DEVOPS TWO TASK

Task document -1

1) Docker

TASK 1

Write a Dockerfile and run a container

Your task is as follows:

1. Create a Docker image:

● Use centos or Ubuntu as the base image

● Install apache web server

● Deploy any web application you want (NGINX )

2. Once you wrote the Dockerfile and created an image, run the container and test the

application. Describe how did you test it and provide output

###Process to do task###

#1st create dockerfile for above task 1

vi Dockerfile

FROM ubuntu

RUN apt update

RUN apt install -y apache2

COPY . /var/www/html/

EXPOSE 80

CMD ["apache2ctl", "-D", "FOREGROUND"]

#now build the image of above file by using below cmd

-->docker build -t myimage2 .

#check the docker image is created or not for that use cmd

-->docker images

#in output you will see --> myimage2

#Now create a container by using above docker image

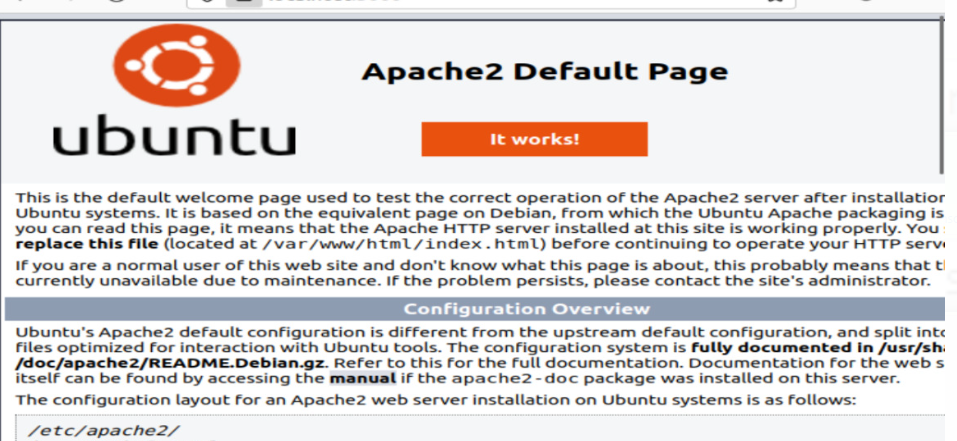
-->docker run -itd --name container23 -p 80:80 myimage2

#Now check the container is created or not by using below cmd

-->docker ps

#and finaly your container is working so access on browser

localhost:80



To troubleshoot container use cmd

* docker

\*\*\*\*\*Task 1 with nginx \*\*\*\*\*\*

#########for this task use the same process as above task ############

1st create a Docker file for nginx

vi Dockerfile

FROM nginx:alpine

COPY ./index.html /usr/share/nginx/html

##This is index.html file to access on nginx page##

Vi Index.html

!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Docker Nginx</title>

</head>

<body>

<h2>Hello from Nginx container. how are you all</h2>

</body>

</html>>

Then: wq enter and you will see your index.html file is created.

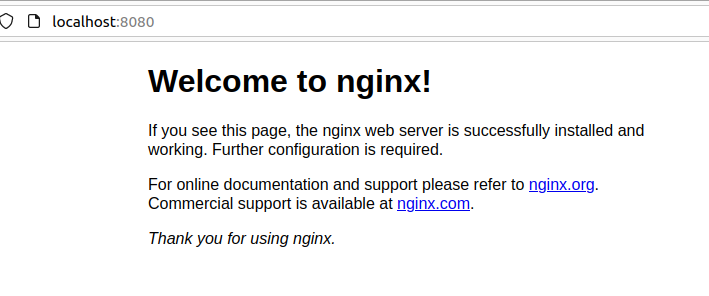
To build the docker image from docker file

-->docker build -t nginximage5 .

🡪docker run -itd --name nginxcontainer -p 8080:80 nginximage5

#and finaly your container is working so access on browser

--->>localhost:8080



TASK 2

2) TASK 2

https://github.com/kumail1232/slack-docker

● Clone this repo

● Create a Dockerfile to Dockerize a react application

● Run that container using a Docker Compose file

● upload this task on a github repo and push these files from local to remote repo

https://github.com/kumail1232/slack-docker

● Clone this repo

● Create a Dockerfile to Dockerize a react application

● Run that container using a Docker Compose file

#1st create a docker file for your 2nd task

vi Dockerfile

FROM node:18-alpine

WORKDIR /frontend

COPY ./\*.json /frontend/

RUN npm install

#RUN npm audit fix --force

RUN yarn add react-scripts

# add app

COPY . .

EXPOSE 3000

CMD ["npm", "start"]

Details of Docker dockerfile

Build an image starting with the node:18-alpine

Set the working directory to /frontend

Copy ./\*.json to workdir

Install npm install

react script

Copy the current directory . in the project to the workdir .

container is listening on port 3000

(it allows you to experiment with express without root access)

Set the default command for the container to "npm", "start".

#Build the docker image by using above file

--->docker build -t reactimage .

##check the docker image is created or not for that use cmd

-->docker images

#output --> reactimage

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

#Now create a docker-compose file for task two

Docker compose file

docker-compose.yml

version: '3.3'

services:

FrontEnd:

image: react-image

ports:

- "3000:3000"

To build the stack, go back to the terminal window, make sure you’re in your build directory, and issue the command:

#docker-compose up

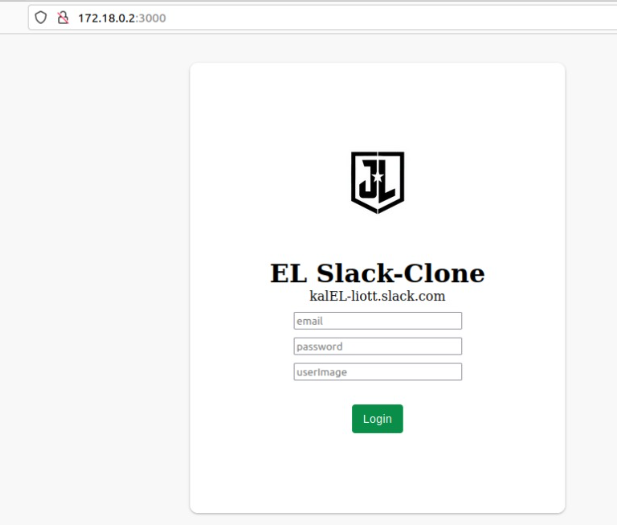
The above command will deploy both the web and db containers.

If you only run that command, the containers will deploy in attached mode, so you won’t get your bash prompt returned. If you want to run them in detached mode, you’d issue the command:

docker-compose up -d

#and finally your container is working so access on browser

--->>localhost:3000



\*\* TO TRUBELSHOOT ANY CONTAINER USE FOLLOWING CMD

# docker inspect caintanername

Now to push your files and document in GIT GUB

#1st create a new repository -->> sty-task-repo

#initialise that repo

--> git init

#now check status of your git acc.

-->git status

#add your files on in staging area

--> git add .

#give commite to your code/document

-->> git commit -m "my 1st commite"

#now add your local repo to remote repository so your

--> git remote add origin and enter your created repo URL

now you are connected to your repo in github.

#Now push your code in gitrepo

--> git push -u origin master

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

Challenges at the time of task.

#when I try to push my code it asks me for token then i create a personal access token and end then my code is pushed.

#no other error come in this task