**Travel Memory Application Deployment  
  
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**](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.

**5. Documentation:**

- Prepare comprehensive documentation detailing each step of the deployment process. Include relevant screenshots to make the process clear and reproducible.

- Design a deployment architecture diagram using [draw.io](https://www.draw.io/) to visualize the flow and connections.

**Expected Deliverables:**

1. A fully functional TravelMemory application hosted on an EC2 instance, accessible via a custom domain.

2. Detailed documentation of the deployment process with appropriate screenshots.

3. A deployment architecture diagram.

**Evaluation Criteria:**

- Accuracy and effectiveness of the deployment.

- Clarity and comprehensiveness of the documentation.

- Adherence to best practices in terms of security, scalability, and resilience.

**Hints:**

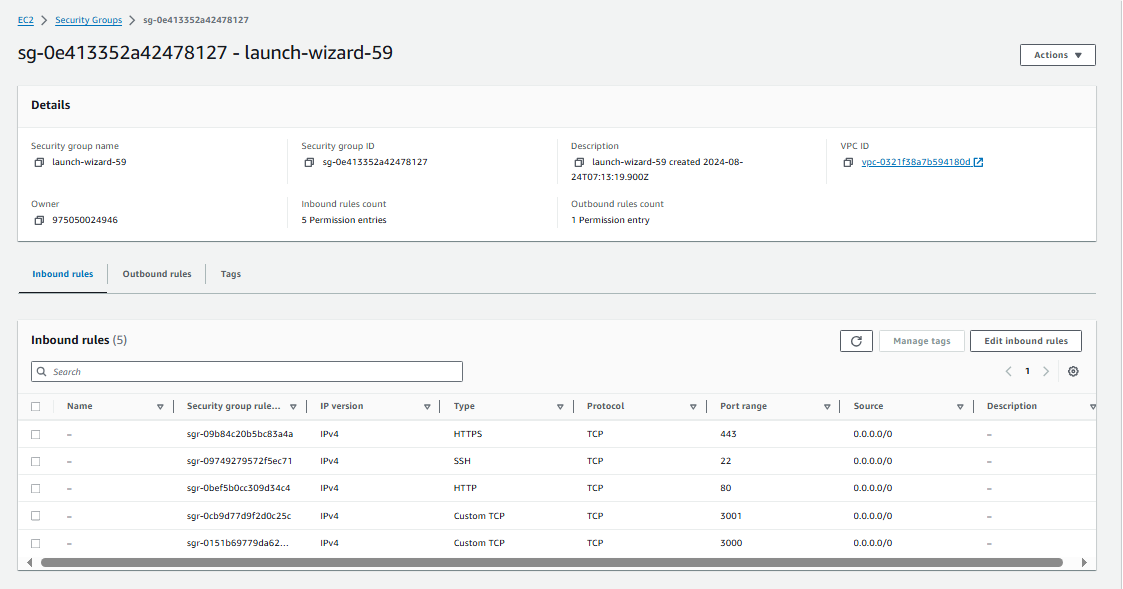
While setting up with Cloudflare, remember that a CNAME record is essential for linking the load balancer endpoint. An A record, on the other hand, connects the EC2 instance via its IP address.

**Steps for deployment process:**

**Step 1:**

Fork the repository<https://github.com/UnpredictablePrashant/TravelMemory.git> toyour own repository **-** [**https://github.com/AkankshaJairath/TravelMemory**](https://github.com/AkankshaJairath/TravelMemory)

**Step 2:**

Create EC2 instance for deployment for TravelMemory. Make sure that port 3001, 80 and 3000 is open in security group.  
  


**Backend configuration:**

* Launch the instance and perform the below configurations  
  Update the package management tool (apt)

sudo apt update

sudo apt-get update

Install the nginx server

sudo apt install nginx

* clone the repository using below command

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

* Create the .env file in the TravelMemory/backend folder and add below details  
    
  MONGO\_URI='mongodb+srv://AkankshaJairath123:AhjunkjbhjRykb@akankshacluster.aulxyf3.mongodb.net/tmbatch7'

PORT=3000

* Install the npm libraries and run the backend

npm install node index.js &

The backend started running at port 3000.  
  


