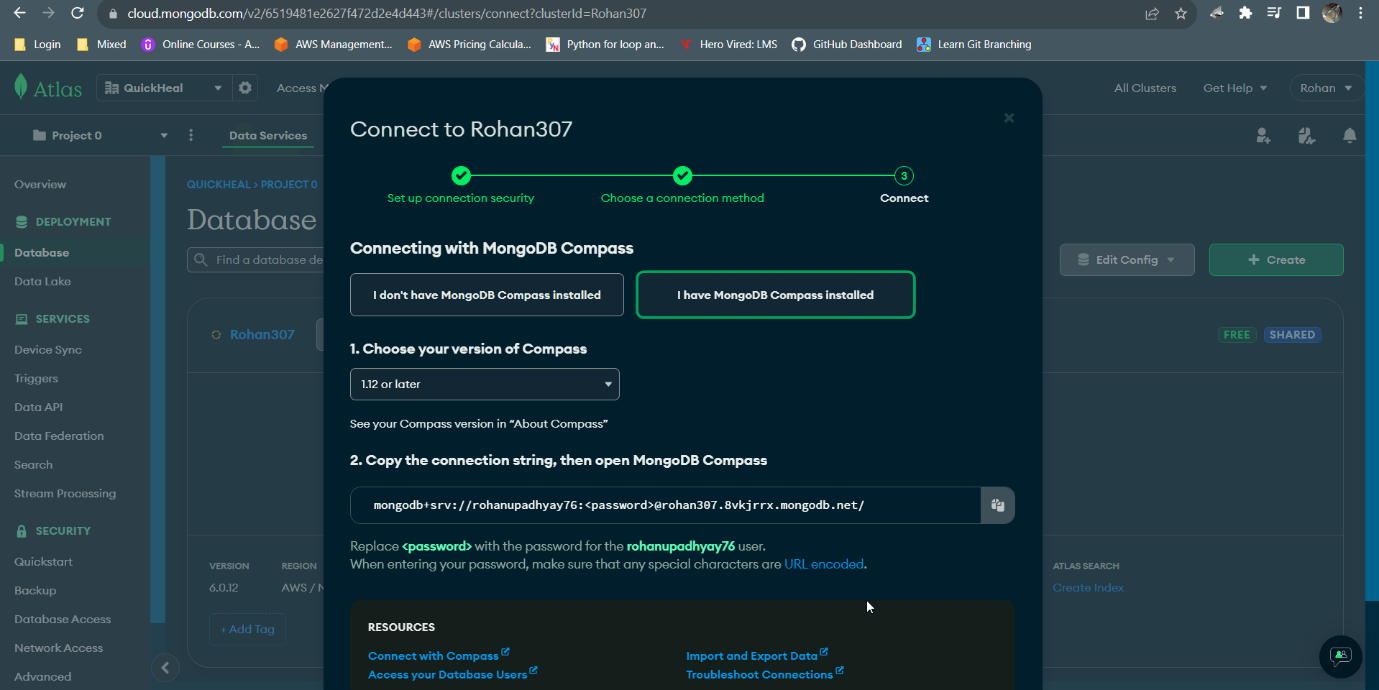
AWS Deployment:

1. We are going to run Travel Memory app in using ec2 instances. So we need to create 2 instances 1st for Fronend and 2nd for backend. Install Nodejs v 18 and clone the repo in both the instances.

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

You can use marketplace AMI which comes with already configured softwares Or you can create your own AMI and put it in community category.

You can find the mongoDB databsae Url from Database>Connect> Compass (installed in system)>I have compass installed > URL



1. Create .env file in the local machine from opening the project in visual studio by cloning the repo in local system.

Get the link of the database from above and add the content like given below in .env file.

