**Introduction:**

The TravelMemory application has been developed using the MERN stack. Your challenge is to deploy this

application on an Amazon EC2 instance. This will provide you with hands-on experience in deploying full-stack

applications, working with cloud platforms, and ensuring scalable architecture.

Project Repository:

Access the complete codebase of the TravelMemory application from the provided GitHub

link: <https://github.com/UnpredictablePrashant/TravelMemory>

Objective: - Set up the backend running on Node.js. - Configure the front end designed with React. - Ensure efficient communication between the front end and back end. - Deploy the full application on an EC2 instance. - Facilitate load balancing by creating multiple instances of the application. - Connect a custom domain through Cloudflare.

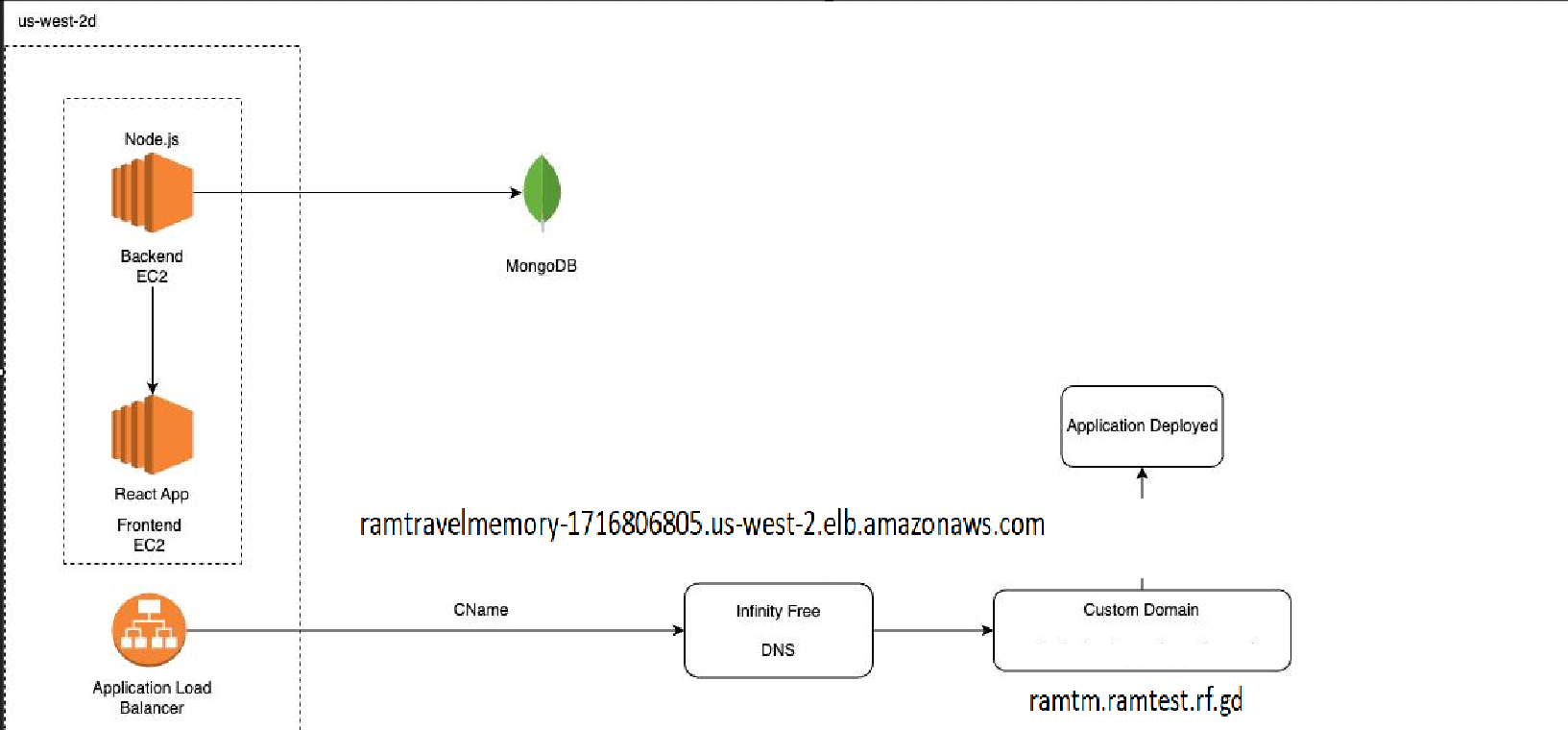
Tasks:

1. Backend Configuration: - Clone the repository and navigate to the backend directory. - The backend runs on port 3000. Set up a reverse proxy using nginx to ensure smooth deployment on EC2. - Update the .env file to incorporate database connection details and port information.

2. Frontend and Backend Connection: - Navigate to the `urls.js` in the frontend directory. - Update the file to ensure the front end communicates effectively with the backend.

3. Scaling the Application: - Create multiple instances of both the frontend and backend servers. - Add these instances to a load balancer to ensure efficient distribution of incoming traffic.

4. Domain Setup with Cloudflare: - Connect your custom domain to the application using Cloudflare. - Create a CNAME record pointing to the load balancer endpoint. - Set up an A record with the IP address of the EC2 instance hosting the front end.



Githuburl: <https://github.com/Ram495-ctrl/Travel-Memory-Application-Deployment.git>

1. Backend Configuration: - Clone the repository and navigate to the backend directory. - The backend runs on port 3000. Set up a reverse proxy using nginx to ensure smooth deployment

on EC2. - Update the .env file to incorporate database connection details and port information.

Created the EC2 Instances for both backend and frontend

In backend clone the repo using -> git clone <https://github.com/UnpredictablePrashant/TravelMemory>

Navigate to the path for backend /home/ubuntu/Travelmemory/backend

Create a .env -> nano .env

Update the details of mongo BD cluster with

MONGO\_URI='ENTER\_YOUR\_URL'

PORT=3001

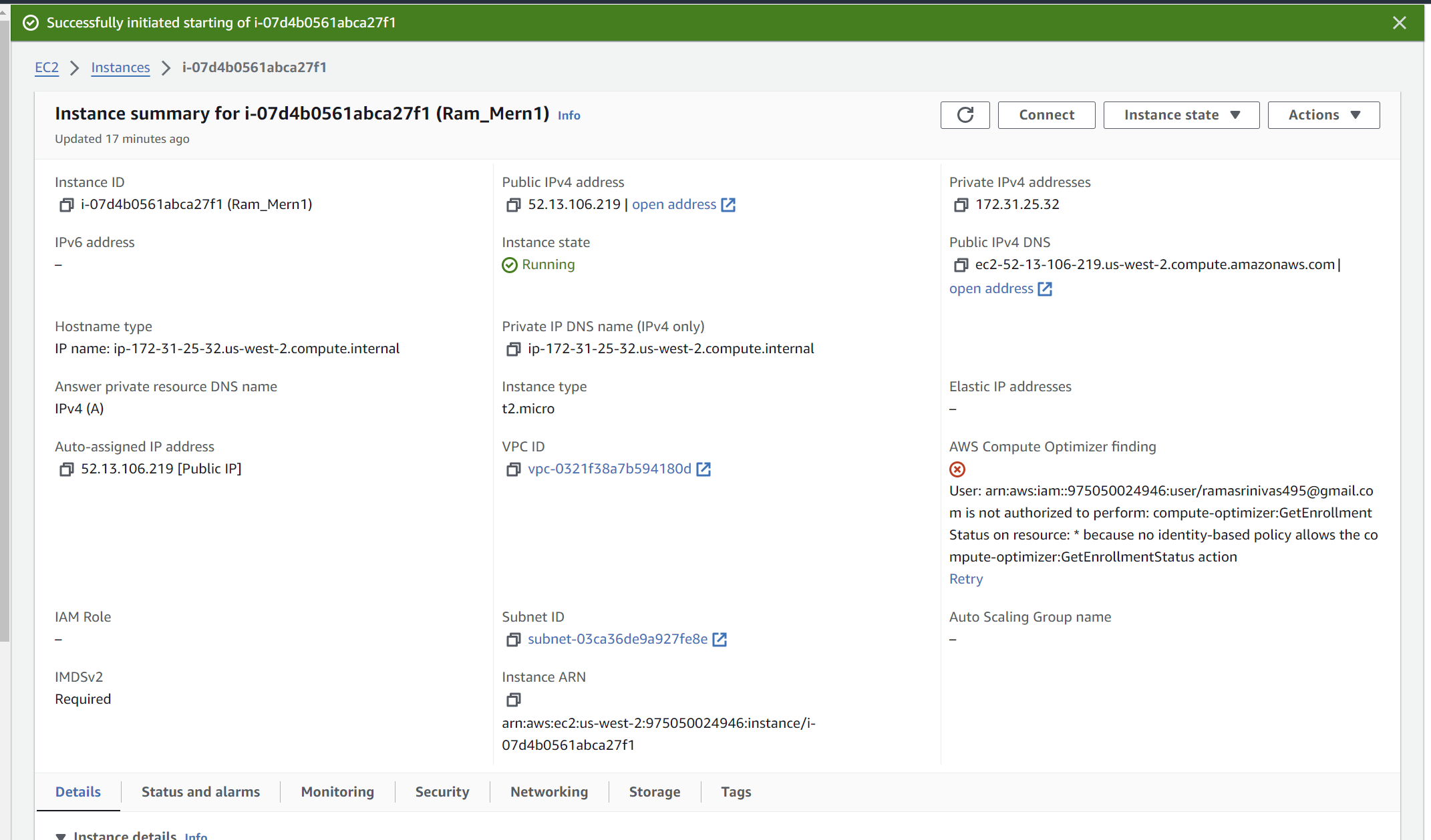
Proceed with the sudo apt install npm command to install the nesscary packages.