**Frontend configuration:**Navigate to the src/frontend Directory in TravelMemory app

cd TravelMemory/frontend/src

Open the url.js file and provide the backend url.

"http://localhost:3000"

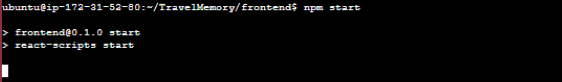
Note:- localhost with instance ip address  
  

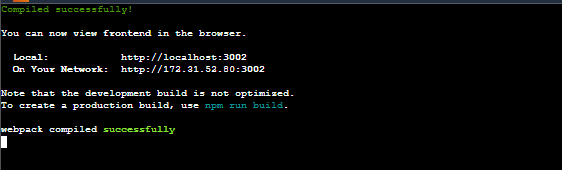

Install the npm libraries and run the Frontend.

cd ..

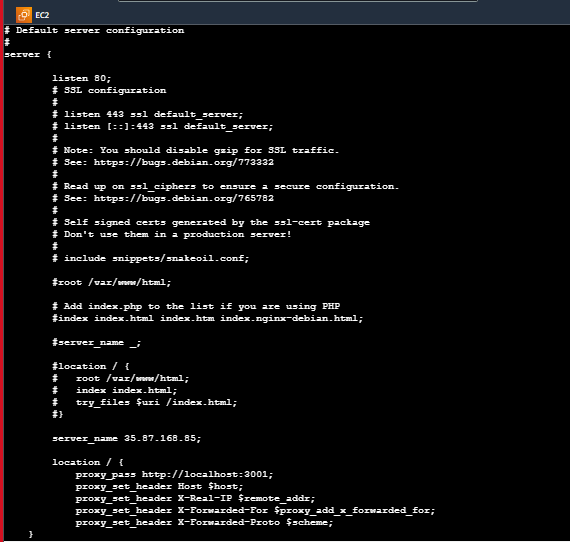
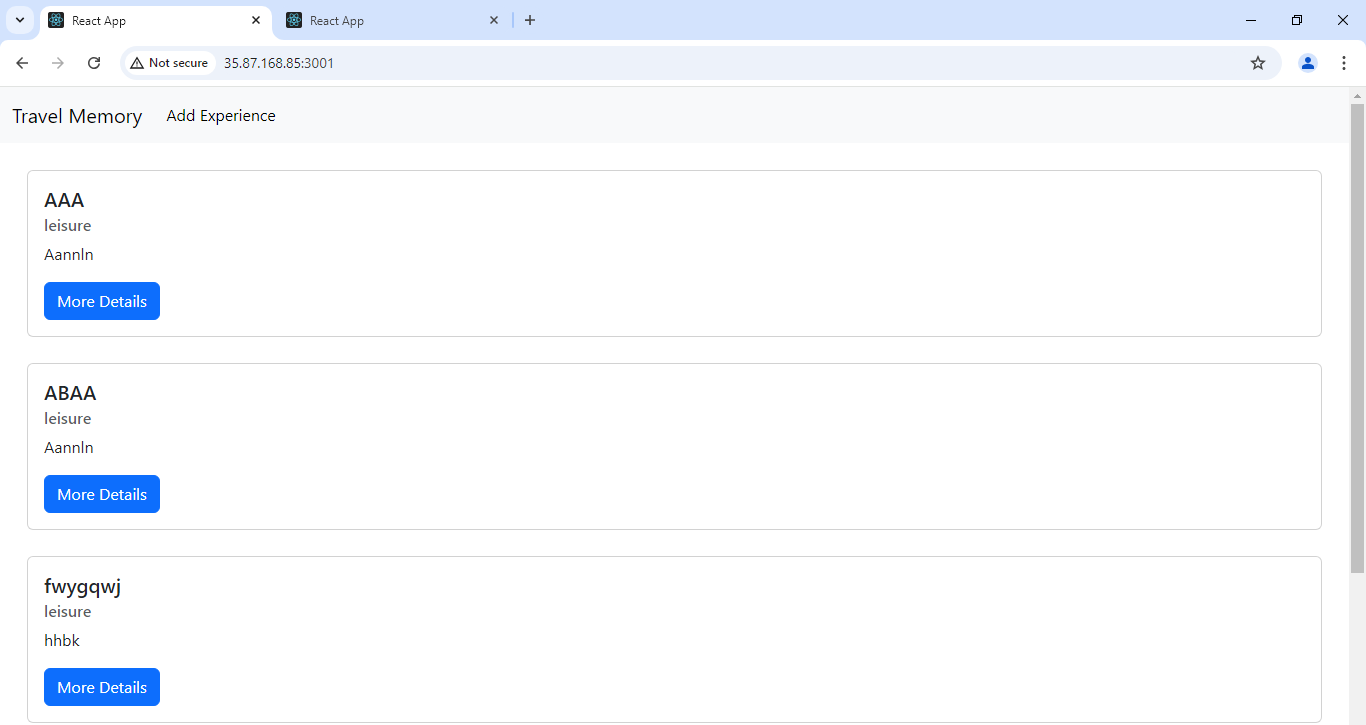
npm install

