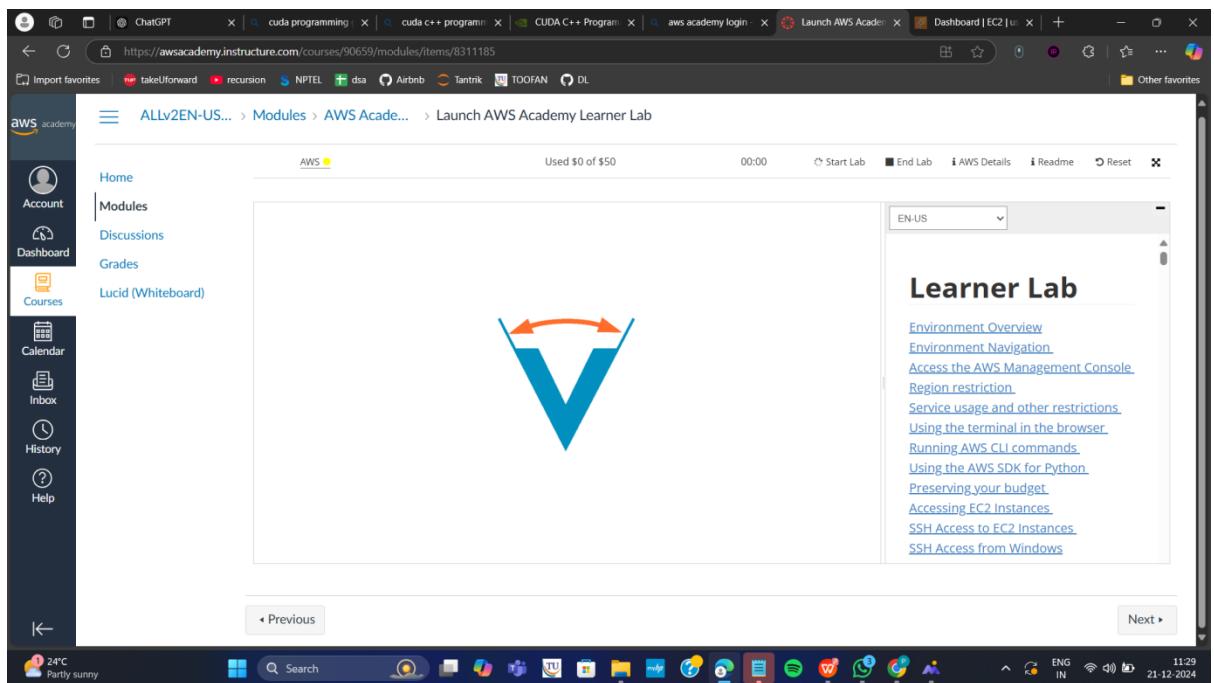


## WEEK-12

### 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 Launch AWS Academy Lab-
  - Click on start Lab and wait AWS  becomes from red to green AWS 



## Click on AWS

ALLv2EN-US... > Modules > AWS Academ... > Launch AWS Academy Learner Lab

AWS

Used \$0 of \$50 04:00 Start Lab End Lab AWS Details Readme Reset

eee\_nl\_394025@runweb155453:-\$

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 >

## Step 2: Click on EC2 to create instance

Console Home Info

Reset to default layout + Add widgets

Recently visited Info

- EC2

View all services

Applications (0) Info

Region: US East (N. Virginia)

us-east-1 (Current Region) Find applications

Name Description Region Originat. ★ ▲

No applications Get started by creating an application.

Create application

Welcome to AWS AWS Health Info Cost and usage Info

Go to myApplications

## Click on Launch Instance

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

Resources

Instances (running)	0	Auto Scaling Groups	0	Capacity Reservations	0
Dedicated Hosts	0	Elastic IPs	0	Instances	0
Key pairs	1	Load balancers	0	Placement groups	0
Security groups	1	Snapshots	0	Volumes	0

Launch instance To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance Migrate a server

Note: Your instances will launch in the US East (N. Virginia) Region

Service health AWS Health Dashboard

Region: US East (N. Virginia) Status: This service is operating normally.

Zones Zone name Zone ID

Account attributes

Default VPC vpc-0da3310938e1da95f

Settings Data protection and security Allowed AMIs Zones EC2 Serial Console Default credit specification EC2 console preferences

Explore AWS Get Up to 40% Better Price Performance T4g instances deliver the best price performance for burstable general purpose workloads in Amazon EC2. Learn more

10 Things You Can Do Today to Reduce AWS Costs Explore how to effectively manage your AWS costs without compromising on performance or capacity. Learn more

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

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Name and tags' step, a name 'MyExampleWebServer' is entered. In the 'Application and OS Images (Amazon Machine Image)' step, the 'ubuntu' button is highlighted, indicating it's selected. Other options like Amazon Linux, macOS, Windows, Red Hat, SUSE Linux, and Debian are also shown.

Select this free tier option of Ubuntu from here

The screenshot shows the 'Quick Start' section where the 'ubuntu' button is highlighted. A tooltip 'Free tier eligible' appears over the 'ubuntu' button. In the 'Summary' section, the 'Software Image (AMI)' is set to Canonical, Ubuntu, 24.04, amd64, and the 'Virtual server type (instance type)' is set to t2.micro. A tooltip 'Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free' is shown for the instance type.

Stage 3 -- Architecture as 64-bit

Make sure AMI and Architecture are there as shown

The screenshot shows the 'Architecture' dropdown set to '64-bit (x86)'. The 'AMI ID' field contains 'ami-02e8ca94b6378db8c'. The 'Username' is set to 'ubuntu' and the 'Verified provider' status is shown as green.

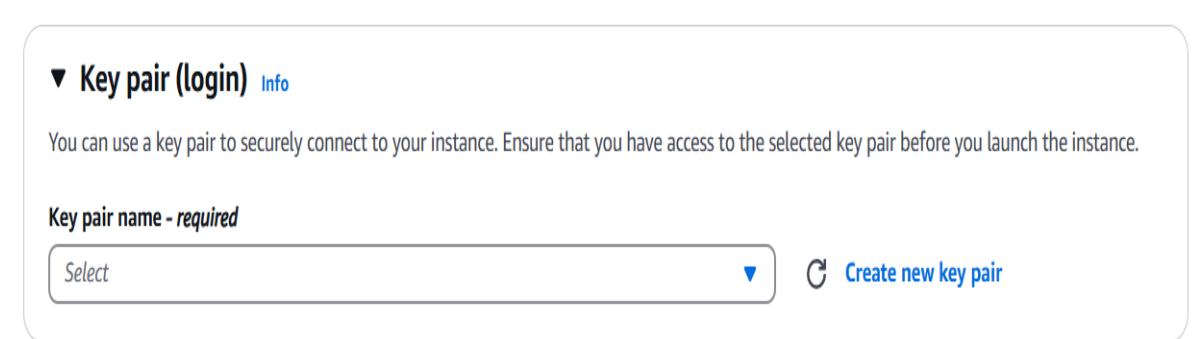
Stage 4 -- Instance type ---- t2.micro(default 1 CPU,1 GB RAM)

The screenshot shows the 'Instance type' selection page. The 't2.micro' instance type is selected, showing details: Family: t2, 1 vCPU, 1 GB Memory, Current generation: true. It is marked as 'Free tier eligible'. Other options like All generations and Compare instance types are available.

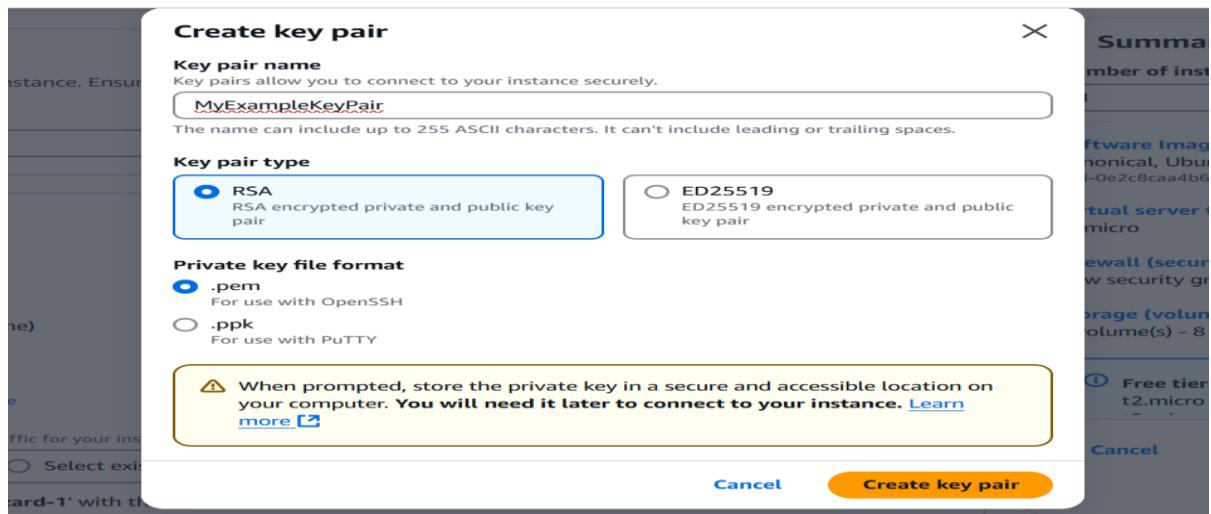
Stage 5 -- Create a new keypair--a keypair will download with extension .pem

Store key in folder AWS

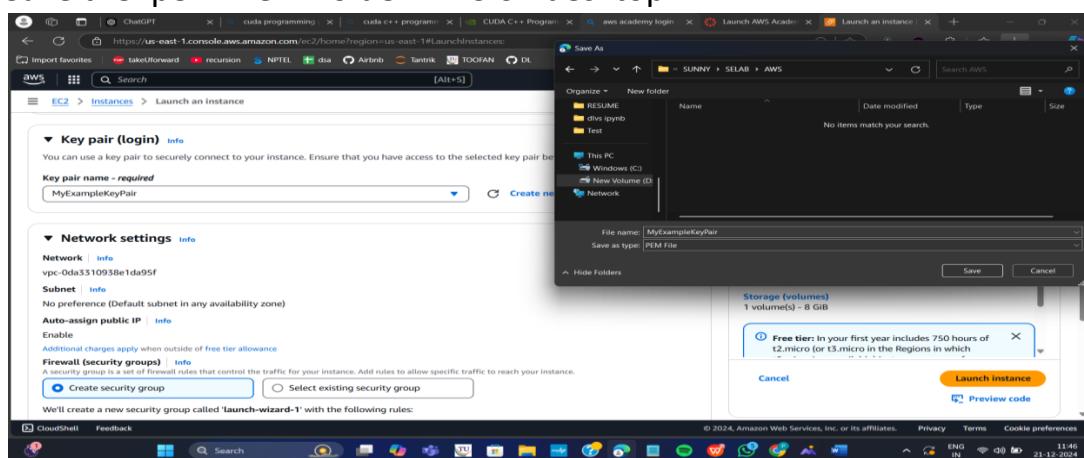
First Click on create-new key pair



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

The screenshot shows the AWS EC2 'Launch an instance' wizard. In the 'Network settings' section, there are three checkboxes under 'Firewall (security groups)': 'Allow SSH traffic from Anywhere', 'Allow HTTPS traffic from the internet', and 'Allow HTTP traffic from the internet'. All three checkboxes are checked. A callout box points to the 'Create security group' button and the checked checkboxes.

