Họ Tên: Trần Đình Kiên

MSSV: 21065701

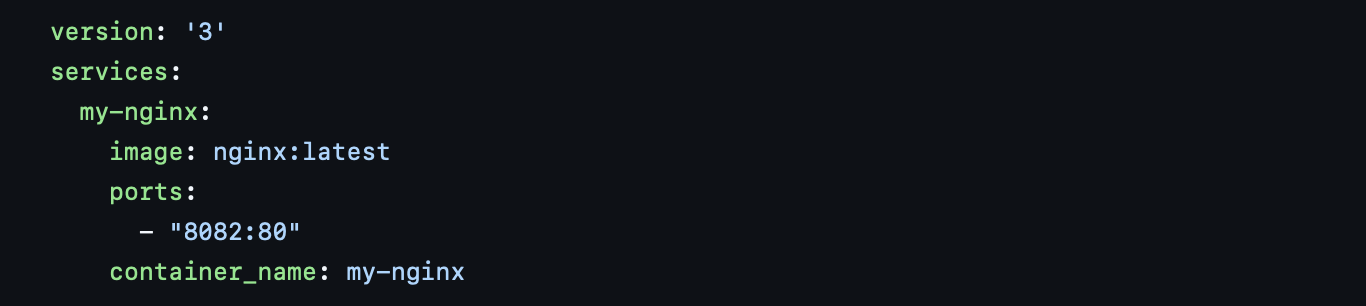
**Phần 2: Docker Compose file**

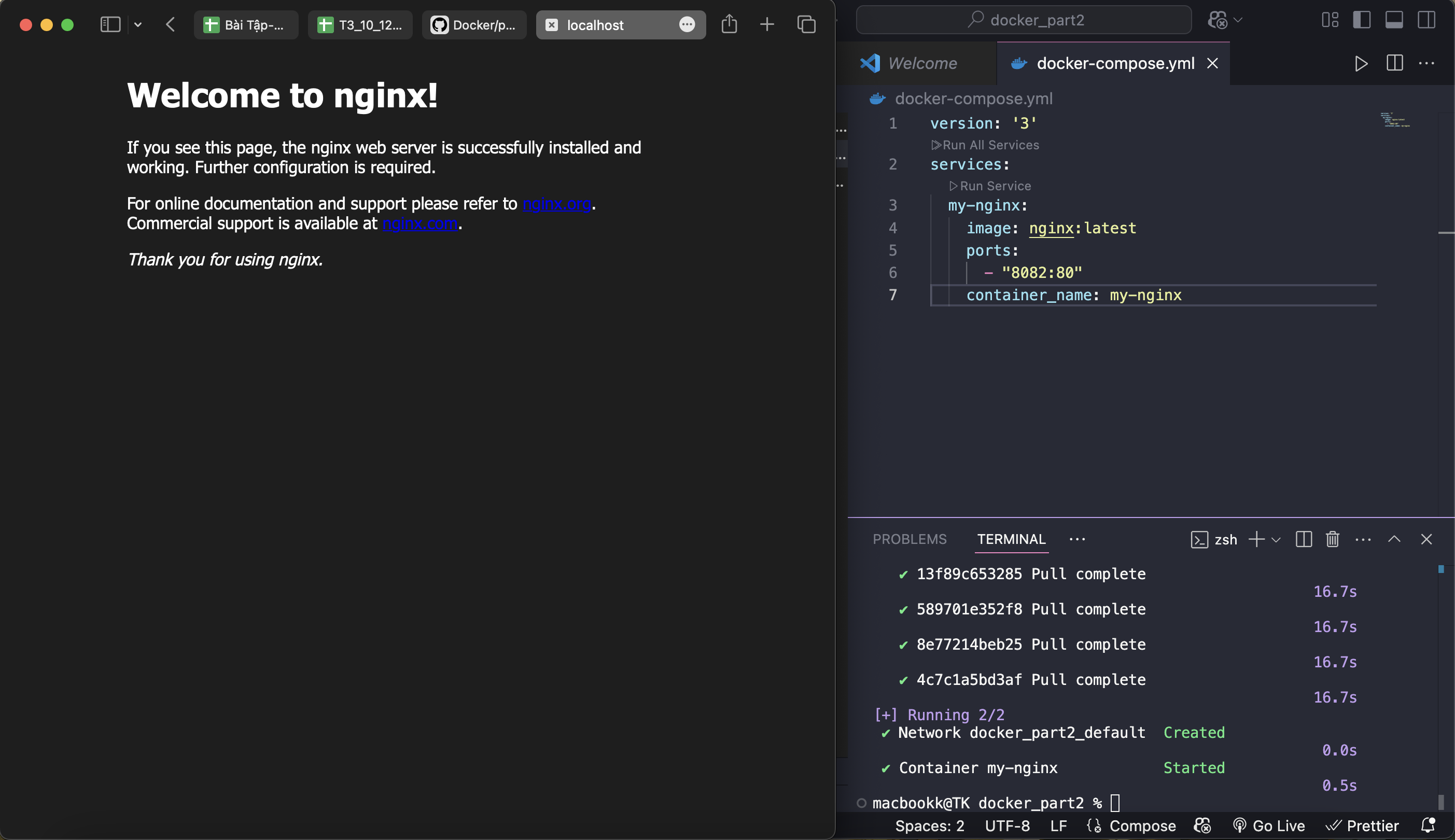
**Bài 1: Chạy một container đơn giản với Docker Compose**

Yêu cầu:

Tạo một container chạy Nginx bằng Docker Compose.

Map cổng 8080 của máy host với cổng 80 của container.



