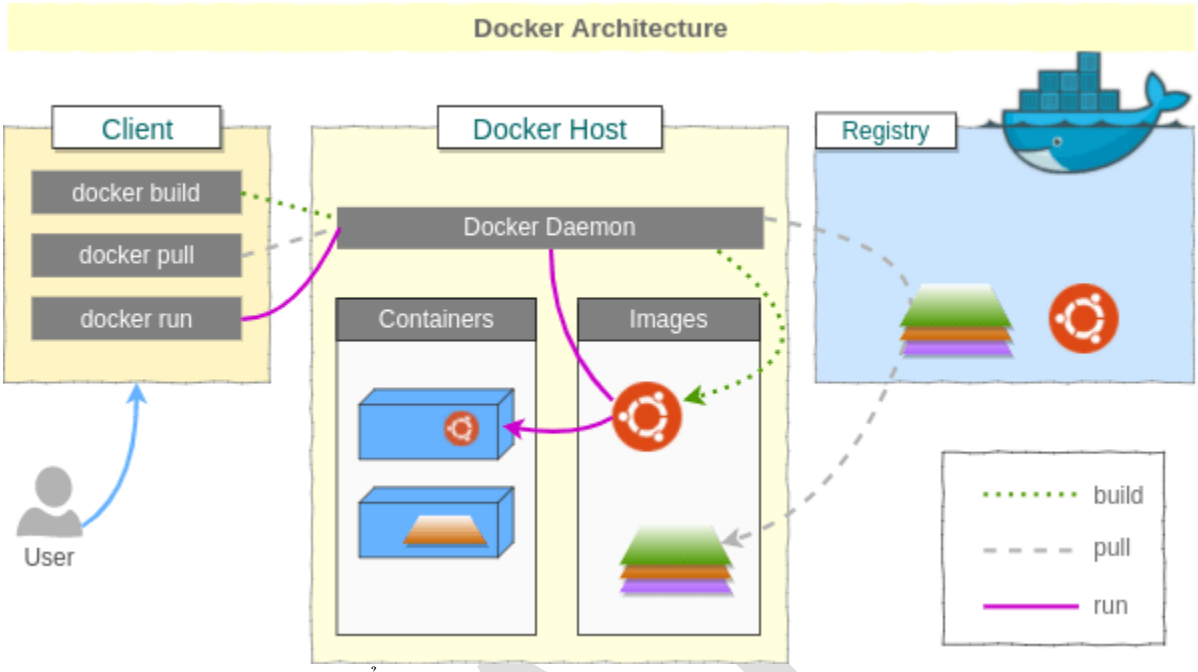
HƯỚNG DẪN DOCKER

Tài liệu dành cho devops-engineer, mức độ cơ bản

MỤC LỤC

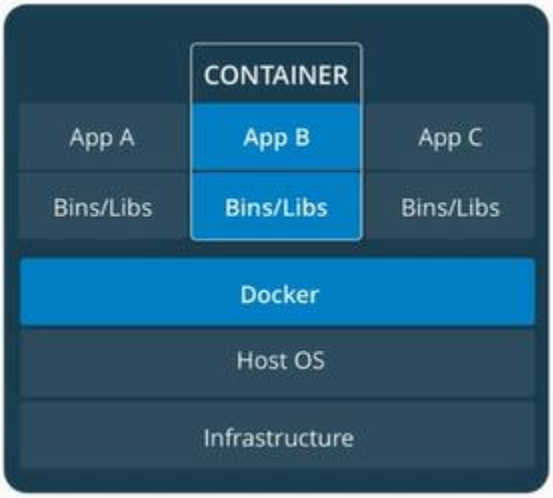
1. Docker là gì?



+ Công cụ đóng gói và dùng để chạy ứng dụng
+ Mã nguồn mở, giúp tạo ra các container để phát triển và triển khai các ứng dụng.

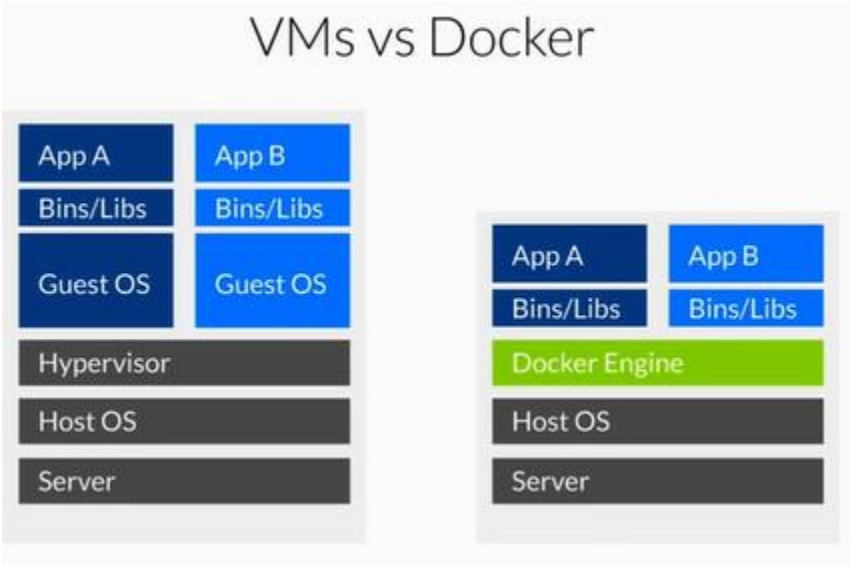
Container là gì?

