

Soulpage IT Solutions Assignment

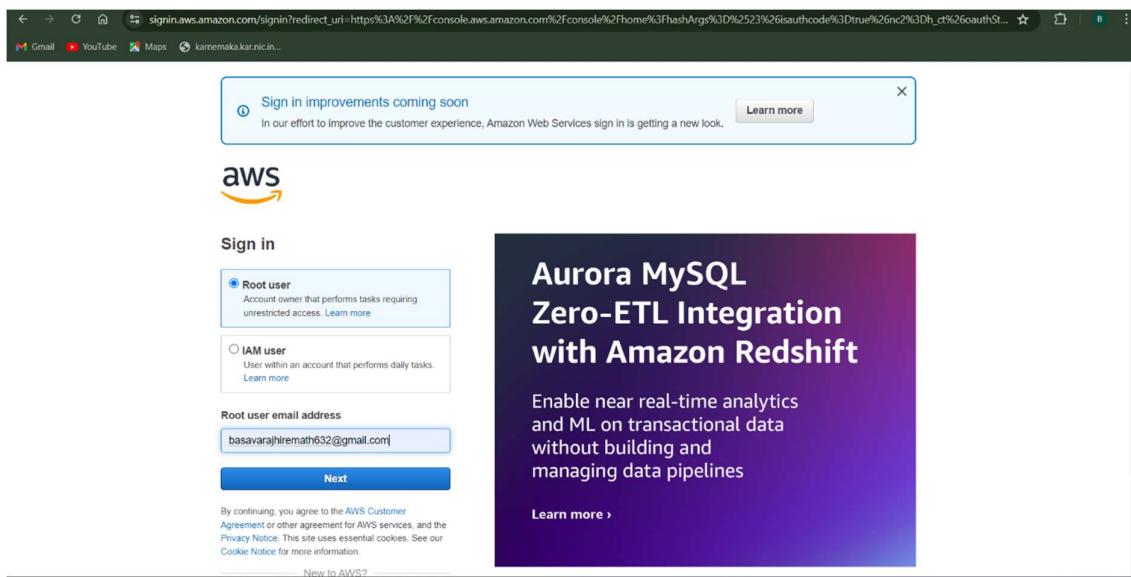
Name: Basawaraj Hiremath

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

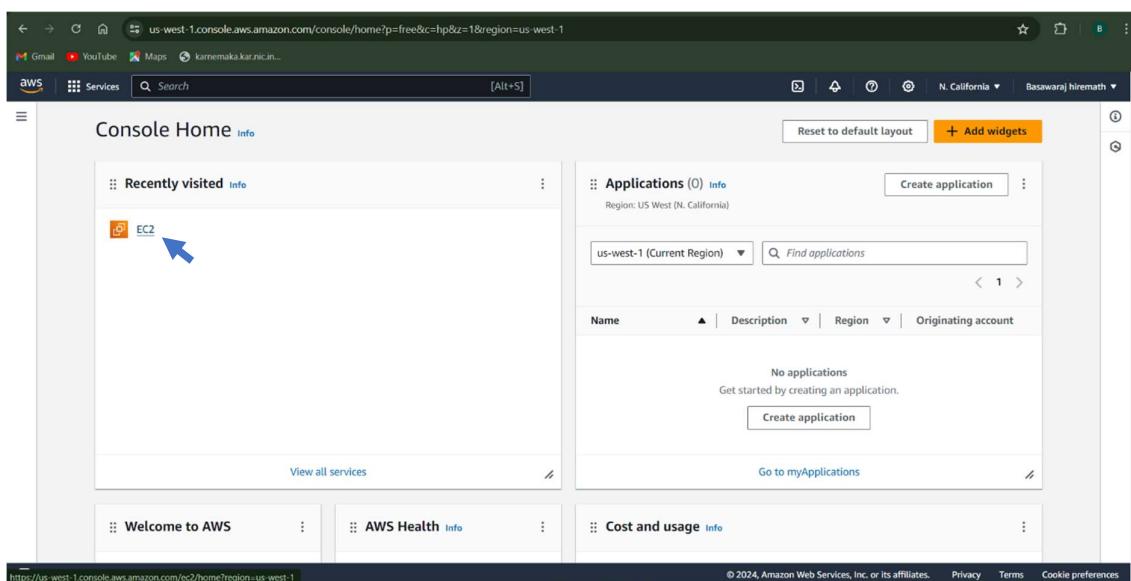
Environment set up: I am using **AWS** for deployment