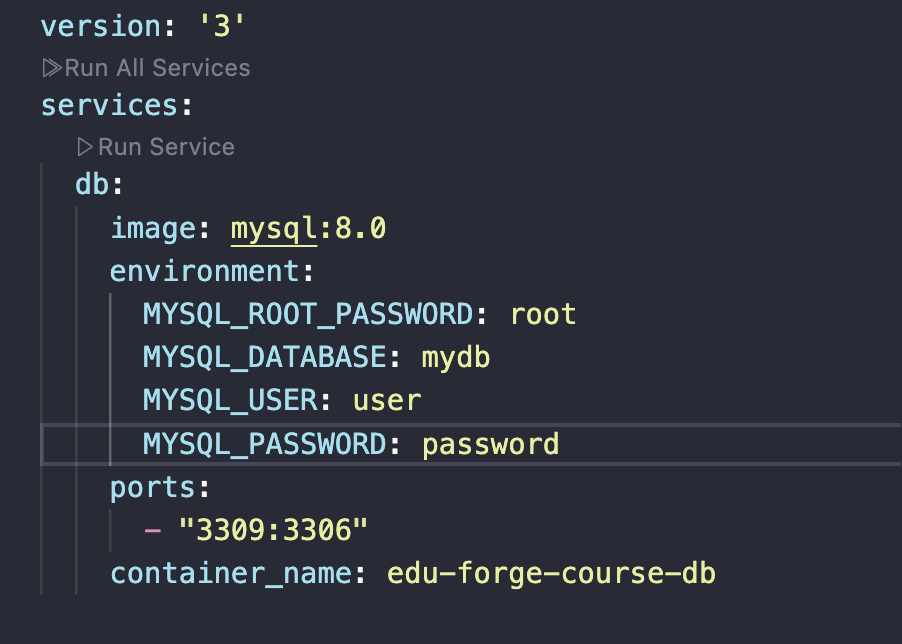


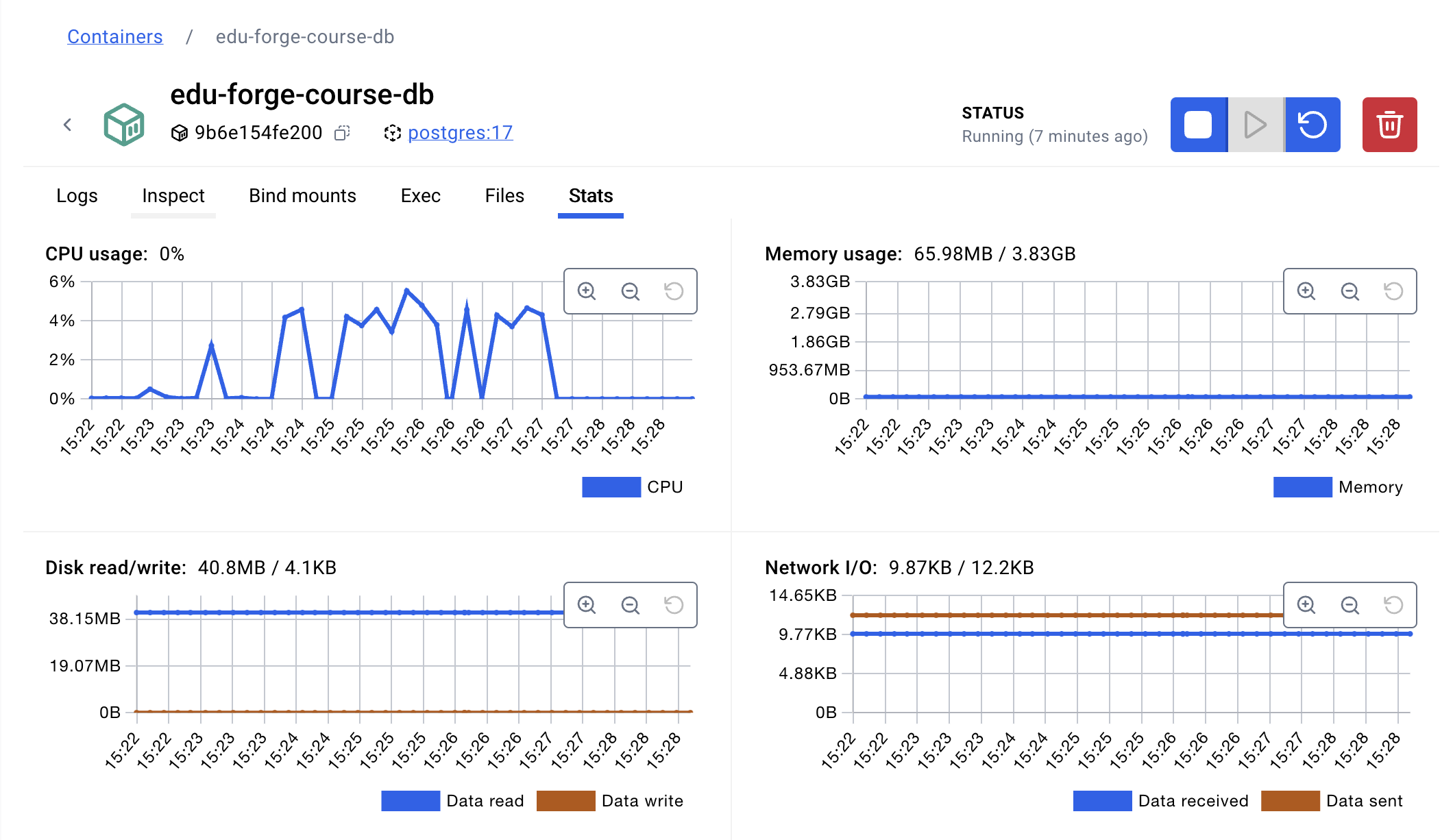
**Bài 2: Chạy MySQL với Docker Compose**

Yêu cầu:

Tạo một container chạy MySQL phiên bản 8.0.

