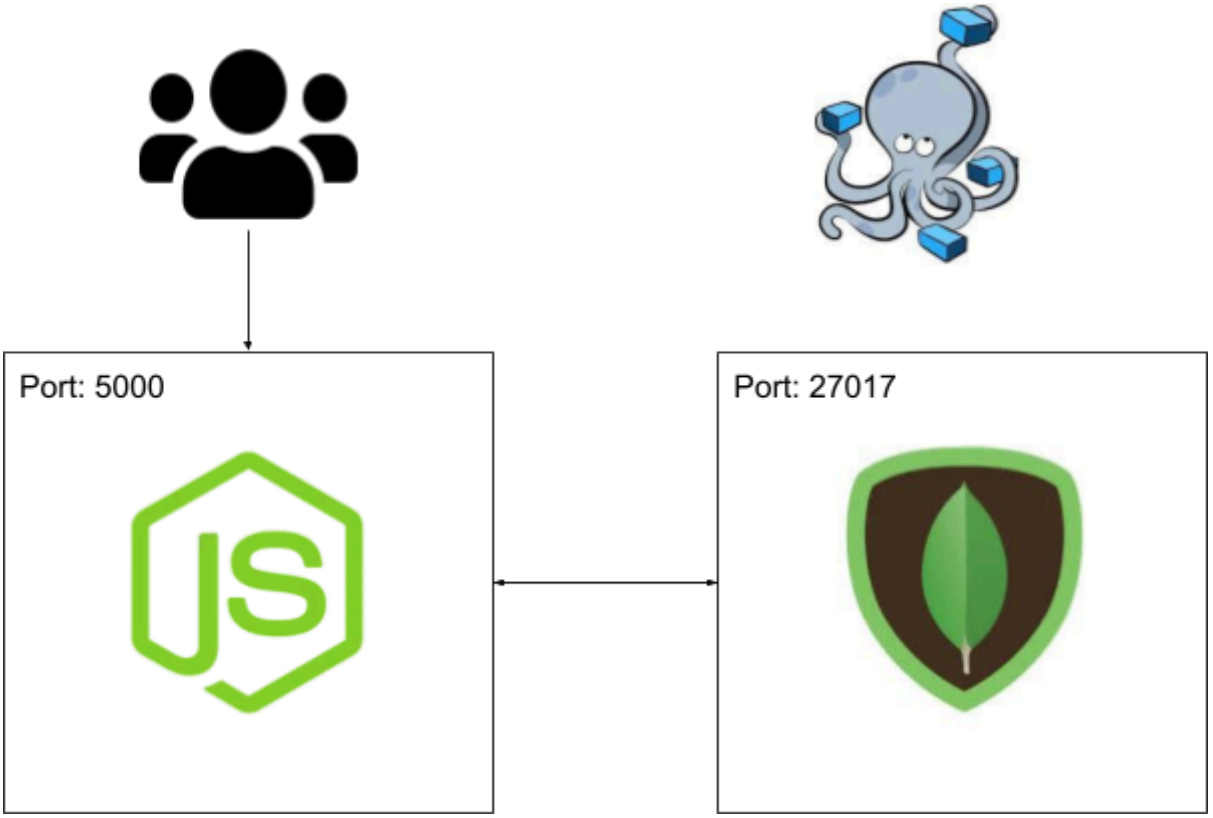
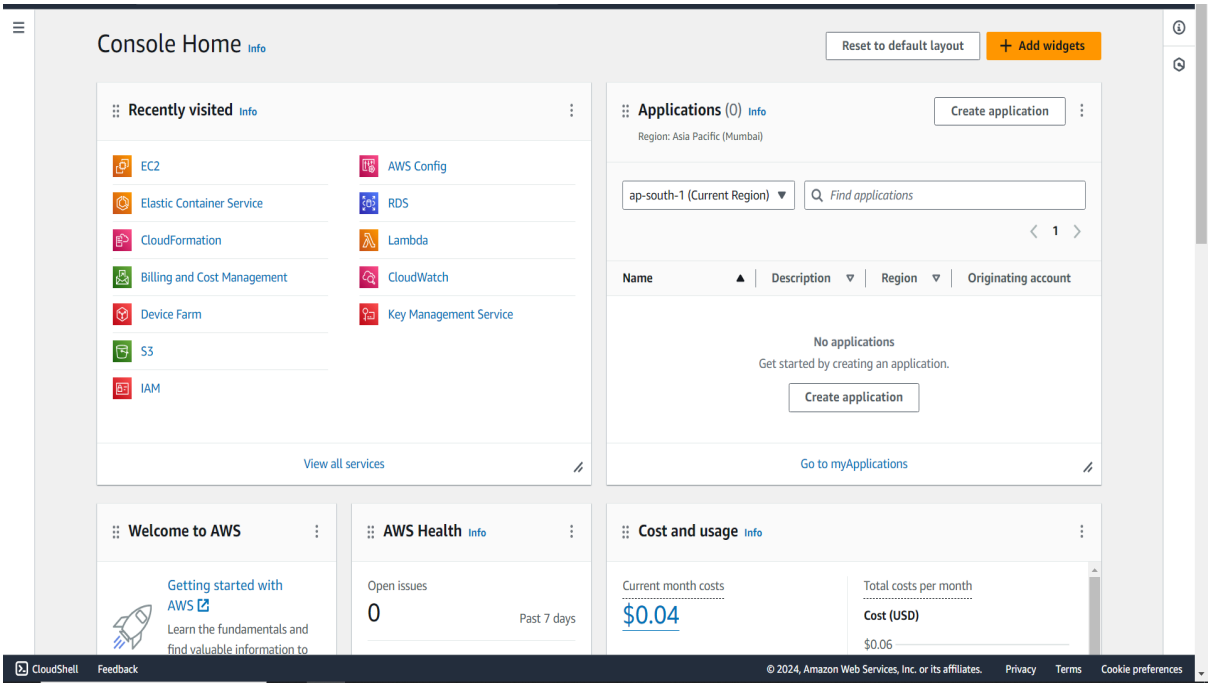
