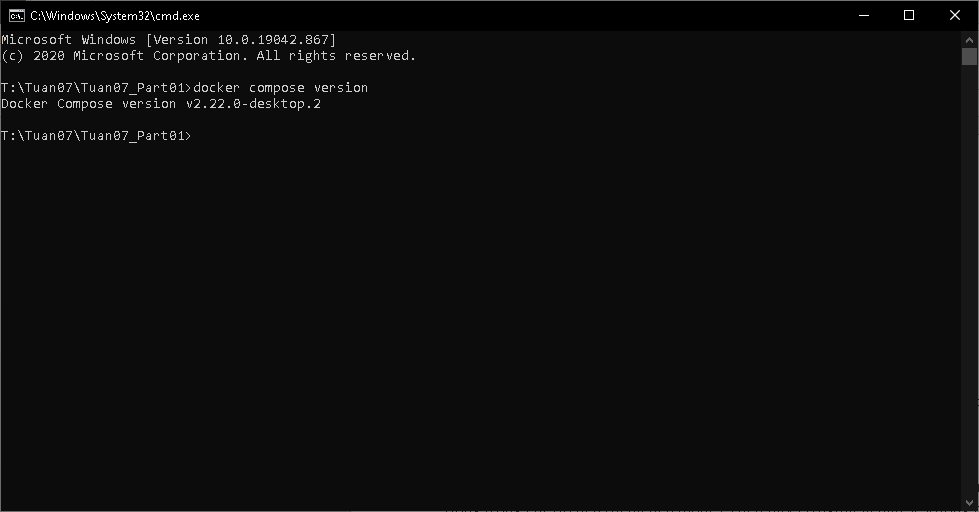
**Phần 1: Một số lệnh Docker Compose cơ bản**

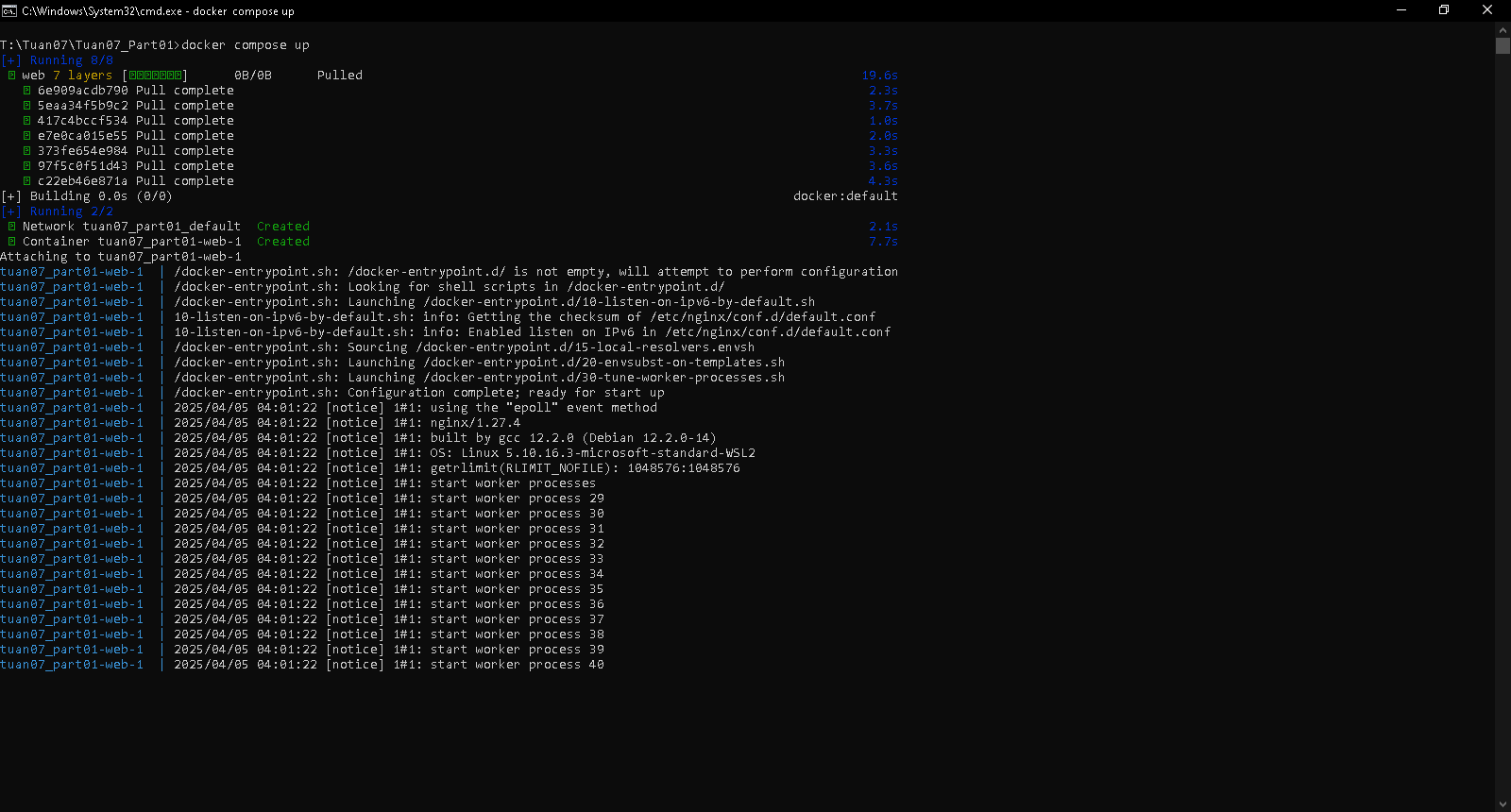
1. docker compose version

Hiển thị phiên bản của Docker Compose đang được cài đặt



1. docker compose up

Khởi động các container trong Docker Compose, xây dựng lại bất kỳ image nào nếu cần và khởi động dịch vụ từ tệp docker-compose.yml



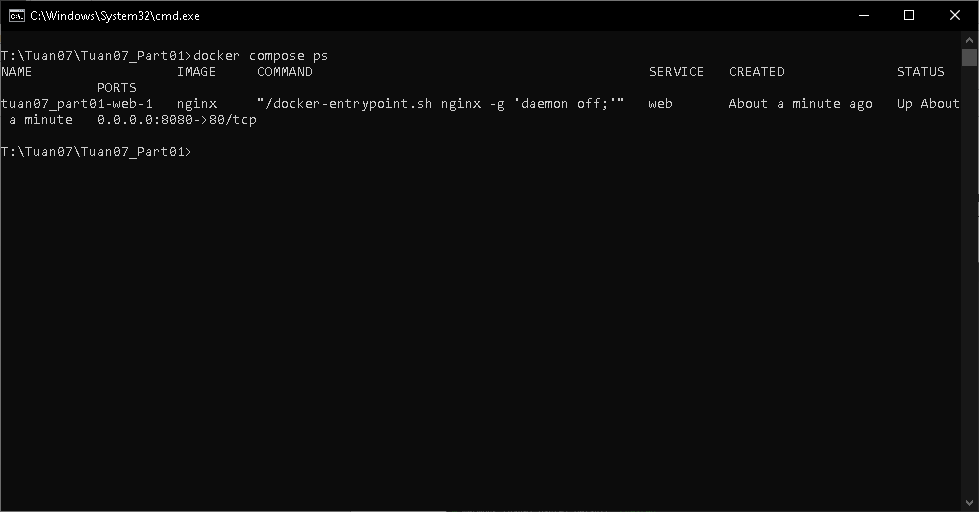
1. docker compose up –d

Tương tự như docker compose up, nhưng với tham số -d, các container sẽ được khởi động trong chế độ nền (detached mode). Điều này có nghĩa là bạn sẽ không nhìn thấy output của các container trong terminal.



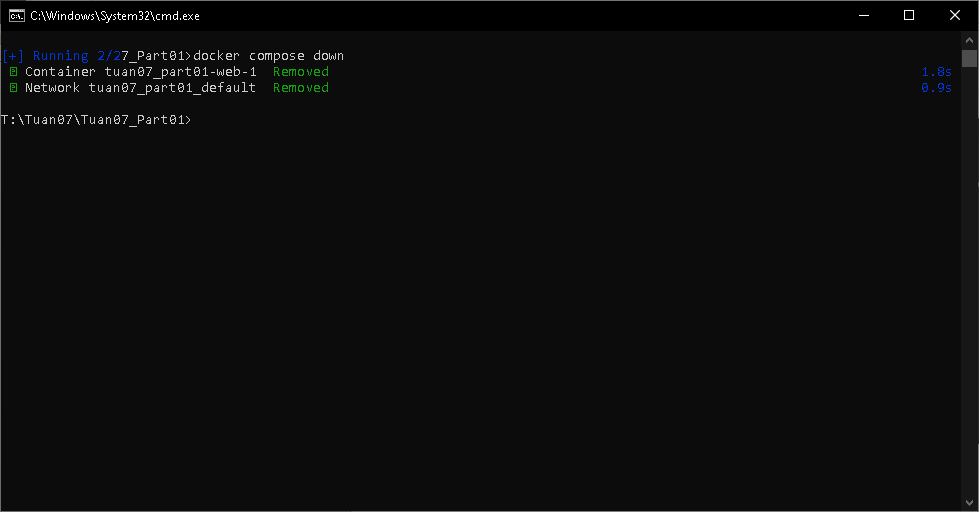
1. docker compose ps

Liệt kê các container đang chạy mà Docker Compose quản lý. Nó sẽ cho bạn thấy thông tin như ID container, tên dịch vụ, trạng thái, và cổng mở.



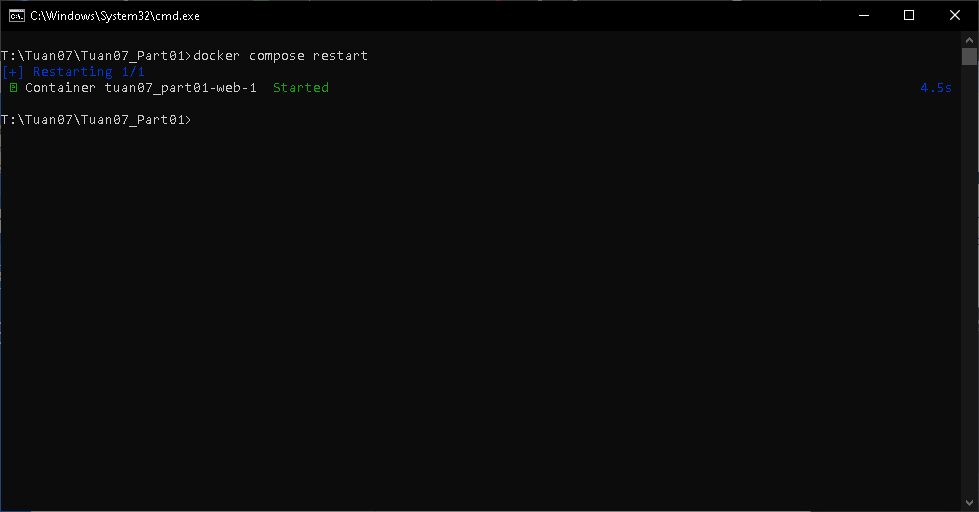
1. docker compose down

Dừng và xóa tất cả các container, network, và volume được Docker Compose tạo ra. Đây là cách để dọn dẹp mọi thứ khi bạn không cần các dịch vụ đang chạy nữa.



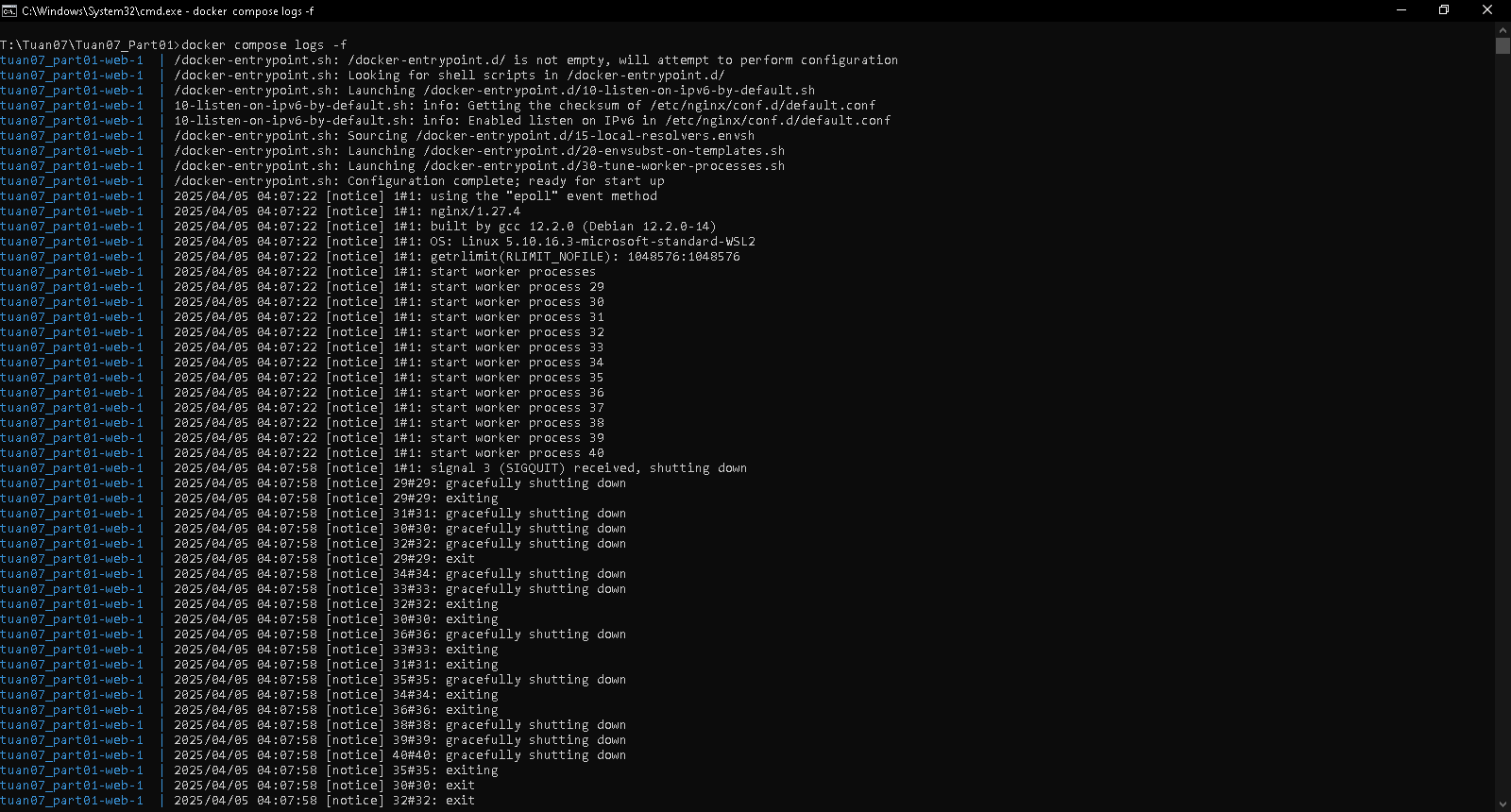
1. docker compose restart

Khởi động lại các container của Docker Compose mà không cần phải dừng chúng trước. Thích hợp khi bạn cần áp dụng thay đổi mà không cần phải xóa container hiện tại.



