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

1. docker –version

- Xem phiên bản Docker hiện tại đang cài trên máy .

A screenshot of a computer

AI-generated content may be incorrect.

1. docker run hello-world

* Nếu image hello-world chưa có trên máy, Docker sẽ tải về từ Docker Hub.
* Chạy container từ image hello-world.
* Container thực thi một chương trình đơn giản in ra “Hello from Docker!” để xác nhận Docker đã cài đặt thành công.

A screenshot of a computer

AI-generated content may be incorrect.

1. Docker pull nginx

* Lệnh này dùng để tải image nginx từ Docker Hub về máy.
* Mặc định, nếu không chỉ định phiên bản (tag), Docker sẽ tải phiên bản latest.
* Nếu image đã có trên máy, Docker sẽ kiểm tra và chỉ tải các phần mới nhất.

A screenshot of a computer

AI-generated content may be incorrect.

1. Docker images

- Lệnh này hiển thị danh sách các imageDocker hiện có trên máy.

A black background with white letters and numbers

AI-generated content may be incorrect.

1. Docker run –d nginx

- Lệnh này khởi chạy container Nginx ở chế độ detached (-d).

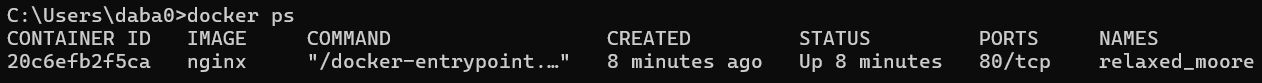
- Container sẽ chạy ở nền và bạn sẽ không thấy nó trực tiếp trên terminal.

- Docker trả về ContainerID, nghĩa là Nginx đã chạy thành công



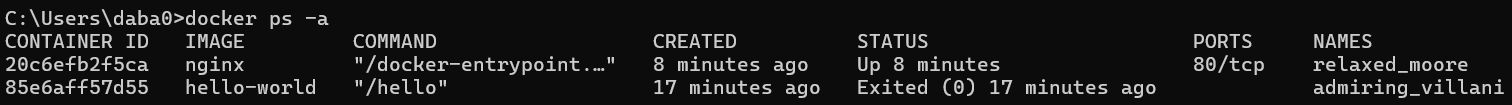
1. Docker ps

- Hiển thị danh sách các container đang chạy, bao gồm ID, image, lệnh khởi chạy, trạng thái và cổng.



1. Docker ps –a

- Hiển thị tất cả container, bao gồm cả container đã dừng.



1. Docker logs <container\_id>

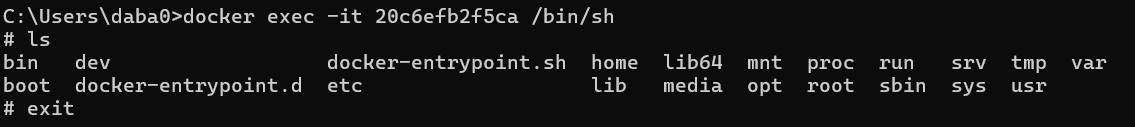
- Hiển thị log (nhật ký) của container, giúp kiểm tra quá trình khởi động và hoạt động của nó.

A screen shot of a computer

AI-generated content may be incorrect.

1. Docker exec –it <container\_id> /bin/sh

* Truy cập vào terminal của container để chạy lệnh bên trong.
* Nếu container chạy Ubuntu/Debian, có thể dùng /bin/bash.
* Để thoát ra thì nhập “exit”.



1. docker stop <container\_id>

- Dừng container có ID 20c6efb2f5ca.



1. docker restart <container\_id>

- Khởi động lại container có ID 20c6efb2f5ca.



1. docker rm <container\_id>

- Xóa container có ID 85e6aff57d55 ( phải dừng container thì mới xóa được).



1. docker container prune

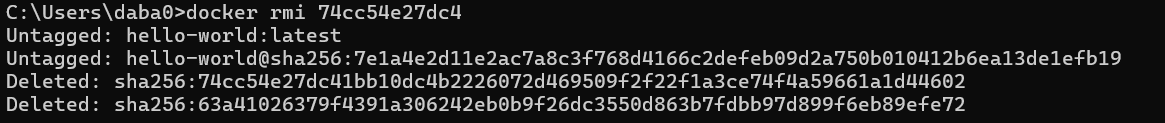
- Xóa tất cả container đã dừng để dọn dẹp hệ thống.

A black background with white text

AI-generated content may be incorrect.

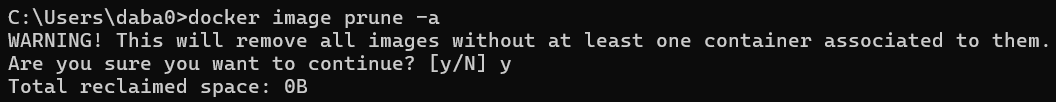
1. docker rmi <image\_id>

- Xóa image hello-world khỏi hệ thống, bao gồm cả các tag và dữ liệu liên quan.



