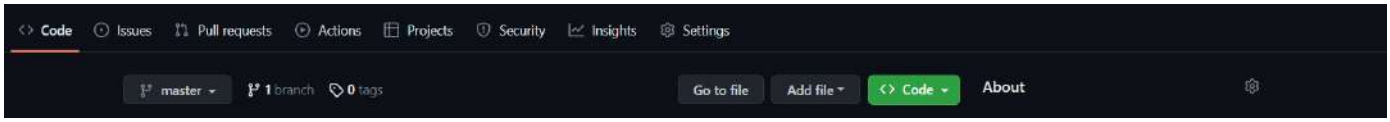
```
ubuntu@ip-172-31-38-127:~$ node -v
v18.15.0
```

(This command displays the version of NodeJS installed in our server system)

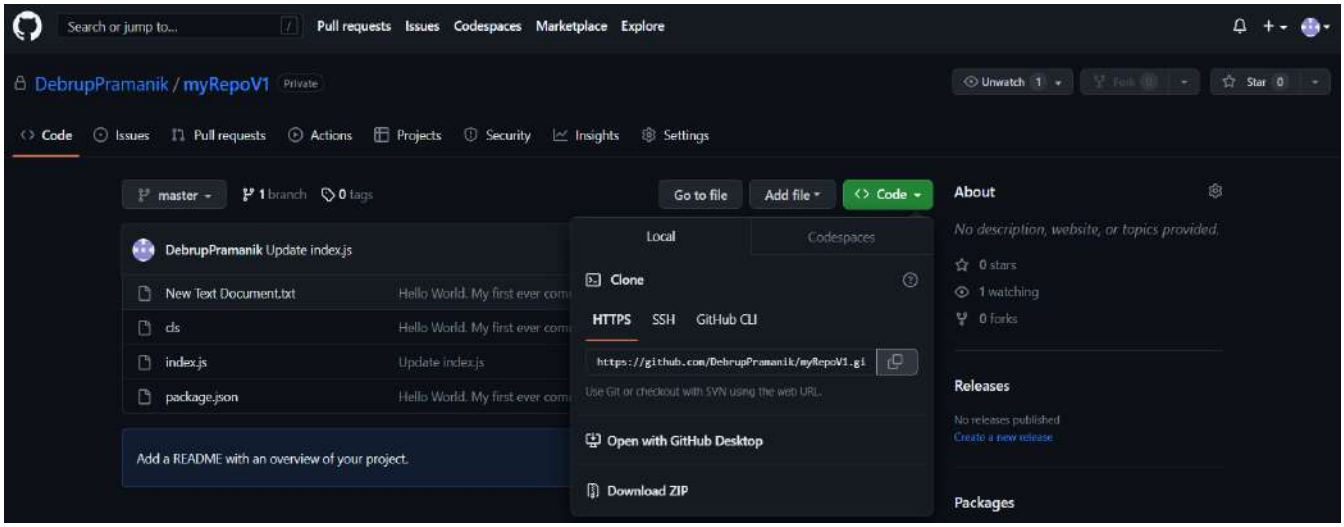
Now, minimize the terminal window. Go to the browser where our GitHub is Logged-In.

7. Go to your GitHub Repository which you want to upload in your EC2 server.

8. Click on the code button (in green color).



9. Now copy the HTTPS address of your Repository



10. Now return to the minimized terminal window and enter the following commands:-

➔ **git clone https-address-you-just-copied-in-step-10**

```
ubuntu@ip-172-31-38-127:~$ git clone https://github.com/DebrupPramanik/myRepoV1.git
Cloning into 'myRepoV1'...
Username for 'https://github.com':
```

(Remember to paste your own https address that you copied in the above command in place of the one given in the screenshot)

(As shown in the screenshot, you will be asked to enter your username for GitHub. So mention your username there.

After that you will be requested to provide your password. However, you have to enter your Account Token you generated instead of your password. If you don't have a Account Token then refer to Ass7 and create one for your GitHub account. Now copy-paste the Account Token (from the text file you haved saved it) where it wants to mention your password. For pasting just Right click for a single time on the terminal where you want to paste)

(Note you won't be able to see your pasted token on the terminal as it is hidden by default. So just press 'Enter' to continue)

```
ubuntu@ip-172-31-38-127:~$ git clone https://github.com/DebrupPramanik/myRepoV1.git
Cloning into 'myRepoV1'...
Username for 'https://github.com': DebrupPramanik
Password for 'https://DebrupPramanik@github.com':
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 2), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (2/2), done.
```

➔ **dir**

```
ubuntu@ip-172-31-38-127:~$ dir
myRepoV1
```

(As seen this is the name of our cloned repository. This means a new directory has been created in our present working directory which has been named automatically to match the name of our Repository.)

➔ **cd myRepoV1/**

```
ubuntu@ip-172-31-38-127:~$ dir
myRepoV1
ubuntu@ip-172-31-38-127:~$ cd myRepoV1/
ubuntu@ip-172-31-38-127:~/myRepoV1$
```

(Now we enter into the directory)

➔ **ls -A**

```
ubuntu@ip-172-31-38-127:~/myRepoV1$ ls -A
 .git  'New Text Document.txt'   cls   index.js   package.json
ubuntu@ip-172-31-38-127:~/myRepoV1$
```

(This command displays all the files in the current directory)

(We observe that we have all the files that we have in our Repository has been cloned in our directory in the server system)

➔ **npm install**

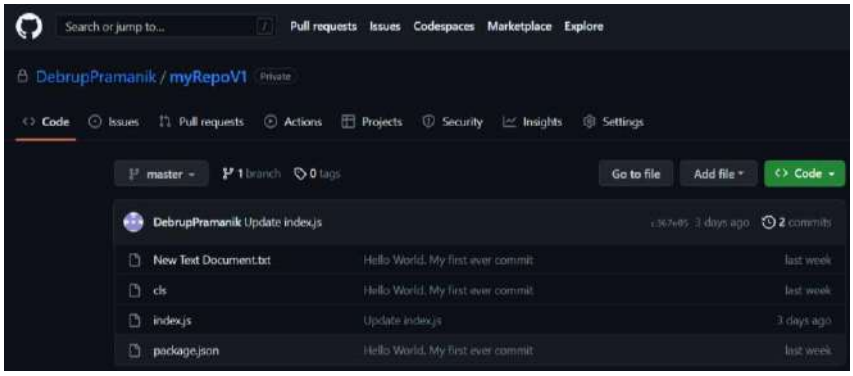
```
ubuntu@ip-172-31-38-127:~/myRepoV1$ npm install
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math
.random() in certain circumstances, which is known to be problematic.  See https://v8.dev/blog/math-
random for details.

added 258 packages, and audited 259 packages in 10s

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

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 9.5.0 -> 9.6.4
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.4
npm notice Run npm install -g npm@9.6.4 to update!
npm notice
ubuntu@ip-172-31-38-127:~/myRepoV1$
```

(This command installs the npm package manager)

Now before proceeding further we need to return back to GitHub. Minimize the terminal for now.

11. Go back to your Repository in Github.

12. Open your “index.js” file



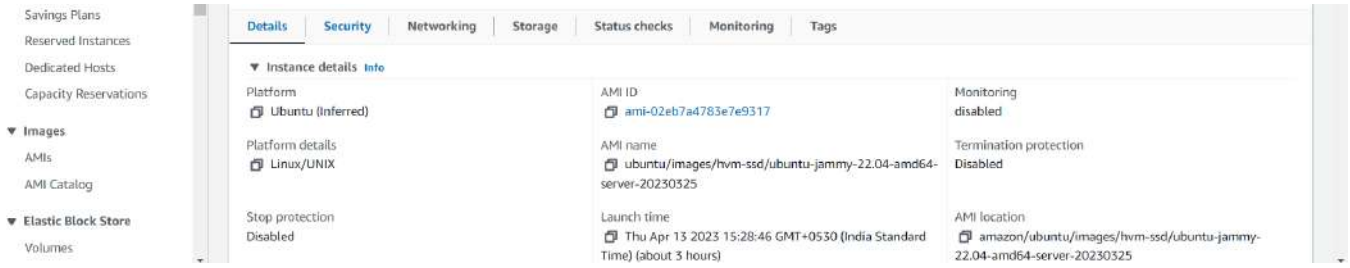
13. Check the port no. specified in the program. It is specified in the app.listen() method as the first parameter. Here it is ‘4000’. Copy or remember this no. as it is the port no. and will be required to connect to our website.

We have to add this port no. to our EC2 instance security group rule otherwise we won’t be able to access the website from anywhere.

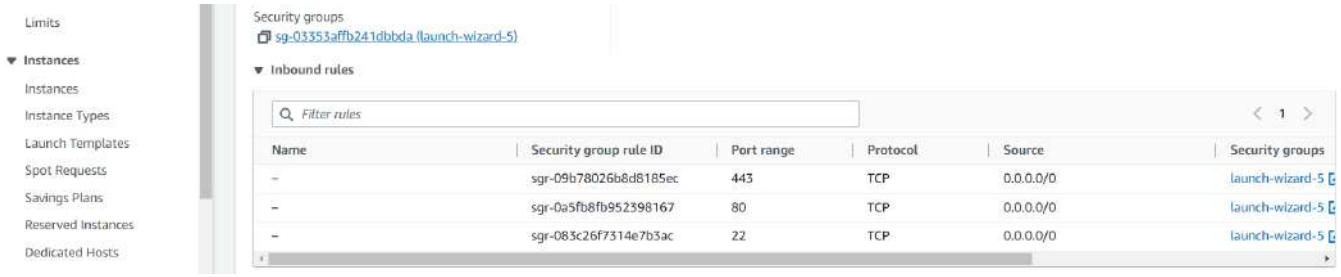
14. Now go back to your AWS EC2 instances page.

15. Click on the instance that is being used.

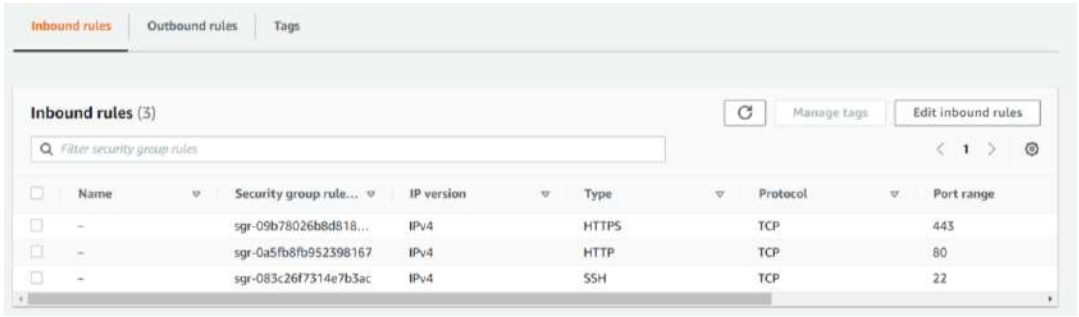
16. Scroll down until you find a section bar where by default the details option is selected. Select the Security option. It is beside the Details option (in blue).



17. Then click on the security groups link under security groups



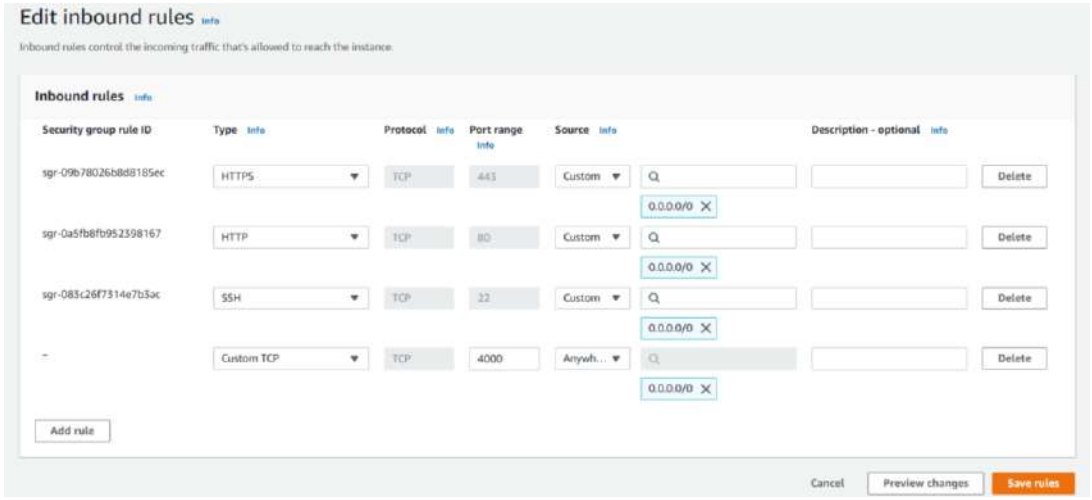
18. Then click on Edit Inbound Rules button.



19. Click on the Add Rule button.

20. A new Row will be generated. Let the type remain Custom TCP. Under Port Range write your Port no. you want to open. In this case we have 4000 port no. as we found out earlier in our index.js code. Next in source click on the search box and the first option with value 0.0.0.0/0 should be selected.

21. Now click on the save rules button.



We have successfully added the Port No. to our Inbound rules. Now we can access the our website. But first we need to start our server.

22. We return back to the terminal and type:-

➔ **node index.js**

```
ubuntu@ip-172-31-38-127:~/myRepoV1$ node index.js
Started server
```

Our server has started and it is also reflected by the terminal prompt. Now to check we need to open another browser and type in the IPv4 address of our EC2 server to access our website.

23. Now copy the IPv4 address of your EC2 server and paste it in another browser. But before pressing Enter add a colon (:) and then mention the Port No. mentioned in the index.js file. For our case it is 4000. In our case the full address resulted in the one in the below screenshot.

```
43.205.140.84:4000
```

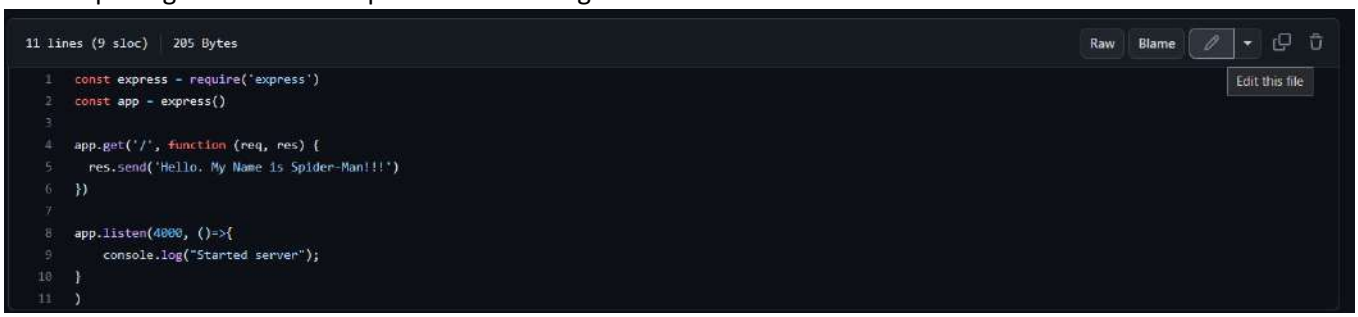
24. Now press Enter to load the website.

25. We have successfully deployed our project from GitHub to our EC2 server.

Now if we want to change something in our project or file or code we will follow these general steps:

26. Suppose we want to modify the displayed message. So open the index.js file in GitHub.

27. After opening we click on the pen icon on the right corner side of the code viewer.



28. We then modiify the string passed through the res.send() method.

29. Now after editing, scroll-down and click the Commit changes button.

30. Our changes have finally been commited in our Repository in GitHub.
However our changes have not been reflected in our Remote Server. For that we have to 'pull' the new updated files into the repository directory in our Remote Server.

31. We have to Close our already running Server. For this, go to the Terminal which we have been working with. Then press <CTRL + C> shortcut to close the server.

```
Started server
^C
ubuntu@ip-172-31-38-127:~/myRepoV1$
```

Our server has been stopped.

32. Now type:-

➔ **git pull**

(Enter the username when asked)
(Enter your account Token as your Password when asked for password)
(Right click once to paste, then press Enter)

```
ubuntu@ip-172-31-38-127:~/myRepoV1$ git pull
Username for 'https://github.com': DebrupPramanik
Password for 'https://DebrupPramanik@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 685 bytes | 228.00 KiB/s, done.
From https://github.com/DebrupPramanik/myRepoV1
   c367e05..188da63  master     -> origin/master
Updating c367e05..188da63
Fast-forward
 index.js | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
ubuntu@ip-172-31-38-127:~/myRepoV1$
```

Now we have to restart the server.

➔ **node index.js**

(We restarted the server)

33. We now have to Refresh our browser where we have our website open.

The changes have been successfully reflected. This is how we have to edit and update our project if required.

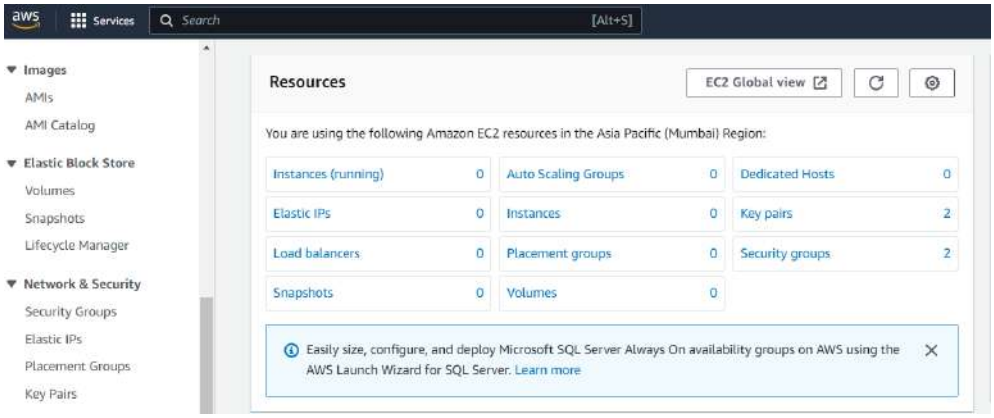
We have successfully completed our task of Deploying our project from GitHub to our EC2 server.

ASSIGNMENT – 10

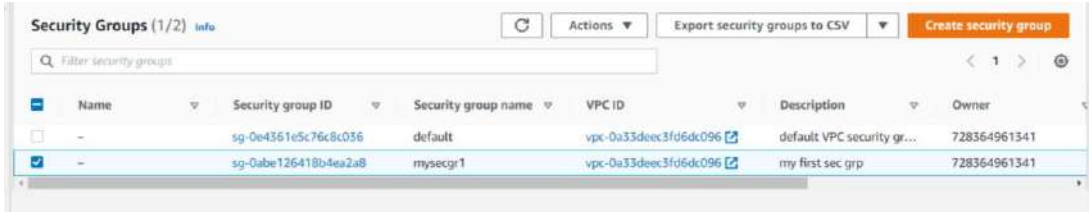
Problem Statement: Deploy project from GitHub to EC2 by creating new security group and user data.

Procedure:

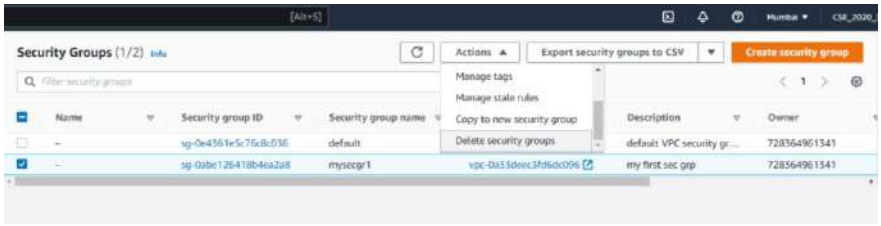
1. Sign in to your AWS account.
2. Go to your EC2 dashboard
3. Scroll down and Click on Security Groups option on the left side nav bar under Network & Security option.



4. Select all the Security Groups other than the one named “default”.



5. Then Click on the Actions button.

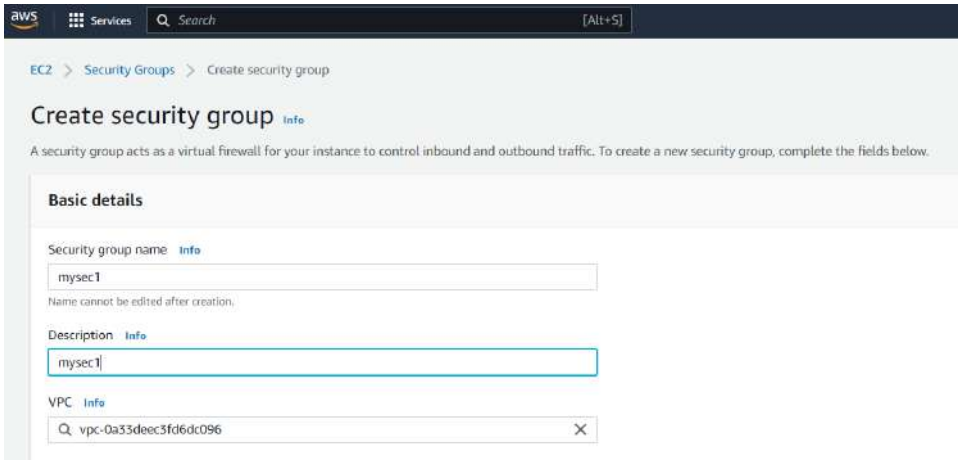


6. Scroll-Down the dropdown list until you find the “delete all security groups” option. Click on it.

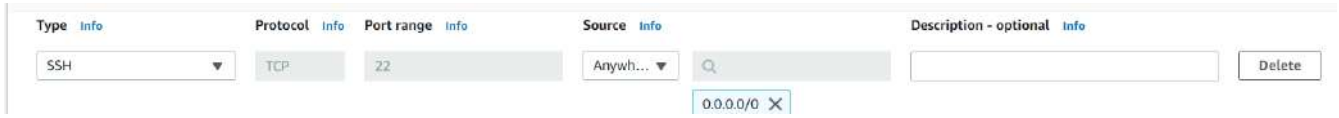
7. Now only the “default” security group remains and we keep it that way.

8. Now click on the “Create Security Group” button.

9. Now start by giving a name to the security group and giving its description (anything).
   Let the VPC remain unchanged.

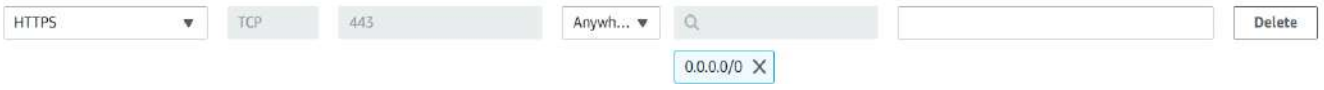


10. Next, we will add Inbound Rules. Start adding by clicking the Add rule button. These include:
    a) SSH

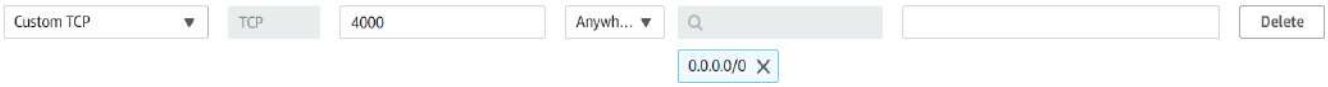


    b) HTTP

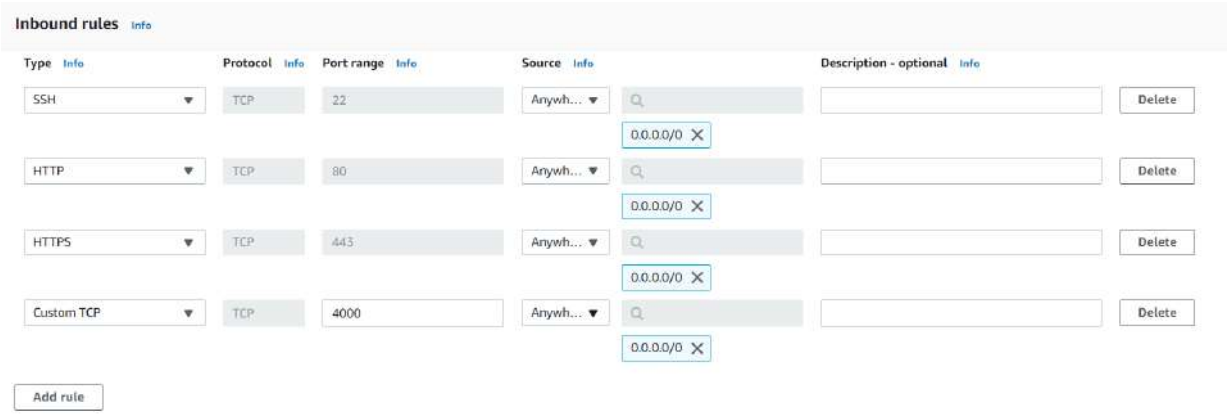
    c) HTTPS



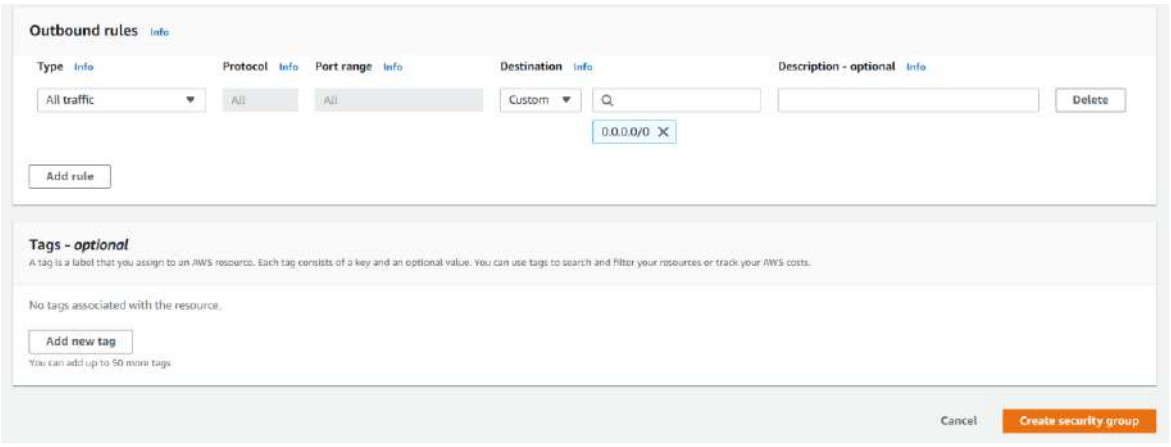
    d) Custom TCP



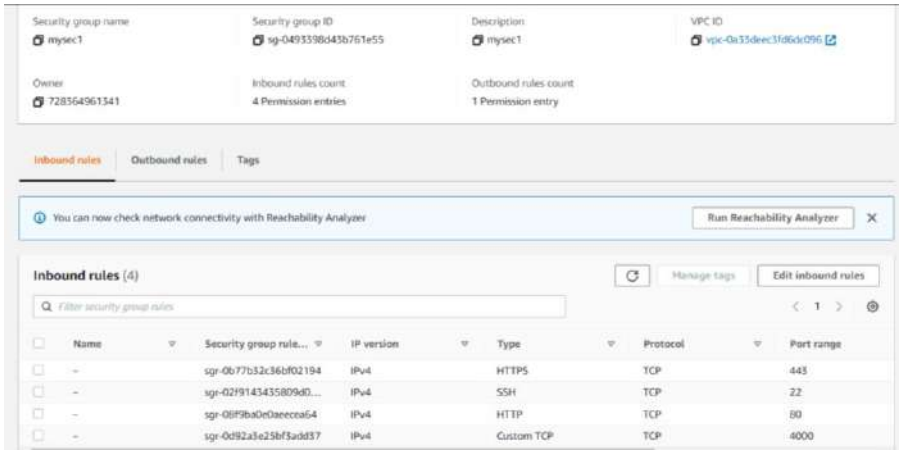
    The last one with custom TCP has a specific port range that we require to connect to our project. It has been specified in our index.js file (refer Ass9).
    Now the final Inbound Rules section should look like this.