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



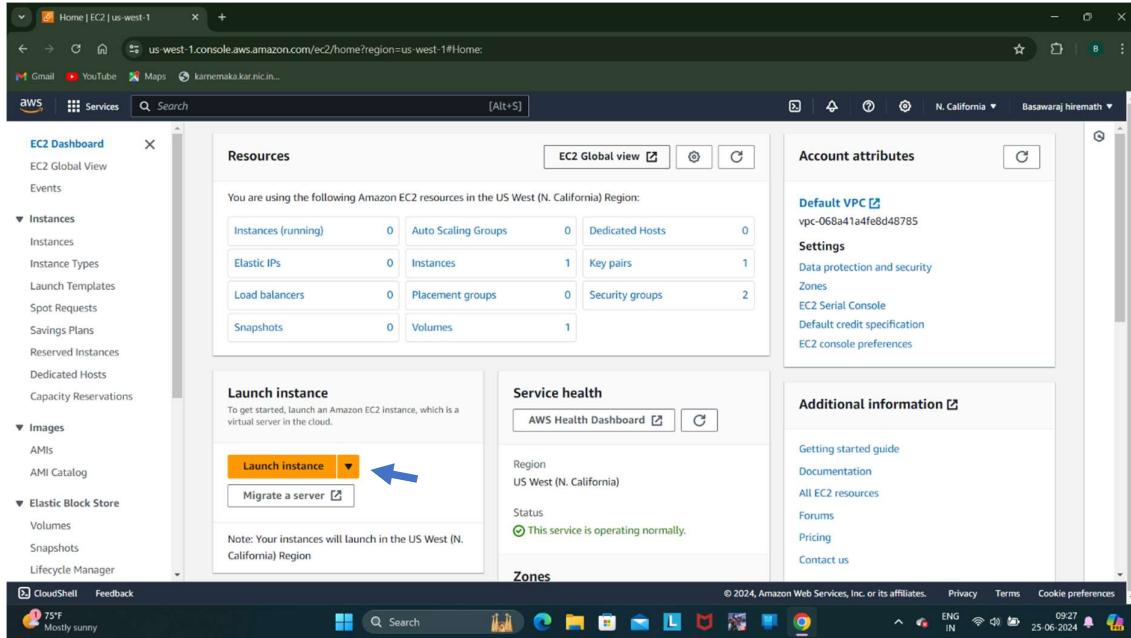
Step 2: Select EC2 service

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



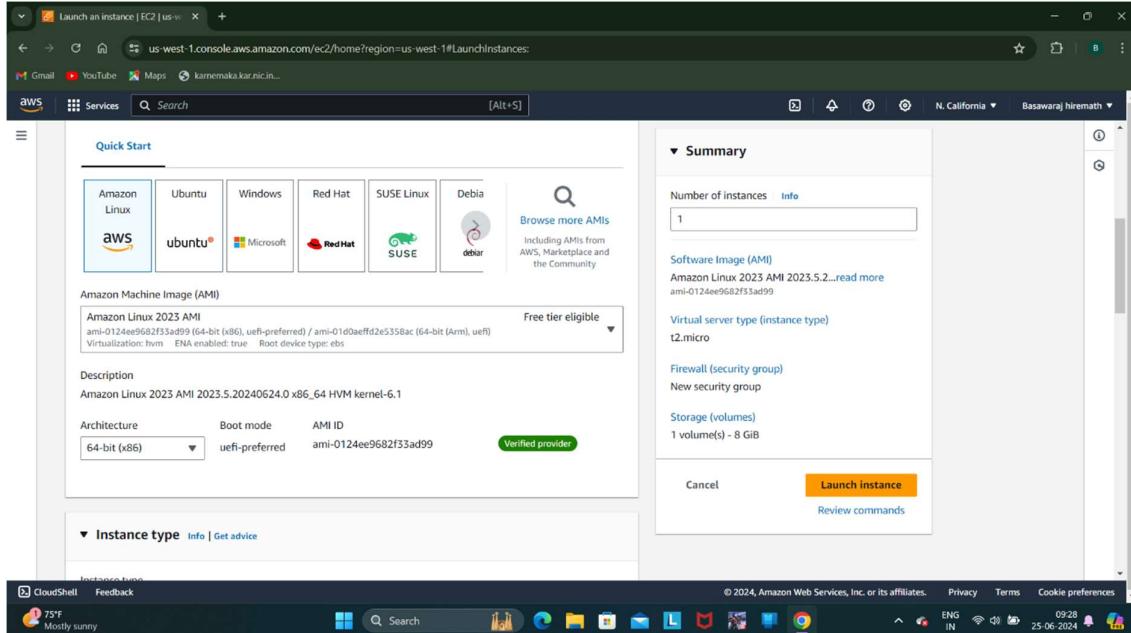
launch virtual servers called **instances** depending on your needs

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

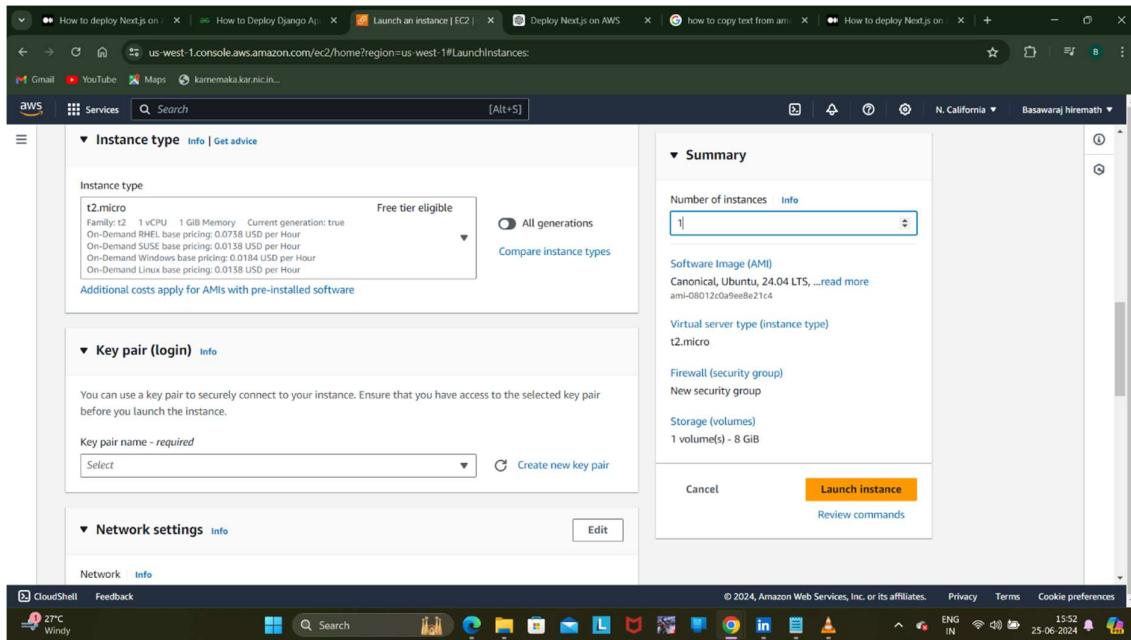


Instance Types:

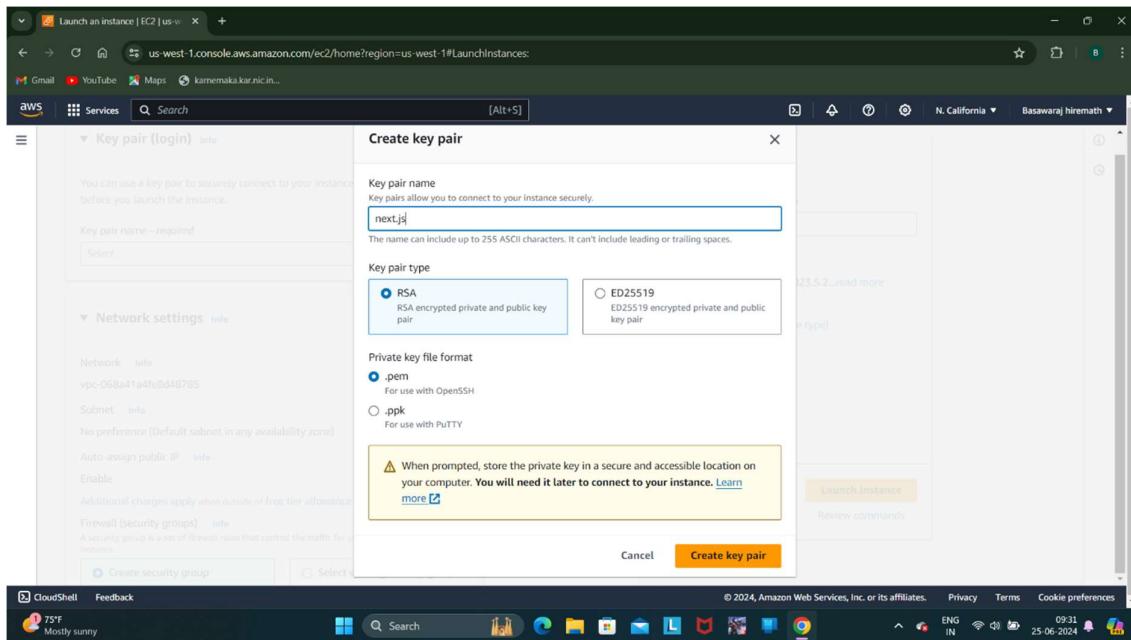
- Virtual CPUs
- Memory

- Storage
- Network Performance

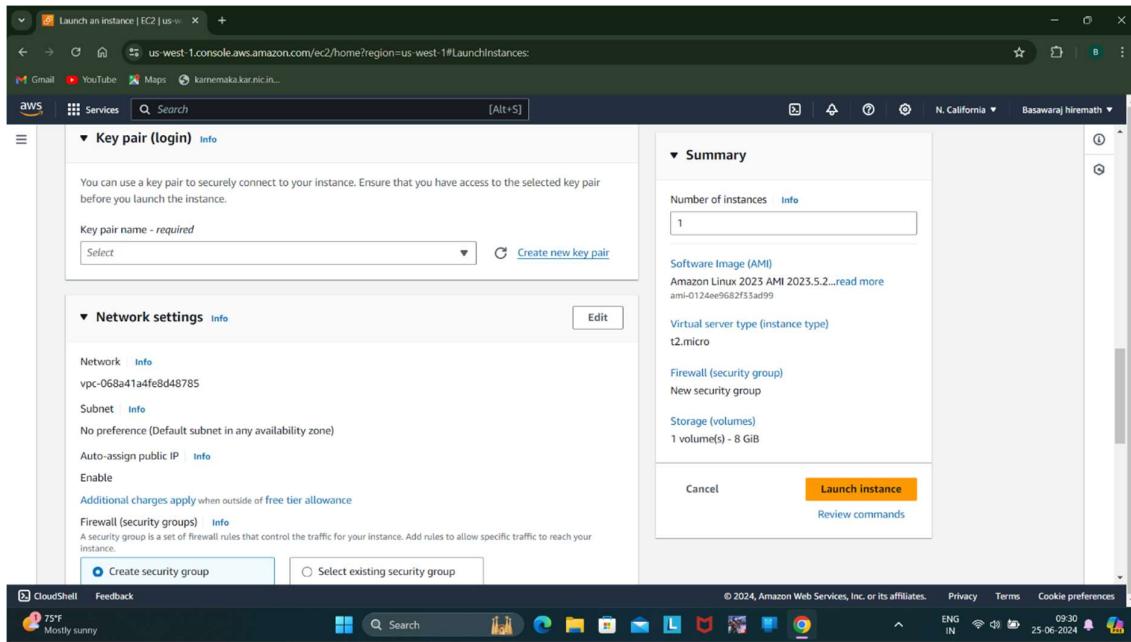
Step 5: Select instance type and create new key pair



