**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.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Nội dung do AI tạo ra có thể không chính xác.

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).

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác. Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Nội dung do AI tạo ra có thể không chính xác.

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).

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

2. 1 service Nginx (port 8080) cấu hình làm reverse proxy: Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

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

3. Tạo custom network và Nginx config.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

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.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

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

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

1. Volume để lưu dữ liệu Prometheus và Grafana.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

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).