Soulpage IT Solutions Assignment

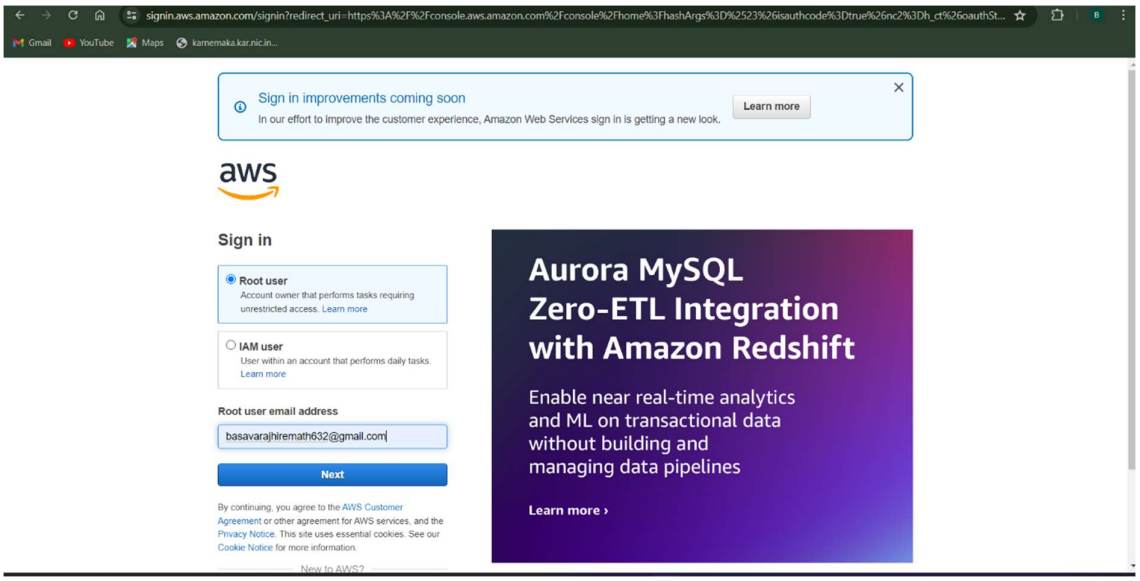
Name: Basawaraj Hiremath

Cloud computing is the on-demand delivery of computing services such as servers, storage, databases, networking, software, and analytics. Rather than keeping files on a proprietary hard drive or local storage device, cloud-based storage makes it possible to save remotely.

Next.js:Deployment and Configuration

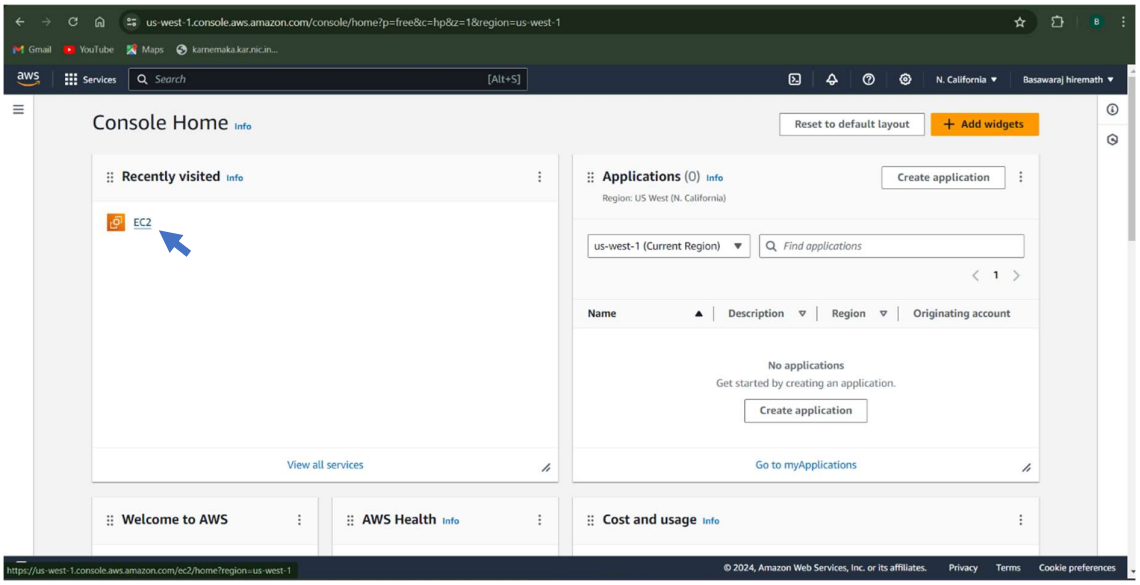
Environment set up: I am using **AWS** for deployment

Step 1: I already have an AWS account and I'm logging in now.



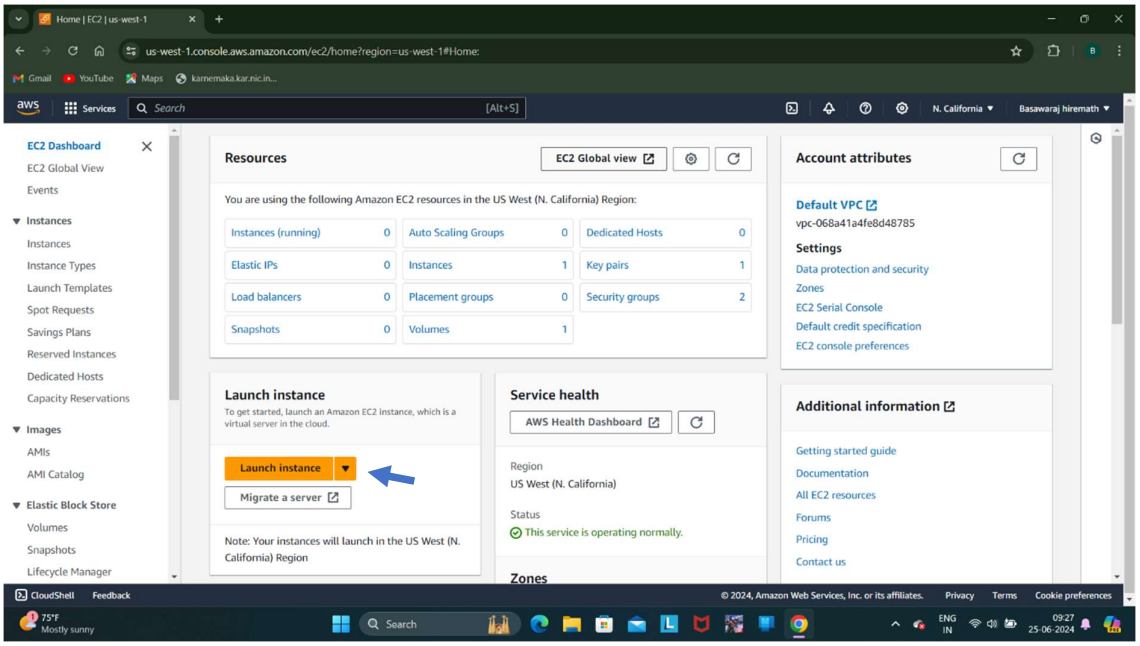
Step 2: Select **EC2** service

Elastic Compute Cloud: AWS Primary Web Service that provides resizable compute capacity.



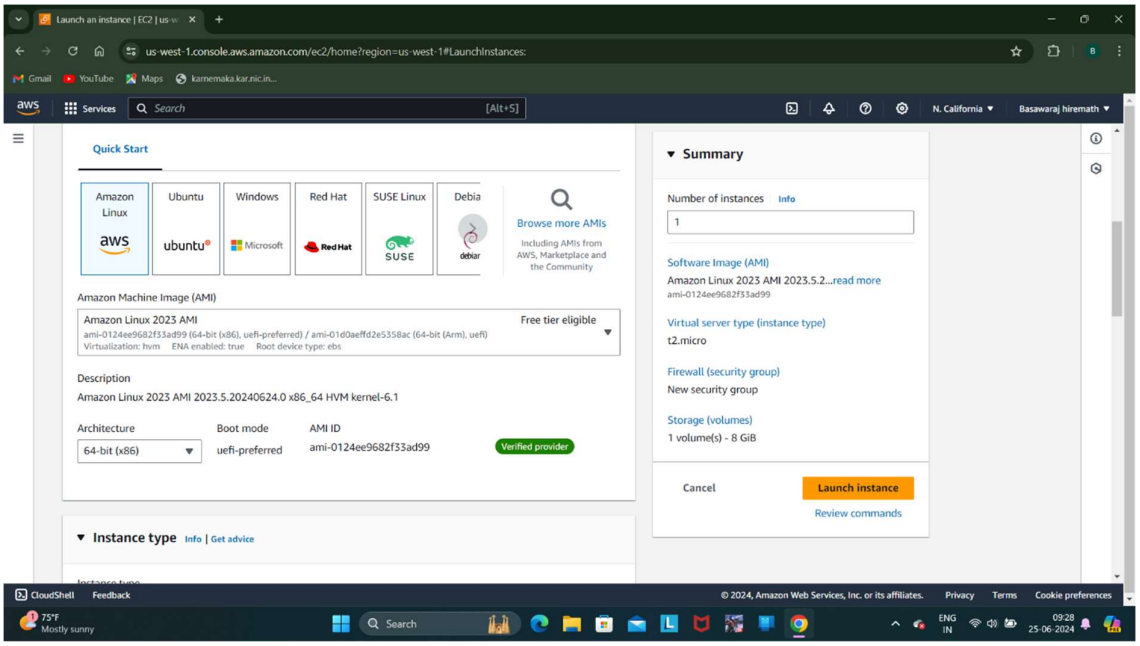
launch virtual servers called **instances** depending on your needs

Step 3: Click on Launch Instance



Amazon Machine Image (AMI): It is the initial software that will be on the instance when the instance is launched.

Step 4: Choose Amazon Linux 2023 AMI

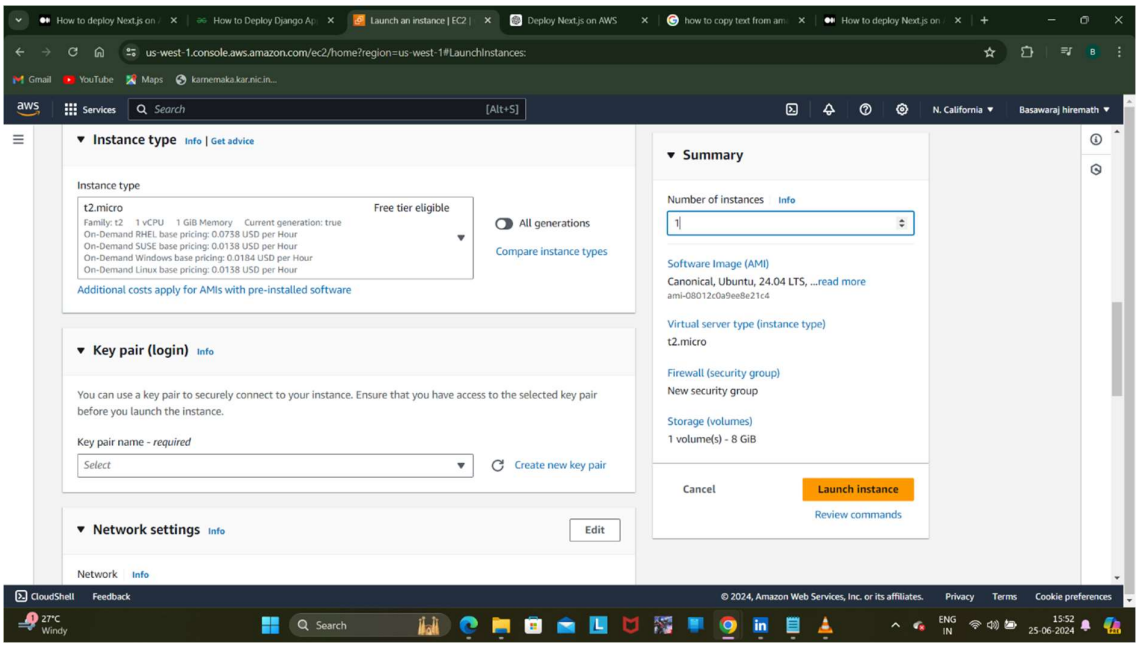


Instance Types:

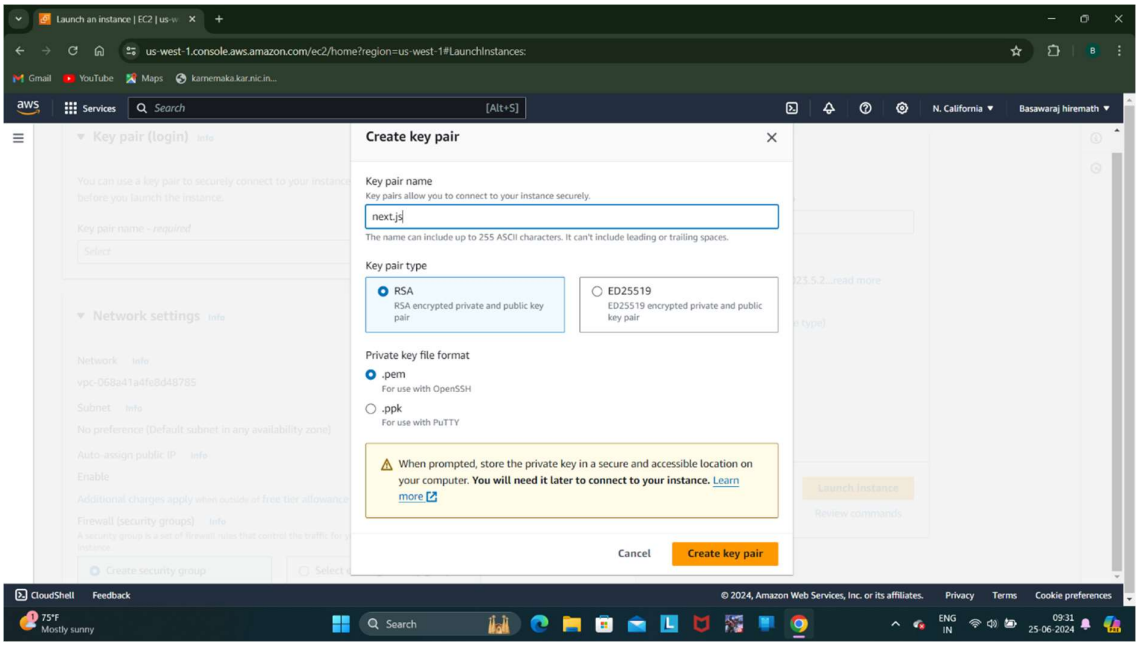
- Virtual CPUs
- Memory

- Storage
- Network Performance

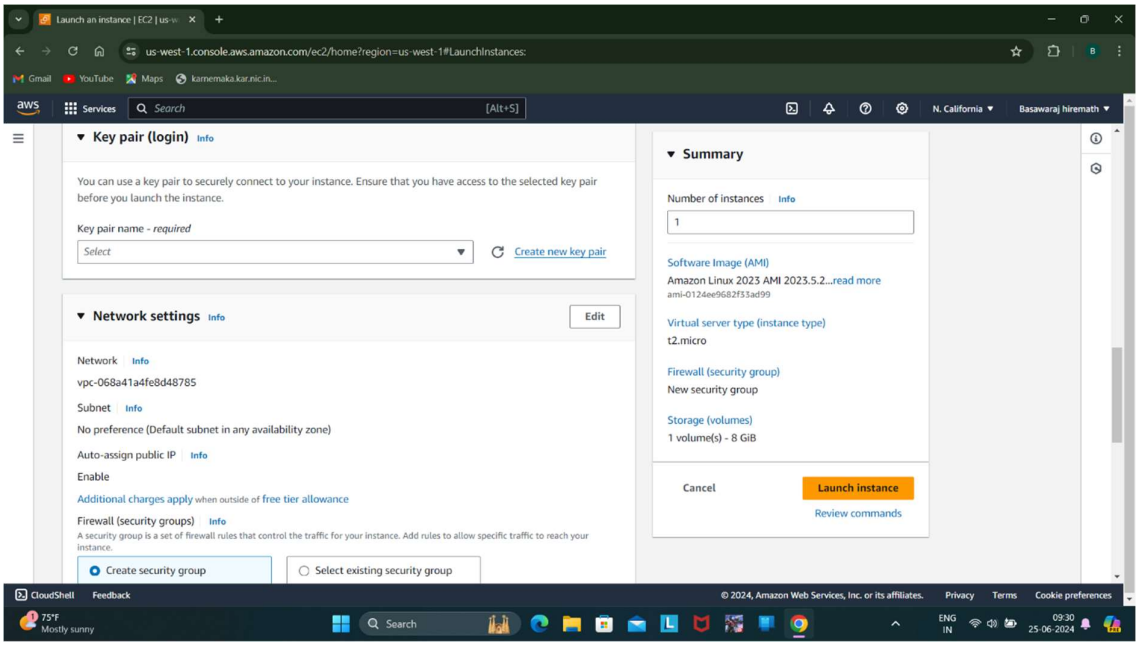
Step 5: Select instance type and create new key pair



Step 6: Set a key pair name

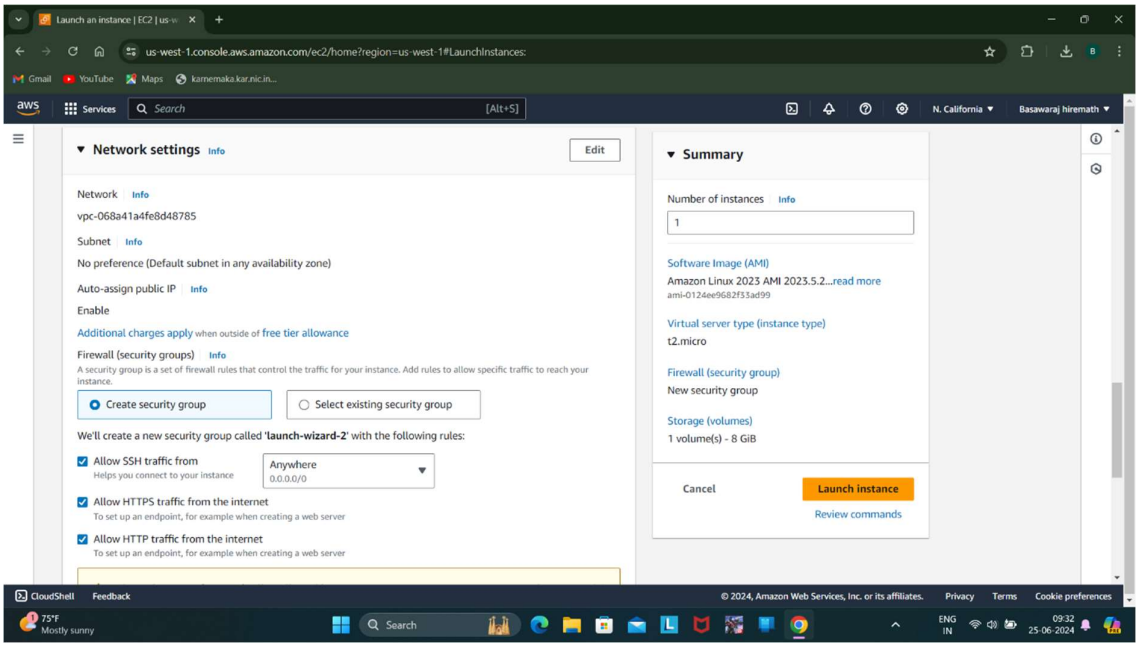


Step 7: Set Network settings as shown below



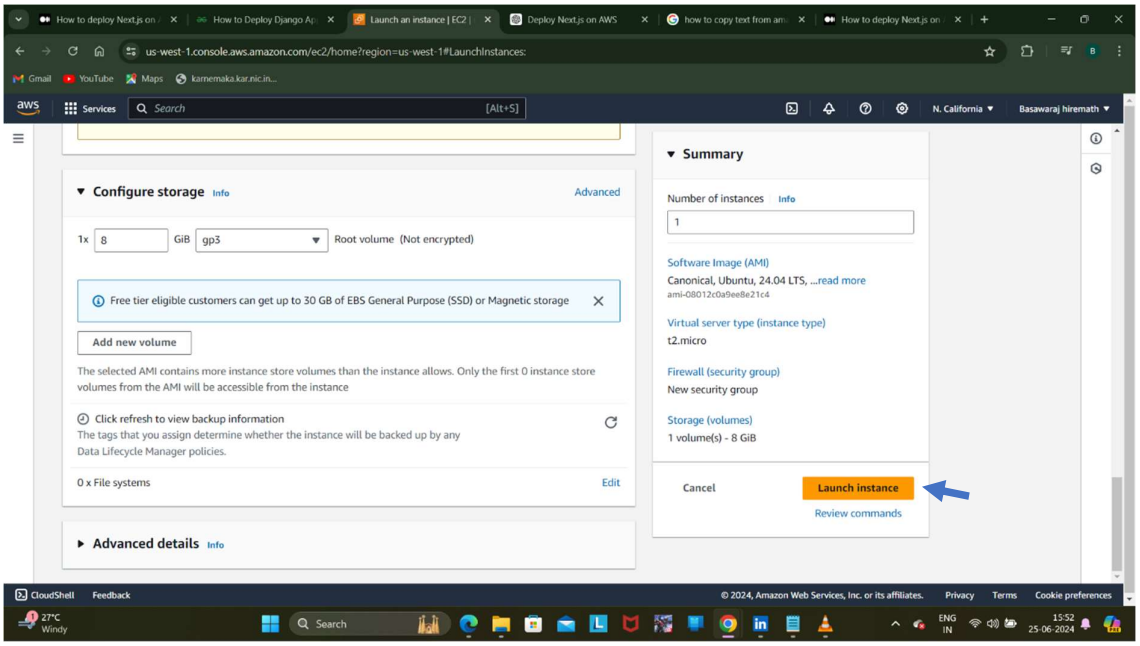
Step 8: Create new security group

Security group: It is a virtual firewall which controls inbound , outbound access to a EC2 instance and other resources

