## **Phần 1: Các lệnh cơ bản thao tác với Docker**

1. **docker --version – Kiểm tra phiên bản Docker.**

****

1. **docker run hello-world – Chạy container kiểm tra Docker hoạt động đúng không.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **docker pull nginx – Tải image Nginx từ Docker Hub.**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

1. **docker images – Liệt kê các image có trong hệ thống.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **docker run -d nginx – Chạy container Nginx ở chế độ nền (detached).**

****

1. **docker ps – Xem danh sách các container đang chạy.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

1. **docker ps -a – Xem danh sách tất cả container (bao gồm cả container đã dừng).**

**A screen shot of a computer program

AI-generated content may be incorrect.**

1. **docker logs <container\_id> – Xem logs của một container.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

1. **docker exec -it <container\_id> /bin/sh – Truy cập vào shell của container.**

****

1. **docker stop <container\_id> – Dừng một container đang chạy.**

****

1. **docker restart <container\_id> – Khởi động lại container.**

**A screen shot of a computer

AI-generated content may be incorrect.**

1. **docker rm <container\_id> – Xóa container đã dừng.**

****

1. **docker container prune – Xóa tất cả container đã dừng.**
2. **docker rmi <image\_id> – Xóa một image Docker.**
3. **docker image prune -a – Xóa tất cả image không dùng đến.**
4. **docker run -d -p 8080:80 nginx – Chạy Nginx và ánh xạ cổng 8080 của máy host sang cổng 80 của container.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **docker inspect <container\_id> – Xem thông tin chi tiết của container.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. **docker run -d -v mydata:/data nginx – Tạo volume mydata và gắn vào /data trong container.**
2. **docker volume ls – Liệt kê các volume Docker.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **docker volume prune – Xóa tất cả volume không sử dụng.**
2. **docker run -d --name my\_nginx nginx – Chạy container Nginx với tên my\_nginx.**

**A black screen with white text

AI-generated content may be incorrect.**

1. **docker stats – Giám sát tài nguyên (CPU, RAM) của container.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

1. **docker network ls – Liệt kê các mạng Docker.**

**A screen shot of a computer

AI-generated content may be incorrect.**

1. **docker network create my\_network – Tạo một mạng Docker tùy chỉnh.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. **docker run -d --network my\_network --name my\_container nginx – Chạy container trong mạng my\_network.**
2. **docker network connect my\_network my\_nginx – Kết nối container my\_nginx vào my\_network.**
3. **docker run -d -e MY\_ENV=hello\_world nginx – Chạy container với biến môi trường MY\_ENV=hello\_world.**
4. **docker logs -f my\_nginx – Xem logs liên tục của container my\_nginx.**
5. **FROM nginx (Dockerfile) – Sử dụng Nginx làm image cơ sở trong Dockerfile.**

**COPY index.html /usr/share/nginx/html/index.html – Sao chép file index.html vào thư mục web của Nginx.**

1. **docker build -t my\_nginx\_image . – Build image mới từ Dockerfile, đặt tên là my\_nginx\_image.**
2. **docker run -d -p 8080:80 my\_nginx\_image – Chạy container từ image my\_nginx\_image, ánh xạ cổng 8080.**

## **Phần 2: Thao tác với Dockerfile**

### **Bài 1: Tạo Dockerfile chạy một ứng dụng Node.js đơn giản**

**Yêu cầu:**

* Viết **Dockerfile** để chạy một ứng dụng Node.js hiển thị "Hello, Docker!" trên cổng 3000.
* Sử dụng node:18 làm base image.

A screenshot of a computer

AI-generated content may be incorrect.

### **Bài 2: Tạo Dockerfile chạy một ứng dụng Python Flask**

**Yêu cầu:**

* Viết Dockerfile để chạy một ứng dụng Flask hiển thị "Hello, Docker Flask!" trên cổng 5000.
* Sử dụng python:3.9 làm base image.

A screenshot of a computer

AI-generated content may be incorrect.

### **Bài 3: Tạo Dockerfile chạy một ứng dụng React**

**Yêu cầu:**

* Viết Dockerfile để build và chạy một ứng dụng React.
* Sử dụng node:18-alpine làm base image.

A screenshot of a computer

AI-generated content may be incorrect.

### **Bài 4:** **Tạo Dockerfile chạy một trang web tĩnh bằng Nginx**

Yêu cầu:

Tạo một file index.html đơn giản và sử dụng nginx:latest để phục vụ trang web.

### **Bài 5: Tạo Dockerfile cho ứng dụng Go**

Yêu cầu:

Viết Dockerfile để build và chạy một ứng dụng Go đơn giản.

### **Bài 6: Sử dụng Multi-stage Build trong Dockerfile**

Viết Dockerfile để build một ứng dụng Node.js với hai stage:

Stage 1: Dùng node:18 để build code.

Stage 2: Dùng node:18-alpine để chạy ứng dụng đã build.

### **Bài 7: Sử dụng biến môi trường trong Dockerfile**

Yêu cầu:

Viết Dockerfile cho ứng dụng Python đọc biến môi trường APP\_ENV và in ra màn hình.

Sử dụng ENV APP\_ENV=development trong Dockerfile.

### **Bài 8: Tạo Dockerfile cho PostgreSQL tùy chỉnh**

Yêu cầu:

Viết Dockerfile để chạy PostgreSQL (postgres:15).

Thêm file SQL để tự động tạo database khi container chạy lần đầu tiên.

### **Bài 9: Tạo Dockerfile chạy Redis với cấu hình tùy chỉnh**

Yêu cầu:

Viết Dockerfile sử dụng redis:latest.

Thêm file redis.conf vào container.

### **Bài 10: Chạy ứng dụng PHP với Apache**

Yêu cầu:

Viết Dockerfile để chạy một ứng dụng PHP đơn giản (php:8.2-apache).

Mount mã nguồn từ máy host vào container.

## **Phần 3: Thao tác với Dockerfile 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).