**GIthub Link:**

**Description**

**Objective:**

Gain practical experience in deploying a MERN stack application on AWS using infrastructure automation with Terraform and configuration management with Ansible.

**MERN Application to use:**<https://github.com/UnpredictablePrashant/TravelMemory>

**Tasks:**

**Part 1: Infrastructure Setup with Terraform**

1. AWS Setup and Terraform Initialization:

   - Configure AWS CLI and authenticate with your AWS account.

   - Initialize a new Terraform project targeting AWS.

2. VPC and Network Configuration:

   - Create an AWS VPC with two subnets: one public and one private.

   - Set up an Internet Gateway and a NAT Gateway.

   - Configure route tables for both subnets.

3. EC2 Instance Provisioning:

   - Launch two EC2 instances: one in the public subnet (for the web server) and another in the private subnet (for the database).

   - Ensure both instances are accessible via SSH (public instance only accessible from your IP).

4. Security Groups and IAM Roles:

   - Create necessary security groups for web and database servers.

   - Set up IAM roles for EC2 instances with required permissions.

5. Resource Output:

   - Output the public IP of the web server EC2 instance**.**

**Part 2: Configuration and Deployment with Ansible**

1. Ansible Configuration:

   - Configure Ansible to communicate with the AWS EC2 instances.

2. Web Server Setup:

   - Write an Ansible playbook to install Node.js and NPM on the web server.

   - Clone the MERN application repository and install dependencies.

3. Database Server Setup:

   - Install and configure MongoDB on the database server using Ansible.

   - Secure the MongoDB instance and create necessary users and databases.

4. Application Deployment:

   - Configure environment variables and start the Node.js application.

   - Ensure the React frontend communicates with the Express backend.

5. Security Hardening:

   - Harden the security by configuring firewalls and security groups.

   - Implement additional security measures as needed (e.g., SSH key pairs, disabling root login).

**Deliverables:**

* Terraform scripts for AWS infrastructure setup.
* Ansible playbooks for configuration and deployment of the MERN application.
* A detailed report documenting the implementation process, including how the application components interact with each other and the infrastructure.
* Screenshots or a video recording demonstrating the working MERN application.

**Submission Instructions:**

To submit your assignment, please follow these guidelines:

- Ensure that your assignment is fully completed.

- Push your code/assignment to a GitHub repository.

- Share the repository link by including it in a text, Word, or PDF file format.

Submit the file/text containing the repository link via Vlearn.

**Part 1: Infrastructure Setup with Terraform**

1. AWS Setup and Terraform Initialization:

   - Configure AWS CLI and authenticate with your AWS account.

