

Lesson 3

Introduction to Components of Docker

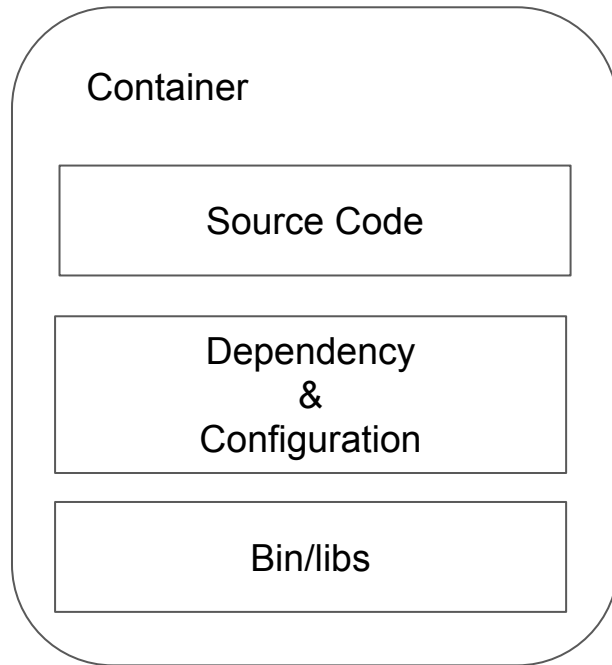
1. Tìm hiểu các thành phần trong Docker

2. Tìm hiểu về Docker Hub

3. Thực hành

1. Các thành phần trong Docker

Container: là một instance được tạo ra từ Image. Container sẽ package Source Code, Dependency, từ đó có thể Run Application.

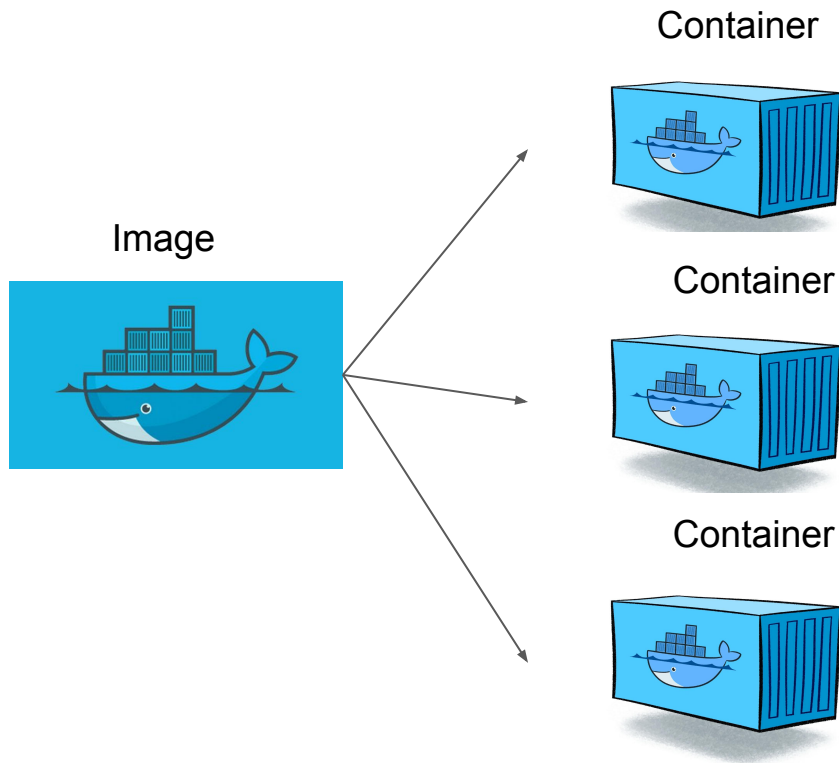


1. Các thành phần trong Docker

Image: Là một template chỉ dẫn và các thiết lập cần thiết để tạo ra Container.

