|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phần 1:** |  | **Các lệnh cơ bản thao tác với Docker** | | |
|  | 1 | docker --version |  | Kiểm tra phiên bản docker |
|  | 2 | docker run hello-world |  | Chạy image hello-world, chưa có thì pull về |
|  | 3 | docker pull nginx |  | Lấy image nginx từ hub.docker |
|  | 4 | docker images |  | Kiểm tra docker images |
|  | 5 | docker run -d nginx |  | Chạy nền |
|  | 6 | docker ps |  | Những container đang chạy |
|  | 7 | docker ps -a |  | Tất cả những container đã và đang chạy gồm trạng thái |
|  | 8 | docker logs <container\_id> |  | Xem log của container |
|  | 9 | docker exec -it <container\_id> /bin/sh |  | Truy cập sh của container |
|  | 10 | docker stop <container\_id> |  | Dừng container |
|  | 11 | docker restart <container\_id> |  | Khởi động lại container |
|  | 12 | docker rm <container\_id> |  | Xóa container |
|  | 13 | docker container prune |  | Xóa tất cả container không sử dụng |
|  | 14 | docker rmi <image\_id> |  | Xóa image |
|  | 15 | docker image prune -a |  | Xóa những image không sử dụng bơi bất kì container nào |
|  | 16 | docker run -d -p 8080:80 nginx |  | Chạy nginx port 8080 dạng daemon |
|  | 17 | docker inspect <container\_id> |  | Xem thông tin của container gồm các trạng thái,… |
|  | 18 | docker run -d -v mydata:/data nginx |  | Mount volume là thư mục mydata (ở máy thật) và /data ở docker |
|  | 19 | docker volume ls |  | Liệt kê những ổ |
|  | 20 | docker volume prune |  | Giải phóng ổ không sử dụng |
|  | 21 | docker run -d --name my\_nginx nginx |  | Đặt tên cho container khi chạy thay vì để mặc định. |
|  | 22 | docker stats |  | Thống kê các tài nguyên như mem, cpu mà container đang dùng |
|  | 23 | docker network ls |  | Xem các network driver đang có |
|  | 24 | docker network create my\_network |  | Tạo mạng tên my\_network |
|  | 25 | docker run -d --network my\_network --name my\_container nginx |  | Chạy nginx trên network my\_network |
|  | 26 | docker network connect my\_network my\_nginx |  | Kết nối mạng my\_network vào ny\_nginx để cho nó giao tiếp với nhau |
|  | 27 | docker run -d -e MY\_ENV=hello\_world nginx |  | Chạy nginx với biến môi trường MY\_ENV |
|  | 28 | docker logs -f my\_nginx |  | Xem logs của mạng my\_nginx theo thời gian thực |
|  | 29 | FROM nginx |  | Chạy dockerfile -> Copy index.html vào /usr/share/nginx/html/index.html sau đó build với tên my\_custom\_nginx Chạy port 8080 |
|  | COPY index.html /usr/share/nginx/html/index.html |
|  | 30 | docker build -t my\_custom\_nginx |
|  | 31 | docker run -d -p 8080:80 my\_custom\_nginx |
|  |  |  |  |  |
| **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. | FROM node:18  COPY app.js /usr/nodeapp/app.js  WORKDIR /usr/nodeapp/app.js  CMD ["node", "app.js"]  // app.js  const http = require("http");  const app = http.createServer((req, res) => {      res.end(          JSON.stringify({              status: 200,              message: "OK"          })      )  })  app.listen(3000, () => {      console.log("Server is running on port 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 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.* |  |  |