

# **Graded Project**

**on**

# **“Travel Memory”**

# **Application Deployment**

**Course 6: AWS Cloud Deployment | DevOps B4**

Access the complete codebase of the TravelMemory application from the provided GitHub link: [GitHub - 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 frontend 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 frontend.

### 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.

# TravelMemory Application Deployment

Capture and cherish your travel adventures.

The TravelMemory application is one of the learning session materials that helped us in the learning process for DevOps from Hero Vired. Prashant Kumar Dey [Profile Link: <https://github.com/UnpredictablePrashant>] develops this application.

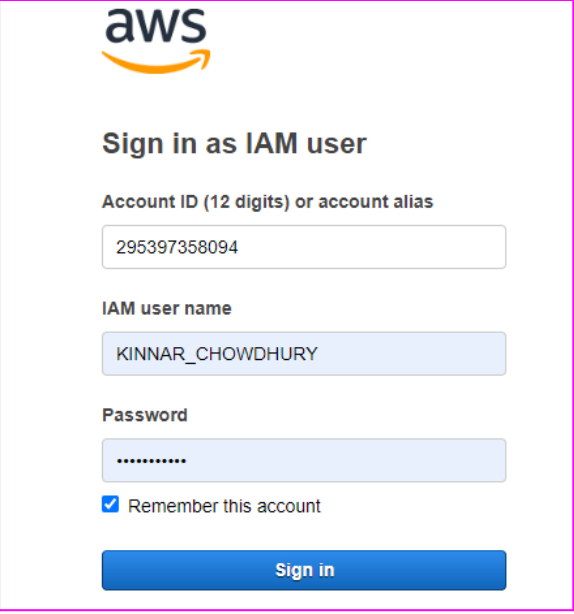
TravelMemory appears to be an application designed to help users document and relive their travel experiences. This app lets you create a digital journal of your trips, complete with photos, locations, expenses, and descriptions. You can even share your travel stories with others!

Detailed information on the [deployment process of the TravelMemory app](#) is referenced below;

Initially, for this project, I have taken two EC2 machines from Amazon Web Services (AWS).

The process regarding commissioning an EC2 instance is mentioned as follows:

To set up an AWS EC2, first, you have to log in to your AWS account and then proceed to create an EC2 Instance

A screenshot of the AWS IAM user sign-in page. The page features the AWS logo at the top left. Below it, the heading "Sign in as IAM user" is displayed. The form includes three input fields: "Account ID (12 digits) or account alias" with the value "295397358094", "IAM user name" with the value "KINNAR\_CHOWDHURY", and "Password" with masked characters. A checkbox labeled "Remember this account" is checked. A blue "Sign in" button is at the bottom.

aws

**Sign in as IAM user**

Account ID (12 digits) or account alias

295397358094

IAM user name

KINNAR\_CHOWDHURY

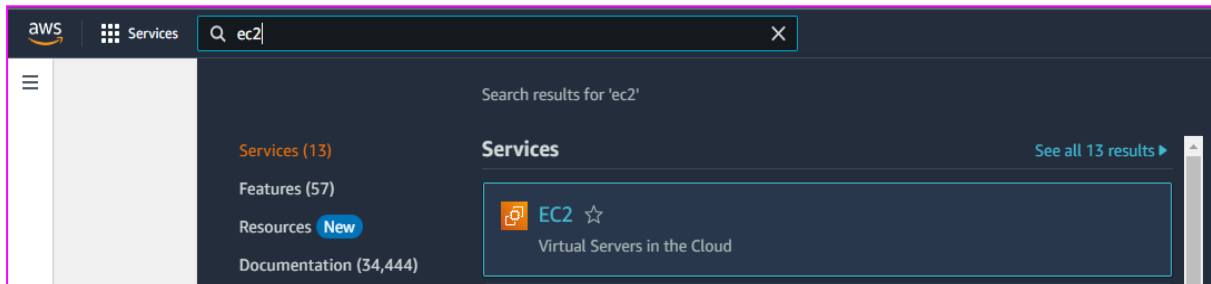
Password

.....

