## Week 12: Creation of virtual machine for Ubuntu OS and Deploying the web application

1. Evaluation of previous experiments
    a. Creating, running and scaling pods in minikube (eg. nginx)
    b. Running Nagios server and Understanding the Monitoring tool using Docker.
    c. Creation of virtual machine for Ubuntu OS and deploying an web server
        (i.e. DEPLOYMENT of index.html using EC2 instance in AWS)
    d. Deploying an Maven web project application into cloud

2. Creation of virtual machine
3. Deploying the web application
4. Accessing it publicly

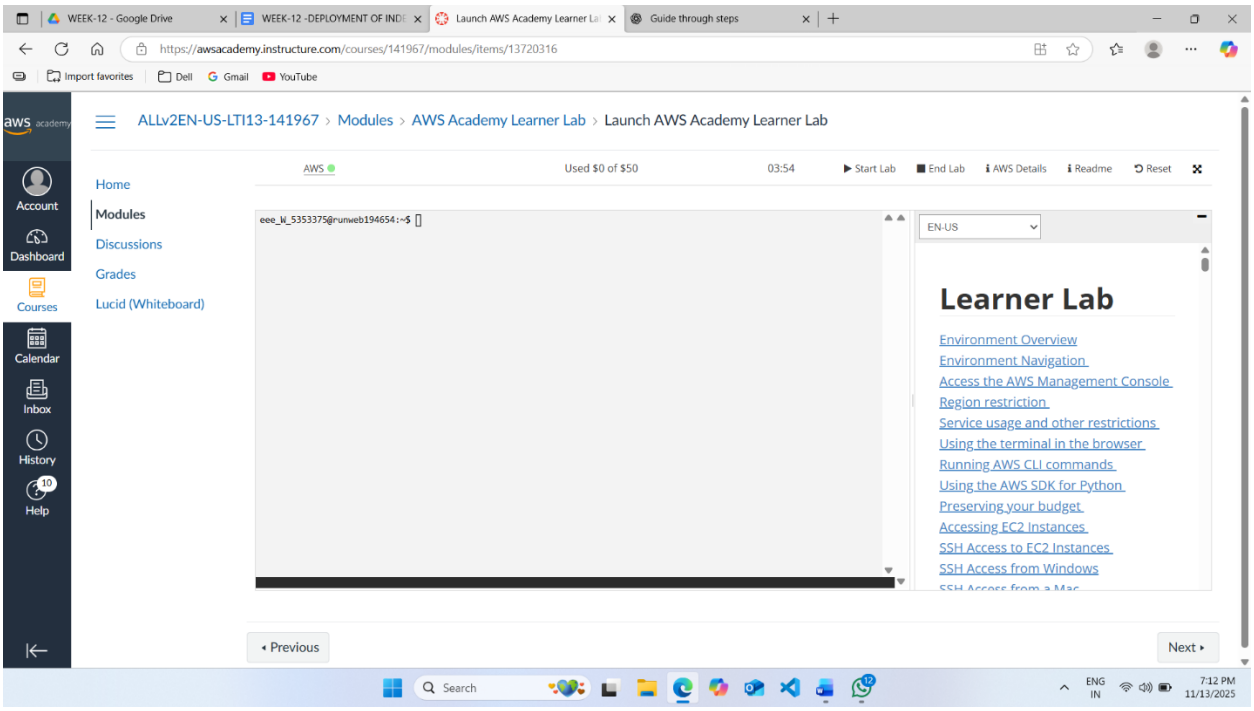
---

### Deploying an application into cloud

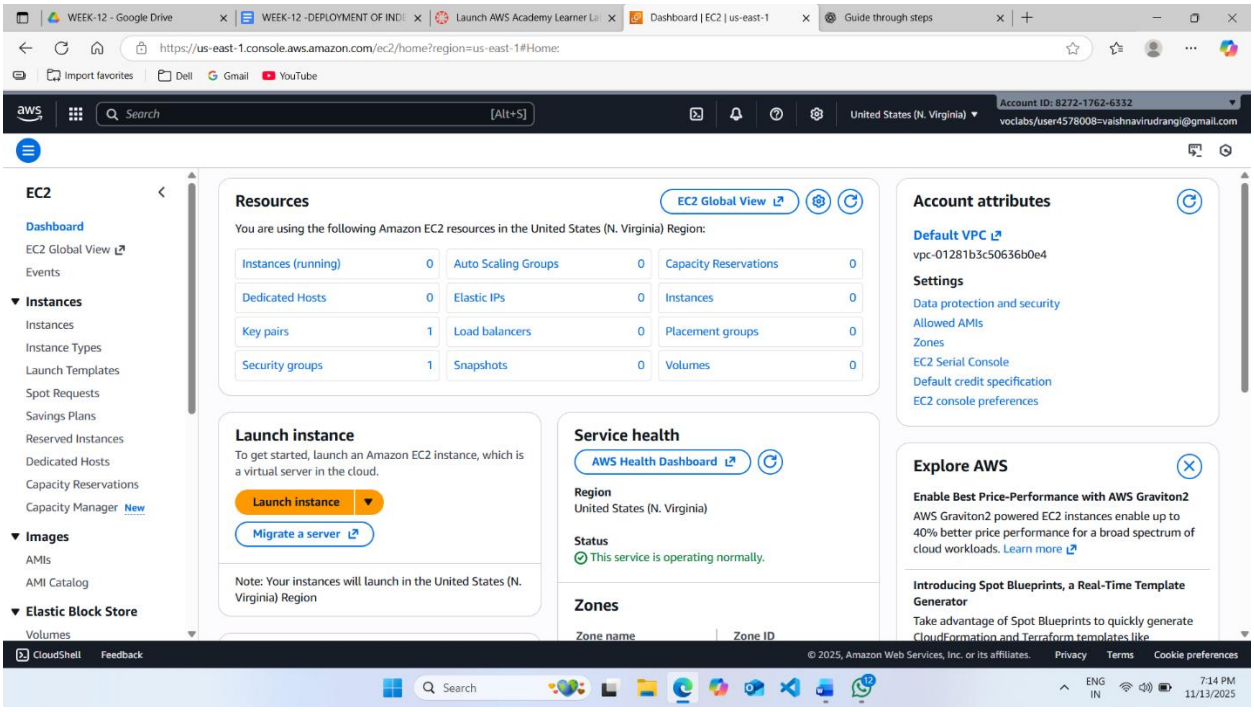
#### Steps for Deploying application into the cloud

I. Create application and Push into github
II. Create the virtual machine and connect to it.
III. Clone the application from github, Write the Dockerfile
IV. Create the image
V. Run the image and access it public ip of virtual machine

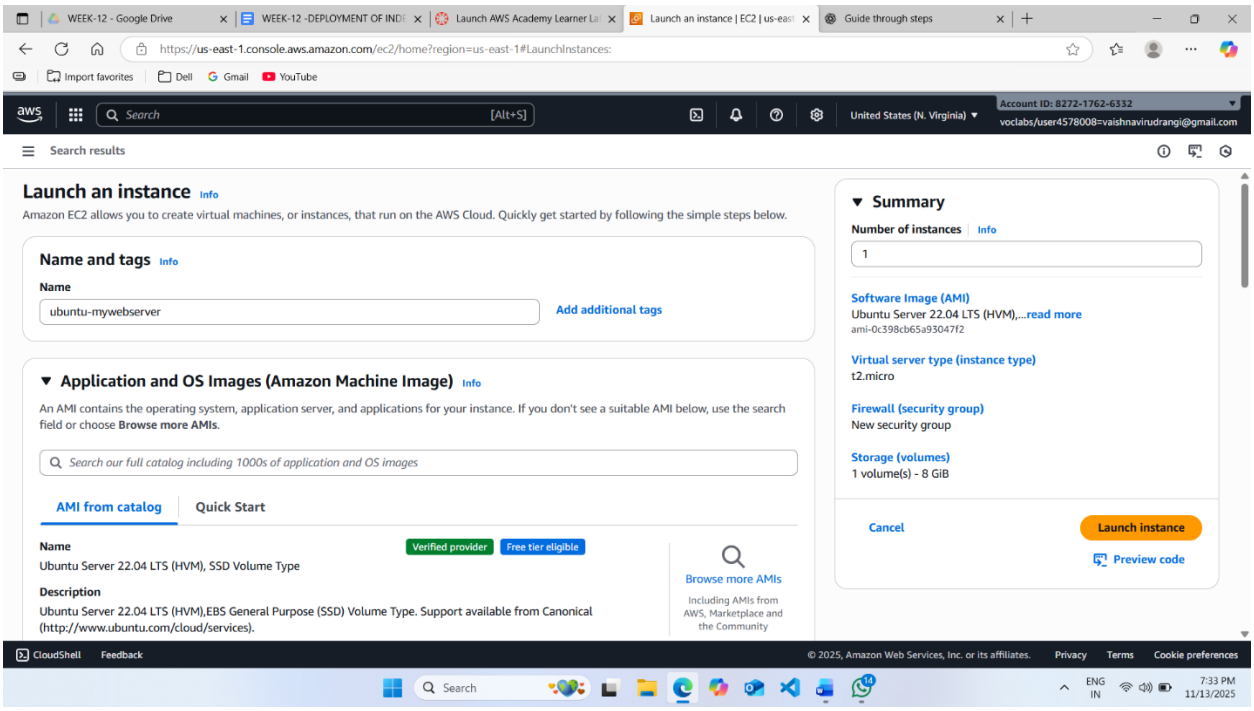
### I. Create Maven-web-java project in eclipse & push into github