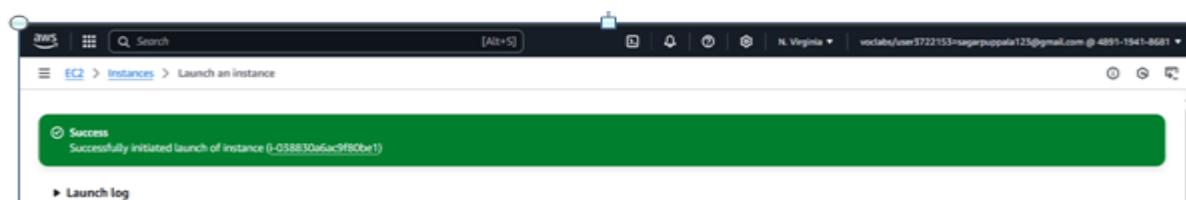
Stage 7 -- Storage - 8GB ( Observation - we have root - it is same as C Drive) it default 8 GB

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

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with navigation links like Dashboard, EC2 Global View, Events, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store. The main area is titled 'Instances (1) Info'. It shows a table with one row: Name (MyExampleWe...), Instance ID (i-038830a6ac9f80be1), Instance state (Running), Instance type (t2.micro), Status check (2/2 checks passed), Alarm status (View alarms), Availability Zone (us-east-1c), and Public IP (ec2-5-80-1). Below the table, a section titled 'Select an instance' is visible. At the top right of the main area, there are buttons for 'Connect', 'Actions', and 'Launch instances'. The status bar at the bottom indicates the instance was last updated less than a minute ago.

You have to get 2 tests passes.

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

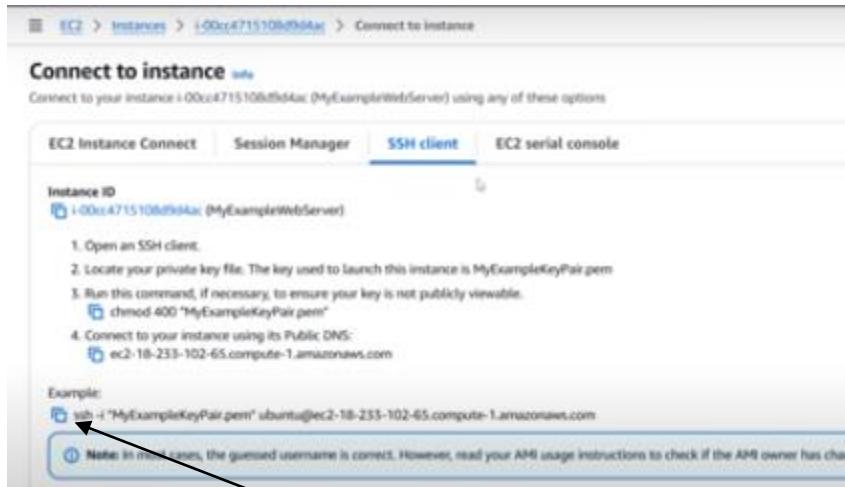
**Do this step:---once it is created select that instance**

**(Tick in checkbox) and click on connect**

This screenshot is similar to the previous one, showing the EC2 Instances page with one running instance. However, two red arrows point to the 'Connect' button at the top right of the main table area. The 'Connect' button is highlighted with a red circle. The rest of the interface and data are identical to the first screenshot.

Thus your EC2 instance is running on server.

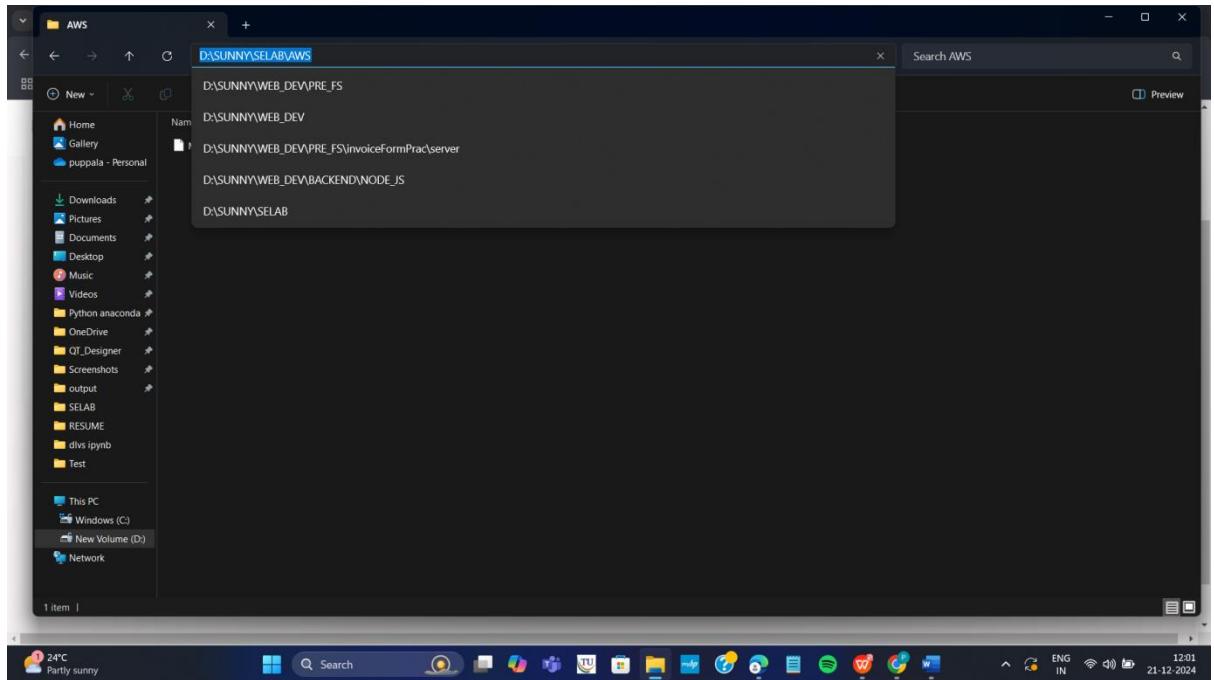
**Step 4: Now you can connect local system to server (EC2 instance) using secure shell SSH.**



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.

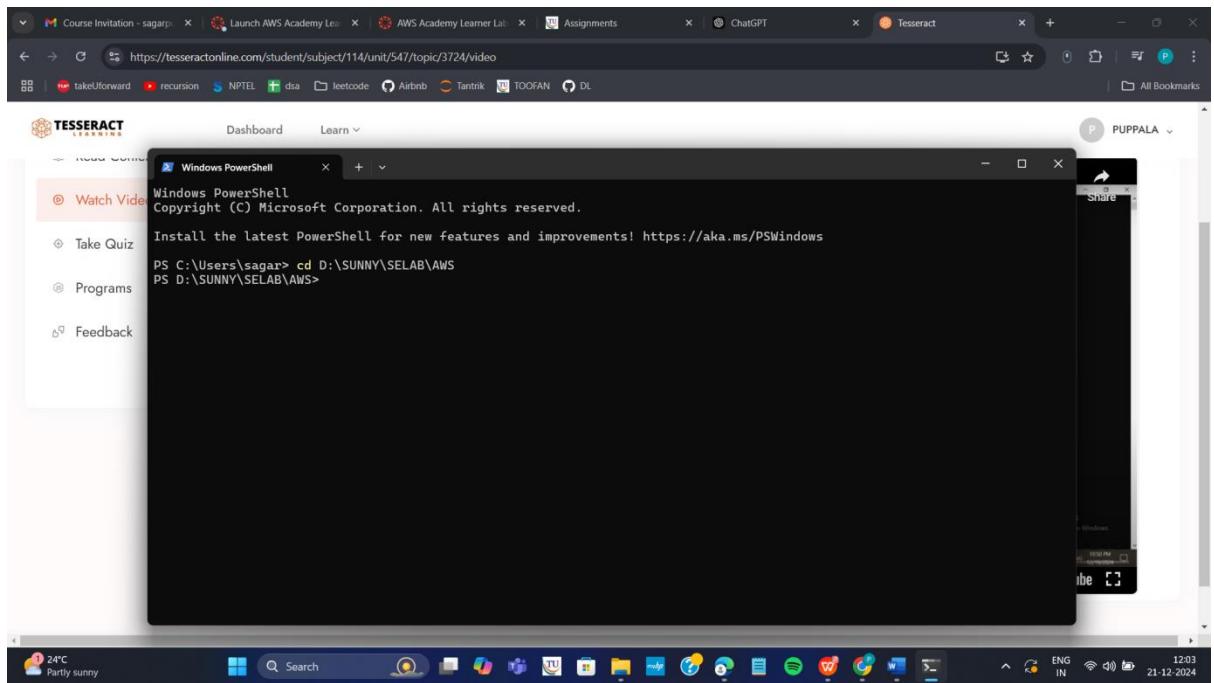
**Step a:** Copy the path of .pem file that you saved earlier in folder AWS eg. as here D:\SUNNY\SELABWEB\AWS



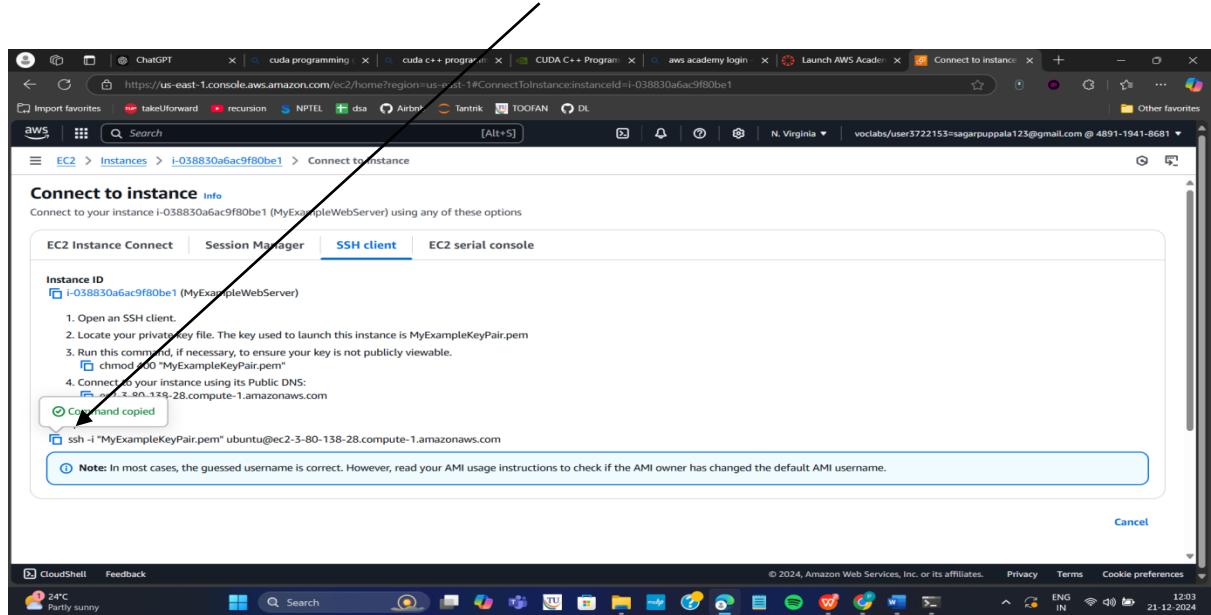
**Step b:** Open Powershell in administrative mode and navigate to that path.