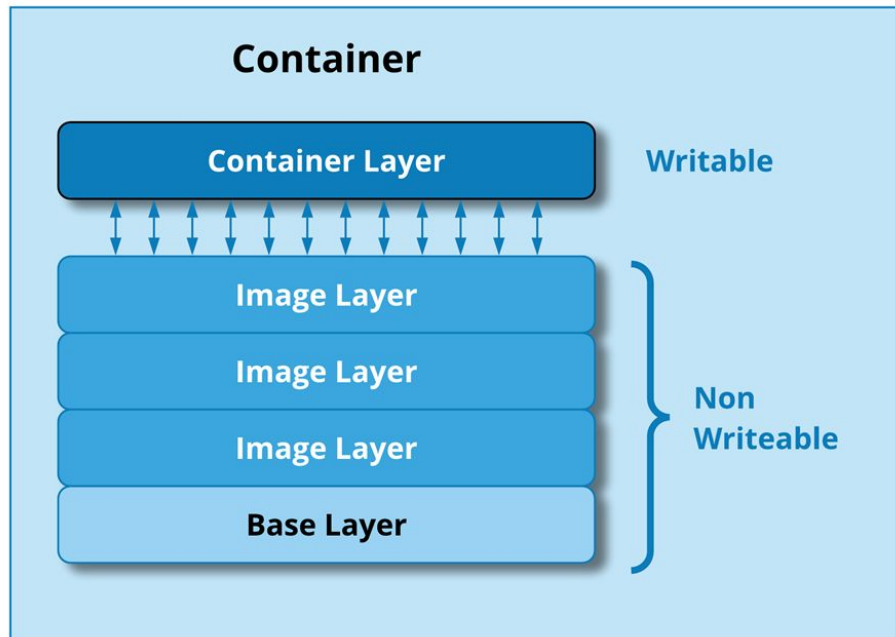
Một Image có thể được tạo ra dựa trên một Image khác có sẵn.

Image được tạo bởi Dockerfile.



Cấu trúc của Image:

- Image được tạo thành từ nhiều Image Layer khác nhau theo dạng Stack và từng Layer xếp chồng lên nhau.
- Image gốc sẽ được gọi là Base Image
- Và các Image Layer chỉ có thể Read, không thể Write.
- Khi tạo Container từ Image, thì Container có thể Write



1. Các thành phần trong Docker

Dockerfile: Là một file chứa các câu lệnh hướng dẫn và các thiết lập cần thiết.
Từ Dockerfile có thể build thành một Image

```
FROM node:16

ENV PORT 3000

# Install OS updates
RUN apt-get update && apt-get install bc

WORKDIR "/app"

# Install nodemon for use with "dev" mode
RUN npm install -g nodemon

# Bundle app source and install dependencies
COPY . /app

RUN npm install

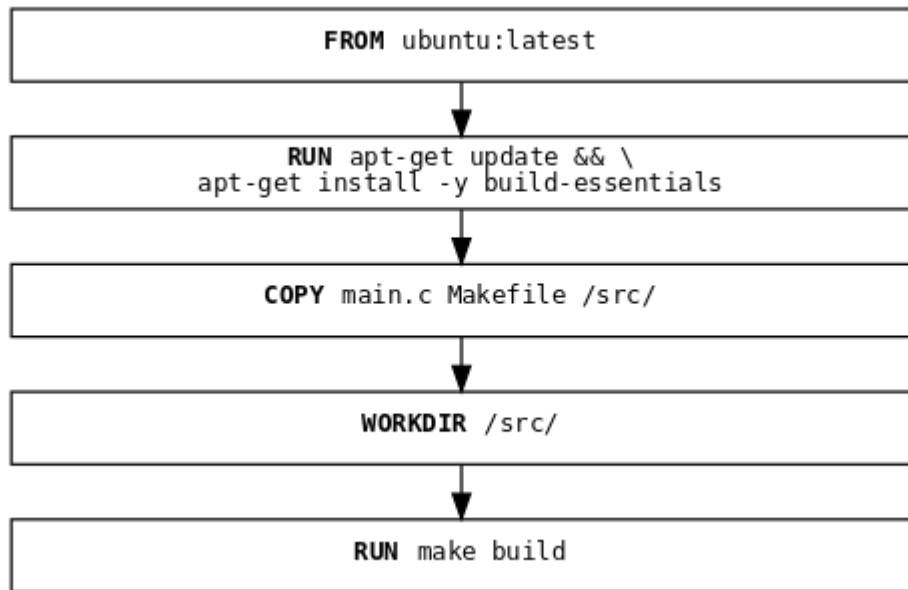
EXPOSE 3000

USER node

CMD ["/bin/bash"]
```

