

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>

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.

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.

Created the EC2 Instances for both backend and frontend

In backend clone the repo using -> git clone <https://github.com/UnpredictablePrashant/TravelMemory/>

Navigate to the path for backend /home/ubuntu/Travelmemory/backend

Create a .env -> nano .env

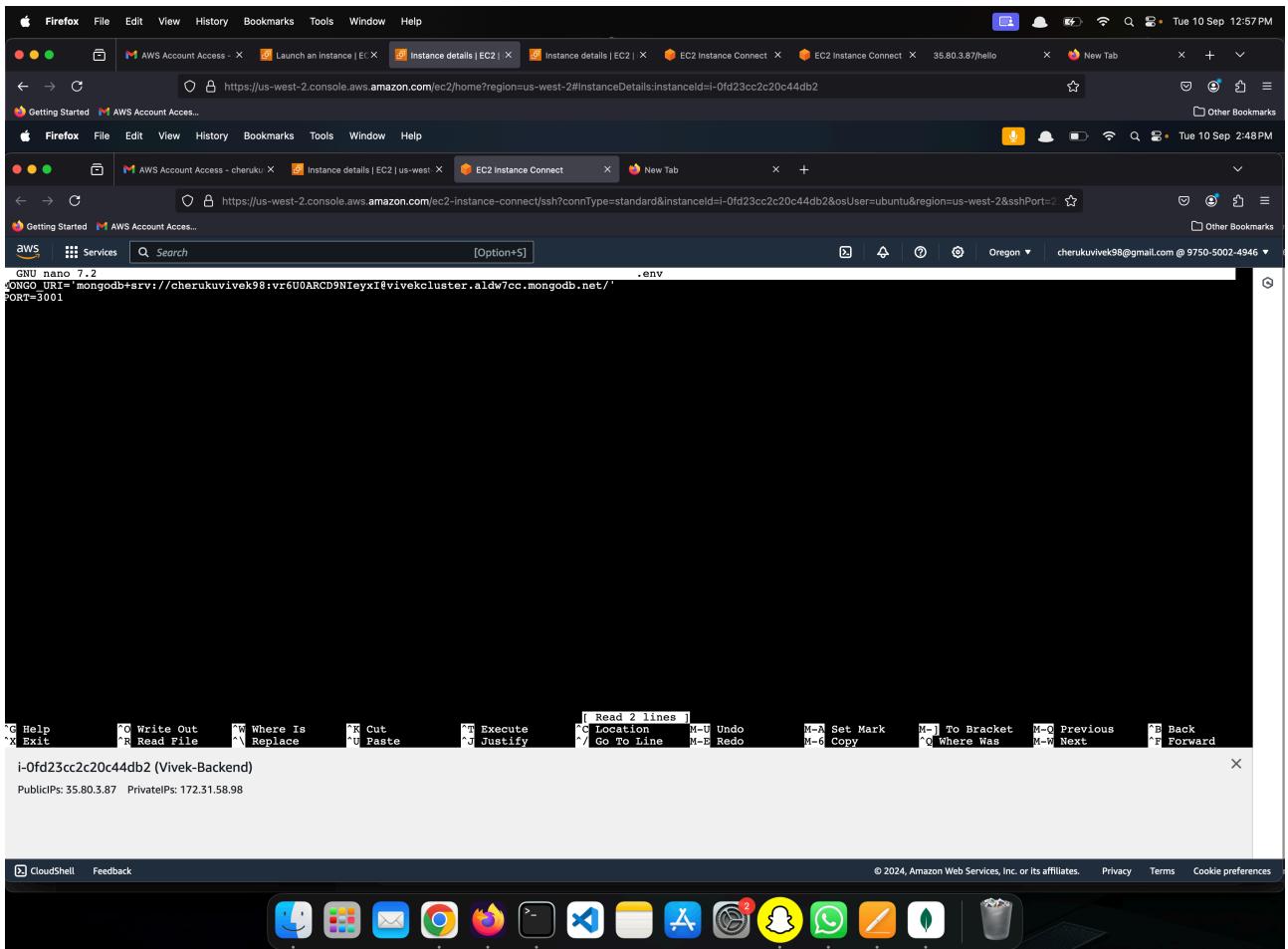
Update the details of mongo BD cluster with
MONGO_URI='ENTER_YOUR_URL'
PORT=3001

Proceed with the sudo apt install npm command to install the nesscary packages.