Type: cd < path>

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



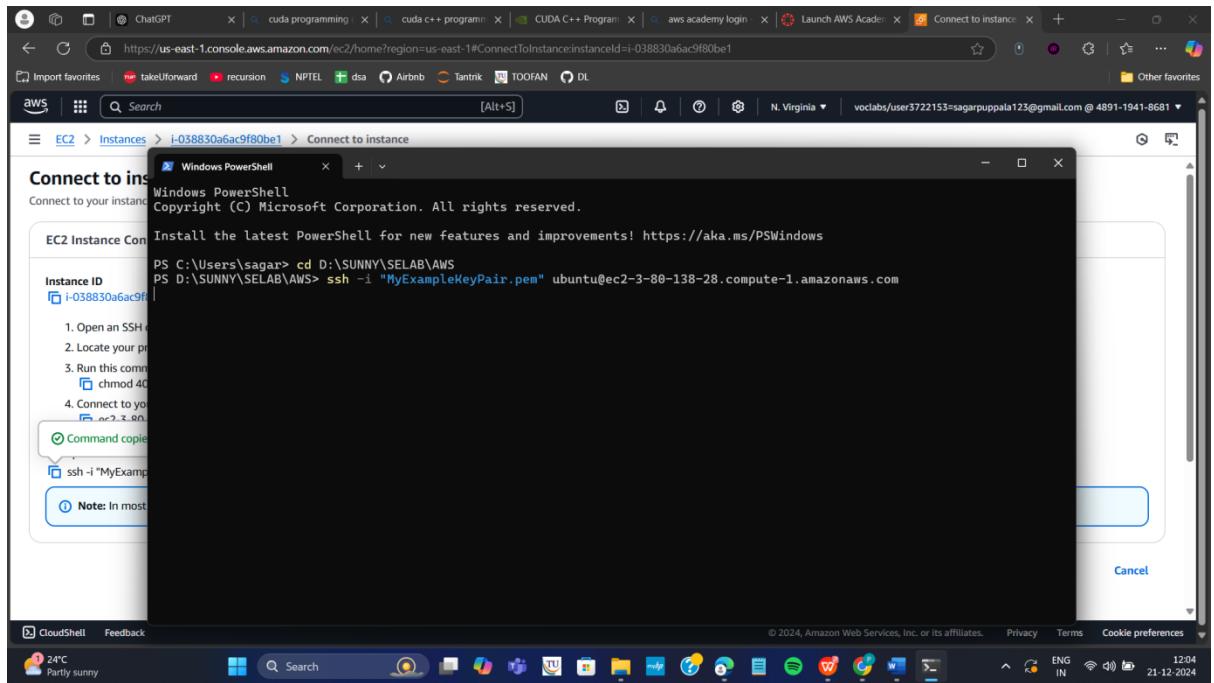
**Step c:** Go to SSH and copy the command ssh which is present at the below



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**

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-93-141:~$ sudo apt update|
```

2. Install docker by command

**sudo apt-get install docker.io**

```
0 packages can be upgraded. Run "apt list --upgradable" to see them.
ubuntu@ip-172-31-93-141:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose-v2 docker
  zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 58 not upgraded.
Need to get 80.1 MB of archives.
```

### 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.
```

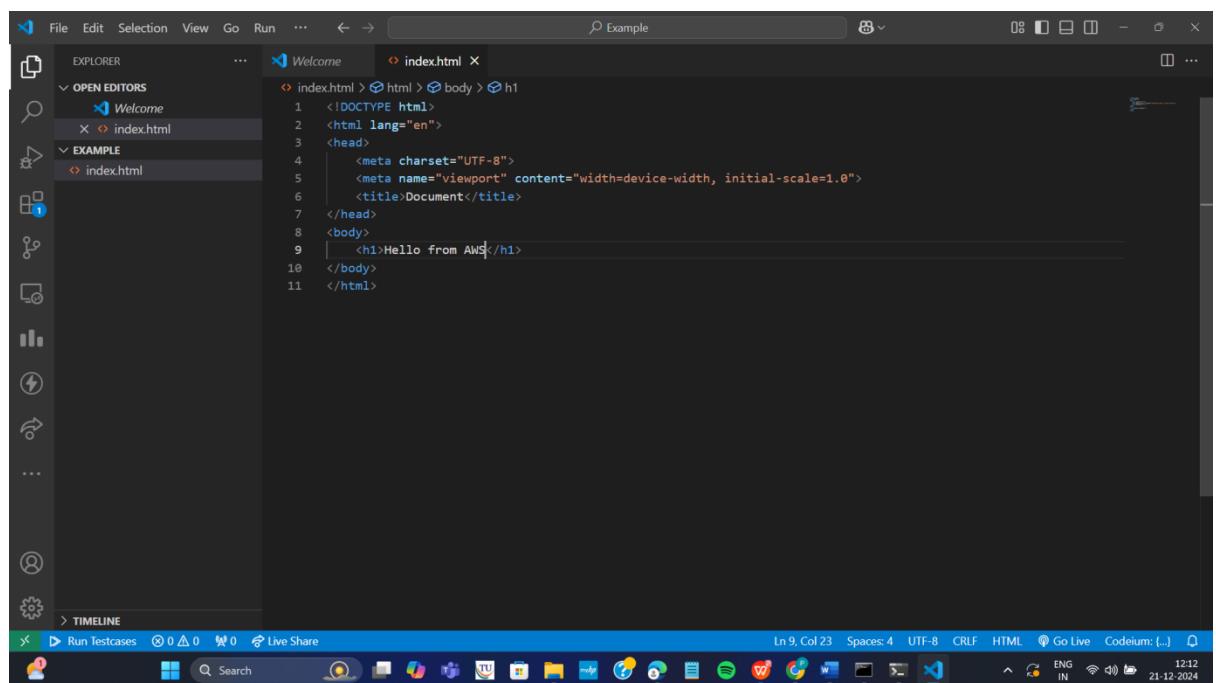
### 4. Install nano( text editor) by command

**Sudo apt install nano**

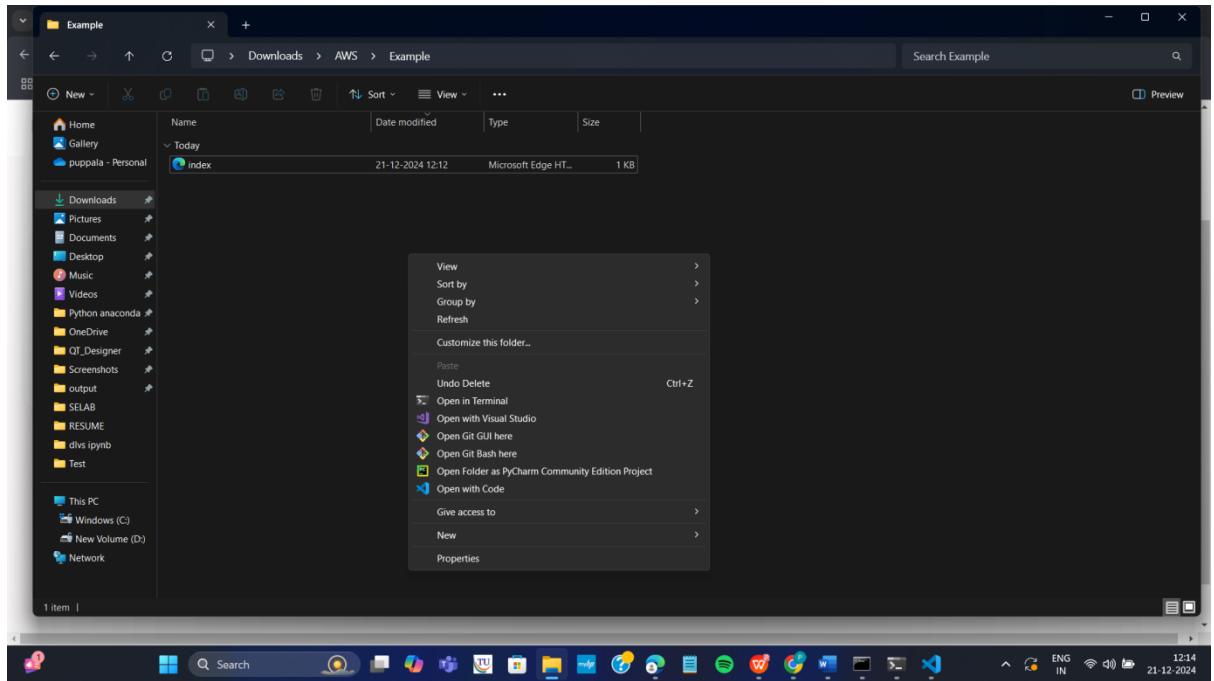
```
ubuntu@ip-172-31-93-141:~$ sudo apt install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (7.2-2ubuntu0.1).
nano set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 58 not upgraded.
```

