**Travel Memory Application**

**OBJECTIVE**

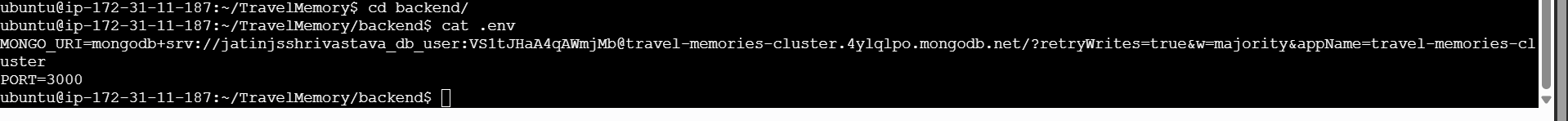
To host the Travel Memory Application (MERN stack) using application load balancer and nginx server to my custom domain using AWS services.  
  
GITHUB repository: <https://github.com/jatinggg/TravelMemory.git>

STEPS TO CONFIGURE:

**BACKEND CONFIGURATION:**

1. Start an EC2 Instance and connect

2. Clone git repo --> sudo apt git clone https://github.com/jatinggg/TravelMemory.git

3. Go to TravelMemory/backend --> sudo nano .env   


4. Configure URI for mongo db database (copy from mongodb compass) and port = 3000

5. Run sudo apt install nginx -y

sudo systemctl start nginx

sudo systemctl enable nginx

sudo systemctl status nginx

6. Run sudo apt install nodejs

7. Run sudo apt install npm

8. Run sudo npm install --> installs the react dependencies.

9. Open sudo nano /etc/nginx/sites-available/default and paste below code (Reverse Proxy Setup for nginx server on port 80)

server {

listen 80;

location / {

proxy\_pass http://<ec2-public-ip>:3000; # Change this to your backend app URL

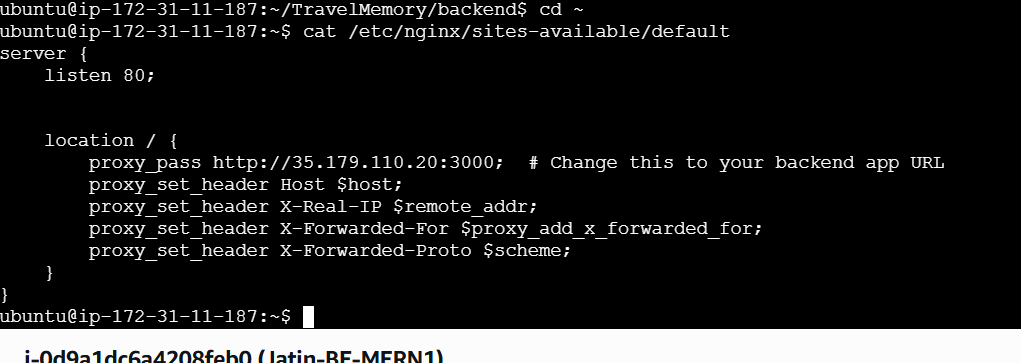
proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

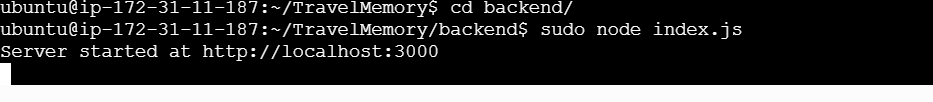
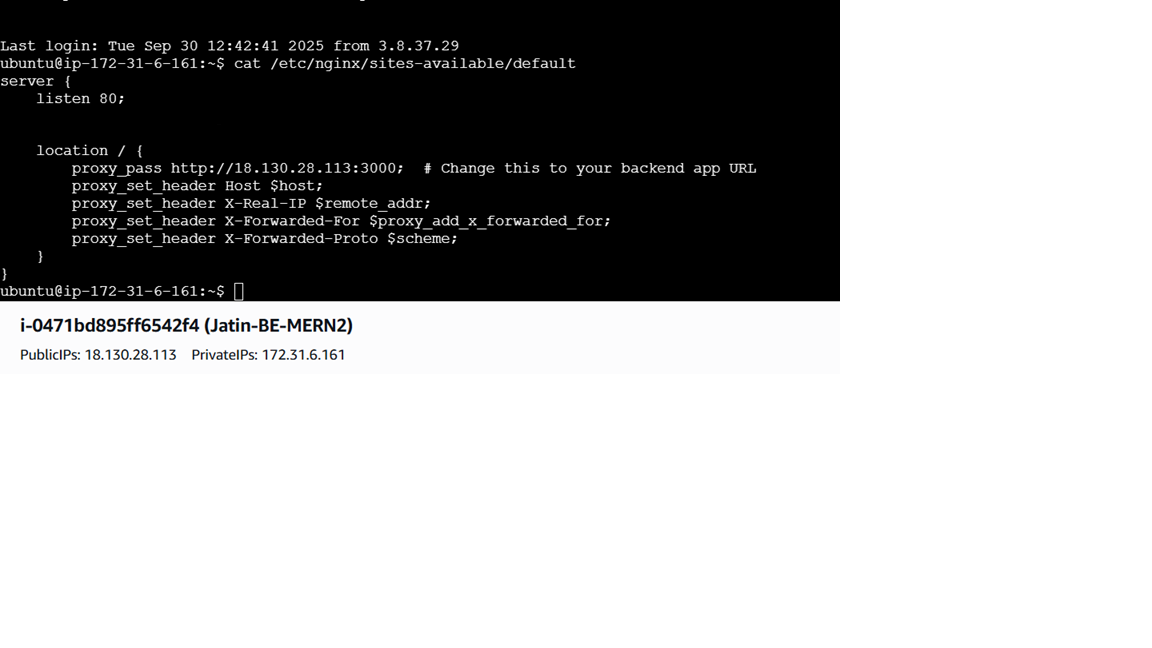
proxy\_set\_header X-Forwarded-Proto $scheme;