## Click on EC2 to create instance

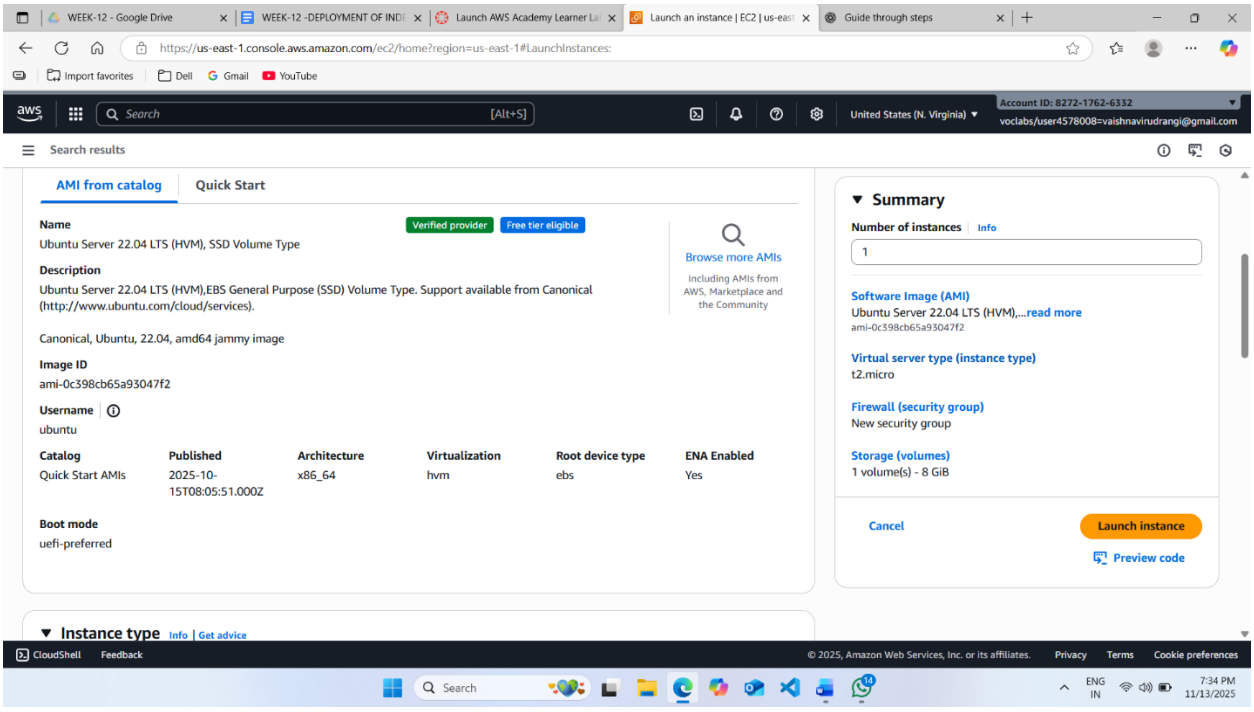



## Stage 1 --Name (Giving name to the machine) ubuntu

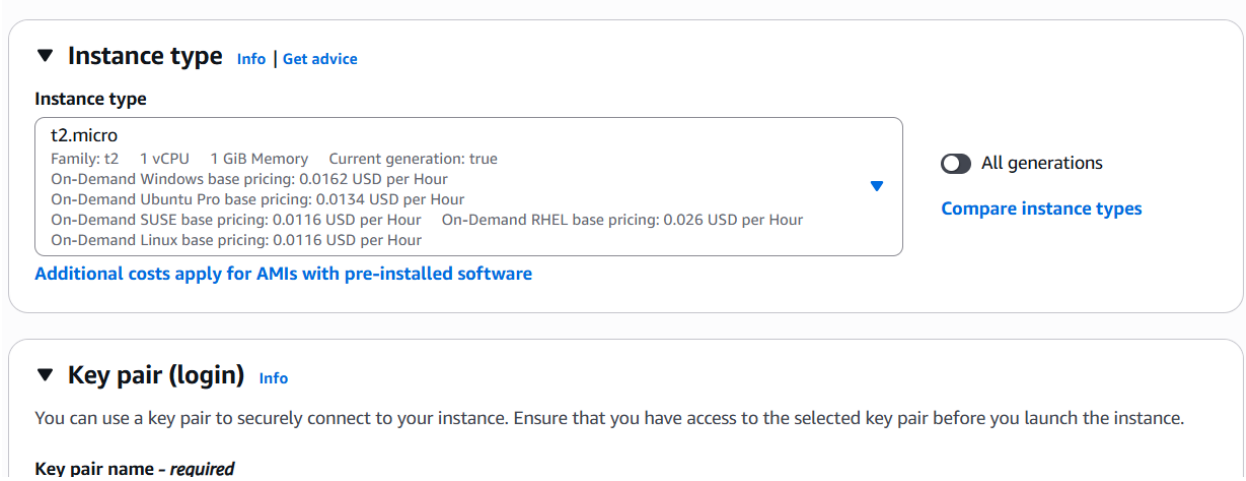



**Stage 2 -- Select AMI ( Note: Select free tier eligible ) ubuntu server**

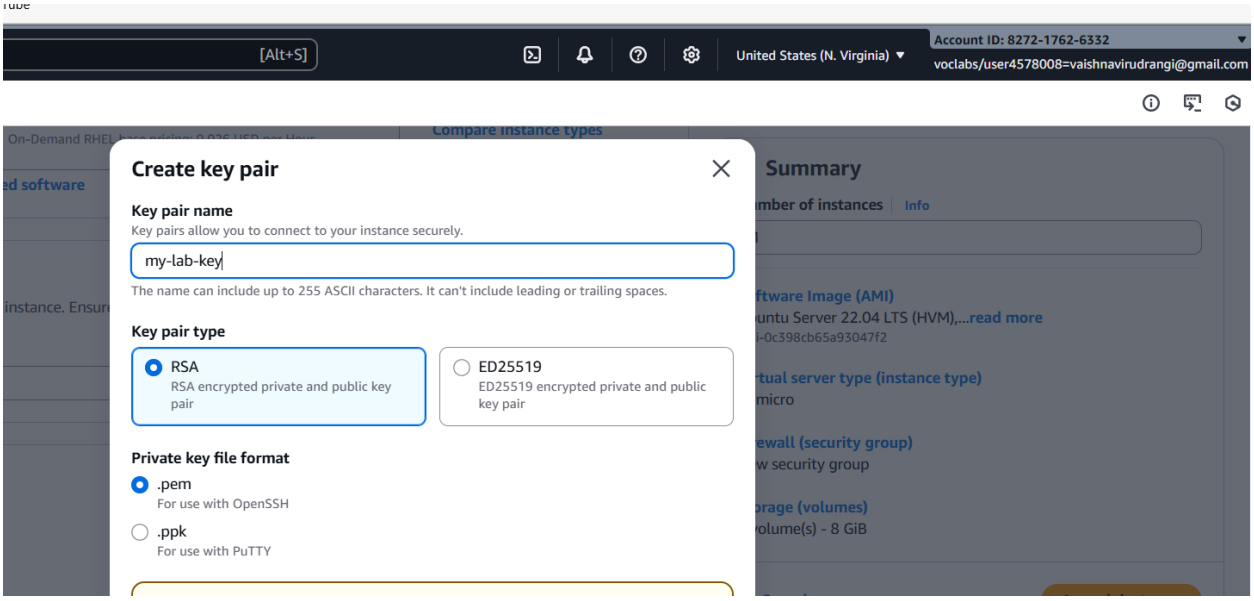
**Stage 3 -- Architecture as 64-bit**

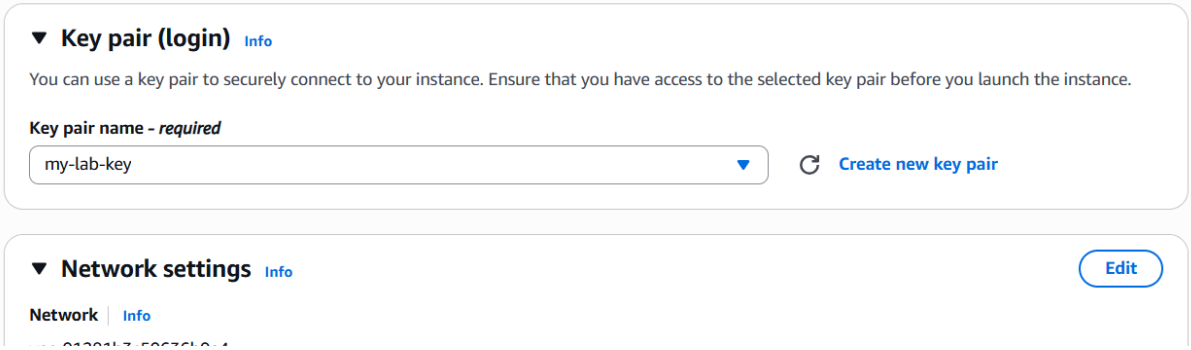



**Stage 4 -- Instance type ---- t2.micro(default 1 CPU,1 GB RAM)**

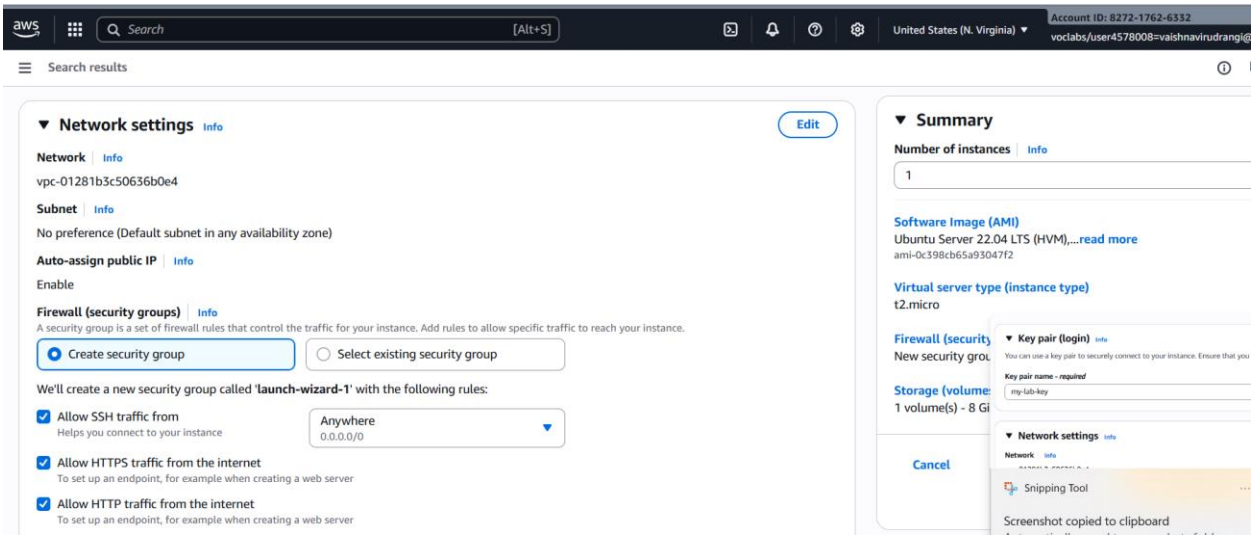



## Stage 5 -- Create a new keypair---a keypair will downloaded with extension .pem

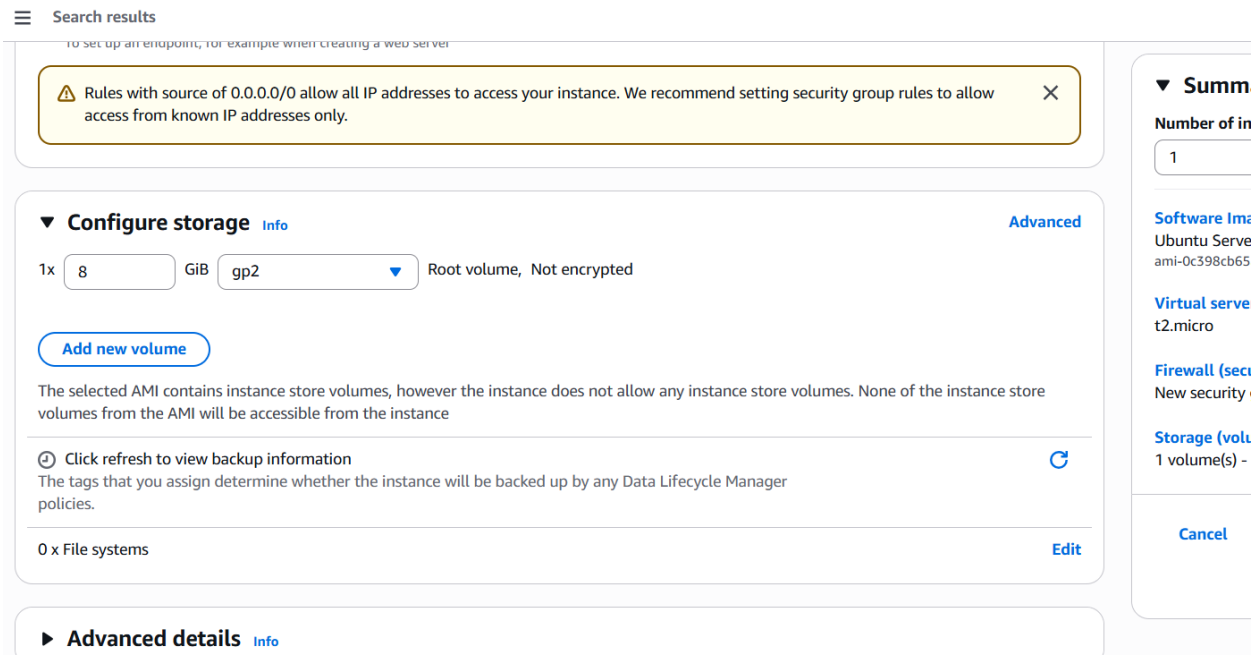


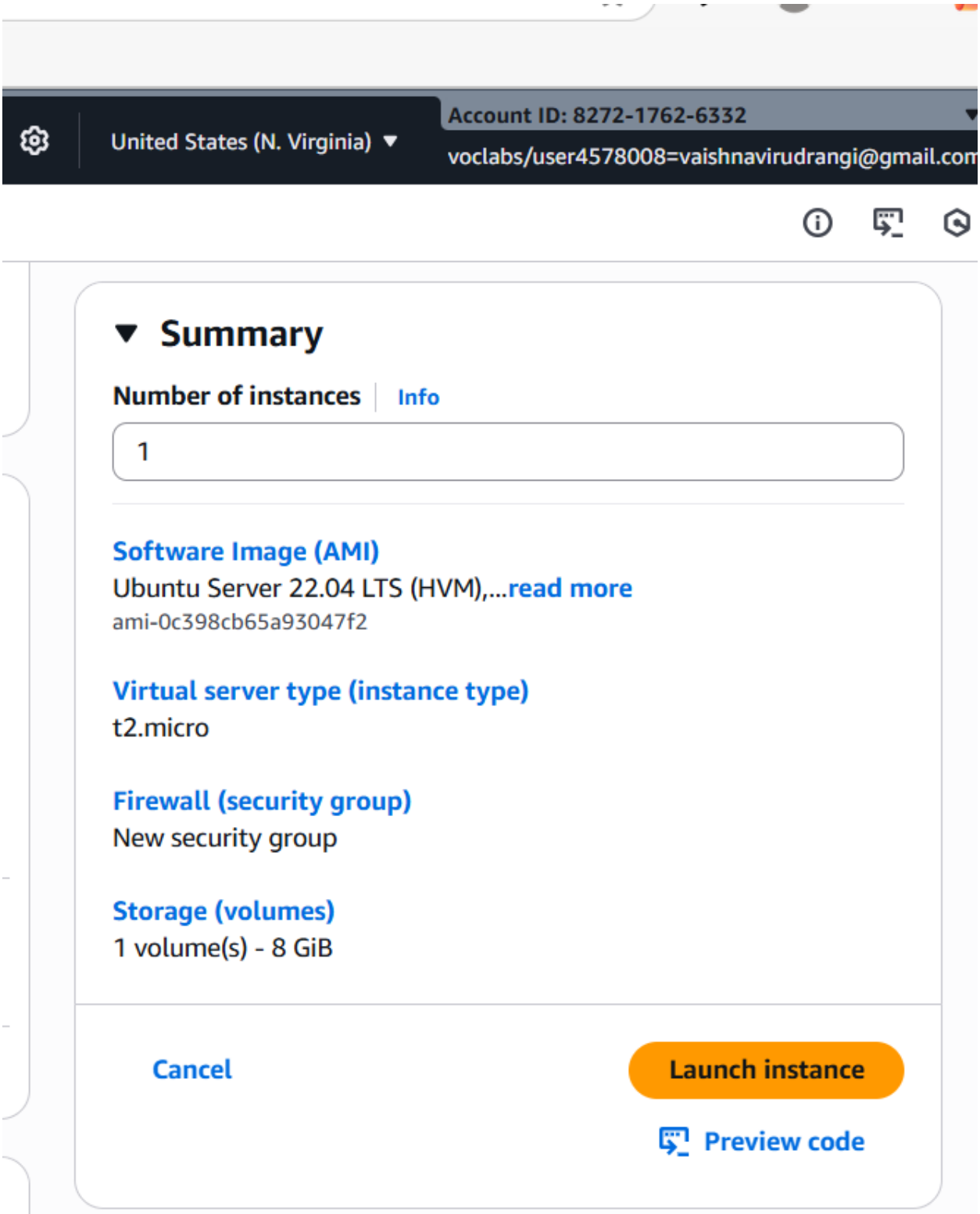
## Stage 6 -- Network Setting ----Create Security group -- ( It deals with ports )