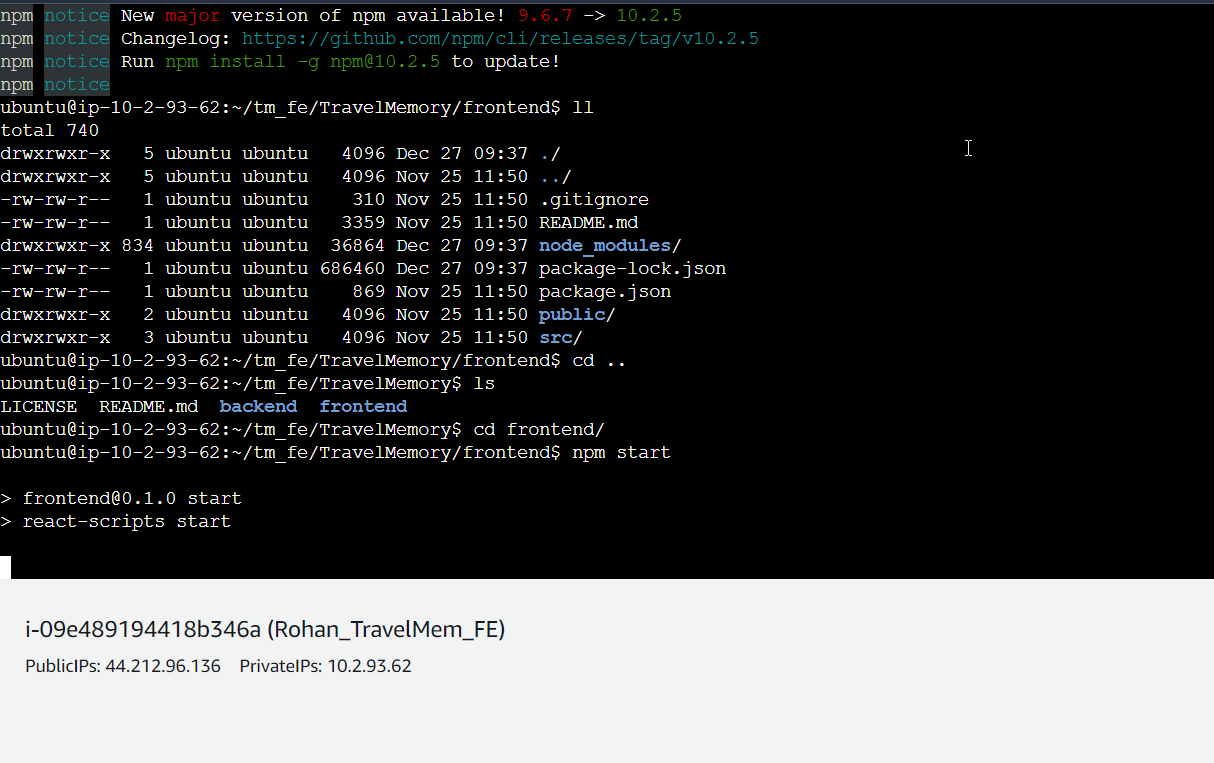
MONGO\_URI='mongodb+srv://RohanDB\_User:Admin123@rohan307.8vkjrrx.mongodb.net/travelm-rohan'

PORT=3000

1. After that run the backend EC2 instance which needs to be created at the start and check whether the nodejs is installed or not.
2. Go to the backend folder where the repo is cloned before and create .env file and copy paste the content as shown above. Mongo DB URI and port.
3. Check whether the DB is getting connected properly by using the URI in compass.
4. Save the file and enter the command npm install. It will check the package.json file and install all the dependencies by itself.
5. Once all things are installed node modules folder will be created in the backend folder itself.
6. Run node index.js and try to access app by giving public IP of the instance and port 3000. At first it will not work as we have not allowed port 3000 in security group. So we need to allow port by editing the security group custom port 3000 then you will get cannot GET / Output. Which means at least it is getting connected.
7. Now the application is running but we have to keep it running and if we press ctrl+c then it will stop so we have to run it in background. So for that we have to enter & after giving node index.js means the command is **node index.js &**

**Now go to another instance where need to run frontend.**

1. Go to frontend folder and run npm install and once it is installed run npm start



1. Now go to src folder and then edit url.js file and enter the backend Url without the / in the end of the URL there in the file and save it.
2. Now run it again with npm start and refresh the page and it will work like a charm!
3. Basically we have to connect backned to frontend so that we can access the app properly.

|  |  |
| --- | --- |
| Kill -9 pid | To kill the process with process id |

1. Now we need to run the app on port 80 because web app works on 80 or 443. but the app is running on 3000 so we need to do the port forwarding so that port 80 traffic can be forwarded to 3000.
2. That also called as reverse proxy. So we need to install nginx because the node app is single threaded app which can not handle multiple requests so we are taking help of nginx.
3. Now nginx by default runs on port 80. For achieving that we need to change default configs of nginx in the backend instance. The folder can be found in /etc/nginx/sites-available/default.
4. Before editing the default file take the backup of that file by cp default default.old command
5. Now edit the default file and comment out root /var/www/html line and index line below the root line and also try files line.
6. Go to location line and write down these lines in there.

proxy\_pass <http://localhost:3000>;

proxy\_http\_version 1.1;

