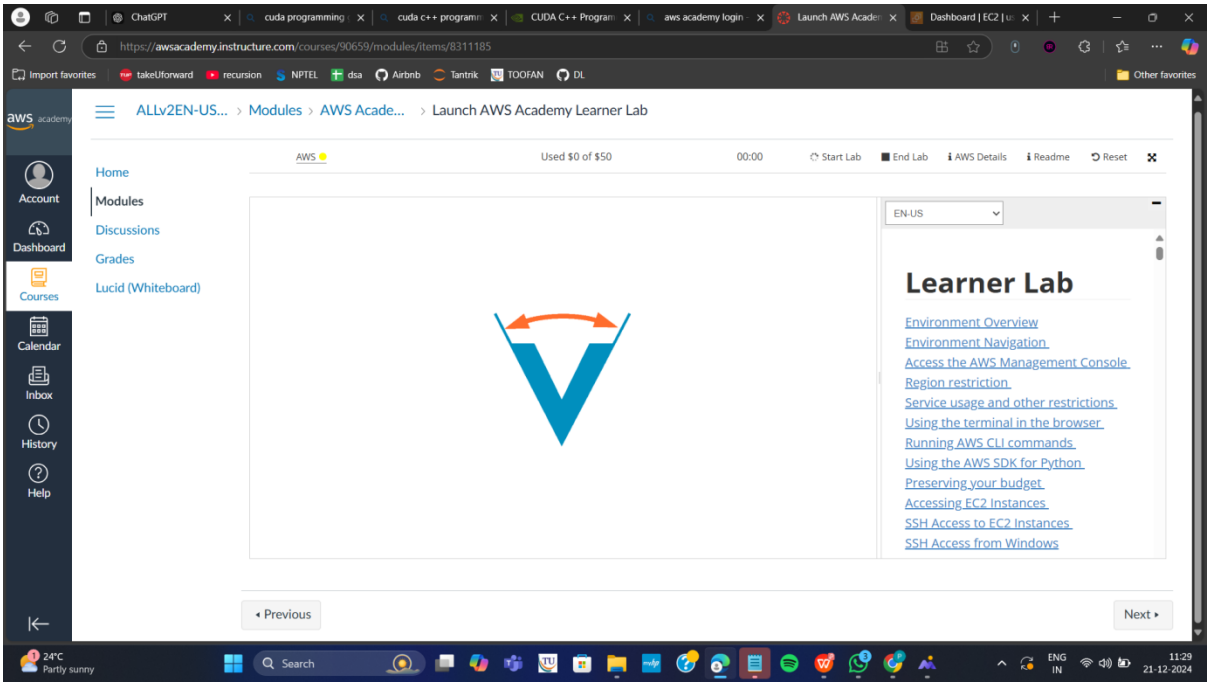
WEEK-12

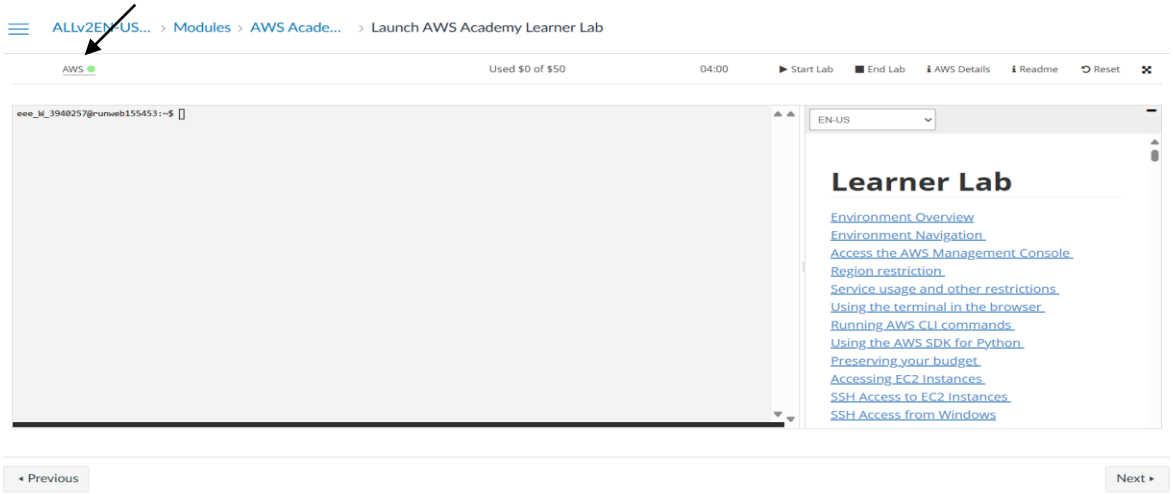
1 DEPLOYMENT OF INDEX.HTML USING EC2 INSTANCE in AWS

Step 1: Login to AWS /canvas account

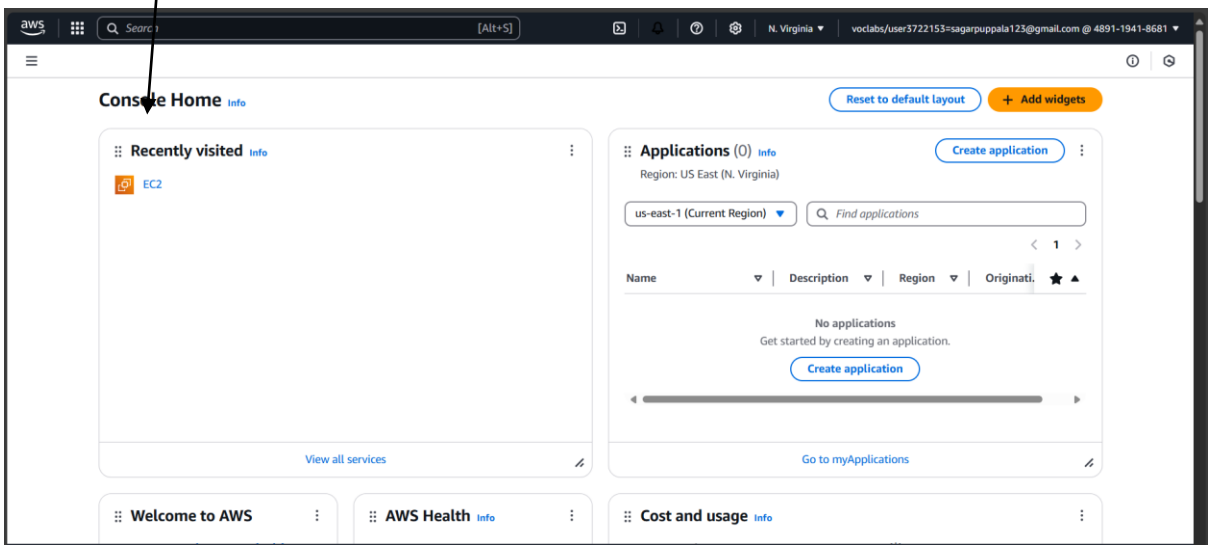
- Go to course invitation mail and click on start
- Opens the aws academy, select the student login and enter the email and password details
- Click on Modules
- Scroll down and select Lunch AWS Academy Lab-
  - Click on start Lab and wait AWS becomes from red to green AWS



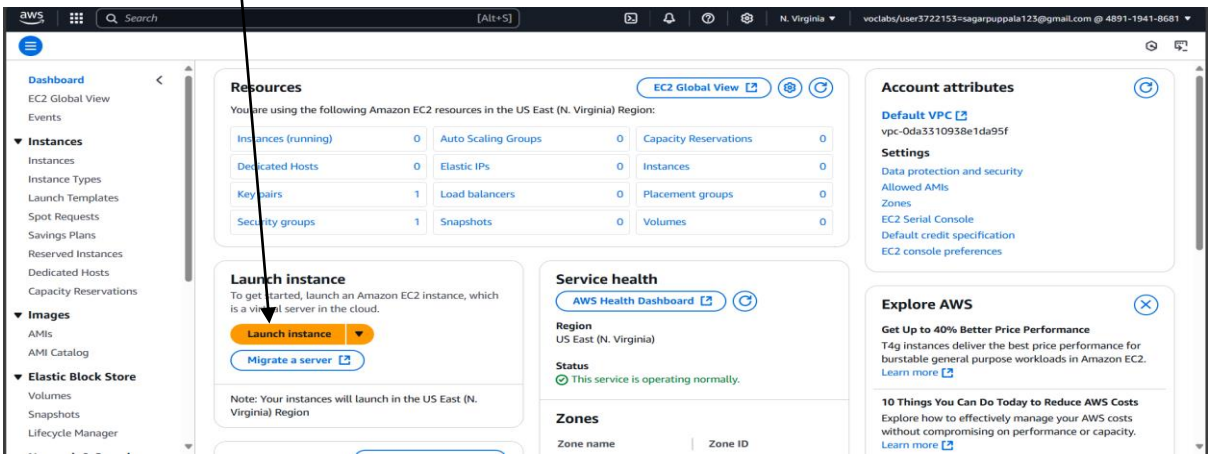
Click on AWS



Step 2: Click on EC2 to create instance



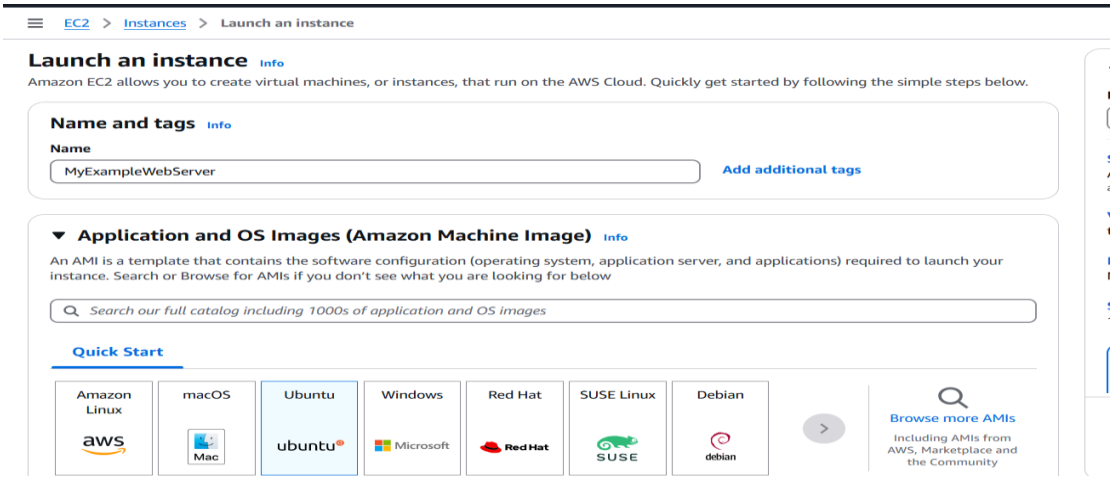
Click on Launch Instance



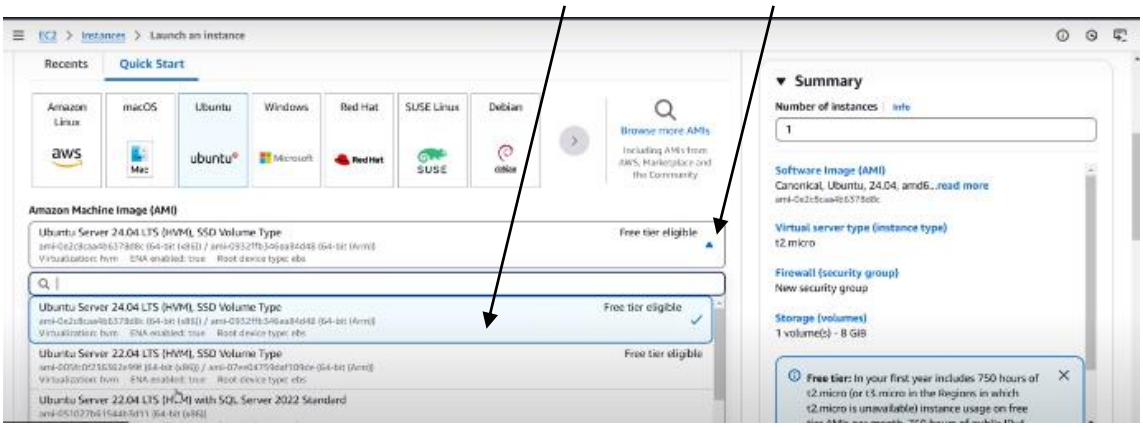
Stage 1 --Name (Giving name to the machine) ubuntu

Stage 2 -- Select AMI (Note: Select free tier eligible) ubuntu server

Give name and select ubuntu under application



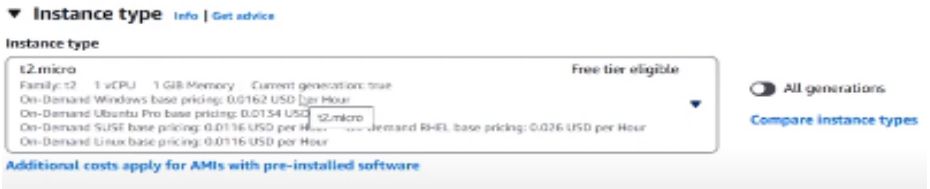
Select this free tier option of Ubuntu from here



Stage 3 -- Architecture as 64-bit

Make sure AMI and Architecture are there as shown

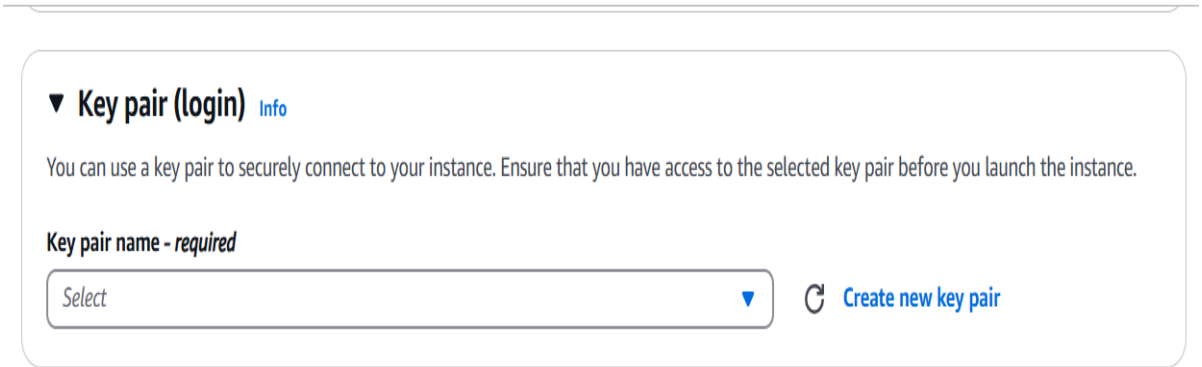
Stage 4 -- **Instance type** ---- t2.micro(default 1 CPU,1 GB RAM)



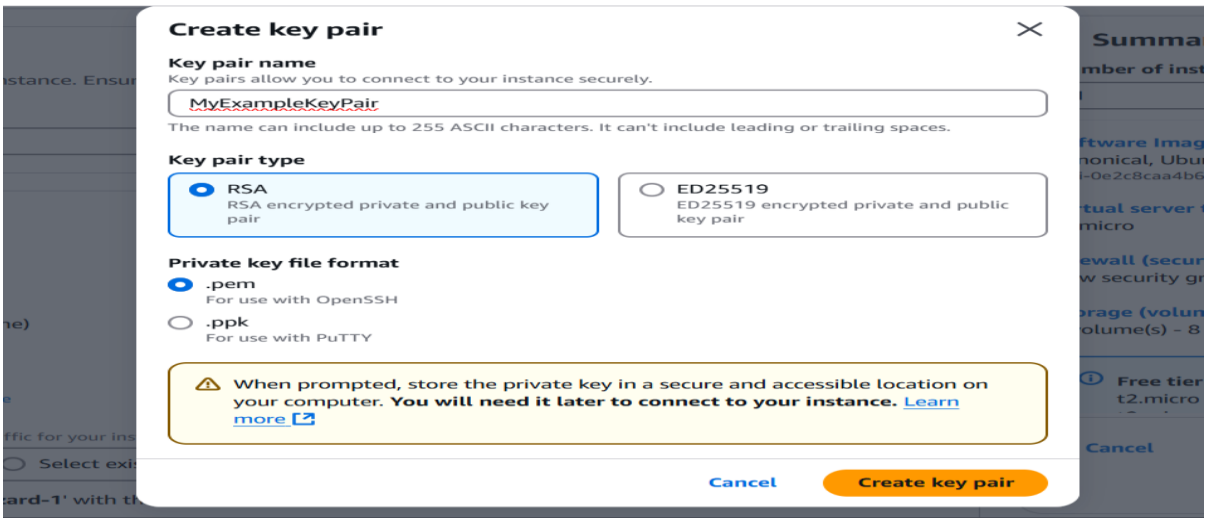
Stage 5 -- Create a new keypair---a keypair will downloaded with extension .pem