For Reverse proxy Nginx-> sudo apt install nginx

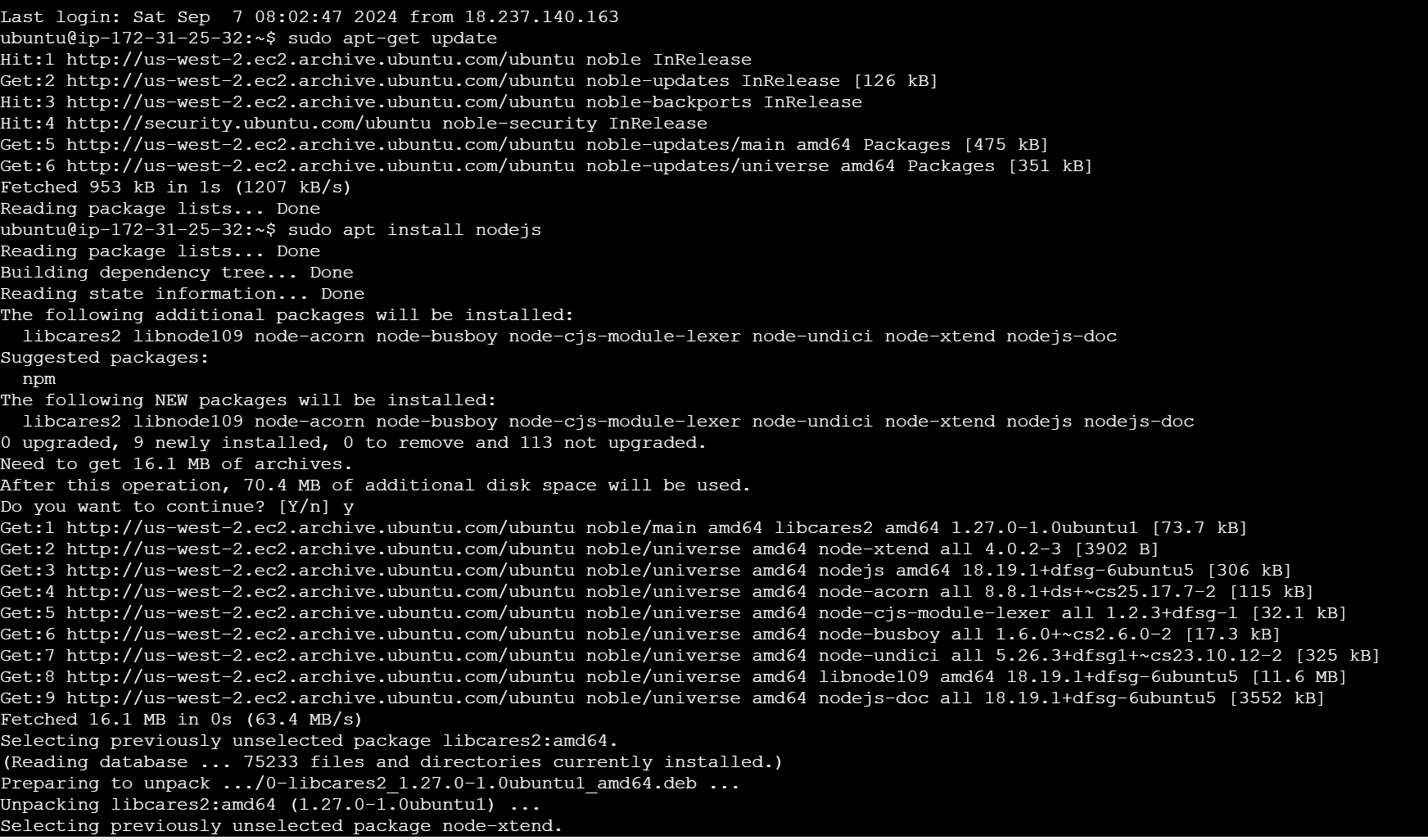
and proceed with the path

Sudo nano /etc/nginx/sites-available/default

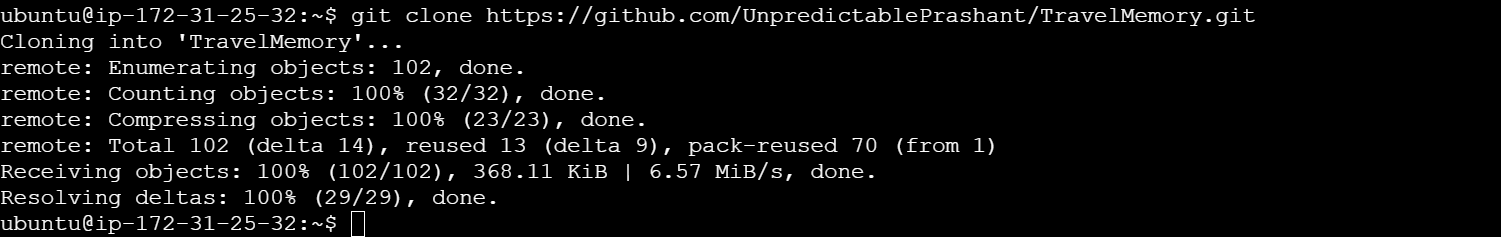
Add the following configuration:

Create EC2 instance :

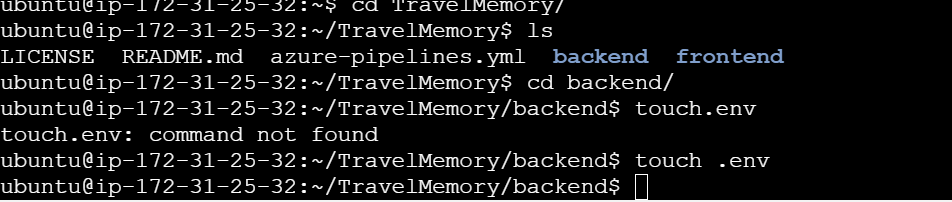
Install nodejs



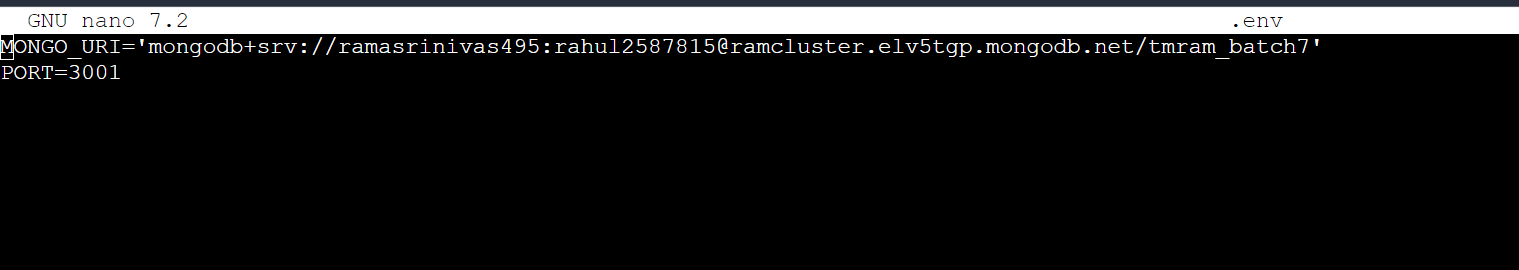
Clone <https://github.com/UnpredictablePrashant/TravelMemory>



Create . env



Add in .env



Change the following in nginx default file

server {

listen 80;

server\_name “Backend\_public\_id”;

location / {

proxy\_pass http://localhost:3000;

proxy\_http\_version 1.1;

proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

}

}

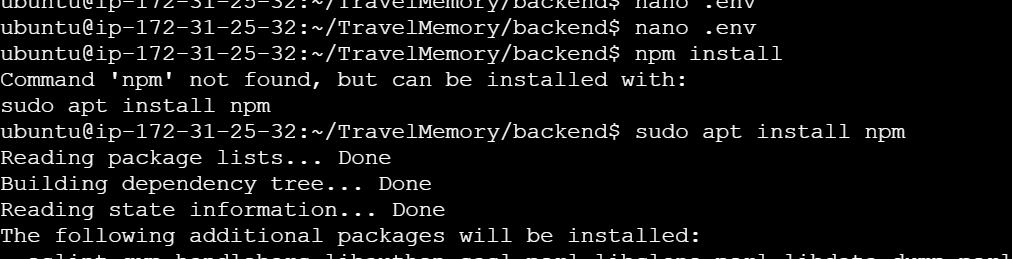
check for the nginx test with -> sudo nginx -t if successful proceed to restart the nginx with sudo

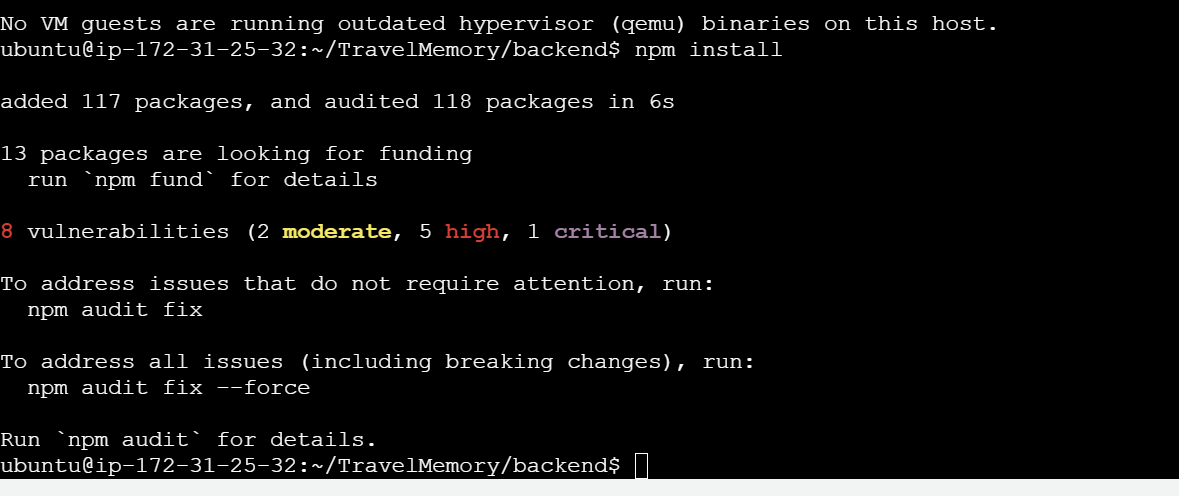
systemctl restart nginx

Once done proceed to backend folder and start the 3001 port via command node index.js