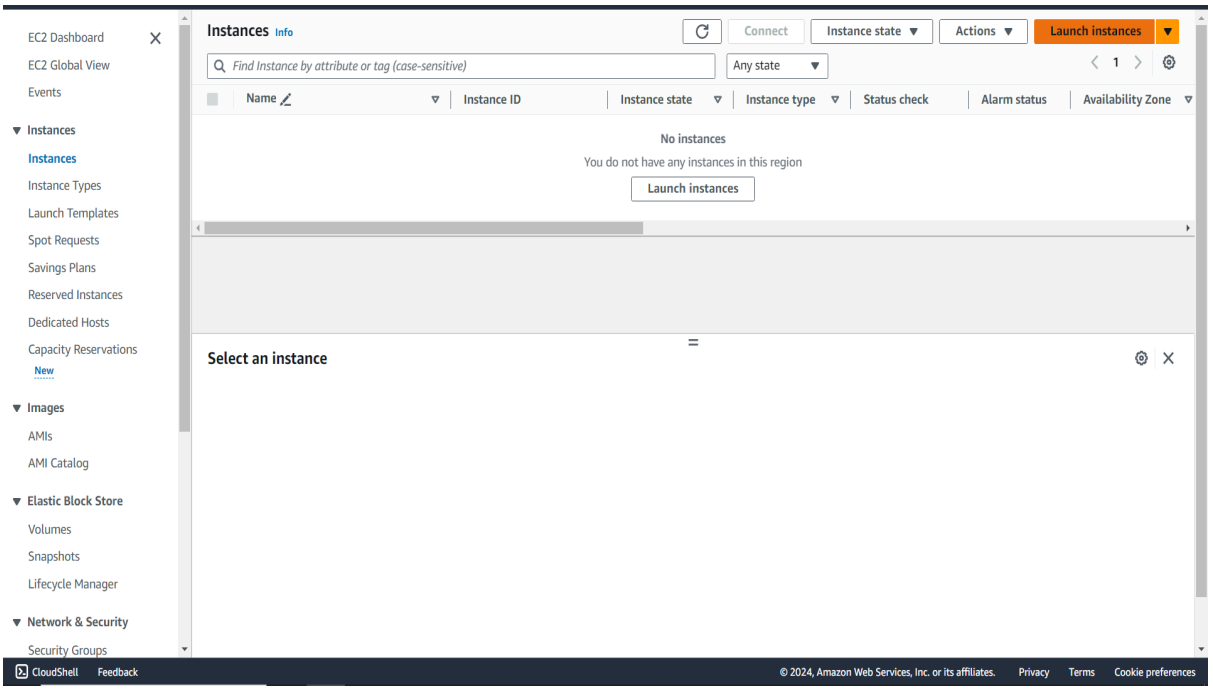
Create and run multi-container applications using Docker Compose