Step 6: Set a key pair name

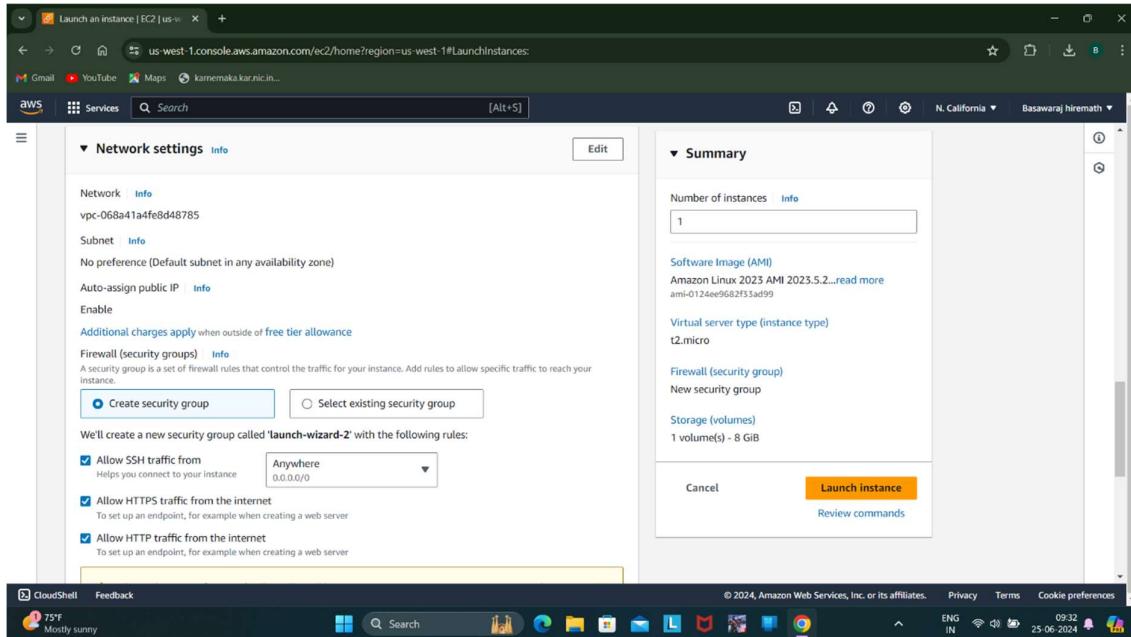


Step 7: Set Network settings as shown below



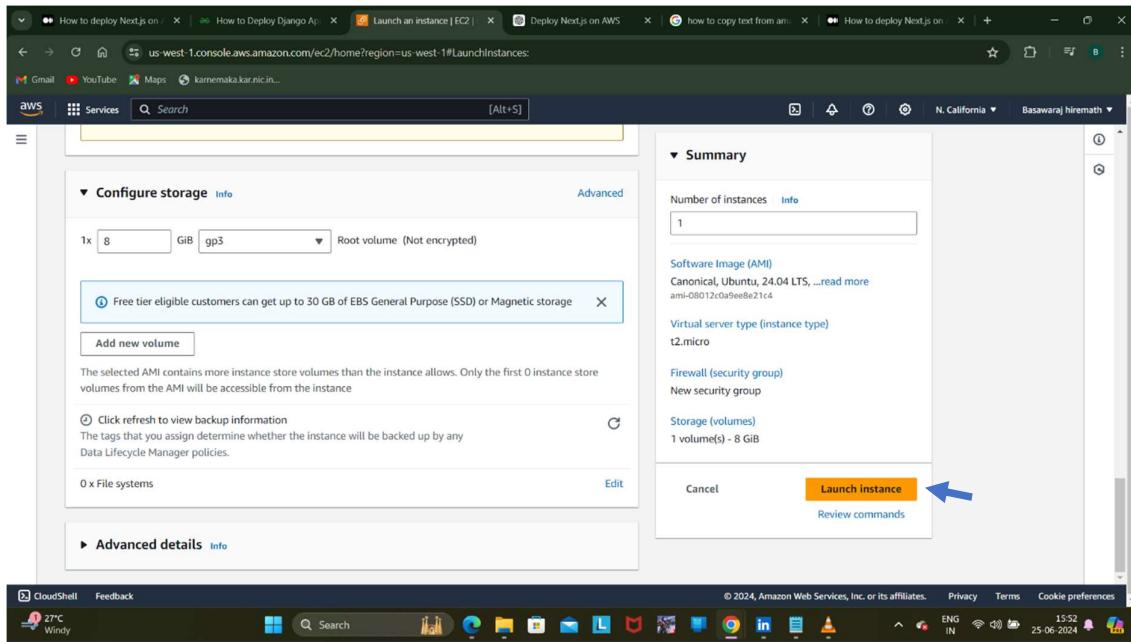
Step 8: Create new security group

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

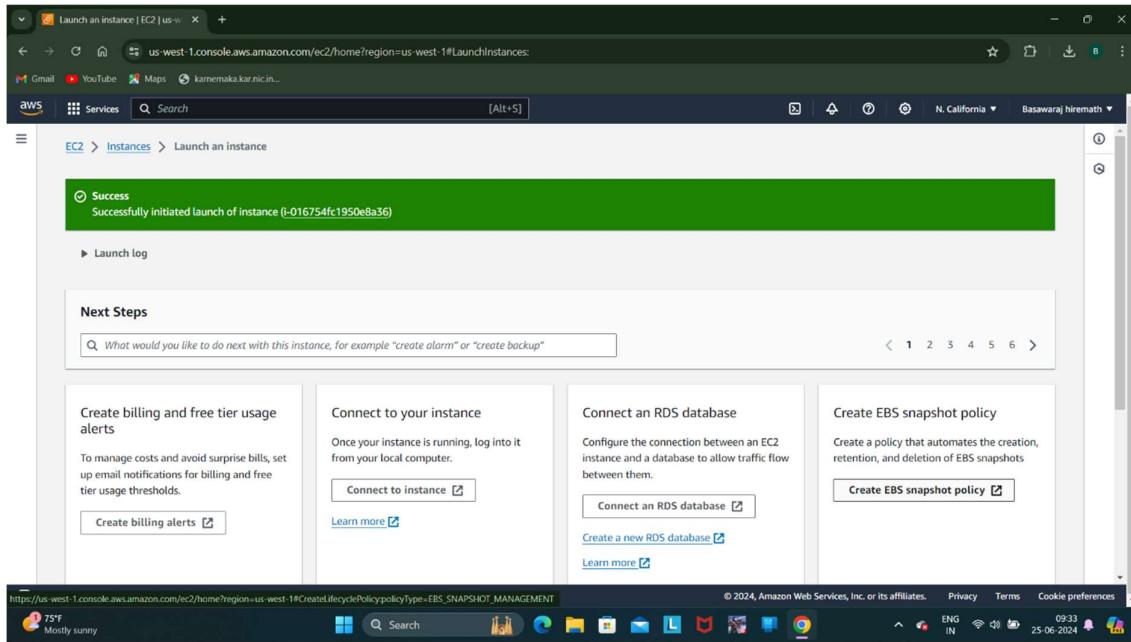


Step 9: Configure storage as shown below

Step 10: Click on Launch instance



Instance have been successfully launched



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

The screenshot shows the AWS EC2 Instances page. On the left, a sidebar lists various services like EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, AMI Catalog, and Elastic Block Store. The main content area displays a table of instances. The first instance, 'Node.js' (Instance ID: i-0cd787698b3ba3c1), is stopped. The second instance, 'next.js' (Instance ID: i-016754fc1950e8a36), is running, as indicated by the green status icon. The table includes columns for Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4 DNS. Below the table, a detailed view for the 'next.js' instance is shown, including sections for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The Security tab is selected, showing Security details (IAM Role: -, Owner ID: S19331243151) and Inbound rules (Filter rules: sg-049f0d9e95ceeb6d1 [launch-wizard-2]).

