Name: Anh Tran
Student ID: 150511795

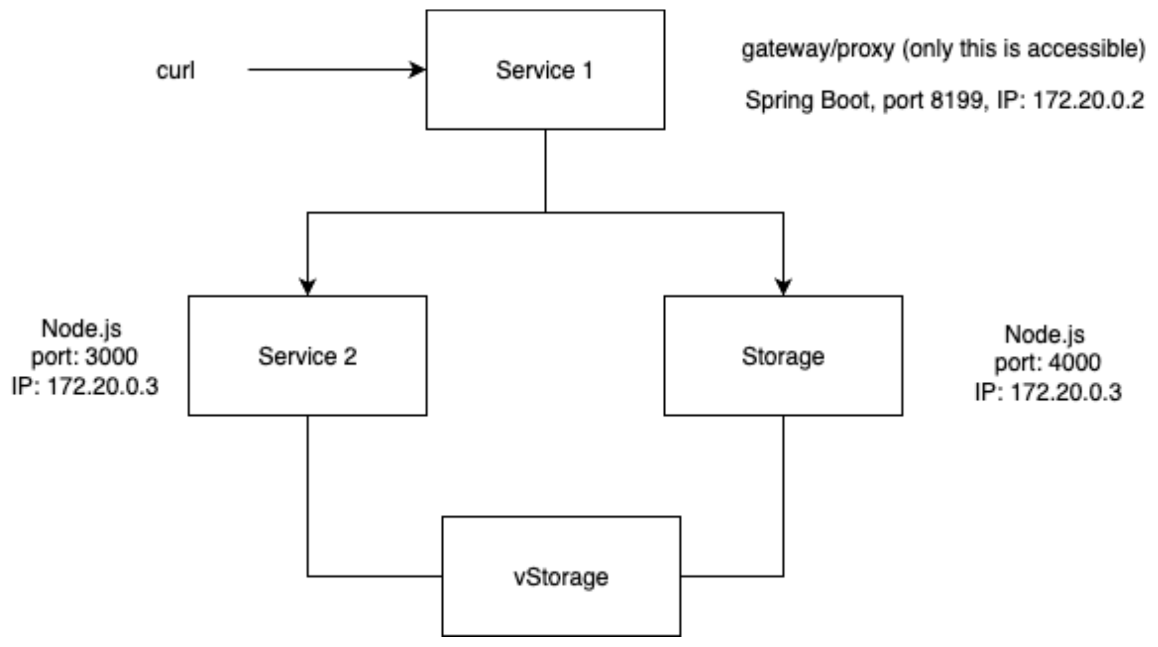
Exercise 1: Basics of containers and microservices

Report

1. Basic information about the platform

- Hardware: Apple Macbook (Apple M3) with 8GB RAM
- Operating system: macOS Sequoia
- Docker version: 28.0.1
- Docker compose version: v2.33.1

2. Diagram of the system



Service 1 is the only accessible service from outside. It acts as a gateway to the system. Every time there's a request, it creates its own status record (log). Then Service 1 forwards the request to Service 2 and Storage service. Service 2 generates its own status record, then logs its record to shared volume (./vstorage) and Storage service, and returns the log to Service 1 when asked. Storage service provides REST API to append new log entries and return all logs to service 1 and service 2.

3. Analysis of Status Records

The status record looks like this (log.txt)

```
Timestamp1: uptime 0.02 hours, free disk in root: 963204.43 Mbytes at 2025-09-
24T14:54:59.255548136Z
Timestamp2: uptime 0.09 hours, free disk in root: 963204.43 Mbytes at 2025-09-
24T14:54:59.512Z
Timestamp1: uptime 0.02 hours, free disk in root: 963204.43 Mbytes at 2025-09-
24T14:55:03.521786388Z
Timestamp2: uptime 0.09 hours, free disk in root: 963204.43 Mbytes at 2025-09-
24T14:55:03.549Z
Timestamp1: uptime 0.03 hours, free disk in root: 963204.43 Mbytes at 2025-09-
24T14:55:26.444431343Z
Timestamp2: uptime 0.10 hours, free disk in root: 963204.43 Mbytes at 2025-09-
24T14:55:26.469Z
```

Uptime

Service1 uses Java runtime uptime and Service2 uses OS uptime (from Node.js). These measure container runtime and OS uptime respectively. However, recorded time logs don't reflect real system uptime, but just container/runtime values.

Free Disk

Measure available space in container's root filesystem. The measured value is usually the free space of Docker VM volume, not the actual host disk. It is not really relevant, but it still demonstrates how to gather resource metrics. For improvement, monitoring tool like Prometheus can be used.

4. Storage solutions comparison

The vStorage host-mounted directory is easy to inspect by cat command. However, it is a very bad design because it breaks container isolation, ties app to host filesystem.
Storage service with its own volume is a more proper microservice approach in this case. It is fully isolated and persistent across restarts. It can also be accessible via REST API, not host-dependent like the vStorage. The only downside is that it is slightly harder to debug logs because you need to access it via API.
Overall, option 2 – build a Storage service is a better solution for microservices system.

5. Instructions for clean up the persistent storage

To stop containers and remove volumes

```
docker-compose down -v
```

To clean logs in vStorage, we can manually delete host-mounted logs using

```
rm ./vstorage/log.txt
```

6. Difficulties

When doing this assignment, it took me a while to understand relations between 3 services (service 1, service 2 and storage). I also got confused about the recorded metrics and their meanings. After some researching, I figured out how the networking works and implemented it.

7. Main problems encountered

- The first error was JAR not found because Spring Boot JAR name in the Dockerfile was incorrect.
- Logs didn’t appear on host until I switched from named volume to bind mount (./vstorage)
- Then I debugged container IPs and tested with Postman to make sure that only service 1 is accessible.