TD2 - Simple Web Server with Docker

Goal

The goal of this lab is to create and run your first Docker container!

Subject

The following subject uses the words *must*, *must not*, *should*, and *should not*. Refer to RFC 2119.

The submission for this TD will be done via GitLab.

Step 0 - GitLab

Verify that you can sign in at https://gitlab.sre.paris/users/sign_in.

If you encounter any issues signing in, reach out to your teacher for assistance as soon as possible.

Step 1 - Required architecture

At the end of the lab, make sure your submission match the following tree.

```
. # Your Gitlab repository must look like this
├── build.sh
├── Dockerfile
├── index.html
└── run.sh
```

Step 2 - Create your first Dockerfile

In this step, you will learn how to create a Dockerfile to containerize a simple Nginx web server and serve a basic HTML file.

A Dockerfile is simply a text document containing instructions that Docker uses to build a container image. It allows you to define a reproducible and portable environment for your application. By specifying base images, dependencies, configurations, and commands, a Dockerfile ensures your application can run consistently across different systems. You can see it as a sort of cooking recipe to create a container. It is not a container per se, but the instructions on how to create one.

Next, let's talk about Nginx, the software you'll use in this step. Nginx (pronounced "engine-x") is a high-performance web server and reverse proxy server. It is widely used to serve static files (testine.sre.paris is exposed by an Nginx server), and can handle high levels of traffic with low resource usage.

Create the following files:

- `index.html`: Create a basic HTML file. Ensure it includes a title and some content to display.
- `Dockerfile`: Write a Dockerfile to set up an Nginx web server that serves the `index.html` file.

Step 3 - Build your first Dockerfile

Let’s briefly recap what it means to "build" a Docker image: When you build a Docker image, Docker reads the instructions in your Dockerfile to create an image layer by layer. Each instruction creates a new layer, allowing Docker to optimize the build process by caching unchanged layers. The resulting image can then be used to create and run containers on any system that supports Docker.

To facilitate the image-building process, you will write a shell script. Shell scripts are a convenient way to automate repetitive tasks, such as building a Docker image with specific options.

Create the following files:

- `build.sh`: Create a shell script that build the Docker image.

Constraint:

- The Docker image **must** be named `my-tiny-web-server`.

Step 4 - Run your first Dockerfile

In this step, you will use the Docker image you built earlier to run a containerized instance of your web server.

Let’s quickly revisit what it means to "run" a Docker container: Running a Docker container involves creating an isolated process on a computer that can run an application regardless of where it is deployed.

To simplify the process, you will write a script that runs the container with the necessary options to respect the constraints.

Here are the constraints you **must** respect:

- The `run.sh` script create a single container named webserver.
- The web server **must** be accessible from the host system on port 8080.
- The container **must** be automatically removed when stopped to avoid clutter.
- The container **must** run in detached mode, allowing it to operate in the background.

Create the following files:

- `run.sh`: Write a script to run the Docker container from the image built earlier by build.sh.

By the end of this step, your web server will be running and accessible in your browser at http://localhost:8080.

Git Submission

Submission for this TD will be done via GitLab.

A repository has already been created for you (`workshop2-<firstname>-<lastname>`). Clone the repository and push your work to the `main` branch.

Testine

Testine is the platform you will use to assess your work. It provides an automated grading system for your submissions, ensuring consistency and transparency.

Refer to the Testine guide for more details on how to use the platform effectively.