Docker achievements so far…

1. Create a docker file for a simple React-based web app, build an image, run a container, and look for desired output in the web browser.

---------------------------------------------------------------------------------------------------------------------

**FROM node:19      //node-base image**

**WORKDIR /app      //work directory**

**COPY ./package\*.json ./  //copy all package.json files into work directory**

**RUN npm install //run the installed npm scripts**

**COPY . . //copy src folders**

**COPY . /app**

**CMD ["npm","run","start"] //command for running app**

**Commands:**

**To build an image,**

**docker build -t {image-name} .**

**To run a container,**

**docker run -d -p 3000:3000 --name {container-name} {image-name}**

---------------------------------------------------------------------------------------------------------------------

1. Create a docker file for a simple Node-based backend, build an image, run a container, and look for desired output by hitting a certain endpoint.

---------------------------------------------------------------------------------------------------------------------

**FROM node:16**

**WORKDIR /app**

**COPY package\*.json ./**

**RUN npm install**

**COPY . .**

**EXPOSE 8000**

**CMD ["npm","run","dev"]**

**Commands:**

**To build an image,**

**docker build -t {image-name} .**

**To run a container,**

**docker run -d -p 8000:8000 --name {container-name} {image-name}**

---------------------------------------------------------------------------------------------------------------------

1. Create a docker file that should be capable of running both the backend and frontend on the same port, also, they must be able to communicate in between.

---------------------------------------------------------------------------------------------------------------------

**# Fetching the latest node image on alpine linux**

**FROM node:alpine AS builder**

**# Declaring env**

**ENV NODE\_ENV production**

**# Setting up the work directory**

**WORKDIR /app**

**# Installing dependencies**

**COPY ./ui/package.json ./**

**RUN npm install**

**# Copying all the files in our project**

**COPY ./ui/. .**

**# Building our application**

**RUN npm run build**

**#create node js image**

**FROM node:14-alpine**

**WORKDIR /app**

**COPY ./api/package.json ./**

**RUN npm install**

**COPY --from=builder /app/build /app/public**

**COPY ./api/app.js ./**

**COPY NodEnginx.conf /etc/nginx/nginx.conf**

**RUN apk add --update nginx**

**EXPOSE 80**

**CMD ["sh", "-c", "nginx && node app.js"]**

**The file name here is NodeDocker (not compulsory )**

**Frontend folder is /ui**

**Backend folder is /api**

**-------------------------------------------------**

**Configuration file for enginx server for running frontend and backend on the same port:**

**user nginx;**

**worker\_processes auto;**

**error\_log /var/log/nginx/error.log;**

**pid /run/nginx/nginx.pid;**

**events {**

**worker\_connections 1024;**

**}**

**http {**

**include /etc/nginx/mime.types;**

**default\_type application/octet-stream;**

**sendfile on;**

**tcp\_nopush on;**

**tcp\_nodelay on;**

**keepalive\_timeout 65;**

**server {**

**listen 80;**

**server\_name localhost;**

**root /app/public;**

**index index.html;**

**location /api {**

**proxy\_pass http://localhost:3000;**

**proxy\_set\_header Host $host;**

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

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

**}**

**location / {**

**try\_files $uri /index.html;**

**}**

**}**

**}**

**The configuration file name should be NodEnginx.conf (for running both frontend and backend on the same port)**

**To build an image,**

**docker build --build-arg GIT\_USERNAME={gitlab\_username}--build-arg GIT\_PASSWORD={gitlab\_password}-f ./NodeDocker -t st:st1 .**

---------------------------------------------------------------------------------------------------------------------

**RUN DOCKER BUILDER PRUNE IN THE TERMINAL FOR CLEARING THE DOCKER CACHE**

1. Create a docker file that should be capable of Pulling a code of the backend and frontend  from GitLab, build a docker image

---------------------------------------------------------------------------------------------------------------------