Store key in folder AWS

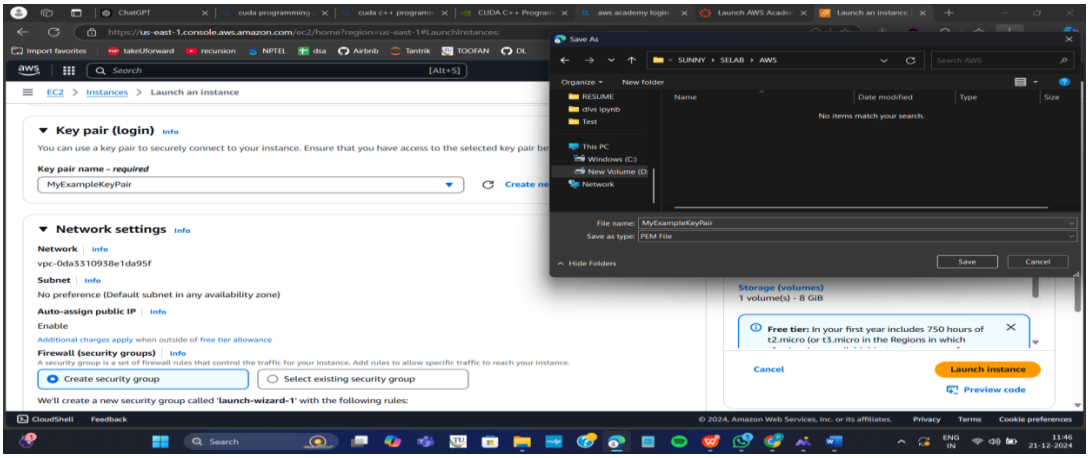
First Click on create-new key pair



Give KeyPair name and click on create key pair



Save the .pem file in folder AWS on desktop

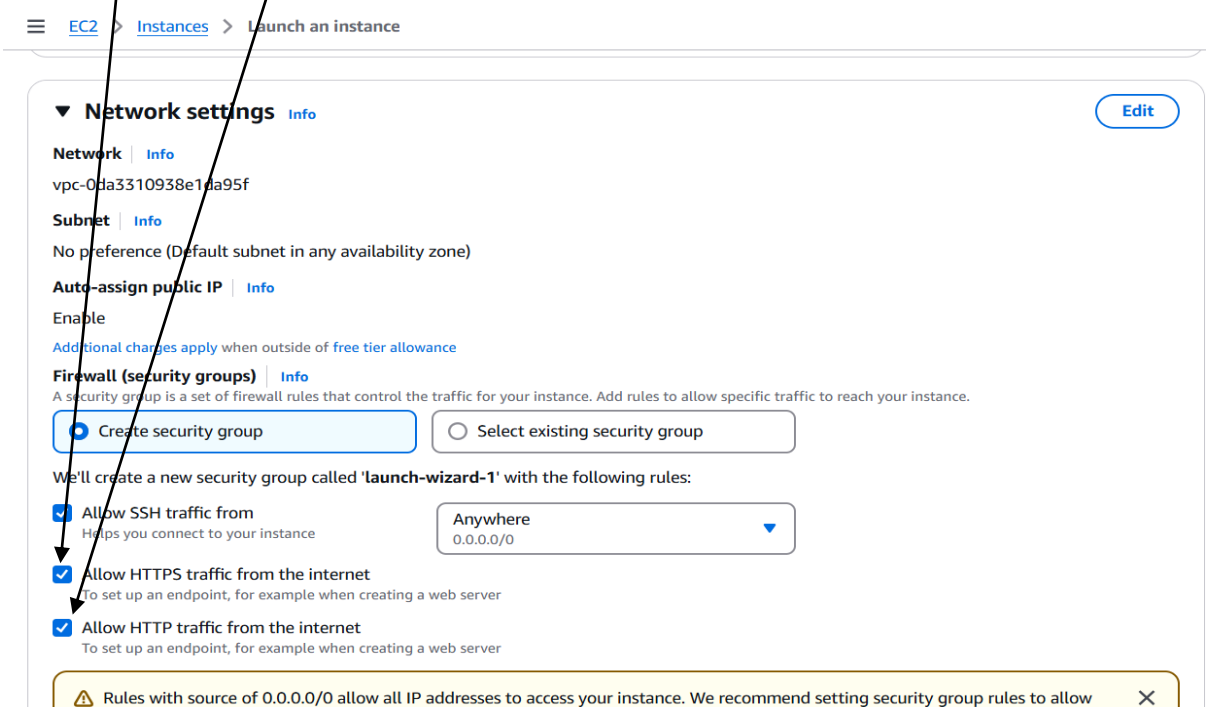


Stage 6 -- Network Setting ----Create Security group -- (It deals with ports)

(Note for understanding We have 0 to 65535 ports. Every port is dedicated to special purpose)

HERE select https and http (which allow to load your web pages while execution)

In network setting check all the checkboxes

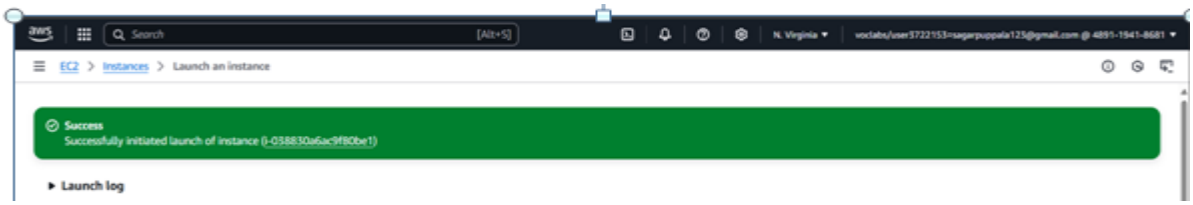


Stage 7 -- Storage - 8GB (Observation - we have root - it is same as C Drive) it default 8 GB

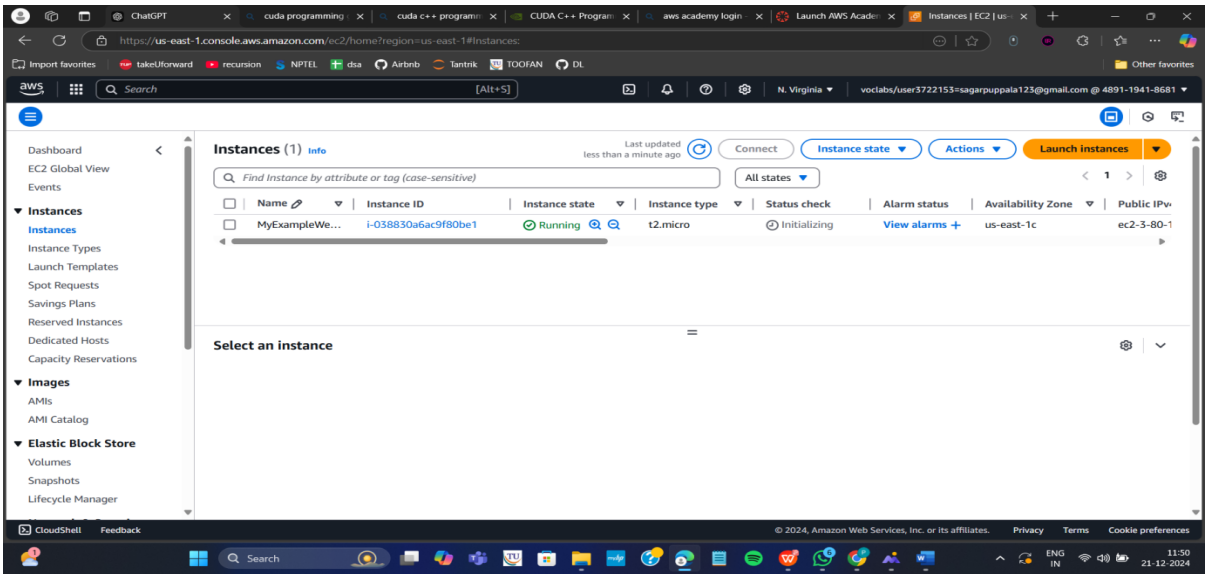
Stage 8 --- click on launch instance

Now click on the instances

Wait for the Success message to apper in green color



You can see MyExampleWebServer is Running and wait for it to initialize

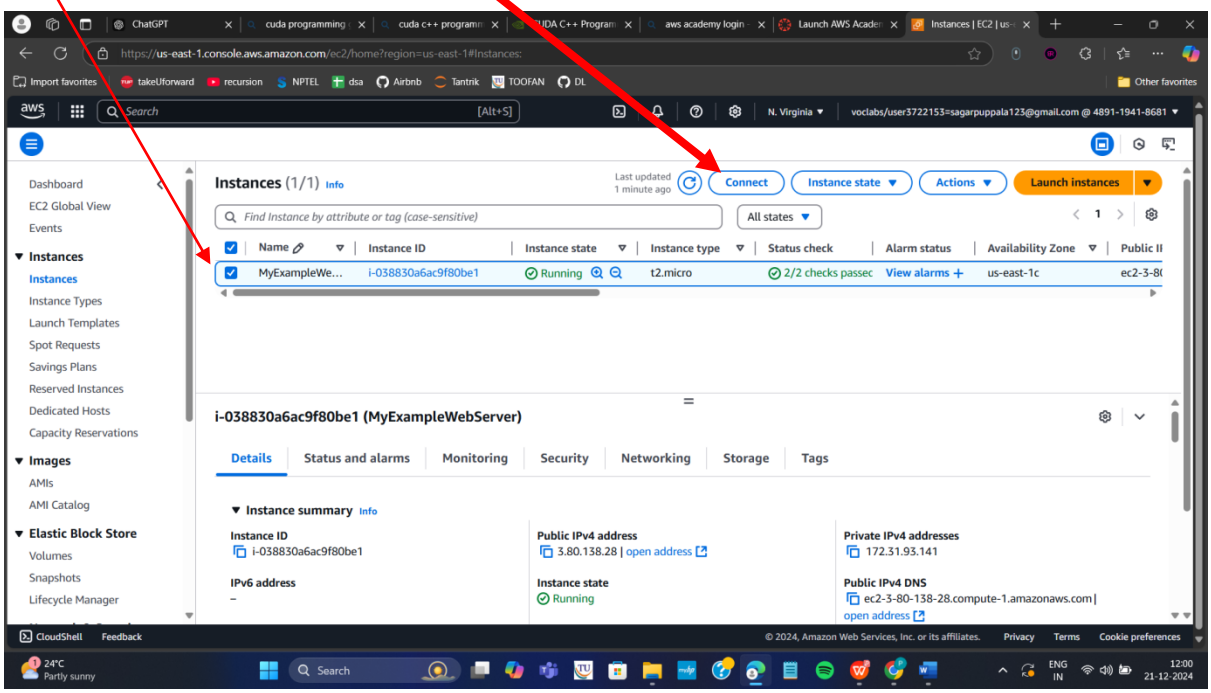


You have to get 2 tests passes.

Important:---- Now check the box and click on connect.

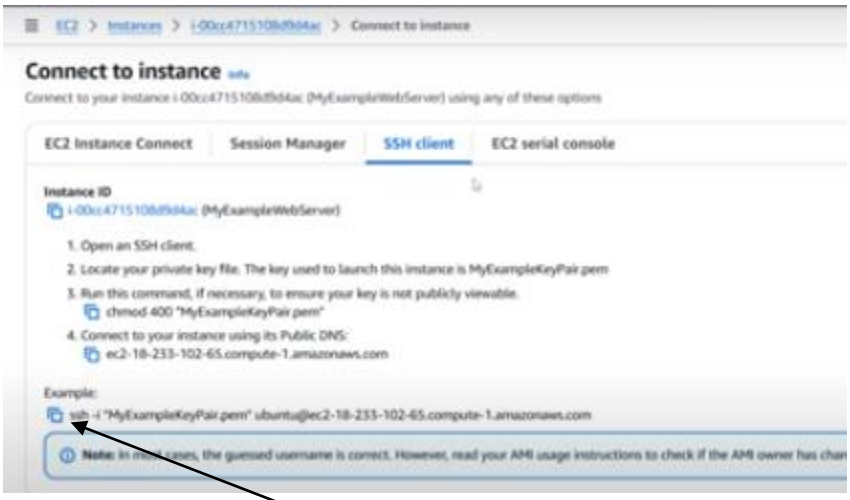
Do this step:---once it is created select that instance

(Tick in checkbox) and click on connect



Thus your EC2 instance is running on server.

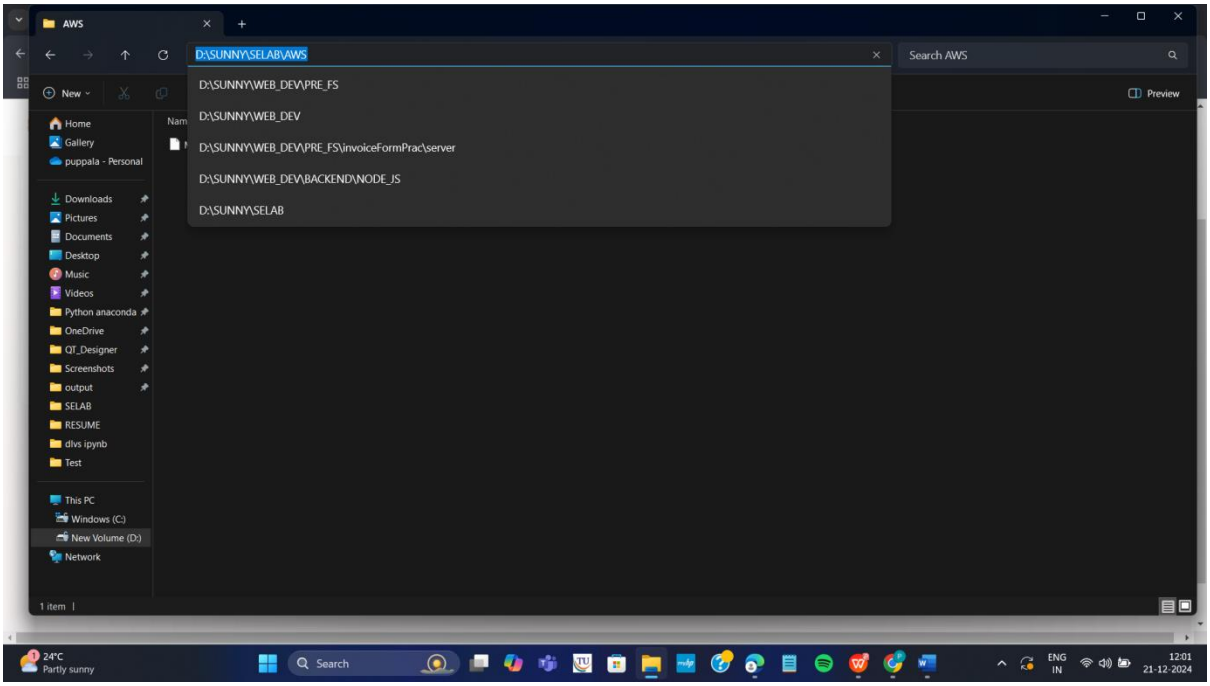
Step 4: Now you can connect local system to server (EC2 instance) using secure shell SSH.



For this, **Here copy the ssh – i command from SSH client connect tab and paste later**

Next We can use powershell /gitbash /webconsole **in administrative mode**, to connect to ubuntu machine.