11. Next outbound rules and all other sections remain unchanged. Now Click on the create security group button.



12. Now go back to the security groups list and click on the security group ID of the newly created Security Group.



After clicking we can view the inbound rules that we added during its creation.

13. Now we go to the instances section from the left side nav bar.
14. Now we Create a new EC2 instance. Click on the Launch Instance button.

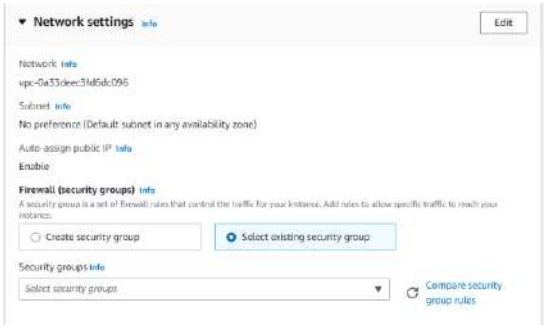
Now,

a) Give the name

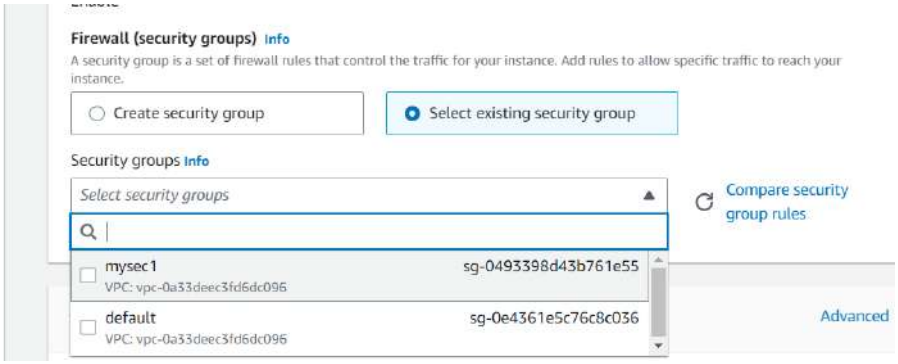
b) Select Ubuntu as OS.

c) Select a keypair or generate a new one if none is available.

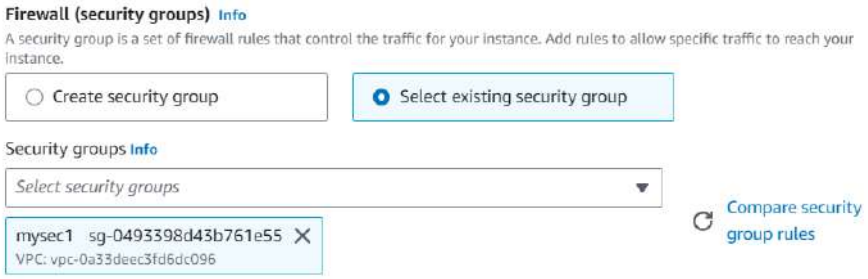
d) Then under Network settings select the Select Existing Security Group option.



e) Now under the security groups dropdown menu select the one we just created.



It should look like this…..



f) Now scroll down and click on the Advanced Details option.

g) Now again scroll-down to the newly appeared sub-sections until you find User Data section.

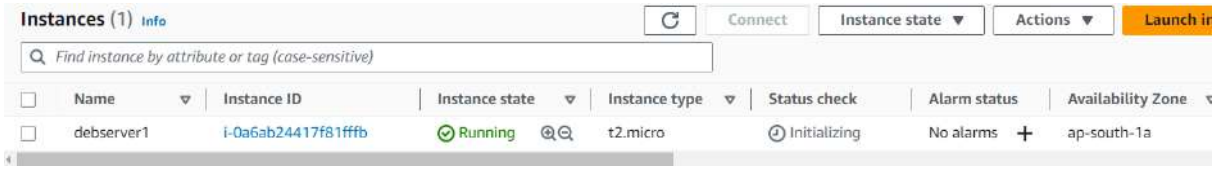
h) Write the following commands in the given box. Remember this user data is given to execute the given commands once the server starts. So essentially, we can provide all commands that we entered in our Assignment 9 previously and execute them without connecting to our server itself!! They will be executed sequentially.

```
#!/bin/bash
apt-get update
apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt-get install -y nodejs
```

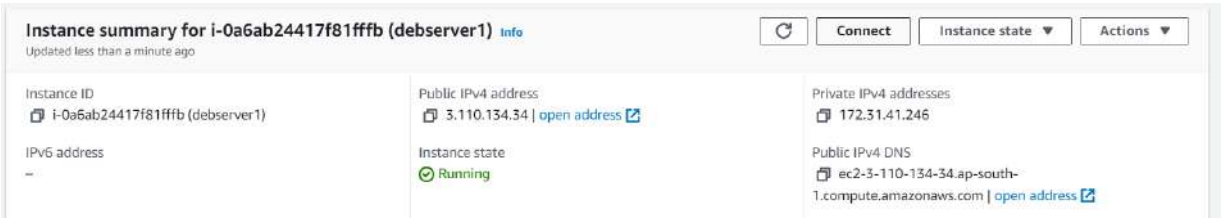
Now, here is a caveat. We have created a private repository in GitHub. So, whenever we run the git clone command it asks for our username and password. Hence this cannot be executed directly through our User Data instructions. We have to connect manually and enter all commands starting from the git clone command.

i) Now we click on the launch instance button.

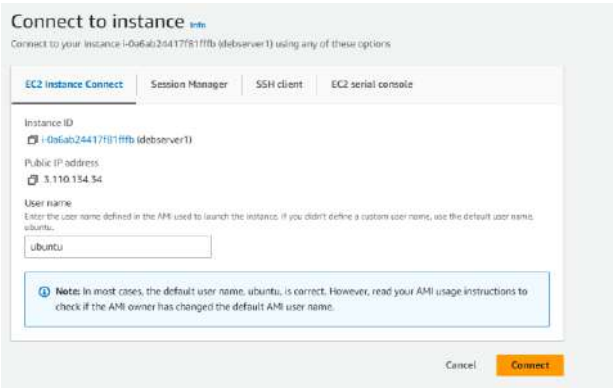
15. Now we Click on the 'Instance Id' link of our newly created server in our Instances list.



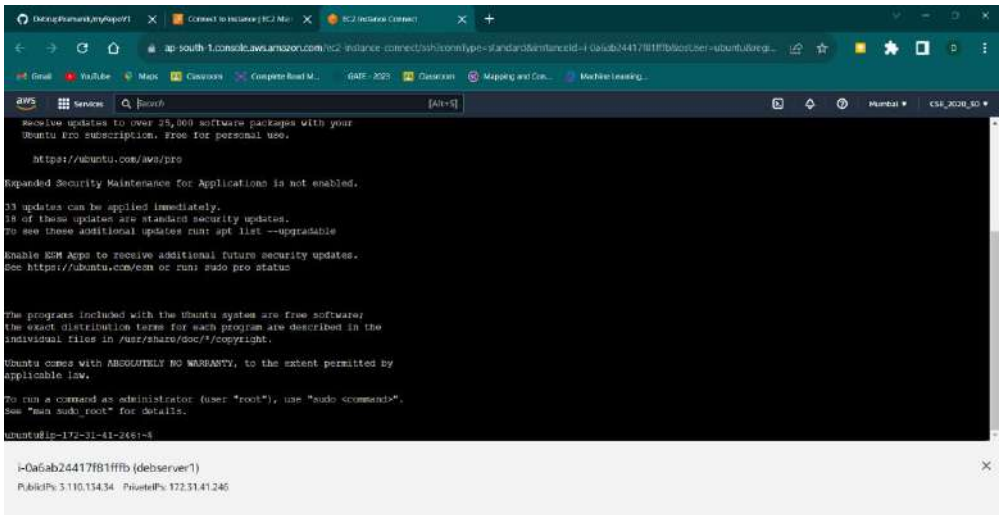
16. Now click on the connect button



17. Again, click on the connect button



18. After this anew Tab will open with a Bash Terminal that is of our remote EC2 server!
    Here we can type all our required commands that we used to type in a similar terminal by connecting to our remote server through our Bitvise SSH client software in our previous assignments.



19. Now type the following commands in the terminal:-

➔ git clone https://github.com/..... //Your GitHub Repository URL
Give your Username of GitHub when asked.
Give your account Token when your Password is asked.

```
ubuntu@ip-172-31-41-246:~$ git clone https://github.com/DebrupPramanik/myRepoV1.git
Cloning into 'myRepoV1'...
Username for 'https://github.com': DebrupPramanik
Password for 'https://DebrupPramanik@github.com':
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 6), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (15/15), done.
Resolving deltas: 100% (6/6), done.
```

➔ cd YourRepositoryname/

```
ubuntu@ip-172-31-41-246:~$ cd myRepoV1/
ubuntu@ip-172-31-41-246:~/myRepoV1$
```

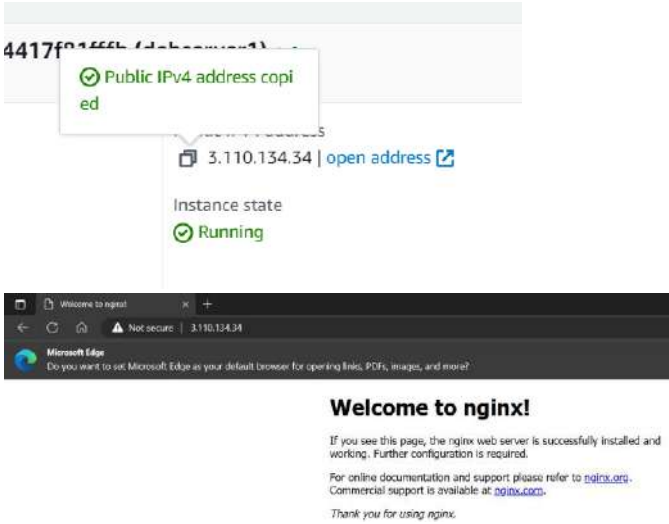
➔ npm install

```
ubuntu@ip-172-31-41-246:~/myRepoV1$ npm install
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  O
ee https://v8.dev/blog/math-random for details.

added 258 packages, and audited 259 packages in 15s

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

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 9.5.1 -> 9.6.5
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.5
npm notice Run npm install -g npm@9.6.5 to update!
npm notice
```

➔ node index.js

```
ubuntu@ip-172-31-41-246:~/myRepoV1$ node index.js
Started server
```

20. Now copy and paste the Public IPv4 address of your EC2 instance in another browser.



21. Now append the port no. 4000 (for our case) to the IP address in the browser with a ":" sign.

We have successfully Deployed a project from GitHub to EC2 by creating a new Security group and User Data.

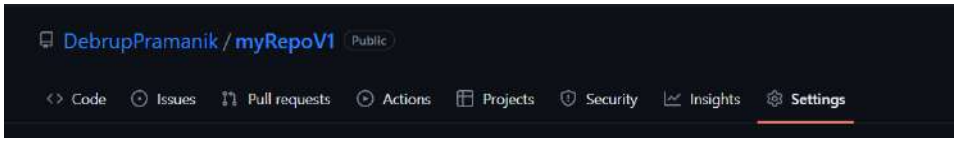
ASSIGNMENT – 11

Problem Statement: Build Scaling plans in AWS that balance load on different EC2 instances.

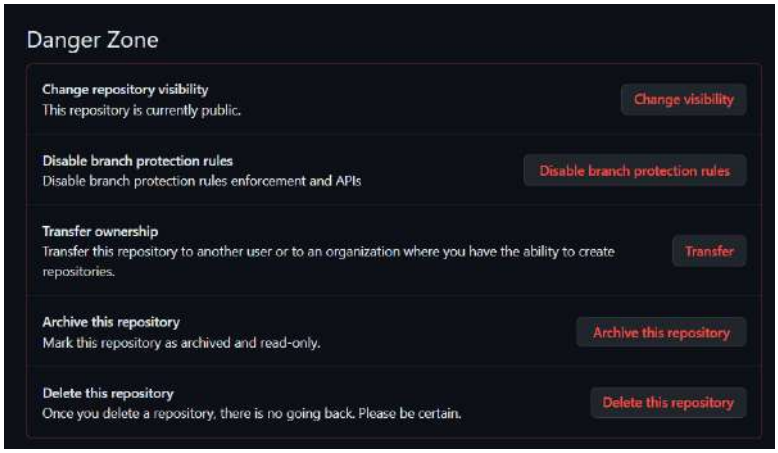
Procedure:

1. Sign-in to GitHub. Make sure the Repository which will be cloned is made public or not.

a. For This, select the “settings” tab of your repository.



b. Next Scroll Down until you reach the danger-zone section.

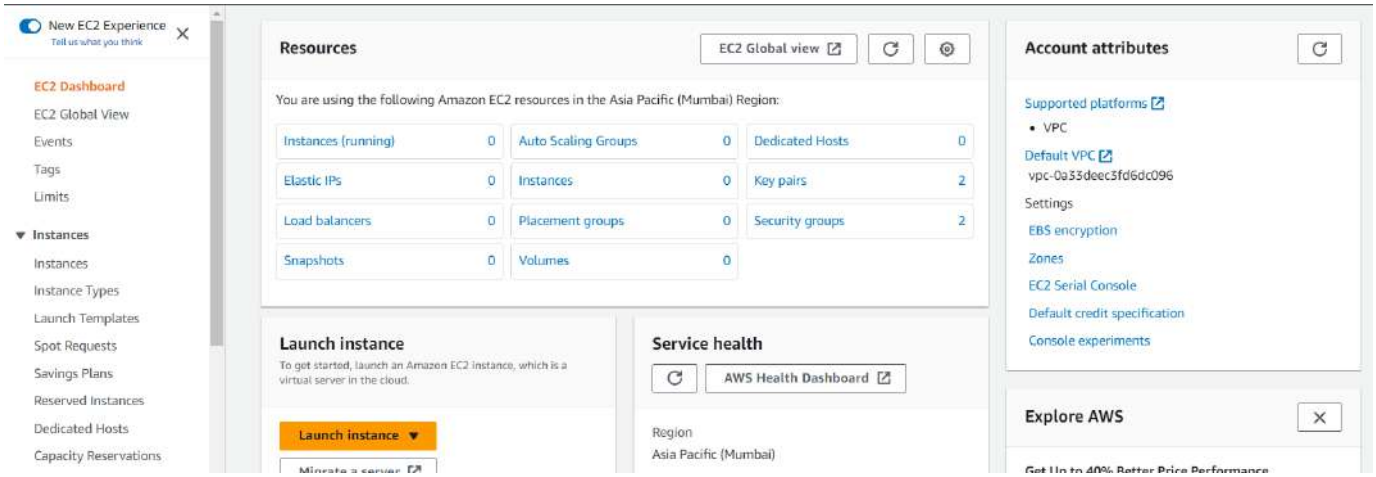


c. In here, check The Change Repository visibility option. Here, we can see the repository is currently Public. If it shows Private then click on the Change Visibility option and follow the on-screen Instructions to make the repository Public.

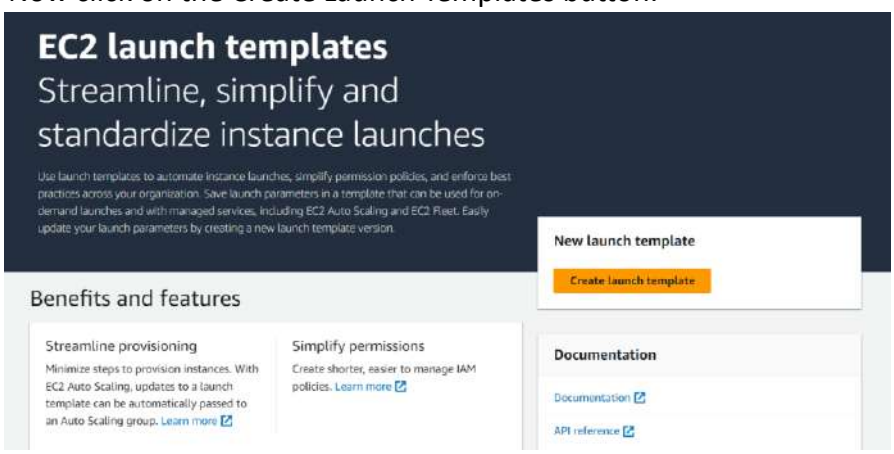
2. Now in another tab open AWS and Sign-in to your console.

3. Now go to your EC2 dashboard.

4. From the Left Side Nav-bar Go to your Instances Section. Under it Click the launch templates button.

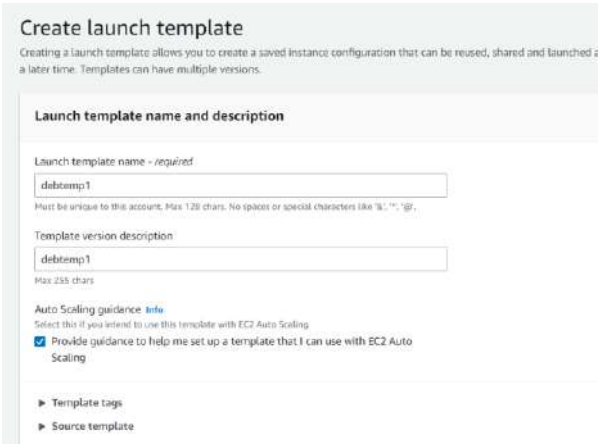


5. Now click on the Create Launch Templates button.

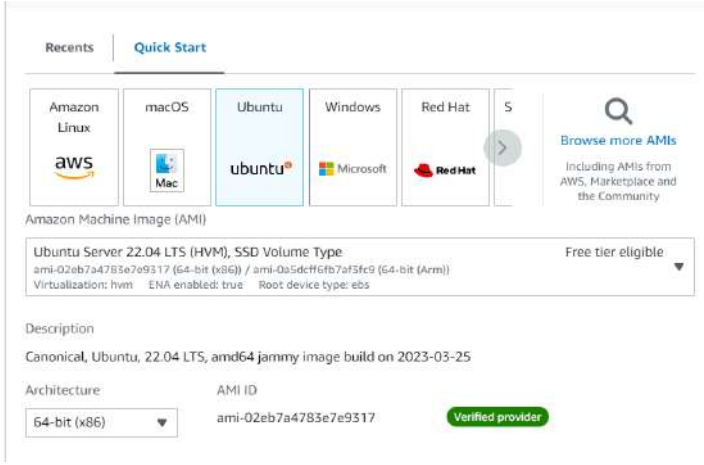


6. Now, give a name and description for your EC2 template you are about to create. Here we gave the same for both the fields.

Next, Check the “Provide Guidance” box.

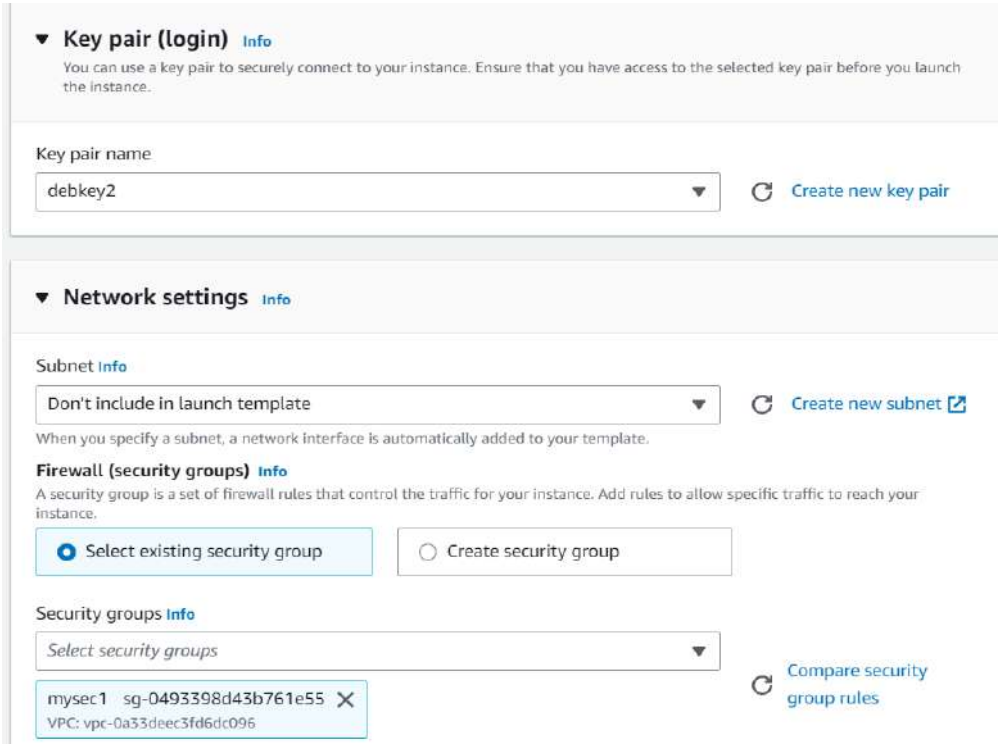


7. Next, under the quick start menu select Ubuntu as the OS.



8. Under Instance type select t2.micro type of configuration.

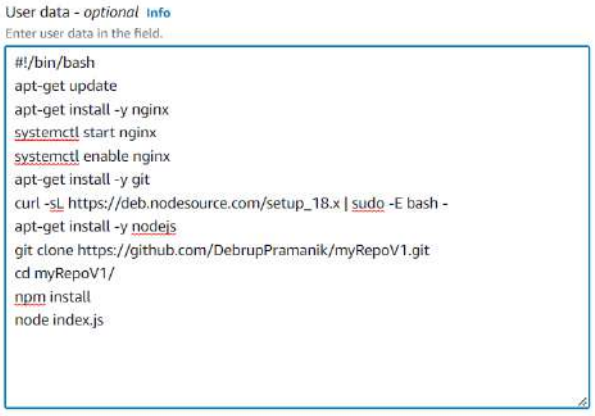
9. Select Existing Key-Pair and Security Group and if not applicable then Generate or Create a Key-Pair or Security Group wherever required.



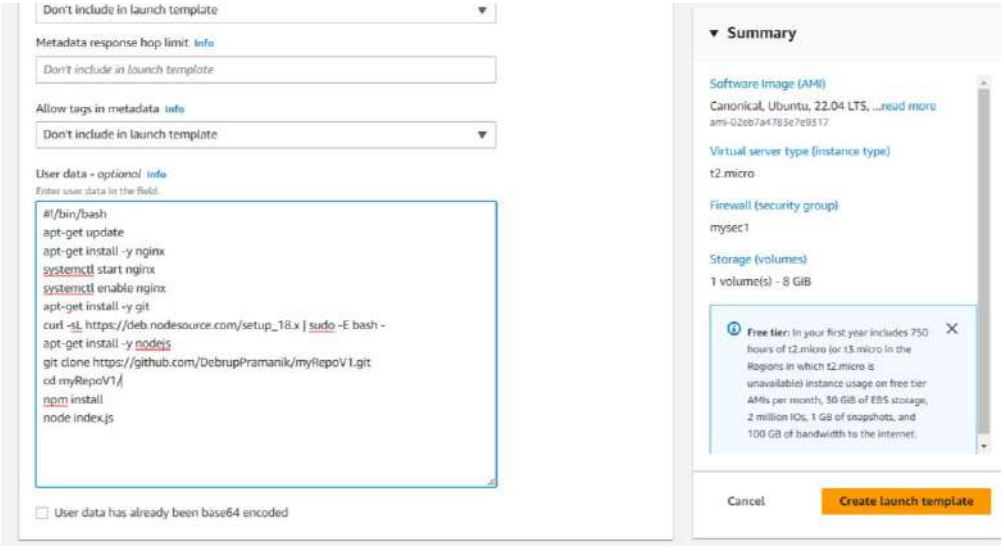
10. Now, Click on the Advanced Group Section at the bottom.
Scroll Down to User Data Section and paste the following commands in the provided box.