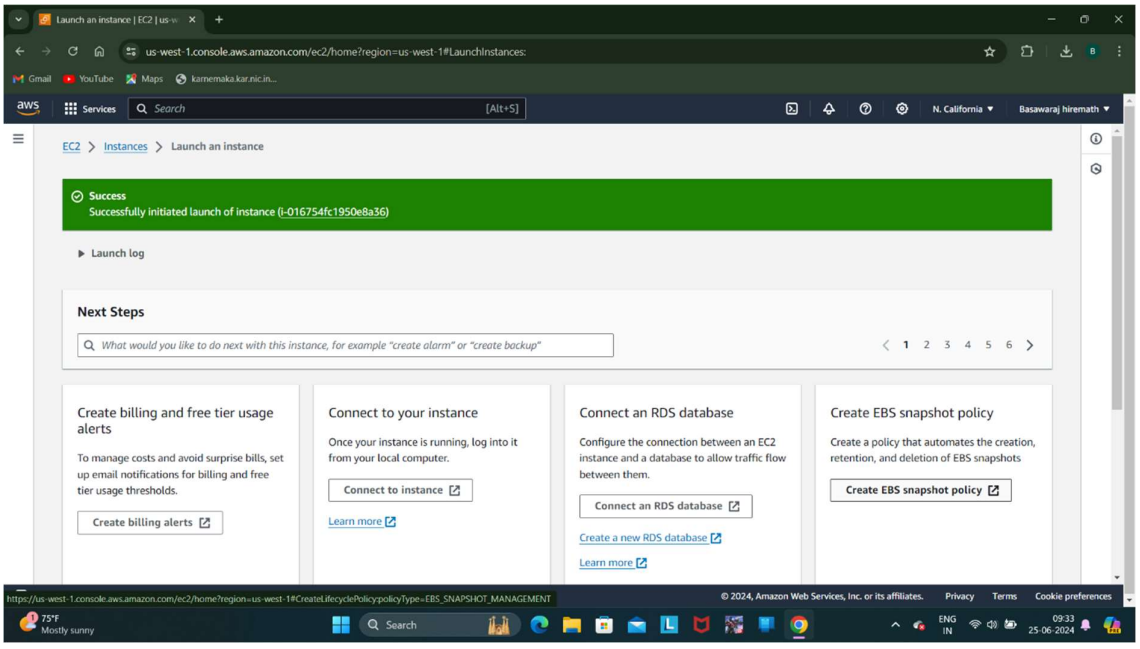


Step 9: Configure storage as shown below

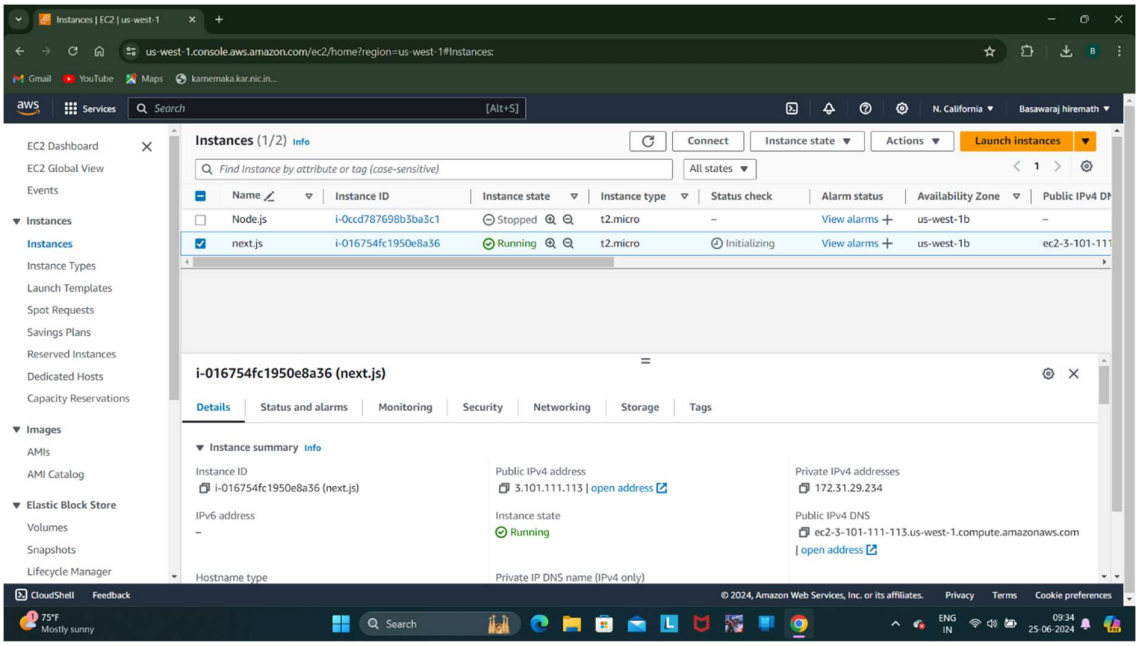
Step 10: Click on Launch instance



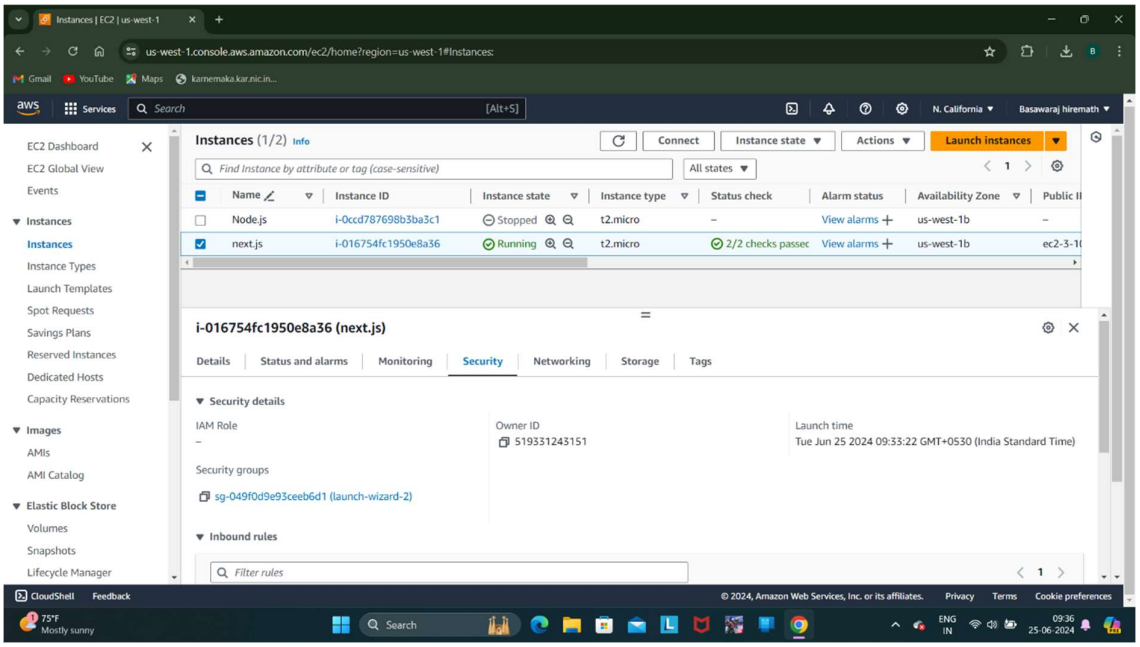
Instance have been successfully launched



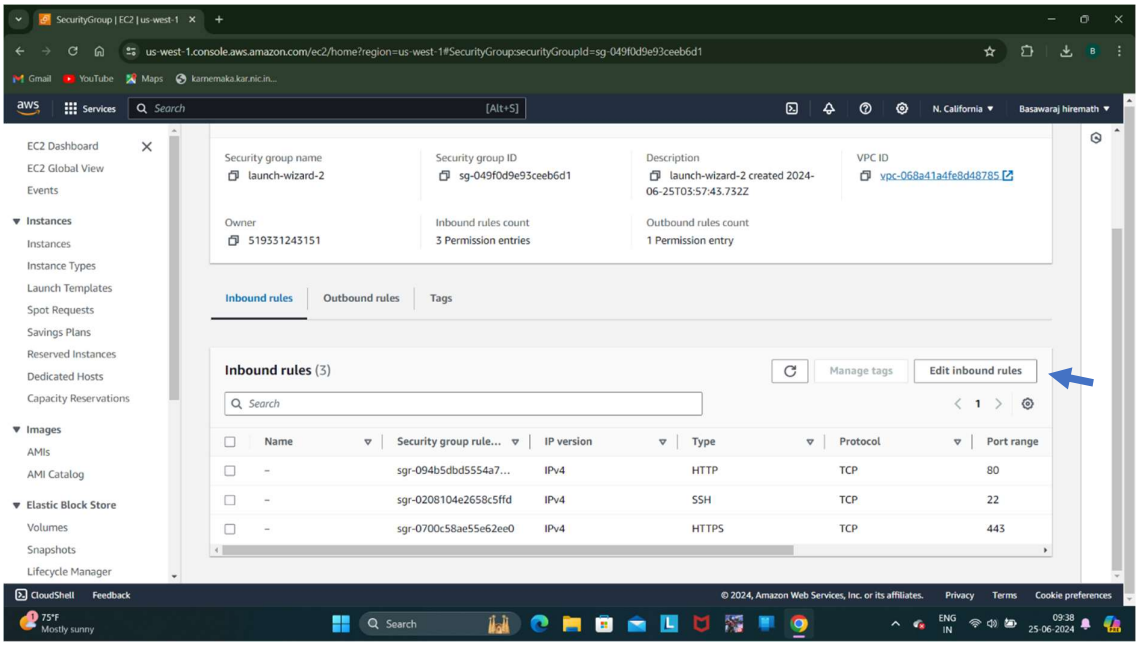
Below you can see an **instance** called **next.js** is running.



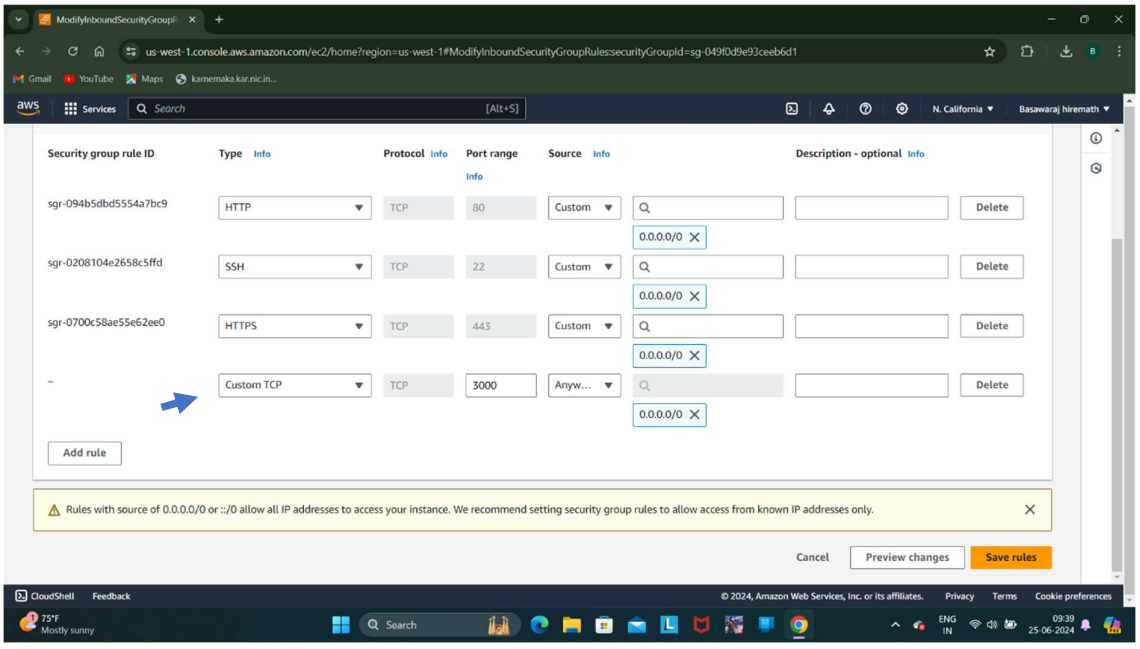
Step 11: Got to **Security** section to edit inbound rules



