

**ĐẠI HỌC QUỐC GIA TP.HCM**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC  
QUẢN TRỊ MẠNG VÀ HỆ THỐNG**

**TÌM HIỂU VÀ TRIỂN KHAI DOCKER**

**NHÓM 13**

Giáo viên hướng dẫn - Bùi Thanh Bình

Lớp: NT132.M21.MMCL

Sinh viên thực hiện:

19521800 Thân Trọng Hoàng Long  
20521919 Nguyễn Lê Minh Thanh  
20521692 Bùi Thị Trúc Nhạn

# MỤC LỤC

1. Tổng quan về Docker.....	3
1.1 Khái niệm Docker.....	3
1.1.1 Container là gì .....	3
1.2 Phân biệt Docker và Virtual Machine... ..	4
1.3 Cách hoạt động của Docker... ..	4
1.4 Các thành phần trong Docker .....	5
1.4.1 Docker Engine .....	5
1.4.2 Docker Image .....	6
1.4.3 Docker Hub .....	6
1.4.4 Docker Machine .....	7
1.4.5 Docker Compose .....	7
1.4.6 Docker swarm.....	8
1.4.7 Docker File .....	9
1.4.8 Docker Toolbox.....	9
1.5 Cấu trúc của Docker .....	10
1.6 Quy trình thực thi của hệ thống .....	12
1.7 Quy trình ảo hoá ứng dụng sử dụng Docker.....	12
1.8 Ưu điểm khi sử dụng.....	13
1.9 Nhược điểm.....	13
2. Mô hình triển khai.....	13
3. Cài đặt, cấu hình.....	15
4. Video demo .....	16
5. Tài liệu tham khảo.....	17

# 1. Tổng quan về Docker

## 1.1 Khái niệm Docker

Để dễ hình dung Docker là gì, thì trước tiên ta cần biết **Container** là gì?

Container là:

+Tạo ra một môi trường chứa mọi thứ mà phần mềm cần để có thể chạy được.

+Không bị các yếu tố liên quan đến môi trường hệ thống làm ảnh hưởng tới.

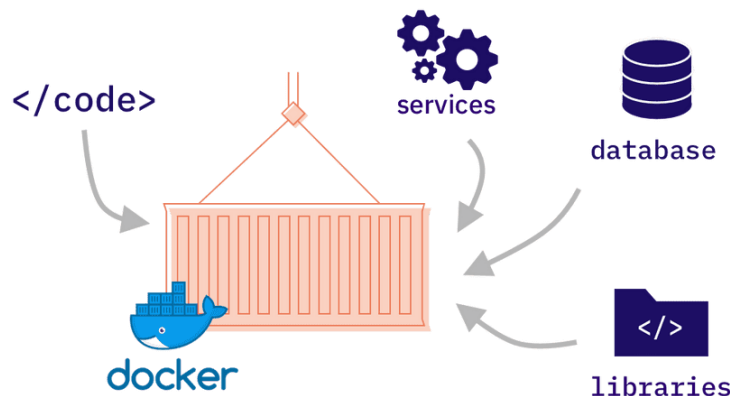
+Cũng như không làm ảnh hưởng tới các phần còn lại của hệ thống

→ có thể hiểu là ruby, rails, mysql ... kia được bỏ gọn vào một hoặc nhiều cái thùng (**container**), ứng dụng chạy trong những chiếc thùng đó, đã có sẵn mọi thứ cần thiết để hoạt động, không bị ảnh hưởng từ bên ngoài và cũng không gây ảnh hưởng ra ngoài.

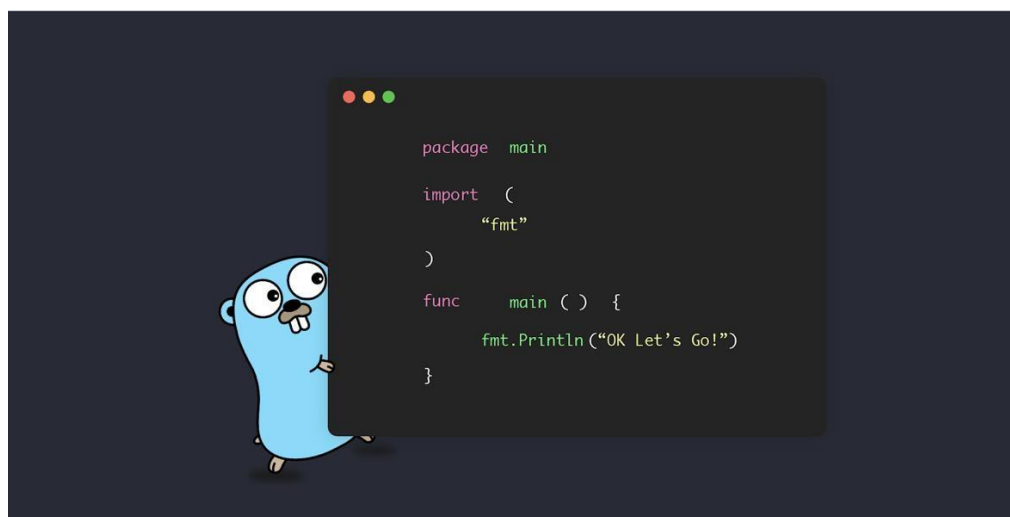


Còn **Docker** là một trình thực thi container, một công cụ giúp cho việc tạo ra và triển khai các container để phát triển, chạy ứng dụng được dễ dàng. Docker hỗ trợ nhiều nền tảng hệ điều hành khác nhau bao gồm Linux, Windows và cả Mac.

