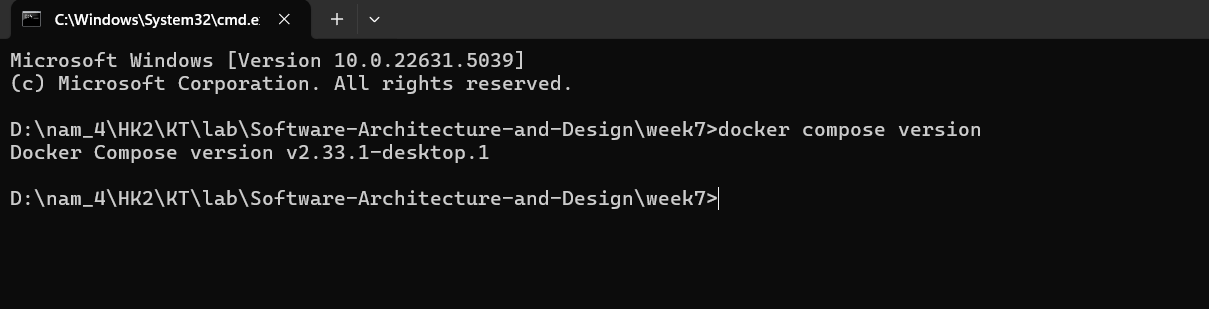
Ngô Văn Toàn – 21077971

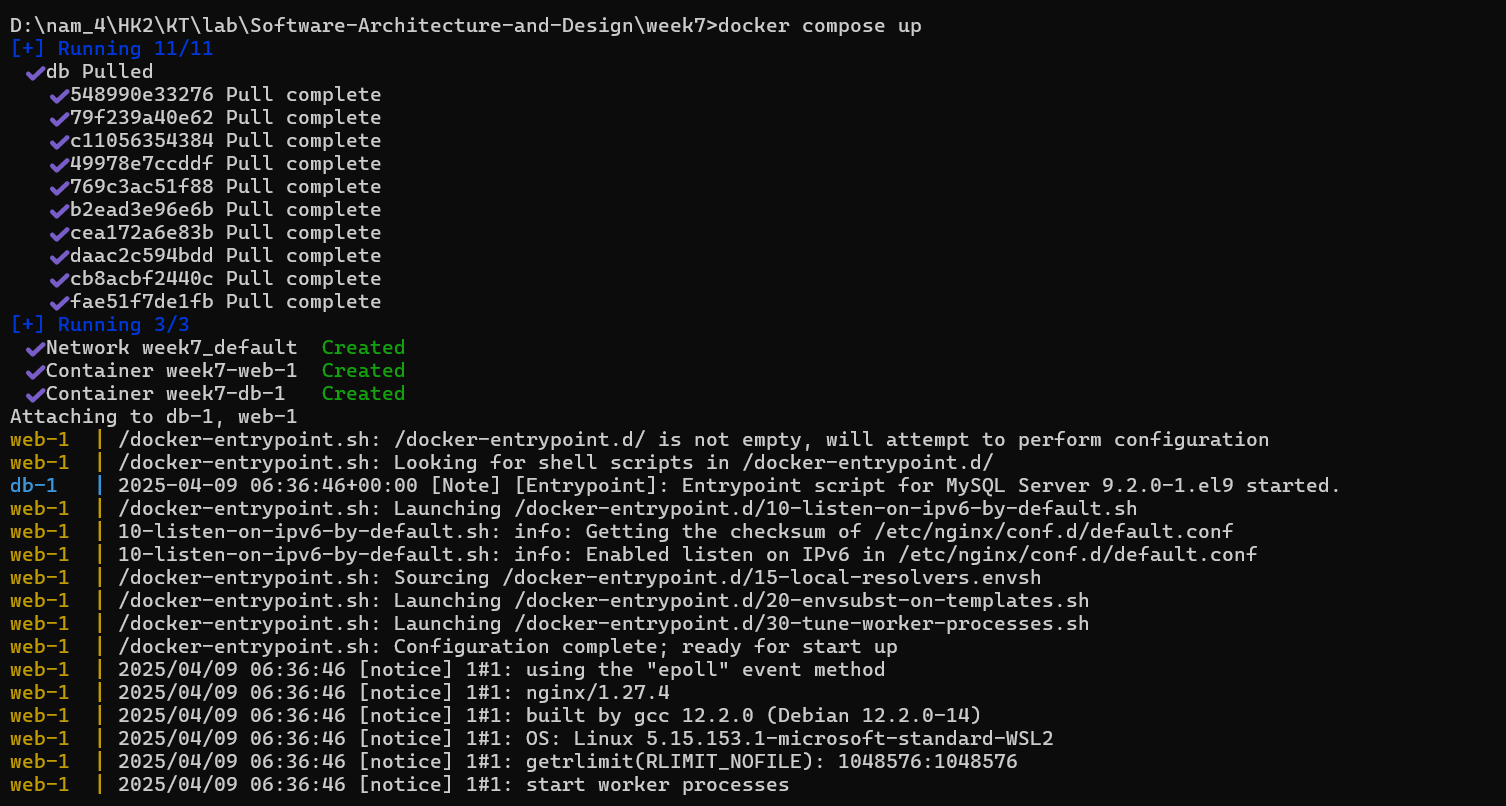
**Week 7: Docker Compose**

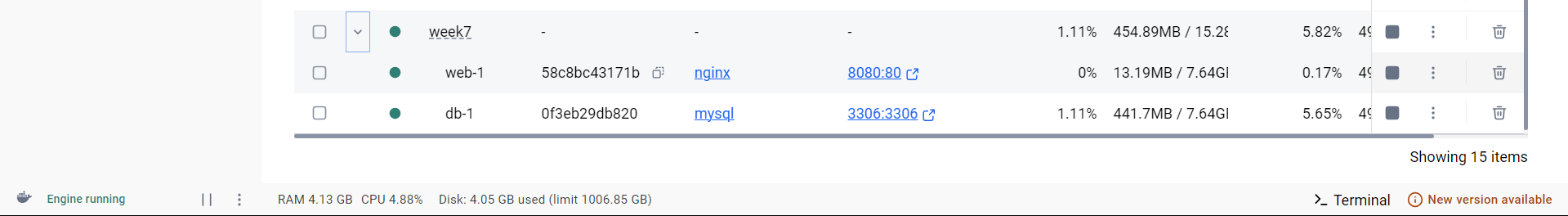
**Phần 1: Một số lệnh Docker Compose cơ bản**

1. docker compose version: Hiển thị phiên bản Docker Compose đang sử dụng



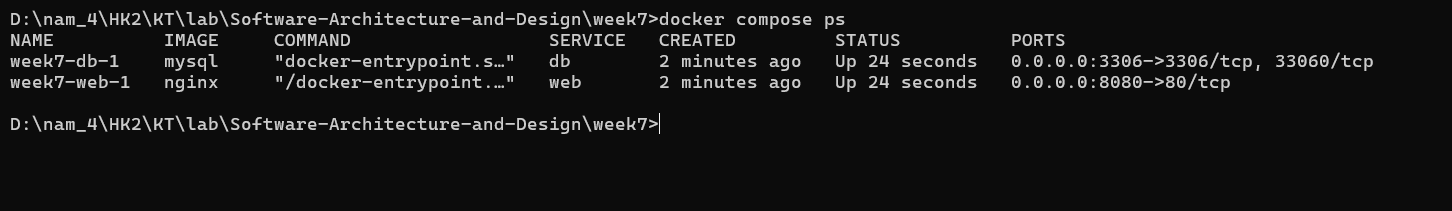
1. docker compose up: Khởi động các service được định nghĩa trong docker-compose.yml, hiển thị log trong terminal.



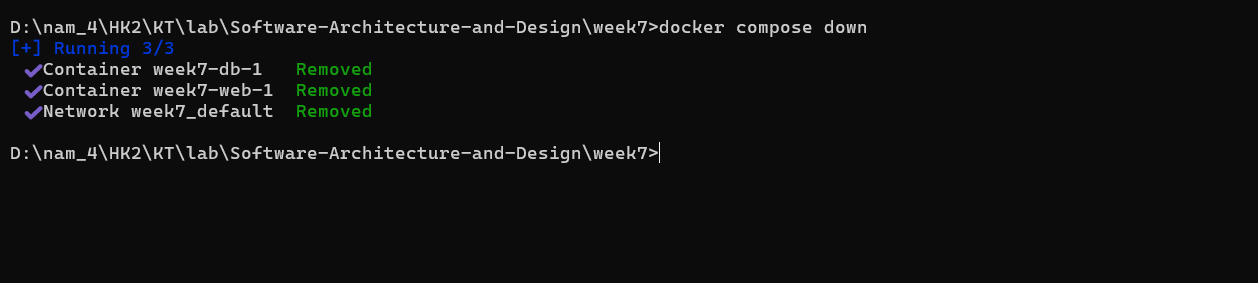


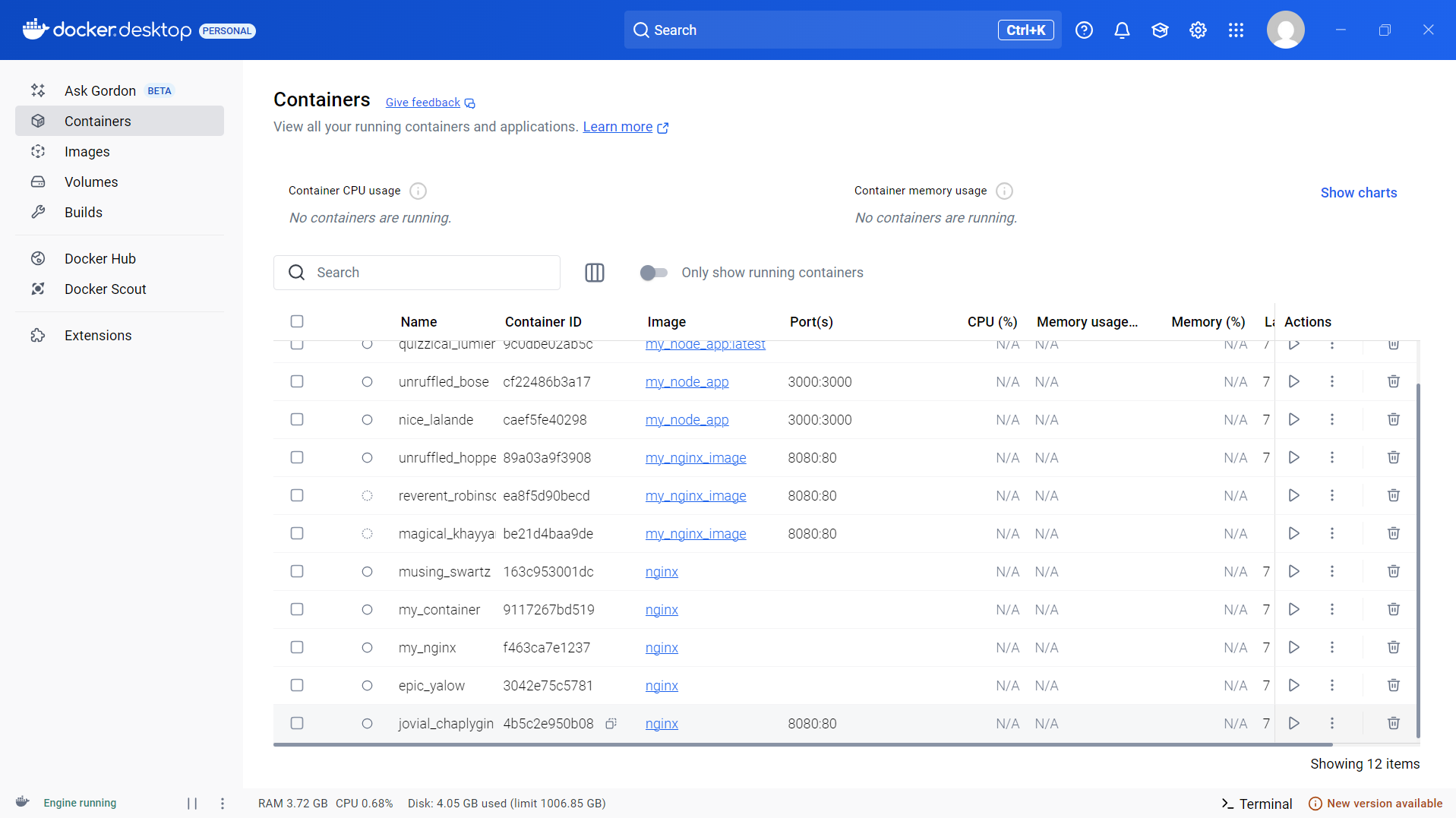
1. docker compose up -d: Khởi động các service ở chế độ nền (detached mode).