Đặt username là user, password là password và database là mydb



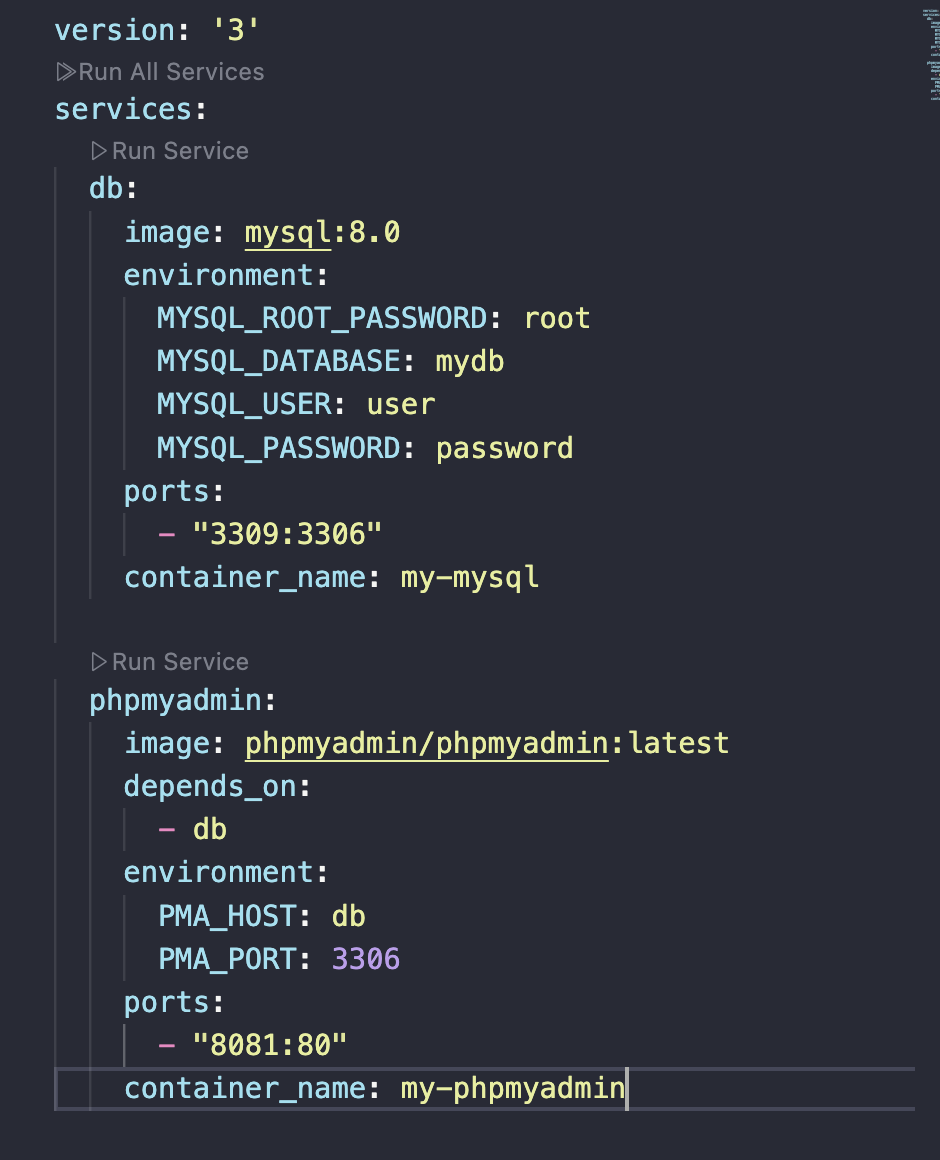


**Bài 3: Kết nối MySQL với PHPMyAdmin**

Yêu cầu:

Chạy MySQL và PHPMyAdmin với Docker Compose.

PHPMyAdmin chạy trên cổng 8081.



**Bài 4: Chạy ứng dụng Node.js với Docker Compose**

Yêu cầu:

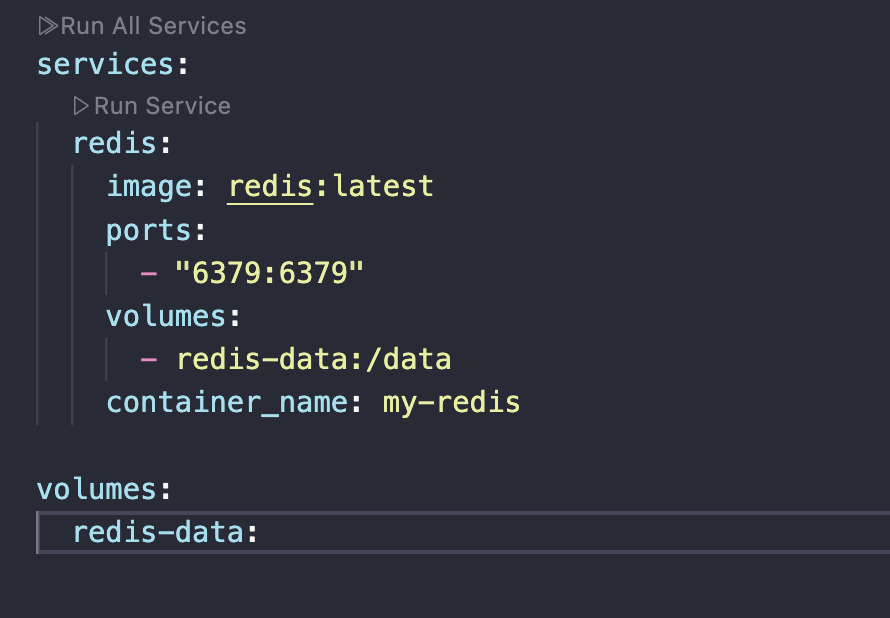
Chạy một ứng dụng Node.js đơn giản với Express.

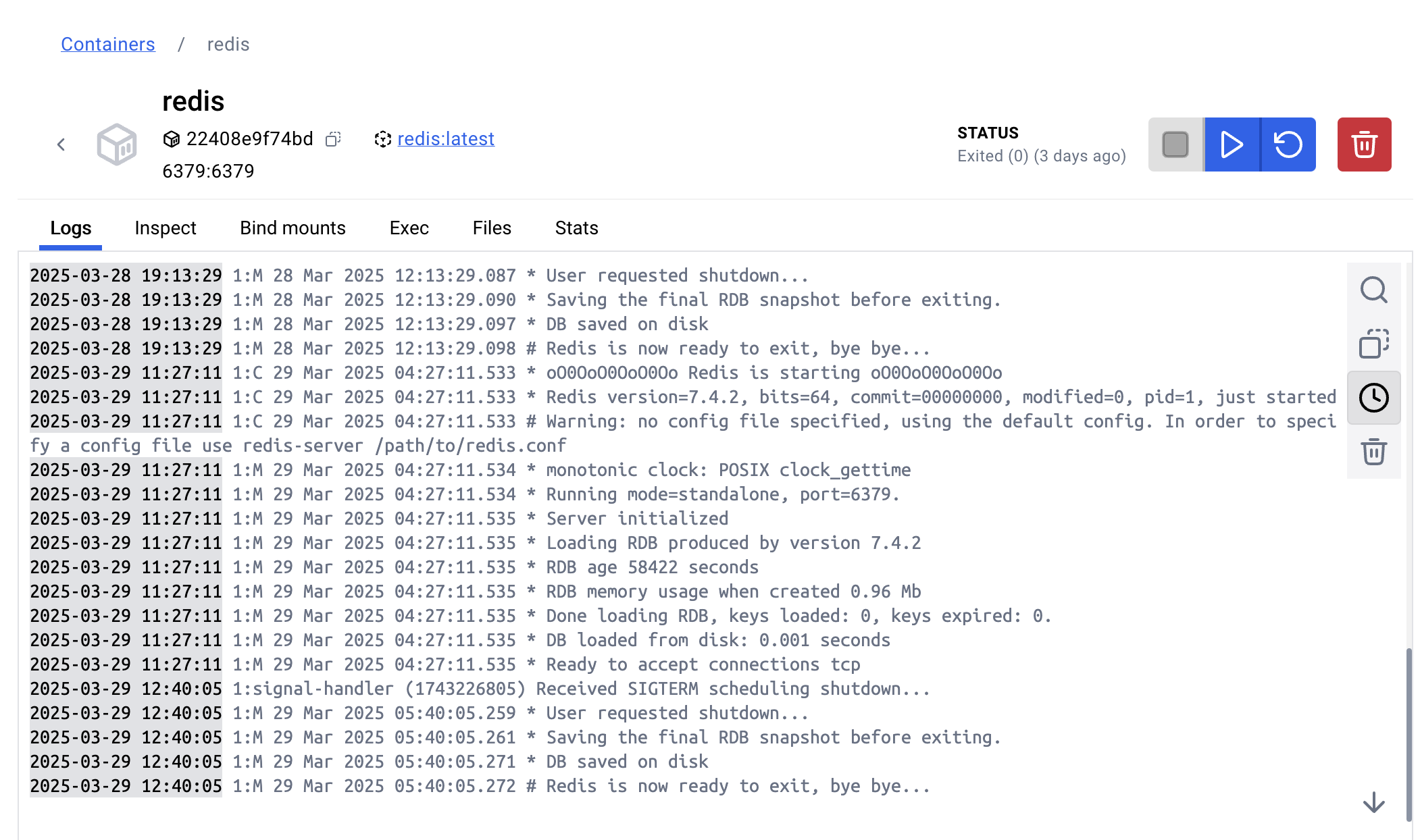


**Bài 5: Chạy Redis với Docker Compose**

Yêu cầu:

Chạy một container Redis trên cổng 6379.