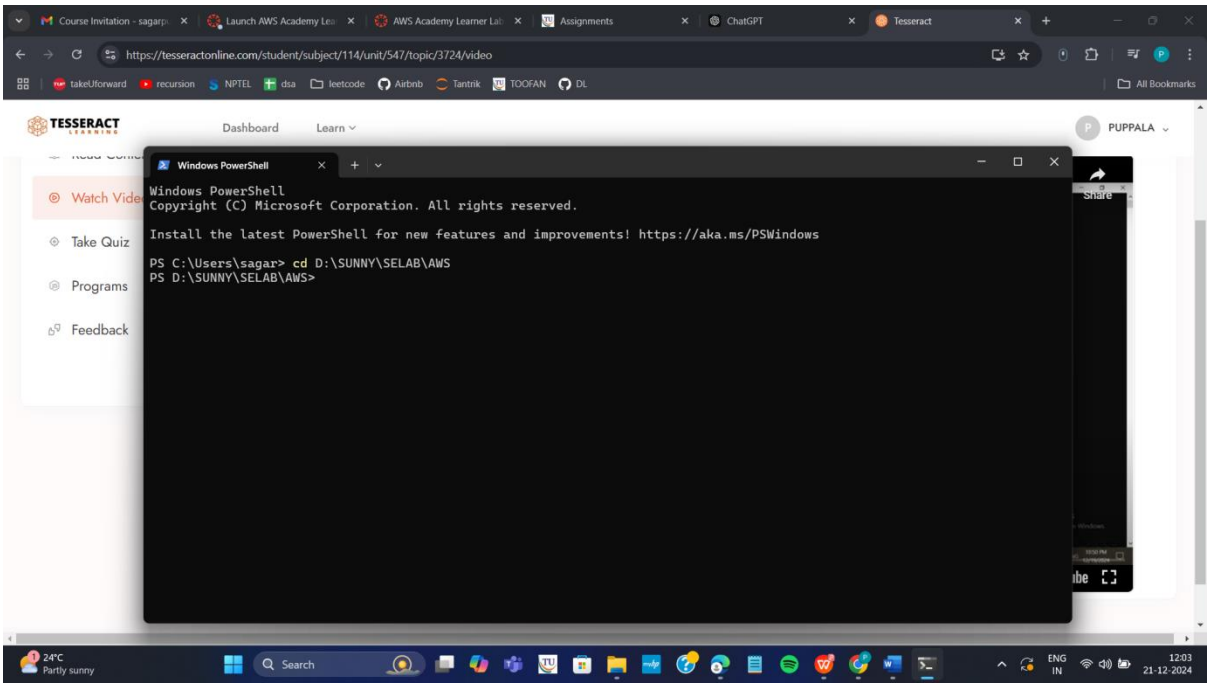
Step a: Copy the path of .pem file that you saved earlier in folder AWS eg. as here D:\SUNNY\SELABWEB\AWS



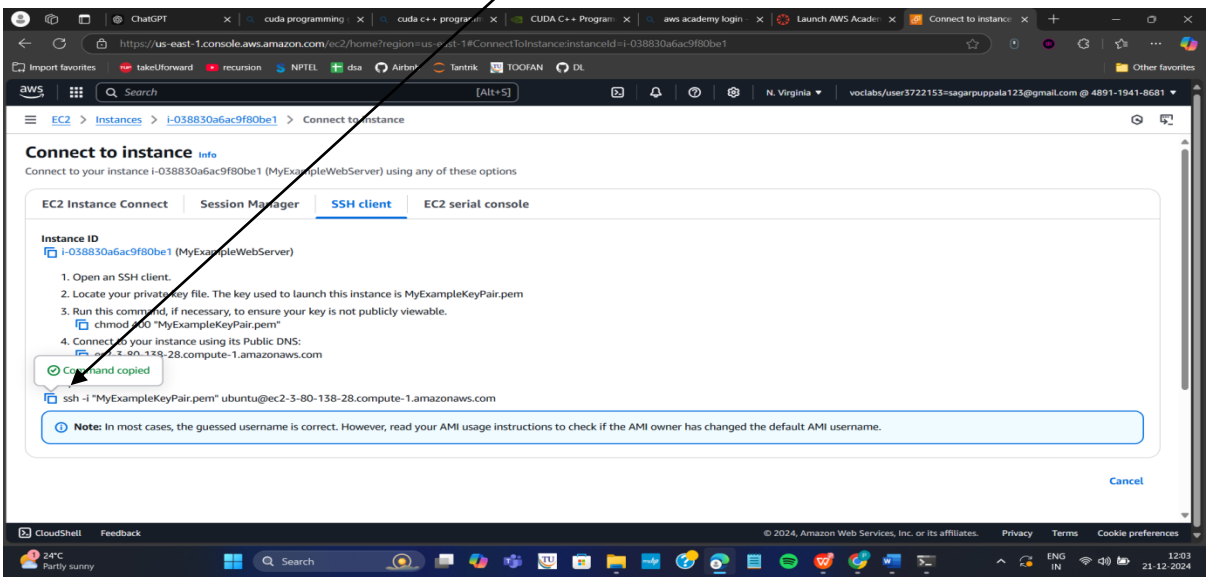
Step b: Open Powershell in administrative mode and navigate to that path.

Type: cd < path>

eg. cd D:\SUNNY\SELABWEB\AWS here specify your .pem path



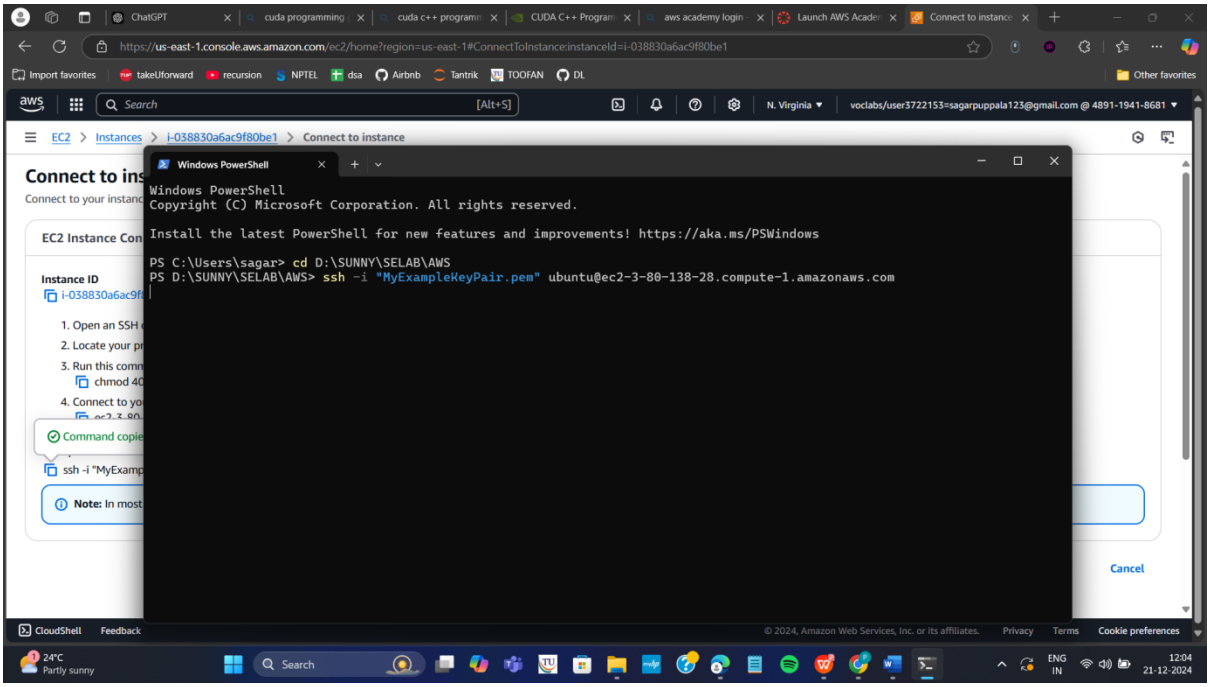
Step c: Go to SSh and copy the command ssh which is present at the below



Note: To connect to above terminals we need to go into the path of the keypair.and

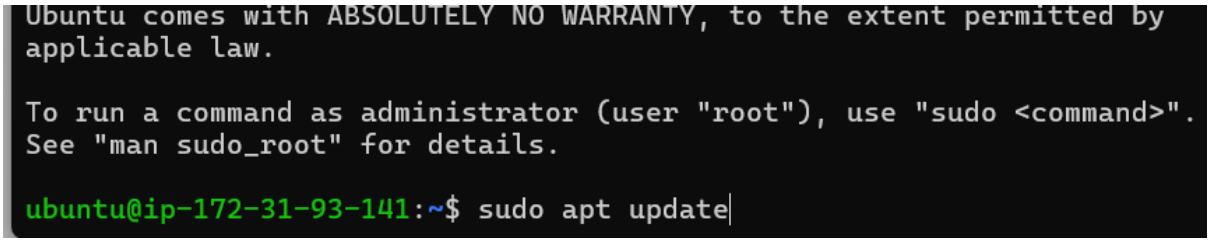
paste the ssh -i command from the aws console

Step d: **Run the pasted** ssh -i **command in the terminal**

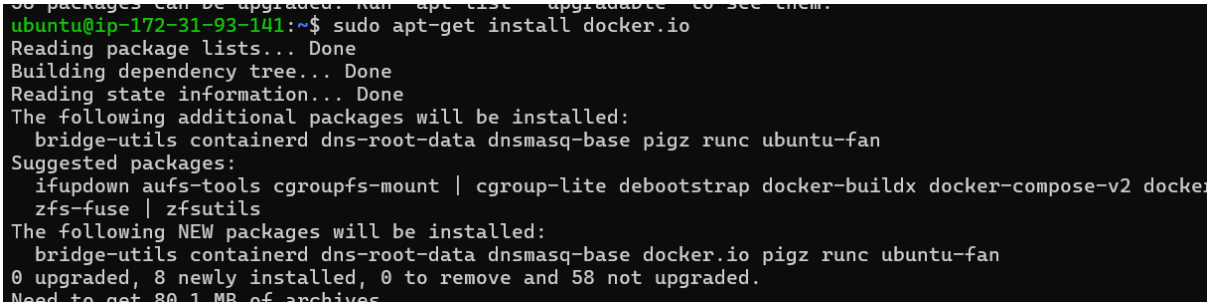


Step 5: Run the following commands to install s/w

1. Update all softwares in Ubuntu by command
   sudo apt update



2. Install docker by command
   sudo apt-get install docker.io



3. Install git by command
   sudo apt install git

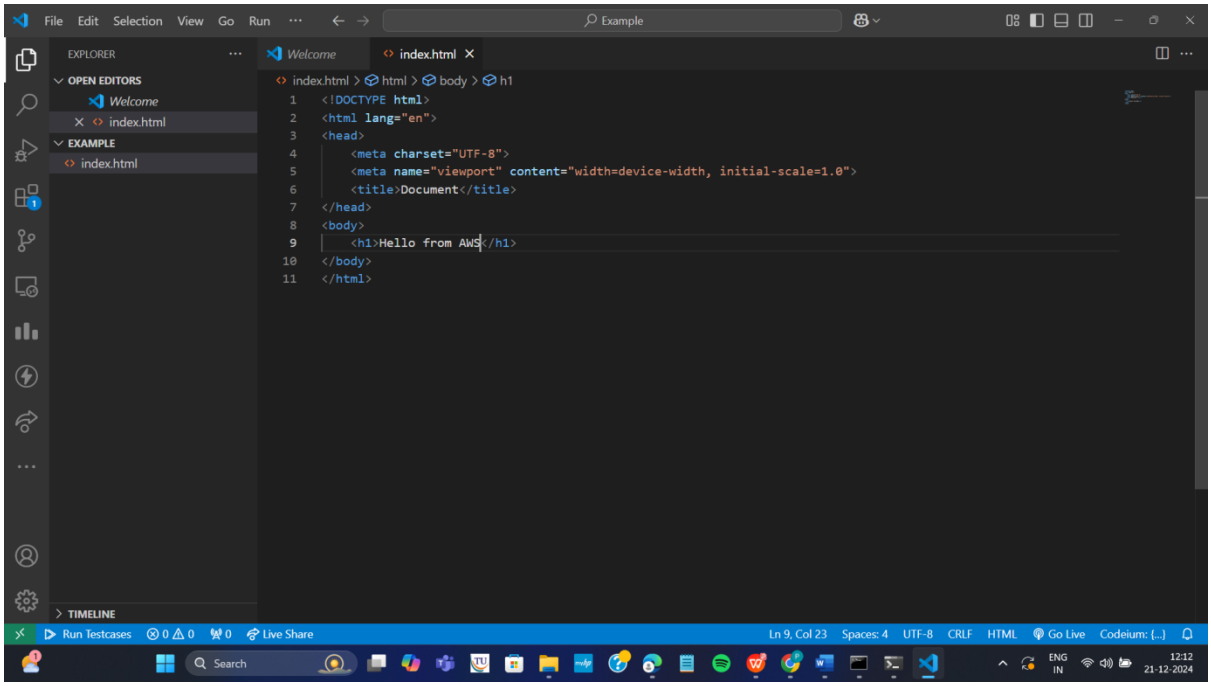
```
ubuntu@ip-172-31-93-141:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.1).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 58 not upgraded.
```

4. Install nano(text editor) by command
   Sudo apt install nano

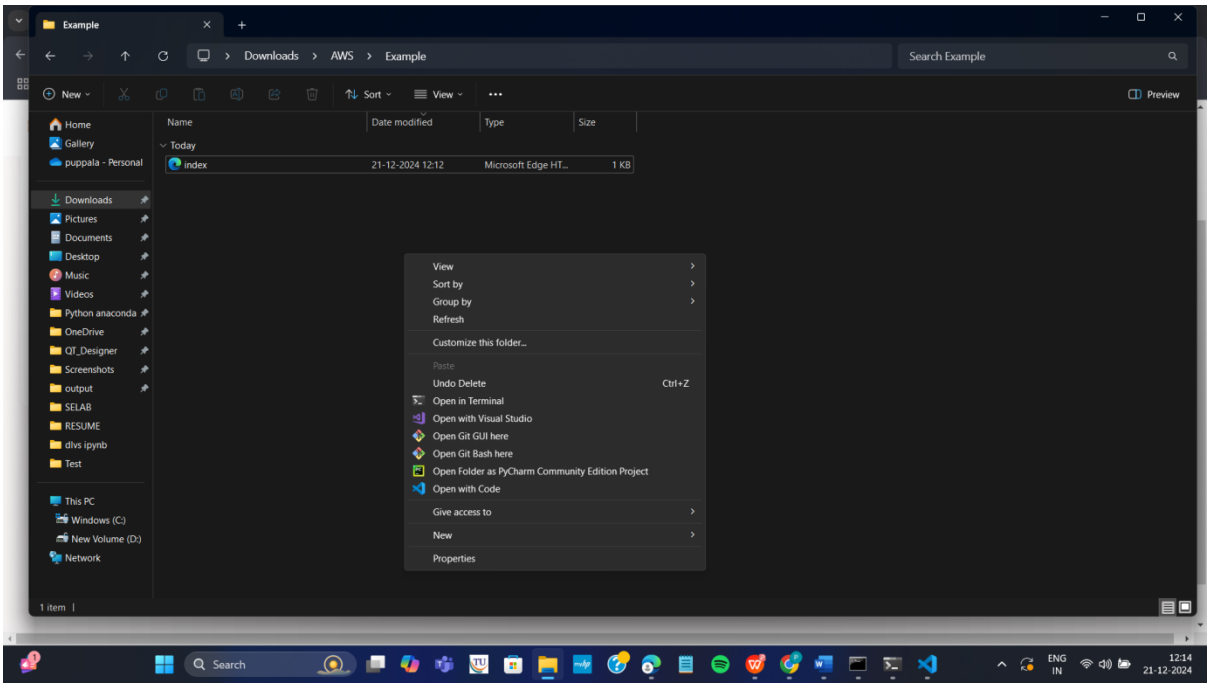
```
ubuntu@ip-172-31-93-141:~$ sudo apt install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (7.2-2ubuntu0.1).
nano set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 58 not upgraded.
```

Step 6: Now we want to create an application, push it into git, create docker image of it and run it

Step a: Create basic index.html file in folder Example and save it



Step b: Open git Bash **in folder Example** by right clicking with mouse

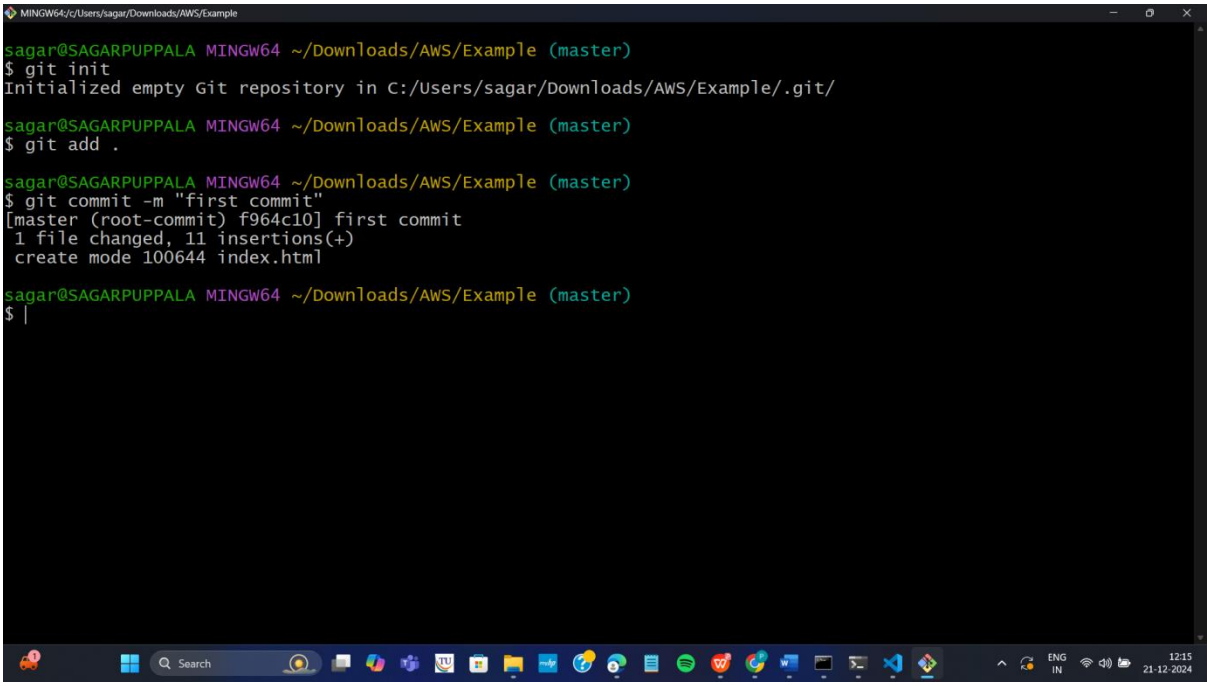


Step c: In git bash run the following commands

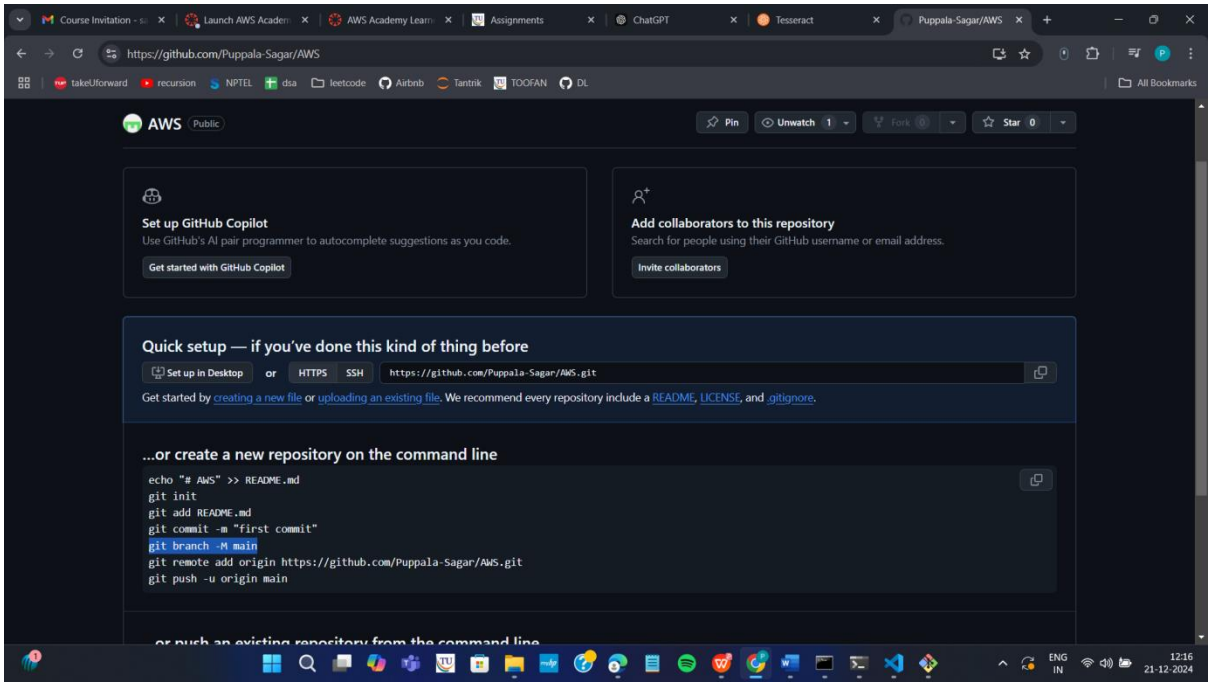
git init

git add .

