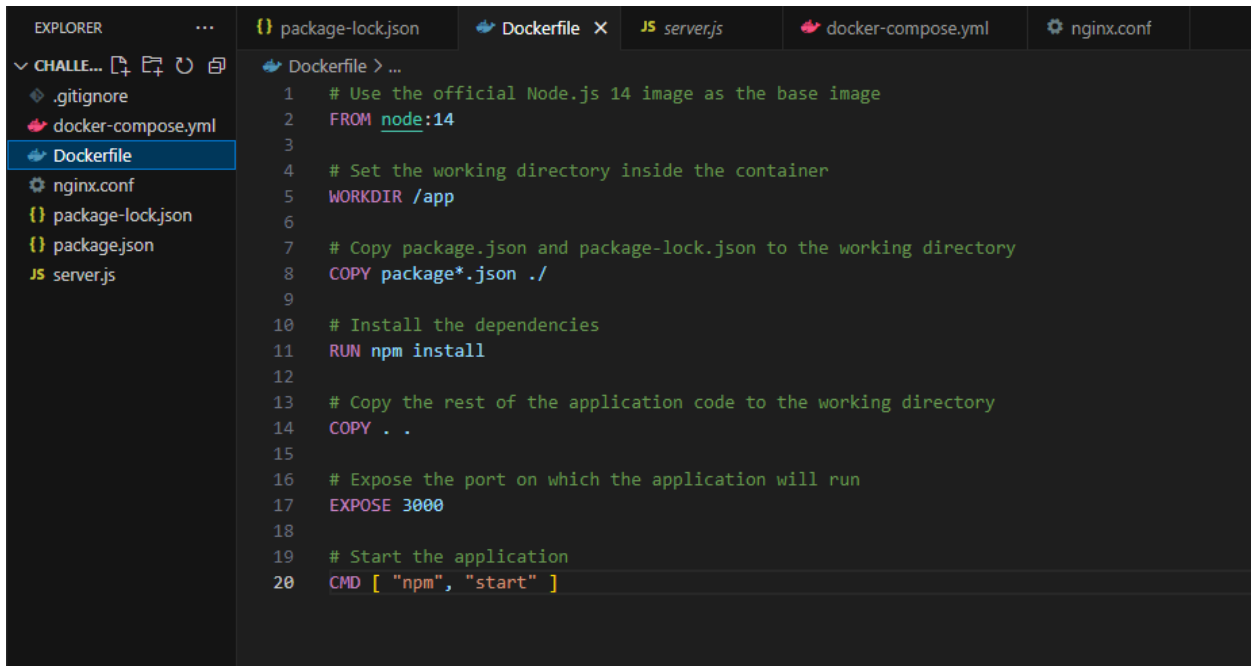


Docker Challenge 2

Danny Horning

000702974

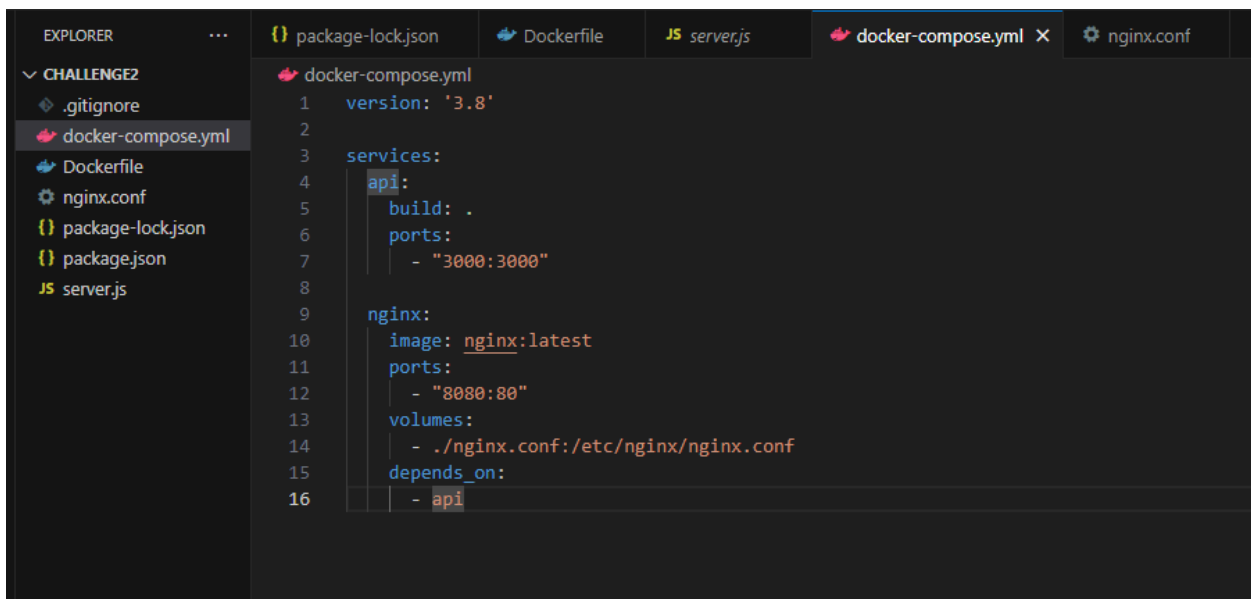
June 26th 2024



The screenshot shows the VS Code interface with the Explorer on the left and the Dockerfile editor in the center. The Explorer shows a project named 'CHALLENGE2' with files: .gitignore, docker-compose.yml, Dockerfile, nginx.conf, package-lock.json, package.json, and server.js. The Dockerfile editor shows the following content:

```
Dockerfile > ...
1  # Use the official Node.js 14 image as the base image
2  FROM node:14
3
4  # Set the working directory inside the container
5  WORKDIR /app
6
7  # Copy package.json and package-lock.json to the working directory
8  COPY package*.json ./
9
10 # Install the dependencies
11 RUN npm install
12
13 # Copy the rest of the application code to the working directory
14 COPY . .
15
16 # Expose the port on which the application will run
17 EXPOSE 3000
18
19 # Start the application
20 CMD [ "npm", "start" ]
```

While setting up this project, I first unzipped the required json files and server into the appropriate. The first step I took was to create the dockerfile, in which we use the node version 14 image to package everything. We then insert the json files into the working directory. Next, to get everything working properly, we need to install node package manager. Finally we tell the application to listen on port 3000 (where the server will be) and to start.



The screenshot shows the VS Code interface with the Explorer on the left and the docker-compose.yml editor in the center. The Explorer shows the same project 'CHALLENGE2' with files: .gitignore, docker-compose.yml, Dockerfile, nginx.conf, package-lock.json, package.json, and server.js. The docker-compose.yml editor shows the following content:

```
docker-compose.yml
1  version: '3.8'
2
3  services:
4    api:
5      build: .
6      ports:
7        - "3000:3000"
8
9    nginx:
10     image: nginx:latest
11     ports:
12       - "8080:80"
13     volumes:
14       - ./nginx.conf:/etc/nginx/nginx.conf
15     depends_on:
16       - api
```

This file is going to manage our separate containers used in our application. The api represents the Node.js application. The ports connect the host machine ports to the container ports. The