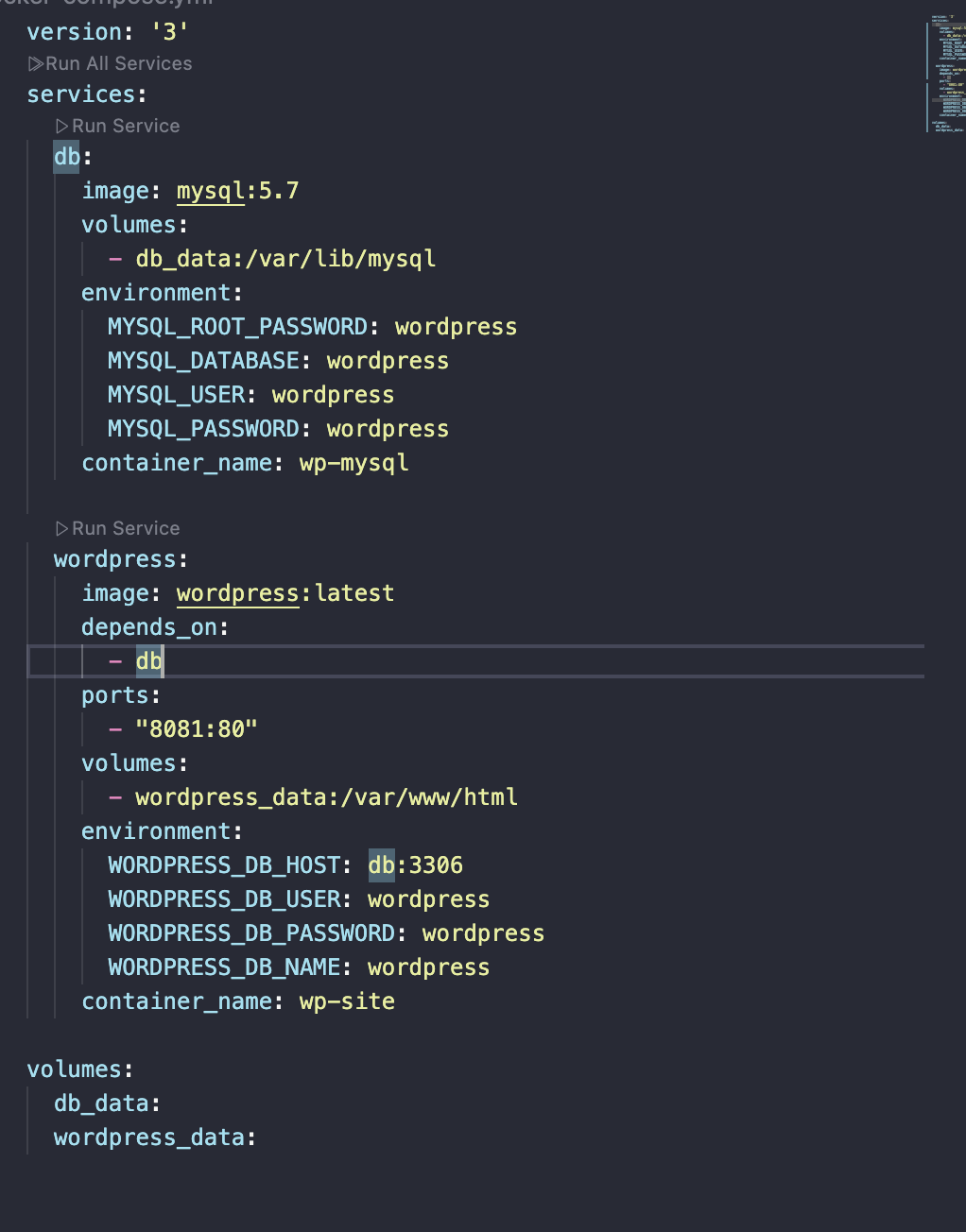




**Bài 6: Chạy WordPress với MySQL**

Yêu cầu:

Chạy WordPress với MySQL bằng Docker Compose.