4 docker compose ps: Liệt kê các container đang chạy theo project. 

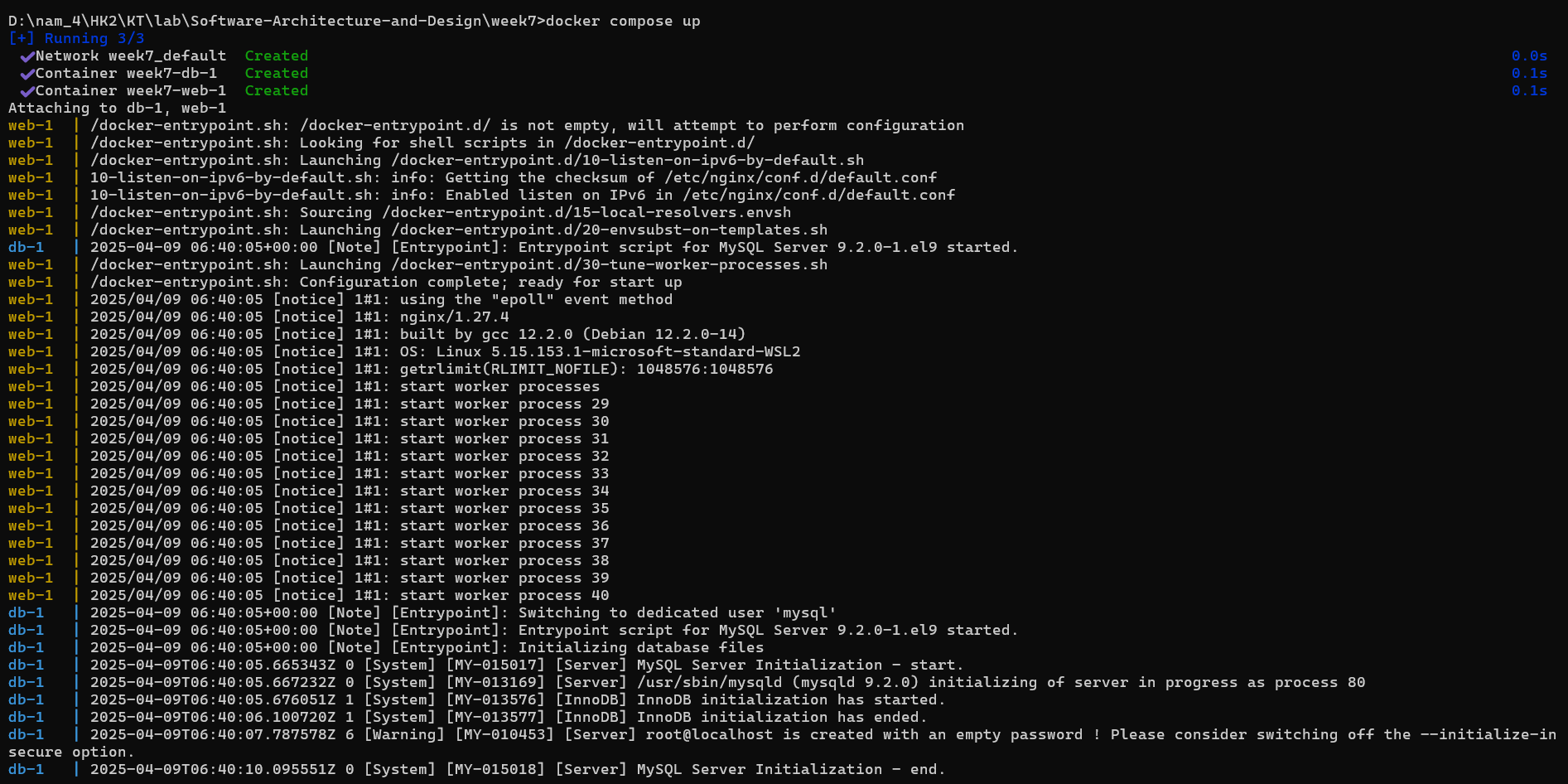
5 docker compose down: Dừng và xóa tất cả container, network (mặc định giữ lại volume và image).

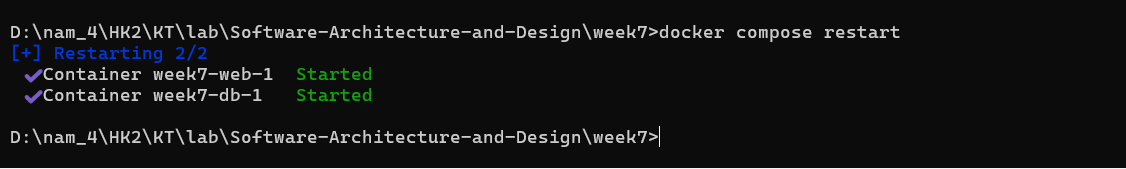




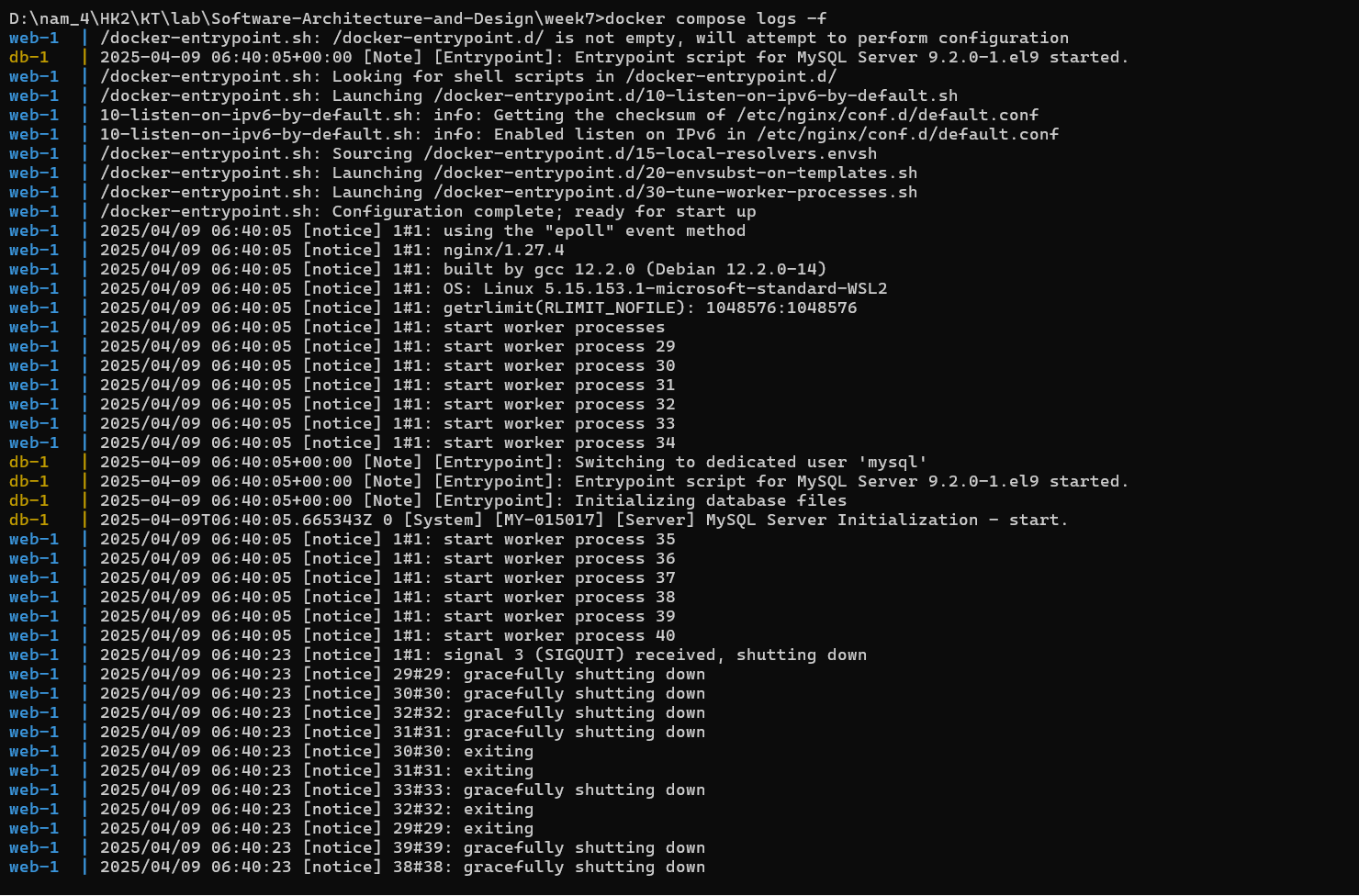
6 docker compose restart: Khởi động lại toàn bộ các service.

Trước đó ta đã xóa toàn bộ container đi, nên muốn khởi động lại thì phải thực hiện khởi động lại bằng lệnh docker compose up trước khi restart

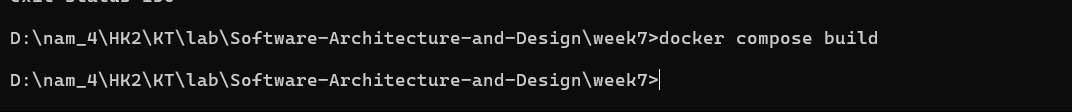




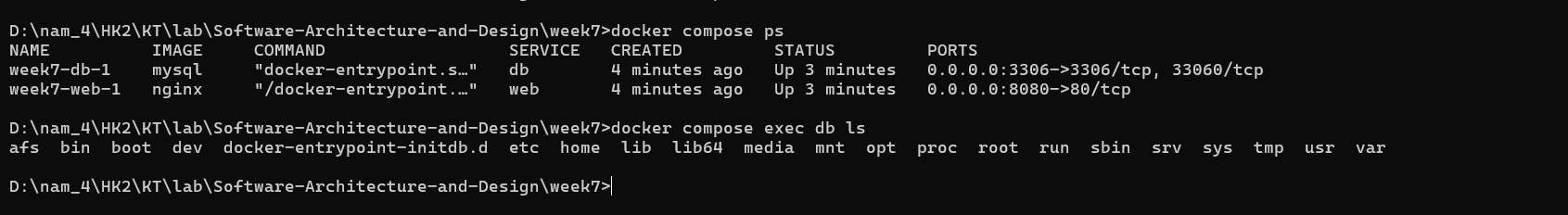
7 docker compose logs -f: Theo dõi log thời gian thực từ các container.



8 docker compose build: Build lại image cho các service từ Dockerfile.

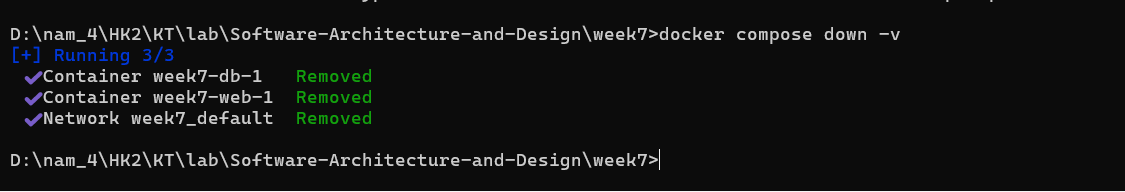


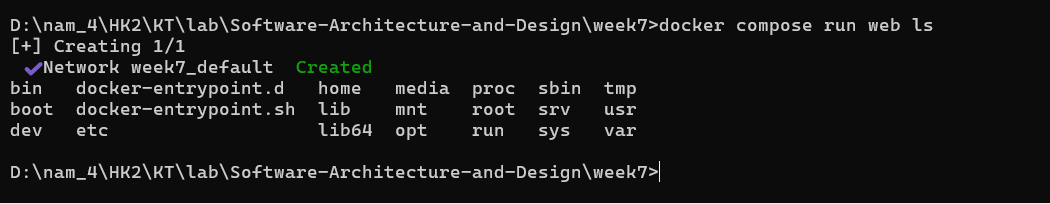
9 docker compose exec <service\_name> <command>: Thực thi một lệnh bên trong container đang chạy (giống docker exec).



