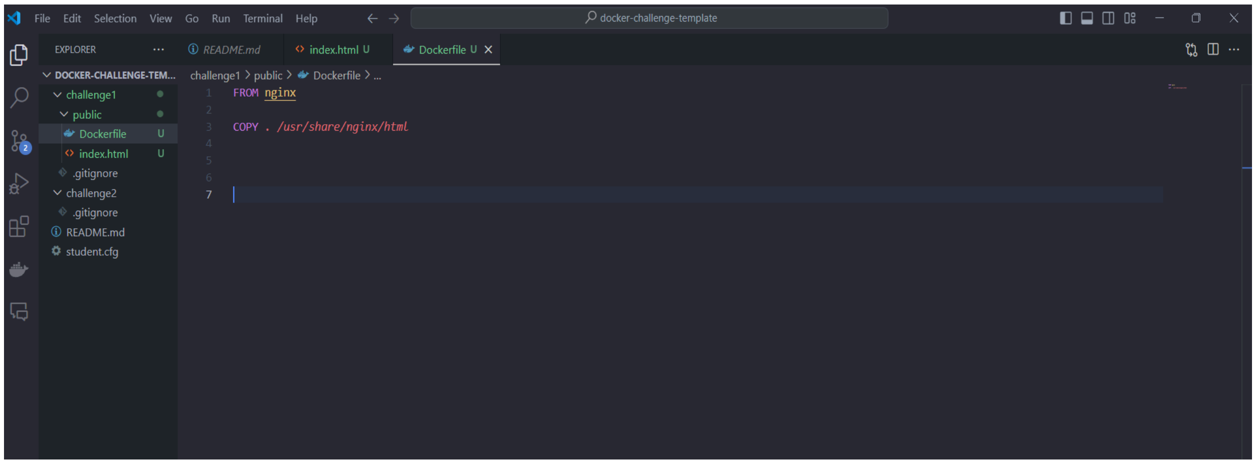
CHALLENGE 1

Docker is like the process of packing your software into marked containers called containers. These containers consist of everything required to run an application, which allows the easy movement, sharing, and executing software reliably across various computers and environments. This helps to implement tasks quicker, allows for an easy and unique development environment, and enables the straightforward scaling of applications.

Containers are just the software packages that are ready for use, and have all the necessary parameters, like code, runtime, and libraries, inside them in a single package. Docker enables to package and distribute apps in a way that those apps run in an identical environment which is isolated from others and therefore they have the same behavior, no matter which the underlying infrastructure is.

Code :

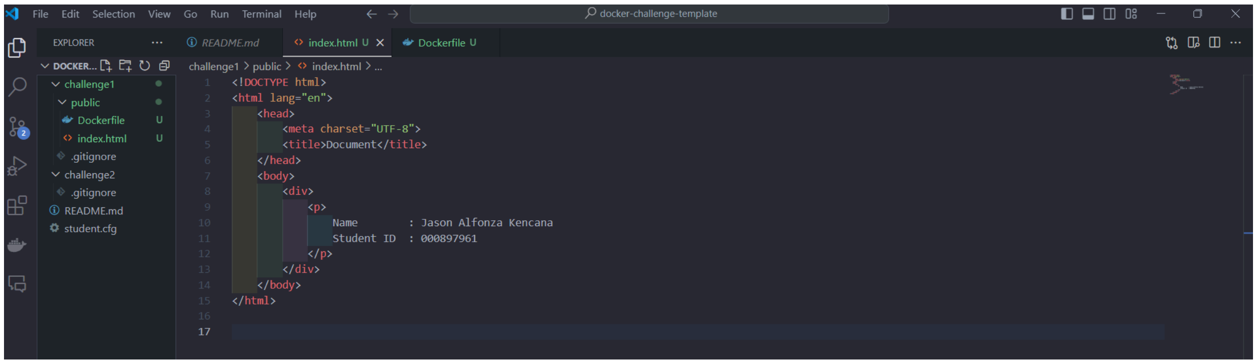
Dockerfile



```
FROM nginx

COPY . /usr/share/nginx/html
```

- FROM nginx: This line specifies the base image to use for the Docker container. In this case, it's using the official Nginx image from Docker Hub as the base image.
- COPY . /usr/share/nginx/html: This line copies the contents of the current directory (where the Dockerfile is located) into the /usr/share/nginx/html directory within the container. This likely includes the index.html file.