For Reverse proxy Nginx
-> sudo apt install nginx
and proceed with the path

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

Add the following configuration:



```

server {
listen 80;
server_name "Backend_public_id";

location / {
    proxy_pass http://localhost: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;
}
}

```

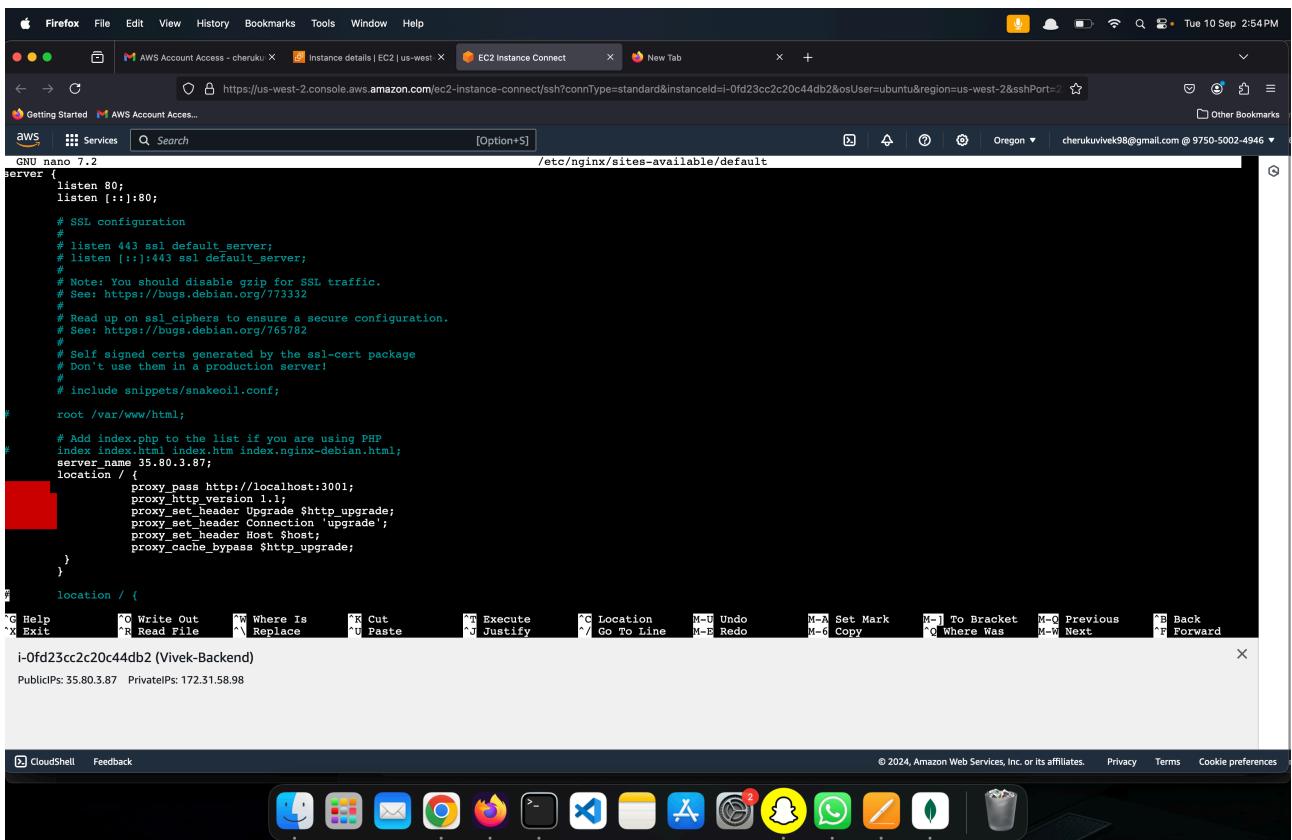
check for the nginx test with -> sudo nginx -t if successful proceed to restart the nginx with sudo systemctl restart nginx

Once done proceed to backend folder and start the 3001 port via command node index.js