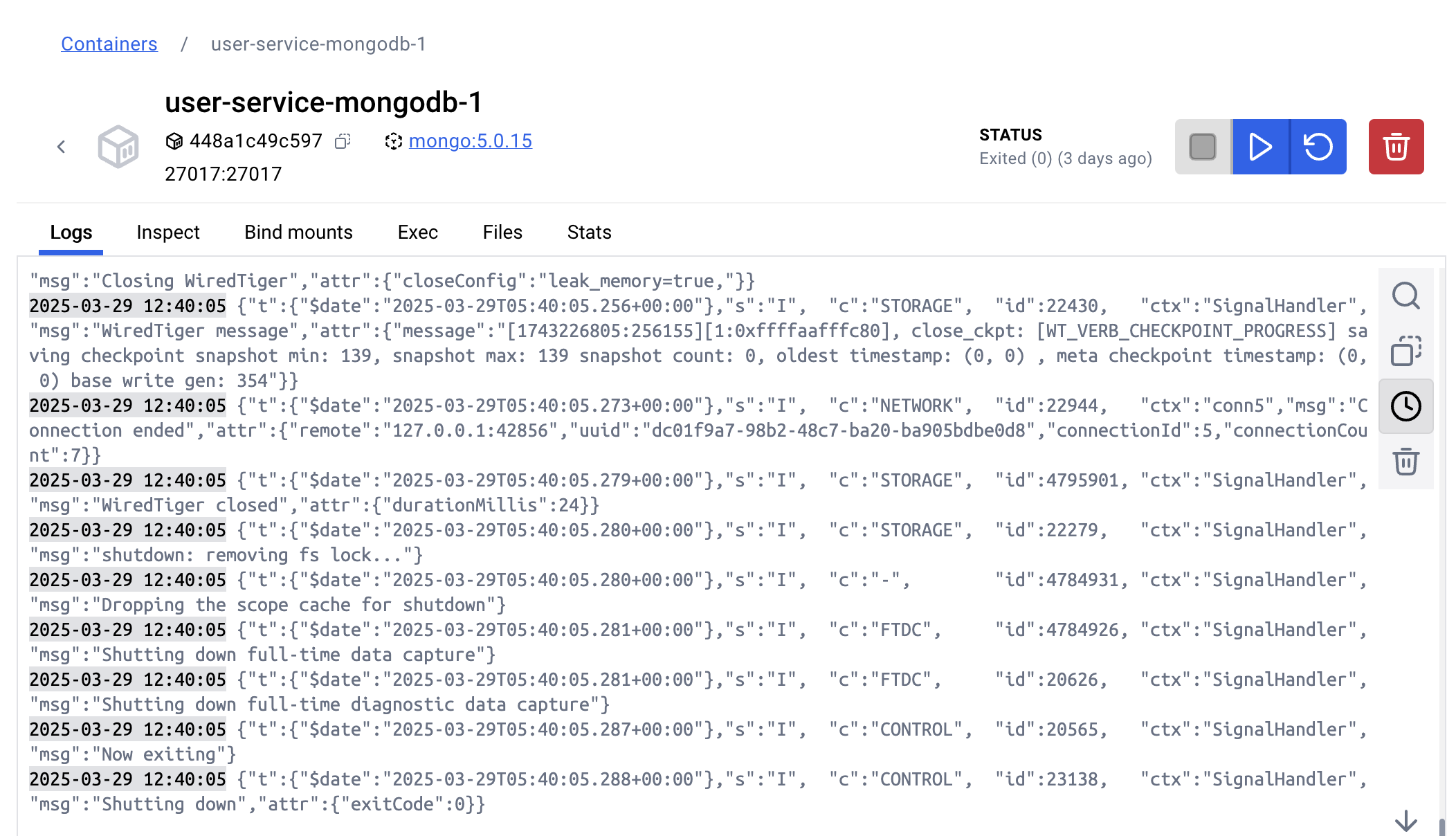


**Bài 7: Chạy MongoDB với Docker Compose**

Yêu cầu:

Chạy MongoDB và Mongo Express để quản lý.





**Bài 8: Kết nối nhiều dịch vụ với Docker Compose**

Yêu cầu:

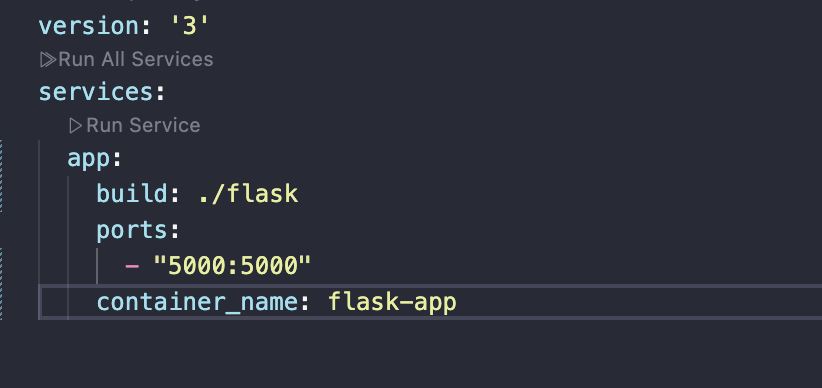
Chạy Node.js kết nối với MySQL.

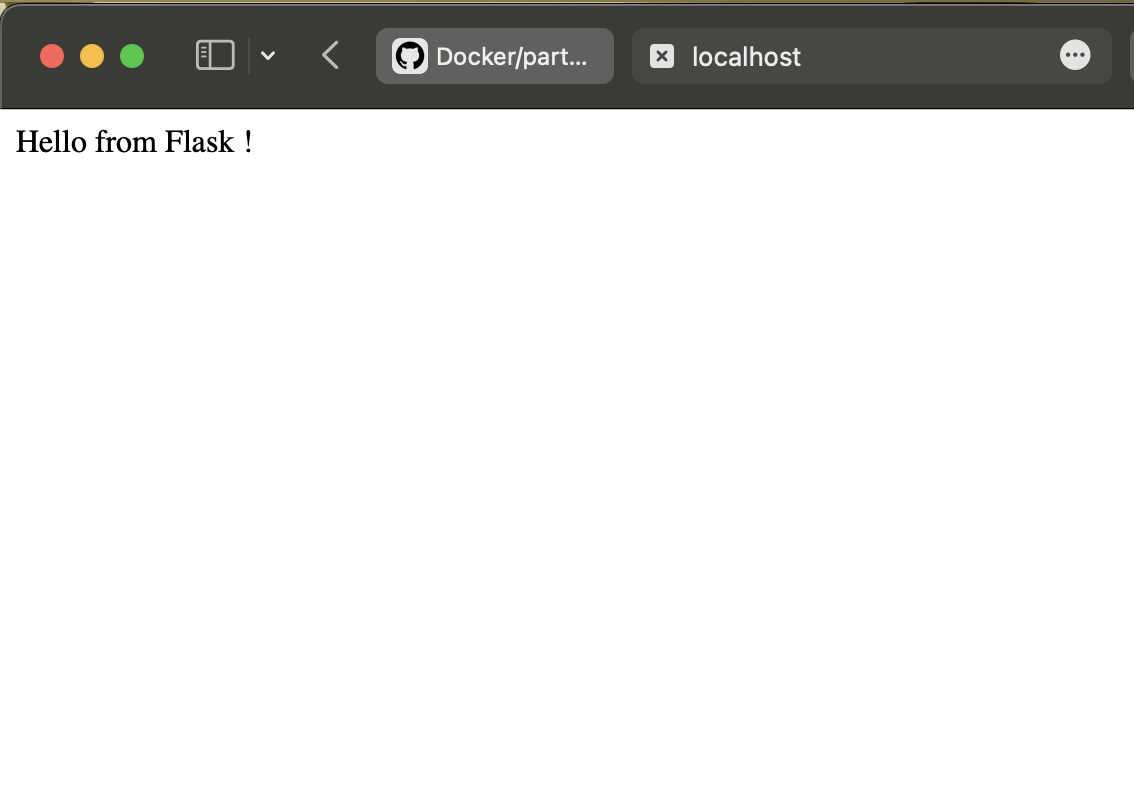


**Bài 9: Chạy ứng dụng Python Flask với Docker Compose**

Yêu cầu:

Chạy ứng dụng Flask đơn giản với Docker Compose.

