Travelmemory MERN Stack Application Deployment

This document provides a comprehensive guide to deploying the TravelMemory MERN stack application on Amazon. EC2 instances. It includes steps to configure the backend and frontend servers, set up Nginx reverse proxy, and integrate the application with a load balancer and Cloudflare.

by Ashraf Khan

Step 1: EC2 Instances Setup

Launch 3 Ubuntu EC2 instances:

Backend Server: backend_server001

Frontend Servers: frontend_server001 and frontend_server002

Configure Security Groups:

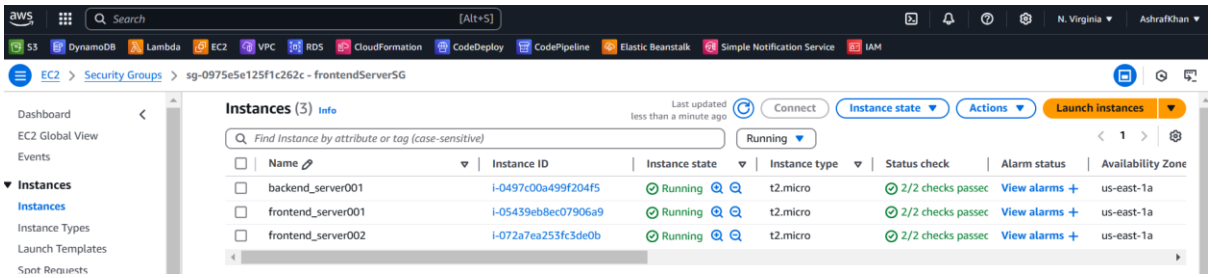
Backend Security Group (backendServerSG):

Open ports: 22 (SSH), 80 (HTTP), 443 (HTTPS), 3000

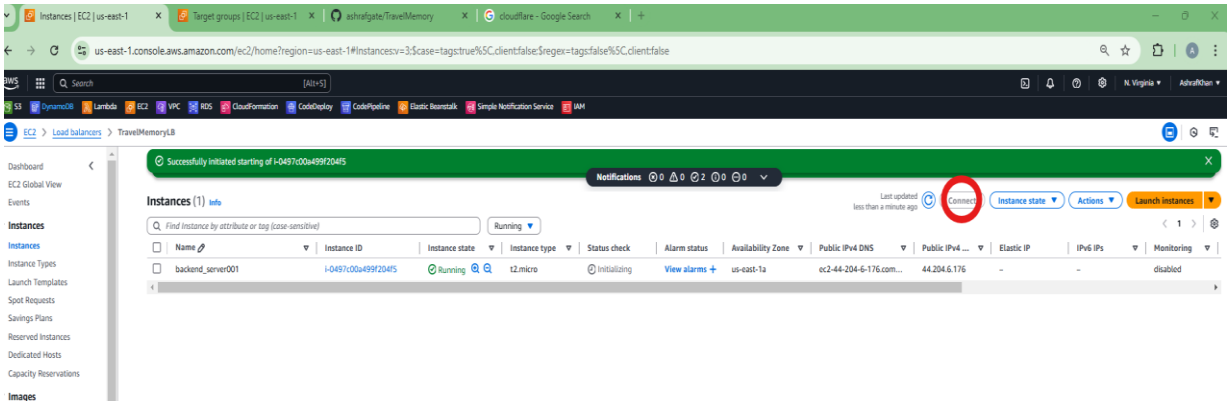
Frontend Security Group (frontendServerSG):

Open ports: 22 (SSH), 80 (HTTP), 443 (HTTPS), 3000

Launched 3 instances



Step 2: Backend Server Configuration



Connect the backend server and install the below things: -

Install prerequisites:

sudo apt update

sudo apt install nodejs

sudo apt install npm

Clone the repository:

git clone https://github.com/ashrafgate/TravelMemory.git

Navigate to the backend folder: cd TravelMemory/backend

Create a .env file:

sudo nano .env

Add the following in .env file, Save and exit:

PORT=3000

MONGO_URI="mongodb+srv://ashrafkhandu:KS1QHYOuV9W2ZLGC@cluster0.gp76h.mongodb.net/ashrafkhan"

Step 3: Nginx Reverse Proxy Setup for Backend

Install Nginx:

sudo apt install -y nginx

Edit the Nginx default configuration:

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

Replace with the following:

server {

listen 80;

server_name 3.83.91.117;

location / {

proxy_pass http://127.0.0.1:3000;

```
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;
}
}
```