**# Frontend**

**FROM node:alpine AS builder**

**ARG GIT\_USERNAME**

**ARG GIT\_PASSWORD**

**RUN apk --no-cache update && \**

**apk --no-cache add git curl**

**WORKDIR /app**

**RUN git clone https://gitlab.silvertouch.com/KaranShah/student\_management.git .**

**RUN git checkout main**

**RUN git pull**

**# RUN git fetch --all**

**RUN cd /app/Client && npm install**

**RUN cd /app/Client &&  npm run build**

**FROM builder AS builder-frontend**

**# Backend**

**FROM node:14-alpine**

**WORKDIR /app/Server**

**COPY --from=builder /app /app**

**COPY --from=builder /app/Client/build /app/public**

**RUN cd /app/Server && npm install**

**COPY NodEnginx.conf /etc/nginx/nginx.conf**

**RUN apk add --update nginx**

**EXPOSE 80**

**CMD ["sh", "-c", "nginx && DEBUG=express:\* node index.js"]**

**NODENGINX CONFIGURATION FILE WILL BE SAME AS ABOVE.**

**To build an image,**

**docker build --build-arg GIT\_USERNAME={gitlab\_username}--build-arg GIT\_PASSWORD={gitlab\_password}-f ./NodeDocker -t st:st1 .**

---------------------------------------------------------------------------------------------------------------------

1. Create a docker file that should be capable of running both the backend and frontend on the same port, also, they must be able to communicate in between. Moreover, data must be fetched from MongoDB.

---------------------------------------------------------------------------------------------------------------------

**# Frontend**

**FROM node:alpine AS builder**

**ARG GIT\_USERNAME**

**ARG GIT\_PASSWORD**

**RUN apk --no-cache update && \**

**apk --no-cache add git curl**

**WORKDIR /app**

**RUN git clone https://gitlab.silvertouch.com/KaranShah/student\_management.git .**

**RUN git checkout main**

**RUN git pull**

**# RUN git fetch --all**

**RUN cd /app/Client && npm install**

**FROM builder AS builder-frontend**

**FROM node:14-alpine**

**# Backend**

**WORKDIR /app/Server**

**COPY --from=builder /app /app**

**COPY --from=builder /app/Client/build /app/public**

**RUN cd /app/Server && npm install**

**COPY NodEnginx.conf /etc/nginx/nginx.conf**

**RUN apk add --update nginx**

**EXPOSE 80**

**CMD ["sh", "-c", "nginx && DEBUG=express:\* node index.js"]**

**-------------------------------------------------**

**Configuration file for enginx server for running frontend and backend on the same port:-**

**user nginx;**

**worker\_processes auto;**

**error\_log /var/log/nginx/error.log;**

**pid /run/nginx/nginx.pid;**

**events {**

**worker\_connections 1024;**

**}**

**http {**

**include /etc/nginx/mime.types;**

**default\_type application/octet-stream;**

**sendfile on;**

**tcp\_nopush on;**

**tcp\_nodelay on;**

**keepalive\_timeout 65;**

**server {**

**listen 80;**

**server\_name localhost;**

**root /app/public;**

**index index.html;**

**location /api {**

**proxy\_pass http://localhost:3000;**

**proxy\_set\_header Host $host;**

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

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

**}**

**location / {**

**try\_files $uri /index.html;**

**}**

**}**

**}**

**The configuration file name should be NodEnginx.conf (for running both frontend and backend on the same port)**

**For building an image:-**

**docker build --build-arg GIT\_USERNAME={gitlab\_username}--build-arg GIT\_PASSWORD={gitlab\_password}-f ./NodeDocker -t st:st1 .**

---------------------------------------------------------------------------------------------------------------------

1. Create a docker file that should be sufficient enough to do #4, although, the docker image should be with project builds not code in it.