Step 11: Got to Security section to edit inbound rules

This screenshot is identical to the one above, showing the AWS EC2 Instances page. The 'Security' tab is now selected for the 'next.js' instance. The security details show the IAM Role is empty and the Owner ID is S19331243151. The Inbound rules section shows a single rule: sg-049f0d9e95ceeb6d1 [launch-wizard-2]. The rest of the interface, including the sidebar and other instance details, remains the same.

Step 12: Click on edit inbound rules

The screenshot shows the AWS EC2 Security Group configuration page for a group named 'launch-wizard-2'. The 'Inbound rules' tab is selected, displaying three existing rules:

Name	Security group rule ID	Type	Protocol	Port range
-	sgr-094b5dbd5554a7bc9	HTTP	IPv4	80
-	sgr-0208104e2658c5ffd	SSH	TCP	22
-	sgr-0700c58ae55e62ee0	HTTPS	TCP	443

A blue arrow points to the 'Edit inbound rules' button at the top right of the table.

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

The screenshot shows the 'ModifyInboundSecurityGroupRules' configuration page. A new rule is being added at the bottom of the list:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-094b5dbd5554a7bc9	HTTP	TCP	80	Custom	0.0.0.0/0
sgr-0208104e2658c5ffd	SSH	TCP	22	Custom	0.0.0.0/0
sgr-0700c58ae55e62ee0	HTTPS	TCP	443	Custom	0.0.0.0/0
-	Custom TCP	TCP	3000	Anyw...	0.0.0.0/0

A blue arrow points to the 'Add rule' button at the bottom left. A warning message at the bottom states: '⚠ 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.' The 'Save rules' button is highlighted in orange at the bottom right.

Inbound security group successfully modified.

The screenshot displays two separate browser tabs within a CloudShell session.

Top Tab (EC2 Security Groups):

- URL: https://us-west-1.console.aws.amazon.com/ec2/home?region=us-west-1#SecurityGroup:group_id=sg-049f0d9e93ceeb6d1
- Content: Shows a success message "Inbound security group rules successfully modified on security group (sg-049f0d9e93ceeb6d1) [launch-wizard-2]".
- Details: Security group name is "launch-wizard-2", ID is "sg-049f0d9e93ceeb6d1", created on "2024-06-25T03:57:45.732Z". Owner is "519331245151". Inbound rules count is 4, Outbound rules count is 1.
- Bottom navigation: Inbound rules (4), Outbound rules, Tags.

Bottom Tab (EC2 Instances):

- URL: <https://us-west-1.console.aws.amazon.com/ec2/home?region=us-west-1#Instances:instanceState=running:sort>tagName>
- Content: Shows a table of instances with one entry: "next.js" (Instance ID: i-016754fc1950e8a36). Status: Running. Type: t2.micro. Status check: 2/2 checks passed. Alarm status: View alarms. Availability Zone: us-west-1b. Public IP: ec2-3-101-111-113.us-west-1.compute.amazonaws.com.
- Bottom navigation: Details, Status and alarms, Monitoring, Security, Networking, Storage, Tags.

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.

```

[D:\] ec2-user@ip-172-31-29-234:/fullstack-assignment
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lenovo>chmod 400 "next.js.pem"
'chmod' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\lenovo>ssh -i ~/Downloads/next.js.pem ec2-user@54.219.186.26
The authenticity of host '54.219.186.26 (54.219.186.26)' can't be established.
ED25519 key fingerprint is SHA256:pSLcIfzId2H0h6H/eF4fW8mPuk1snOZSyj3lhvyxCM.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '54.219.186.26' (ED25519) to the list of known hosts.

      _###_
     ~ \###\
     ~ \##|
     ~ \#|
     ~ \_V_-'-->
     ~ \_/_/ \
     ~ \_/_/
     ~ \_/_/ \
Last login: Tue Jun 25 14:32:30 2024 from 13.52.6.116
[ec2-user@ip-172-31-29-234 ~]$ sudo yum update -y
Last metadata expiration check: 10:56:41 ago on Tue Jun 25 04:04:07 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-29-234 ~]$ sudo yum install -y docker git
Last metadata expiration check: 10:56:56 ago on Tue Jun 25 04:04:07 2024.
Package docker-23.0.3-1.amzn2023.0.1.x86_64 is already installed.
Package git-2.40.1-1.amzn2023.0.3.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-29-234 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-29-234 ~]$ start docker.service
-bash: start: command not found
[ec2-user@ip-172-31-29-234 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-29-234 ~]$ start docker.service
-bash: start: command not found
[ec2-user@ip-172-31-29-234 ~]$ sudo usermod -a -G docker ec2-user
[ec2-user@ip-172-31-29-234 ~]$ sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)"
Warning: Binary output can mess up your terminal. Use "--output" to tell
Warning: curl to output it to your terminal anyway, or use "--output"
Warning: <FILE>" to save to a file.

```

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

```

[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
Warning: Binary output can mess up your terminal. Use "--output " to tell
curl to output it to your terminal anyway, or consider "--output
warning: curl --output > /dev/null -z 234 -j -l sudo chmod +x /usr/local/bin/docker-compose
chmod: cannot access '/usr/local/bin/docker-compose': No such file or directory
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[bash: cd /path/to/your/docker-compose: No such file or directory
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[j] ls -l /usr/local/bin/docker-compose
ls: cannot access '/usr/local/bin/docker-compose': No such file or directory
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
ls: cannot access '/users/lenovo/bin/docker-compose': No such file or directory
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
ls: cannot access '/users/lenovo/bin/docker-compose': No such file or directory
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
curl: (35) error:1E0B0000:SSL routines:SSL3_GET_SERVER_CERTIFICATE:peer certificate or
[fatal: destination path '/fullstack-assignment' already exists and is not an empty directory.
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[bash: cd fullstack-assignment: No such file or directory
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[j] cd frontend
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[nano dockerfile , then ctrl+x then y
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[nano index.html
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[nano dockerfile
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[j] cd ..
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[nano dockerfile
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[j] cd ..
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[nano dockerfile
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[j] cd ..
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[nano dockerfile
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[j] nano docker-compose.yml
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[sudo yum update -y
Last metadata expiration check: 12:04:26 ago on Tue Jun 25 04:04:07 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-29-234:~/fullstack-assignment]
[sudo yum install -y libcrypt-compat
Last metadata expiration check: 12:04:43 ago on Tue Jun 25 04:04:07 2024.
Dependencies resolved.
=====
Repository           Size
=====
libcrypt-compat     x86_64      4.4.33-7.amzn2023          amazonlinux      92 k
=====
Transaction Summary
=====
Install 1 Package
=====

