Challenge-1:

Please find below the detailed steps to host a static html page on the nginx web server:

1. Go to the following link and download the latest version of Docker desktop for your OS:

<https://www.docker.com/products/docker-desktop/>

1. Follow the instructions in the docker installer and complete the installation with the default configurations provided.
2. Run the docker desktop application and verify that it is installed correctly by opening a command prompt and typing the following command:

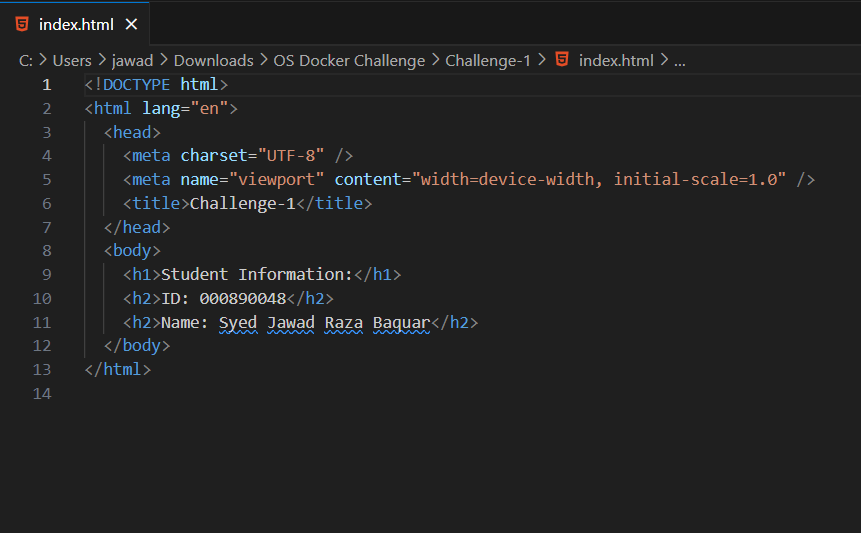
docker –version

A black background with white text

Description automatically generated

Seeing the docker version as output verifies that our installation was successful.

1. Create a file named ‘index.html’ with the following content:



This is a simple html file that display the student’s information such as ID and name.

1. Create a ‘Dockerfile’ in the same directory as your ‘index.html’ file with the following content:

A screenshot of a computer

Description automatically generated

The ‘Dockerfile’ contains the instructions that are used to build layers for the docker image that will later be hosted in the container.

‘FROM’ command tells docker to look for the nginx web version in the Docker hub repository. We are using the stable-apline version of the nginx server.

‘COPY’ command copies our ‘index.html’ file into the nginx server’s directory which the web server uses to store and host static content like html files.

‘EXPOSE’ command documents the fact that we will host the image inside the container on port 80. We will have to map this port with a port on the host machine that the nginx server will use to listen to any requests.

1. Navigate to the directory that has your ‘index.html’ and ‘Dockerfile’ and run the following command to build the docker image:

docker build -t challenge-1 .

A screenshot of a computer

Description automatically generated

‘docker build’ is used to build docker images using the instructions provided in the ‘Dockerfile’.

‘-t challenge-1’ tells docker that we want to tag or name our image as challenge-1.

‘ . ’ specifies the build context. It tells docker to look for the ‘Dockerfile’ in the current directory.

1. Run the following command to run the docker container:

docker run -d -p 8080:80 challenge-1

A black background with blue text

Description automatically generated

‘docker run’ command is used to create a docker container and start the container using the provided docker image.

‘-d’ tells docker to run the container in detached mode or in the background. Which means the container keeps running even after we close the command prompt.

‘-p 8080:80’ flag tells docker that the docker container will be published on port 8080 on the host machine which will be mapped to port 80 on the container. Any request on port 8080 will be forwarded to port 80 on the container.

‘challenge-1’ is the name of the image from which the container will be created and run.

1. Navigate to ‘http://localhost:8080/’ in a web browser to see the output shown below:

A screenshot of a computer

Description automatically generated

1. Run the command ‘docker ps -a’ to list all the services or docker containers we have created so far. See which containers are still in running status. Note down their Container ID. Then run the command ‘docker stop <Container ID>’ to stop that container.

Verify that no container are currently running by using the command ‘docker container ls’

A black screen with white text

Description automatically generated

We perform the above step to free up any ports that we will need to use in the future.

Challenge-2:

Please find below the step by step instructions to host an api server using node and using nginx as a web server to act as reverse proxy and redirect all the requests it receives to the api-server:

1. Create a simple node server application with the following code:

A screen shot of a computer program

Description automatically generated

1. In the same directory as our node server, create a ‘Dockerfile.api-server’ file with the following code:

A screenshot of a computer program

Description automatically generated

This ‘Dockerfile.api-server’ will be used to build the image for our api-server and run it in a container.

We use the ‘node:lts-alpine’ version of the node server to host our APIs.

We set our working directory as /app so that all further commands and actions will be performed in this directory in the container.

We copy all the files in our directory to use while building the image.

Then we install any dependencies needed to run our application.

We document the port that will be used by this image in the container.

Then we provide ‘node app.js’ command to run our API server once the image has been built.

1. Create a ‘nginx.conf’ file in the same directory with the following code:

A computer screen shot of a code

Description automatically generated

We are basically configuring our nginx web server to work as reverse proxy. It will listen to network traffic on port 8080 and redirect all the requests it receives to our node api-server running on port 3000.

1. Create a ‘docker-compose.yml’ file in the same directory with the following code:

A screen shot of a computer

Description automatically generated

We use docker compose to create and run two services, our api server and our web server as reverse proxy together at the same time. Docker compose can be used to create multiple containers as services and combine them together as one service or container and host it.

We use the ‘Dockerfile.api-server’ file that we created to build and run our api-server service.

We use the ‘nginx:stable-alpine’ version of the nginx server to create the image for our web server. It will listen on port 8080.

We use volume mount feature to use our ‘nginx.conf’ file in the current directory as the configuration file for our nginx server and provide the server with read-only permission to our configuration file.

1. Open command prompt and navigate to the directory that has all our files. Run the following command to use docker compose to build and run our services:

docker-compose up -d

A screenshot of a computer

Description automatically generated

1. We can see that both our services our running. Use the command ‘docker ps’ to verify that both out containers are indeed up and running.
2. Open a web browser. When we navigate to ‘http://localhost:8080/api/books’ we see that all the books in our list are displayed.

A screenshot of a computer

Description automatically generated

1. When we navigate to ‘http://localhost:8080/api/books/<Book ID>’ we can see that the book with the provided Book ID is displayed.

A white and blue screen

Description automatically generated

A white and blue screen

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Run the command ‘docker-compose down’ to stop all the services and free up the ports for future use.

A black screen with white text

Description automatically generated

References:

<https://www.freecodecamp.org/news/the-docker-handbook/>