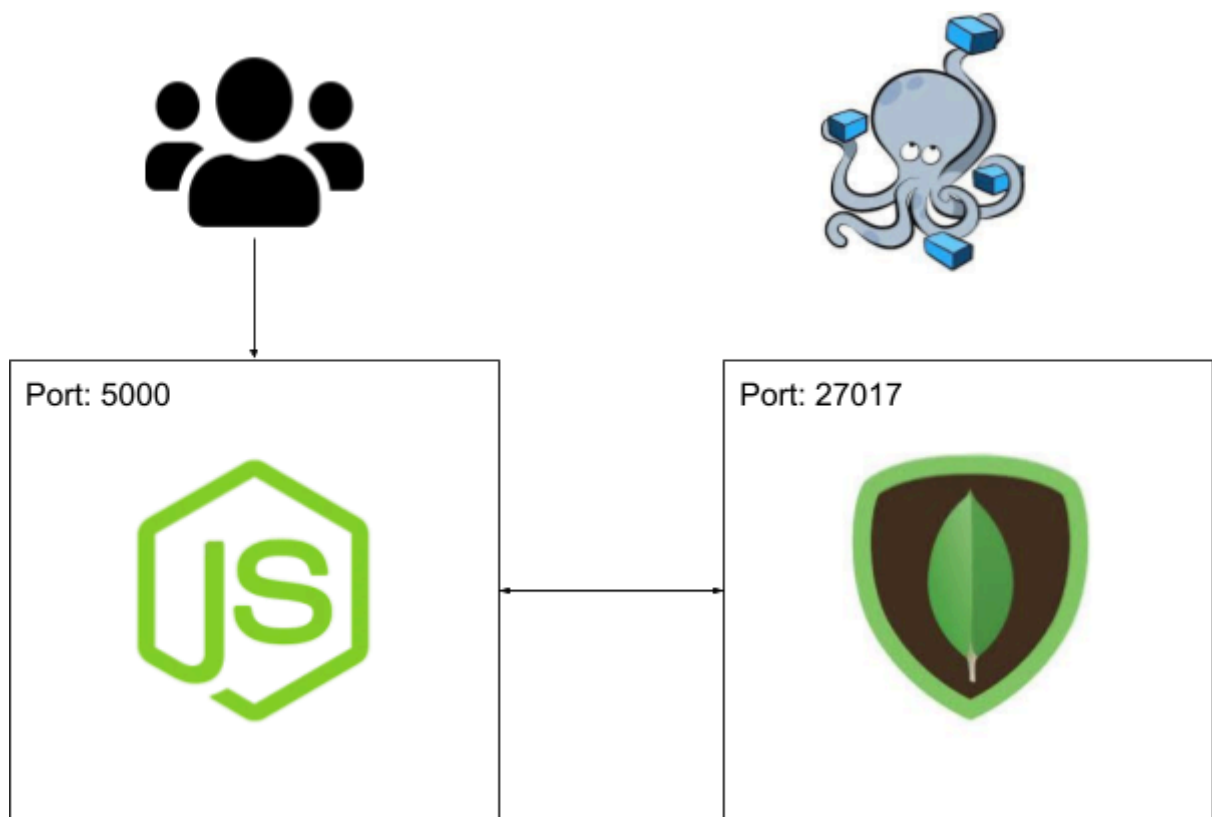
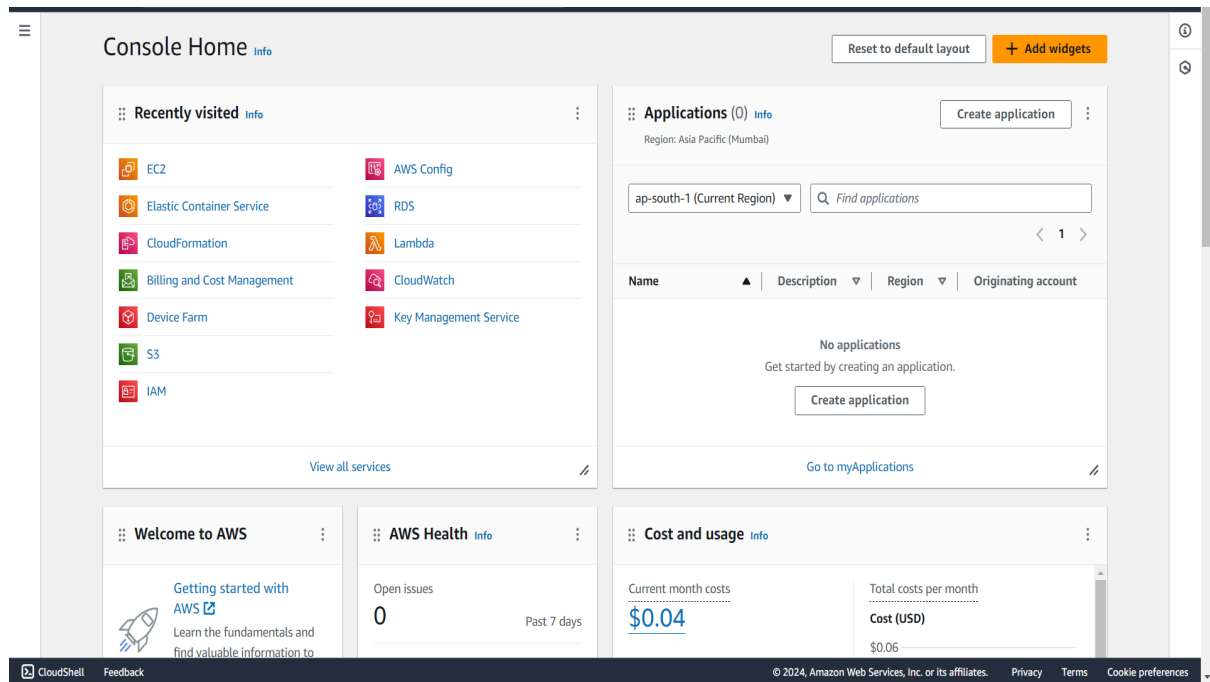