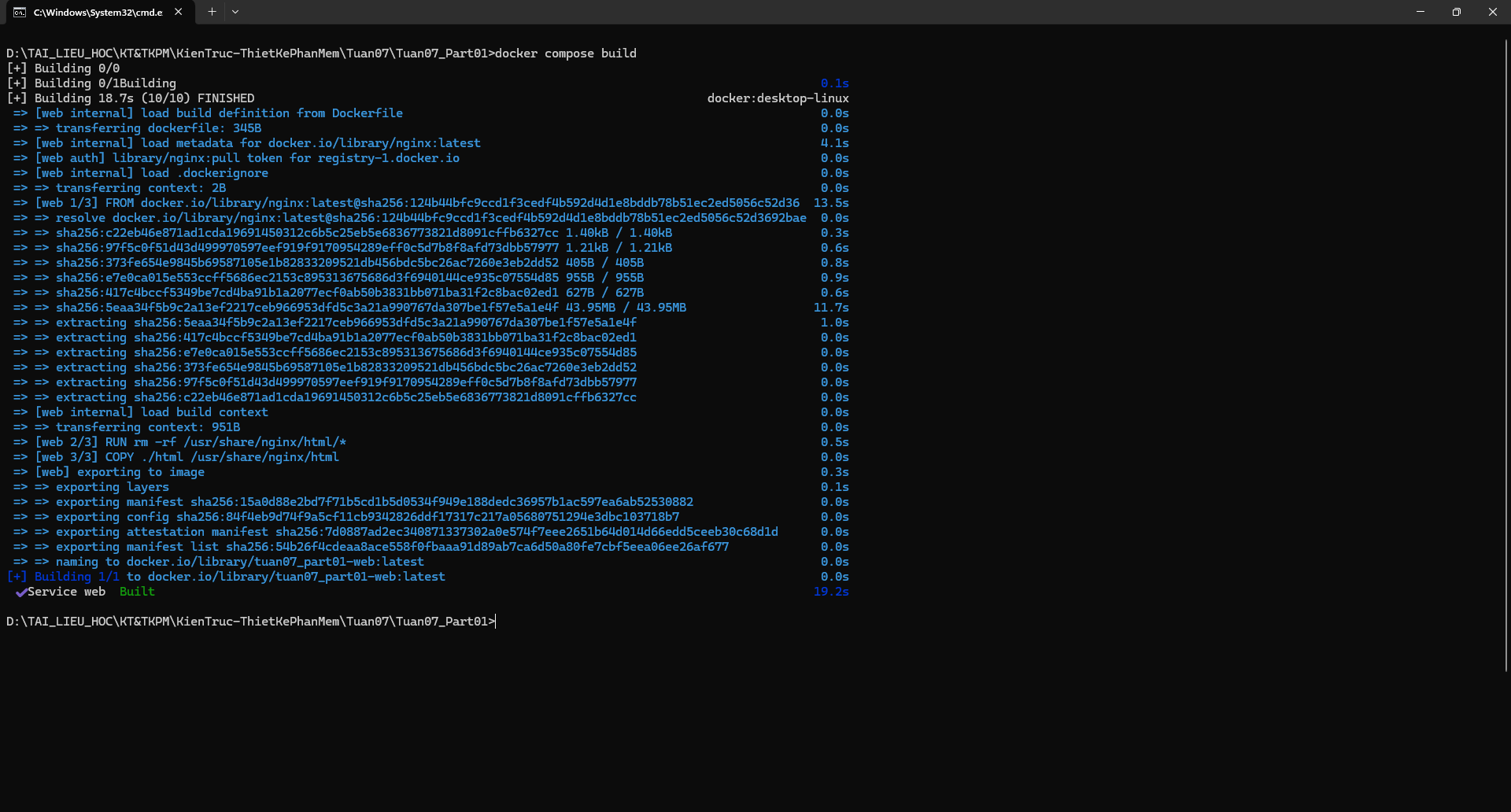
1. docker compose logs –f

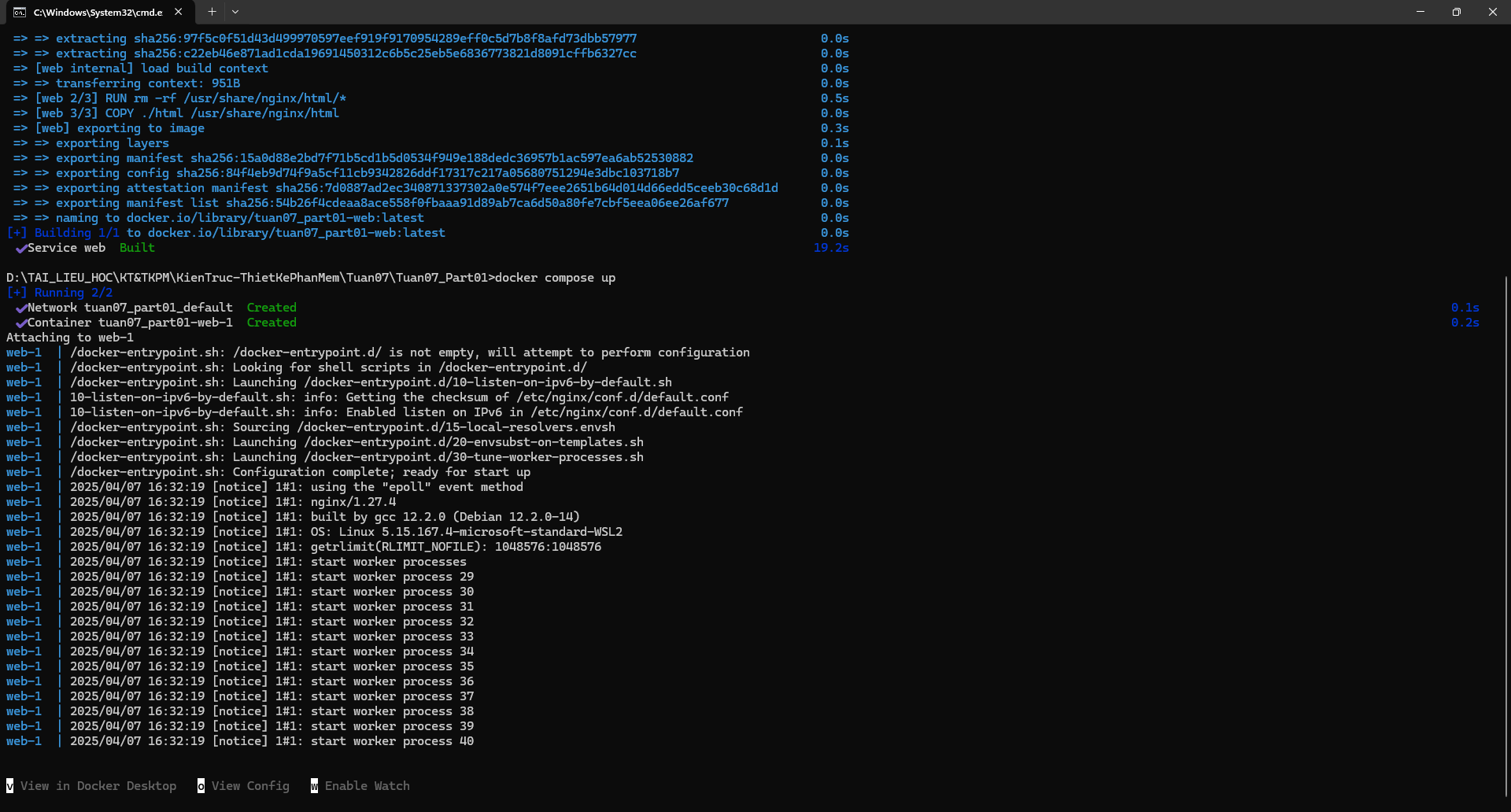
Hiển thị log của các container đang chạy. Tham số -f khiến lệnh này tiếp tục theo dõi log theo thời gian thực, tương tự như lệnh tail -f.

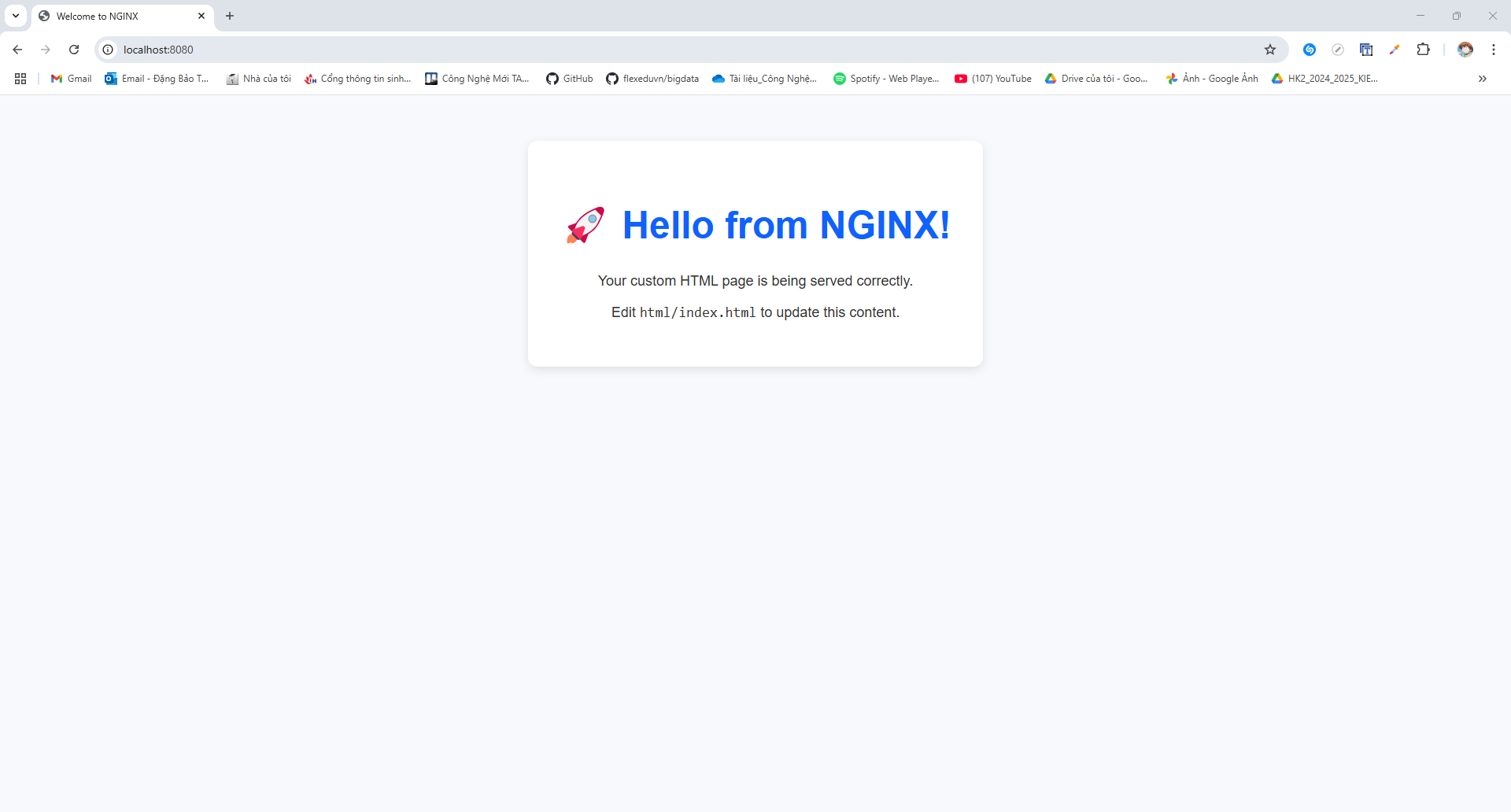


1. docker compose build

Xây dựng lại các image từ Dockerfile được chỉ định trong tệp docker-compose.yml. Lệnh này cần thiết nếu có sự thay đổi trong các Dockerfile hoặc cấu hình dịch vụ.





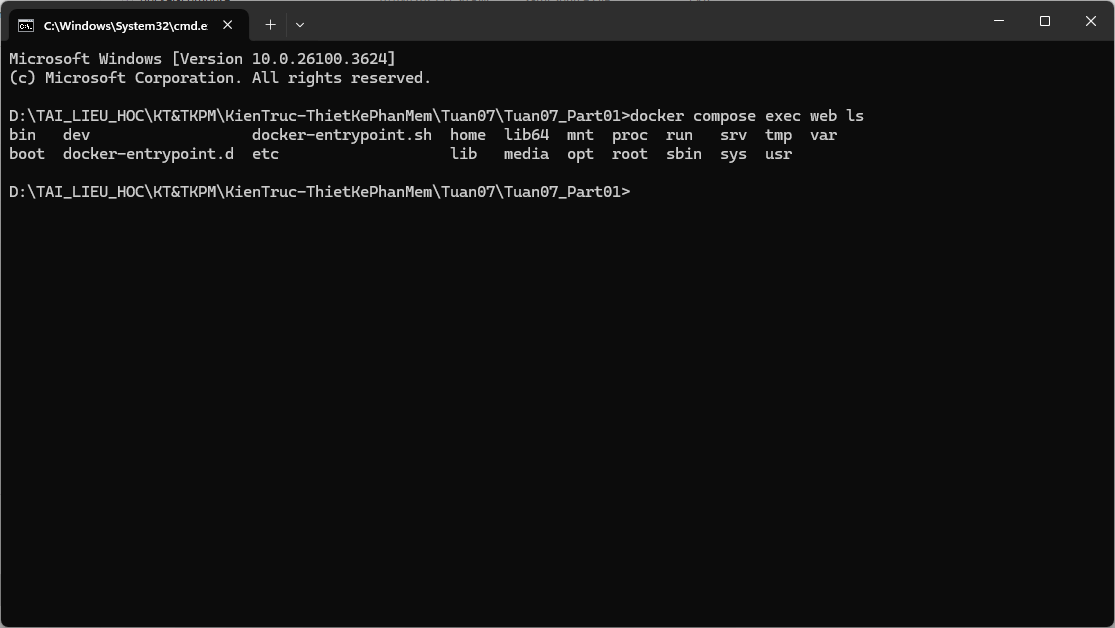


1. docker compose exec <service\_name> <command>

Chạy một lệnh trong một container đang chạy của một dịch vụ cụ thể. <service\_name> là tên của dịch vụ bạn muốn chạy lệnh, và <command> là lệnh bạn muốn thực thi.

Chạy lệnh ls trong container của service web:

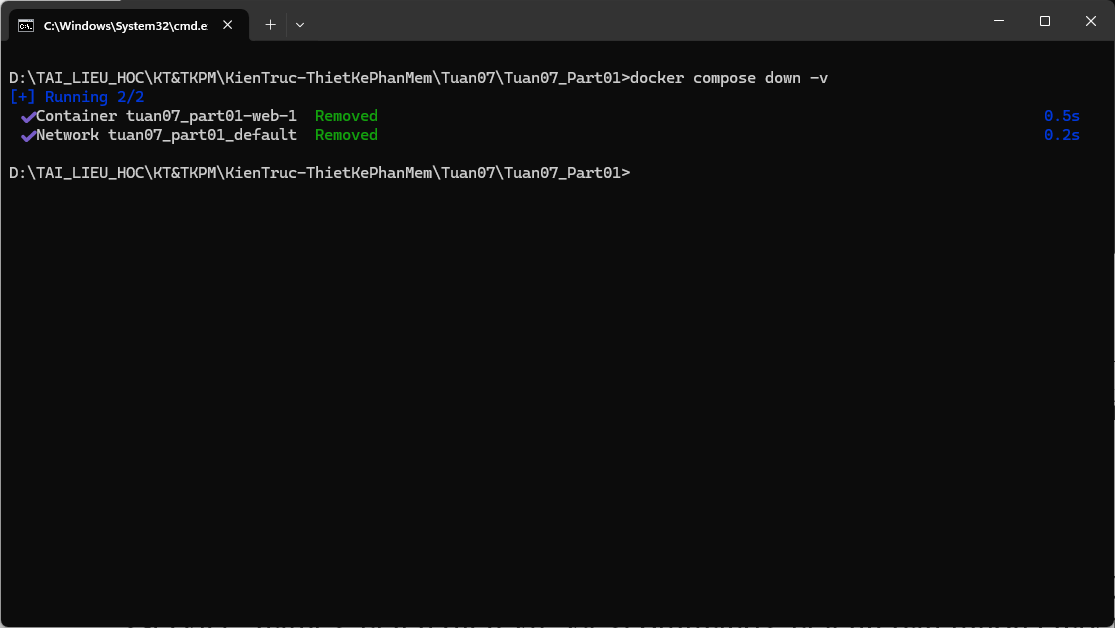
*docker compose exec web ls*



1. docker compose down –v

Tương tự như docker compose down, nhưng với tham số -v, nó sẽ xóa luôn các volumes được Docker Compose tạo ra. Dùng khi bạn muốn dọn dẹp hoàn toàn tất cả, bao gồm cả dữ liệu lưu trữ.

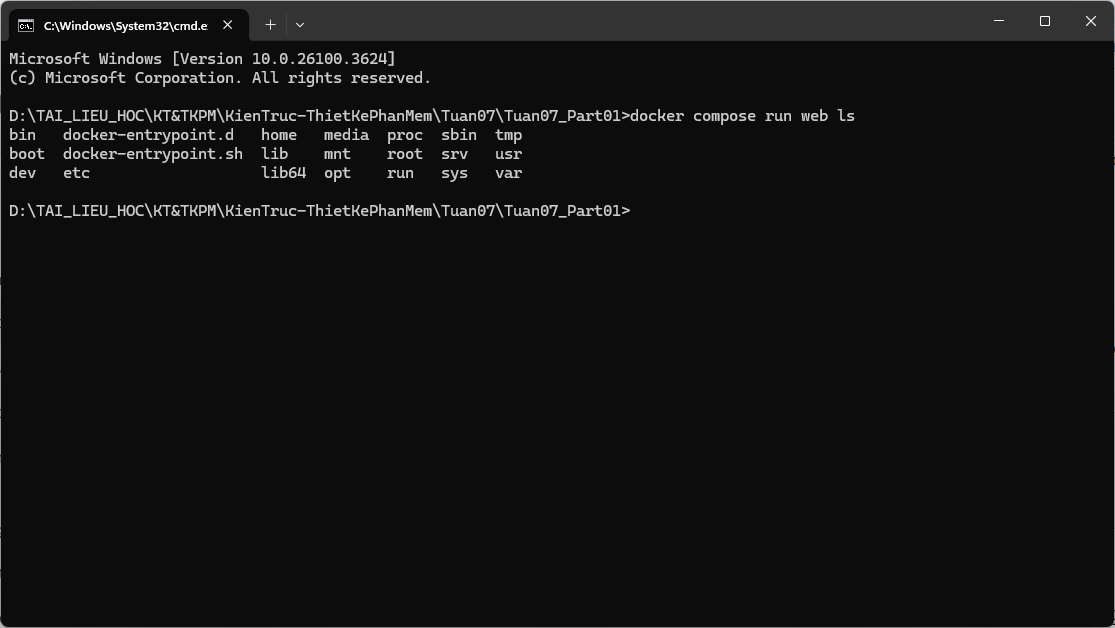
Tắt toàn bộ các service và xóa luôn volumes:



1. docker compose run <service\_name> <command>

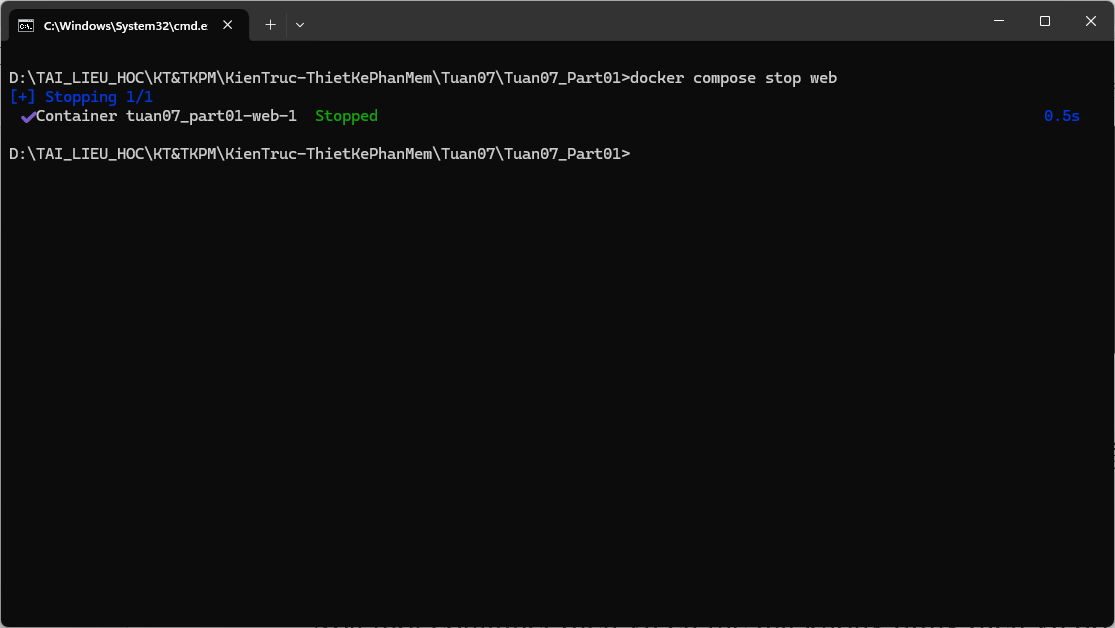
Chạy một lệnh trong một container mới, không phải container đang chạy. Tham số <service\_name> là tên dịch vụ, và <command> là lệnh bạn muốn chạy trong container đó.