```

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

```

[ec2-user@ip-172-31-29-234 ~]$ sudo yum install -y libcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm
Transaction Summary
Install 1 Package
Total download size: 92 k
Installed size: 198 k
Downloading Packages:
libcrypt-compat-4.4.33-7.amzn2023.x86_64.rpm           1.2 MB/s | 92 kB   00:00
Total
711
[kB/s] | 92 kB  00:00
Running transaction check
Transaction check succeeded.
Performing transaction test
Transaction test succeeded.
Running transaction
Preparing : 1/1
Installing  : libcrypt-compat-4.4.33-7.amzn2023.x86_64 1/1
Running scriptlet: libcrypt-compat-4.4.33-7.amzn2023.x86_64 1/1
Verifying   : libcrypt-compat-4.4.33-7.amzn2023.x86_64 1/1
Installed:
libcrypt-compat-4.4.33-7.amzn2023.x86_64

Complete!
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ ls -l /var/run/docker.sock
srw-rw----. 1 root docker 0 Jun 25 14:44 /var/run/docker.sock
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo chmod 666 /var/run/docker.sock
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ docker-compose up --build -d
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ docker ps
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ cd frontend
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ nano dockerfile
[ec2-user@ip-172-31-29-234 frontend]$ cd backend
[bash: cd: backend: No such file or directory]
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ cd ..
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ cd ./fullstack-assignment/backend
[ec2-user@ip-172-31-29-234 backend]$ nano dockerfile
[ec2-user@ip-172-31-29-234 backend]$ cd ./fullstack-assignment
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ nano docker-compose.yml
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo yum update -y
Last metadata expiration check: 12:20:50 ago on Tue Jun 25 04:04:07 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo yum install -y libcrypt-compat-4.4.33-7.amzn2023.x86_64
Last metadata expiration check: 12:21:12 ago on Tue Jun 25 04:04:07 2024.
Package libcrypt-compat-4.4.33-7.amzn2023.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Step 17: 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"

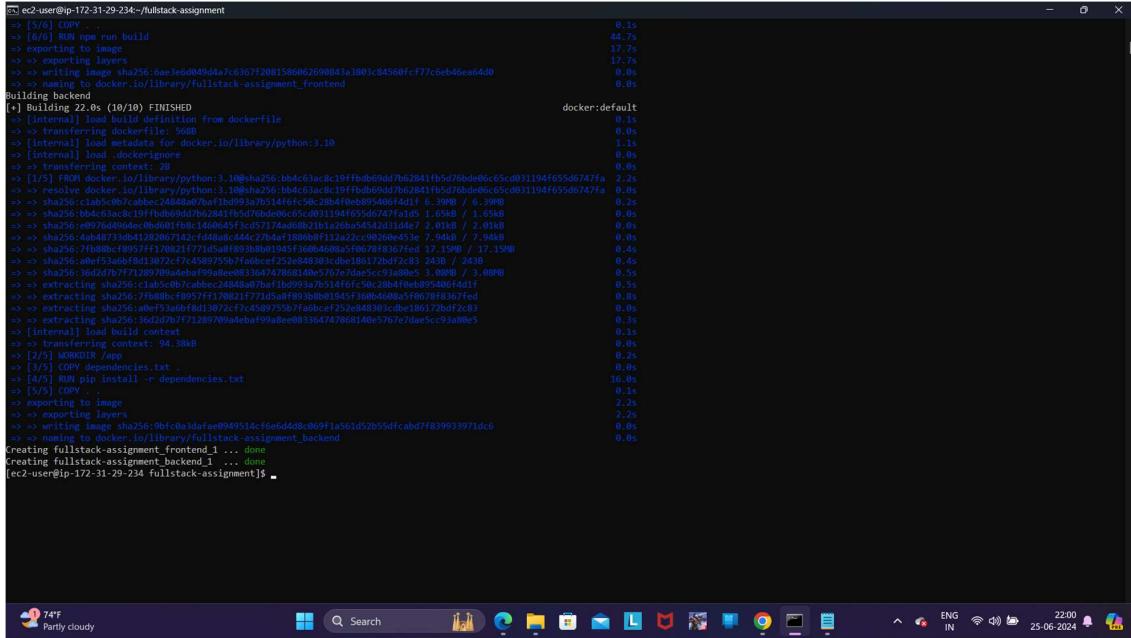
again, we should update the machine

- 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



```
ec2-user@ip-172-31-29-234:~/fullstack-assignment
>> [5/6] COPY . .
>> [6/6] FROM python:3.10-slim
>> exporting to image
>> >> exporting layers
>> >> writing image sha256:iaeue3edbd4da7c3c07f2081596082690843a3b01c84560ffcf77c6eb46ea64d0
>> >> naming to docker.io/library/fullstack-assignment_frontend
Building backend
[+] Building 22.0s (10/10) FINISHED
   > [internal] load build context from dockerfile          docker:default
   > [internal] load metadata for docker.io/library/python:3.10
   > [internal] load .dockerignore
   >> >> extracting context: 2B
   > [1/5] FROM python:3.10-slim
      >> [internal] load build context from dockerfile          docker:default
      >> [internal] load metadata for docker.io/library/python:3.10
      >> [internal] load .dockerignore
      >> >> extracting context: 2B
      >> sha256:bbdc61ac0c197fbdb0dd762841f5c570d9e9c65c201194f655d0727fa 2.3s
      >> sha256:c1ab5c007cabee7459e619514fcf5bc2084febe8940a3f4d1 6.399B / 6.399B 0.0s
      >> sha256:bbdc61ac0c197fbdb0dd762841f5c570d9e9c65c201194f655d0727fa 0.0s
      >> sha256:e9f764996d4e08fb10d001fb81460645f1cd5774a0d80211a26ba5452d31a4e7 2.01kB / 2.01kB 0.0s
      >> sha256:4a4048733d8d1292067142cf448c44a27f0da1f880f11a2a22c96265e53e 7.94kB / 7.94kB 0.0s
      >> sha256:7f1880c189577f13f08217723a5a81933b000194513134608a59067818367fed 17.19kB / 17.19kB 0.4s
      >> sha256:7f1880c189577f13f08217723a5a81933b000194513134608a59067818367fed 0.000B / 0.000B 0.0s
      >> sha256:bbdc61ac0c197fbdb0dd762841f5c570d9e9c65c201194f655d0727fa 3.099B / 3.099B 0.0s
      >> extracting sha256:c1ab5c007cabee7459e619514fcf5bc2084febe8940a3f4d1f 0.5s
      >> extracting sha256:7f1880c189577f13f0821771d5a0f893b0b01945f3601608a5f067818367fed 0.8s
      >> extracting sha256:aef53ebf58d13072cf7c45897550f7ad0ce252e848031cde186172bf2c83 0.0s
      >> [internal] load build context
      >> >> extracting context: 94.30kB 0.3s
      >> [4/5] WORKDIR /app 0.0s
      >> [5/5] COPY dependencies.txt .
      >> [4/5] RUN pip install --dependencies.txt 10.0s
      >> [5/5] COPY .
      >> >> extracting context: 2.23s
      >> >> extracting layers 2.23s
      >> >> writing image sha256:9bf4ca3d1ae0949514cf66d4d8c009f1a561d52b55dfcabdfff839933971dc6 0.0s
      >> >> naming to docker.io/library/fullstack-assignment_backend
Creating fullstack-assignment_frontend_1 ... done
Creating fullstack-assignment_backend_1 ... done
[ec2-user@ip-172-31-29-234 fullstack-assignment]$
```

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

```

[ec2-user@ip-172-31-29-234:/fullstack-assignment]
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo yum update -y
Last metadata expiration check: 12:48:47 ago on Tue Jun 25 04:04:07 2024.
Dependencies resolved.
Nothing to do.
[completed]
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ cat /etc/os-release
NAME="Amazon Linux"
VERSION="2023"
ID="amzn"
ID_LIKE="fedora"
VERSION_ID="2023"
PLATFORM_ID="platform:al2023"
PRETTY_NAME="Amazon Linux 2023.5.20240624"
ANSI_COLOR="0;33"
CPE_NAME="cpe://a-amazon-amazon-linux:2023"
HOME_URL="https://aws.amazon.com/amazon-linux-2023/"
DOCUMENTATION_URL="https://docs.aws.amazon.com/linux/"
SUPPORT_URL="https://aws.amazon.com/premiumsupport/"
BUG_REPORT_URL="https://github.com/amazonlinux/amazon-linux-2023"
VENDOR_NAME="AWS"
VENDOR_URL="https://aws.amazon.com/"
CPU_VENDOR="Amazon Linux 2023.5"
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo dnf update -y
Last metadata expiration check: 12:49:23 ago on Tue Jun 25 04:04:07 2024.
Dependencies resolved.
Nothing to do.
[completed]
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo dnf install nginx -y
Last metadata expiration check: 12:49:40 ago on Tue Jun 25 04:04:07 2024.
Package nginx-1:24.0.1.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
[ec2-user@ip-172-31-29-234 fullstack-assignment]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server, and stream
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
     Active: active (running) since Tue 2024-06-25 14:30:33 UTC; 2h 2min ago
       Main PID: 2138 (nginx)
          Tasks: 2 (limit: 1114)
        Memory: 3.3M
         CPU: 69ms
        CGroup: /system.slice/nginx.service
            ├─2138 "nginx: master process /usr/sbin/nginx"
            └─2140 "nginx: worker process"