Index.html

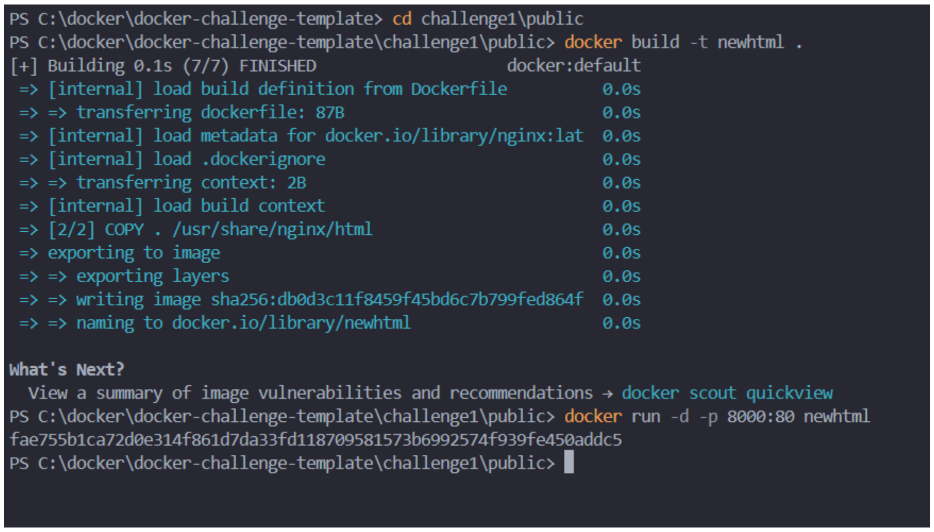


```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Document</title>
    </head>
    <body>
        <div>
            <p>
                Name        : Jason Alfonza Kencana
                Student ID  : 000897961
            </p>
        </div>
    </body>
</html>
```

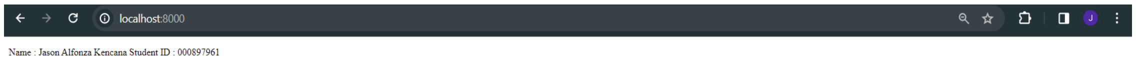
- This is just a simple html file that contains my name and my student ID

How to run the code

1. Open a terminal
2. Navigate to the directory containing the Dockerfile
    a. In the assignment I provided these are the complete directory : cd C:\docker\docker-challenge-template\challenge1\public
3. Create a docker image
    a. Docker build -t newhtml . (newhtml is the tag of the docker image using the dockerfile in the current directory)
4. Once the image is built, run the docker container based on that image
    a. Docker run -d -p 8000:80 newhtml
    b. This command will start the docker container in the detached mode (-d) and port 8000 on my host machine to port 80 in the docker container
5. Now we can test it out by opening a web browser and navigating to http://localhost:8000, we can see my name and studentID there.



```
PS C:\docker\docker-challenge-template> cd challenge1\public
PS C:\docker\docker-challenge-template\challenge1\public> docker build -t newhtml .
[+] Building 0.1s (7/7) FINISHED                    docker:default
 => [internal] load build definition from Dockerfile          0.0s
 => => transferring dockerfile: 87B                           0.0s
 => [internal] load metadata for docker.io/library/nginx:lat  0.0s
 => [internal] load .dockerignore                             0.0s
 => => transferring context: 2B                               0.0s
 => [internal] load build context                             0.0s
 => [2/2] COPY . /usr/share/nginx/html                        0.0s
 => exporting to image                                        0.0s
 => => exporting layers                                       0.0s
 => => writing image sha256:db0d3c11f8459f45bd6c7b799fed864f  0.0s
 => => naming to docker.io/library/newhtml                    0.0s

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\docker\docker-challenge-template\challenge1\public> docker run -d -p 8000:80 newhtml
fae755b1ca72d0e314f861d7da33fd118709581573b6992574f939fe450addc5
PS C:\docker\docker-challenge-template\challenge1\public> 
```