1. docker image prune –a

- Xóa tất cả image không được sử dụng bởi container nào, nhưng không có image nào để xóa.



1. docker run -d -p 8080:80 nginx

- Chạy container Nginx ở chế độ nền (-d), ánh xạ cổng 8080 trên máy host với cổng 80 trong container.



1. docker inspect <container\_id>

- Hiển thị thông tin chi tiết về container 4865b3d7c8ec, bao gồm trạng thái, cấu hình và ID image.

A computer screen shot of white text

AI-generated content may be incorrect.

1. docker run -d -v mydata:/data nginx

- Chạy container Nginx ở chế độ nền (-d), gắn volume tên mydata vào thư mục /data trong container.



1. docker volume ls

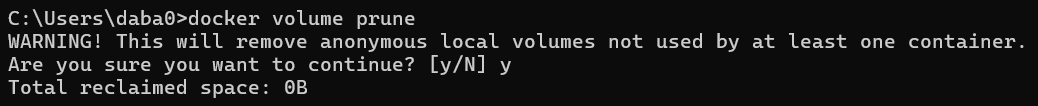
- Hiển thị danh sách các volume Docker hiện có, bao gồm volume mydata.

A black background with white text

AI-generated content may be incorrect.

1. docker volume prune

- Xóa các volume không còn được sử dụng, nhưng không có volume nào để xóa.



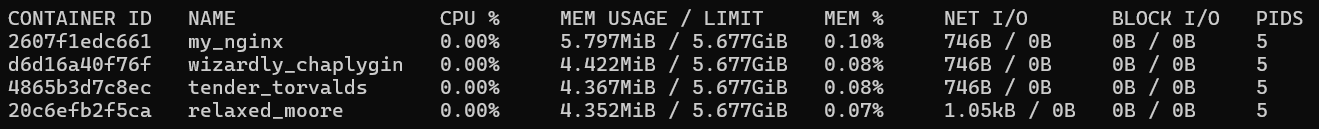
1. docker run -d --name my\_nginx nginx

- Chạy container Nginx ở chế độ nền (-d) với tên tùy chỉnh là my\_nginx.



1. docker stats

- Hiển thị thông tin thời gian thực về mức sử dụng CPU, RAM, mạng và I/O của các container đang chạy.



1. docker network ls

- Hiển thị danh sách các mạng Docker hiện có, bao gồm các mạng mặc định (bridge, host, none) và mạng tùy chỉnh (ls, my\_network).

A black screen with white text

AI-generated content may be incorrect.

1. docker network create my\_network

- Tạo thành công mạng Docker my\_network.



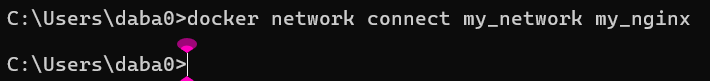
1. docker run -d --network my\_network --name my\_container nginx

- Chạy container my\_container với Nginx và kết nối nó vào mạng my\_network.



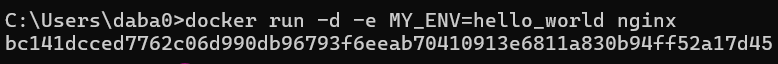
1. docker network connect my\_network my\_nginx

- Kết nối container my\_nginx vào mạng my\_network.



1. docker run -d -e MY\_ENV=hello\_world nginx

- Chạy container nginx và đặt biến môi trường MY\_ENV=hello\_world.



1. docker logs -f my\_nginx

-  Theo dõi log của container my\_nginx theo thời gian thực (-f giữ log chạy liên tục).

A screenshot of a computer program

AI-generated content may be incorrect.

1. FROM nginx   
   COPY index.html /usr/share/nginx/html/index.html

A screenshot of a computer

AI-generated content may be incorrect.

* + Tạo thư mục chứa file index.html và file Dockerfile

1. docker build -t my\_nginx\_image .
   * Chạy lệnh “docker build -t my-nginx .” để build image từ Dockerfile

A computer screen with blue and white text

AI-generated content may be incorrect.

1. docker run -d -p 8080:80 my\_nginx\_image
   * Chạy lệnh “docker run -d -p 8080:80 my-nginx” ở chế độ nền (-d) để ánh xạ cổng 8080 của máy sang cổng 80 của container.
   * Truy cập <http://localhost:8080/> ở trình duyệt, nếu thành công, sẽ thấy trang “**Welcome to My Nginx Server!”.**



A screenshot of a computer

AI-generated content may be incorrect.

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.

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.

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.

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

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.