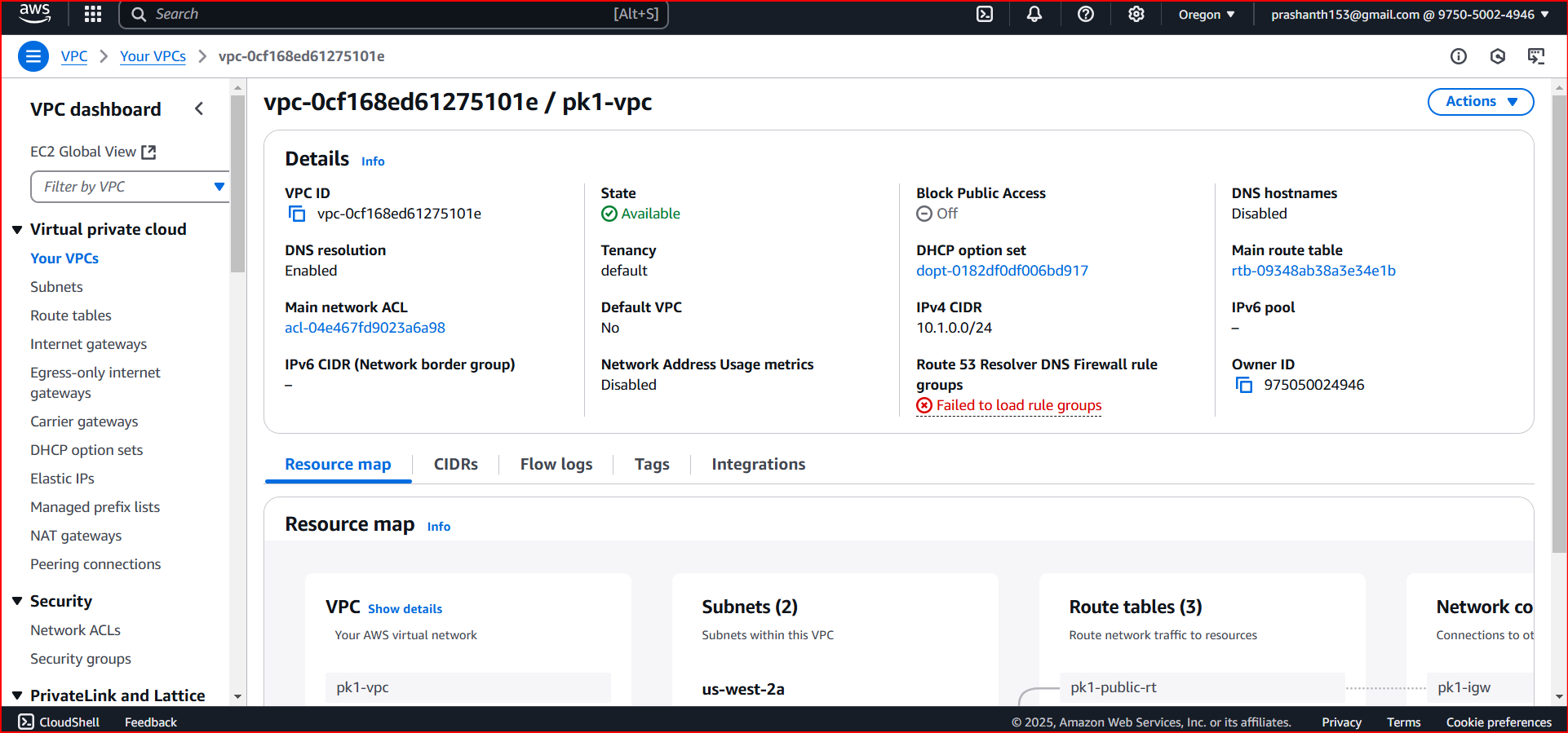
   - Initialize a new Terraform project targeting AWS.

***Done using the Terraform script as shown in the GITHUB repository***

2. VPC and Network Configuration:

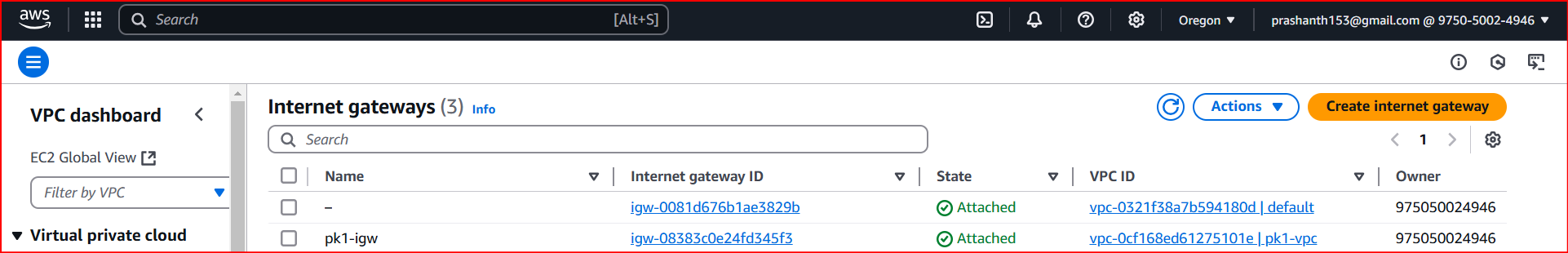
   - Create an AWS VPC with two subnets: one public and one private.