**Docker** là nền tảng dành cho lập trình viên, quản trị hệ thống dùng để xây dựng, vận chuyển và chạy các ứng dụng phân tán ( lập trình viên tập trung vào việc viết code chứ không lo lắng về việc triển khai, không lo lắng ở máy của lập trình viên chạy được, máy khác lại không chạy được )



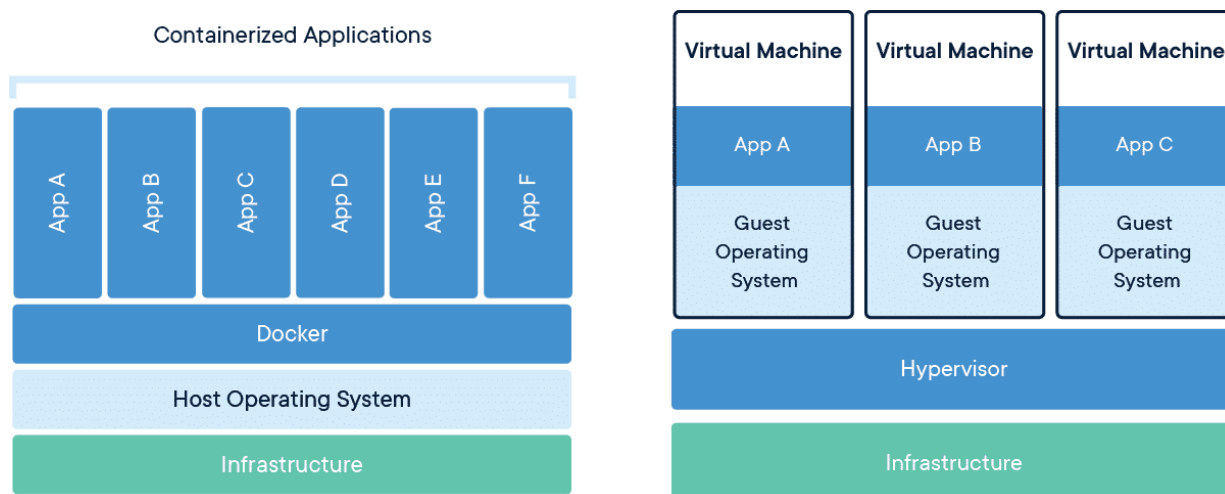
Ban đầu Docker được viết bằng ngôn ngữ Python, bây giờ chuyển sang Golang



*Go là một ngôn ngữ lập trình mới do google thiết kế và phát triển*

## 1.2 Phân biệt Docker và Virtual Machine

- **Docker** : Dùng chung kernel, chạy độc lập trên Host Operating System và có thể chạy trên bất kì hệ điều hành nào cũng như cloud. Khởi động và làm cho ứng dụng sẵn sàng chạy trong 500ms, mang lại tính khả thi cao cho những dự án cần sự mở rộng nhanh.
- **Virtual Machine** : Cần thêm một Guest OS cho nên sẽ tốn tài nguyên hơn và làm chậm máy thật khi sử dụng. Thời gian khởi động trung bình là 20s có thể lên đến hàng phút, thay đổi phụ thuộc vào tốc độ của ổ đĩa



*Docker vs Virtual Machine*

### 1.3 Cách hoạt động của Docker

Thông qua một **docker engine** có sự kết hợp của 2 yếu tố:

- Server và Client
- Server và Client giao tiếp với nhau thông qua REST API.

Các thành phần chính của Docker:

- **Docker Engine:** là thành phần chính của Docker, như một công cụ để đóng gói ứng dụng, tạo và khởi chạy docker container từ các docker image
- **Docker Image:** tương tự file .gho để ghost win. Một docker image thường chứa OS(Windows, ubuntu, CentOS) và các môi trường lập trình được cài sẵn (httpd, mysqld, nginx, python, git, ...). Bạn có thể tải các image từ người khác
- **Docker Hub:** là nơi để mọi người upload, chia sẻ các images Docker của mình

### 1.4 Các thành phần trong Docker

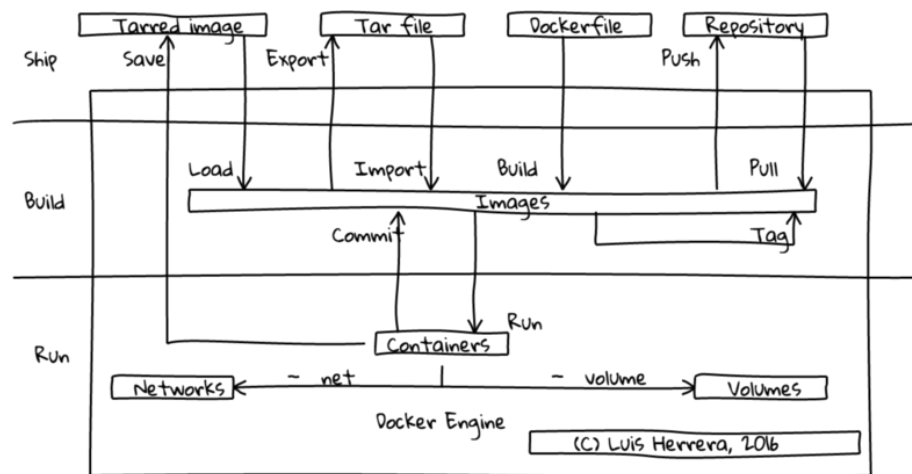
1.4.1 **Docker engine:** Là một ứng dụng Client – Server. Docker hoạt động trên sự kết hợp giữa 1 docker engine và 2 yếu tố :

- 1 server và 1 client: là tạo ra một quy trình daemon(server) phía máy chủ lưu trữ *images, containers, networks* và *storage volumes*. Daemon cũng cung cấp giao diện dòng lệnh phía máy khách (CLI) cho phép người dùng tương tác với daemon thông qua giao diện lập trình ứng dụng Docker.
- Giao tiếp giữa server và client nhờ REST API thì Docker Client nói chuyện với Docker Engine thông qua một RESTful API, để thực thi các lệnh như *build, ship* và *run* một container.

