Docker

▼ Khái Niệm

Docker là một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang.

image: Docker image là một file bất biến - không thay đổi, chứa các source code, libraries, dependencies, tools và các files khác cần thiết cho một ứng dụng để chạy.

container: Docker container là một run-time environment mà ở đó người dùng có thể chạy một ứng dụng độc lập. Những container này rất gọn nhẹ và cho phép bạn chạy ứng dụng trong đó rất nhanh chóng và dễ dàng.

▼ Các lệnh Docker

▼ docker ps

Dùng để liệt kê ra các container đang chạy. Khi sử dụng với các tham số

▼ docker pull

Hầu hết các image sẽ được tạo dựa trên các image cơ sở từ Docker Hub. Docker Hub chứa rất nhiều các image được dựng sẵn, mà ta có thể pull về và dùng mà không cần phải định nghĩa và cấu hình lại từ đầu. Để tải một image cụ thể hoặc một tập hợp image ta dùng docker pull.

docker pull nginx
docker pull mysql

▼ docker build

Lệnh này dùng để build một image từ Dockerfile và context. Context ở đây là một tập file được xác đinh bởi đường dẫn hoặc url cụ thể. Ta có thể sử dụng thêm tham số -t để gắn nhãn cho image.

dùng để tạo image từ dockerfile

```
docker build -t your_name_container
```

▼ docker run

Lệnh này dùng để chạy một container dựa trên một image mà ta có sẵn. Ta có thể thêm vào sau lệnh này một vài câu lệnh khác như -it bash để chạy bash từ container này.

```
docker run image_name -it bash
```

▼ docker logs

Lệnh này được sử dụng để hiển thị logs của một container, ta cần phải chỉ rõ container để hiển thị logs thông qua tên của nó. Ngoài cũng có thể sử dụng thêm một số flag như --follow để giữ việc theo dõi logs.

```
docker logs --follow your_name_container
```

▼ docker volume ls

Lệnh này dùng để liệt kê ra các volumn mà các container sử dụng, volume là một cơ chế dùng để lưu trữ dữ liệu sinh ra và sử dụng bởi Docker.

▼ docker rm

Lệnh này dùng để xóa một hoặc nhiều container.

```
docker rm <list_container_name_or_id>
```

▼ docker rmi

Lệnh này dùng để xóa một hoặc nhiều images.

```
docker rmi <list_image_id>
```

▼ docker stop

Lệnh này dùng để stop một hoặc nhiều container. Ngoài ra ta có thể dung docker kill để bắt buộc container dừng lại.

docker stop <list_container_name_or_id>

▼ **Sử dụng Docker trong dự án NodeJS**

▼ **Bước 1**

Tạo file Dockerfile cùng thư mục với app.js có nội dung:

FORM node:9-slim #phiên bản image nodejs
WORKDIR /app   #thư mục làm việc
COPY package.json /app   # copy package.json vào thư mục làm việc
RUN npm install   # chạy npm install để tải các thư viện
COPY . /app   hoặc COPY . . # copy thư mục hiện tại
CMD ["npm","start"] // các lệnh

▼ **Bước 2**

Trong terminal chạy lệnh docker build -t your_name_container document_name

▼ **Bước 3**

sau đó chạy lệnh docker run image_name -it

có thể thêm các lệnh như -p 9000:3000 để thay đổi cổng

▼ **Docker compose**

▼ Khái niệm

Docker compose là công cụ để chúng ta có thể định nghĩa và chạy multi-container trong Docker application.

▼ Cách tạo Docker compose

▼ **tạo file docker-compose.yml**

bao gồm:

version: “3.7”

service:

tên service:

build: ./đường dẫn tới thư mục chứa project

ports:

-ví dụ 3000:3000

volumes:

-./:/app

environment:

MYSQL_HOST: mysql
MYSQL_USER: root
MYSQL_PASSWORD: secret
MYSQL_DB: todos

mysql:
image: mysql:5.7
volumes:
- ./được khai báo bên dưới:/var/lib/mysql
environment:
MYSQL_ROOT_PASSWORD: secret
MYSQL_DATABASE: todos

volumes:

- được khai báo bên dưới:

sau đó chạy lệnh `docker-compose up -d`

▼ Docker networking

▼ Khái Niệm

Docker network sẽ đảm nhiệm nhiệm vụ kết nối mạng giữa các container với nhau, kết nối giữa container với bên ngoài, cũng như kết nối giữa các cụm (swarm) docker containers

▼ Tại sao Docker networking lại quan trọng

Trong thực tế, phần lớn các dự án mà ta sẽ phải làm việc sẽ có nhiều hơn một container. Container là môi trường cô lập vì vậy không thể kết nối, các container hoàn toàn cách biệt với nhau và không để ý đến sự tồn tại của nhau. **Vì vậy, làm thế nào để kết nối hai container? Đó sẽ không phải là một thách thức?**

Vì vậy phải kết nối các bằng cách đặt chúng dưới một mạng cầu nối do người dùng xác định và Docker network sẽ giúp chúng ta thực hiện điều đó .