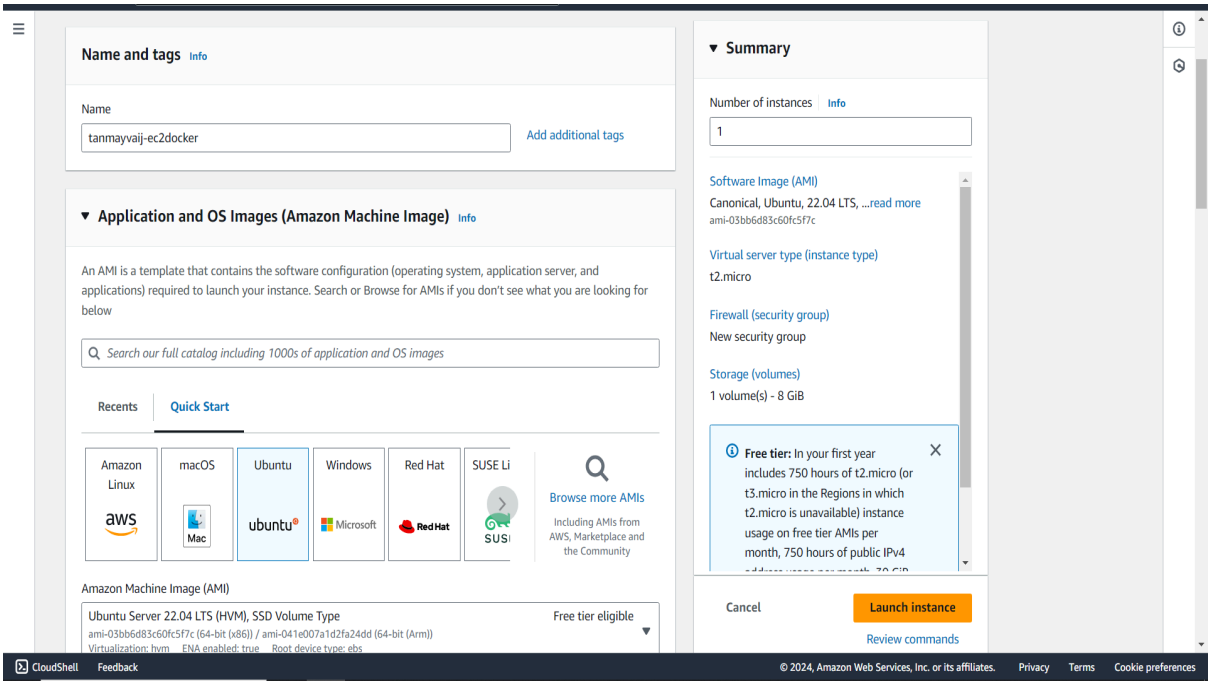
Step 1: Sign in to aws account



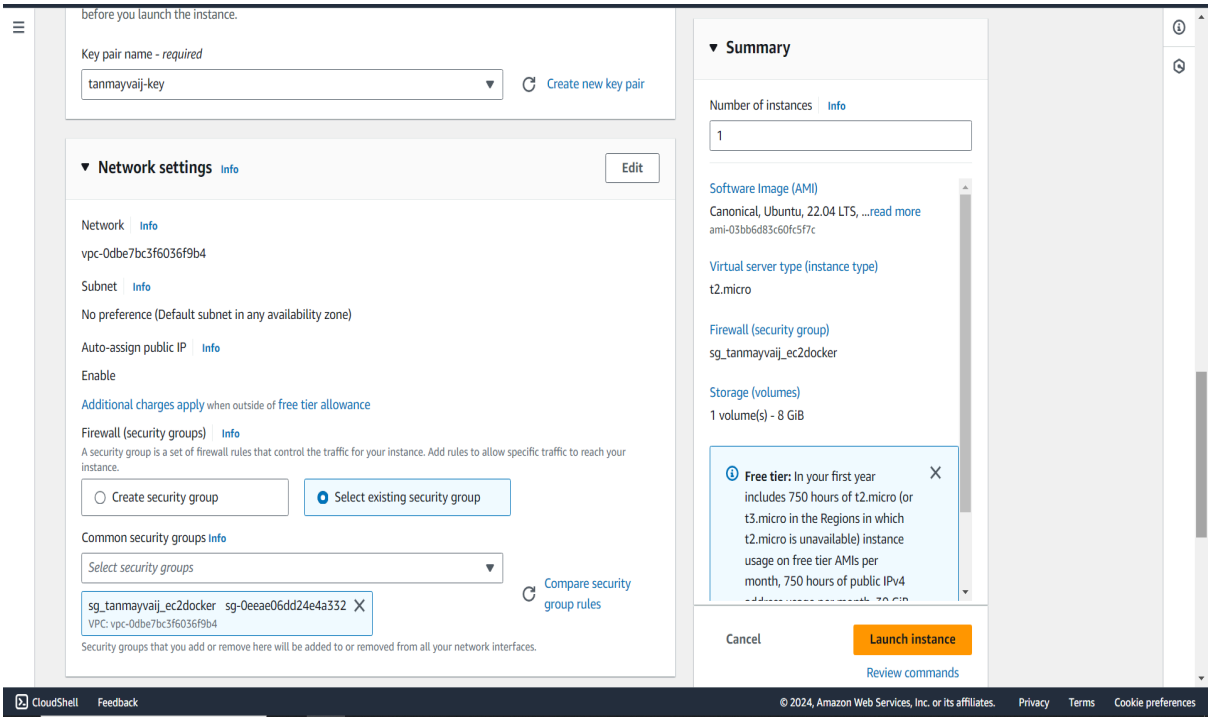
Step 2: Go to 'instances' in the EC2 dashboard and click on 'Launch Instances'