4 đối tượng *images, containers, network, volume* của Engine đều có ID để xác định và phối hợp với nhau để *build, ship* và *run* application ở bất cứ đâu

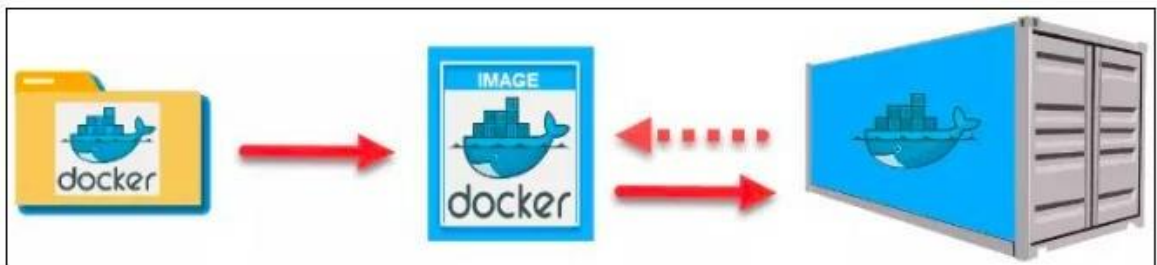
- **Images:** là thành phần để đóng gói ứng dụng và các thành phần mà ứng dụng phụ thuộc để chạy. Và image được lưu trữ ở trên local hoặc trên một Registry (là nơi lưu trữ và cung cấp kho chứa các image)
- **Containers:** là một instance của image, và nó hoạt động như một thư mục, chứa tất cả những thứ cần thiết để chạy một ứng dụng

- **Network:** cung cấp một private network chỉ tồn tại giữa container và host
- **Volume:** Volume trong Docker được dùng để chia sẻ dữ liệu cho container



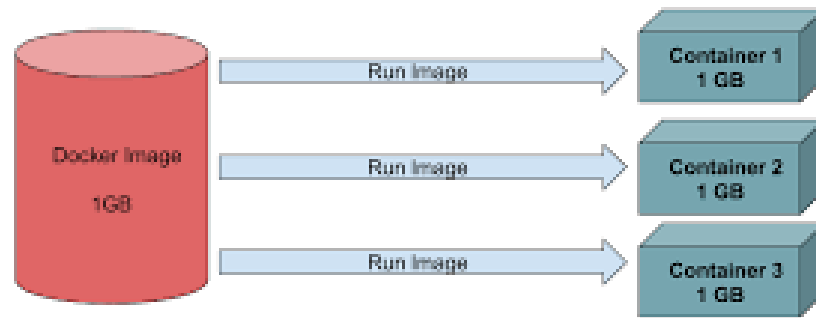
Hình ảnh minh họa cho các mối liên hệ giữa các thành phần trên

**1.4.2 Docker Image:** là nền tảng của container, có thể hiểu Docker image như khung xương giúp định hình cho container, nó sẽ tạo ra container khi thực hiện câu lệnh chạy image đó. Nếu nói với phong cách lập trình hướng đối tượng, Docker image là class, còn container là thực thể (instance, thể hiện) của class đó.

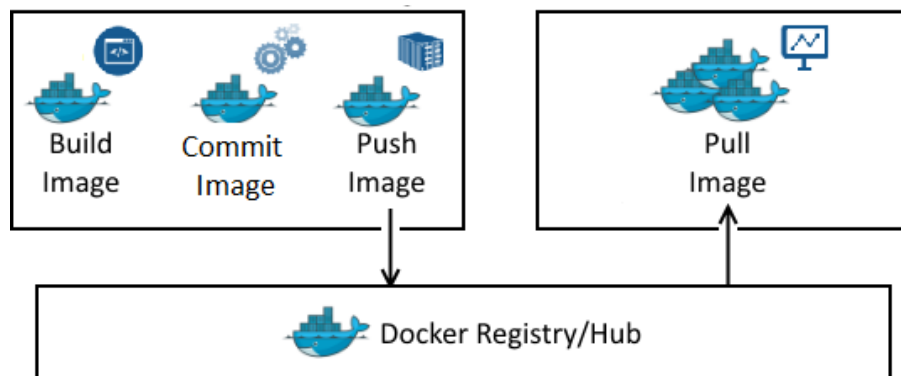


- Image có thể tồn tại mà không cần container, trong khi container chạy thì cần có image đã tồn tại. Vì vậy, container phụ thuộc vào image và sử dụng nó để tạo ra run-time environment và chạy ứng dụng trên đó.
- Docker image là cực kì quan trọng để chi phối và định hình 1 Docker container.

# Docker image vs Docker Container



1.4.3 **Docker Hub:** ( giống GitHub hoặc BitBucket ) là nơi lưu trữ các image của docker. Nó là một kho lưu trữ trực tuyến dựa trên đám mây lưu trữ cả hai loại kho lưu trữ, tức là kho lưu trữ công khai cũng như kho lưu trữ riêng. Các kho lưu trữ công khai có thể truy cập được cho mọi người, nhưng kho riêng có thể truy cập được đối với chủ sở hữu có liên quan của kho lưu trữ; cũng có một chi phí liên quan đến nó nếu chúng ta lưu trữ nhiều hơn một số kho nhất định dưới dạng riêng tư.