Test the configuration:

```
sudo nginx -t
```

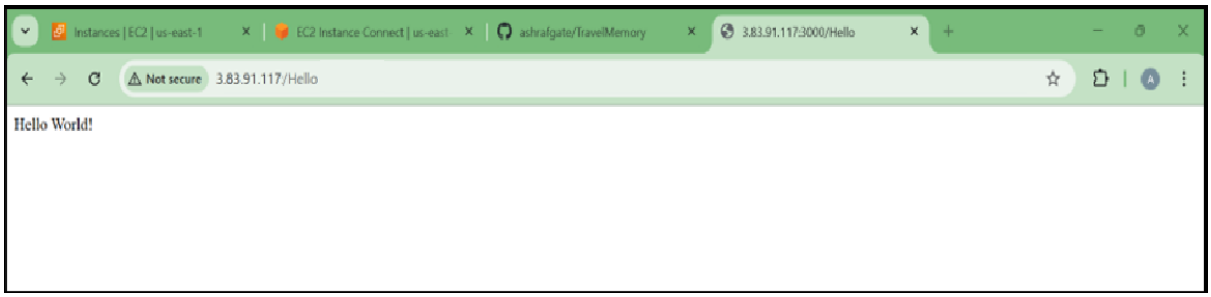
Restart Nginx:

```
sudo systemctl restart nginx
```

Install dependencies and start the server:

```
sudo npm install
sudo node index.js &
```

Verify the backend is running at http://3.83.91.117/hello in the browser



backend server running successfully on port 80

Step 6: Frontend002 Server Configuration

Connect to the frontend server002:

Frontend Server Configuration

Install prerequisites:

```
sudo apt update -y
```

sudo apt install npm

Clone the repository:

git clone https://github.com/ashrafgate/TravelMemory.git

Navigate to TravelMemory/frontend/src and edit url.js to replace 'localhost' with the backend server's public IP(3.83.91.117).

Step 7: Nginx Reverse Proxy Setup for frontend001

Install Nginx:

sudo apt install -y nginx

Edit the Nginx default configuration:

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

Replace with the following:

server {

listen 80;

server_name 44.204.34.81;

location / {

proxy_pass http://127.0.0.1:3000;

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;

}

}

Test the configuration:

sudo nginx -t

Restart Nginx:

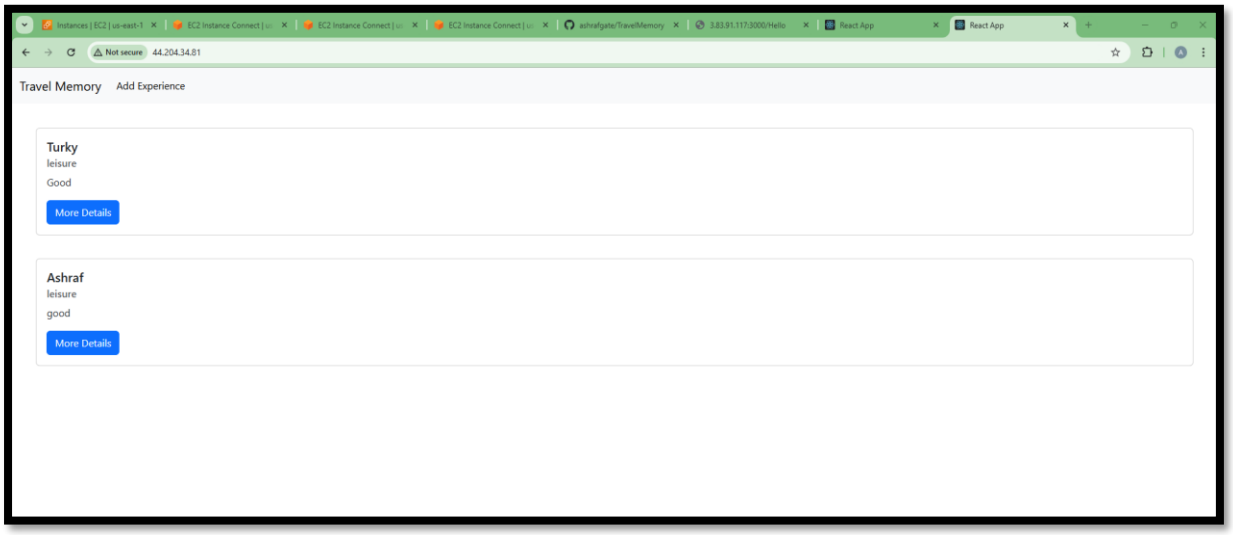
sudo systemctl restart nginx

Install dependencies and start the frontend server001:

sudo npm install

sudo npm start

Verify the frontend is running at http://44.204.34.81 in the browser



Step 8: Frontend002 Server Configuration

Connect to the frontend server002:

Frontend Server Configuration

Install prerequisites:

sudo apt update

sudo apt install -y

sudo apt install git

sudo apt install npm

Clone the repository:

git clone https://github.com/ashrafgate/TravelMemory.git

Navigate to TravelMemory/frontend/src and edit url.js to replace 'localhost' with the backend server's public IP(3.83.91.117).

Step 9: Nginx Reverse Proxy Setup for frontend002

Install Nginx:

sudo apt install -y nginx

Edit the Nginx default configuration:

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

Replace with the following:

server {

listen 80;

server_name 3.87.229.206;

location / {

proxy_pass http://127.0.0.1:3000;

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;

}

}

Test the configuration:

sudo nginx -t

Restart Nginx:

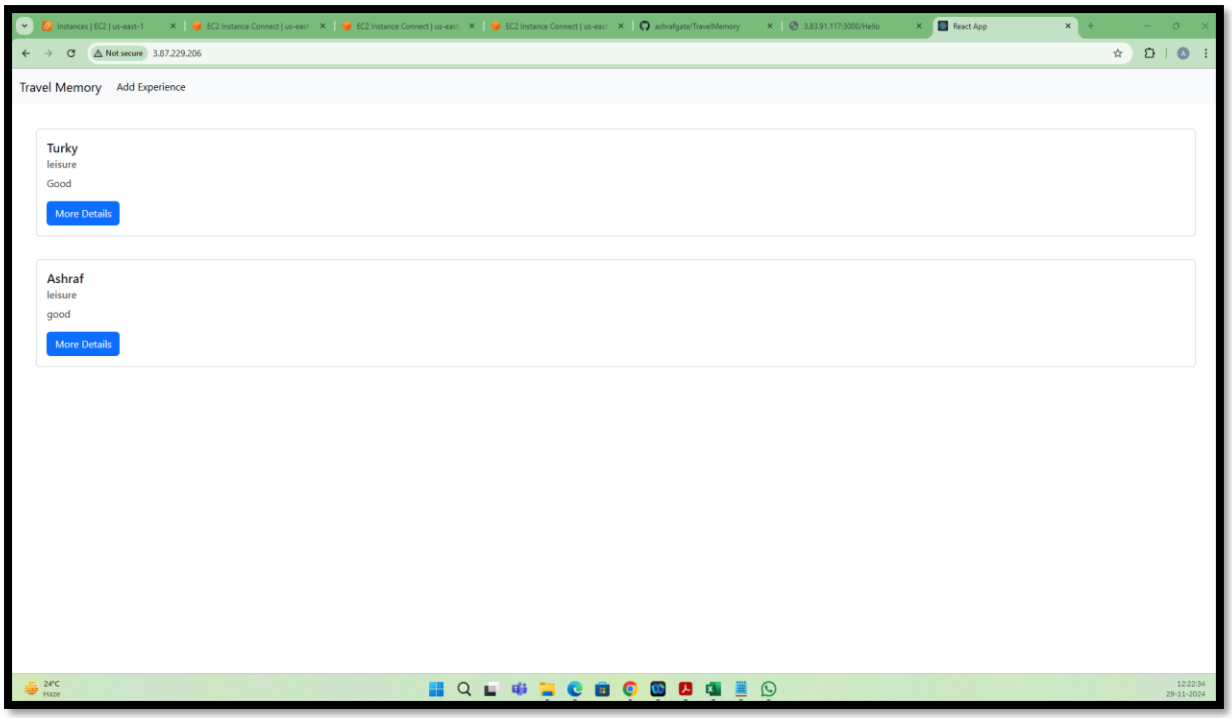
sudo systemctl restart nginx

Install dependencies and start the frontend server:

sudo npm install

sudo npm start

Verify the frontend is running at http://3.87.229.206 in the browser



Step 10: Load Balancer Configuration

Create a target group:

Include both frontend servers (frontend_server001 and frontend_server002).