Back end would be started

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.



```
GNU nano 7.2
server {
    listen 80;
    listen [::]:80;

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/77332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

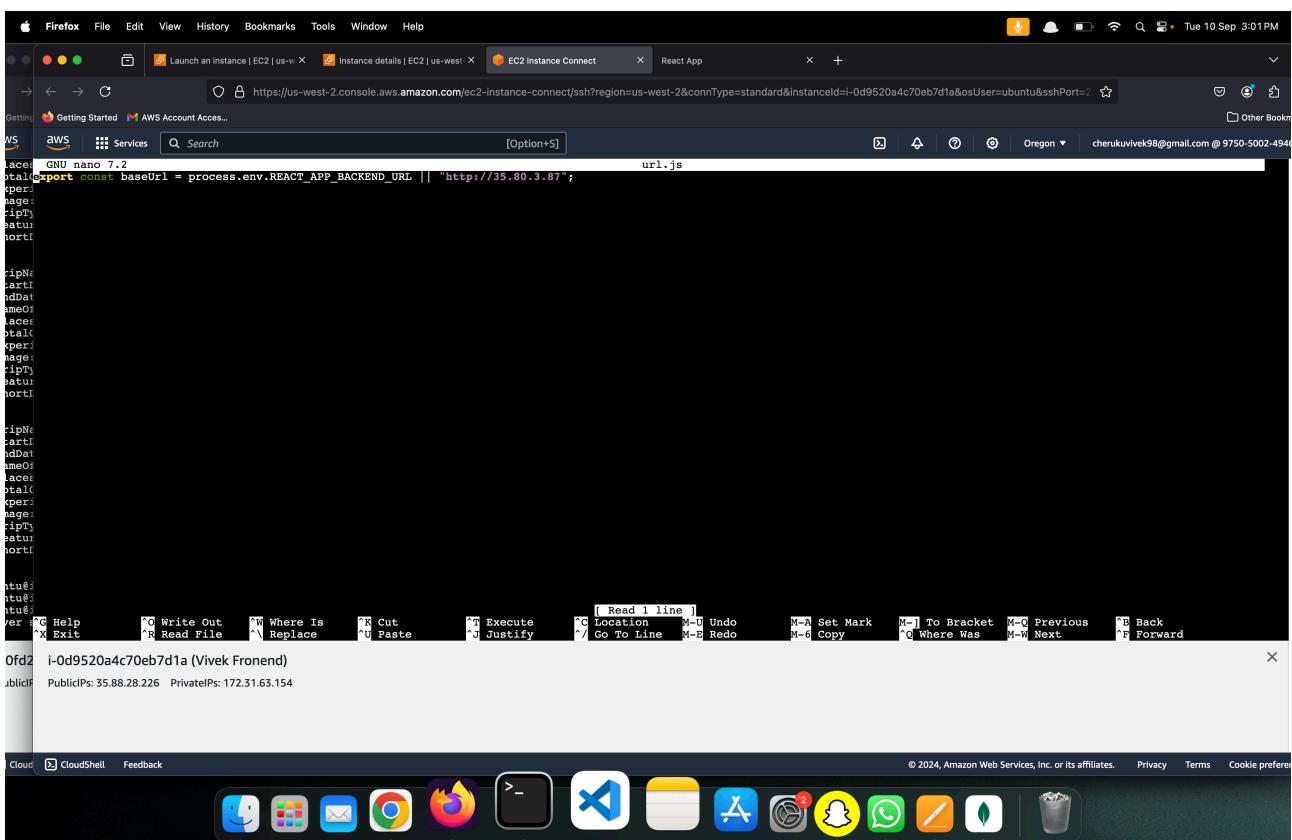
    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
    server_name 35.80.3.87;
    location / {
        proxy_pass http://localhost:3001;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

location / {
```

Help F1 Write Out F2 Where Is F3 Cut F4 Execute F5 Location M-U Undo M-A Set Mark M-J To Bracket M-Q Previous M-B Back
X Exit F6 Read File F7 Replace F8 Paste F9 Justify F10 Go To Line M-E Redo M-G Copy M-Q Where Was M-W Next M-F Forward

i-0fd23cc2c20c44db2 (Vivek-Backend)
PublicIPs: 35.80.3.87 PrivateIPs: 172.31.58.98