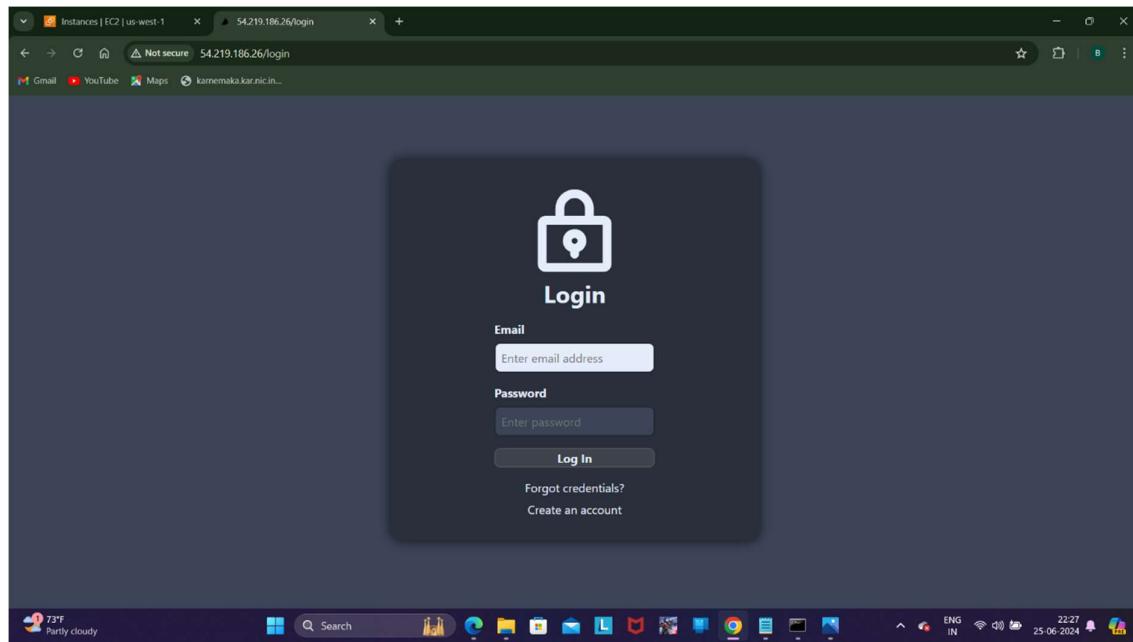
Jun 25 14:30:32 ip-172-31-29-234.us-west-1.compute.internal systemd[1]: Starting nginx.service - The nginx HTTP and reverse proxy server...
Jun 25 14:30:32 ip-172-31-29-234.us-west-1.compute.internal nginx[2138]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Jun 25 14:30:32 ip-172-31-29-234.us-west-1.compute.internal nginx[2138]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Jun 25 14:30:33 ip-172-31-29-234.us-west-1.compute.internal systemd[1]: Started nginx.service - The nginx HTTP and reverse proxy server.