+ Cho phép các ứng dụng chạy trong môi trường cách ly gọi là container.
+ Để hiểu đơn giản thì có thể coi container như một máy ảo.



Một máy vật lý có phần cứng (ổ đĩa, CPU, RAM,…) chạy trên một hệ điều hành nào đó, được gọi là máy host. Máy host này được dùng để cài docker. Muốn một ứng dụng nào đó chạy trên môi trường docker , thì docker sẽ tạo ra môi trường cách ly, gọi là container. Container chứa đầy đủ các thư viện, các gói ứng dụng nhằm đảm bảo ứng dụng chạy được trong docker. Container là môi trường cách ly. Docker có thể tạo ra nhiều container khác nhau để chạy các ứng dụng khác nhau. Do các container này cách ly, nên chúng không có sự tương tác trực tiếp qua lại lẫn nhau. Như vậy việc docker cài đặt vào máy host và tạo ra các container khác nhau để chạy những ứng dụng khác

nhau rất giống việc chúng ta cài đặt các máy ảo khác nhau để chạy những ứng dụng khác nhau. Tuy nhiên so với công cụ ảo hóa, thì việc sử dụng docker có sự khác biệt với nhau rất lớn về bản chất.



Đối với công nghệ ảo hóa, như chúng ta đã biết việc sử dụng ảo hóa được thiết lập bằng cách tạo một lớp ảo hóa, lớp ảo hóa phân chia tài nguyên của máy hos như CPU,RAM từ đó tạo ra máy ảo. Trên mỗi máy ảo này sẽ chạy những hệ điều hành và ứng dụng của chúng ta. Như vậy khi càng nhiều máy ảo, chúng ta càng sử dụng nhiều tài nguyên khác nhau. Đồng thời, việc chúng ta cài đặt một máy ảo giống như việc chúng ta cài trên máy vật lý thật. Các ứng dụng sẽ sử dụng các thư viện của máy ảo đó.

Đối với Docker, thì khi chạy một ứng dụng nào đó, khi chạy thì nó sẽ sử dụng chính hệ điều hành của máy host và bổ sung thêm các thư viện, gói phần mềm cần thiết để chạy được ứng dụng của nó. Do ứng dụng chạy trên docker cùng sử dụng hệ điều hành của máy host nên nó có thể chia sẻ tài nguyên và chạy nhanh hơn so với công nghệ ảo hóa.

2. Cài đặt docker

2.1. Cho window 10

+ Kích hoạt Hyper-V:

1. **Mở PowerShell bằng quyền Admin.**
2. **Chạy lệnh:**

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All
```

+ Cài đặt docker như một ứng dụng bình thường theo link:

https://hub.docker.com/editions/community/docker-ce-desktop-windows

+ Kiểm tra thông tin:

```
docker --version
```

2.2. Cho Ubuntu

Chạy các lệnh sau:

```
sudo apt update
sudo apt install apt-transport-https ca-certificates curl software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
sudo apt update
apt-cache policy docker-ce
sudo apt install docker-ce
sudo systemctl status docker
sudo usermod -aG docker $USER
```

Cài đặt thêm:

```
sudo apt install docker-compose
sudo apt install virtualbox
```

2.3. Cho centos 7

Chạy các lệnh sau:

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install docker-ce
sudo usermod -aG docker $(whoami)
sudo systemctl enable docker.service
sudo systemctl start docker.service

#Cài thêm Docker Compose
sudo yum install epel-release
sudo yum install -y python-pip
sudo pip install docker-compose
sudo yum upgrade python*
docker-compose version
```

3. Docker Image

Đầu tiên, để biết có tất cả những lệnh gì? Thì cách xem đơn giản nhất là gõ **docker** và ấn enter.
Khi muốn sử dụng một lệnh con thì ta sẽ sử dụng cú pháp: docker <lệnh con>.
Ví dụ: docker image
Trong trường hợp không hiểu lệnh đó là gì thì chỉ cần thêm --help để biết thông tin của nó.
Ví dụ: docker image save --help