**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 .**

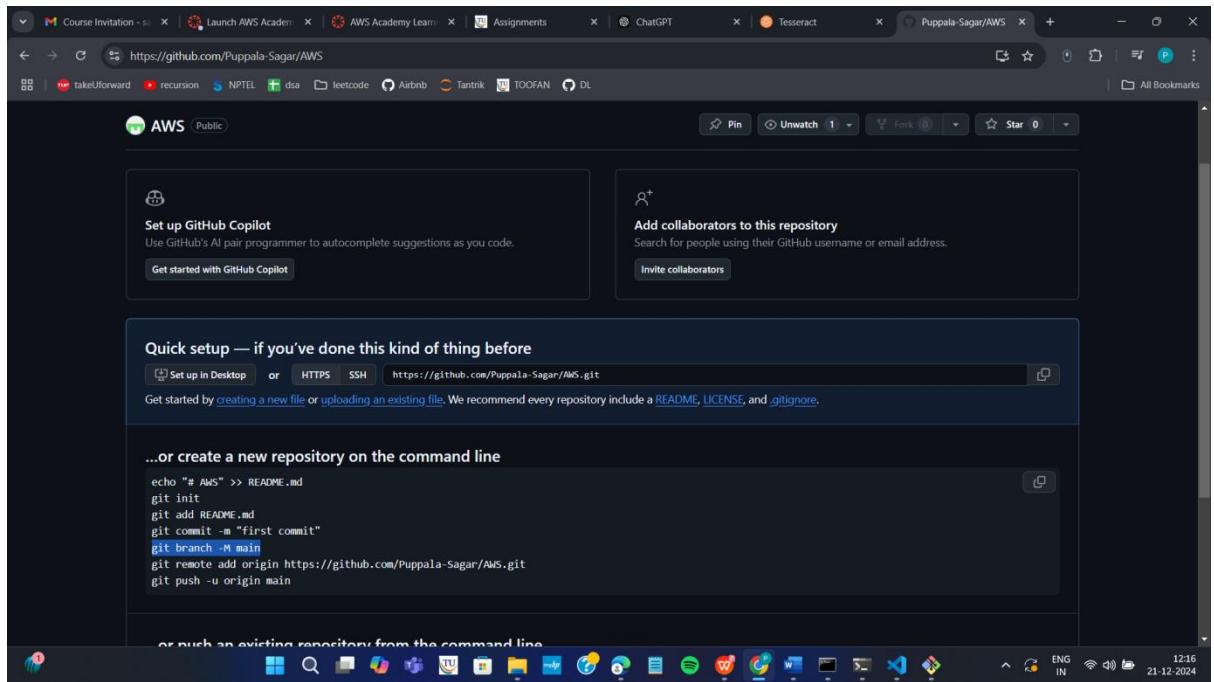
**Git commit -m "first commit"**

```
sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (master)
$ git init
Initialized empty Git repository in C:/Users/sagar/Downloads/AWS/Example/.git/
sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (master)
$ git add .

sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (master)
$ git commit -m "first commit"
[master (root-commit) f964c10] first commit
 1 file changed, 11 insertions(+)
 create mode 100644 index.html

sagar@SAGARPUPPALA MINGW64 ~/Downloads/AWS/Example (master)
$ |
```

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

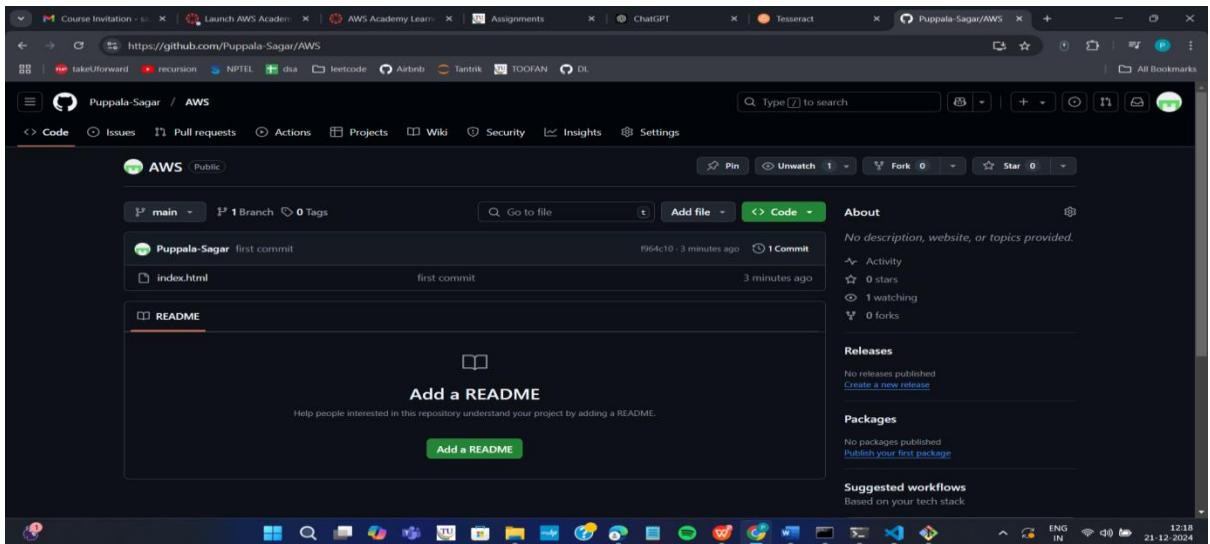


**Step e:** Copy command **one by one** from above repository and run as below

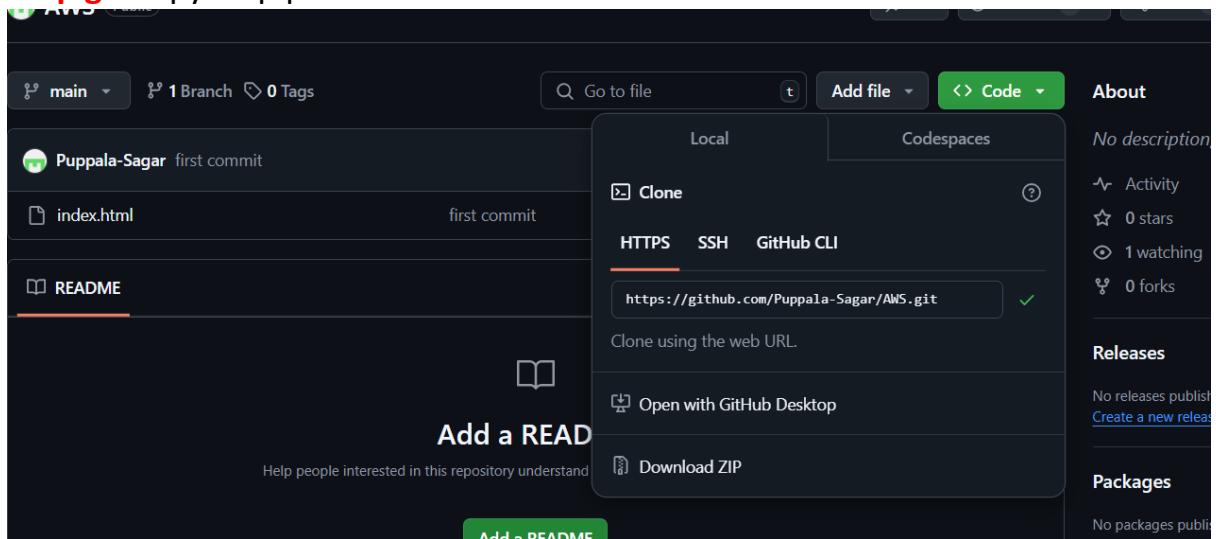
```
git branch -M main  
git remote add origin <https url>  
git push -u origin main
```

```
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

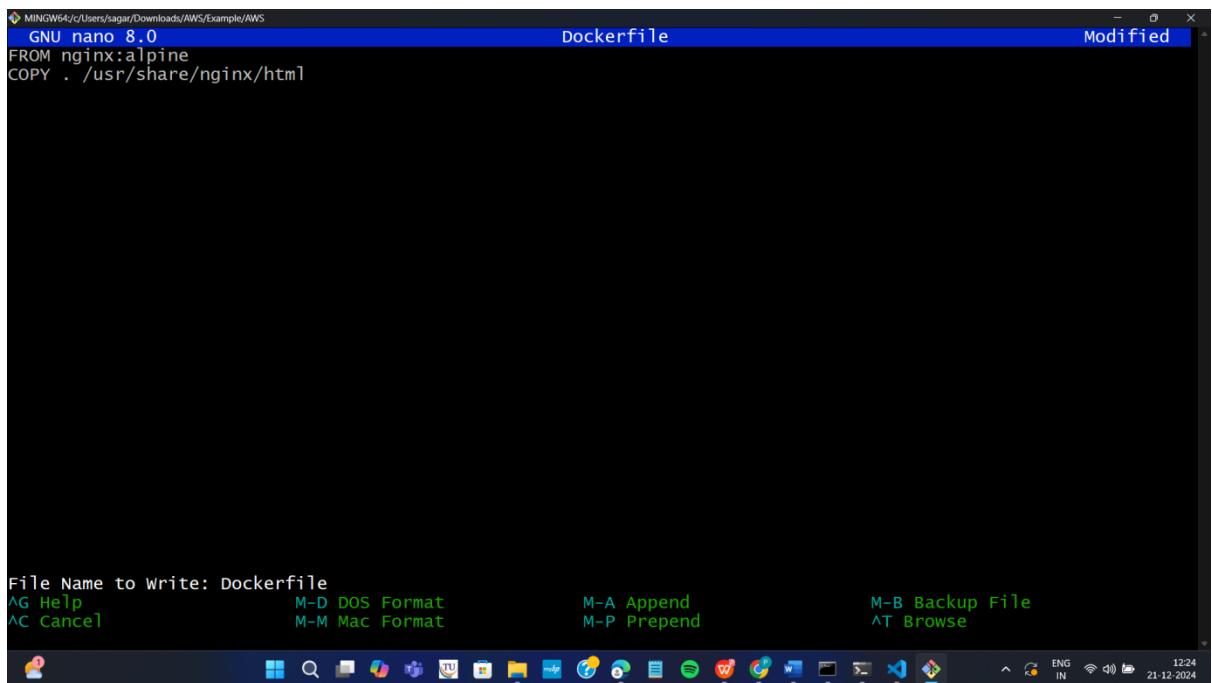
```
ubuntu@ip-172-31-93-141:~$ git clone https://github.com/Puppala-Sagar/AWS.git
Cloning into 'AWS'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)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



The screenshot shows a Windows terminal window titled "Dockerfile". The nano editor is open with the following content:

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

The terminal window has a blue header bar with the title "Dockerfile" and a "Modified" indicator. The bottom of the window shows a standard Windows taskbar with various icons for applications like File Explorer, Edge, and File Manager. A status bar at the bottom right displays "ENG IN" and the date "21-12-2024".