[ec2-user@ip-172-31-29-234 fullstack-assignment]$

```

73°F Partly cloudy 22:26 ENG IN 25-06-2024

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 &&

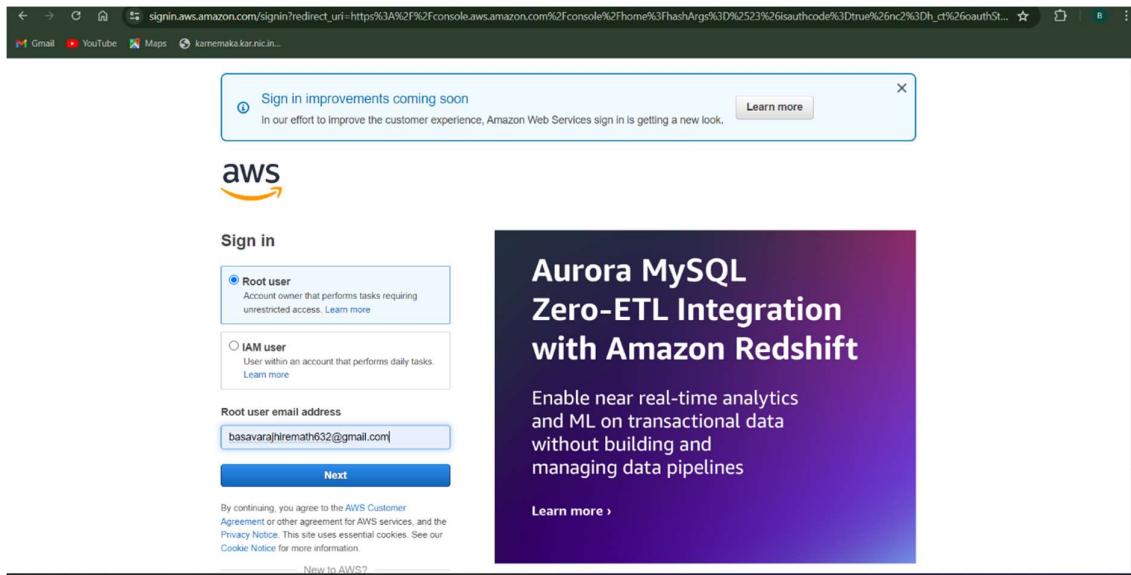
docker-compose up --build -d

Go to Secrets and click New repository secret .

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

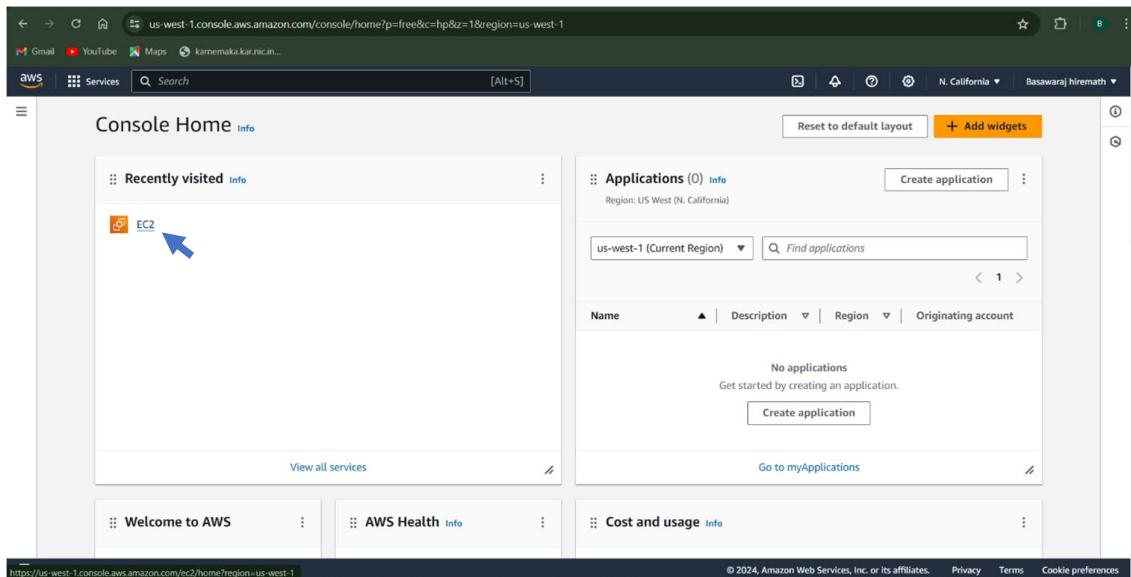
Steps to deploy Django on AWS Environment.

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

The screenshot shows the AWS EC2 Dashboard. On the left, there's a sidebar with navigation links like 'EC2 Dashboard', 'Instances', 'Images', and 'Elastic Block Store'. The main area is titled 'Resources' and displays various EC2 metrics: Instances (running) 0, Auto Scaling Groups 0, Dedicated Hosts 0, Elastic IPs 0, Instances 1, Key pairs 1, Load balancers 0, Placement groups 0, Security groups 2, Snapshots 0, and Volumes 1. Below this is a 'Launch instance' section with a large orange 'Launch instance' button. To the right of the launch button is a 'Service health' section showing 'US West (N. California)' and a status message 'This service is operating normally.' At the bottom right of the dashboard is an 'Additional information' section with links to 'Getting started guide', 'Documentation', 'All EC2 resources', 'Forums', 'Pricing', and 'Contact us'.

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

The screenshot shows the 'Launch an instance' wizard. On the left, there's a 'Quick Start' section with icons for Amazon Linux, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. Below this is a detailed view of the 'Amazon Linux 2023 AMI'. The AMI ID is 'ami-0124ee9682f3ad99' (64-bit (x86), uefi-preferred) / 'ami-01d0aefdf2e5358ac' (64-bit (Arm), uefi). The instance type is 't2.micro'. The summary on the right indicates 1 instance, the software image is 'Amazon Linux 2023 AMI 2023.5.20240624.0 x86_64 HVM kernel-6.1', and the storage is 1 volume(s) - 8 GiB. At the bottom right is a large orange 'Launch instance' button.

Instance Types:

- Virtual CPUs
- Memory

- Storage
- Network Performance

Step 5: Select instance type and create new key pair

The screenshot shows the AWS EC2 Launch Instances wizard at Step 3: Set Instance Details. In the 'Instance type' section, 't2.micro' is selected. The 'Key pair (login)' section has a dropdown set to 'Select' and a 'Create new key pair' button. The 'Network settings' section shows 'Network' tab selected. The 'Summary' section on the right shows 1 instance, using the 'Canonical, Ubuntu, 24.04 LTS' AMI, and contains a large 'Launch Instance' button.