}

}   
  


10. Run sudo nginx -t

11. Run sudo systemctl reload nginx

12. Under EC2 security group make sure to allow TCP at port 80 for 0.0.0.0/0   
13. Go to TravelMemory/backend 🡪 run sudo node index.js  
  
  
14. Create a machine image of this instance and launch another instance running the backend server.  
15. Configure the /etc/nginx/sites-available/default and replace the IP address with the public IP of second backed server.  
  
16. Run 🡪 sudo nginx -t

Sudo systemctl reload nginx

17. Navigate to TravelMemory/backend 🡪 Run sudo node index.js to run the backend server.

**FRONTEND CONFIGURATION:**

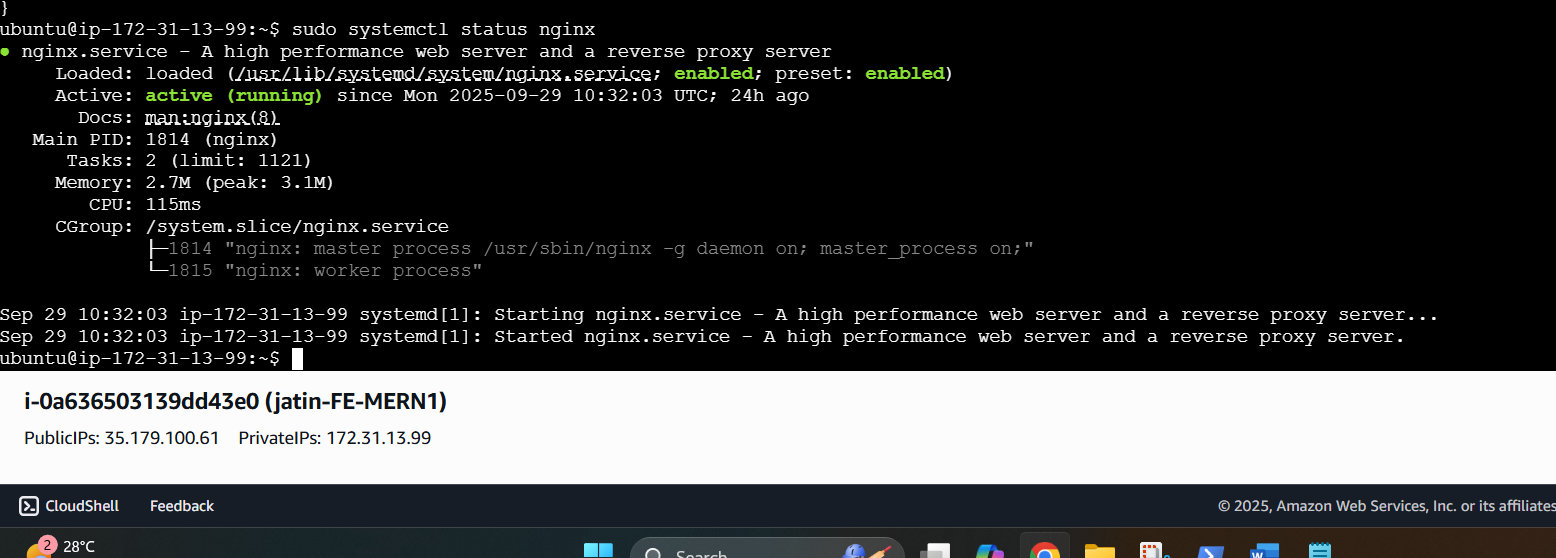
