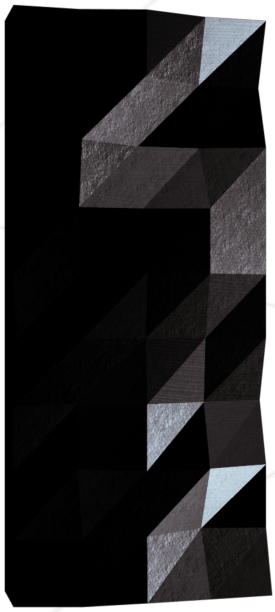


Inception

Summary: This document is a System Administration related exercise.

Version: 5.1

Contents

Chapter I

Preamble

Chapter II

Introduction

This project aims to broaden your knowledge of system administration by using Docker. You will virtualize several Docker images, creating them in your new personal virtual machine.

Chapter III

General guidelines

- This project needs to be done on a Virtual Machine.
- All the files required for the configuration of your project must be placed in a `srcs` folder.
- A `Makefile` is also required and must be located at the root of your directory. It must set up your entire application (i.e., it has to build the Docker images using `docker-compose.yml`).
- This subject requires putting into practice concepts that, depending on your background, you may not have learned yet. Therefore, we advise you not to hesitate to read a lot of documentation related to Docker usage, as well as anything else you will find helpful in order to complete this assignment.

Chapter IV

AI Instructions

Context

During your learning journey, AI can assist with many different tasks. Take the time to explore the various capabilities of AI tools and how they can support your work. However, always approach them with caution and critically assess the results. Whether it's code, documentation, ideas, or technical explanations, you can never be completely sure that your question was well-formed or that the generated content is accurate. Your peers are a valuable resource to help you avoid mistakes and blind spots.

Main message

- Use AI to reduce repetitive or tedious tasks.
- Develop prompting skills — both coding and non-coding — that will benefit your future career.
- Learn how AI systems work to better anticipate and avoid common risks, biases, and ethical issues.
- Continue building both technical and power skills by working with your peers.
- Only use AI-generated content that you fully understand and can take responsibility for.

Learner rules:

- You should take the time to explore AI tools and understand how they work, so you can use them ethically and reduce potential biases.
- You should reflect on your problem before prompting — this helps you write clearer, more detailed, and more relevant prompts using accurate vocabulary.
- You should develop the habit of systematically checking, reviewing, questioning, and testing anything generated by AI.
- You should always seek peer review — don't rely solely on your own validation.

Phase outcomes:

- Develop both general-purpose and domain-specific prompting skills.
- Boost your productivity with effective use of AI tools.
- Continue strengthening computational thinking, problem-solving, adaptability, and collaboration.

Comments and examples:

- You'll regularly encounter situations — exams, evaluations, and more — where you must demonstrate real understanding. Be prepared, keep building both your technical and interpersonal skills.
- Explaining your reasoning and debating with peers often reveals gaps in your understanding. Make peer learning a priority.
- AI tools often lack your specific context and tend to provide generic responses. Your peers, who share your environment, can offer more relevant and accurate insights.
- Where AI tends to generate the most likely answer, your peers can provide alternative perspectives and valuable nuance. Rely on them as a quality checkpoint.

✓ Good practice:

> I ask AI: "How do I test a sorting function?" It gives me a few ideas. I try them out and review the results with a peer. We refine the approach together.

✗ Bad practice:

> I ask AI to write a whole function, copy-paste it into my project. During peer-evaluation, I can't explain what it does or why. I lose credibility — and I fail my project.

✓ Good practice:

> I use AI to help design a parser. Then I walk through the logic with a peer. We catch two bugs and rewrite it together — better, cleaner, and fully understood.

✗ Bad practice:

> I let Copilot generate my code for a key part of my project. It compiles, but I can't explain how it handles pipes. During the evaluation, I fail to justify and I fail my project.

Chapter V

Mandatory part

This project consists of having you set up a small infrastructure composed of different services under specific rules. The whole project has to be done in a virtual machine. You have to use `docker compose`.

