# Operating Systems

# Midterm Project

CPSY-300-C

**Docker**

**Challenges 1 & 2**

## Name:

Rommel Hipos

**Date Submitted:**

June 29, 2024

## Docker Introduction

Docker is an open-source platform that allows you to automate the deployment, scaling, and management of applications using containerization. Containers are lightweight, portable units that encapsulate an application and all its dependencies, ensuring that the application runs consistently across different environments [1].

## Importance of Docker in Software Development

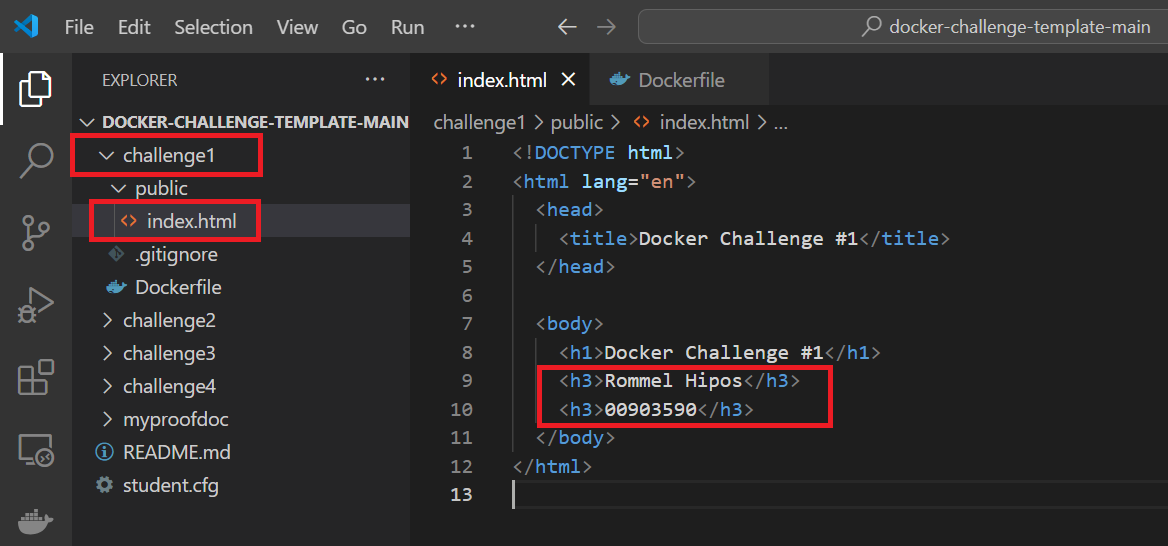
Using Docker Desktop facilitates teamwork by enabling one-click sharing of work via Git or Docker Hub. Additionally, it features an intuitive user interface (UI) for numerous common tasks, such as initiating, pausing, and ending containers. Through our Docker Community Slack channel, users of Docker may also communicate, learn from, and work together with one another. By leveraging Docker's powerful containerization technology, developers can streamline their workflows, enhance collaboration, and ensure that applications run reliably in any environment.

Additionally, Docker users can learn, connect, and collaborate with each other via our Docker Community Slack channel. We can chat with Docker community leaders, Docker Captains, and your fellow local developers in the channel. [2].

## Challenge 1 - Simple static page server

### Steps

* Use the folder challenge1.
* Add a "public" folder with some assets.
* Add a file with the name index.html. It should contain your **name** and **SAIT ID** in the contents.



***Lessons Learned****: The purpose of these steps is to set up a basic Docker image that includes a simple web application structure. This structure includes static assets in a public folder and a minimal index.html file that can be served by a web server running inside the Docker container.*

* Create a Dockerfile to use NGinx to serve pages existent in the public folder.
* Create the Docker image.
* Execute docker with the right parameters.
* *Command to build the image*: ***docker build -t <name of the image> .***

A screenshot of a computer program

Description automatically generated

* *Once the build is successfully executed, open the Docker desktop to view the image created.*

A screenshot of a computer

Description automatically generated

* Run the image to generate the container and set the port to 8080

A screenshot of a computer

Description automatically generated

* Now it’s ready to run inside the container.