1. Start an EC2 Instance and connect

2. Clone git repo --> sudo apt git clone <https://github.com/jatinggg/TravelMemory.git>

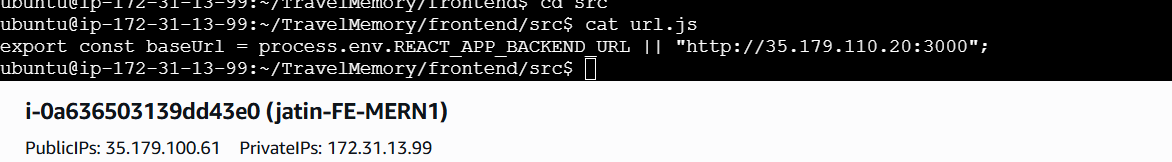
3. Run sudo apt install nginx -y

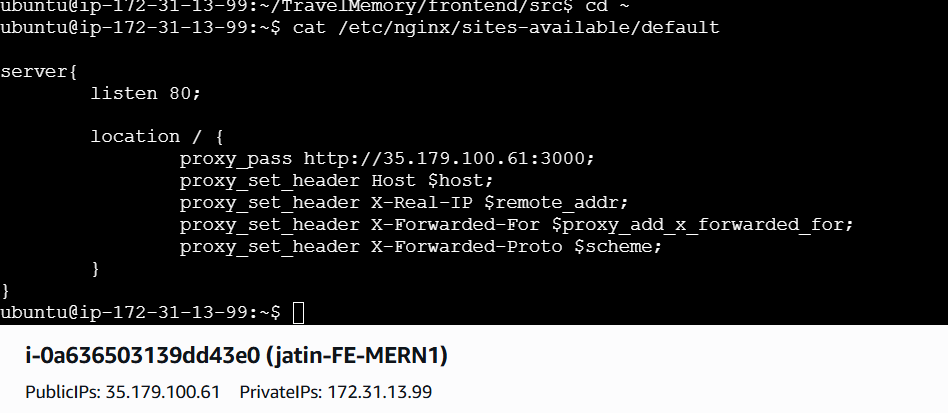
Sudo systemctl start nginx

Sudo systemctl enable nginx

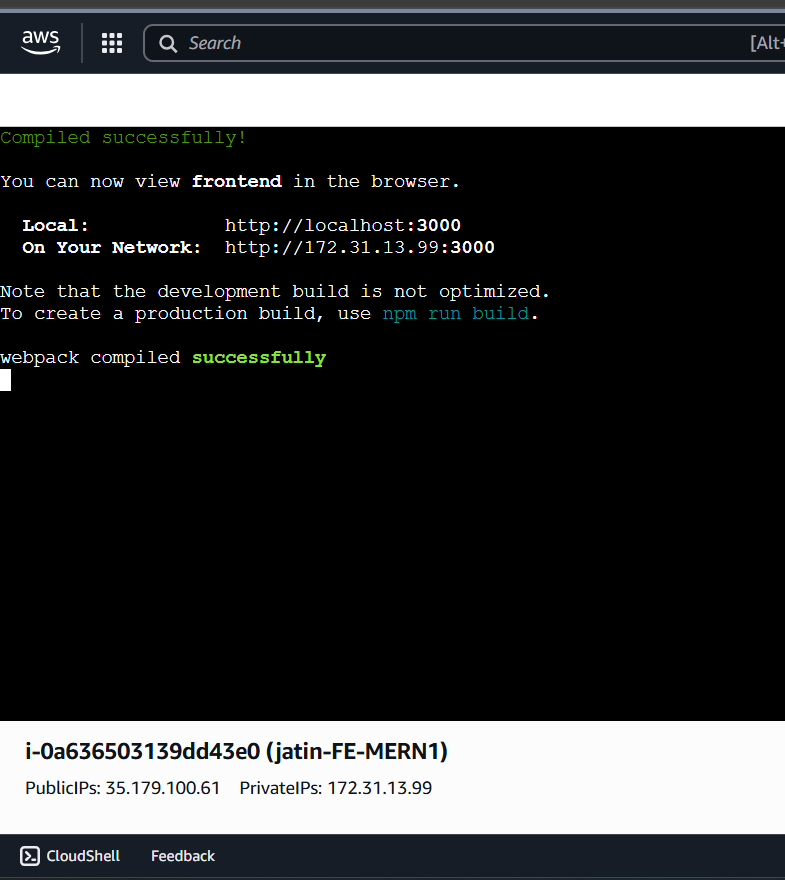
Sudo systemctl status nginx  


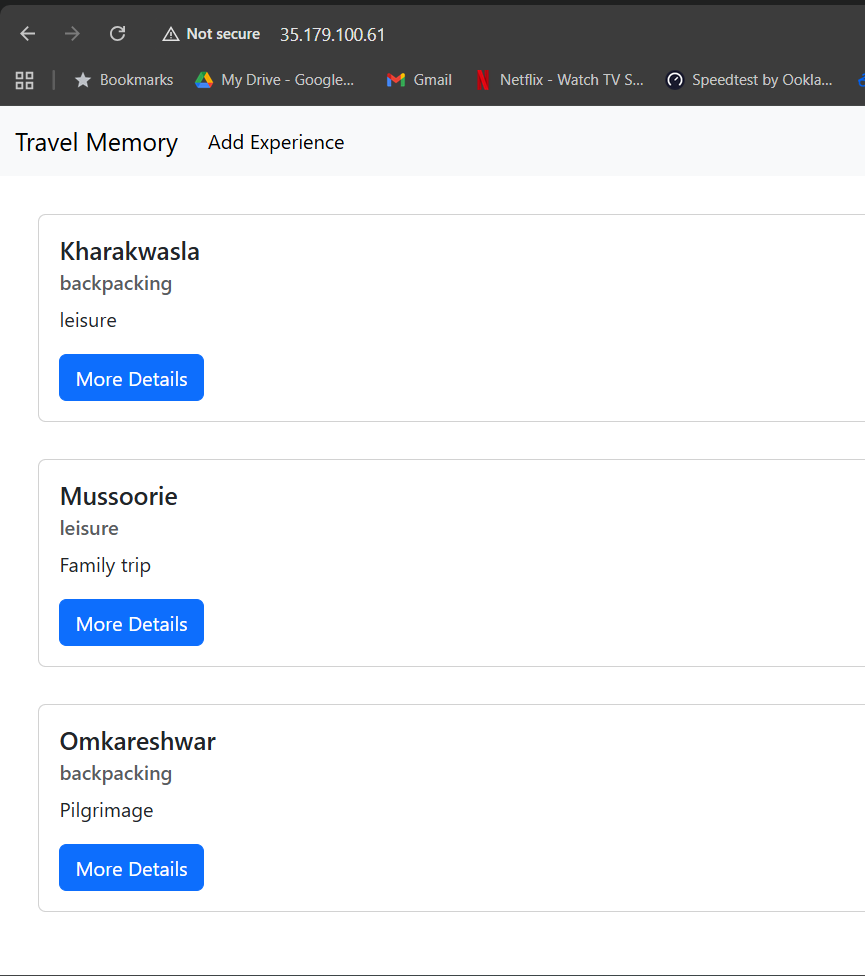
4. Run sudo apt install npm

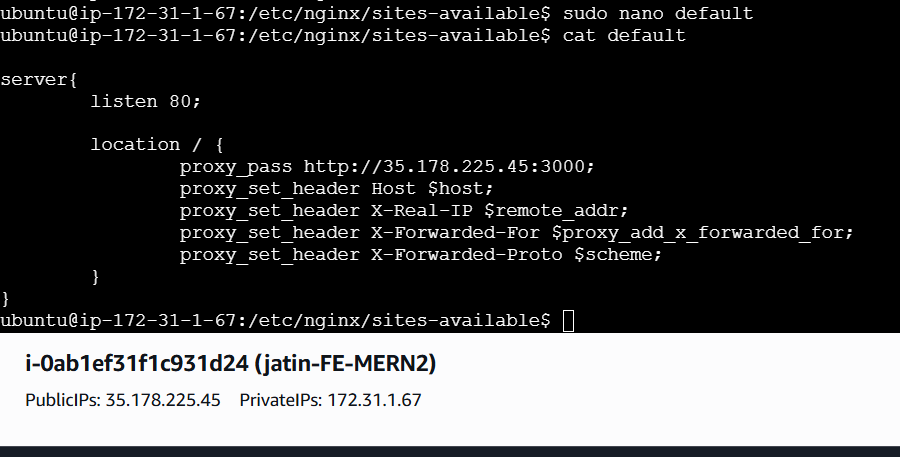
5. Go to TravelMemory/frontend/src 🡪 sudo nano url.js   
  
configure for your backend url (“http://<your-backend-public-ip:3000>”)  
  