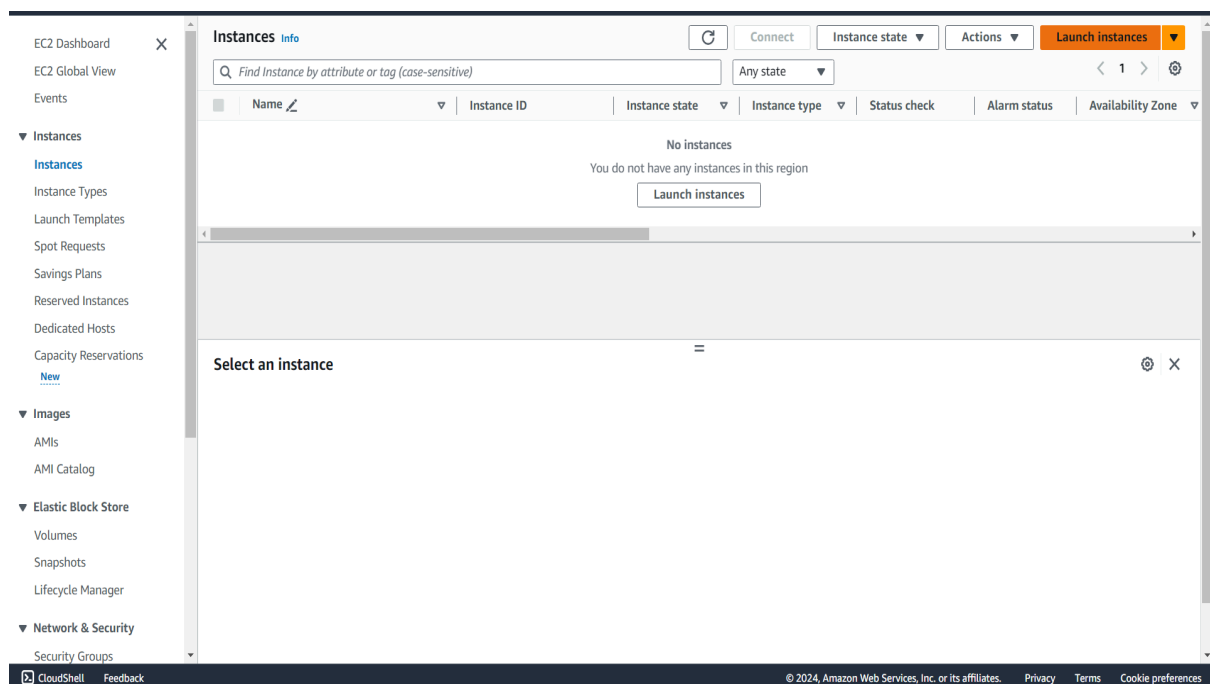
## Create and run multi-container applications using Docker Compose