3.1. Docker Image là gì?

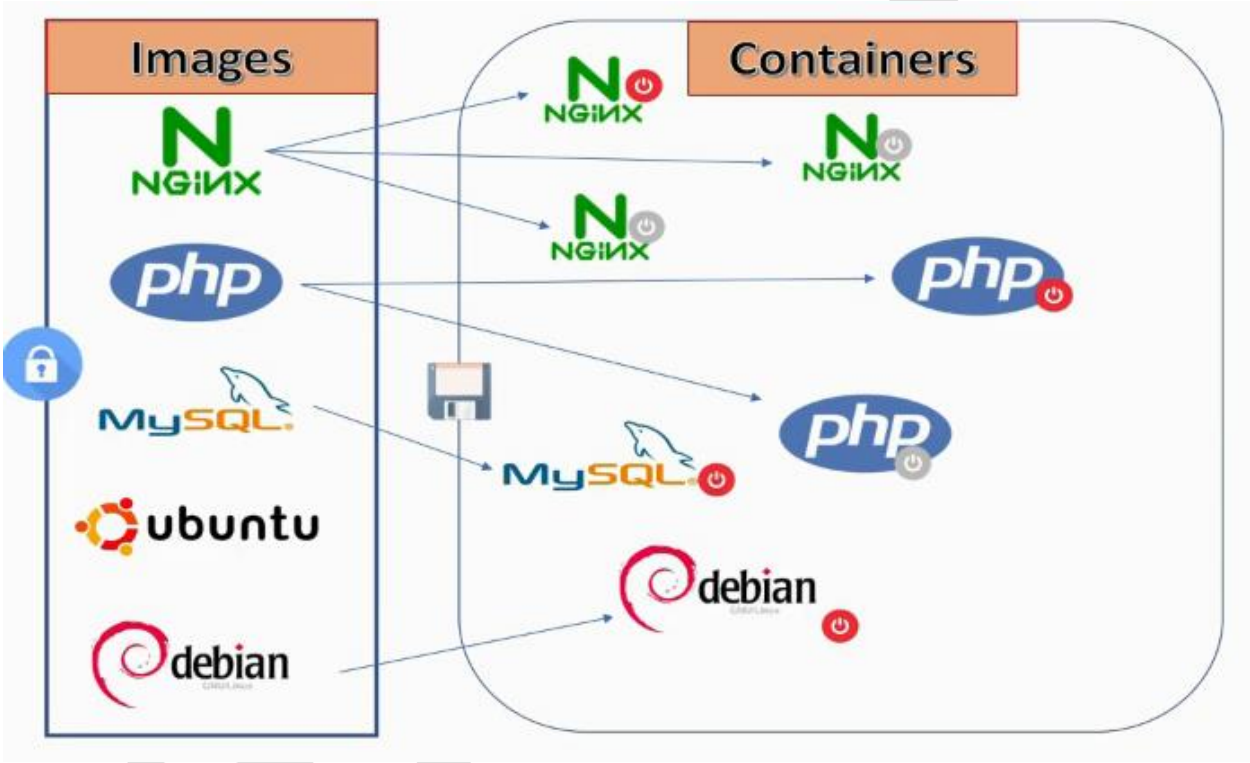


Image trong docker là những phần mềm được đóng gói và quản lý bởi docker. Trong docker các image chỉ có thể đọc, không thể sửa đổi. Khi image được docker khởi chạy thì phiên bản thực thi của image được gọi là các container. Các container có thể ghi được các dữ liệu trong nó. Như vậy để chạy các container, ta phải có các image trước.

3.2. Các lệnh cơ bản với docker image

Chúng ta phải kiểm tra danh sách các image mà chúng ta có trong hệ thống:

```
docker images
```

Ví dụ:

```
PS C:\Users\h> docker images
REPOSITORY                                    TAG                 IMAGE ID            CREATED             SIZE
myimage                                       v1                  95d8cc259a2f        7 months ago        1.14 GB
ubuntu-vim                                    version-1           d82b705682e5        7 months ago        152 MB
php                                           7.3-fpm             888cc7ac7127        7 months ago        401 MB
mariadb                                       latest              3be89eaa5c2b        8 months ago        355 MB
ubuntu                                        latest              3556258649b2        8 months ago        64.2 MB
mysql                                         latest              2151acc12881        8 months ago        445 MB
busybox                                       latest              db8ee88ad75f        8 months ago        1.22 MB
httpd                                         latest              ee39f68eb241        8 months ago        154 MB
docker.elastic.co/kibana/kibana               7.0.1               2c25d2be5a7f        11 months ago       712 MB
docker.elastic.co/elasticsearch/elasticsearch 7.0.1               eb1e40835f76        11 months ago       862 MB
centos                                        latest              9f38484d220f        12 months ago       202 MB
d4w/nsenter                                   latest              9e4f13a0901e        3 years ago         83.8 kB
```