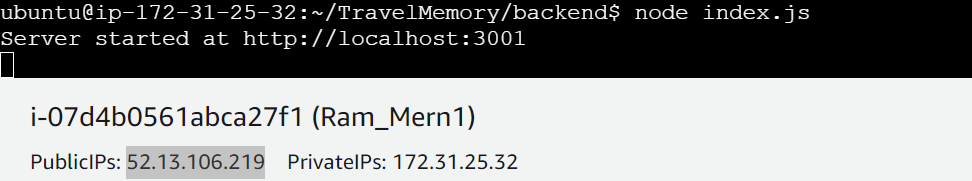
Back end would be started

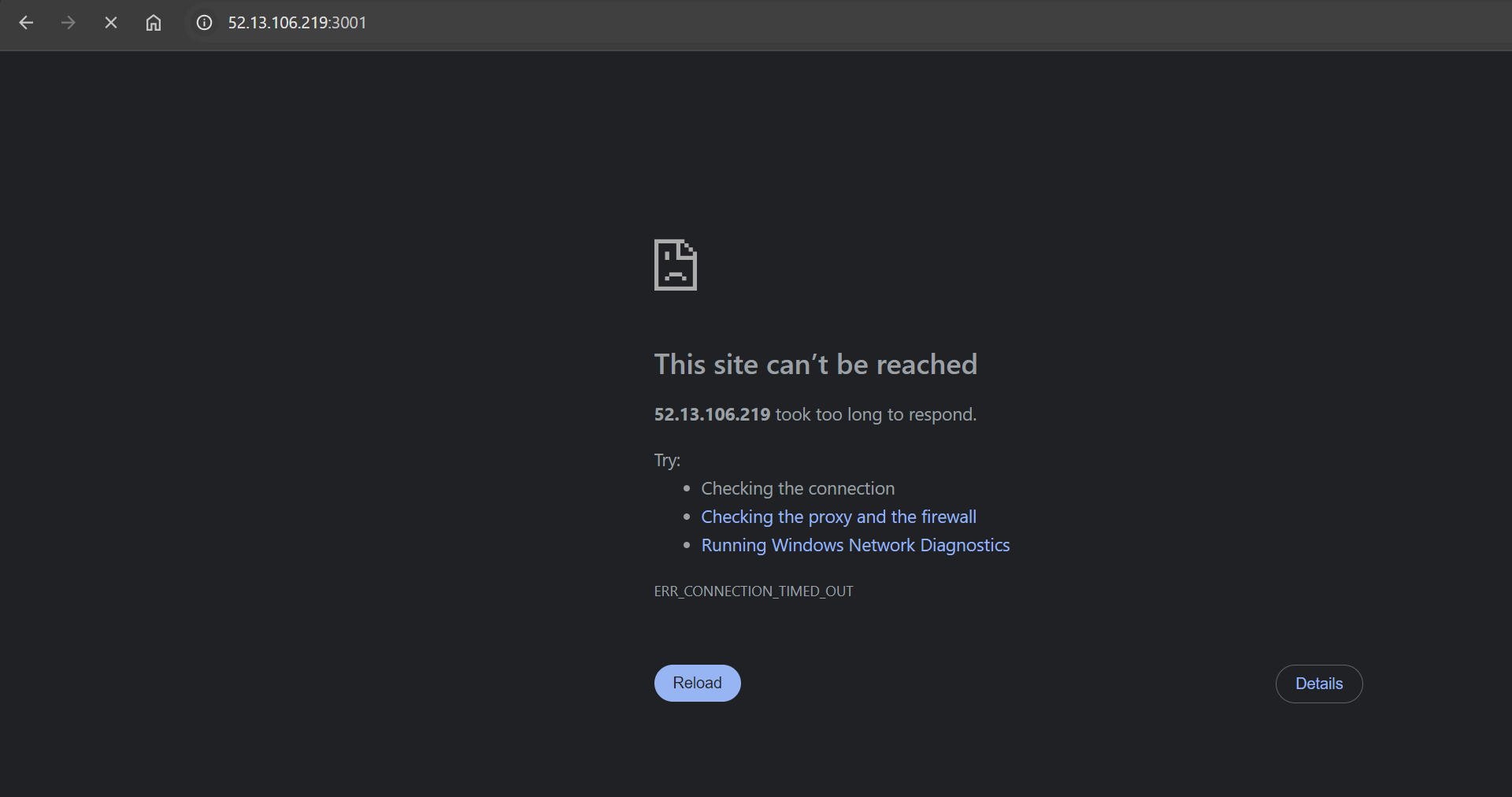
Install npm



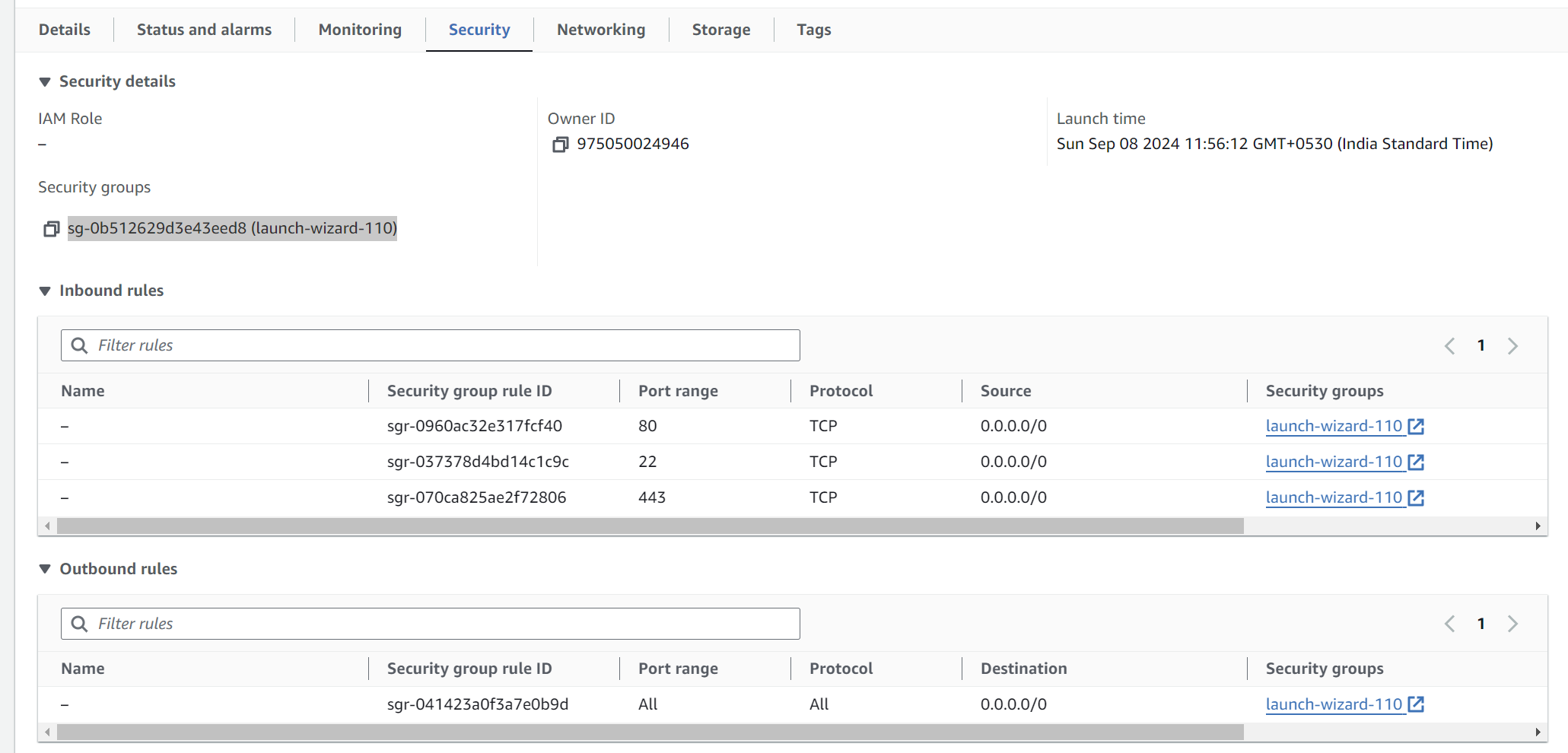


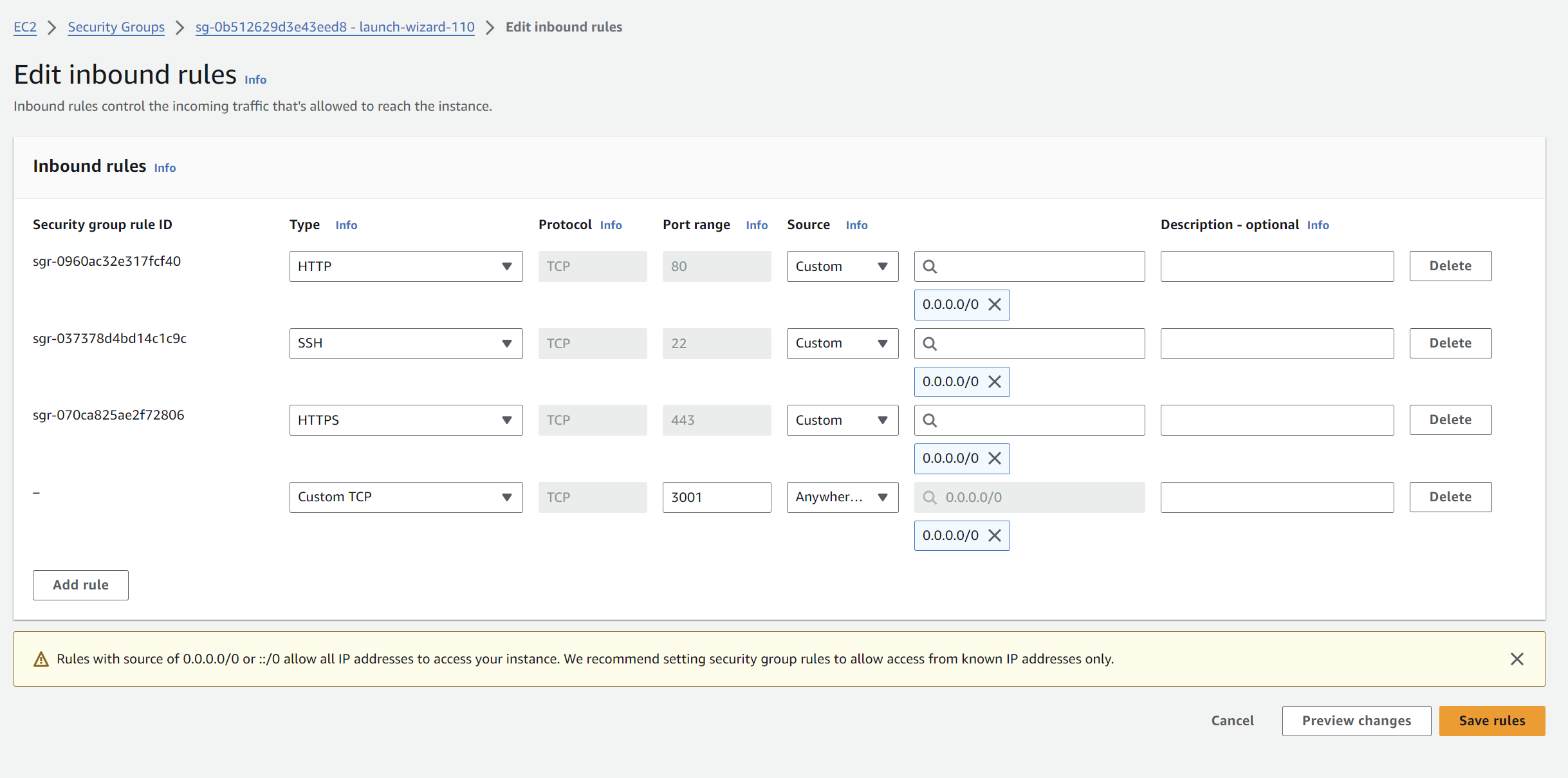
Run the index.js on

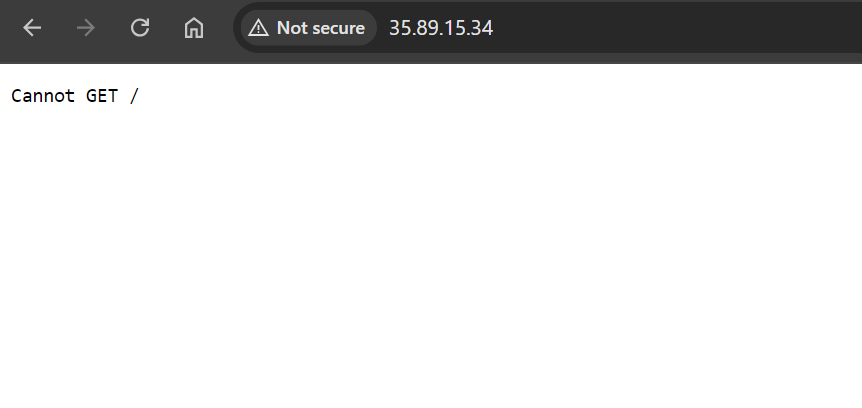