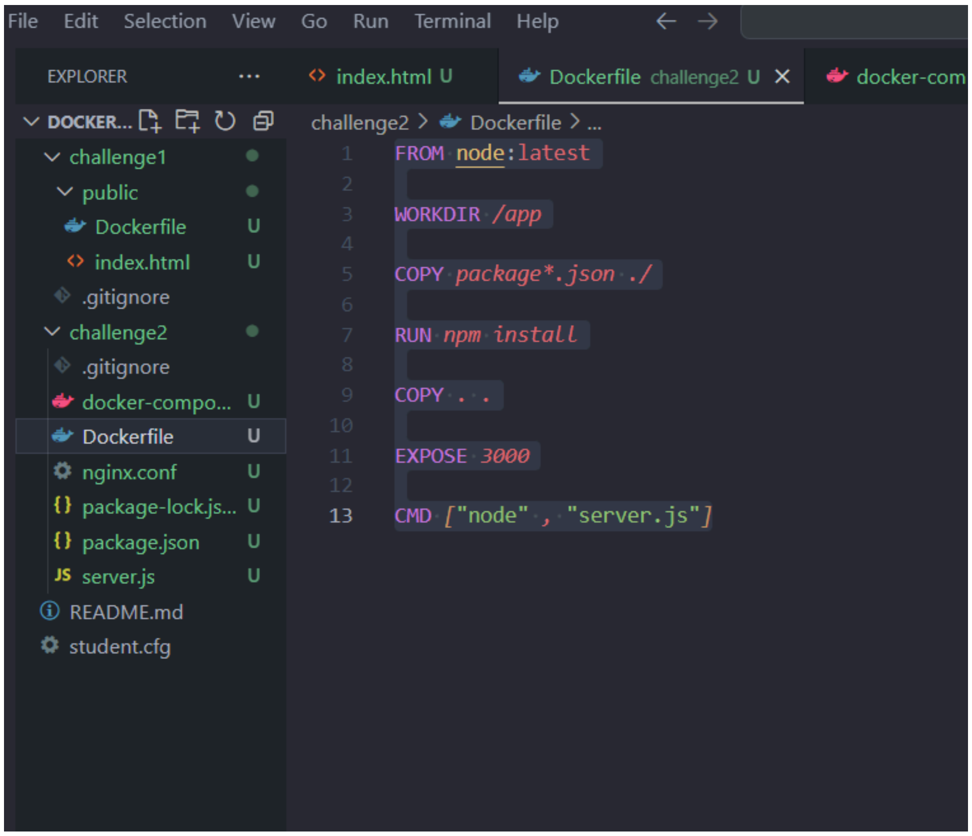


Name : Jason Alfonza Kencana Student ID : 000897961

CHALLENGE 2

Code:

Dockerfile

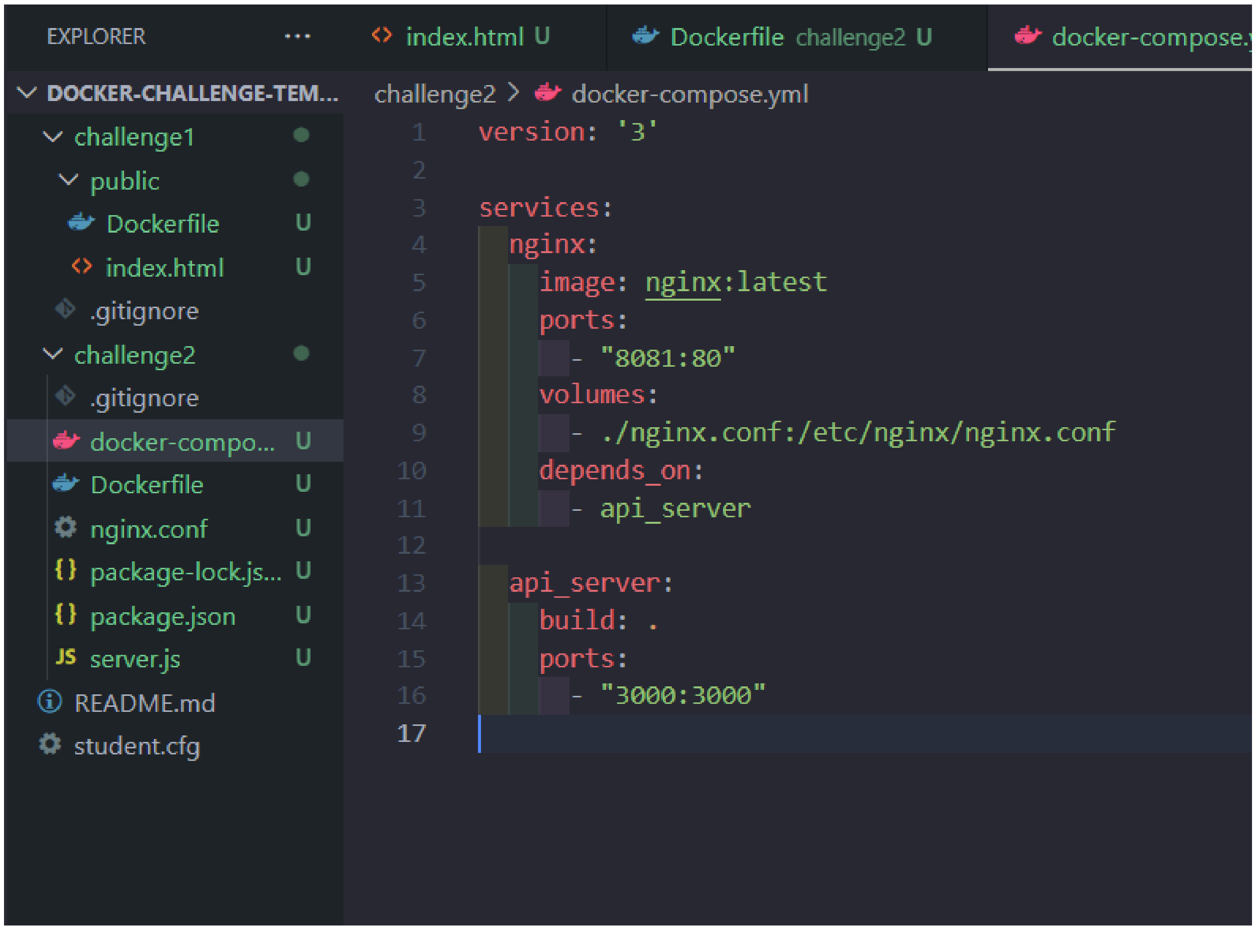


```dockerfile
FROM node:latest

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node" , "server.js"]
```

- FROM node:latest: Specifies the base image for the container, in this case, the latest version of Node.js.
- WORKDIR /app: Sets the working directory inside the container to /app.

- *COPY package.json ./**: Copies package.json and package-lock.json from the current directory on the host to the /app directory in the container.
- RUN npm install: Runs npm install inside the container to install dependencies listed in package.json.
- COPY . .: Copies the rest of the application files from the current directory on the host to the /app directory in the container.
- EXPOSE 3000: Exposes port 3000 to allow connections to the server running inside the container.
- CMD ["node" , "server.js"]: Specifies the command to run when the container starts. In this assignment, it runs the server.js file using Node.js.