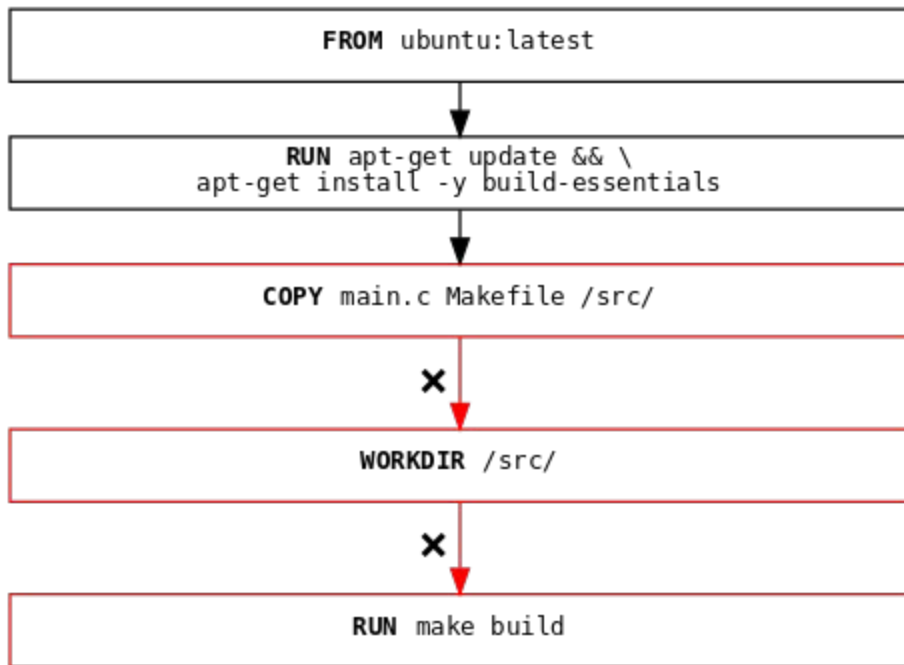
Cách hoạt động khi Build Image:

- Trong Dockerfile các **Instruction** sẽ được chạy theo thứ tự từ trên xuống dưới
- Nếu một bước **Failed**, thì toàn bộ sẽ dừng lại quá trình Build
- Mỗi một **Instruction** Docker sẽ xem là một **New Layer**



Cache Layer:

- Trong quá trình Build Image, nếu Instruction Command lớn hoặc xử lý phức tạp, có thể khiến quá trình Build mất nhiều thời gian.
- Khi Docker phát hiện một Layer có sự thay đổi, thì Docker chỉ Run Layer đó, và những Layer khác không cần Run Again mà chỉ sử dụng lại Cache trước đó.



Instruction phổ biến trong Dockerfile:

FROM:

- Được sử dụng để Build Image từ một Image được chỉ định
- Là Instruction khai báo đầu tiên trong Dockerfile
- Trong một file Dockerfile có thể có nhiều FROM Instruction để tạo nhiều Image
- Những Image phía sau có dependency với Image trước đó

Syntax: `FROM [--platform=<platform>] <image>[:<tag>] [AS <name>]`

ARG:

- Instruction duy nhất được đứng trước FROM
- Có thể khai báo nhiều ARG trong một Dockerfile
- Dùng để tạo Argument trong quá trình Build Image

Syntax: ARG <key>=<value>

ENV:

- Dùng để khai báo biến môi trường khi Build Image, và sử dụng trong Container
- Có thể khai báo nhiều ENV trong một Dockerfile
- Sử dụng trong khi Container Running

Syntax: ENV `<key>=<value>`

METADATA: LABEL, MAINTAINER

- LABEL Dùng để định nghĩa thêm các thông tin Metadata cho Container
- MAINTAINER sử dụng để thêm thông tin Author cho Container, nhưng Docker đã **Predecated**, và thay thế bởi LABEL, vì LABEL sẽ flexible hơn MAINTAINER
- Ví dụ như: Version, Author, ...

