# Assignment: Containerization with Docker

### Some resources

* <https://www.freecodecamp.org/news/the-docker-handbook/>
* <https://www.youtube.com/watch?v=pTFZFxd4hOI>
* <https://www.geeksforgeeks.org/docker-tutorial/>
* <https://www.tutorialspoint.com/docker/>

### Guidelines

Here are some basic rules for these challenges:

1. This is an **individual assignment**.
2. At the end of each challenge, we expect that each student will deliver:
   1. Their personal, public repository on GitHub, with all the necessary files to make the produced applications work.
   2. A report with all the steps that they went through, to achieve the results. See more details about this journal below.
   3. Answer to eventual questions.
3. **Copies (or shared work) will not be tolerated**.

#### Report guidelines

Think of your report as a tutorial for someone who has never used Docker before and needs some guidance.

Submitted on D2L:

* Your report as a single file in PDF format:
  + You can add links to web pages used as references.
  + Add screenshots of your terminal, and your browser, showing the commands you used, and the results.

# Challenges

## Challenge 1 - Simple static page server

### Steps

* Use the folder challenge1.
* Setup Docker Environment : Check the docker version in your device by running the – docker --version command in the command prompt.

A black screen with white text

Description automatically generated

* Run the docker desktop application.
* Navigate to the directory of Challenge 1 Folder in the VS code.
* Create a "public" folder in this directory under challenge 1.
* Create an index.html file under this folder.
* Create a static page by using !element.
* In the body write your name and student ID in h1 and p tags respectively.
* Create another file in the challenge 1 directory and name it as dockerfile.
* Write the following code in that file : FROM nginx:latest #This will create an image of nginx distribution

COPY public /usr/share/nginx/html #This will the public folder to the shared nginx folder on a particular port

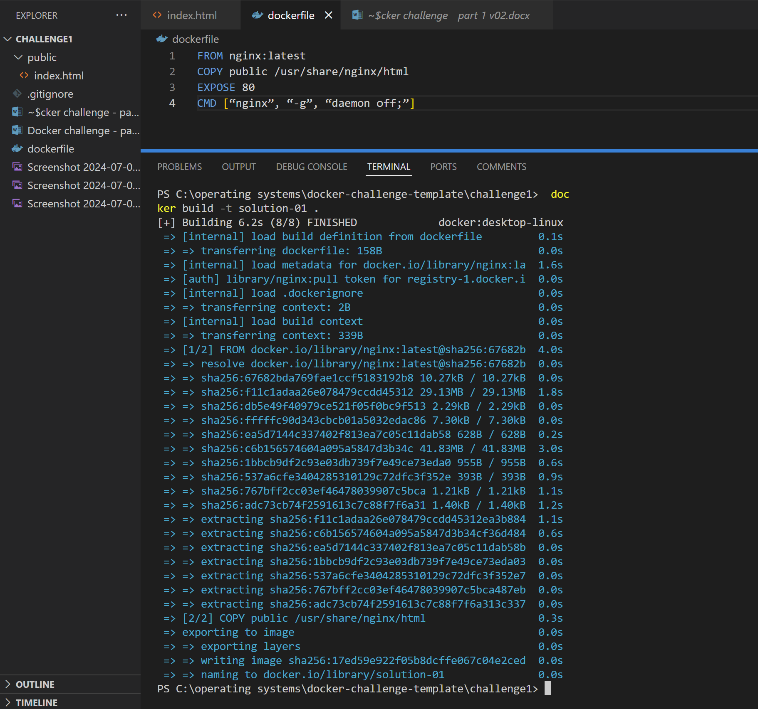
EXPOSE 80 #This will expose port 80 to the html requests for the localhost

CMD [“nginx”, “-g”, “daemon off;”] # With the daemon configuration turned off, this command will launch the default nginx container and execute nginx in the foreground.

A screenshot of a computer

Description automatically generated

* Now run this command in the terminal of the VS code : docker build -t solution-01 .
* This will build the image in the docker.



