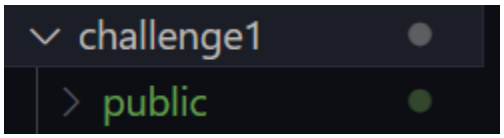
Docker challenge

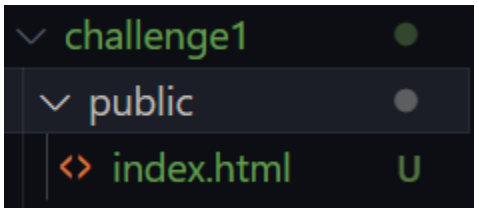
Challenge 1

Steps to complete challenge 1

1. Fork the repo and clone it to your machine: https://github.com/eduluz1976/docker-challenge-template
2. Open the project on visual studio code
3. In the challenge 1 folder, create another folder called public.



4. Next, create another file inside the public folder called index.html



5. In that index.html file, make sure the page can display your name and id:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Challenge 1</title>
  </head>
  <body>
    <h1>Challenge 1</h1>
    <p>id: </p>
    <p>name: </p>
  </body>
```

6. Create a dockerfile and configure it with this:

```
FROM nginx:alpine
COPY ./public /usr/share/nginx/html
EXPOSE 8080
```

7. To create the image, run this command:

   docker build -t challenge1-webserver:v1 .

The output should look like this:

```
+] Building 1.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 106B
=> [internal] load metadata for docker.io/library/nginx:alpine
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 259B
=> CACHED [1/2] FROM docker.io/library/nginx:alpine@sha256:31bad00311cb5eeb8a6648beadcf67277a175da89989f14727420a80e2e76742
=> [2/2] COPY ./public /usr/share/nginx/html
=> exporting to image
=> => writing image sha256:d247b5c0d4f36c1dee53ed83e5ab0273f3e5642f87f877693b9cf561b347994e
=> => naming to docker.io/library/challenge1-webserver:v1

View build details: docker-desktop://dashboard/build/default/default/kk08ny225lh9b0eqq71uqwxr9
```

8. To run, use this command:

```
PS C:\docker-challenge-template\challenge1> docker run -d -p 8080:80 --name webserver challenge1-webserver:v1
>>
```

Output:

```
33d0c5abb019be7e6b2f40e9c0d014cacd3f0ff16e85
```

9. To test, navigate to http://localhost:8080/
   The site should look like:

Challenge 1

id: 000871973

name: arvie sangalang

Challenge 2

1. Unzip and add the challenge2.zip files in the challenge 2 folder
2. Create a docker file in the challenge 2 root.
3. Configure the dockerfile so that it can run the node server:

```
FROM node:14
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 8080
ENV NODE_ENV=production
CMD ["npm", "start"]
```

4. Create a file called docker-compose.yml in the challenge2 root folder. This file is responsible for defining multi-container applications [1].
5. Configure your docker-compose.yml file, so the app runs on port 3000 and nginx to run on 8080:

```
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3000:3000"
  nginx:
    image: nginx:alpine
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    ports:
      - "8080:80"
    depends_on:
      - app
```

6. Create a nginx.conf file in the challenge2 root folder. This file is responsible for configuring the nginx server [2].
7. Configure the nginx.conf file:

```
events {}

http {
    server {
        listen 80;

        location / {
            proxy_pass http://app:3000;
            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;
        }
    }
}
```

8. Once all is said and done, this command needs to be called in order to build the images and start the containers:

docker-compose up --build
The output should look like this in the terminal:

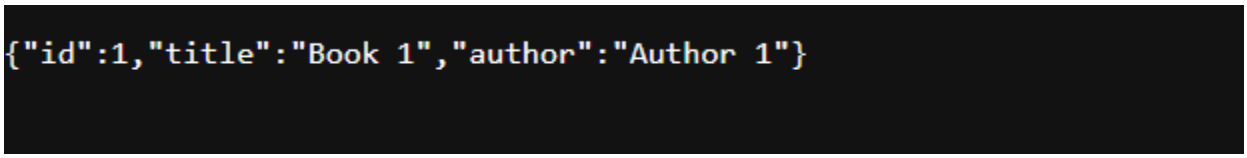
```
[+] Building 0.0s (0/0)  docker:default
[+] Building 1.5s (11/11) FINISHED                                                                                                  docker:default
 => [app internal] load build definition from Dockerfile                                                                                      0.0s
 => => transferring dockerfile: 185B                                                                                                          0.0s
 => [app internal] load metadata for docker.io/library/node:14                                                                                1.2s
 => [app auth] library/node:pull token for registry-1.docker.io                                                                               0.0s
 => [app internal] load .dockerignore                                                                                                         0.0s
 => => transferring context: 2B                                                                                                               0.0s
 => [app 1/5] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa                          0.0s
 => [app internal] load build context                                                                                                         0.0s
 => => transferring context: 383B                                                                                                             0.0s
 => CACHED [app 2/5] WORKDIR /usr/src/app                                                                                                     0.0s
 => CACHED [app 3/5] COPY package*.json ./                                                                                                    0.0s
 => CACHED [app 4/5] RUN npm install                                                                                                          0.0s
 => [app 5/5] COPY . .                                                                                                                        0.1s
 => [app] exporting to image                                                                                                                  0.1s
 => => exporting layers                                                                                                                       0.0s
 => => writing image sha256:382762b5952d3359e08038bd3d5c579fec8cfed30453276d4da8ba7abd9cbb84                                                  0.0s
 => => naming to docker.io/library/challenge2-app                                                                                             0.0s
[+] Running 2/2
 ✓ Container challenge2-app-1    Recreated                                                                                                    0.2s
 ✓ Container challenge2-nginx-1  Recreated                                                                                                    0.2s
Attaching to app-1, nginx-1
```

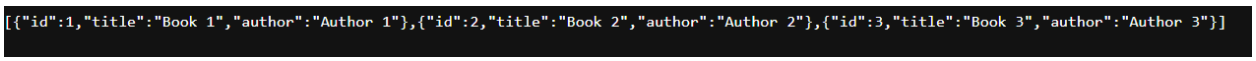
And in docker:

9. Next, in order to test that the application works, navigate to http://localhost:8080/api/books, and it should return json with several books. Then,

navigate to http://localhost:8080/api/books/1/ to test if it will return a json with only one book

Expected outputs:





References

[1] J. Ellingwood, “Understanding the nginx configuration file structure and configuration contexts,” DigitalOcean, https://www.digitalocean.com/community/tutorials/understanding-the-nginx-configuration-file-structure-and-configuration-contexts (accessed Apr. 5, 2024).

[1] A. B, “What is Docker compose? how to use it with an example,” freeCodeCamp.org, https://www.freecodecamp.org/news/what-is-docker-compose-how-to-use-it/#:~:text=Docker%20Compose%20is%20a%20tool,with%20NodeJS%20and%20MongoDB%20together. (accessed Apr. 5, 2024).