## - Tính năng của Docker Hub:

+ **Kho ảnh (Repositories):** Nó giúp chúng tôi tìm và kéo hình ảnh container từ Docker Hub. Nó cũng giúp chúng tôi đẩy hình ảnh thành kho lưu trữ công khai hoặc riêng tư đến Docker Hub.

+ **Nhóm và tổ chức (Teams & Organizations):** Nó cho phép chúng tôi tạo các nhóm làm việc và đẩy các kho lưu trữ thành một kho riêng, chỉ có sẵn để sử dụng trong tổ chức của chúng tôi. Bằng cách này, chúng tôi đã quản lý quyền truy cập vào kho riêng của chúng tôi về hình ảnh container.

+ **Tích hợp GitHub và Bitbucket:** Nó cho phép tích hợp với các kho mã nguồn như GitHub và BitBucket.

+ **Xây dựng tự động (Builds):** Nếu bất kỳ thay đổi nào trong mã nguồn được đẩy đến kho lưu trữ mã nguồn, nó sẽ tự động phát hiện và xây dựng hình ảnh chứa từ GitHub hoặc BitBucket và đẩy chúng vào Docker Hub.

+ **Webhooks:** Khi chúng tôi đã đẩy hình ảnh của mình thành công, với sự trợ giúp của webhook, nó kích hoạt một hành động để tích hợp Docker Hub với các dịch vụ khác.

+ **Hình ảnh chính thức và nhà xuất bản:** Các hình ảnh chất lượng cao được cung cấp bởi các docker được coi là hình ảnh chính thức, và nó có thể được kéo và sử dụng. Tương tự, hình ảnh chất lượng cao được cung cấp bởi các nhà cung cấp bên ngoài là hình ảnh của nhà xuất bản, còn được gọi là hình ảnh được chứng nhận, mang lại sự hỗ trợ và đảm bảo khả năng tương thích với doanh nghiệp Docker. Chúng tôi sẽ thảo luận thêm về hình ảnh được chứng nhận sau trong bài viết này.

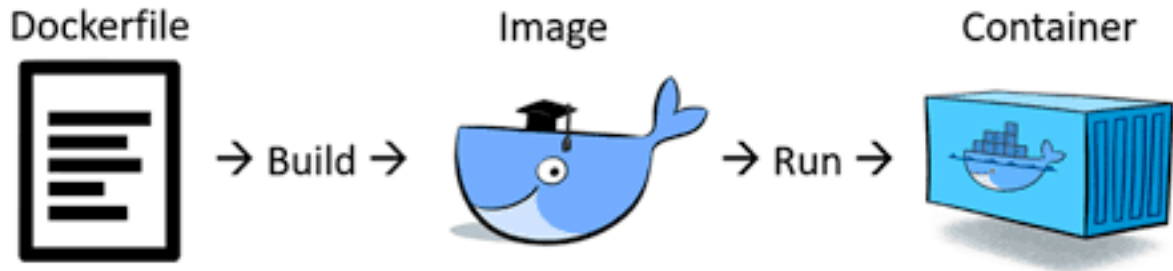
1.4.4 **Docker Machine:** ( là hệ thống tạo ra các *docker engine* trên máy chủ ) Machine tạo Docker Engine trên laptop của bạn hoặc trên bất cứ dịch vụ cloud phổ biến nào như AWS, Azure, Google Cloud, Softlayer hoặc trên hệ thống data center như VMware, OpenStack. Docker Machine sẽ tạo các máy ảo và cài Docker Engine lên chúng và cuối cùng nó sẽ cấu hình Docker Client để giao tiếp với Docker Engine một cách bảo mật

1.4.5 **Docker Compose:** ( thực hiện chạy ứng dụng thông qua các định nghĩa cấu hình các docker container, thực hiện thông qua file cấu hình ) là công cụ giúp định nghĩa và khởi chạy multi-container Docker applications

1.4.6 **Docker Swarm:** là một công cụ giúp chúng ta tạo ra một clustering Docker. Nó giúp chúng ta gom nhiều Docker Engine lại với nhau và ta có thể "nhìn" nó như duy nhất một virtual Docker Engine