***Done pk1-vpc created with 2 subnets 1 provate and 1 public***

******

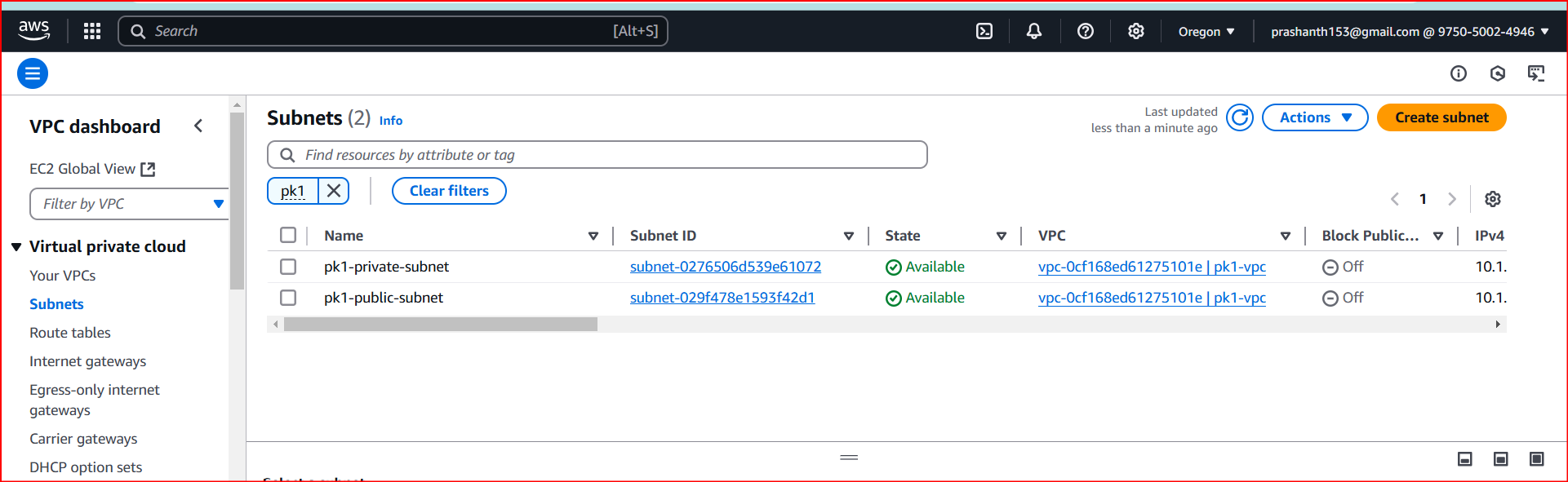
   - Set up an Internet Gateway and a NAT Gateway.

D***one***



   - Configure route tables for both subnets.