6. Go to TravelMemory/frontend 🡪 run Sudo npm install to install all dependencies.

7. Go to /etc/nginx/sites-available/default and configure the reverse proxy for frontend   
under proxy\_pass add your frontend instance’s public IP address  
(make sure your instance’s inbound rule allow TCP traffic to port 80 from 0.0.0.0/0)  
  
8. Run sudo nginx -t

9. Run sudo systemctl reload nginx

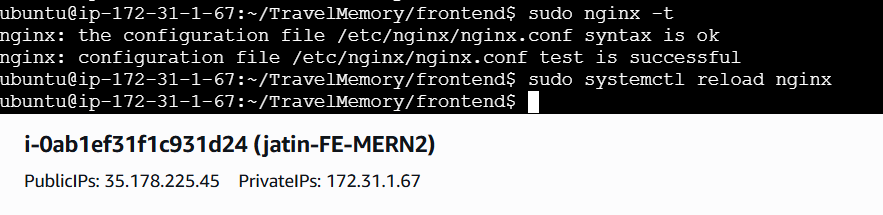
10 . Navigate to TravelMemory/frontend 🡪 run Sudo npm start   
  
  
  
11. Navigate to the web browser and open “http://<-frontend-public-ip->”  
and check the application is working

  
  
Front end configuration is completed along with revese proxy setup using nginx  
  
  
**CREATING IMAGE FOR LOAD BALANCING :**  
1. Navigate to EC2 on AWS console 🡪 Select your front end instance 🡪 ACTIONS 🡪 Image and Templates 🡪 CREATE IMAGE.  
2. Launch a new instance with this image