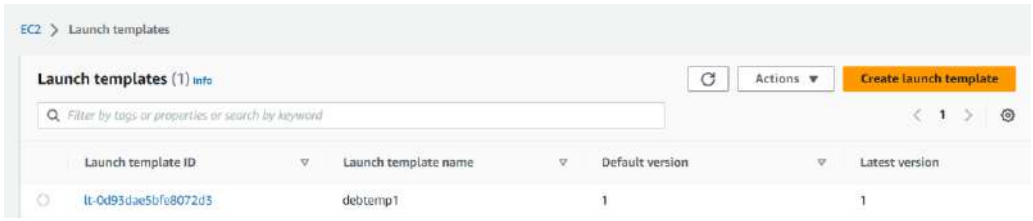

```
#!/bin/bash
apt-get update
apt-get install -y nginx
systemctl start nginx
systemctl enable nginx
apt-get install -y git
curl -sL https://deb.nodesource.com/setup_18.x | sudo -E bash -
apt-get install -y nodejs
git clone YourRepositoryURLhere
cd YourRepositoryNamehere/
npm install
node index.js
```

After adding the commands, it will look like this……

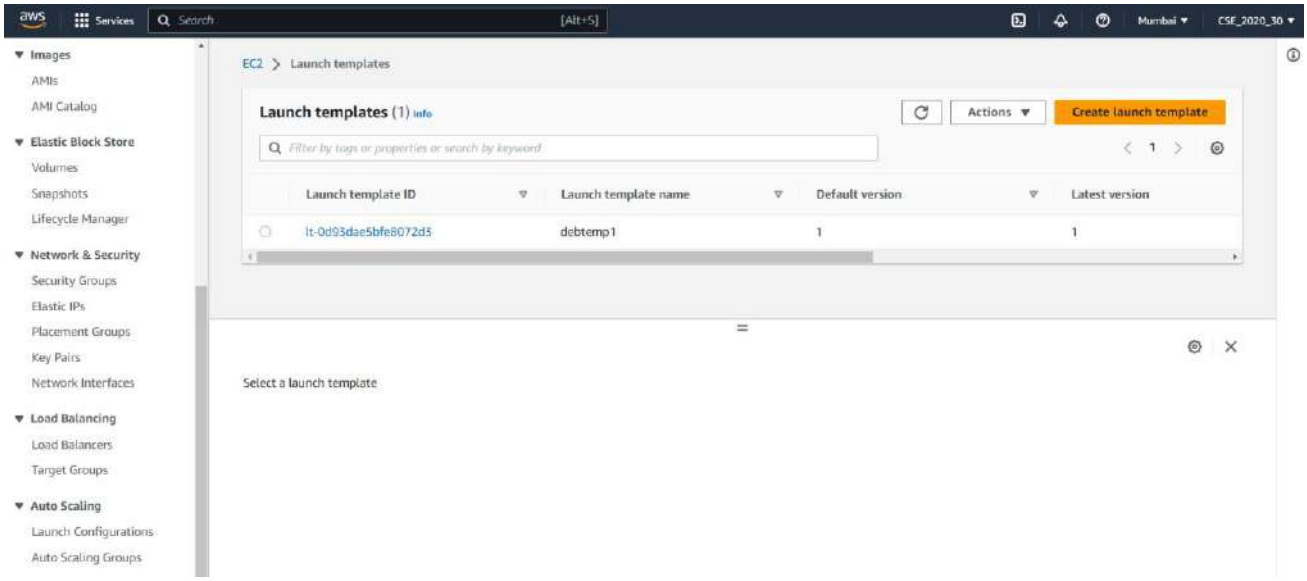


11. Now click on the Create Launch Template button.

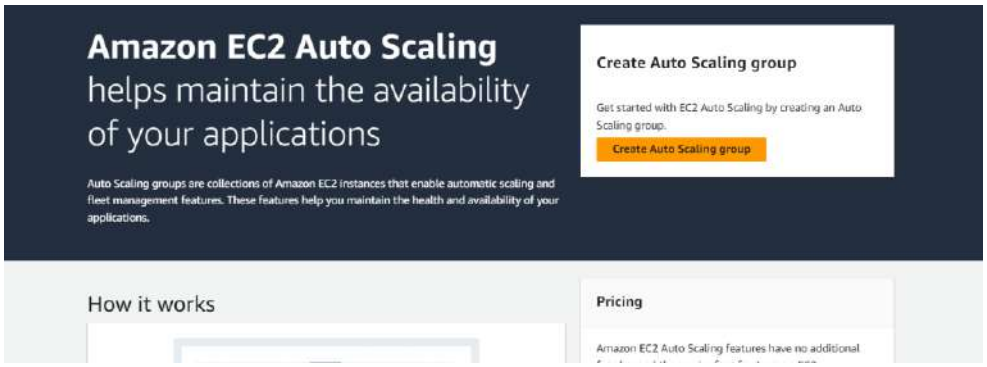




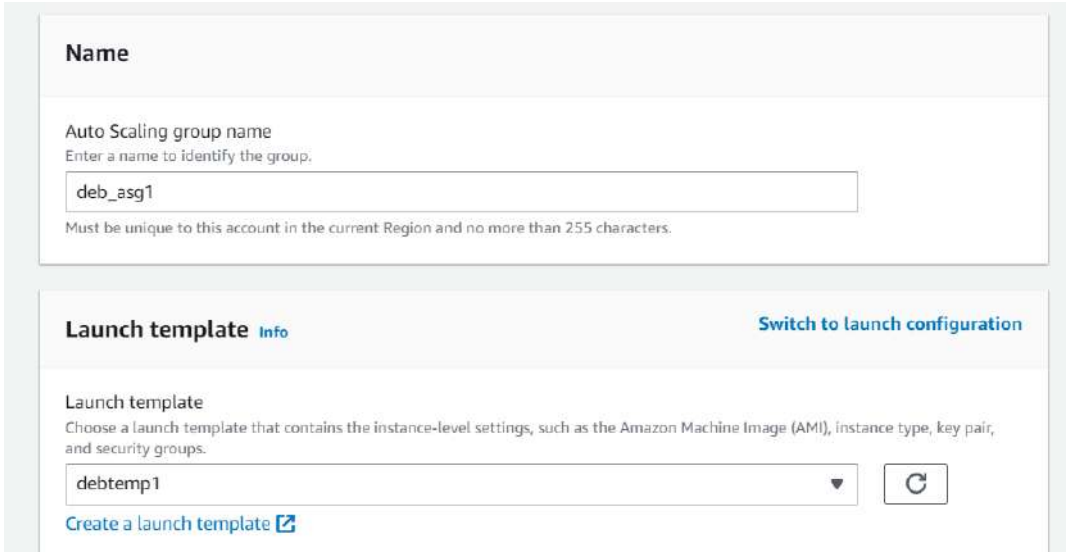
12. Now from the Left side Nav Bar go to Auto Scaling Groups under Auto-Scaling section.



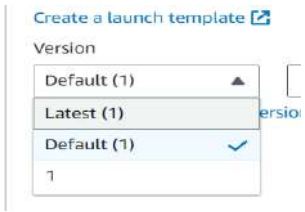
13. Now click on Create Auto Scaling Group.



14. Now, Give a unique name to your new Auto Scaling Group. Also select the Launch Template that we recently created by using the drop-down menu under Launch Templates section.

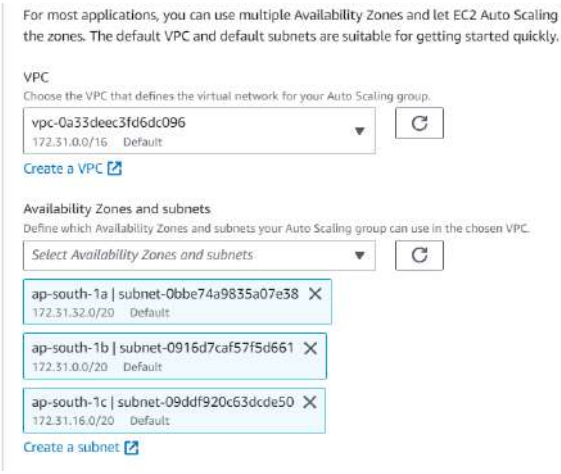


15. Under the selected Launch Template click on the version option and select Latest.



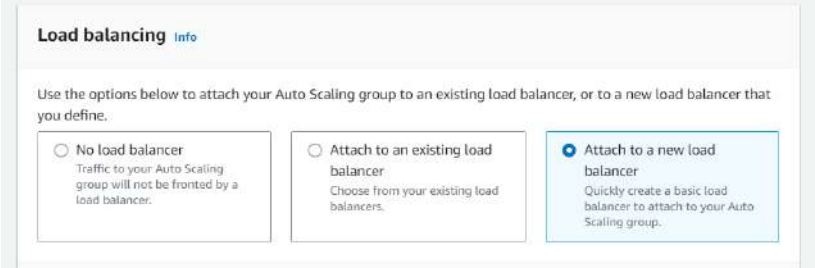
16. Now click on the Next button.

17. After that, Under Availability Zones and Subnets select all the zones that appear.

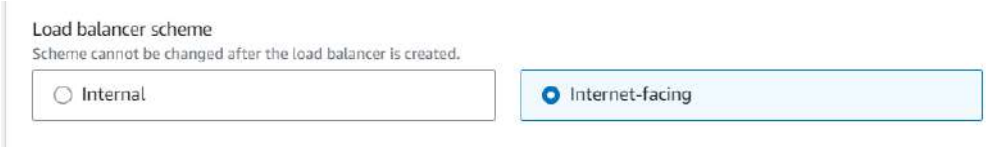


18. Again, click on the Next button.

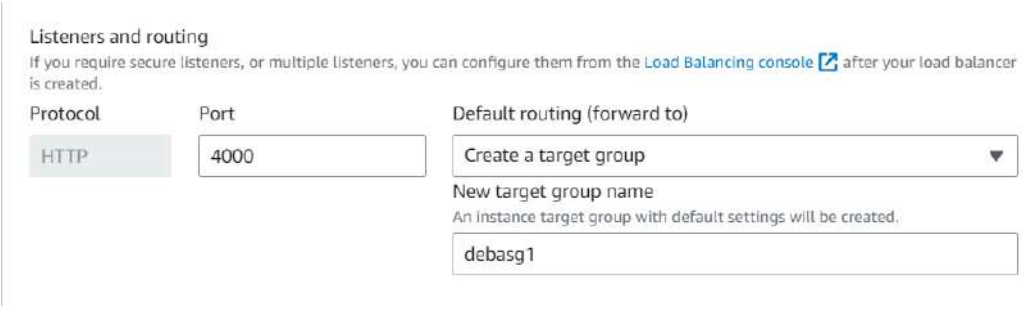
19. Now Under Load Balancers select the Attach to a New Load balancer option.



20. Now select Internet-Facing under Load balancer scheme.



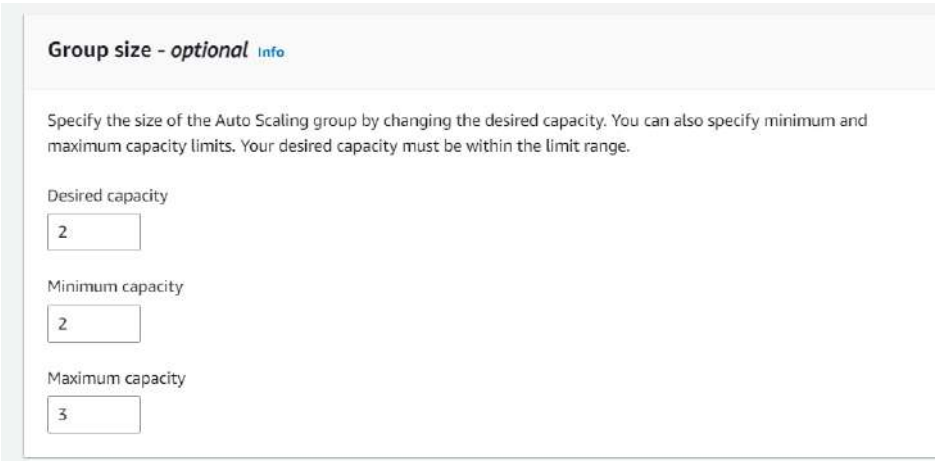
21. Under Listeners and Routing enter the port no. of the project and select Create target group followed by giving the target group a name.



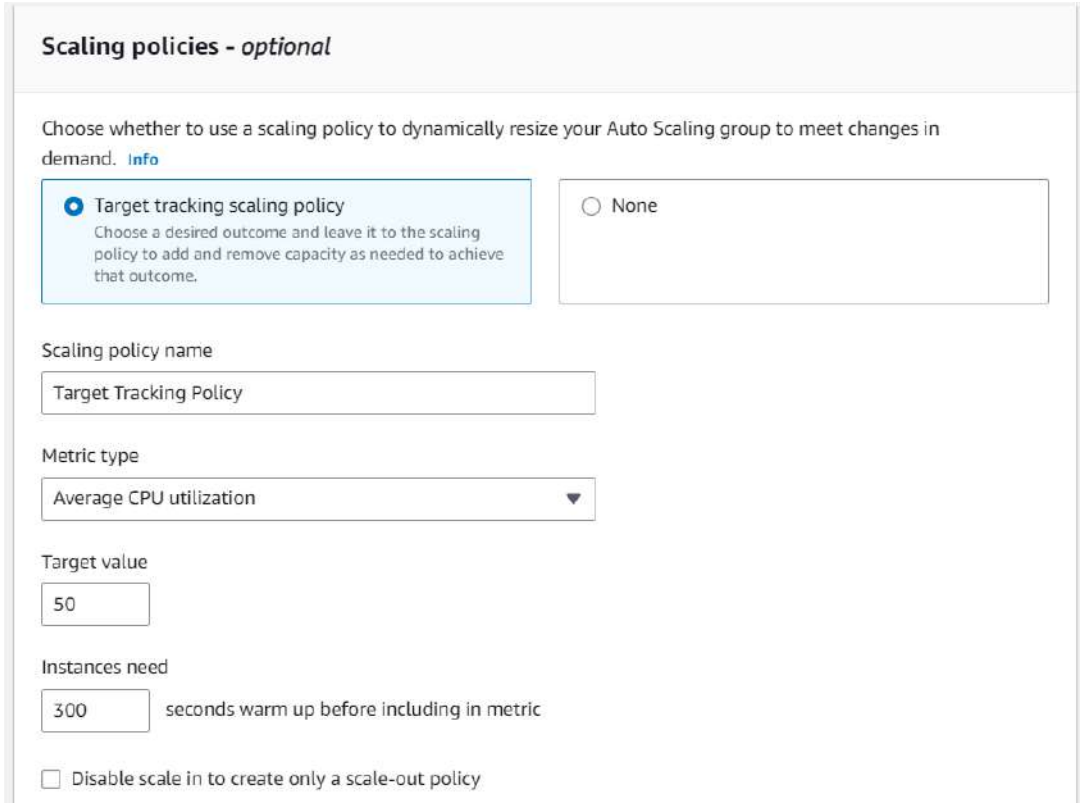
22. Now click on the next button.

23. After clicking on the Next button, a new page will open. Under Group Size mention:

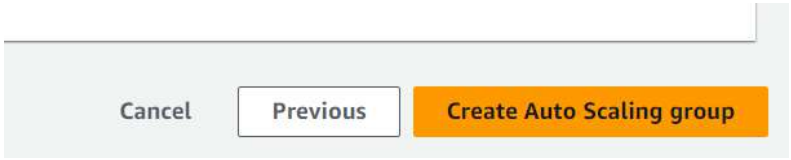
a) Desired Capacity = 2
b) Minimum Capacity = 2
c) Maximum Capacity = 3



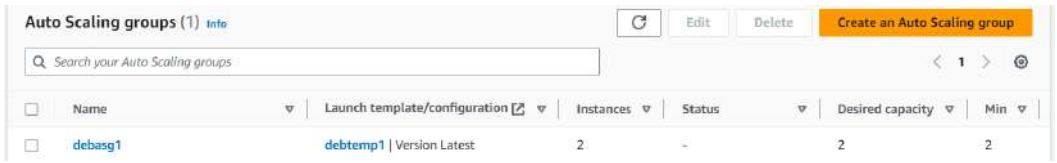
24. Now under Scaling policies Choose the Target Tracking Scaling policy option.
Select the metric type as Average CPU utilization.
Set Target Value to 50.
Set Warm-Up time to 300 seconds under Instances Need.



25. Then click on next.
26. Nothing to do in Notifications page. So again, click on the Next button.
27. No tags needed. Again, click on the Next button.
28. Now Review your Auto-Scaling Group you are going to create. Now click on the Create Auto-Scaling Group button.

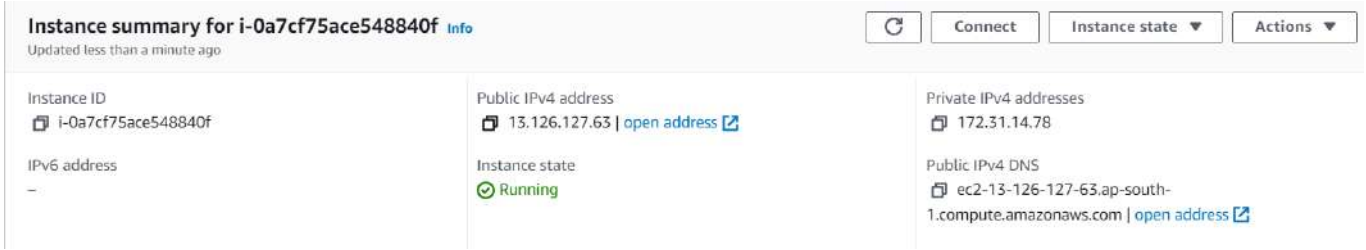


29. Now we can go to the Auto Scaling Groups section and find our newly created Auto Scaling Group.



30. Return to the Instances Page using the Left side Nav bar.

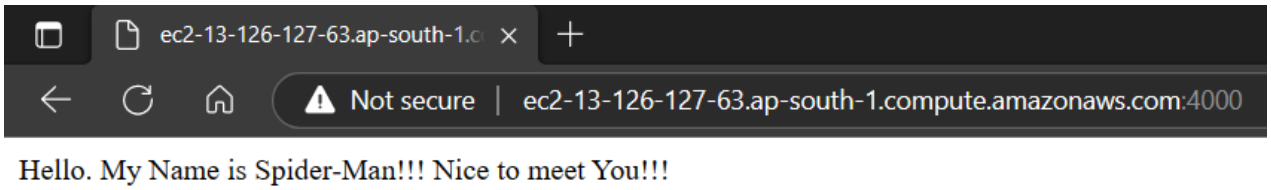
31. Click on the First instance. Copy its public IPV4 DNS.



32. Paste it in another browser.

We can successfully access the webpage.

33. Now to access our project webpage we need to append the port no. (4000) of our project with a ":"

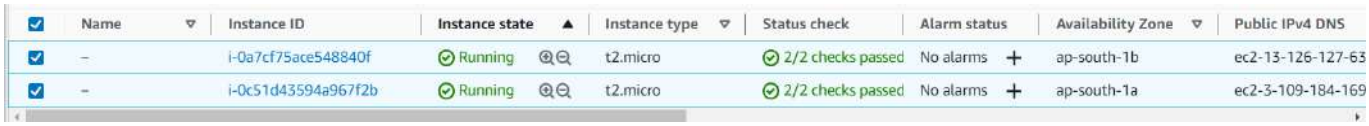


Hello. My Name is Spider-Man!!! Nice to meet You!!!

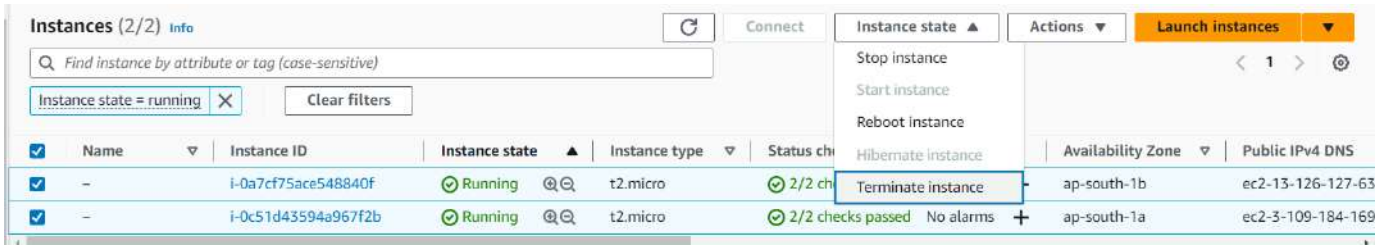
34. To test our Auto-Scaling Group actually works we need to crash or overload the existing instance servers. Then only our Auto-Scaling Group will provide fresh instance servers automatically in case of crash; or it can provide extra servers to handle overloads.

35. We will now **CRASH THE SERVER INSTANCES** by terminating them.

36. Go to the instances page. Select the server instances.



37. Now click on the Instance State button up top. From the drop-down select the Terminate instance option.



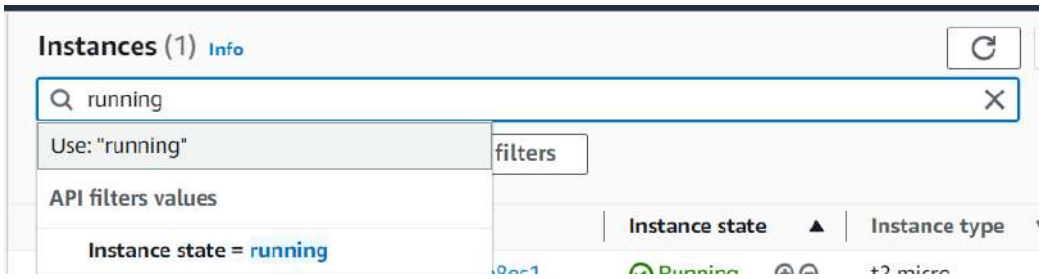
38. Refresh the instances the page from time to time. After few seconds the instances will get terminated.

39. Wait for some time. Keep refreshing using the refresh button on the left side of Instance state button.

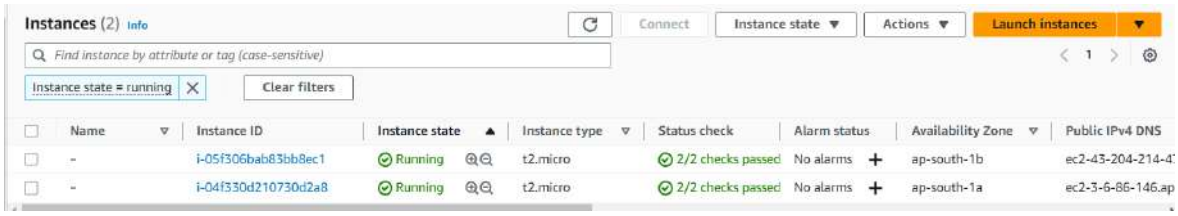
40. After some you will notice a new instance server will appear automatically! To help finding it more easily we need to activate the instance running filter. Click on the search box below the Instances Section Heading. Start typing running.

Select the option "Instance state = running option in the suggestion dropdown. The filter will be activated. You can alternatively type the command directly in the search-box directly.

Now, only the running instances will be shown in the list.



41. After some few seconds of refreshing we will be able to see two new servers are running.



Now again copy paste the new public IPv4 DNS of the first instance and port no. in the other browser to see if the instances are working. It will be working.
So, our Auto-Scaling Group can handle instance crashing by providing new fresh instances.

42. We will now **OVERLOAD THE SERVER INSTANCES** by running scripts and increasing CPU utilization value above the threshold that we specified during Configuration of the Auto scaling group.

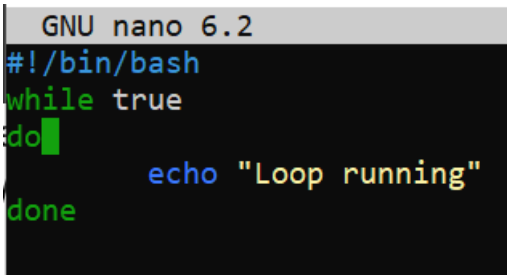
43. For it we will use:

a) Use Bitvise SSH client for instance 1.
b) Use direct connect terminal in AWS for instance 2.

