# Project Summary

**Introduction:**

The Travel Memory 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 Travel Memory 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.

**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 Travel Memory 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 TO PRODUCE

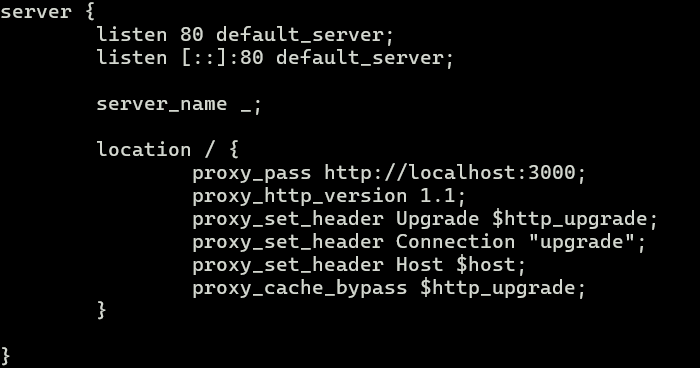
1. Start an ec2 instance

* Details
* Ubuntu 24.04 LTS with **MEMORY of 1 GB** and **STORAGE of 8 GB**
* Set key and download it as PEM [ Privacy Enhanced mail] save it
* Use chmod 600 to <pem file name>
* Use this command to connect to ec2

\* **ssh -I <pem name> ubuntu@<public\_ip\_address>\***

* Now after connected install nginx
  + **sudo apt update**
  + **curl -sL** [**https://deb.nodesource.com/setup\_18.x -o/tmp/nodesource\_setup.sh**](https://deb.nodesource.com/setup_18.x%20-o/tmp/nodesource_setup.sh)
  + **sudo bash /tmp/nodesource\_setup.sh**
  + **sudo apt install nodejs**
  + **node -v**
  + **history**
* Configure firewall
  + **ufw allow ‘Nginx Full’**
  + **ufw enable**
* Clone the repository
  + git clone <https://github.com/UnpredictablePrashant/TravelMemory>
  + change directory to TravelMemory/backend
  + create .env file
  + Fill the mongo db url and the port to 3001
  + Do npm install
  + Now go to directory of TravelMemory/Frontend
  + Do npm install