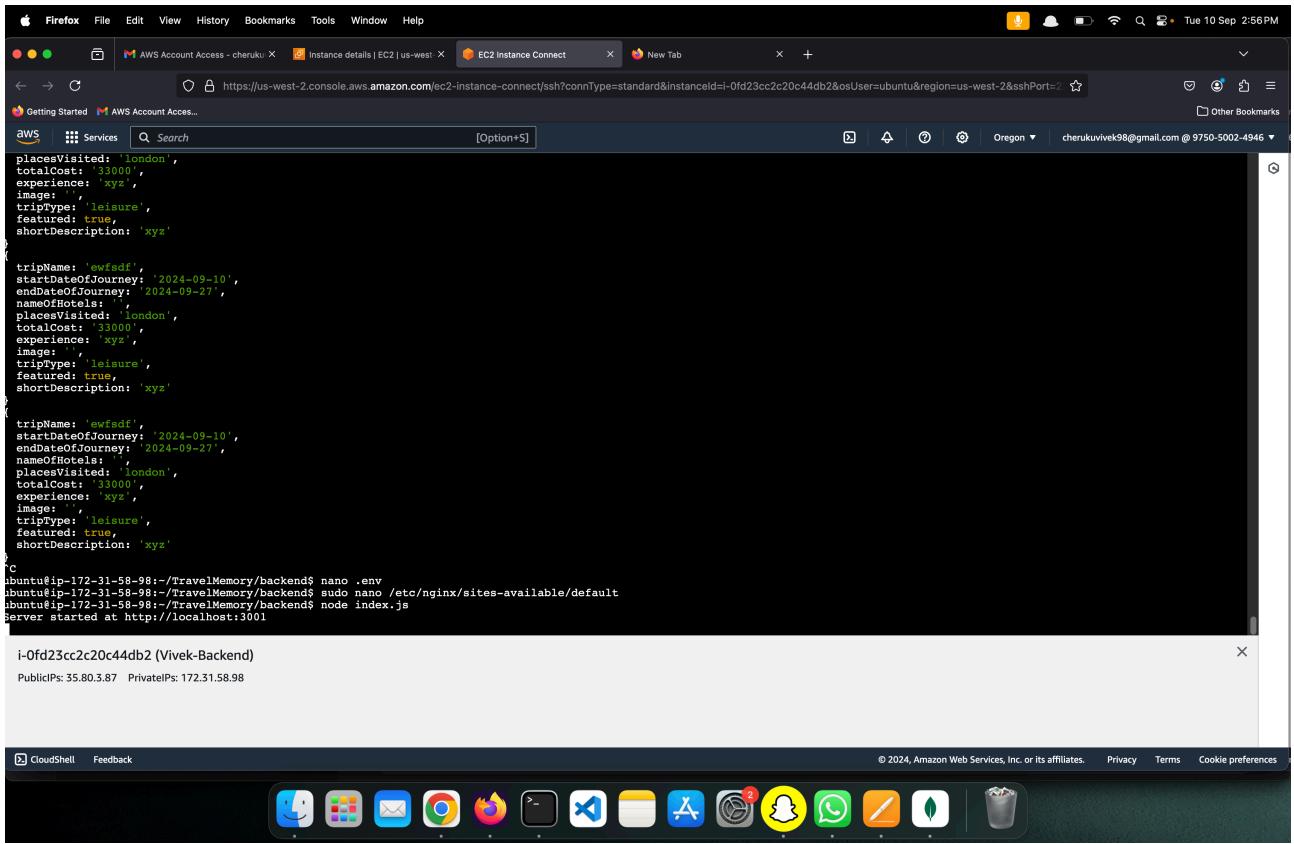
In frontend EC2 instance proceed with the path /home/ubuntu/Travelmemory/frontend and proceed with the command sudo apt install npm for packages.
Go into the src package and update the url.js with the backend public id.



```
url.js
export const baseUrl = process.env.REACT_APP_BACKEND_URL || "http://35.80.3.87";
```