Chạy một lệnh ls trong container của dịch vụ web:



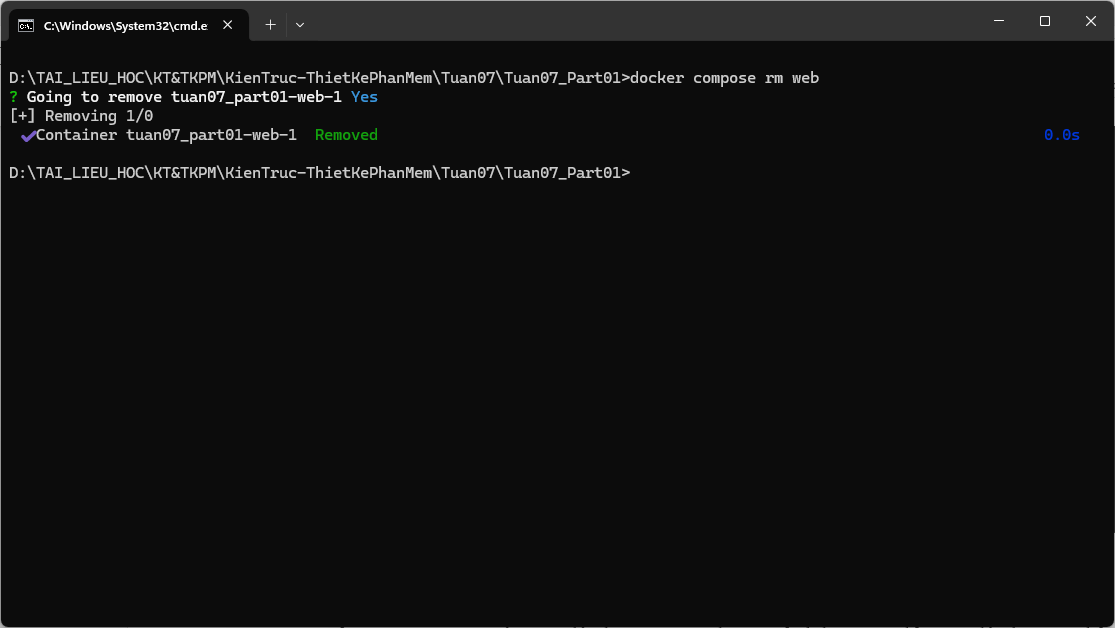
1. docker compose stop <service\_name>

Dừng một dịch vụ cụ thể mà không xóa container. Dịch vụ vẫn có thể được khởi động lại sau này mà không cần phải xây dựng lại.



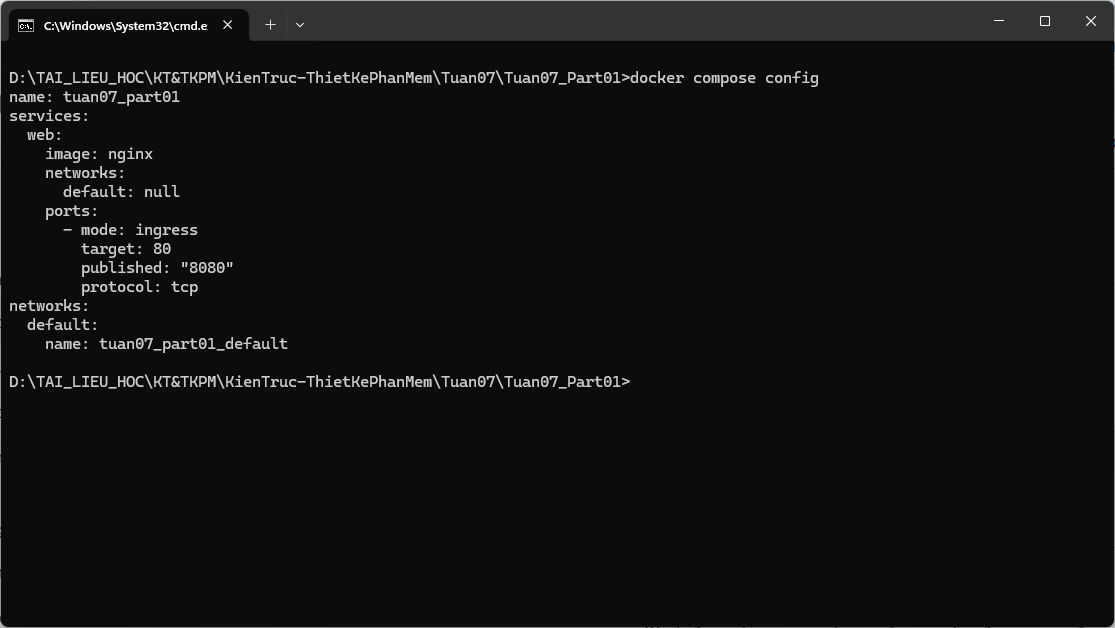
1. docker compose rm <service\_name>

Xóa một container dịch vụ cụ thể mà không dừng dịch vụ đó. Điều này có thể hữu ích khi bạn muốn dọn dẹp các container cũ nhưng vẫn giữ các container khác.



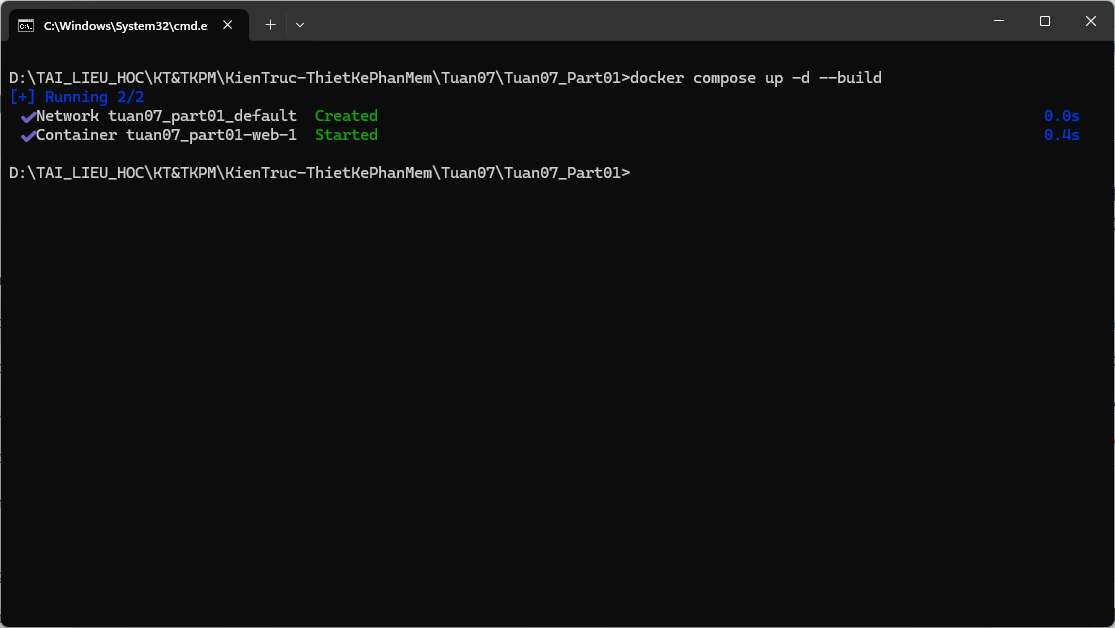
1. docker compose config

Kiểm tra cấu hình của tệp docker-compose.yml, giúp kiểm tra lỗi cấu hình hoặc xem các thông tin cấu hình được tóm tắt từ tệp YAML.



1. docker compose up -d –build

Khởi động lại các container trong chế độ nền (-d) và đồng thời xây dựng lại các image nếu có thay đổi (--build). Câu lệnh này hữu ích khi bạn muốn cập nhật Image trước khi khởi động lại dịch vụ.



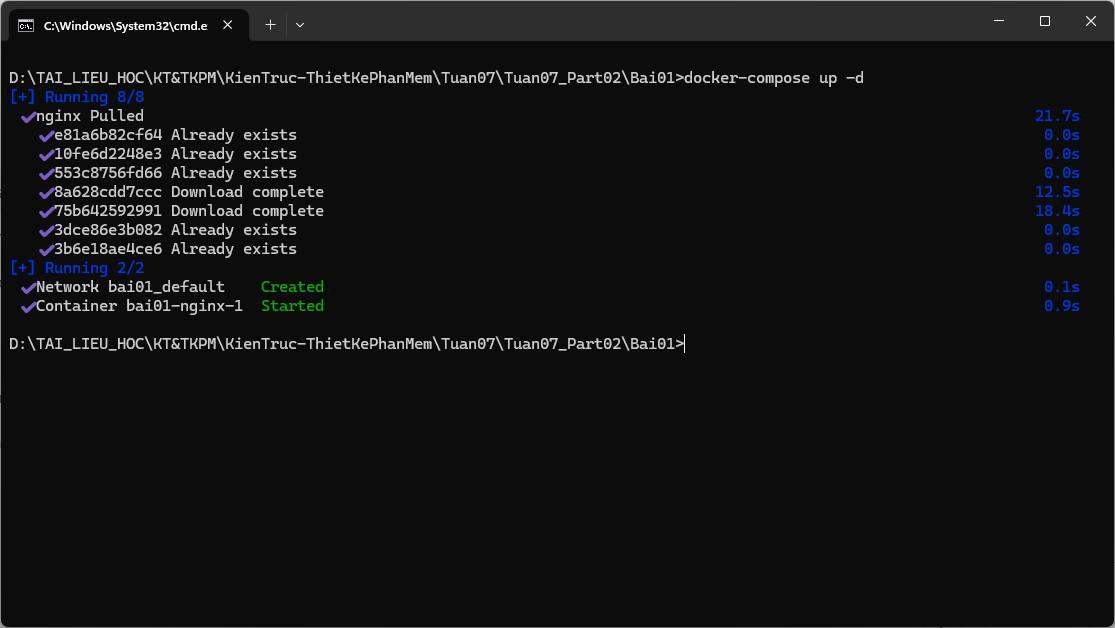
**Phần 2: Docker Compose file**

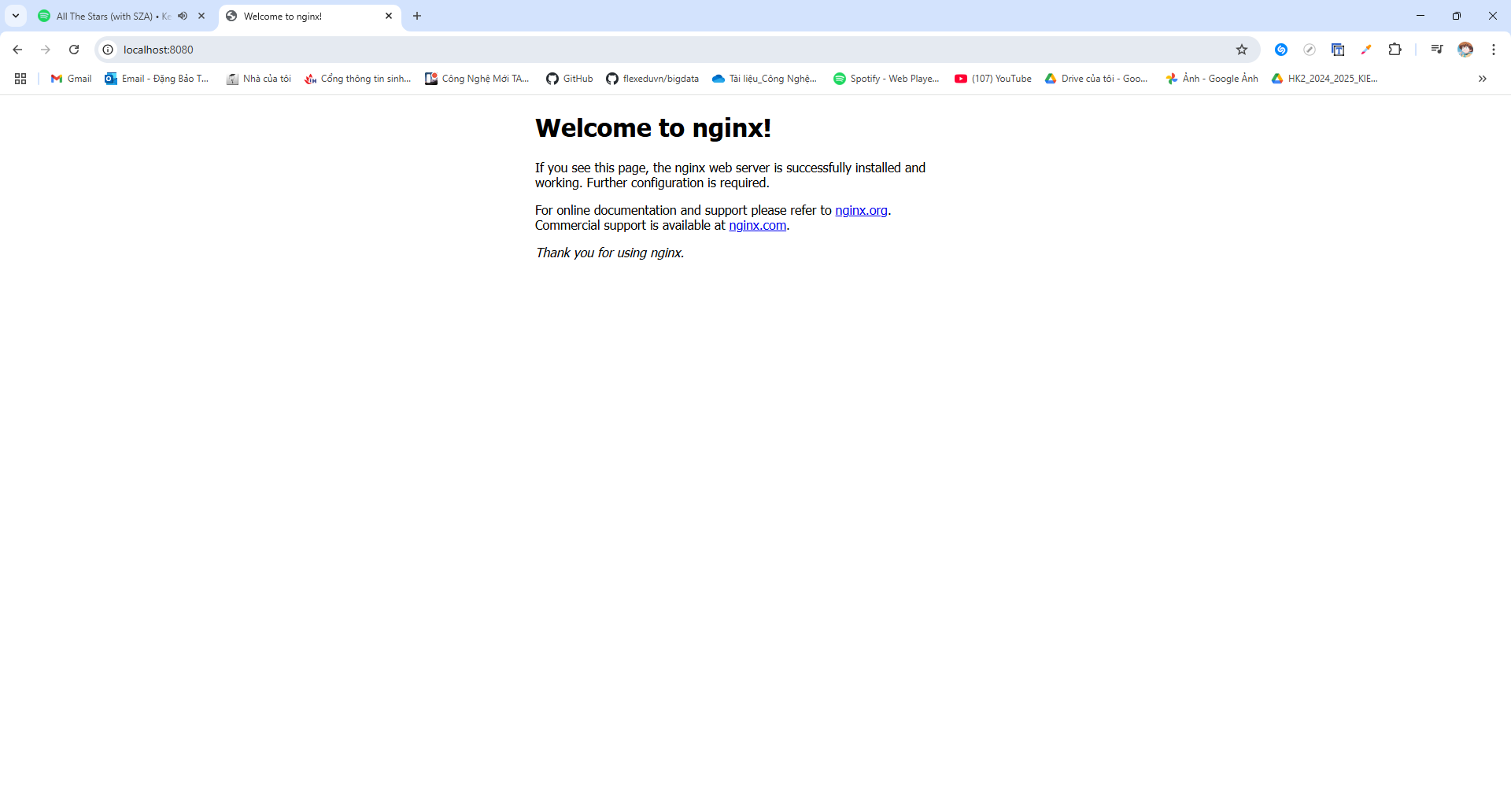
Bài 1: Chạy một container đơn giản với Docker Compose

Yêu cầu:

Tạo một container chạy Nginx bằng Docker Compose.

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



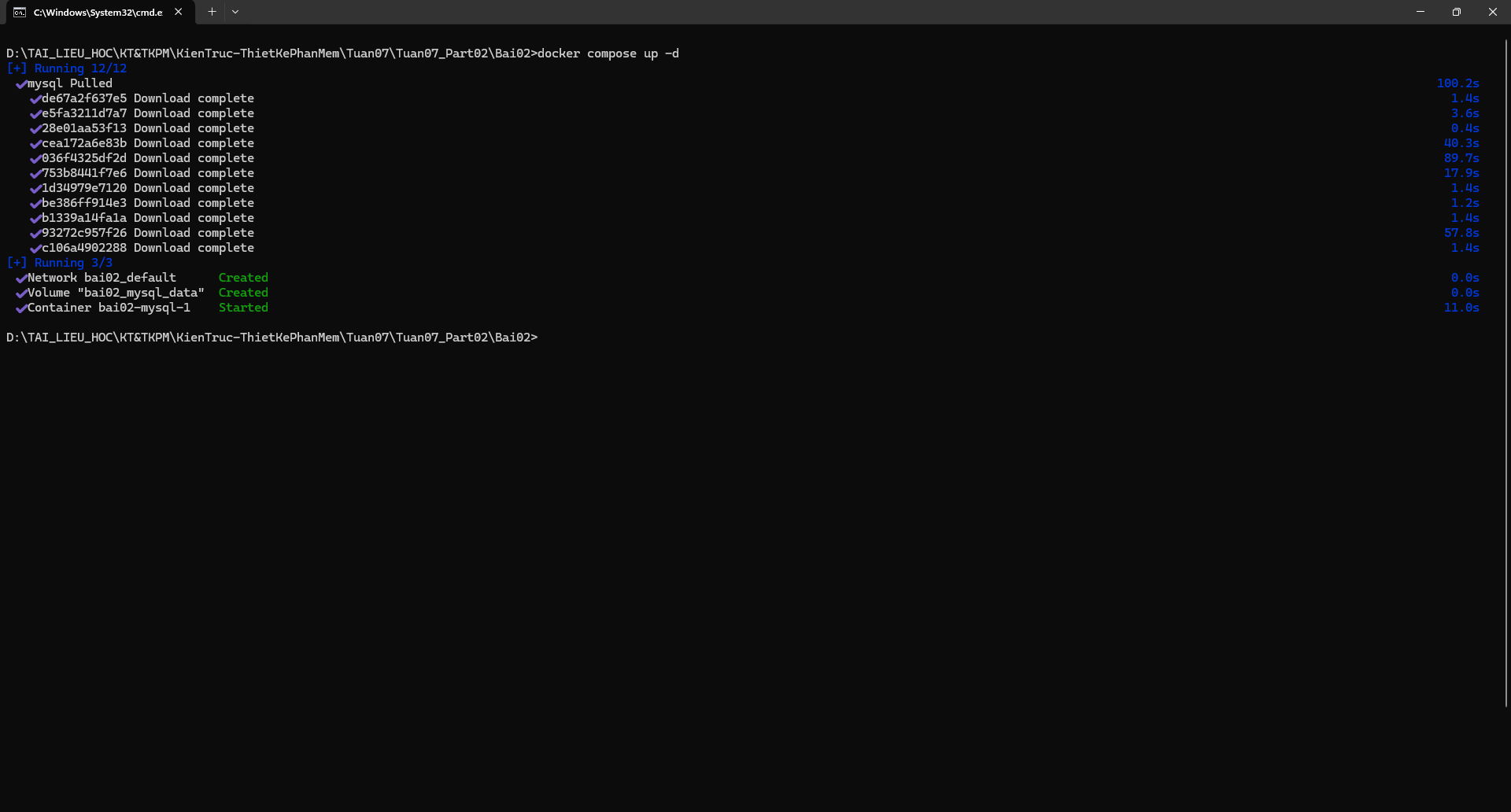


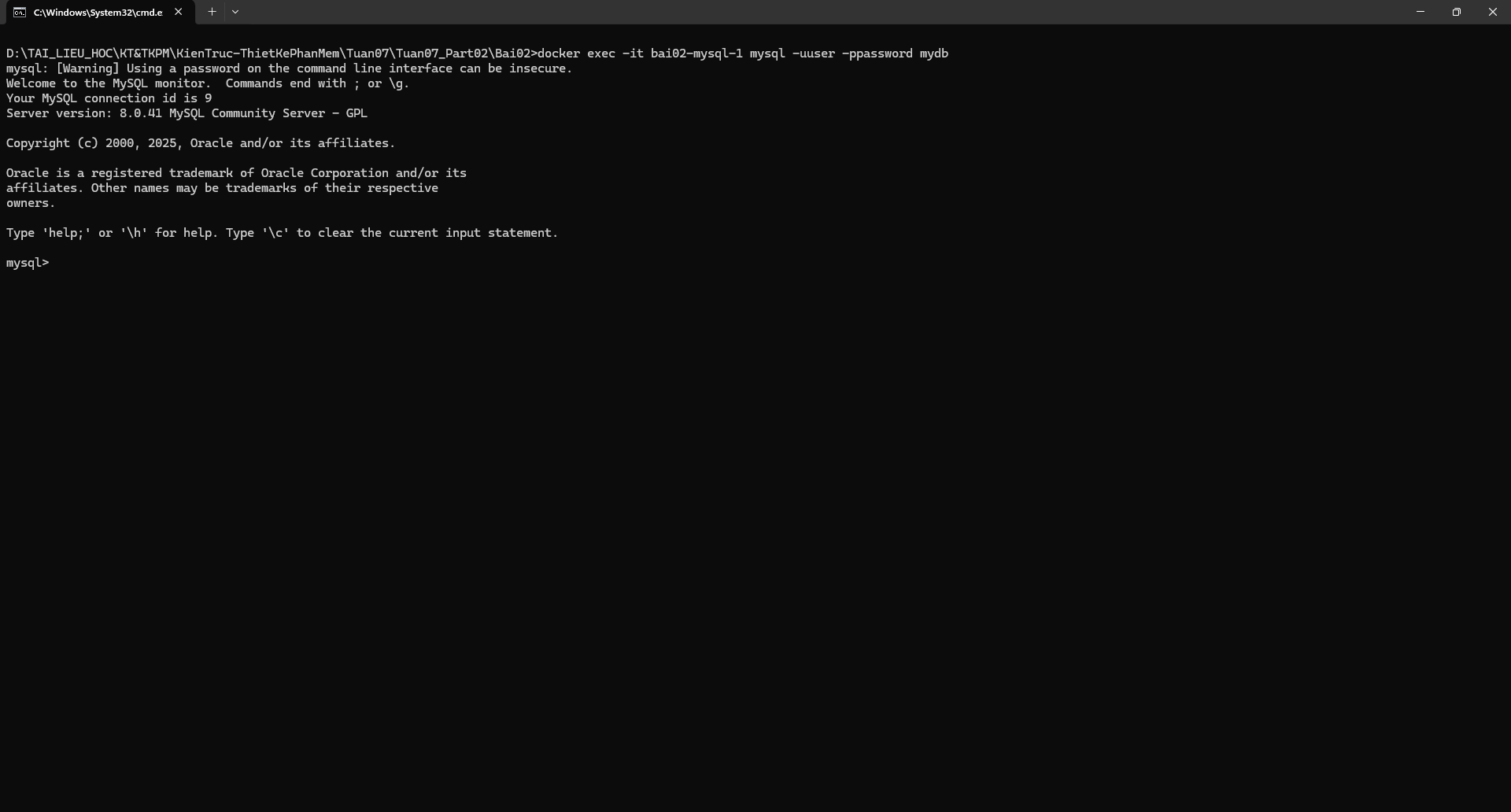
Bài 2: Chạy MySQL với Docker Compose

Yêu cầu:

Tạo một container chạy MySQL phiên bản 8.0.