---------------------------------------------------------------------------------------------------------------------

**# Frontend**

**FROM node:alpine AS builder**

**ARG GIT\_USERNAME**

**ARG GIT\_PASSWORD**

**RUN apk --no-cache update && \**

**apk --no-cache add git curl**

**WORKDIR /app**

**RUN git clone https://gitlab.silvertouch.com/KaranShah/student\_management.git .**

**RUN git checkout main**

**RUN git pull**

**# RUN git fetch --all**

**RUN cd /app/Client && npm install**

**RUN cd /app/Client &&  npm run build**

**FROM builder AS builder-frontend**

**FROM node:14-alpine**

**# Backend**

**WORKDIR /app/Server**

**COPY --from=builder /app /app**

**COPY --from=builder /app/Client/build /app/public**

**RUN cd /app/Server && npm install && npm test**

**COPY NodEnginx.conf /etc/nginx/nginx.conf**

**RUN apk add --update nginx**

**EXPOSE 80**

**CMD ["sh", "-c", "nginx && DEBUG=express:\* node index.js"]**

**-------------------------------------------------**

**Configuration file for enginx server for running frontend and backend on the same port:-**

**user nginx;**

**worker\_processes auto;**

**error\_log /var/log/nginx/error.log;**

**pid /run/nginx/nginx.pid;**

**events {**

**worker\_connections 1024;**

**}**

**http {**

**include /etc/nginx/mime.types;**

**default\_type application/octet-stream;**

**sendfile on;**

**tcp\_nopush on;**

**tcp\_nodelay on;**

**keepalive\_timeout 65;**

**server {**

**listen 80;**

**server\_name localhost;**

**root /app/public;**

**index index.html;**

**location /api {**

**proxy\_pass http://localhost:3000;**

**proxy\_set\_header Host $host;**

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

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

**}**

**location / {**

**try\_files $uri /index.html;**

**}**

**}**

**}**

**The configuration file name should be NodEnginx.conf (for running both frontend and backend on the same port)**

**To build an image,**

**docker build --build-arg GIT\_USERNAME={gitlab\_username}--build-arg GIT\_PASSWORD={gitlab\_password} -f ./NodeDocker -t st:st1 .**

---------------------------------------------------------------------------------------------------------------------

1. Create a React-based web app (including Jest test cases) and run test cases using docker. If it gets successful, create an image

---------------------------------------------------------------------------------------------------------------------

**# Frontend**

**FROM node:alpine AS builder**

**ARG GIT\_USERNAME**

**ARG GIT\_PASSWORD**

**RUN apk --no-cache update && \**

**apk --no-cache add git curl**

**WORKDIR /app**

**RUN git clone https://gitlab.silvertouch.com/KaranShah/student\_management.git .**

**RUN git checkout main**

**RUN git pull**

**# RUN git fetch --all**

**RUN cd /app/Client && npm install && npm test**

**RUN cd /app/Client &&  npm run build**

**FROM builder AS builder-frontend**

**FROM node:14-alpine**

**# Backend**

**WORKDIR /app/Server**

**COPY --from=builder /app /app**

**COPY --from=builder /app/Client/build /app/public**

**RUN cd /app/Server && npm install && npm test**

**COPY NodEnginx.conf /etc/nginx/nginx.conf**

**RUN apk add --update nginx**

**EXPOSE 80**

**CMD ["sh", "-c", "nginx && DEBUG=express:\* node index.js"]**

**-------------------------------------------------**

**Configuration file for enginx server for running frontend and backend on the same port:-**

**user nginx;**

**worker\_processes auto;**

**error\_log /var/log/nginx/error.log;**

**pid /run/nginx/nginx.pid;**

**events {**

**worker\_connections 1024;**

**}**

**http {**

**include /etc/nginx/mime.types;**

**default\_type application/octet-stream;**

**sendfile on;**

**tcp\_nopush on;**

**tcp\_nodelay on;**

**keepalive\_timeout 65;**

**server {**

**listen 80;**

**server\_name localhost;**

**root /app/public;**

**index index.html;**

**location /api {**

**proxy\_pass http://localhost:3000;**

**proxy\_set\_header Host $host;**

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

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

**}**

**location / {**

**try\_files $uri /index.html;**

**}**

**}**

**}**

**The configuration file name should be NodEnginx.conf (for running both frontend and backend on the same port)**

**To build an image,**

**docker build --build-arg GIT\_USERNAME={gitlab\_username}--build-arg GIT\_PASSWORD={gitlab\_password} -f ./NodeDocker -t st:st1 .**