44. For Instance-1:

a) Copy the public IPv4 address
b) Open Bitvise SSH client.
c) Paste the IP and select/specify the necessary options. (**Refer Ass7**)
d) Now Log-In to your server.
e) Open the new Terminal.
f) Now enter the command:
➔ **nano infil.sh**
g) After the command a new nano Editor window will open. Type the following in it.

```
#!/bin/bash
while true
do
        echo “Loop running”
done
```



h) Now, to save and close the shell script we need to press the following shortcuts and keys sequentially:
Ctrl+X
Y
Enter
i) Now you will be returned back to the terminal.
j) Now type the following commands:

➔ **chmod +x infil.sh**
(Used to give the execute permission for the infil.sh file)

```
ubuntu@ip-172-31-13-76:~$ nano infil.sh
ubuntu@ip-172-31-13-76:~$ chmod +x infil.sh
```

➔ **./infil.sh**
(Used to execute the infil.sh script)

```
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
Loop running
```

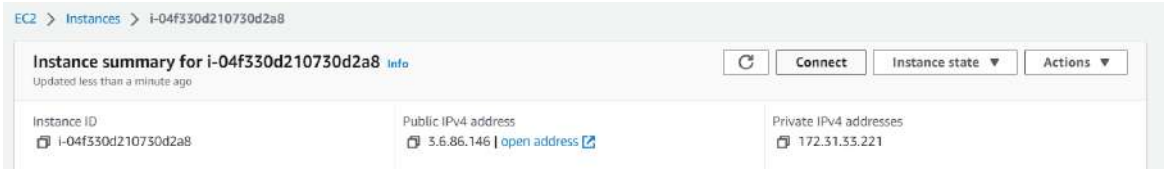
k) Now the script will start running infinitely!

l) DO NOT CLOSE THE TERMINAL. Keep it minimized.

45. For Instance-2:

a) Click on the instance 2.

b) Now click on the connect button



c) Click on connect again

d) After the terminal gets opened we again follow the same steps as we did for instance-2

e) Enter the command:

➔ **nano infil.sh**

f) After the command a new nano Editor window will open. Type the following in it.

```
#!/bin/bash
while true
do
        echo “Loop running”
done
```

g) Now, to save and close the shell script we need to press the following shortcuts and keys:
Ctrl+X
Y
Enter

h) Now you will be returned back to the terminal.

i) Now type the following commands:

➔ **chmod +x infil.sh**
(Used to give the execute permission for the infil.sh file)

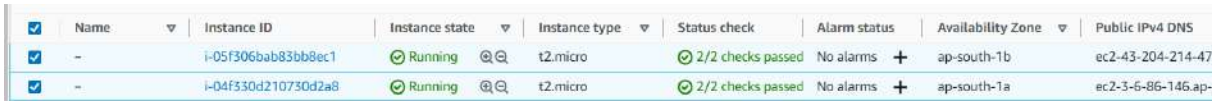
➔ **./infil.sh**
(Used to execute the infil.sh script)

j) Now the script will start running infinitely!

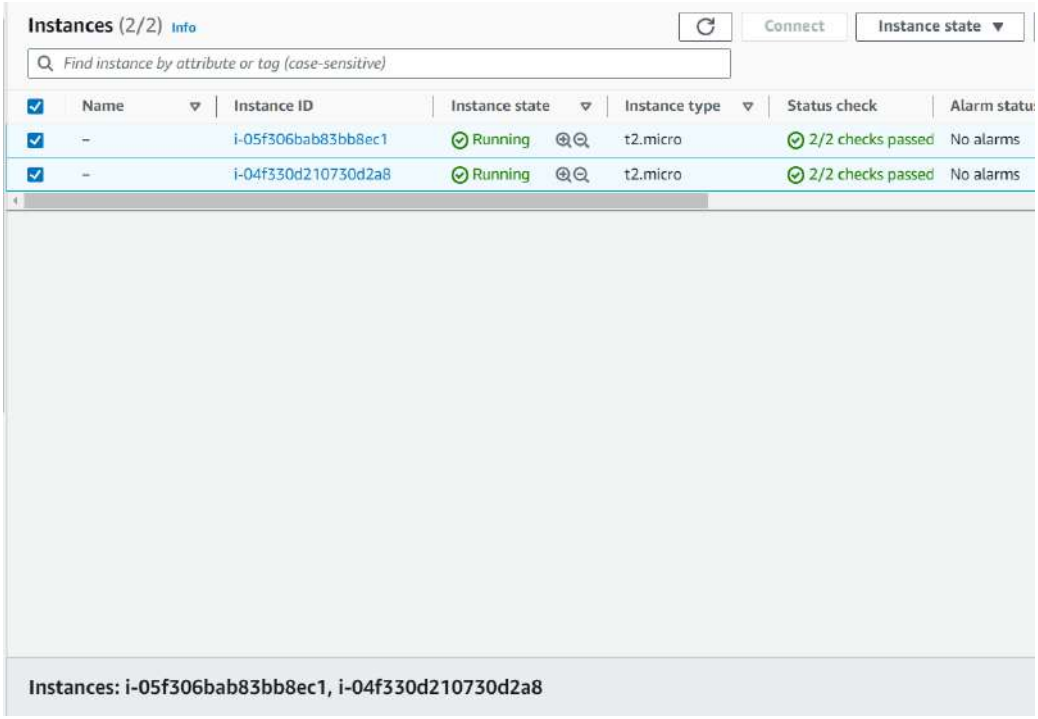
k) DO NOT CLOSE THE TERMINAL TAB. Go back to the previous tab to keep working in AWS.

46. Now go to the instances page.

47. Select both the instances.

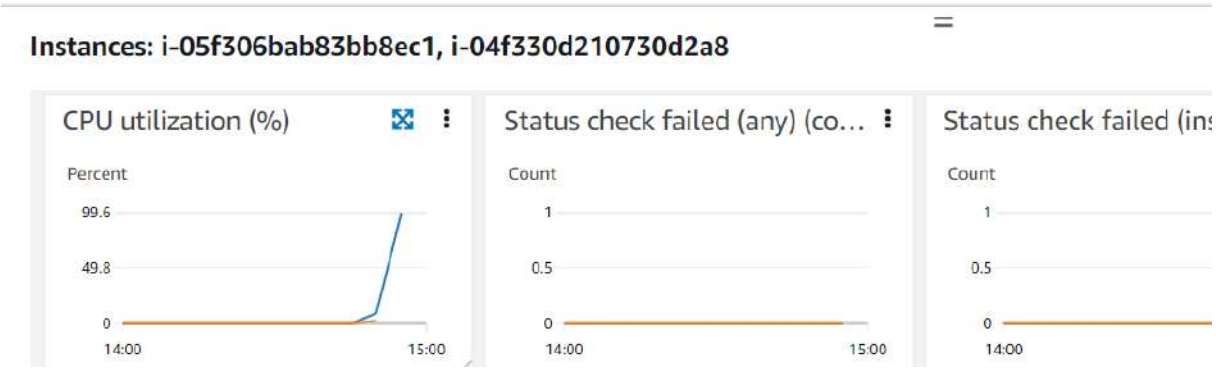


48. Click on the instances white bar at the bottom of the page.

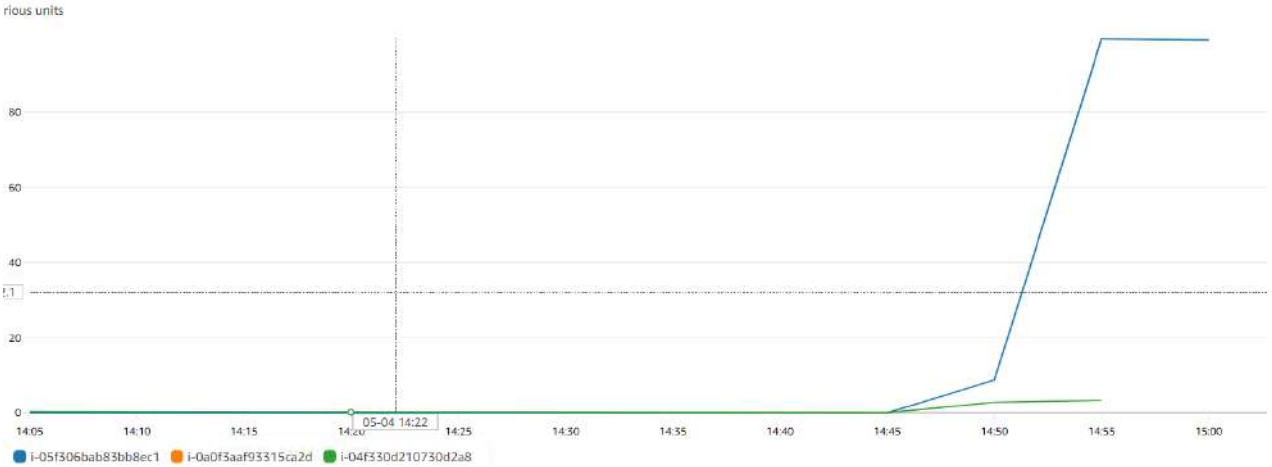


49. Now drag the two bars to expand the view.

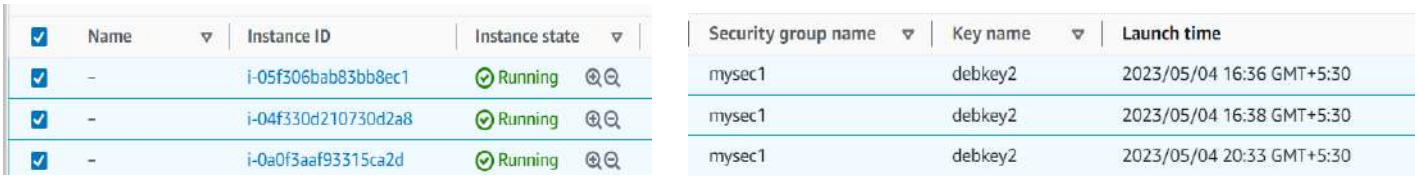
50. We are interested only in the CPU utilization graph. Click on the maximize icon by hovering over the graph as shown in the fig to maximize this graph.



51. Our 1st instance has already crossed over 50% utilization. That's why we can see already a new third instance has been initiated by our auto-scaling group to compensate for the overload.



52. There can only be 3 servers running at a time for us as specified in our Auto-Scaling group when we were creating it. Hence, we have reached our maximum limit of instances running concurrently.



Hence, our Auto-Scaling Group can handle instance overloading by providing new instances to handle the overloading.

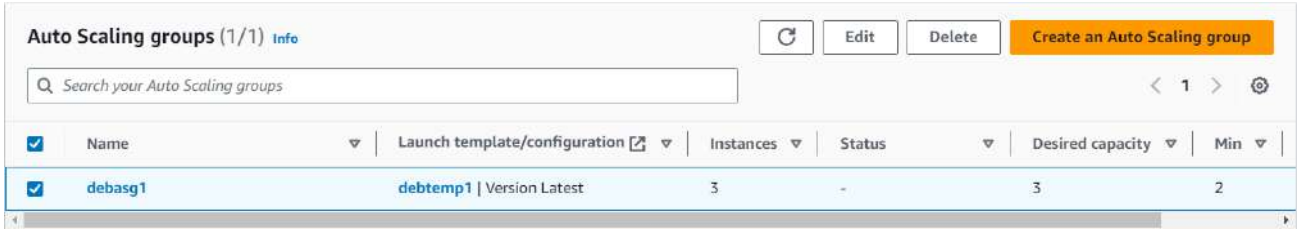
Note that our project webpage was not at all disconnected because of this.

We have successfully created, configured and tested our Auto-Scaling Group.

Now observe that whenever we close or terminate any instance then a new instance gets created. Hence, we cannot delete them if we want to delete them finally.

Follow these steps to cleanly remove Auto-Scaling Groups and everything related to it.

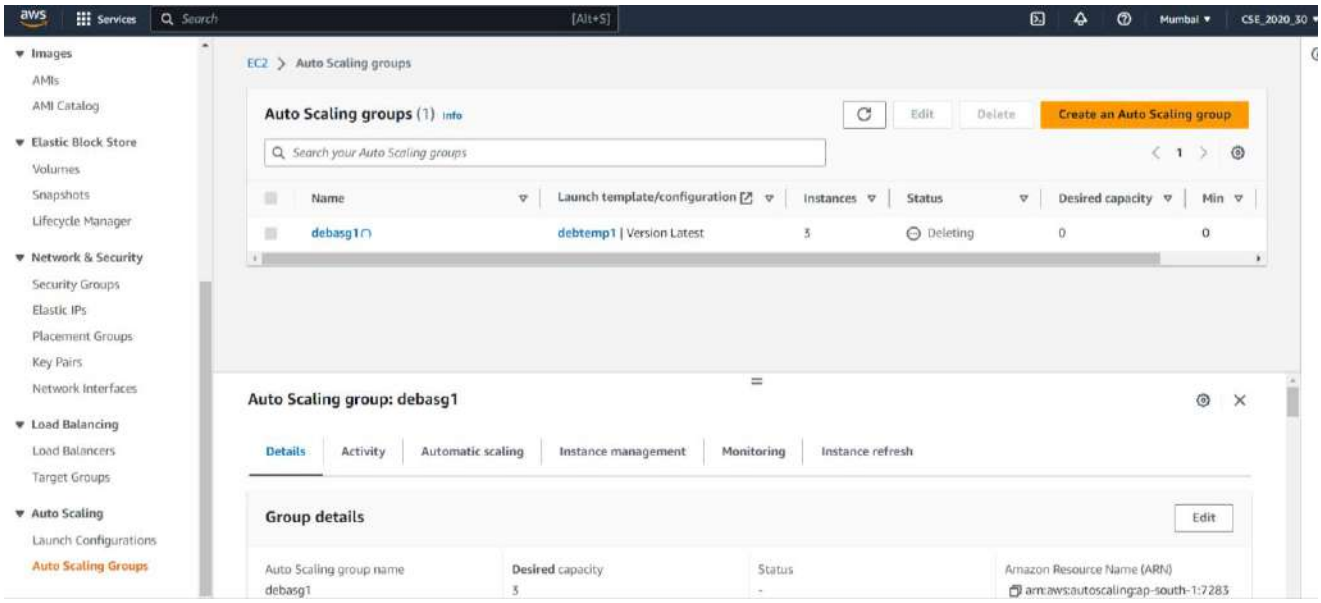
1) Go to Auto-Scaling groups and select the one which we are using.

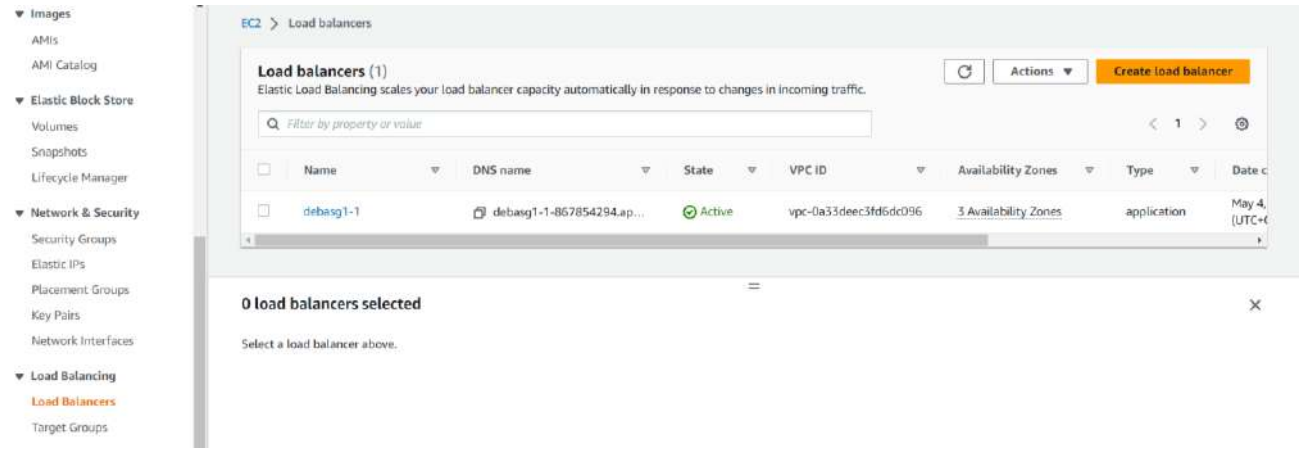


2)Now click on the delete button.

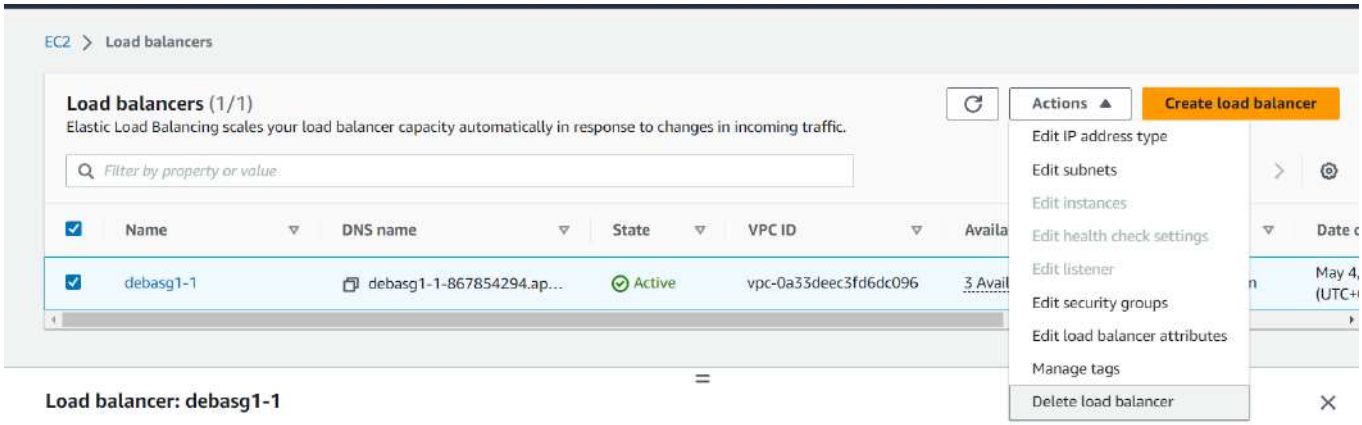
3)Type delete and finally delete it.

4)Notice the status changes to Deleting. It will take some minutes to fully delete. Now go to the Load balancer section from the left side nav bar.





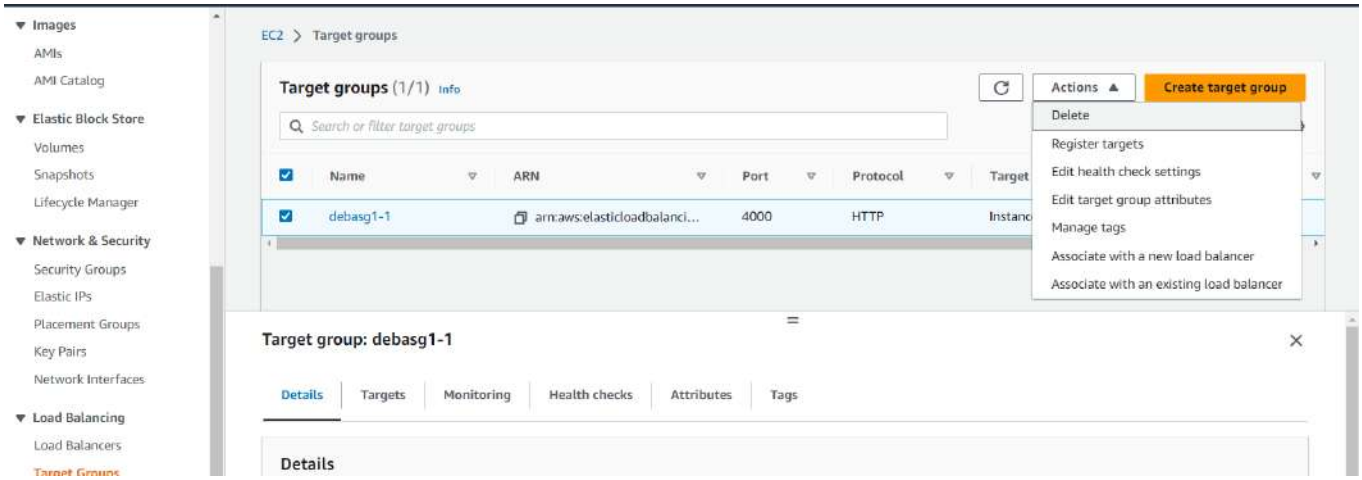
5)Now select the load balancer and click on the action button on the top.



Now select the Delete Load balancer option to delete it.

6) Now go to the Target Groups section.

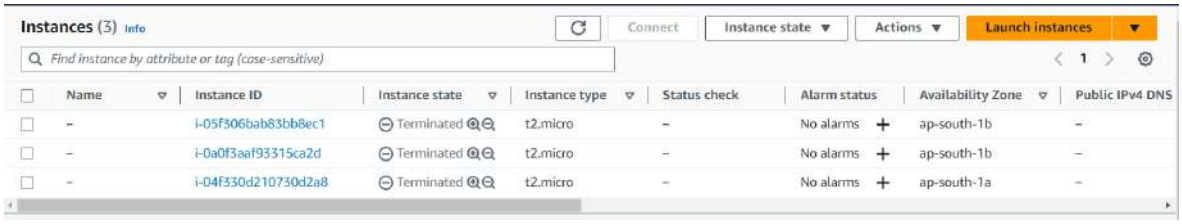
7)Select the target group and click on the action button on the top. Select the delete option.



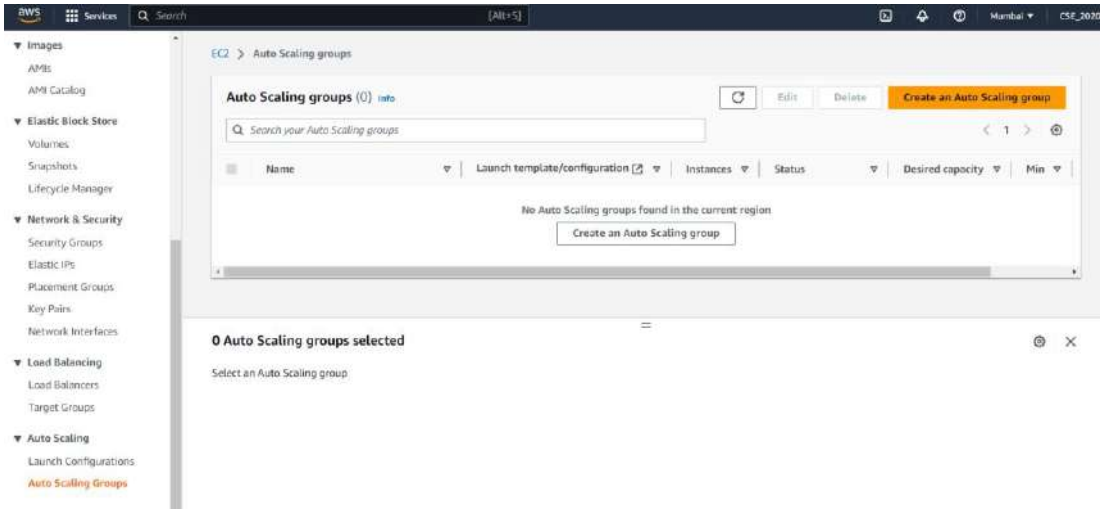
8)Now go to the instances page.

9)You will find that all the instances created by the Auto-Scaling group will automatically be terminated.

(If not wait for some time. Check if the Auto Scaling Group has been deleted by now.)



10)Finally check the Auto-Scaling Group section to see if it is completely removed/deleted.



Everything was deleted successfully.

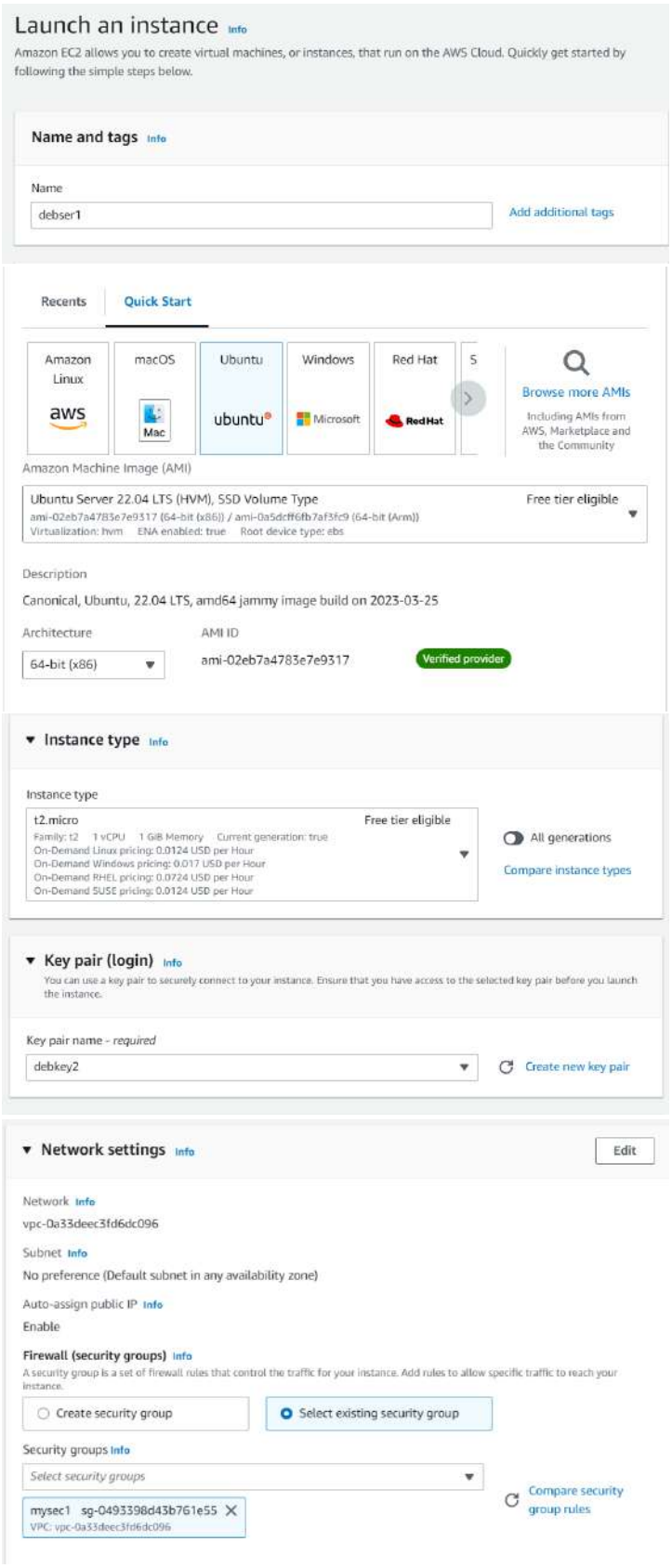
Hence, we successfully deleted our Auto-Scaling Group and all the load balancers, target groups, instances that was created.