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



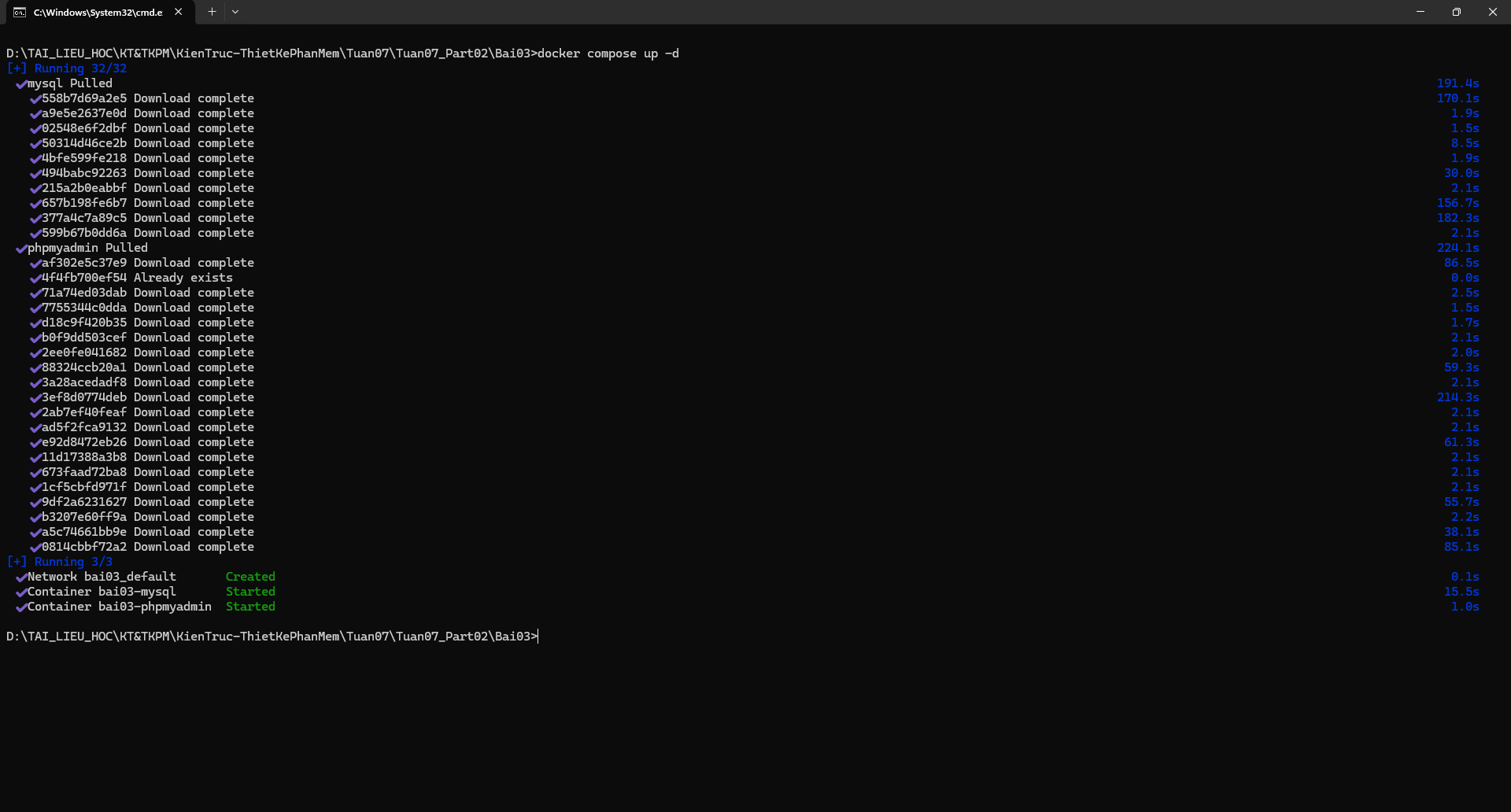


Bài 3: Kết nối MySQL với PHPMyAdmin

Yêu cầu:

Chạy MySQL và PHPMyAdmin với Docker Compose.

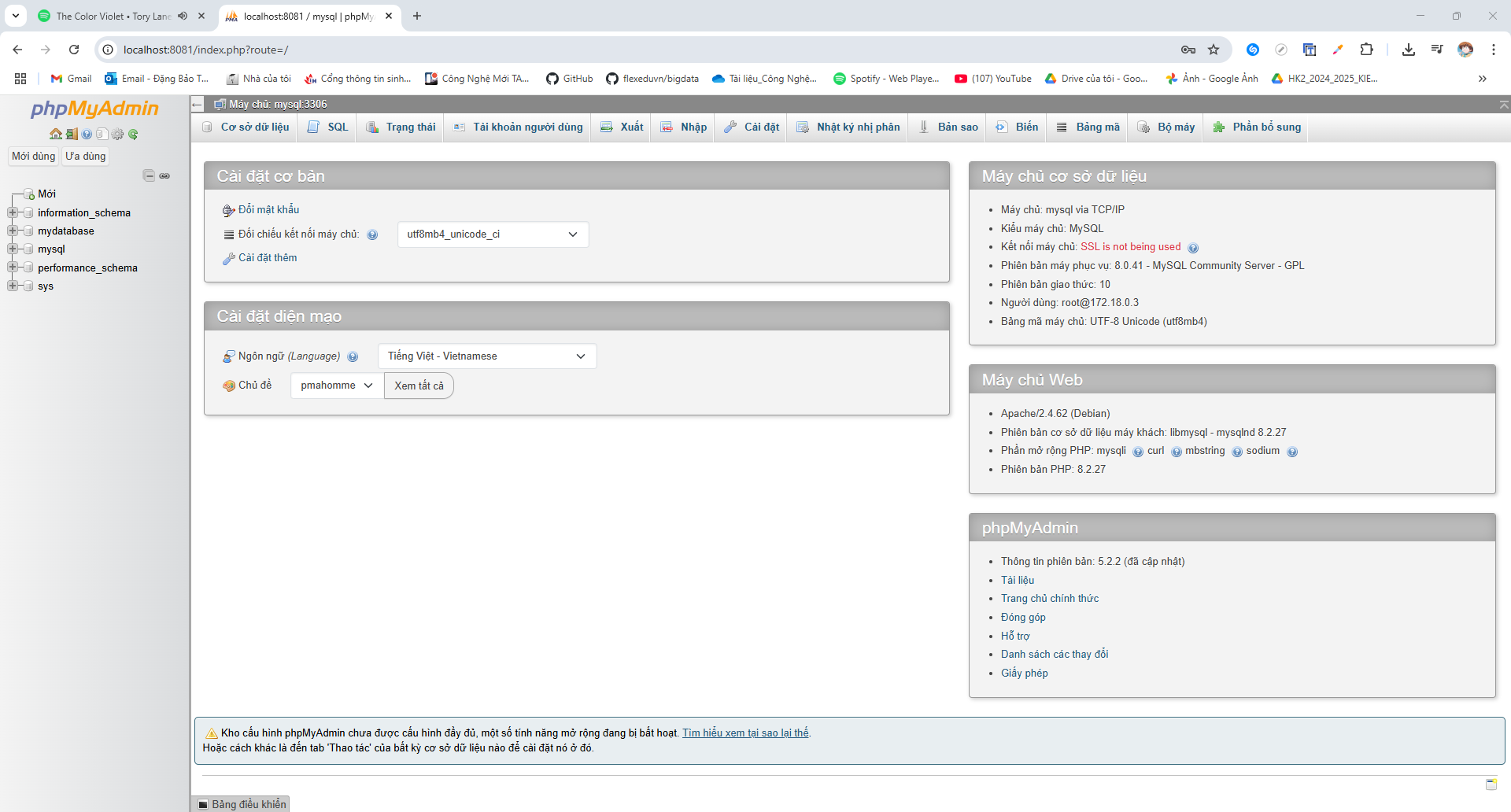
PHPMyAdmin chạy trên cổng 8081.



http://localhost:8081

Username: root

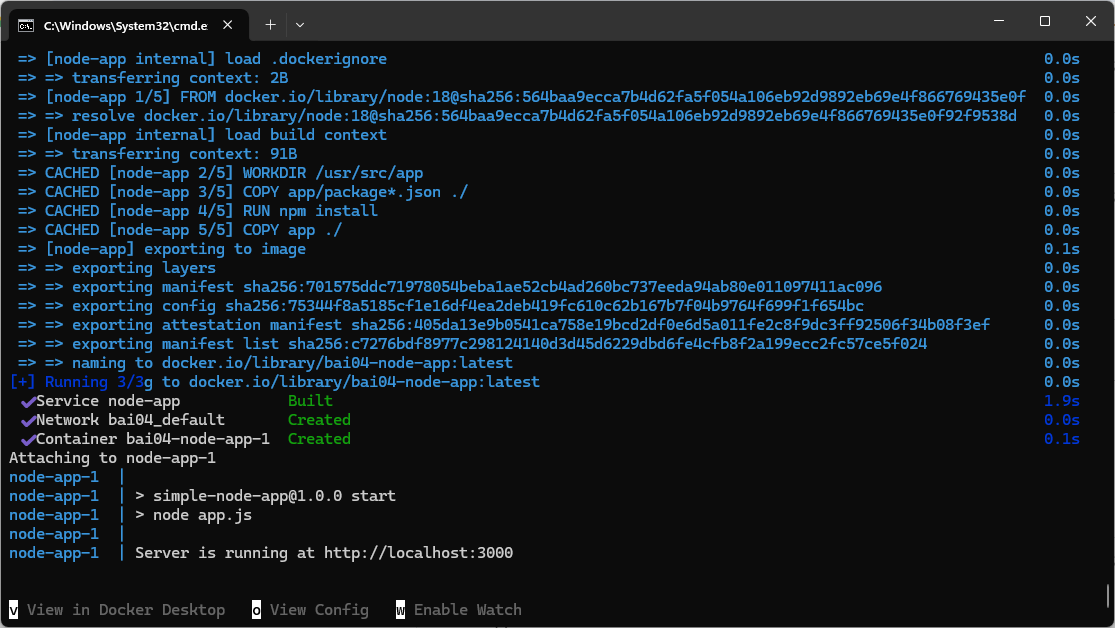
Password: rootpassword



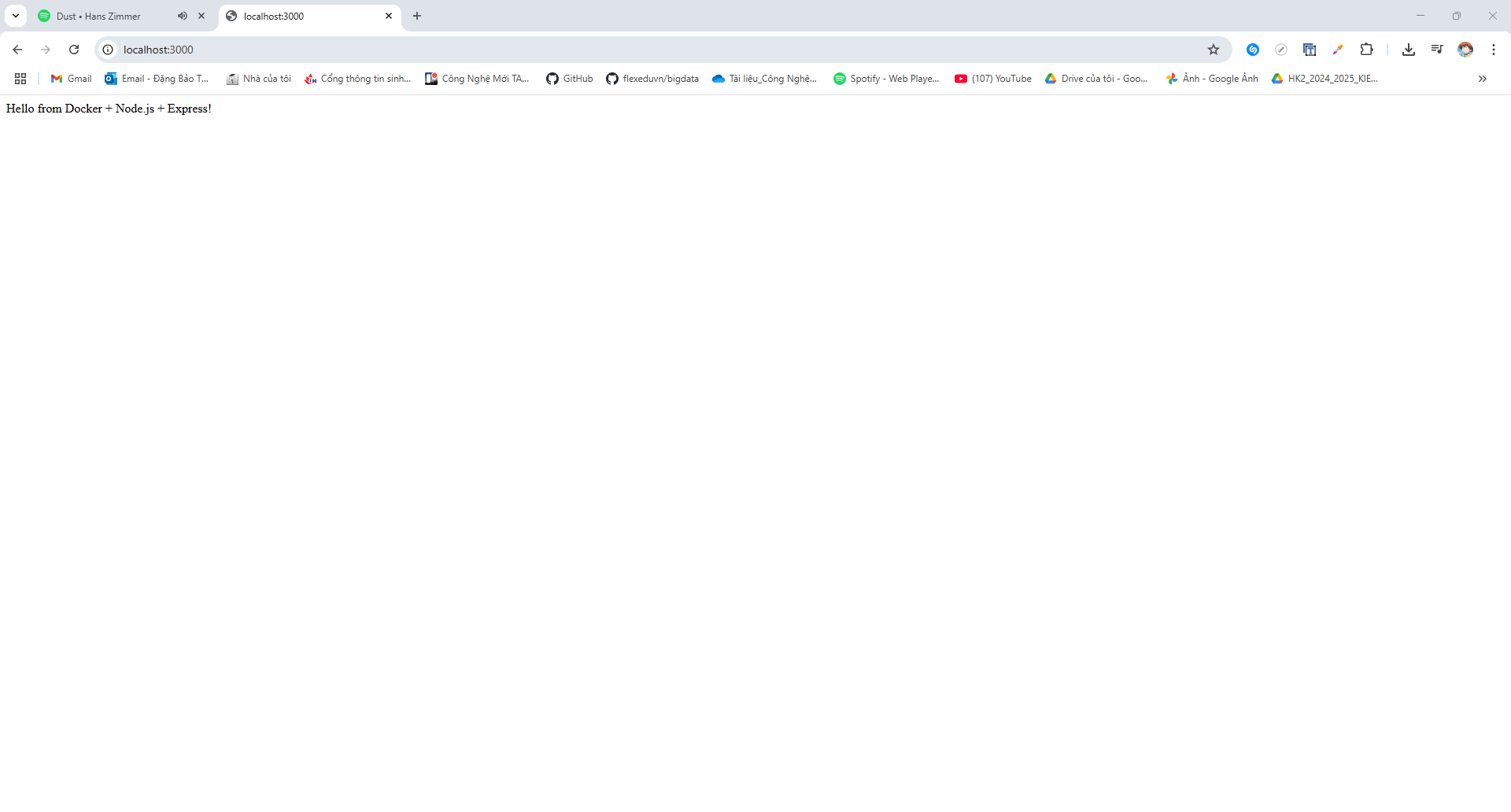
Bài 4: Chạy ứng dụng Node.js với Docker Compose

Yêu cầu:

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



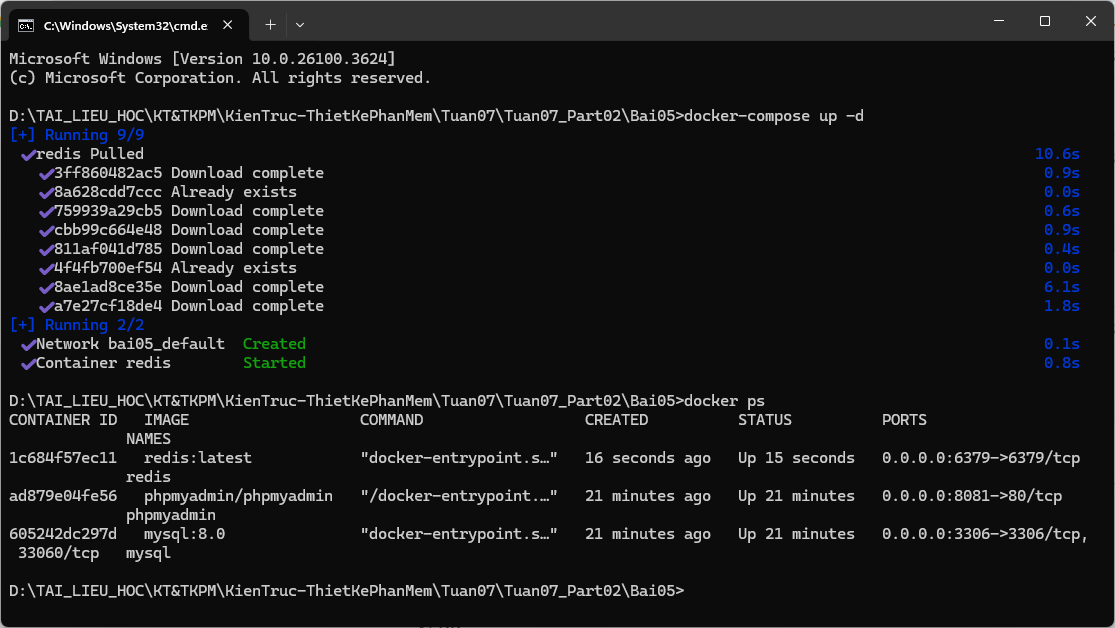
http://localhost:3000



Bài 5: Chạy Redis với Docker Compose

Yêu cầu:

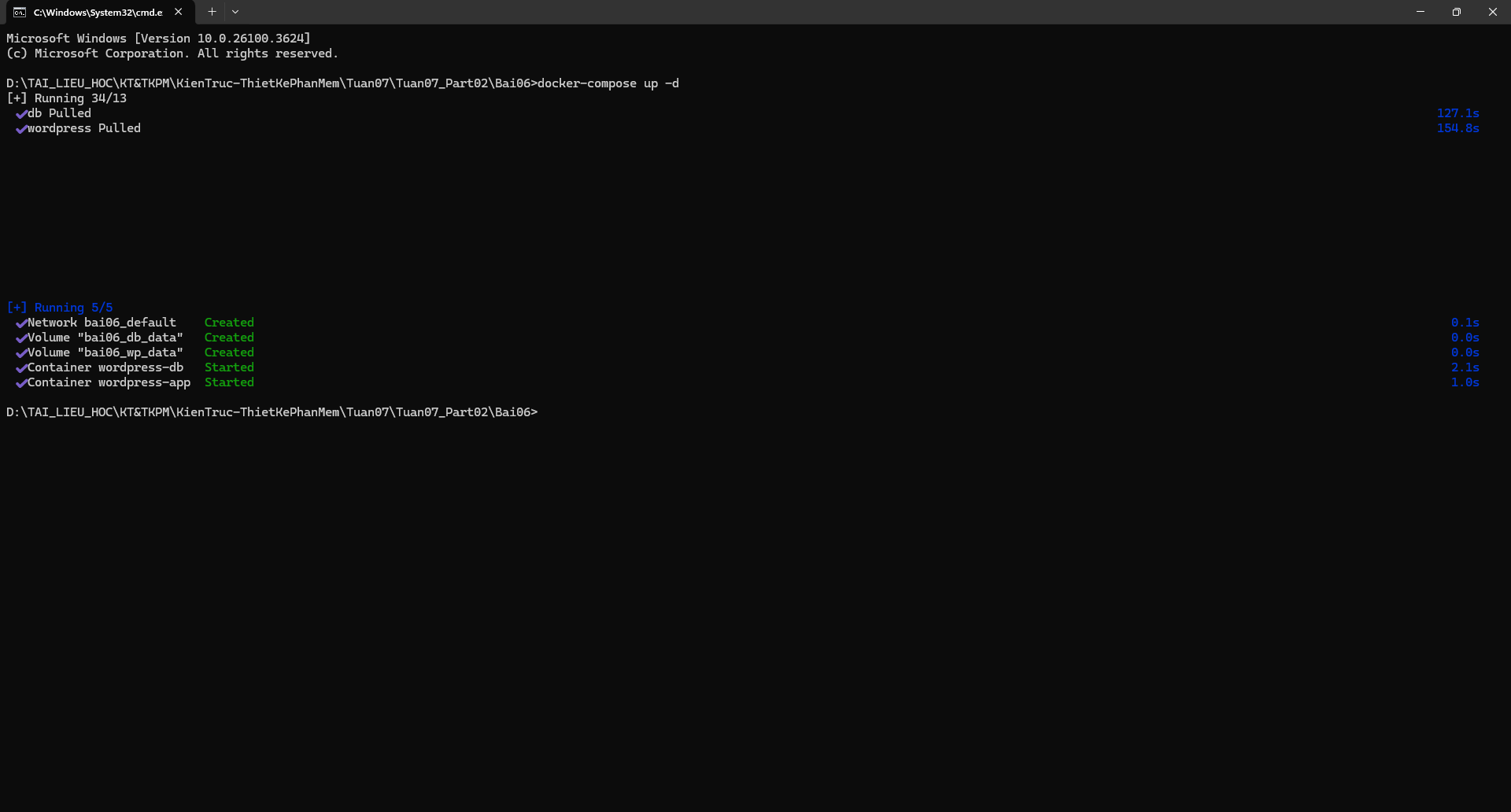
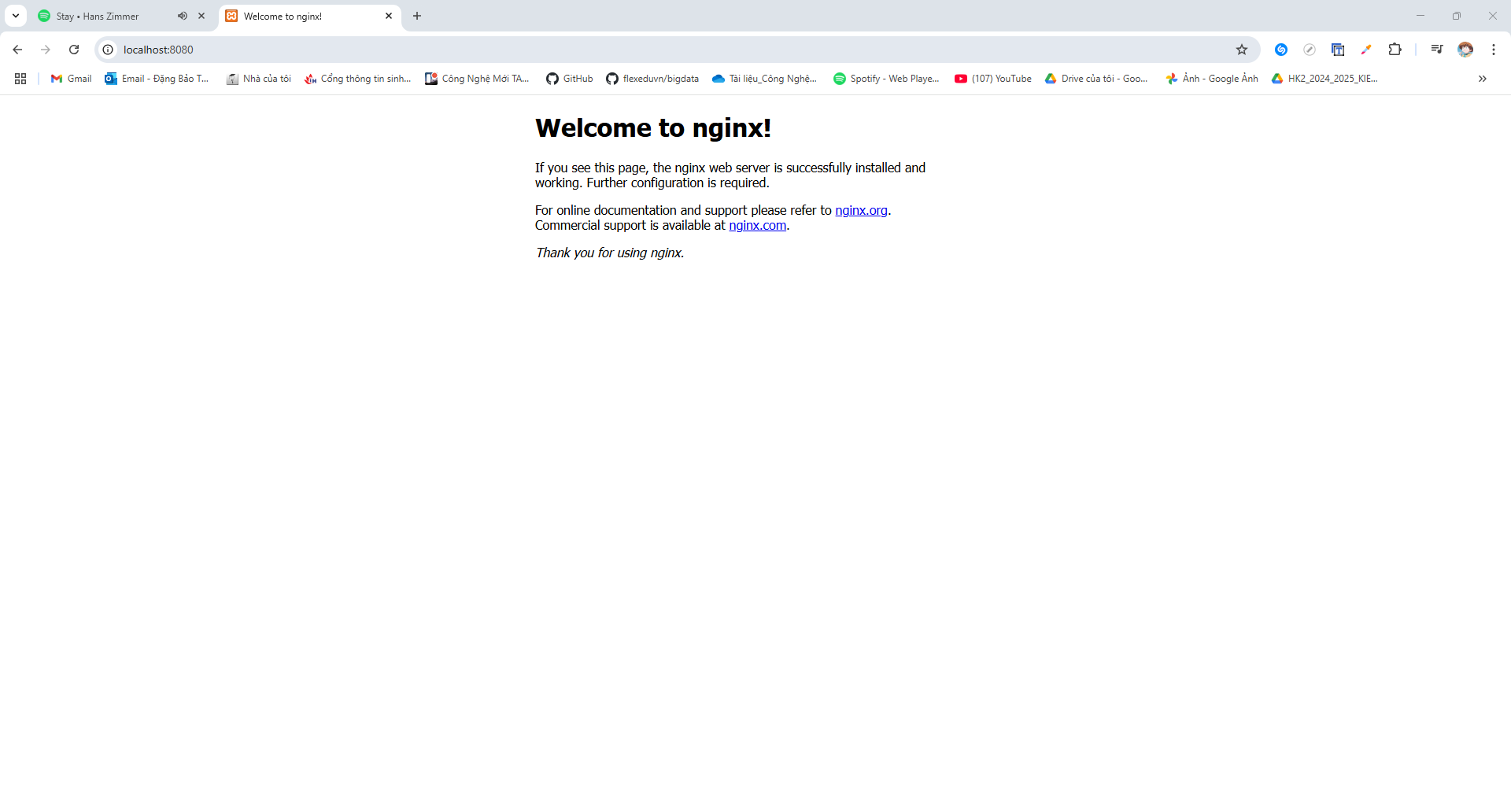
Chạy một container Redis trên cổng 6379.

Bài 6: Chạy WordPress với MySQL

Yêu cầu:

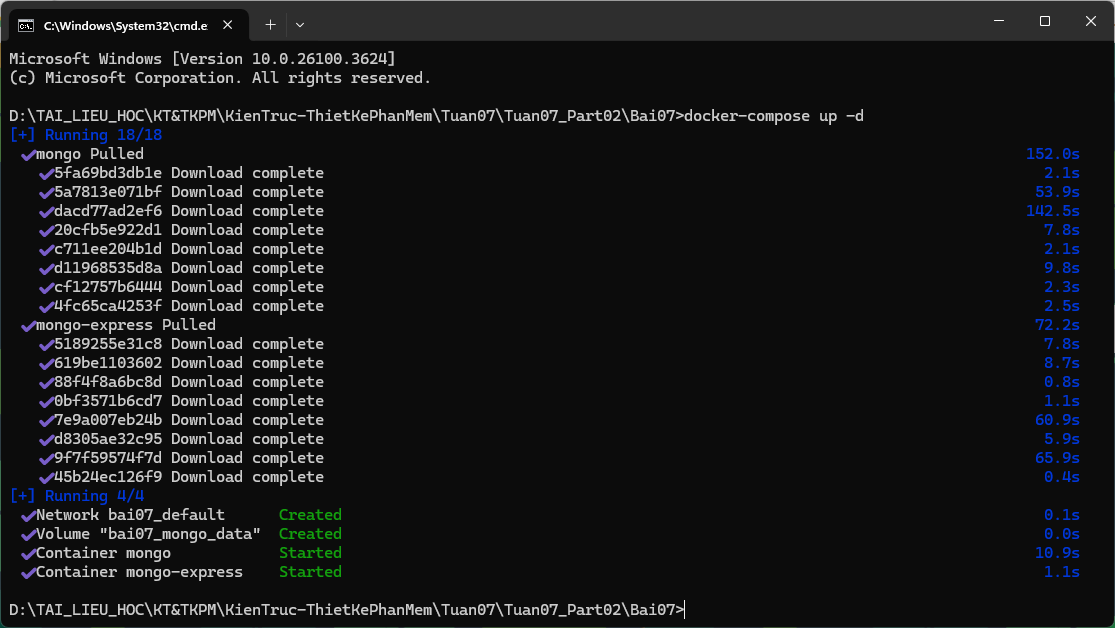
Chạy WordPress với MySQL bằng Docker Compose.

 http://localhost:8080

Bài 7: Chạy MongoDB với Docker Compose

Yêu cầu:

Chạy MongoDB và Mongo Express để quản lý.



<http://localhost:8081>

Username: admin

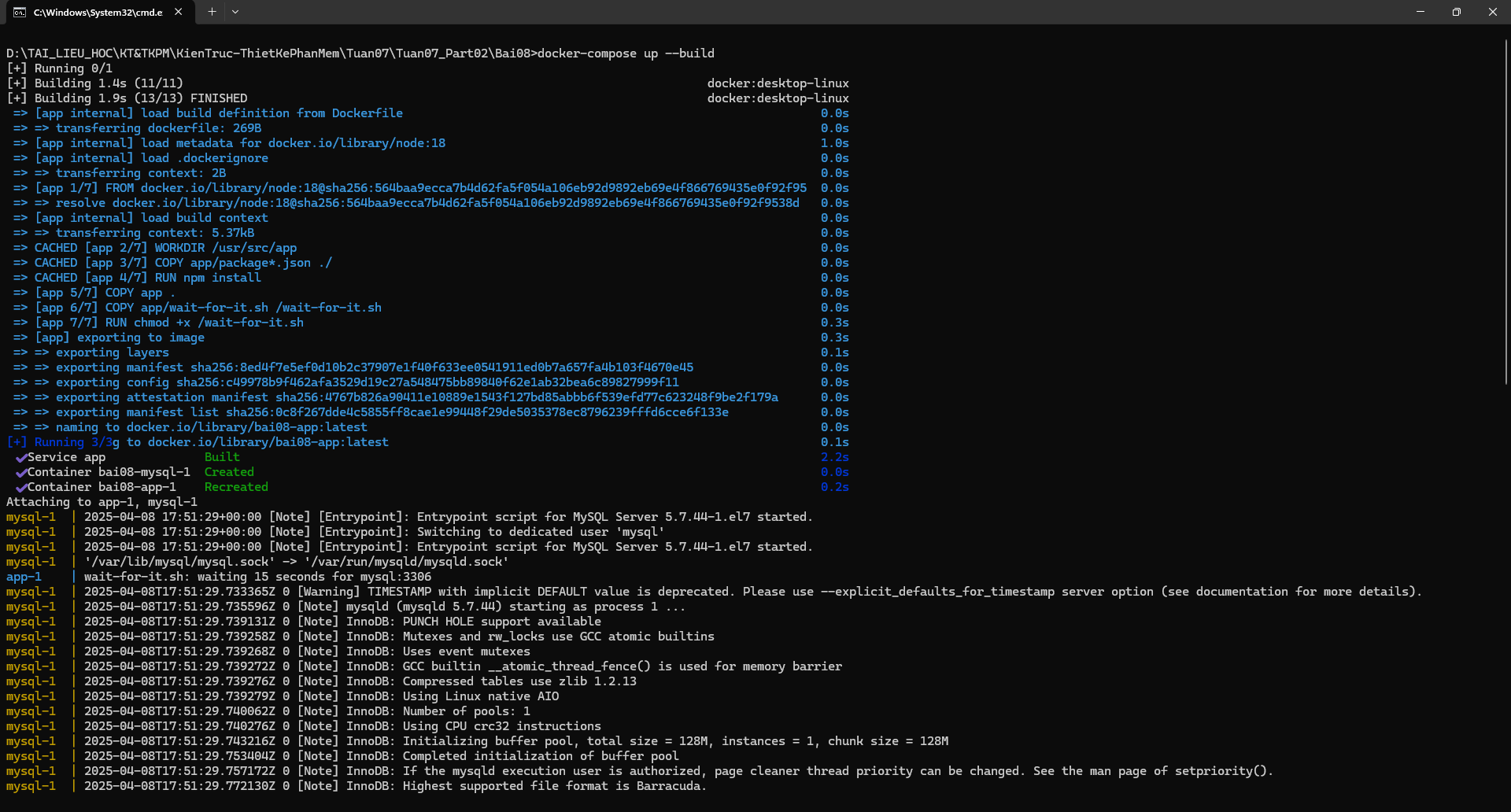
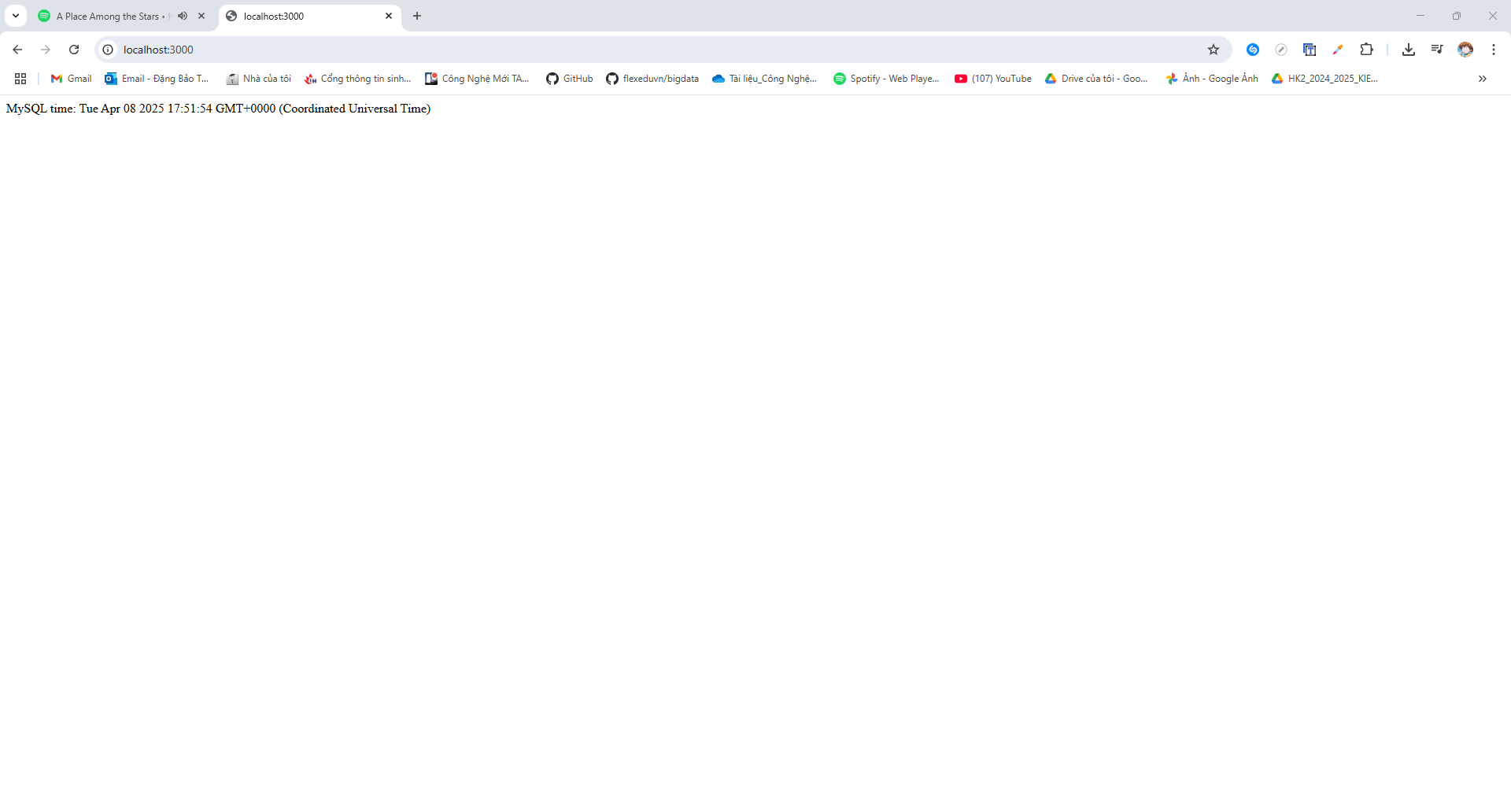
Password: admin123



Bài 8: Kết nối nhiều dịch vụ với Docker Compose

Yêu cầu:

Chạy Node.js kết nối với MySQL.