**NOTE - we have to add  "scripts": { "test": "jest"} in package.json file for running test cases from local and docker as well and we have to include ex. RUN cd /app/Server && npm install && npm test**

---------------------------------------------------------------------------------------------------------------------

1. Send email notifications containing the status report of test cases

---------------------------------------------------------------------------------------------------------------------

**SAME CODE AS POINT NO 7, BUT WE DO NEED TO INCLUDE  "scripts": {"test": "jest --reporters=default --reporters=jest-html-reporter && node send-email.js || node send-email.js"} IN BOTH FRONTEND AND BACKEND FOLDERS’ package.json file.**

**CODE FOR SENDING EMAIL FOR TEST CASES.**

**const nodemailer = require('nodemailer');**

**const fs = require('fs');**

**const transporter = nodemailer.createTransport({**

**service: 'gmail',**

**auth: {**

**user: '{EMAIL\_ID}',**

**pass: '{PASSWORD}’,**

**}**

**});**

**const reportContents = fs.readFileSync('./test-report.html', 'utf8');**

**const mailOptions = {**

**from: `{CREDENTIALS}`,**

**to: "{CREDENTIALS}",**

**subject: "Client Test Results",**

**html: reportContents**

**};**

**transporter.sendMail(mailOptions, (error, info) => {**

**if (error) {**

**console.log(error);**

**} else {**

**console.log(`Email sent: ${info.response}`);**

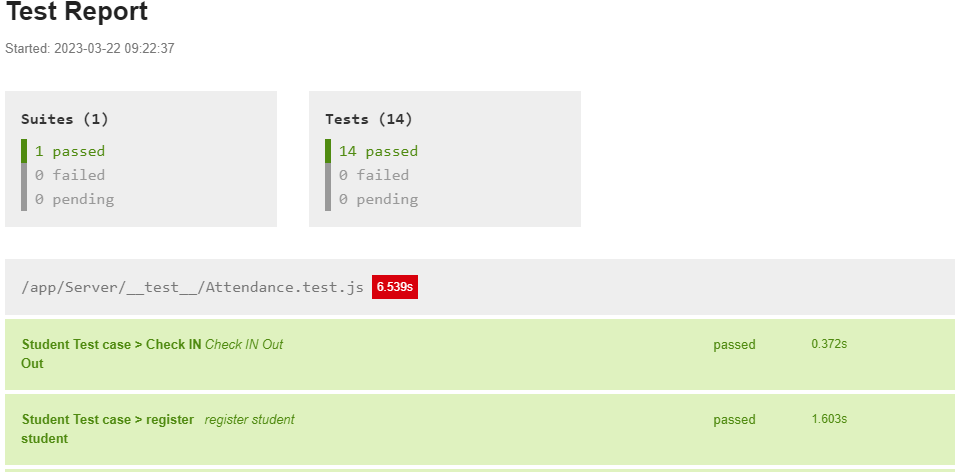
**}**

**});**

**FILE NAME SHOULD BE send-email.js**

It will generate report of case results and send it on given mail.

Mail will look like this.





---------------------------------------------------------------------------------------------------------------------

1. Find  the solution to communicate multiple docker images in between

---------------------------------------------------------------------------------------------------------------------

**SAME DOCKERFILE AND CONFIGURATION FILES AS ABOVE POINT BUT WE DO NEED TO INCLUDE ONE ANOTHER docker-compose.yml FILE.**

docker-compose.yml :-

version: '3'

services:

  app:

    build:

      context: .

      dockerfile: NodeDocker

    ports:

      - '80:80'

    depends\_on:

      - db

    environment:

      MONGODB\_URI: {mongo uri}

  db:

    image: mongo:4.4.4-bionic

    volumes:

      - db\_data:/data/db

volumes:

  db\_data:

**Command :- docker compose up**

This file will act as mediator between our main dockerfile and mongodb container. It will connect our main file to database.

---------------------------------------------------------------------------------------------------------------------