ASSIGNMENT-12

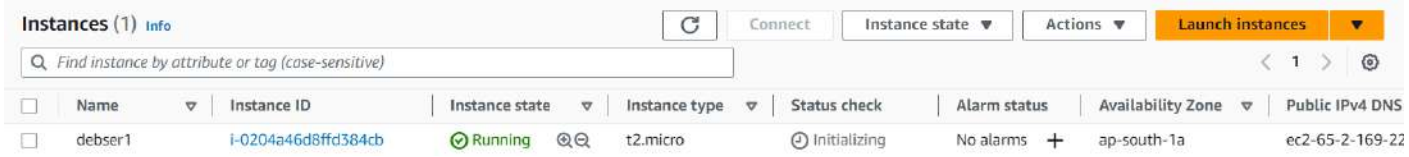
Problem Statement: Deploy a project from GitHub to EC2 without using Port.

Procedures:

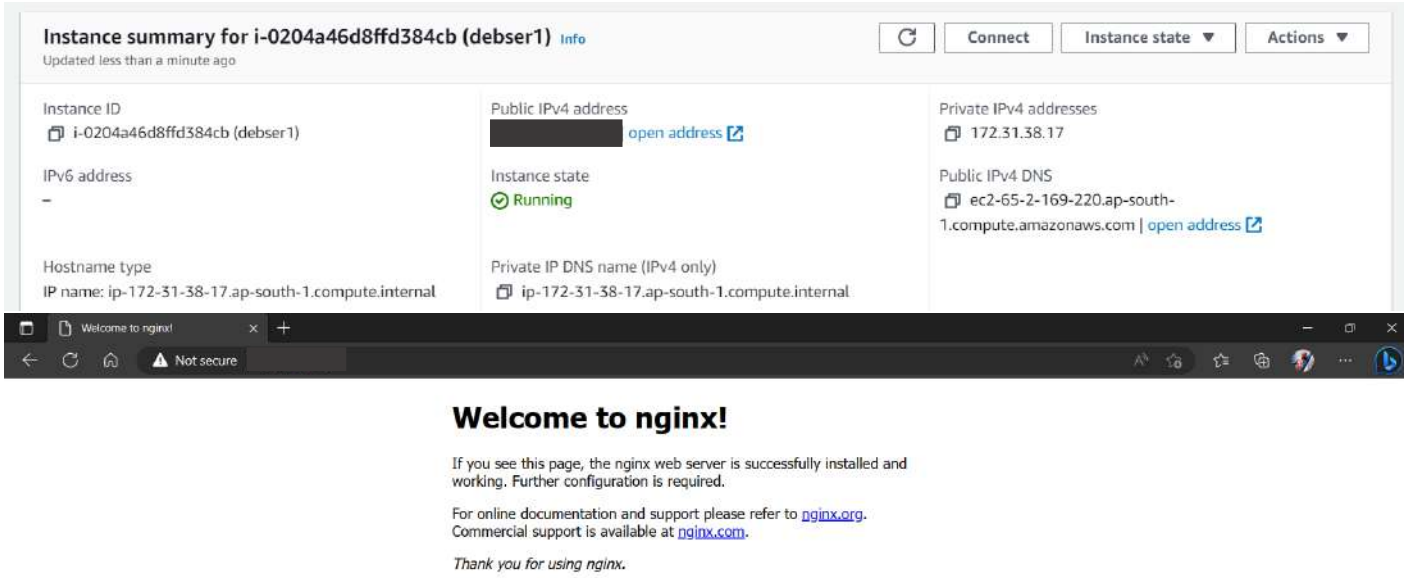
1. Sign-in to AWS console.
2. Go to the EC2 dashboard. Now go to the instances page.
3. Click on the create new instance button.
4. Now create an EC2 server using the Security Group created earlier and enter the user data **(Refer to Ass10)**



5. Create the instance and click on the instance after creation.

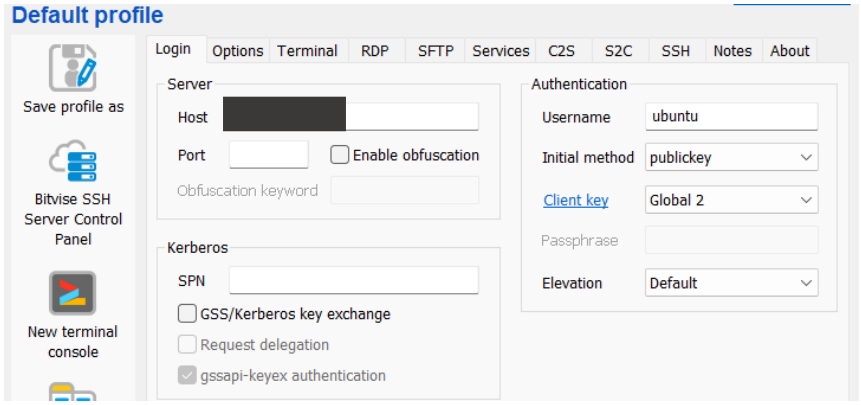


6. Copy the public IPv4 address and paste it in another browse. The nginx homepage will show up.

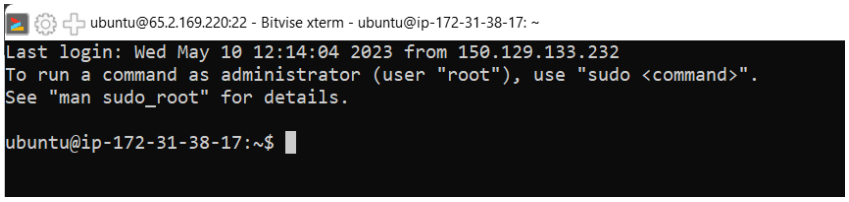


Our server is working perfectly. Note, in previous assignments we used to connect to our project webpages using port no. However, in this exercise we are going to access our project webpage without using any port no.

7. Copy the Public IPv4 address of the server instance and use this to connect it to the server using Bitvise SSH client. **(Refer Ass7)**

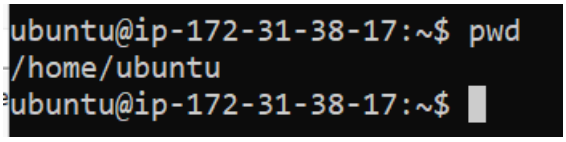


8. Now open the terminal in Bitvise.



9. Enter the following commands in it.

➔ **pwd**

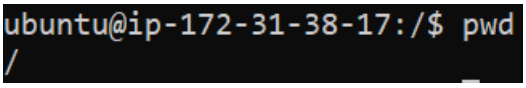


(To check current directory)

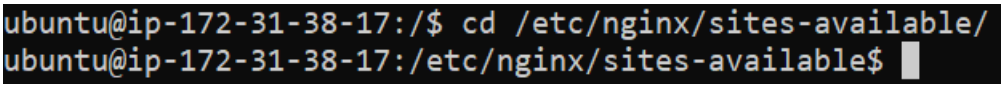
➔ **cd /**

(To go to root folder)

➔ pwd

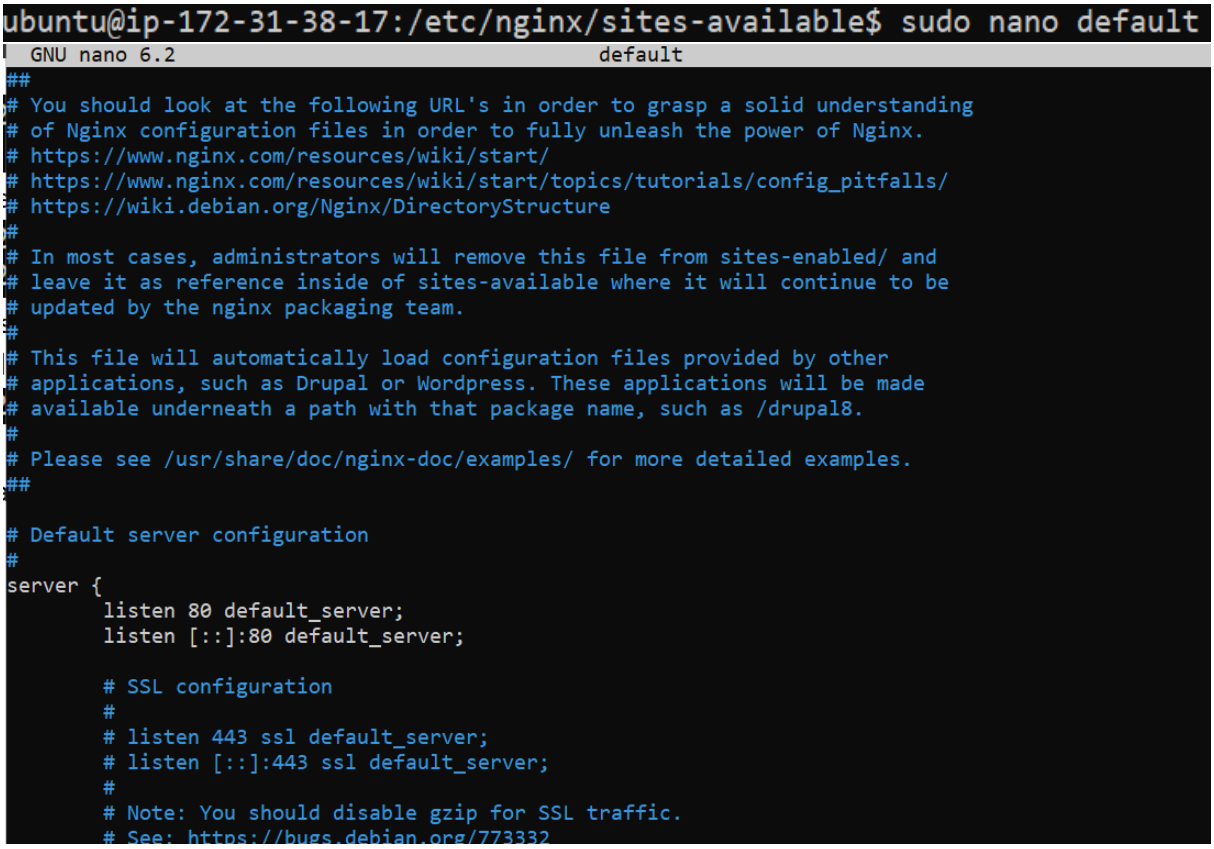


➔ cd /etc/nginx/sites-available/



(To open the sites-available directory under nginx)

➔ sudo nano default



(To open the default file in the nano editor)

10. After opening the default file in the nano editor, search for the location / {}. It should be after server_name_;

```
        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
```

11. Comment out the location block and each and every line inside the block.

```
        server_name _;

        #location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                #try_files $uri $uri/ =404;
        #}
```

12. Now paste the following code just under the closing curly bracket.

```
location / {
        proxy_pass http://localhost:4000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'Upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
```

```
        server_name _;

        #location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                #try_files $uri $uri/ =404;
        #}
        location / {
                proxy_pass http://localhost:4000;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'Upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
        }
```

13. Now save it by Ctrl+X and exit nano editor.

14. You will be reverted back to the terminal. Type the following command…..

➔ **sudo systemctl restart nginx**

```
ubuntu@ip-172-31-2-192:/etc/nginx/sites-available$ sudo systemctl start nginx
```

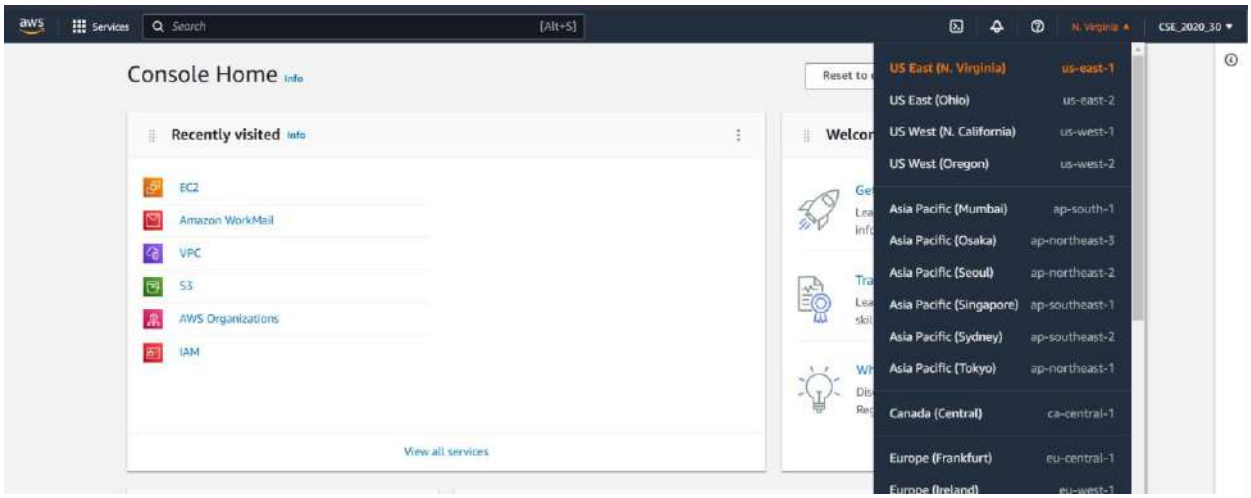
15. Now paste the public IPv4 address in your browser. Now press Enter. Our project page will show up without entering our port no.

We have successfully deployed a project from GitHub to EC2 without using port.

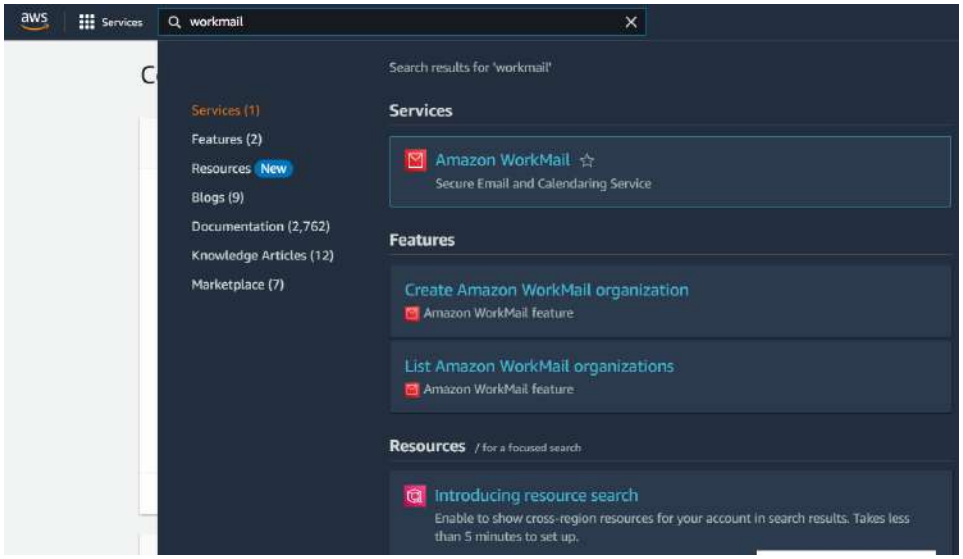
ASSIGNMENT-13

Problem Statement: Create a workmail for your profile.

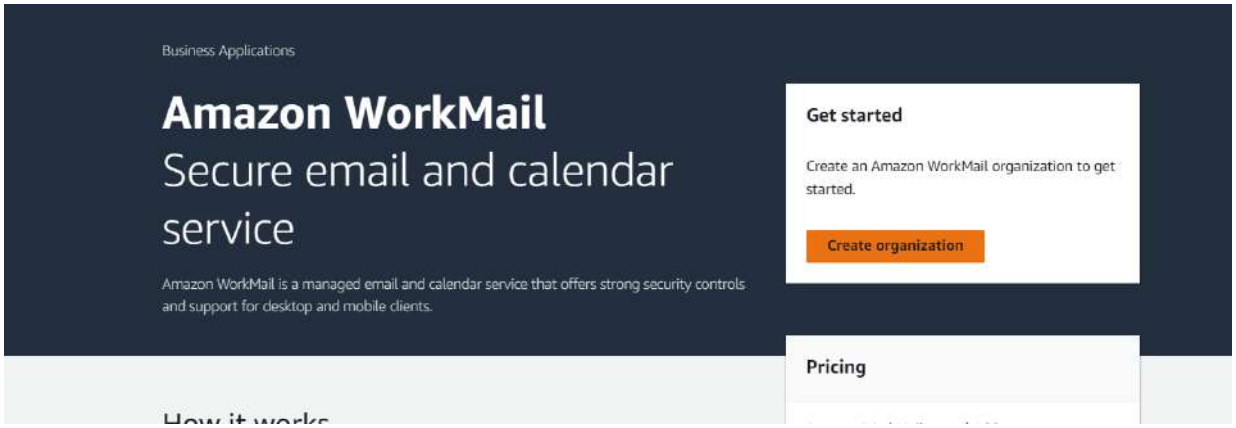
1. Sign-in to your AWS console.
2. Change your region to N.Virginia from the dropdown on the left-side of the username in the top bar of the console homepage.



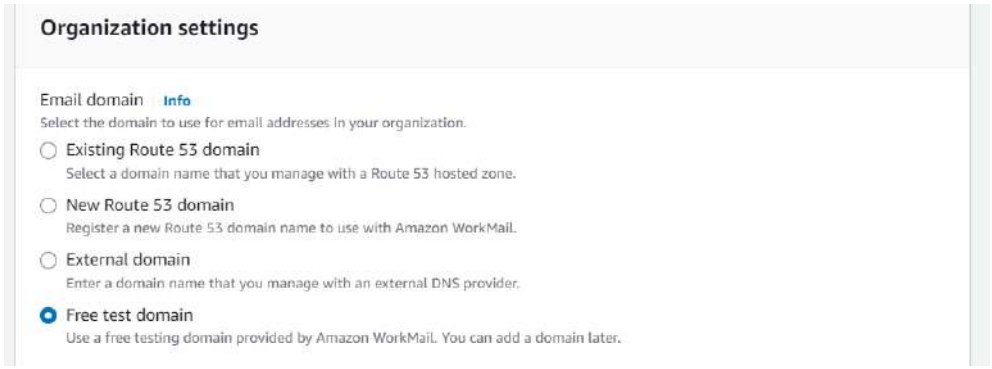
3. Search for workmail.
4. Select the AWS workmail option.



5. Click on the create organization button.



6. Select the free test domain option under email domain.

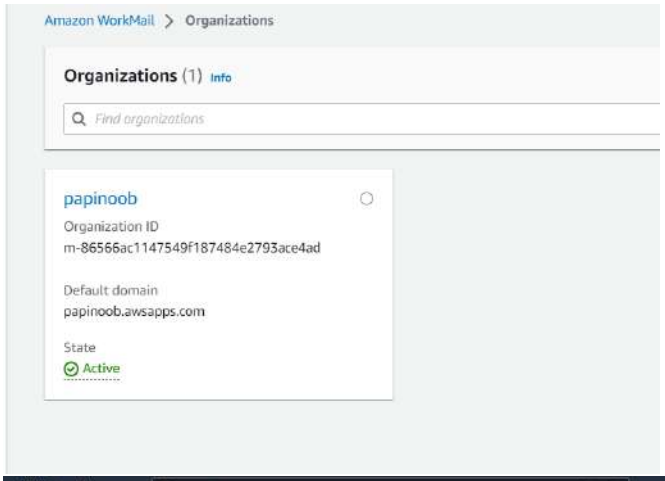


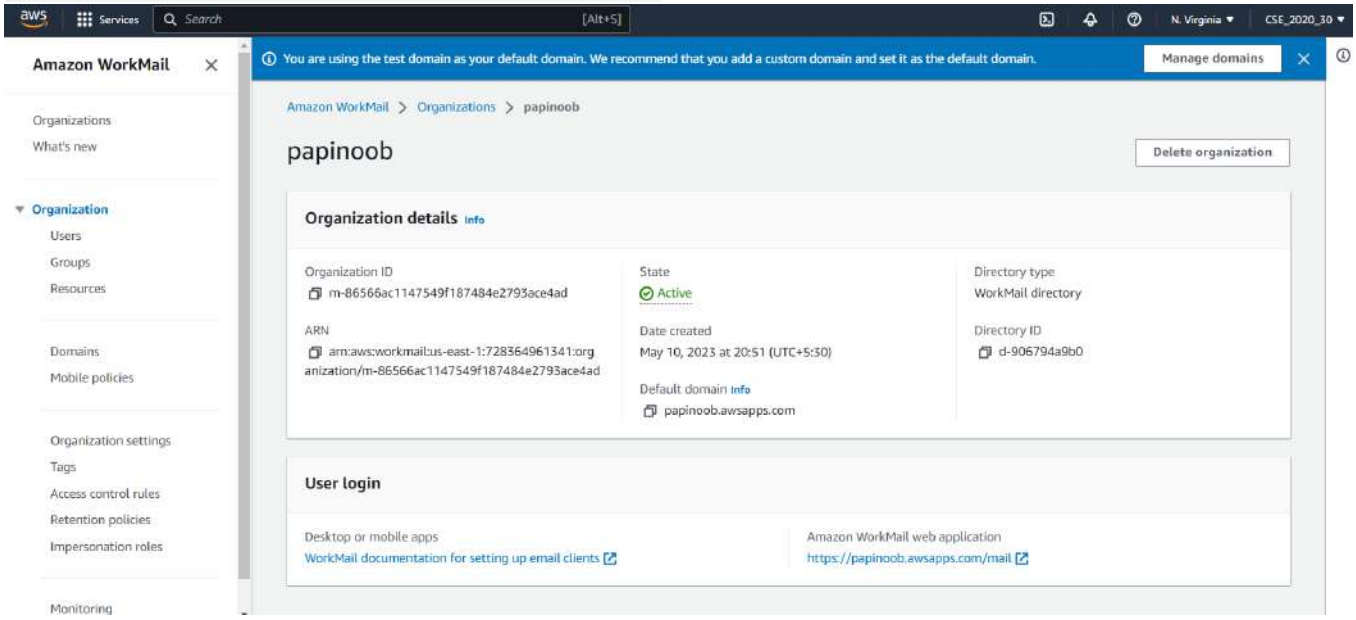
7. Give the organization name/ alias. Make sure it is unique in the region.

8. Next click on create organization button.

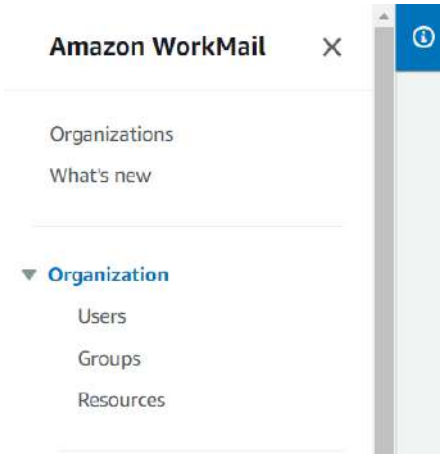
9. Wait for two minutes.

10. After waiting the organization will become active. Now click on the organization name.

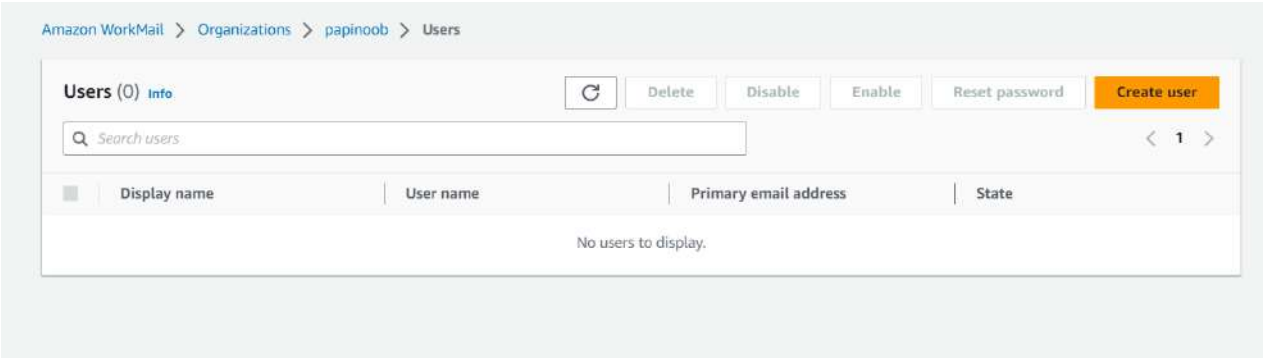




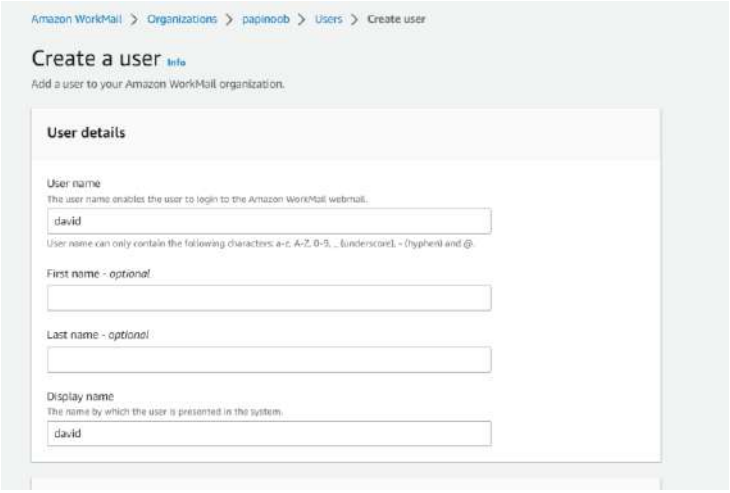
11. Now go to the Users section under Organization tab on the left side nav bar.