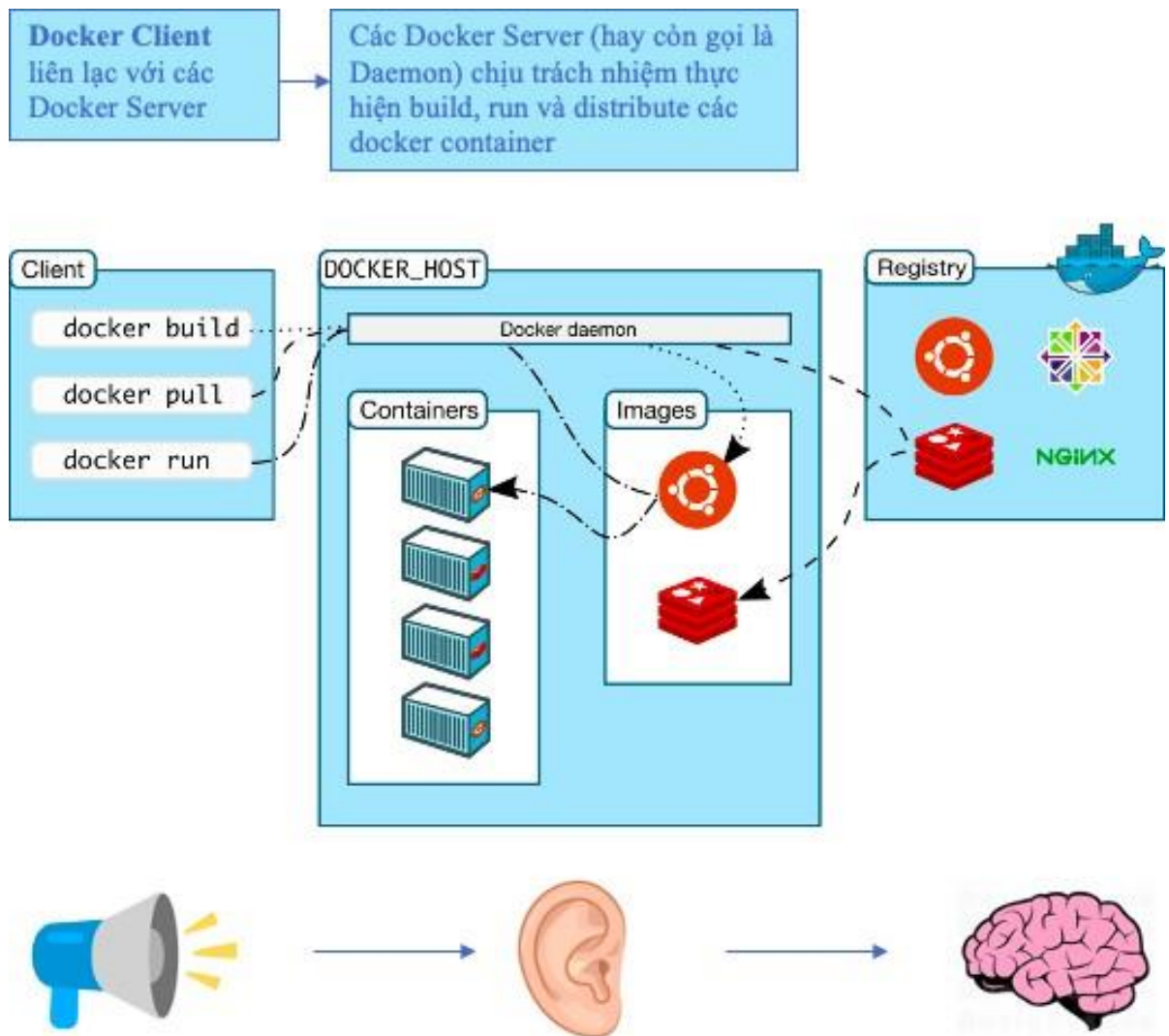
1.4.7 **Docker File:** như một script dùng để build các image trong container. Dockerfile bao gồm các câu lệnh liên tiếp nhau được thực hiện tự động trên một image gốc để tạo ra một image mới. Dockerfile giúp đơn giản hóa tiến trình từ lúc bắt đầu đến khi kết thúc