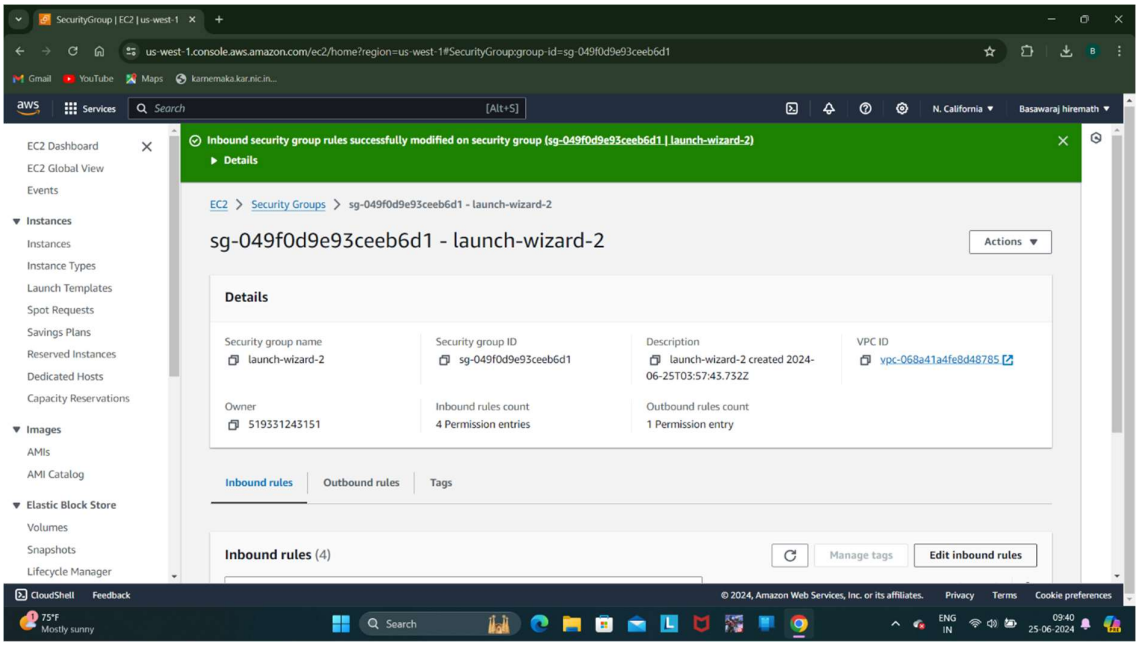
Step 12: Click on edit inbound rules

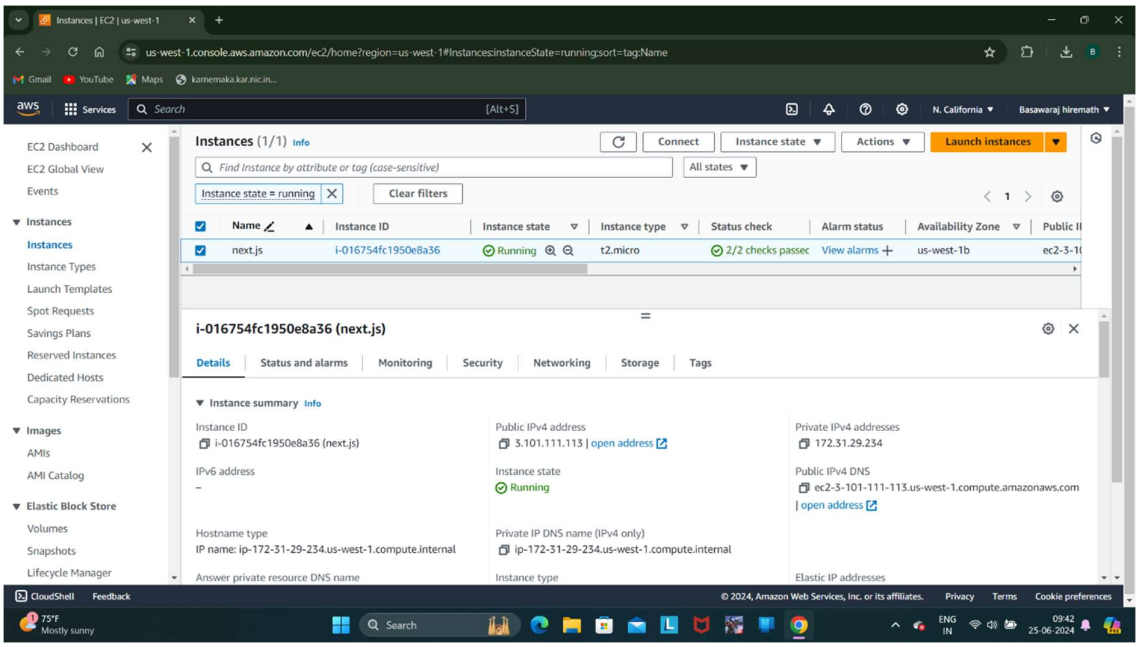


Step 13: Add new rule and Click on save rules.



Inbound security group successfully modified.

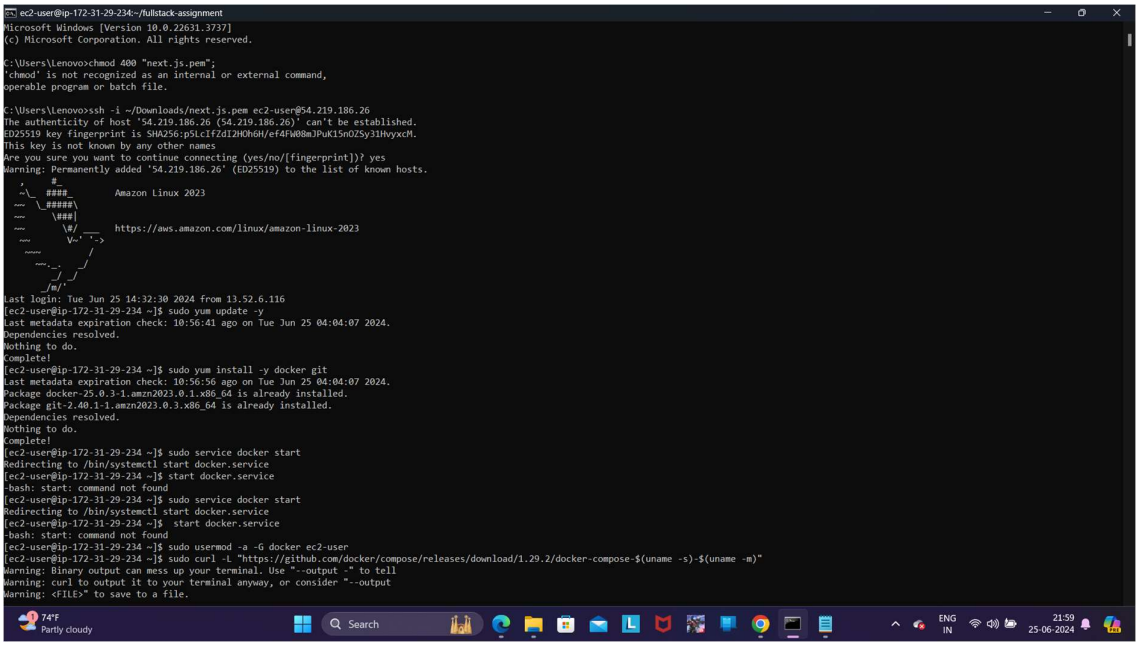




Step 14: Connect the EC2 Instance

Open a terminal where your .pem file is present and run the following commands:

- ssh -i ~/Downloads/next.js.pem ec2-user@54.219.186.26
- After this we have to upgrade our machine.
- we need docker and git hub, these are few prerequisites.

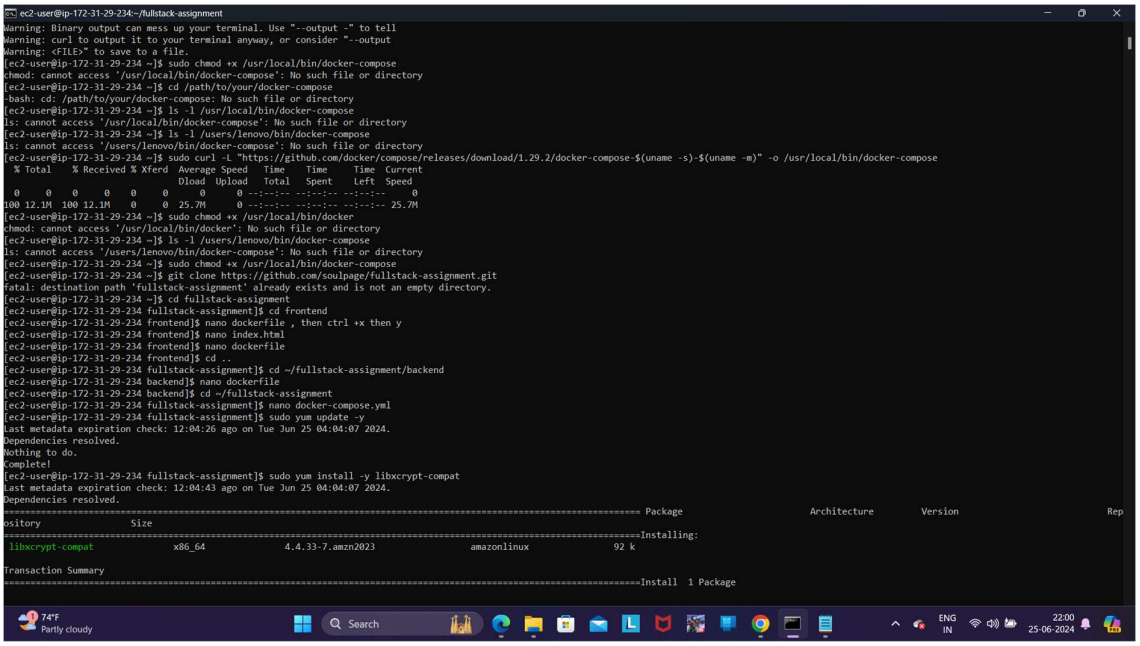


Step 15: To set up an environment follow below commands

- sudo yum update -y
- 2 sudo yum install -y docker git
- 3 sudo service docker start
- start docker. service
- sudo service docker start
- start docker. service
- sudo usermod -a -G docker ec2-user
- sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)"
- sudo chmod +x /usr/local/bin/docker-compose

Then, we need to clone with the provided repository.

- git clone https://github.com/soulpage/fullstack-assignment.git
- cd fullstack-assignment
- cd frontend

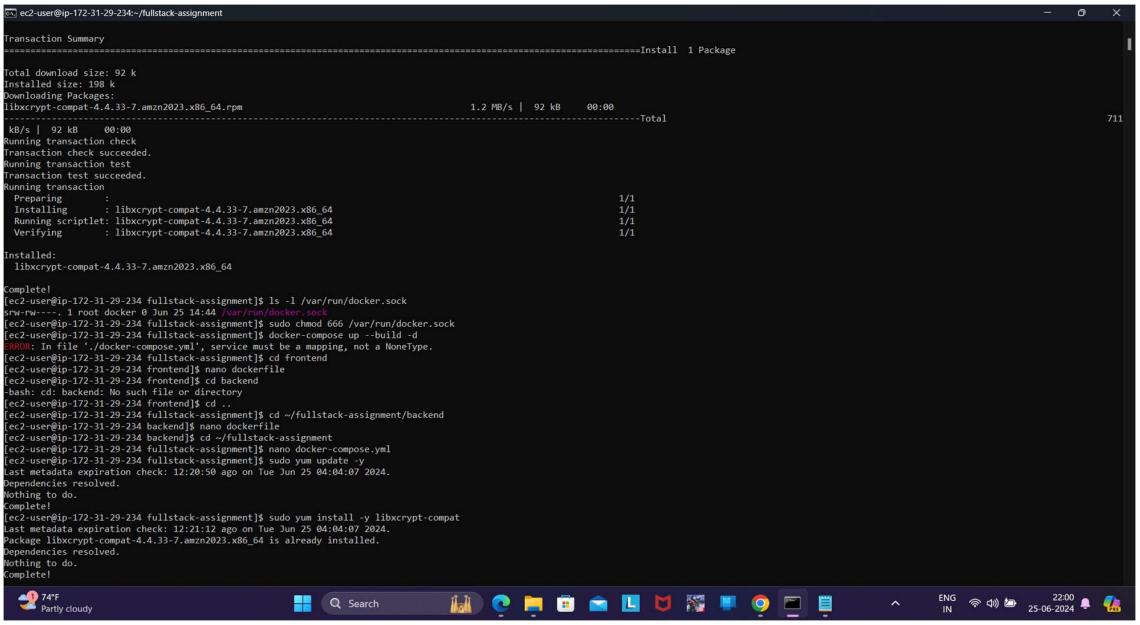


Step 16: Then, create the **dockerfile** for the **frontend** → in the terminal type nano dockerfile

- FROM node:18
- WORKDIR /app
- COPY package*. json ./
- RUN npm install
- COPY . .
- RUN npm run build
- EXPOSE 3000
- CMD ["npm", "start"]

Then type

- cd ..
- cd ~/fullstack-assignment/backend
- nano dockerfile
- type ctrl+x then yes and enter



<u>Step 17</u>: backend dockerfile

- FROM python:3.10
- WORKDIR /app
- COPY dependencies.txt .
- RUN pip install -r dependencies.txt
- COPY . .
- EXPOSE 8000
- CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

Then type following commands

- cd ~/fullstack-assignment
- nano docker-compose.yml

we have to combine front end and backend and should create a build as docker-compose.yml in the machine created in cloud for this follow the below command.

- version: '3'
- services:
- frontend:
- build: ./frontend
- ports:"3000:3000"
- backend:
- build: ./backend
- ports:"8000:8000"

again, we should update the machine

- sudo yum update -y

- sudo yum install -y libxcrypt-compat
- ls -l /var/run/docker.sock
- chmod 666 /var/run/docker.sock

Then, run the below command -for creating the build.