D***one***

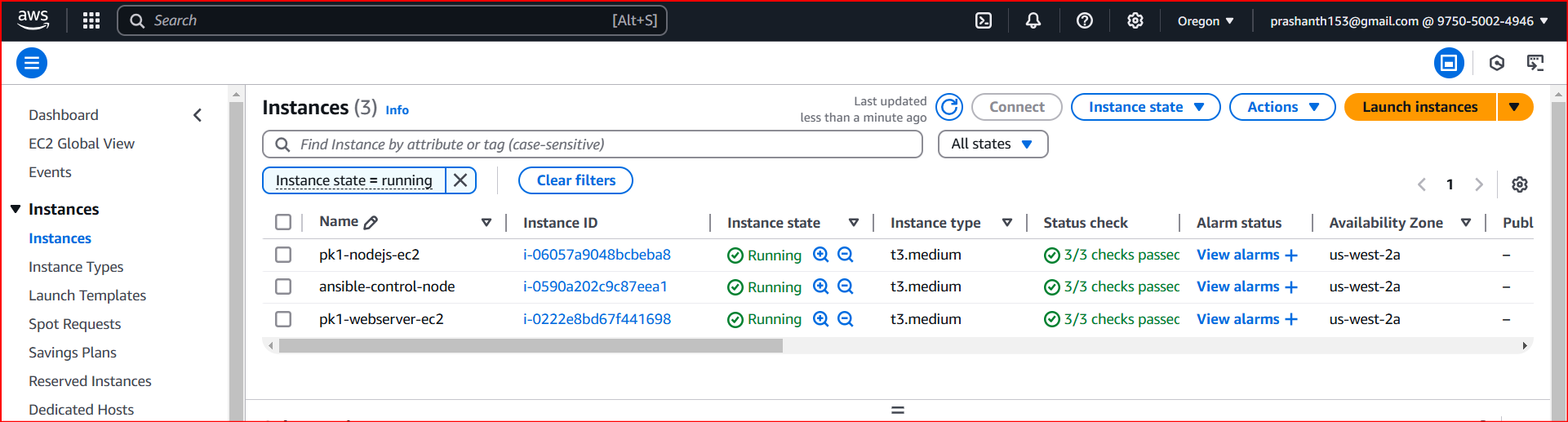


3. EC2 Instance Provisioning:

   - Launch two EC2 instances: one in the public subnet (for the web server) and another in the private subnet (for the database).

   - Ensure both instances are accessible via SSH (public instance only accessible from your IP).

All the 3 isntance are launches as shown below



4. Security Groups and IAM Roles:

   - Create necessary security groups for web and database servers.

***Done in the terraform script itself***

   - Set up IAM roles for EC2 instances with required permissions.

5. Resource Output:

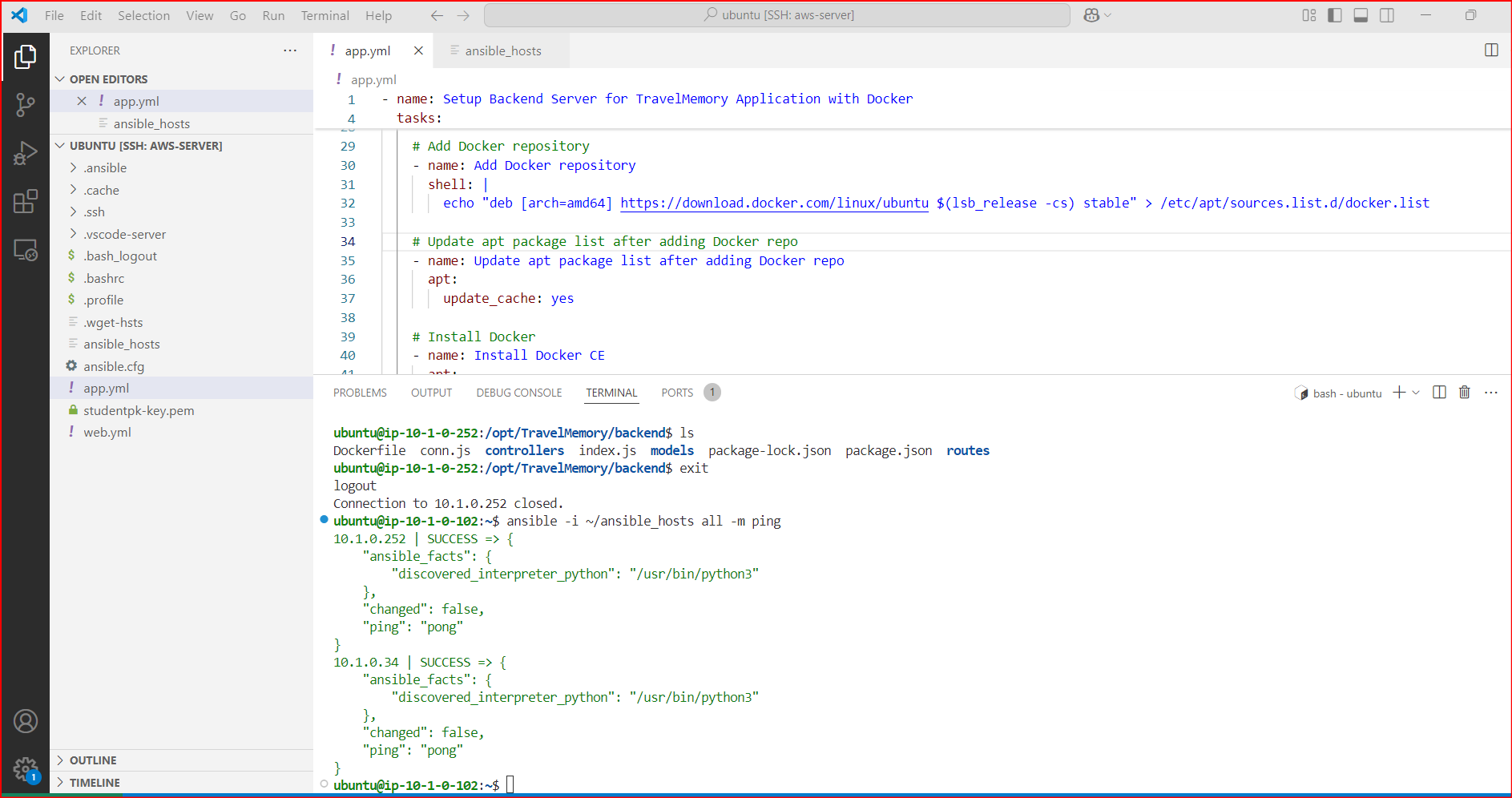
   - Output the public IP of the web server EC2 instance**.**