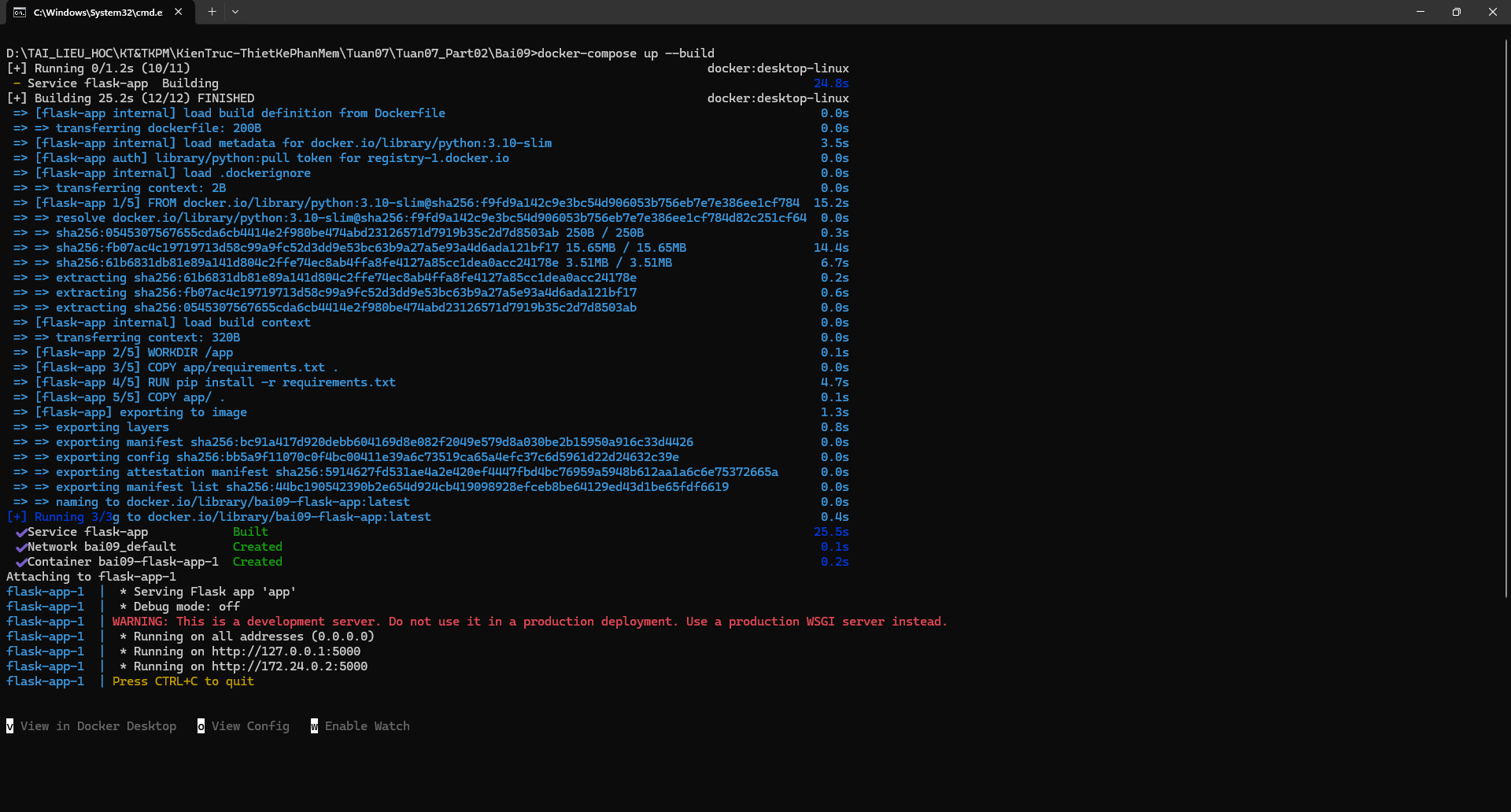
 

http://localhost:3000

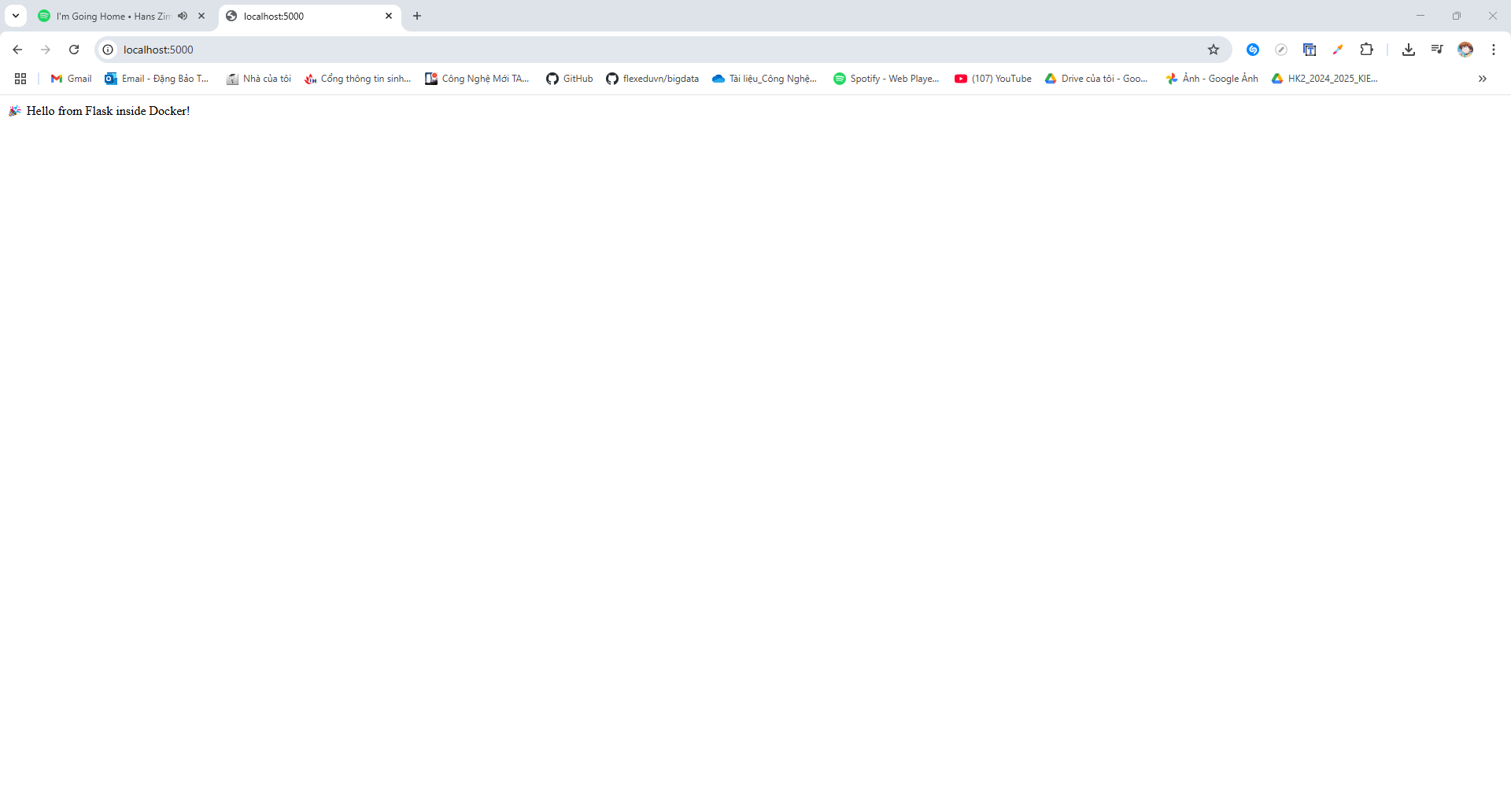
Bài 9: Chạy ứng dụng Python Flask với Docker Compose

Yêu cầu:

Chạy ứng dụng Flask đơn giản với Docker Compose.



http://localhost:5000/

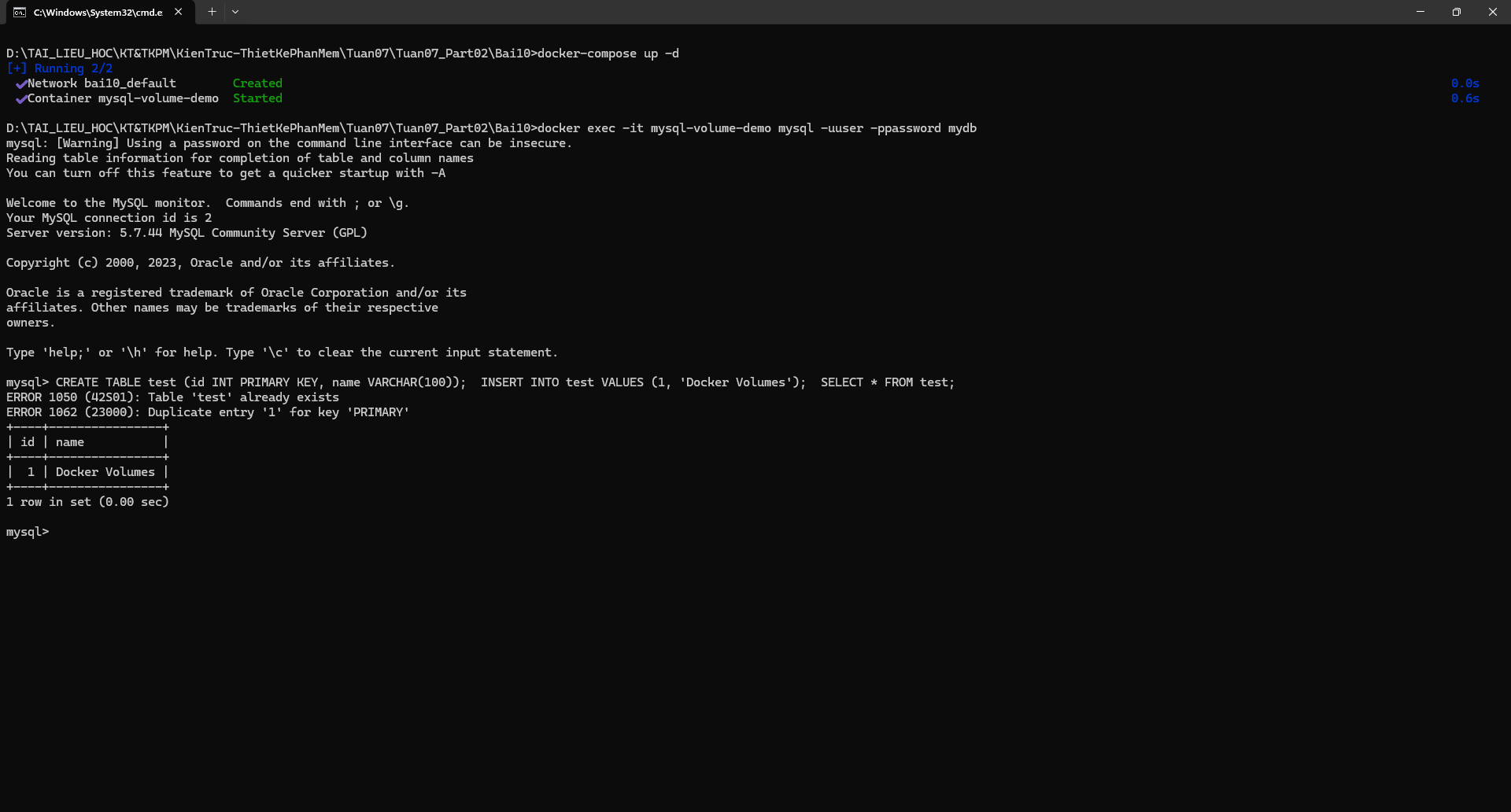


Bài 10: Lưu trữ dữ liệu với Docker Volumes

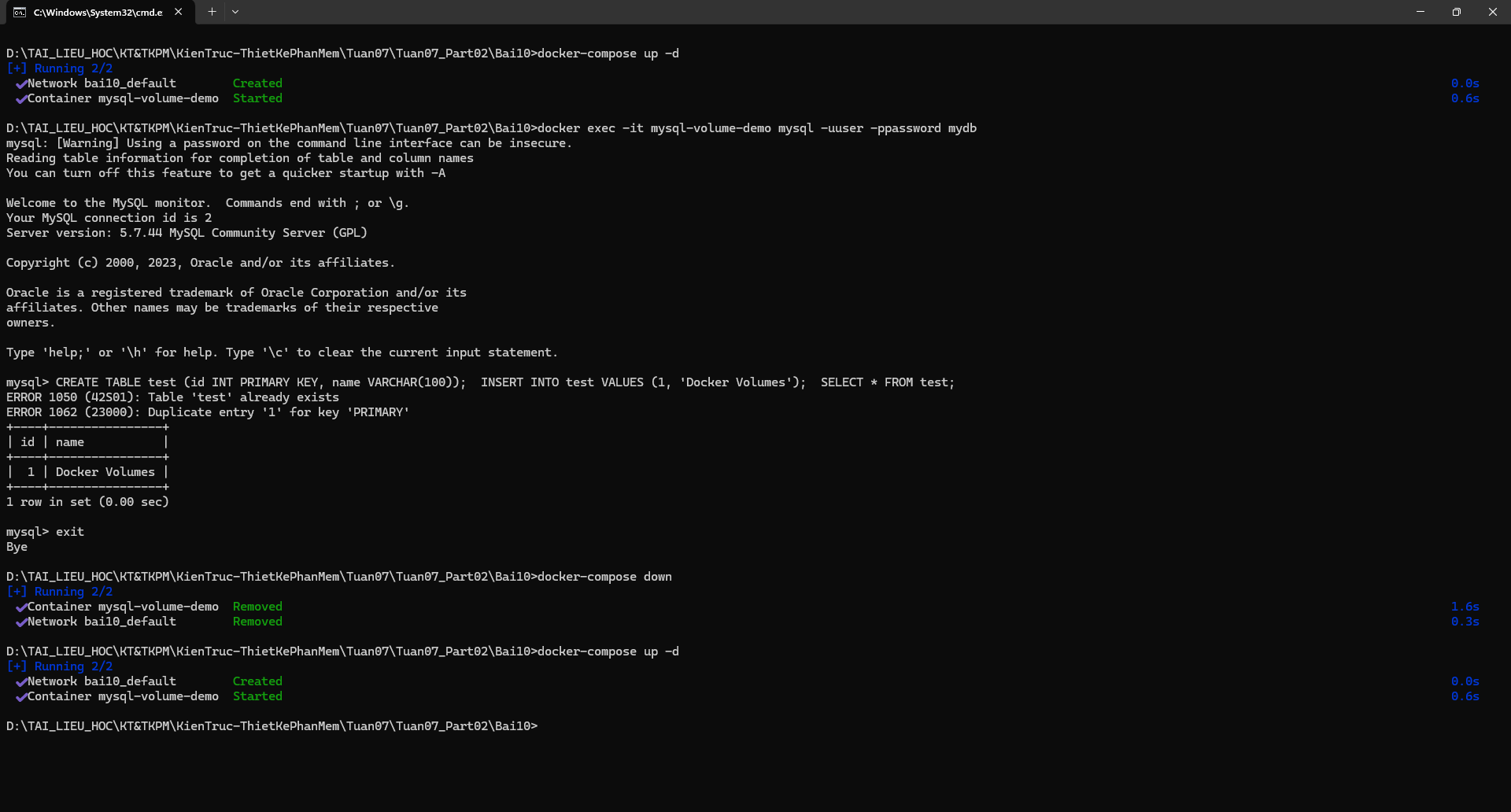
Yêu cầu:

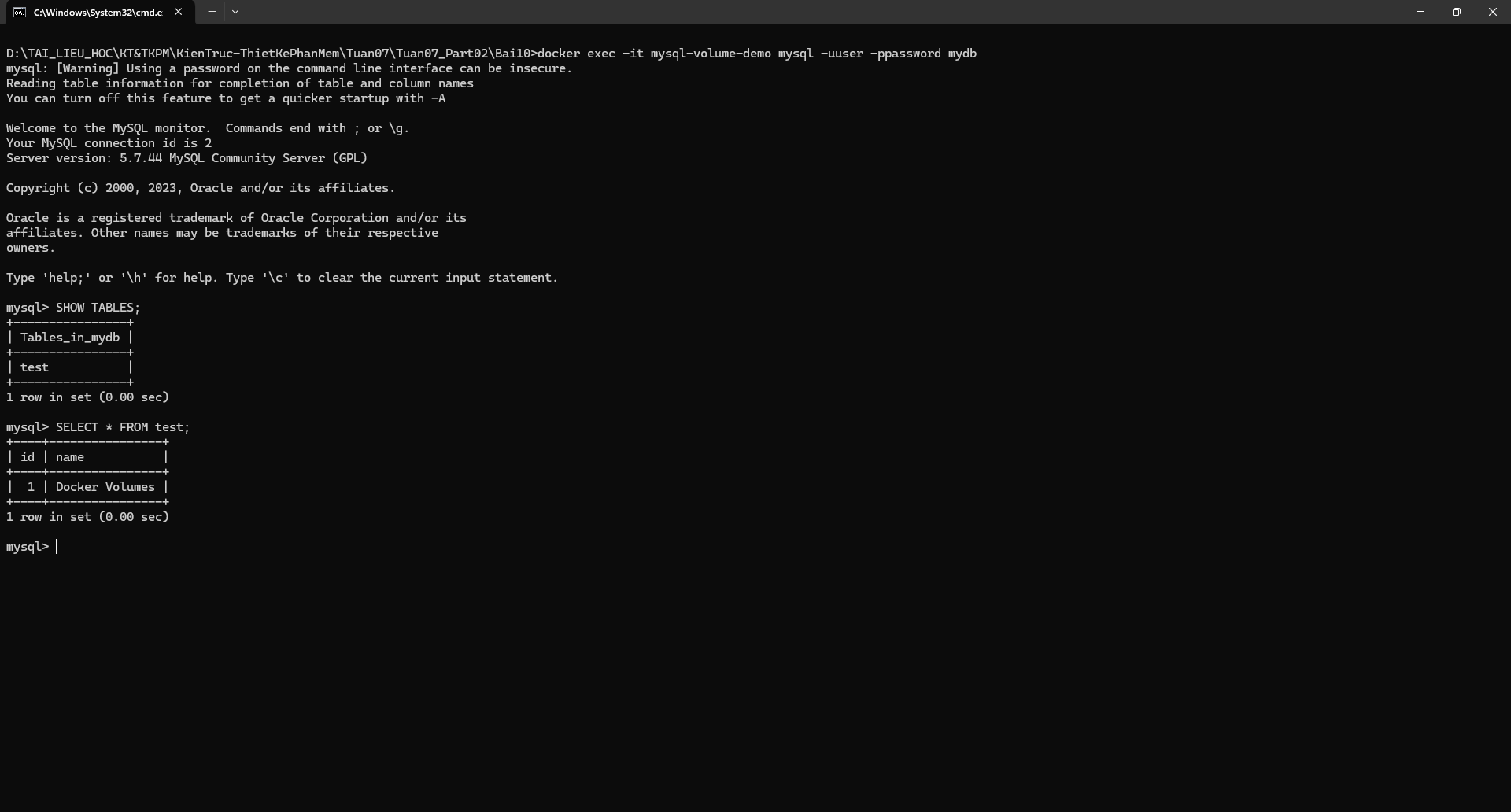
Chạy MySQL và gắn volume để dữ liệu không bị mất.