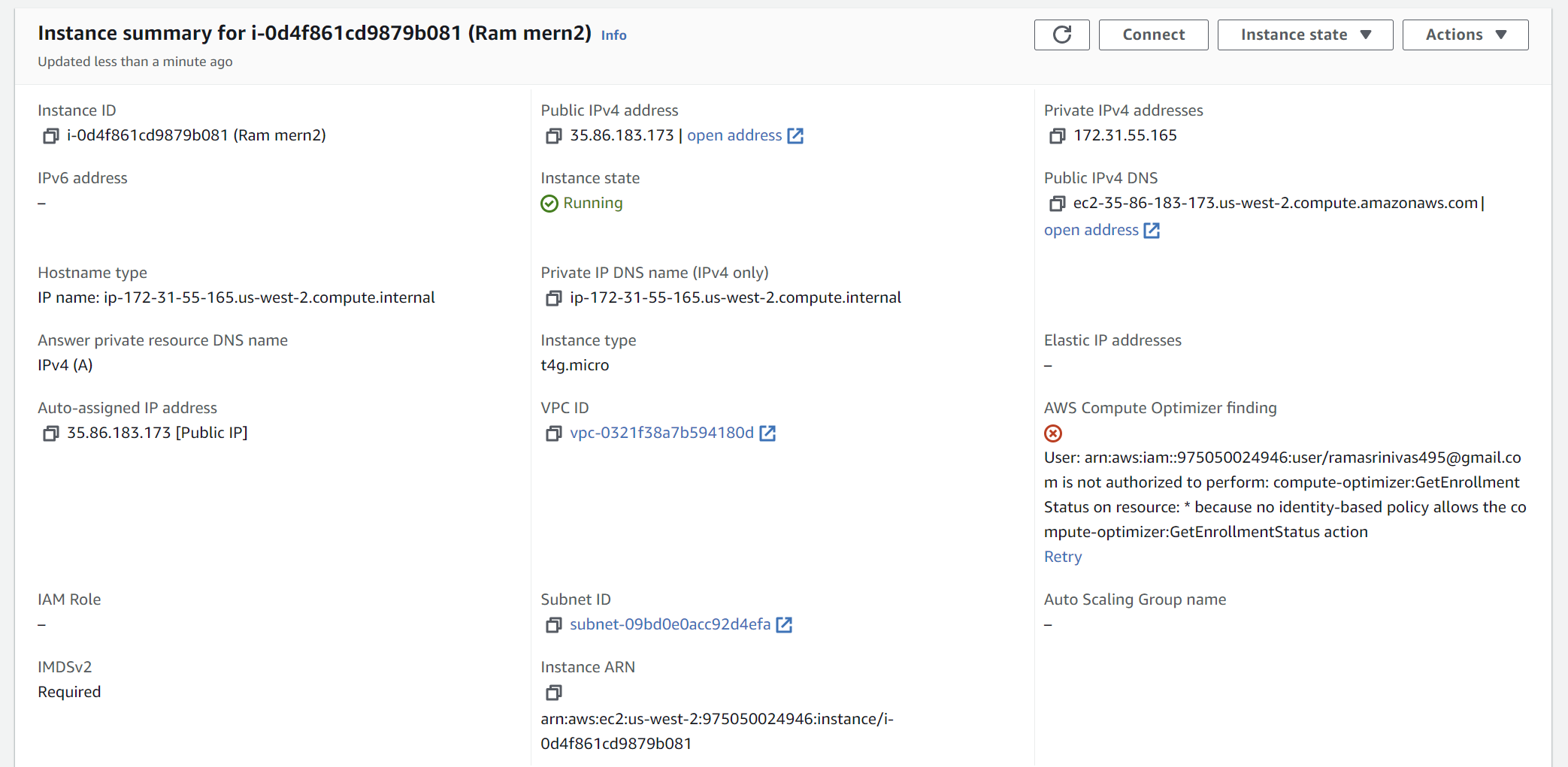
Go to security group and add 3001 port

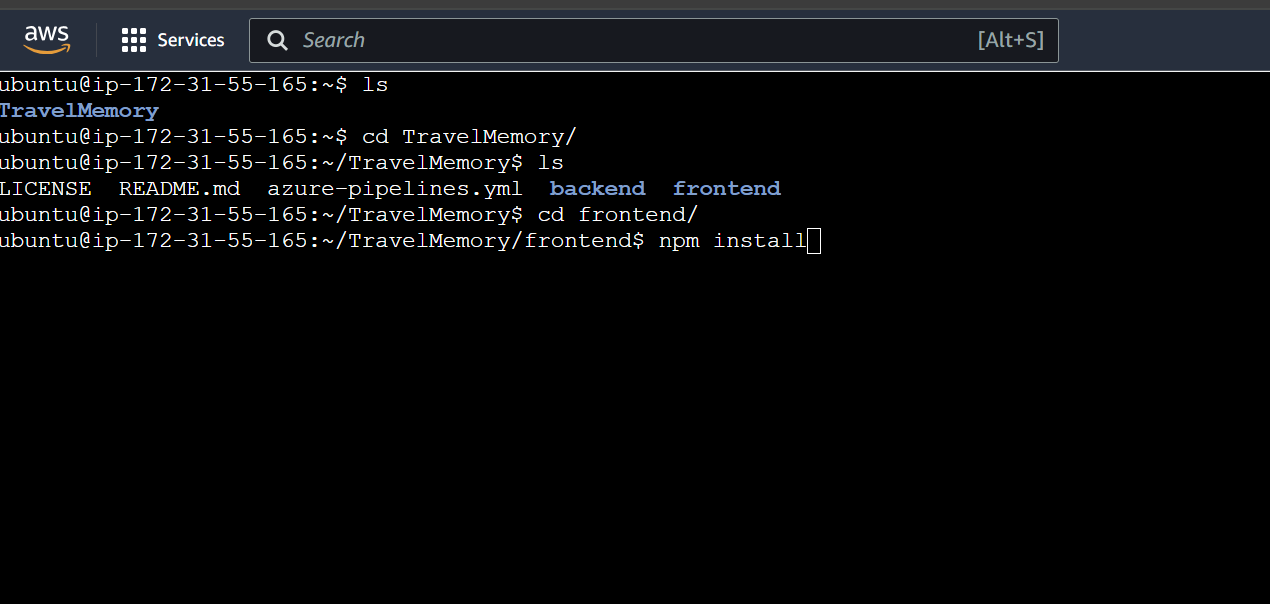




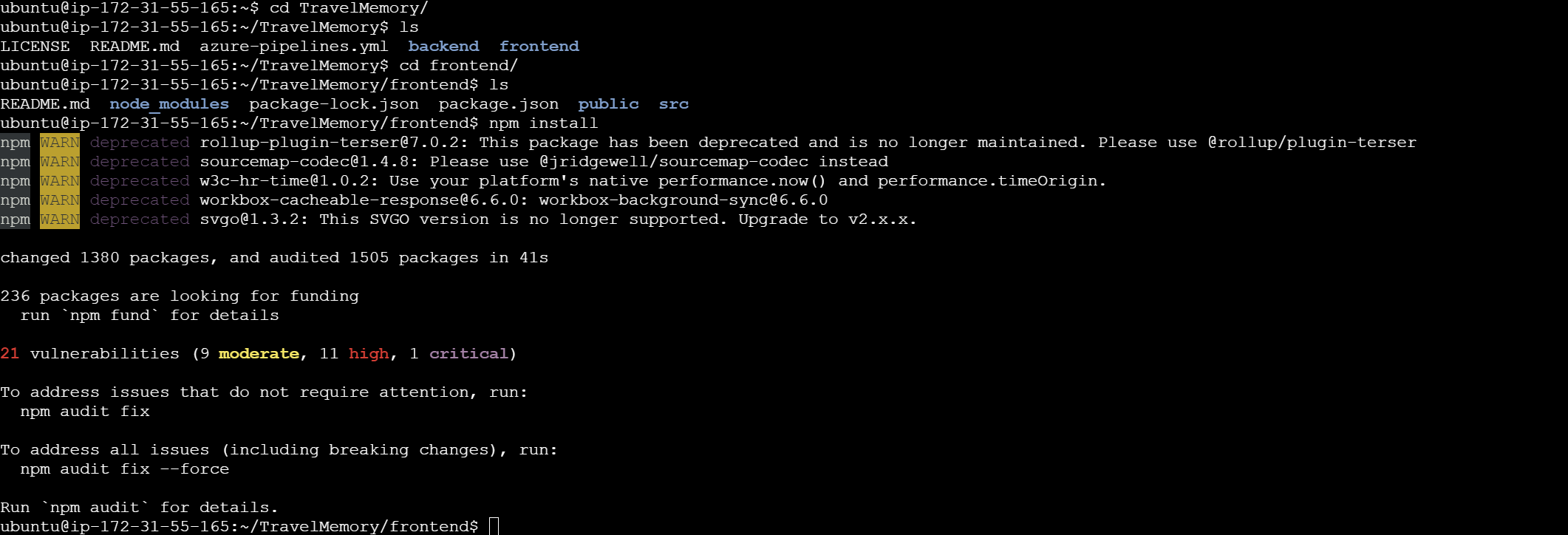


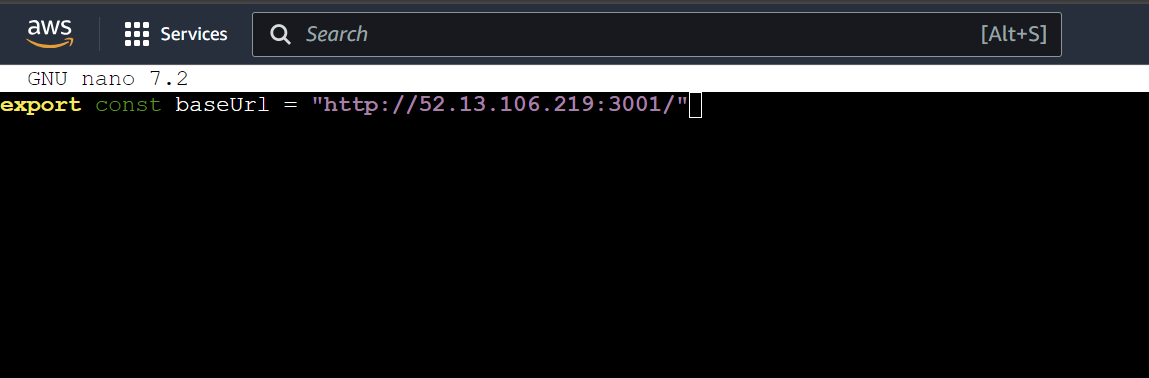
**Now create another EC2 instance for frontend:**



