# # Introduction

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

## ## For your reference resources

* <https://www.docker.com/products/docker-desktop/>
* <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/index.htm>
* <https://www.markdownguide.org/>

## ## 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 so far. Adding to that, these challenges will bring some use-case scenarios from the industry, which will be useful for building your portfolio, 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.

At the end of each challenge, we expect that each student will deliver:

1. To prepare a document with the steps followed to implement these challenges **(1 and 2)** as detailed as possible.
2. Document should have step by step explanation.

## ## Some basic rules

Here are some basic rules for these challenges:

1. This is **individual work**.
2. **It is not necessary to mention that copies (or shared work) and AI written text will not be tolerated.**

## #### Deliverables Submission

The student should submit a write-up with enough details so that a Docker beginner can be able to reproduce the results. You can write it in the form of a tutorial in the following format:

* A single file, in PDF format.
* Add links for the pages you used as reference.
* Add screenshots of your terminal, and your browser, showing the commands you used, and the results.

# # Challenges

## ## Challenge 1 - Simple web server for static web pages

This is an introduction of 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 browser.
* Publish the Dockerfile and related files on GitHub.

### ### Steps

* Add a file with name index.html. It should contain your **name** and **student ID number**.
  + ID=<write here your student ID>
  + NAME=<write your name>
* Create a Dockerfile to use NGinx to serve pages existent on the public folder.
* Create a Docker image.
* Execute docker with the right parameters.
* Run your application and take a screenshot of the server running and the page being served.
* Submit on D2L:
* Zip the docker file and other artifacts and upload to D2L as proof of submission.
  + **When asked**:
    - Answer the questions.
    - Add the screenshots.

### ### Expected outcomes

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

## ## Challenge 2 – Creating a dynamic application

Create a simple application (NodeJs/Java/Python or anything else), with a Dockerfile that allows to expose the endpoints to external clients.

### ### Goals

* Introduce the concept of Docker Compose.
* Create a simple application, with a Dockerfile.
* Watch a dynamic application running on Docker.

### ### Steps:

* Create a Dockerfile to build the server’s Docker container
* Using NGinx and the API server from previous step.
  + NGinx should listen on post 8080
* Verify 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.
* Zip all the files and upload to D2L

### ### Expected outcomes

* When you make a request to URL:
  + “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 the steps you went through to achieve your results.