Help F1 Write Out F2 Where Is F3 Cut F4 Execute F5 Location M-U Undo M-A Set Mark M-J To Bracket M-Q Previous M-B Back
X Exit F6 Read File F7 Replace F8 Paste F9 Justify F10 Go To Line M-E Redo M-G Copy M-Q Where Was M-W Next M-F Forward

i-0fd23cc2c20c44db2 (Vivek Fronend)
PublicIPs: 35.88.28.226 PrivateIPs: 172.31.63.154



```
aws Services Search [Option+S] Getting Started AWS Account Access... AWS Instance details | EC2 | us-west-2 EC2 Instance Connect New Tab

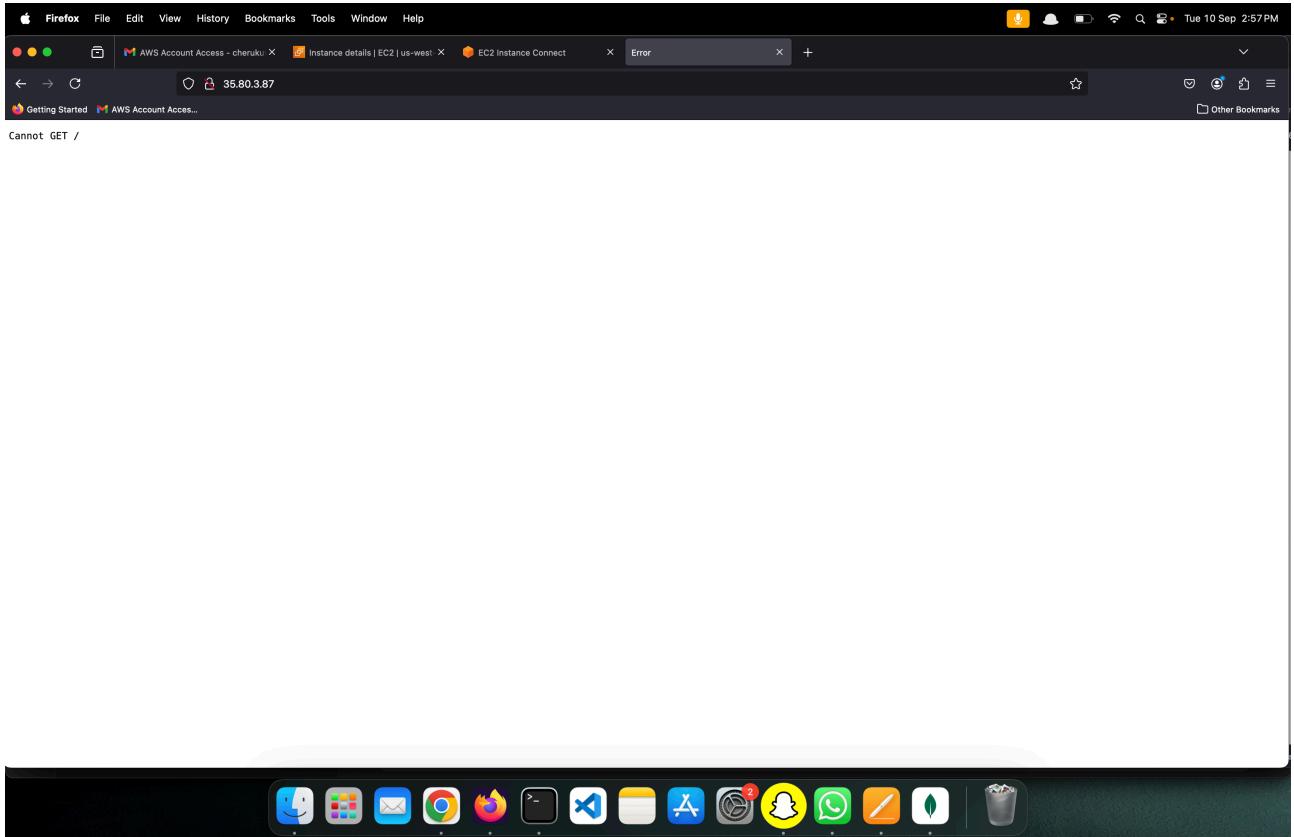
placesVisited: "london",
totalCost: 33000,
experience: "xyz",
image: '',
tripType: 'leisure',
featured: true,
shortDescription: "xyz"

tripName: 'ewfsdf',
startDateOfJourney: '2024-09-10',
endDateOfJourney: '2024-09-27',
nameOfHotels: '',