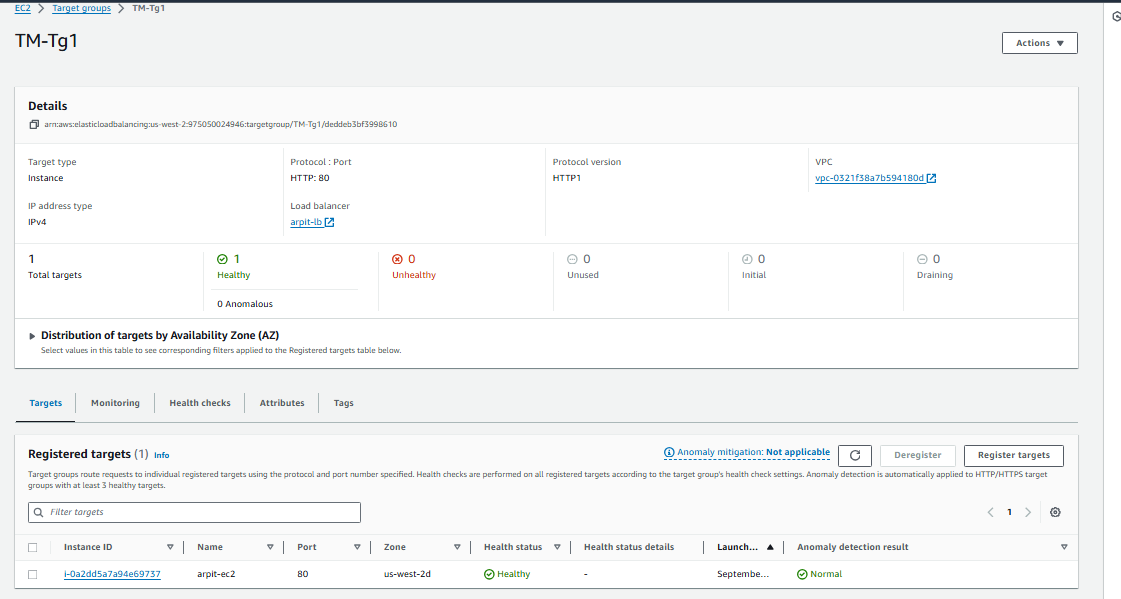
npm start

Frontend started  


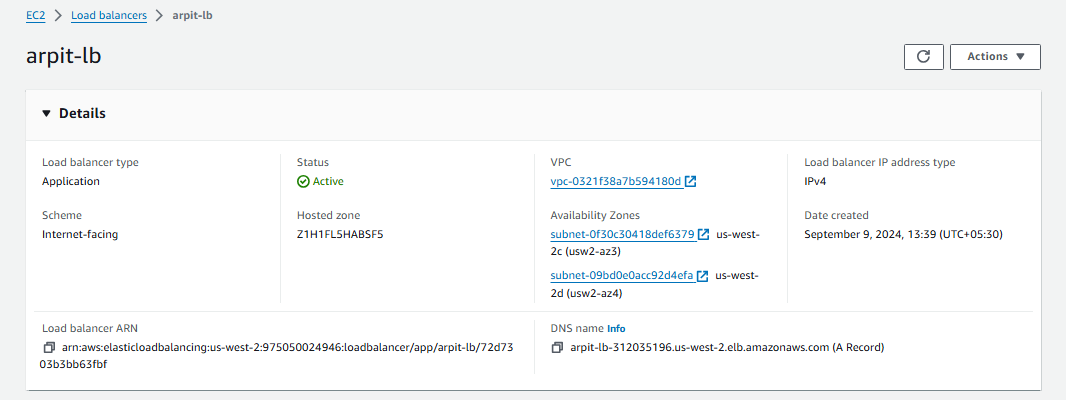
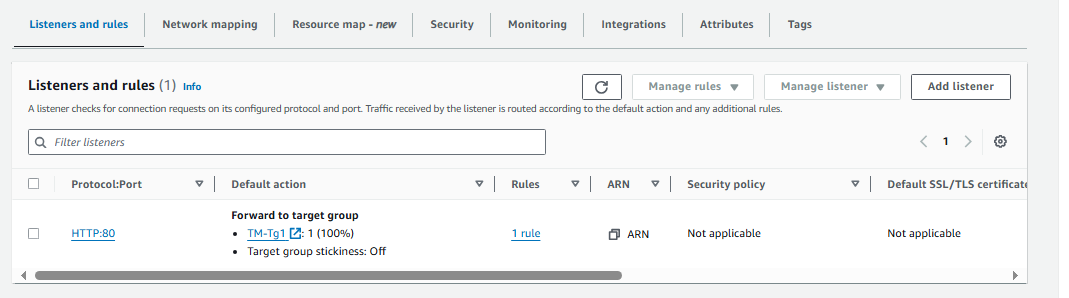