Các thông tin cần lưu tâm:
REPOSITORY: tên image trên kho chứa
TAG: phiên bản của image (latest là phiên bản cuối)
IMAGE ID: chuỗi định danh duy nhất của image trên hệ thống
CREATED: thời gian tạo image
SIZE: kích cỡ image
Những image này được lấy ở đâu ra? Những image được lấy từ kho chứa image được gọi là registry, mặc định là chúng ta đang làm việc với kho chứa công cộng của công ty docker. Chúng ta có thể tìm kiếm theo đường dẫn link: https://hub.docker.com/search?q=&type=image
Ngoài ra, ta cũng có thể tìm kiếm image theo câu lệnh:

docker search <tên image>

Ví dụ:

```
PS C:\Users\h> docker search oracle
NAME                                DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
oraclelinux                         Official Docker builds of Oracle Linux.         641       [OK]
jaspeen/oracle-11g                  Docker image for Oracle 11g database            155                  [OK]
oracleinanutshell/oracle-xe-11g                                                     88
oracle/openjdk                      Docker images containing OpenJDK Oracle Linux   60                   [OK]
oracle/graalvm-ce                   GraalVM Community Edition Official Image        59                   [OK]
absolutapps/oracle-12c-ee           Oracle 12c EE image with web management co...   38
araczkowski/oracle-apex-ords        Oracle Express Edition 11g Release 2 on Ub...   29                   [OK]
oracle/nosql                        Oracle NoSQL on a Docker Image with Oracle...   23                   [OK]
bofm/oracle12c                      Docker image for Oracle Database                23                   [OK]
datagrip/oracle                     Oracle 11.2 & 12.1.0.2-se2 & 11.2.0.2-xe        18                   [OK]
truevoly/oracle-12c                 Copy of sath89/oracle-12c image (https://g...   13
oracle/weblogic-kubernetes-operator Docker images containing the Oracle WebLog...   11
openweb/oracle-tomcat               A fork off of Official tomcat image with O...   8                    [OK]
iamseth/oracledb_exporter           A Prometheus exporter for Oracle modeled a...   3
paulosalgado/oracle-java8-ubuntu-16 Oracle Java 8 on Ubuntu 16.04 LTS.              2                    [OK]
18fgsa/oracle-client                Hosted version of the Oracle Container Ima...   2
softwareplant/oracle                oracle db                                       2                    [OK]
publicisworldwide/oracle-core       This is the core image based on Oracle Lin...   1                    [OK]
roboxes/oracle7                     A generic Oracle Linux 7 base image.            1
toolsmiths/oracle7-test                                                             0
bitnami/oraclelinux-runtimes        Oracle Linux runtime-optimized images           0                    [OK]
amd64/oraclelinux                   Official Docker builds of Oracle Linux.         0
arm64v8/oraclelinux                 Official Docker builds of Oracle Linux.         0
pivotaldata/oracle7-test            Oracle Enterprise Linux (OEL) image for GP...   0
bitnami/oraclelinux-extras          Oracle Linux base images                        0                    [OK]
PS C:\Users\h>
```

Các thông tin cần lưu tâm:
NAME: tên của image
DESCRIPTION: mô tả của image
STARS: số lượng star
OFFICIAL: phiên bản chuẩn

AUTOMATED:

Muốn sử dụng một image thì ta phải tải nó xuống thông qua câu lệnh sau:

```
docker pull <tên image>: <tên tag>
```

Muốn sử dụng với phiên bản mới nhất của một image thì ta không cần thêm tag, lúc đó câu lệnh:

```
docker pull <tên image>
```

Để muốn xóa một image thì chúng ta có lệnh:

```
docker image rm <tên image>:<tên tag>
```

Ngoài ra còn một cách xóa khác mà không cần sử dụng tên image mà dùng image id (chúng ta chỉ cần gõ một số kí tự đầu tiên của image id, nếu nó xác định được đúng là image duy nhất nó sẽ xóa), lệnh như sau:

```
docker image rm <image id>
```

4. Docker container

4.1. Docker container là gì?

Khi một phiên bản của image đang chạy, phiên bản chạy đó gọi là container. Do đó, có thể nói muốn có container thì trước tiên phải có image. Bất kỳ thời điểm nào, chúng ta cũng có thể kiểm tra xem có bao nhiêu container đang chạy và chúng được sinh ra từ những image nào.

4.2. Các lệnh cơ bản với docker container

Tạo một container thì dùng lệnh:

docker run <tham số> <tên image hoặc image id> <tên lệnh> <tham số lệnh>

Ví dụ 1:
docker run –it ubuntu:latest
Trong đó:
-i: tương tác
-t: kết nối với terminal
Ví dụ 2:
docker run –it --name “XYZ” –h centos1 centos:latest
Trong đó:
--name: tên container
-h: tên host của container
Ví dụ 3:
docker run –it --rm debian ls –la
Mô tả: --rm là container tự xóa khi kết thúc. ls –la là lệnh thực thi của container khi nó bắt đầu chạy.
Kiểm tra danh sách container đang chạy, dùng lệnh:

docker ps

Ví dụ:

```
PS C:\Users\h> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
74630e9b48cc        ubuntu              "/bin/bash"         11 seconds ago      Up 9 seconds                            flamboyant_visvesvaraya
```