**Part 2: Configuration and Deployment with Ansible**

1. Ansible Configuration:

   - Configure Ansible to communicate with the AWS EC2 instances.

***Created ansible host file and able to ping to the ec2 instances***

******

2. Web Server Setup:

   - Write an Ansible playbook to install Node.js and NPM on the web server.

   - Clone the MERN application repository and install dependencies.

***Created web.yaml and ran it using ansible. It is is able to update the REACT server with all the dependencies.***

3. Database Server Setup:

   - Install and configure MongoDB on the database server using Ansible.

   - Secure the MongoDB instance and create necessary users and databases.

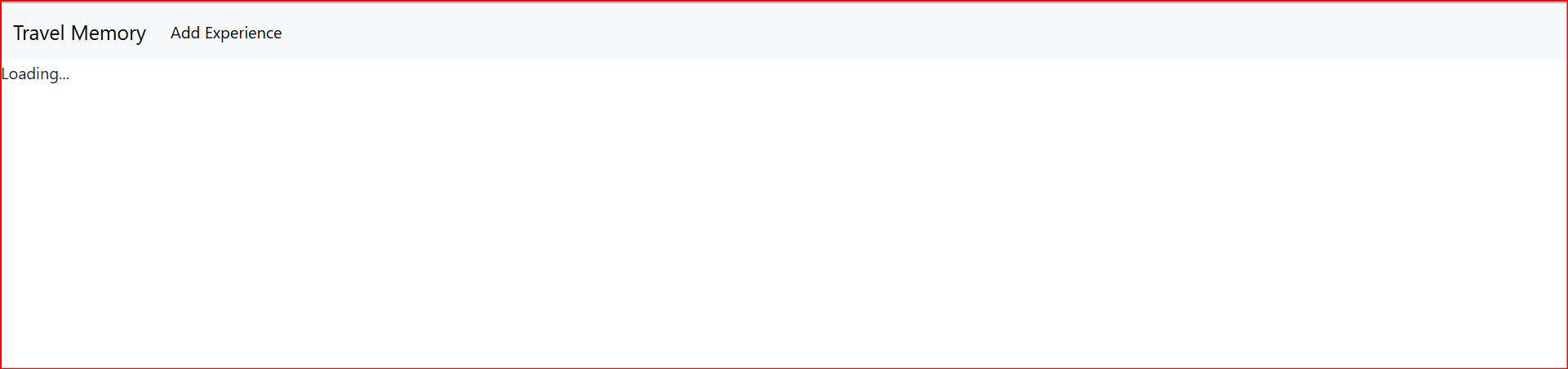
***Created app.yaml and ran it using ansible. It is is able to update the REACT server with all the dependencies.***