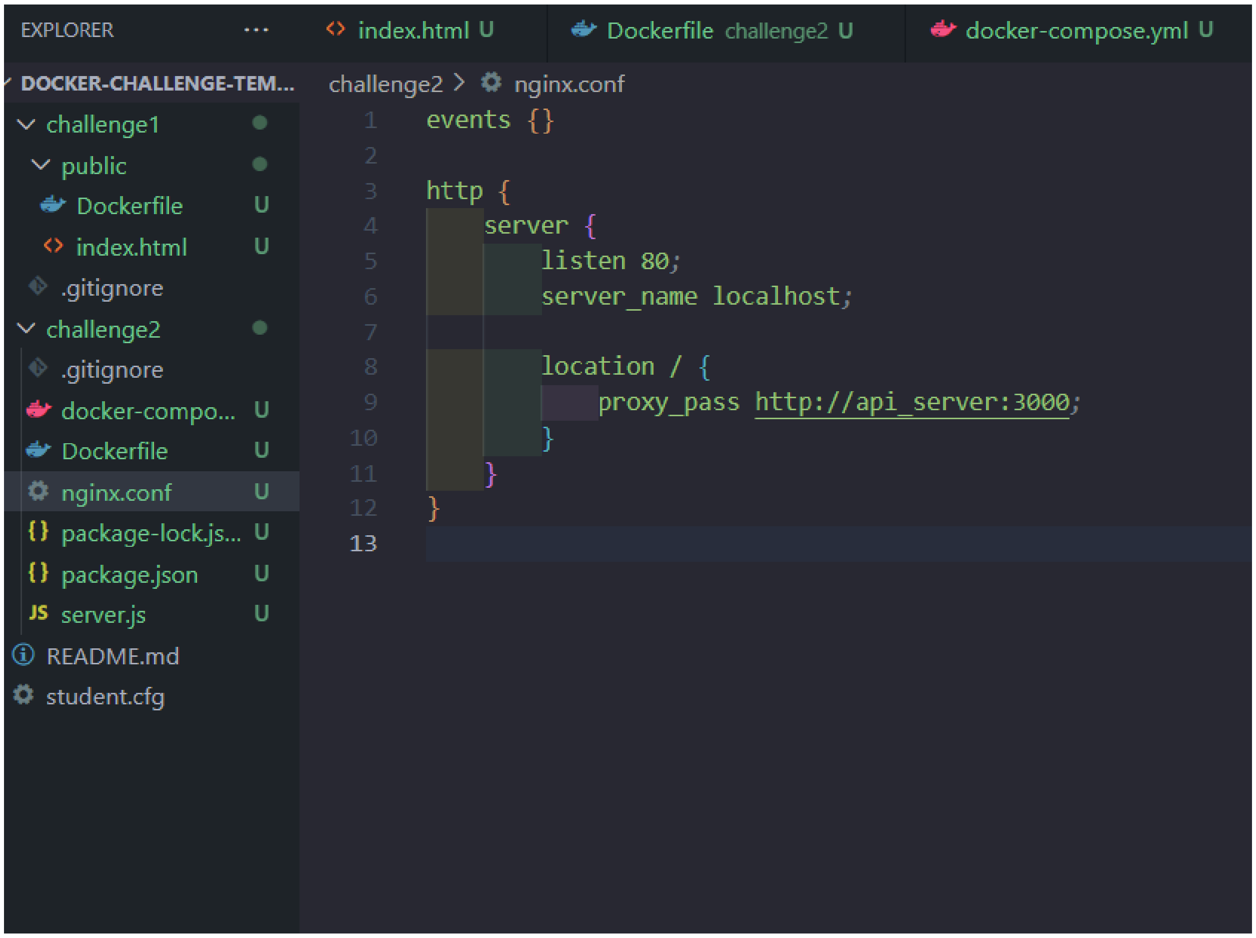
Docker-compose.yml



```yaml
version: '3'

services:
  nginx:
    image: nginx:latest
    ports:
      - "8081:80"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - api_server

  api_server:
    build: .
    ports:
      - "3000:3000"
```

- nginx service: Configures a container running Nginx, exposing port 8081 on the host and mounting a custom nginx.conf file.
- api_server service: Configures a container for the API server, building it from the Dockerfile in the current directory and exposing port 3000 on the host.