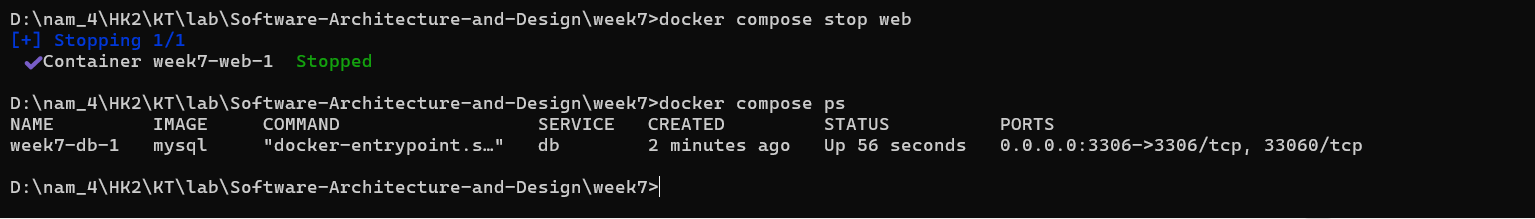
- Một số command thông dụng:

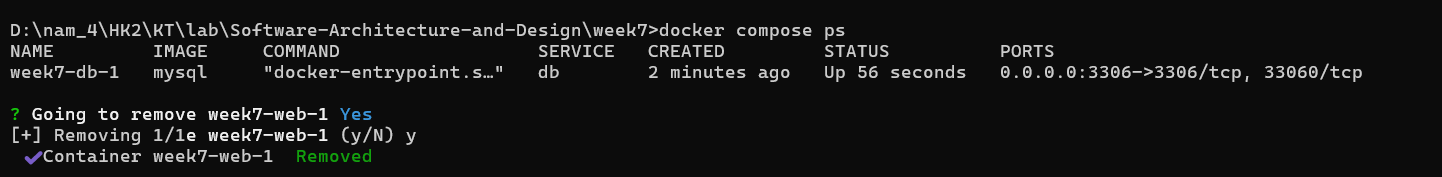
* bash hoặc sh: Mở một terminal shell trong container
* ls: Liệt kê các file và thư mục trong thư mục hiện tại của container
* pwd: Hiển thị đường dẫn thư mục hiện tại trong container
* cat <filename>: Đọc và hiển thị nội dung của một file trong container
* top: Hiển thị danh sách các tiến trình đang chạy trong container
* ps: Hiển thị danh sách các tiến trình trong container
* df -h: Hiển thị thông tin về dung lượng đĩa trong container
* curl: Gửi yêu cầu HTTP từ trong container

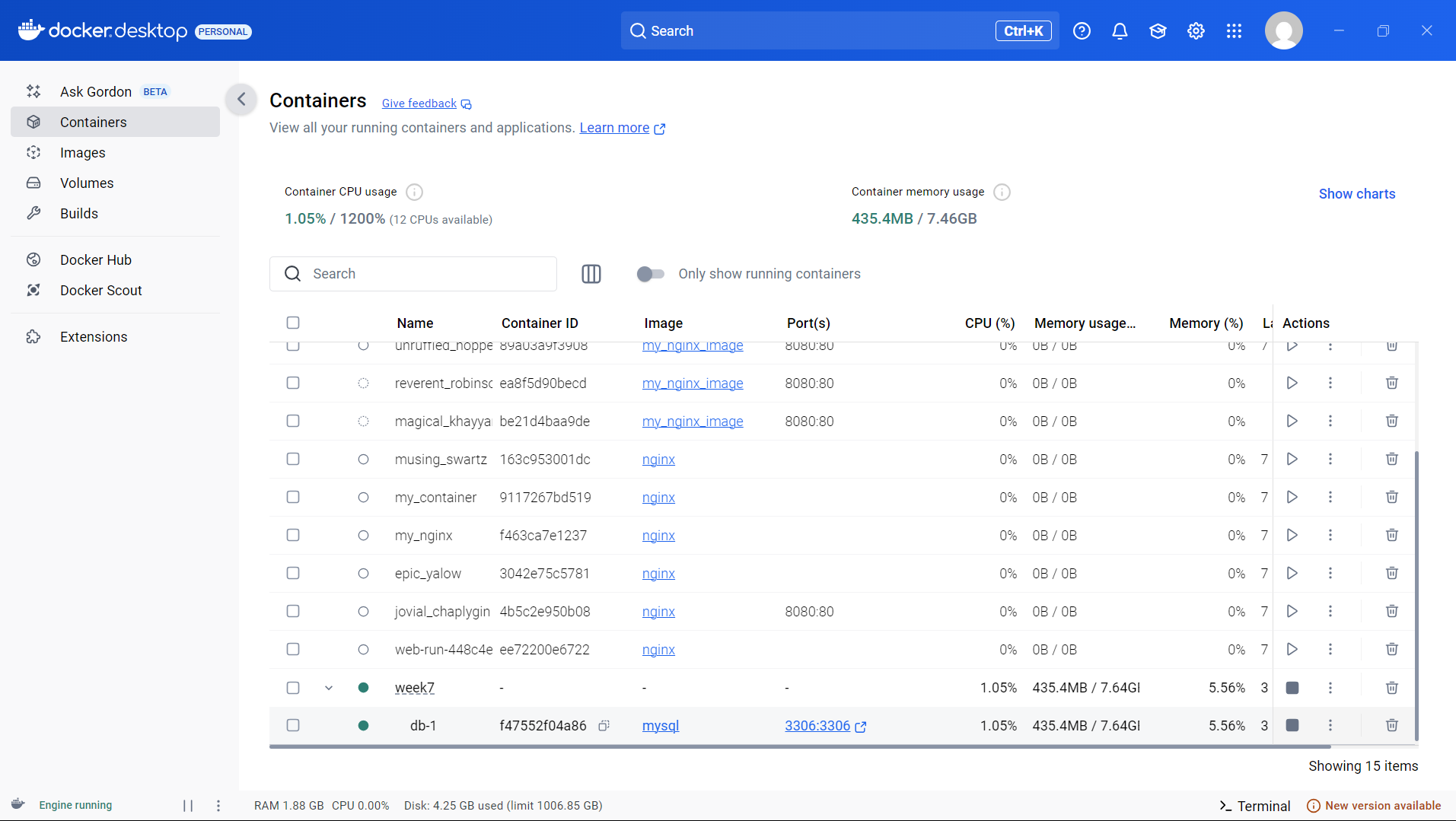
10 docker compose down -v: Dừng container và xóa luôn volume liên quan. 

11 docker compose run <service\_name> <command>: Tạo container mới để chạy một lệnh riêng biệt (khác exec là không cần container đang chạy trước đó). 

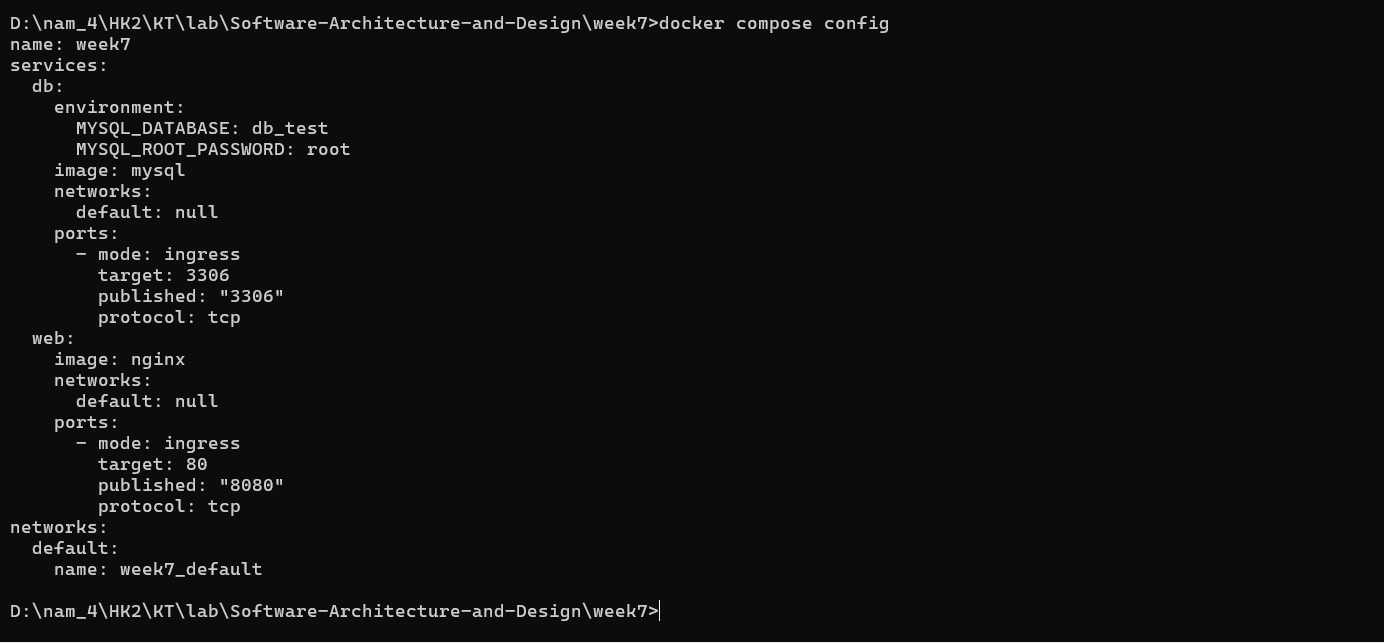
12 docker compose stop <service\_name>: Dừng service cụ thể (không xóa container).



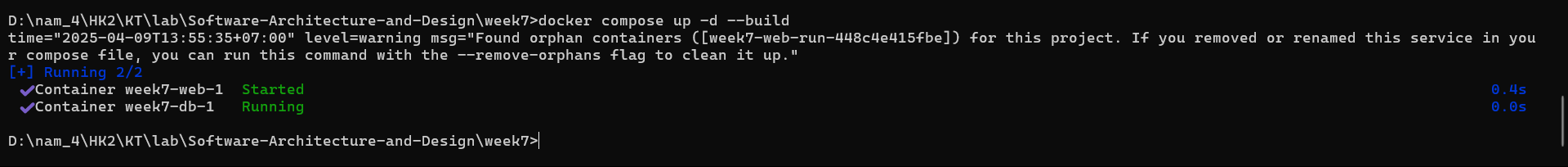
13 docker compose rm <service\_name>: Xóa container của service đã dừng. 



14 docker compose config: Hiển thị cấu hình đã hợp nhất từ docker-compose.yml, bao gồm giá trị mặc định và env.



15 docker compose up -d --build: Build lại image và khởi động service ở chế độ nền.