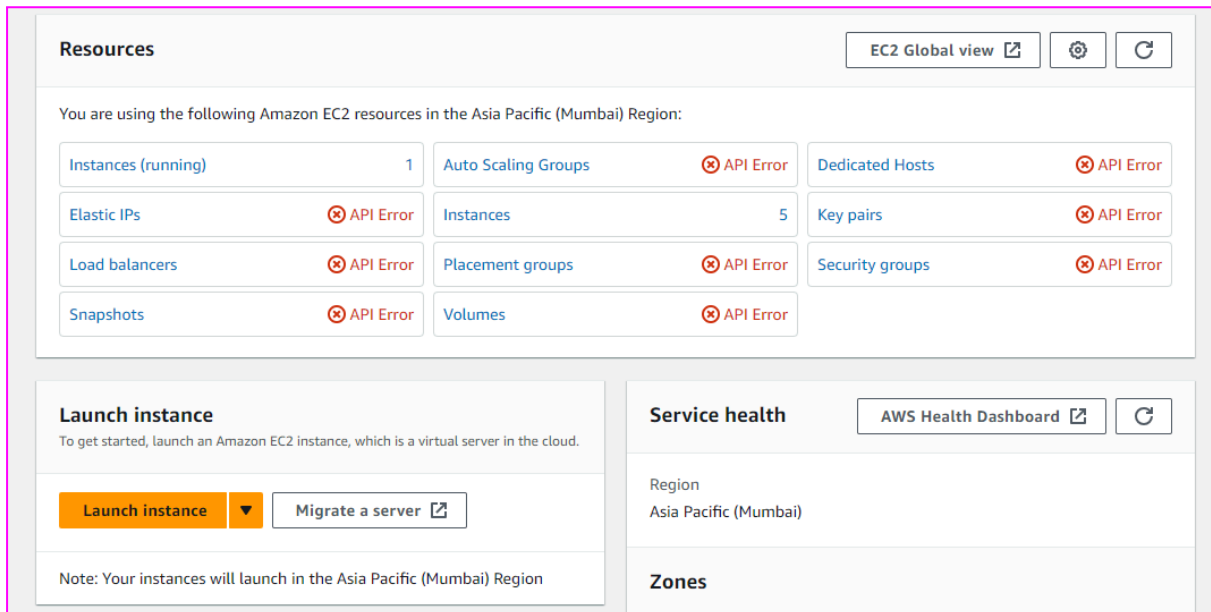
☒ Remember this account

Sign in

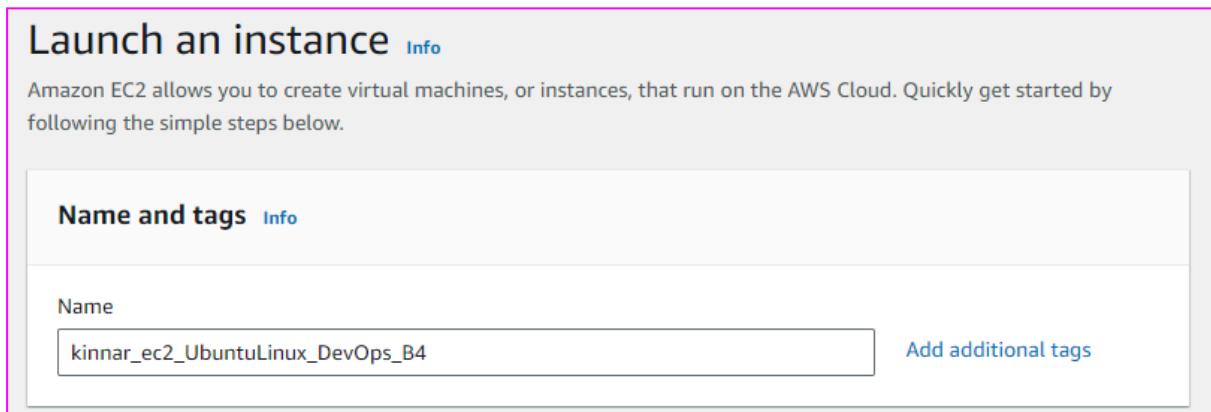
1. Search for EC2 in the AWS console dashboard and click on it.