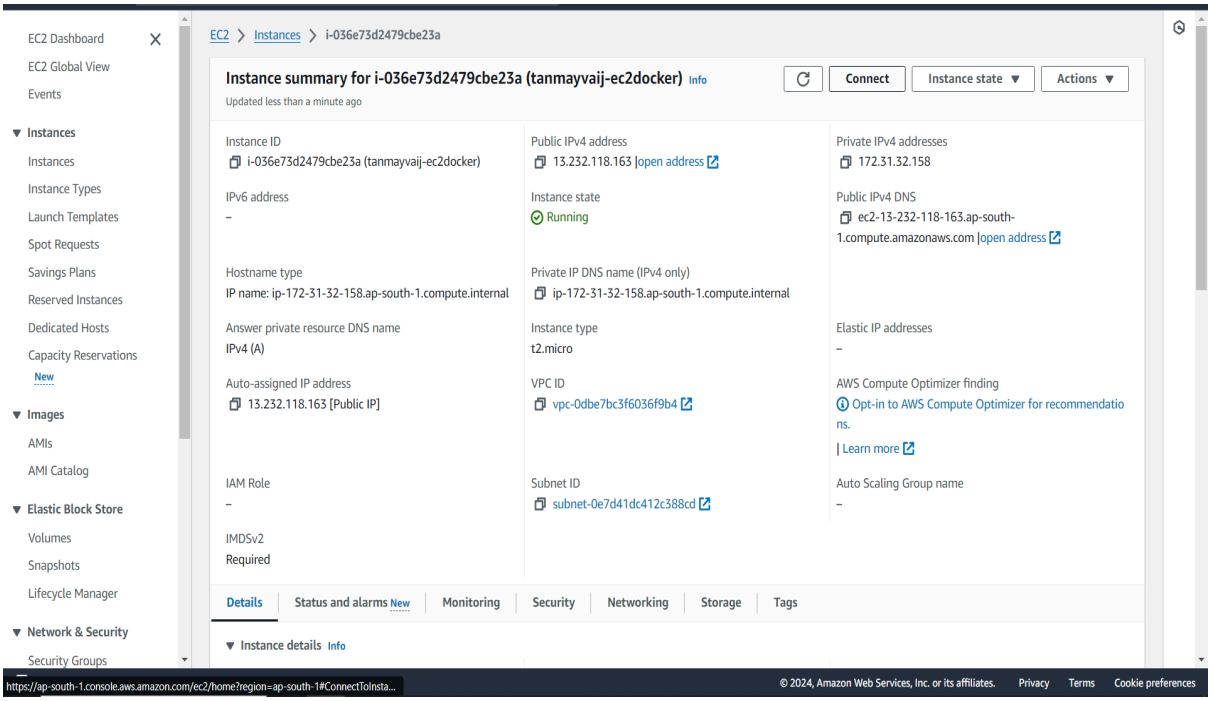
Step 3: Give a name to the instance and select ubuntu os



Step 4: Select key pair and security group and click on 'Launch Instance'



Step 5: Click on the created running instance and click on ‘Connect’