Git commit –m “first commit”



Step d: Create git repository (here with name AWS)



Step e: Copy command **one by one** from above repository and run as below

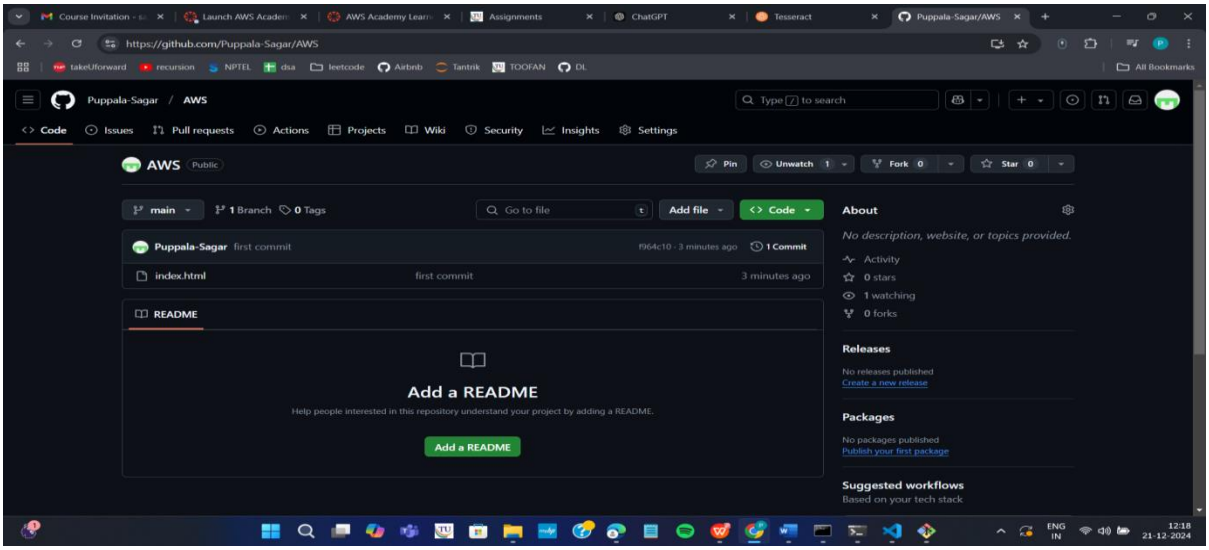
git branch –M main

git remote add origin <https url>

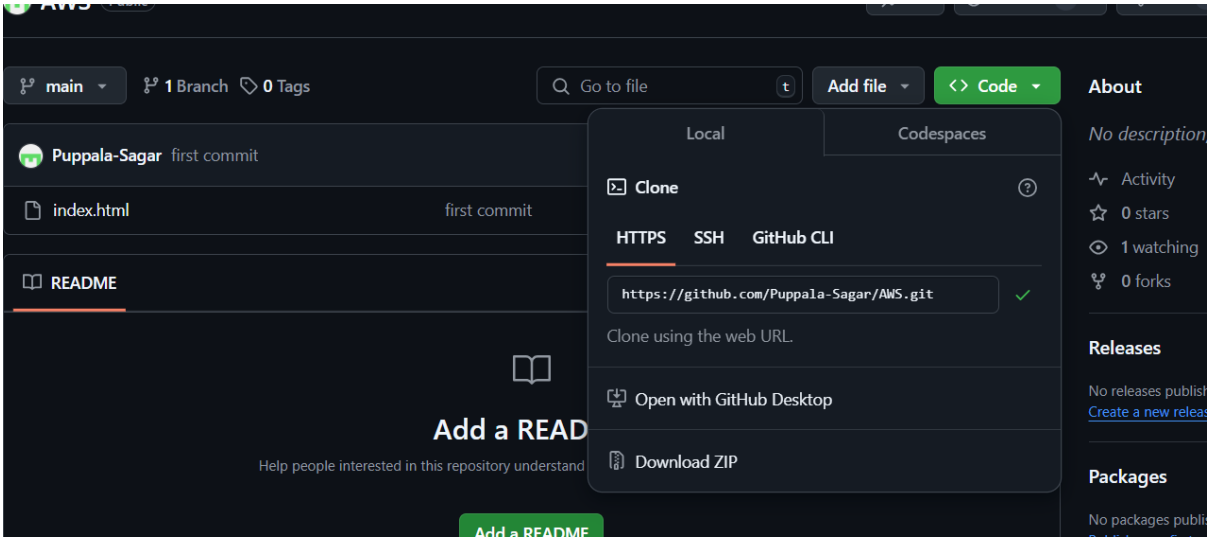
git push –u origin main

```
sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (master)
$ git branch -M main

sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (main)
$ git remote add origin https://github.com/Puppala-Sagar/AWS.git

sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 380 bytes | 380.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Puppala-Sagar/AWS.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Step f: refresh repository and You can now see index.html in github



Step g: Copy http path



Step h: Clone the repository with copied http path by command **in command prompt** **git clone <copied http url>**

Step i: Navigate to the cloned folder. Type **cd AWS** as below, next **ls** to

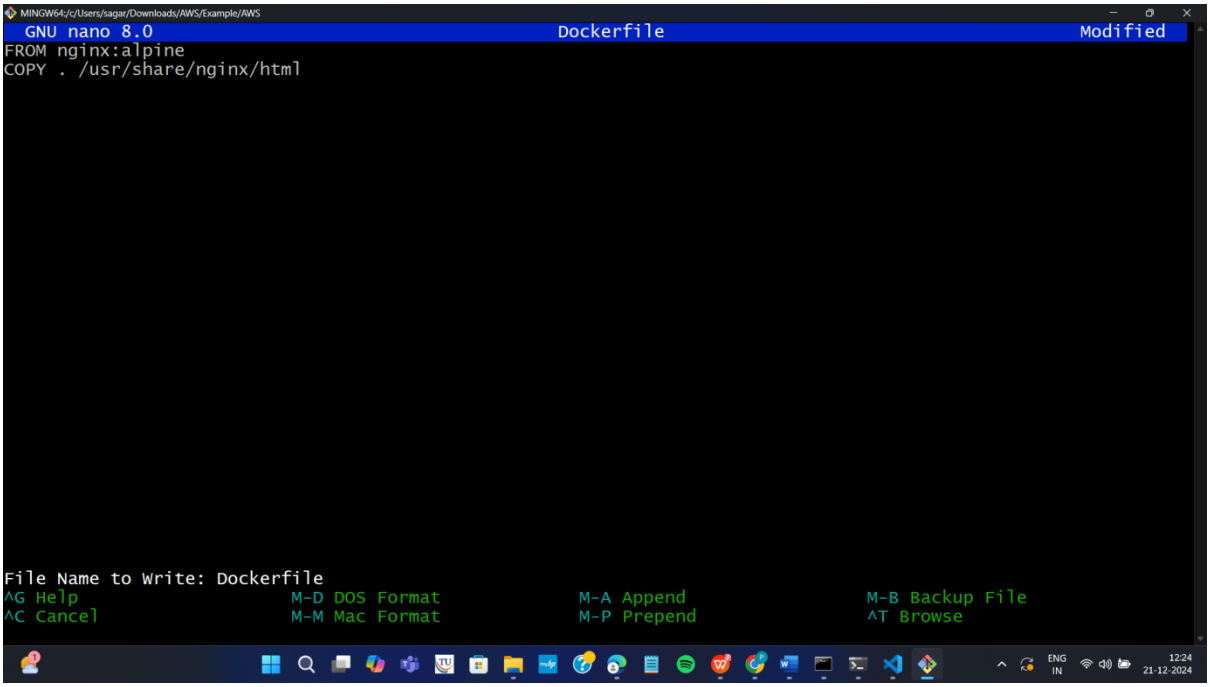
```
ubuntu@ip-172-31-93-141:~$ git clone https://github.com/Puppala-Sagar/AWS.git
Cloning into 'AWS'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
ubuntu@ip-172-31-93-141:~$ ls
AWS
ubuntu@ip-172-31-93-141:~$ cd AWS
ubuntu@ip-172-31-93-141:~/AWS$ ls
index.html
ubuntu@ip-172-31-93-141:~/AWS$
```

Step j: create Dockerfile in above command prompt in ububtu ie in power shell

Nano Dockerfile

```
sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example/AWS (main)
$ nano Dockerfile
```

Step k: Write the following data in Dockerfile and click ctrl+o Enter and then ctrl-x



Step L: Build docker image by executing the following command

sudo docker build –t mywebapp .

```
ubuntu@ip-172-31-93-141:~/AWS$ sudo docker build -t mywebapp .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  65.02kB
Step 1/2 : FROM nginx:alpine
alpine: Pulling from library/nginx
da9db072f522: Pull complete
e10e486de1ab: Pull complete
af9c0e53c5a4: Pull complete
b2eb2b8af93a: Pull complete
e351ee5ec3d4: Pull complete
fbbf7d28be71: Pull complete
471412c08d15: Pull complete
a2eb5282fbec: Pull complete
Digest: sha256:41523187cf7d7a2f2677a80609d9caa14388bf5c1fbca9c410ba3de602aaaab4
Status: Downloaded newer image for nginx:alpine
 ---> 91ca84b4f577
Step 2/2 : COPY . /usr/share/nginx/html
 ---> a3737b372d23
Successfully built a3737b372d23
Successfully tagged mywebapp:latest
```

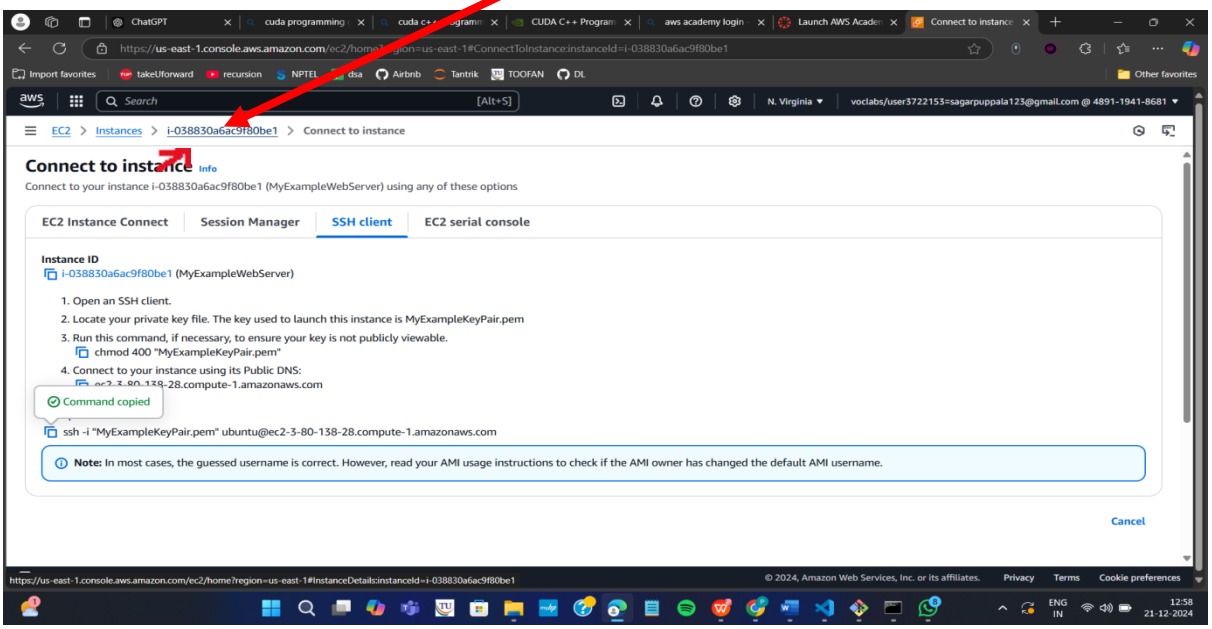
Thus image is created

Step m: run the image and map it to port 80

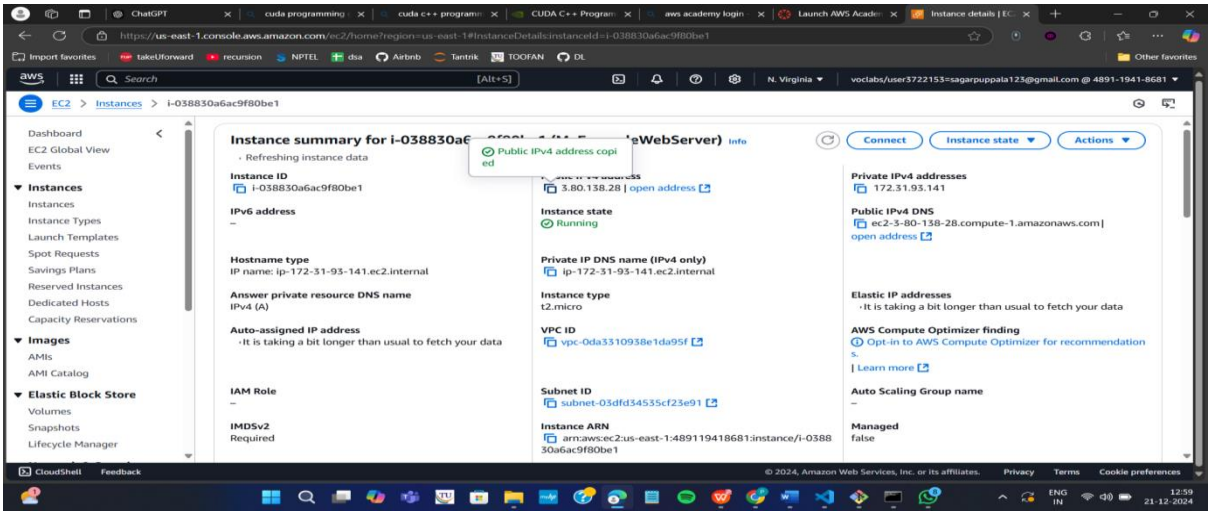
sudo docker run –d –p 80:80 mywebapp