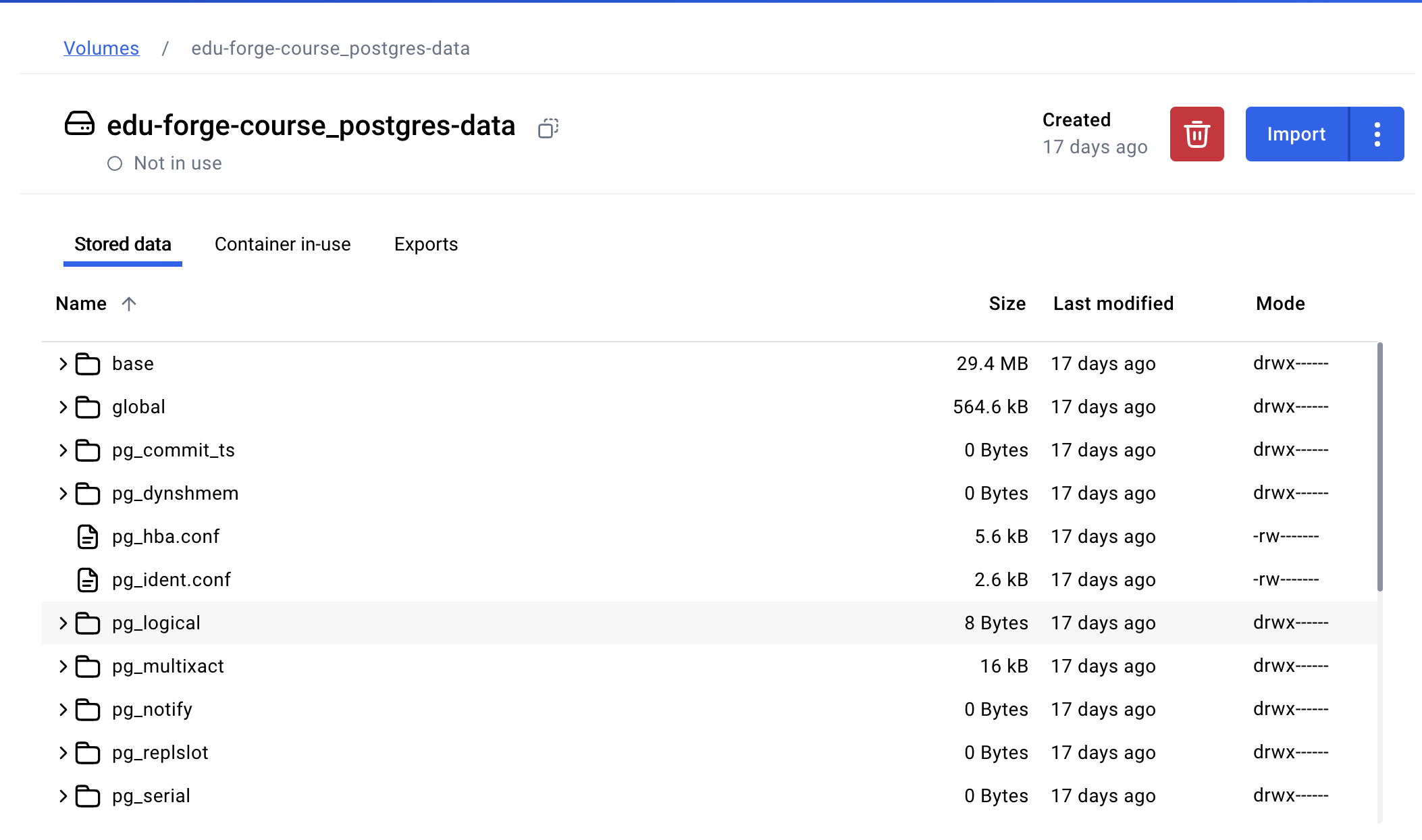


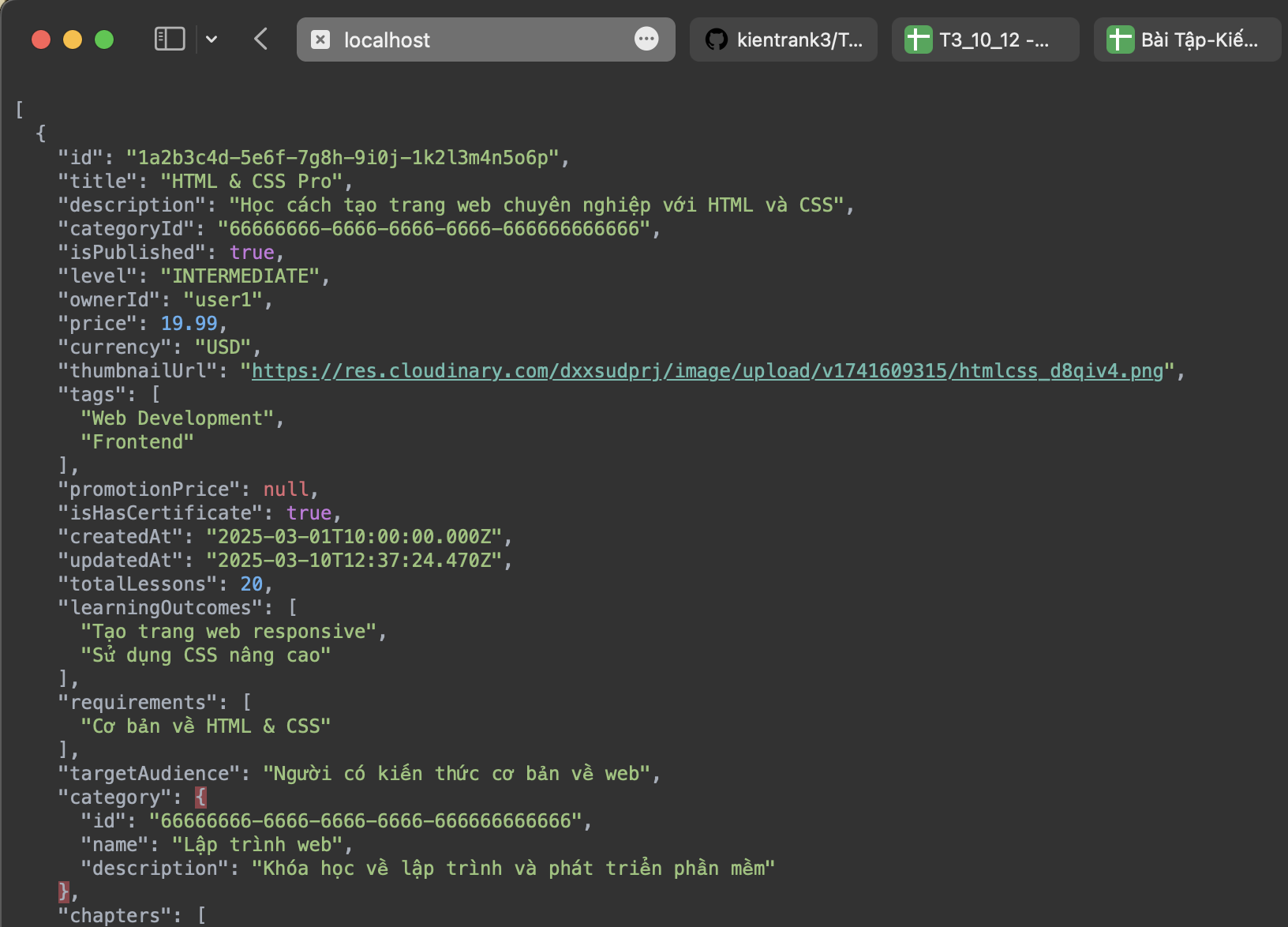


**Bài 10: Lưu trữ dữ liệu với Docker Volumes**

Yêu cầu:

Chạy MySQL và gắn volume để dữ liệu không bị mất.