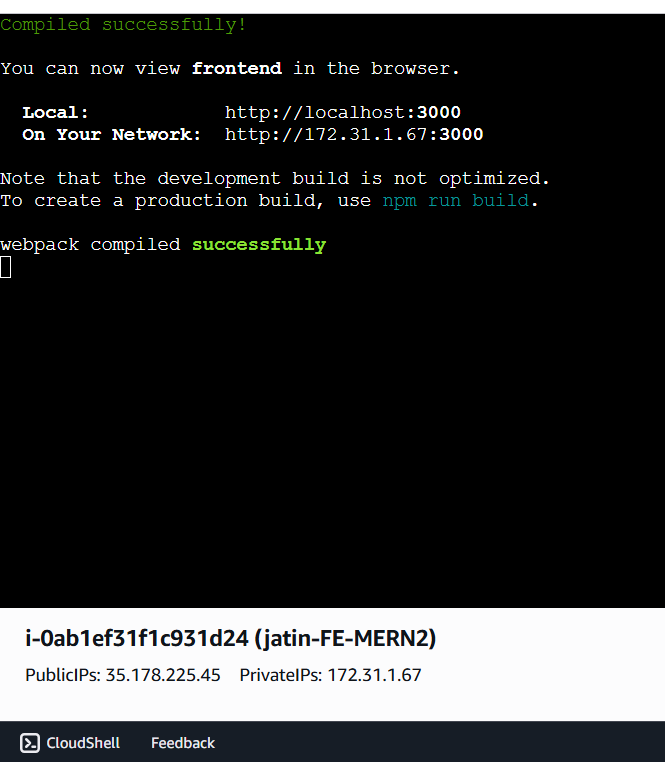
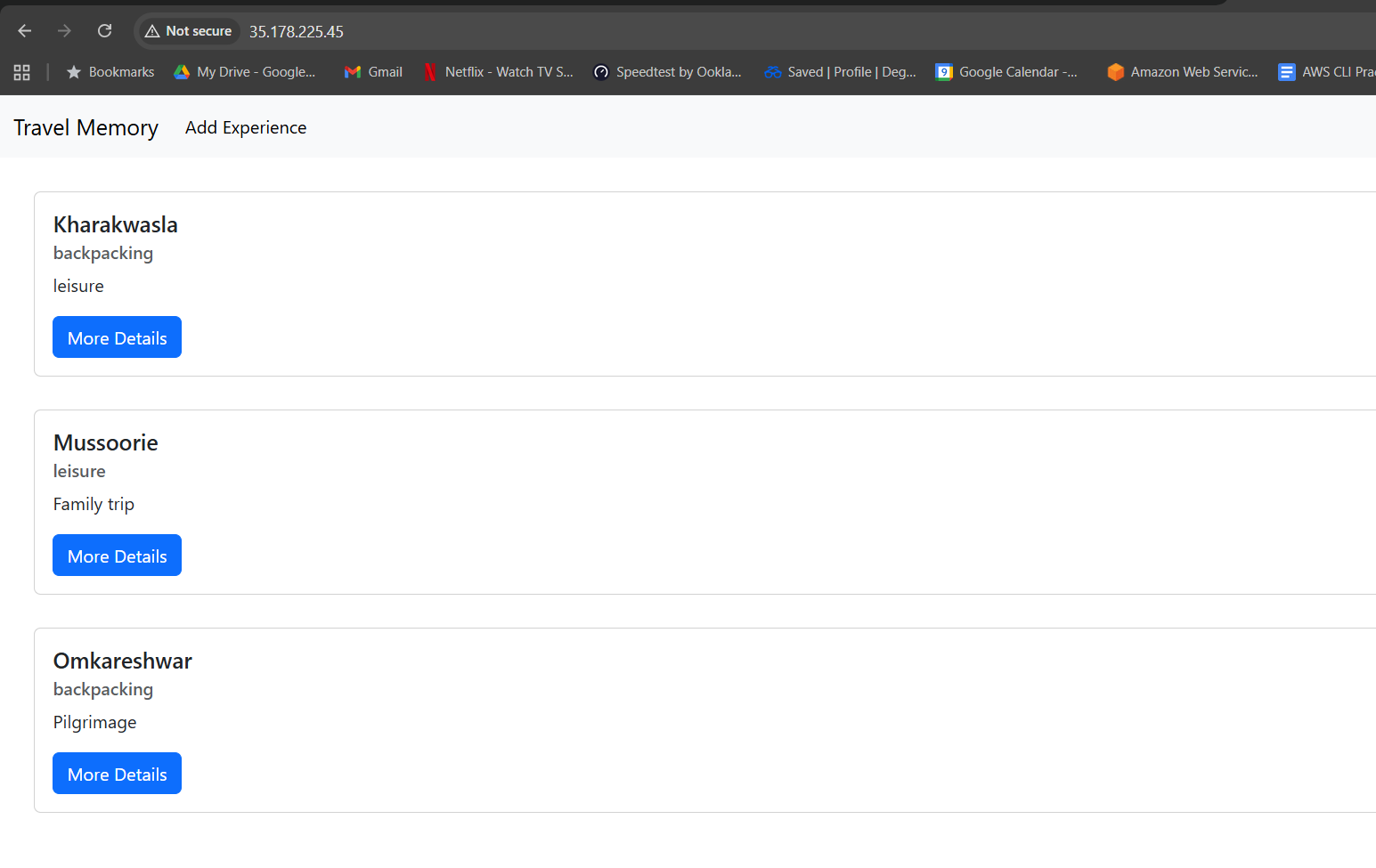
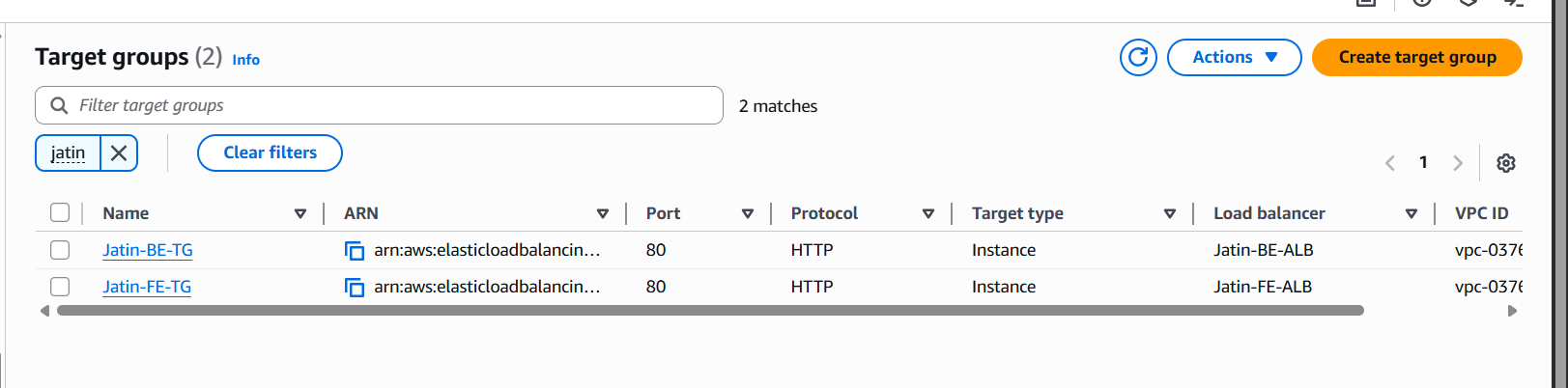
3. Connect to the second frontend instance,  
4. Naviagte to TravelMemory/frontend  
5. Run Sudo nano /etc/nginx/sites-available/default  
   
  
update the public IP of the second instance in nginx configuration.