2. Click on Launch EC2 Instances



3. Provide Name and Tags.



#### 4. Choose an OS Image for Amazon Machine Image.

▼ **Application and OS Images (Amazon Machine Image)** [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Q Search our full catalog including 1000s of application and OS images

Recents

My AMIs

Quick Start

Amazon Linux

aws

macOS

Mac

Ubuntu

ubuntu

Windows

Microsoft

Red Hat

Red Hat

SUSE Linux

SUS

Q

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

Free tier eligible

ami-05fb0b8c1424f266b (64-bit (x86)) / ami-0748d13ffbc370c2b (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2023-12-07

Architecture

AMI ID

64-bit (x86)

ami-05fb0b8c1424f266b

Verified provider

#### 5. Choose Instance Type.

▼ **Instance type** [Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true

On-Demand Linux base pricing: 0.0116 USD per Hour

On-Demand SUSE base pricing: 0.0116 USD per Hour

On-Demand Windows base pricing: 0.0162 USD per Hour

On-Demand RHEL base pricing: 0.0716 USD per Hour

All generations

Compare instance types

Additional costs apply for AMIs with pre-installed software

SUBMITTED BY- KINNAR CHOWDHURY, Hero Vired, DevOps, Batch 4


6. You can use an existing key pair or create an EC2 key pair

▼ Key pair (login) Info

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

kinnar\_ec2\_key ▼

 Create new key pair

Create key pair

Key pair name

Key pairs allow you to connect to your instance securely.

kinnar\_ec2\_key

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type



☐ RSA  
RSA encrypted private and public key pair

☒ ED25519  
ED25519 encrypted private and public key pair

Private key file format

☒ .pem  
For use with OpenSSH

☐ .ppk  
For use with PuTTY

 When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn more](#) 

Cancel

Create key pair

7. Set up Network settings as per your requirements.

▼ Network settings Info

Edit

Network Info

vpc-02c848fd2d7a8634e

Subnet Info

No preference (Default subnet in any availability zone)

Auto-assign public IP Info

Enable

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

We'll create a new security group called 'launch-wizard-14' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere  
0.0.0.0/0

☒ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

⚠ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

×

8. Allocate storage for your ec2

▼ Configure storage Info

Advanced

1x 8 GiB gp3

Root volume (Not encrypted)

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

×

Add new volume

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

⊗ Failed to fetch policy coverage

↻

0 x File systems

Edit

9. Now review your settings and click on Launch Instance and wait for some time to be provisioned by AWS.

### ▼ Summary

Number of instances

Info

#### Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...[read more](#)  
ami-05fb0b8c1424f266b

#### Virtual server type (instance type)

t2.micro

#### Firewall (security group)

New security group

#### Storage (volumes)

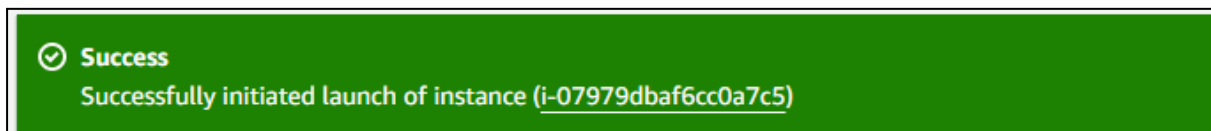
1 volume(s) - 8 GiB

Cancel

Launch instance

[Review commands](#)

10. After successfully launching, Check the EC2 running dashboard to get the EC2 IP and other details.



Instances (1/1) Info

Any state

Clear filters

1

⌵

✓	Name ↗	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
✓	kinnar_ec2_UbuntuLinux_DevOps_...	i-07979dbaf6cc0a7c5	Running	t2.micro	Initializing	User: arn:aws:iam::295397358094:user/KINNAR_CHOWDHURY is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: * because no identity-based policy allows the compute-optimizer:GetEnrollmentStatus action	us-east-2c	ec2-3-15-43-39.us-east-2...	3.15.43.39