▼ Các trình điều khiển mạng của Docker

Theo mặc định, Docker có năm trình điều khiển mạng. Chúng như sau:

- `bridge` - Trình điều khiển mạng mặc định trong Docker. Nó có thể được sử dụng khi nhiều container đang chạy ở chế độ tiêu chuẩn và cần giao tiếp với nhau.
- `host` - Loại bỏ hoàn toàn cách ly mạng. Bất kỳ container nào chạy trong một mạng `host` về cơ bản đều được gắn vào mạng của hệ thống máy chủ.
- `none` - Trình điều khiển này vô hiệu hóa hoàn toàn mạng cho các container. Tôi chưa tìm thấy bất kỳ trường hợp sử dụng nào cho loại mạng này.
- `overlay` - Trình điều khiển này được sử dụng để kết nối nhiều Docker daemon trên các máy tính và nằm ngoài phạm vi của hướng dẫn này.
- `macvlan` - Cho phép gán địa chỉ MAC cho các container, làm cho chúng hoạt động giống như các thiết bị vật lý trong mạng.

▼ Tại sao cần phải dùng cầu nối do người dùng xác định

Các container được gắn với mạng cầu nối mặc định có thể giao tiếp với nhau bằng địa chỉ IP điều này khá bất tiện

Tuy nhiên, cầu do người dùng xác định có một số tính năng bổ sung so với cầu mặc định:

- **Cầu nối do người dùng xác định cung cấp khả năng phân giải DNS tự động giữa các container**: Điều này có nghĩa là các container được gắn vào cùng một mạng có thể giao tiếp với nhau bằng cách sử dụng tên container. Vì vậy, nếu bạn có hai container được đặt tên `notes-api` và `notes-db`, container API có thể kết nối với container cơ sở dữ liệu bằng cách sử dụng tên `notes-db`.
- **Các cầu nối do người dùng xác định cung cấp khả năng cách ly tốt hơn**: Tất cả các container được gắn vào mạng cầu nối mặc định theo mặc định, điều này có thể gây ra xung đột giữa chúng. Việc gắn các container vào một cầu nối do người dùng xác định có thể đảm bảo cách ly tốt hơn.
- **Các container có thể được gắn vào và tách ra khỏi các mạng do người dùng xác định khi đang di chuyển**: Trong thời gian tồn tại của container, có thể kết nối hoặc ngắt kết nối nó khỏi các mạng do người dùng

xác định một cách nhanh chóng. Để xóa container khỏi mạng cầu nối mặc định, phải dừng container và tạo lại container đó với các tùy chọn mạng khác nhau.

▼ **Cách xem các mạng trên hệ thống**

sử dụng lệnh docker network ls

▼ **Cách tạo cầu nối do người dùng xác định trong Docker**

sử dụng lệnh `docker network create <network name>`

▼ **Cách gắn container vào một mạng**

```
docker network connect <network identifier> <container identifier>
```

▼ **Cách tách container khỏi mạng**

```
docker network disconnect <network identifier> <container identifier>
```

▼ **Cách xóa mạng trong Docker**

```
docker network rm <network identifier>
```

▼ **Docker volume**

1. **Khái Niệm:** Volume trong Docker được dùng để chia sẻ dữ liệu cho container.
2. **Tại sao ta lại cần Docker Volume?**

- Volumes giản hóa việc backup hoặc migrate hơn bind mount.
- Bạn có thể quản lý volumes sử dụng các lệnh Docker CLI và Docker API.
- Volumes làm việc được trên cả Linux và Windows container.
- Volumes có thể an toàn hơn khi chia sẻ dữ liệu giữa nhiều container.
- Volume drivers cho phép bạn lưu trữ volumes trên remote hosts or cloud providers, để mã hóa nội dung của volumes, hoặc thêm các chức năng khác.
- Các nội dung của volume mới có thể được điền trước bởi một container.

3. **Ta dùng Docker Volume khi nào?**

- Sử dụng volume để gắn (mount) một thư mục nào đó trong host với container.
- Sử dụng volume để chia sẻ dữ liệu giữa host và container

- Sử dụng volume để chia sẽ dữ liệu giữa các container
- Backup và Restore volume.