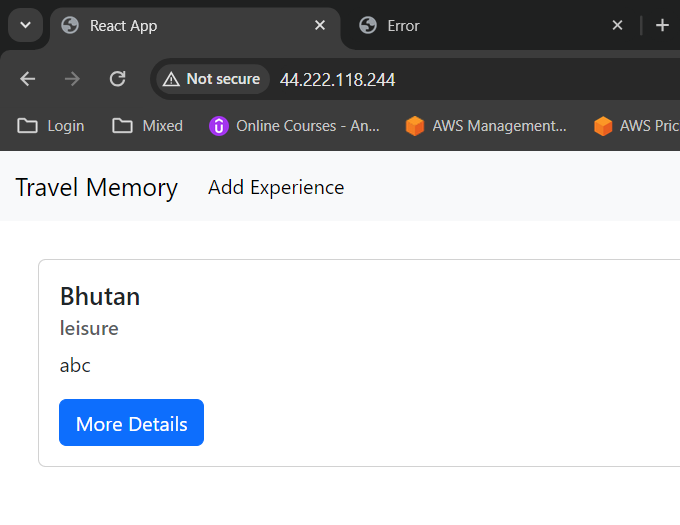
proxy\_set\_header Upgrade $http\_upgrade;

proxy\_set\_header Connection 'upgrade';

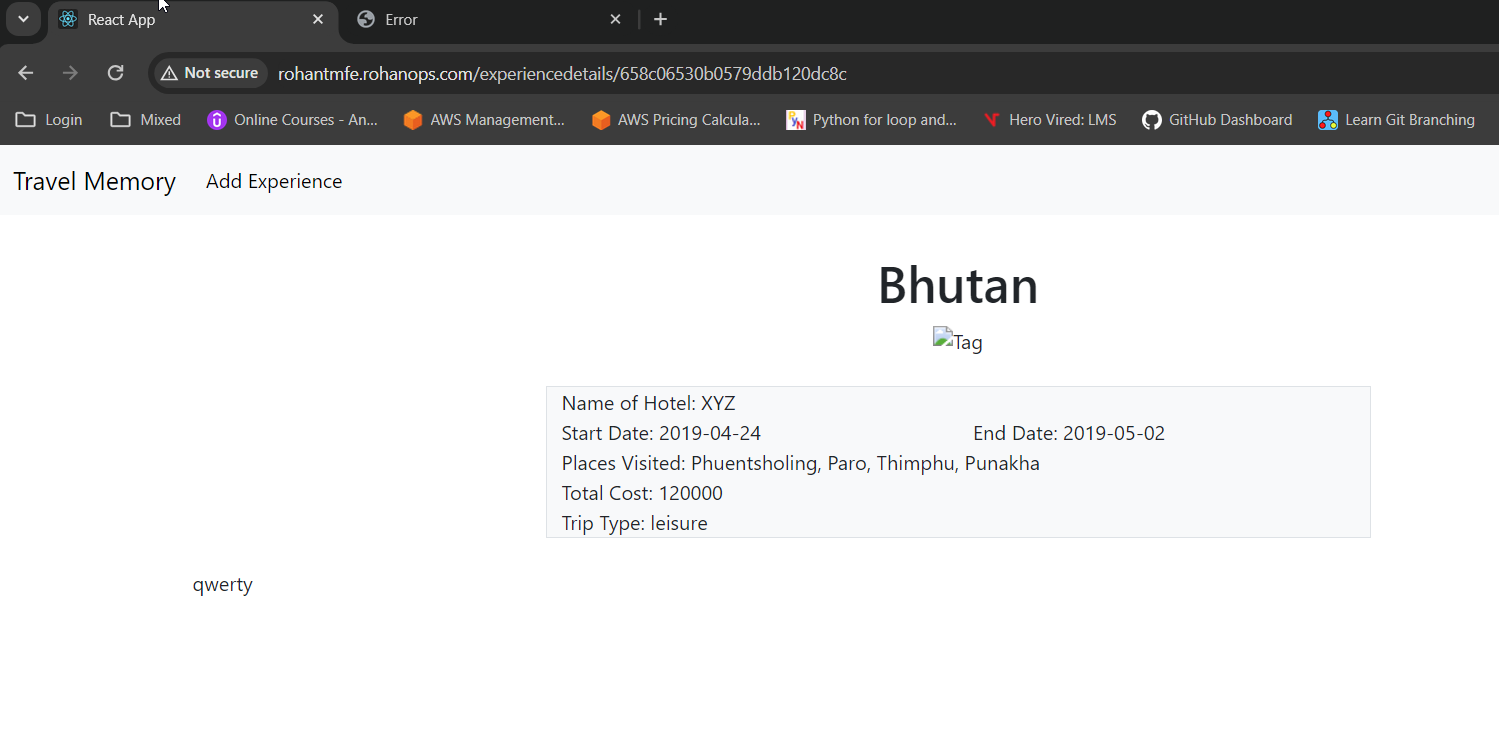
proxy\_set\_header Host $host;

proxy\_cache\_bypass $http\_upgrade;