## Step 1: Sign in to aws account



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



Name and tags [Info](#)

Name

[Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Recents

Quick Start

Amazon Linux  
aws

macOS

Ubuntu  
ubuntu®

Windows  
Microsoft

Red Hat  
Red Hat

SUSE Li  
SUSE

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-03bb6d83c60fc5f7c (64-bit (x86)) / ami-041e007a1d2fa24dd (64-bit (Arm))  
Virtualization: hvm    ENA enabled: true    Root device type: ebs

▼ Summary

Number of instances [Info](#)

Software Image (AMI)  
Canonical, Ubuntu, 22.04 LTS, ...read more  
ami-03bb6d83c60fc5f7c

Virtual server type (instance type)  
t2.micro

Firewall (security group)  
New security group

Storage (volumes)  
1 volume(s) - 8 GiB

❗ Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address per month, 30 GB of EBS storage per month.

Cancel

Launch instance

Review commands

before you launch the instance.

Key pair name - *required*  

tanmayvaij-key

Create new key pair

▼ Network settings

Info

Edit

Network 

Info

  
vpc-0dbe7bc3f6036f9b4  
  
Subnet 

Info

  
No preference (Default subnet in any availability zone)  
  
Auto-assign public IP 

Info

  
Enable  
  
Additional charges apply when outside of free tier allowance  
  
Firewall (security groups) 

Info

  
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.  

Create security group

Select existing security group

  
Common security groups 

Info

Select security groups

sg\_tanmayvaij\_ec2docker sg-0eeae06dd24e4a332 X  
VPC: vpc-0dbe7bc3f6036f9b4

Compare security group rules

  
Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Summary

Number of instances 

Info

1

  
Software Image (AMI)  
Canonical, Ubuntu, 22.04 LTS, ...[read more](#)  
ami-03bb6d83c60f5f7c  
  
Virtual server type (instance type)  
t2.micro  
  
Firewall (security group)  
sg\_tanmayvaij\_ec2docker  
  
Storage (volumes)  
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 750 hours of public IPv4 address space, 30 GiB of EBS storage, and 100,000 requests per month to Amazon S3.

Cancel 

Launch instance

Review commands

CloudShell Feedback

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

## Step 5: Click on the created running instance and click on 'Connect'

The screenshot shows the AWS Management Console interface. On the left is a navigation menu with categories like EC2 Dashboard, Instances, Images, Elastic Block Store, and Network & Security. The main area displays the 'Instance summary for i-036e73d2479cbe23a (tanmayvaj-ec2docker)'. The instance is in a 'Running' state. Key details include: Public IPv4 address 13.232.118.163, Private IPv4 address 172.31.32.158, Instance type t2.micro, and VPC ID vpc-0dbe7bc3f6036f9b4. A 'Connect' button is located in the top right of the summary card. Below the summary are tabs for Details, Status and alarms, Monitoring, Security, Networking, Storage, and Tags. The 'Details' tab is currently selected, showing instance details like IP name, hostname type, and DNS names.

## Step 6: Go to root mode and update the system

Commands:-

```
$ sudo su
```

```
$ apt update -y
```

The screenshot shows a terminal window with the following output:

```
System load: 0.22607421875   Processes:      104
Usage of /:  20.3% of 7.57GB   Users logged in: 0
Memory usage: 21%            IPv4 address for eth0: 172.31.32.158
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

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

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

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

ubuntu@ip-172-31-32-158:~$ sudo su
root@ip-172-31-32-158:/home/ubuntu# apt update -y
```

Below the terminal output, a small box shows the instance details for i-036e73d2479cbe23a (tanmayvaj-ec2docker), including Public IPs: 13.232.118.163 and Private IPs: 172.31.32.158.

