Docker là gì?

[Docker là một nền tảng](https://www.geeksforgeeks.org/introduction-to-docker/)[chứa ứng dụng](https://www.geeksforgeeks.org/containerization-using-docker/) nguồn mở mà qua đó bạn có thể đóng gói ứng dụng của mình và tất cả các phần phụ thuộc của nó vào một đơn vị tiêu chuẩn hóa được gọi là bộ chứa.

Các thùng chứa có trọng lượng nhẹ nên có thể di chuyển được và chúng được cách ly với cơ sở hạ tầng bên dưới cũng như với các thùng chứa khác.

Các câu lệnh docker run.

Docker Run

# Để đặt tên cho container

docker run --name <container\_name> <image\_name>

### **Docker Pull**

# Lệnh này cho phép bạn lấy bất kỳ hình ảnh nào Docker hub

docker pull <image\_name>

Docker Ps

# Lệnh này (theo mặc định) hiển thị cho chúng ta danh sách tất cả các container đang chạy

docker ps [options..]

Options:

-a:  hiển thị cho chúng ta tất cả các vùng chứa.

-l: hiển thị cho chúng tôi vùng chứa mới nhất.

-q: chỉ hiển thị Id của vùng chứa.

Docker Stop

# lệnh dừng một container.

docker stop <container\_ID>

Docker Start

# Lệnh khởi động lại container

docker start <container\_ID>

### **Docker rm**

# xóa một container

docker rm {options} <container\_name or ID>

Options:

-f: loại bỏ container một cách mạnh mẽ.

-v: xóa các tập.

–l: xóa liên kết cụ thể được đề cập.

### Docker RMI

# Xóa một image

docker rmi <image ID/ image name>

**Docker Images**

**# Hiển thị dánh sách các image**

**docker images**

Docker exec

# Lệnh chạy các lệnh mới trong một container đang chạy

docker exec {options}

Options:

-d: để chạy các lệnh ở chế độ nền.

-i: nó sẽ giữ STDIN mở ngay cả khi không được đính kèm.

-e: đặt các biến môi trường

**Docker Ports**

# Để truy cập container docker từ thế giới bên ngoài, chúng tôi phải ánh xạ cổng trên máy chủ của chúng tôi tới cổng trên container.

docker run -d -p <port\_on\_host>:<port\_on\_container> Container\_name

Docker Login

# Lệnh đăng nhập Docker sẽ giúp bạn xác thực với trung tâm Docker mà qua đó bạn có thể đẩy và kéo hình ảnh của mình.

docker login

Docker Push

# Khi bạn xây dựng hình ảnh tùy chỉnh của riêng mình bằng cách sử dụng Dockerfile, bạn cần lưu trữ hình ảnh trong sổ đăng ký từ xa là DockerHub để bạn cần đẩy hình ảnh của mình bằng cách sử dụng lệnh sau

docker push <Image name/Image ID>

Docker Build

# Lệnh docker build được sử dụng để xây dựng hình ảnh docker với sự trợ giúp của Dockerfile.

docker build -t image\_name:tag .

Docker Stop

# Bạn có thể dừng và khởi động các container docker nơi bạn có thể thực hiện việc bảo trì các container

docker stop container\_name\_or\_id

Stop Multiple Containers

# Thay vì dừng một container duy nhất. Bạn có thể dừng nhiều vùng chứa cùng một lúc bằng cách sử dụng các lệnh sau

docker stop container1 container2 container3

Docker Restart

# Khởi động lại các vùng chứa để giải quyết các vùng chứa bằng cách sử dụng các lệnh sau

docker restart container\_name\_or\_id

Docker Inspection

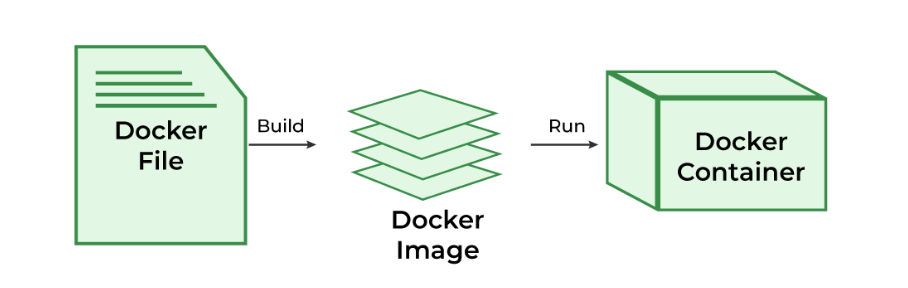
# Các vùng chứa Docker sẽ gặp một số lỗi trong thời gian thực để gỡ lỗi của vùng chứa

