I, Docker

1. Docker là gì?

* Docker là một nền tàng cung cấp cách building và deploying và running ứng dụng một cách dễ dàng hơn bằng cách sử dụng các container. Các container cho phép lập trình viên đúng gói ứng dụng với tất cả thành phần cần thiết như thư viện và cấc phụ thuộc khác. Bằng cách đó ứng dụng có thể chạy trên mọi nền tảng cho dù cấu hình có khác nhau.
* Dựa vào sự tiện lợi đó có thể đóng gói ứng dụng và chạy ở tất cả mọi nơi
* Nói tóm lại: docker docker giúp ứng dụng depoloyng và runniung một cách dễ dàng nhất

III, docker file

1. Docker file là gì?

* Docker file Dockerfile là một file dạng text không có phần đuôi mở rộng, chứa các đặc tả về một trường thực thi phần mềm, cấu trúc cho Docker Image. Từ những câu lệnh đó, Docker sẽ build ra Docker image (thường có dung lượng nhỏ từ vài MB đến lớn vài GB).
* Docker file tự động hóa quá trình xây dụng images, đảm bảo rằng môi trường chạy container có thể tái tạo và mô tả được. mỗi lệnh docker file tạo ra một layer trong iamges giúp quản lý tài nguyên và tăng tốc độ xây dựng.

1. Hiểu các viết docker file để xây dụng iamges

# sử dụng một images cơ bản từ docker hub

FROM ubuntu:20.04

#mô tả thông tin người tạo images

LABEL maintainer = “your name <your email>”

#cài đặt các dependency và ứng dụng

RUN apt-get update && \

Apt-get install -y nginx && \

Apt-get clean && \

Rm -rf /var/lib/apt/list/\*

#copy các lệnh cần thiết từ máy host vào container

COPY index.html/ var/www/html/

#mở cổng cho ứng dụng

#EXPOSE 80

# đặt lệnh mặc định khi chạy container

CMD [“nginx ”, “-g”, “demo off;”]

1. Hiểu các thành phần của một images
2. FROM

* Chỉ định rằng image nào sẽ được dùng làm image cở sở để build image customize. Các image base này thường được tải về từ public hoặc private ( vd: openjdk17:latest )
* FROM <<images\_name>>:<<tag>>: tag ở đâyu là các phiên bản nếu không chỉ định tag mặc định sẽ là latest

1. LABEL

* Chỉ định các metadat vào image, giúp mô tả, xác định và tổ chức.
* Label được lưu dưới dạng key-value, có thể lưu nhiều label nhưng key-value phải là độc nhất
* LABEL <key>=<value> <key>=<value>

1. RUN

* Chỉ thị run dùng để chạy một lệnh nào đó trong quá trình build image và thường là các câu lệnh linix. Tùy vào image gốc được khai báo trong phần FROM thì sẽ có các câu lệnh tương ứng.
* RUN <command>

1. ADD

* Chỉ thị add thực hiện sao chép các tập, thư mục từ máy đang build hoặc remote file urls từ src và thêm chúng vào filesystem của image dest.
* ADD <src> … <dest>: trong đó src: có thể khai báo nhiều file, thư mục. dest phải là đường dẫn tuyệt đối hoặc có quan hệ chỉ thị đối với WORKDIR

1. COPY

* Chỉ thị copy cũng giống với ADD là copy file. Thư mục từ src và thêm chúng vào dest của container. Khác với ADD, nó không hỗ trợ thêm các file remote, file urls từ các nguồn trên mạng
* COPY <src> … <dest>

1. ENV

* Chỉ thị ENV dùng để khai báo các biến môi trường. các biến này được khai bao dưới dạng key-value bằng các chuỗi. giá trị của các biến này sẽ có hiện hữu cho các chỉ thị tiếp theo của dockerfile.
* ENV <key>=<value>

1. CMD

* Chỉ thị CMD định nghĩa các câu lệnh sẽ dược chạy sau khi container được khởi động từ image đã build. Có thể khai báo được nhiều nhưng chỉ có duy nhất CMD cuối cùng được chạy
* CMD echo vible

III, docker images

1. Docker image

* Docker image là một phiên bản chụp (snapshot) của một ứng dụng cùng với môi trường chạy và tất cả các dependency cần thiết để ứng dụng đó chạy.
* Image là đơn vị đóng gói độc lập, có thể chia sẻ và triển khai trên nhiều môi trường khác nhau mà không cần phải cài đặt môi trường.
* Image có thể được xây dựng bằng dockerfile, một tập chứa các hướng dẫn để tạo image

1. Tạo image

* Sử dụng lệnh docker build để xây images:
* docker build -t your-image-name:tag .

1. chia sẻ image:

* sử dụng lệnh docker login để đăng nhập vào docker hub
* gán tag và push image: docker tag local-image your-dockerhub-username/your-imaga:tag
* đẩy image lên docker hub: docker push your-dockerhub-username/your-image-name:tag

1. quản lý images

* liệt kê images đang tồn tại trên may: docker images
* xóa images: docker rmi image\_id
* tìm kiếm images: docker search image\_name

IV, docker containers

1. container là gì?