Nginx.conf



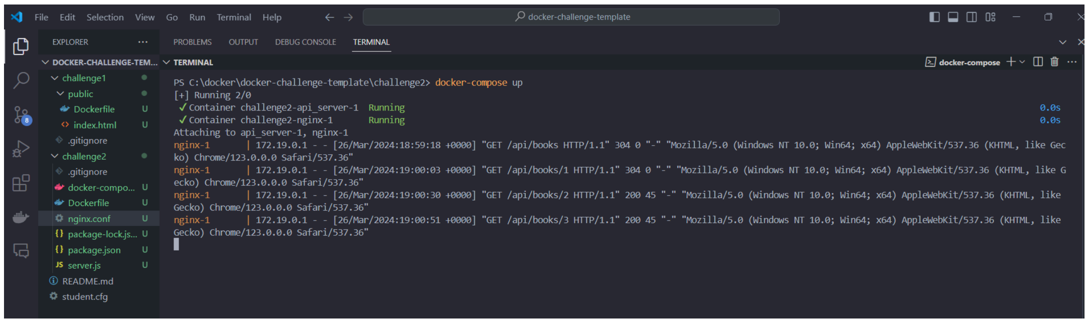
```
events {}

http {
    server {
        listen 80;
        server_name localhost;

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

\

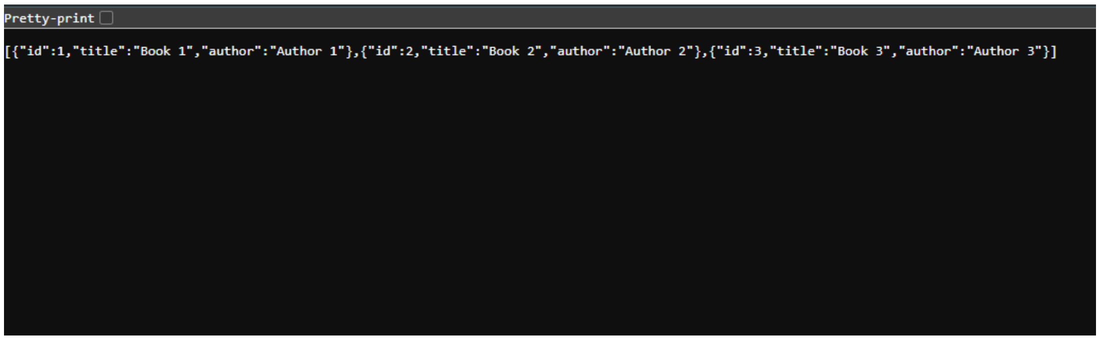
Configures Nginx to listen on port 80 and proxy requests to the API server running on port 3000.

How to run the code :

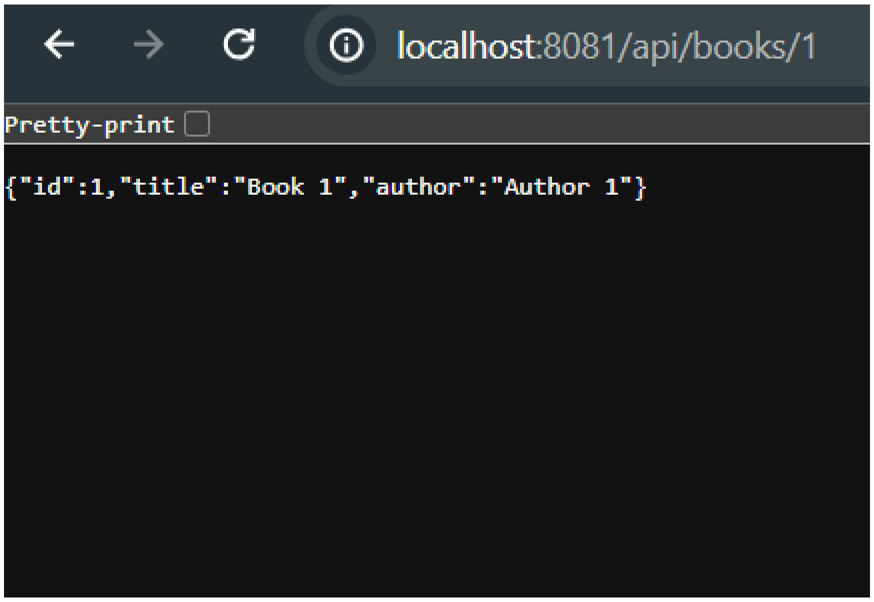
1. Open a terminal
2. Navigate to the directory containing the Dockerfile
   a. In the assignment I provided these are the complete directory :
      cd C:\docker\docker-challenge-template\challenge2
3. Run "docker-compose up" on the terminal as it will build the docker images and start the containers
4. Now we can test it out by opening a web browser and navigating to http://localhost:8081/api/books, we can see the id, title, and author for all the books.
5. If we navigate to http://localhost:8081/api/books/1 or http://localhost:8081/api/books/2, http://localhost:8081/api/books/3 we can see each id, title, and the author for the specific book.



