# Assignment: Containerization with Docker

## Introduction

This project aims to stimulate students to develop their technical skills using containerization tools like Docker, relating to some content covered during the Operating Systems and Cloud Computing course.

### Dependencies

To perform these challenges, each student must

1. Have **git** installed in their machine.
2. Have a **GitHub account**.
3. Have **Docker installed** on their computer (or in their Linux VM if you consider it more convenient).

### Getting started

The first step is to clone the repository and **keep it private**:

<https://github.com/eduluz1976/docker-challenge-template>

The repository folders should follow this structure:

- README.md

- student.cfg

+ challenge1

+ challenge2

The file **student.cfg** will contain the student information, and the README.md some information about your work.

### 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/>

### Goals

Each challenge has a set of goals. Aside from each challenge goal, this laboratory aims to stimulate all students to learn more about Docker, and how it relates to all Operating Systems and Cloud Computing content covered in the course. Moreover, these challenges will bring some real case scenarios from the industry, which will be useful for job interviews, software development activities, and open space for new learnings in related areas, such as DevOps, Architecture, and Cybersecurity.

Remember: much more than grades, these challenges will bring you knowledge and experience valid for your professional life. This is a great opportunity to boost your knowledge in a very required subject nowadays.

### 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

This is a simple introduction to how to use Docker to serve some static pages.

### Goals

* Learn the basics of Docker.
* Build a Docker image.
* Be able to run the container and see the results on a browser.
* Publish the Dockerfile and related files on GitHub.

### 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.
* Create a Dockerfile to use NGinx to serve pages existent in the public folder.
* Create the Docker image.
* Execute docker with the right parameters.
* Commit the Dockerfile and public folder and push it to the remote repository.
* Run your application and take a screenshot of it.
* **Submit on D2L**:
  + Your report.
  + The repository's URL. Don’t forget to add your instructors as a collaborator.
  + **When asked**:
    - Answer the questions.
    - Add the screenshots.

### Expected outcomes

* When you request the URL “http://localhost:8080/” you will get a home page with your name and code.
* On D2L you will submit the document with all steps necessary to achieve the results.

## Challenge 2 - NodeJS application

Given a NodeJS application, create the Dockerfile that allows exposing the endpoints for external clients and the **Docker Compose** file to orchestrate the web server and the application.

### Goals

* Introduce the concept of Docker compose.
* Given a simple NodeJS application, create the Dockerfile.
* Watch a dynamic application running on Docker.
* Build a foundation for the next step.

### 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.
* Create the Docker compose file using NGinx and the API server from the previous step.
  + NGinx should listen on port 8080.
* Check if all services are running properly.
* Open a browser and point it to the address <http://localhost:8080/api/books>.
* If the result is not expected, then return and fix it.
* Commit all files and push them to the remote repository.
* Submit the repository's URL in D2L and answer the questions.

### Expected outcomes

* When you access the following URLs:
  + “http://localhost:8080/api/books” you will get a JSON message with all books.
  + “http://localhost:8080/api/books/1” you will get a JSON message with just one book.
* On D2L you will submit the document with all steps necessary to achieve the results.