- docker-compose up --build -d

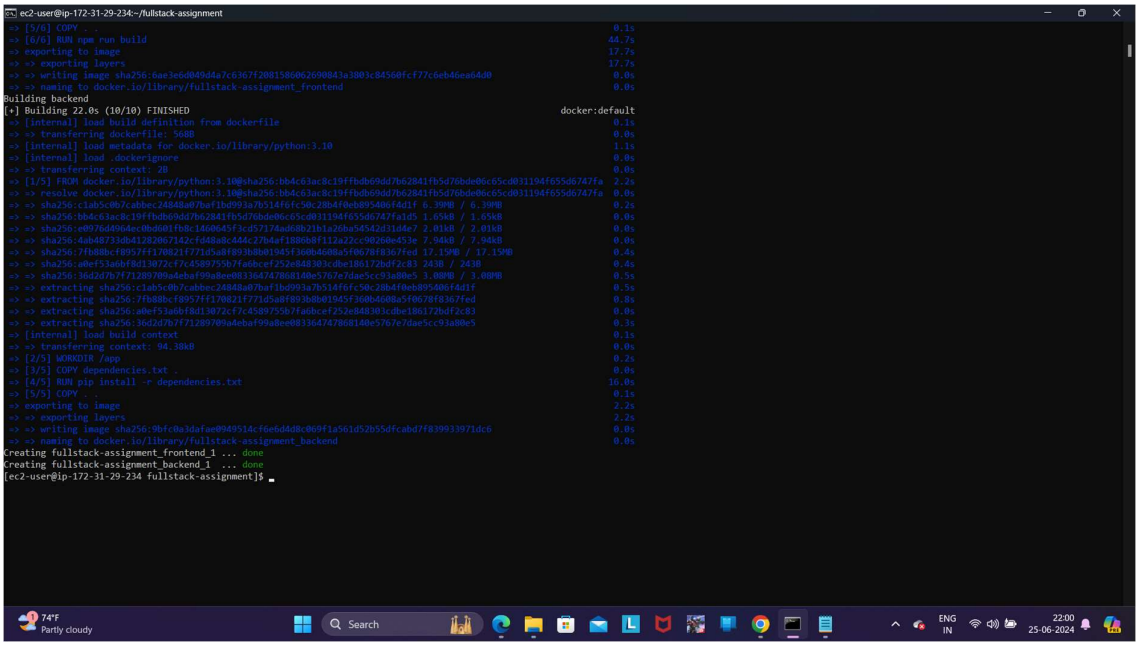
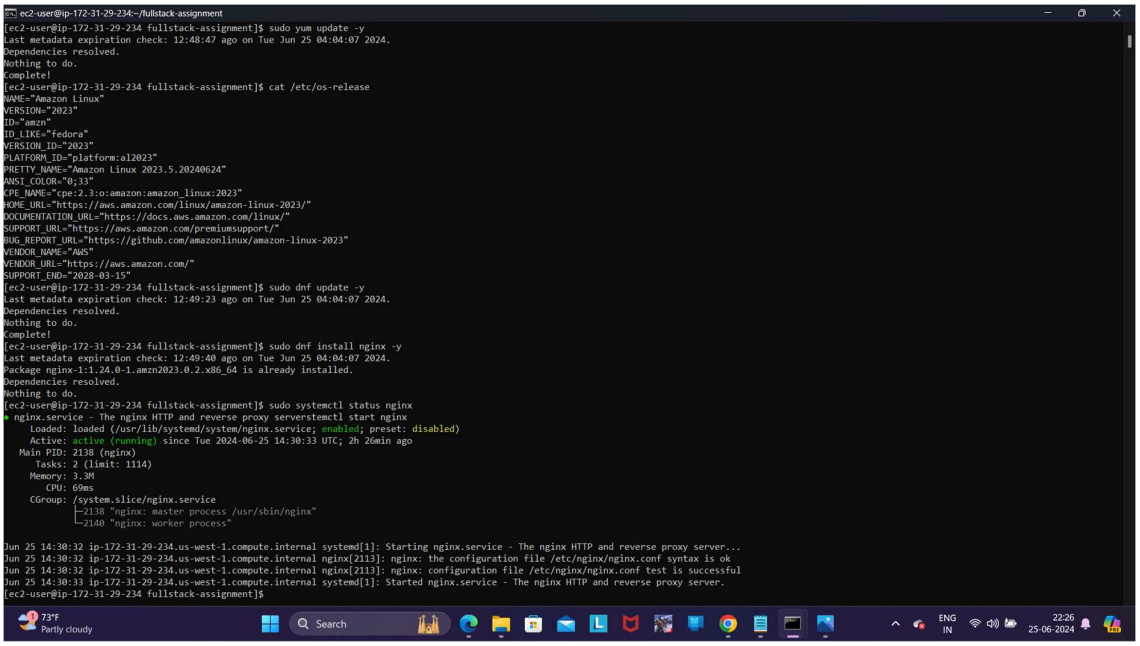


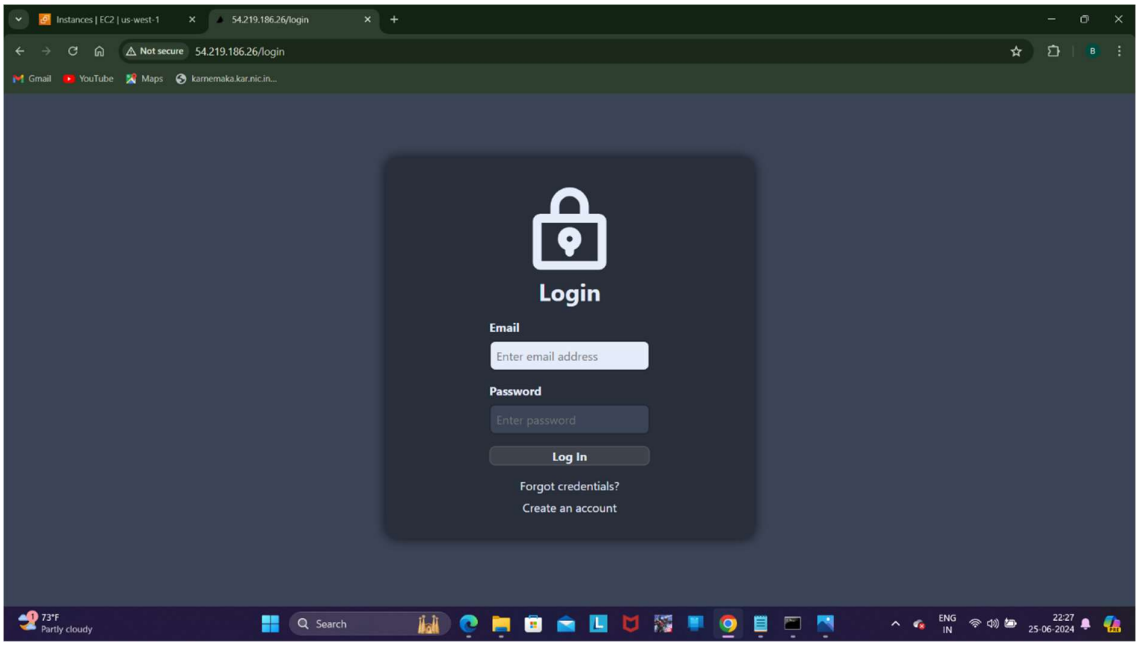
Image is successfully built.

Step now we need to expose our content to outside world for which we need to setup nginx. So, after image is built, then run the below command for the nginx-setup

- sudo yum update -y
- cat /etc/os-release
- sudo dnf update -y
- sudo dnf install nginx -y
- sudo systemctl start nginx
- sudo systemctl enable nginx
- sudo systemctl status nginx
- sudo nano /etc/nginx/conf.d/fullstack-app.conf
- sudo nginx -t



Step 18: Then check the url as your public ip ex - http://54.219.186.26/login, you will see the output like below-



To automate this process, we should go for GitHub Actions.

Go to your GitHub repository.

Click on Settings .

name: Deploy to EC2

on:

push:

branches: [main]

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

name: Deploy to EC2

env:

PRIVATE_KEY: ${{ secrets.EC2_PRIVATE_KEY }}

HOST: ${{ secrets.EC2_HOST }}

run: |

echo "$PRIVATE_KEY" > private_key && chmod 600 private_key

ssh -o StrictHostKeyChecking=no -i private_key ec2-user@${HOST} '

cd fullstack-assignment &&

git pull origin main &&

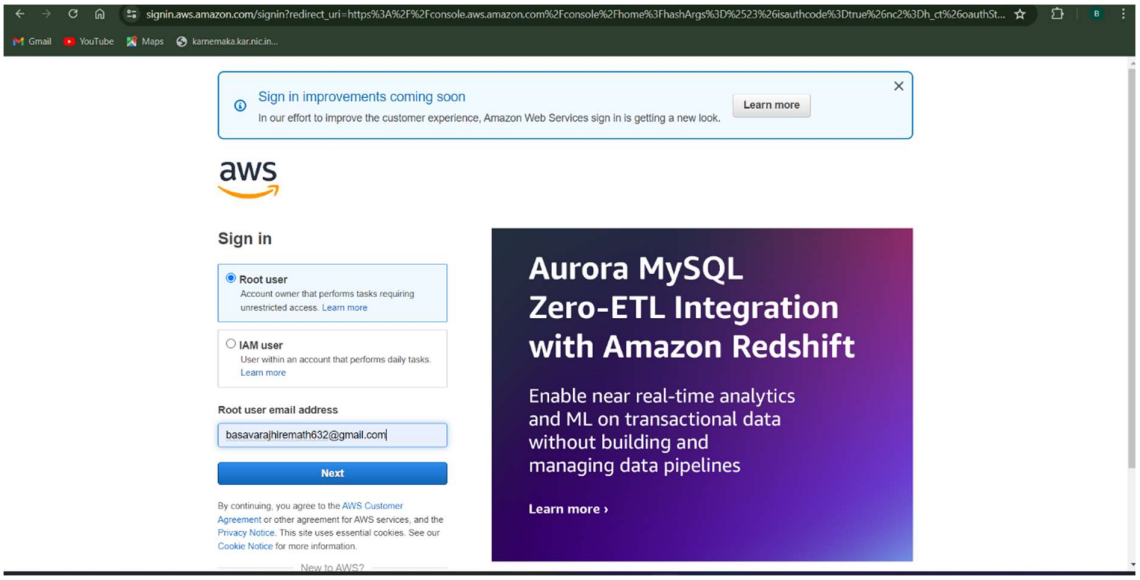
docker-compose up --build -d

Go to Secrets and click New repository secret .

Add AWS EC2_PRIVATE_KEY (Public Ip) and EC2_HOST (Region) with their respective values

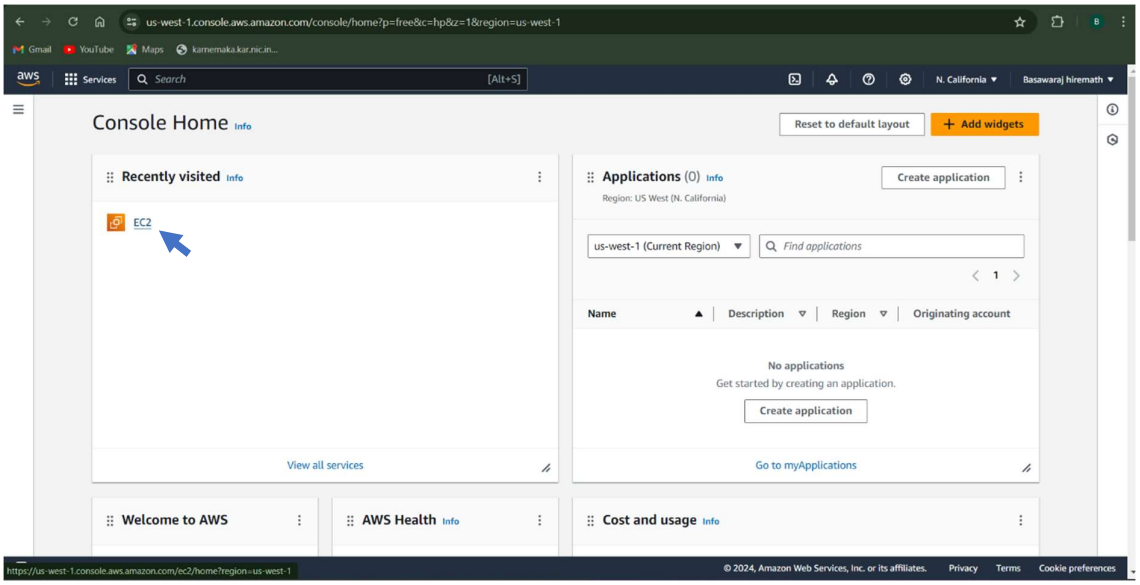
Steps to deploy Django on AWS Environment.

Step 1: I already have an AWS account and I'm logging in now.