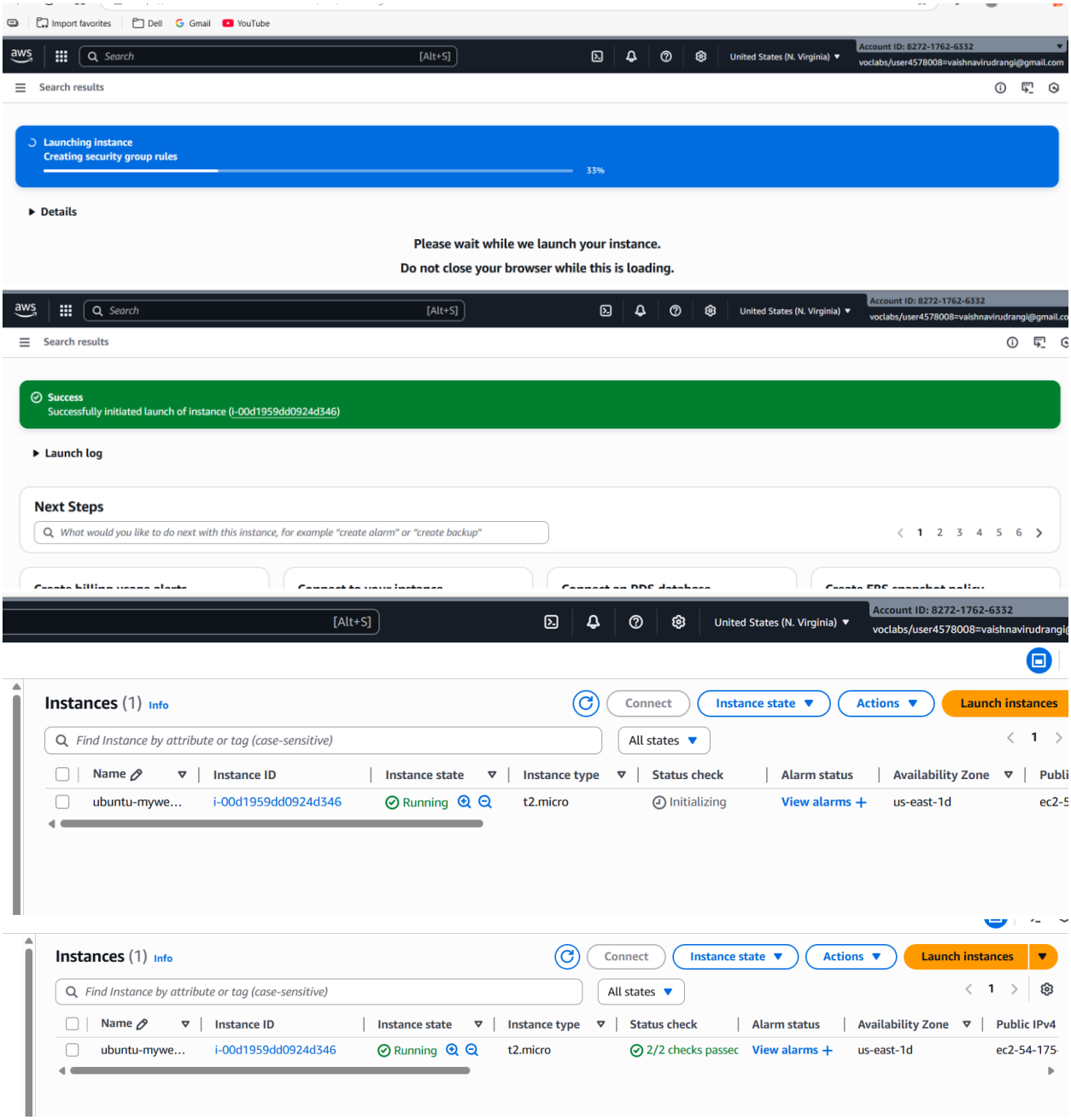



## Stage 7 -- Storage - 8GB ( Observation - we have root - it is same as C Drive) it default 8 GB

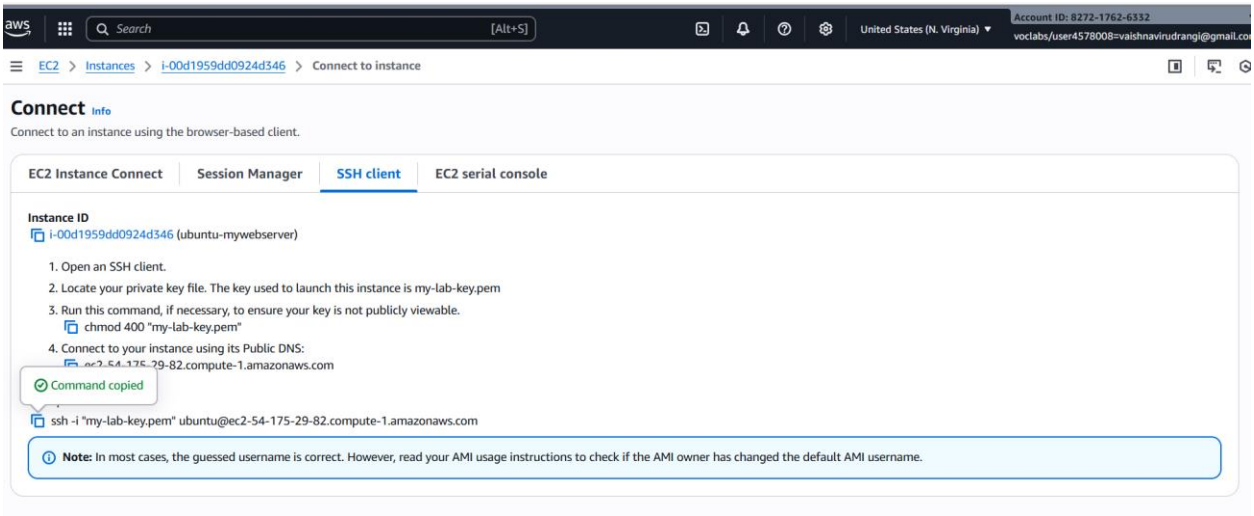



## Stage 8 --- click on launch instance

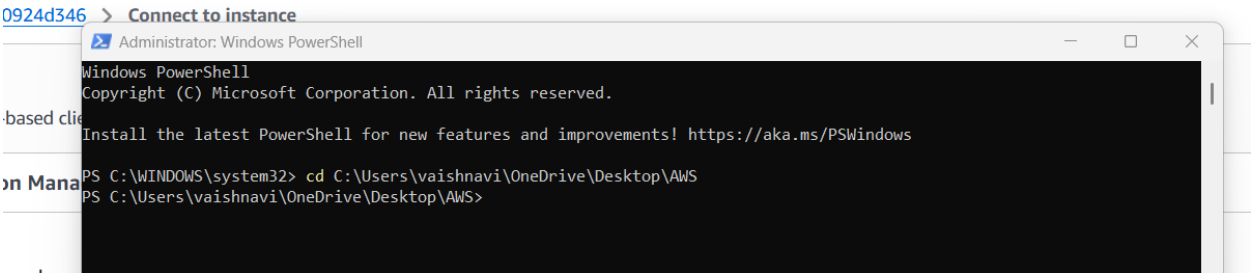

United States (N. Virginia)
Account ID: 8272-1762-6332
voclabs/user4578008=vaishnavirudrangi@gmail.com
Summary
Number of instances
Info
1
Software Image (AMI)
Ubuntu Server 22.04 LTS (HVM),...read more
ami-0c398cb65a93047f2
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code



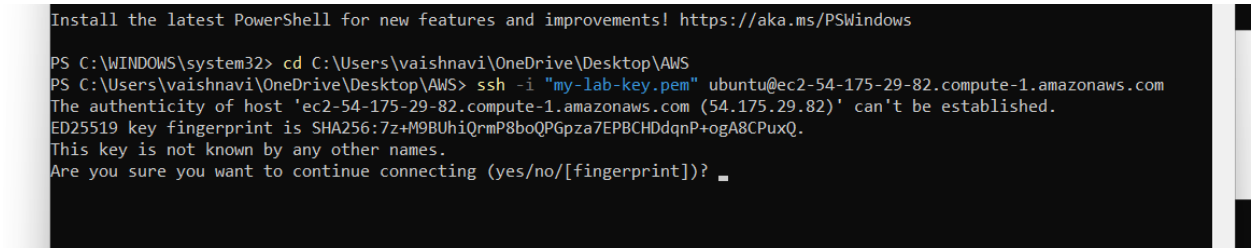

**Step a: Now you can connect local system to server (EC2 instance) using secure shell SSH.**

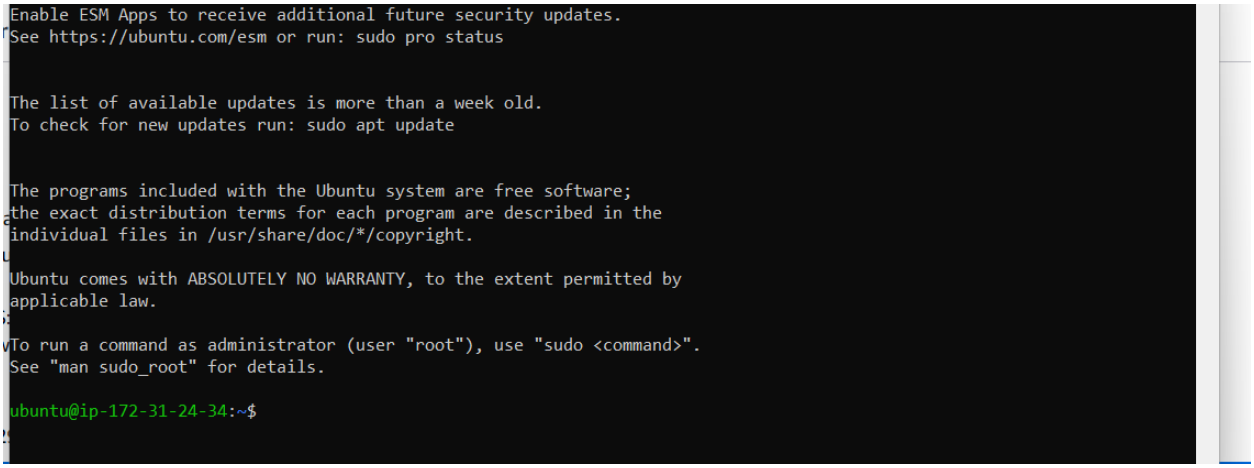



**Step b: Open Powershell in administrative mode and navigate to that path.**
**Type: cd < path>**

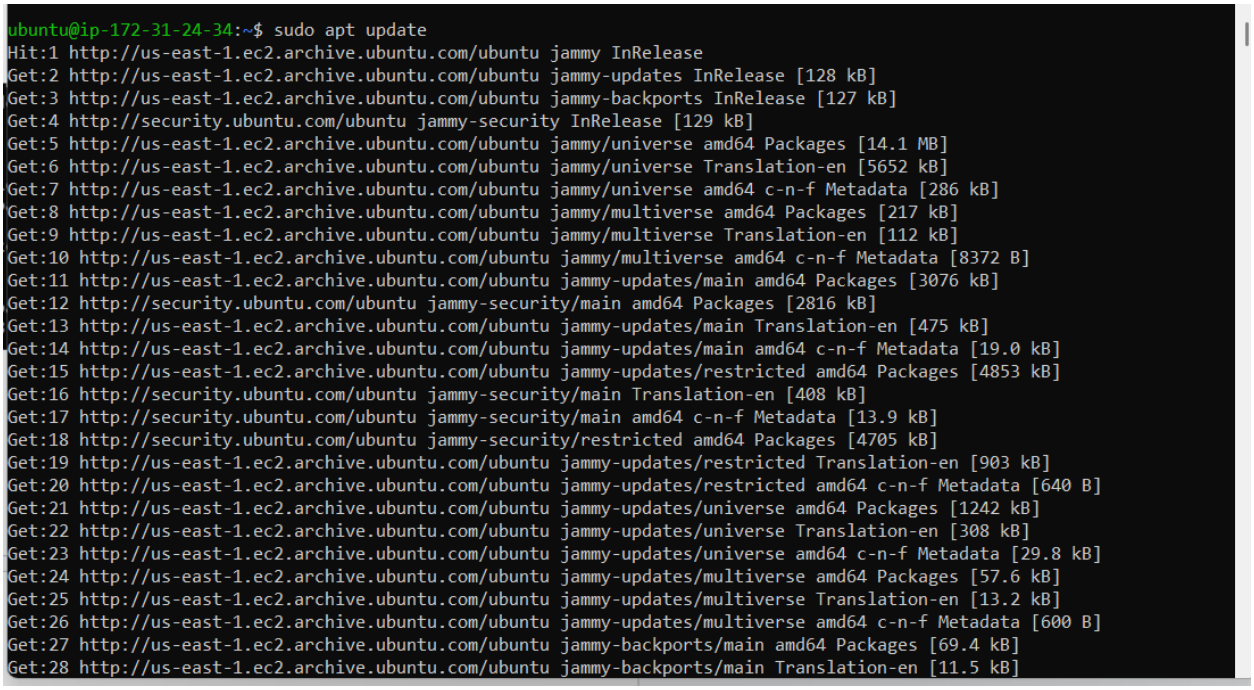



**Step c: Go to SSh and copy the command ssh which is present at the below**
**Step d: Run the pasted ssh -i command in the terminal**






## Run the following commands to install s/w



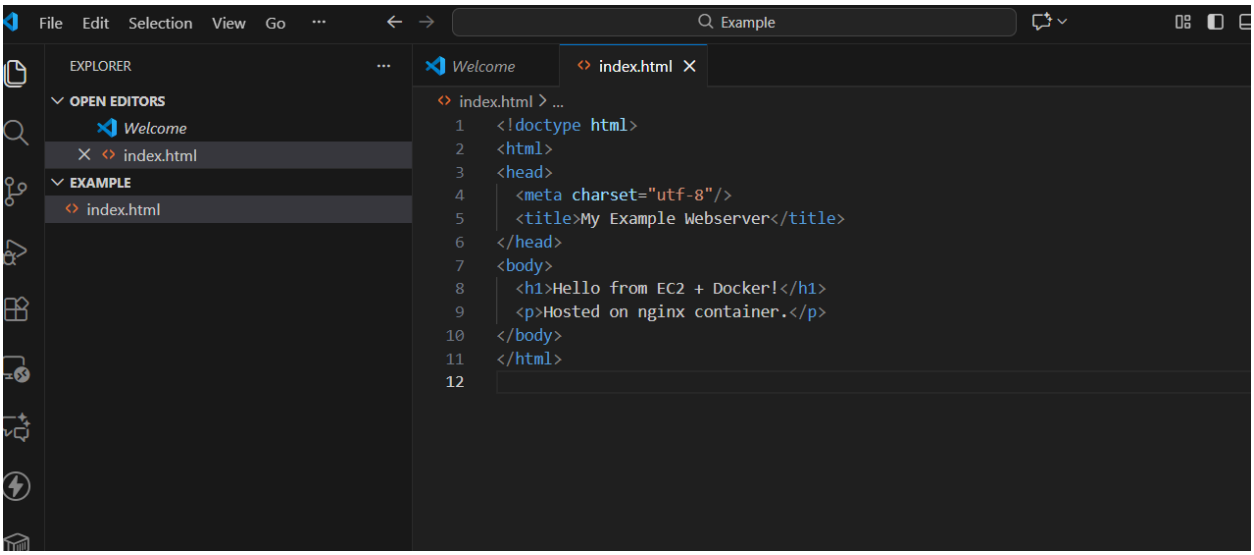
```
19 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-24-34:~$ sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd 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-doc rinse
  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 19 not upgraded.
Need to get 76.3 MB of archives.
After this operation, 289 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.3.3-0ubuntu1~22.04.2 [8856 kB
]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.28-0ubuntu1~22.04.1 [
38.5 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2024071801~ubuntu0.22.04
.1 [6132 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0.22.04.1 [
374 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 28.2.2-0ubuntu1~22.04.
1 [28.4 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
```

```
ubuntu@ip-172-31-24-34:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.15).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
ubuntu@ip-172-31-24-34:~$
```