Syntax: LABEL <key>=<value> <key>=<value> ...

Những Instruction dùng để Execute Command:

RUN:

Được sử dụng để chạy các câu lệnh được khai báo trong quá trình Build Image.

CMD:

Giống như **RUN** để thực thi các câu lệnh khai báo. Instruction này sẽ Set câu lệnh thực thi mặc định cho Container, câu lệnh này chỉ được thực thi một lần duy nhất khi Run Container

ENTRYPOINT:

Giống như **CMD** để thực thi các câu lệnh khai báo, và sẽ chỉ được thực thi một lần duy nhất khi Run Container

Vậy sự khác nhau giữa CMD và ENTRYPOINT là gì ?

CMD:

Khi run Container mà không truyền Parameters ở command thì CMD sẽ được thực thi, ngược lại nếu có Parameters thì CMD sẽ bị ignored

ENTRYPOINT:

Khi run Container mà không truyền Parameters ở command thì ENTRYPOINT sẽ được thực thi, ngược lại nếu có Parameters thì ENTRYPOINT sẽ lấy Parameters làm Value cho câu lệnh ENTRYPOINT

Những Instruction dùng để copy thư mục:

ADD:

Được sử dụng để **COPY** file hoặc thư mục từ Source đến Destination của Container.

COPY:

Giống như **ADD** để **COPY** file hoặc thư mục từ Source đến Destination của Container

Vậy sự khác nhau giữa ADD và COPY là gì ?

	ADD	COPY
Có thể copy	Yes	No
Copy từ Local	Yes	Yes
Copy từ URL Remote	Yes	No
Cache Layer	No	Yes

EXPOSE:

- Dùng để chỉ định Port được mở từ Container
- Có thể chỉ định mở cổng theo mạng TCP hoặc UDP
- Mặc định sẽ là TCP

Syntax: EXPOSE <port>/<protocol>

WORKDIR:

- Được sử dụng để di chuyển tới thư mục
- Có thể khai báo nhiều WORKDIR trong Dockerfile
- Thường khai báo kèm với những Instruction như: RUN, COPY, ...

Syntax: `WORKDIR <path>`

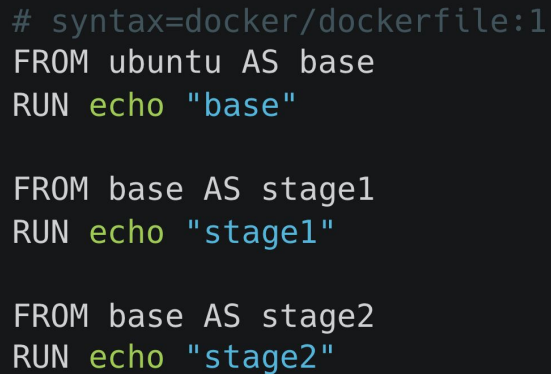
USER:

- Sử dụng để set USER mặc định trong quá trình Build Image
- Những Instruction như RUN sẽ sử dụng USER để cấp quyền execute command
- Sử dụng USER cực kì quan trọng trong vấn đề **SECURE**

Syntax: `USER <name>`

Build Multi-Stage:

Sau khi có Build Multi-Stage: Có thể sử dụng một file Dockerfile để build multi image bằng cách sử dụng multi câu lệnh FROM. Image phía sau có thể gọi tới Image trước đó để lấy data bằng Name Alias. Và câu lệnh Docker có hỗ trợ để build từng Image cụ thể theo chỉ định. Vì vậy quá trình setup được rút ngắn.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It displays a Dockerfile with three stages: 'base', 'stage1', and 'stage2'. Each stage starts with a 'FROM' instruction and followed by a 'RUN echo' command.