Each Docker image must have the same name as its corresponding service.
Each service has to run in a dedicated container.
For performance reasons, the containers must be built either from the penultimate stable version of Alpine or Debian. The choice is yours.
You also have to write your own `Dockerfiles`, one per service. The `Dockerfiles` must be called in your `docker-compose.yml` by your `Makefile`.
This means you have to build the Docker images of your project yourself. It is then forbidden to pull ready-made Docker images, as well as using services such as DockerHub (Alpine/Debian being excluded from this rule).

You then have to set up:

- A Docker container that contains NGINX with TLSv1.2 or TLSv1.3 only.
- A Docker container that contains WordPress + php-fpm (it must be installed and configured) only, without nginx.
- A Docker container that contains MariaDB only, without nginx.
- A volume that contains your WordPress database.
- A second volume that contains your WordPress website files.
- A `docker-network` that establishes the connection between your containers.

Your containers have to restart in case of a crash.

`A Docker container is not a virtual machine. Thus, it is not recommended to use any hacky patches based on 'tail -f' and similar methods when trying to run it. Read about how daemons work and whether it's a good idea to use them or not.`

Of course, using network: host or --link or links: is forbidden. The network line must be present in your docker-compose.yml file. Your containers must not be started with a command running an infinite loop. Thus, this also applies to any command used as entrypoint, or used in entrypoint scripts. The following are a few prohibited hacky patches: tail -f, bash, sleep infinity, while true.

Read about PID 1 and the best practices for writing Dockerfiles.

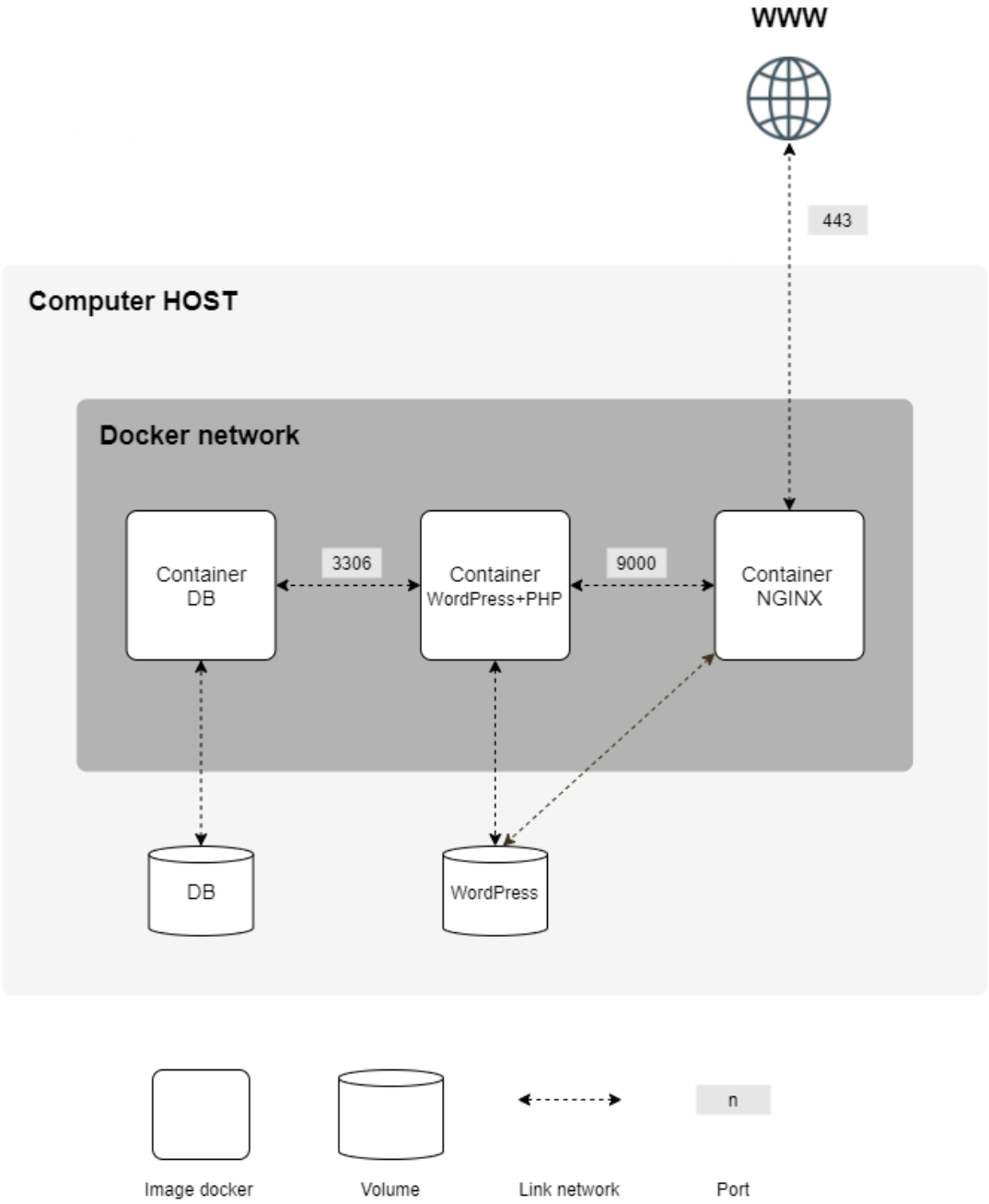
- In your WordPress database, there must be two users, one of them being the administrator. The administrator's username can't contain admin/Admin or administrator/Administrator (e.g., admin, administrator, Administrator, admin-123, and so forth).

