NodeJs Application Using Docker File

Introduction :

This document provides a guide for implementing a Docker container for the Node.js Chat App using the provided Dockerfile. The Dockerfile defines the environment and dependencies required to run the application within a Docker container.

Prerequisites:

- Docker installed on the host machine.
- Internet connectivity to download dependencies during the Docker build process.

Step 1:

Login to AWS Console. Start your docker instance.

Connect the aws docker instance.

Step 2:

In docker instance create a project directory. E.g. NodeJS_APP

 # mkdir NodeJS_APP

 # ls

```
root@ip-172-31-9-128:~# mkdir NodeJS_APP
root@ip-172-31-9-128:~# ls
Docker  NodeJS_APP  snap  webapp
root@ip-172-31-9-128:~#
```

Step 3:

Create Dockerfile inside the project directory.

 # cd NodeJS_APP/

 # vim Dockerfile

```
root@ip-172-31-9-128:~# cd NodeJS_APP/
root@ip-172-31-9-128:~/NodeJS_APP# vim Dockerfile
root@ip-172-31-9-128:~/NodeJS_APP#
```

Step 4 :

Paste the following content inside the Dockerfile

```
FROM ubuntu:latest
LABEL app="nodejs"
LABEL Author="Mayur"

RUN  apt update
RUN  apt install nodejs npm -y

RUN git clone https://github.com/owanhunte/nodejs-chat-app.git

WORKDIR nodejs-chat-app

RUN npm install

EXPOSE 3000

CMD [ "npm", "start" ]
```

Step 5 :

Create a docker image using docker build.

 # docker build .

```
root@ip-172-31-9-128:~/NodeJS_APP# ls
Dockerfile
root@ip-172-31-9-128:~/NodeJS_APP# docker build .
[+] Building 6.4s (5/10)
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 290B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [auth] library/ubuntu:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/6] FROM docker.io/library/ubuntu:latest@sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44eff17e
 => => resolve docker.io/library/ubuntu:latest@sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44eff17e
 => => sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44eff17e 1.13kB / 1.13kB
 => => sha256:aa772c98400ef833586d1d517d3e8de670f7e712bf581ce6053165081773259d 424B / 424B
 => => sha256:ca2b0f26964cf2e80ba3e084d5983dab293fdb87485dc6445f3f7bbfc89d7459 2.30kB / 2.30kB
 => => sha256:bccd10f490ab0f3fba61b193d1b80af91b17ca9bdca9768a16ed05ce16552fcb 29.54MB / 29.54MB
 => => extracting sha256:bccd10f490ab0f3fba61b193d1b80af91b17ca9bdca9768a16ed05ce16552fcb
 => [2/6] RUN  apt update
```

Step 6 :

List the images using

 # docker images

```
root@ip-172-31-9-128:~/NodeJS_APP# docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
<none>        <none>    e530af557417   3 minutes ago    964MB
<none>        <none>    4c8beb6a583a   10 hours ago     478MB
```

Step 7 :

Create a container using the created image.

 # docker run -d -p 3000:3000 <image id>

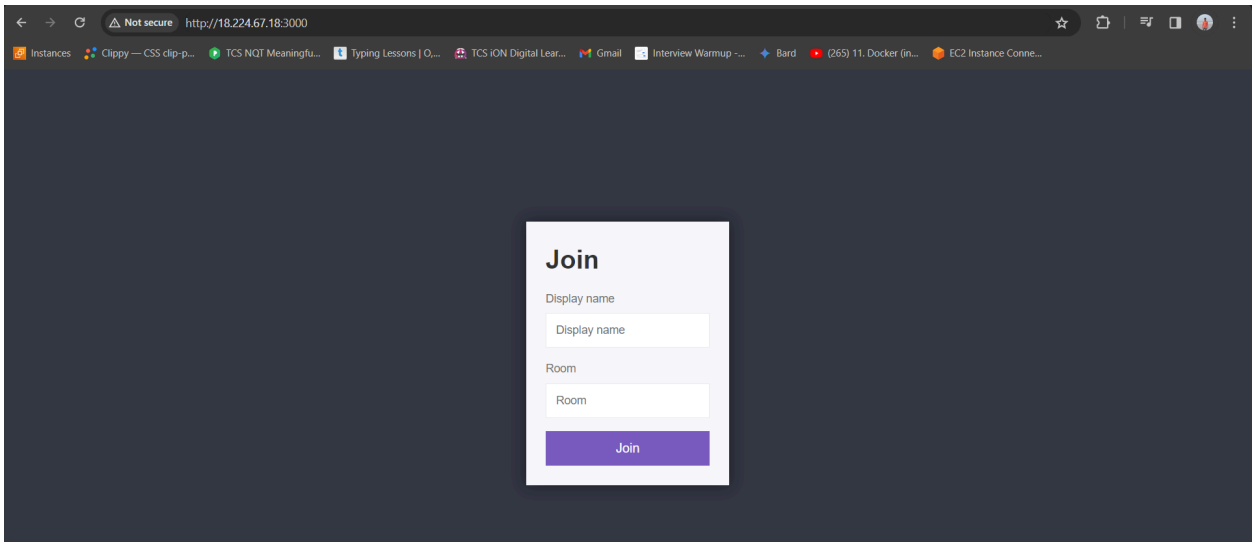
Eg.