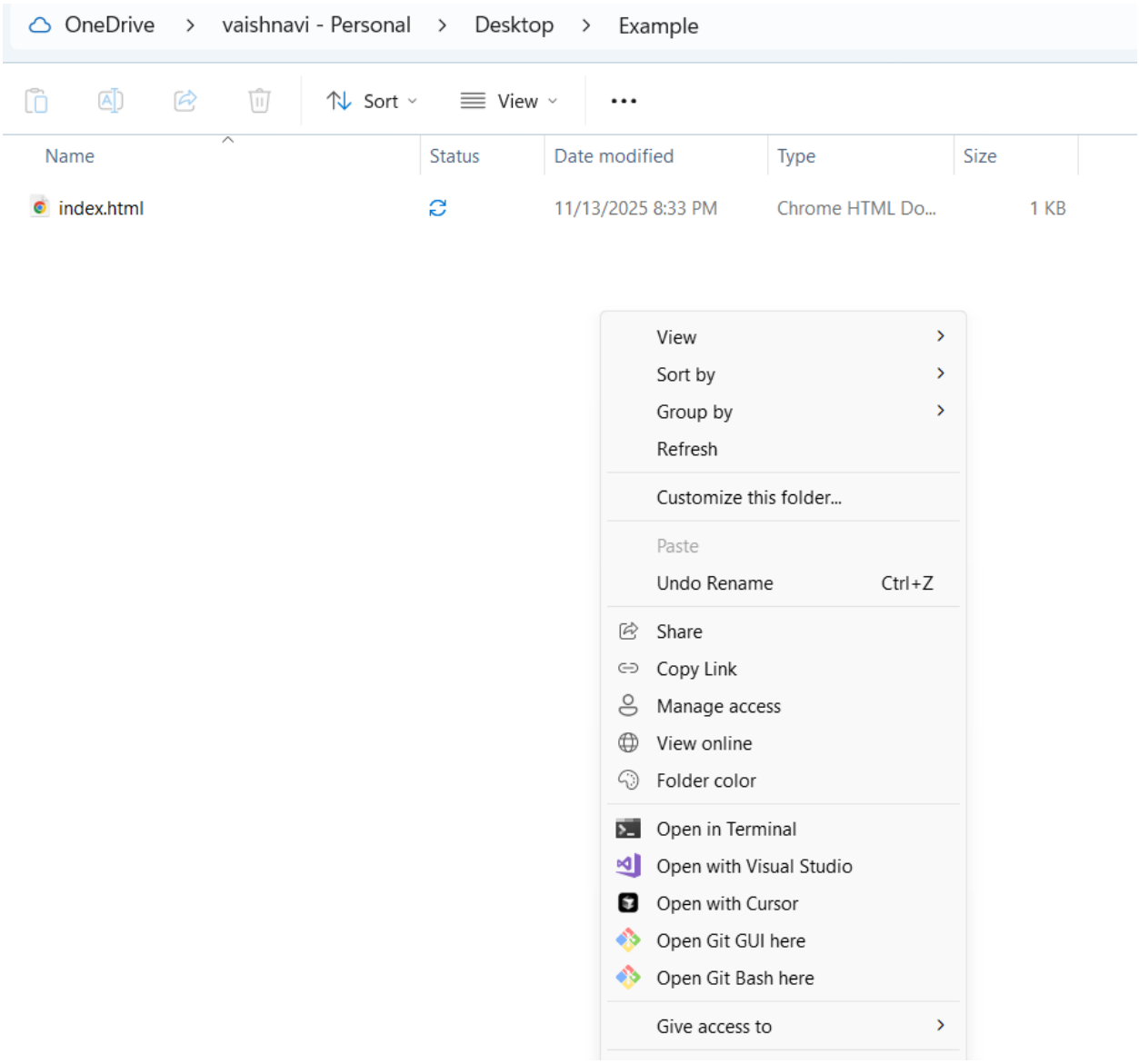
```
ubuntu@ip-172-31-24-34:~$ sudo apt install nano
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nano is already the newest version (6.2-1ubuntu0.1).
nano set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
ubuntu@ip-172-31-24-34:~$
```

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

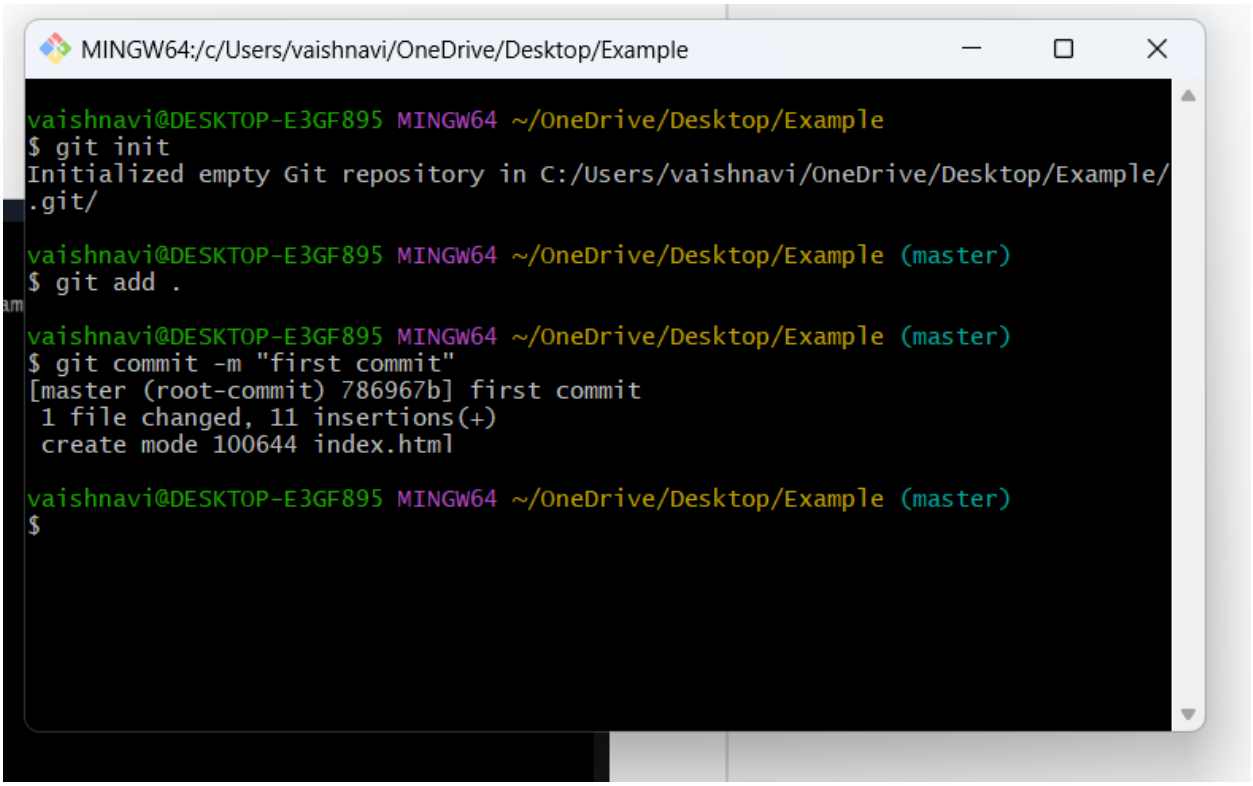
**Step a: Create basic index.html file in folder Example and save it**




**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”**

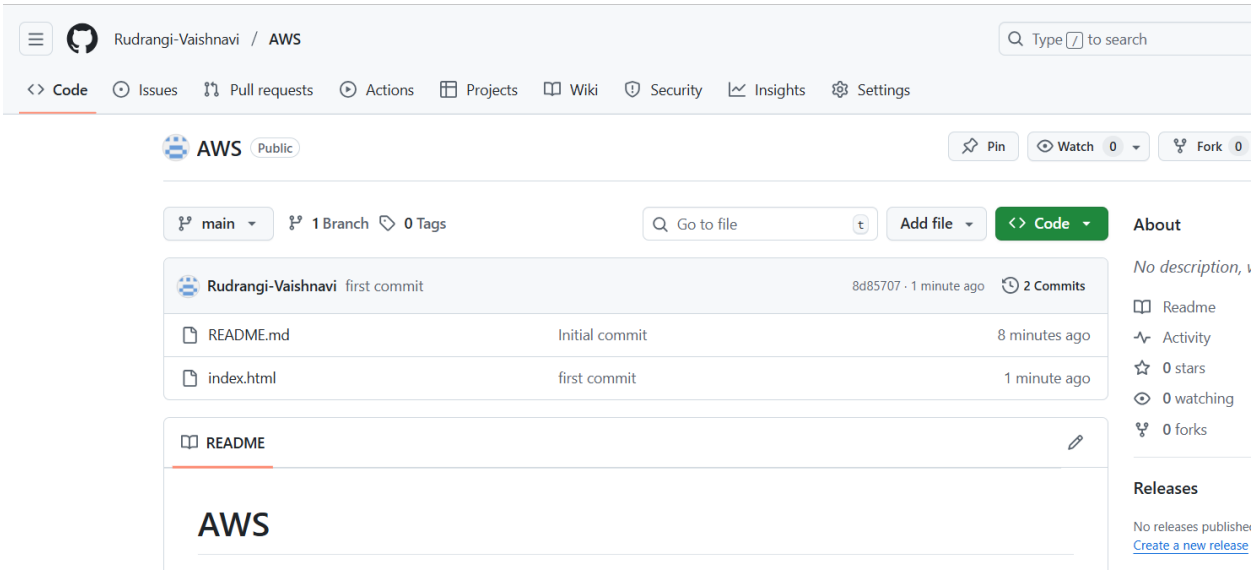



**Step d: Create git repository (here with name AWS)**
**Step e: Copy command one by one from above repository and run as below**
**git branch –M main**
**git remote add origin <https url>**
**git push –u origin main**

```
vaishnavi@DESKTOP-E3GF895 MINGW64 ~/OneDrive/Desktop/Example (master)
$ git branch -M main

vaishnavi@DESKTOP-E3GF895 MINGW64 ~/OneDrive/Desktop/Example (main)
$ git remote add origin https://github.com/Rudrangi-Vaishnavi/AWS.git
```

```
vaishnavi@DESKTOP-E3GF895 MINGW64 ~/OneDrive/Desktop/Example (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 439 bytes | 219.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Rudrangi-Vaishnavi/AWS.git
   c4e5a44..8d85707  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>**

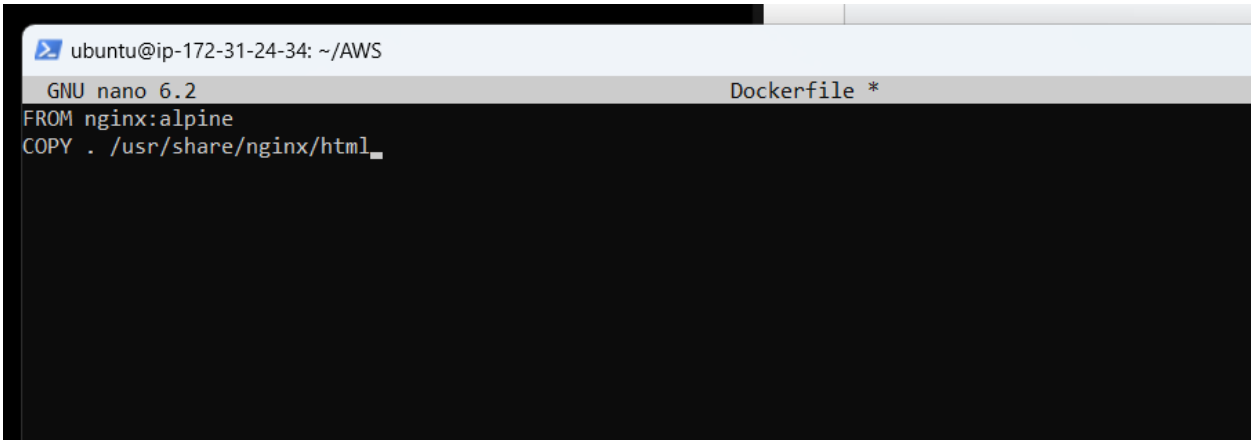
```
0 upgraded, 0 newly installed, 0 to remove and 19 not upgraded.
ubuntu@ip-172-31-24-34:~$ git clone https://github.com/Rudrangi-Vaishnavi/AWS.git
Cloning into 'AWS'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
ubuntu@ip-172-31-24-34:~$
```

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