***Lessons Learned****: With the help of these methods, I can quickly create a Docker image that encapsulates the whole web server setup into a portable and managed containerized environment, using NGinx to serve static web content from a public folder.*

* Commit the Dockerfile and public folder and push it to the remote repository.

A screenshot of a computer

Description automatically generated

### Expected outcomes

* When you request the URL “http://localhost:8080/” you will get a home page with your name and code.

A screenshot of a computer

Description automatically generated

***Lessons Learned****: Once the Docker image has been successfully created, you can run it using Docker Desktop to instantiate the container, making it ready to access.*

## Challenge 2 - NodeJS application

### Steps

* Use the folder challenge2.
* Extract the files present on challenge2.zip to the challenge’s root folder.
* Create a Dockerfile to build the server’s Docker container.

A screenshot of a computer

Description automatically generated

* Create the Docker compose file using NGinx and the API server from the previous step.
  + NGinx should listen on port 8080.

A screenshot of a computer screen

Description automatically generated

***Lessons Learned****: This ensures that NGinx is configured to handle requests on port 8080, acting as a gateway or reverse proxy to your API server.*

* *Command to build the image*: ***docker build -t <name of the image> .***

A screenshot of a computer program

Description automatically generated

* *Once the build is successfully executed, open the Docker desktop to view the image created*