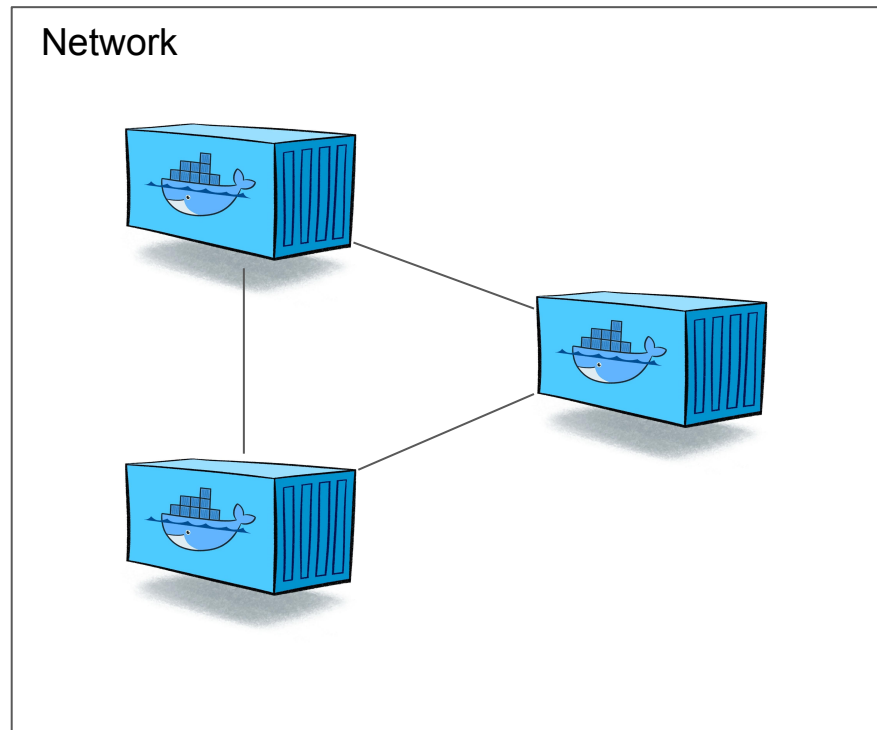
```
# syntax=docker/dockerfile:1
FROM ubuntu AS base
RUN echo "base"

FROM base AS stage1
RUN echo "stage1"

FROM base AS stage2
RUN echo "stage2"
```

1. Các thành phần trong Docker

Network: là thành phần để thiết lập mạng cho Container. Giúp cho các Container có thể giao tiếp với nhau



Host: khi khởi tạo sẽ sử dụng Network của Host Machine bằng cách là Container sẽ mở một hoặc nhiều Port và Host Machine sẽ Map Port từ Host Machine vào Port của Container, từ đó bên ngoài có thể gọi vào Container thông qua Network của Host Machine

Nên sử dụng trong trường hợp nào:

- Nếu như Container không quan tâm về địa chỉ IP, mà chỉ quan tâm về Range Port thì có thể sử dụng loại này.

Bridge: Docker sẽ có một phần mềm Bridge để gom tất cả các Container cùng một Bridge có thể giao tiếp được với nhau. Docker sẽ tạo những quyền hạn chế để tránh Container khác Bridge giao tiếp với nhau.

Nên sử dụng khi nào:

- Khi chúng ta muốn Group một số Container sử dụng chung Network với nhau, và chỉ những Container cùng chung Network có thể giao tiếp được với nhau.
- Và phạm vi hoạt động của Bridge chỉ sử dụng trong cùng một Docker Host