**Phần 2: Docker Compose file**

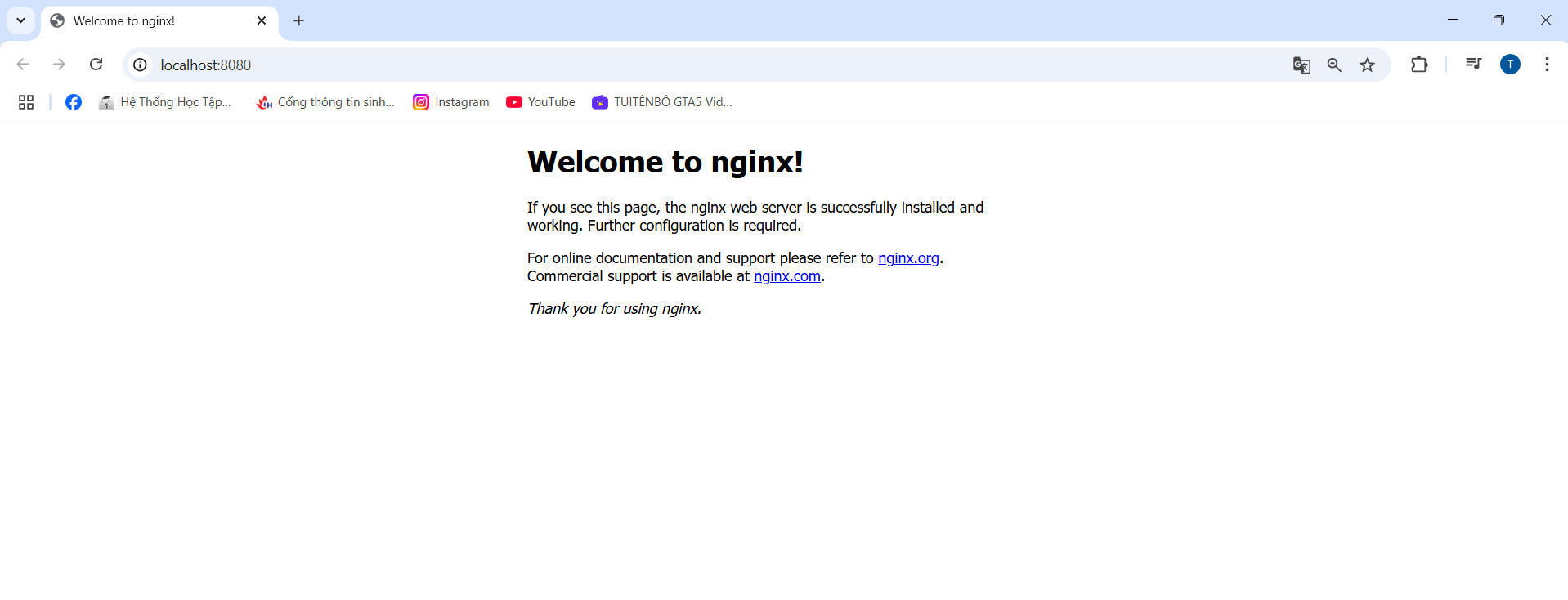
Bài 1: Chạy một container đơn giản với Docker Compose

Yêu cầu:

Tạo một container chạy Nginx bằng Docker Compose.

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

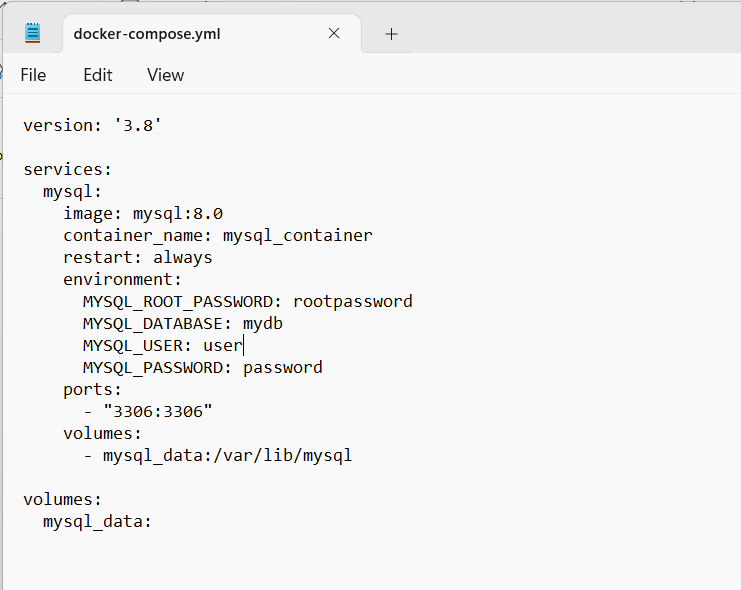
docker compose up -d

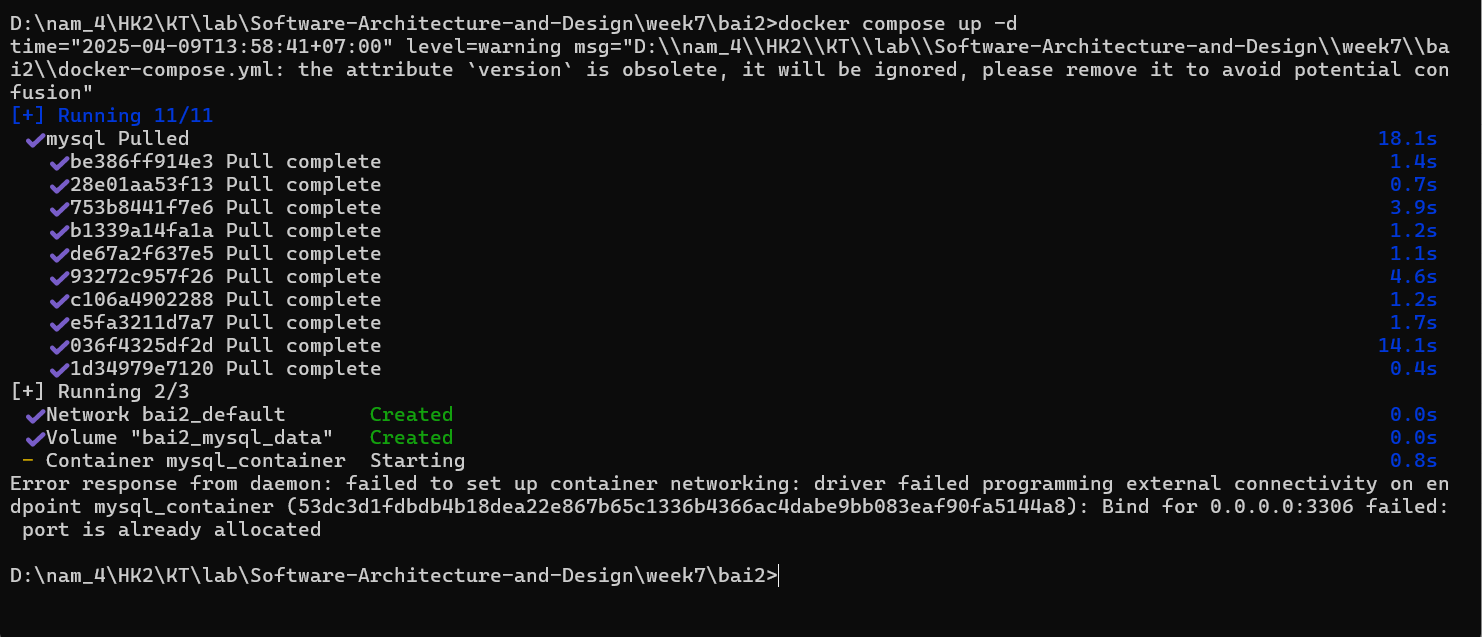


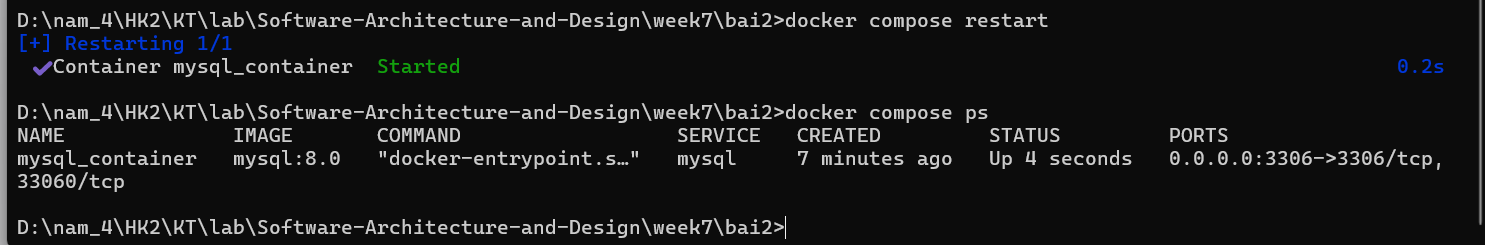
Bài 2: Chạy MySQL với Docker Compose

Yêu cầu:

Tạo một container chạy MySQL phiên bản 8.0.

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





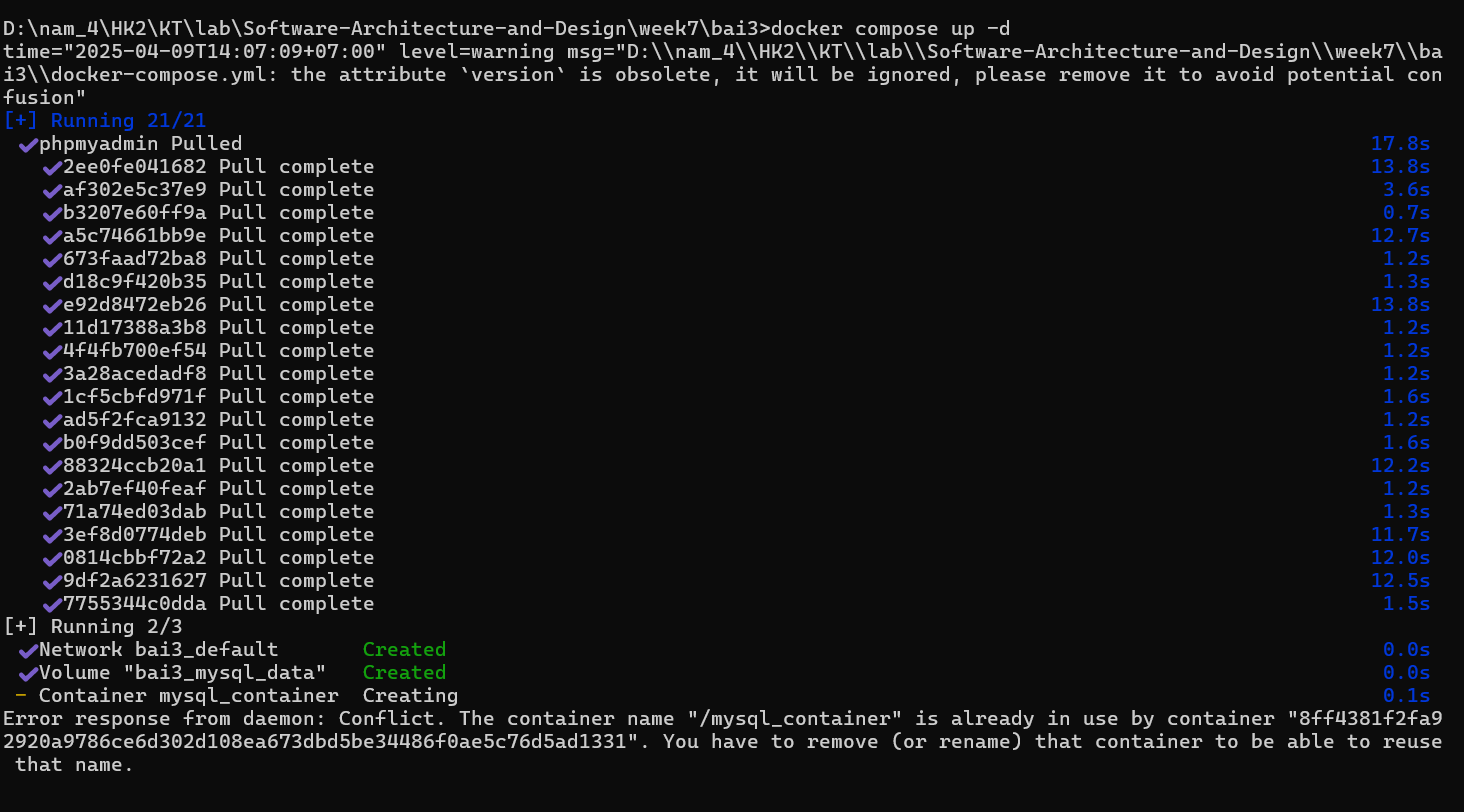
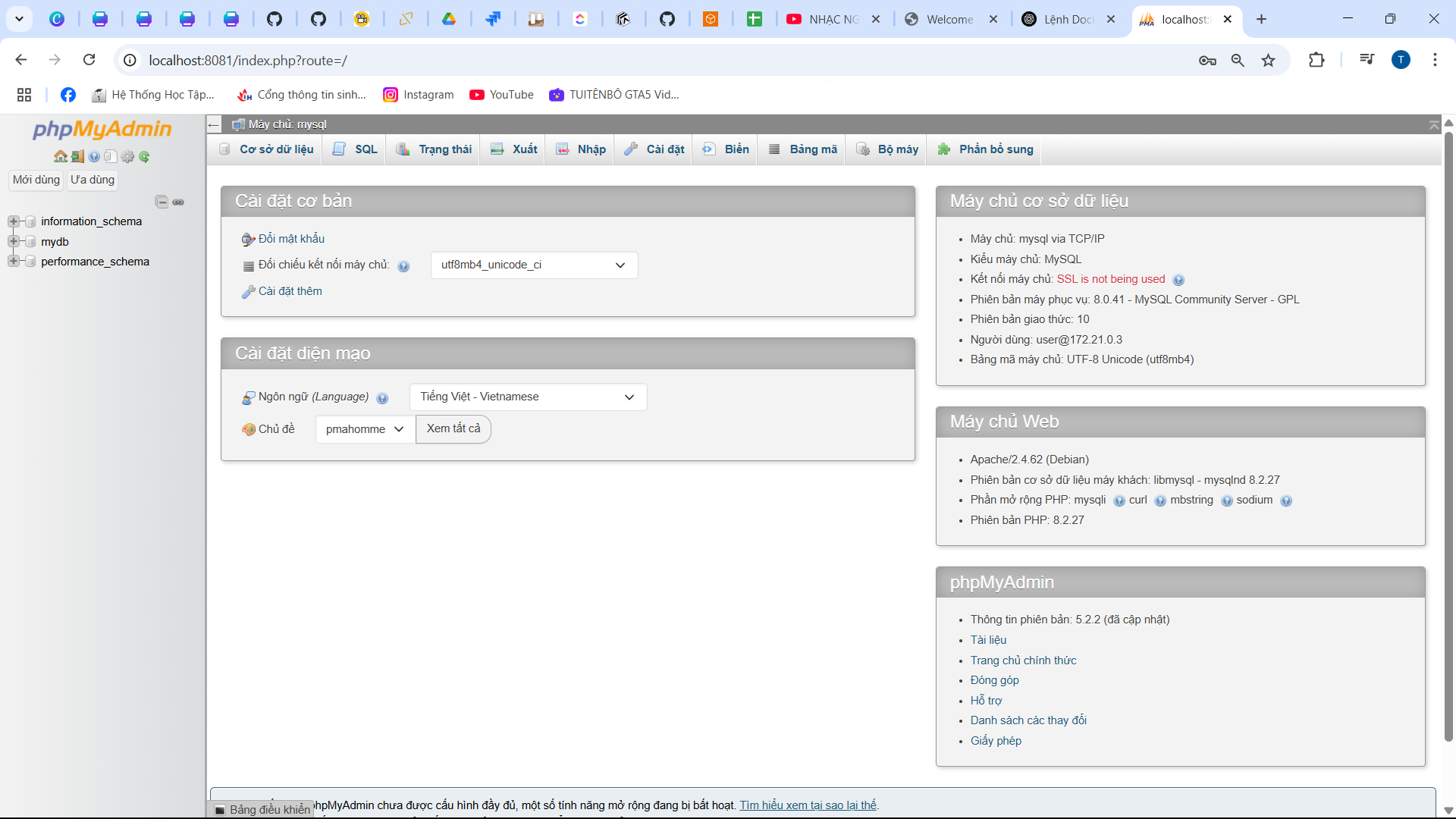
Bài 3: Kết nối MySQL với PHPMyAdmin