```
remote: Total 6 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.
ubuntu@ip-172-31-24-34:~$ ls
AWS
ubuntu@ip-172-31-24-34:~$ cd AWS
ubuntu@ip-172-31-24-34:~/AWS$ ls
README.md  index.html
ubuntu@ip-172-31-24-34:~/AWS$
```

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

**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**

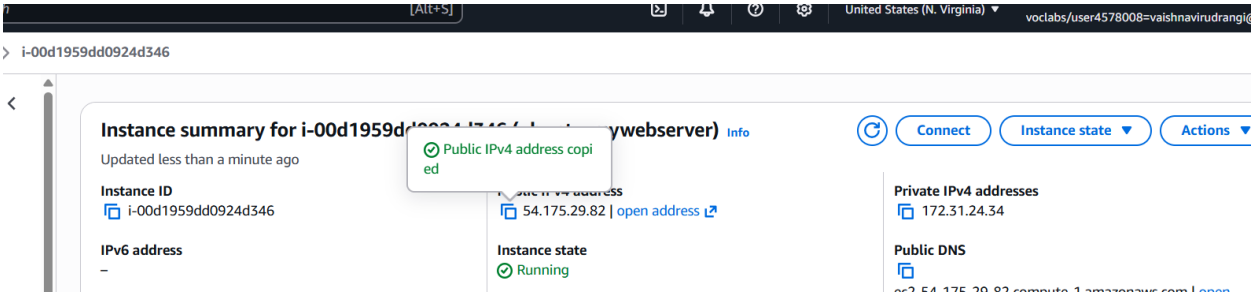
```
ubuntu@ip-172-31-24-34:~/AWS$ sudo docker build -t mywebapp .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  61.44kB
Step 1/2 : FROM nginx:alpine
alpine: Pulling from library/nginx
2d35ebdb57d9: Pulling fs layer
8f6a6833e95d: Pulling fs layer
194fa24e147d: Pulling fs layer
3eaba6cd10a3: Pulling fs layer
df413d6ebdc8: Pulling fs layer
d9a55dab5954: Pulling fs layer
ff8a36d5502a: Pulling fs layer
bdabb0d44271: Pulling fs layer
3eaba6cd10a3: Waiting
df413d6ebdc8: Waiting
d9a55dab5954: Waiting
ff8a36d5502a: Waiting
bdabb0d44271: Waiting
194fa24e147d: Verifying Checksum
194fa24e147d: Download complete
2d35ebdb57d9: Verifying Checksum
```

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

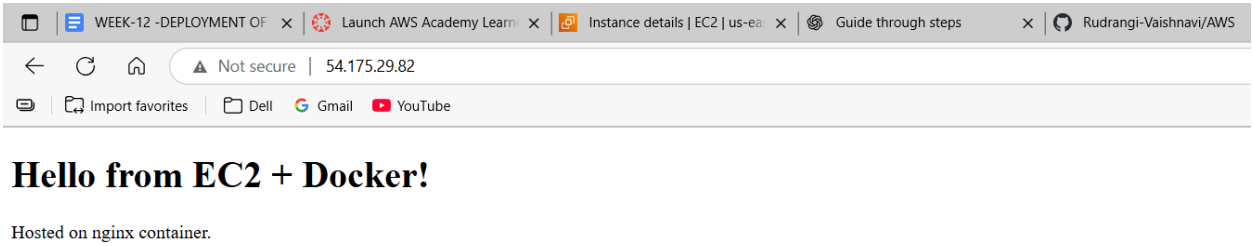
```
Status: Downloaded newer image for nginx:alpine
 ---> d4918ca78576
Step 2/2 : COPY . /usr/share/nginx/html
 ---> 41a8b396d4e1
Successfully built 41a8b396d4e1
Successfully tagged mywebapp:latest
ubuntu@ip-172-31-24-34:~/AWS$ sudo docker run -d -p 80:80 mywebapp
6d01321d73d4d21c80ba9b4697ec01fe63c863590444b203774c306a5a74eb3c
ubuntu@ip-172-31-24-34:~/AWS$
```

**Step n: Go to instances and click on instance here**

**Step o: Copy public ipv4 address**

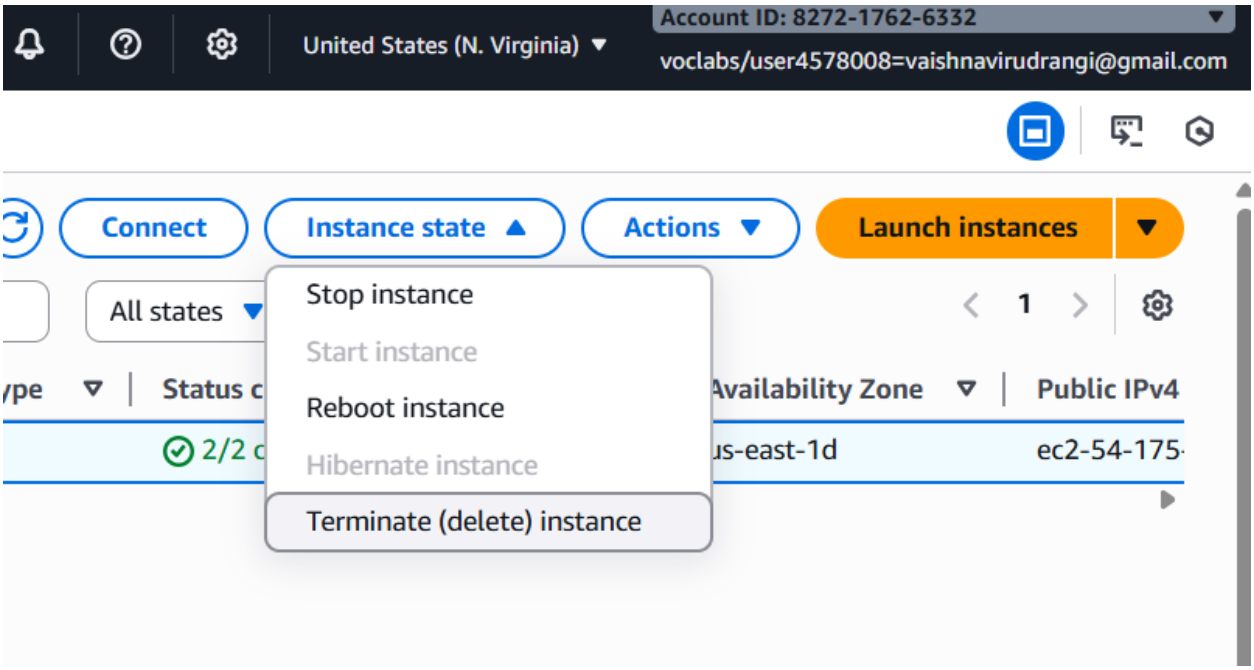


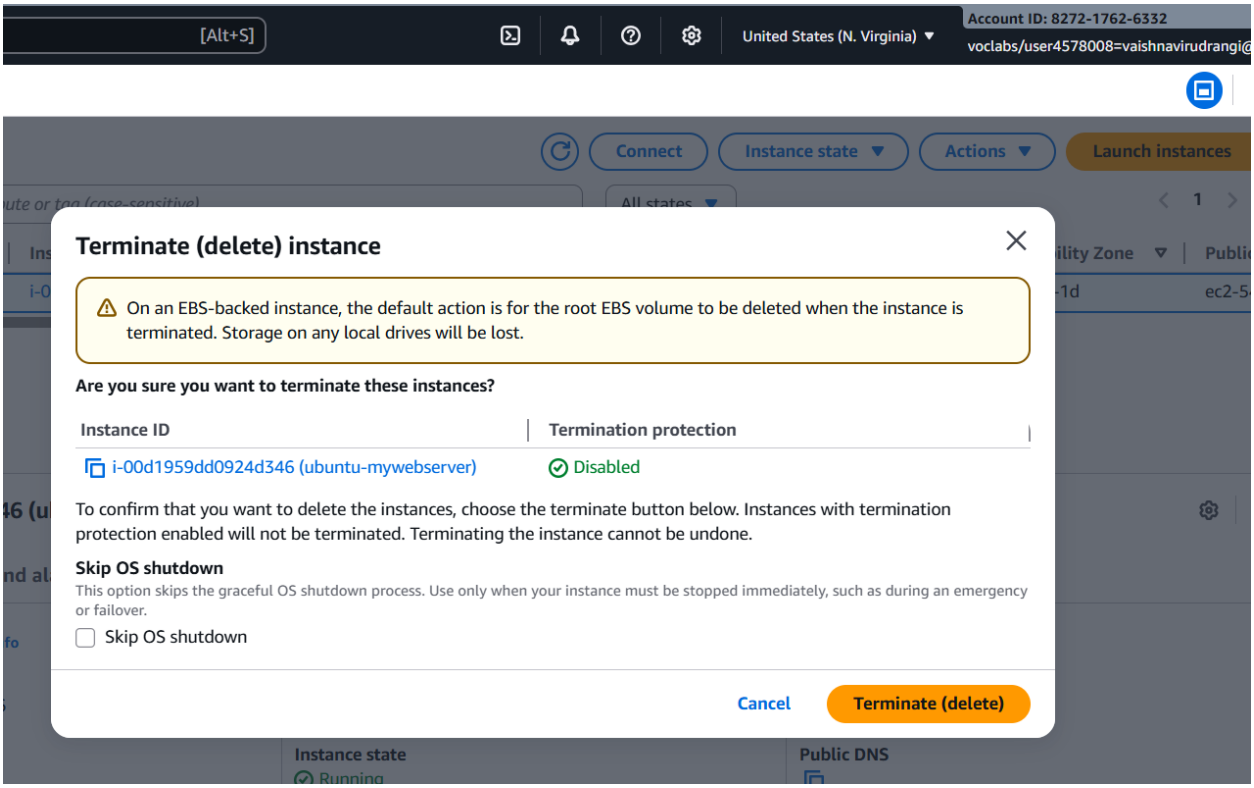

**Step p: Paste it in the browser to get below output**

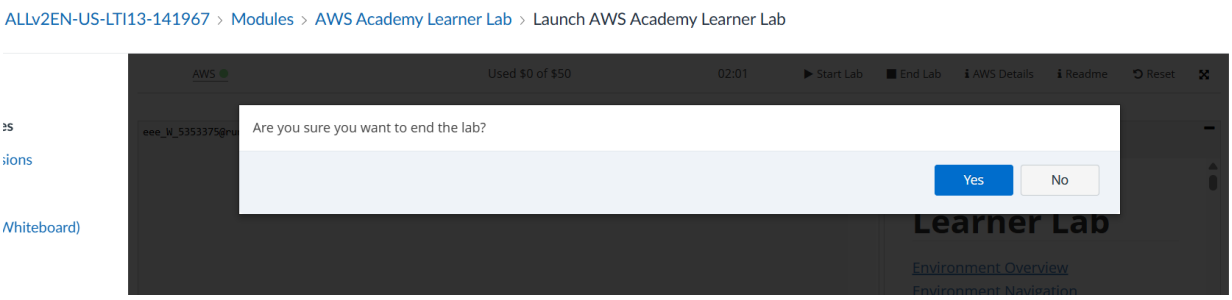



**Step Q: stop container**

