TAMPERE UNIVERSITY

EXERCISE REPORT
COMP.SE.140

EXERCISE 01

BY
NADON, Alexis

SEPTEMBER 29th 2025

Table of content

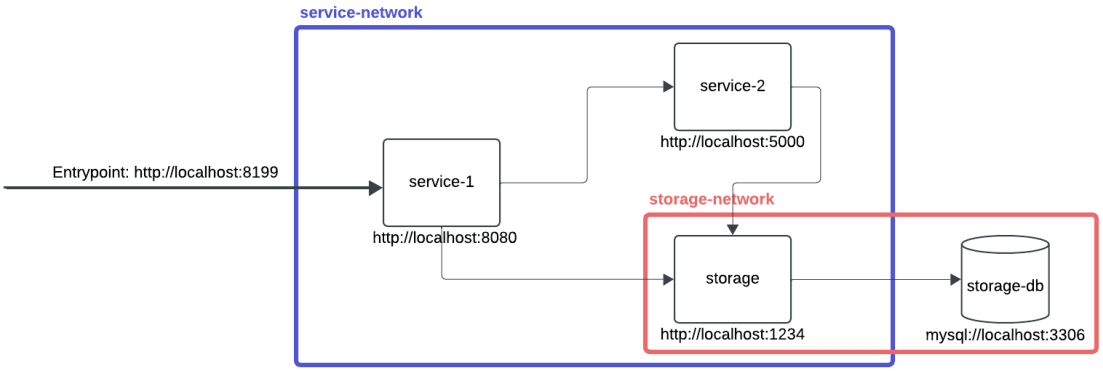
1. Basic information

In order to create this small project, here is my computer configuration to be able to run the project.

Hardware	MacBook Pro M1
Operating system	MacOS
Docker version	28.4.0, build d8eb465
Docker compose version	v2.39.2-desktop.1

2. Overview of the architecture

To illustrate the architecture of this exercise, here's a diagram that shows all the different services and the docker networks.



As we can see, the only entry point is the port 8199 which is redirected to the Service-1 container. Service-1 will then act as a proxy, since it will redirect incoming requests to the correct service.

3. Status record analysis

When the API endpoint /status is called, Service 1 and Service 2 perform an analysis of the system. Both services return the same information. The first piece of information returned is the uptime of the service. This measurement is calculated in the same way for both services: by saving the timestamp when the service is initialized. Each time the endpoint is called, the current time is subtracted from the saved timestamp to calculate the uptime. Therefore, it always returns the uptime of the container, not the host. For the free space measurement, since each service reads the free space from the root directory, the value reflects the free space available on the host. This makes the measurement less relevant, since it should return the available space of the container to ensure the data is binded to the container state.

A better approach would be to check the available space of a volume mounted to the container. With Docker Compose, it is possible to limit the size of a volume, and by checking the available space of this specific folder, we would obtain a more accurate and relevant measurement.

4. Persistent storage comparison

This project includes two different ways to store information. One is using a file mounted to some containers, and the other uses a separate container that encapsulates the storing logic.

4.1 Mounted file

This solution is the easiest to implement, since it does not require much code. We only need to implement some file writing logic in the containers that need to store persistent data, and Docker will mount the specific volume to ensure persistence. Although it is the easiest, it is not the best solution since it doesn't encapsulate the concerns. This means that each container needs to implement the storing logic on the specific mounted file. This can involve a lot of code duplication, and moreover, writing to a file doesn't handle concurrency well. Two different containers could try to write to the same file at the same time and corrupt some data.

4.2 Storage container

The second solution can be more complicated to implement, but it is more robust. By encapsulating the storing logic in its own container, we have more control on how we want to store the data. For this small project, I have created a small API in Python and connected a database to it to be able to store data. This way, I had the choice of how I wanted to store the logs. If I want to change container technology, I am not bound to the volume functionality of Docker. In addition, my implementation is more robust to concurrency since the container handles the read/write operations. Although this advantage does not necessarily come from the storage container, it is related to my own implementation.

5. Cleanup instruction

Here is the following procedure to be able to cleanup the logs from the Storage container.

```
docker compose down
docker volume rm exercise-01_storage-db-data
```

6. Difficulties and problems

During this exercise, I had some problems with the Docker volume system. It was the first time that I needed to mount one specific file. Although I tried numerous times to follow the official documentation, Docker was always creating a folder for me instead of a file. To make it work, I followed the instructions on the StackOverflow forum cited in the **e1Compose_v13.pdf** file. I needed to manually create the vStorage file instead of letting Docker handle it.

Another difficulty I encountered was with my API in Python. It was the first time that I created a python API, so it took me more time than usual. I had problems with the framework that initiated the connection to the database. With the help of the Internet, and a lot of trial and error, I finally succeeded.

7. Conclusion

Overall, this exercise was great for practicing some concepts that I already knew. It was a good way to see the different features that Docker offers and how easy it is to let different containers communicate with each other. I find it very interesting to compare different ways to persistently store data. It helped me a lot to refresh my knowledge about volumes, a concept that is really important.