```
ubuntu@ip-172-31-93-141:~/AWS$ sudo docker run -d -p 80:80 mywebapp
12429f3ea7b7b0750424000e6404b7c34ba8213745f54708e5f1e9e716177b58
```

Step n: Go to instances and click on instance here

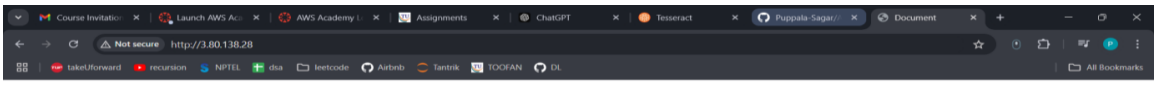


Step o: Copy public ipv4 address


ChatGPT
cuda programming
cuda c++ programm
CUDA C++ Program
aws academy login
Launch AWS Acade
Instance details | EC
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#InstanceDetails:instanceId=i-038830a6ac9f80be1
Import favorites
takeUforward
recursion
NPTEL
dsa
Airbnb
Tantrik
TOOFAN
DL
Other favorites
Search
[Alt+S]
N. Virginia
voclabs/user3722153=sagarpuppala123@gmail.com @ 4891-1941-8681
EC2 > Instances > i-038830a6ac9f80be1
Dashboard
EC2 Global View
Events
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Instance summary for i-038830a6 eWebServer) Info
Refreshing instance data
Public IPv4 address copied
Connect
Instance state
Actions
Instance ID
i-038830a6ac9f80be1
IPv6 address
–
Hostname type
IP name: ip-172-31-93-141.ec2.internal
Answer private resource DNS name
IPv4 (A)
Auto-assigned IP address
It is taking a bit longer than usual to fetch your data
IAM Role
–
IMDSv2
Required
3.80.138.28 | open address
Instance state
Running
Private IP DNS name (IPv4 only)
ip-172-31-93-141.ec2.internal
Instance type
t2.micro
VPC ID
vpc-0da3310938e1da95f
Subnet ID
subnet-03dfd34535cf23e91
Instance ARN