**Configuration of nginx**



**Instance Settings**

Edit inbound rules

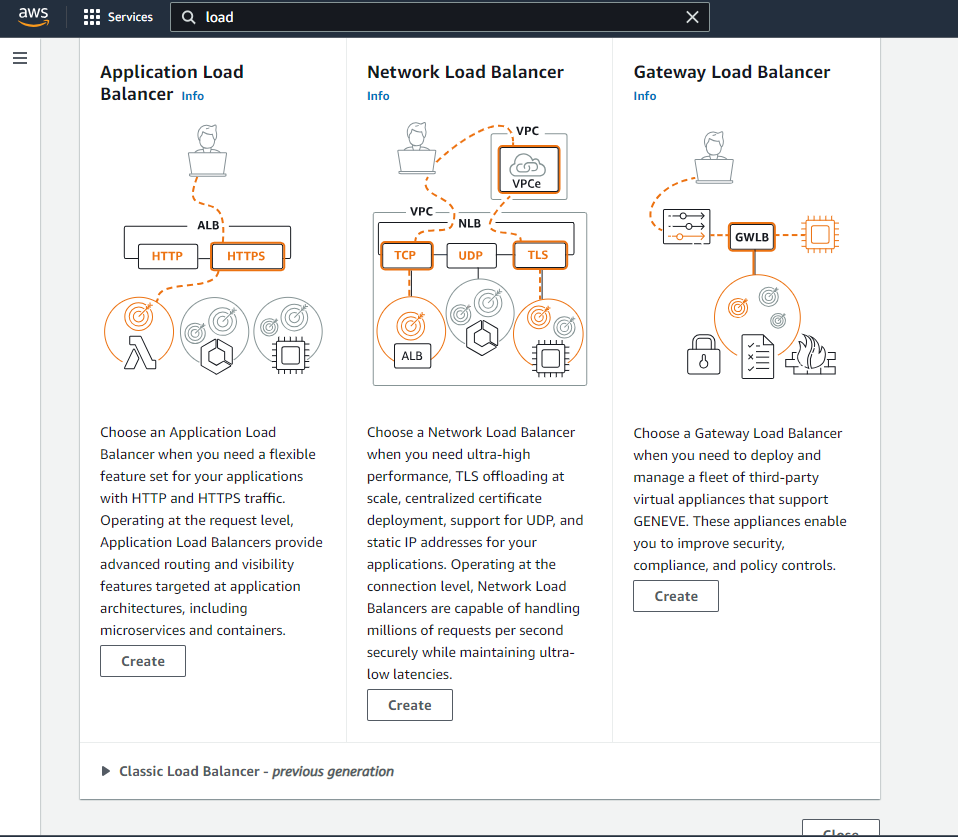
* Open 3000, 3001 ports

A screenshot of a computer

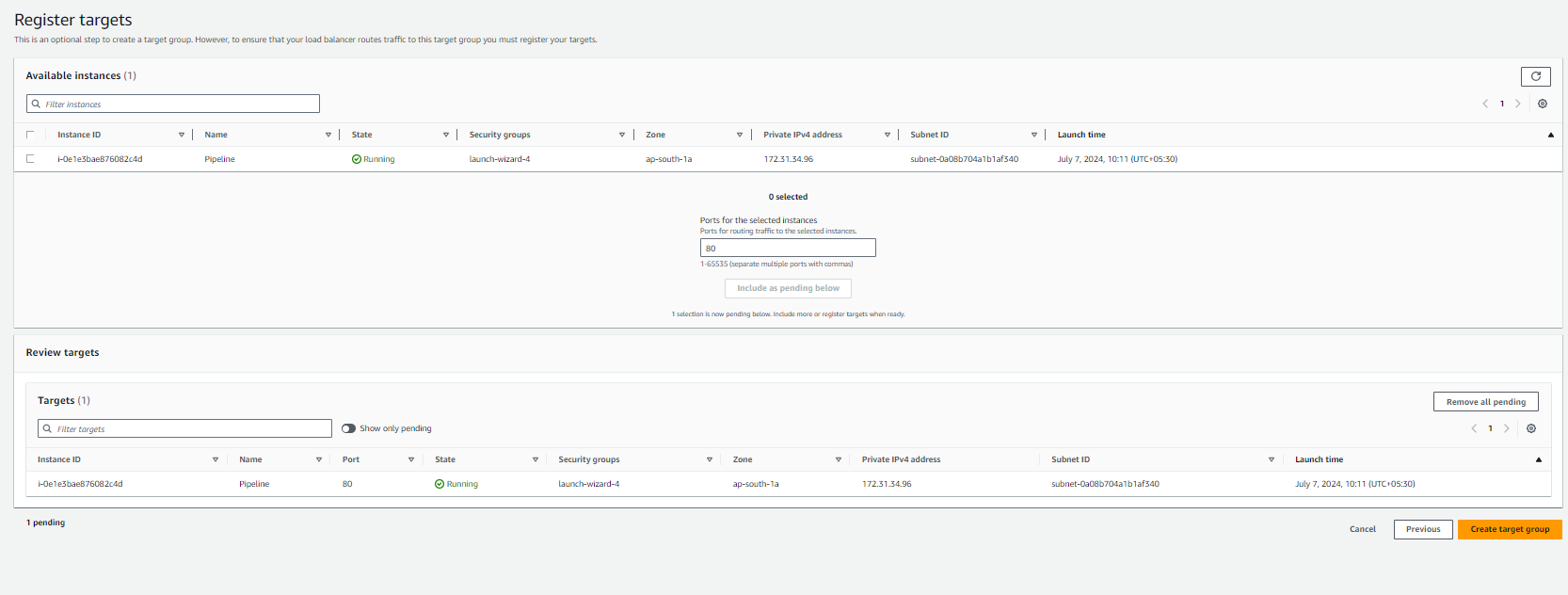