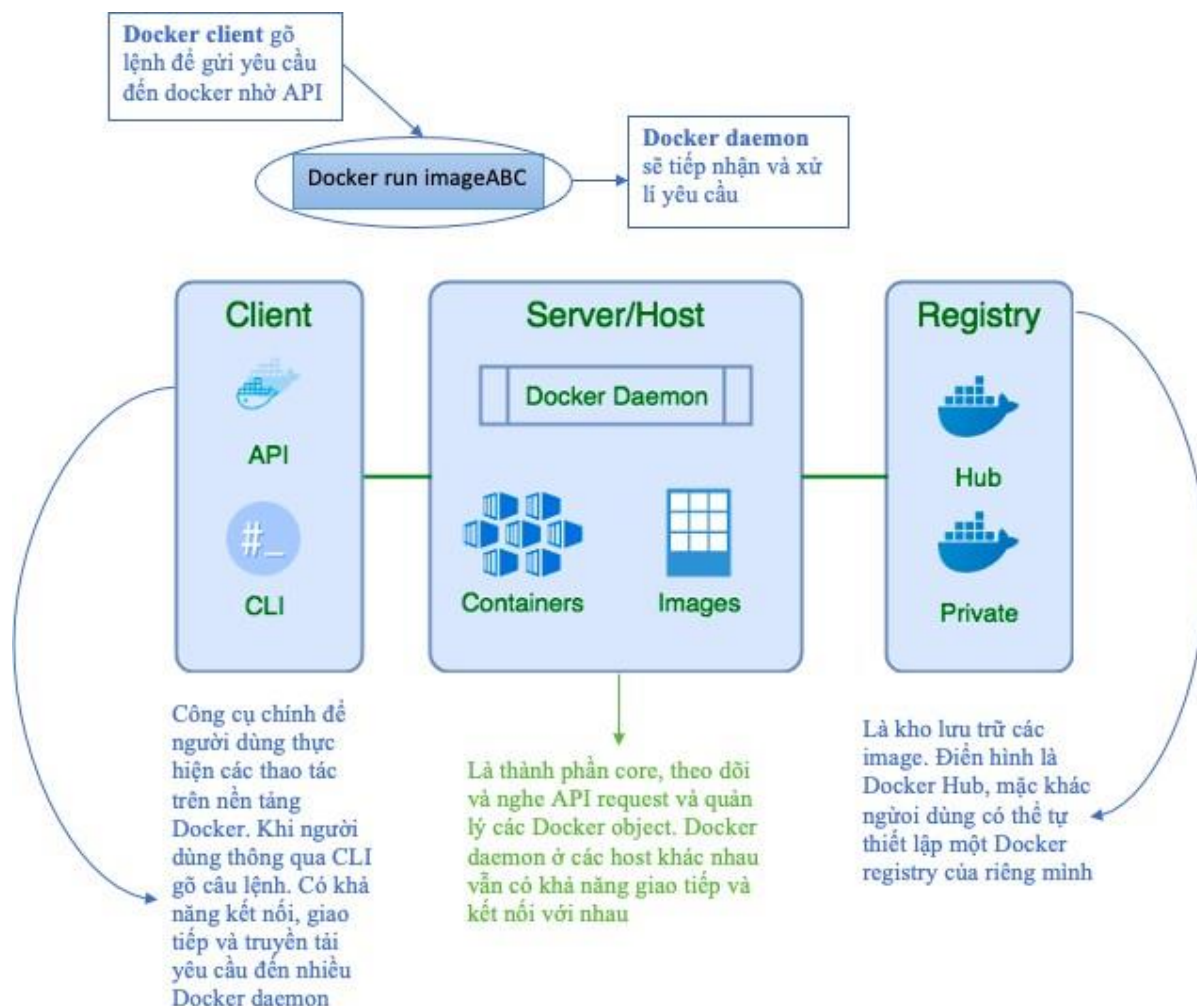


1.4.8 **Docker Toolbox:** Bởi vì Docker Engine dùng một số feature của kernel Linux nên ta sẽ không thể chạy Docker Engine natively trên Windows hoặc BSD được. Ở các phiên bản trước đây thì ta sẽ cần một máy ảo cài một phiên bản Linux nào đó và sau đó cài Docker Engine lên máy ảo đó

## 1.5 Cấu trúc của Docker

Docker sử dụng kiến trúc *client-server*.



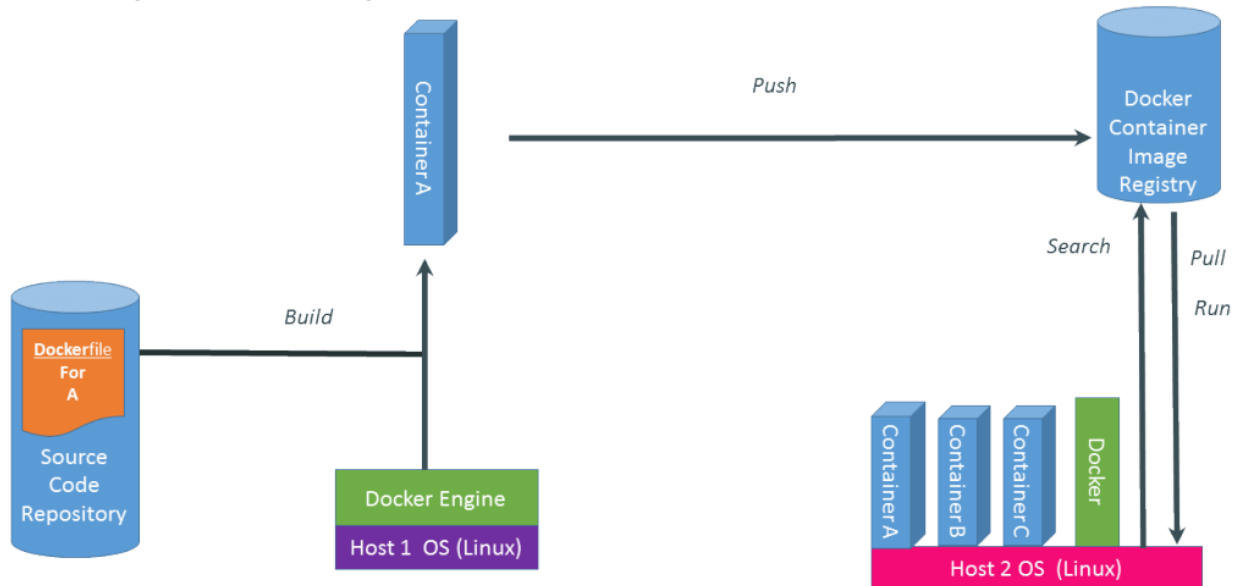


Docker **client** và Docker **server** có thể nằm trên cùng một server hoặc khác server. Chúng giao tiếp với nhau thông qua REST API dựa trên UNIX sockets hoặc network interface.

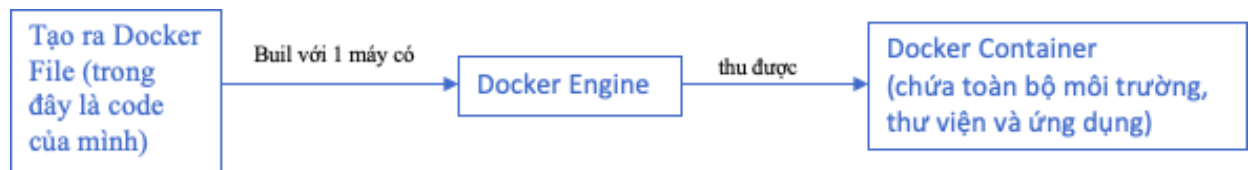
Docker daemon chạy trên các máy host. Người dùng sẽ không tương tác trực tiếp với các daemon, mà thông qua Docker Client.

## 1.6 Quy trình thực thi

Hệ thống Docker bao gồm 3 bước chính: Build ➡ Push ➡ Pull, Run



- Build:



- Push:

Sau khi có được Container thì thực hiện push Container này lên đám mây và lưu trữ ở đó (việc push này được thực hiện qua môi trường mạng Internet) thông qua Docker Hub

- Pull, Run

Một máy muốn sử dụng Container (bắt buộc phải cài đặt Docker Engine) mà Container đó đã được push lên đám mây thì bắt buộc máy phải thực hiện pull container này về máy. Sau đó thực hiện run container