Yêu cầu:

Chạy MySQL và PHPMyAdmin với Docker Compose.

PHPMyAdmin chạy trên cổng 8081.

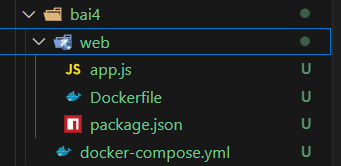


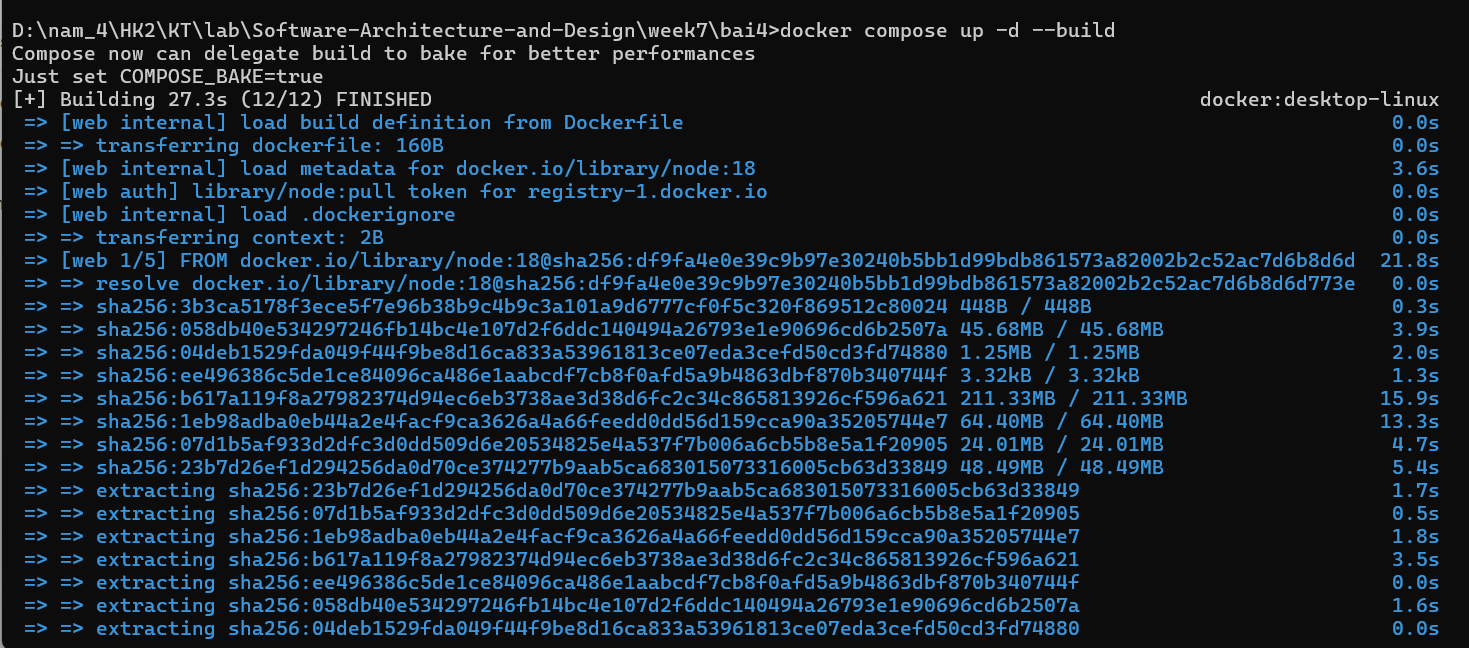
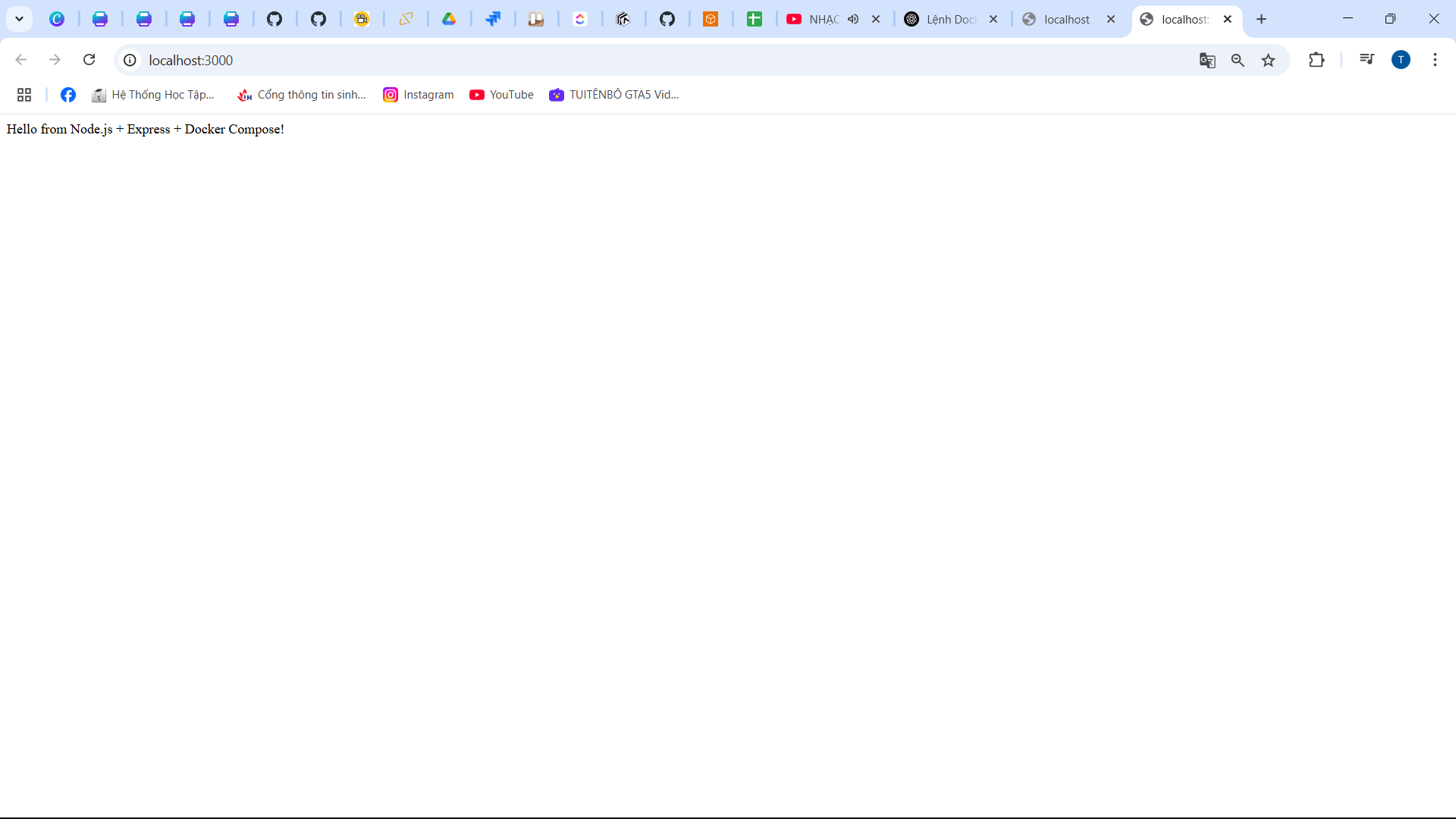
 

Bài 4: Chạy ứng dụng Node.js với Docker Compose

Yêu cầu:

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

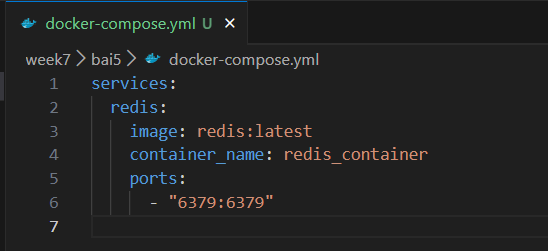


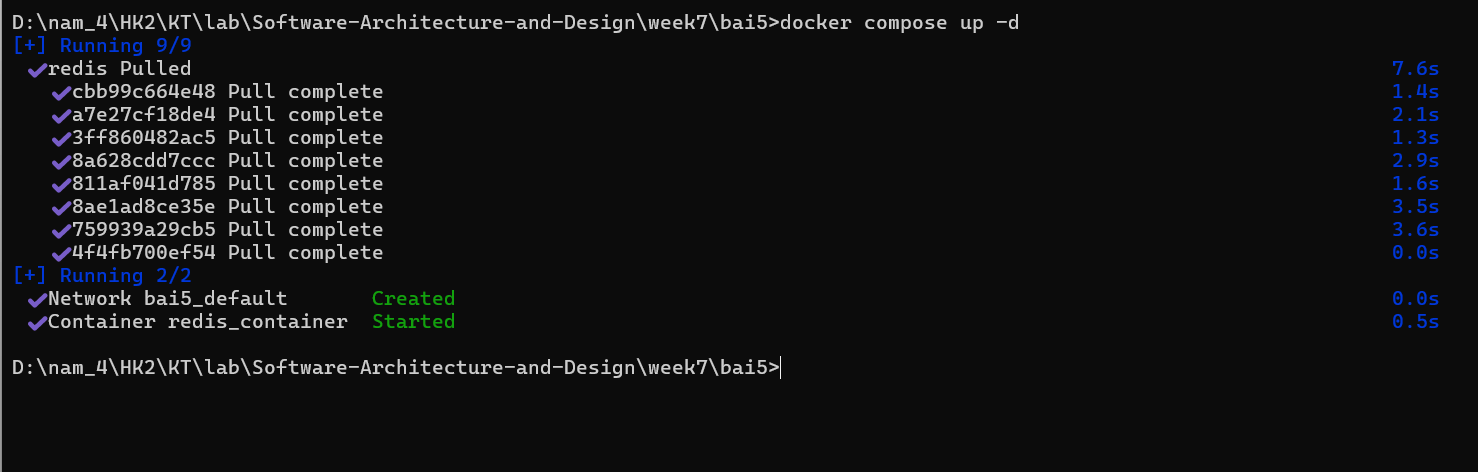
 

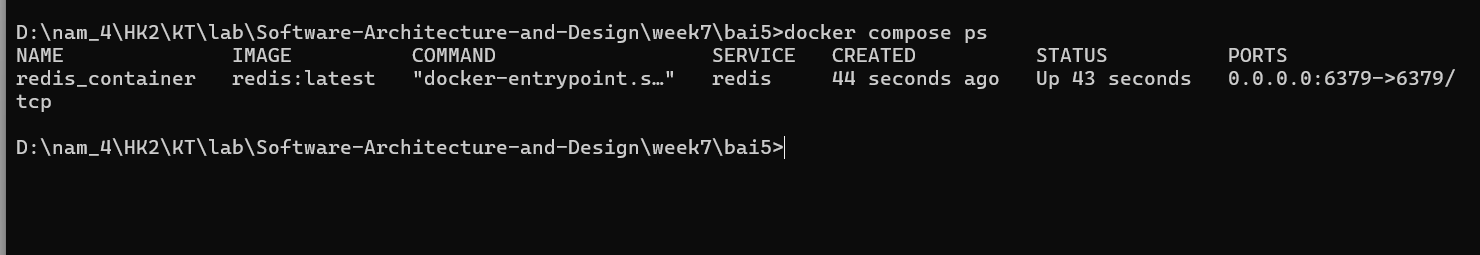
Bài 5: Chạy Redis với Docker Compose

Yêu cầu:

Chạy một container Redis trên cổng 6379.



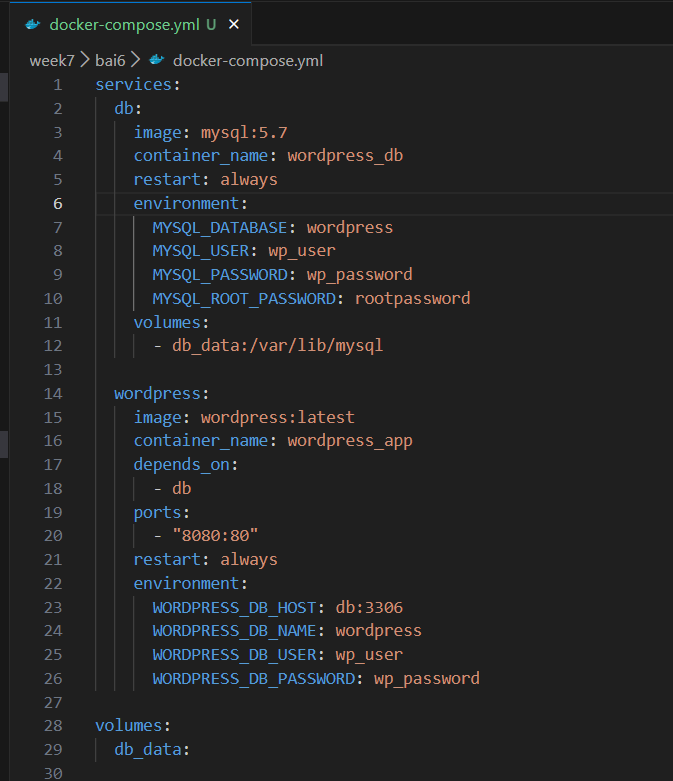


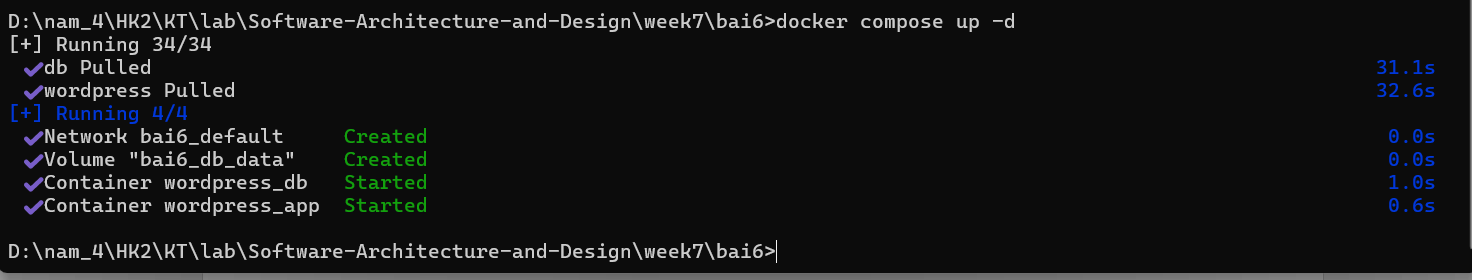


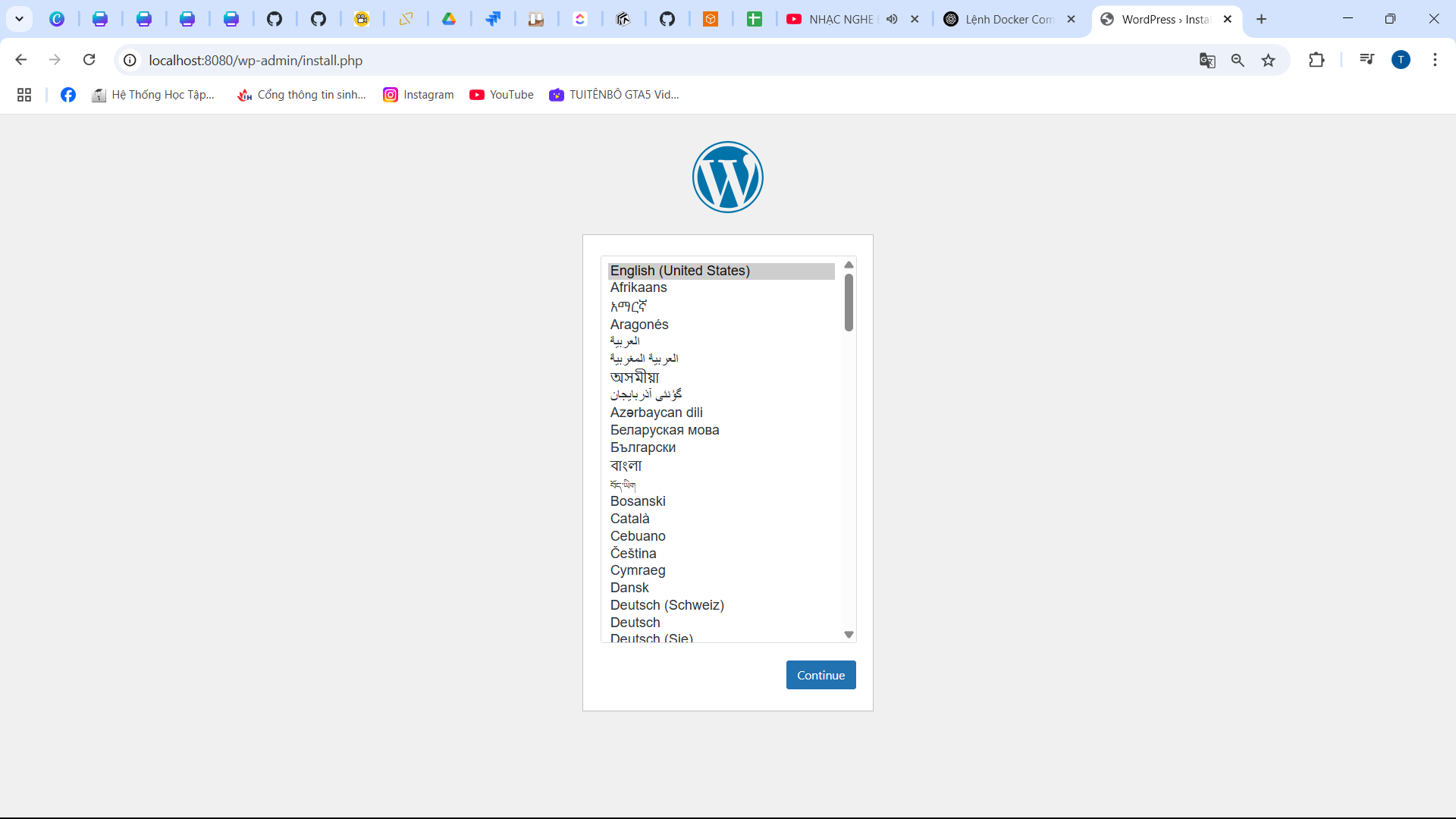
Bài 6: Chạy WordPress với MySQL

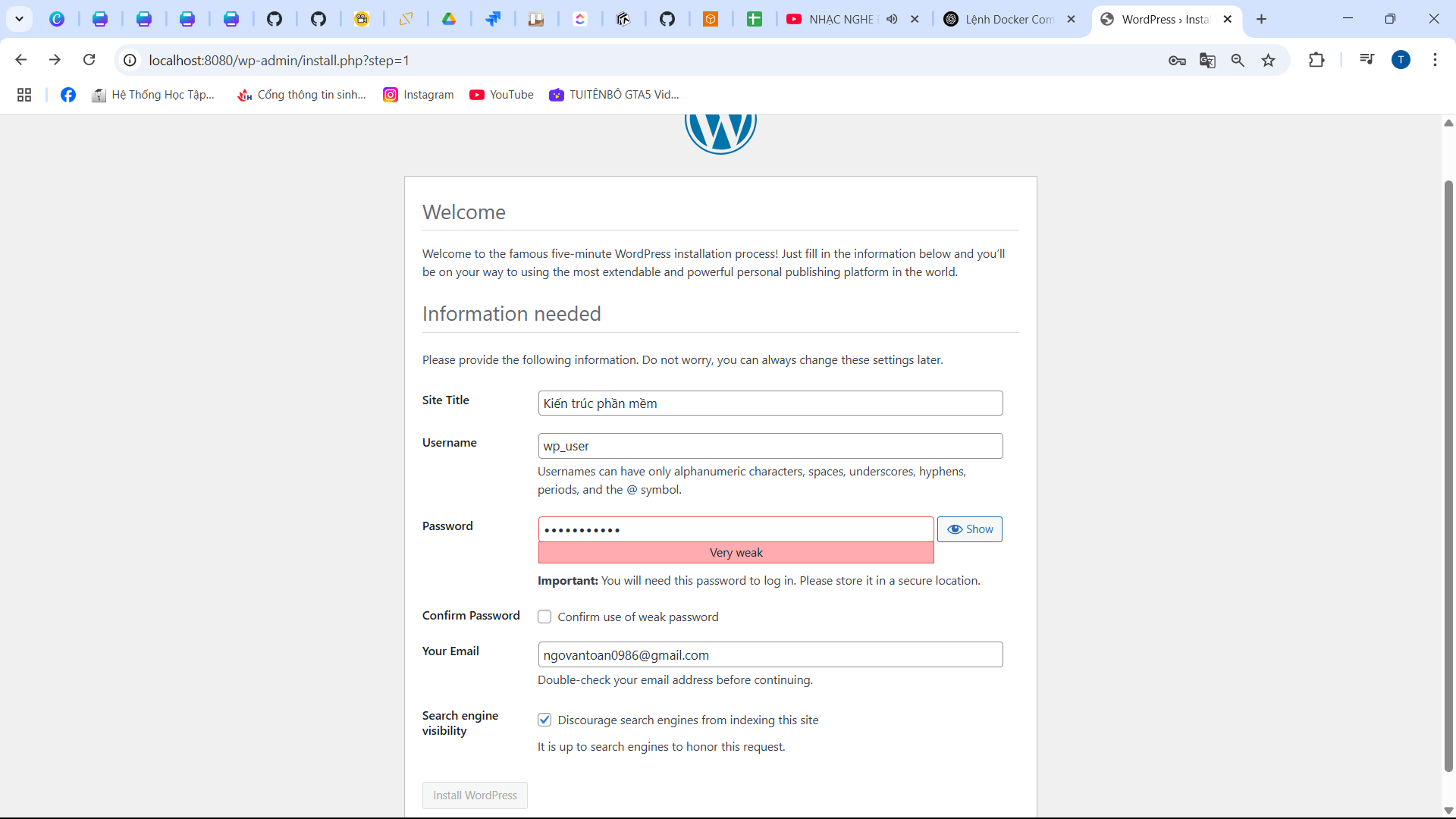
Yêu cầu:

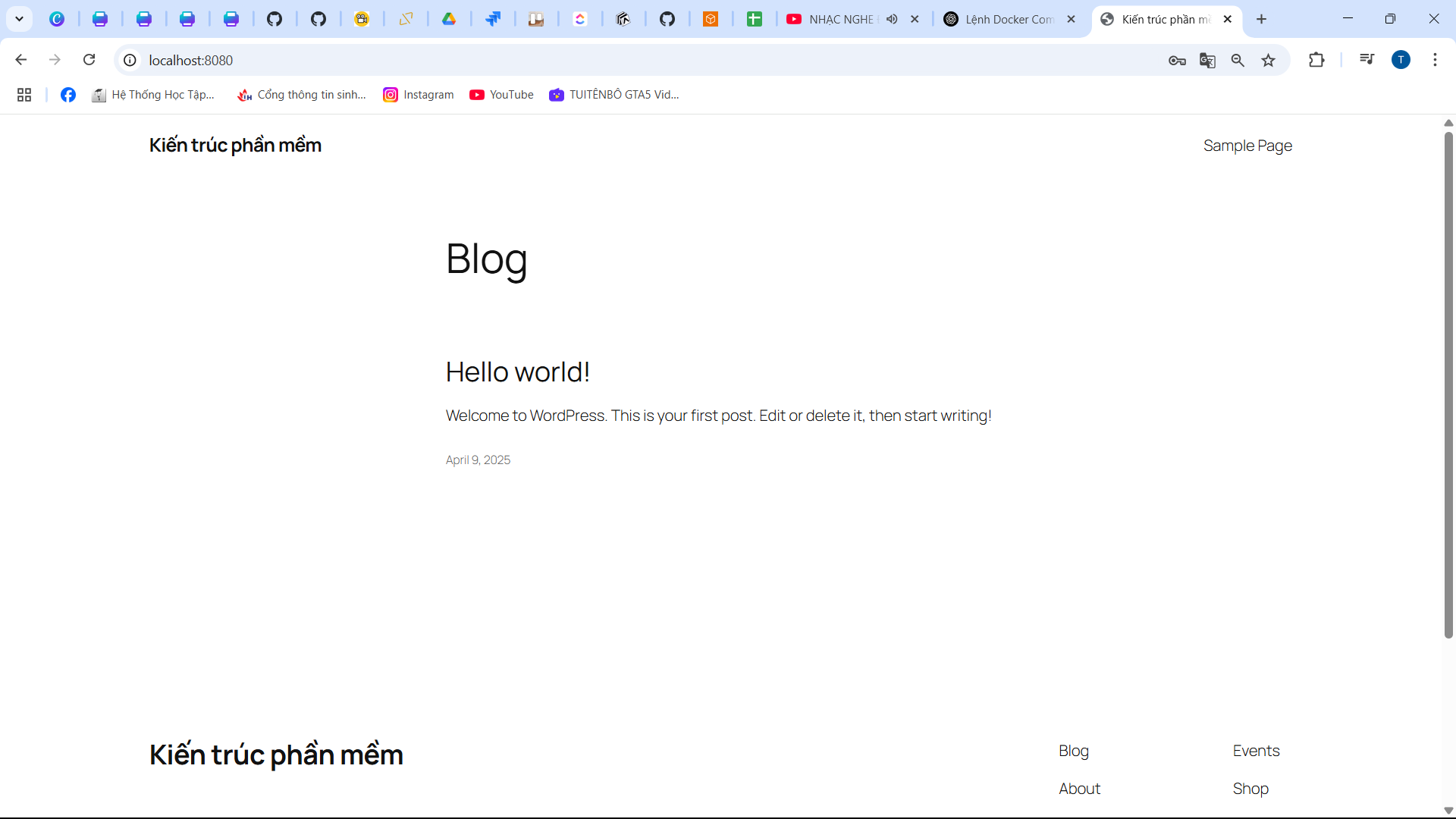
Chạy WordPress với MySQL bằng Docker Compose.







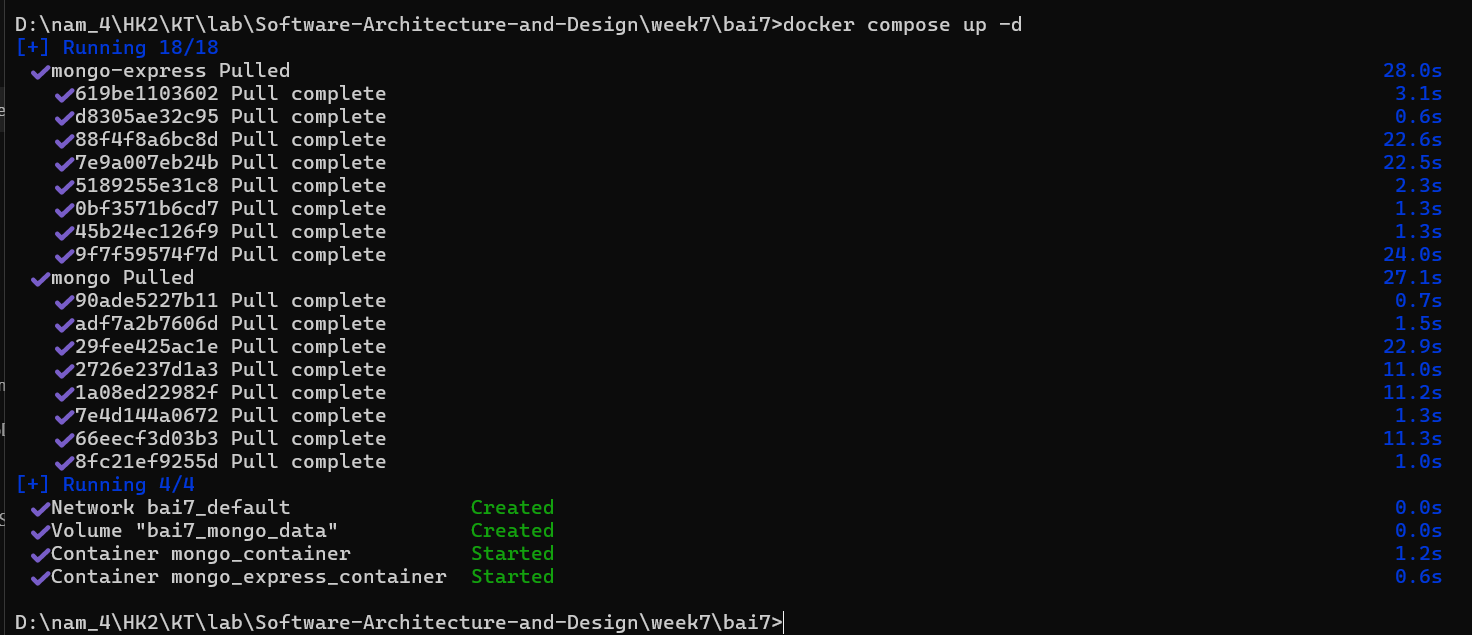
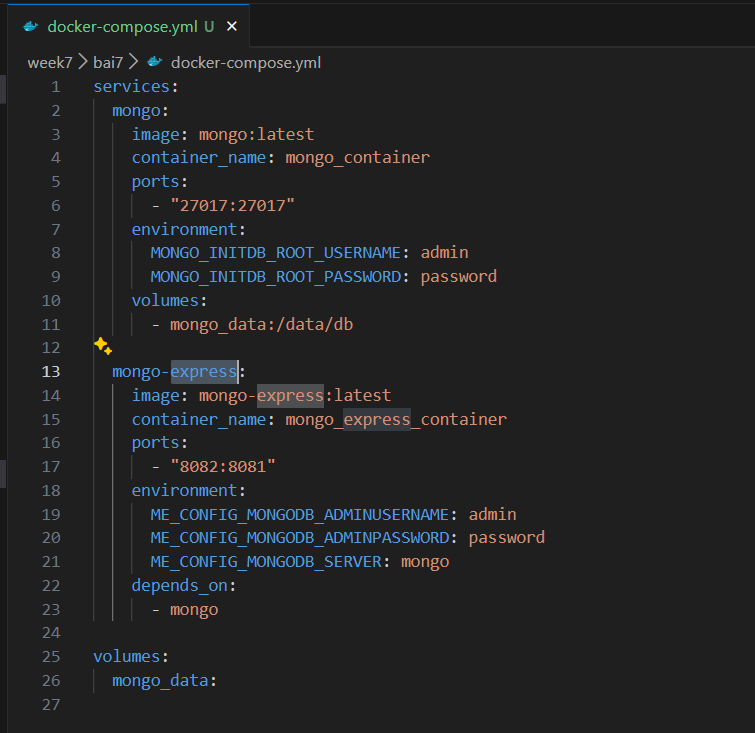