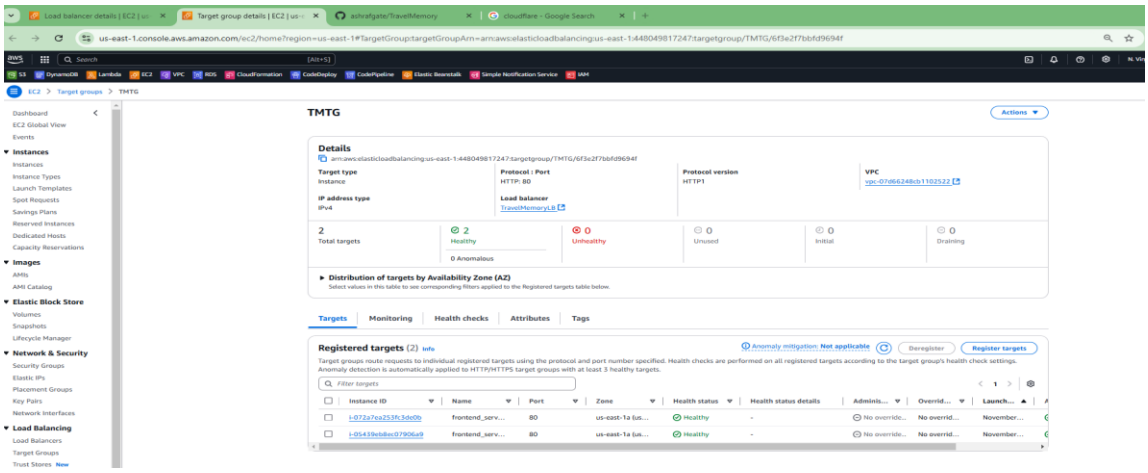
Create an Application Load Balancer (ALB):

Attach the target group.

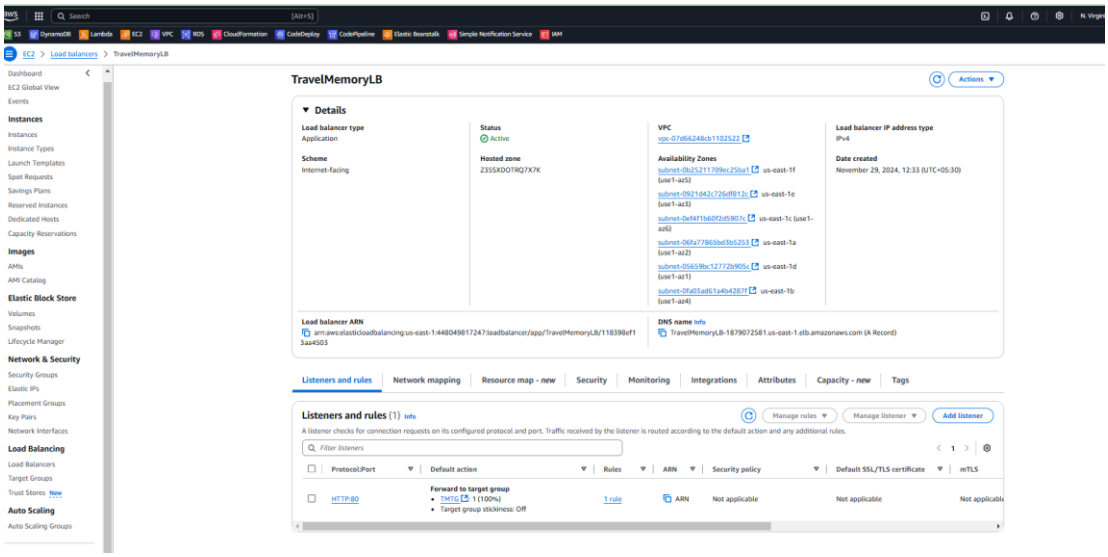
Verify the ALB DNS name and ensure it routes traffic correctly to the frontend servers.

Access the application via the ALB DNS name.