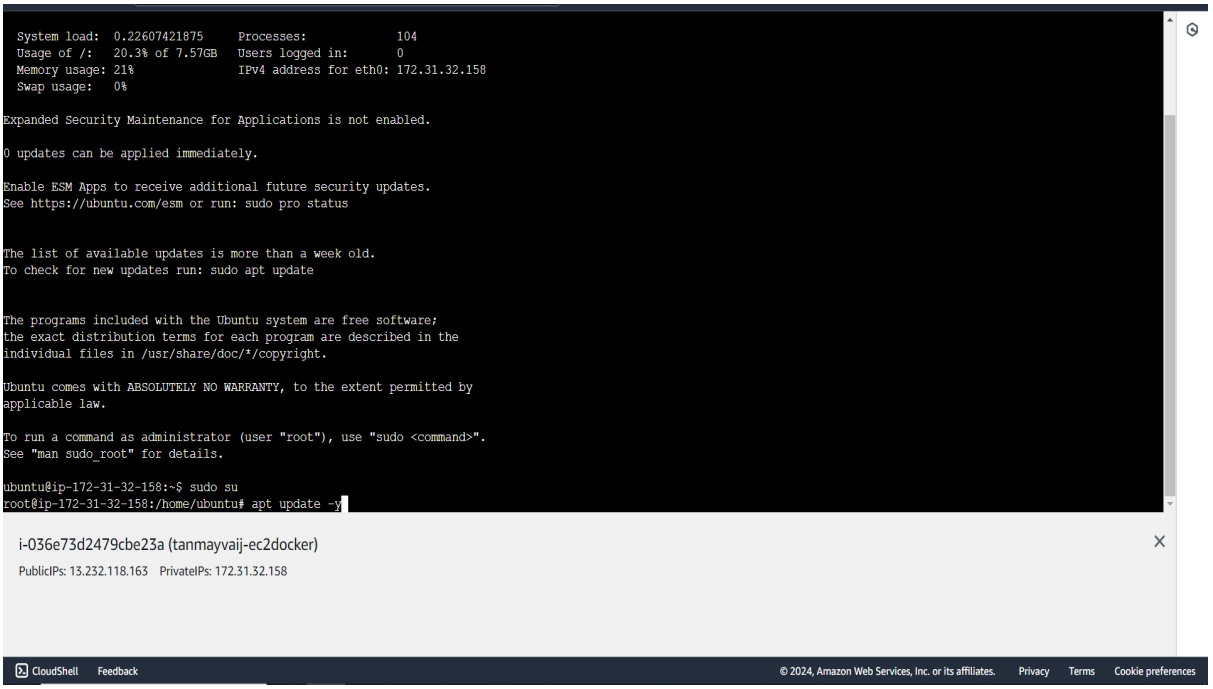


Step 6: Go to root mode and update the system

Commands:-

$ sudo su

$ apt update -y



Step 7: After entering the aws ec2 terminal, install docker on the instance using following command.

Command:- $ snap install docker

```
Get:15 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1055 kB]
Get:16 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [238 kB]
Get:17 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [22.1 kB]
Get:18 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [42.1 kB]
Get:19 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.1 kB]
Get:20 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [472 B]
Get:21 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [67.1 kB]
Get:22 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.0 kB]
Get:23 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [388 B]
Get:24 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [116 B]
Get:25 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [28.4 kB]
Get:26 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.6 kB]
Get:27 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [644 B]
Get:28 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1245 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [223 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1531 kB]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [255 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [849 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [162 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [16.8 kB]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.1 kB]
Get:37 http://security.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [7476 B]
Get:38 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [260 B]
Fetched 30.1 MB in 6s (5371 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
39 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-32-158:/home/ubuntu# snap install docker
```

i-036e73d2479cbe23a (tanmayvaij-ec2docker)