Sau khi chạy docker-compose up -d thì vào MySQL và thêm 1 row vào bảng test



Ta thoát khỏi MySQL -> xóa container -> khởi động lại



Vào lại thì thấy dữ liệu vẫn còn 

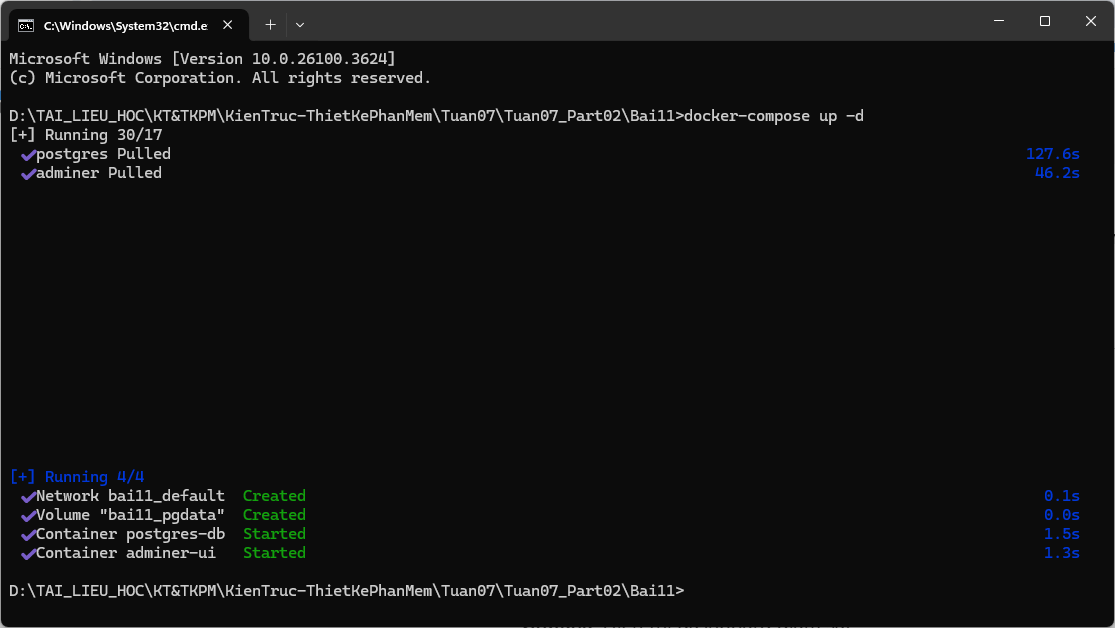
Bài 11: Chạy dịch vụ Postgres với Adminer

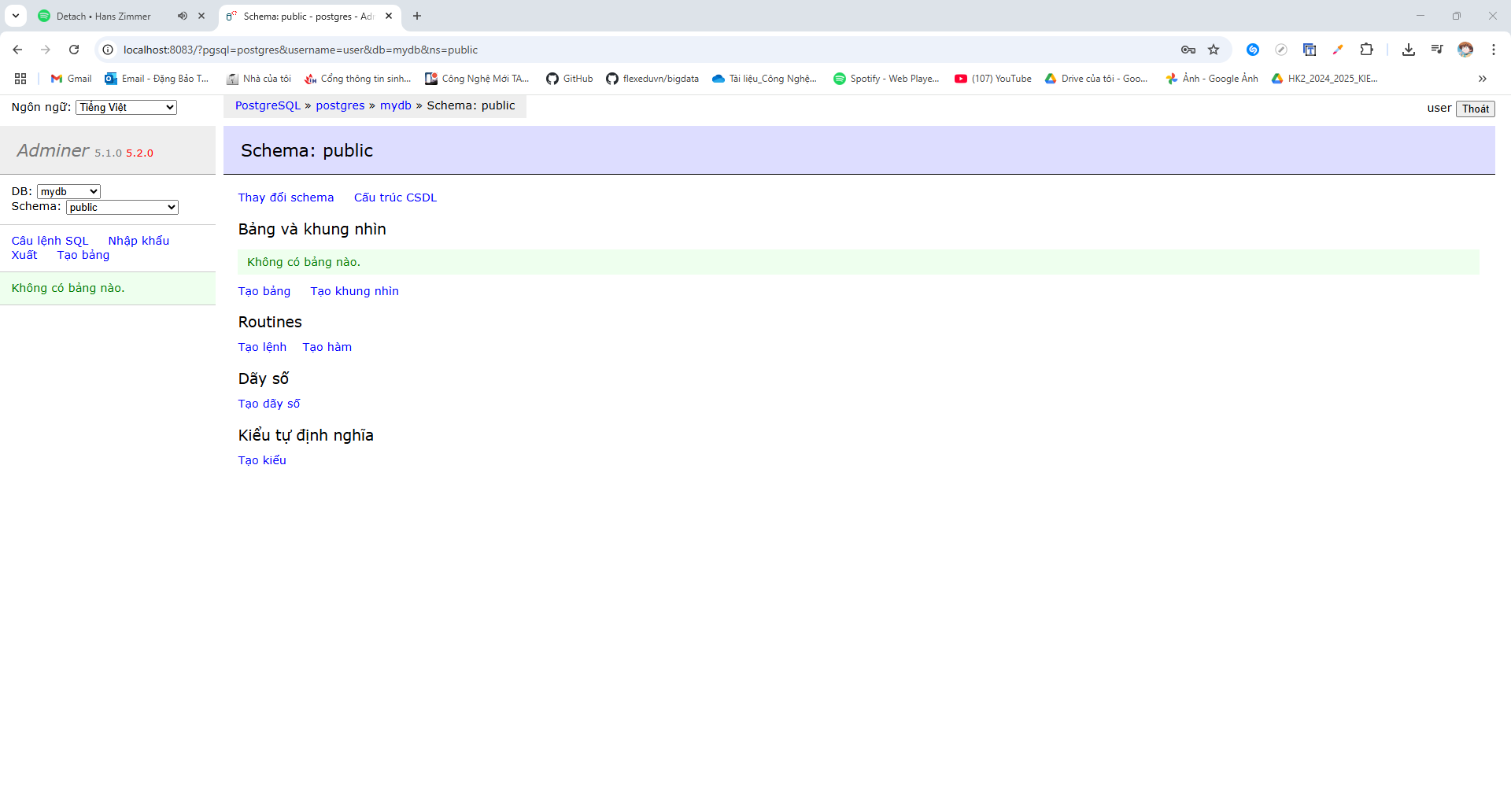
Yêu cầu:

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





<http://localhost:8083>

System: PostgreSQL

Server: postgres (vì chạy chung trong mạng Docker Compose)

Username: user

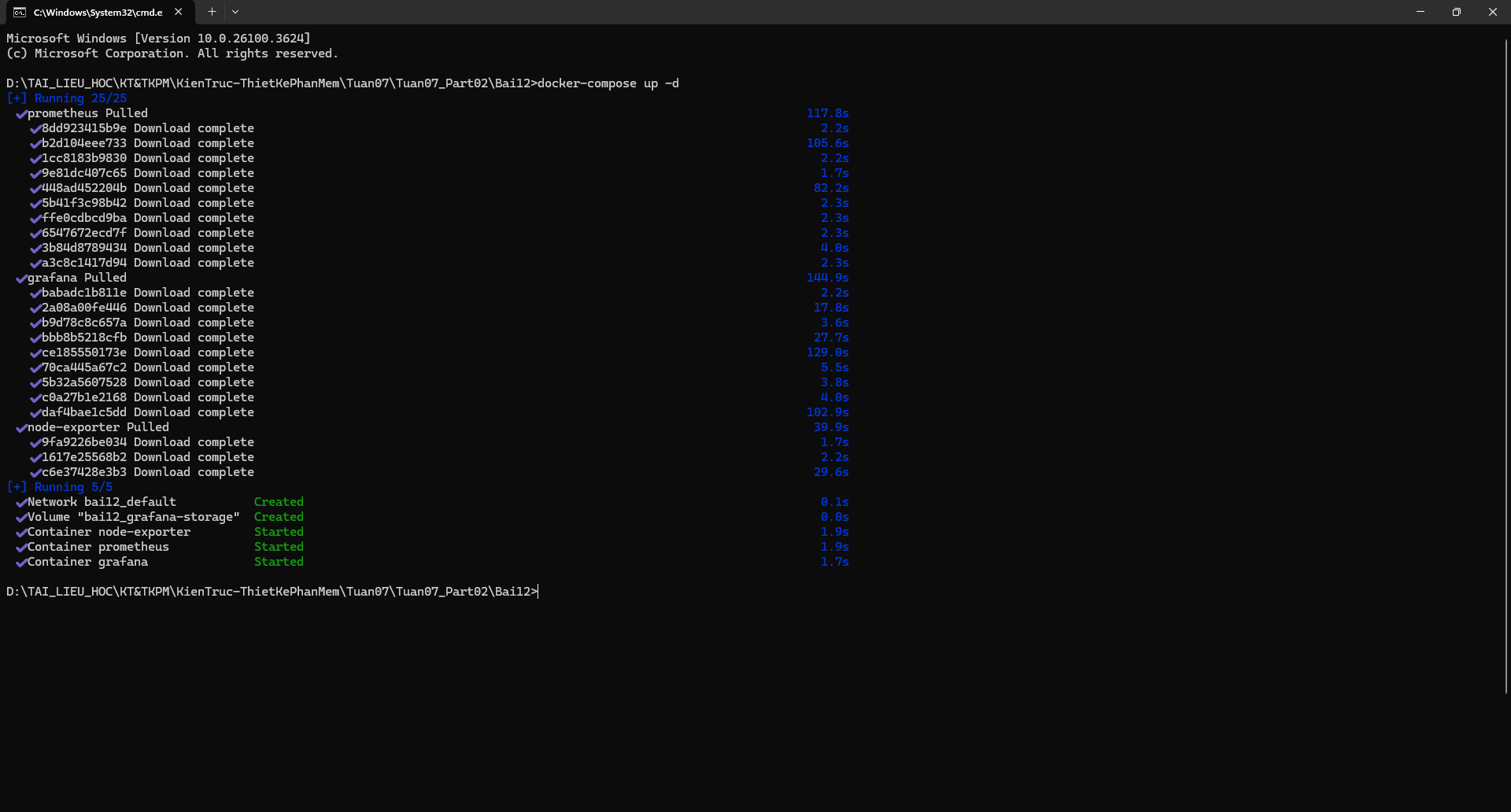
Password: password

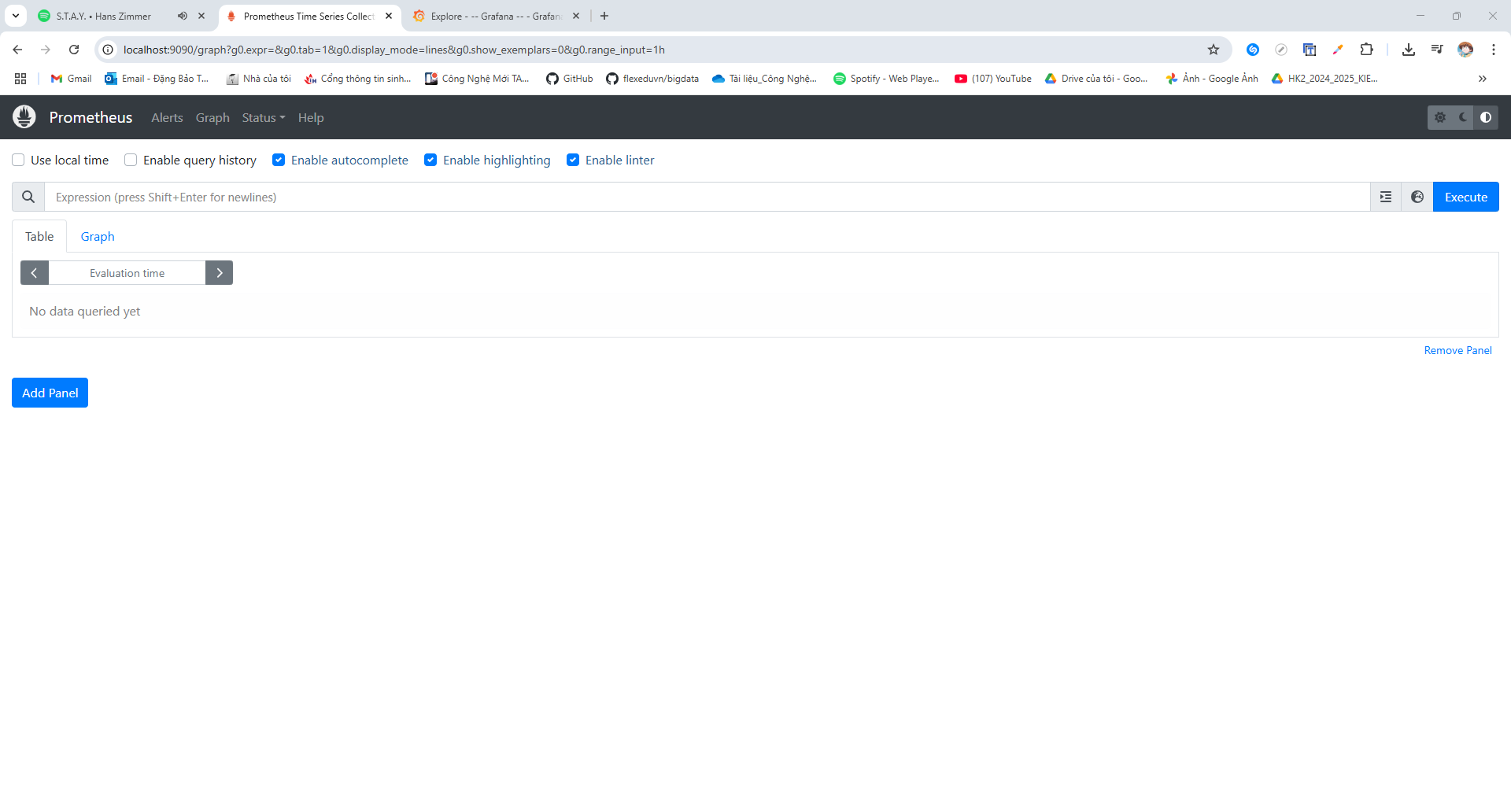
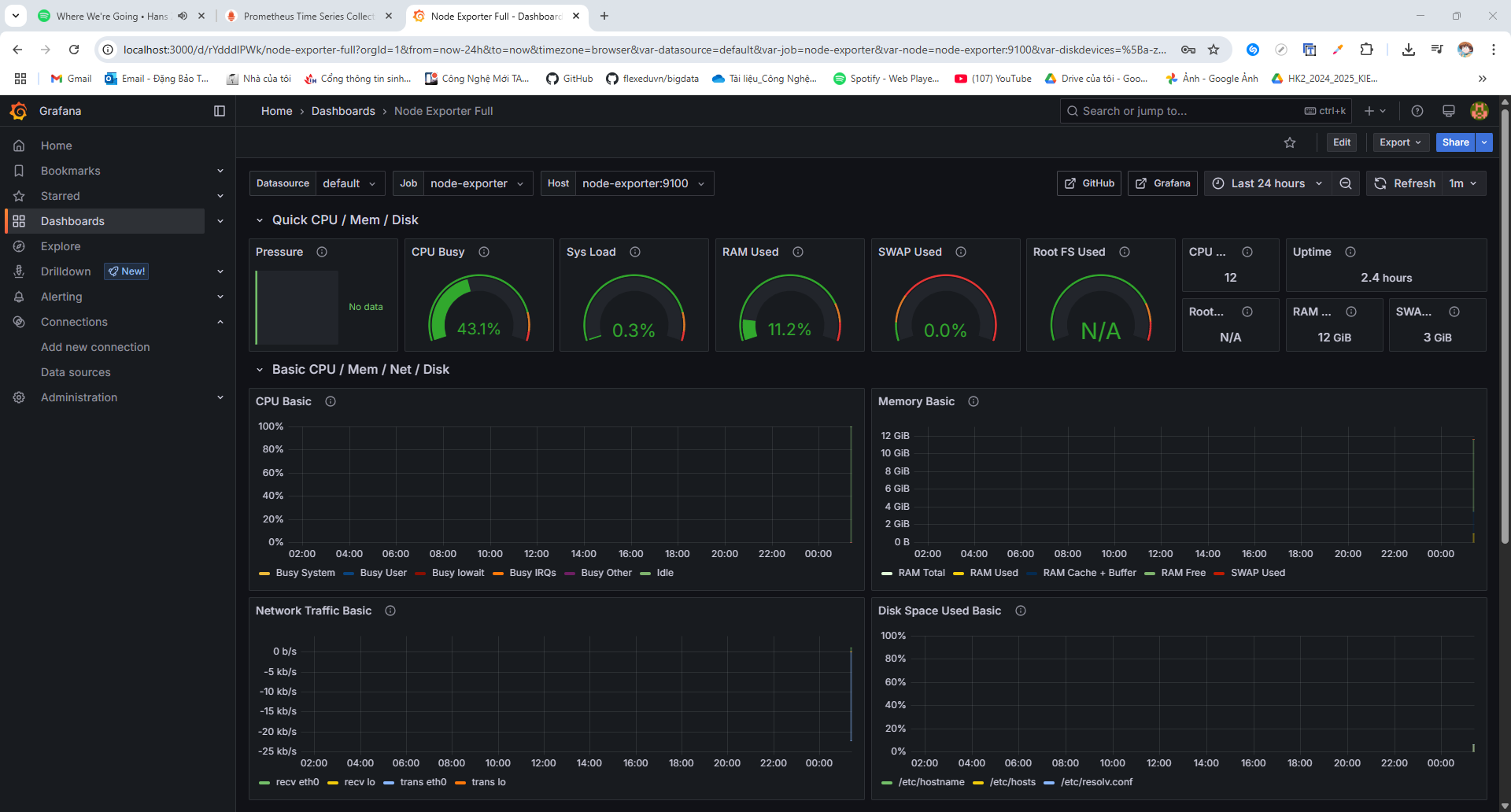
Database: mydb

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.