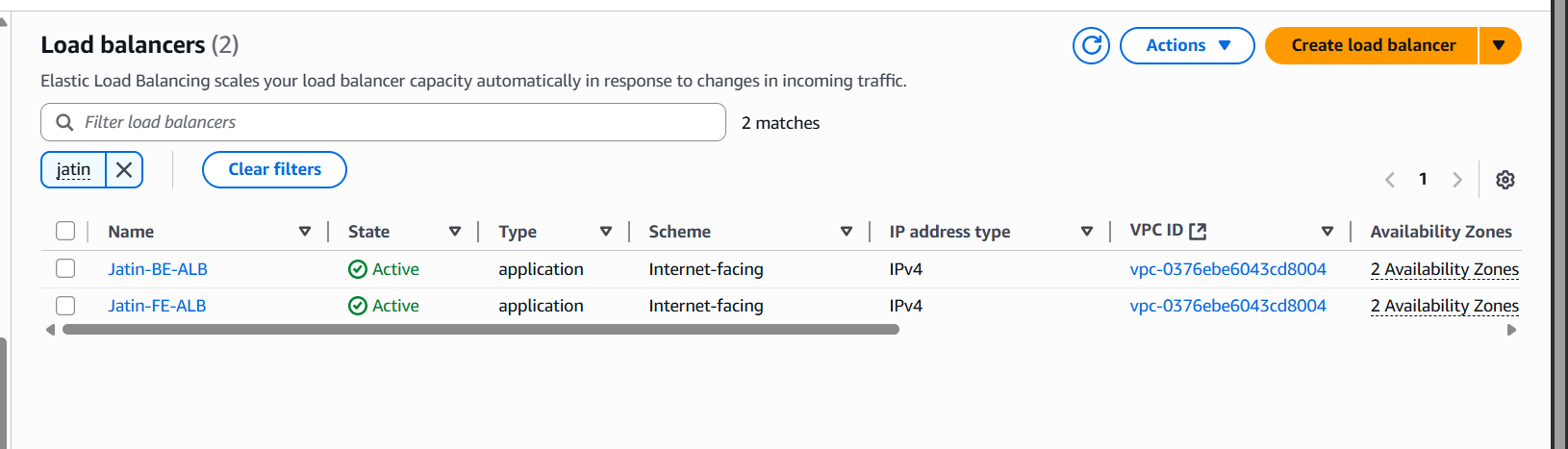
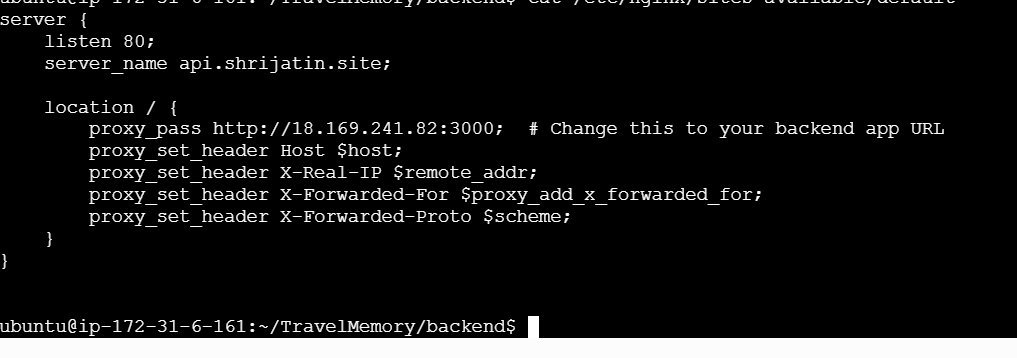
6 .Run sudo nginx -t