```
 ---> 41a8b396d4e1
Successfully built 41a8b396d4e1
Successfully tagged mywebapp:latest
ubuntu@ip-172-31-24-34:~/AWS$ sudo docker run -d -p 80:80 mywebapp
6d01321d73d4d21c80ba9b4697ec01fe63c863590444b203774c306a5a74eb3c
ubuntu@ip-172-31-24-34:~/AWS$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS         PORTS
NAMES
6d01321d73d4   mywebapp   "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   0.0.0.0:80->80/tcp, [::]:80->80/tcp
peaceful_blackwell
ubuntu@ip-172-31-24-34:~/AWS$ sudo docker stop 6d01321d73d4
6d01321d73d4
ubuntu@ip-172-31-24-34:~/AWS$
```





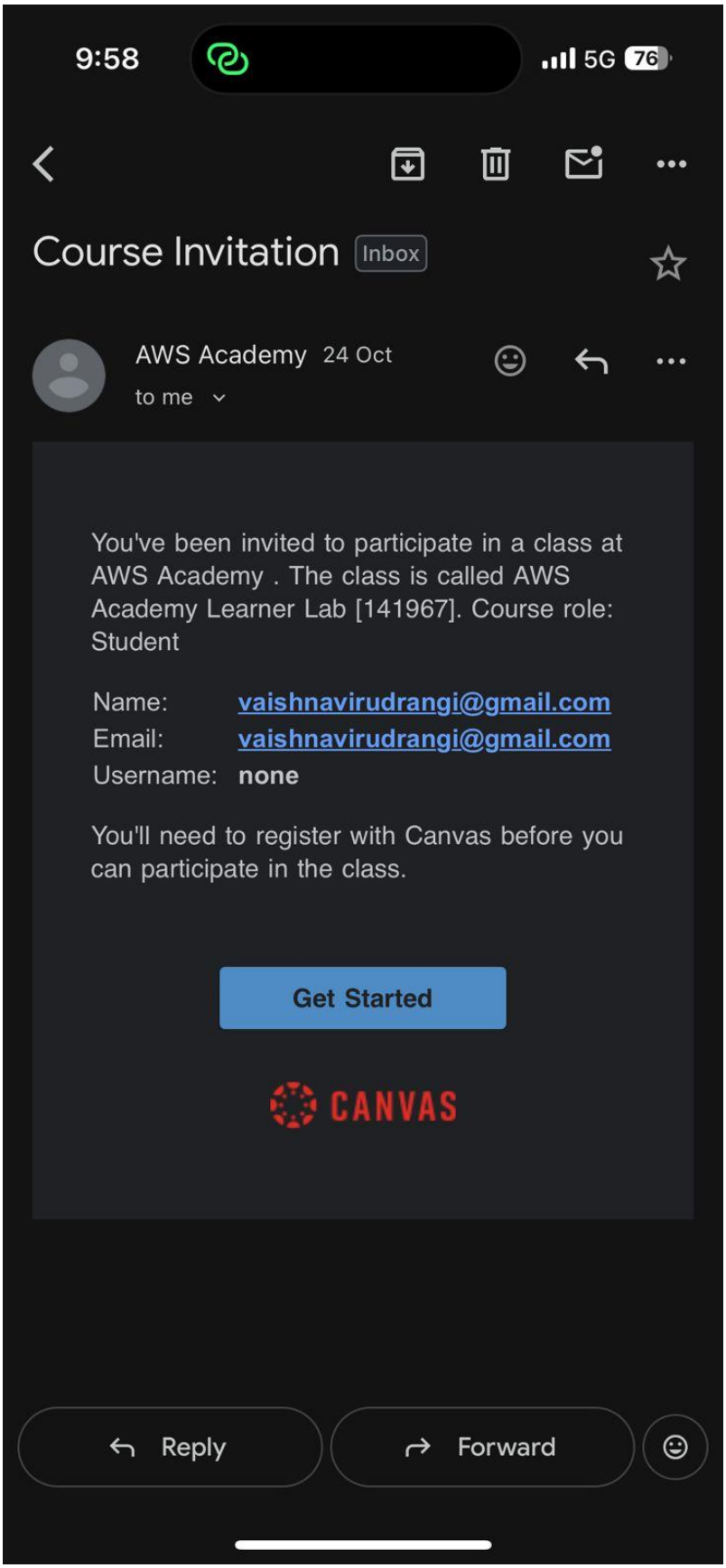



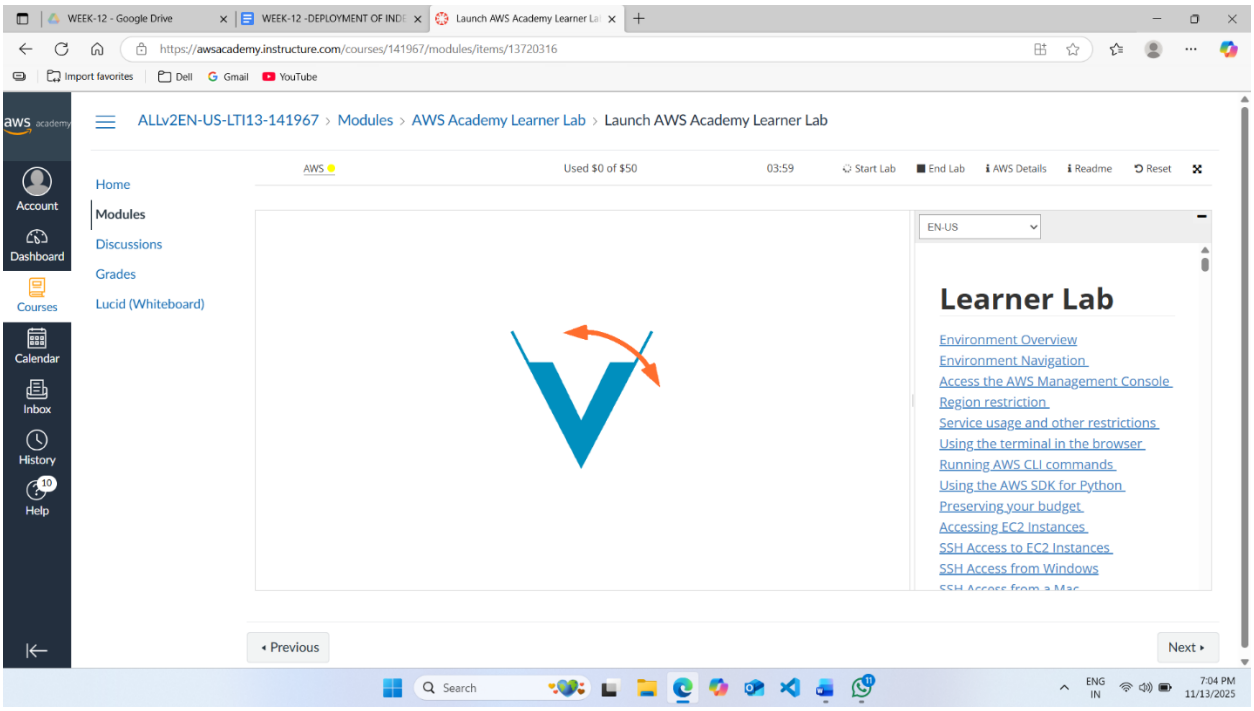

## II. Create the virtual machine (EC2--instance) in aws and connect to

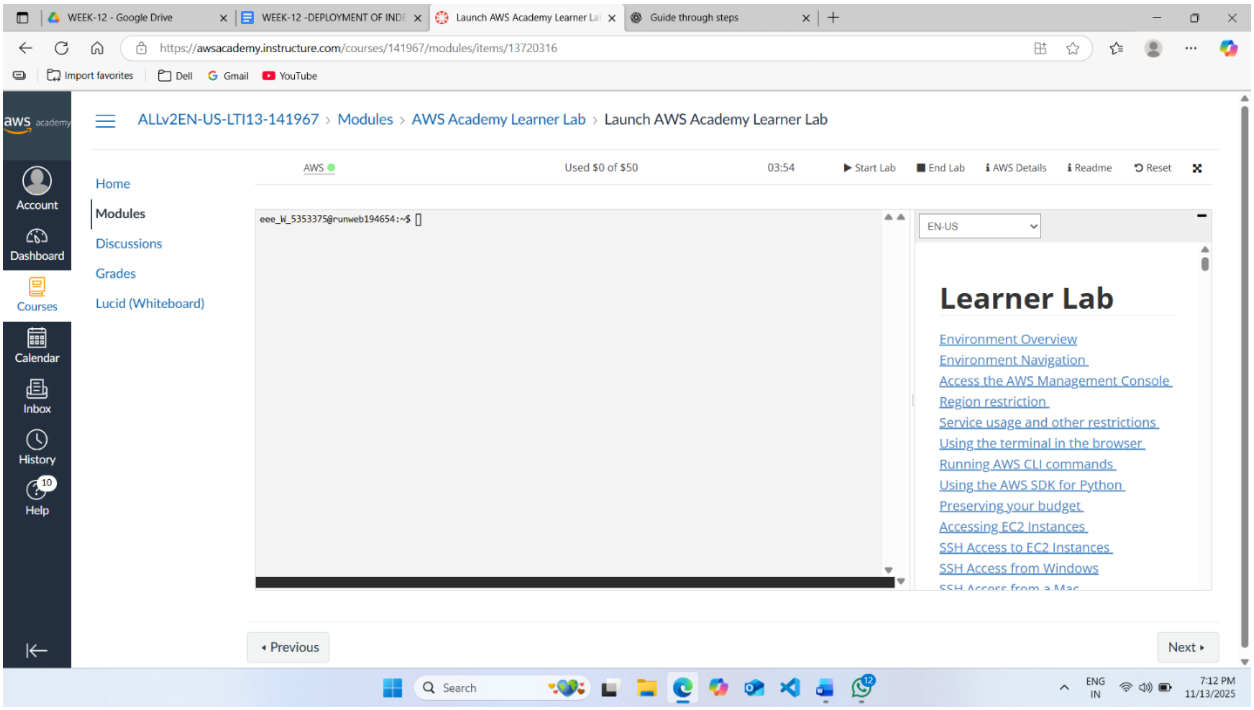
Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster.

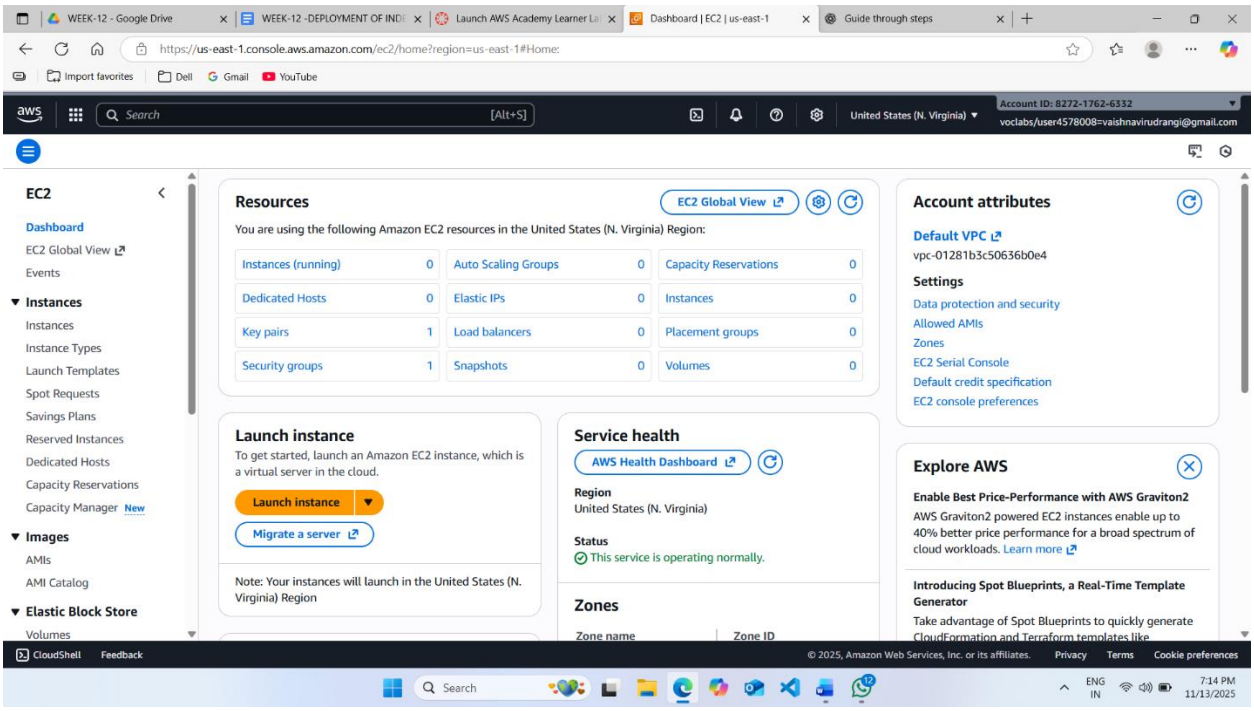
Ex : **Launch ubuntu instnace**

Step 1: Login to AWS /canvas account


9:58
5G
76
Course Invitation
Inbox
AWS Academy 24 Oct
to me
You've been invited to participate in a class at AWS Academy . The class is called AWS Academy Learner Lab [141967]. Course role: Student
Name: vaishnavirudrangi@gmail.com
Email: vaishnavirudrangi@gmail.com
Username: none
You'll need to register with Canvas before you can participate in the class.
Get Started
CANVAS
Reply
Forward


WEEK-12 - Google Drive
WEEK-12 -DEPLOYMENT OF INDI
Launch AWS Academy Learner La
https://awsacademy.instructure.com/courses/141967/modules/items/13720316
Import favorites
Dell
Gmail
YouTube
ALLv2EN-US-LTI13-141967 › Modules › AWS Academy Learner Lab › Launch AWS Academy Learner Lab
Account
Dashboard
Courses
Calendar
Inbox
History
Help
Home
Modules
Discussions
Grades
Lucid (Whiteboard)
AWS
Used $0 of $50
03:59
Start Lab
End Lab
AWS Details
Readme
Reset
EN-US
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
Search
ENG
IN
7:04 PM
11/13/2025

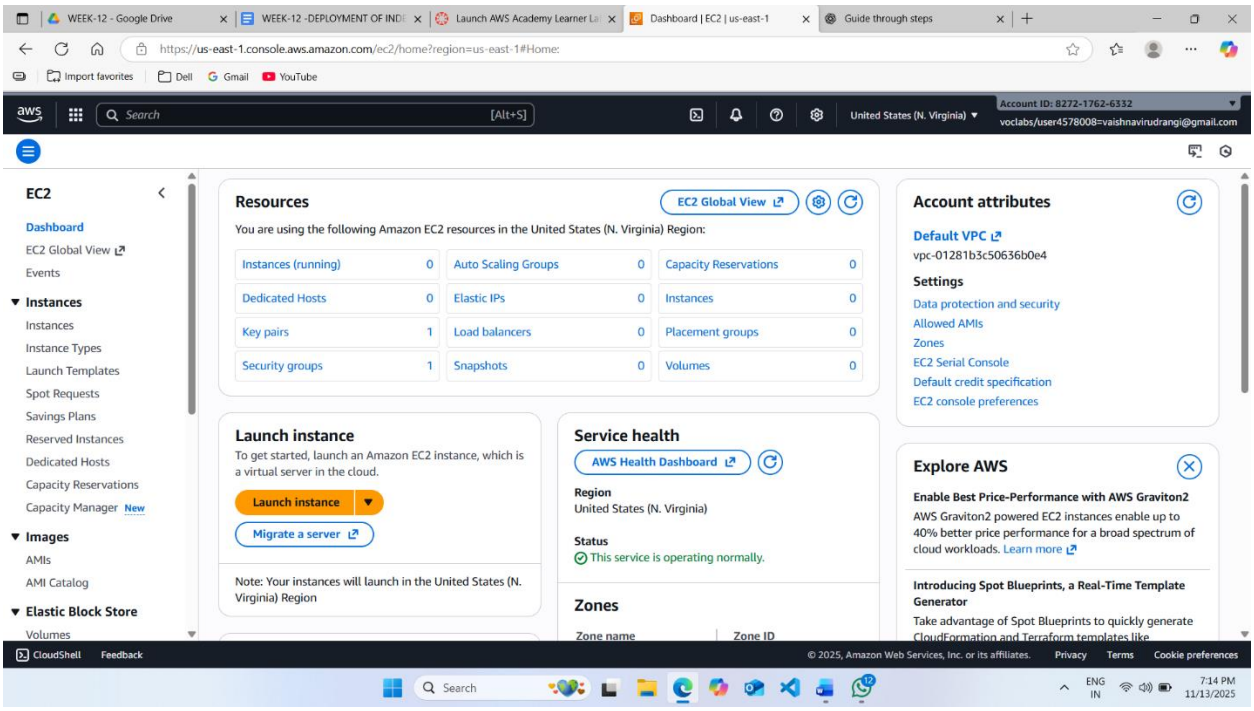

WEEK-12 - Google Drive
WEEK-12 -DEPLOYMENT OF INDI
Launch AWS Academy Learner La
Guide through steps
https://awsacademy.instructure.com/courses/141967/modules/items/13720316
Import favorites
Dell
Gmail
YouTube
ALLv2EN-US-LTI13-141967 › Modules › AWS Academy Learner Lab › Launch AWS Academy Learner Lab
Account
Dashboard
Courses
Calendar
Inbox
History
Help
Home
Modules
Discussions