docker inspect container\_name\_or\_id

Docker Commit command

# Sau khi chạy các vùng chứa bằng cách sử dụng hình ảnh hiện tại, bạn có thể thực hiện cập nhật cho các vùng chứa bằng cách tương tác với các vùng chứa từ vùng chứa đó

docker commit container\_name\_or\_id new\_image\_name:tag



Dockerfile là một tập lệnh sử dụng nền tảng Docker để tự động tạo các vùng chứa. Về cơ bản nó là một tài liệu văn bản chứa tất cả các hướng dẫn mà người dùng có thể sử dụng để tạo hình ảnh từ dòng lệnh. Nền tảng Docker là [nền tảng dựa trên Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/) cho phép các nhà phát triển tạo và thực thi các bộ chứa, chương trình độc lập và hệ thống độc lập với cơ sở hạ tầng cơ bản.

Dockerfile commands/Instructions

FROM

# Câu lệnh FROM xác định hình ảnh nào cần tải xuống và bắt đầu từ đó. Đây phải là lệnh đầu tiên trong Dockerfile của bạn. Một Dockerfile có thể có nhiều câu lệnh FROM, điều đó có nghĩa là Dockerfile tạo ra nhiều hơn một hình ảnh.

*FROM <ImageName>*

COPY

# Lệnh sao chép được sử dụng để sao chép tệp/thư mục vào hình ảnh trong khi xây dựng hình ảnh

COPY <Source> <Destination>

ADD

# Nếu chúng ta xác định thêm một số tệp, câu lệnh ADD sẽ được sử dụng. Về cơ bản, nó cung cấp hướng dẫn sao chép các tệp, thư mục mới hoặc URL tệp từ xa và sau đó thêm chúng vào hệ thống tệp của hình ảnh.

ADD <URL>

RUN

# Câu lệnh RUN xác định việc chạy một lệnh thông qua shell, đợi nó hoàn thành và lưu kết quả. Nó cho biết tiến trình nào sẽ chạy bên trong container vào thời gian chạy.

RUN < Command + ARGS>

CMD

# CMD chỉ định toàn bộ lệnh để chạy. Chúng ta có thể nói CMD là đối số mặc định được truyền vào ENTRYPOINT. Mục đích chính của lệnh CMD là khởi chạy phần mềm được yêu cầu trong vùng chứa

CMD [command + args]

ENTRYPOINT

# Một vùng chứa sẽ hoạt động như một tệp thực thi được định cấu hình bởi ENTRYPOINT. Khi bạn khởi động vùng chứa Docker, một lệnh hoặc tập lệnh có tên ENTRYPOINT sẽ được thực thi.

# Không thể ghi đè nó. Sự khác biệt duy nhất giữa CMD và ENTRYPOINT là CMD có thể bị ghi đè còn ENTRYPOINT thì không.

ENTRYPOINT [command + args]

ARG

# Lệnh ARG xác định một biến mà người dùng có thể chuyển vào thời điểm xây dựng cho trình xây dựng bằng lệnh docker build bằng cách sử dụng cờ --build-arg <varname>=<value>

ARG <name>[=<default value>]

ENV

# Câu lệnh ENV đặt các biến môi trường cả trong quá trình xây dựng và khi chạy kết quả. Nó có thể được sử dụng trong Dockerfile và bất kỳ tập lệnh nào mà nó gọi. Nó có thể được sử dụng trong Dockerfile cũng như bất kỳ tập lệnh nào mà Dockerfile gọi.

ENV <key>=<value> ...

EXPOSE

# Câu lệnh EXPOSE ánh xạ một cổng vào vùng chứa. Các cổng có thể là TCP hoặc UDP nhưng theo mặc định là TCP.

EXPOSE <port> [<port>/<protocol>...]

LABEL

# Lệnh LABEL thêm siêu dữ liệu vào hình ảnh

LABEL <key>=<value> <key>=<value> <key>=<value> ...