PublicIPs: 13.232.118.163 PrivateIPs: 172.31.32.158

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

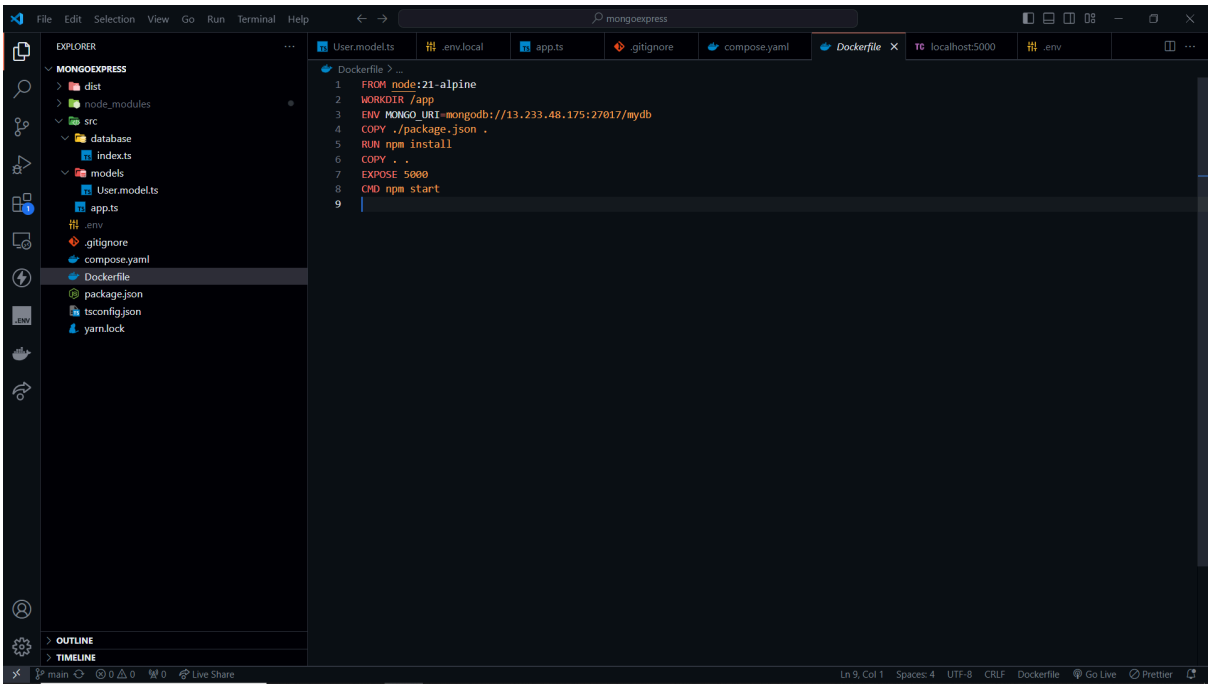
Step 8: Write a simple web server, for this example i am taking express as the web server and mongodb as database

Code used for this example:- https://github.com/tanmayvaij/express-mongodb-app

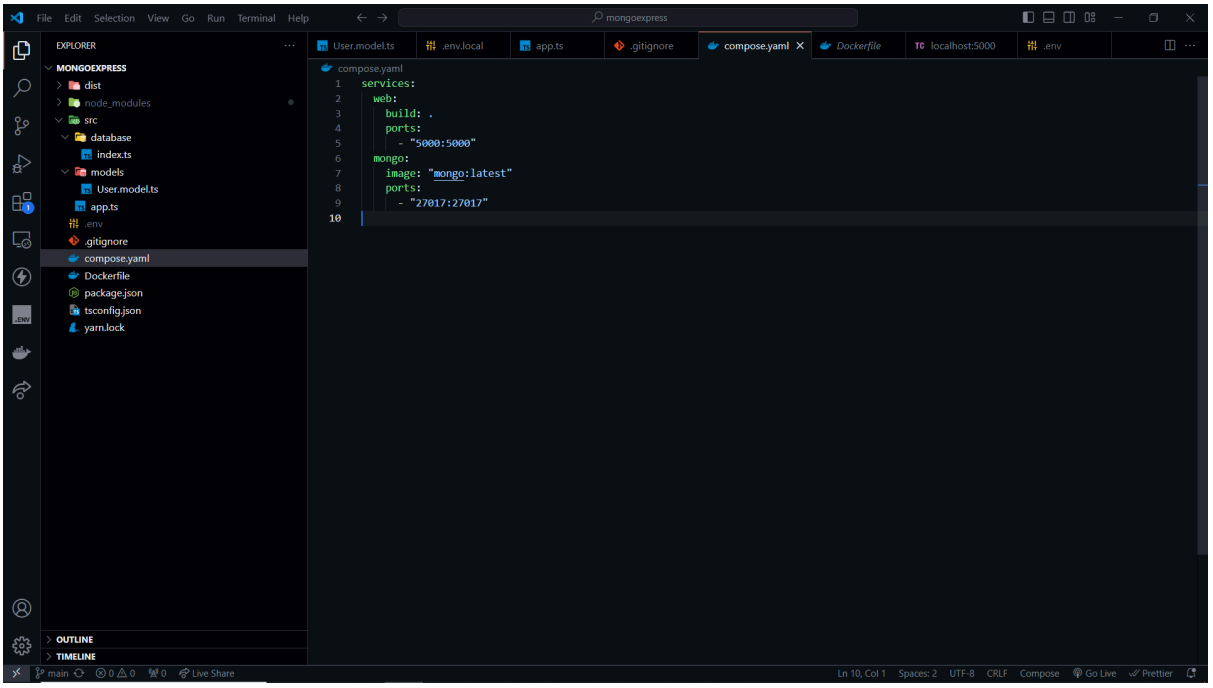
```
import cors from "cors";  5k (gzipped: 2.1k)
import { connectToDatabase } from "./database";
import { User } from "./models/User.model";

config();

const app = express();

app.use(cors());

app.get("/", async (req, res) => {

  try {

    const user = await User.findOne({ firstName: "Tanmay" });
    res.json(user);

  }
  catch (err) {

    console.log(err);

    res.json({
      err: "Error occurred",
    });
  }
});

app.listen(5000, async () => {
  connectToDatabase();

  await User.create({
    firstName: "Tanmay",
    lastName: "Vaij",
    email: "tanmayvaij22@gmail.com",
  });

  console.log("Server started successfully");
});
```

