# Solution to Challenge-1

Steps to be reproduces to produce a solution for challenge-1:-

1. Check if the host machine have docker installed by executing the following command in the administrator terminal
    - docker --version

If the terminal provides with a string showcasing a version of docker, the docker is installed in the host machine as

```
PS C:\Users\amrit> docker --version
Docker version 26.1.1, build 4cf5afa
```

Otherwise Visit:-

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

And download docker desktop from there and follow standard installation instructions.

2. Once confirmed the host machine have a docker stable version, run the docker desktop application
3. Now navigate to the directory of the challenge 1 folder.
4. Open this directory in the vs code.
5. Create a folder named '*public*'
6. In that folder proceed to create a index.html file.
7. In the index.html proceed to create a html static page by using ! emmet.
8. In the body write your name and student ID.
9. Create a Dockerfile with no extension in challenge 1 folder.
10. In that file proceed with the following code:-

    FROM nginx:latest // This creates an image of nginx distribution[1]

    COPY public /usr/share/nginx/html //This copies the public folder to the shared nginx folder to host it on a particular port

    EXPOSE 80 //This exposes the port 80 to the html requests for the localhost

    CMD ["nginx", "-g", "daemon off;"] // This command runs the default nginx container with the config of daemon off to run nginx in foreground
11. Now proceed to the terminal of VS code and build this image by running the following command[1]
    - docker build -t solution-01 .

This command creates a docker image with the following command configured dockerfile

```
PS C:\OS\Solutions\docker-challenge-template> docker build -t solution-01 .
```

12. Now start this image with the following command:-
    - docker run -p 8080:80 -d –name docker-container solution-01
      This command runs a new container from the solution-01 image.

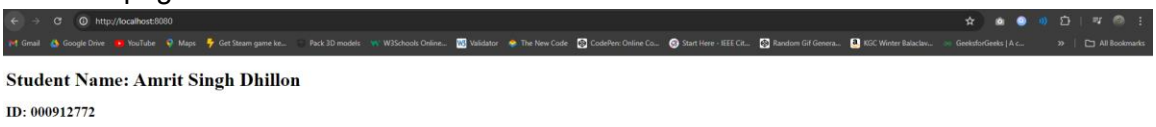
-p 8080:80: Maps port 8080 on the host machine to port 80 in the container.
-d: Runs the container in detached mode so the terminal does not gets bombarded with the logs.
--name docker -container: Gives the container the name docker -container

```
PS C:\OS\Solutions\docker-challenge-template> docker run -p 8080:80 -d --name docker-container solution-01
```

13. If the terminal produces a string of character then the docker container is running

```
PS C:\OS\Solutions\docker-challenge-template> docker run -p 8080:80 -d --name docker-container solution-01
53f00f92309163abc3617e972d871af13a464de182fbea1d76de86c677874a4a
```

14. Now if you open any browser in host machine and goto localhost:8080/ then you can see that html page rendered in the browser




And with this it concludes the ending of challenge 1.

## Solution to Challenge-2

The following steps can be followed in order to proceed a solution to challenge 2

1. Proceed with the step 1 and 2 from the challenge 1 solution and this concludes the setting up of our docker environment.
2. As provided challenge2.zip, extract it into an appropriate directory.
3. Now proceed to open this folder in VS Code where we begin our journey.
4. Create a Dockerfile in this directory and write the following code :-

```
FROM node:latest
WORKDIR /app
COPY package.json ./
RUN npm i
COPY . .
```

```
EXPOSE 3000
CMD npm start
```

*Explaining futher, this code works as follows, the first line FROM node:latest pulls the node image from the docker hub and then it instantiates this image into the docker once started. Now we proceed to separate our work directory to /app directory inside the image file structure. After that we copy the package.json into this app directory that provides us with all the dependencies required for this node server to execute properly and after that we install these packages by running npm I command that is also npm install and then we copy all the files particularly the sever.js into this /app folder. Now we expose port 3000 for this server and finally we proceed to start this server by executing npm start command.*

5. Now once this dockerfile is successfully created, we go onto creating the docker compose file that is going to include the instructions for our api and nginx services.
6. Create new file named docker-compose.yml in the host folder with challenge2 files.
7. In this file write the following code:-

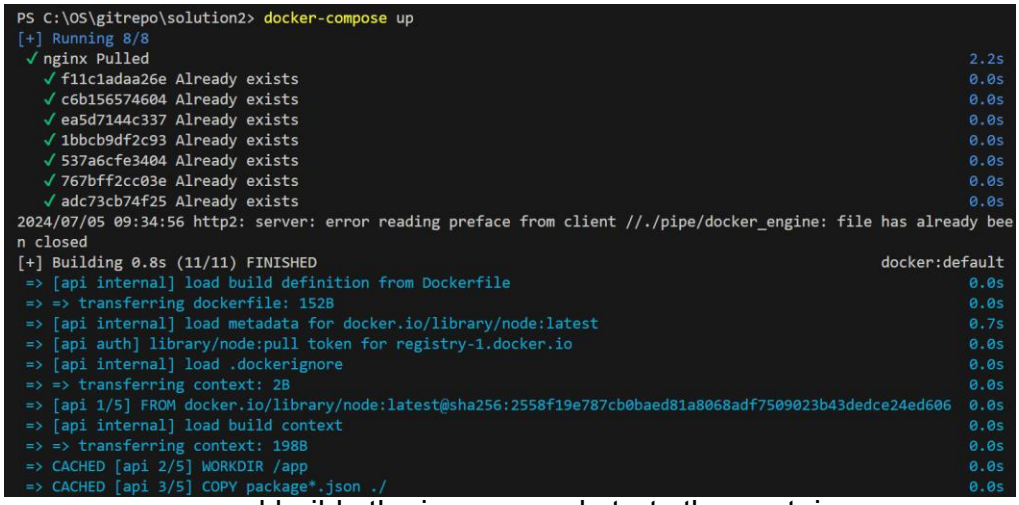
```
events {
        worker_connection 1024;
}
services:
        api:
          build: .
          ports:
                -“3000:3000”
          Environment:
                - NODE_ENV=production
        nginx:
          image: nginx:latest
          ports:
                - “8080:80”
          volumes:
                - ./nginx.conf:/etc/nginx/nginx.conf:ro
          depends_on:
                - api
```