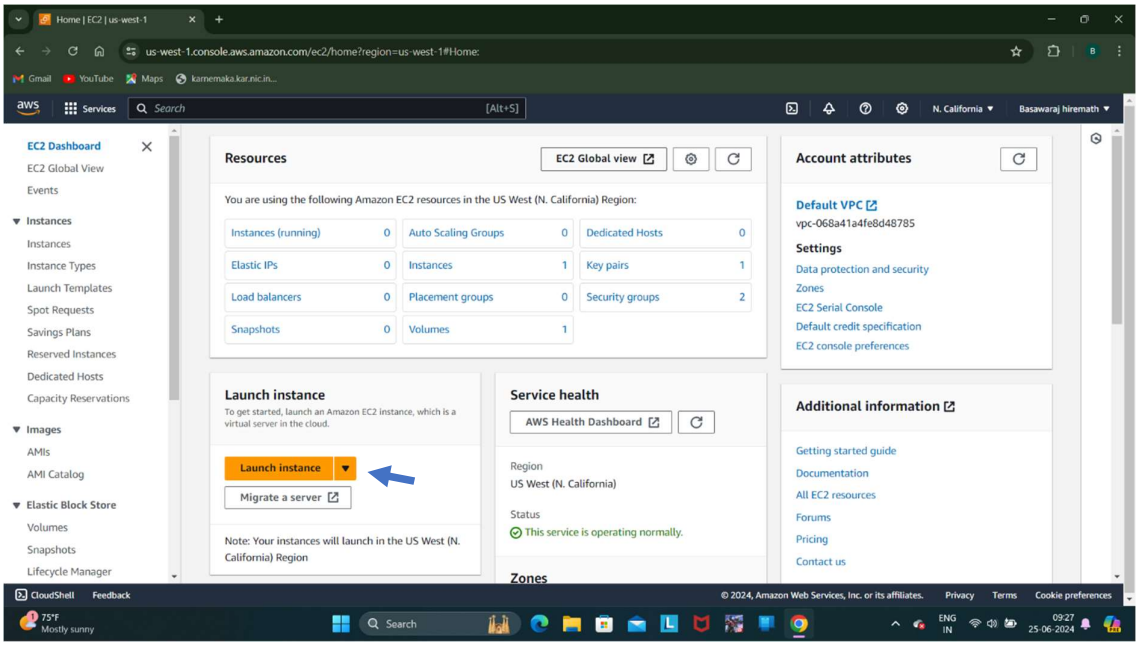
Step 2: Select **EC2** service

Elastic Compute Cloud: AWS Primary Web Service that provides resizable compute capacity.



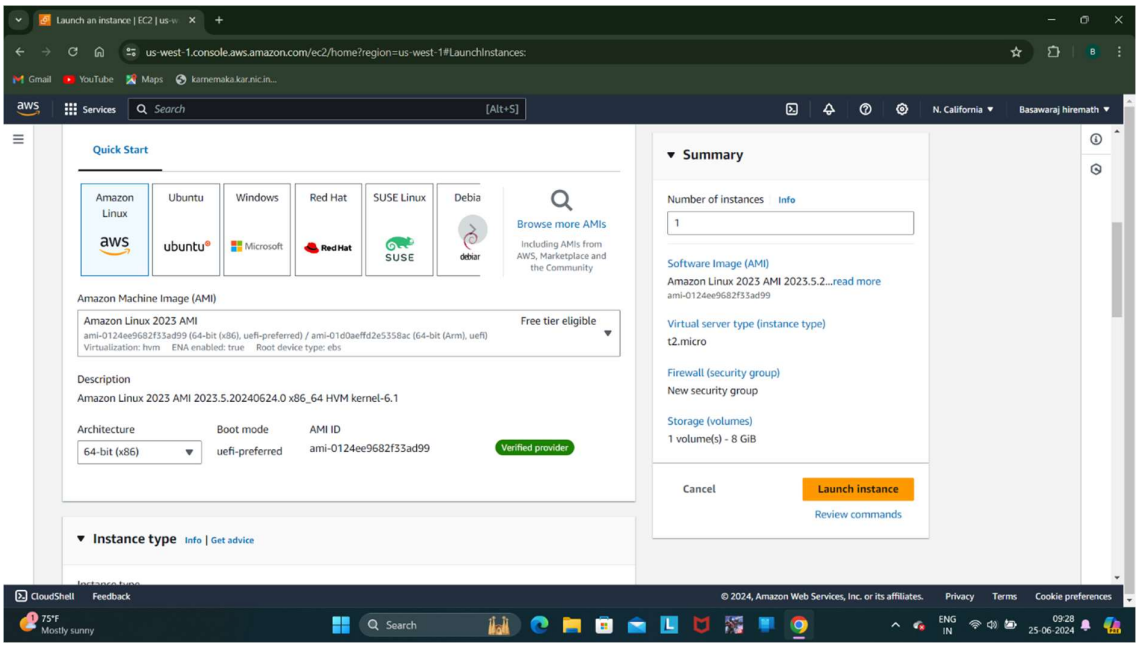
launch virtual servers called **instances** depending on your needs

Step 3: Click on Launch Instance



Amazon Machine Image (AMI): It is the initial software that will be on the instance when the instance is launched.

Step 4: Choose Amazon Linux 2023 AMI

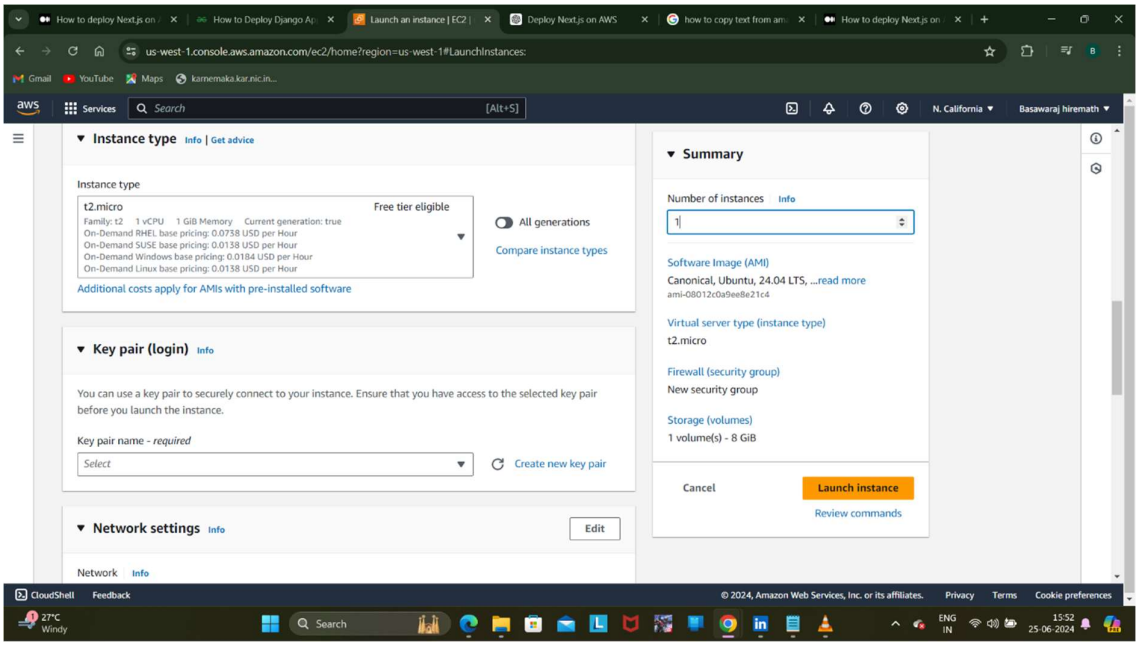


Instance Types:

- Virtual CPUs
- Memory

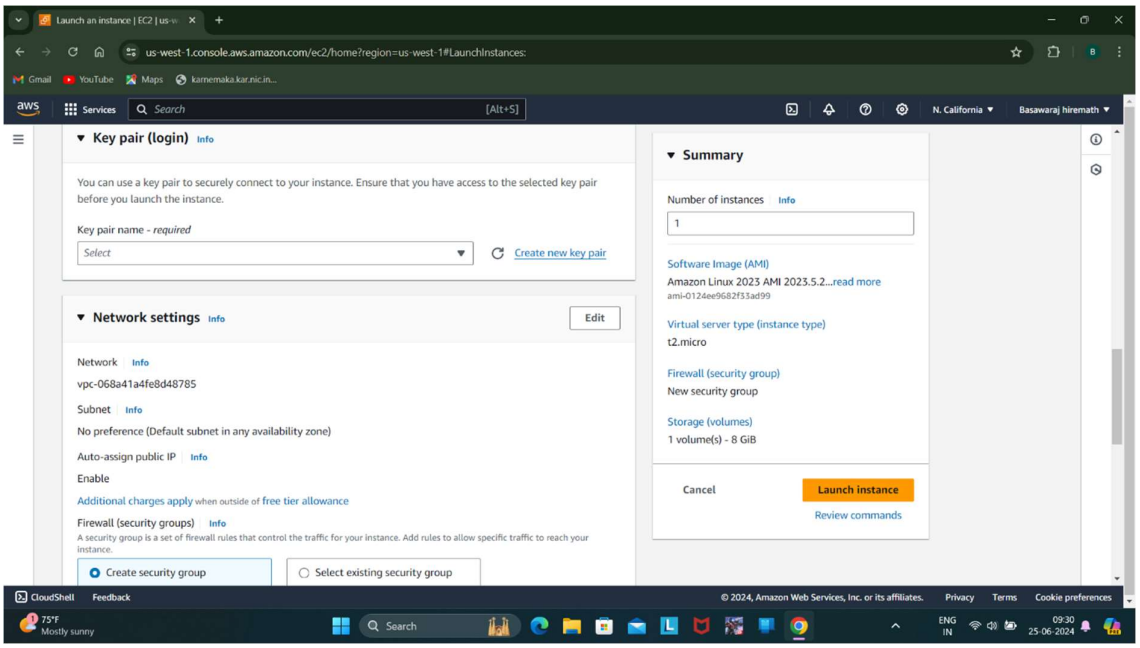
- Storage
- Network Performance

Step 5: Select instance type and create new key pair



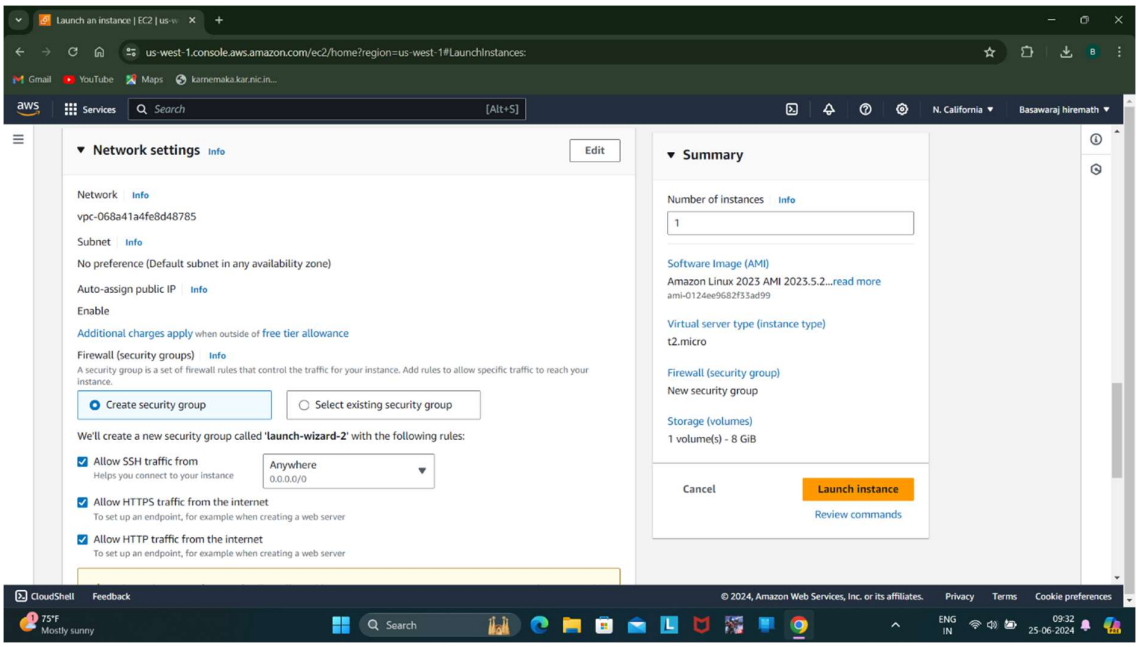
Step 6: Create new key pair

Step 7: Set Network settings as shown below



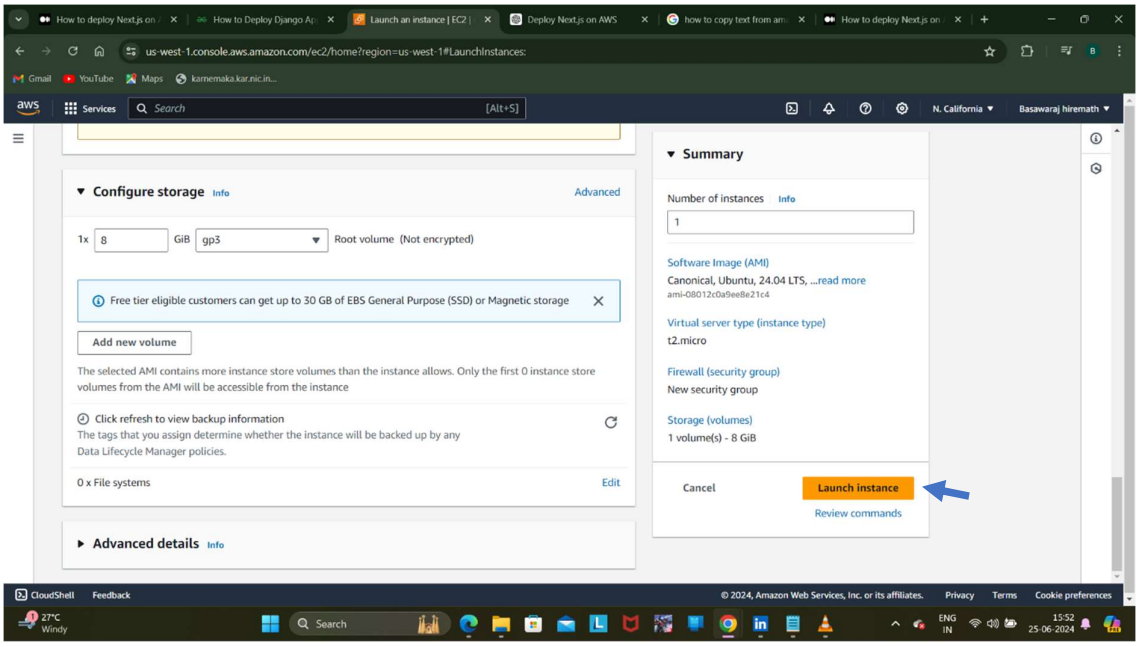
Step 8: Create new security group

Security group: It is a virtual firewall which controls inbound , outbound access to a EC2 instance and other resources

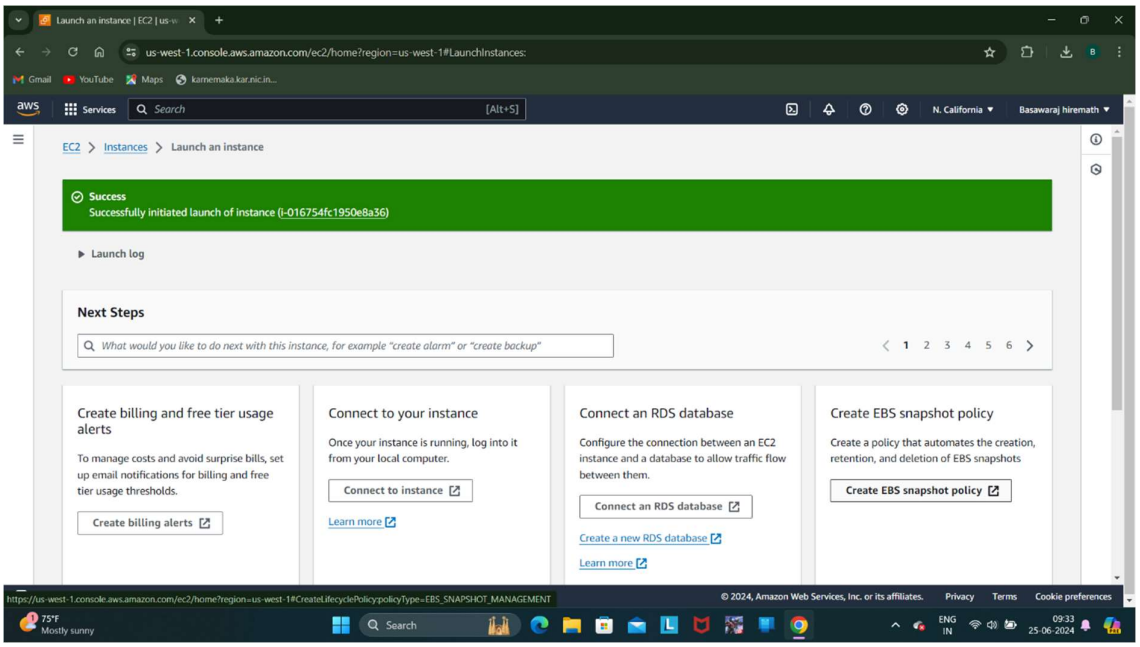


Step 9: Configure storage as shown below

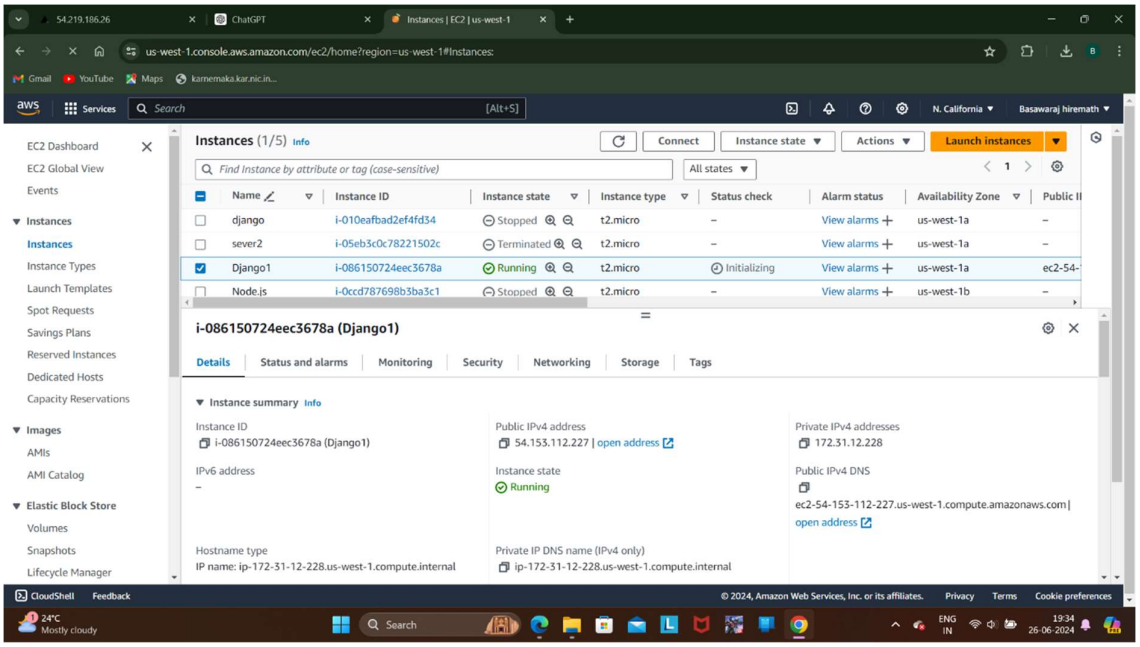
Step 10: Click on Launch instance



Instance have been successfully launched

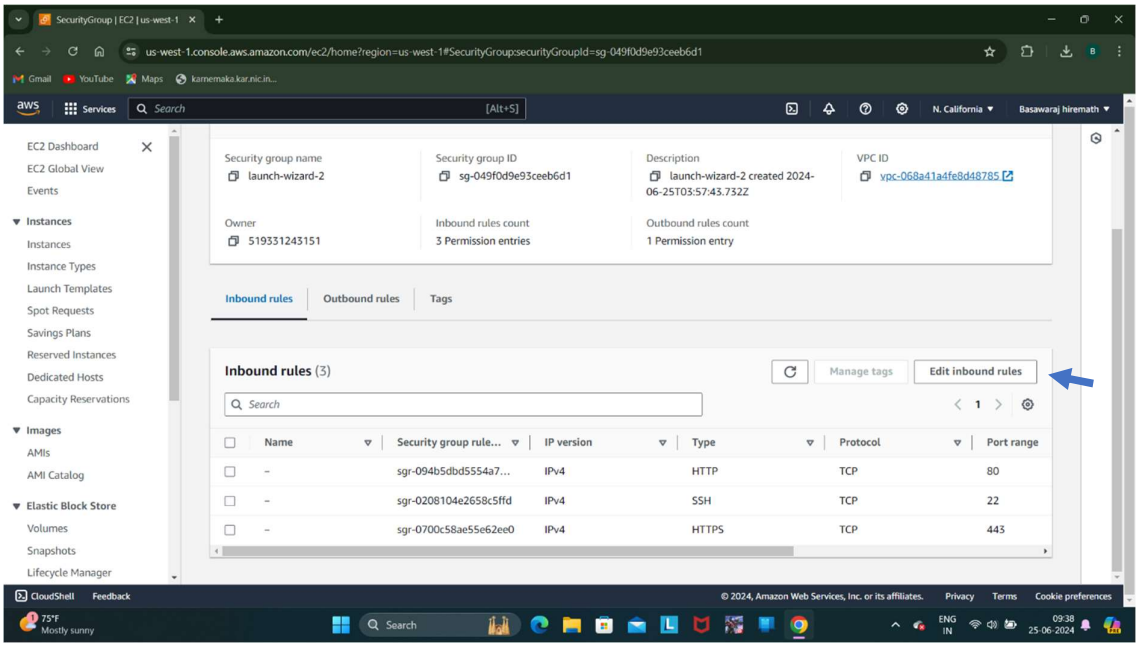


Below you can see an **instance** called **Django1** is running.