## 1.7 Quy trình ảo hoá ứng dụng

- + Tạo Dockerfile
- + Build image
- + Chạy container

## 1.8 Ưu điểm khi sử dụng

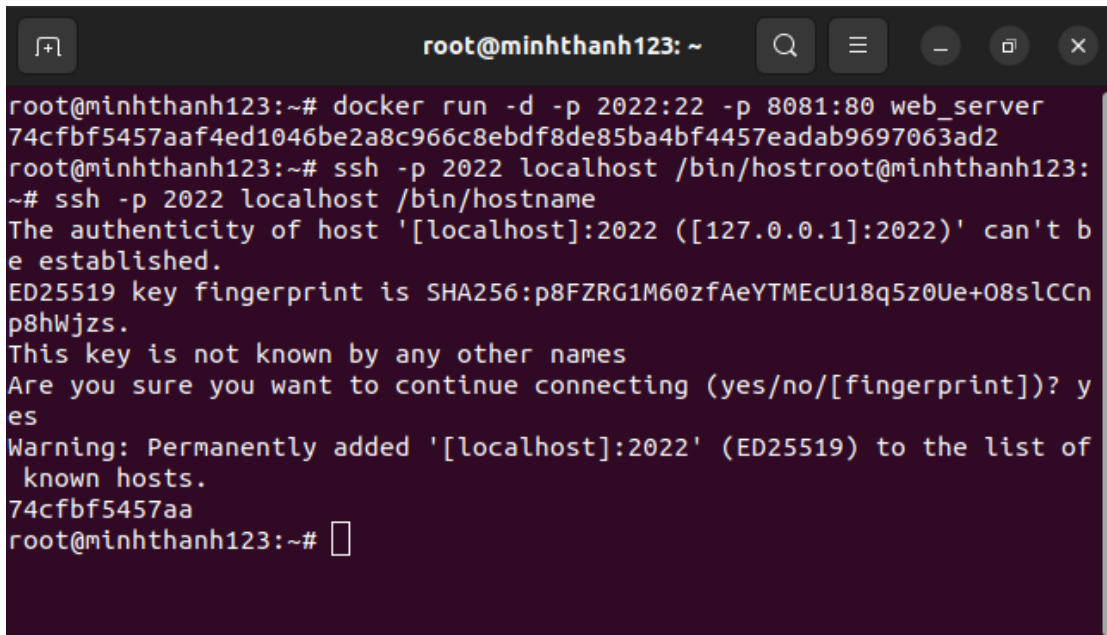
- Tính dễ ứng dụng
- Tốc độ
- Môi trường chạy và khả năng mở rộng
- Giúp kết nối dễ dàng, nhanh gọn, đỡ phải cài

## 1.9 Nhược điểm

- Tốn thời gian học tập
- Mới, cập nhật thay đổi thường xuyên
- Do dùng chung hệ điều hành nên nếu có lỗi hỏng nào đấy ở kernel của host hệ điều hành thì nó sẽ ảnh hưởng tới toàn bộ container có trong host OS đấy;
- Ngoài ra với host hệ điều hành là Linux, nếu trong trường hợp ai đấy hoặc một ứng dụng nào đấy có trong container chiếm được quyền superuser. Về lý thuyết thì tầng hệ điều hành sẽ bị crack và ảnh hưởng trực tiếp tới máy host bị hack cũng như các container khác trong máy đó (hacker sử dụng quyền chiếm được để lấy dữ liệu từ máy host cũng như từ các container khác trong cùng máy host bị hack chẳng hạn).

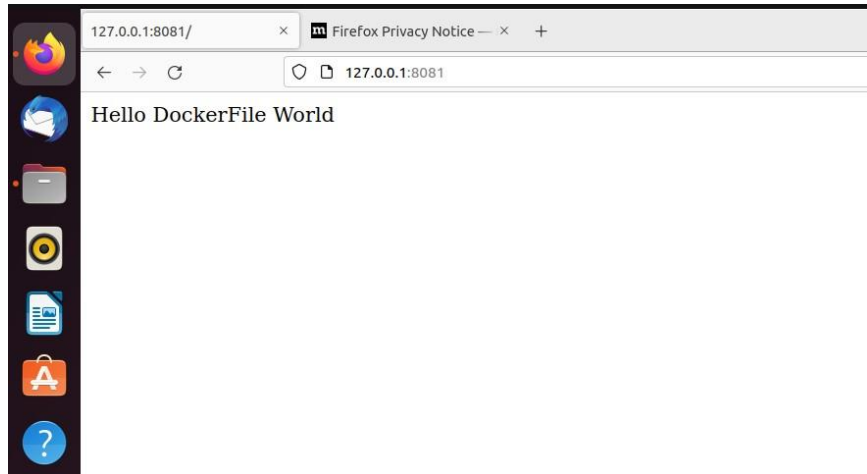
## 2. Mô hình triển khai

- Chạy container web server và kết nối ssh tới webserver

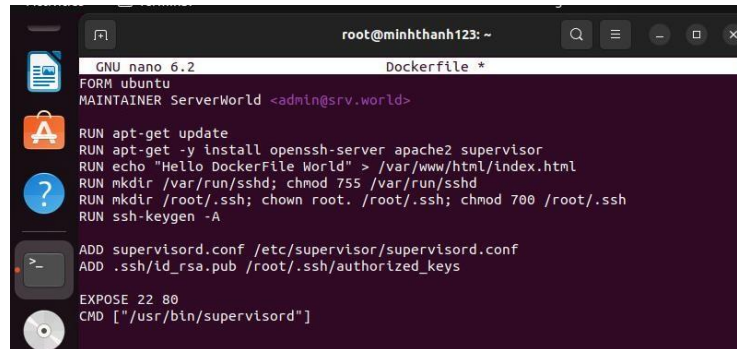


```
root@minhthanh123: ~  
root@minhthanh123:~# docker run -d -p 2022:22 -p 8081:80 web_server  
74cfbf5457aaf4ed1046be2a8c966c8ebdf8de85ba4bf4457eadab9697063ad2  
root@minhthanh123:~# ssh -p 2022 localhost /bin/hostroot@minhthanh123:  
~# ssh -p 2022 localhost /bin/hostname  
The authenticity of host '[localhost]:2022 ([127.0.0.1]:2022)' can't b  
e established.  
ED25519 key fingerprint is SHA256:p8FZRG1M60zfAeYTMecU18q5z0Ue+08slCCn  
p8hWjzs.  
This key is not known by any other names  
Are you sure you want to continue connecting (yes/no/[fingerprint])? y  
es  
Warning: Permanently added '[localhost]:2022' (ED25519) to the list of  
known hosts.  
74cfbf5457aa  
root@minhthanh123:~#
```