Target Group:



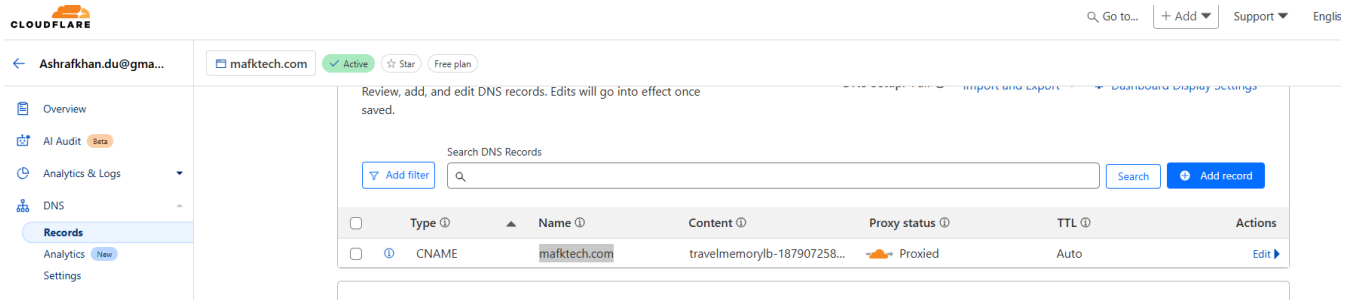
Load Balancer:



Take the DNS name of the Load balancer and run in browser.

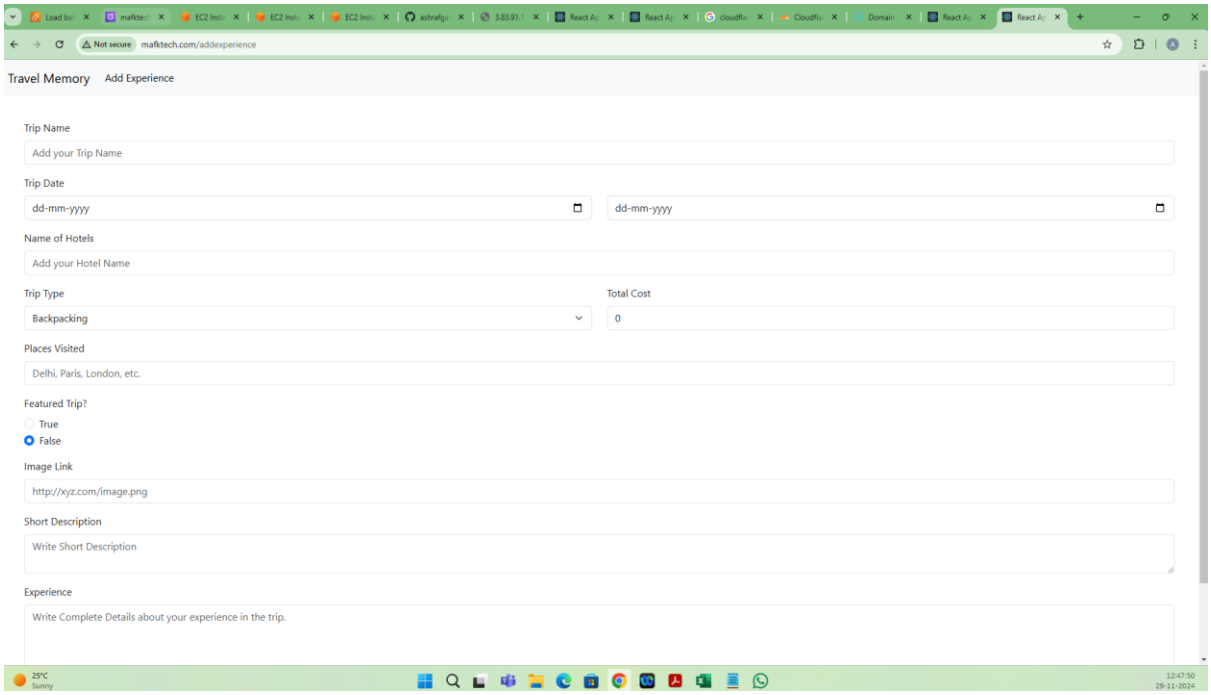
Step 11: Cloudflare Integration

Add Load balancer DNS name in Cloudflare. Type should be CNAME.



After integrating the DNS name of Load balancer with your purchased domain you run your domain name in browser and frontend server will reflect.

My domain name is http://mafktech.com



Conclusion

This document provides a detailed procedure for deploying the TravelMemory MERN stack application on Amazon EC2 instances. By following these steps, you can successfully configure your backend and frontend servers, implement Nginx reverse proxy, integrate with a load balancer, and secure your application with Cloudflare. With this setup, you can enjoy a robust, scalable, and reliable application architecture for your TravelMemory platform.