placesVisited: "london",
totalCost: 33000,
experience: "xyz",
image: '',
tripType: 'leisure',
featured: true,
shortDescription: "xyz"

tripName: 'exxfdf',
startDateOfJourney: '2024-09-10',
endDateOfJourney: '2024-09-27',
nameOfHotels: '',
placesVisited: "london",
totalCost: 33000,
experience: "xyz",
image: '',
tripType: 'leisure',
featured: true,
shortDescription: "xyz"

i-0fd23cc2c20c44db2 (Vivek-Backend)
PublicIPs: 35.80.3.87 PrivateIPs: 172.31.58.98

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```



To ensure the front end communicated effectively with the backend.
Once done save and exit

Also reverse proxy for nginx

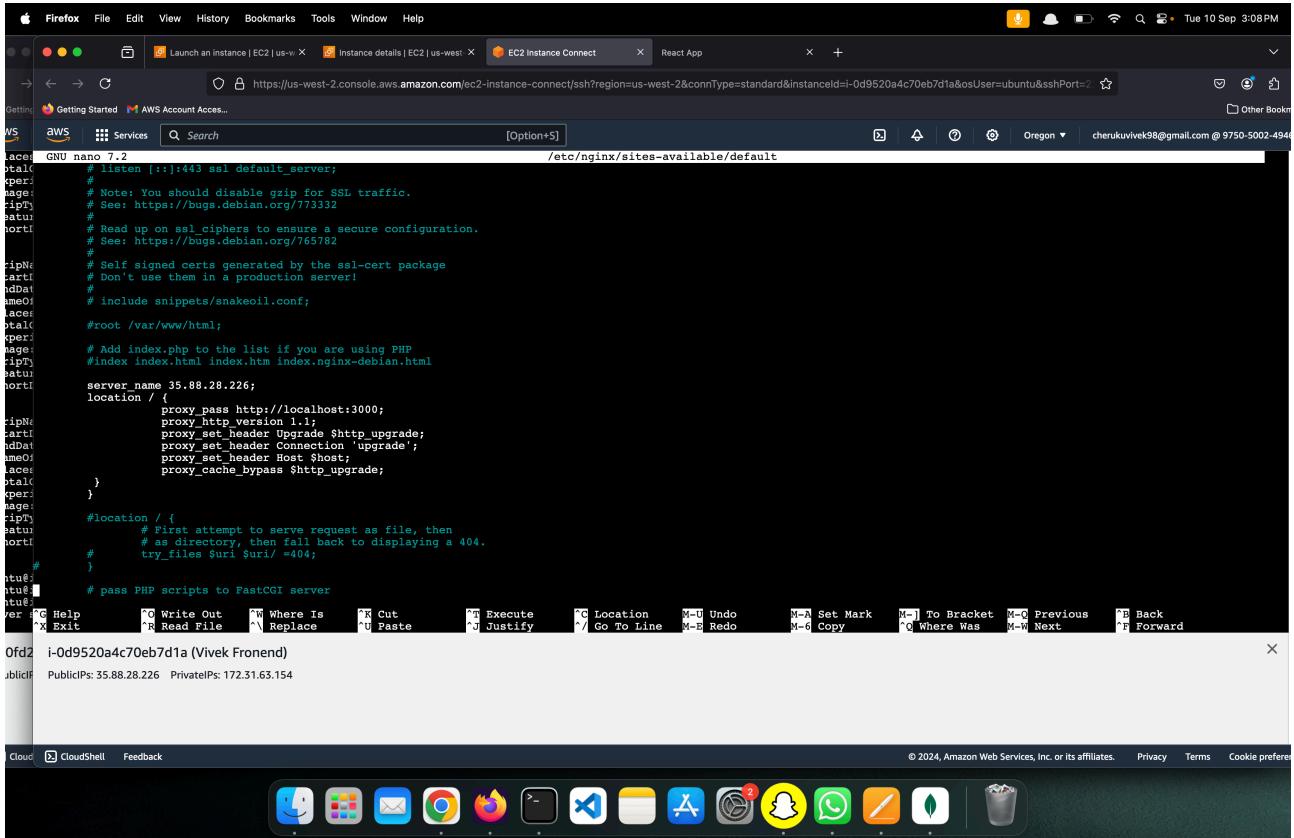
-> sudo apt install nginx
and proceed with the path