arn:aws:ec2:us-east-1:489119418681:instance/i-038830a6ac9f80be1
Private IPv4 addresses
172.31.93.141
Public IPv4 DNS
ec2-3-80-138-28.compute-1.amazonaws.com | open address
Elastic IP addresses
It is taking a bit longer than usual to fetch your data
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations. | Learn more
Auto Scaling Group name
–
Managed
false
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ENG
IN
12:59
21-12-2024

Step p: Paste it in the browser to get below output

Note : if https :__ won't work then type just http:___



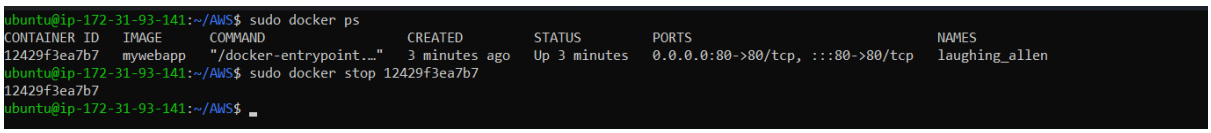
Hello from AWS

Important do it without fail. **Step Q:** stop container

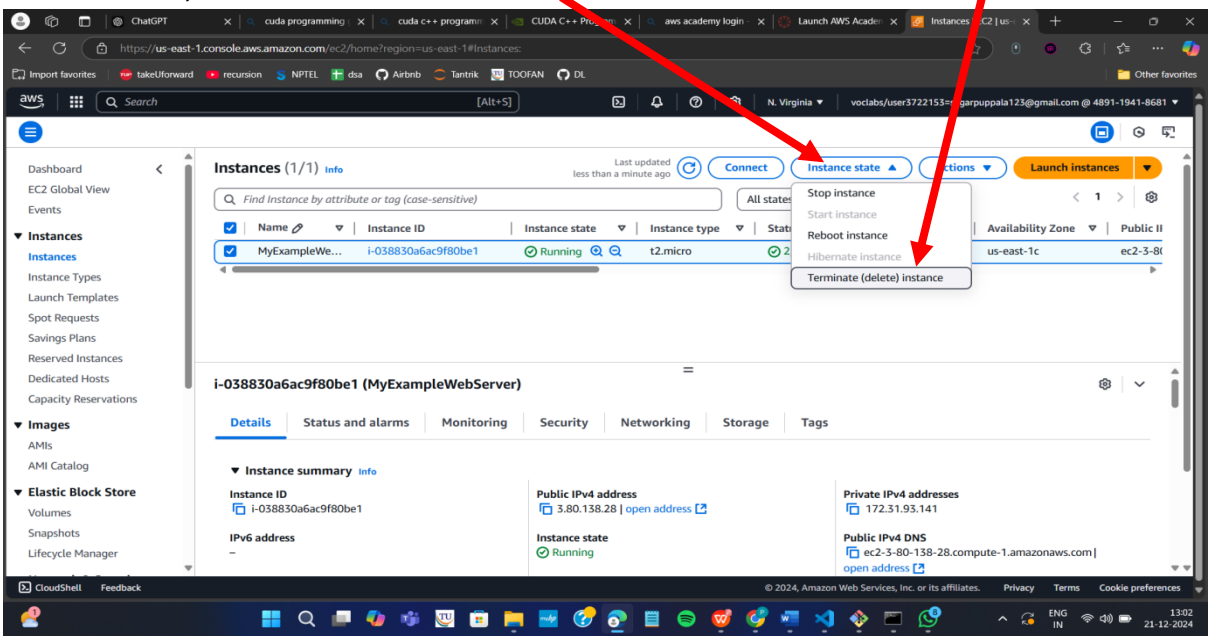
Run the following command to stop the container

i. sudo docker ps

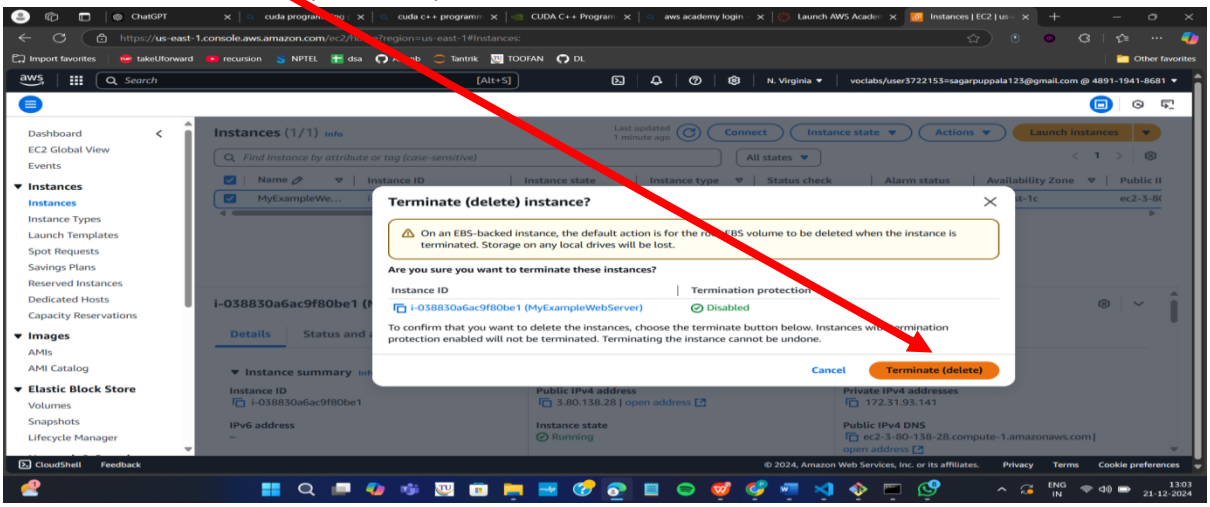
ii. sudo docker stop <container-id>



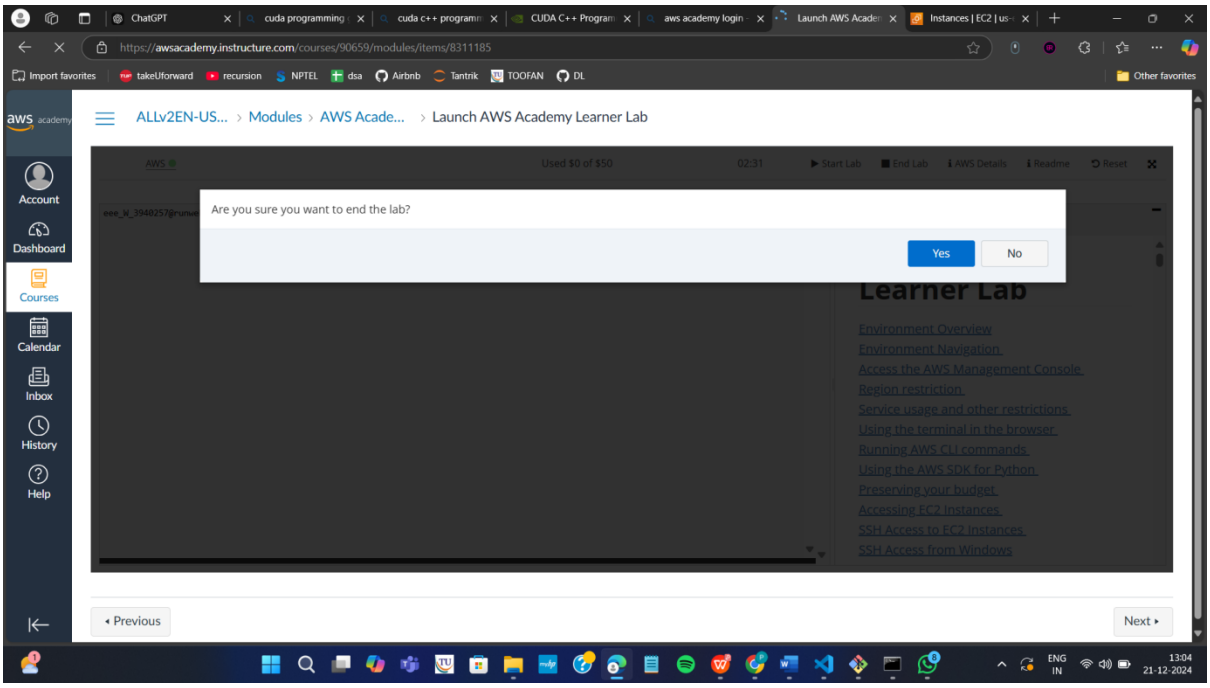
iii. Go back to instances and click on option instance. Once load as below, then Click on Instance state and select terminate instances



Click on Terminated (deleted)



iv. Now click on End lab

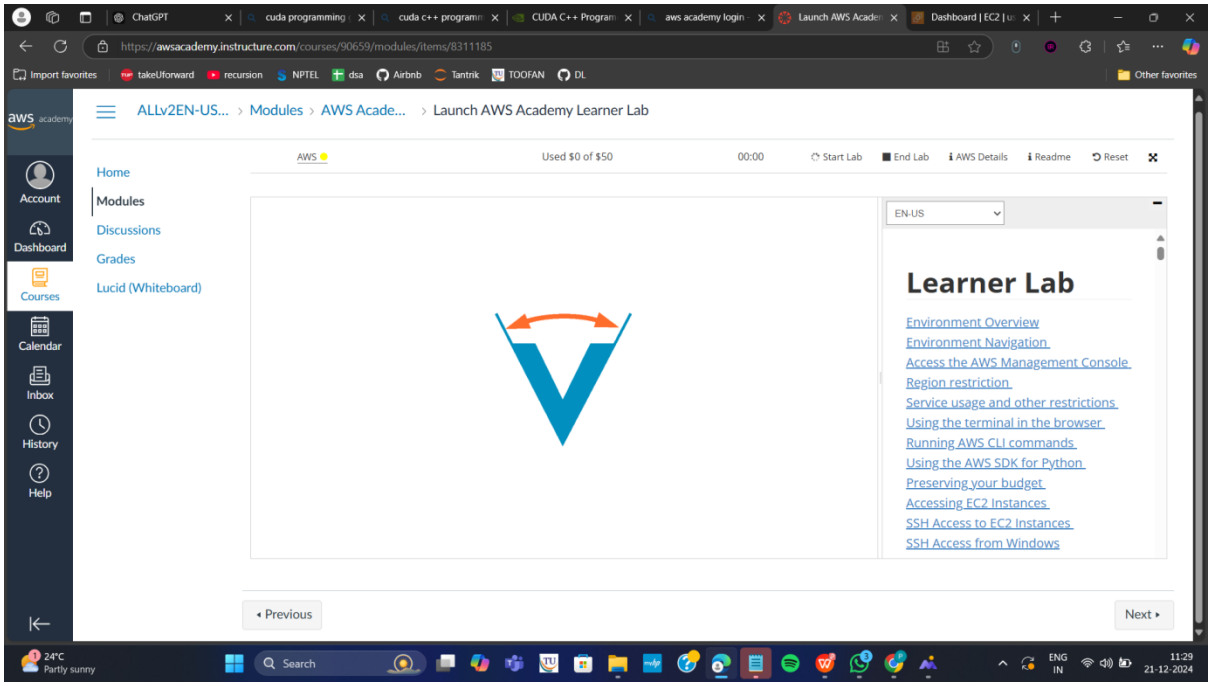


End of exercise 1 (DEPLOYMENT of index.html using EC2 instance in AWS) consisting of:

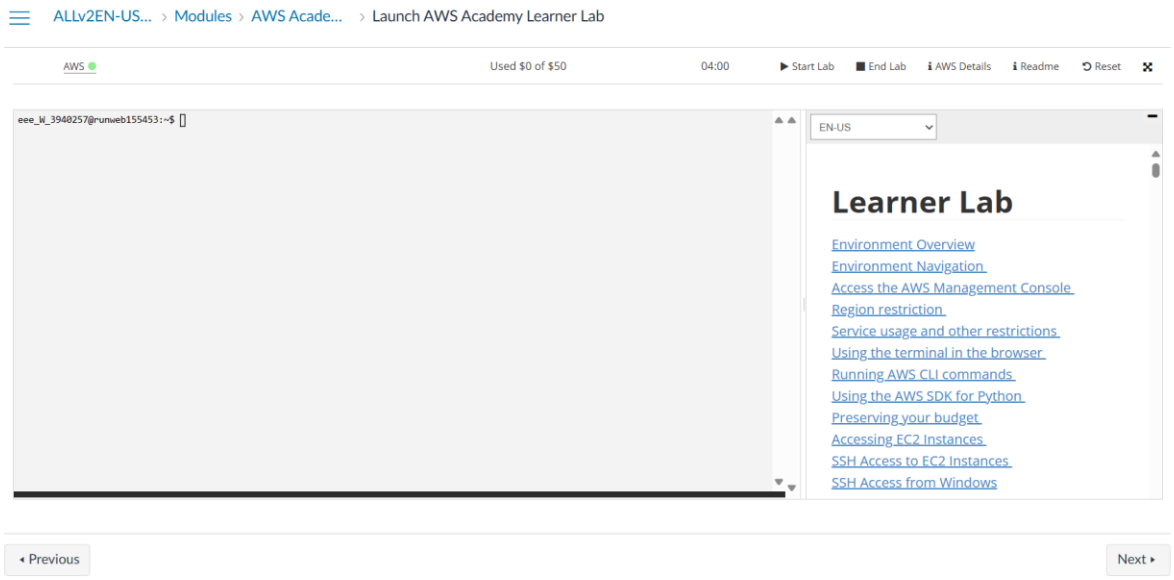
1. creating EC2 instance to install Ubuntu, docker, git , and nano.
2. creating web application (index.html) using Visual studio
3. Pushing index.html to github and cloning the it