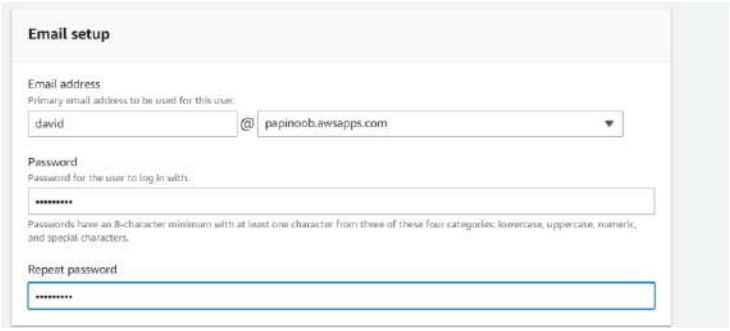
12. Click on the create user button.



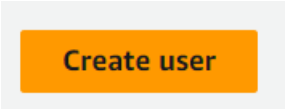
13. Give username. Then display name.



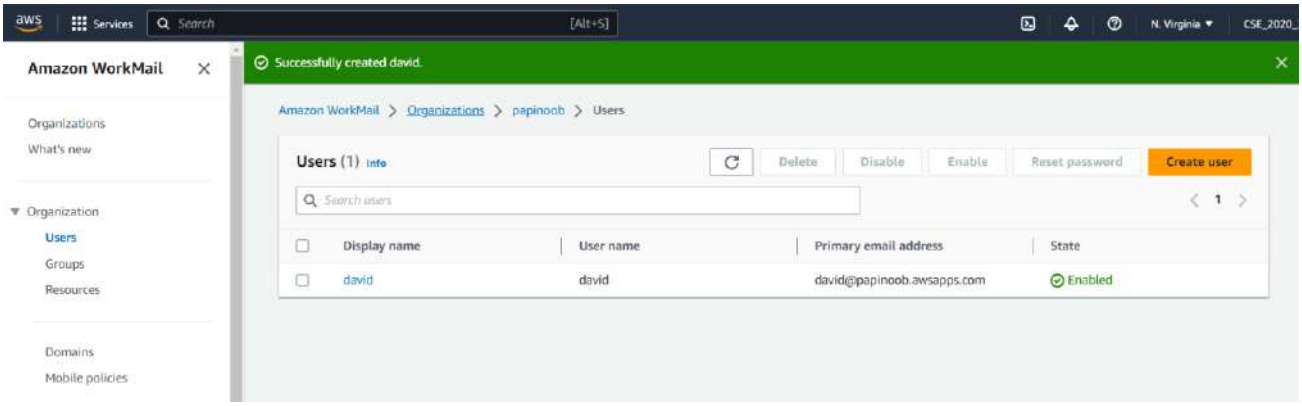
14. Then provide the password.



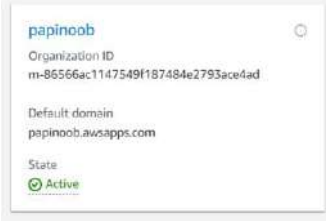
15. Next click on the Create User button.



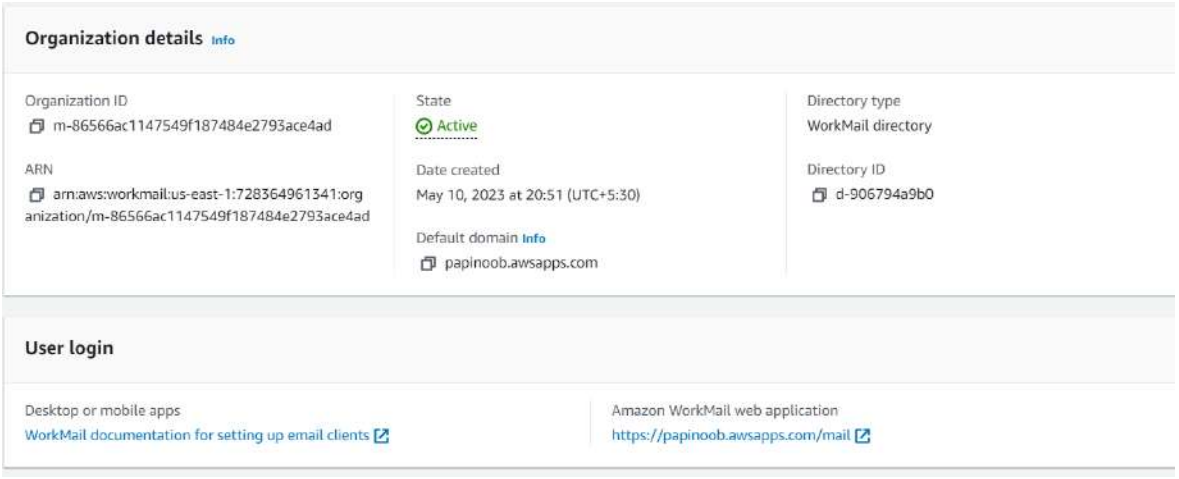
16. After the user is created return back to your organization page using the following link.



17. Click on the organization name.

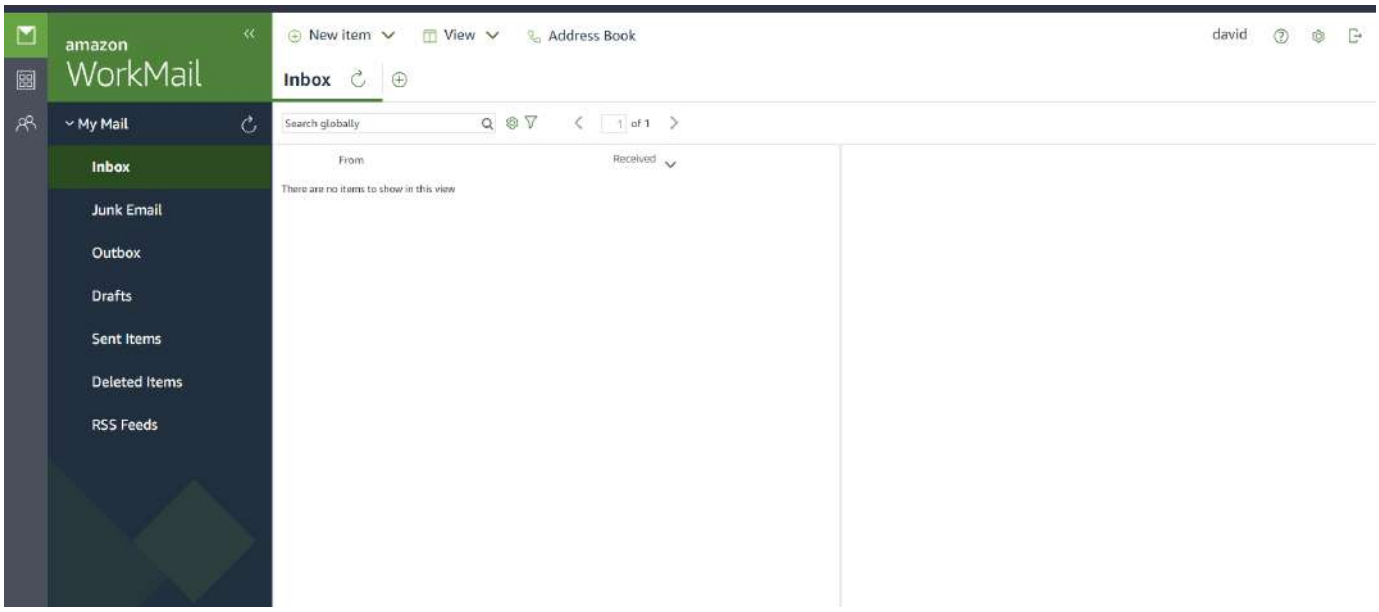


18. Now click on the Amazon Workmail web application link.

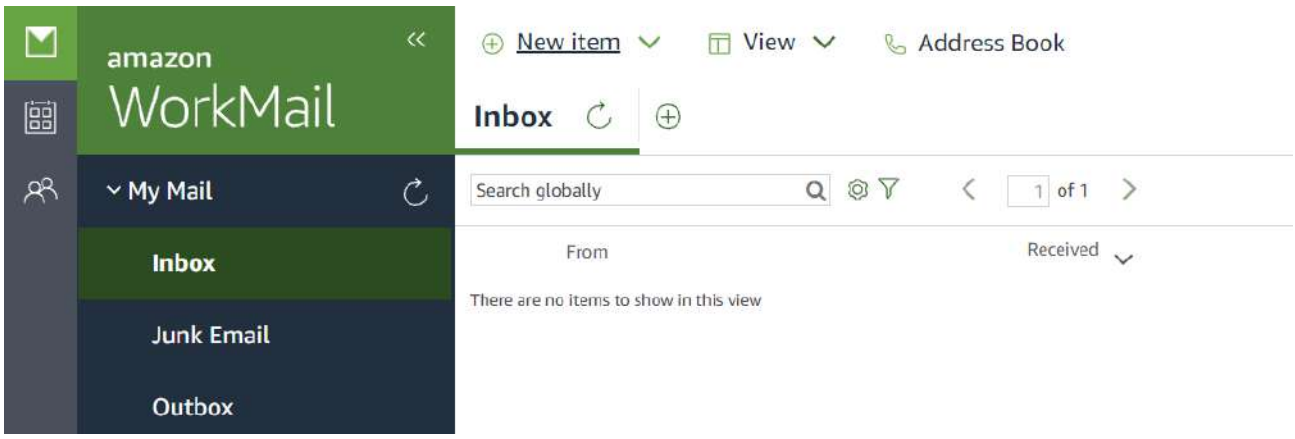


19. Enter the credentials of the user you just created in it.

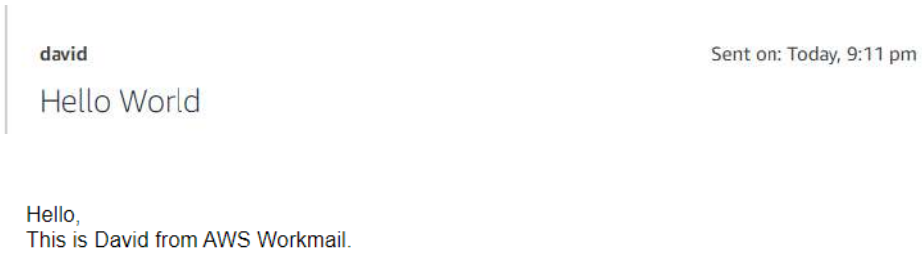
20. Our mail will be opened as the entered user.



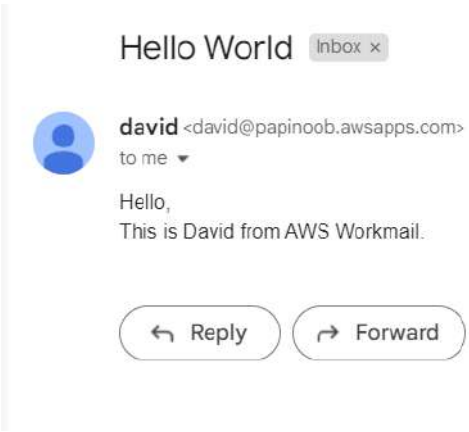
21. Now let's test our mail server by seeing whether it can send and receive mail. Send some mail to your Gmail from this by creating a mail using the new item option.



22. Select new email. Now write an email to your Gmail address.

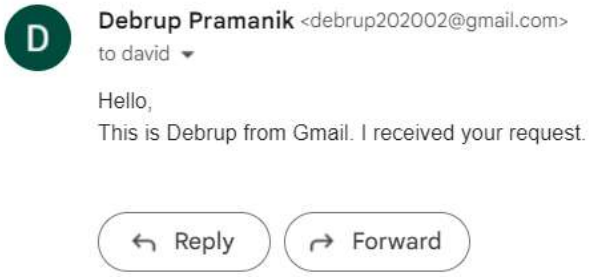


23. Check your Gmail.

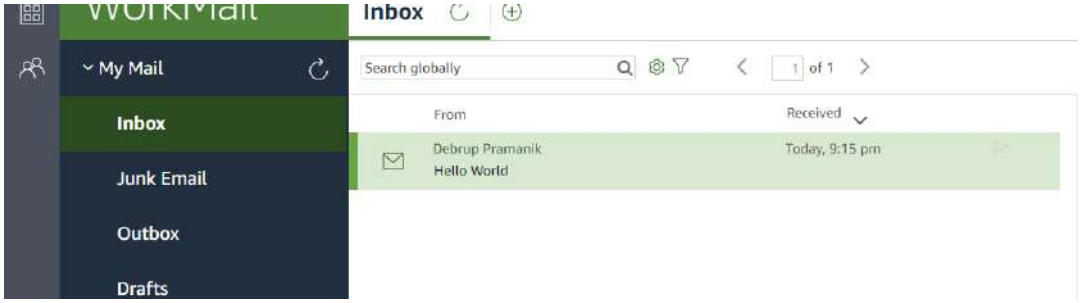


We received it.

24. Now send something back to the workmail.



25. Check your workmail if you have received the mail. It should be in inbox.



Hello World

To david

Hello,
This is Debrup from Gmail. I received your request.

We have successfully received a mail.

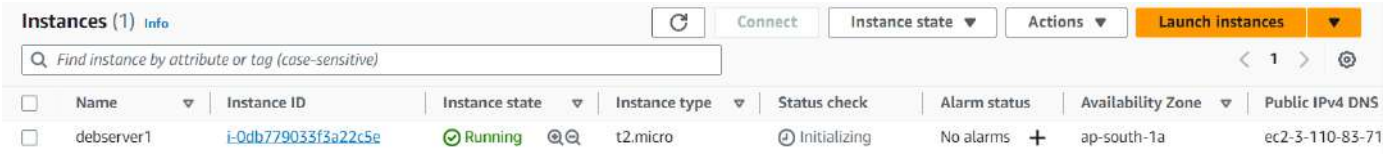
Hence our mail is working perfectly fine.

We have successfully created a workmail for our profile.

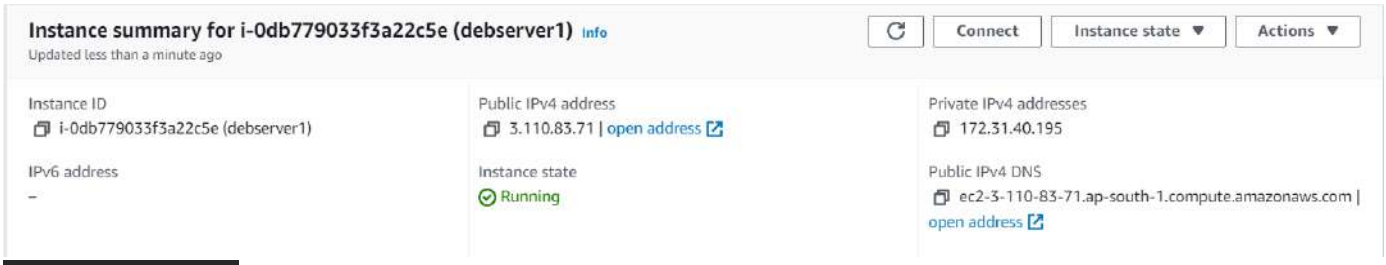
ASSIGNMENT – 14

Problem Statement: Create an elastic IP for an instance.

1. Sign-in to your AWS console.
2. Create an EC2 instance. (We do not need any user-data or any custom security group for this assignment)

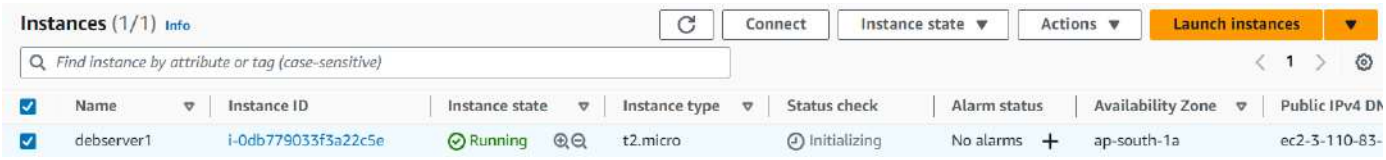


3. After the instance gets created click on it. Copy the public IPv4 address and paste it in a simple text file anywhere in your pc.

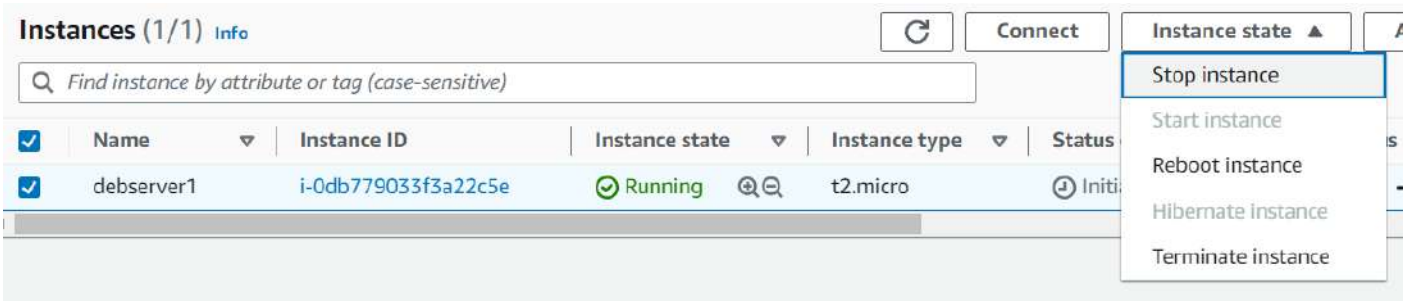



```
3.110.83.71
```

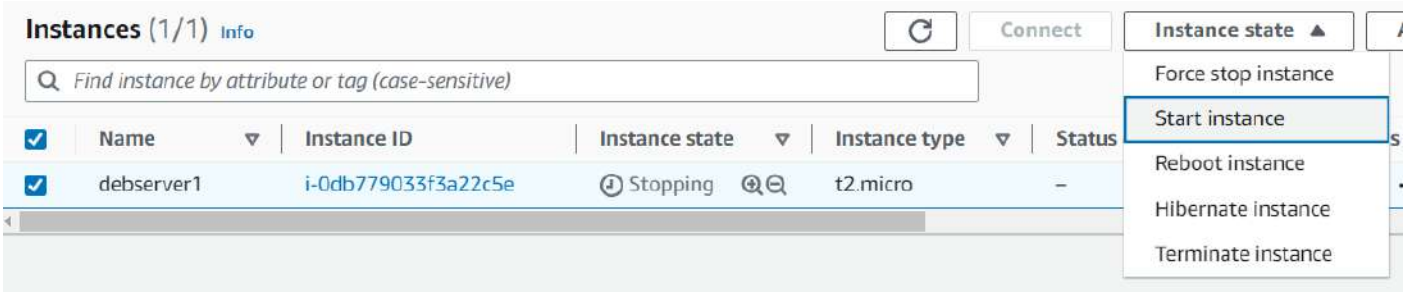
4. Now go back to the instances list and select our instance.



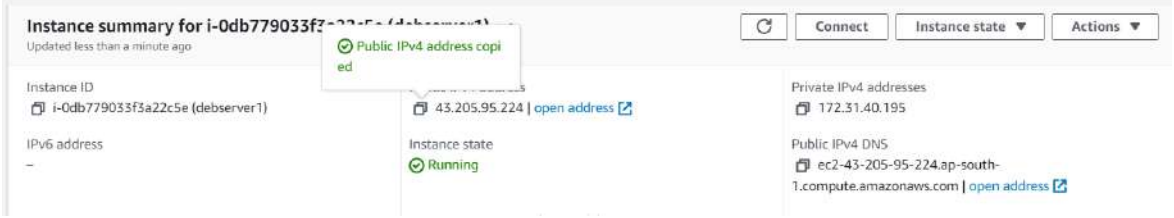
5. After selection click on the Instance state button and click on the Stop Instance option.



6. Wait for few seconds.
7. Now again select the instance and click on the Instance state button. Now click on the start instance button.



8. Click on the instance and copy the IPv4 address again and paste it in the same text file.

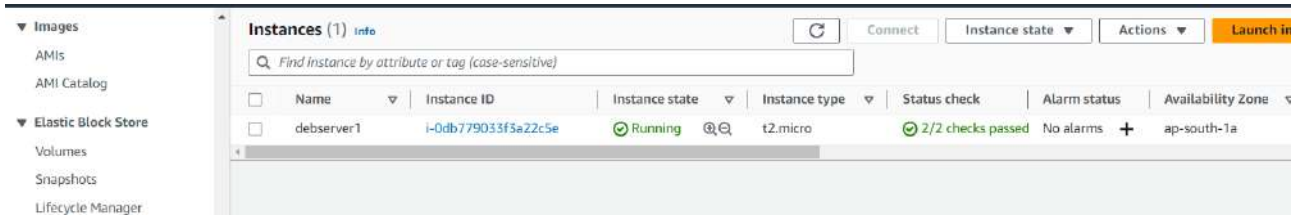


9. Now compare both the new and old IP address and notice that they are not the same.

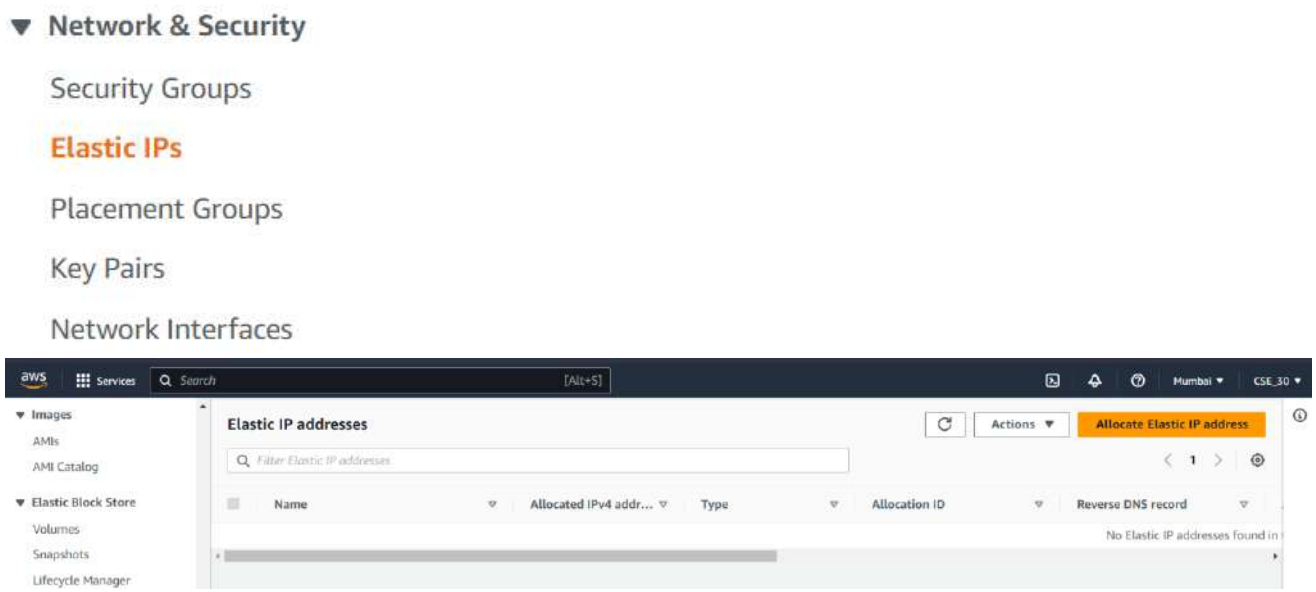
```
3.110.83.71
43.205.95.224
```

So even if we stop and restart our same instance it changes it public IPv4 address. This may not be desirable in some situations. So, to ensure that our instance does not change its public IPv4 address under any circumstances, we need to create an Elastic IP and associate/bind the instance to it. After that it will always be assigned the same Elastic IP as its public IPv4 address (static) all the time.