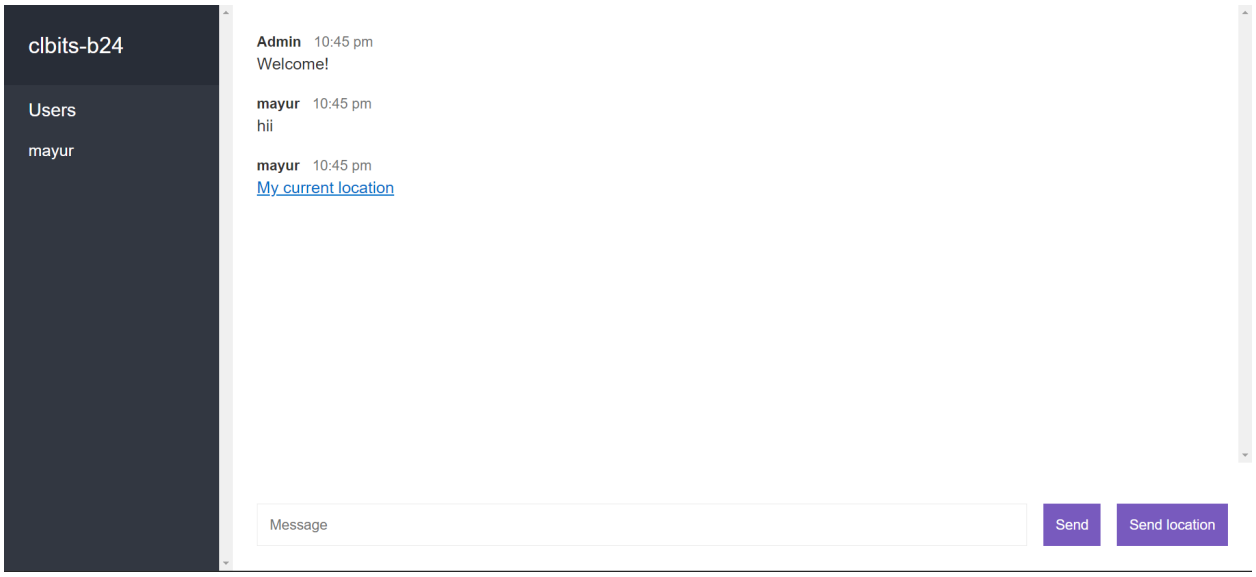
 # docker run -d -p 3000:3000 e53

```
root@ip-172-31-9-128:~/NodeJS_APP# docker run -d -p 3000:3000 e53
db8b8ef746fa7c90fdacdbdb21193df4af4ff30c642230363dfd0972bf3d49c6
root@ip-172-31-9-128:~/NodeJS_APP# docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED          STATUS         PORTS                                       NAMES
db8b8ef746fa   e53       "npm start"   10 seconds ago   Up 9 seconds   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   peaceful_knuth
root@ip-172-31-9-128:~/NodeJS_APP#
```

Step 8:
Check the application
http://<ip>:3000

In my case it is: ***http://18.224.67.18:3000/***





Project Github Repository :
https://github.com/owanhunte/nodejs-chat-app.git