```
Sudo nano /etc/nginx/sites-available/default
```

Add the following configuration:

```
server {  
    listen 80;  
    server_name "front_end_Public ID";  
  
    location / {  
        proxy_pass http://localhost: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;  
    }  
}
```

check for the nginx test with -> sudo nginx -t if successful proceed to restart the nginx with sudo systemctl restart nginx



```
GNU nano 7.2 /etc/nginx/sites-available/default  
server {  
    listen [::]:443 ssl default_server;  
    # Note: You should disable gzip for SSL traffic.  
    # See: https://bugs.debian.org/773332  
    # Read up on ssl_ciphers to ensure a secure configuration.  
    # See: https://bugs.debian.org/765782  
    # Self signed certs generated by the ssl-cert package  
    # Don't use them in a production server!  
    # include snippets/snakeoil.conf;  
    root /var/www/html;  
    # Add index.php to the list if you are using PHP  
    #index index.htm index.html index.nginx-debian.html;  
    server_name 35.88.28.226;  
    location / {  
        proxy_pass http://localhost: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;  
    }  
    location / {  
        # First attempt to serve request as file, then  
        # as directory, then fall back to displaying a 404.  
        # try_files $uri $uri/ =404;  
    }  
    # pass PHP scripts to FastCGI server  
    #  
};
```

Come back with the frontend folder and proceed with the npm install command following with npm run build

Once done proceed with the npm start to start the front_end

The app is successfully deployed with the backend connected and the database as a mongodb
Enter the details and submit the form check that the details are saved in the data base

Once checking the EC2 and MongoDB

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.

Create a target group in AWS for frontend with the 3000 port HTTP

Add the available instance which is running the frontend

Now create the load balancer and add the target group which is created earlier with the instances created availability zones

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Tools', 'Window', 'Help'. Below the navigation bar, the URL is https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#CreateFunctionWizard:Step1. The main area has a title 'Create function' and a sub-section 'HelloWorld'. It includes fields for 'Function name' (HelloWorld), 'Runtime' (Node.js 18.x), 'Handler' (index.handler), and 'Role' (Lambda execution role). A large 'Create function' button is at the bottom.

The screenshot shows the AWS CloudWatch Metrics console interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Tools', 'Window', 'Help'. Below the navigation bar, the URL is https://us-west-2.console.aws.amazon.com/cloudwatchmetrics/home?region=us-west-2#CreateMetricFilterWizard:Step1. The main area has a title 'Create metric filter' and a sub-section 'HelloWorld'. It includes fields for 'Metric filter name' (HelloWorld), 'Metric namespace' (aws/lambda/HelloWorld), and 'Metric name' (HelloWorld). A large 'Create metric filter' button is at the bottom.

The load balance is created and the target is healthy

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.

Setup a Domain. As the cloud flare is a paid one
Hence using <https://dash.infinityfree.com/> to setup a domain

Now create a CNAME with and add the DNS name of the load balancer

The screenshot shows a web browser window with multiple tabs open. The active tab is 'DNS Records' for the domain 'vivekappdepo.free.nf'. The main content area displays a green success message: 'The CNAME record for vivekappdepo.free.nf has been created!' Below this, there's a note: 'Don't see your website yet? Please note that it can take up to 72 hours for a new domain name or hosting account to start working everywhere. Learn more.' On the left, a sidebar titled 'Manage if0_37282976' lists various account options like Home, Upgrade to Premium, Statistics, Domains, FTP Details, MySQL Databases, Deactivation History, and Account Settings. The 'Domains' option is currently selected. The right side shows a table of DNS records:

DOMAIN	TYPE	TARGET	ACTIONS
vivekappdepo.free.nf	A	185.27.134.231	
www.vivekappdepo.free.nf	A	185.27.134.231	
vivek.vivekappdepo.free.nf	CNAME	vivekappdepo-786768774.us-west-2.elb.amazonaws.com	Delete

At the bottom, there's a link to 'Upgrade to Premium Hosting'. To the right of the main content, there's a small preview window showing a video call interface with three participants.

Using the DNS Records name the website will be opened

The screenshot shows a web browser window displaying the homepage of the website 'vivek.vivekappdepo.free.nf'. The page features a 'Featured' section with a card for 'Meghalaya' travel, which includes the text 'leisure', 'One week trip', and a 'More Details' button. This pattern repeats three times. To the right of the main content, there's a small preview window showing a video call interface with three participants.

Check if the backend and frontend working fine by updating the details which should be reflected on to the frontend and mongo DB