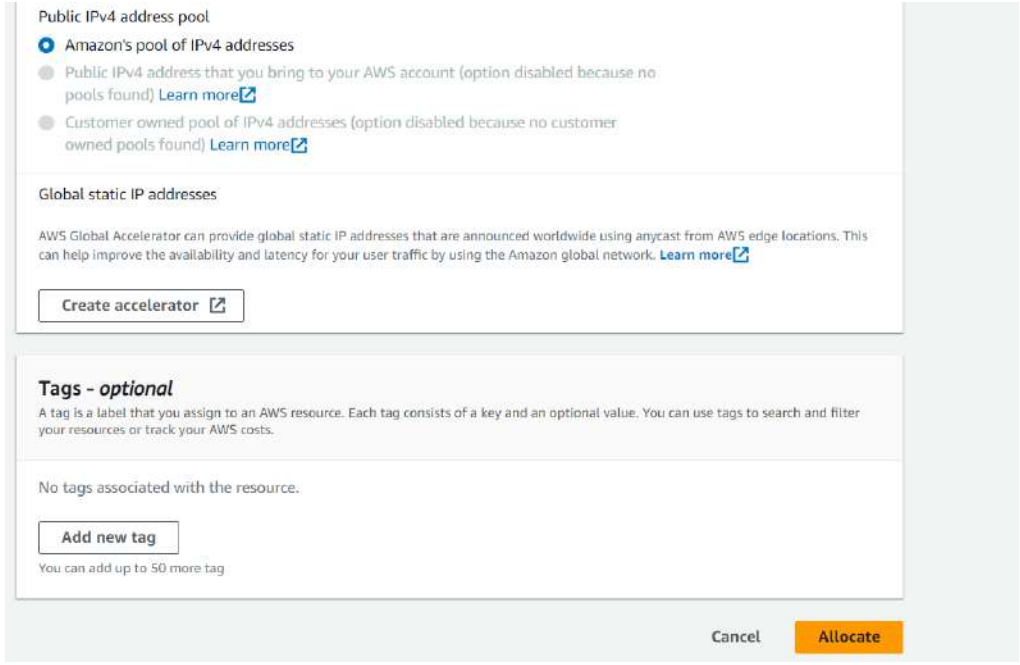
10. For creating an Elastic IP, we need to go scroll down the left side Nav bar and find the Network and security section.



11. Under it click on the Elastic IPs option.

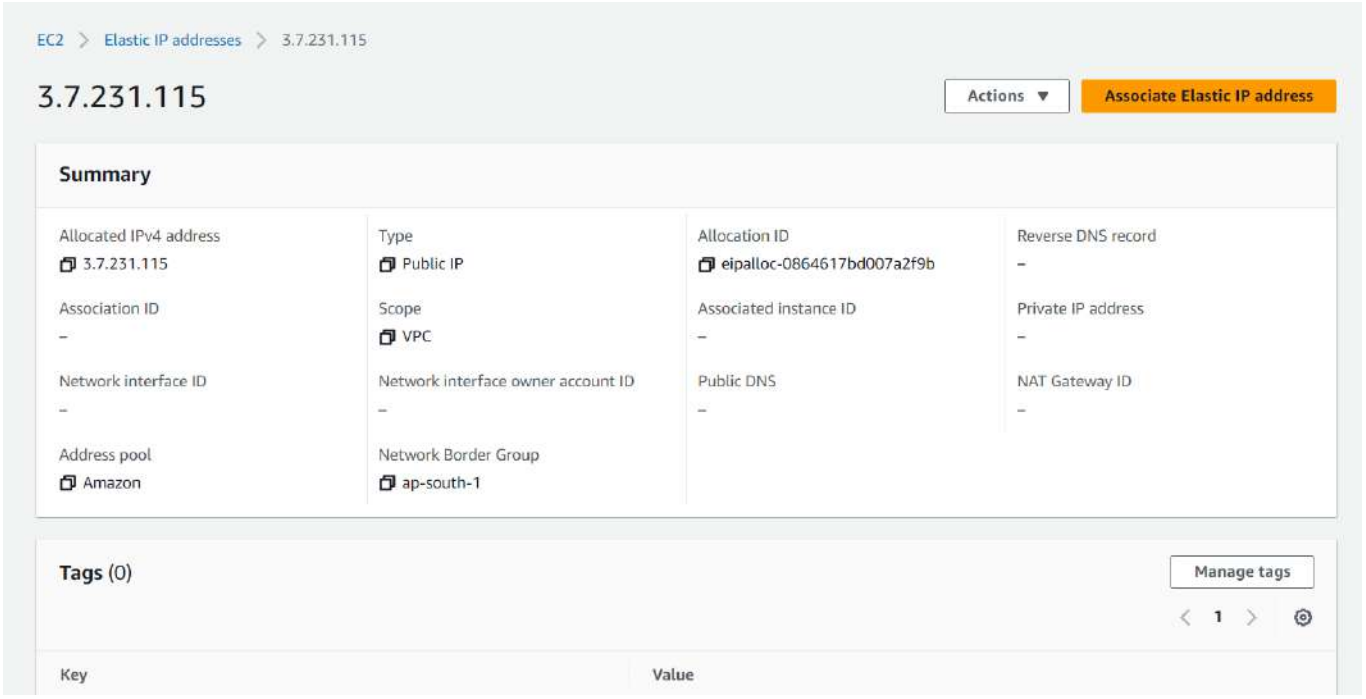


12. Now, click on the Allocate Elastic IP address button on the right side. No need to change any options. Just click on the Allocate button.



13. Now click on the Elastic IP address (in blue).

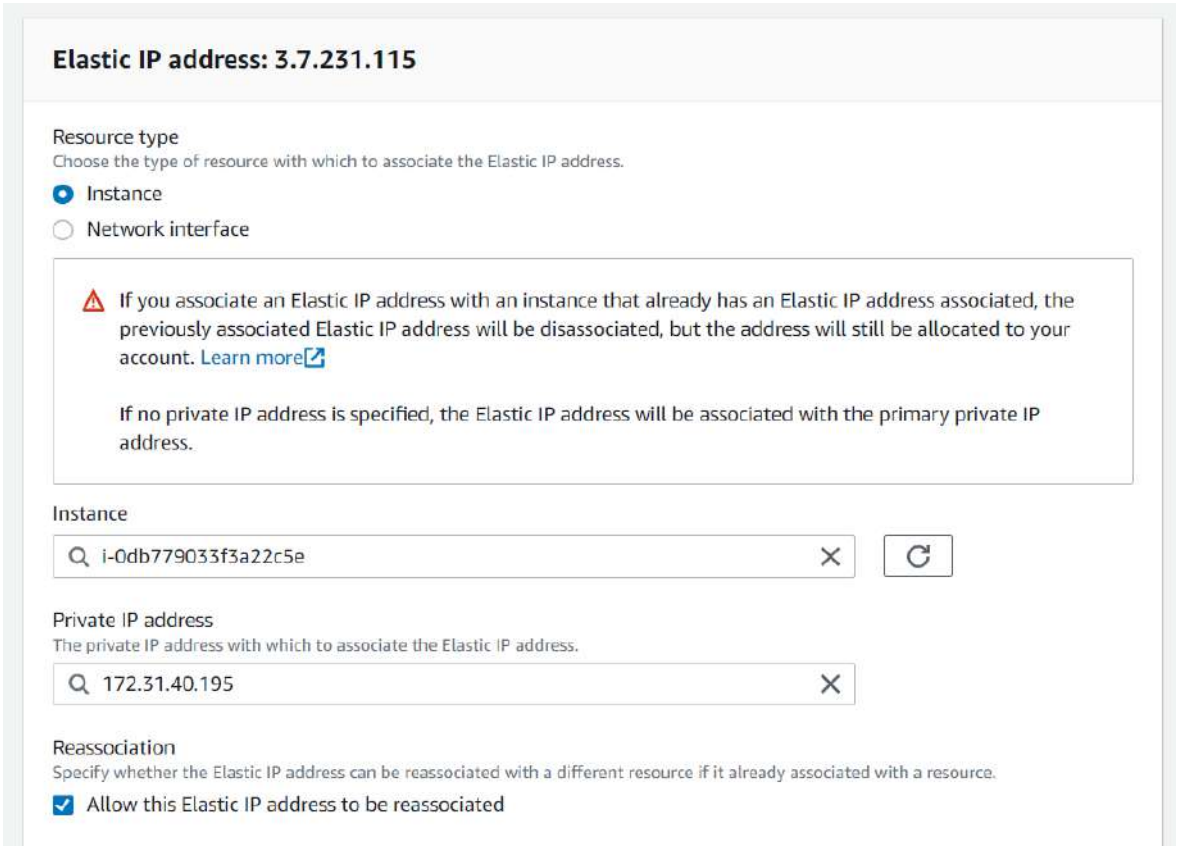
14. Next click on the Associate Elastic IP address button.



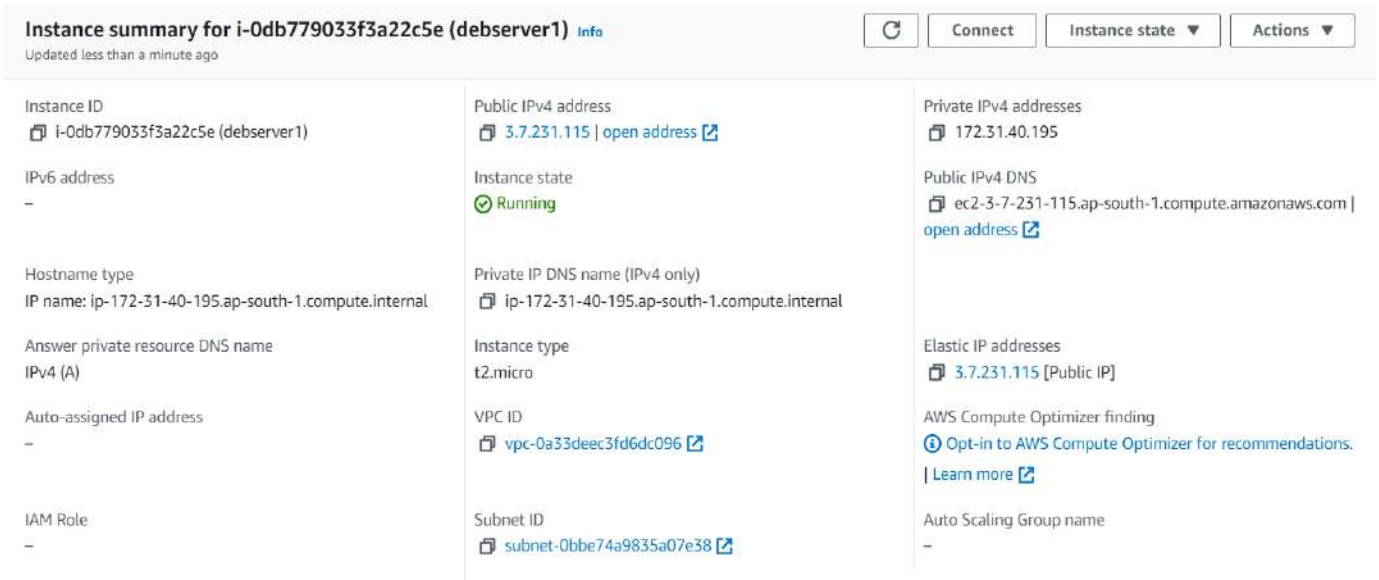
15. Choose your instance you want to associate with it.

16. Keep the Private IP address as specified in the dropdown when clicking for the Private Address.

17. Select the Allow Elastic IP to be reassociated option if we want to reuse it again for another instance.



18. Now click the associate button.
19. The Elastic IP should have been successfully associated with the instance.
20. To check it go back to the instances page. Click on the Instance and see the Public IPv4 address and the Elastic IP address. They should be same. Also notice that the public IPv4 address has turned into a hyperlink to the Elastic IP page.



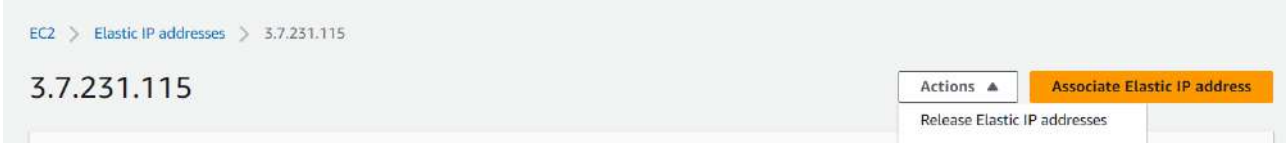
Now stop and restart the instance and see if the public IPv4 address changes or not. It will not change.

Hence, we have successfully created an Elastic IP for an instance.

To delete the Elastic IP, follow these steps:

1. Click on the Elastic IP.
2. Click on the actions button.

3. From the drop-down menu select Disassociate Elastic IP address. Then again click on disassociate on the pop-up.
4. Next again click on the Actions button and this time select Release Elastic IP address.

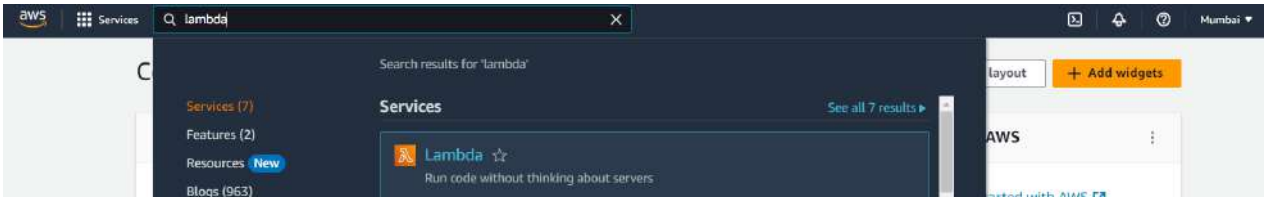


5. Now you can go back to your instance and see that the IPv4 address has already changed to a random one and it has no Elastic IP address associated with it. Now you can terminate the instance.

ASSIGNMENT – 15

Problem Statement: Create a serverless computing service.

1. Sign-in to your AWS console.
2. Search for Lambda



Click on the first result named Lambda.

3. Now click on the Create Function button on the top right corner.

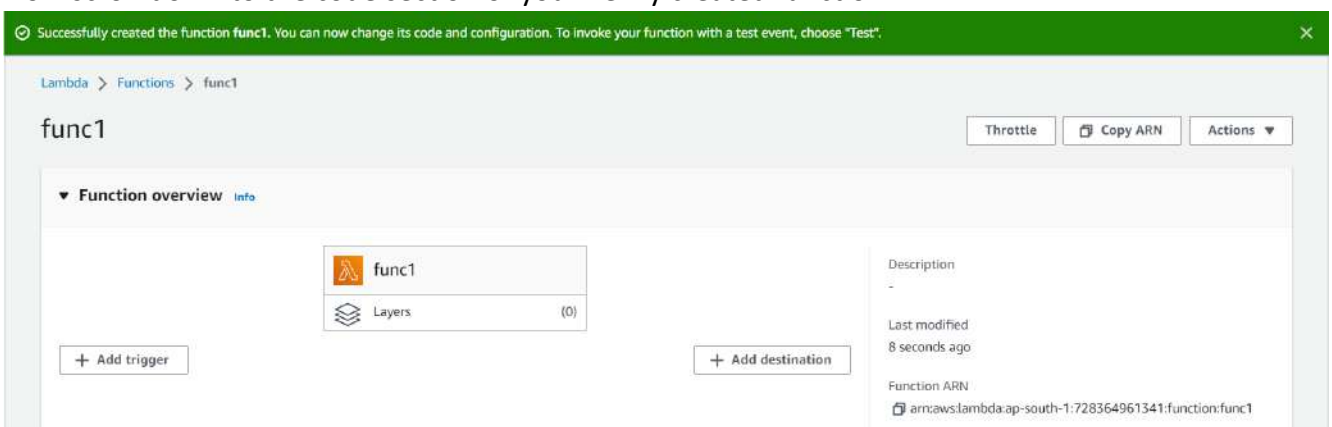
4. Select Author from scratch option.

5. Give the name of the function.

6. Choose Node.js as the Runtime. (No need to change architecture.)

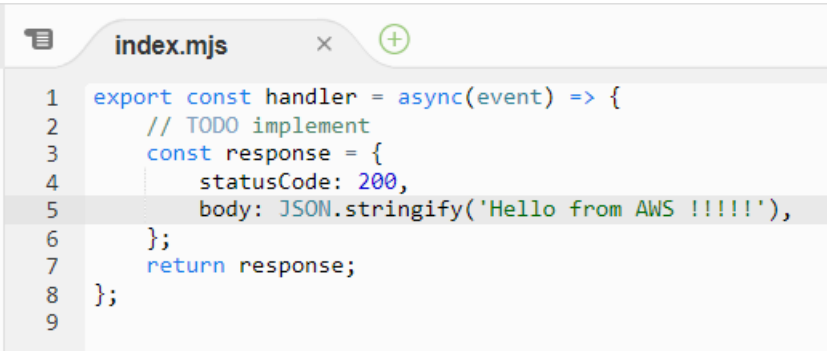
7. Now click on the create function button

8. Now scroll-down to the code section of your newly created function.

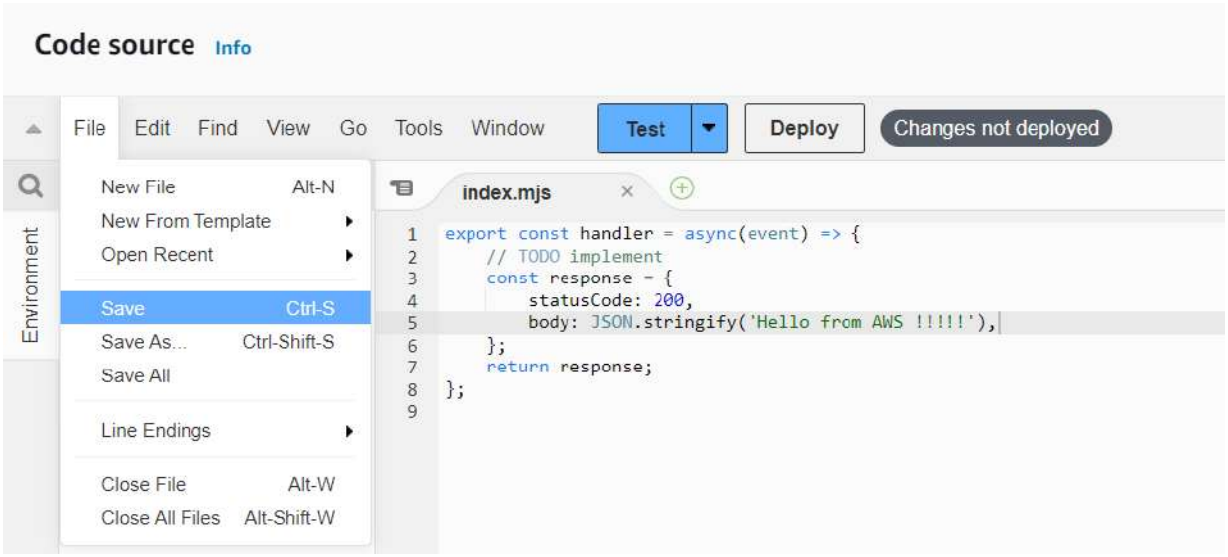


9. Change the string in the code to be displayed.

We changed it to this……

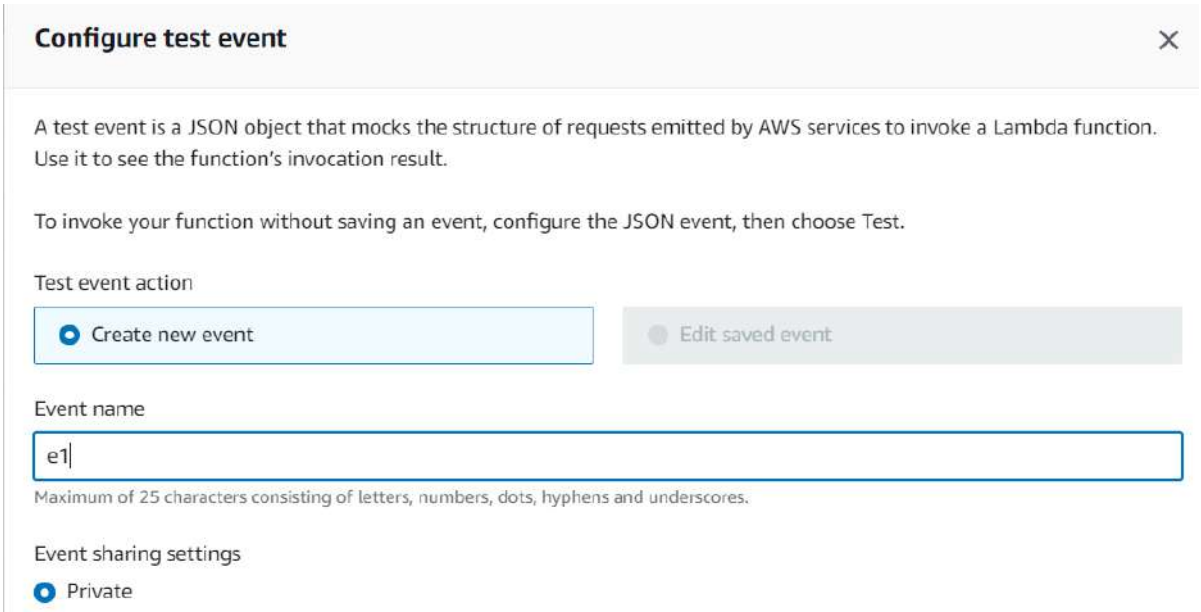


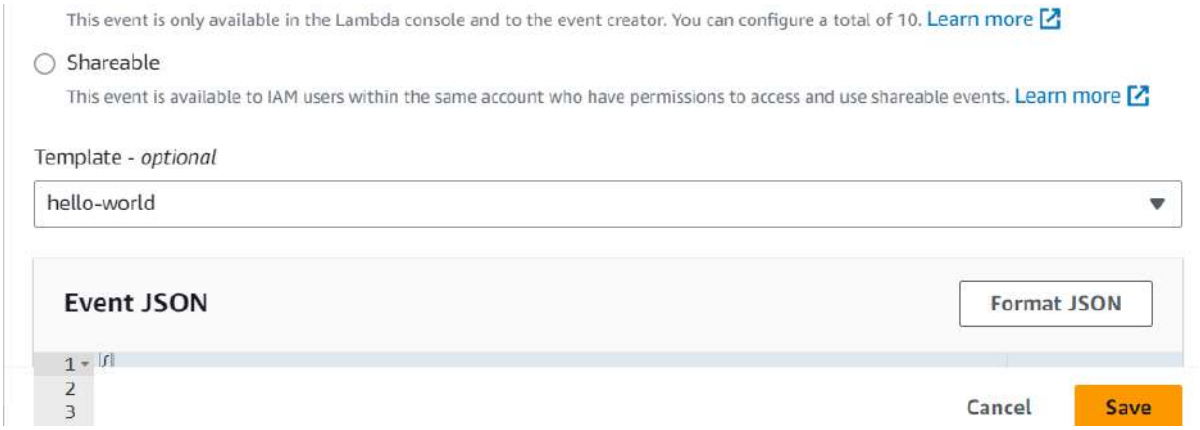
10. Now go to File option and click on save to save the changes.



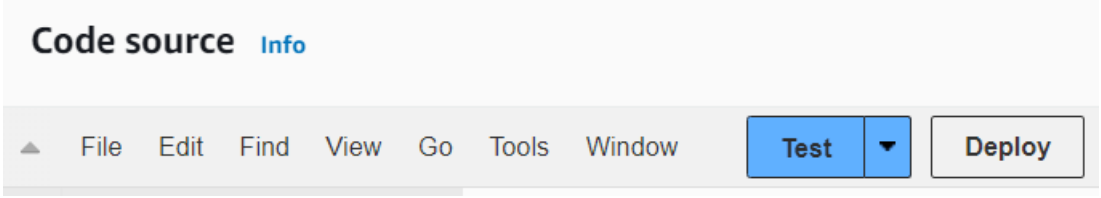
11. Now click on the Test button.

12. Select Create New Event. Then give a name. Then click on save.



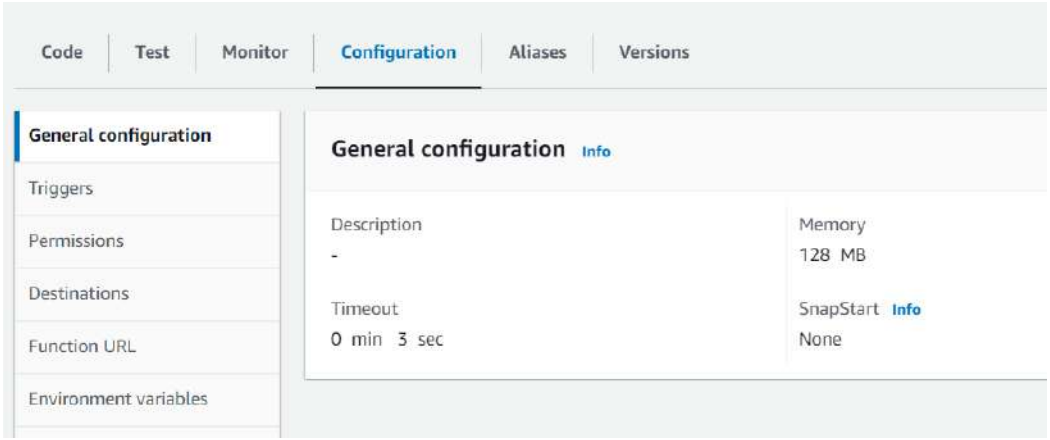


13. After saving. Now click on the Deploy button.

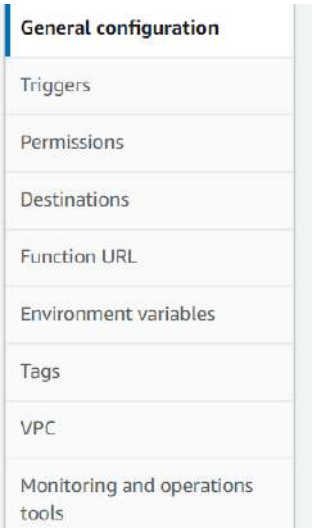


14. After successful deployment a message will pop as mentioned below and the deploy button will be locked out, indicating that our Function has been successfully deployed.

