NAMASTE NODE.JS SEASON 3

Episode-2

Hand Written Notes

-By Shanmuga Priya

www.linkedin.com/in/shanmuga-priya-e-tech2

Episode - 2

Backend APP Deployment and connecting Frontend & Backend

Steps involved in deploying Backend APP:

we have already created an instance and cloned our backend project in previous episode now we will deploy it.

Step 1: Move to Backend folder & install necessary dependencies

```
npm install
```

Step 2: Copy Ip address to MongoDB

Inorder to run a project in instance we use npm start. Before that we need to allow access of our DB to the instance Ip Address. Save the local Ip address of instance in the DB atlas. Now the DB connection is successfull

Step 3: Enable the port

to enable the port go to security then securitygroups there we set a new inbound rules to include the port NO 7777.

Issue with current implementation:

whenever the terminal is closed the ~~server~~ app will also be shutdown but we cannot keep the terminal open all the time we need to run this application in the background all the time for that we use a package called "PM2".

what is PM2?

→ PM2 is a process manager that will help us manage and keep our application online 24/7. It is an open source Process manager for Node.js application.

```
npm install pm2 -g.
```

→

Step4: Start our app in Pm2.

Now we start our app in the Pm2 process manager to run our application 24/7 using

Pm2 start npm -- start

This command will run npm start in the background by creating a new process that keeps on running 24/7 in the background.

→ This make our app running even after the terminal is closed or logged out from the instance.

Pm2 logs → it gives logs of all running process.

Pm2 flush name of the process/app → to clear the logs.

Pm2 list → list of processess started by Pm2.

Pm2 stop processname → to stop the process.

Pm2 delete Processname → to delete the process.

we can also give a custom name to the process before starting

Pm2 start npm --name "devtinder" -- start.

Connecting frontend & backend

Step1: Configure nginx.

we need to configure nginx to map /api to port num 7777 inorder to ease access for users this way user need not deal with ports in URL's.

without nginx:

frontend: http://ex.com

Backend: http://ex.com:7777/

with nginx:

http://ex.com/api (both front end & backend)

Behind the scence when the browser sends a req to http://ex.com/api nginx receives this & forward to http://localhost:7777/api.

frontend → nginx → backend

req → ↓ ↳ ↓

Step a: Open the nginx configuration file

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

step b: edit the nginx file.

1) change the server name from _ to domainname (or) localIPaddress.

server_name Domainname/Ipaddress.

2) add the proxy-pass lines below it.

```
location /api/ {
    Proxy_pass http://localhost:7777/;
    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;
}
```

It says whenever you see a ~~location~~ URL /api just redirect it to localhost:7

step c: Save the file.

Ctrl + C → to exit it will ask to save it type Y. and enter.

step 2: Restart nginx.

after the nginx file updated we need to restart the nginx

sudo systemctl restart nginx

step 3: Replace the backend URL in the frontend file with the "/api".

const BASEURL = "http://localhost:7777/" → const BASEURL = "/api".

and push it to github.

step 4: Make a pull req in the instance terminal to get updated with latest commit.

git pull.

Step 5: Build the frontend once again.

As we updated the code in frontend inorder to reflect the changes made we need to bundle it again and copy the dist folder to nginx. like we did when deploying frontend.

```
npm run build

sudo scp -r dist/* /var/www/html.
```

Now, both frontend & backend are connected successfully.