Grades
Lucid (Whiteboard)
AWS
Used $0 of $50
03:54
Start Lab
End Lab
AWS Details
Readme
Reset
eee_lk_5353375@runweb194654:~$
EN-US
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
Search
ENG
IN
7:12 PM
11/13/2025



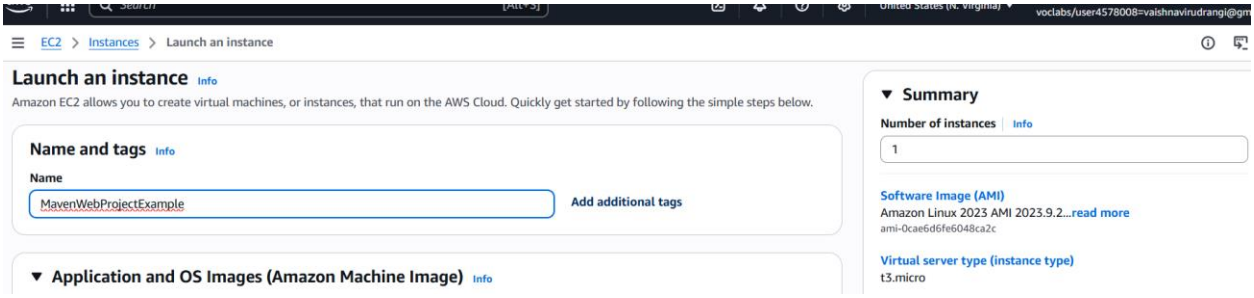

Step 2: Services -- EC2

Step 3: Choose region which is near ?

**Services -- EC2 --- Launch Instance**

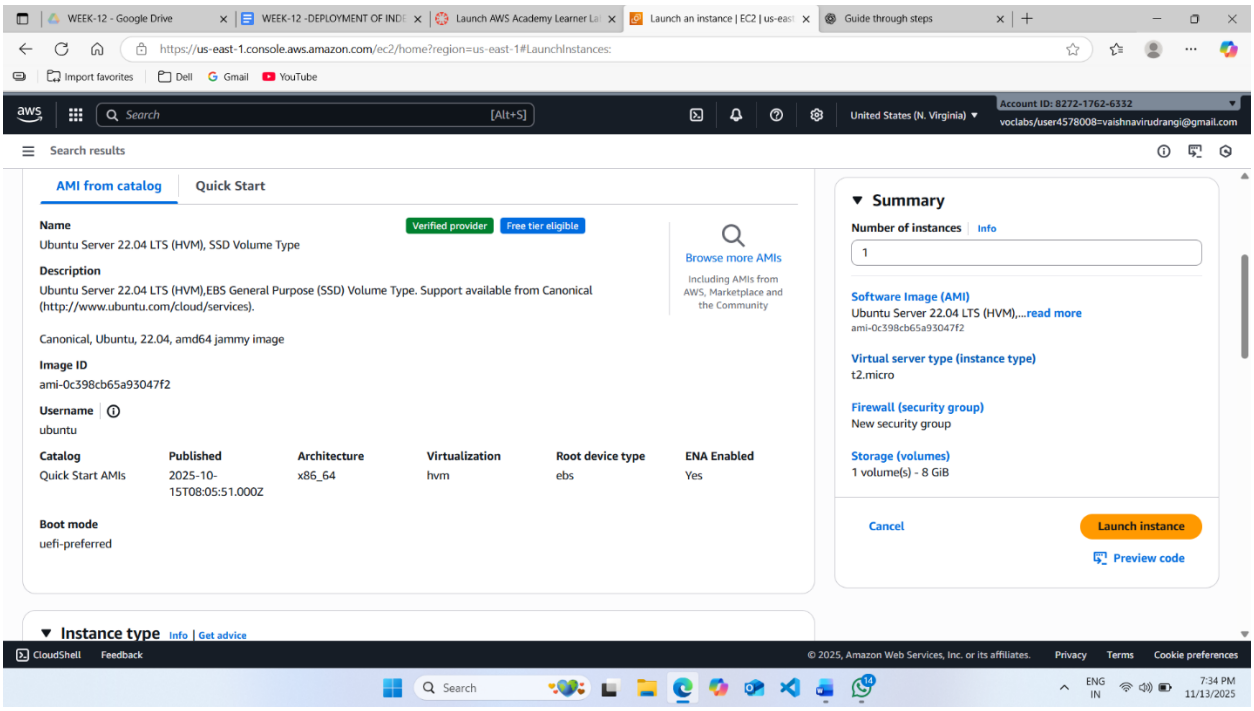



Stage 1 --Name (Giving name to the machine) ubuntu

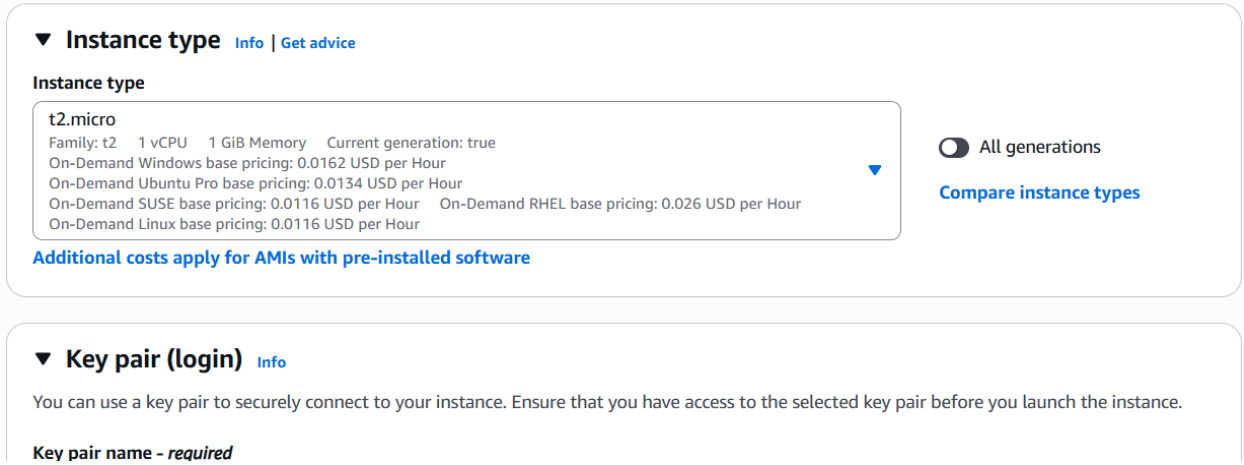



Stage 2 -- Select AMI ( Note: Select free tier eligible ) ubuntu server

Stage 3 -- Architecture as 64-bit

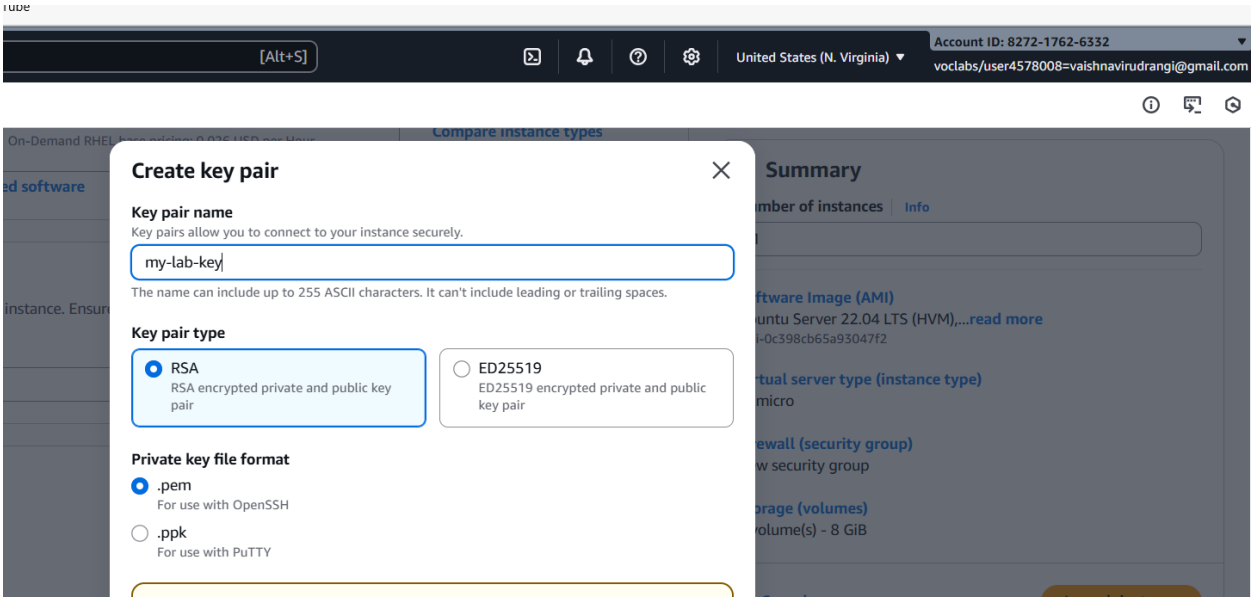


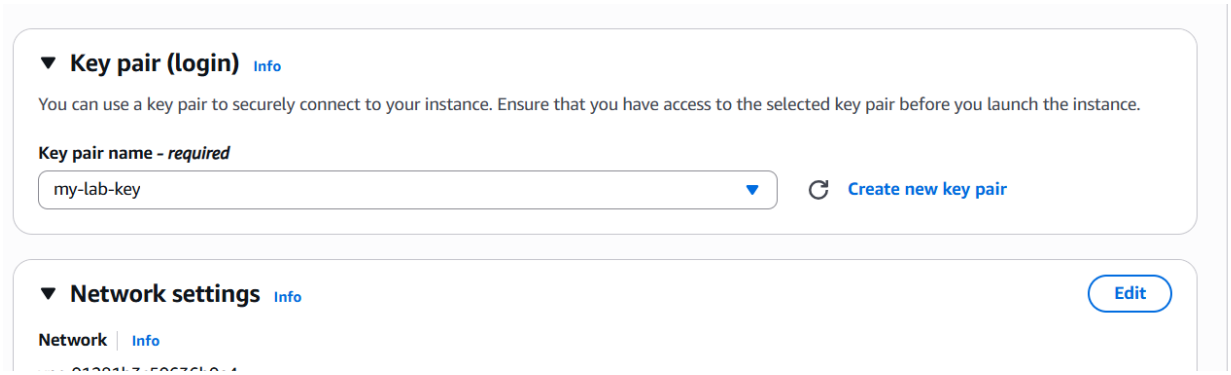

Stage 4 -- **Instance type** ---- t2.micro(default 1 CPU,1 GB RAM)




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

Store key in folder AWS

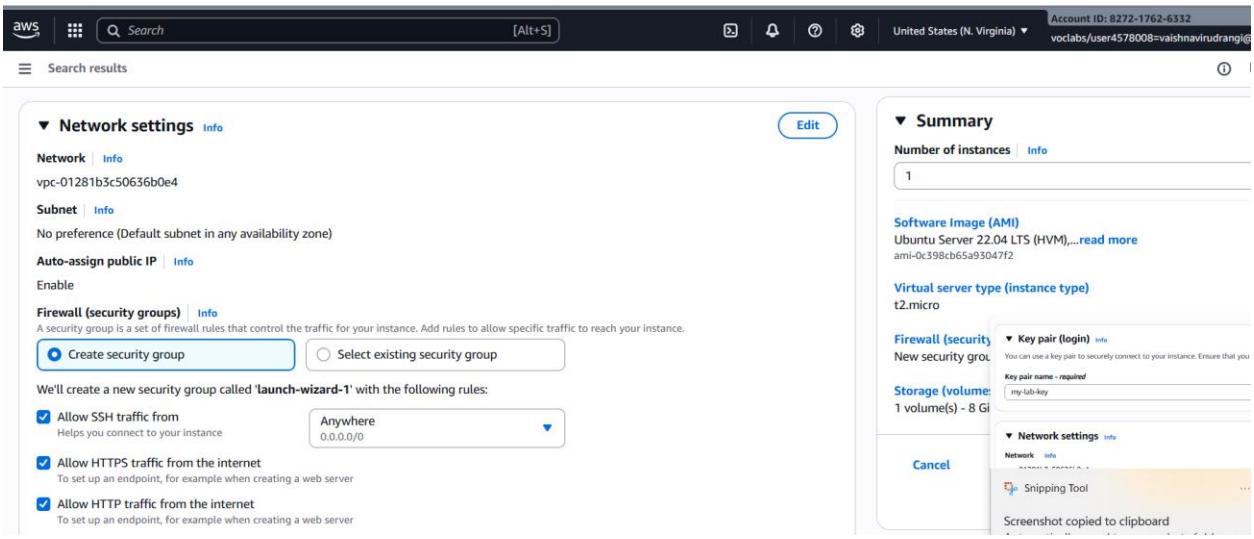


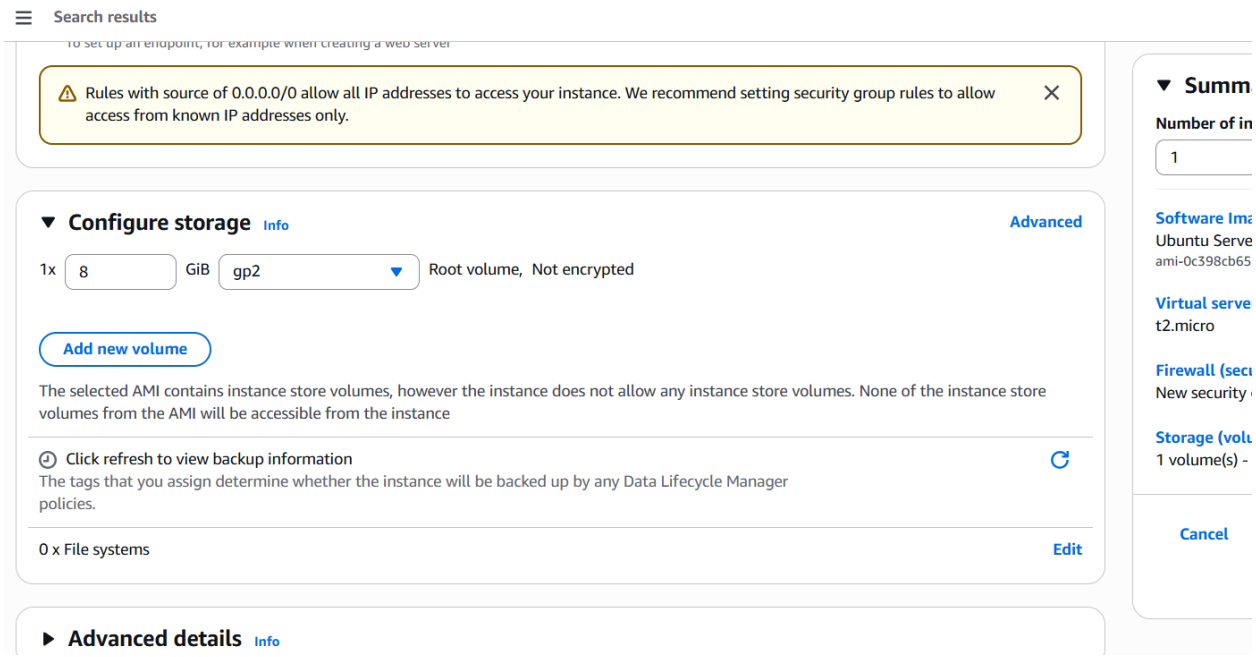

Stage 6 -- Network Setting ----Create Security group -- ( It deals with ports )

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

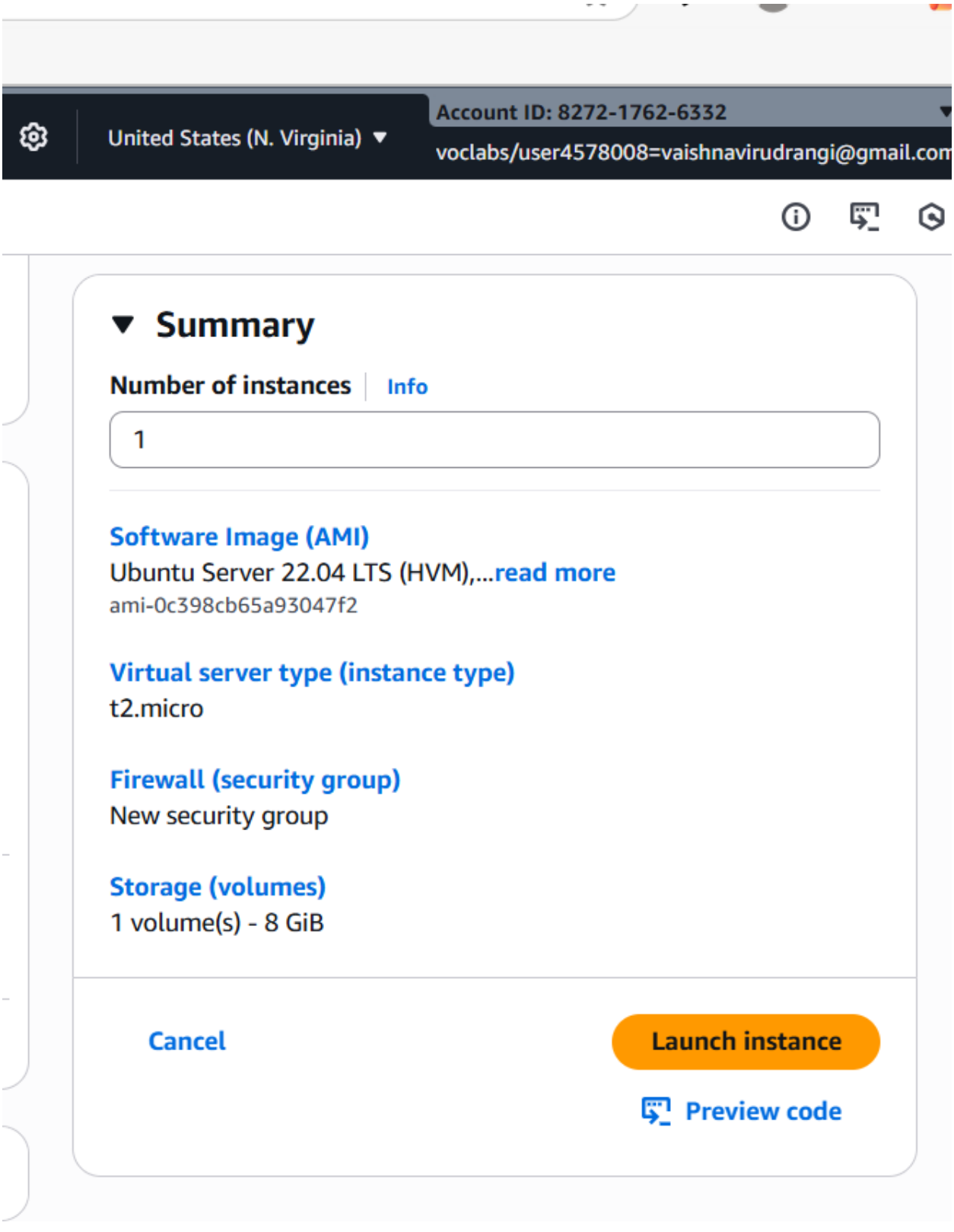
**Do this step : HERE select http and https**



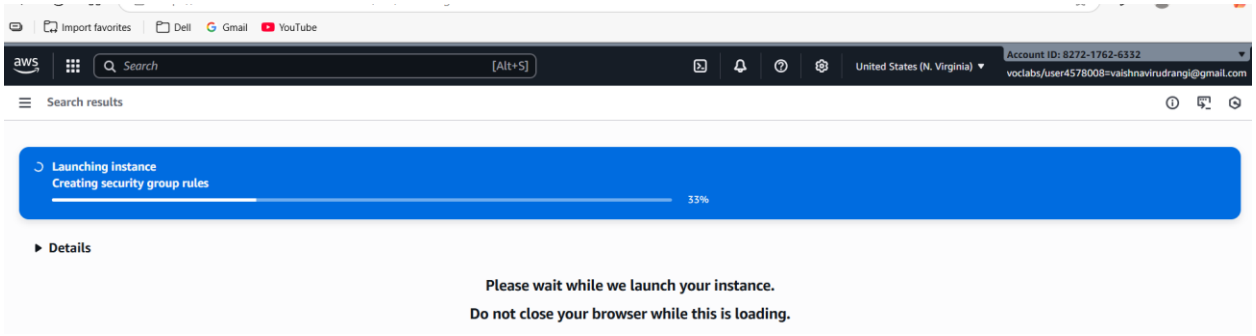

Stage 7 -- Storage - 8GB ( Observation - we have root - it is same as C Drive)

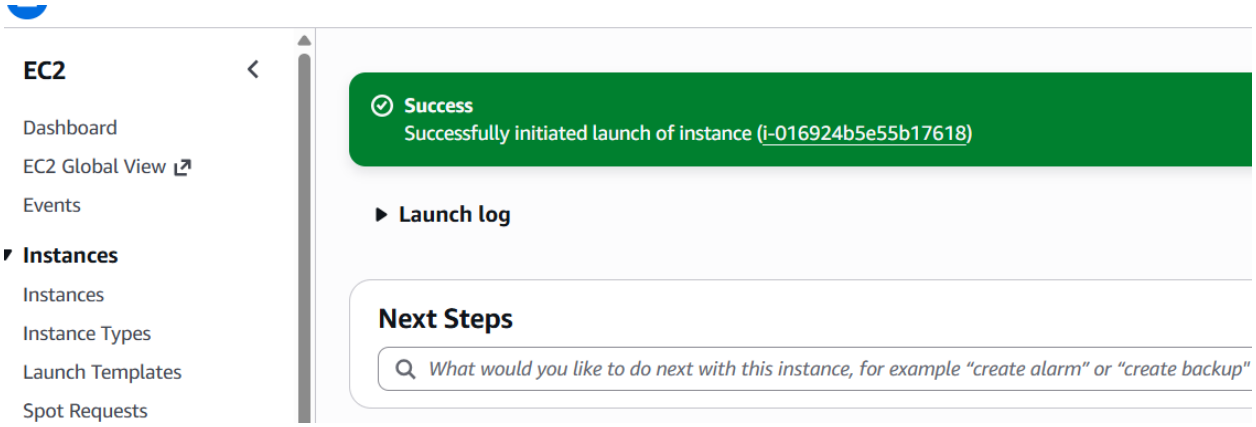


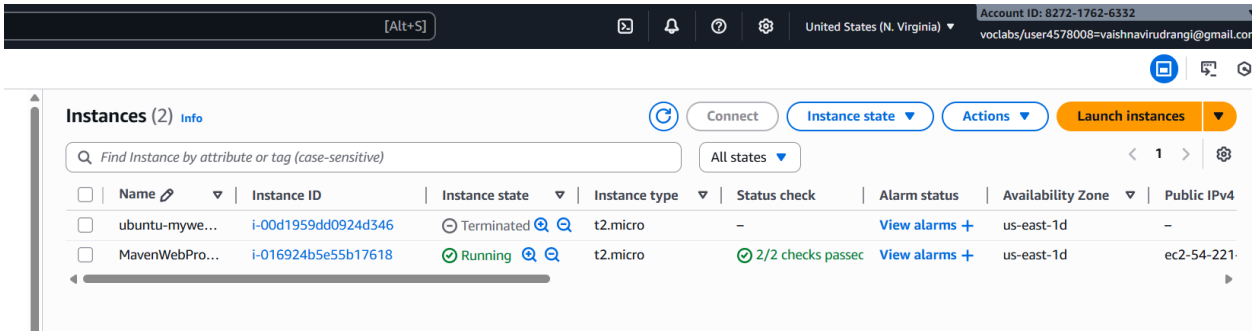

Stage 8 --- click on launch instance


United States (N. Virginia)
Account ID: 8272-1762-6332
voclabs/user4578008=vaishnavirudrangi@gmail.com
Summary
Number of instances
Info
1
Software Image (AMI)
Ubuntu Server 22.04 LTS (HVM),...read more
ami-0c398cb65a93047f2
Virtual server type (instance type)
t2.micro
Firewall (security group)
New security group
Storage (volumes)
1 volume(s) - 8 GiB
Cancel
Launch instance
Preview code

Stage 9: Number of instances ---1



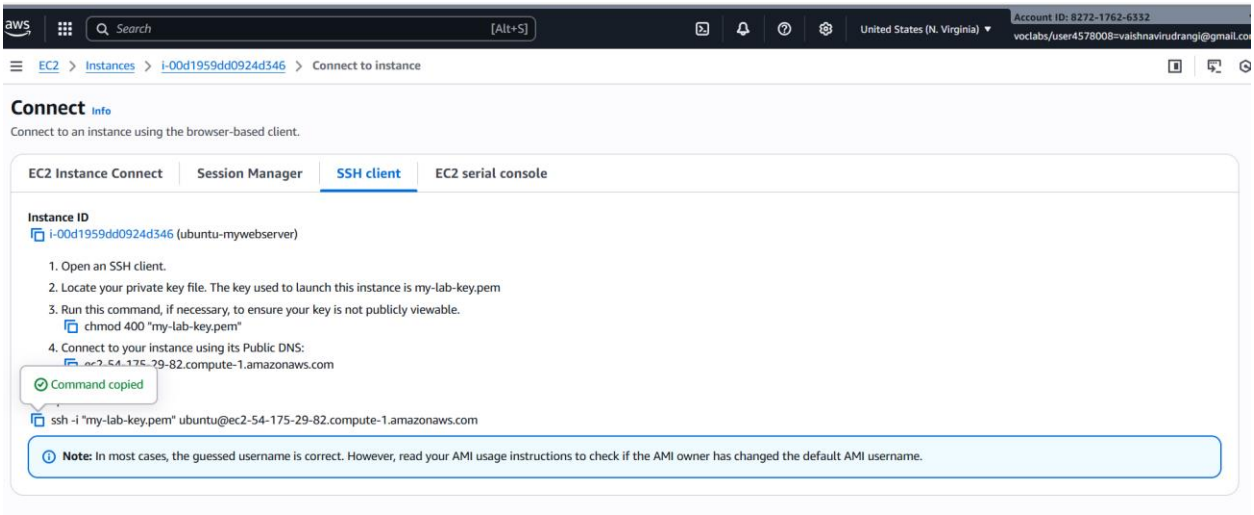




++++++++++++++++++++++++++++++++

Observation - One machines created

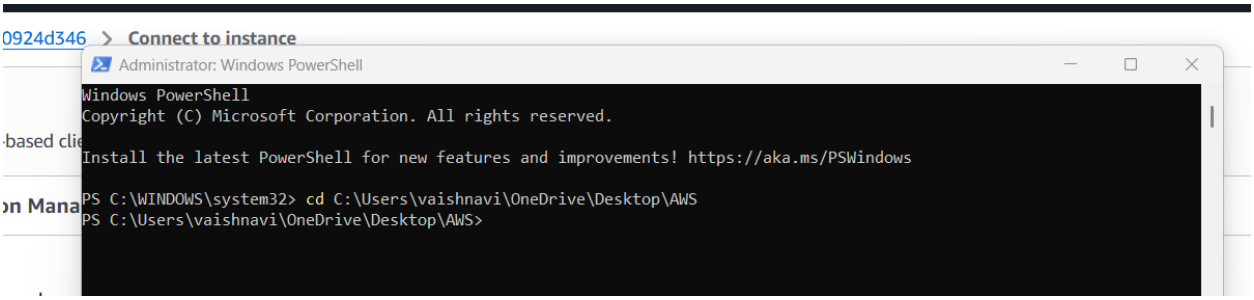
**Do this step:---once it is created select that instance and click on connect**

**Here copy the ssh – i command from SSH client connect tab**




We can use powershell /gitbash /webconsole , to connect to ubuntu machine.

**NOTE:- cd path of AWS folder // change path**




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