Description automatically generated

Setting Load balancers

* Go to aws console open application load balancers

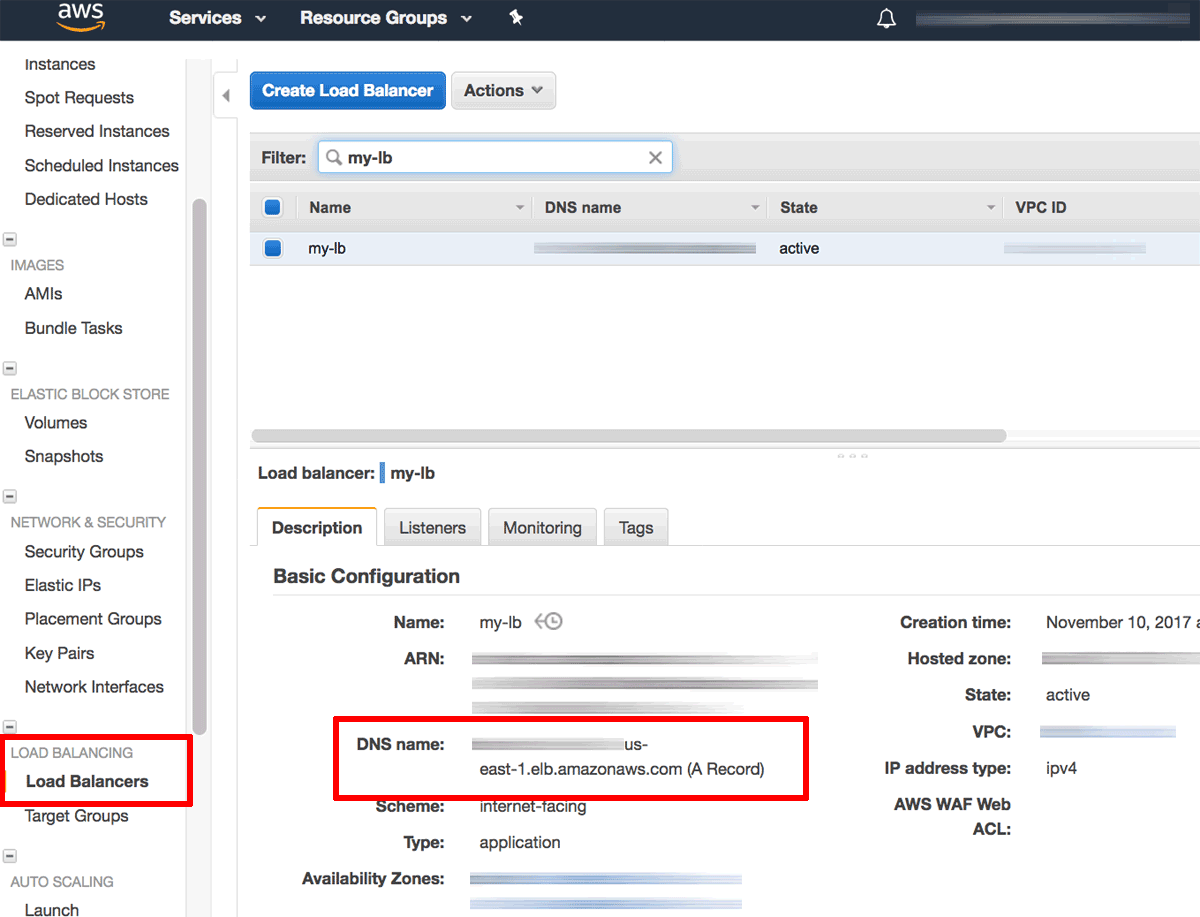


Set Load balancers also create a target group selecting your instance