* Docker container là một đối tượng chứa thực thi của một image ( là image khi chạy ). Container là một môi trường đóng gói độc lập chứa tất cả những gì cần thiết để một ứng dụng chạy. bao gồm mã nguồn, runtime, thư viện, và các tài nguyên hệ thống khác.

1. Chạy container.

* Docker run -d –name <tên container> <tên image>: chạy từ image
* Docker container ls: xem tất cả có trong máy
* Docker container ps: xem danh sách các container đang chạy.
* Docker container đã dừng + đagn chạy.
* Docker rm <tên hoặc id container>

V, docker volumn

1. Volume là gì?

* Là cách để lưu trữ và quản lý dữ liệu bên trong các container docer một cách linh hoạt và bảo đảm tính liên tục của dữ liệu. khi một container docker được tạo, nó có thể tạo và sử dụng các Docker volume để lưu trữ dữ liệu, và các volume này có thể tồn tại và độc lập với các container.

+ độc lập với container: có nghĩa là giữ liệu được giữ lại ngay cả khi container đã đsong hoặc bị xóa

+ chia sẻ dữ liệu giữa các container: các container có thể chia sẻ volume, giúp chúng truy cập và cập nhật dữ liệu chung

+ là cách tiện lợi để quản lý dữ liệu liên quan đến các ứng dụng trong container.

+ có thể dùng để lưu trữ tập tin đặc biệt, chẳng hạn như cơ sở dữ liệu hoặc tệp cấu hình.

* Docker volume create, docker volume ls, docker volume rm…

1. Câu lệnh

* Chạy container liên kết với volume tạo mới
* Docker run -d –name mycontainer -v mydata:/path/to/container/mount/point myimage
* Docker volume create <tên volume>: tạo volume
* Docker volume ls: xem danh sách volume trên máy
* Docker volume inspect <tên volume>: xêm chi tiết volume
* Docker volume rm <tên volume>: xóa volume
* Docker volume prune: < xóa tất cả volume không liên kết với container nào
* Docker run -d –name <tên container> -v <tên volume>: /đường dẫn trong container

VI, docker network

1. Network là gì?

* Là một phần ucar docker có nhiệm vujq uản lý và kết nối các container với nhau haowjc các dihcj vụ bên ngoại. nó cho phép các container trong một ứng dụng có thể giao tiếp với nhau hoặc kết nối đến các nguồn tài nguyên mạng bên ngoài một cách linh hoạt và an toàn

+ isolation (cô lập): mỗi container có thể nằm trong một network độc lập, giúp cô lập và bảo vệ chúng khỏi các container khác.

\_ connectivity (kết nối): docker network cho phép các container kết nối với nhau, cung cấp khả năng giao tiếp giữa chúng thông qua tên container haowjc DNS.

+ external connectivity: các container có thể kết nối với các nguồn tài nguyên bên ngoài mạng docker, chẳng hạn nhưu internet hoặc các dịch vụ bên ngoài.

+ user-defined networks (mạng do người dùng định nghĩa): docker cho phép tạo ra các network do người dùng định nghĩa để quản lý các kết nối theo cách tùy chỉnh.

1. Câu lệnh

* Xem danh sách: docker network ls
* Gắn kết network vào container đagn chạy: docker run -d --name tên-container --network=tên-network tên-image
* Ngắt kết nối container từ một network: docker network disconnect <ten network> <ten container>

VII, docker compose

1. Docker compose là một công cụ giúp quản lý và triển khai ứng dụng đa container trong docker. Nó chophesp bạn định nghĩa cấu trúc của ứng dụng, bao gồm các container, network, volumes, và các cài đặt khác, trong một tệp yaml duy nhất. sau đó bạn có thể sử dụng moojet lệnh đơn để triển khai và quản lý toàn bộ ứng dụng.
2. Câu lệnh

* Tạo và cahyj ứng dụng từ tệp yaml: docker-compose up
* Tạo và chạy ứng dụng ở chế độ nền: docker – compose up -d
* Dừng ứng dụng: docker -compose down
* Hiển thị log của các contaienr: docker-conpose logs
* Xem danh sách các conatiner đang chạy: docker -compose ps

1. File yaml demo



VIII, các lệnh viết tắt của docker

IX, docker với java

X, docker hub và docker registry

5. Debugging và Logging:

docker logs: Đọc logs từ các containers.

docker exec: Thực hiện lệnh trong một container đang chạy để kiểm tra và debug.

6. Dockerfile Best Practices:

Layers: Hiểu về layers và cách chúng ảnh hưởng đến hiệu suất và kích thước của images.

Caching: Sử dụng caching hiệu quả trong quá trình build.

8. Các Thực Hành Thêm:

Build và Deploy Ứng dụng: Thực hành xây dựng một ứng dụng và triển khai nó sử dụng Docker.

Multi-Container Applications: Xây dựng và quản lý ứng dụng với nhiều containers liên quan.