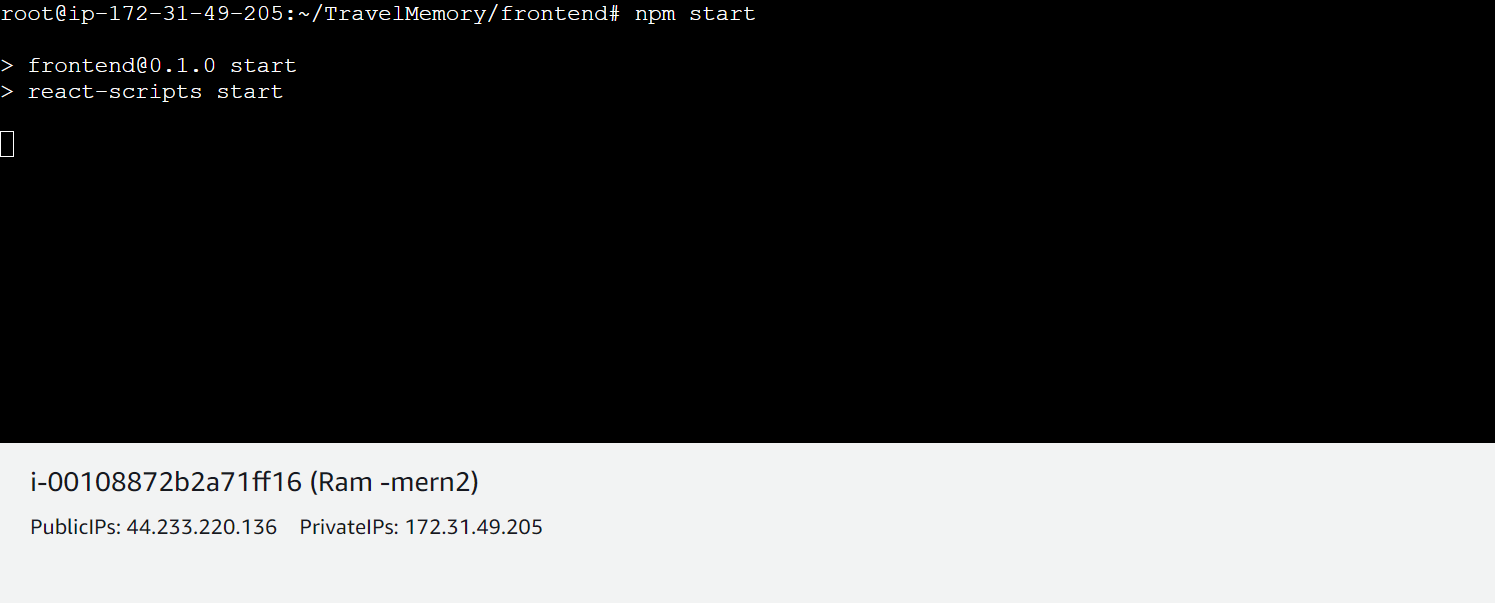


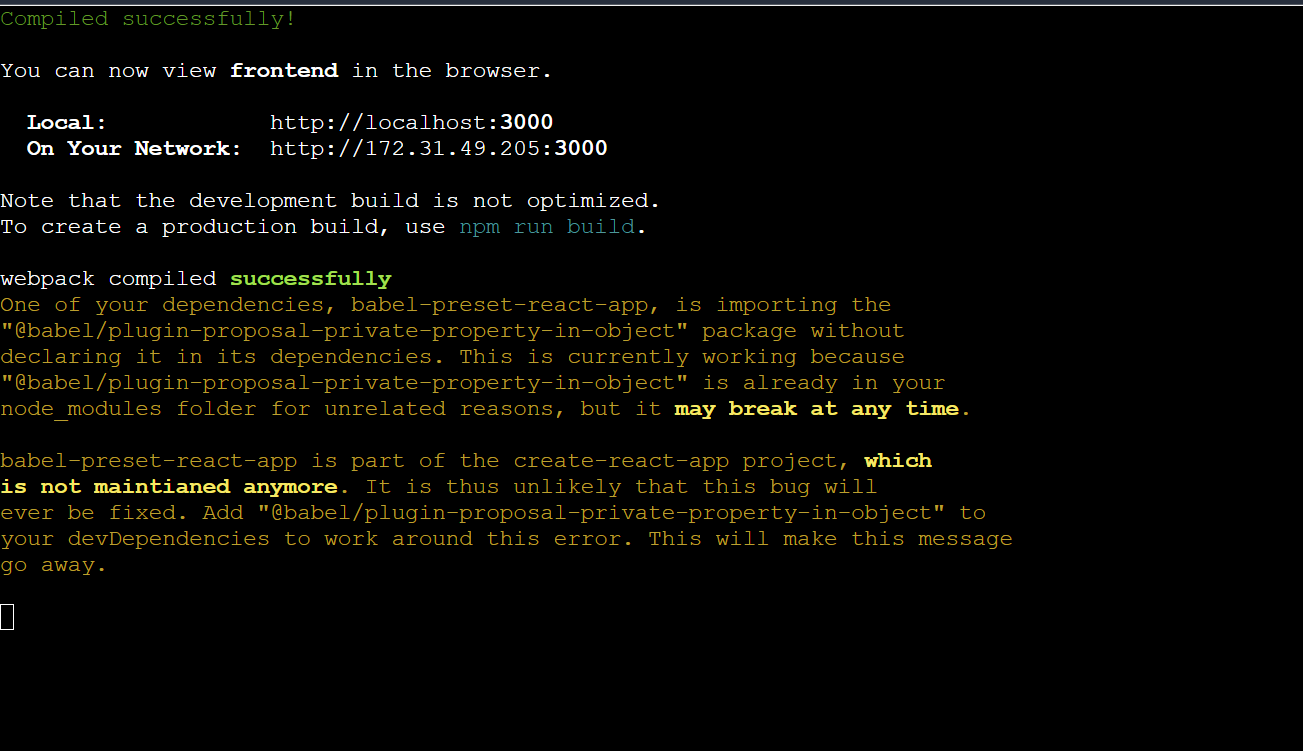
2. Frontend and Backend Connection: - Navigate to the `urls.js` in the frontend directory. - Update the file to ensure the front end communicates effectively with the backend.