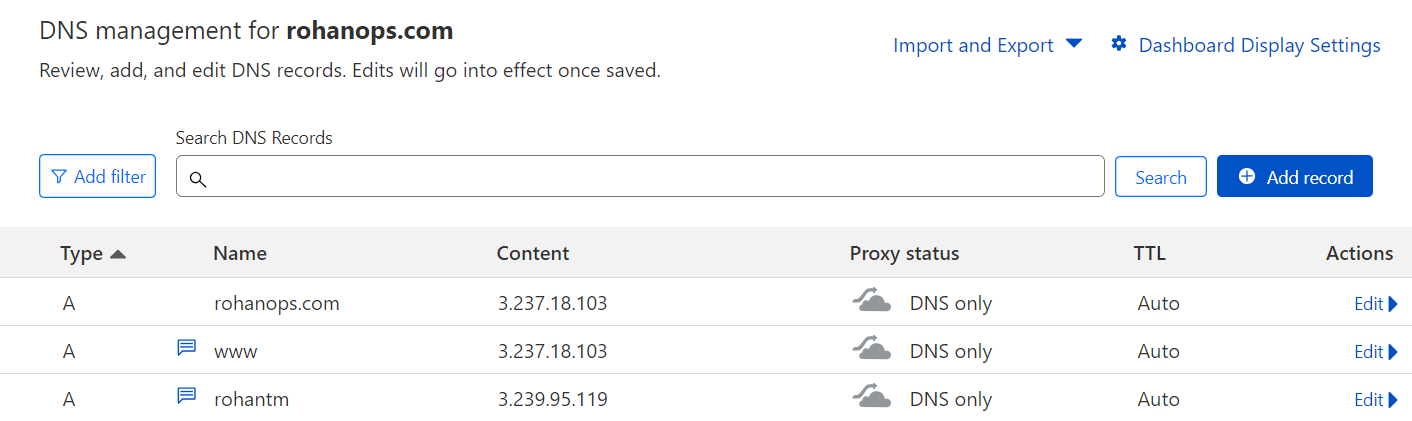
1. Now after saving the file, you can run the command for checking the syntax of the content in the file. Run sudo nginx -t. It will show you whether the syntax is correct or not of the file.
2. Now try to run the node index.js and check with the IP address whether the page is opening or not on port 80. It should get cannot get / output on the page.
3. Now go to front end instance and copy paste the same content in the /etc/nginx/default file as above.
4. Go to Travelmemory folder /src folder and edit the url.js file and remove the port 3000 as the backend is running on port 80 now.
5. Now after saving both the files take the public IP of the front end and check whether the app is working on port 80 or not. It should show the output like below.



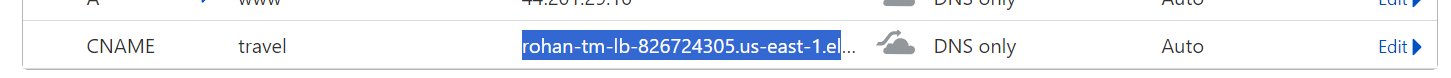
1. Now we will try to bind the IP with the domain name which we need to purchase from any of the website like hotslinger or godaddy or cloudflare etc.
2. Go to cloudfalre.com as they provide free domain as well. So sign up there and create the domain name in free plan. I have selected rohanops.com as domain.
3. Now you can configure DNS with subdomain and IP address for backend and then access the backend with that URL.
4. If you want to access frontend through the domain name you need to give dns for the frontend in cloudflare and then give the backend URL in /src/url.js file.
5. Make sure both the URLs backend and FrontEnd should be in the same domain. Now you can access the frontend with the URL given in cloudflare i.e. rohantmfe.rohanops.com