Instance: i-07979dbaf6cc0a7c5 (kinnar\_ec2\_UbuntuLinux\_DevOps\_B4)

Details

Status and alarms New

Monitoring

Security

Networking

Storage

Tags

#### ▼ Instance summary Info

Instance ID

i-07979dbaf6cc0a7c5 (kinnar\_ec2\_UbuntuLinux\_DevOps\_B4)

IPv6 address

-

Hostname type

IP name: ip-172-31-32-207.us-east-2.compute.internal

Answer private resource DNS name

IPv4 (A)

Auto-assigned IP address

3.15.43.39 [Public IP]

IAM Role

-

Public IPv4 address

3.15.43.39 [open address](#)

Instance state

Running

Private IP DNS name (IPv4 only)

ip-172-31-32-207.us-east-2.compute.internal

Instance type

t2.micro

VPC ID

vpc-02c848fd2d7a8634e [open address](#)

Subnet ID

subnet-02f192d52f1c25421 [open address](#)

Private IPv4 addresses

172.31.32.207

Public IPv4 DNS

ec2-3-15-43-39.us-east-2.compute.amazonaws.com [open address](#)

Elastic IP addresses

-

AWS Compute Optimizer finding

User: arn:aws:iam::295397358094:user/KINNAR\_CHOWDHURY is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: \* because no identity-based policy allows the compute-optimizer:GetEnrollmentStatus action

[Retry](#)

Auto Scaling Group name

-



11. Now connect to your AWS instance as follows.

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID  
i-07979dbaf6cc0a7c5 (kinnar\_ec2\_UbuntuLinux\_DevOps\_B4)

1. Open an SSH client.

2. Locate your private key file. The key used to launch this instance is kinnar\_ec2\_key.pem

3. Run this command, if necessary, to ensure your key is not publicly viewable.  
chmod 400 "kinnar\_ec2\_key.pem"

4. Connect to your instance using its Public DNS:  
ec2-3-15-43-39.us-east-2.compute.amazonaws.com

Example:  
ssh -i "kinnar\_ec2\_key.pem" ubuntu@ec2-3-15-43-39.us-east-2.compute.amazonaws.com

Note:

In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

So, for this project, I have taken two ec2 instances to deploy this application. One ec2 will have the backend deployment, and the other one will be used to host the frontend. Now, I have to set up the backend server that will be connected to the front end of the TravelMemory application.

Kinnar-TravelMemory-Backend

Kinnar-TravelMemory-Frontend

## Application System Requirements

Operating System - Ubuntu Linux (22.04)  
Application - Node (Version 18.x)  
Database - Mongo DB  
Web Server - Nginx

Now, we will install Node JS in the both servers. Please follow these steps sequentially:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install -y curl
$ curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
$ sudo apt install -y nodejs
After finish check node version:
$ node --version
```

```
37 sudo apt update
38 sudo apt upgrade
39 sudo apt install -y curl
40 curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
41 sudo apt install -y nodejs
42 node -v
43 clear
44 history
ubuntu@ip-172-31-7-122:~$ node --version
v18.20.2
ubuntu@ip-172-31-7-122:~$
```

## 1. Backend Configuration:

After logging into the ec2 here mentioned as Kinnar-TravelMemory-Backend through ssh, we will first update the apt repository of the ec2 instance.

```
$ sudo apt update
```

We can also upgrade the packages to the latest version.

```
$ sudo apt upgrade -y
```

As of now we already have accessed the complete codebase of the TravelMemory app from the provided GitHub link

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

```
$ git clone https://github.com/UnpredictablePrashant/TravelMemory.git
```

```
ubuntu@ip-172-31-7-122:~/travel_memory_backend$ git clone https://github.com/UnpredictablePrashant/TravelMemory.git
Cloning into 'TravelMemory'...
remote: Enumerating objects: 92, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 92 (delta 8), reused 5 (delta 5), pack-reused 70
Receiving objects: 100% (92/92), 217.75 KiB | 3.40 MiB/s, done.
Resolving deltas: 100% (23/23), done.
ubuntu@ip-172-31-7-122:~/travel_memory_backend$ ls
TravelMemory
ubuntu@ip-172-31-7-122:~/travel_memory_backend$ cd TravelMemory/backend/
```

Created an environment file (.env) in the below location:  
/home/ubuntu/travel\_memory\_backend/TravelMemory/backend