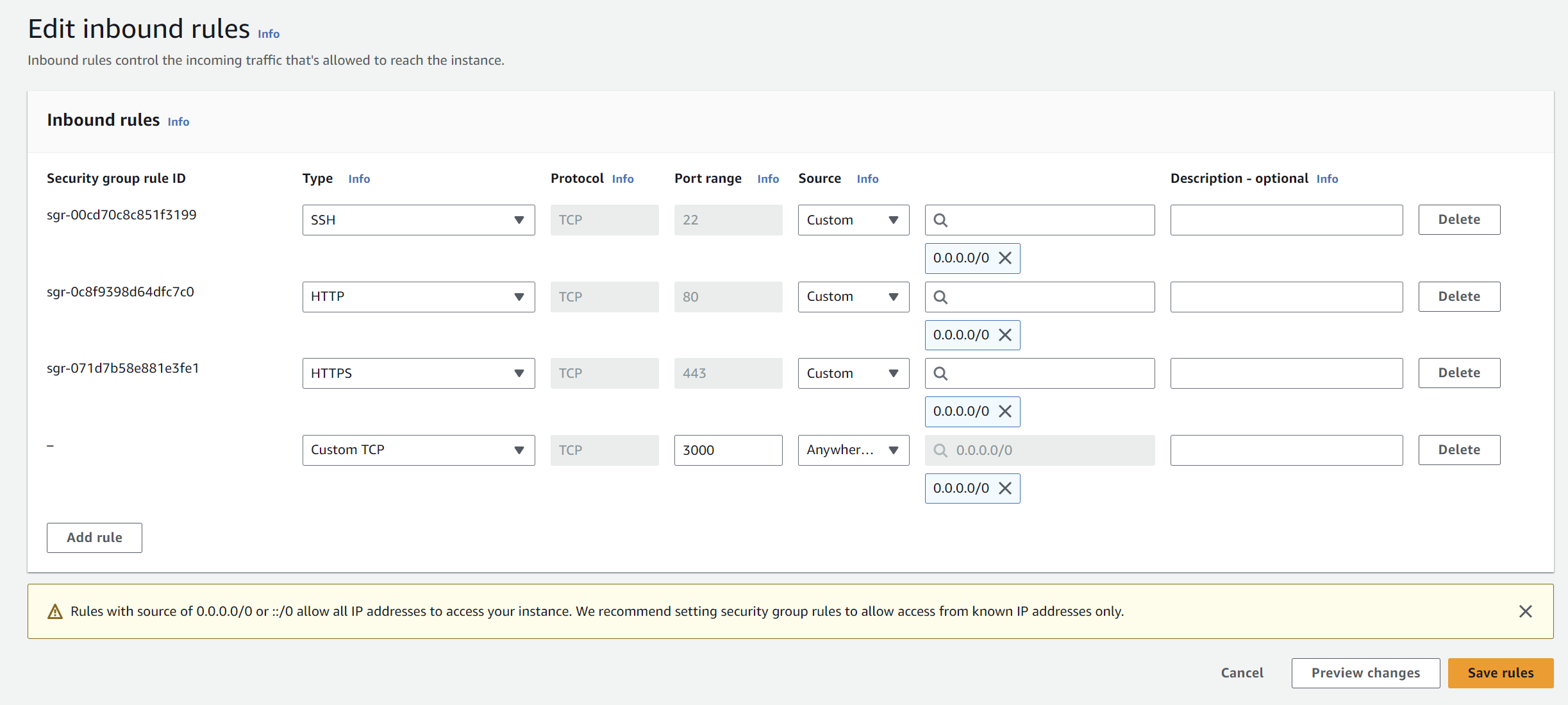
Do npm Install in frontend

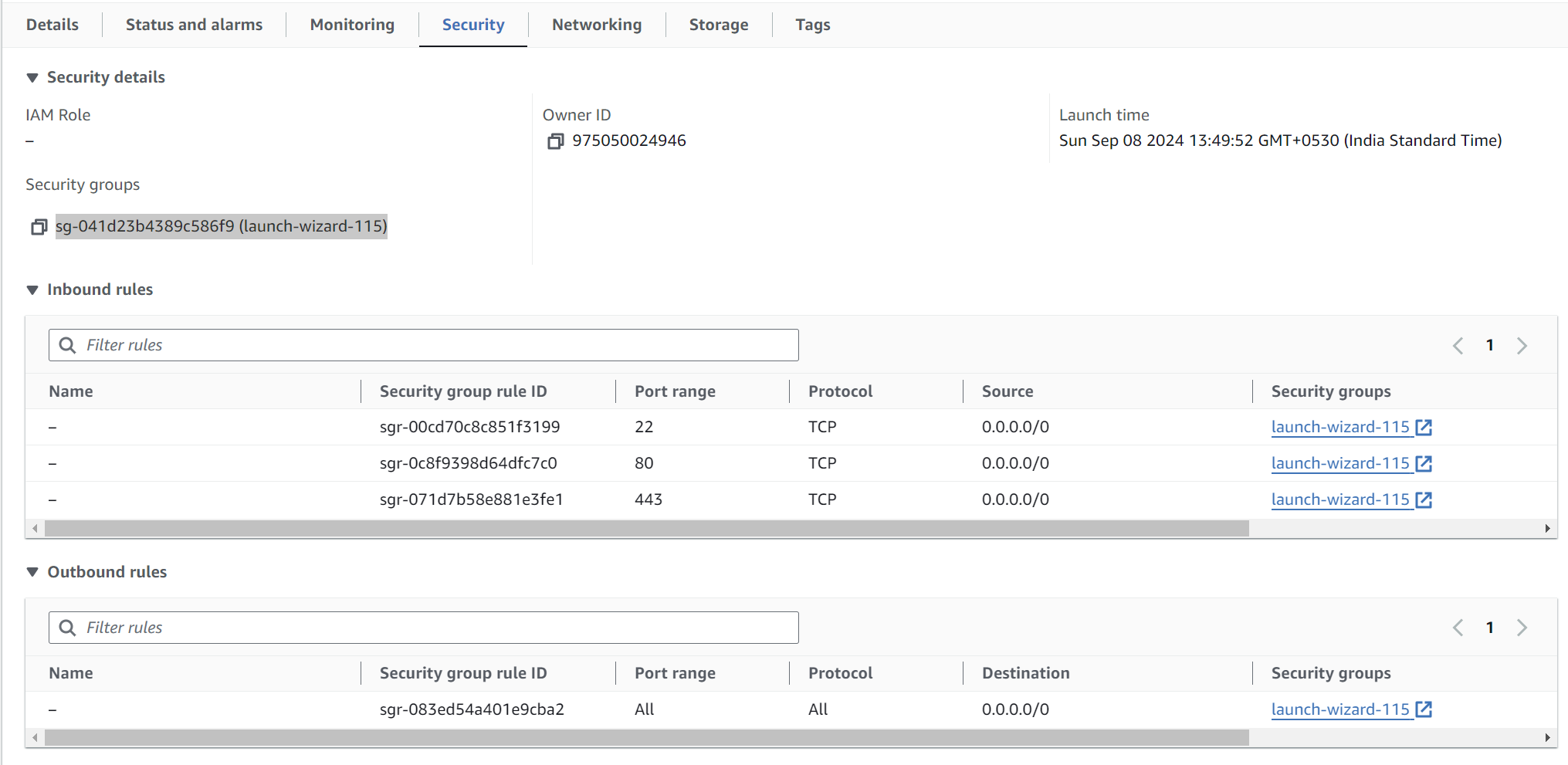
Edit the src url file

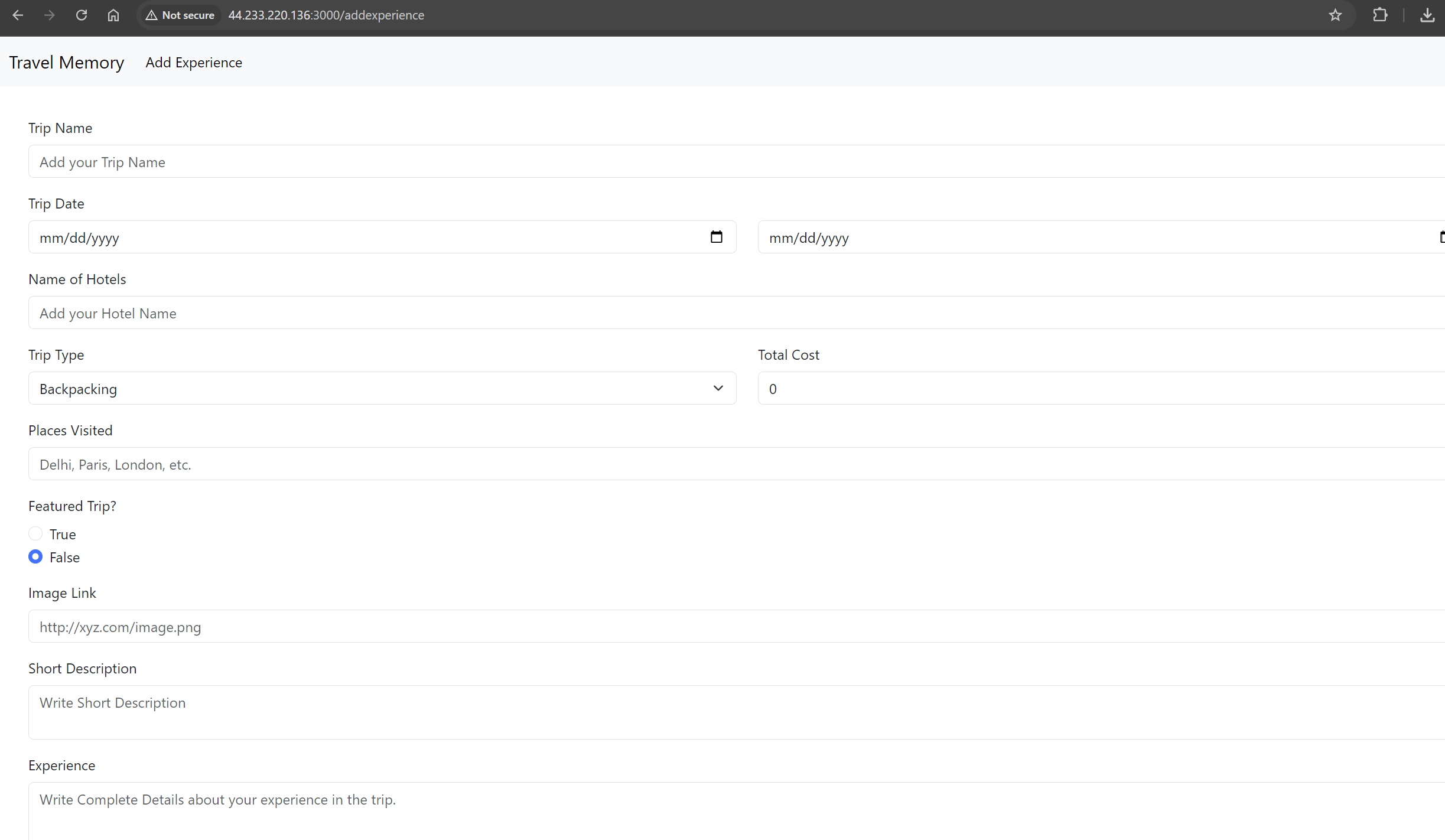
Npm start

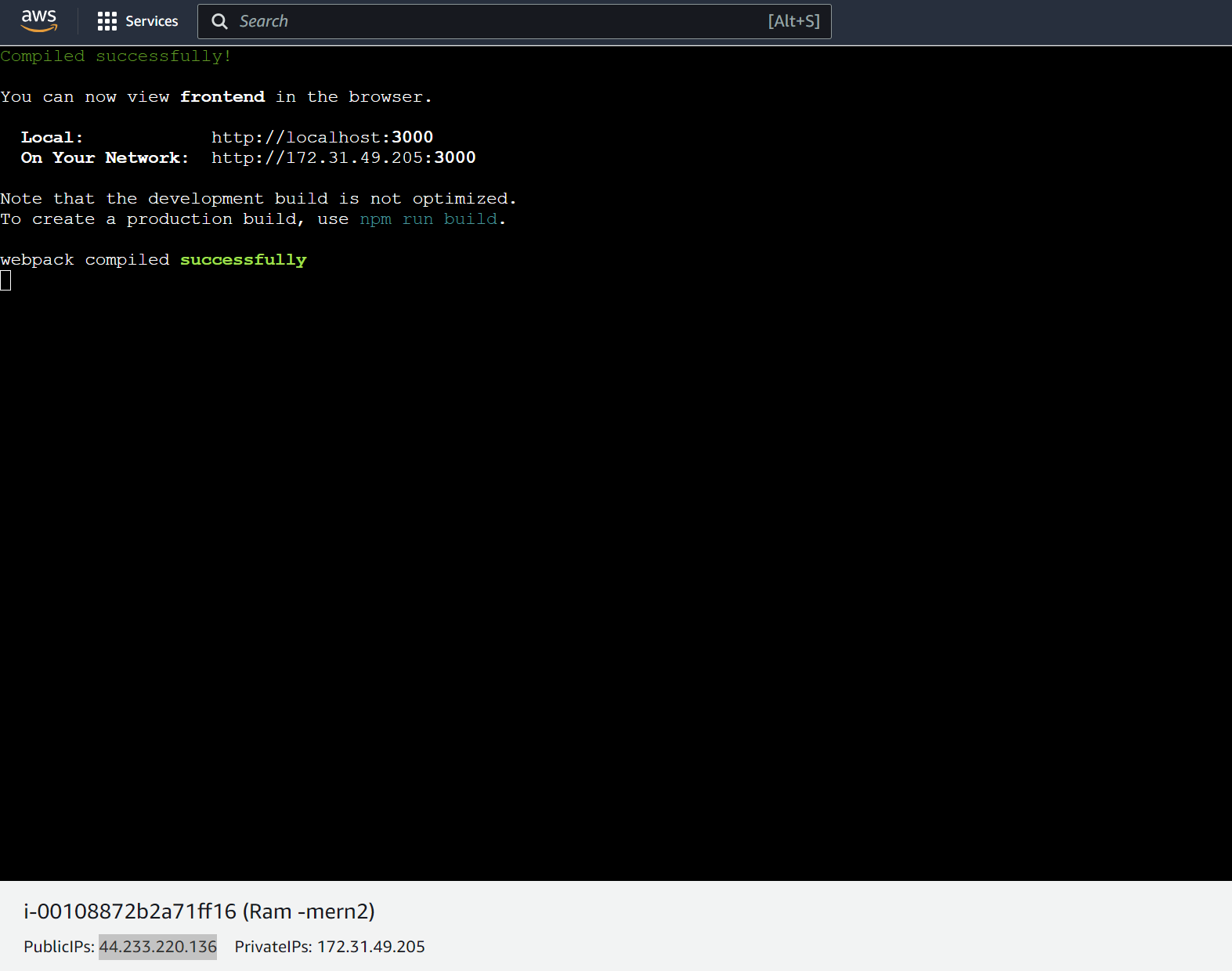


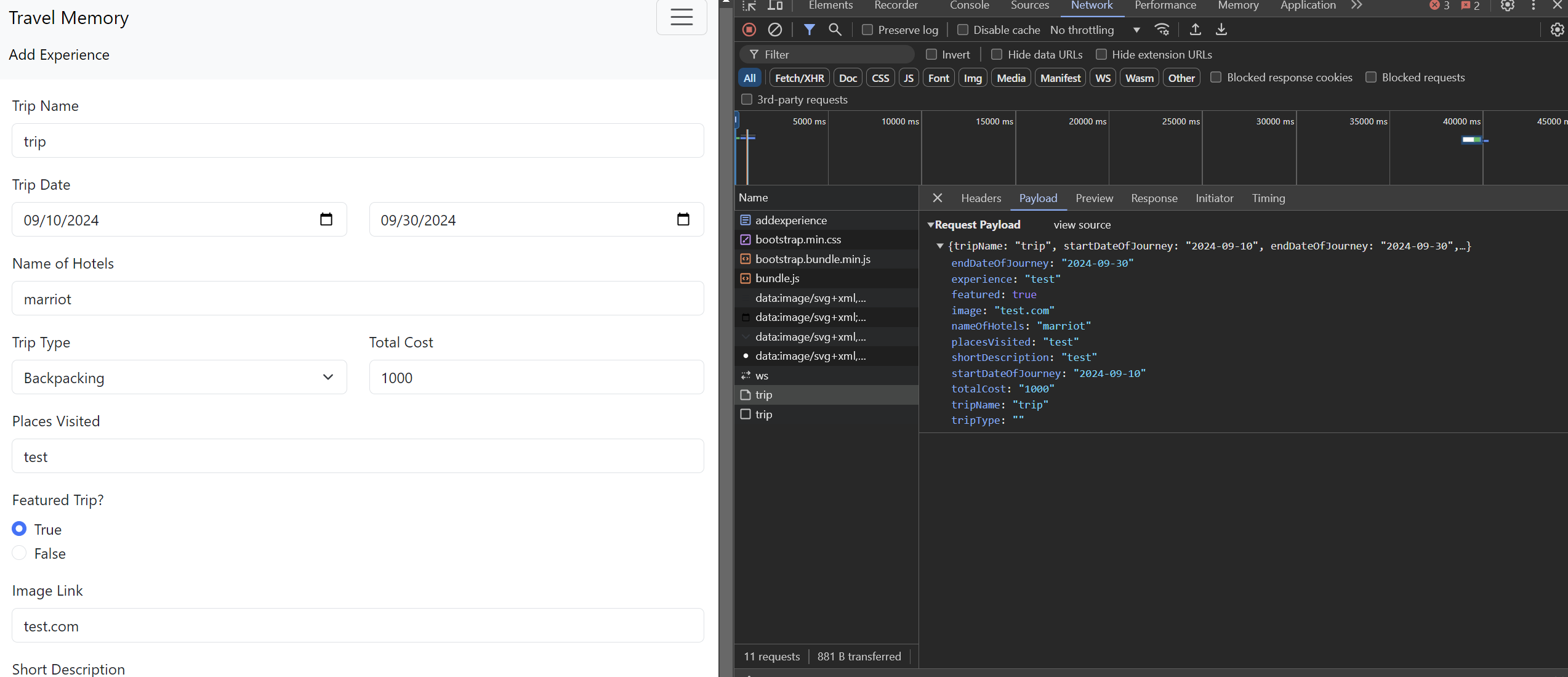


Add port 3000 in security group 



PublicIp : 44.233.220.136:3000