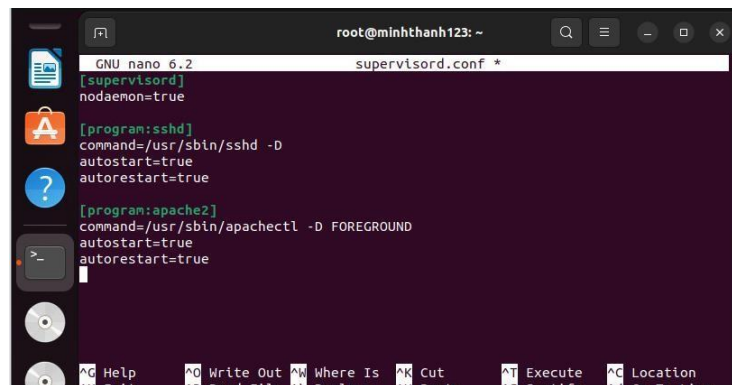
- Webserver chạy thành công , thông qua cổng 8081 trênlocalhost



- Nội dung Docker File



- Nội dung của Docker Supervisord



### 3. Cài đặt, cấu hình

- Nội dung Docker File

```
FROM ubuntu
MAINTAINER ServerWorld <admin@srv.world>

RUN apt-get update
RUN apt-get -y install apache2

EXPOSE 80
CMD ["/usr/sbin/apachectl", "-D", "FOREGROUND"]
```

- Nội dung Docker Compose

```
version: '3'
services:
  db:
    image: mariadb
    volumes:
      - /var/lib/docker/disk01:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_USER: bionic
      MYSQL_PASSWORD: password
      MYSQL_DATABASE: bionic_db
    ports:
      - "3306:3306"
  web:
    build: .
    ports:
      - "80:80"
    volumes:
      - /var/lib/docker/disk02:/var/www/html
```



- 2 container được tạo thành công

```
manager [Running] - Oracle VM VirtualBox
root@exeld:~/docke# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
3e3472cbe500   mariadb   "docker-entrypoint.s..." 43 seconds ago Up 39 seconds 0.0.0.0:3306->3306/tcp, :::3306->3306/tcp
dd24c8a123ea   docke_web "/usr/sbin/apachectl..." 43 seconds ago Up 39 seconds 0.0.0.0:80->80/tcp, :::80->80/tcp
root@exeld:~/docke# _
```

- Join swarm

```
root@exeld:~# docker swarm init --advertise-addr 10.1.1.4
Swarm initialized: current node (u6zdyeijm1asfw1oo97fmvijz) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-2jk00uy722cfau85prz193yqwxqsq3rikispxiy5rrz9yc7434-9115r1u1v1uoo1nn42xyjvdaw 10.1.1.4:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

root@exeld:~# docker node ls
ID                HOSTNAME        STATUS        AVAILABILITY        MANAGER STATUS        ENGINE VERSION
u6zdyeijm1asfw1oo97fmvijz *   exeld          Ready         Active               Leader                 20.10.14
c80e900u7brzu4u1n7nig8nnp     exzeld         Ready         Active               -                     20.10.12
root@exeld:~#
```

```
worker [Running] - Oracle VM VirtualBox
root@exzeld:~# docker swarm join --token SWMTKN-1-2jk00uy722cfau85prz193yqwxqsq3rikispxiy5rrz9yc7434-9115r1u1v1uoo1nn42xyjvdaw 10.1.1.4:2377
This node joined a swarm as a worker.
root@exzeld:~# _
```

- Chạy service trênswarm

```
manager [Running] - Oracle VM VirtualBox
On failure:      pause
Monitoring Period: 5s
Max failure ratio: 0
Update order:    stop-first
RollbackConfig:
  Parallelism:    1
  On failure:     pause
  Monitoring Period: 5s
  Max failure ratio: 0
  Rollback order: stop-first
ContainerSpec:
  Image:          apache2_server:latest
  Init:           false
Resources:
Endpoint Mode:   vip
Ports:
  PublishedPort = 80
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress

root@exeld:~/cluster# docker service ps swarm_cluster
ID                NAME                IMAGE                NODE        DESIRED STATE
d08ccgmrtgj       swarm_cluster.1     apache2_server:latest exeld       Running
o2ca9n111uxu      swarm_cluster.2     apache2_server:latest exeld       Running
```



#### 4. Video demo

[https://drive.google.com/drive/folders/1GPEsASkyC\\_Q\\_P0SniEQoh-tVK\\_KDkazW?usp=sharing](https://drive.google.com/drive/folders/1GPEsASkyC_Q_P0SniEQoh-tVK_KDkazW?usp=sharing)

Tài liệu tham khảo

Internet: [1] <https://www.docker.com>

[2] <https://github.com>

[3] <https://www.wikipedia.org>

[4] <https://stackify.com/docker-tutorial/>

Tiếng Anh: [5] Karl Matthias & Sean P. Kane(2015), Docker - Up & Running