```
$ cd /home/ubuntu/travel_memory_backend/TravelMemory/backend/
$ sudo nano .env
```

```
MONGO_URI='mongodb+srv://development:xxxxx-kc@cluster0-kc.kodulgi.mongodb.net/travelmemory'
PORT=3000
```

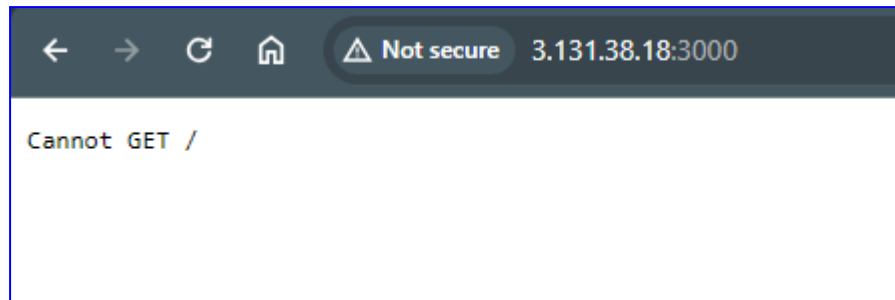
```
ubuntu@ip-172-31-7-122:~$ cd /home/ubuntu/travel_memory_backend/TravelMemory/backend/
ubuntu@ip-172-31-7-122:~/travel_memory_backend/TravelMemory/backend$ pwd
/home/ubuntu/travel_memory_backend/TravelMemory/backend
ubuntu@ip-172-31-7-122:~/travel_memory_backend/TravelMemory/backend$ ls -a
.  .. .env conn.js controllers index.js models package-lock.json package.json routes
ubuntu@ip-172-31-7-122:~/travel_memory_backend/TravelMemory/backend$ sudo nano .env
ubuntu@ip-172-31-7-122:~/travel_memory_backend/TravelMemory/backend$ cat .env
MONGO_URI='mongodb+srv://development:Dev2024-kc@cluster0-kc.kodulgi.mongodb.net/travelmemory'
PORT=3000
ubuntu@ip-172-31-7-122:~/travel_memory_backend/TravelMemory/backend$ █
```

After the successful installation of node version 18, we will proceed for the backend server to be ready for use. For this process, we will simply move to the backend directory located at **/home/ubuntu/travel\_memory\_backend/TravelMemory/backend/** and install the required dependencies using Node Package Manager from **package.json**.

```
$ cd /home/ubuntu/travel_memory_backend/TravelMemory/backend/
$ npm install
$ sudo node index.js
```

```
ubuntu@ip-172-31-7-122:~/travel_memory_backend/TravelMemory/backend$ sudo node index.js
Server started at http://localhost:3000
```

Now check the application in the browser:



**Note:** If you are seeing this page, it means the backend application is running healthy.

Now We need to configure a reverse proxy to ensure smooth running of application deployment. So, we install nginx in the backend server and configure a reverse proxy in the `/etc/nginx/sites-available/default` file. We have added `proxy_pass` method in the location directives in the default file.

```
location / {
    proxy_pass http://127.0.0.1:3001/;
}
```

Another Twick:- `sudo nohup node index.js > /home/ubuntu/travel_memory_backend/backend.log &`

## 2. Frontend Configuration:

After logging into the ec2, here mentioned as Kinnar-TravelMemory-Frontend through ssh, we will first update the apt repository of the ec2 instance.

```
$ sudo apt update
```

We can also upgrade the packages to the latest version.