Your volumes will be available in the /home/login/data folder of the host machine using Docker. Of course, you have to replace the login with yours.

To make things simpler, you have to configure your domain name so it points to your local IP address.
This domain name must be `login.42.fr`. Again, you have to use your own login.
For example, if your login is wil, `wil.42.fr` will redirect to the IP address pointing to wil's website.

The latest tag is prohibited.
No password must be present in your Dockerfiles.
It is mandatory to use environment variables.
Also, it is mandatory to use a .env file to store environment variables. It is strongly recommended that you use Docker secrets to store any confidential information. Any credentials, API keys, or passwords found in your Git repository (outside of properly configured secrets) will result in project failure.
Your NGINX container must be the only entrypoint into your infrastructure via the port 443 only, using the TLSv1.2 or TLSv1.3 protocol.

Here is an example diagram of the expected result:



Below is an example of the expected directory structure:

```
$> ls -alR
total XX
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 .
drwxrwxrwt 17 wil wil 4096 avril 42 20:42 ..
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 Makefile
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 secrets
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 srcs

./secrets:
total XX
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 6 wil wil 4096 avril 42 20:42 ..
-rw-r--r-- 1 wil wil XXXX avril 42 20:42 credentials.txt
-rw-r--r-- 1 wil wil XXXX avril 42 20:42 db_password.txt
-rw-r--r-- 1 wil wil XXXX avril 42 20:42 db_root_password.txt

./srcs:
total XX
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 ..
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 docker-compose.yml
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 .env
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 requirements

./srcs/requirements:
total XX
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 3 wil wil 4096 avril 42 20:42 ..
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 bonus
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 mariadb
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 nginx
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 tools
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 wordpress

./srcs/requirements/mariadb:
total XX
drwxrwxr-x 4 wil wil 4096 avril 42 20:45 .
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 ..
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 conf
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 Dockerfile
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 .dockerignore
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 tools
[...]
./srcs/requirements/nginx:
total XX
drwxrwxr-x 4 wil wil 4096 avril 42 20:42 .
drwxrwxr-x 5 wil wil 4096 avril 42 20:42 ..
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 conf
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 Dockerfile
-rw-rw-r-- 1 wil wil XXXX avril 42 20:42 .dockerignore
drwxrwxr-x 2 wil wil 4096 avril 42 20:42 tools
[...]

$> cat srcs/.env
DOMAIN_NAME=wil.42.fr
# MYSQL SETUP
MYSQL_USER=XXXXXXXXXXX
[...]
$>
```

For obvious security reasons, any credentials, API keys, passwords, etc., must be saved locally in various ways/files and ignored by git. Publicly stored credentials will lead you directly to a failure of the project.