**Phần 3: Docker Compose file**

**Bài tập 1: Triển khai WordPress với MySQL**

Mục tiêu: Tạo stack WordPress kết nối với MySQL, sử dụng volumes để lưu trữ dữ liệu.

Yêu cầu:

1. Sử dụng image wordpress:latest (port 80).

2. Sử dụng image mysql:5.7 (port 3306).

3. Volume cho database (/var/lib/mysql).

4. Biến môi trường cho MySQL:

MYSQL\_ROOT\_PASSWORD, MYSQL\_DATABASE, MYSQL\_USER, MYSQL\_PASSWORD

Gợi ý:

WordPress cần khai báo depends\_on MySQL.

Sử dụng network tùy chỉnh để kết nối giữa 2 service.



**Bài tập 2: Ứng dụng Node.js + MongoDB**

Mục tiêu: Triển khai ứng dụng Node.js (lưu dữ liệu vào MongoDB) và MongoDB với volume.

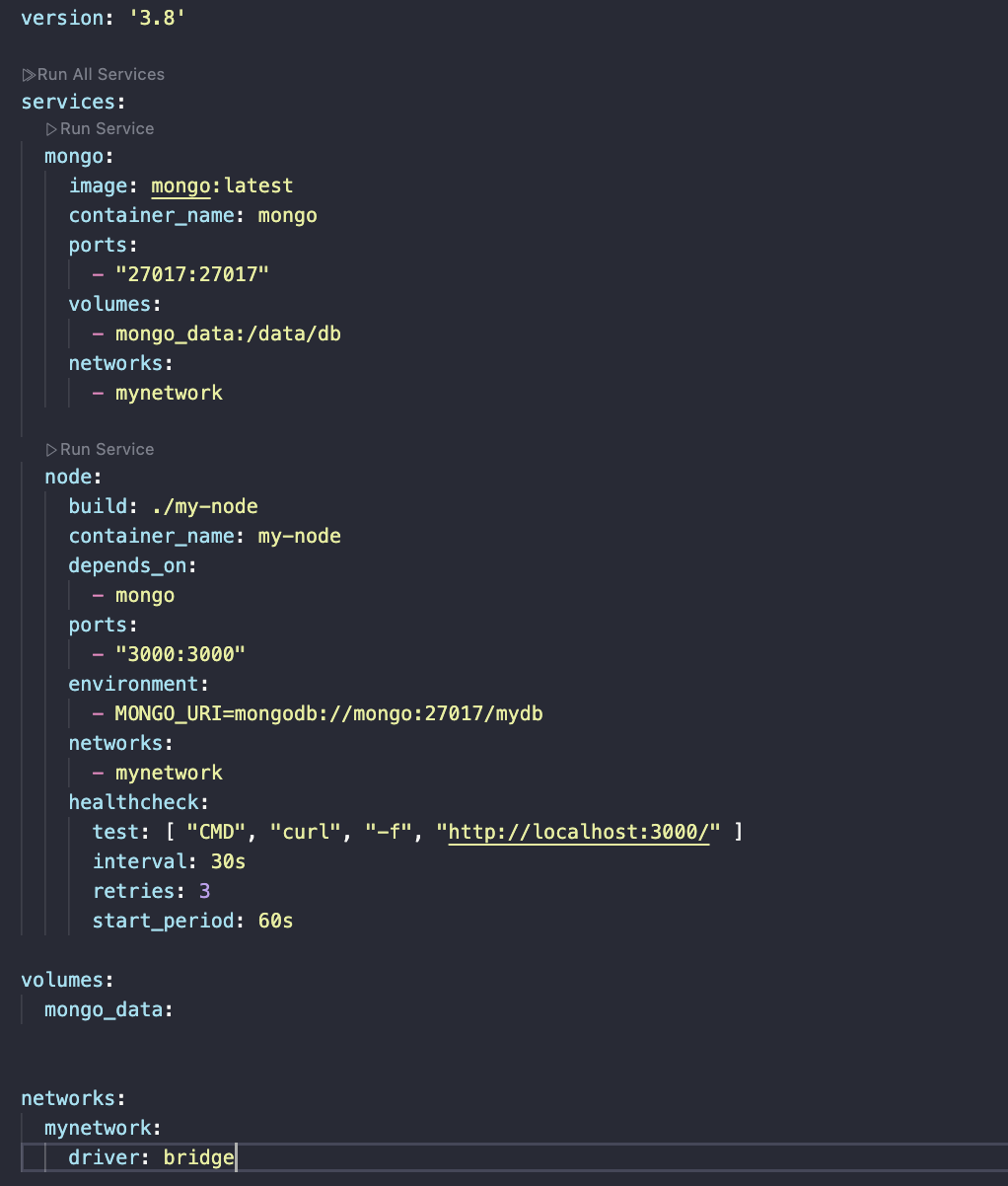
Yêu cầu:

1. Viết Dockerfile cho ứng dụng Node.js (ví dụ: REST API đơn giản).

2. Sử dụng image mongo:latest (port 27017).

3. Volume cho MongoDB (/data/db).

4. Đảm bảo Node.js service khởi động sau MongoDB (depends\_on + healthcheck).



**Bài tập 3: Load Balancing với Nginx + Flask**

Mục tiêu: Cân bằng tải giữa 2 instance Flask dùng Nginx.