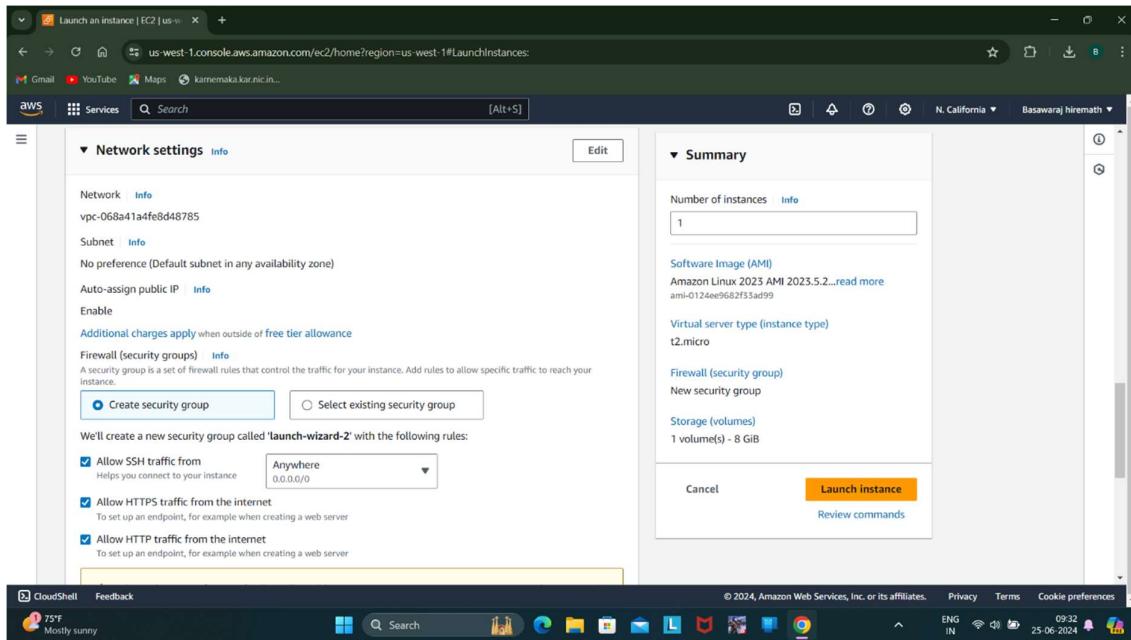
Step 6: Create new key pair

Step 7: Set Network settings as shown below

The screenshot shows the AWS EC2 Launch Instances wizard at Step 4: Set Network & Security. The 'Key pair (login)' section has a dropdown set to 'Select' and a 'Create new key pair' button. The 'Network settings' section shows 'Network' tab selected, with 'vpc-068a41a4fe8d48785' and 'Subnet' info. The 'Summary' section on the right shows 1 instance, using the 'Amazon Linux 2023 AMI 2023.5.2...' AMI, and contains a large 'Launch Instance' button.

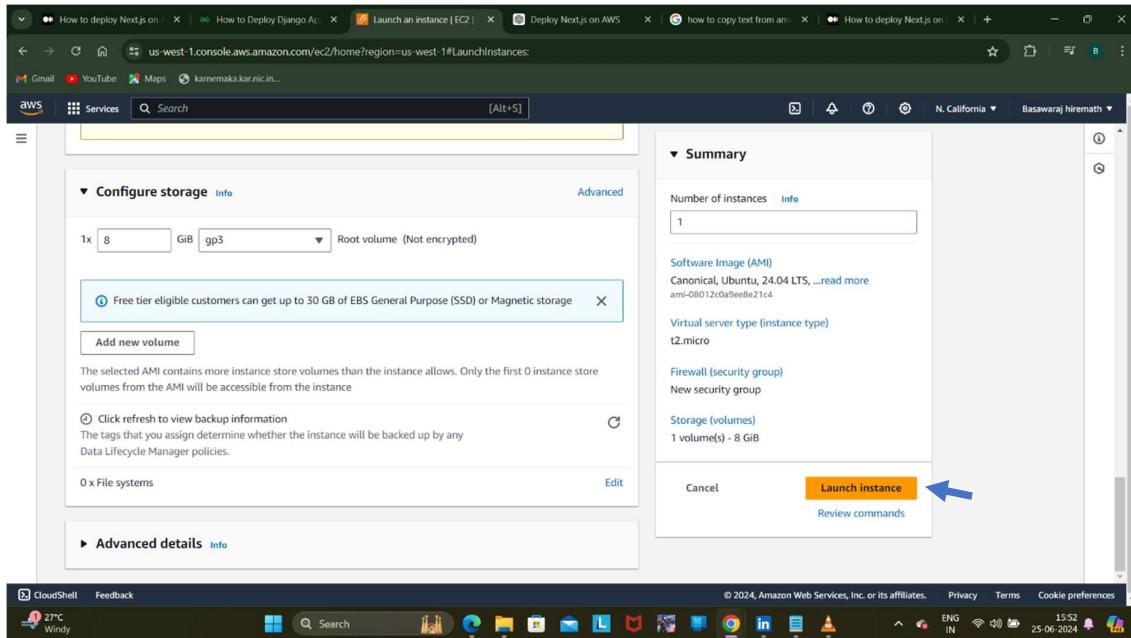
Step 8: Create new security group

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

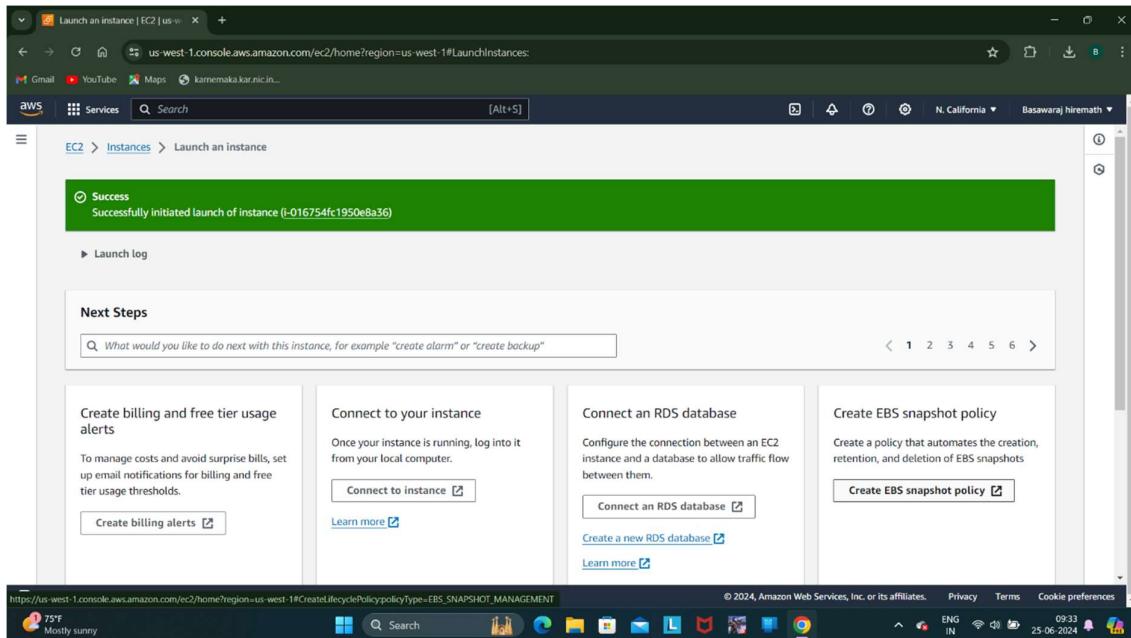


Step 9: Configure storage as shown below

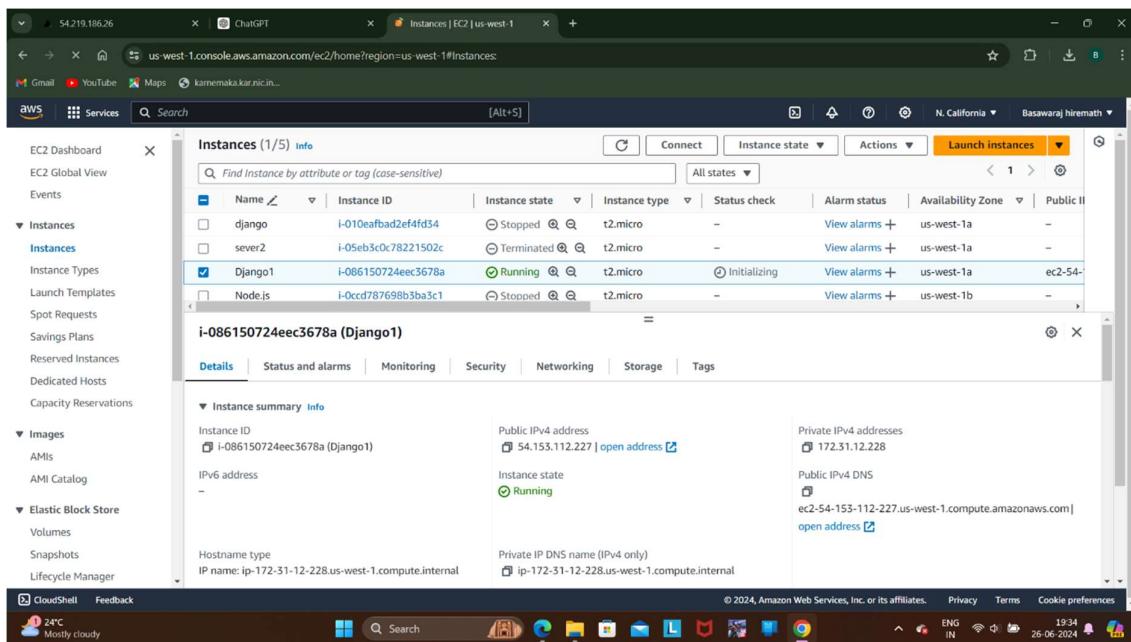
Step 10: Click on Launch instance



Instance have been successfully launched



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



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

Step 12: Click on edit inbound rules

The screenshot shows the AWS EC2 Security Group Inbound Rules page. The security group name is 'launch-wizard-2'. There are three inbound rules listed:

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-094b5dbd5554a7...	IPv4	HTTP	TCP	80
-	sgr-0208104e2658c5ff	IPv4	SSH	TCP	22
-	sgr-0700c58ae55e62eo	IPv4	HTTPS	TCP	443

An arrow points to the 'Edit inbound rules' button at the top right of the table.

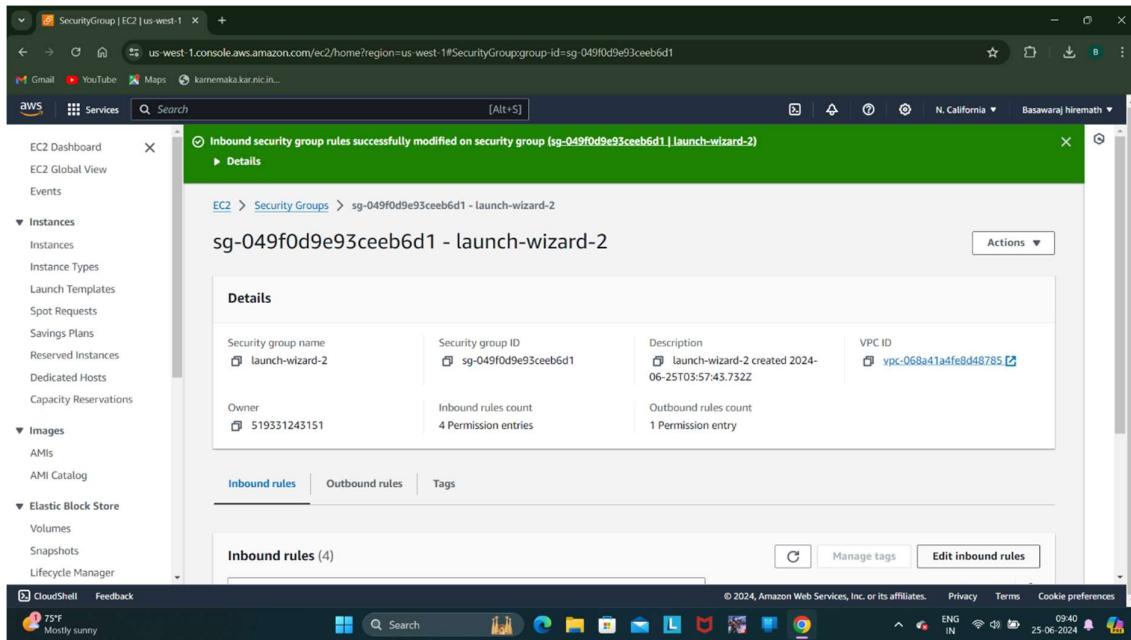
Step 13: Add new rule and Click on save rules.

The screenshot shows the AWS EC2 Security Group Inbound Rules page. The security group name is 'launch-wizard-5'. There are four inbound rules listed:

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-03c944601f5bc0df	IPv4	HTTPS	TCP	443
-	sgr-0e0480725fa7851da	IPv4	HTTP	TCP	80
-	sgr-033a73f2d86b84ac4	IPv4	SSH	TCP	22
-	sgr-05e1d81b4ac7d11...	IPv4	Custom TCP	TCP	8000

An arrow points to the 'Edit inbound rules' button at the top right of the table.

Inbound security group successfully modified.



Step 14: Connect the EC2 Instance

```
kgt@kgts-MacBook-Pro-2 desktop % chmod 400 "backend.pem"

kgt@kgts-MacBook-Pro-2 desktop % ssh -i "backend.pem" ec2-user@ec2-54-173-33-218.compute-1.amazonaws.com
Last login: Wed Jun 26 10:42:28 2024 from 139.167.42.18
[REDACTED]
```

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`