***A screenshot of a computer

Description automatically generated***

* Run the image to generate the container and set the port to 8080

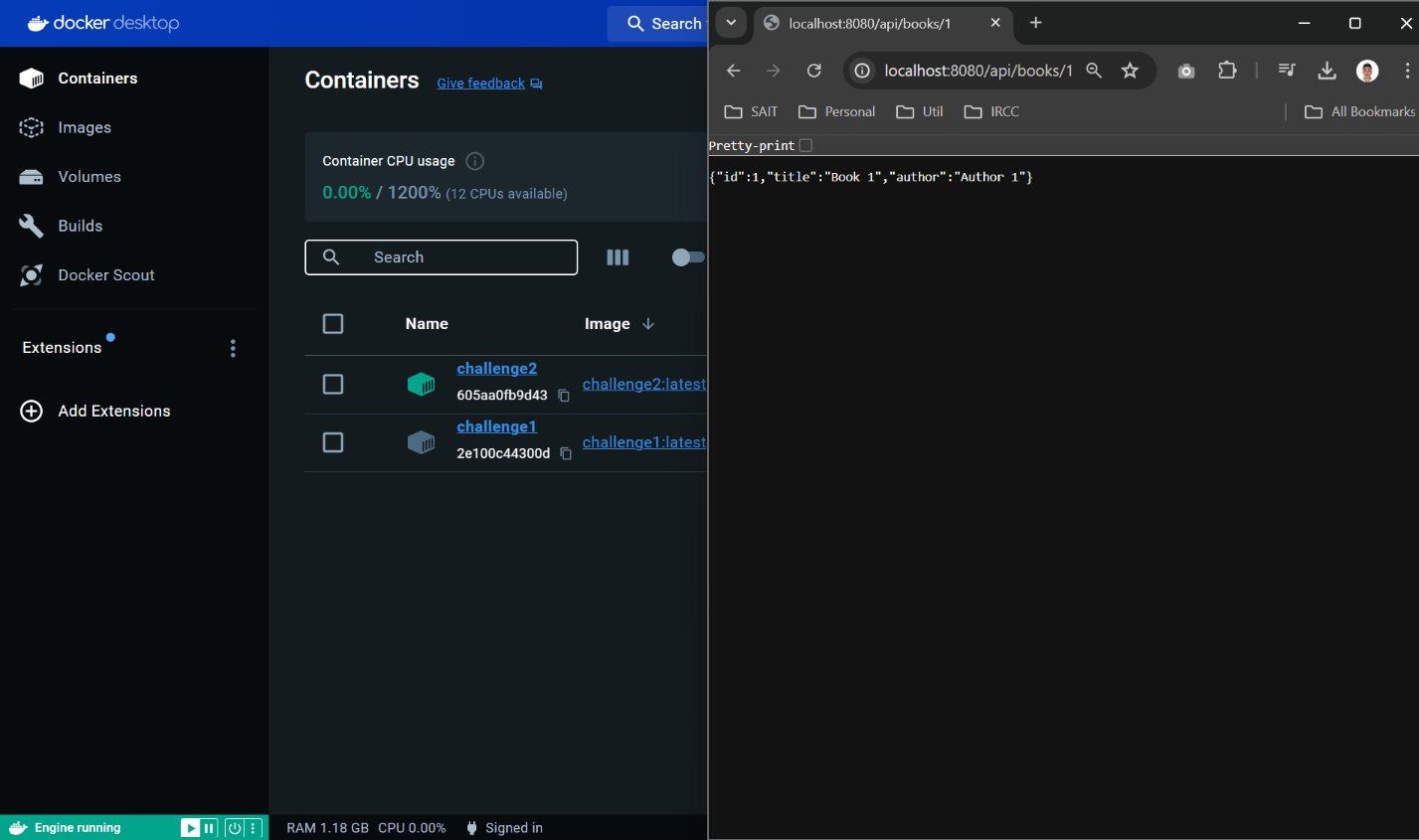
***A screenshot of a computer

Description automatically generated***

* Now it’s ready to run inside the container.

***Lessons Learned****: The goal of all these steps is to set up and get ready a development or deployment environment inside the challenge2 directory. These consist of establishing the directory structure, incorporating required project resources from a compressed file, and setting up a Dockerfile to enable the deployment of containerized servers.*

* Open a browser and point it to the address <http://localhost:8080/api/books>.



* Commit all files and push them to the remote repository.

A screenshot of a computer

Description automatically generated

### Expected outcomes

* When you access the following URLs:
  + “http://localhost:8080/api/books” you will get a JSON message with all books.

A screenshot of a computer

Description automatically generated

* + “http://localhost:8080/api/books/1” you will get a JSON message with just one book.

A screenshot of a computer

Description automatically generated

**Test Access Using my local VM:**

1. **Windows VM**
2. **Challenge 1**

**A screenshot of a computer

Description automatically generated**

1. **Challenge 2**

**A screenshot of a computer

Description automatically generated**

1. **Linux VM**
2. **Challenge 1**

**A screenshot of a computer

Description automatically generated**

1. **Challenge 2**

**A screenshot of a computer

Description automatically generated**

**Prerequisites:**

1. Install IDE to load the challenge template, in my case I use visual code as my editor.

<https://code.visualstudio.com/>

1. Install Docker Desktop based on your current operating system.

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

1. This is optional, I used Postman to easily test the API (Challenge2). Postman is a powerful and user-friendly tool that allows developers to design, test, and document APIs.

<https://www.postman.com/downloads/>

# References

|  |  |
| --- | --- |
| [1] | [Online]. Available: https://docs.docker.com/. |
| [2] | [Online]. Available: https://www.docker.com/blog/getting-started-with-docker-desktop/#:~:text=Docker%20Desktop%20makes%20collaboration%20easy,Pausing%20and%20resuming%20a%20container. |
| [3] | [Online]. Available: https://github.com/IamStudentRommel/rommel-hipos-docker-challenge.git. |
| [4] | [Online]. Available: https://medium.com/@ajitfawade/how-to-create-a-docker-project-for-a-node-js-web-application-90-days-of-devops-e3623f46bf7. |
| [5] | [Online]. Available: https://medium.com/@ajitfawade/how-to-create-a-docker-project-for-a-node-js-web-application-90-days-of-devops-e3623f46bf7. |
| [6] | [Online]. Available: https://www.docker.com/blog/how-to-use-the-official-nginx-docker-image/. |