## 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 (tanmayvaj-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/tanmayvaj/express-mongodb-app>

```
File Edit Selection View Go Run Terminal Help
src > app.ts > app.get("/") callback
3 import cors from "cors";
4 import { connectToDatabase } from "../database";
5 import { User } from "../models/User.model";
6
7 config();
8
9 const app = express();
10
11 app.use(cors());
12
13 app.get("/", async (req, res) => {
14   try {
15     const user = await User.findOne({ firstName: "Tanmay" });
16     res.json(user);
17   } catch (err) {
18     console.log(err);
19     res.json({
20       err: "Error occurred",
21     });
22   }
23 });
24
25 app.listen(5000, async () => {
26   connectToDatabase();
27
28   await User.create({
29     firstName: "Tanmay",
30     lastName: "Vaij",
31     email: "tanmayvaj22@gmail.com",
32   });
33   console.log("Server started successfully");
34 });
```

## Step 9: Write a Dockerfile for it

FROM node:21-alpine

WORKDIR /app

ENV MONGO\_URI=mongodb://<public-url-of-ec2-instance>:27017/mydb

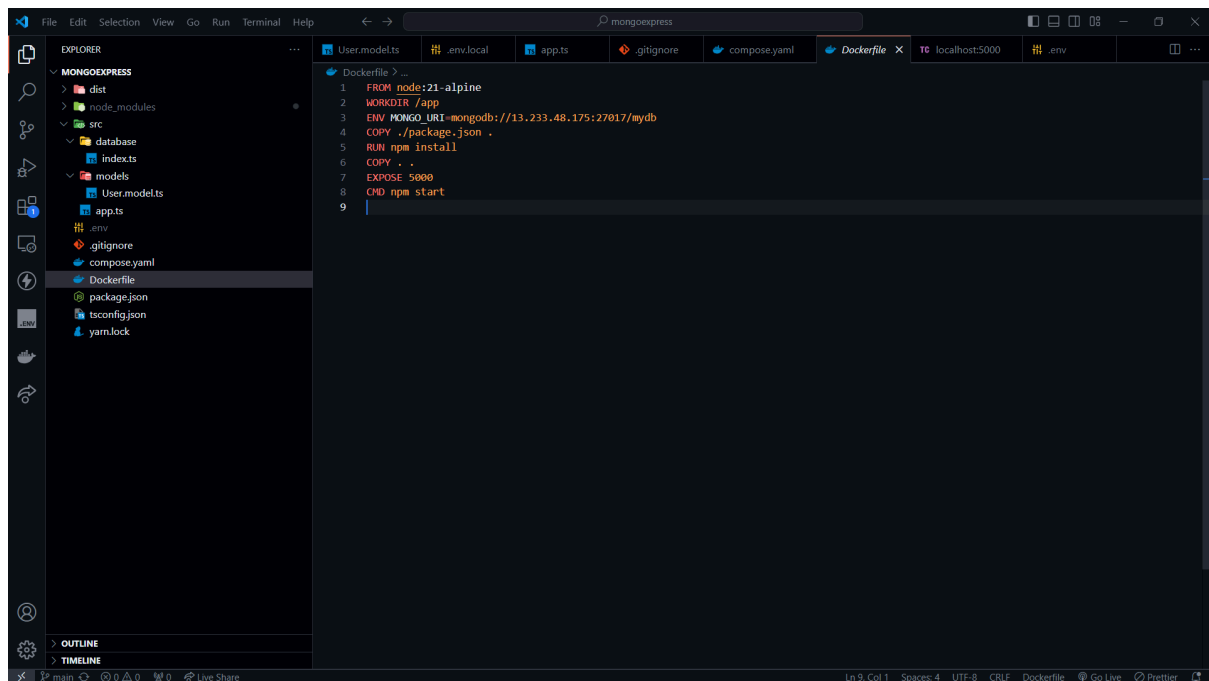
COPY ./package.json .