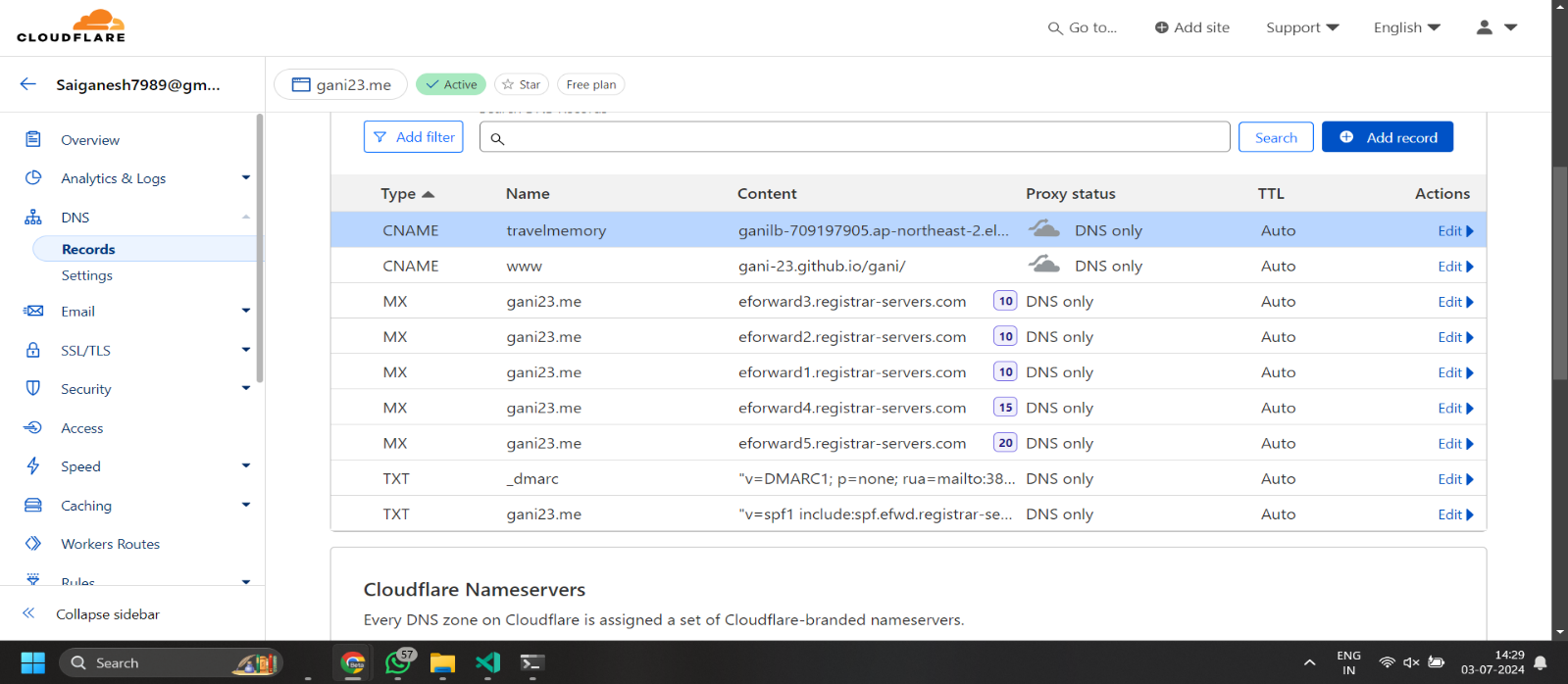
**Check for DNS NAME**

* Wait for few minutes while we get load balancers provisioned for us
* Now copy the DNS target, DNS name