**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

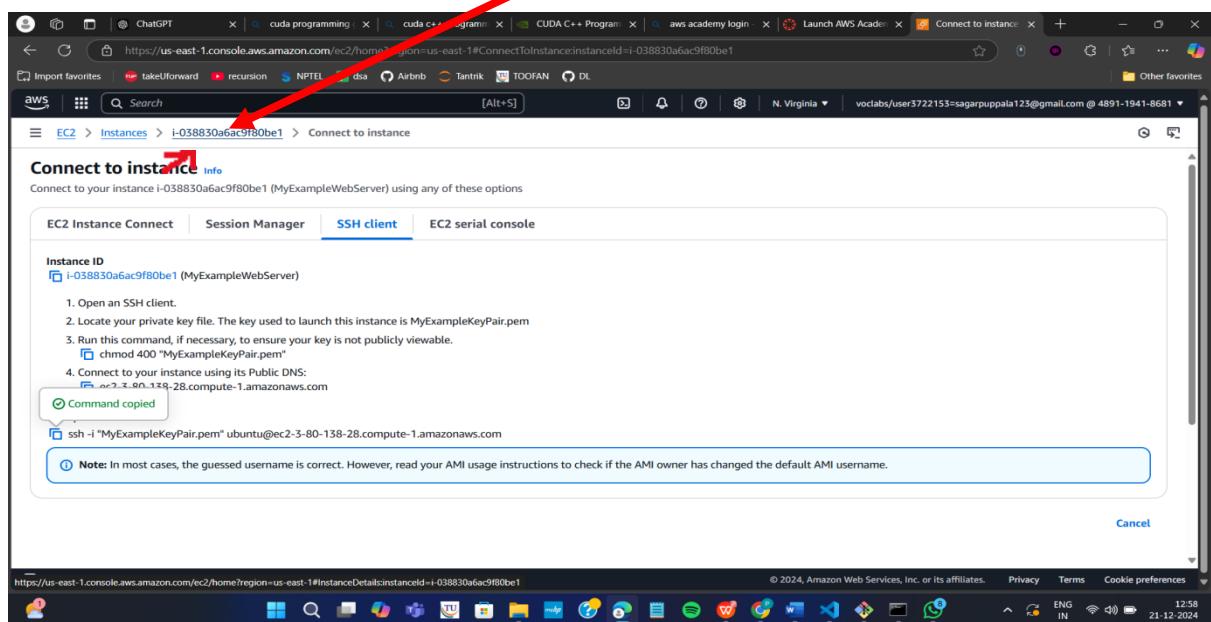
**sudo docker run -d -p 80:80 mywebapp**

```

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

Screenshot of the AWS CloudWatch Metrics Insights interface showing a query for CloudWatch Metrics Insights usage over time.

The interface includes a top navigation bar with tabs like ChatGPT, CUDA C++ programming, CUDA C++ Program, AWS Academy login, Launch AWS Academy, Instance details | EC2, and others. The URL is https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#functionDetails:functionId=i-038830a6ac9f80be1.

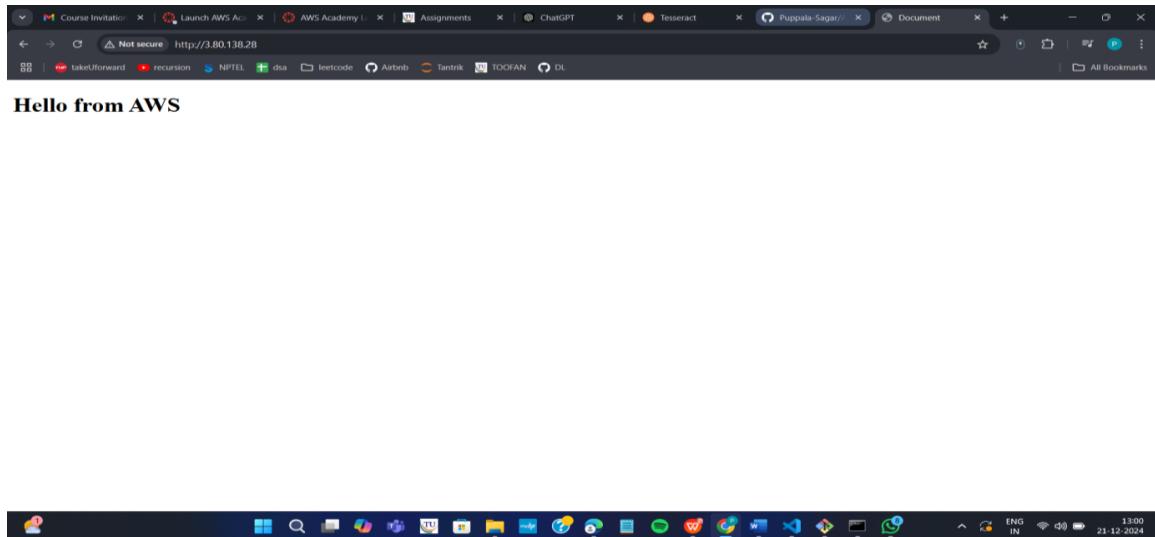
The main content area shows a table with columns for Time, Function Name, and Duration. The data shows a single entry for "CloudWatch Metrics Insights" with a duration of 0.000000 seconds.

Time	Function Name	Duration
2024-07-10T12:00:00Z	CloudWatch Metrics Insights	0.000000

Below the table, there are sections for "Metrics Insights metrics" and "Metrics Insights metrics (CloudWatch Metrics Insights)" with detailed descriptions and links to CloudWatch Metrics Insights.

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

**Note :** if https :\_\_ won't work then type just http:\_\_



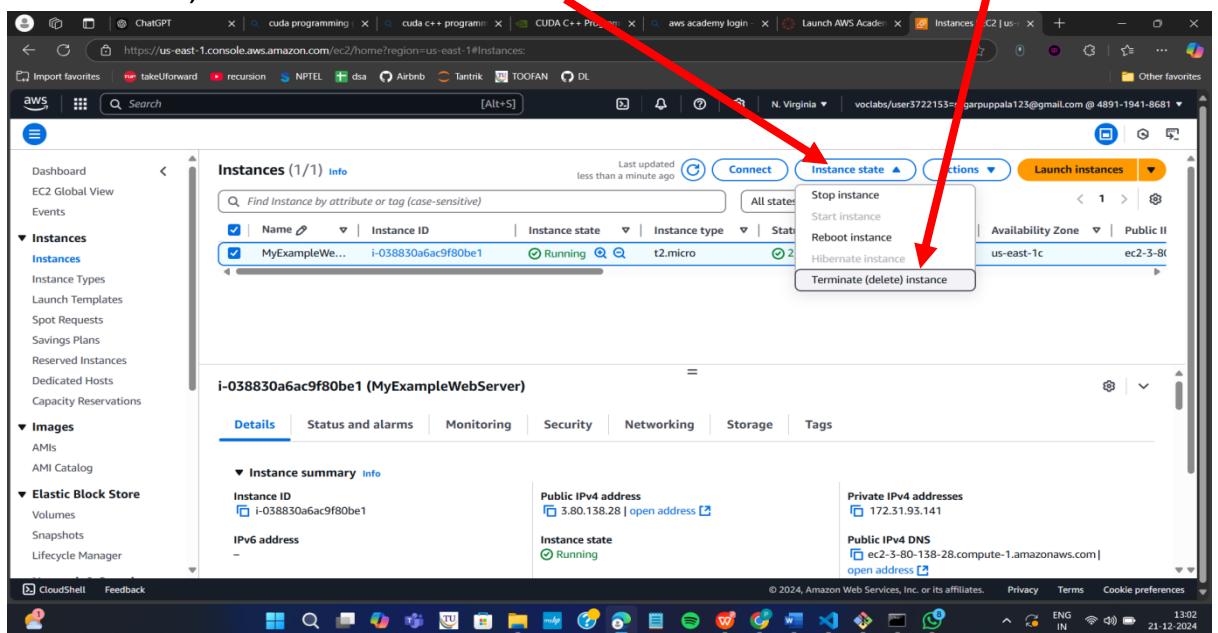
**Important do it without fail. Step Q: stop container**

Run the following command to stop the container

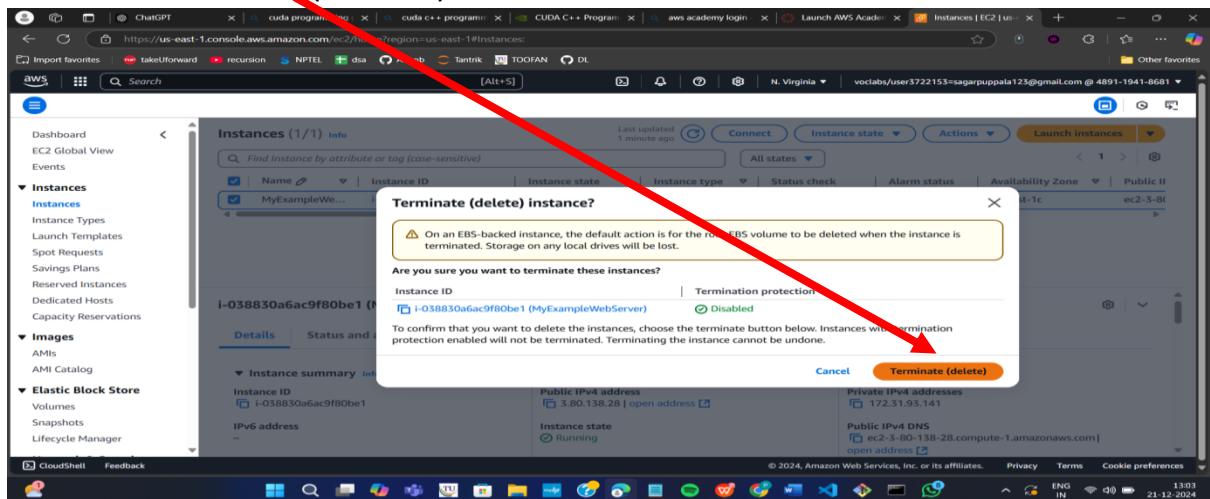
- i. sudo docker ps
- ii. sudo docker stop <container-id>