Các tham số cần lưu ý:
CONTAINER ID: chuỗi string định danh duy nhất của container
IMAGE: image mà container đang dùng.
COMMAND: cho biết lệnh mà container đang chạy (bin/bash tức là terminal)
CREATED: thời gian tạo của container
STATUS: Trạng thái hoạt động của container (Up là đang chạy, Exited là đã dừng).
PORTS: Port của container
NAMES: tên của container, nếu không gán tên khi chạy thì nó sinh ra tự động.
Muốn xem tất cả các container mà cả chạy và không chạy thì dùng lệnh:

docker ps –a

Muốn thoát container mà vẫn chạy thì ấn tổ hợp phím:

Ctrl P Q

Muốn thoát container mà container cũng dừng chạy thì dùng lệnh:

exit

Khi container đang dừng, mà muốn chạy thì ta dùng lệnh:

docker start < tên container hoặc container id>

Muốn vào container để thực thi lệnh nào đó của một container đang chạy thì dùng lệnh:

docker attach <tên container hoặc container id>

Khi container đang chạy mà muốn dừng thì dùng lệnh:

docker stop <tên container hoặc container id>

Xóa container khi nó dừng dùng lệnh:

docker rm < tên container hoặc container id>

Xóa container khi nó đang chạy thì dùng lệnh:

docker rm –f <tên container hoặc container id>

5. Làm việc với docker image và docker container

5.1. Docker exec

Mục đích: Thực thi lệnh ở bên ngoài container. Như trên đã giới thiệu thì dùng lệnh docker attach để vào bên trong một container rồi dùng các lệnh. Tuy nhiên, nhiều trường hợp không muốn vào trong container mà muốn tác động container đó từ host của docker, khi đó ta dùng lệnh:

```
docker exec <tên container hoặc container id> <lệnh thực thi>
```

Ví dụ:

docker exec ubuntu ls

Mô tả: thực hiện lệnh ls vào container ubuntu từ host.

5.2. Lưu container thành một image

Lưu ý trước khi muốn lưu một container thành một image thì container đó phải ở trạng thái dừng (exited). Khi đó ta dùng lệnh sau:

```
docker commit <tên container hoặc container id> <tên image>:<tag>
```

Ví dụ:

docker commit ubuntu ubuntu-ssh:V1

5.3. Xuất image ra file

Ta muốn lưu image của mình ra một file (mặc định imge lưu ra file dưới định dạng tar), chúng ta dùng lệnh sau:

```
docker save –output <tên file có đuôi tar> <tên image hoặc image id>
```

Ví dụ:

docker save --output myimage.tar ubuntu-ssh

5.4. Load file để tạo image

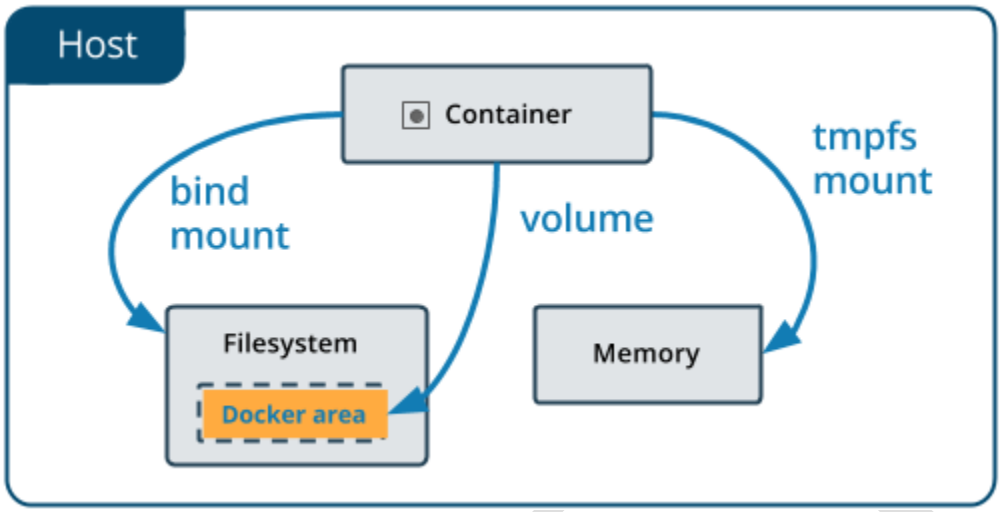
Khi muốn sử dụng một file chứa image để tạo một image, ta dùng lệnh:

```
docker load –i <tên file có chứa image>
```

Sau đó, muốn đổi tên image được tạo thì dùng lệnh:

```
docker tag <image id>  <new  tên>:<new tag>
```

6. Chia sẻ dữ liệu



6.1. Chia sẻ dữ liệu giữa host và container

Mục đích khi xóa container đi thì dữ liệu vẫn còn trên máy host. Để làm được điều này, thì ta dùng lệnh:

docker run –it –v <path lưu thư mục chia sẻ của host>:<path vào thư mục container> <tên image hoặc image id>