ONBUILD

# Lệnh ONBUILD thêm vào hình ảnh một lệnh kích hoạt sẽ được thực thi sau đó, khi hình ảnh được sử dụng làm cơ sở cho một bản dựng khác

ONBUILD <INSTRUCTION>

SHELL

# Lệnh SHELL cho phép ghi đè shell mặc định được sử dụng cho dạng lệnh shell

SHELL ["executable", "parameters"]

USER

# Lệnh USER đặt tên người dùng (hoặc UID) và tùy chọn nhóm người dùng (hoặc GID) để sử dụng làm người dùng và nhóm mặc định cho phần còn lại của giai đoạn hiện tại. Người dùng được chỉ định sẽ được sử dụng cho các lệnh RUN và trong thời gian chạy, chạy các lệnh ENTRYPOINT và CMD có liên quan

USER <user>[:<group>] or USER <UID>[:<GID>]

VOLUME

# Câu lệnh VOLUME xác định các khối được chia sẻ hoặc các khối tạm thời tùy thuộc vào việc bạn có một hay hai đối số

VOLUME ["/data"]

WORKDIR

# Lệnh WORKDIR thiết lập thư mục làm việc cho mọi lệnh RUN, CMD, ENTRYPOINT, COPY và ADD theo sau nó trong Dockerfile

# Nếu WORKDIR không tồn tại, nó sẽ được tạo ngay cả khi nó không được sử dụng trong bất kỳ lệnh Dockerfile nào tiếp theo.

WORKDIR /path/to/workdir

**Docker** Image là gói phần mềm thực thi bao gồm mọi thứ cần thiết để chạy một ứng dụng. Hình ảnh này cho biết cách khởi tạo vùng chứa, xác định thành phần phần mềm nào sẽ chạy và chạy như thế nào.

# Hình ảnh Docker được xây dựng bằng Dockerfile bao gồm một bộ hướng dẫn cần thiết để chứa một ứng dụng. Hình ảnh docker bao gồm những phần sau để chạy một phần mềm. Hình ảnh docker là hình ảnh độc lập với nền tảng, có thể được xây dựng trong môi trường Windows và nó có thể được đẩy đến trung tâm docker và được kéo bởi những người khác có môi trường hệ điều hành khác nhau như Linux.

Các thành phần của Docker Image

* Layers: Các lớp hệ thống tập tin bất biến được xếp chồng lên nhau để tạo thành một hình ảnh hoàn chỉnh.
* Base Image: Lớp nền tảng, thường là hệ điều hành tối thiểu hoặc môi trường thời gian chạy.
* Dockerfile: Một tệp văn bản chứa hướng dẫn xây dựng Docker image.
* Image ID: Một mã định danh duy nhất cho mỗi Docker image.
* Tags: Nhãn dùng để quản lý và phiên bản Docker images.

Commands of Docker Image

| Command | Description |
| --- | --- |
| docker image build | Lệnh này được sử dụng để xây dựng một hình ảnh từ Dockerfile  docker build -t your\_image\_name:tag -f path/to/Dockerfile . |
| docker image history | Nó được sử dụng để biết lịch sử của docker image |
| docker image inspect | Nó được sử dụng để hiển thị thông tin chi tiết trên một hoặc nhiều hình ảnh |
| docker image prune | Nó được sử dụng để xóa các hình ảnh không sử dụng không được liên kết với bất kỳ vùng chứa nào |
| docker image save | Lệnh này giúp lưu hình ảnh docker vào tệp lưu trữ tar |
| docker image tag | Nó giúp tạo thẻ cho hình ảnh mục tiêu đề cập đến hình ảnh nguồn. |
| docker images ls or  docker images | Liệt kê docker image  Để liệt kê tất cả các Docker Images bao gồm cả các image trung gian, bạn có thể cung cấp cờ -a. |
| docker image pull <image-name> | **Pull an Docker Image From a Registry** |
| docker rmi <id-of-image> | **Xóa Docker Image** |
| docker search <name\_image> | **Tìm kiếm 1 hình ảnh** |

## Cấu trúc của **Docker** Image:

**Base Image**: Hình ảnh cơ bản sẽ là điểm khởi đầu cho phần lớn các Dockerfile và nó có thể được tạo từ đầu.

**Parent Image**: Hình ảnh gốc là hình ảnh mà hình ảnh của chúng ta dựa trên. Chúng ta có thể tham chiếu đến hình ảnh gốc trong Dockerfile bằng lệnh FROM và mỗi khai báo sau đó sẽ ảnh hưởng đến hình ảnh gốc.

**Layers**: Hình ảnh Docker có nhiều lớp. Để tạo một chuỗi các hình ảnh trung gian, mỗi lớp được tạo lên trên lớp trước nó.

**Docker Registry**: Tham khảo trang này trên Docker Đăng ký để biết thêm thông tin.

Tối ưu docker Image.

**Optimize Image Size**: Cố gắng sử dụng các bản dựng nhiều giai đoạn và hình ảnh cơ sở tối thiểu để giảm kích thước hình ảnh.