```
ubuntu@ip-172-31-93-141:~/AWS$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
12429f3ea7b7 mywebapp "/docker-entrypoint..." 3 minutes ago Up 3 minutes 0.0.0.0:80->80/tcp, ::80->80/tcp laughing_allen
12429f3ea7b7
ubuntu@ip-172-31-93-141:~/AWS$ sudo docker stop 12429f3ea7b7
ubuntu@ip-172-31-93-141:~/AWS$
```

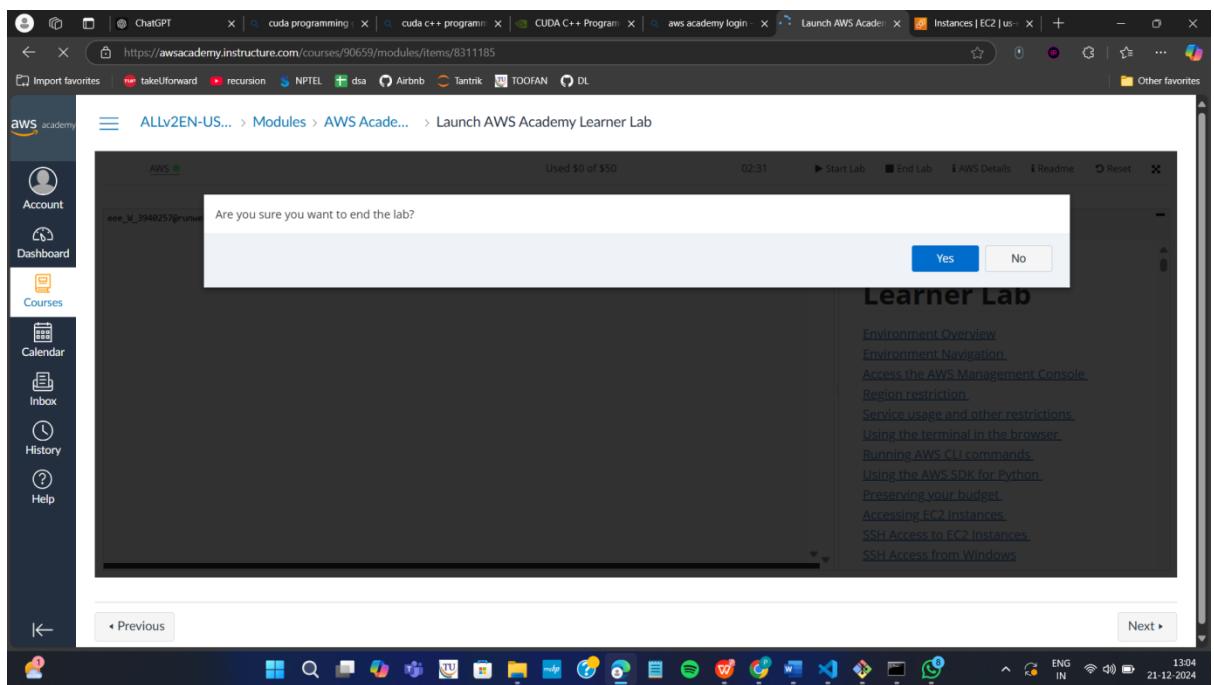
- iii. Go back to instances and click on option instance. Once load as below, then Click on Instance state and select terminate instances



Click on Terminated (deleted)



iv. Now click on End lab

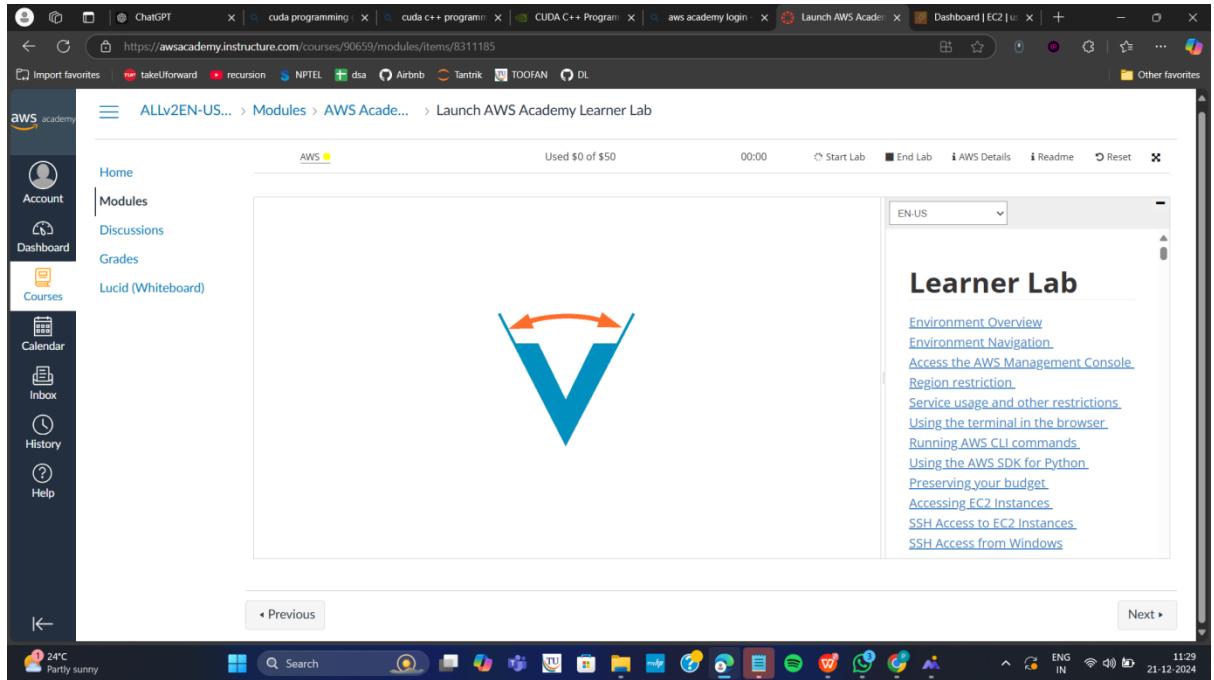


End of exercise 1 (DEPLOYMENT of index.html using EC2 instance in AWS) consisting of:

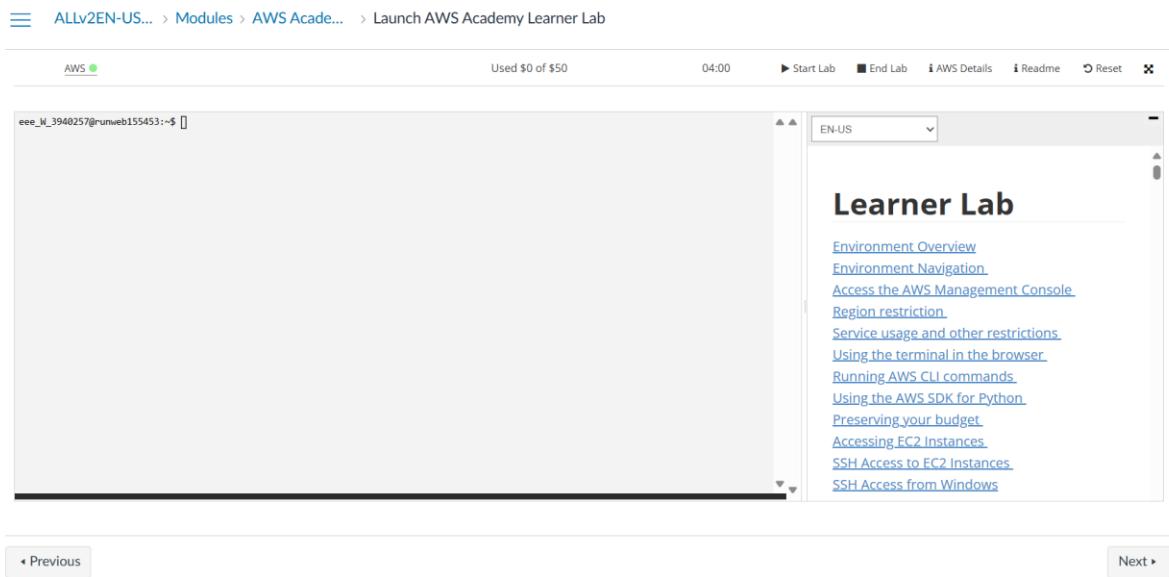
1. creating EC2 instance to install Ubuntu, docker, git , and nano.
2. creating web application (index.html) using Visual studio
3. Pushing index.html to github and cloning the it
4. Creating Dockerfile to build image for running index.html on nginx server
5. Run it as container using docker command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

The screenshot shows the AWS Console Home page. On the left, there's a sidebar with 'Recently visited' (EC2) and other service links like Welcome to AWS, AWS Health, and Cost and usage. The main area has a heading 'Applications (0)' with a 'Create application' button. Below it, there's a message: 'Get started by creating an application.' and another 'Create application' button. At the bottom, there are 'View all services', 'Go to myApplications', and navigation icons.

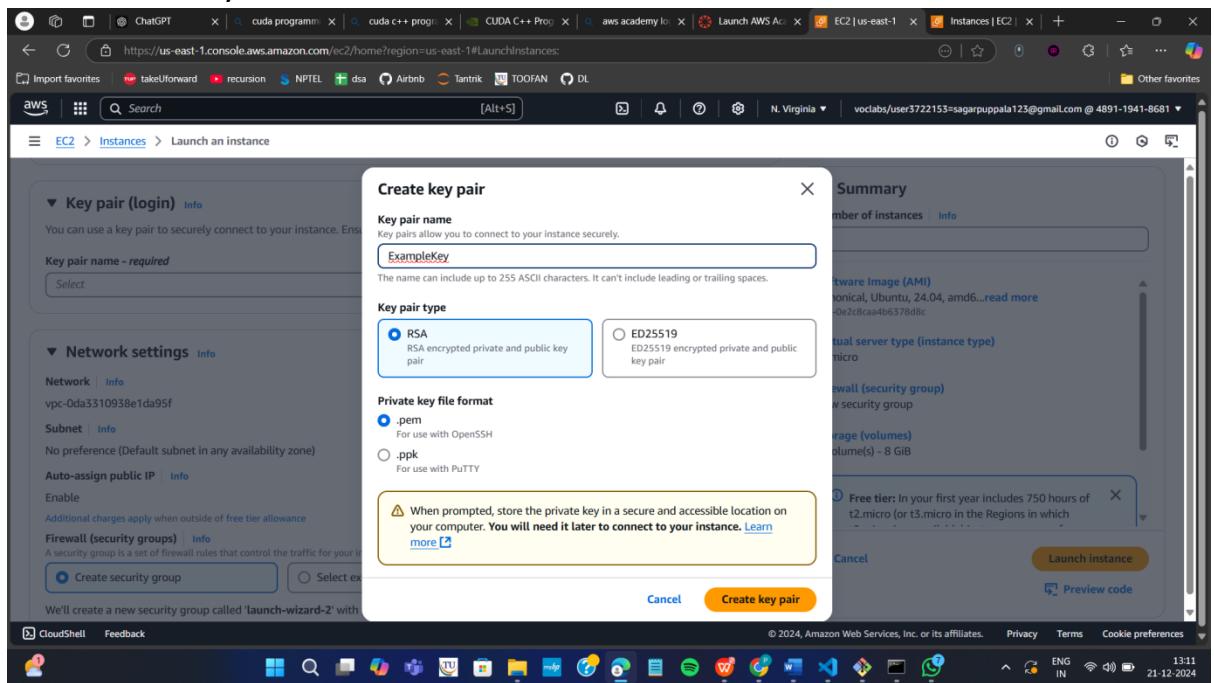