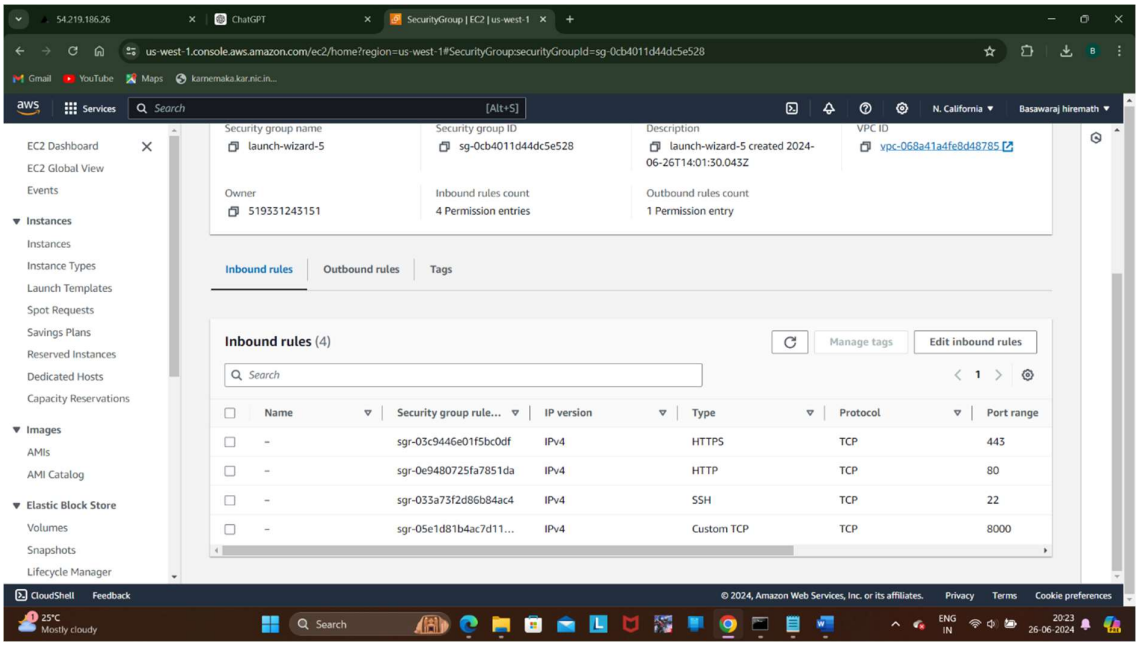


<u>Step 11:</u> Got to **Security** section to edit inbound rules

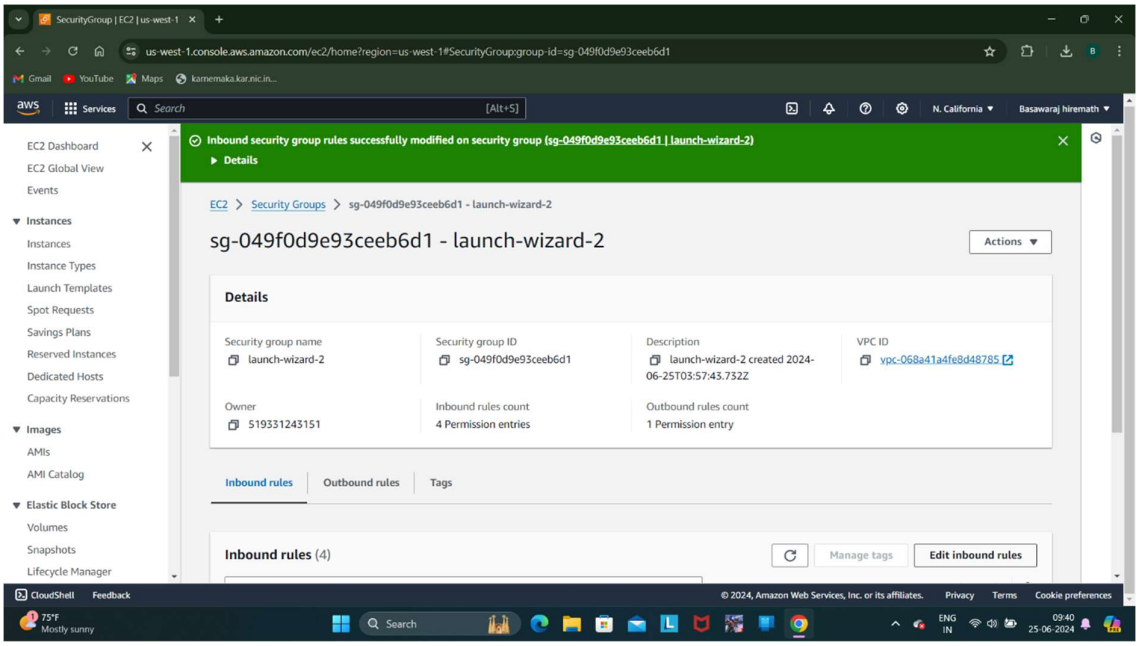
Step 12: Click on edit inbound rules



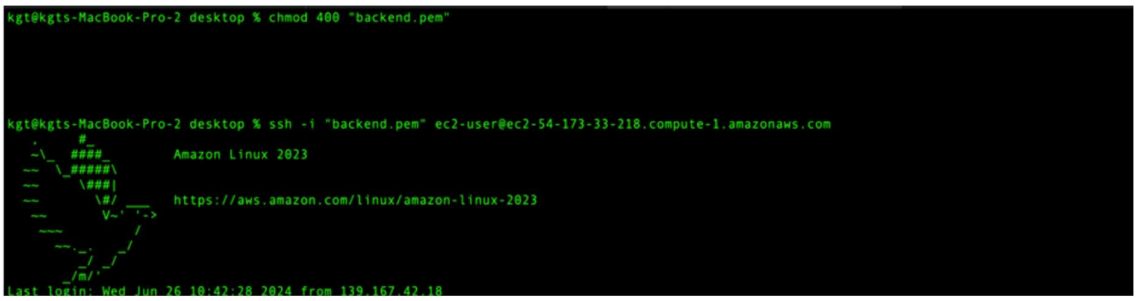
Step 13: Add new rule and Click on save rules.



Inbound security group successfully modified.



Step 14: Connect the EC2 Instance



Open a terminal where your .pem file is present and run the following commands:

- ssh -i ~/Downloads/server.pem ec2-user@54.153.112.227
- After this we have to upgrade our machine.

we need docker and git hub, these are few prerequisites.

- sudo yum update -y
- sudo yum install python3 python3-devel python3-pip gcc git -y
- python3 -m venv venv
- source venv/bin/activate
- git clone https://github.com/soulpage/fullstack-assignment.git
- cd fullstack-assignment
- cd backend

```
  Verifying        : make-1:4.3-5.amzn2023.0.2.x86_64
  Verifying        : python3-devel-3.9.16-1.amzn2023.0.8.x86_64

Installed:
  annobin-docs-10.93-1.amzn2023.0.1.noarch          annobin-plugin-gcc-10.93-1.amzn2023.0.1.x86_64    cpp-11.4.1-2.amzn2023.0.2.x86_64
  gc-8.0.4-5.amzn2023.0.2.x86_64                    gcc-11.4.1-2.amzn2023.0.2.x86_64                  glibc-devel-2.34-52.amzn2023.0.10.x8
  glibc-headers-x86-2.34-52.amzn2023.0.10.noarch    guile22-2.2.7-2.amzn2023.0.3.x86_64               kernel-headers-6.1.94-99.176.amzn202
  libmpc-1.2.1-2.amzn2023.0.2.x86_64                libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64          libxcrypt-devel-4.4.33-7.amzn2023.x8
  make-1:4.3-5.amzn2023.0.2.x86_64                  python3-devel-3.9.16-1.amzn2023.0.8.x86_64

Complete!
[ec2-user@ip-172-31-24-24 ~]$ source venv/bin/activate
(venv) [ec2-user@ip-172-31-24-24 ~]$ git clone https://github.com/soulpage/fullstack-assignment.git
fatal: destination path 'fullstack-assignment' already exists and is not an empty directory.
(venv) [ec2-user@ip-172-31-24-24 ~]$ cd backend
-bash: cd: backend: No such file or directory
(venv) [ec2-user@ip-172-31-24-24 ~]$ ls
fullstack-assignment  venv
(venv) [ec2-user@ip-172-31-24-24 ~]$ cd fullstack-assignment
(venv) [ec2-user@ip-172-31-24-24 fullstack-assignment]$ cd backend
(venv) [ec2-user@ip-172-31-24-24 backend]$
```

Then run the command for the Django

- pip install -r dependencies.txt

```
  Downloading yarl-1.9.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (269 kB)
     |████████████████████████████████| 269 kB 52.4 MB/s
Requirement already satisfied: setuptools in /home/ec2-user/venv/lib/python3.9/site-packages (from django-softdelete==0.10.5->-r dependencies.txt (line
 16)) (59.6.0)
Collecting wheel
  Downloading wheel-0.43.0-py3-none-any.whl (65 kB)
     |████████████████████████████████| 65 kB 7.3 MB/s
Installing collected packages: typing-extensions, multidict, idna, frozenlist, yarl, urllib3, sqlparse, six, platformdirs, filelock, distlib, charset-n
ormalizer, certifi, attrs, async-timeout, asgiref, aiosignal, wheel, virtualenv, tqdm, tomli, requests, PyYAML, pytz, python-monkey-business, python-da
teutil, pyflakes, pycodestyle, pathspec, packaging, nodeenv, mypy-extensions, mccabe, identify, h11, Django, click, cfgv, aiohttp, uvicorn, tzdata, pyt
hon-dotenv, pre-commit, openai, isort, freezegun, flake8, djangorestframework, django-softdelete, django-nested-admin, django-cors-headers, colorama, b
lack
Successfully installed Django-4.2.7 PyYAML-6.0.1 aiohttp-3.8.5 aiosignal-1.3.1 asgiref-3.7.2 async-timeout-4.0.3 attrs-23.1.0 black-23.9.1 certifi-2023
.7.22 cfgv-3.4.0 charset-normalizer-3.3.0 click-8.1.7 colorama-0.4.6 distlib-0.3.7 django-cors-headers-4.3.1 django-nested-admin-4.0.2 django-softdelet
e-0.10.5 djangorestframework-3.14.0 filelock-3.12.4 flake8-6.1.0 freezegun-1.2.2 frozenlist-1.4.0 h11-0.14.0 identify-2.5.30 idna-3.4 isort-5.12.0 mcca
be-0.7.0 multidict-6.0.4 mypy-extensions-1.0.0 nodeenv-1.8.0 openai-0.28.1 packaging-23.2 pathspec-0.11.2 platformdirs-3.11.0 pre-commit-3.4.0 pycodest
yle-2.11.0 pyflakes-3.1.0 python-dateutil-2.8.2 python-dotenv-1.0.0 python-monkey-business-1.0.0 pytz-2023.3.post1 requests-2.31.0 six-1.16.0 sqlparse-
0.4.4 tomli-2.0.1 tqdm-4.66.1 typing-extensions-4.8.0 tzdata-2023.3 urllib3-2.0.5 uvicorn-0.27.1 virtualenv-20.24.5 wheel-0.43.0 yarl-1.9.2
WARNING: You are using pip version 21.3.1; however, version 24.1 is available.
You should consider upgrading via the '/home/ec2-user/venv/bin/python3 -m pip install --upgrade pip' command.
(venv) [ec2-user@ip-172-31-24-24 backend]$
```

- nano backend/settings.py 1

Add your EC2 public IP to ALLOWED_HOSTS

- ALLOWED_HOSTS = ['your-ec2-public-ip', 'localhost', '127.0.0.1']

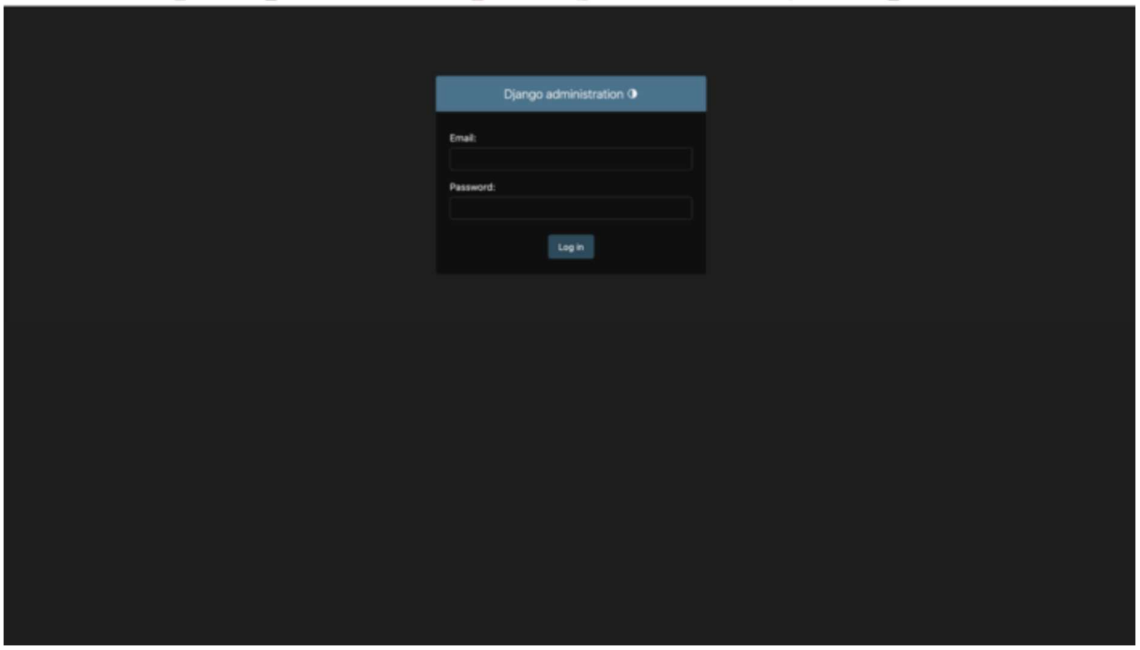
```
DEBUG = True

ALLOWED_HOSTS = ['54.173.33.218', 'localhost', '127.0.0.1']

# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "corsheaders",
    "rest_framework",
    "nested_admin",

^G Help      ^O Write Out  ^W Where Is   ^K Cut       ^T Execute   ^C Location  M-U Undo    M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify   ^/ Go To Line M-E Redo    M-6 Copy       ^Q Where Was
```

- python manage.py migrate

Then create the user for the app, give the email id and password of your choice

- python manage.py createsuperuser
- python manage.py runserver 0.0.0.0:8000

Now we can access it using http:// 54.153.112.227:8000/admin if everything is done perfect you will see the below given screen.


Django administration
Email:
Password:
Log in