* To start this image run the following command:

docker run -p 8080:80 -d –name docker-container solution-01

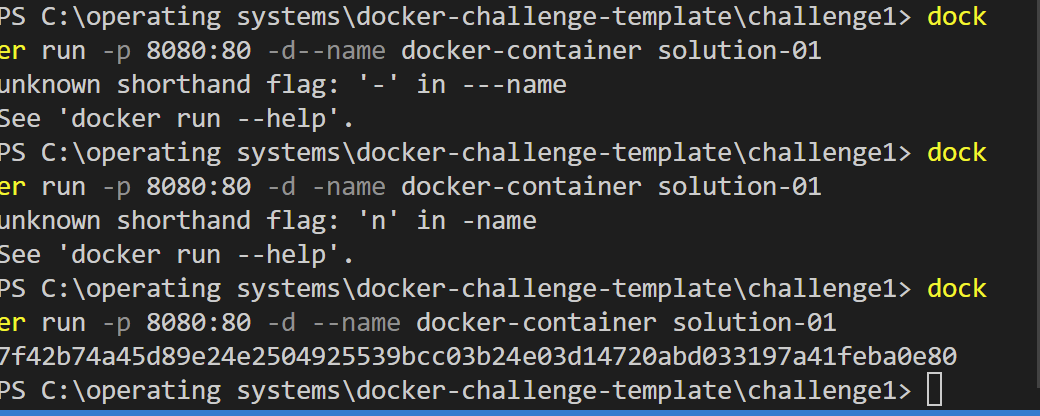
This command launches a new container from the solution-01 image.

-p 8080:80: Binds port 8080 on the host to port 80 within the container.

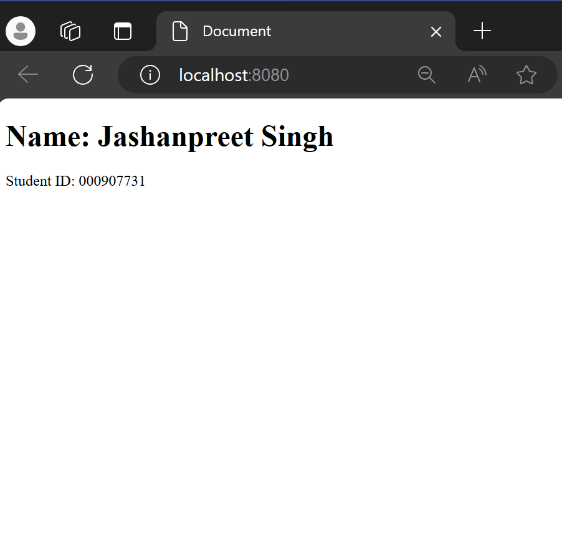
-d: Operates the container in detached mode, preventing the terminal from being overwhelmed with logs.

--name docker-container: Assigns the name docker-container to the running container.

* The result will be shown as a string.



* Open any browser in the host machine and run localhost:8080, the output will be shown as an html page output oon pport 8080.



This output completes our challenge 1.

## Challenge 2 - NodeJS application

* Setup the environment as we did in the challenge1.
* Extract the files in the directory named Challenge2.
* Open the extracted directory in Visual Studio Code to begin the setup.
* In the project directory, create a file named Dockerfile with the following content:

FROM node:latest

WORKDIR /app

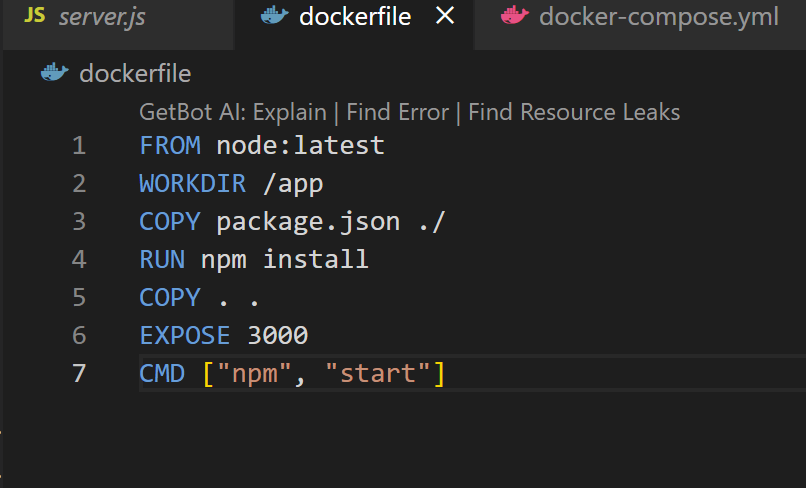
COPY package.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]



* This Dockerfile performs the following actions:

Pulls the latest Node.js image.