The screenshot shows the VS Code interface. The Explorer sidebar on the left lists files: .gitignore, docker-compose.yml, Dockerfile, nginx.conf (selected), package-lock.json, package.json, and server.js. The Editor pane on the right displays the content of nginx.conf:

```
1  events {}
2
3  http {
4      server {
5          listen 80;
6
7          location /api {
8              proxy_pass http://api:3000;
9          }
10     }
11 }
```

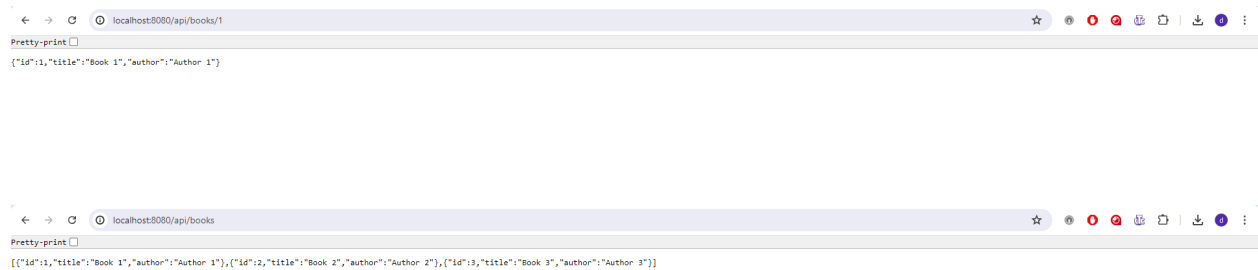
[illegible]

```

Network challenge2_default  [vulnerable]
Container challenge2-apl-1  [vulnerable]
Container challenge2-nginx-1 [vulnerable]
Attaching to apl-1, nginx-1

apl-1 > sr@18.0.0 start /asp
nginx-1 > mode server-js
nginx-1
nginx-1 > Server running on port 3000
nginx-1 /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx-1 /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx-1 /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx-1 10-listen-on-ipv6-by-default.sh info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx-1 10-listen-on-ipv6-by-default.sh info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
nginx-1 /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx-1 /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx-1 /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx-1 /docker-entrypoint.sh: Configuration complete; ready for start up
nginx-1 172.18.0.1 -- [26/Jun/2024:19:08:58 +0000] "GET /api/pooks HTTP/1.1" 200 130 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0 Safari/537.36"
nginx-1 172.18.0.1 -- [26/Jun/2024:19:07:24 +0000] "GET /api/pooks HTTP/1.1" 200 45 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0 Safari/537.36"
nginx-1 172.18.0.1 -- [26/Jun/2024:19:07:20 +0000] "GET /api/pooks HTTP/1.1" 200 45 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0 Safari/537.36"
nginx-1 172.18.0.1 -- [26/Jun/2024:19:07:19 +0000] "GET /api/pooks HTTP/1.1" 200 45 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0 Safari/537.36"

```



Once the container had finished building, I opened a browser (in this case google chrome) and tested both the localhost:8080/api/books and the localhost:8080/api/books/1 to ensure that I was receiving the correct json call.

References:

<https://docs.docker.com/compose/>

<https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose>

<https://www.baeldung.com/ops/docker-compose>

https://www.youtube.com/watch?v=JiJeZOHx0ow&ab_channel=AmichaiMantinband