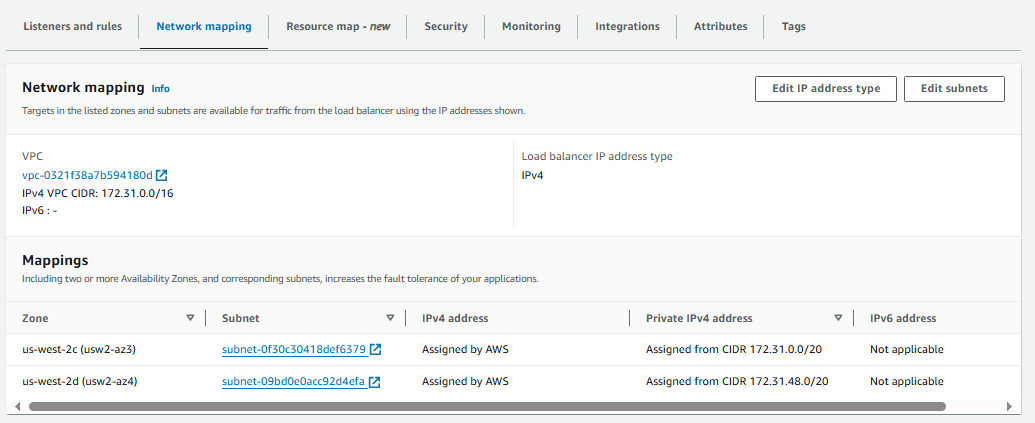
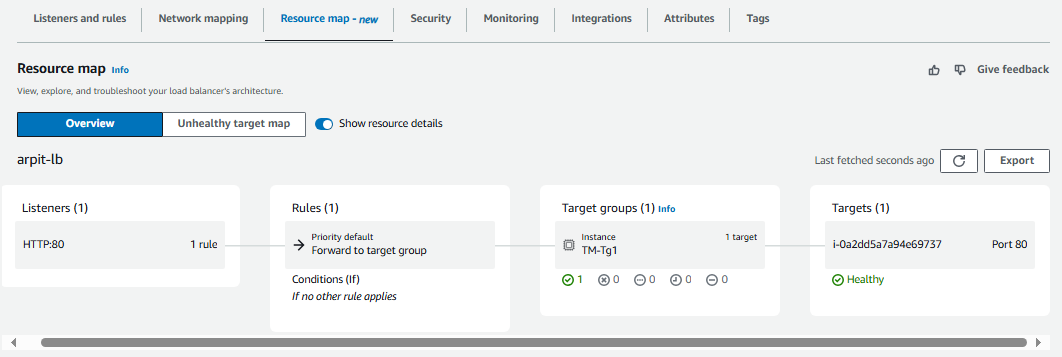
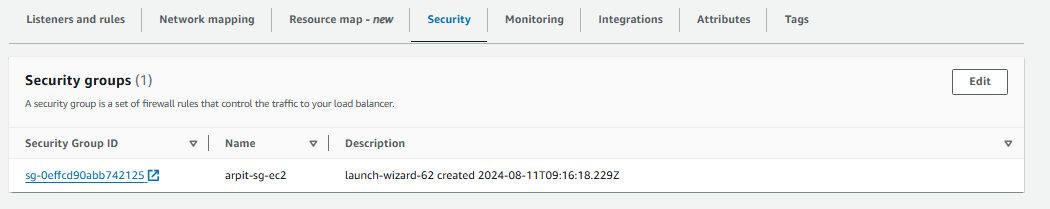
  