Step 9: Write a Dockerfile for it

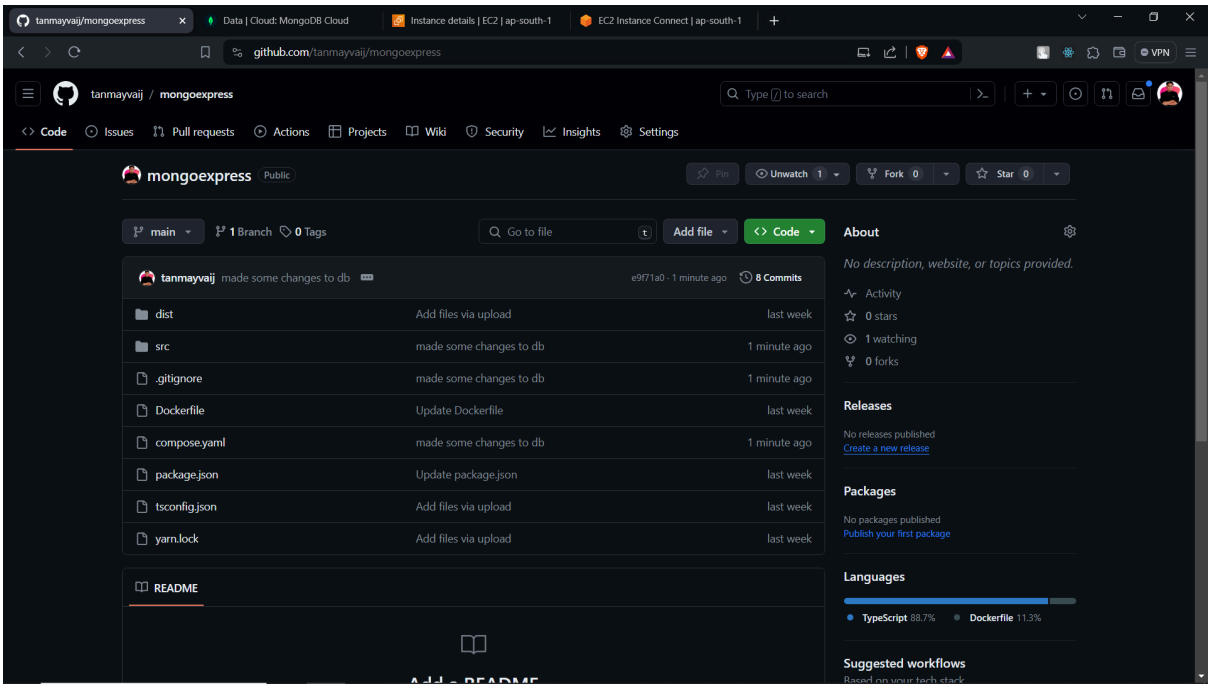
```
FROM node:21-alpine
WORKDIR /app
ENV MONGO_URI=mongodb://<public-url-of-ec2-instance>:27017/mydb
COPY ./package.json .
RUN npm install
COPY . .
EXPOSE 5000
CMD npm start
```



