**Deploy a Calculator application on a cloud virtual machine**

**(aws ec2) using Docker**

**Step 1:Create react app**

Create a calculator application using nodejs and react.

npx create-react-app react-docker-example

cd react-docker-example && npm install

npm start

Now your React app should be available at [<http://localhost:3000/>]

**Step 2: Docker --- to implement container**

Dockerized react application by creating docker image and docker hub to store data globally.

→ Now let's create a Dockerfile in the root of the app.

# build stage

FROM node:lts-alpine as build-stage

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

# production stage

FROM nginx:stable-alpine as production-stage

COPY --from=build-stage /app/build /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

→ I built a docker image.

docker build -t <docker hub username>/<repository name> .

→ run it.

docker run -p 3000:80 -d <docker hub username>/<repository name>:<tagname>

Now your React app should be available again at [<http://localhost:3000/>]

→ Next, we push the docker image to a repository. Let's use a docker hub public repository.

You need to docker login first with your user and create a public repository. We will be pushing our image to this repository.

→ Let's check the image ID first

docker image ls

We get a list of our images and their IDs

REPOSITORY TAG IMAGE ID

460021/calculator latest bf3e546c6845

→ Next we tag the image

docker tag <image id> <docker hub username>/<repository name>:<tagname>

→ Now we can push it to our docker hub public repository

docker push <docker hub username>/<repository name>:<tagname>

Now the image is pushed to a public repository accessible to everyone. We are going to be pulling it on our ec2 instance next.

**Step 3: Deploy on EC2**

I have an aws account and have launched and started an ec2 instance, installed docker.(sudo yum install docker -y)

→ Pull the previously created image from dockerhub.

docker pull <docker hub username>/<repository name>:<tagname>

→ Then, run it

docker run -p 80:80 -d <docker hub username>/<repository name>:<tagname>

That's it, since we bound it to port 80 the app should be running on the public IP of the instance now.

**My details:**

**Docker hub id : 460021**

**Repository name: calci**

**Image id: 565c2ace38b3**

**Github link: https://github.com/keerthi4600/test-project/tree/project**