Click on launch instance

This screenshot shows the AWS EC2 Instances dashboard. The left sidebar includes sections for Dashboard, Instances (with sub-options like Instances, Instance Types, Launch Templates, etc.), Images, and Elastic Block Store. The main content area has a 'Resources' summary table and a 'Launch instance' section with 'Launch instance' and 'Migrate a server' buttons. It also features a 'Service health' card for AWS Health Dashboard and a 'Zones' table. The right side has a 'Account attributes' panel and an 'Explore AWS' section with various tips and links.

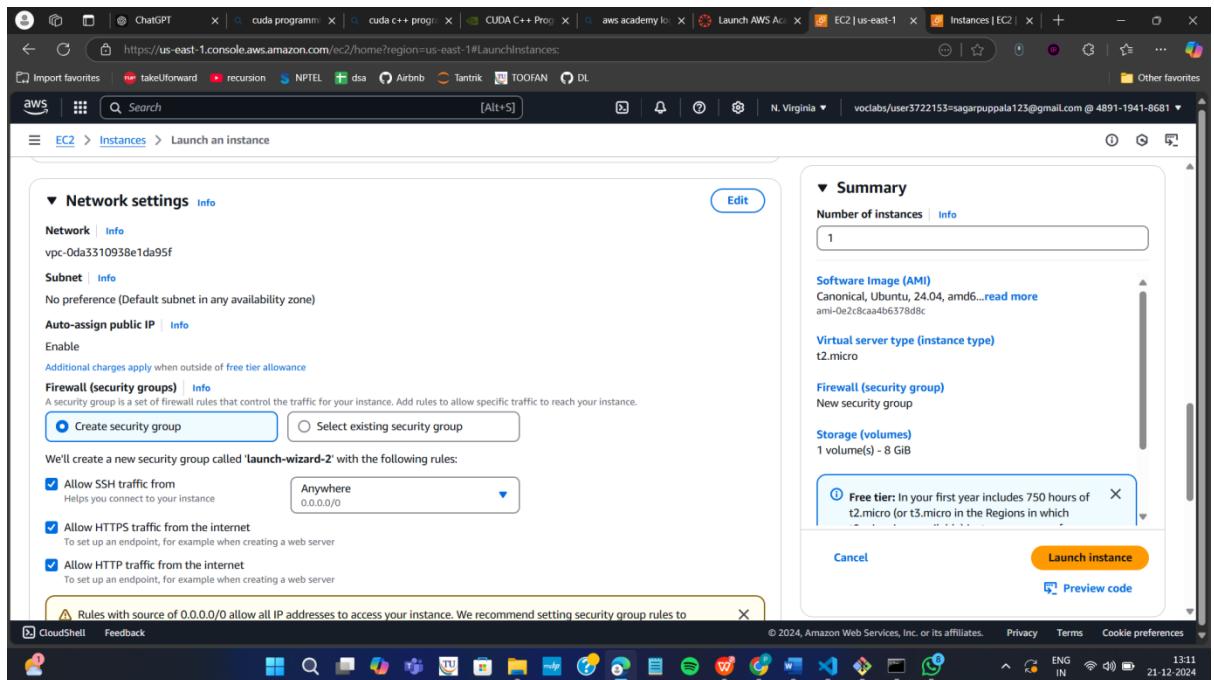
Give name

This screenshot shows the 'Launch an instance' wizard. Step 1 is 'Name and tags'. It has a 'Name' field containing 'MavenWebProjectExample' and a 'Add additional tags' button. Step 2 is 'Application and OS Images (Amazon Machine Image)'. It shows a search bar, a 'Recent' tab, and a 'Quick Start' tab. Under 'Recent', there are cards for Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. A 'Browse more AMIs' button is available. Step 3 is 'Summary', which shows 'Number of instances: 1'. It includes sections for Software Image (Amazon Linux 2023.6.2...), Virtual server type (t2.micro), Firewall (New security group), Storage (1 volume(s) - 8 GiB), and a note about Free tier. Buttons for 'Cancel', 'Launch instance', and 'Preview code' are at the bottom.

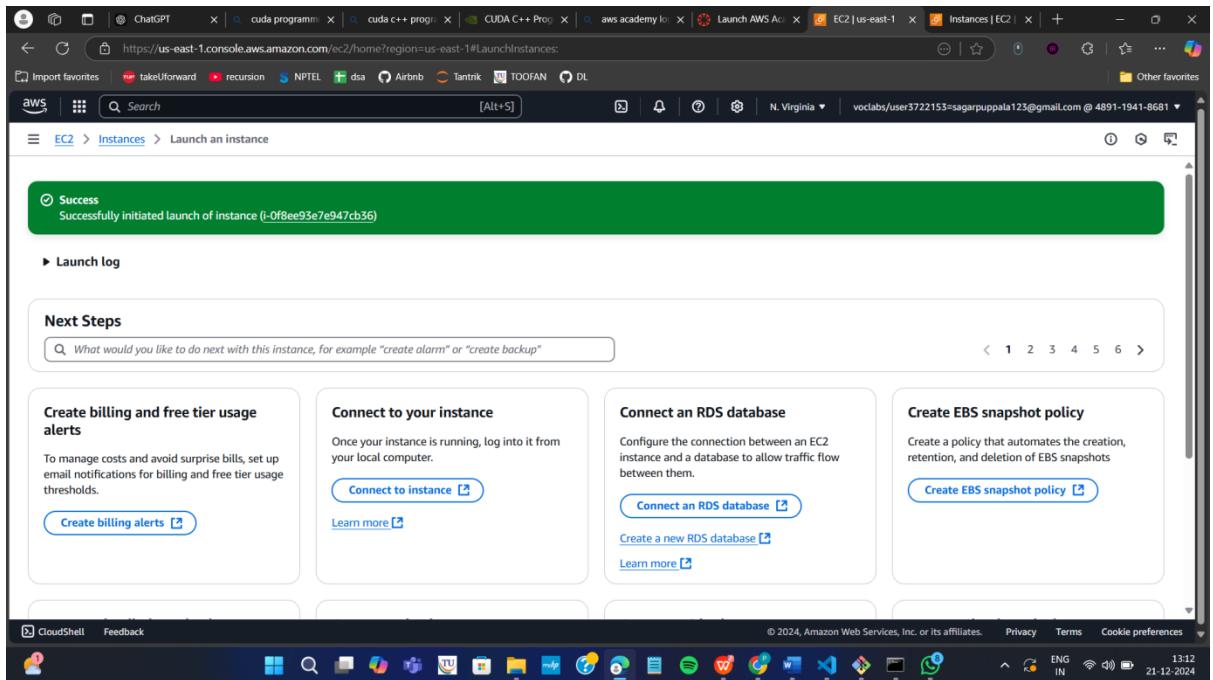
## Create new key



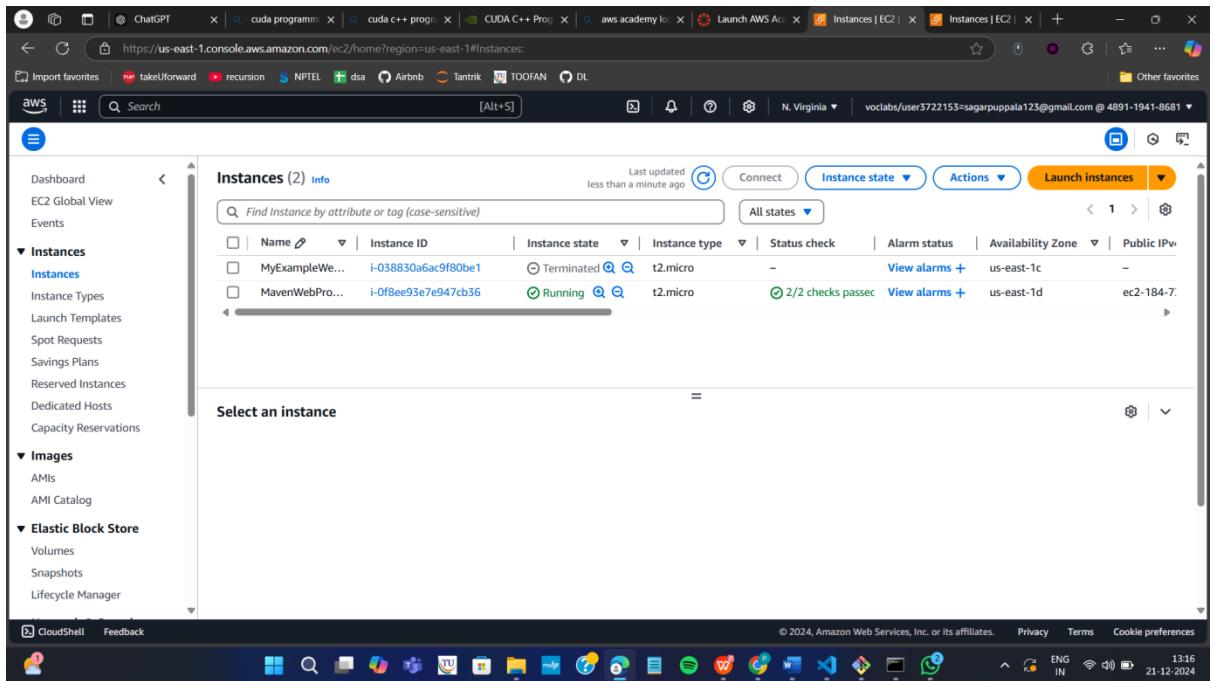
Check all the boxes under network and click on launch instance



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