**Leverage Caching**: Bằng cách sắp xếp các hướng dẫn như hướng dẫn ít thay đổi trước và thường xuyên thay đổi hướng dẫn ở dưới cùng, chúng tôi có thể tối đa hóa bộ nhớ đệm và tăng tốc độ xây dựng.

**Keep Image Secure**: Bằng cách thường xuyên cập nhật hình ảnh và quét tìm lỗ hổng, chúng tôi có thể bảo mật hình ảnh.

**Use Tags Wisely**: Bằng cách thực hành sử dụng các thẻ cụ thể cho hình ảnh thay vì chỉ định thẻ mới nhất, chúng ta có thể đảm bảo tính ổn định và nhất quán.

Khắc phục sự cố thường gặp liên quan đến Docker Image.

Large Image Sizes: Sử dụng các bản dựng nhiều giai đoạn và dọn sạch các tệp không cần thiết để giảm kích thước hình ảnh.

Dependency Conflicts: Đảm bảo tính tương thích bằng cách chỉ định chính xác các phiên bản phụ thuộc trong Dockerfile.

Build Failures: Kiểm tra cú pháp Dockerfile và tính chính xác của lệnh, đồng thời xác minh tất cả các tệp cần thiết được bao gồm.

Slow Performance: Tối ưu hóa thứ tự hướng dẫn Dockerfile và tận dụng bộ nhớ đệm để cải thiện thời gian xây dựng.

Docker Commit image:

được sử dụng để tạo hình ảnh mới từ những thay đổi được thực hiện đối với vùng chứa Docker. Cam kết của Docker có thể lưu trạng thái hiện tại của vùng chứa dưới dạng hình ảnh Docker

docker commit [OPTIONS] CONTAINER [REPOSITORY[:TAG]]

CONTAINER: Tên hoặc ID của vùng chứa Docker.

REPOSITORY: Tên của kho lưu trữ mà bạn muốn đẩy hình ảnh docker vào.

TAG: Gắn thẻ hình ảnh mới.

Các tùy chọn của Docker Commit

| Option | Description |
| --- | --- |
| -a, --author | Chỉ định tên tác giả cho hình ảnh |
| -c, --change | Áp dụng hướng dẫn Dockerfile cho hình ảnh |
| -m, --message | Chỉ định một thông điệp cam kết cho hình ảnh |
| --pause | Tạm dừng vùng chứa trong khi cam kết |
| -p, --pause-file | Tạm dừng vùng chứa bằng tệp tạm dừng trong khi cam kết |
| --platform | Đặt nền tảng nếu không được chỉ định trong Dockerfile |

**Docker Container** là một môi trường ảo kết hợp mã ứng dụng với tất cả các phần phụ thuộc cần thiết để chạy ứng dụng. Ứng dụng chạy nhanh chóng và đáng tin cậy từ môi trường điện toán này sang môi trường điện toán khác.

| Diện mạo | Người lái tàu | Máy ảo (VM) |
| --- | --- | --- |
| Sự cách ly | Docker đi kèm với nền tảng nhẹ,  sử dụng công nghệ container hóa cho các môi trường cô lập. | Máy ảo đi kèm phần mềm nặng cung cấp khả năng ảo hóa hoàn toàn để cô lập toàn bộ hệ điều hành. |
| Sử dụng tài nguyên | Nó tiêu thụ ít tài nguyên hơn từ hạt nhân Host OS và sử dụng chúng hiệu quả hơn | Máy ảo (VM) kém hiệu quả hơn trong việc sử dụng tài nguyên, mỗi VM sử dụng nhân hệ điều hành riêng và tiêu tốn nhiều tài nguyên hơn. |
| Thời gian bắt đầu | Những cách này nhanh hơn trong việc thiết lập nền tảng hệ điều hành (thùng chứa), thường chỉ mất vài giây. | Những giải pháp này khởi động chậm hơn và thường mất vài phút để thiết lập các Phiên bản. |
| Khả năng mở rộng | Việc mở rộng quy mô container trong Docker rất dễ dàng vì nó chiếm ít diện tích hơn. | Sẽ tốn nhiều tài nguyên khi mở rộng quy mô các phiên bản. |
| Tính di động | Docker cung cấp các ứng dụng được đóng gói có tính di động cao, nhất quán trên nhiều môi trường khác nhau. | Những hệ điều hành này cũng dễ di chuyển nhưng sẽ nặng hơn do phải bao gồm toàn bộ hệ điều hành. |
| Trường hợp sử dụng | Nó sẽ có hiệu quả đối với các dịch vụ siêu nhỏ, ứng dụng nhẹ và môi trường chứa container. | Thích hợp để chạy nhiều ứng dụng trong nhiều môi trường khác nhau. |