Come back with the frontend folder and proceed with the npm install command following with

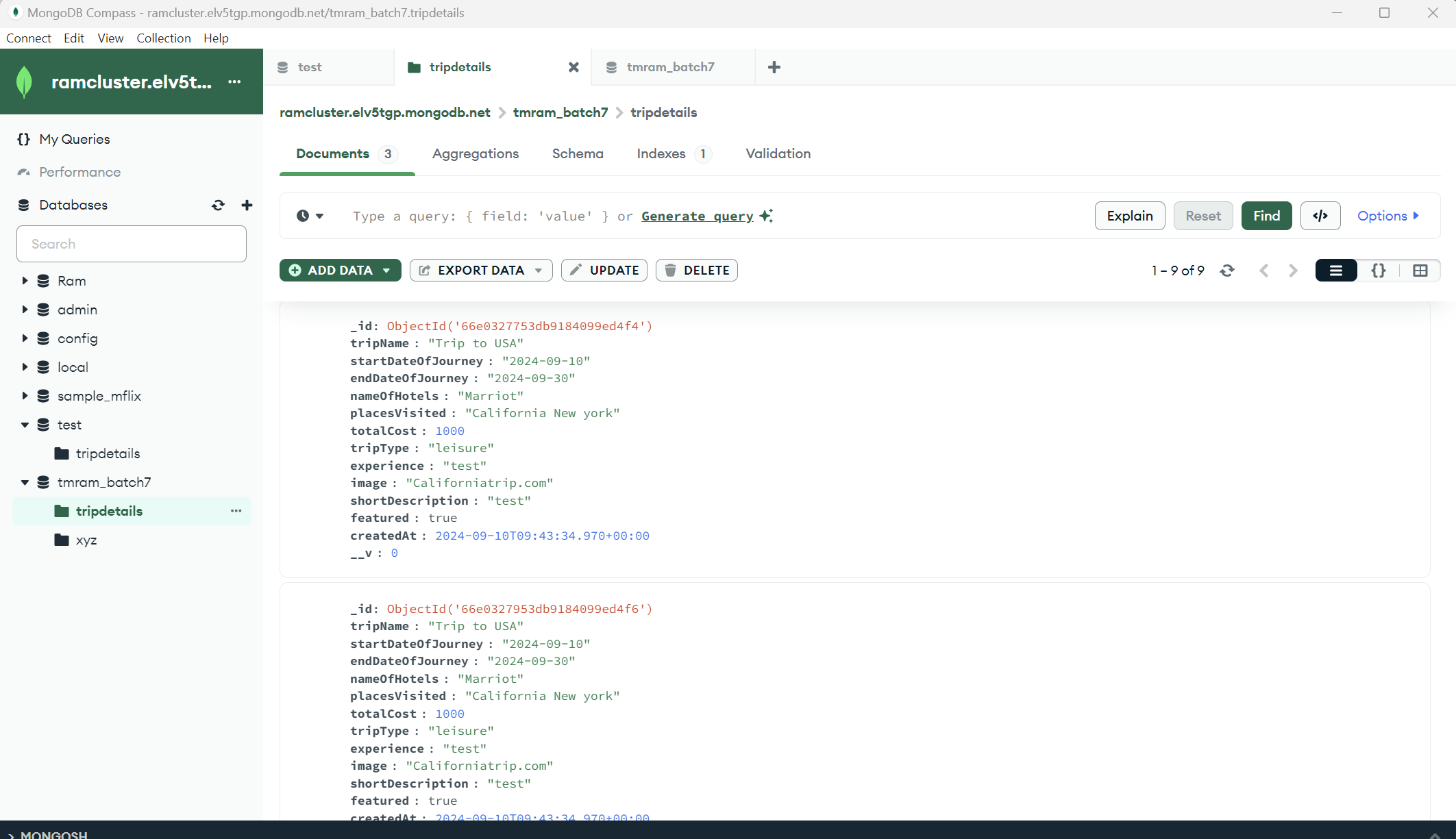
npm run build

Once done proceed with the npm start to start the front\_end

The app is successfully deployed with the backend connected and the database as a mongobd

Enter the details and submit the form check that the details are saved in the data base