  
  
**Configuring the path forwarding**

Navigate to sites-available path in nginx and add the proxy pass details in the default named file.

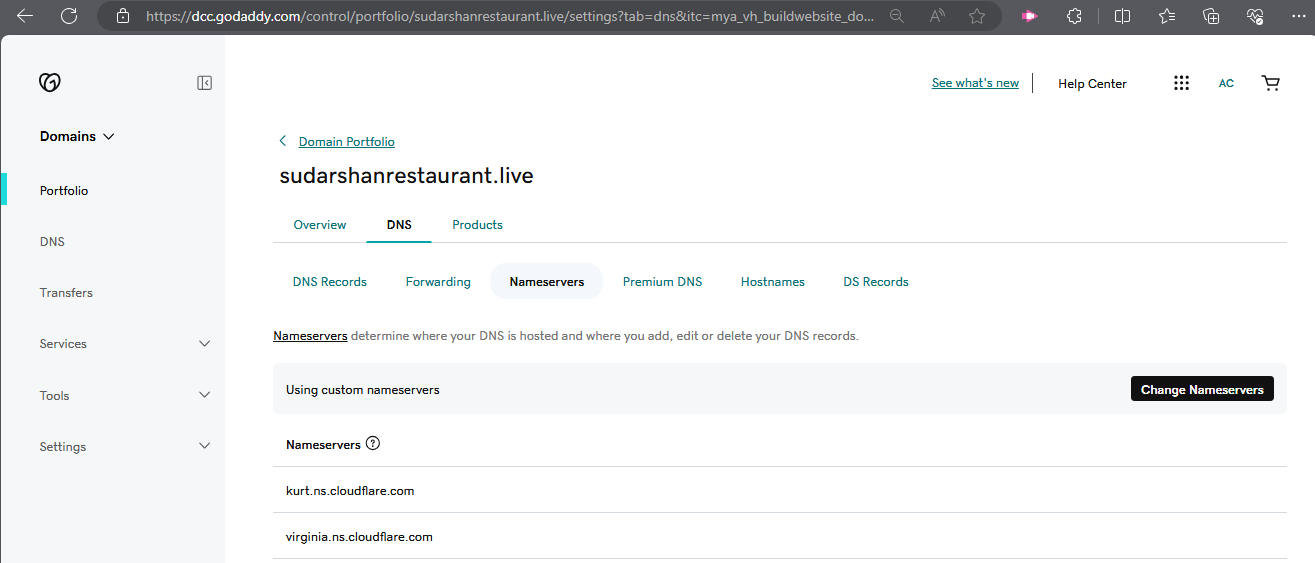
sudo nano /etc/nginx/sites-available/default  
  