Sets the working directory to /app.

Copies package.json into the working directory.

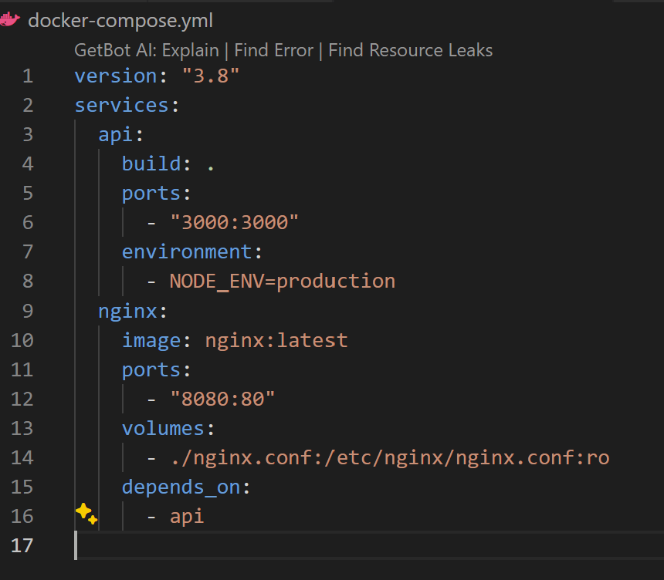
Installs the necessary dependencies.

Copies the rest of the application files.

Exposes port 3000.

Starts the application with npm start.

* In the same directory, create a file named docker-compose.yml with the following content:



This file defines the services for Docker Compose:

api: Builds the image from the current directory, maps port 3000, and sets the environment variable.

nginx: Uses the latest NGINX image, maps port 80 in the container to port 8080 on the host, and mounts the nginx.conf file. It depends on the api service.

* In the project directory, create a file named nginx.conf with the following configuration:

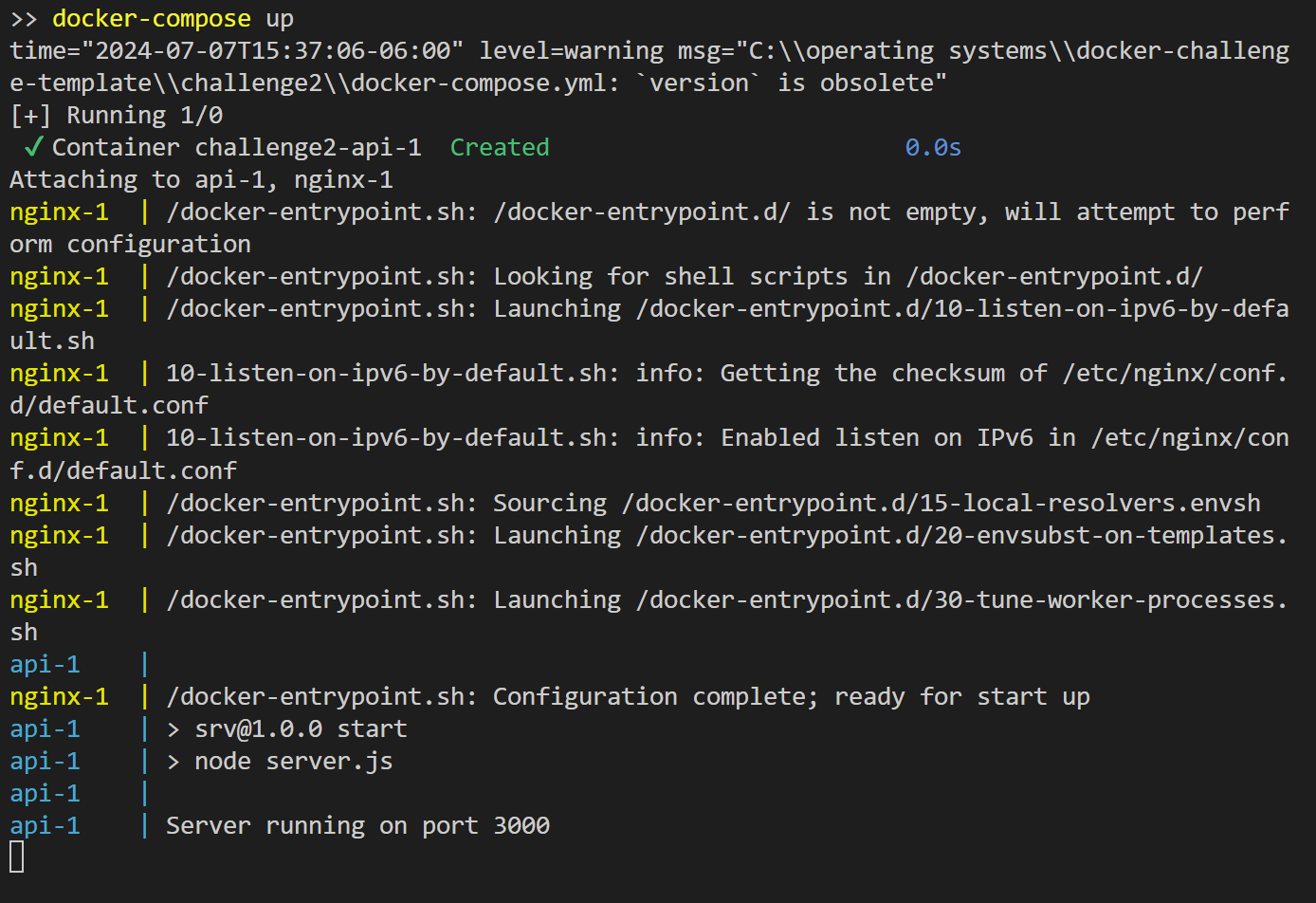
A screen shot of a computer

Description automatically generated

This configuration file sets up NGINX to act as a reverse proxy, forwarding requests from http://localhost:8080/api to the Node.js API running on http://api:3000.

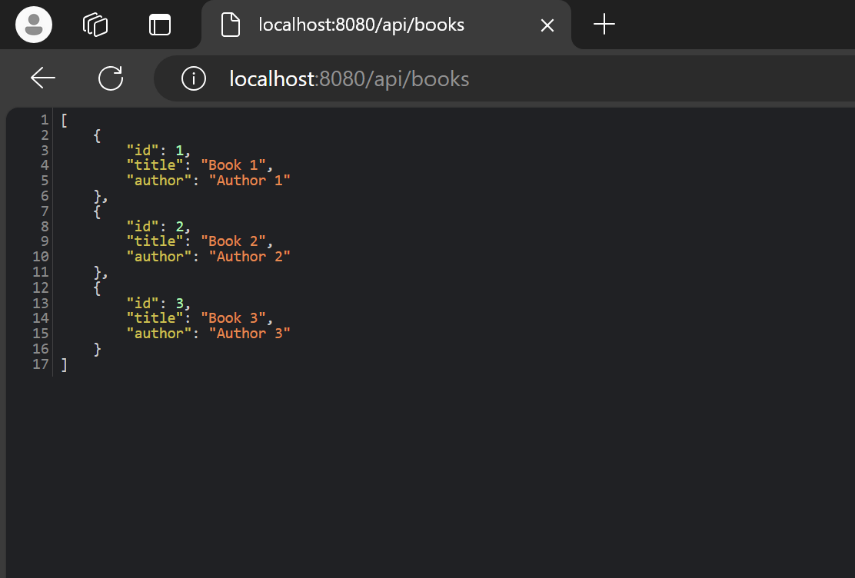
* Start the Docker services by running the following command in the terminal:

docker-compose up



* This command builds the images and starts the containers as defined in the docker-compose.yml file.
* Open a web browser and navigate to the following URLs to verify the setup:

http://localhost:8080/api/books: Should display a JSON list of books.



http://localhost:8080/api/books/1: Should display the JSON of a specific book with ID 1.