 Prometheus: <http://localhost:9090>

Cấu hình Grafana

* Vào <http://localhost:3000>
* Đăng nhập (admin/admin)
* Add data source:

Chọn Prometheus

URL: <http://prometheus:9090>

Save & Test

* Import dashboard:

Chọn + → Import

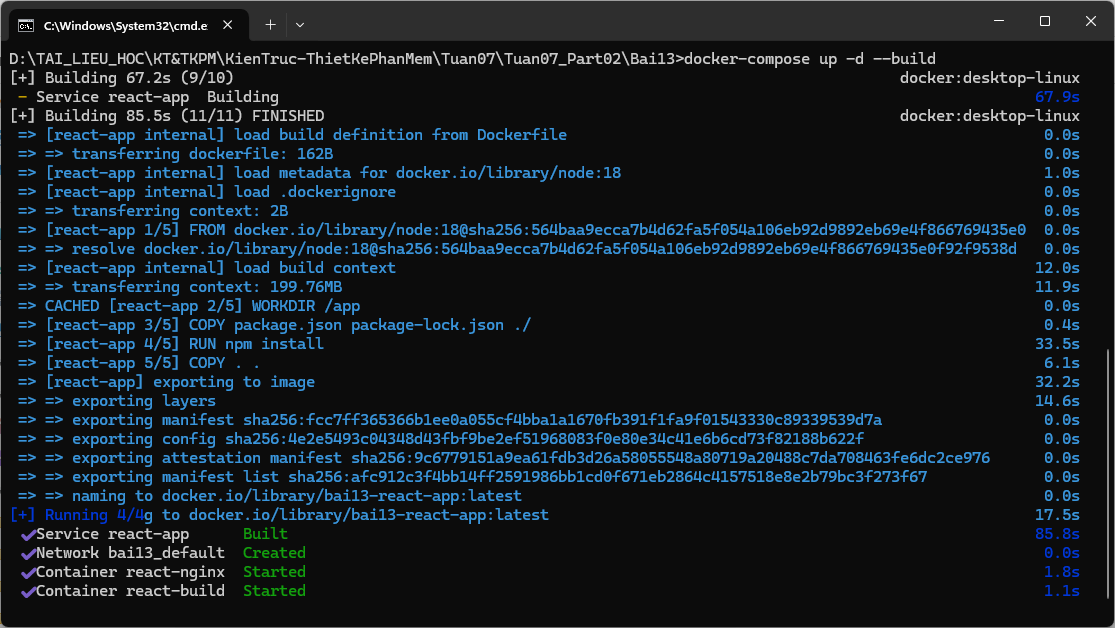
Nhập dashboard ID: 1860 (Node Exporter Full)

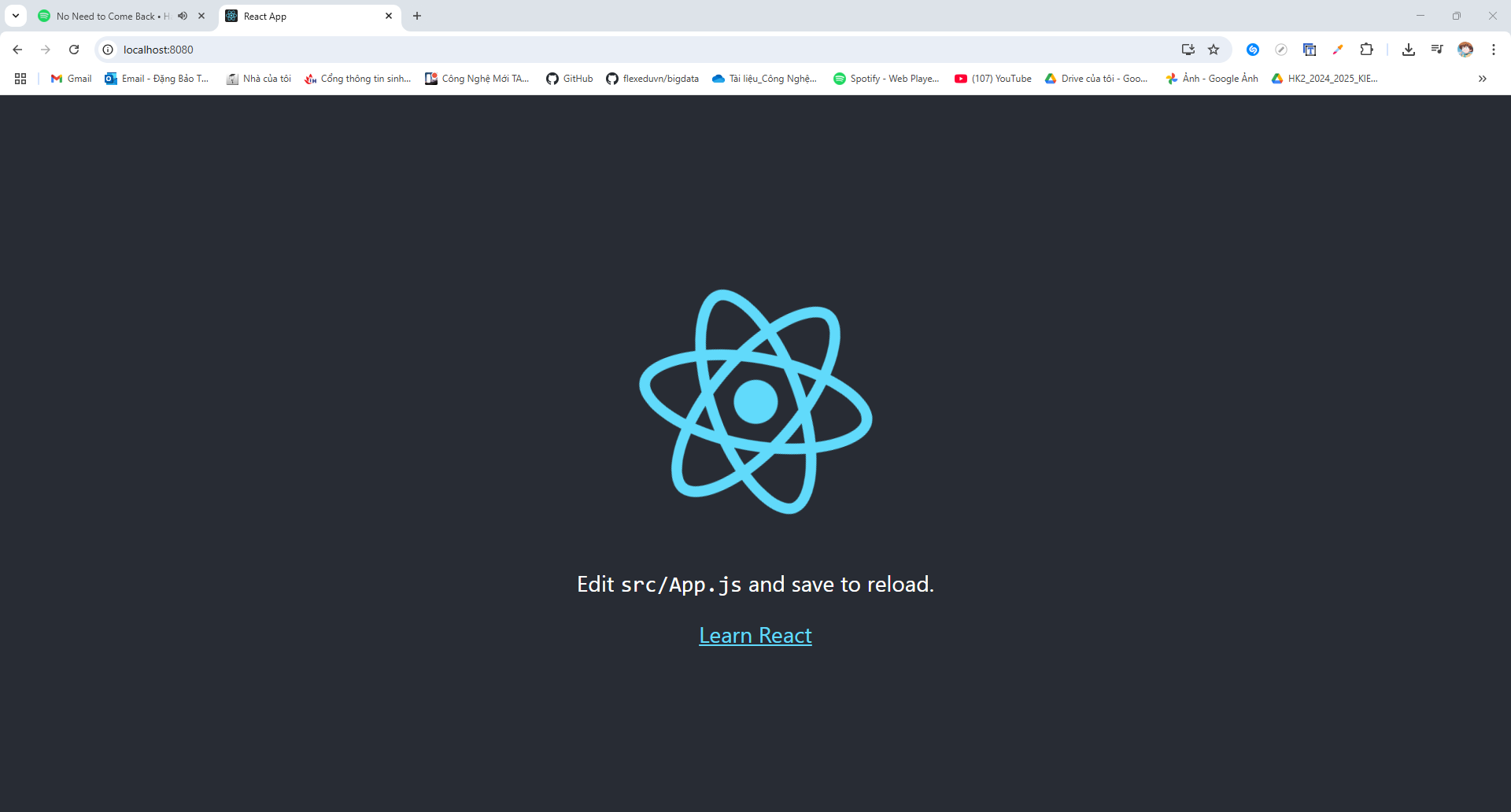
Click Load, chọn Prometheus làm data source → Import

Bài 13: Chạy ứng dụng React với Nginx

Yêu cầu:

Chạy một ứng dụng React và serve nó bằng Nginx.



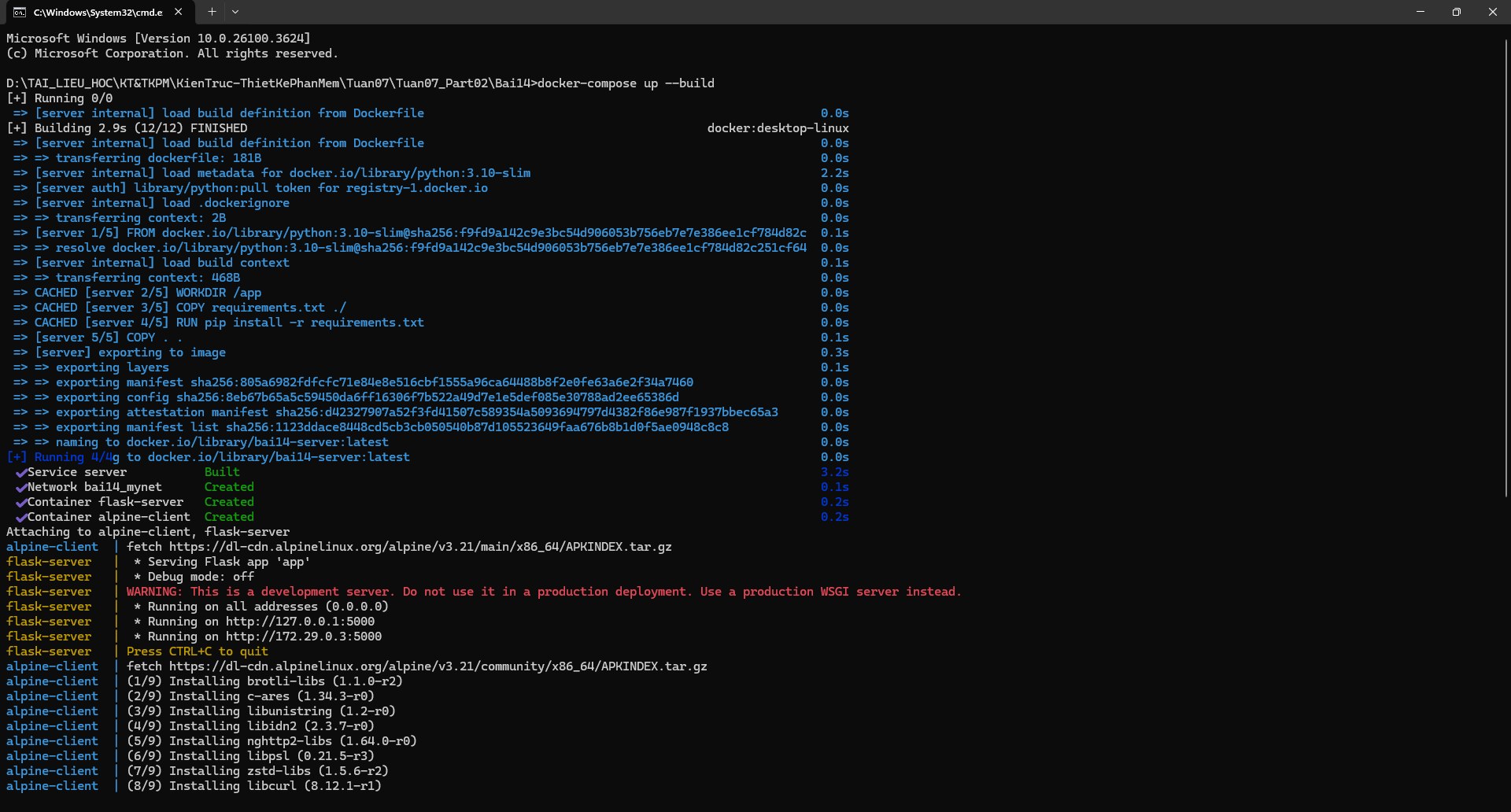
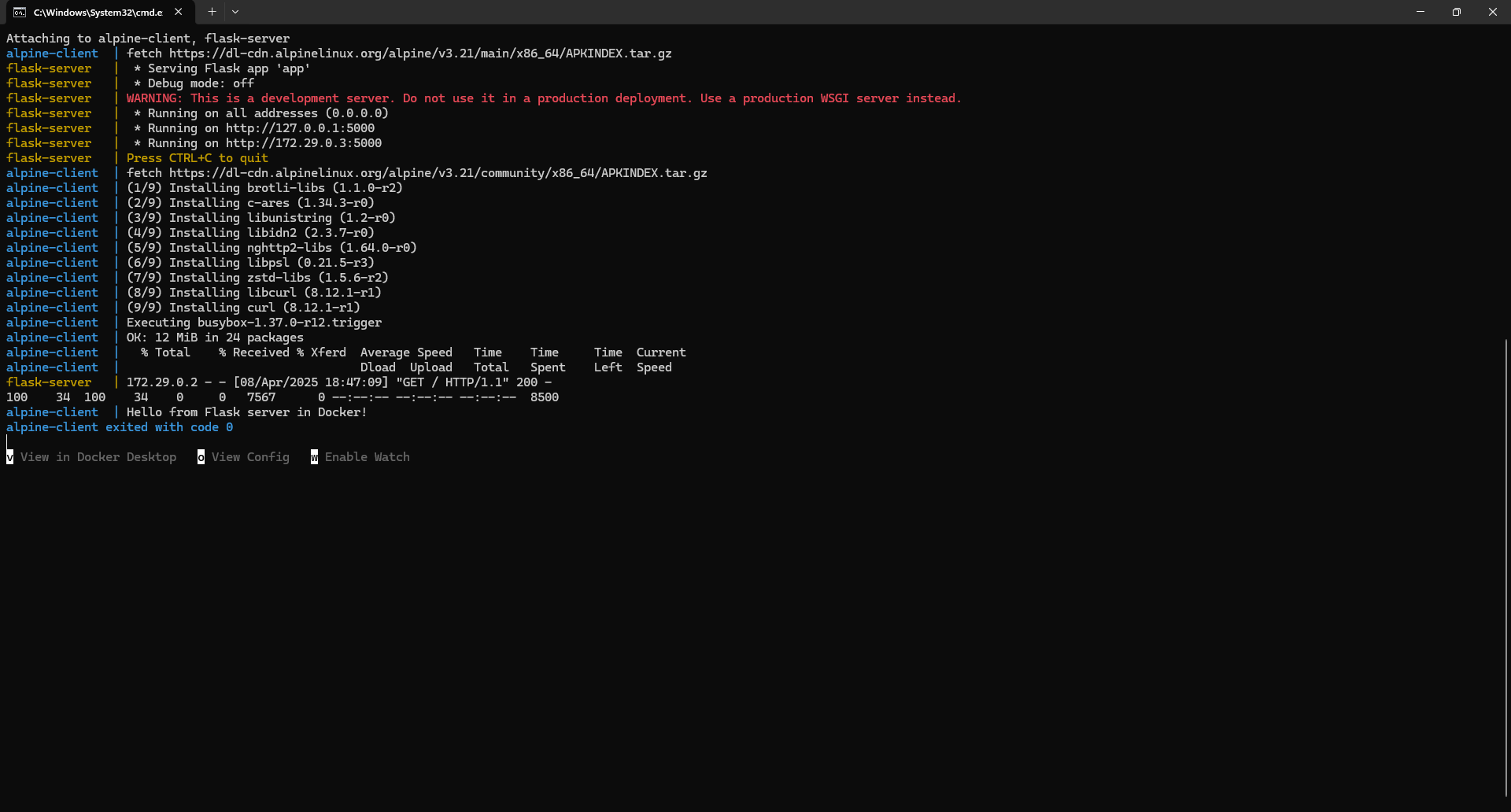


http://localhost:8080/

Bài 14: Cấu hình mạng riêng giữa các container

Yêu cầu:

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

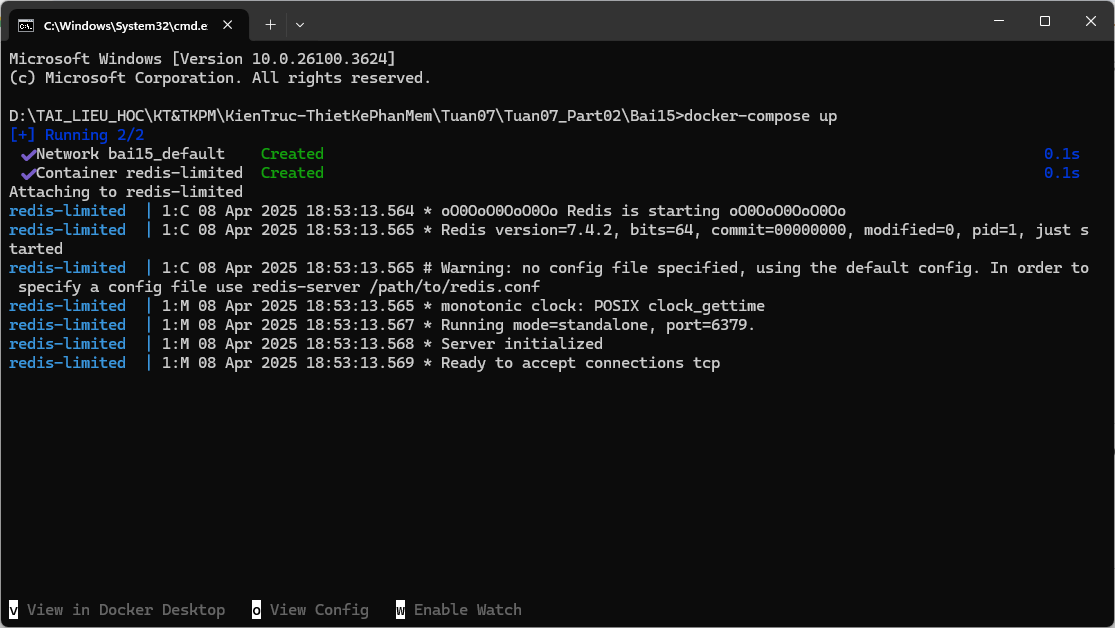
alpine-client | Hello from Flask server in Docker!

* Đây là nội dung trả về từ Flask server, nghĩa là **container client đã truy cập được container server** qua tên service trong mạng riêng.

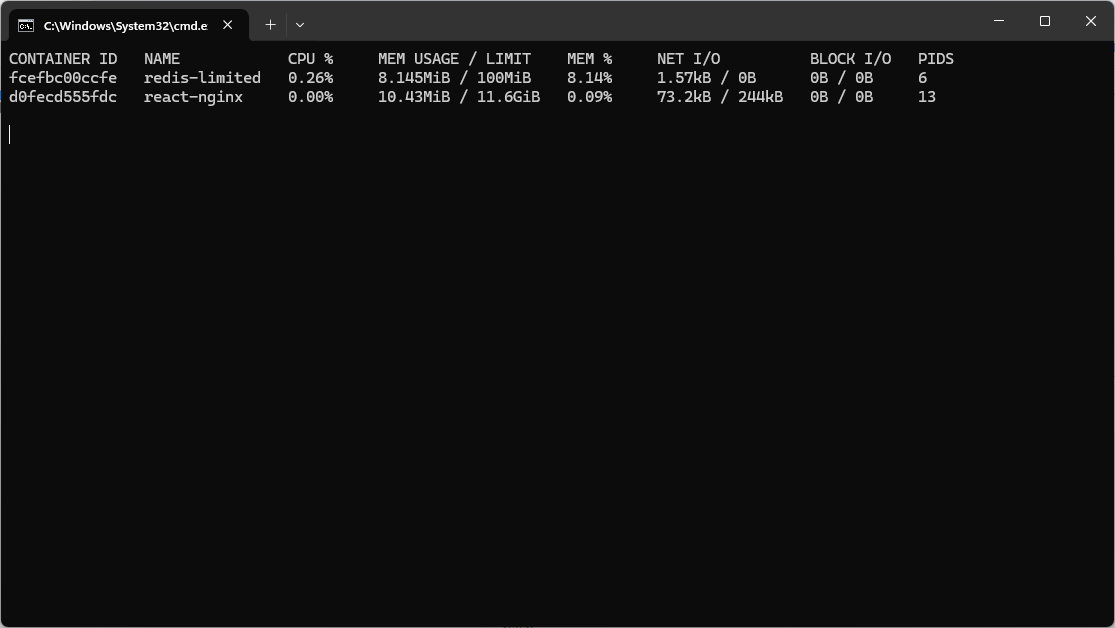
Bài 15: Giới hạn tài nguyên cho container

Yêu cầu:

Giới hạn CPU và RAM cho một container Redis.

mem\_limit: 100m         # RAM giới hạn 100MB

cpus: 0.5               # CPU giới hạn 50% 1 core



**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.