15. Now go to the configuration tab.

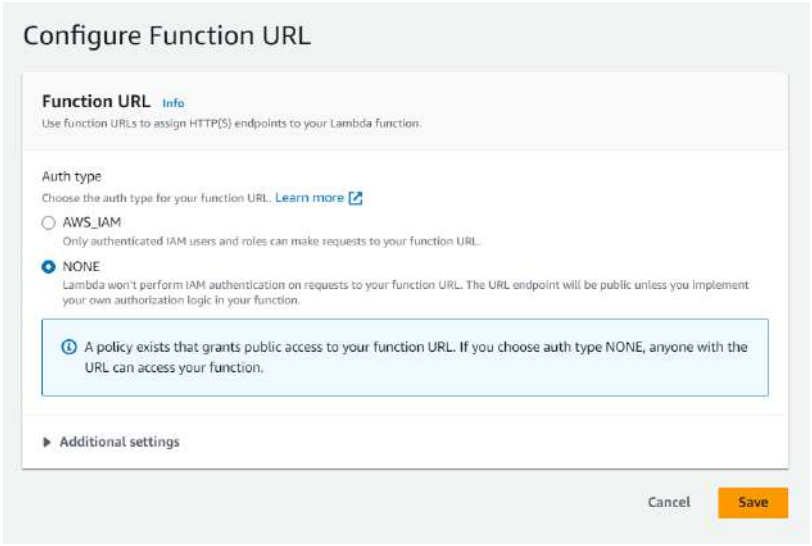


16. Click on the Function URL option in the left side Bar.

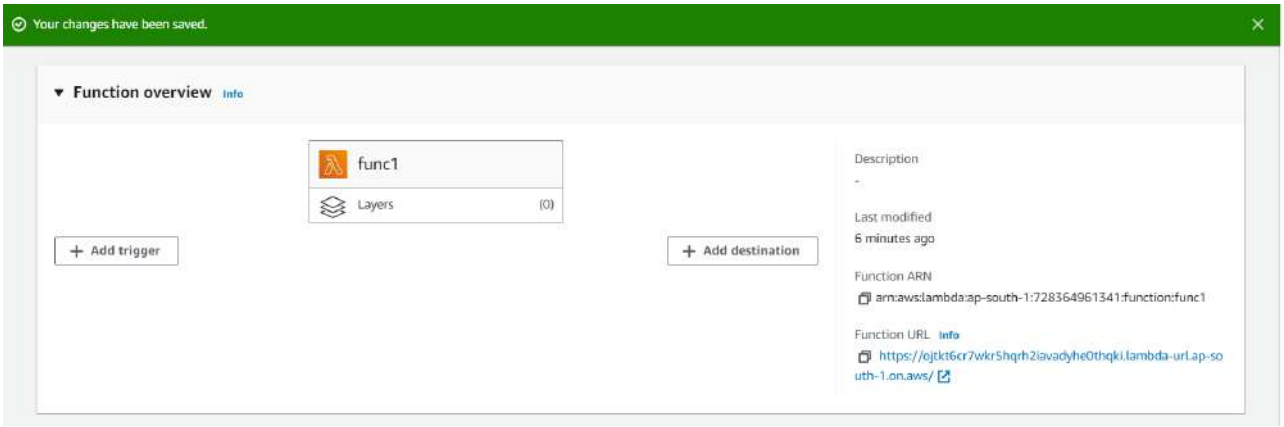


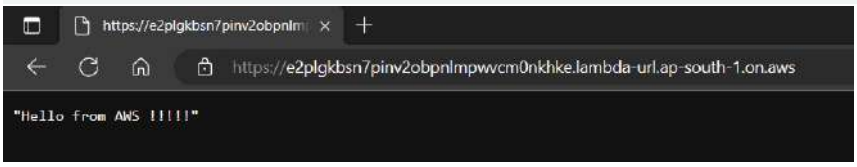
17. Click on Create function URL.

18. Choose NONE and click on the save button.



19. Now copy the newly created Function URL and paste it in a different browser.

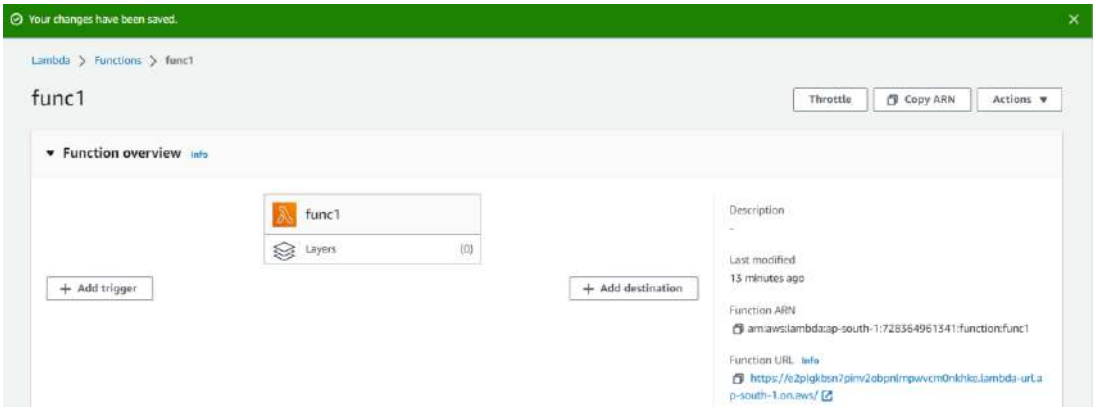




We have successfully Created a Serverless Computing service.

To delete the Lambda Function, follow these steps:

1. Click on the Actions button on the top right side.

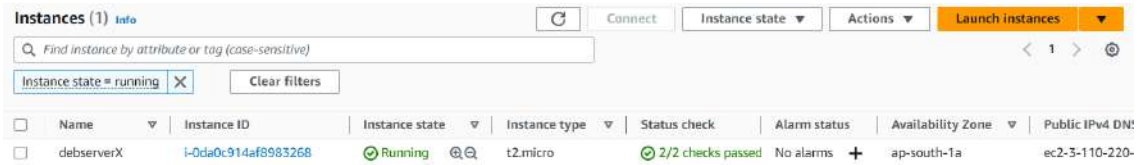


2. Select the Delete function option and then click on delete button in the pop-up.

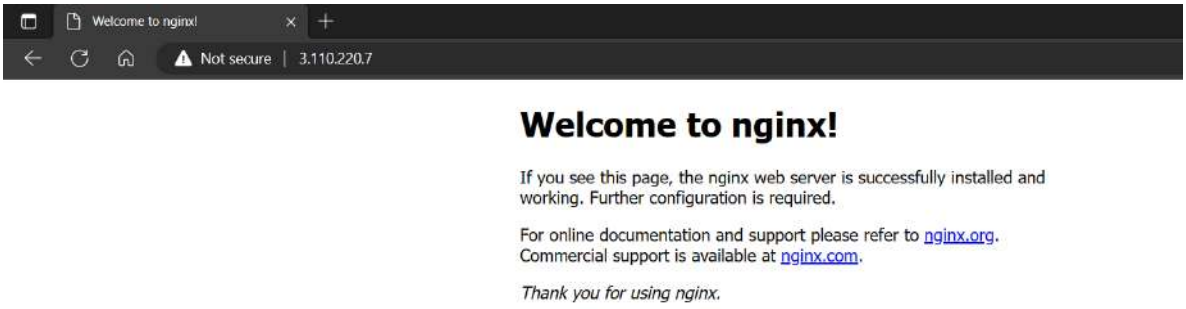
ASSIGNMENT – 16

Problem Statement: Manage Amazon DNS service and run a project using domain-name and URL

1. Sign-in to your AWS console.
2. Create an instance with custom security group and user data (**Refer Ass10**)
3. Click on the instance



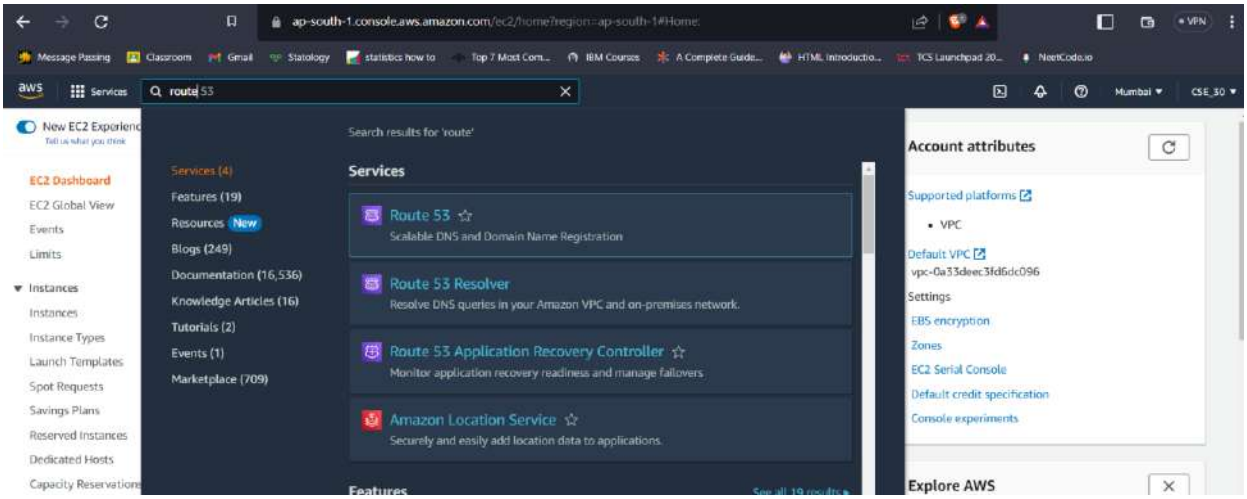
4. Copy the public IPv4 address and paste it in another browser.



5. Check if the project webpage is accessible by appending :4000 to your address.

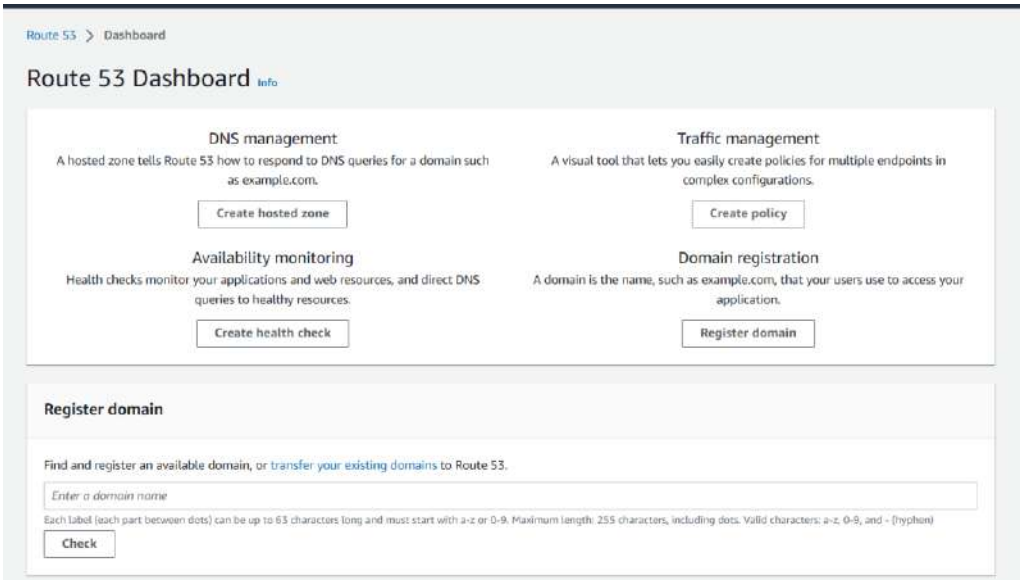
Our EC2 instance works as intended. However, to access our webpage one always requires the public IPv4 address of our server instance which is very complicated/less accessible for end-users of our webpage/web application.
So, to make it easier for our end-users, we need to bind a domain name to the server instance. Now anyone can use the domain name and the URL to access our project.

6. Search Route 53 in the search bar of AWS console. Select the first result.



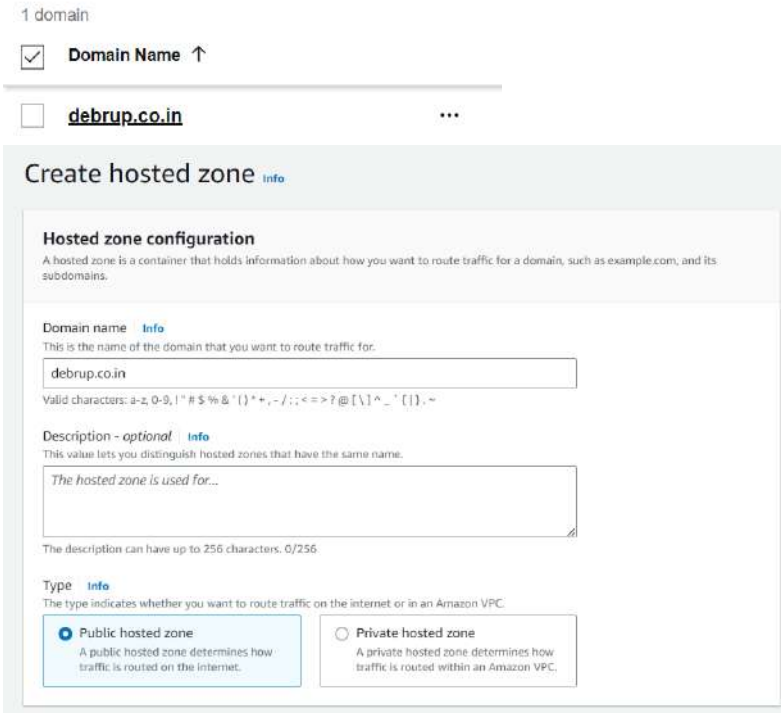
We require a registered Domain name for this assignment. So, after obtaining one (free or paid) go to the Webpage of your Domain provider and log-in to your account where you can find all the details of your purchased Domains.
This may vary from site to site, so you will have to do this based on what site you are using.
We (for now) will be using GoDaddy.com, because we have purchased a Domain from them.

7. After Reaching the Route 53 dashboard click on the Create Hosted Zone button.



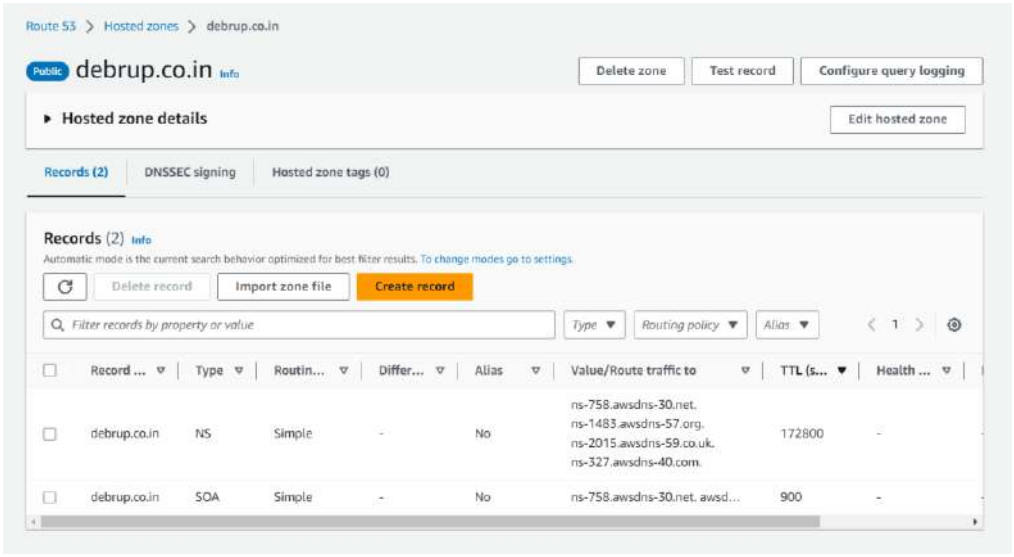
Alternatively, you can go to hosted zones from the left-side bar and then select create hosted zone option.

8. Now, copy your Domain name from your Domain providers website. Here we used GoDaddy.com. Paste the domain name in the given field in Hosted Zone configuration page.



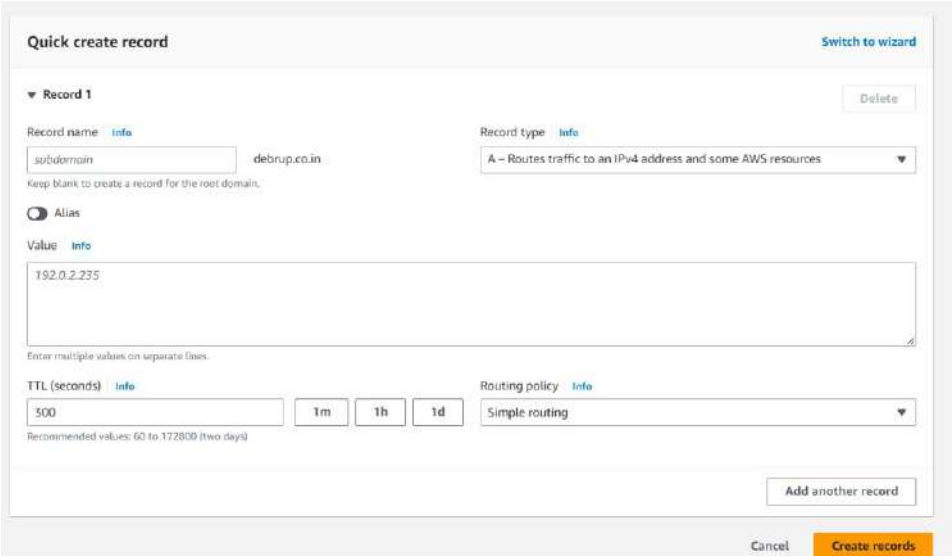
9. Now scroll-down and click on the Create Hosted Zone button.

10. Now click on the Create a record button.

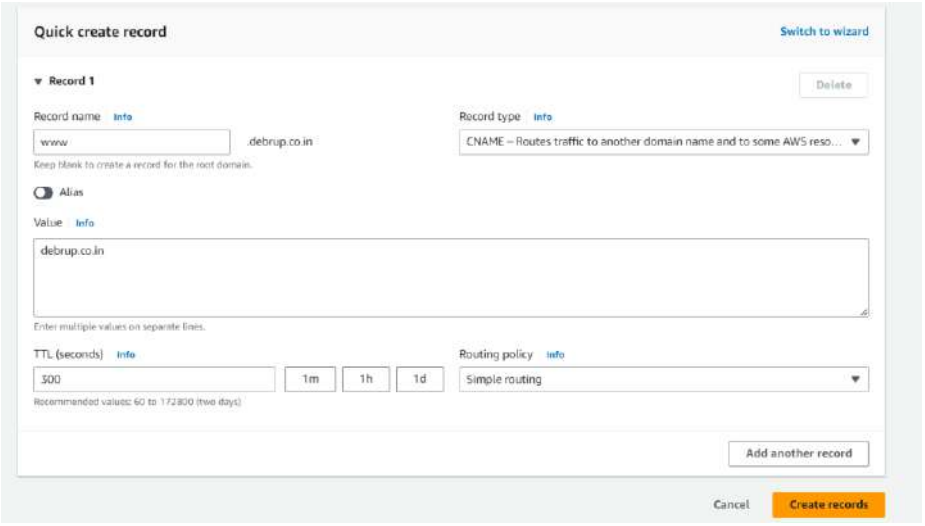


11. Follow these Steps:

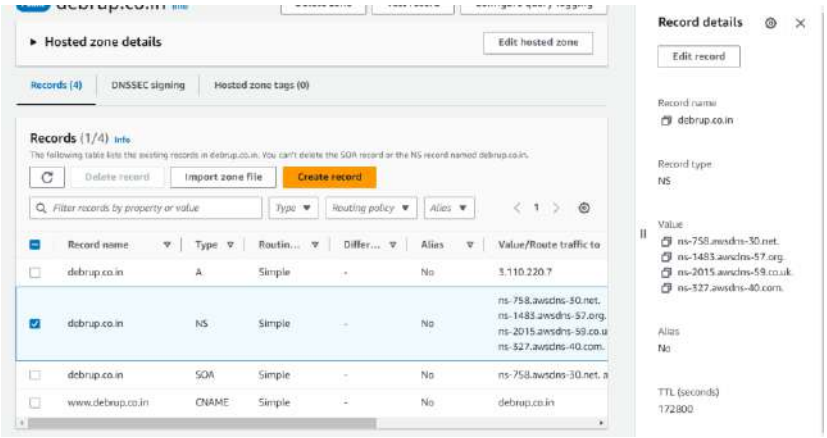
a. Do not give any name. Keep the record name blank.
b. Keep record type as it is. No change required.
c. Under the value, copy and paste your server instance public IPv4 address which you want to route to using your DNS.
d. Then click on create records button



12. Now again click on the Create Record button like the previous step.
a. But this time give the record name as → **www**
b. Select Record type as CNAME
c. In the text box under value, write the full domain-name there. (For example: example.com)
d. Click on create records button



13. Now select the record with type nameserver (NS).

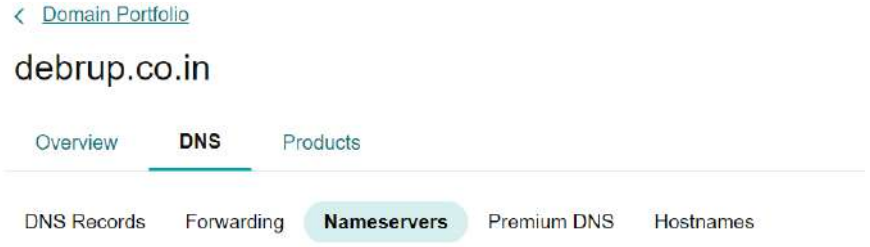


The values seen on the right-hand side are required for the next steps.

14. Now go to your Domain providers webpage. Go to your purchased Domains settings.

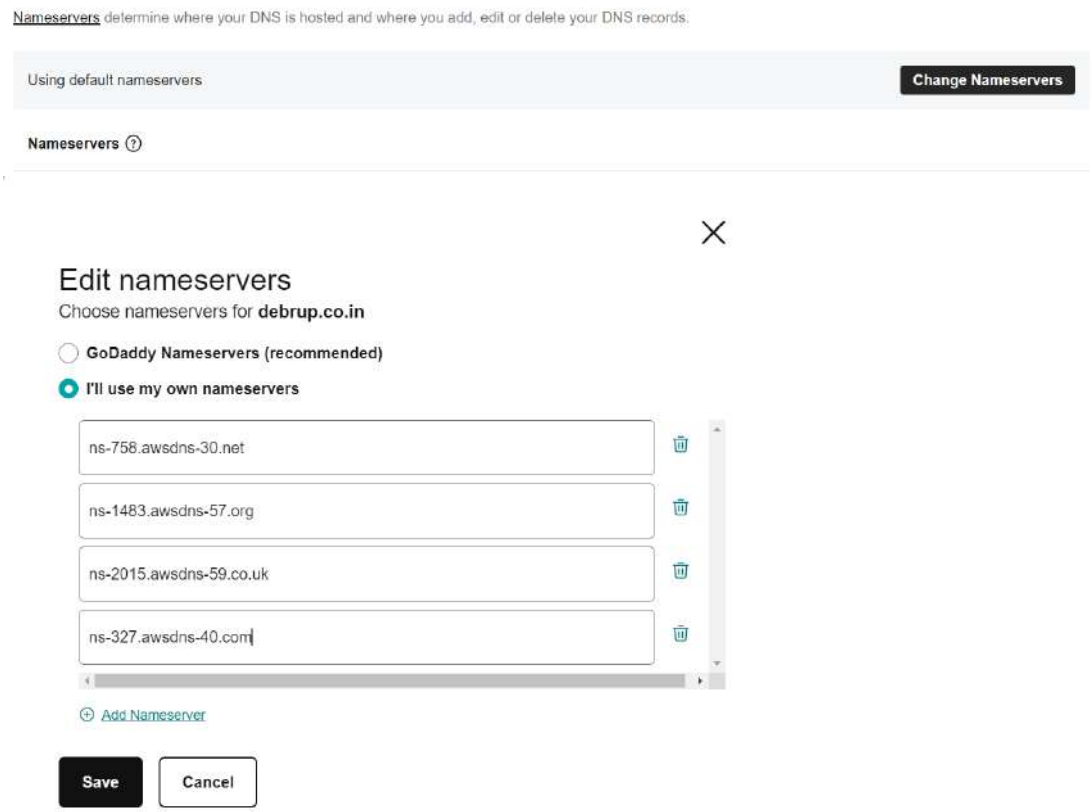
15. Click on DNS section. (This may vary from provider to provider)

16. Click on the nameservers option.



17. Click on the Change nameservers and add here all the values opened in the Route 53 page.

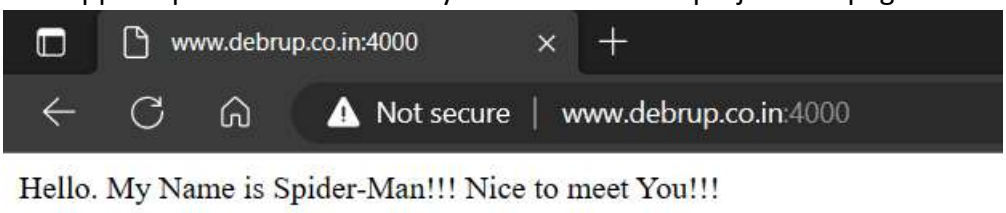
a. Select use my own nameservers option.
b. Add nameservers.
c. Then click on the save button.



18. Wait for few minutes.

19. Now try searching from any browser using your domain name with www.
(For example: www.example.com)

20. Also append port no. like we always do to access our project webpage.



We have successfully run our project using our custom domain-name and URL.