RUN npm install

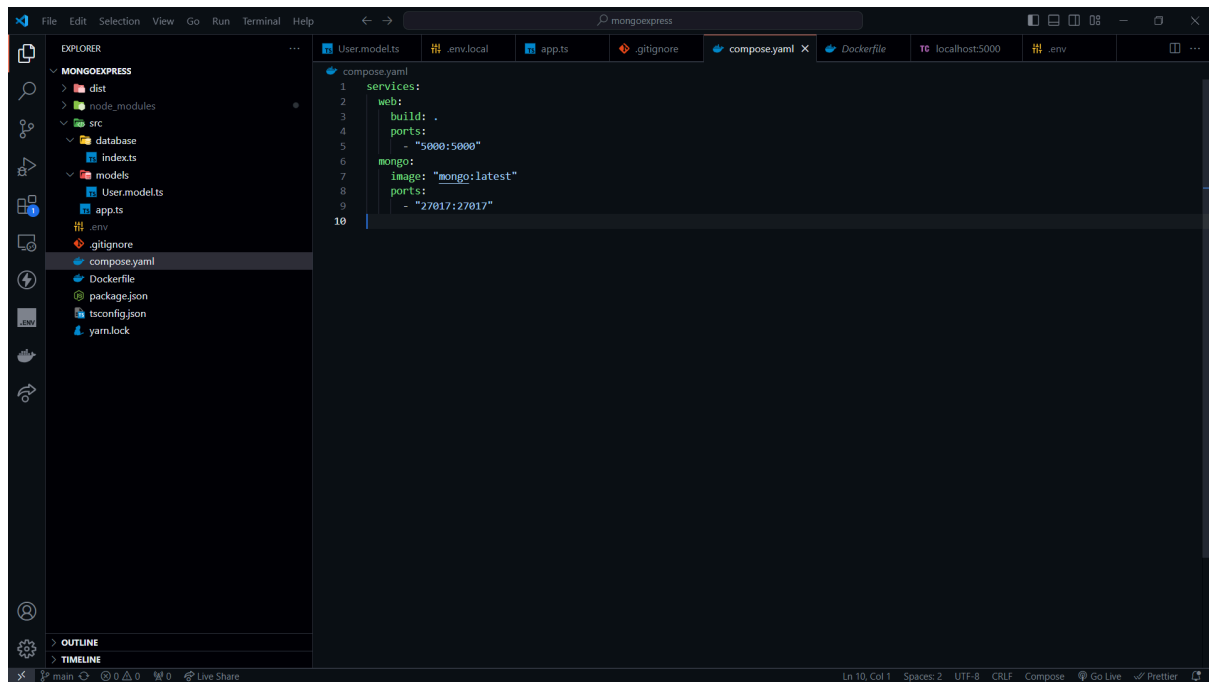
COPY . .