4. Creating Dockerfile to build image for running index.html on nginx server
5. Run it as container using docker command and get the output on browser
6. Terminate the instance
7. End lab

2 MAVEN WEB PROJECT DEPLOYMENT IN AWS

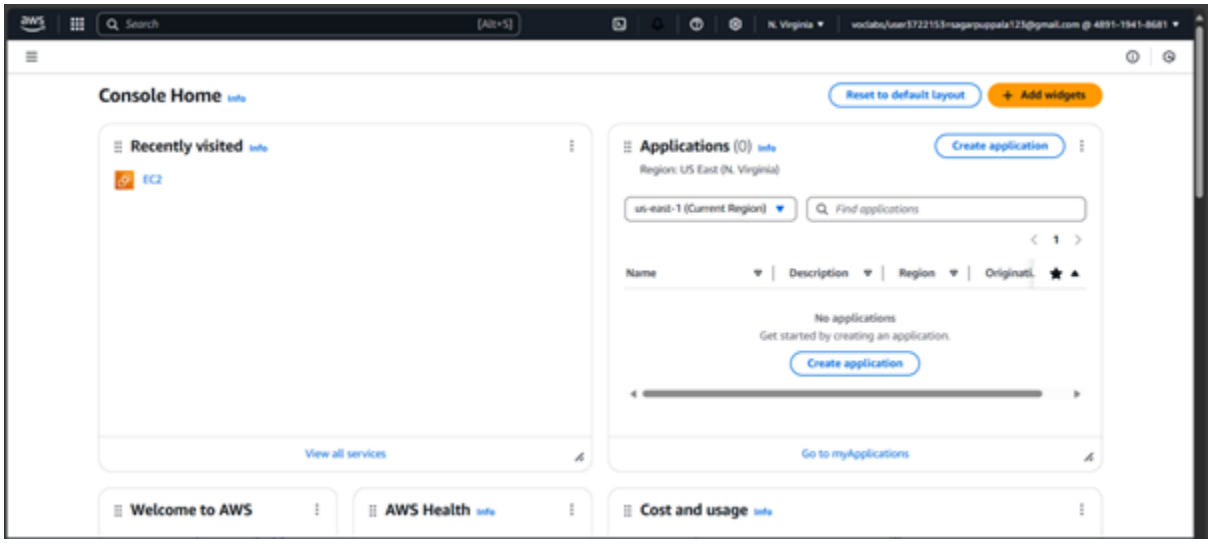
Click on start lab



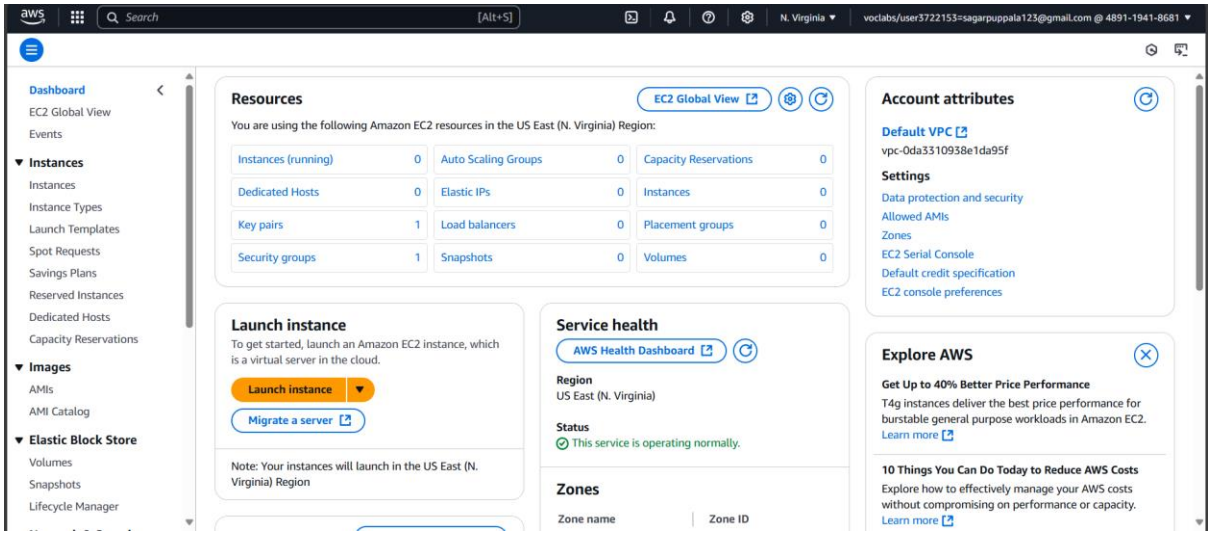
Click on AWS



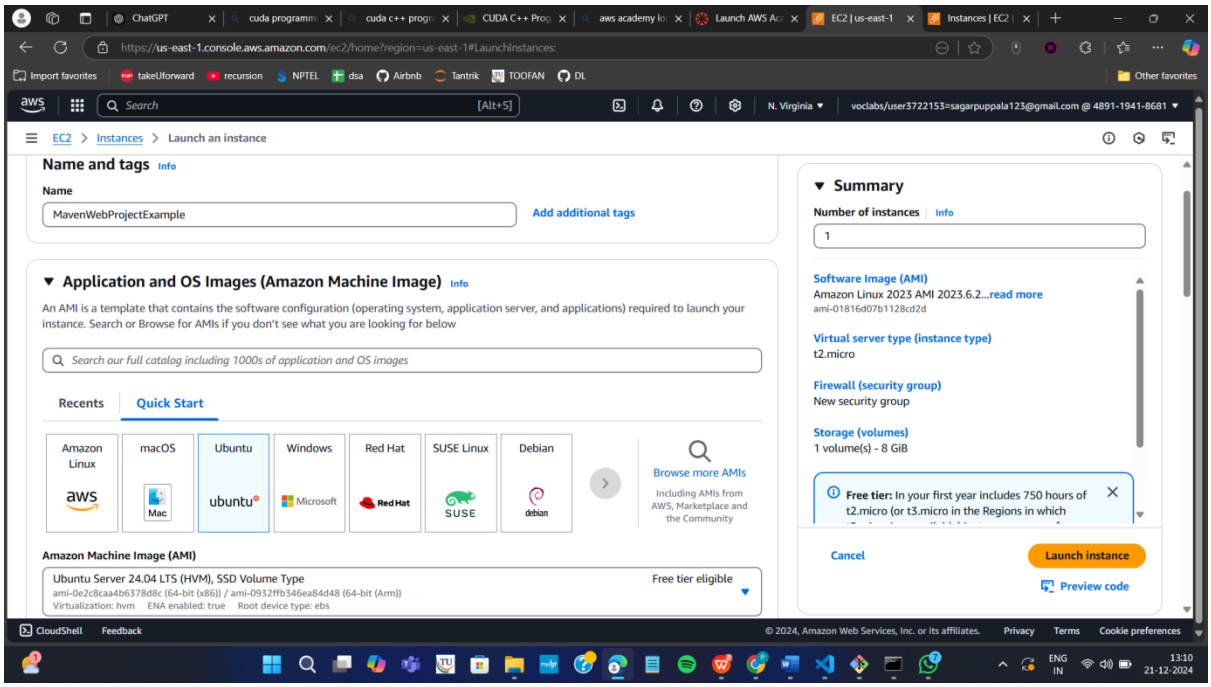
Click on EC2



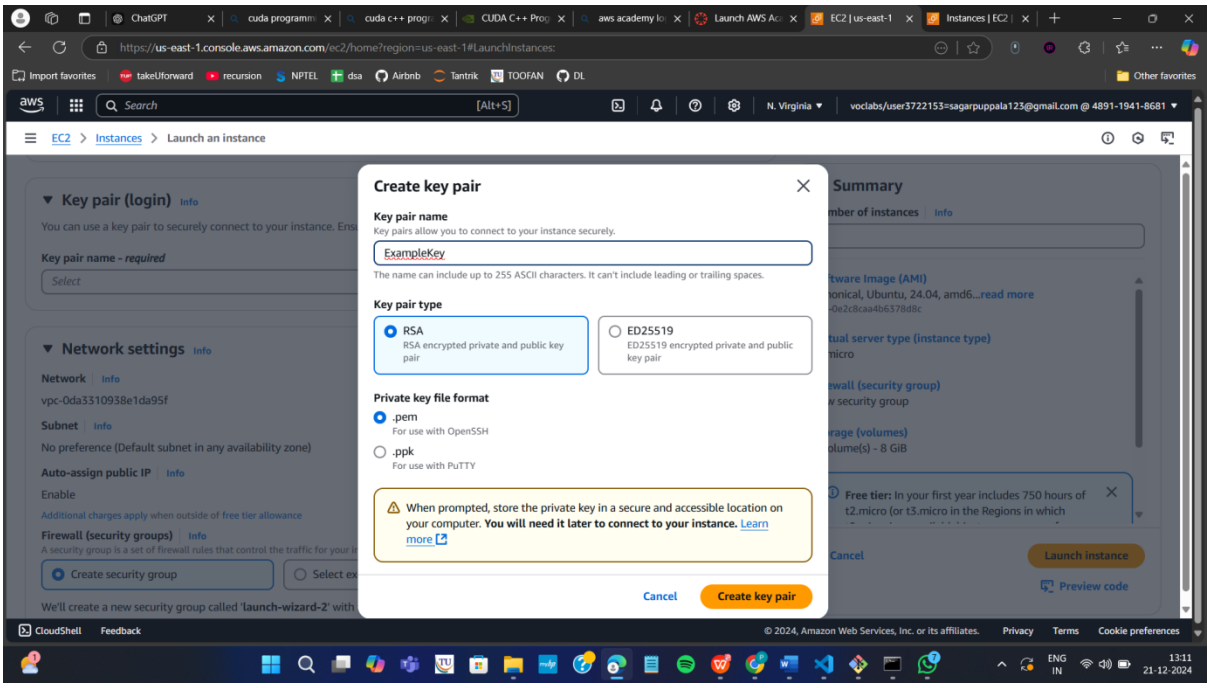
Click on launch instance



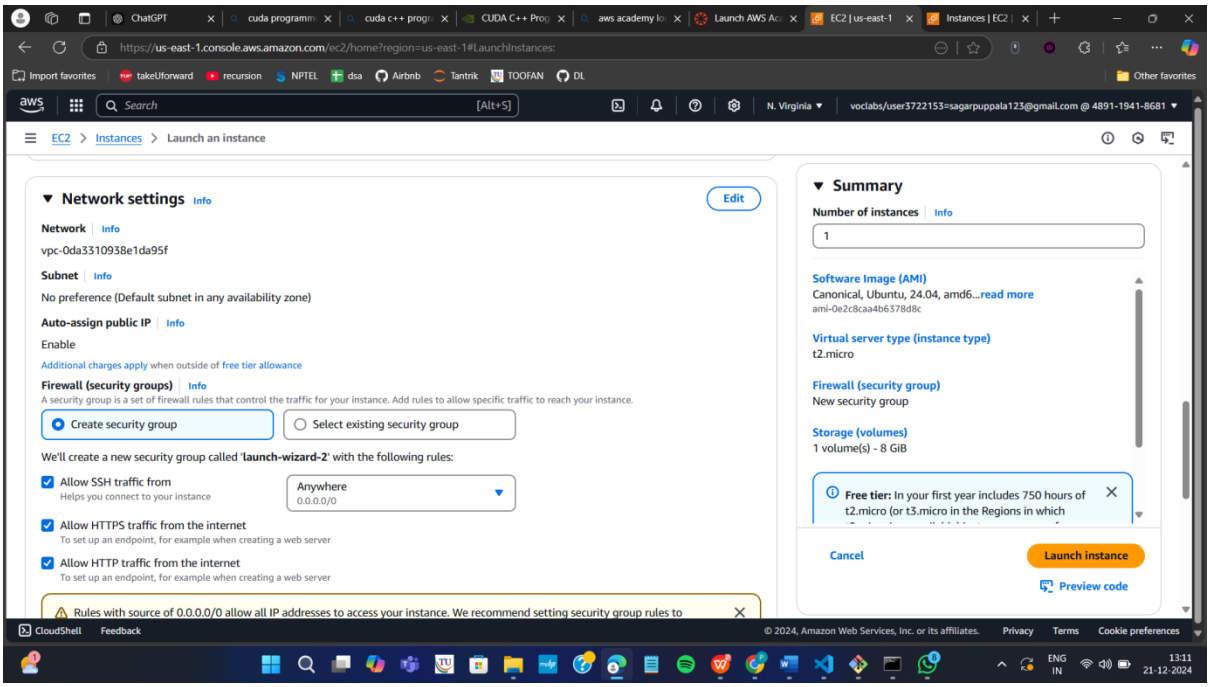
Give name



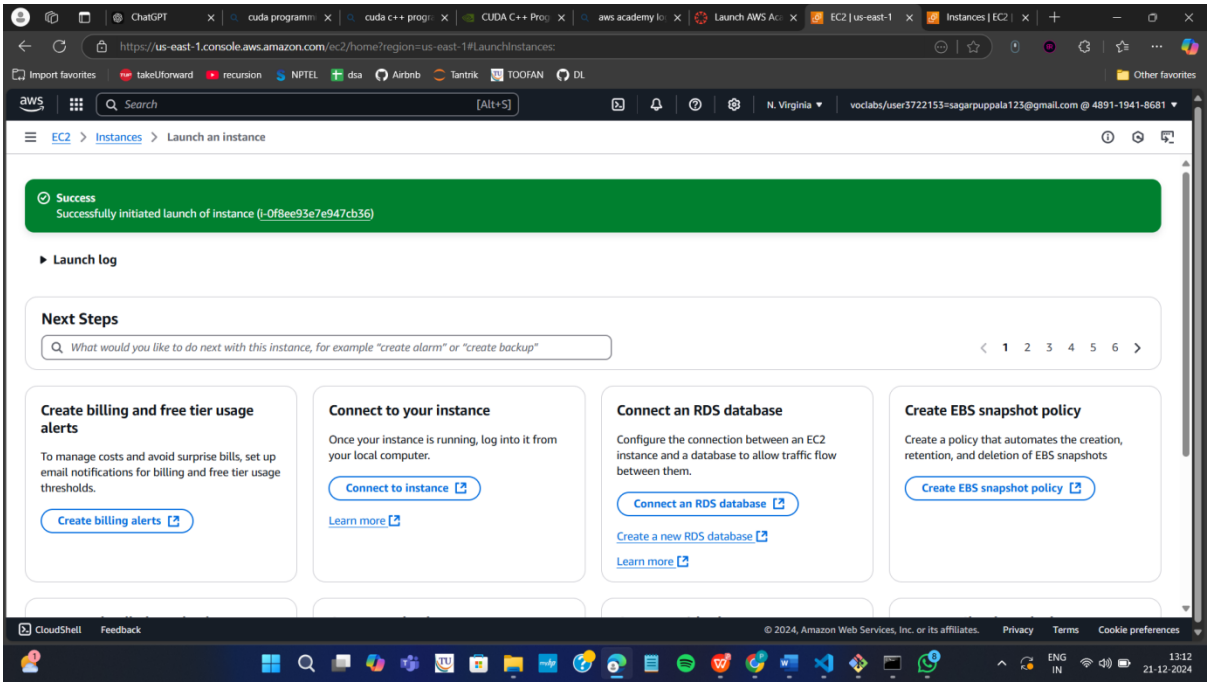
Create new key



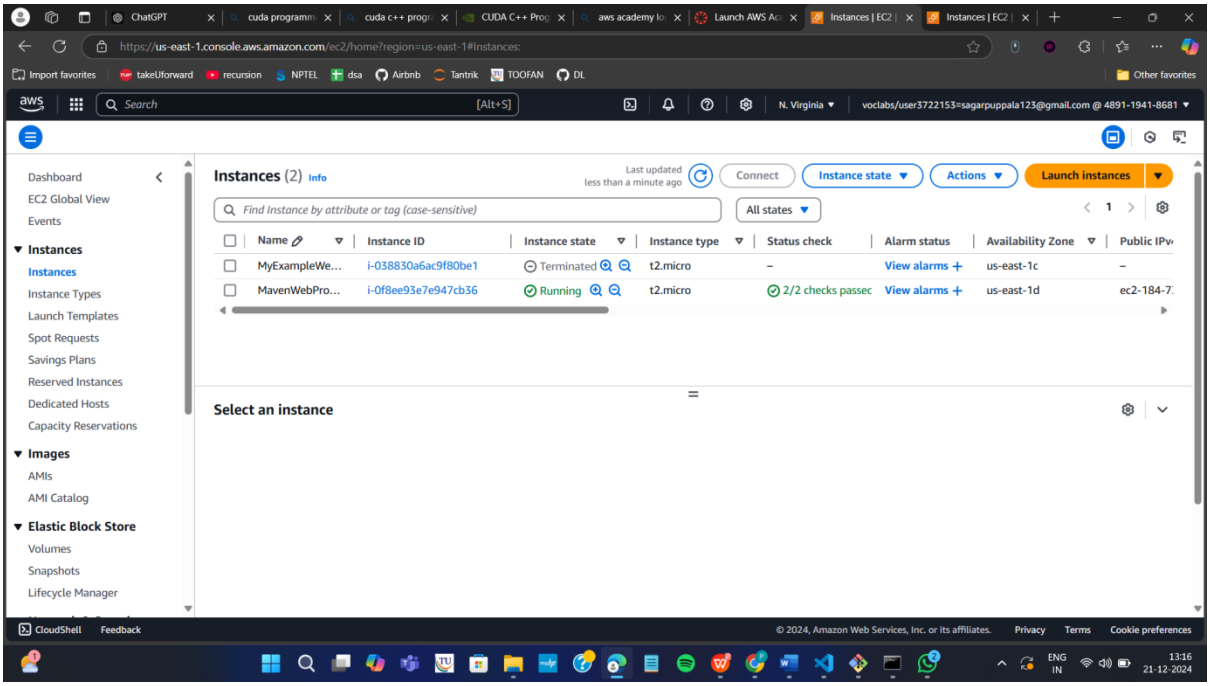
Check all the boxes under network and click on launch instance



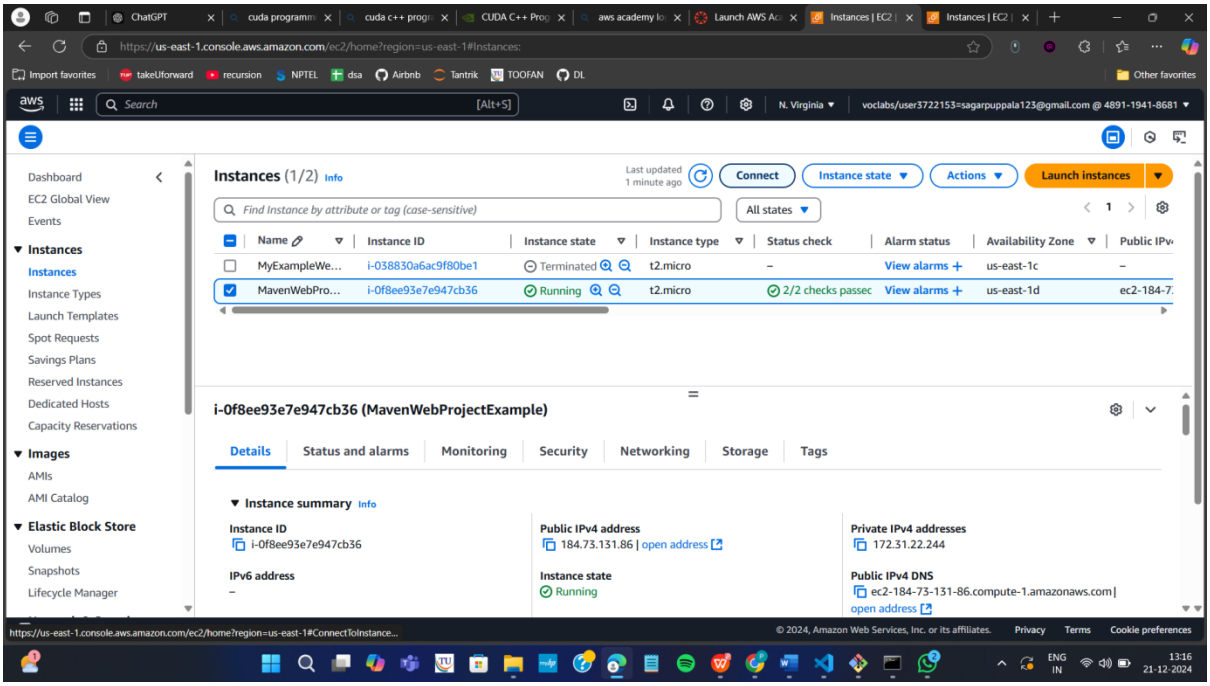
Click on instances



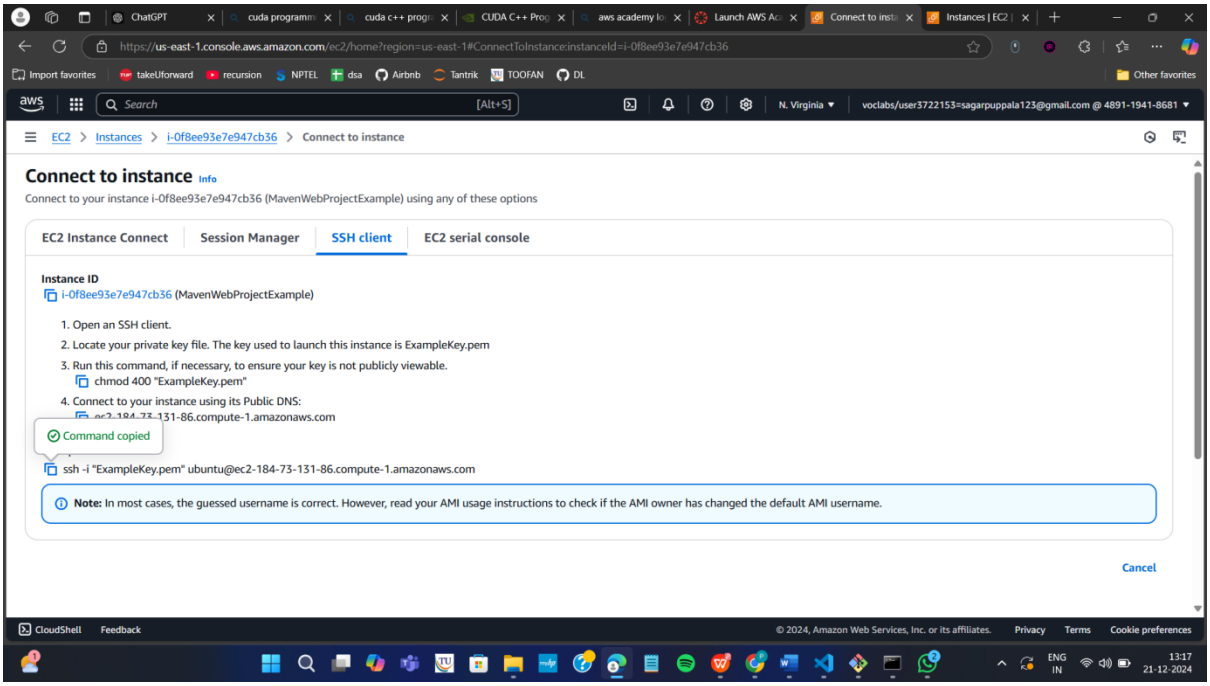
Wait until you get 2 tests passes



Click on connect after checking the box



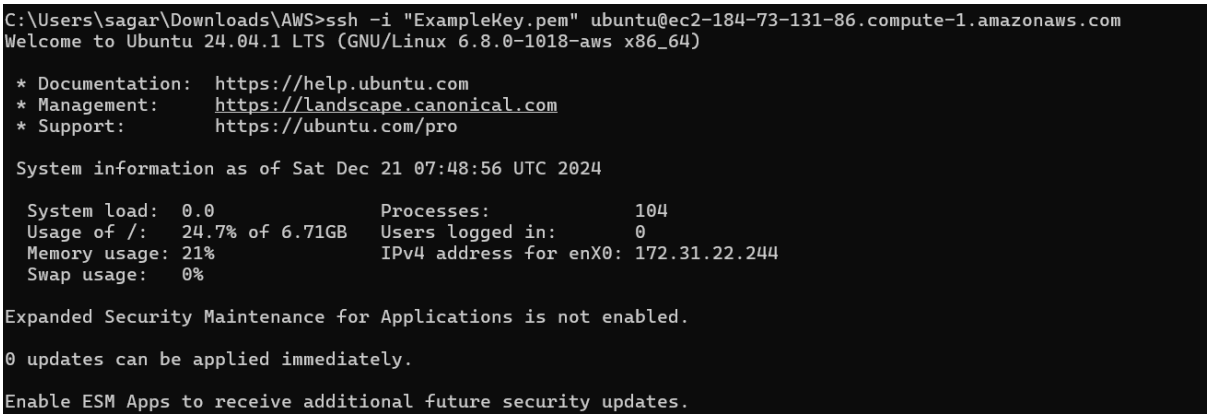
Copy ssh command



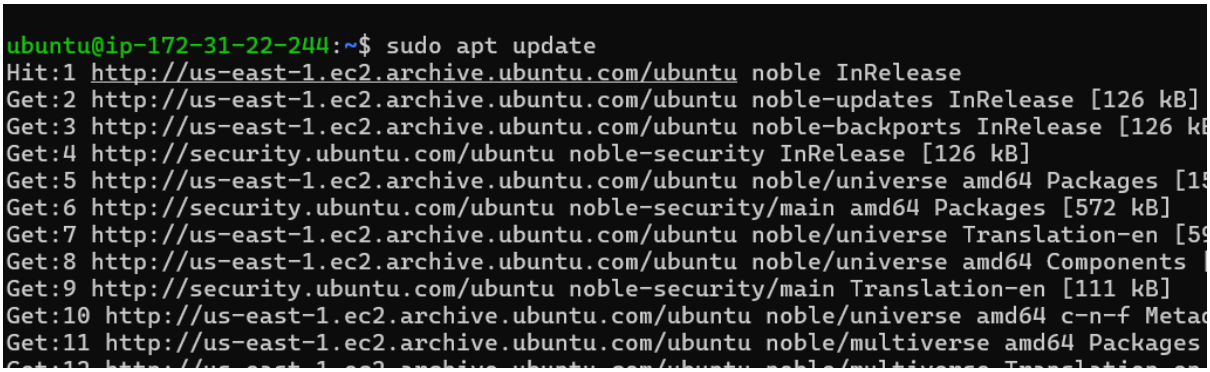
Open the terminal, Navigate to the path

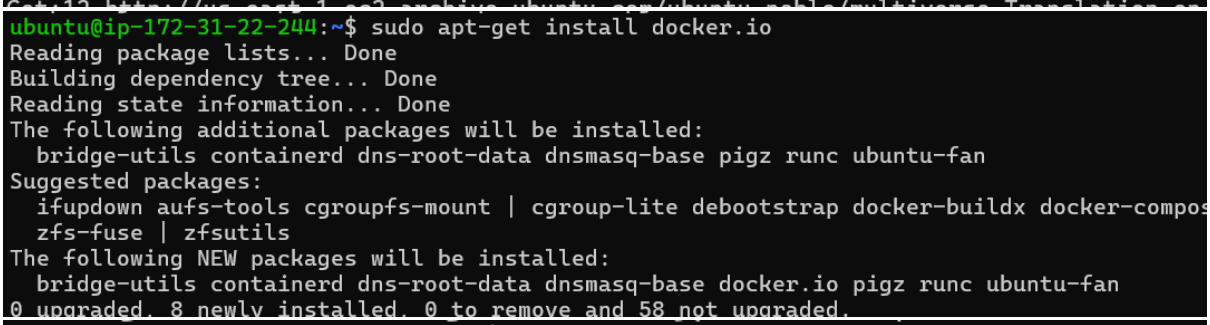
```
C:\Users\sagar>cd C:\Users\sagar\Downloads\AWS
```

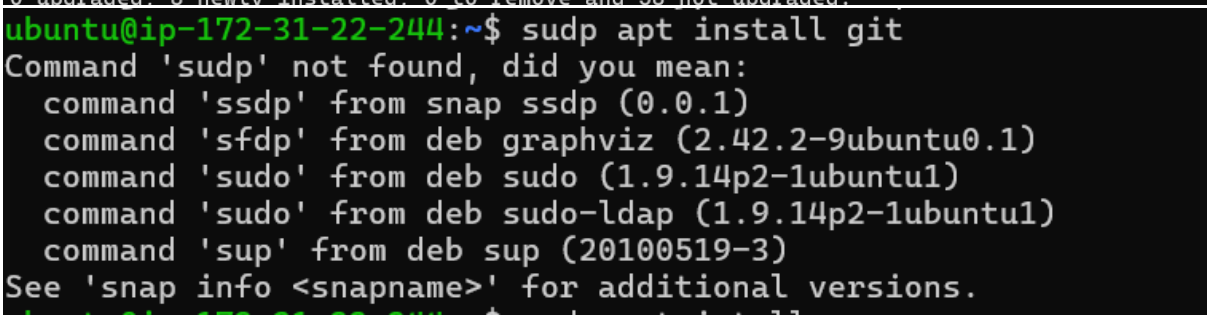
Run the ssh command

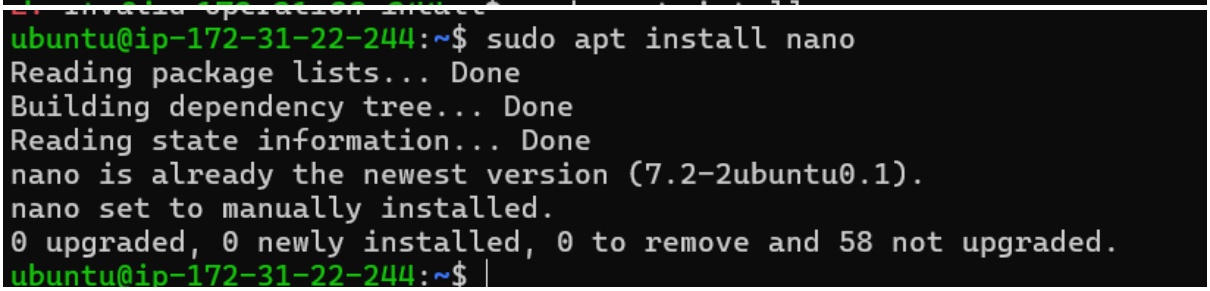


Run the following commands

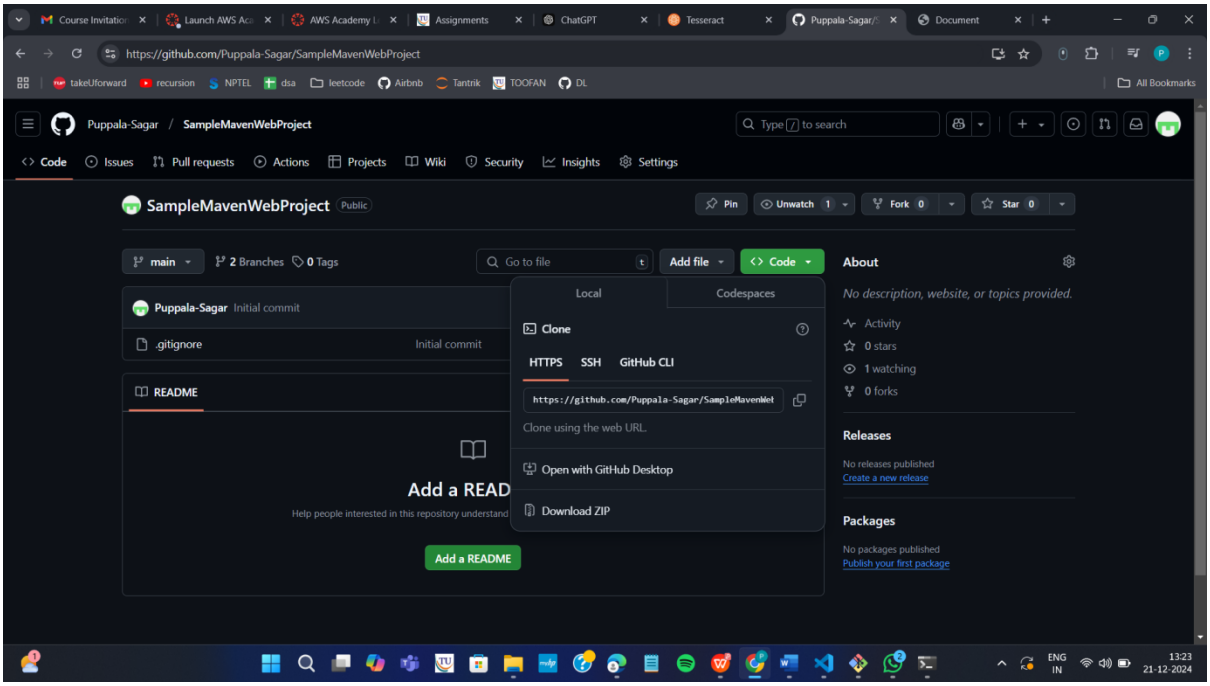








Open MAVEN WEB PROJECT REPO in github and copy http link



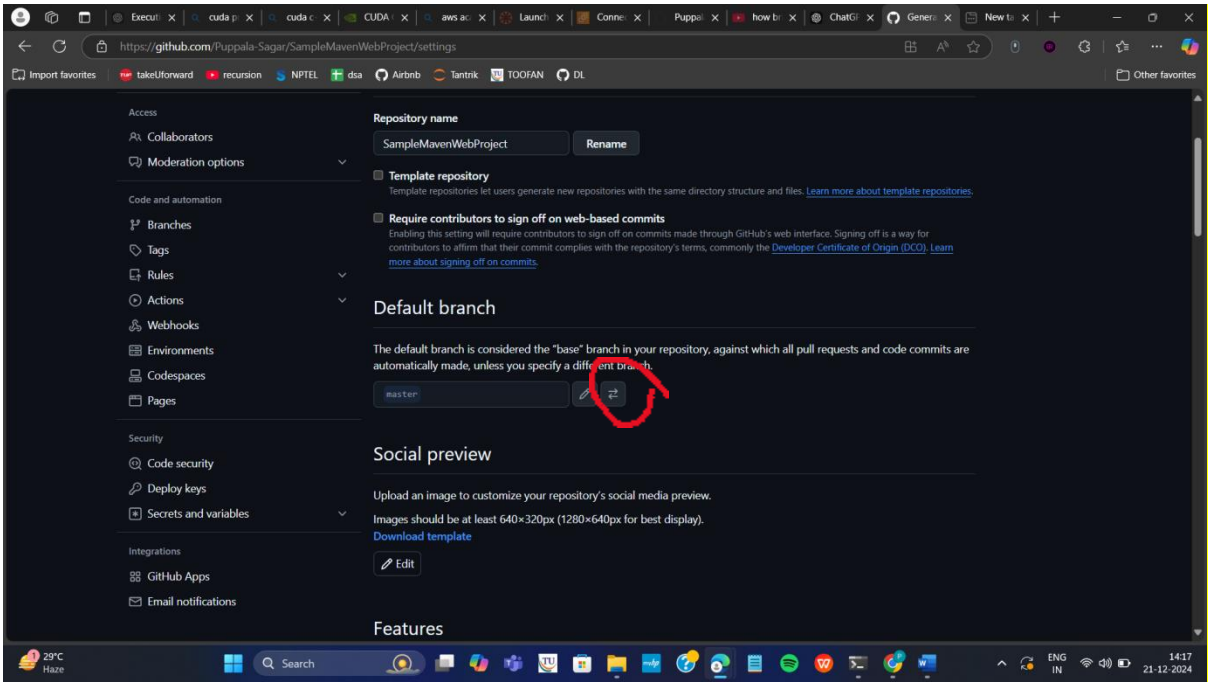
Clone the repository