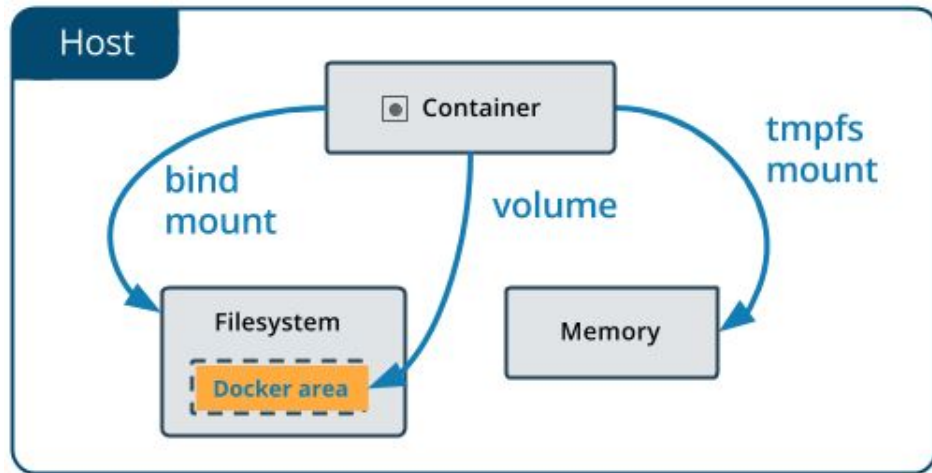
Overlay: Đơn giản Overlay sẽ giúp cho các Container khác Docker Host có thể giao tiếp được với nhau.

Nên sử dụng trong trường hợp nào:

- Khi các Container khác Docker Host giao tiếp với nhau

1. Các thành phần trong Docker

Volume: là một nơi lưu trữ và sử dụng để backup và migrate data của Container.



Bind Mount:

- Một thành phần lưu trữ xuất hiện trước Volume.
- Bind mount sẽ bị giới hạn một số tính năng so với Volume
- Performance của Bind mount sẽ phụ thuộc vào filesystem của Docker Host
- Khi mount từ Host tới Container, Bind mount sẽ sử dụng đường dẫn tuyệt đối, vì vậy nếu Docker Host không có File hay Folder thì quá trình Mount sẽ bị lỗi
- Bind mount có thể Mount tới bất kỳ thư mục nào của Docker Host

Vậy nên sử dụng khi nào:

- Nếu chúng ta muốn sử dụng chung cấu hình của Docker Host
- Có thể Mount được nhiều nơi trong Docker Host

Volume:

- Cách hoạt động của Volume cũng giống như Bind Mount, nhưng Volume sẽ được quản lý bởi Docker
- Volume có nhiều tính năng hơn so với Bind Mount,
- Volume có thể được quản lý thông qua CLI hoặc API

Vậy nên sử dụng khi nào:

- Hoặc chia sẻ dữ liệu giữa các Container
- Lưu dữ liệu lên Remote Server hoặc Cloud
- Khi bạn muốn Backup, Migrate hoặc chia sẻ dữ liệu giữa các Docker Host

tmpfs mount: Docker sẽ mount data của Container lên Memory của Docker Host thay vì Disk và khi Container stop hoặc xóa, Docker sẽ xóa phần data đã mount.

Vậy nên sử dụng khi nào:

- Với kiểu này chúng ta thường sẽ lưu dữ liệu tạm thời hoặc với những dữ liệu nhạy cảm.

3. Tìm hiểu Docker Hub

Giống như Github hay Gitlab, Docker Hub là một nơi giúp Developer có thể lưu trữ Image. Image có thể set Private hoặc Public tùy vào người Push Image, Nếu là Private thì chỉ người Push Image hoặc những người được chia sẻ mới có thể sử dụng Image. Còn với Public mọi người có thể sử dụng được. Nổi tiếng và được sử dụng nhiều nhất là DockerHub, một Host Image mà mọi người có thể chia sẻ và sử dụng những Image có sẵn được Build từ cộng đồng.

Cách sử dụng Docker Hub:

1. Đăng ký tài khoản trên Docker Hub
2. Truy cập vào trang web Docker Hub và đăng nhập
3. Tạo và đặt tên cho Repository
4. Thiết lập quyền truy cập (Public hoặc Private)
5. Từ Image đã build trước đó, sử dụng câu lệnh Docker để Push Image lên Docker Hub
6. Kiểm tra Repository trên Docker Hub
7. Sau khi Push Image thành công, sử dụng câu lệnh Docker để Pull Image và run Container

3.Thực hành

- Thực hành Network
- Thực hành Volume
- Thực hành Dockerfile
- Build Image từ Dockerfile
- Push & Pull Docker Hub