EXPOSE 5000

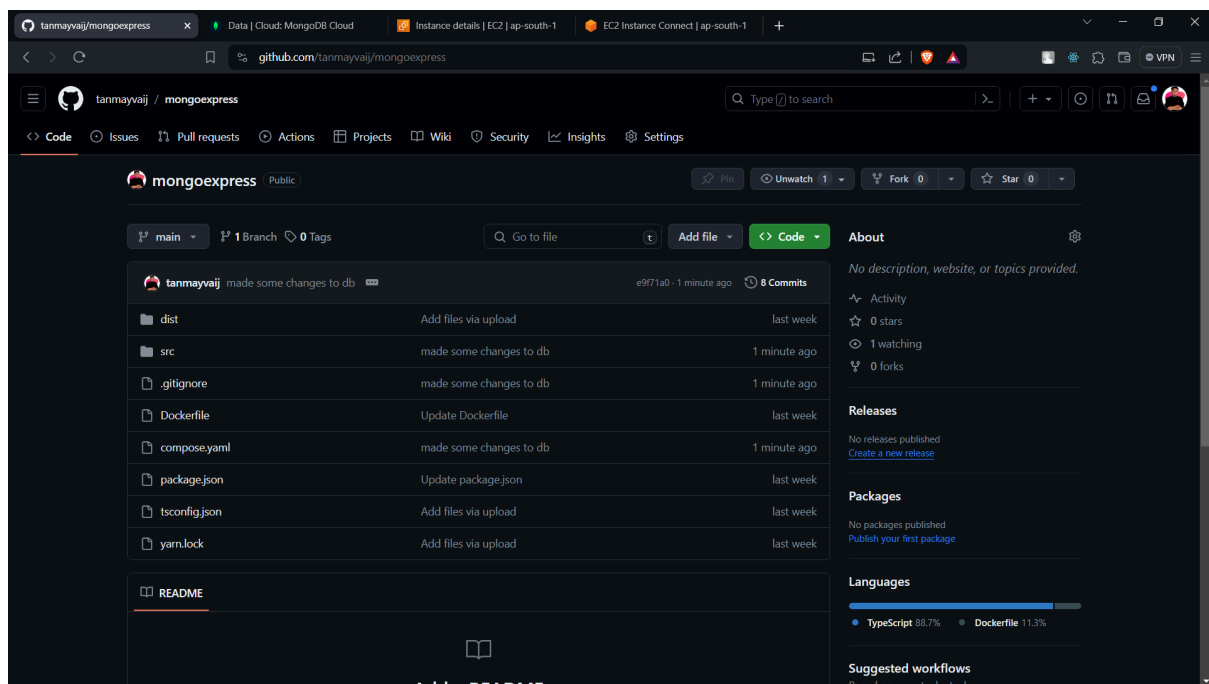
CMD npm start



## 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

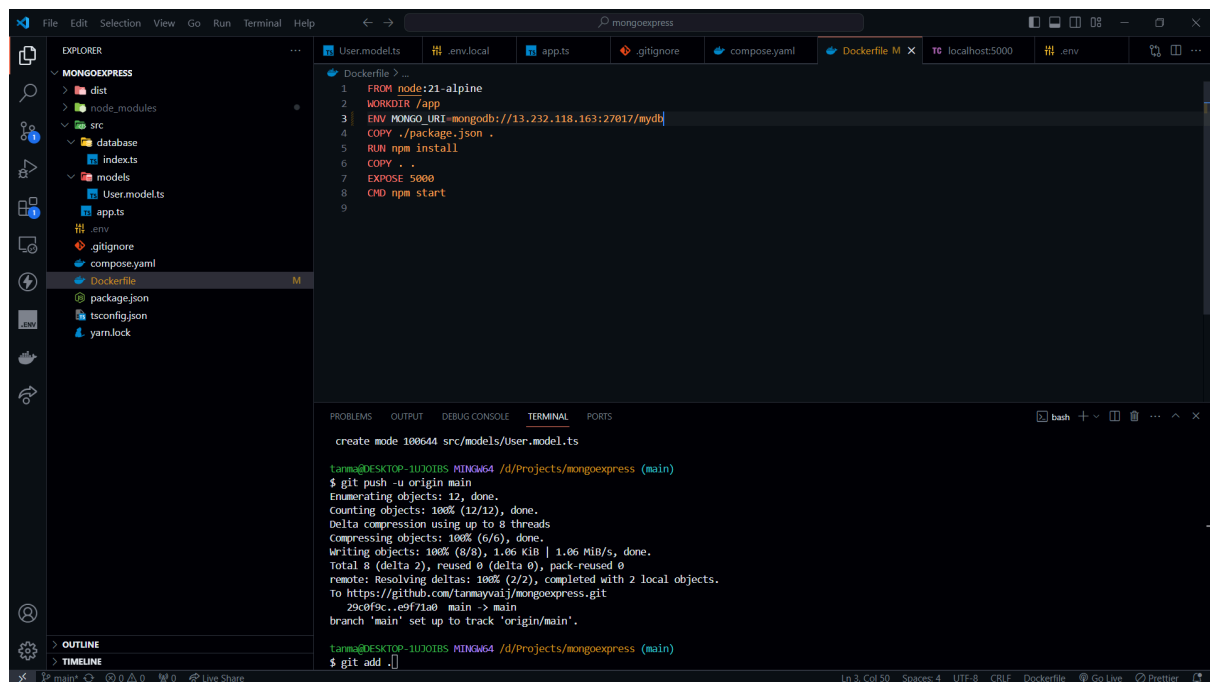
```
root@ip-172-31-32-158:/home/ubuntu# git clone https://github.com/tanmayvaij/mongoexpress.git
Cloning into 'mongoexpress'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 41 (delta 14), reused 10 (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



The screenshot shows the VS Code editor interface. The Explorer panel on the left displays the project structure for 'MONGOEXPRESS', including files like 'dist', 'node\_modules', 'src', 'database', 'indexes', 'models', 'User.model.ts', 'app.ts', '.env', '.gitignore', 'compose.yaml', 'Dockerfile', 'package.json', 'tsconfig.json', and 'yarn.lock'. The Dockerfile is open in the editor, showing the following content:

```
1 FROM node:21-alpine
2 WORKDIR /app
3 ENV MONGO_URI=mongodb://13.232.118.163:27017/mydb
4 COPY ./package.json .
5 RUN npm install
6 COPY . .
7 EXPOSE 5000
8 CMD npm start
```

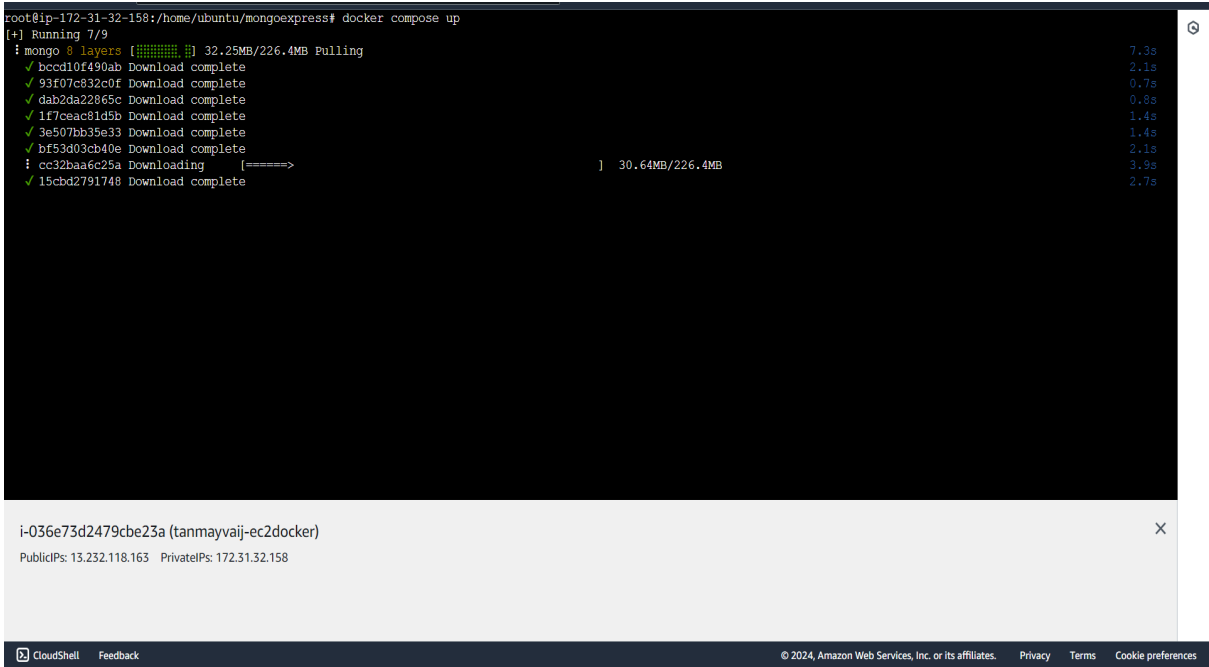
The terminal at the bottom shows the output of a git push command:

```
create mode 100644 src/models/User.model.ts
tanmay@DESKTOP-1UJ0I8S 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'.
tanmay@DESKTOP-1UJ0I8S 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