```
ubuntu@ip-172-31-22-149: ~
PS C:\Users\vaishnavi\OneDrive\Desktop\AWS> ssh -i "ExampleKey.pem" ubuntu@ec2-54-221-108-179.compute-1.amazonaws.com
The authenticity of host 'ec2-54-221-108-179.compute-1.amazonaws.com (54.221.108.179)' can't be established.
ED25519 key fingerprint is SHA256:WBRxDAgR5cG4Mwt9UzhX64DU4UGcNkX52g4sC6ph00s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-221-108-179.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

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

 System information as of Thu Nov 13 15:51:46 UTC 2025

  System load:  0.14              Processes:             108
  Usage of /:   25.9% of 6.71GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for enX0: 172.31.22.149
  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.
See https://ubuntu.com/esm or run: sudo pro status
```

---

## IV .Clone the application from github, Write the Dockerfile

once connected to instance

Step 1:- install the docker

**install docker ---apt-get update**

```
ubuntu@ip-172-31-22-149:~$ 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 [15.0 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe Translation-en [5982 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1309 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Components [3871 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 c-n-f Metadata [301 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [269 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse Translation-en [118 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 Components [35.0 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/multiverse amd64 c-n-f Metadata [8328 B]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1585 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [299 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.7 kB]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1498 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [303 kB]
Get:20 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [378 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2244 kB]
```

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

```
ubuntu@ip-172-31-22-149:~$ 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-doc rin
  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 14 not upgraded.
Need to get 76.0 MB of archives.
After this operation, 288 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-1ubuntu2 [33.9
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.3.3-0ubuntu1~24.04
]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.28-0ubuntu
38.4 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dns-root-data all 2024071801~ub
.1 [5918 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 dnsmasq-base amd64 2.90-2ubuntu
]
```

-
-
-

**apt-get install nano**

```
ubuntu@ip-172-31-22-149:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 14 not upgraded.
ubuntu@ip-172-31-22-149:~$ 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 14 not upgraded.
ubuntu@ip-172-31-22-149:~$
```

step 2:- git clone <**paste the github link of maven-web-java project**>

```
ubuntu@ip-172-31-22-149:~$ git clone https://github.com/Rudrangi-Vaishnavi/MavenWeb.git
Cloning into 'MavenWeb'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 26 (delta 1), reused 26 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (26/26), 4.60 KiB | 1.53 MiB/s, done.
Resolving deltas: 100% (1/1), done.
ubuntu@ip-172-31-22-149:~$
```

**step 3:- navigate to the maven-web-java project**

```
esolving deltas: 100% (1/1), done.
buntu@ip-172-31-22-149:~$ ls
avenWeb
buntu@ip-172-31-22-149:~$ cd MavenWeb
buntu@ip-172-31-22-149:~/MavenWeb$ ls
om.xml  src  target
buntu@ip-172-31-22-149:~/MavenWeb$
```

## VI. Create the image

Step 4:- nano Dockerfile

```
ubuntu@ip-172-31-22-149: ~/MavenWeb
 GNU nano 7.2                                  Dockerfile *
FROM tomcat:9-jdk11
COPY target/*.war /usr/local/tomcat/webapps
```

## V. Run the image and access it with public ip of virtual machine

Step 1:- build your image

docker build –t <imagename> .(dot)

```
ubuntu@ip-172-31-22-149:~/MavenWeb$ nano Dockerfile
ubuntu@ip-172-31-22-149:~/MavenWeb$ 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  97.79kB
Step 1/2 : FROM tomcat:9-jdk11
9-jdk11: Pulling from library/tomcat
4b3ffd8ccb52: Pulling fs layer
a4b8822e712a: Pulling fs layer
305f2a30fb03: Pulling fs layer
ac56e5323a09: Pulling fs layer
e46c95531a6b: Pulling fs layer
c3224bb24617: Pulling fs layer
4f4fb700ef54: Pulling fs layer
5c56d23d6b20: Pulling fs layer
ac56e5323a09: Waiting
e46c95531a6b: Waiting
c3224bb24617: Waiting
4f4fb700ef54: Waiting
5c56d23d6b20: Waiting
4b3ffd8ccb52: Verifying Checksum
4b3ffd8ccb52: Download complete
ac56e5323a09: Verifying Checksum
ac56e5323a09: Download complete
```

Step 2:- check for images

Step 3:- run image

docker run –d --name app-demo –p 6060:8080 <image name>

```
---> 179d7b7f6fee
Step 2/2 : COPY target/*.war /usr/local/tomcat/webapps
 ---> efdce9986d3a
Successfully built efdce9986d3a
Successfully tagged mavenwebproject:latest
ubuntu@ip-172-31-22-149:~/MavenWeb$ sudo docker run -d -p 9090:8080 mavenwebproject
9d68533d1b416ea9e9325d3669d8f929377162deb31857145def0a9a98d5dce7
ubuntu@ip-172-31-22-149:~/MavenWeb$
```

Step:-4 **Accessing the app by public ip of virtual machine**

**Note:-if your are not able to connect change the inbound rules..**