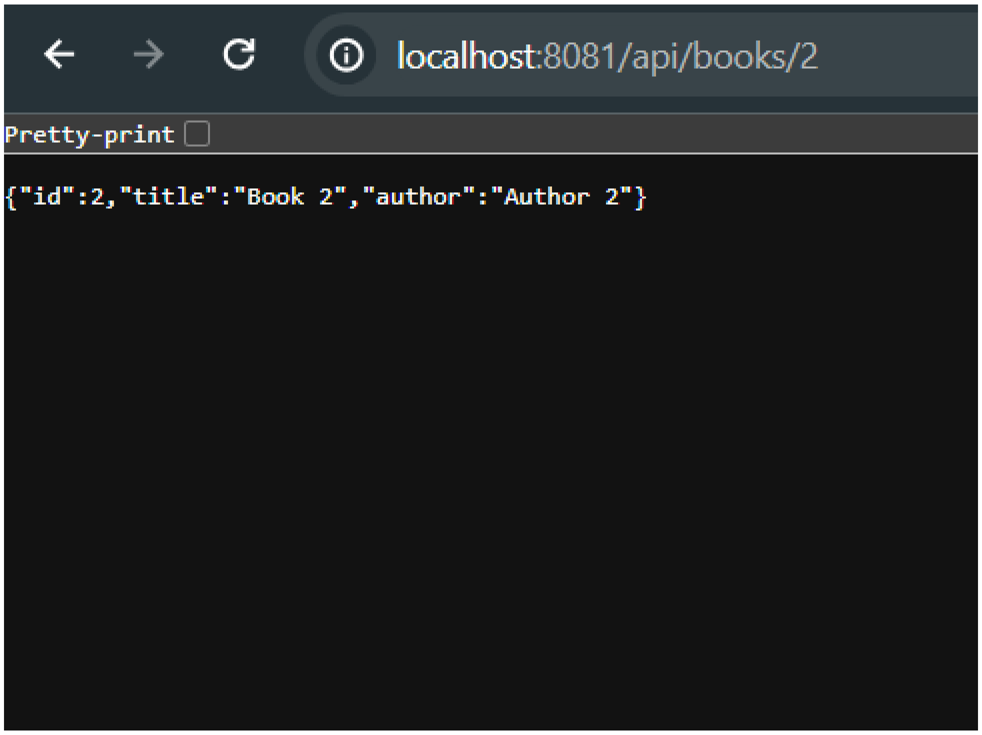
```
PS C:\docker\docker-challenge-template\challenge2> docker-compose up
[+] Running 2/0
 ✓Container challenge2-api_server-1  Running                                                                            0.0s
 ✓Container challenge2-nginx-1       Running                                                                            0.0s
Attaching to api_server-1, nginx-1
nginx-1       | 172.19.0.1 - - [26/Mar/2024:18:59:18 +0000] "GET /api/books HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
nginx-1       | 172.19.0.1 - - [26/Mar/2024:19:00:03 +0000] "GET /api/books/1 HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
nginx-1       | 172.19.0.1 - - [26/Mar/2024:19:00:30 +0000] "GET /api/books/2 HTTP/1.1" 200 45 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
nginx-1       | 172.19.0.1 - - [26/Mar/2024:19:00:51 +0000] "GET /api/books/3 HTTP/1.1" 200 45 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36"
