DOCKER

[Image 1](#_Toc76992406)

[Docker Container 2](#_Toc76992407)

[Docker file 3](#_Toc76992408)

[Các chỉ thị Dockerfile 3](#_Toc76992409)

[FROM Trong Dockerfile 4](#_Toc76992410)

[COPY và ADD Trong Dockerfile 4](#_Toc76992411)

[ENV Trong Dockerfile 4](#_Toc76992412)

[RUN Trong Dockerfile 4](#_Toc76992413)

[VOLUME Trong Dockerfile 4](#_Toc76992414)

[USER Trong Dockerfile 4](#_Toc76992415)

[WORKDIR Trong Dockerfile 4](#_Toc76992416)

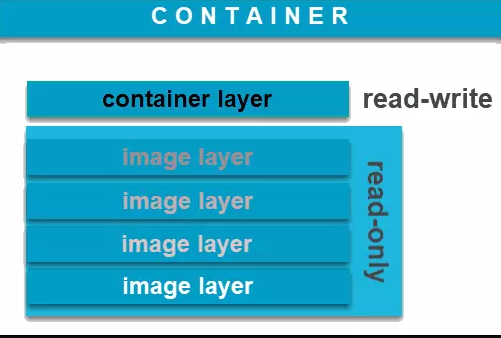
[EXPOSE Trong Dockerfile 4](#_Toc76992417)

[ENTRYPOINT, CMD Trong Dockerfile 4](#_Toc76992418)

Docker là một phần mềm nguồn mở được thiết kế để hỗ trợ và đơn giản hóa việc phát triển ứng dụng. Nó là một tập hợp các platform-as-a-service products tạo ra các virtualized environments cô lập để building, deploy , và testing các ứng dụng.

# 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.
* Do tính chất **read-only** của chúng, những images này đôi khi được gọi là **snapshots**. Chúng đại diện cho một **application** và **virtual environment** của nó tại một thời điểm cụ thể. Tính nhất quán này là một trong những tính năng tuyệt vời của Docker. Nó cho phép các developers **test** và **thử nghiệm phần mềm** trong **điều kiện ổn định, thống nhất**.
* Vì theo một cách nào đó**, images chỉ là các mẫu**, bạn không thể start hoặc run chúng. Những gì bạn có thể làm là **sử dụng mẫu đó** làm cơ sở để **xây dựng một container.** **Một container** cuối cùng **chỉ là một image** đang chạy. Khi bạn tạo một container, nó sẽ thêm một lớp có thể ghi lên trên image bất biến, nghĩa là bây giờ bạn có thể sửa đổi nó.
* **Image** mà **ở** **đó bạn tạo một container tồn tại riêng biệt** và không thể thay đổi. Khi bạn chạy một môi trường containerized, về cơ bản, bạn tạo một bản sao đọc-ghi của image bên trong container. Lúc này nó sẽ thêm một Container Layer cho phép sửa đổi toàn bộ bản sao của image.



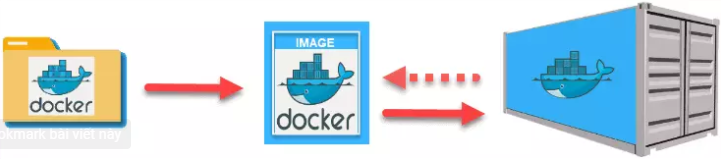
* Bạn có thể **tạo số lượng Docker image không giới hạn** **từ một image** base. Mỗi khi bạn thay đổi trạng thái ban đầu của một image và lưu trạng thái hiện có, bạn tạo một image mới với một layer mới ở trên nó.
* Do đó, Docker image có thể bao gồm một loạt các layer, mỗi layer khác nhau nhưng cũng có nguồn gốc từ layer trước. Các image layer đại diện cho các readonly-file mà chứa thêm các container layer mỗi khi bạn sử dụng nó để khởi động một virtual environment.

# Docker 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.
* Một tính năng quan trọng của **container** là tính chuẩn xác cho việc chạy các ứng dụng trong container. Không chỉ **đảm bảo cho ứng dụng hoạt động như nhau trong các môi trường giống nhau**, nó còn làm **đơn giản việc cài đặt** và **chia sẻ** cài đặt này cho các thành viên trong team.
* **Vì container hoạt động độc lập**, nó đảm bảo không làm ảnh hưởng xấu đến các container khác, cũng như server mà nó đang chạy trong đó. Docker được cho là "**tạo ra sự độc lập tuyệt vời**". Vì vậy, bạn sẽ **không cần lo lắng việc máy tính của bạn bị xung đột** do ứng dụng đang được phát triển được chạy trong container.
* *Không giống như máy ao (VM) nơi mà sự ảo hóa (virtualization) xảy ra ở tầng phần cứng (hardware level),* ***container chỉ ảo hóa ở lớp ứng dụng (app level).*** *Nó có thể dùng 1 máy, chia sẻ kernel và giả môi trường để* ***chạy process độc lập****. Điều này làm cho* ***container cực kì nhẹ****, không* ***chiếm nhiều tài nguyên của máy***.