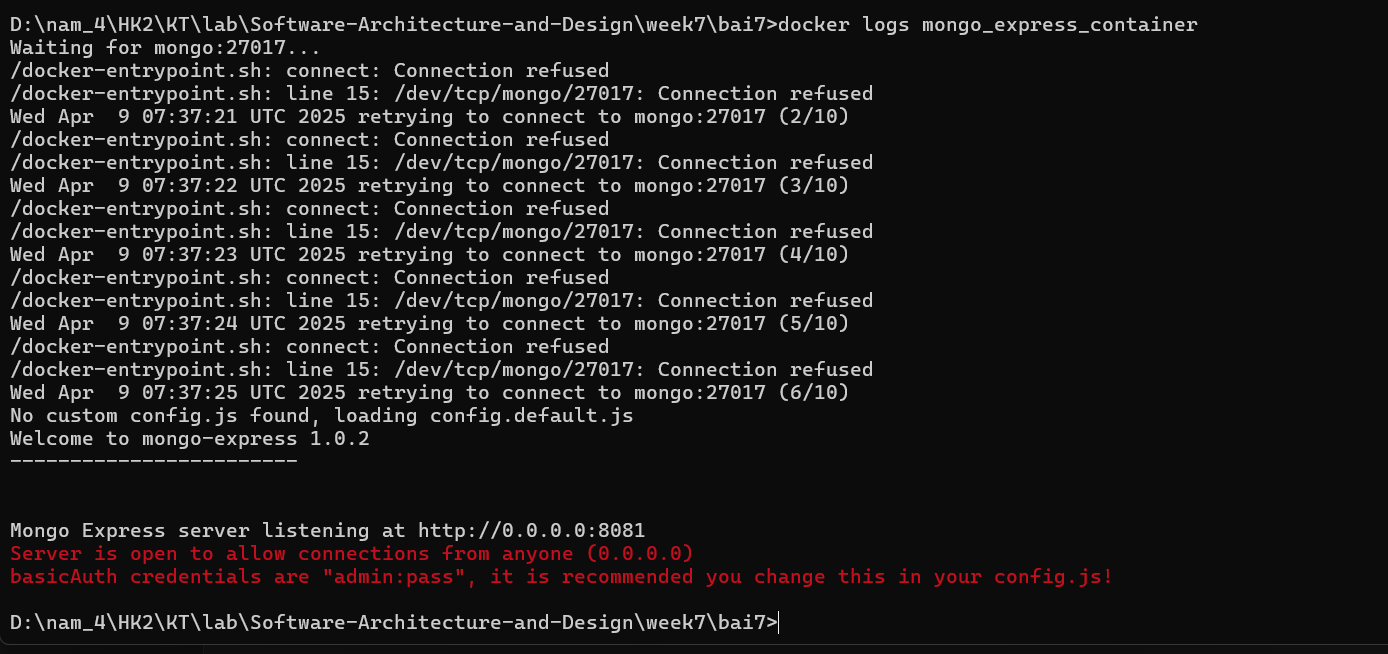


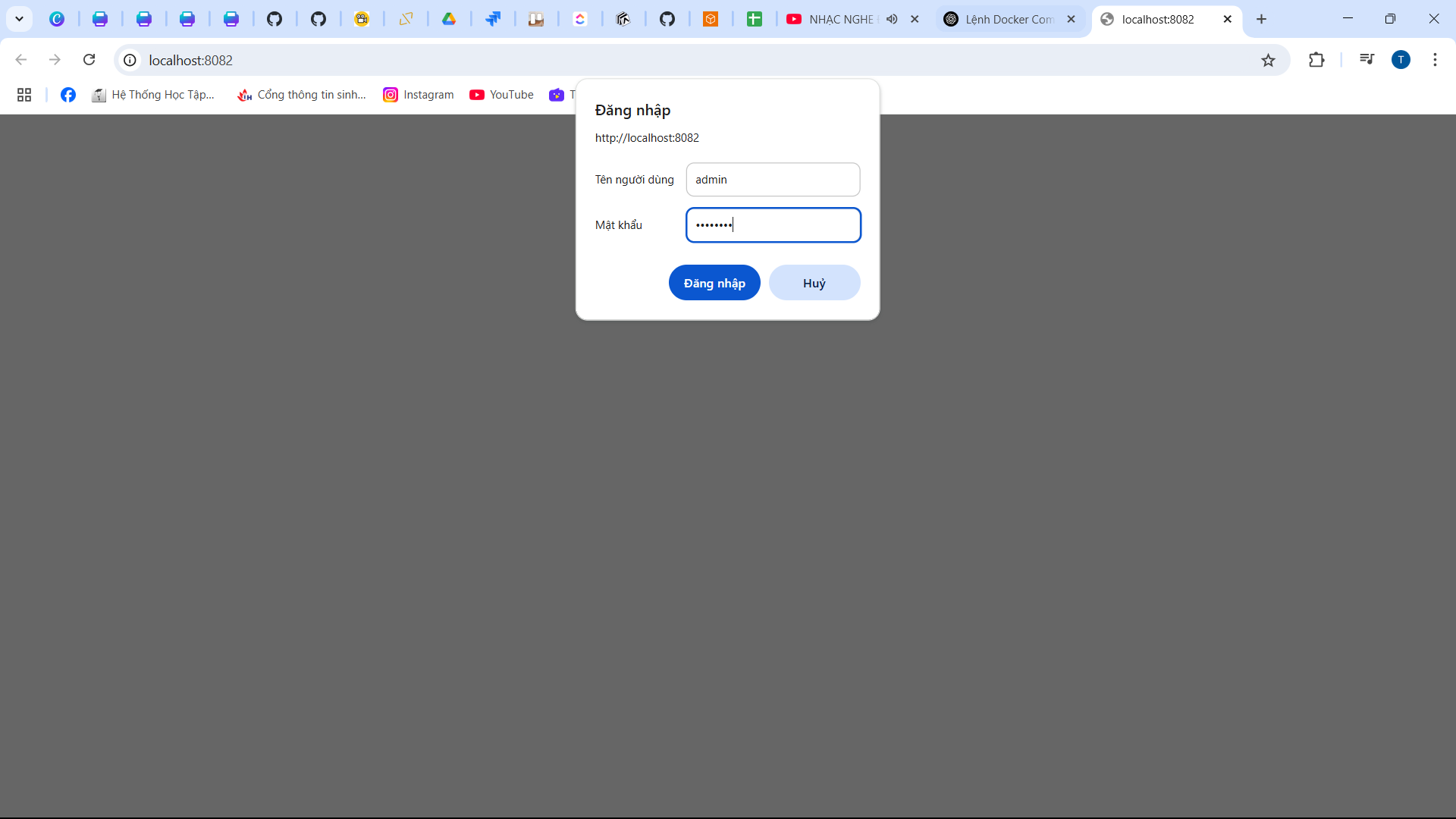
Bài 7: Chạy MongoDB với Docker Compose

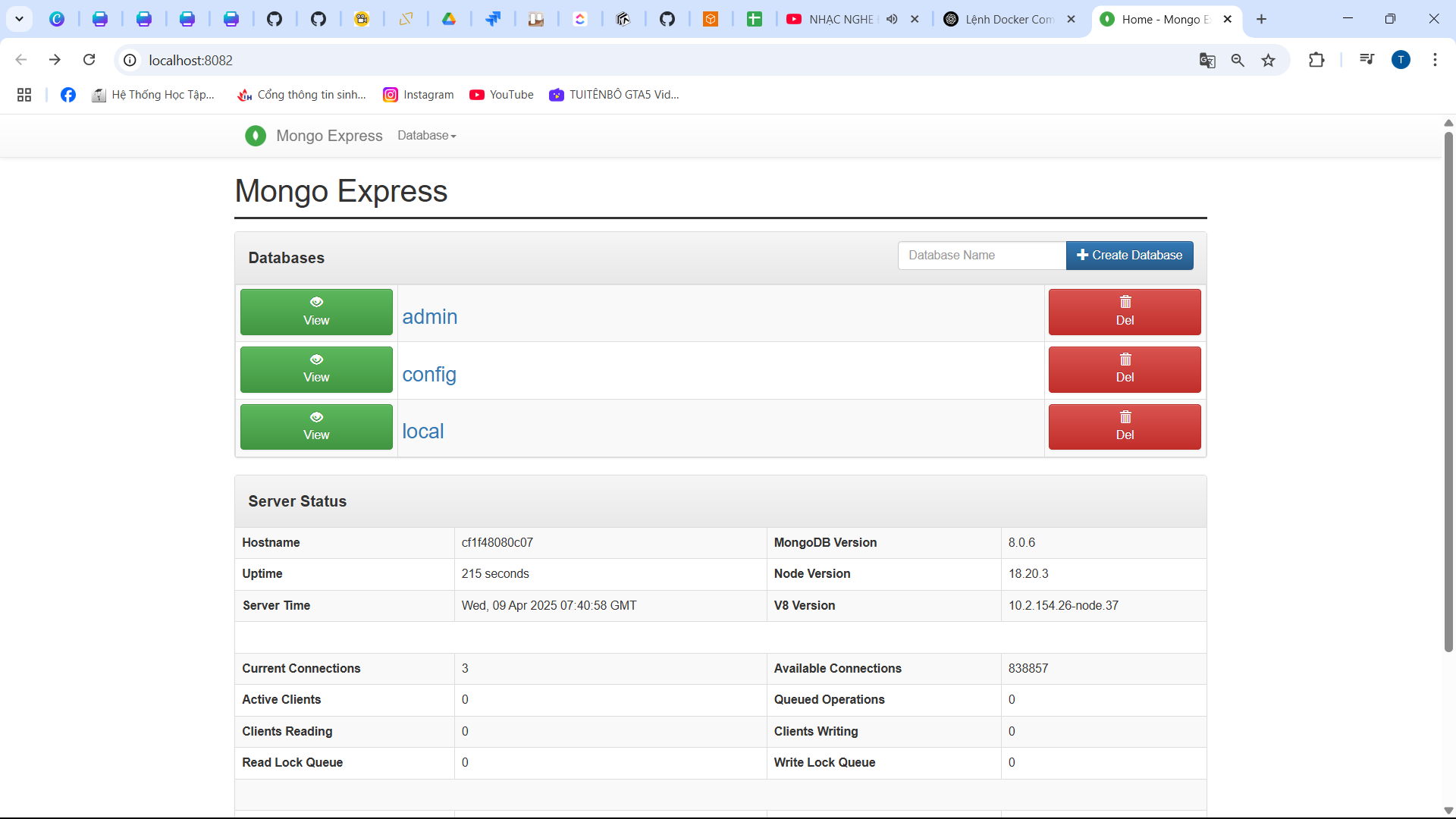
Yêu cầu:

Chạy MongoDB và Mongo Express để quản lý.





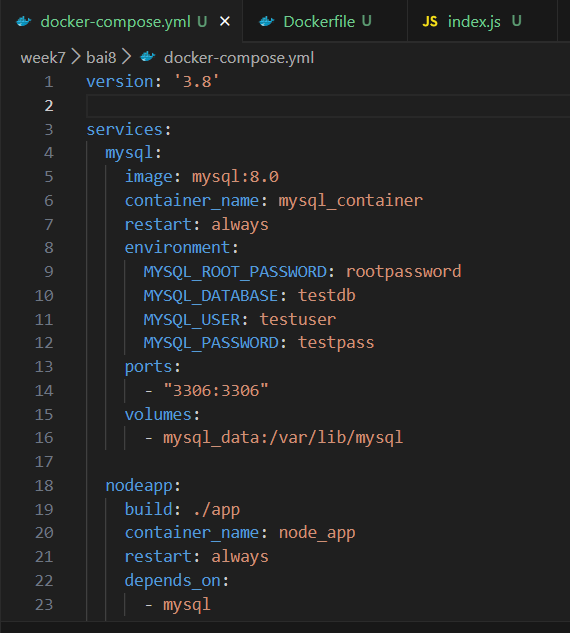


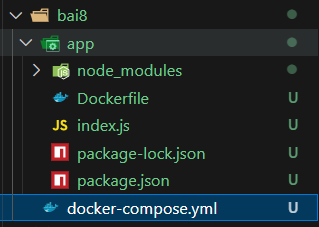


Bài 8: Kết nối nhiều dịch vụ với Docker Compose

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.

Bài 11: Chạy dịch vụ Postgres với Adminer

Yêu cầu:

Chạy PostgreSQL và Adminer (công cụ quản lý database) bằng Docker Compose.

PostgreSQL phải có database tên mydb, user là user, password là password.

Adminer chạy trên cổng 8083

Bài 12: Giám sát container với Prometheus và Grafana

Yêu cầu:

Chạy Prometheus, Grafana và Node Exporter bằng Docker Compose để giám sát hệ thống.

Bài 13: Chạy ứng dụng React với Nginx

Yêu cầu:

Chạy một ứng dụng React và serve nó bằng Nginx.

Bài 14: Cấu hình mạng riêng giữa các container

Yêu cầu:

Chạy 2 container có thể giao tiếp với nhau trong một mạng riêng.

Bài 15: Giới hạn tài nguyên cho container

Yêu cầu:

Giới hạn CPU và RAM cho một container Redis.

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: Multi-tier Voting App

Mục tiêu: Triển khai ứng dụng voting gồm 5 services (Tham khảo từ Docker Docs).

Yêu cầu:

1. Frontend: vote (Python, port 5000).

2. Backend: result (Node.js, port 5001).

3. Redis (lưu tạm vote).

4. Worker (Java) xử lý vote từ Redis sang DB.

5. Postgres (lưu kết quả).

Bài tập 6: CI/CD Pipeline với Docker Compose

Mục tiêu: Mô phỏng pipeline dev/test bằng Docker Compose.

Yêu cầu:

1. Service app (Python/Node.js) với code được mount từ host (development mode).

2. Service tests chạy unit tests khi code thay đổi (sử dụng volumes + entrypoint).

3. Service nginx (production mode) dùng image build sẵn từ app.

Gợi ý:

1. Dùng 2 file compose khác nhau (docker-compose-dev.yml và docker-compose-prod.yml).

2. Sử dụng docker-compose -f <file> up để chọn môi trường.

Bài tập 7: 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.

Bài tập 8: Django + Celery + Redis

Mục tiêu: Triển khai Django với Celery worker và Redis làm message broker.

Yêu cầu:

1. Django app (port 8000).

2. Celery worker chạy song song.

3. Redis service cho task queue.

Bài tập 9: Nextcloud với MariaDB + Redis Caching

Mục tiêu: Triển khai Nextcloud (self-hosted cloud) với MariaDB và Redis.

Yêu cầu:

1. Nextcloud (port 80).

2. MariaDB (volume cho dữ liệu).

3. Redis cache.

Bài tập 10: Traefik as Reverse Proxy

Mục tiêu: Dùng Traefik để định tuyến request đến các service (Flask, WordPress, etc.).

Yêu cầu:

1. Cấu hình Traefik với Docker provider.

2. Đặt labels cho services để Traefik nhận diện.

Tips khi làm bài tập:

1. Luôn dùng docker-compose down -v để xóa volumes khi test lại.

2. Kiểm tra log bằng docker-compose logs <service>.

3. Sử dụng docker-compose config để validate file YAML.