```
$ sudo apt upgrade -y
```

As of now we already have accessed the complete codebase of the TravelMemory app from the provided GitHub link

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

```
$ git clone https://github.com/UnpredictablePrashant/TravelMemory.git
```

```
ubuntu@ip-172-31-7-122:~/travel_memory_backend$ git clone https://github.com/UnpredictablePrashant/TravelMemory.git
Cloning into 'TravelMemory'...
remote: Enumerating objects: 92, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 92 (delta 8), reused 5 (delta 5), pack-reused 70
Receiving objects: 100% (92/92), 217.75 KiB | 3.40 MiB/s, done.
Resolving deltas: 100% (23/23), done.
ubuntu@ip-172-31-7-122:~/travel_memory_backend$ ls
TravelMemory
ubuntu@ip-172-31-7-122:~/travel_memory_backend$ cd TravelMemory/backend/
```

We need to modify url.js file inside src folder in the below location:

/home/ubuntu/travel\_memory\_backend/TravelMemory/frontend/

```
$ cd /home/ubuntu/travel_memory_frontend/TravelMemory/frontend/
```

```
$ cd src && sudo nano url.js
```

```
export const baseUrl = "http://3.139.60.175:3001"
```

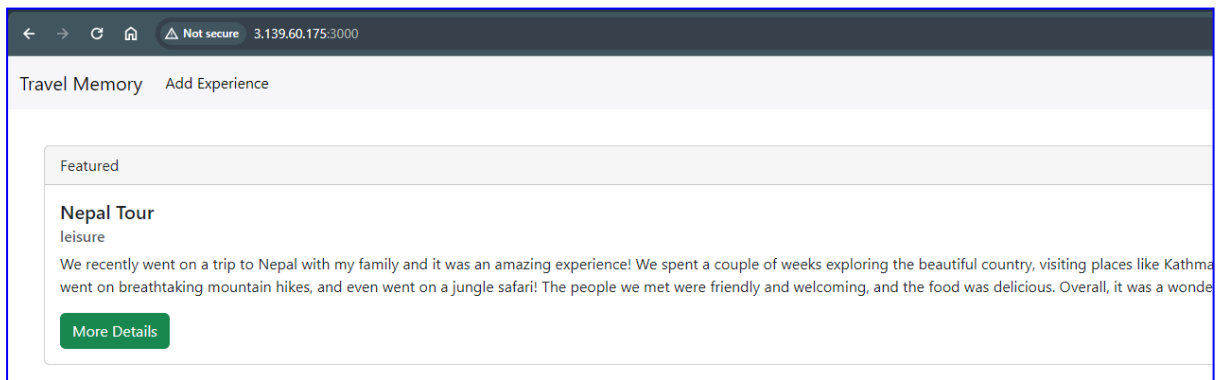
```
ubuntu@ip-172-31-34-120:~$ cd /home/ubuntu/travel_memory_frontend/TravelMemory/frontend/
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend$ cd src/ && sudo nano url.js
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend/src$ ls
App.css  App.js  App.test.js  components  index.css  index.js  logo.svg  reportWebVitals.js  setupTests.js  url.js
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend/src$ cd ..
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend$ ls
README.md  node_modules  package-lock.json  package.json  public  src
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend$ cd src/
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend/src$ cat url.js
export const baseUrl = "http://52.14.109.168:3001"
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend/src$ cd ..
ubuntu@ip-172-31-34-120:~/travel_memory_frontend/TravelMemory/frontend$
```

After the successful installation of node version 18, we will proceed for the backend server to be ready for use. For this process, we will simply move to the backend directory located at **/home/ubuntu/travel\_memory\_frontend/TravelMemory/frontend/** and install the required dependencies using Node Package Manager from **package.json**.

```
$ cd /home/ubuntu/travel_memory_frontend/TravelMemory/frontend/  
$ npm install  
$ sudo npm start
```

```
Compiled successfully!  
  
You can now view frontend in the browser.  
  
Local:      http://localhost:3001  
On Your Network: http://172.31.34.120:3001  
  
Note that the development build is not optimized.  
To create a production build, use npm run build.  
  
webpack compiled successfully
```

Now check the application in the browser:



**Note:** If you are seeing this page, it means the frontend application is running healthy.

Now We need to configure another reverse proxy for the frontend application to ensure smooth running of application deployment. So, we install nginx in the frontend server and configure a reverse proxy in the `/etc/nginx/sites-available/default` file. We have added `proxy_pass` method in the location directives in the default file.