You can store your variables (as a domain name) in an environment variable file like .env

Chapter VI

Readme Requirements

A `README.md` file must be provided at the root of your Git repository. Its purpose is to allow anyone unfamiliar with the project (peers, staff, recruiters, etc.) to quickly understand what the project is about, how to run it, and where to find more information on the topic.
The `README.md` must include at least:

- The very first line must be italicized and read: *This project has been created as part of the 42 curriculum by <login1>[, <login2>[, <login3>[...]]].*
- A "**Description**" section that clearly presents the project, including its goal and a brief overview.
- An "**Instructions**" section containing any relevant information about compilation, installation, and/or execution.
- A "**Resources**" section listing classic references related to the topic (documentation, articles, tutorials, etc.), as well as a description of how AI was used — specifying for which tasks and which parts of the project.

➠ **Additional sections may be required depending on the project** (e.g., usage examples, feature list, technical choices, etc.).

Any required additions will be explicitly listed below.

- A **Project description** section must also explain the use of Docker and the sources included in the project. It must indicate the main design choices, as well as a comparison between:
  - Virtual Machines vs Docker
  - Secrets vs Environment Variables
  - Docker Network vs Host Network
  - Docker Volumes vs Bind Mounts

Your README must be written in English.

Chapter VII

Prerequisites for validation

In addition to the existing requirements, the following documentation files must be present at the root of your repository. They must be written in `Markdown` format (`.md`).

- `USER_DOC.md` — **User documentation** This file must explain, in clear and simple terms, how an end user or administrator can:
  - Understand what services are provided by the stack.
  - Start and stop the project.
  - Access the website and the administration panel.
  - Locate and manage credentials.
  - Check that the services are running correctly.
- `DEV_DOC.md` — **Developer documentation** This file must describe how a developer can:
  - Set up the environment from scratch (prerequisites, configuration files, secrets).
  - Build and launch the project using the Makefile and Docker Compose.
  - Use relevant commands to manage the containers and volumes.
  - Identify where the project data is stored and how it persists.

Chapter VIII

Bonus part

For this project, the bonus part is intended to be simple.

A Dockerfile must be written for each additional service. Thus, each service will run inside its own container and will have, if necessary, its dedicated volume.

Bonus list:

- Set up `redis cache` for your WordPress website in order to properly manage the cache.
- Set up a `FTP server` container pointing to the volume of your WordPress website.
- Create a simple static website in the language of your choice except PHP (yes, PHP is excluded). For example, a showcase site or a site for presenting your resume.
- Set up `Adminer`.
- Set up a service of your choice that you think is useful. During the defense, you will have to justify your choice.

To complete the bonus part, you have the possibility to set up extra services. In this case, you may open more ports to suit your needs.

The bonus part will only be assessed if the mandatory part is completed perfectly. Perfect means the mandatory part has been fully completed and functions without any malfunctions. If you have not passed ALL the mandatory requirements, your bonus part will not be evaluated at all.

Chapter IX

Submission and peer-evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double-check the names of your folders and files to ensure they are correct.
During the evaluation, a brief **modification of the project** may occasionally be requested. This could involve a minor behavior change, a few lines of code to write or rewrite, or an easy-to-add feature.

While this step may **not be applicable to every project**, you must be prepared for it if it is mentioned in the evaluation guidelines.

This step is meant to verify your actual understanding of a specific part of the project. The modification can be performed in any development environment you choose (e.g., your usual setup), and it should be feasible within a few minutes — unless a specific timeframe is defined as part of the evaluation.
You can, for example, be asked to make a small update to a function or script, modify a display, or adjust a data structure to store new information, etc.

The details (scope, target, etc.) will be specified in the **evaluation guidelines** and may vary from one evaluation to another for the same project.