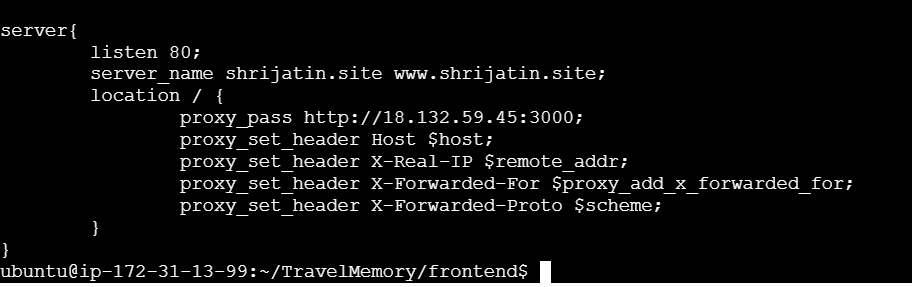
Sudo systemctl reload nginx



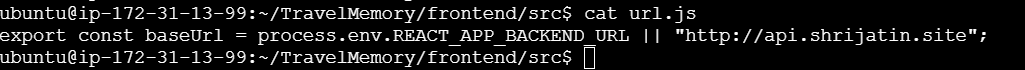
7. Run sudo npm start

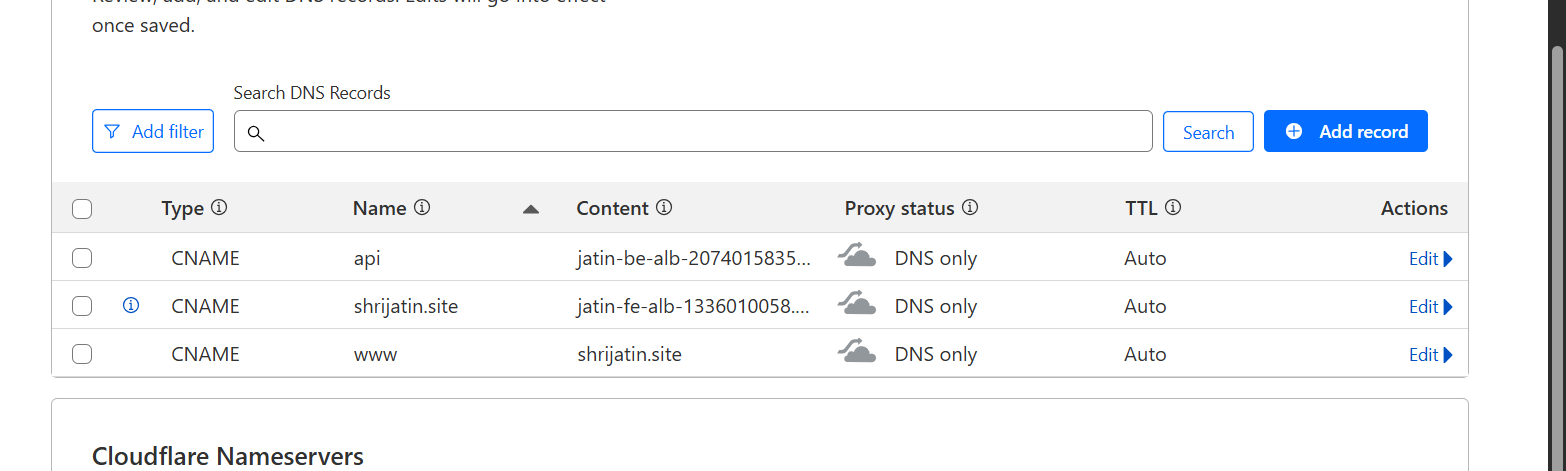
  
  
8. Navigate to the web browser and check “http://<-second-frontend-Public-IP>” and verify the application is running .  
  
  
   
  
**LOAD-BALANCER CONFIGURATRION:**  
1. Create Front end Target Group and attached the two front-end servers.  
2 .Create Backend Target group and attach the two backend servers   
  
  
3. Create application load balancers one for backend and one for front end. Attach the respective target groups.   
4 .Add the SecurityGroupId of Backend ALB into the inbound rule of backend servers for TCP at port 3000 and HTTP at port 80

5. Add the securitygroupid of Frontend ALB into the inbound rule of frontend servers to allow tcp at port 3000.  
  
  
  
  
  
  
NGINX CONFIGURATION FOR RERVERSE PROXY WITH ALB.  
  
1. Backend servers : (identical changes for both servers)  
 update the server\_name field to your custom domain name   
 example 🡪 server\_name api.shrijatin.site  
  
  
  
2. FrontEnd servers :(Identical changes for both)

Update the server\_name to your custom cname field.  
 example 🡪 server\_name shrjatin.site [www.shrijatin.site](http://www.shrijatin.site)  


Update Travelmemory/frontend/src/url.js 🡪

set backend url to “http://api.<customdomain.com>”  
  
  
  
Reload the nginx server for all instances and start the servers.

**CLOUDFLARE CONFIGURATION:**  
  
  
  
TYPE: CNAME  
NAME: api  
Content: <backend-ALB-DNS>  
  
TYPE: CNAME  
NAME: shrijatin.site  
Content: <frontend-ALB-DNS>

TYPE: CNAME  
NAME: www  
Content: shrijatin.site

**Application is successfully deployed on my custom domain.**  