  
Delete the build folder and again Build the frontend  
The Working App shows below:  
  


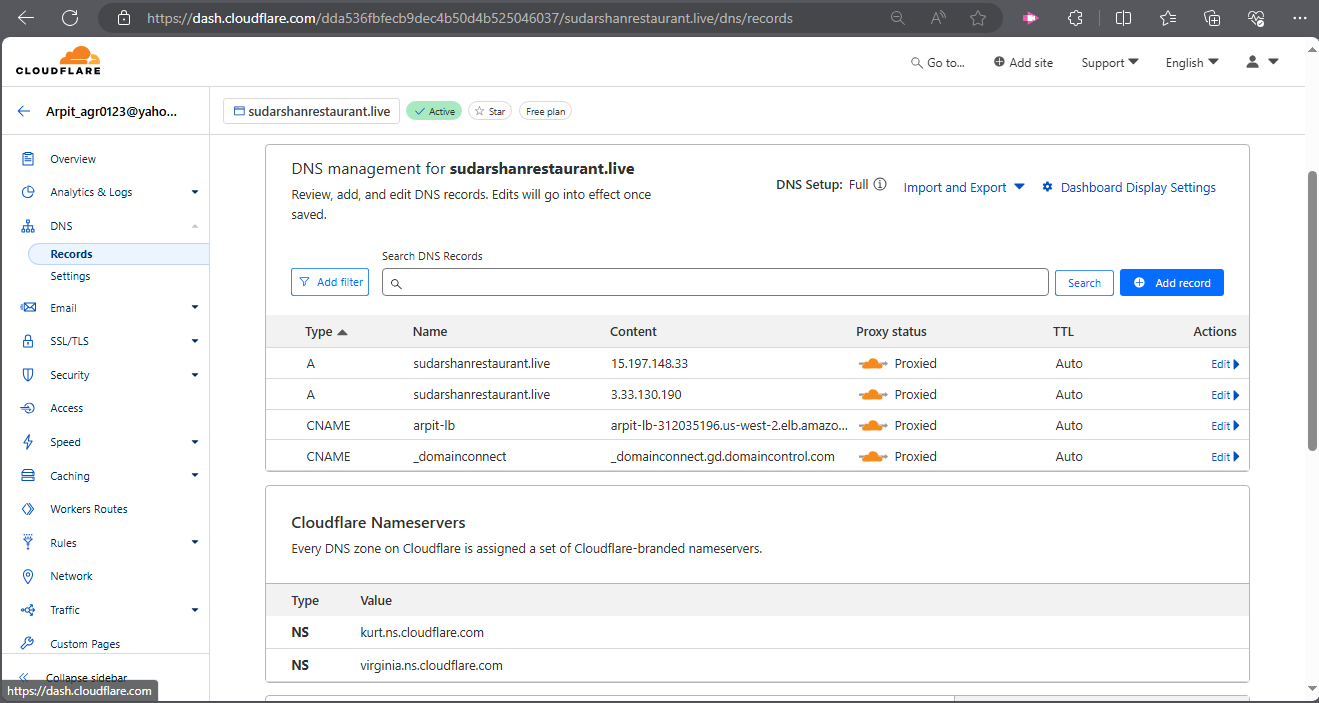
**Load Balancer Configuration:  
  
For LoadBalancer, First, we create a Target Group**

**Then Attach the TG for LoadBalancer.:**

  
  
  
  
  
  
**Configuring Domain Setup**

**Created a Cname record in CloudFare and added the DNS name from the load balancer**

**Note: MakeSure DNS has the correct nameserver configured**

  
  
  
Now the Application working fine with DNS Name:  
  