```



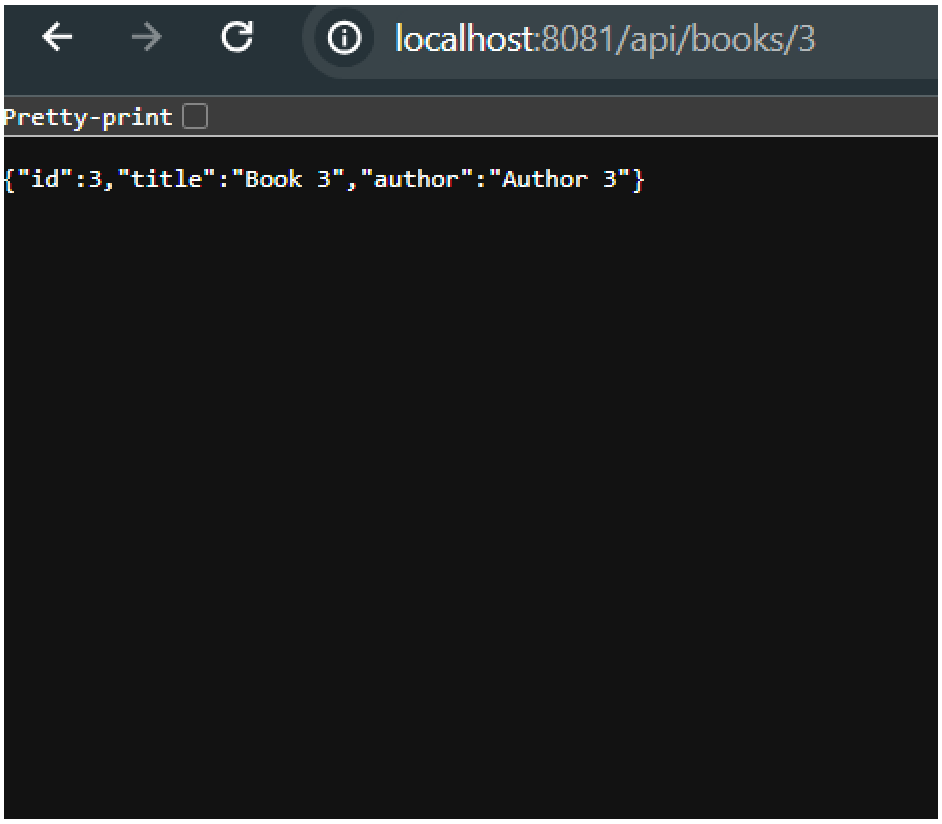
```
[{"id":1,"title":"Book 1","author":"Author 1"},{"id":2,"title":"Book 2","author":"Author 2"},{"id":3,"title":"Book 3","author":"Author 3"}]
```



```
{"id":1,"title":"Book 1","author":"Author 1"}
```



```
{"id":2,"title":"Book 2","author":"Author 2"}
```



```
{"id":3,"title":"Book 3","author":"Author 3"}
```

References

[1] “Containerization explained,” YouTube, https://www.youtube.com/watch?v=0qotVMX-J5s (accessed Mar. 26, 2024).
[2] “How to install Docker on Windows 11,” YouTube, https://www.youtube.com/watch?v=WDEdRmTCSs8 (accessed Mar. 26, 2024).
[3] What is containerization? - containerization explained - AWS, https://aws.amazon.com/what-is/containerization/ (accessed Mar. 26, 2024).
[4] “Use Docker compose,” Docker Documentation, https://docs.docker.com/get-started/08_using_compose/ (accessed Mar. 26, 2024).
[5] 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/ (accessed Mar. 26, 2024).