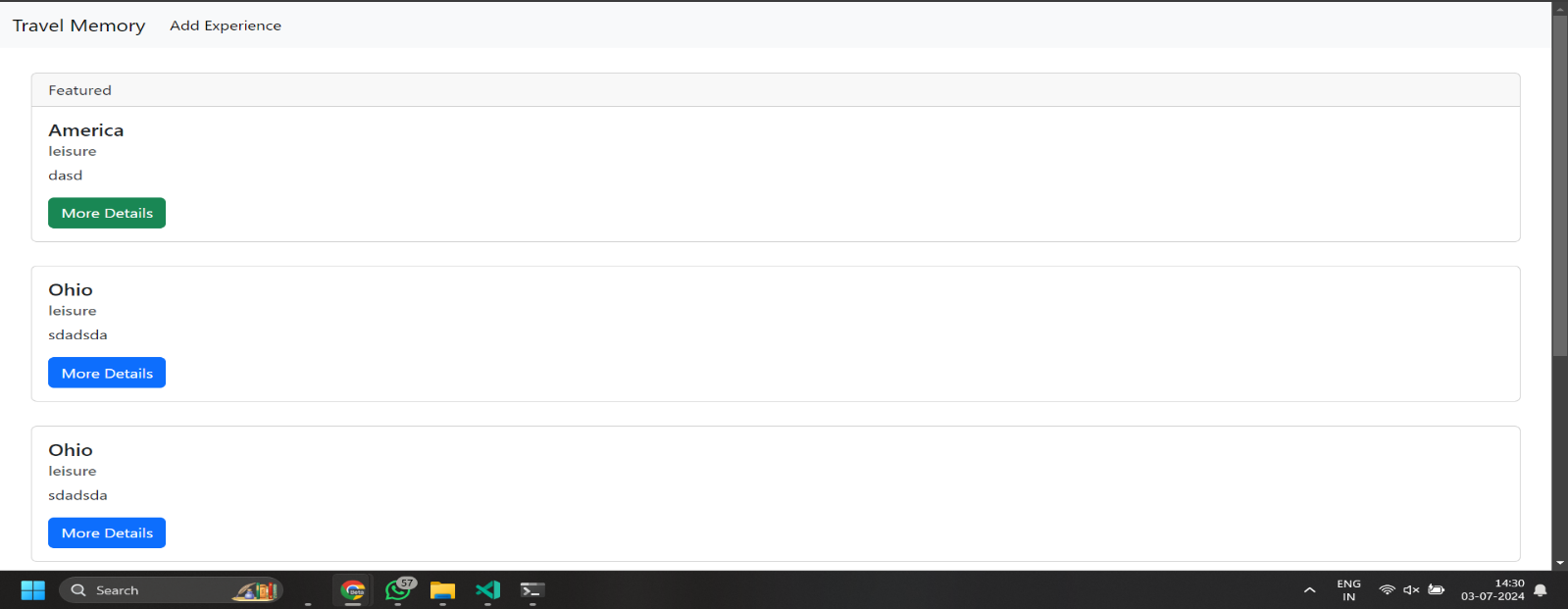
**Setting up with cloudflare**

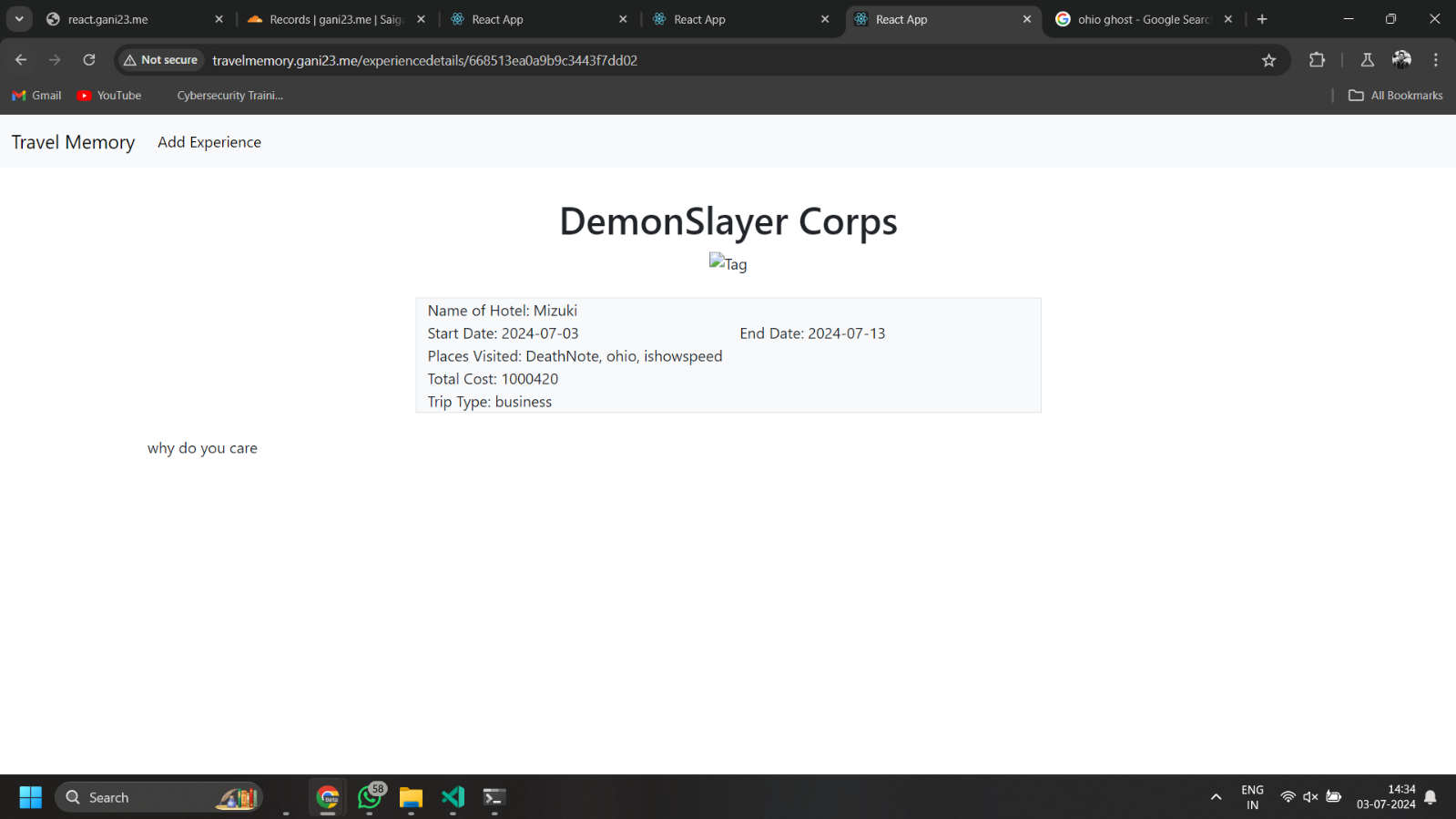
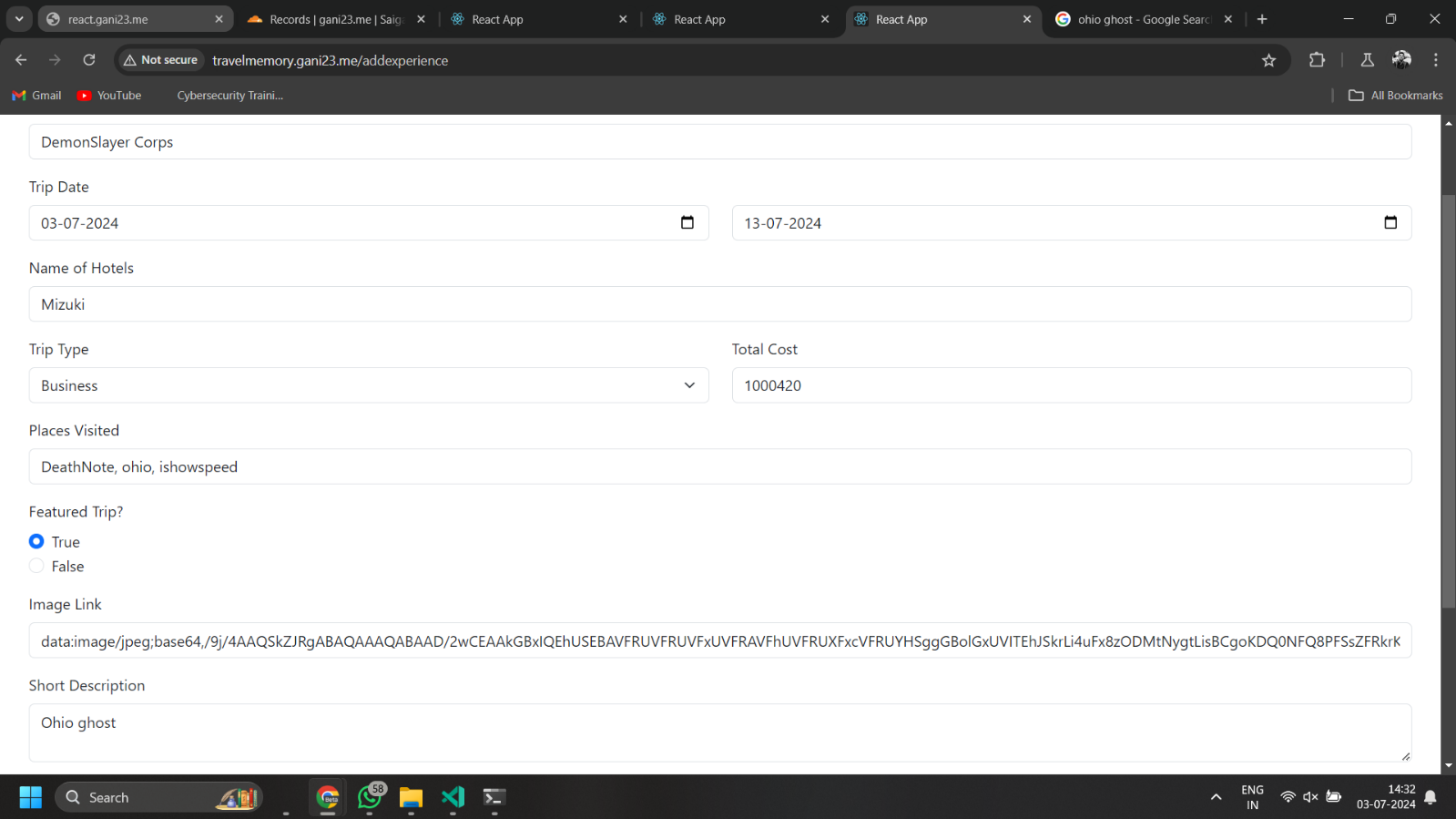
* Click on add record
* Choose CName
* Name could be of your choice and the paste the copied dns name In the content and keep the proxies off
* In the end it should look something like this



Wait for few moments

**Results**



****