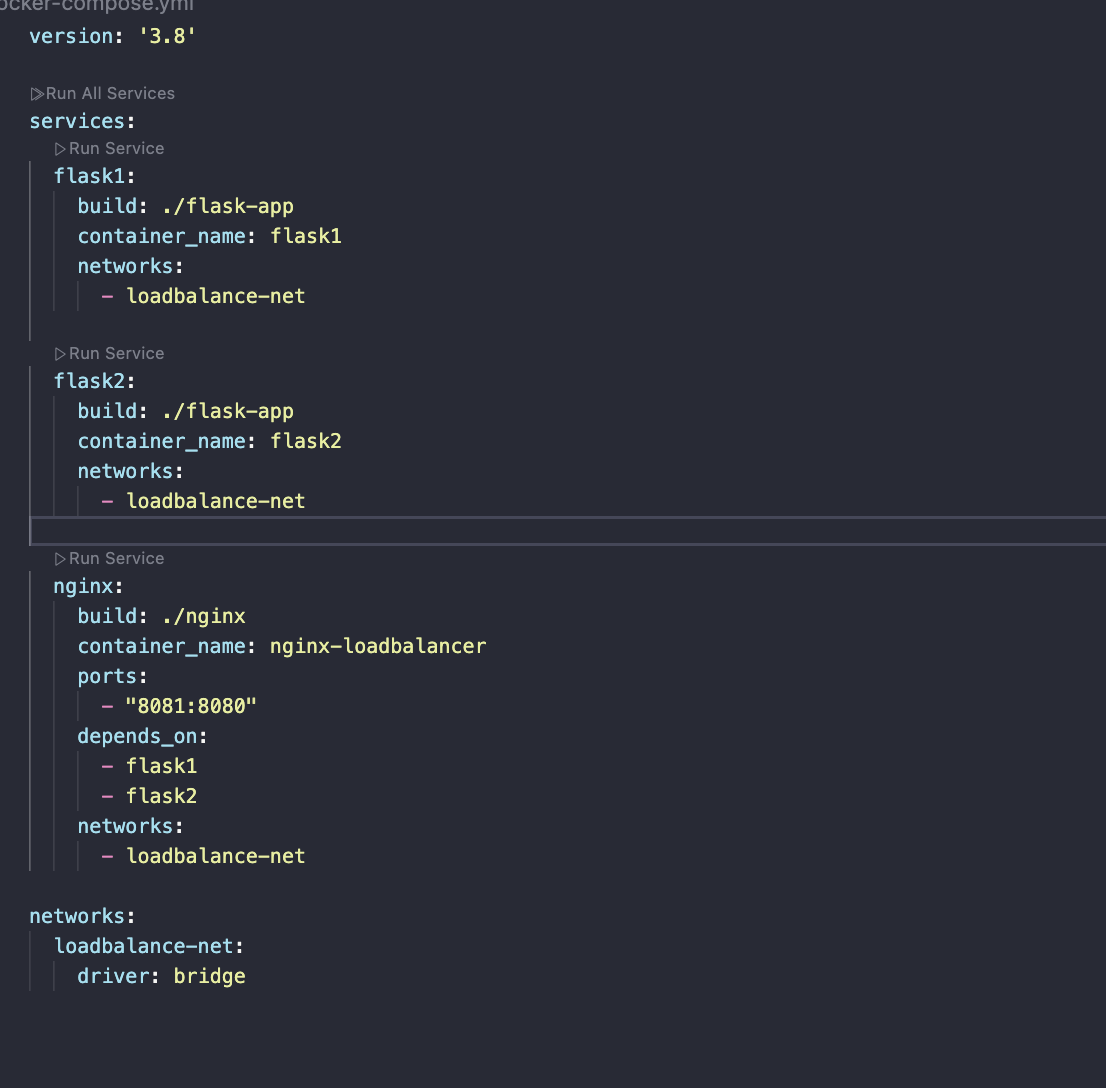
Yêu cầu:

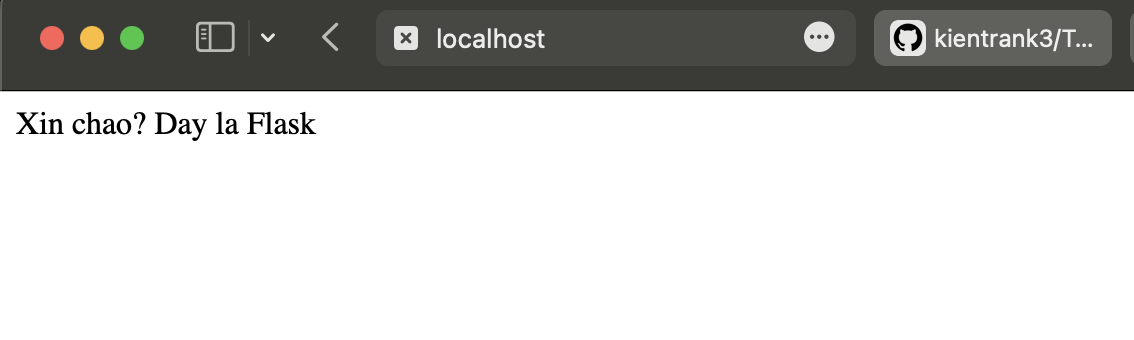
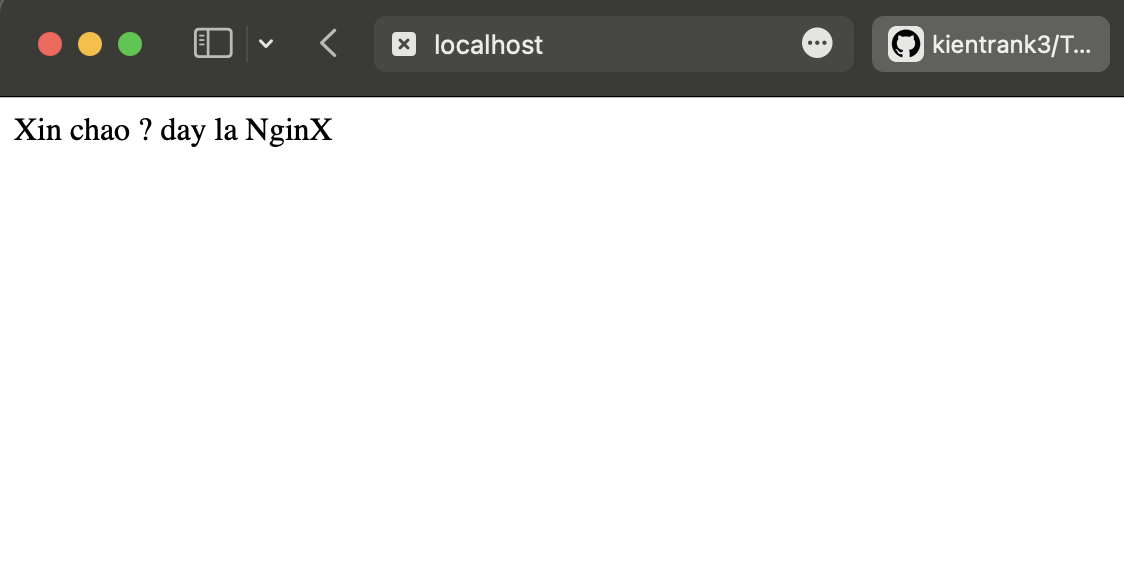
1. 2 service Flask (sử dụng app.py từ bài tập trước, port 5000).

2. 1 service Nginx (port 8080) cấu hình làm reverse proxy:

Chuyển request / đến các Flask instance (round-robin).

3. Tạo custom network và Nginx config.





**Bài tập 4: Prometheus + Grafana Monitoring**

Mục tiêu: Giám sát Docker containers dùng Prometheus và Grafana.

Yêu cầu:

1. Service Prometheus (port 9090) với file cấu hình thu thập metrics từ Docker.

2. Service Grafana (port 3000) kết nối đến Prometheus.

3. Volume để lưu dữ liệu Prometheus và Grafana.

**Bài tập 5: Elasticsearch + Kibana**

Mục tiêu: Triển khai ELK stack đơn giản.

Yêu cầu:

1. Service Elasticsearch (port 9200) với volume.

2. Service Kibana (port 5601) kết nối với Elasticsearch.

3. Thiết lập environment variables cho credentials.