Ví dụ:

docker run –it –v /Users/h/Desktop/data:/home/u01/app/data centos

6.2. Chia sẻ dữ liệu giữa hai container

Ta có thể cho hai container cùng chia sẻ dữ liệu với host, khi đó ta dùng lệnh:

docker run –it –name <tên container 2> --volumes-from <tên container 1> <tên image hoặc image id>

Ngoài việc chia sẻ như trên, thì docker còn cho phép tạo ra các ổ đĩa. Các ổ đĩa này cũng được quản lý độc lập so với container, tức là khi xóa container thì ổ đĩa vẫn còn và vẫn còn lưu dữ liệu. Để hiển thị danh sách ổ đĩa ta dùng lệnh:

docker volume ls

Ví dụ:

```
PS C:\Users\h> docker volume ls
DRIVER              VOLUME NAME
local               4d1286f1152df26849371b05487f1e0a97e045307377642db596f9851d878d11
local               mycode
local               other_esdata
local               truyen_du_lieu
```

Để tạo một ổ đĩa, ta dùng lệnh:

docker volume create <tên volume>

Để kiểm tra thông tin một ổ đĩa, ta dùng lệnh:

docker volume inspect <tên volume>

Để xóa một volume, ta dùng lệnh:

docker volume rm <tên volume>

Để xóa tất cả các volume không sử dụng (tức là không có container nào sử dụng) ta dùng lệnh:

```
docker volume prune
```

Để gán ổ đĩa vào một container lúc chạy, ta sử dụng lệnh sau:

```
docker run –it --name <tên container> --mount source=<tên volume>,target=<path ánh xạ ổ đĩa> <tên image hoặc image id>
```

Lưu ý mount chỉ sử dụng được cho phiên bản Docker 17.06 trở về sau, ta có thể sử -v như lệnh sau cho các phiên bản khác:

```
docker run –it --name <tên container> -v <tên volume>:< path ánh xạ ổ đĩa> <tên image hoặc image id>
```

Ví dụ:

docker run –it --name C1 –v D1:/home/u01/app/disk1 ubuntu:latest

Ngoài ra ta muốn tạo ra một ổ đĩa mà ánh xạ với thư mục nào đó của máy host, ta sử dụng lệnh sau:

```
docker volume create --opt device=<path mong muốn> --opt type=none –opt o=bind <tên volume>
```

Và đối với loại ổ đĩa này, thì ta bắt buộc phải sử dụng tham số -v mà không được sử dụng tham số --mount.

7. Networking trong Docker

Trong docker có hệ thống để tạo và quản lý network. Những network này để các container kết nối vào để tạo ra liên kết mạng giữa các container với nhau cũng như là kết nối mạng ra bên ngoài.
Để liệt kê danh sách các network đang có của docker ta dùng lệnh:

```
docker network ls
```

Mặc định ban đầu, docker tạo ra 3 network:

```
PS C:\Users\h> docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
f89046704359        bridge              bridge              local
6832b7e7b39c        host                host                local
3df35c7460fd        none                null                local
```

Trong phạm vi làm việc, ta chỉ cần quan tâm dockẻ có kiểu driver là bridge.
Muốn tra cứu xem một network đang có container nào, ta có thể kiểm tra bằng lệnh:

```
docker network inspect <tên network>
```

Ví dụ:

```
PS C:\Users\h> docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "f8904670435936 38f99a45fc7ac6de546120049f1bc9dec48243e0bb80dee9d6",
        "Created": "2020-03-28T15:55:42.7055914Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Containers": {
            "49839273705d4a6eaffe80fb67a0bd16b8916a381d5faf4fc4cf541be2bb2117": {
                "Name": "C1",
                "EndpointID": "cf88facce860514b18a4f94b21cb196f8897b6158ec1a24aa1d8dcdaa4173a63",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

Để tạo một network ta sử dụng lệnh:

```
docker network create <tên network>
```

Để xóa một network ta sử dụng lệnh:

```
docker network rm <tên network>
```

Để xóa những network đang không sử dụng, ta sử dụng lệnh:

```
docker network prune
```

Để chạy một container mà gắn vào một network mong muốn ta sử dụng lệnh:

```
docker run –it --network <tên network mong muốn> <tên image hoặc image id>
```

8. Giám sát hoạt động của các container

Từ image cơ sở để hình thành image hiện tại, nó trải qua các lệnh mà ta đã dùng để cập nhật. Khi đó, ta có thể xem lịch sử các lệnh đó bằng cách dùng lệnh:

```
docker image history <tên image hoặc image id>
```

Kiểm tra container đã thay đổi gì thì ta sử dụng lệnh:

```
docker diff <tên container>
```

Để kiểm tra log thì ta sử dụng lệnh:

```
docker logs <tên containe>
```

Ngoài ra còn có thể xem N dòng cuối cùng và realtime của log bằng cách sử dụng lệnh sau:

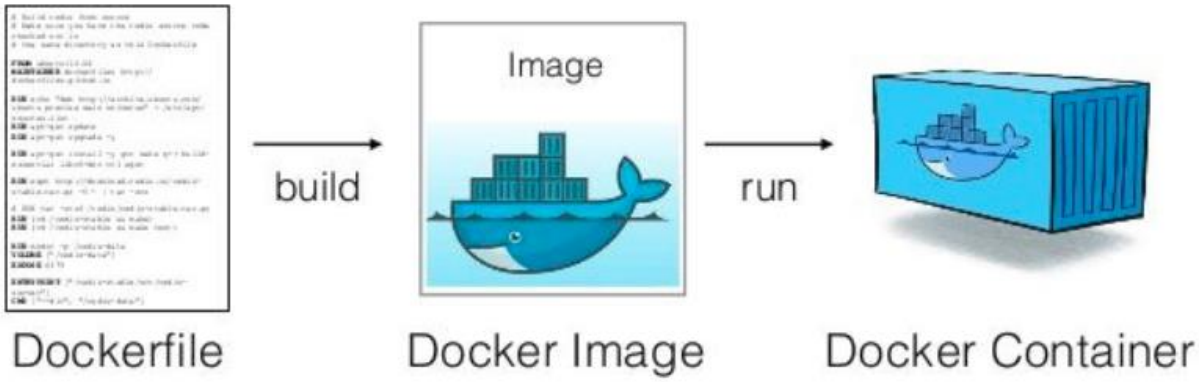
```
docker logs –f --tail=N <tên container>
```

Kiểm tra dung lượng mạng, CPU của container thì ta sử dụng lệnh:

```
docker stats <tên container>
```

9. Docker file

9.1. Docker file là gì?



Là một file text, có khả năng xây dựng image bằng việc đọc các câu lệnh trong file đó để pull image cơ sở và thực hiện thay đổi một số dữ liệu trong đó.

9.2. Cách xây dựng docker file

Các config cơ bản:

- FROM — chỉ định image gốc: python, unbutu, alpine…
- LABEL — cung cấp metadata cho image. Có thể sử dụng để add thông tin maintainer. Để xem các label của images, dùng lệnh docker inspect.
- ENV — thiết lập một biến môi trường.
- RUN — Có thể tạo một lệnh khi build image. Được sử dụng để cài đặt các package vào container.
- COPY — Sao chép các file và thư mục vào container.
- ADD — Sao chép các file và thư mục vào container.
- CMD — Cung cấp một lệnh và đối số cho container thực thi. Các tham số có thể được ghi đè và chỉ có một CMD.
- WORKDIR — Thiết lập thư mục đang làm việc cho các chỉ thị khác như: RUN, CMD, ENTRYPOINT, COPY, ADD,…
- ARG — Định nghĩa giá trị biến được dùng trong lúc build image.
- ENTRYPOINT — cung cấp lệnh và đối số cho một container thực thi.
- EXPOSE — khai báo port lắng nghe của image.
- VOLUME — tạo một điểm gắn thư mục để truy cập và lưu trữ data.

Ví dụ:

 # xây dựng image mới từ image centos:latest (CENTOS 7)
FROM centos:latest

 # Cập nhật các gói và cài vào đó HTTPD, HTOP, VIM

```
RUN yum update -y
RUN yum install httpd httpd-tools -y
RUN yum install epel-release -y \
   && yum update -y \
   && yum install htop -y \
   && yum install vim -y

#Thiết lập thư mục hiện tại
WORKDIR /var/www/html
# Copy tất cả các file trong thư mục hiện tại (.)  vào WORKDIR
ADD . /var/www/html

#Thiết lập khi tạo container từ image sẽ mở cổng 80
# ở mạng mà container nối vào
EXPOSE 80

# Khi chạy container tự động chạy ngay httpd
ENTRYPOINT ["/usr/sbin/httpd"]

#chạy terminate
CMD ["-D", "FOREGROUND"]
```

Sau đó, chạy lệnh để tạo thành image mong muốn:

```
docker build -t <tên image>:<tag> -f<tên docker file> .
```

Sau đó bạn có thể chạy docker run image vừa tạo để thành một container.

10. Docker compose

10.1. Docker compose là gì?

Compose là công cụ giúp định nghĩa và khởi chạy multi-container Docker applications. Trong Compose, chúng ta sử dụng Compose file để cấu hình application's services. Chỉ với một câu lệnh, lập trình viên có thể dễ dàng create và start toàn bộ các services phục vụ cho việc chạy ứng dụng.

10.2. Cách xây dựng docker compose

Khi triển khai ứng dụng, các thành phần chạy trên các container thì được gọi là các dịch vụ (service). Một ứng dụng trên docker có thể có nhiều service, ngoài các service thì nó còn có thể gồm các thành phần như mạng, ổ đĩa,.. thì tất cả những thành phần này được mô tả bằng file, gọi là docker-compose file có đuôi yml. Qua file này và những câu lệnh, thì docker có thể xây dựng, thiết lập và chạy các dịch vụ.

File docker-compose.yml được tổ chức gồm 4 phần:

Directive	Ý nghĩa
version	Chỉ ra version của file Compose
services	Với Docker, một service là tên của một container
networks	Phần này được sử dụng để cấu hình network cho ứng dụng. Bạn có thể cài đặt network mặc định hoặc định nghĩa network chỉ định cho ứng dụng
volumes	Gắn đường dẫn trên host machine được sử dụng trên container

Với phần config services, có một vài directive thường được sử dụng:

Directive	Ý nghĩa
image	Chỉ ra image được sử dụng để build container. Sử dụng directive với các image chỉ định trên host mechine hoặc trên Docker Hub.
build	Directive này có thể được sử dụng thay directive image. Chỉ ra vị trí của Dockerfile để build container
restart	Yêu cầu container restart khi hệ thống restart
volumes	Gắn đường dẫn trên host machine được sử dụng trên container
environment	Định nghĩa các biến môi trường truyền vào Docker khi chạy command
depends_on	Chọn các service được dùng là dependency cho container được xác định trong service hiện tại.
port	Kết nối port từ container đến host theo cách thức host:container

Directive	Ý nghĩa
links	Liên kết service này với bất kỳ service nào khác trong **Docker Compose** file bằng các chỉ rõ tên

Ta sẽ làm ví dụ, xây dựng một hệ thống gồm:

+ dịch vụ mysql
+ dịch vụ httpd
+ dịch vụ php

Chuẩn bị dữ liệu:

+ Đối với PHP, ta tạo file Dockerfile nằm trong đường dẫn mycode/php/Dockerfile

Nội dung Dockerfile như sau:

```
#image cơ sở
FROM php:7.3-fpm

# cài phần mở rộng
RUN docker-php-ext-install mysqli
RUN docker-php-ext-install pdo_mysql

#thiết lập thư mục làm việc mặc định
WORKDIR /home/sites/default
```

+ Đối với httpd:

Bước 1 ta lấy file config httpd.conf để trong thư mục mycode/httpd.conf bằng lệnh sau:

docker run --rm -v E:\3_Hoc_Trien_Khai_Du_An\1_Docker\2_Hoc\mycode\:/home/ httpd cp /usr/local/apache2/conf/httpd.conf /home/

Bước 2 chỉnh sửa file httpd.conf, sử dụng lệnh:

code httpd.conf

Bỏ comment phần load module:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so
```

Phần DocumentRoot và Directory set lại thư mục làm việc thành: home/sites/default

Sửa DirectoryIndex thành index.pho index.html

Thêm vào cuối file dòng lệnh:

```
AddHandler "proxy:fcgi://php-product:9000" .php
```

+ Đối với Mysql:

Bước 1 ta lấy file my.cnf để trong thư mục mycode/my.cnf bằng lệnh sau:

docker run --rm -v E:\3_Hoc_Trien_Khai_Du_An\1_Docker\2_Hoc\mycode\:/home/ mysql cp /etc/mysql/my.cnf /home/

Bước 2 chỉnh sửa file my.cnf, sử dụng lệnh:

code my.cnf

Thêm dòng sau ở cuối file:

```
default-authentication-plugin=mysql_native_password
```

Tạo file docker-compose.yml với nội dung sau:

```
#phiên bản của docker compose
version: '3'

#định nghĩa network có tên my-network
networks:
    my-network:
        driver: bridge

#định nghĩa volume có tên dir-site
volumes:
    dir-site:
        driver_opts:
            device: /mycode/sites
            o: bind


#định nghĩa services
services:
    #container PHP
    my-php:
        container_name: php-product
        build:
            dockerfile: Dockerfile
            context: ./php/
        hostname: php
        restart: always
        networks:
            - my-network
        volumes:
            - dir-site:/home/sites/

    #container HTTPD
    my-httpd:
        container_name: c-httpd01
        image: "httpd:latest"
        hostname: httpd
        restart: always
        networks:
            - my-network
        volumes:
            - dir-site:/home/sites/
```

```
      - ./httpd.conf:/usr/local/apache2/conf/httpd.conf
    ports:
      - "9999:80"
      - "443:443"

  #container MYSQL
  my-mysql:
    container_name: mysql-product
    image: "mysql:latest"
    restart: always
    networks:
      - my-network
    volumes:
      - ./db:/var/lib/mysql
      - ./my.cnf:/etc/mysql/my.cnf
    environment:
      - MYSQL_ROOT_PASSWORD=123abc
      - MYSQL_DATABASE=db_site
      - MYSQL_USER=siteuser
      - MYSQL_PASSWORD=sitepass
```

Sau đó chạy lệnh để start hệ thống:

```
docker-compose up
```