Once checking the EC2 and MongoDB

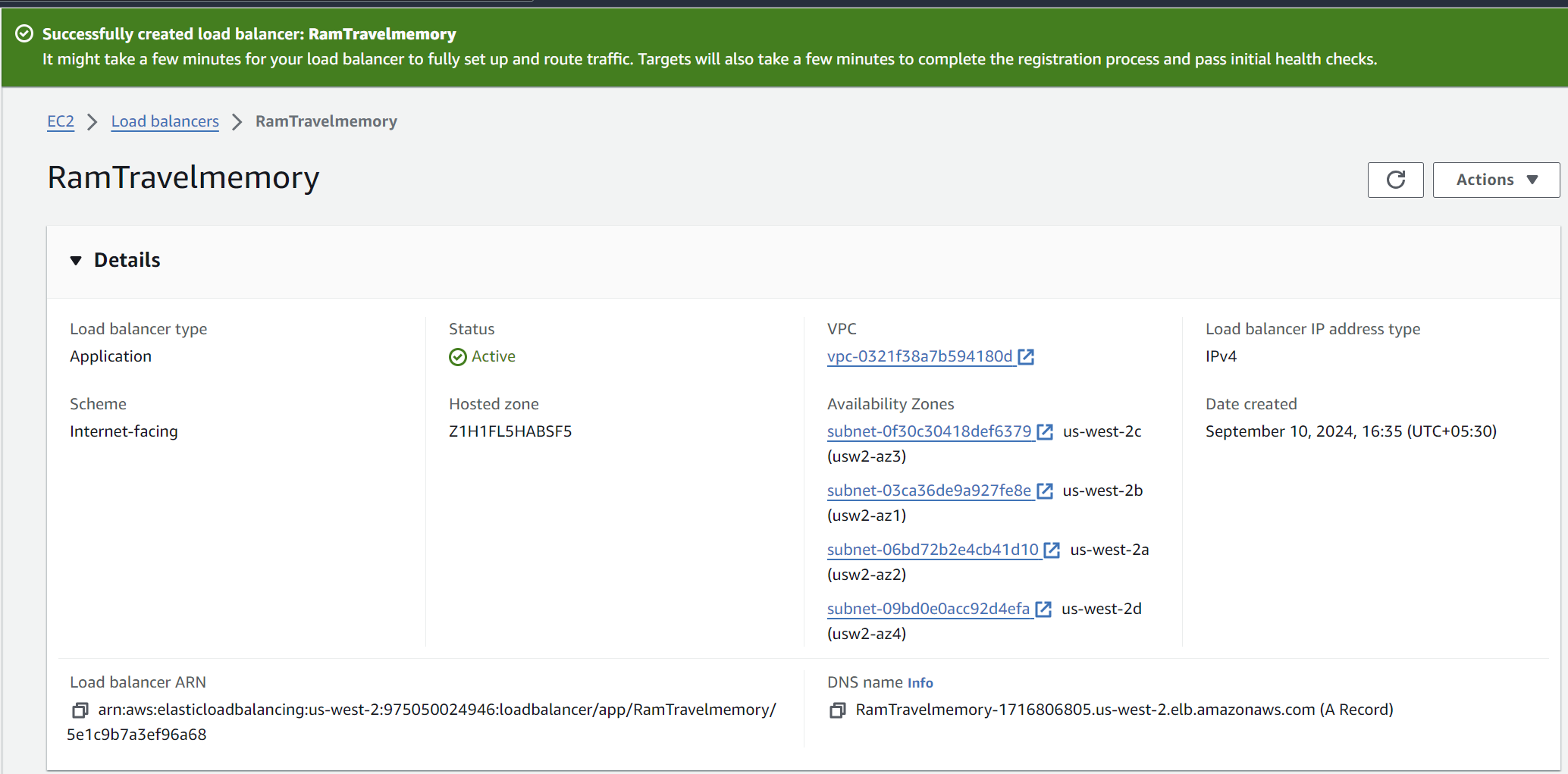


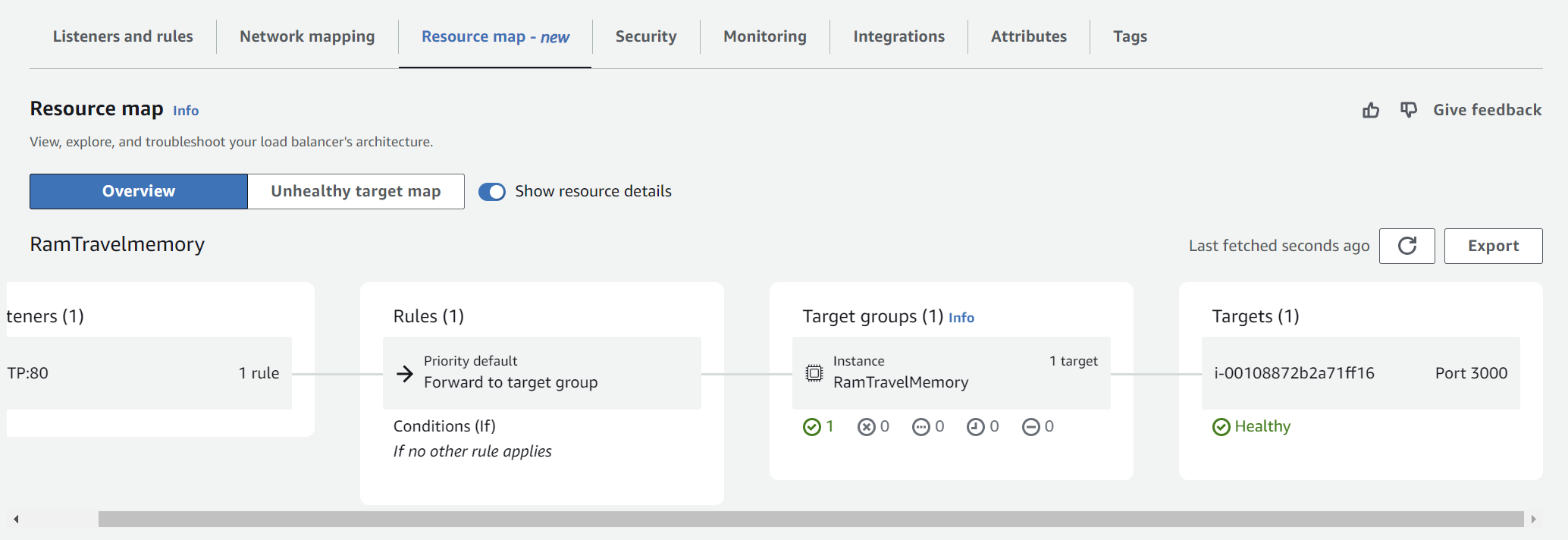
Scaling the Application: - Create multiple instances of both the frontend and backend servers. - Add these instances to a load balancer to ensure efficient distribution of incoming

traffic.

Create a target group in AWS for frontend with the 3000 port HTTP

Create load balacer:

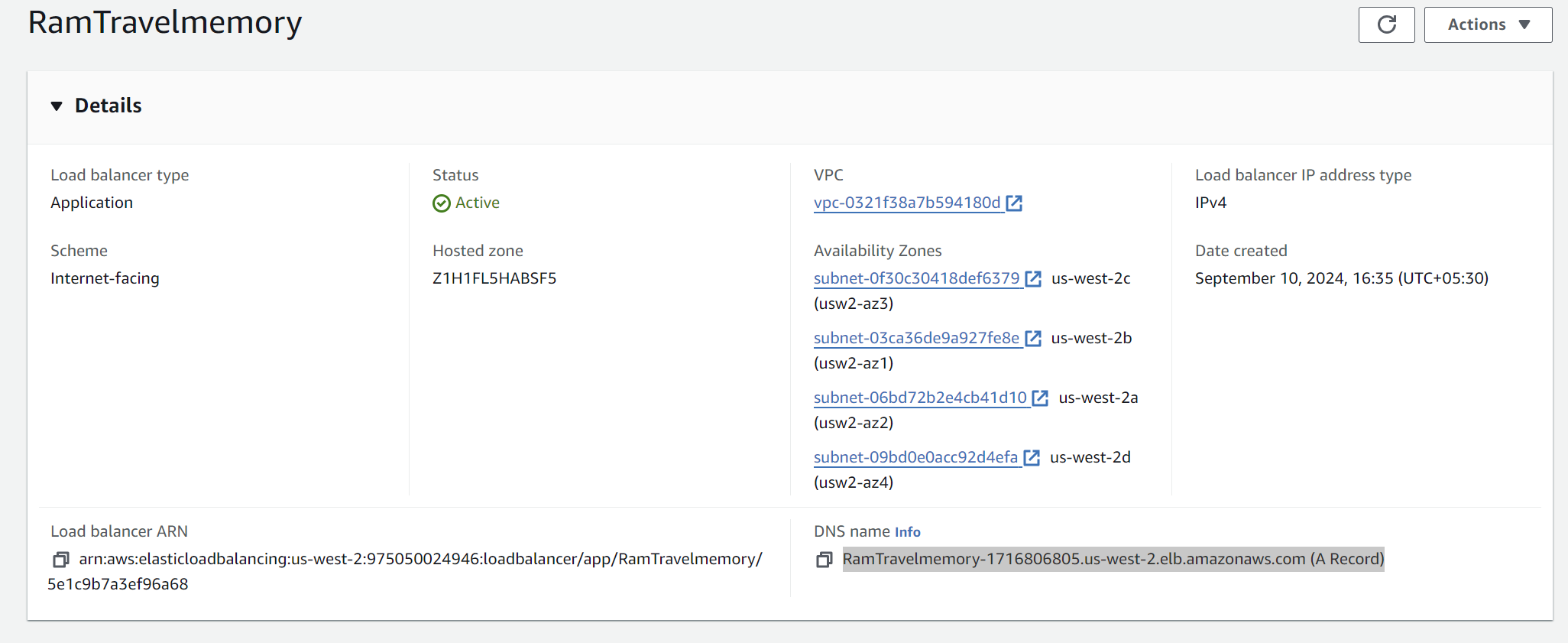




Add the available instance which is running the frontend

Now create the load balancer and add the target group which is created earlier with the instances

created availability zones

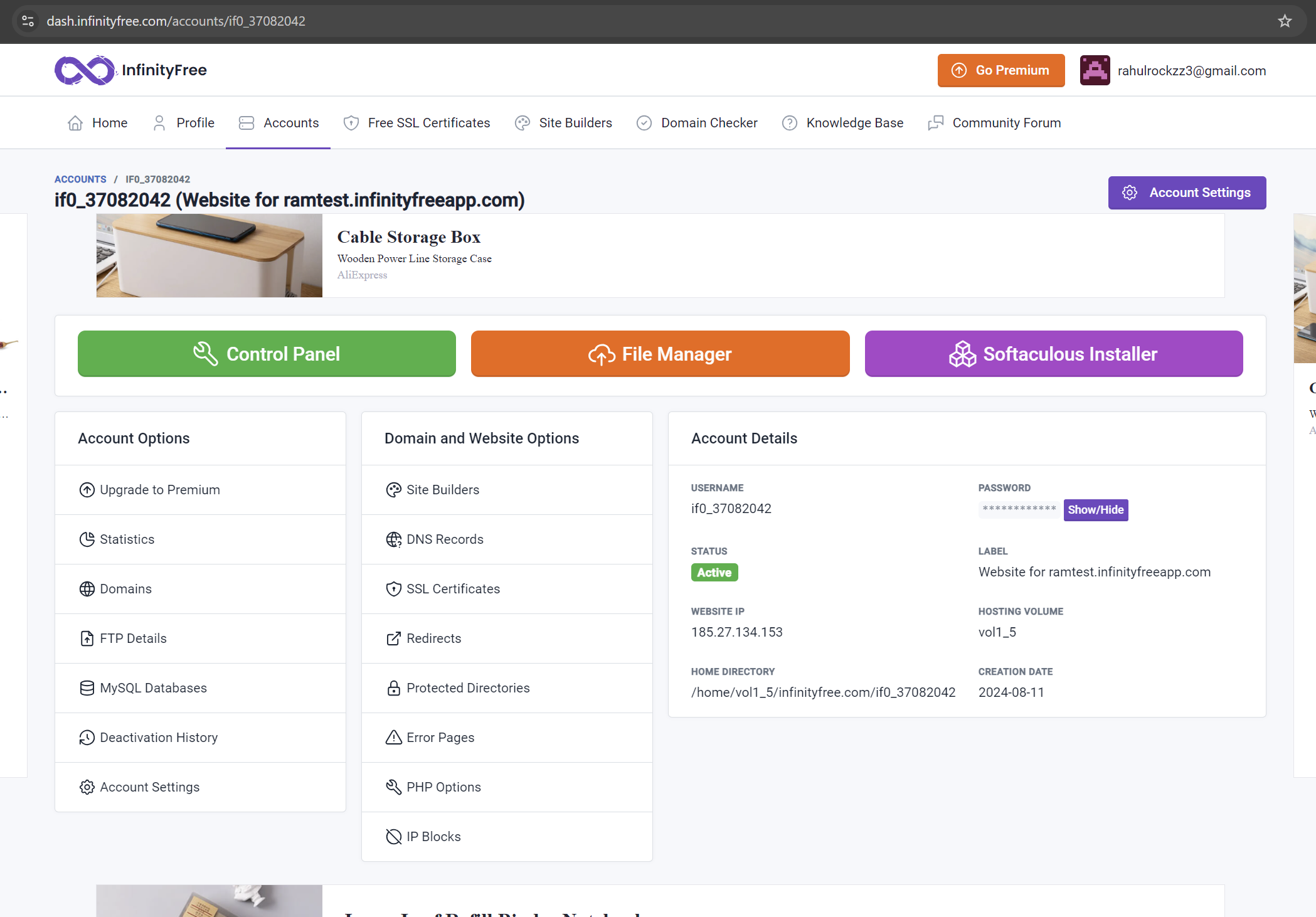


Domain Setup with Cloudflare: - Connect your custom domain to the application using Cloudflare. - Create a CNAME record pointing to the load balancer endpoint.

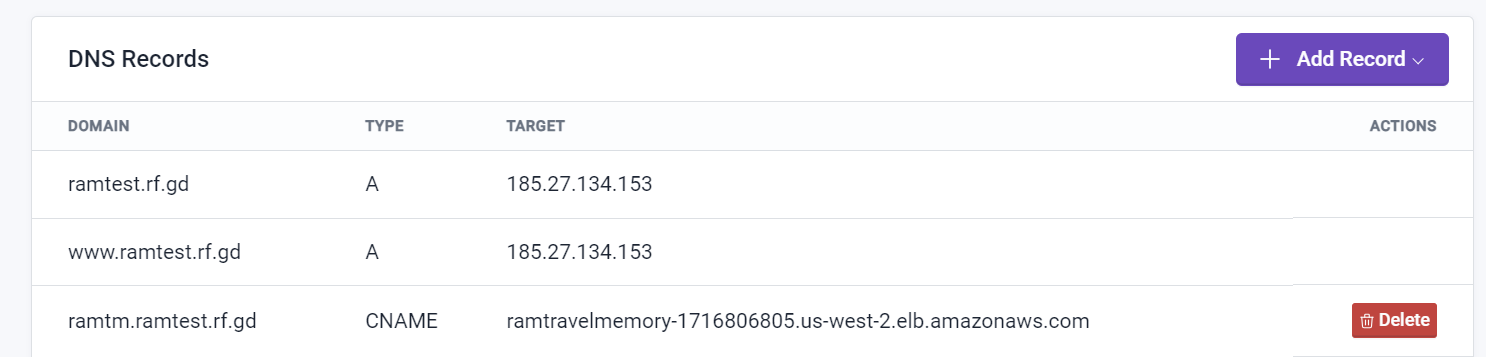
Set up an A record with the IP address of the EC2 instance hosting the front end.

Setup a Domain. As the cloud flare is a paid one

Hence using https://dash.infinityfree.com/ to setup a domain



Now create a CNAME with and add the DNS name of the load balancer



Using the DNS Records name the website will be opened