```
C:\Users\sagar\Downloads\AWS>ssh -i "ExampleKey.pem" ubuntu@ec2-184-73-131-86.compute-1.amazonaws.com
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1018-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Sat Dec 21 07:48:56 UTC 2024

System load:  0.0          Processes:      104
Usage of /:   24.7% of 6.71GB  Users logged in:  0
Memory usage: 21%          IPv4 address for enX0: 172.31.22.244
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

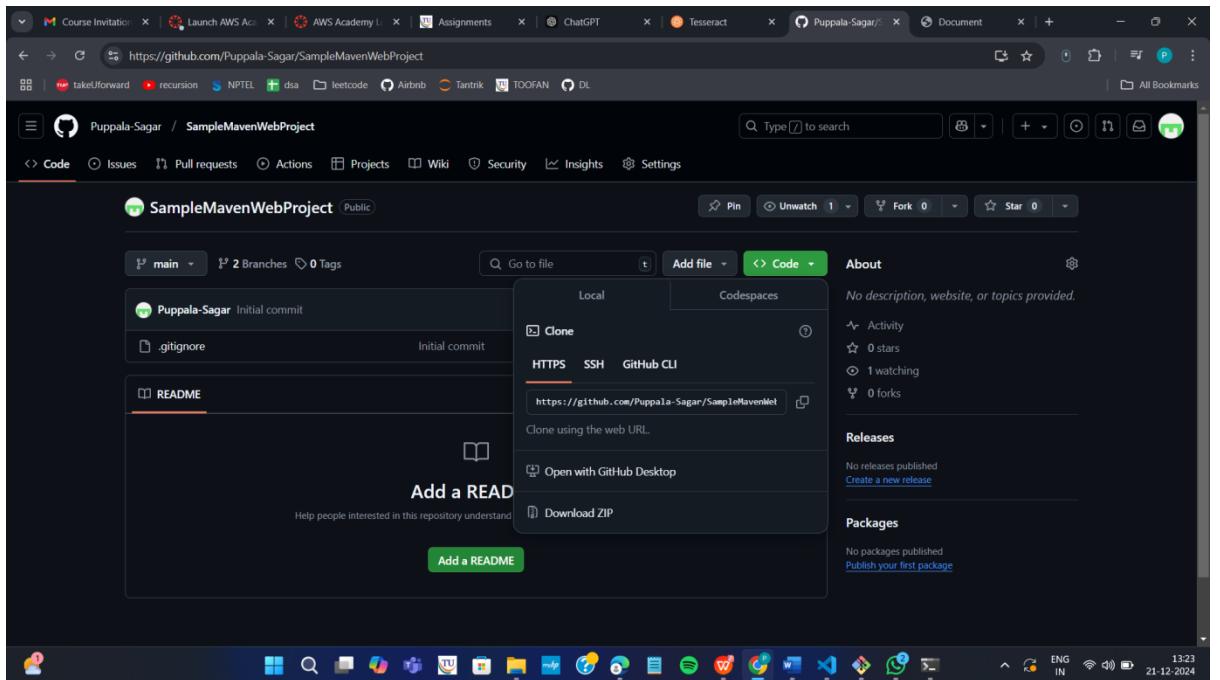
Enable ESM Apps to receive additional future security updates.
```

Run the following commands

```
ubuntu@ip-172-31-22-244:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [572 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [59]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [1]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [111 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [1]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [1]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [1]

ubuntu@ip-172-31-22-244:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-buildx docker-compose
  zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 58 not upgraded.
ubuntu@ip-172-31-22-244:~$ sudo apt install git
Command 'sudp' not found, did you mean:
  command 'ssdp' from snap ssdp (0.0.1)
  command 'sfdf' from deb graphviz (2.42.2-9ubuntu0.1)
  command 'sudo' from deb sudo (1.9.14p2-1ubuntu1)
  command 'sudo' from deb sudo-ldap (1.9.14p2-1ubuntu1)
  command 'sup' from deb sup (20100519-3)
See 'snap info <snapname>' for additional versions.
ubuntu@ip-172-31-22-244:~$ 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.
ubuntu@ip-172-31-22-244:~$ |
```

**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.
```

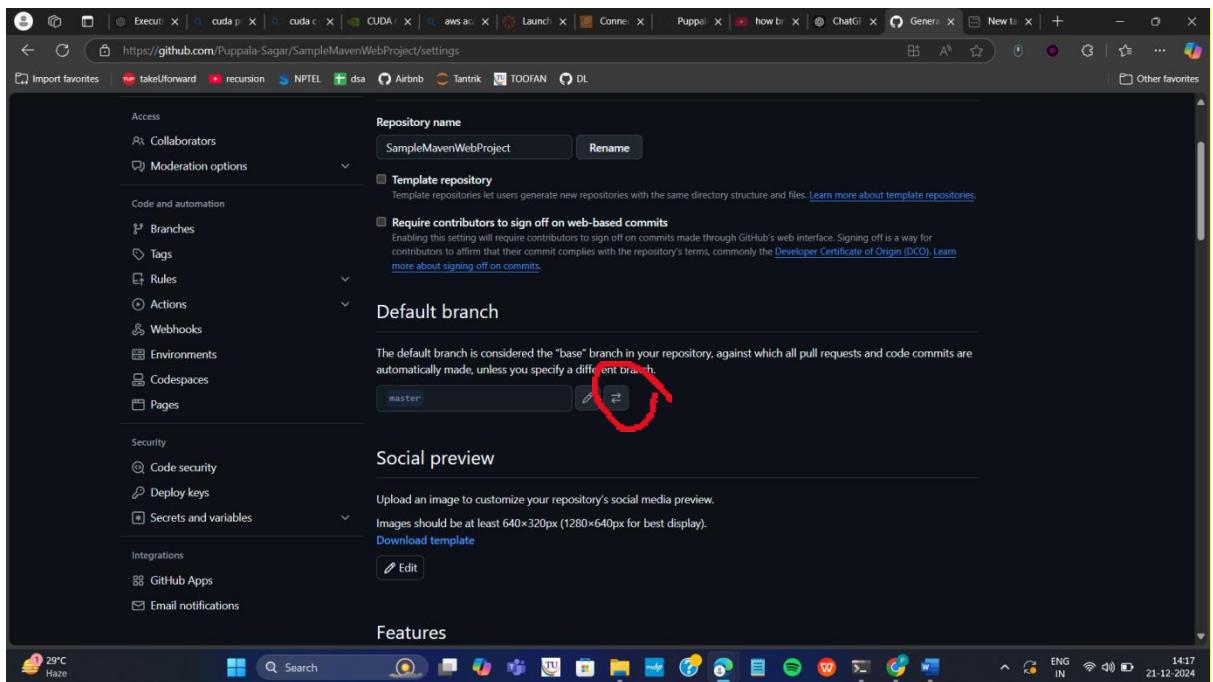
If your repository is in main run following

```
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

```

Successfully tagged mavenwebproject:latest
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

The screenshot shows the AWS Management Console with the EC2 Instances page open. The instance summary for the instance `i-090ce930c7cc2155d` is displayed. Key details shown include:

- Public IPv4 address:** 54.87.134.240
- Instance state:** Running
- Private IP DNS name (IPv4 only):** ip-172-31-92-238.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-090ce930c7cc2155d

Enter this url in browser and click enter

The screenshot shows a browser window with two tabs open:

- Tab 1: `54.87.134.240:9090\SampleMavenWebProject`
- Tab 2: `54.87.134.240:9090/SampleMavenWebProject`

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

EC2 > Instances > i-090ce930c7cc2155d

**IAM Role**  
IMDSv2 Required

**Subnet ID**  
subnet-03dfd34535cf23e91

**Instance ARN**  
arn:aws:ec2:us-east-1:489119418681:instance/i-090ce930c7cc2155d

**Auto Scaling Group name**  
-

**Managed**  
false

**Details** **Status and alarms** **Monitoring** **Security** **Networking** **Storage** **Tags**

**Security details**

**IAM Role**  
-

**Owner ID**  
489119418681

**Launch time**  
Sat Dec 21 2024 14:01:45 GMT+0530 (India Standard Time)

**Security groups**  
sg-030d1847efe1b36cd (launch-wizard-3)

**Inbound rules**

Name	Security group rule ID	Port range	Protocol	Source	Security groups

**Click on edit inbound rules**

sg-030d1847efe1b36cd - launch-wizard-3

**Details**

**Security group name**  
launch-wizard-3

**Security group ID**  
sg-030d1847efe1b36cd

**Description**  
launch-wizard-3 created 2024-12-21T08:31:06.033Z

**VPC ID**  
vpc-0da3310938e1da95f

**Owner**  
489119418681

**Inbound rules count**  
3 Permission entries

**Outbound rules count**  
1 Permission entry

**Inbound rules** **Outbound rules** **Sharing - new** **VPC associations - new** **Tags**

**Inbound rules (3)**

Name	Security group rule ID	IP version	Type	Protocol	Port range
-	sgr-0a364df2380486ba1	IPv4	HTTPS	TCP	443
-	sgr-07d05f05d7474f15d	IPv4	SSH	TCP	22
-	sgr-0d0fb3233a424968	IPv4	HTTP	TCP	80

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

EC2 > Security Groups > sg-030d1847efe1b36cd - launch-wizard-3 > Edit inbound rules

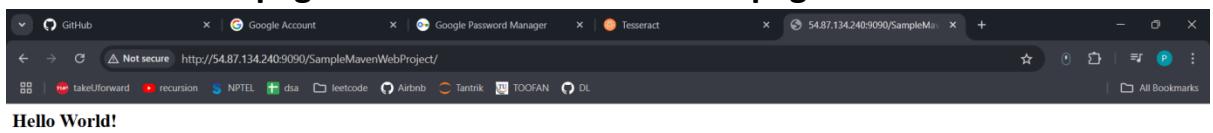
Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0e3eea5774341aee8	Custom TCP	TCP	9090	Cu... ▾	<input type="text"/> 0.0.0.0/0 <span>X</span>
sgr-0a364df2380486ba1	HTTPS	TCP	443	Cu... ▾	<input type="text"/> 0.0.0.0/0 <span>X</span>
sgr-07d05f05d7474f15d	SSH	TCP	22	Cu... ▾	<input type="text"/> 0.0.0.0/0 <span>X</span>
sgr-0d0fb3c3233a424968	HTTP	TCP	80	Cu... ▾	<input type="text"/> 0.0.0.0/0 <span>X</span>

[Add rule](#)

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

[Cancel](#) [Preview changes](#) [Save rules](#)

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



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
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

The screenshot shows the AWS Management Console for the EC2 service. The left sidebar has sections for Dashboard, EC2 Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and AMI Catalog. Under Instances, there are sub-options for Instances, Instance Types, Launch Templates, and Spot Requests. The main content area displays 'Instances (1/2) Info' with a table showing two instances. The first instance is 'MavenWebPro...' with ID 'i-090ce930c7cc2155d', state 'Running', type 't2.micro', and status '2/2'. The second instance is 'MavenWebPro...' with ID 'i-0f8ee93e7e947cb36', state 'Running', type 't2.micro', and status '2/2'. A context menu is open over the first instance, with the 'Terminate (delete) instance' option highlighted. The top navigation bar shows the URL as <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#Instances>. The bottom status bar shows the date and time as 21-12-2024.

## End your lab

The screenshot shows the AWS Academy Learner Lab interface. The left sidebar includes links for Account, Dashboard, Courses (selected), Calendar, Inbox, History, and Help. The main content area shows a terminal window with the command 'eee\_nl\_3949257@runweb155440:~\$'. To the right is a 'Learner Lab' panel with a sidebar containing links: 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, and SSH Access from Windows. At the bottom, there are 'Previous' and 'Next' buttons. The top navigation bar shows the URL as <https://awsacademy.instructure.com/courses/90659/modules/items/8311185>. The bottom status bar shows the date and time as 21-12-2024.