```
location / {  
    proxy_pass http://127.0.0.1:3000;  
}
```

Another Twick:- `sudo nohup npm start > /home/ubuntu/travel_memory_frontend/frontend.log &`

### 3. Scaling the Application:

We have registered two new application load balancers for Backend & Frontend to scale up the application.

<input checked="" type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
<input checked="" type="checkbox"/>	Kinnar-TravelMemory-Frontend-LB	Kinnar-TravelMemory-Fron...	Active	vpc-7faf5614	3 Availability Zones	application	April 27, 2024, 13:42 (UTC+05:30)
<input checked="" type="checkbox"/>	Kinnar-TravelMemory-Backend-LB	Kinnar-TravelMemory-Bac...	Active	vpc-7faf5614	3 Availability Zones	application	April 27, 2024, 13:31 (UTC+05:30)

### Kinnar-TravelMemory-Frontend-LB

Kinnar-TravelMemory-Frontend-LB

▼ Details

Load balancer type

Application

Status

Active

VPC

vpc-7faf5614

IP address type

IPv4

Scheme

Internet-facing

Hosted zone

Z3AADJG6KTTL2

Availability Zones

us-east-2b (use2-az2)  
us-east-2c (use2-az3)  
us-east-2a (use2-az1)

Date created

April 27, 2024, 13:42 (UTC+05:30)

Load balancer ARN

arn:aws:elasticloadbalancing:us-east-2:515210271098:loadbalancer/app/Kinnar-TravelMemory-Frontend-LB/13d4426c25279a01

DNS name

Kinnar-TravelMemory-Frontend-LB-1191351255.us-east-2.elb.amazonaws.com (A Record)

Listeners and rules

Network mapping

Resource map - new

Security

Monitoring

Integrations

Attributes

Tags

Listeners and rules (1) info

Manage rules

Manage listener

Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

Protocol:Port

Default action

Rules

ARN

Security policy

Default SSL/TLS certificate

mTLS

Trust store

Tags

HTTP:80

Forward to target group  
Kinnar-TravelMemory-Frontend-TG (100%)  
Group-level stickiness: Off

1 rule

ARN

Not applicable

Not applicable

Not applicable

Not applicable

2 tags

### Kinnar-TravelMemory-Frontend-TG

Kinnar-TravelMemory-Frontend-TG

Details

Target type

Instance

Protocol : Port

HTTP: 80

Protocol version

HTTP1

VPC

vpc-7faf5614

IP address type

IPv4

Load balancer

Kinnar-TravelMemory-Frontend-LB

1 Total targets

1 Healthy  
0 Anomalous

0 Unhealthy

0 Unused

0 Initial

0 Draining

Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Targets

Monitoring

Health checks

Attributes

Tags

Registered targets (1) info

Anomaly mitigation: Not applicable

Deregister

Register targets

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

Filter targets

Instance ID

Name

Port

Zone

Health status

Health status details

Launch...

Anomaly detection result

I-0e7b81bb2517016d0

Kinnar-TravelM...

80

us-east-2c

Healthy

-

April 25, 2...

Normal

SUBMITTED BY- KINNAR CHOWDHURY, Hero Vired, DevOps, Batch 4

## Kinnar-TravelMemory-Backend-LB

Kinnar-TravelMemory-Backend-LB

▼ Details

Load balancer type

Application

Status

Active

VPC

[vpc-7faf5614](#)

IP address type

IPv4

Scheme

Internet-facing

Hosted zone

Z3AADJGK6K7TL2

Availability Zones

[subnet-6c683716](#) us-east-2b (use2-az2)  
[subnet-b00483fc](#) us-east-2c (use2-az3)  
[subnet-63fa1b08](#) us-east-2a (use2-az1)

Date created

April 27, 2024, 13:31 (UTC+05:30)

Load balancer ARN

[arn:aws:elasticloadbalancing:us-east-2:515210271098:loadbalancer/app/Kinnar-TravelMemory-Backend-LB/b08b073851f2a808](#)

DNS name info

[Kinnar-TravelMemory-Backend-LB-2111625864-us-east-2.elb.amazonaws.com \(A Record\)](#)

Listeners and rules

Network mapping