Step 10: Then write a compose.yaml file



Step 11: Push all the code to your github account



Step 12: Pull the github repo in the ec2 instance

```
root@ip-172-31-32-158:/home/ubuntu# git clone https://github.com/tanmayvaij/mongoexpress.git
Cloning into 'mongoexpress'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 41 (delta 14), reused 10 (delta 2), pack-reused 0
Receiving objects: 100% (41/41), 32.54 KiB | 4.07 MiB/s, done.
Resolving deltas: 100% (14/14), done.
root@ip-172-31-32-158:/home/ubuntu# cd ./mongoexpress/
root@ip-172-31-32-158:/home/ubuntu/mongoexpress# ls
Dockerfile  compose.yaml  dist  package.json  src  tsconfig.json  yarn.lock
root@ip-172-31-32-158:/home/ubuntu/mongoexpress#
```

i-036e73d2479cbe23a (tanmayvaij-ec2docker)

PublicIPs: 13.232.118.163 PrivateIPs: 172.31.32.158

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

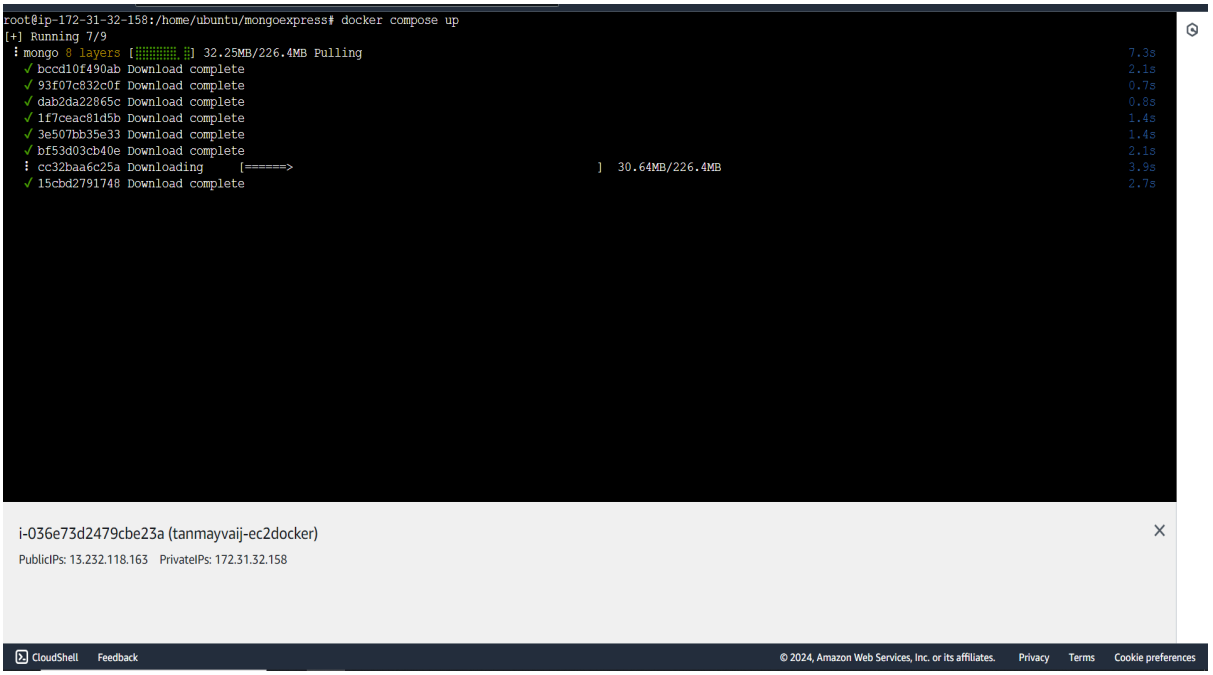
Step 13: Change the ip address of MONGO_URI in the dockerfile to the public ip of ec2

```
FROM node:21-alpine
WORKDIR /app
ENV MONGO_URI=mongodb://13.232.118.163:27017/mydb
COPY ./package.json .
RUN npm install
COPY . .
EXPOSE 5000
CMD npm start
```

```
create mode 100644 src/models/User.model.ts

tanma@DESKTOP-1UJOIBS MINGW64 /d/Projects/mongoexpress (main)
$ git push -u origin main
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 1.06 KiB | 1.06 MiB/s, done.
Total 8 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/tanmayvaij/mongoexpress.git
   29c0f9c..e9f71a0  main -> main
branch 'main' set up to track 'origin/main'.

tanma@DESKTOP-1UJOIBS MINGW64 /d/Projects/mongoexpress (main)
$ git add .
```

Step 14: Allow required ports (5000, 27017) in the security group and then start both containers with docker compose

Command:- docker compose up



Step 15: Now, verify the setup by visiting port 5000