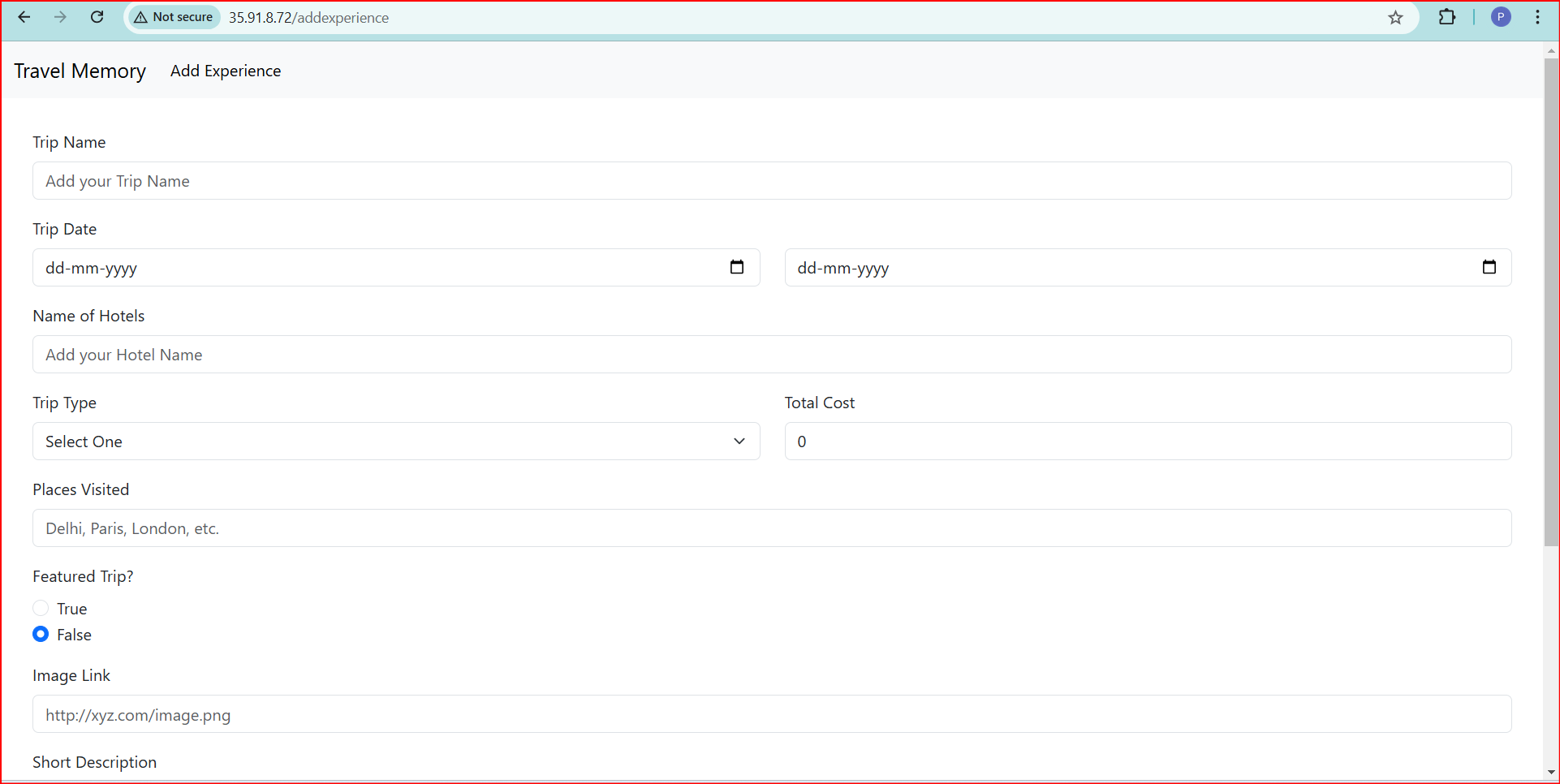
4. Application Deployment:

   - Configure environment variables and start the Node.js application.

   - Ensure the React frontend communicates with the Express backend.

***Application deployment is complete and able to see the following image.***

******



5. Security Hardening:

   - Harden the security by configuring firewalls and security groups.

   - Implement additional security measures as needed (e.g., SSH key pairs, disabling root login).