Resource map - new

Security

Monitoring

Integrations

Attributes

Tags

Listeners and rules (1) info

Manage rules ▼

Manage listener ▼

Add listener

A listener checks for connection requests on its configured protocol and port. Traffic received by the listener is routed according to the default action and any additional rules.

Filter listeners

1

Protocol/Port	Default action	Rules	ARN	Security policy	Default SSL/TLS certificate	mTLS	Trust store	Tags
<a href="#">HTTP:80</a>	<div>Forward to target group</div> <ul style="list-style-type: none"> <li> <a href="#">Kinnar-TravelMemory-Backend-TG</a> (100%)               <div>0 rules</div> </li> <li>Group-level stickiness: Off</li> </ul>	<div>ARN</div> <div>Not applicable</div>	<div>Not applicable</div>	<div>Not applicable</div>	<div>Not applicable</div>	<div>Not applicable</div>	<div>Not applicable</div>	<div>2 tags</div>

## Kinnar-TravelMemory-Backend-TG

Kinnar-TravelMemory-Backend-TG

Actions

Details

aws-elasticloadbalancing-us-east-2:515210271098:targetgroup/Kinnar-TravelMemory-Backend-TG/a7f6fad83d34d811

Target type Instance	Protocol : Port HTTP: 80	Protocol version HTTP1	VPC <a href="#">vpc-7fa5614</a>
IP address type IPv4	Load balancer <a href="#">Kinnar-TravelMemory-Backend-LB</a>		

1  
Total targets

0  
Healthy

1  
Unhealthy

0  
Unused

0  
Initial

0  
Draining

0 Anomalous

► Distribution of targets by Availability Zone (AZ)

Select values in this table to see corresponding filters applied to the Registered targets table below.

Targets

Monitoring

Health checks

Attributes

Tags

Registered targets (1) info

Anomaly mitigation: Not applicable

Deregister

Register targets

Target groups route requests to individual registered targets using the protocol and port number specified. Health checks are performed on all registered targets according to the target group's health check settings. Anomaly detection is automatically applied to HTTP/HTTPS target groups with at least 3 healthy targets.

☐

Instance ID

☐

Name

☐

Port

☐

Zone

☐

Health status

☐

Health status details

☐

Launch...

☐

Anomaly detection result

☐

<a href="#">i-Oeb4c4fb69b936bd8</a>	Kinnar-TravelM...	80	us-east-2c	Unhealthy	Health checks failed wi...	April 25, 2...	Normal
-------------------------------------	-------------------	----	------------	-----------	----------------------------	----------------	--------

#### 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 frontend.

Setting up DNS for **https://kinnar-travelmemory.shrikarlonkar.co.in/**

DNS management for **shrikarlonkar.co.in**

Review, add, and edit DNS records. Edits will go into effect once saved.

DNS Setup: Full ⓘ Import and Export ▾ ⚙ Dashboard Display Settings

Search DNS Records

▼ Add filter

Search

+ Add record

kinnar-travelmemory.shrikarlonkar.co.in is an alias of Kinnar-TravelMemory-Frontend-LB-1191351255.us-east-2.elb.amazonaws.com and has its traffic proxied through Cloudflare.

Type	Name (required)	Target (required)	Proxy status	TTL
CNAME	kinnar-travelmemory <small>Use @ for root</small>	2.elb.amazonaws.com <small>E.g. www.example.com</small>	<input checked="" type="checkbox"/> Proxied	Auto

Record Attributes ⓘ Documentation

The information provided here will not impact DNS record resolution and is only meant for your reference.

Comment



**Reference:**

1. <https://stackoverflow.com/questions/76318653/how-can-i-install-node-js-version-18-on-ubuntu-22-04>
- 2.

# THANK YOU