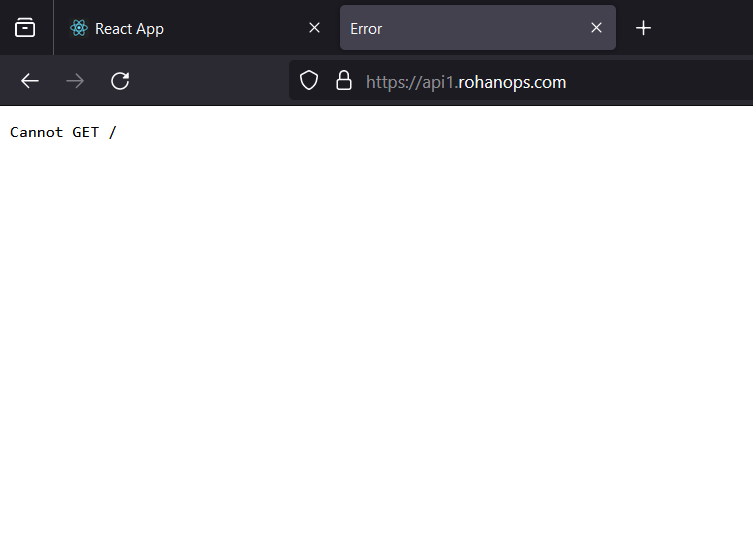
1. Now we will try to work with SSL means how to run the website with port 443.
2. Go to let's encrypt website and then select certbot <https://certbot.eff.org/> . It provides the certificate for free for 3 months and after that you can renew.
3. Select the software on which the app is running and OS as well. It will show the steps to install SSL. Follow all the steps given there.
4. First in cloudflare we have to give A record for domain and root as well. Then you can give domains while it asks for domain for which you want certificates.



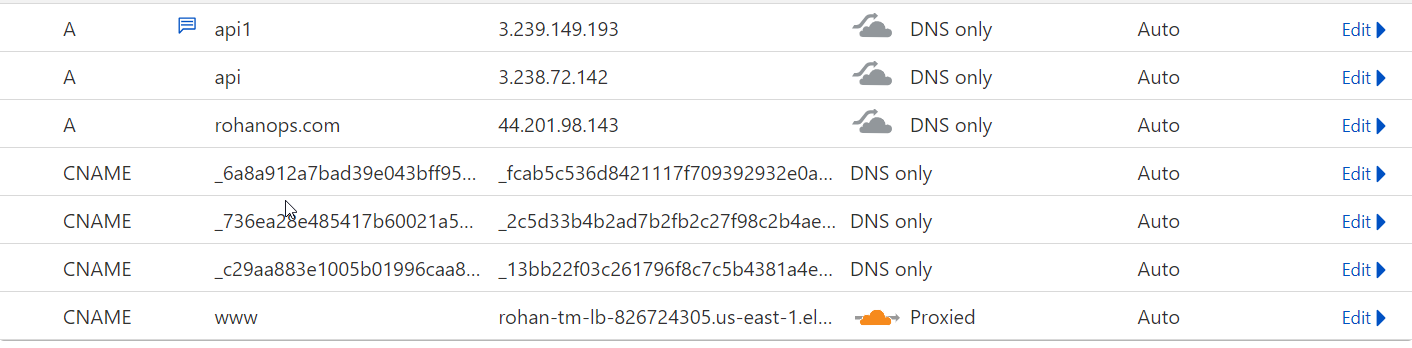
1. You can have certificate for all the subdomains as well. So while entering the domains you can mention the subdomain as well.
2. Now in the backend also you need to fetch the certificate for api.rohanops.com the name changed from rohantm to api in cloudflare itself.
3. Once the certificate got installed you can simply change the url.js in frontend and make backend url as <https://api.rohanops.com>.
4. Now if you will check both the URL will work on HTTPS.
5. Now we need to configure the load balancer so that we can achieve high availability for the website.
6. For that we have to clone the Ec2 instances for FE and BE by using AMI. There is an option in Actions>Images & Templates > Create Image. From that created image you can directly run the new instance.
7. Once the instances are up and running create a load balancer and create target group in which you can select two FE instances and once the load balancer is created you can make CNAME entry in cloudflare for that load balancer DNS.



2. Try to access the website through load balancer record name.



1. Now after configuring load balancer, we can add the port 443 as the traffic will be coming on 443. For that we have to use ACM for installing certificate. So we have to give domain and create new certificate for that domain and select that certificate while giving the port 443. For this I had given travel.rohanops.com and then once the certificate is created you can give that load balancer dns in to cloudflare also CNAME entry we need to do in cloudflare while configuring ACM i.e. AWS certificate manager.



1. Now once the entry is done in cloudflare try to up the app and then access the app. It should work. Basically, we have to create 2 load balancers one for backend and one for frontend and give the entry accordingly in cloudflare.