```
ubuntu@ip-172-31-22-244:~$ git clone https://github.com/Puppala-Sagar/SampleMavenWebProject.git
Cloning into 'SampleMavenWebProject'...
remote: Enumerating objects: 39, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 39 (delta 3), reused 23 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (39/39), 7.82 KiB | 2.60 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

If your repository is in main run following

```
ubuntu@ip-172-31-22-244:~$ ls
SampleMavenWebProject
ubuntu@ip-172-31-22-244:~$ cd SampleMavenWebProject/
ubuntu@ip-172-31-22-244:~/SampleMavenWebProject$ ls
```

If not goto your github repo
Click on settings
Under default branch section click on the icon shown and you're your master branch as default and execute above commands




```
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$ nano Dockerfile
```

```
  GNU nano 7.2                          Dockerfile *
FROM tomcat:9-jdk11
COPY target/*.war /usr/local/tomcat/webapps
```

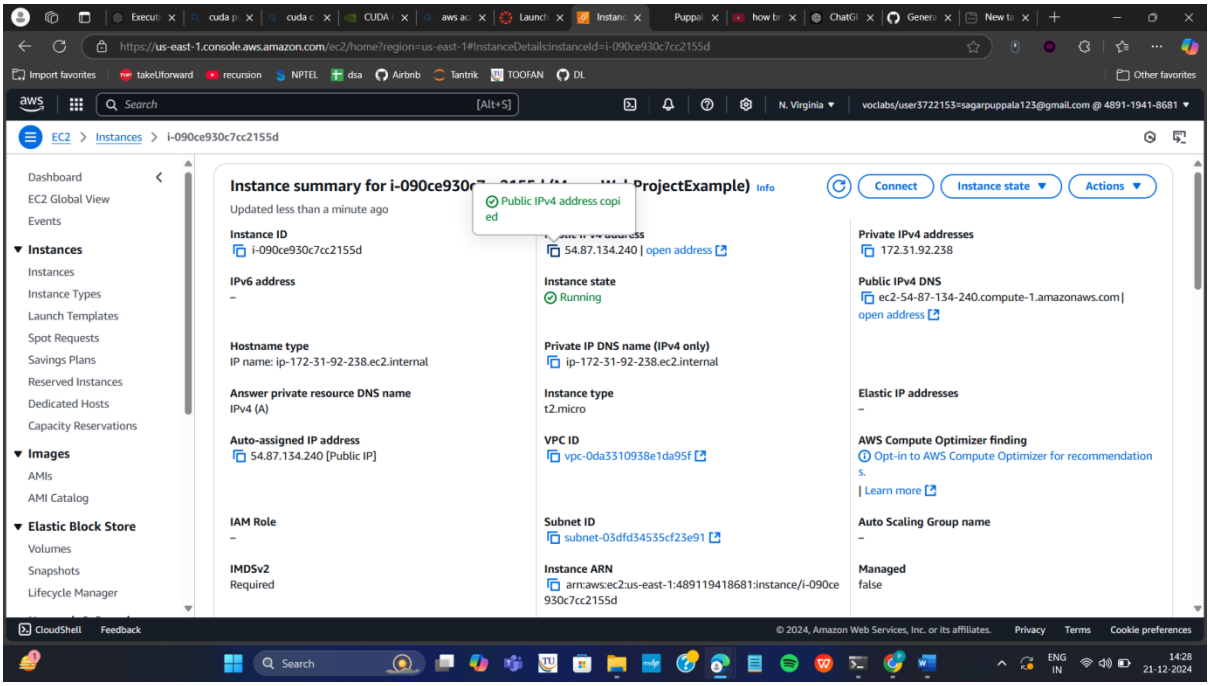
Build docker image

```
Build an image from a Dockerfile
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$ sudo docker build -t mavenwebproject .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  101.4kB
Step 1/2 : FROM tomcat:9-jdk11
9-jdk11: Pulling from library/tomcat
de44b265507a: Extracting [========================>                          ]  15.73MB/29.75MB
4d0025a6d227: Download complete
5a7ece70ec66: Downloading [====================================>              ]    111MB/145.6MB
623a4ff914ca: Download complete
6d3bf2e80222: Download complete
9ab2f23fa0e1: Download complete
4f4fb700ef54: Download complete
cdbea10bf012: Download complete
```

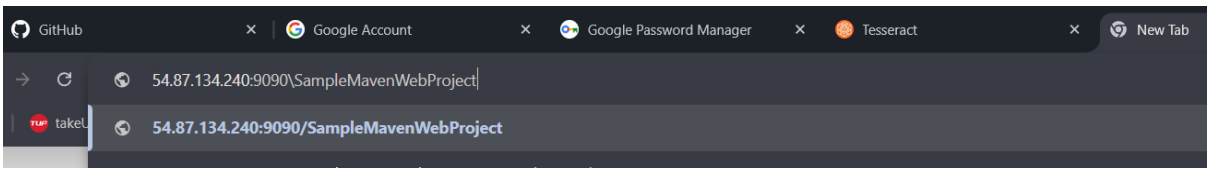
Run the container

```
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$ sudo docker run -d -p 9090:8080 mavenwebproject
e83304610c89a12dd5eab6379ea697e0f445baaf8f45d85edb009d356beda8e7
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$
```

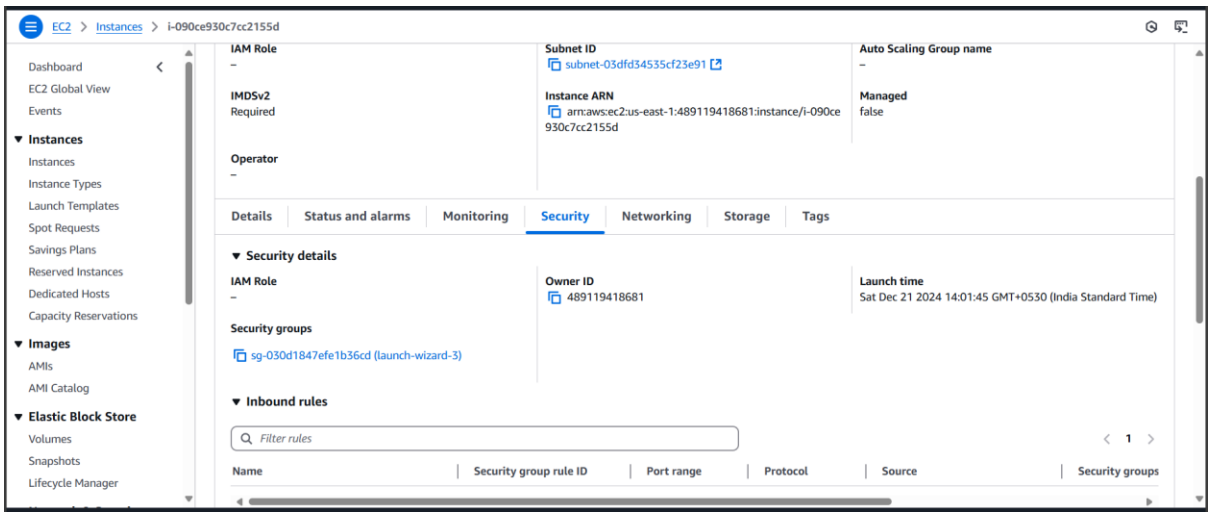
Copy public ipv4



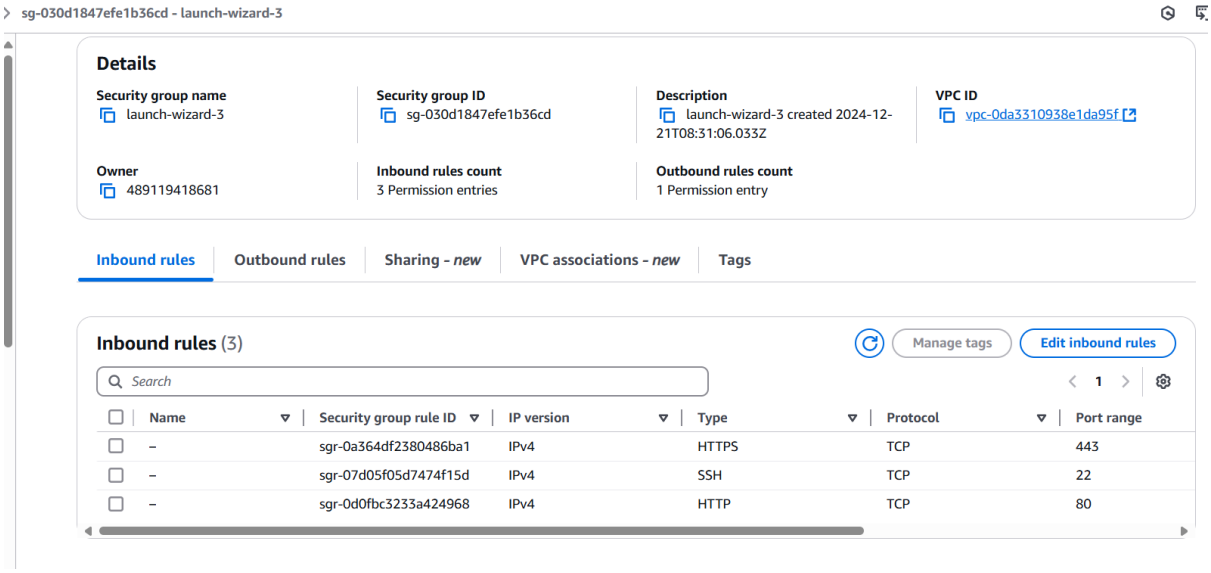
Enter this url in browser and click enter



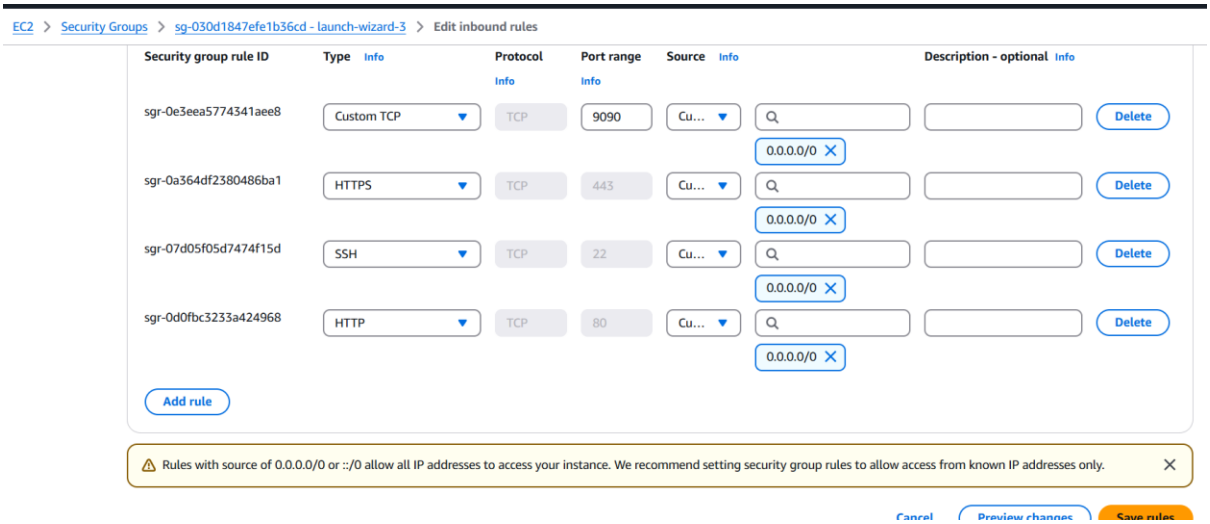
If your app is not running goto security and Click on security groups



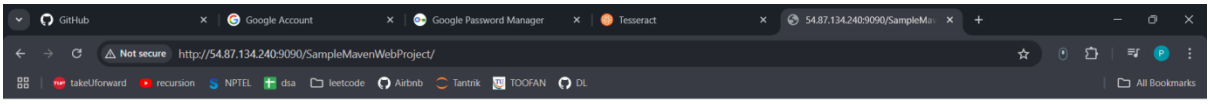
Click on edit inbound rules



Click on add rule give port as 9090 and 0.0.0.0/0 click on save



And refresh the page to check whether the web page has loaded or not



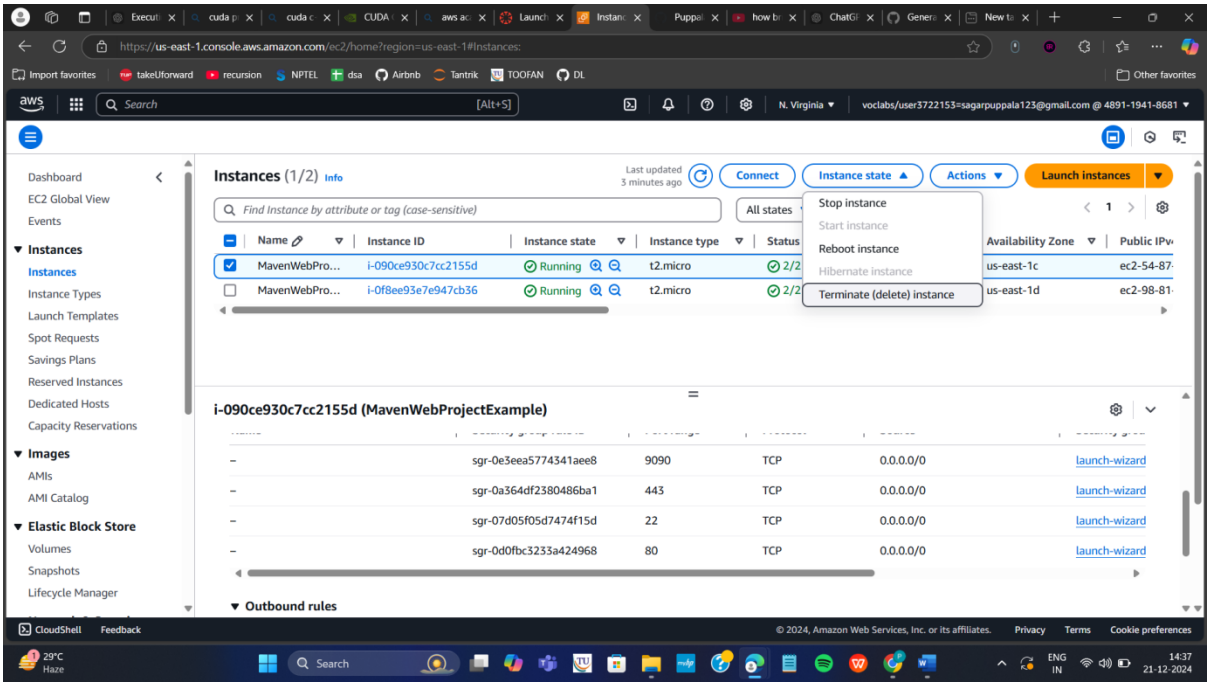
Hello World!

If it is loaded you have successfully deployed you maven web project In your ec2 instance

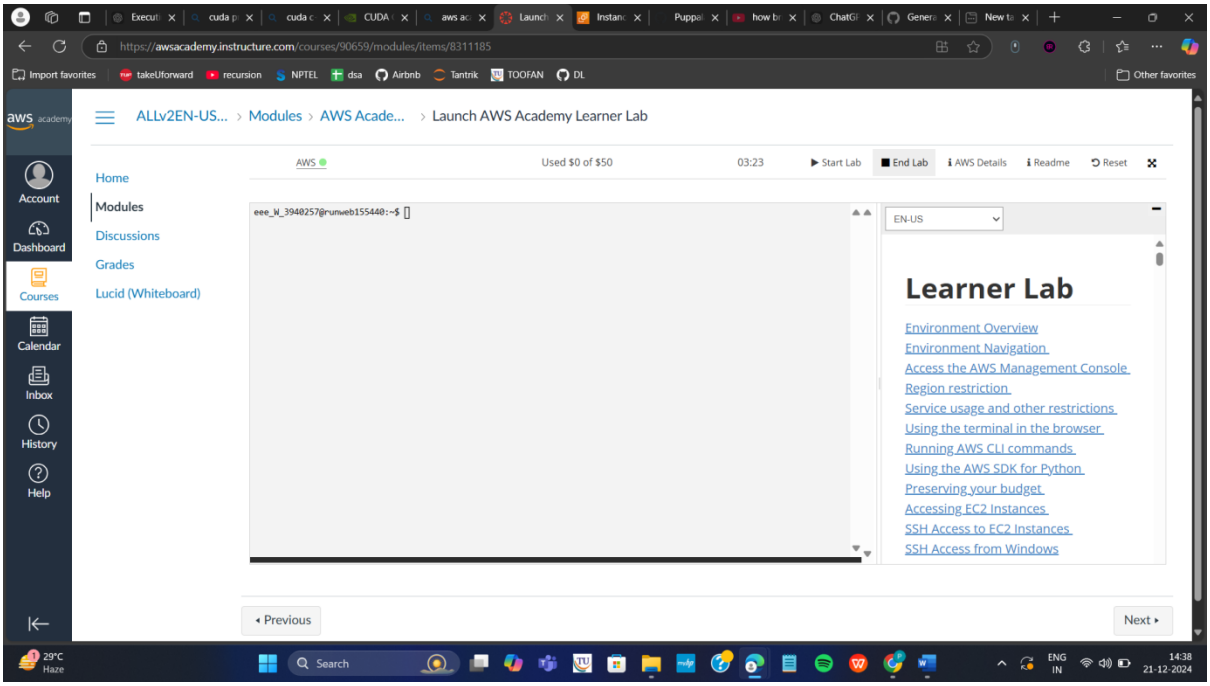
Run the following commands to stop the container

```
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$ sudo docker ps
CONTAINER ID   IMAGE            COMMAND             CREATED         STATUS         PORTS                                                NAMES
e83304610c89   mavenwebproject   "catalina.sh run"   9 minutes ago   Up 9 minutes   0.0.0.0:9090->8080/tcp, :::9090->8080/tcp   keen_elbakyan
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$ sudo docker stop e83304610c89
e83304610c89
ubuntu@ip-172-31-92-238:~/SampleMavenWebProject$
```

Terminate your instance


Instances (1/2) Info
Last updated 3 minutes ago
Connect
Instance state
Actions
Launch instances
Stop instance
Start instance
Reboot instance
Hibernate instance
Terminate (delete) instance
MavenWebPro... i-090ce930c7cc2155d Running t2.micro
MavenWebPro... i-0f8ee93e7e947cb36 Running t2.micro
us-east-1c
us-east-1d
i-090ce930c7cc2155d (MavenWebProjectExample)
sgr-0e3eea5774341aee8 9090 TCP 0.0.0.0/0 launch-wizard
sgr-0a364df2380486ba1 443 TCP 0.0.0.0/0 launch-wizard
sgr-07d05f05d7474f15d 22 TCP 0.0.0.0/0 launch-wizard
sgr-0d0fbc3233a424968 80 TCP 0.0.0.0/0 launch-wizard
Outbound rules

End your lab


ALLv2EN-US... Modules AWS Acade... Launch AWS Academy Learner Lab
Used $0 of $50
03:23
Start Lab
End Lab
AWS Details
Readme
Reset
Home
Modules
Discussions
Grades
Lucid (Whiteboard)
Learner Lab
Environment Overview
Environment Navigation
Access the AWS Management Console
Region restriction
Service usage and other restrictions
Using the terminal in the browser
Running AWS CLI commands
Using the AWS SDK for Python
Preserving your budget
Accessing EC2 Instances
SSH Access to EC2 Instances
SSH Access from Windows
Previous
Next