This file includes the instructions for docker compose how it is going to operate once we initiate this docker compose

8. Once we have stated the instructions for the docker compose, we also need to instruct the nginx, how this container is going to map the services of the node server to the host machine for particular urls.
9. For that we create another file nginx.conf and write the following instructions:

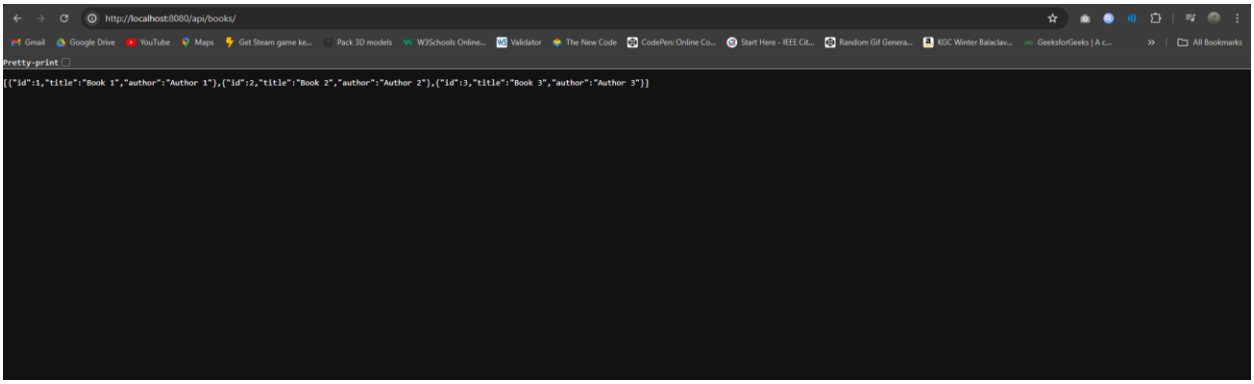
```
http{
        upstream api{
                server api: 3000;
        }
        server{
                listen: 80;
                location /api {
                        proxy_pass http://api;
                        proxy_http_version 1.1;
                        proxy_set_header Upgrade $http_upgrade;
                        proxy_set_header Connection 'upgrade';
                        proxy_set_header Host $host;
                        proxy_cache_bypass $http_upgrade;
                }
        }
}
```

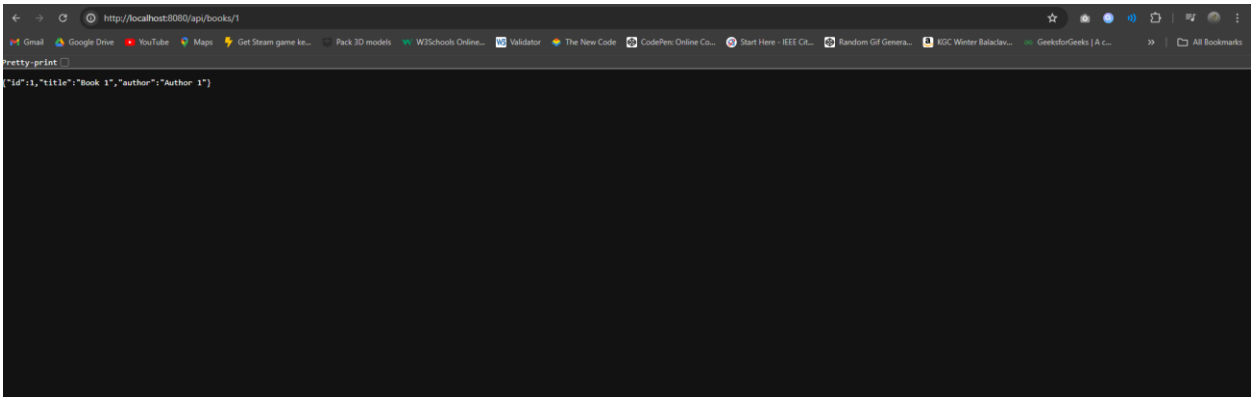
   This file includes the instructions of how a request from port 80 is going to be mapped to the server in simple sense this only instructs the structure of the http req sent to the server running on port 3000.
   This nginx.conf file configures nginx to act as a reverse proxy, forwarding requests from http://localhost:8080/api to http://api:3000.
10. Now once these files are ready proceed to initiate this compose
    - docker-compose up




This command builds the images and starts the containers

11. Once the containers are up and running proceed to open a browser on host machine and go to http://localhost:8080/api/books to test the containers. This shows the JSON of all the books and then again go to http://localhost:8080/api/books/1 to check if the server responds with the JSON of only one book.




12. With this we have completed the challenge 2.

# References

[1] P. McKee, "How to Use the NGINX Docker Official Image ," docker. Accessed: Abbreviated July 3,2024.. [Online]. Available: https://www.docker.com/blog/how-to-use-the-official-nginx-docker-image/