# Docker file

* Tất cả bất **đầu từ 1 script** gồm các **instruction** mà định nghĩa cách để tạo ra **1 Docker image**. Đoạn script này được gọi là Docker file . File này chạy 1 loạt các command và **tạo ra 1 Docker image**. Câu lệnh để tạo ra 1 image từ Dockerfile là **docker build .**
* Source code, files, dependencies, và binary libraries, tất cả đều được chứa trong Docker image, cái mà tạo nên container.
* Để tạo ra 1 container từ 1 image, sử dụng lệnh **docker create**



## Các chỉ thị Dockerfile

* FROM : mọi Docker file đều có chỉ thị này, chỉ định image cơ sở
* COPY ADD : sao chép dữ liệu
* ENV : thiết lập biến môi trường
* RUN : chạy các lệnh.
* VOLUME : gắn ổ đĩa, thư mục
* USER : user
* WORKDIR : thư mục làm việc
* EXPOSE : thiết lập cổng

### FROM Trong Dockerfile

* Như trên đã nói, chỉ thị này chỉ ra image cơ sở để xây dựng nên image mới. Để xây dựng từ image nào đó thì bạn cần đọc document của Image đó để biết trong đó đang chứa gì, có thể chạy các lệnh gì trong đó .

### COPY và ADD Trong Dockerfile

* Được dùng để thêm thư mục, file vào Image. Cú pháp viết đó là:
* ADD thư\_mục\_nguồn thư\_mục\_đích
* Trong đó thư\_mục\_nguồn là thư mục ở máy chạy Dockerfile, chứa dữ liệu cần thêm vào. thư\_mục\_đích là nơi dữ liệu được thêm vào ở container.

### ENV Trong Dockerfile

* Chỉ thị này dùng để thiết lập biến môi trường, như biến môi trường PATH ..., tùy hệ thống hay ứng dụng yêu cầu biến môi trường nào thì căn cứ vào đó để thiết lập.

### RUN Trong Dockerfile

* Thi hành các lệnh, tương tự bạn chạy lệnh shell trên OS từ terminal.
* RUN lệnh-và-tham-số-cần-chạy
* RUN ["lệnh", "tham số1", "tham số 2" ...]

### VOLUME Trong Dockerfile

* Chỉ thị tạo một ổ đĩa , chia sẻ được giữa các container.
* VOLUME /dir\_vol

### USER Trong Dockerfile

* Bạn thêm user được dùng khi chạy các lệnh ở chỉ thị RUN CMD WORKDIR.

### WORKDIR Trong Dockerfile

* Thiết lập thư mục làm việc hiện tại chi các chỉ thị CMD, ENTRYPOINT, ADD thi hành.
* WORKDIR path\_current\_dir

### EXPOSE Trong Dockerfile

* Để thiết lập cổng mà container lắng nghe, cho phép các container khác trên cùng mạng liên lạc qua cổng này hoặc đỉ ánh xạ cổng host vào cổng này.
* EXPOSE port

### ENTRYPOINT, CMD Trong Dockerfile

* Chạy lệnh trong chỉ thị này khi container được chạy.
* ENTRYPOINT commnad\_script
* ENTRYPOINT ["command", "tham-số", ...]
* CMD ý nghĩa tương tự như ENTRYPOINT, khác là lệnh chạy bằng shell của container.
* CMD command param1 param2
* Chú ý ở dạng sau của CMD thì nó lại là thiết lập tham số cho ENTRYPOINT

# Docker-compose

* **docker-compose.yml** để tạo ra các dịch vụ chạy ứng dụng cụ thể .
* Service
* Lệnh docker-compose
* Tạo Image HTTPD
* Tạo Image PHP (FPM)
* Chuẩn bị my.cnf cho MySQL
* Biên tập docker-compose.yml
* Cài đặt Joomla

## Chia nhỏ ứng dụng của bạn thành các Service Docker

* Khi phân phối các ứng dựng hoàn thành, **từng phần của ứng dụng** đó ở dạng sản phẩm gọi nó là các **service** (dịch vụ).

## Docker Compose với file docker-compose.yml

* File docker-compose.yml gần giống ý nghĩa với file Dockerfile đã tìm hiểu trong [Sử dụng Dockerfile](https://xuanthulab.net/su-dung-dockerfile-de-tu-dong-tao-cac-image-trong-docker.html), là một file text, viết với định dạng [YAML](https://yaml.org/) (Ain’t Markup Language, đọc nhanh định dạng [Định dạng YML](https://xuanthulab.net/tim-hieu-file-yaml-co-ban.html)) là cấu hình để tử đó lệnh docker compose sinh ra và quản lý các service (container), các network, các ổ đĩa ... cho một ứng dụng hoàn chỉnh.

## **Lệnh**docker-compose

* Về nội dung bên trong file docker-compose.yml sẽ tìm hiểu phía sau, giờ giả sử đang có file này, từ thư mục chứa file này gõ lệnh docker-compose với tham số phù hợp để thi hành những tác vụ như:
* Tạo và chạy các thành phần định nghĩa trong docker-compose.yml (các dịch vụ, image, container, mạng, đĩa ...).
* docker-compose up
* Dừng và xóa: image, container, mạng, đĩa tạo ra bởi docker-compose up
* docker-compose down
* Theo dõi Logs từ các dịch vụ
* docker-compose logs [SERVICES]