Tài liệu hướng dẫn sử dụng Docker

- Dùng để đóng gói toàn bộ các thông tin về dữ liệu trong 1 container để đảm bảo hoạt động trên các thiết bị khác ngoài thiết bị được dùng để phát triển
- Giảm thiểu việc thiết lập các thư viện, bộ phụ thuộc trên các thiết bị khác
- Phân biệt giữa container và virtual machine
  - Container
    - Môi trường biệt lập để khởi chạy các ứng dụng
    - Đòi hỏi ít tài nguyên hơn
    - Hoạt động trên OS của host
    - Khởi động nhanh hơn
  - Virtual machine
    - Giả lập phần cứng của một machine trên 1 phần cứng thật
    - Liên quan đến Hypervisors như
      - Virtual Box
      - VMware
      - Hyper-V (Windows only)
    - Cho phép chạy các phần mềm khác nhau với các thư viện, bộ phụ thuộc khác nhau giữa các máy ảo
    - Đòi hỏi chạy trên một OS hoàn chỉnh
    - Khởi động chậm và tiêu tốn nhiều tài nguyên
- Kiến trúc của Docker
  - Sử dụng kiến trúc client-server
    - Các thành phần client sẽ giao tiếp với thành phần server thông qua REST-ful API
    - Các thành phần server còn được gọi là Docker Engine
      - Các Docker Engine hỗ trợ việc xây dựng và vận hành các container
- Container
  - Là các process chạy trong OS
  - Được chia sẻ kernel của OS
    - Kernel OS
      - Là lõi nhân hoạt động của OS
      - Quản lý các ứng dụng và tài nguyên hệ thống
- Cách tải Docker trên Linux Fedora
  - Chạy câu lệnh 1 :
    - `sudo dnf -y install dnf-plugins-core`
    - `sudo dnf config-manager --add-repo https://download.docker.com/linux/fedora/docker-ce.repo`
  - Chạy câu lệnh 2 :
    - `sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`
  - Chọn `y` nếu khóa GPG trùng theo mẫu sau:
    - `060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35`
- Cách khởi chạy Docker:
  - Chạy câu lệnh `sudo systemctl start docker`
  - Để kiểm tra Docker Engine chạy thành công thì chạy câu lệnh
    - `sudo docker run hello-world`
    - Nếu có in ra confirmation và exit thì thành công
- Workflow khi làm việc với Docker
  - Để đảm bảo một ứng dụng có thể chạy trên Docker, ta cần phải thêm Docker-file trên ứng dụng đó
  - Docker-file
    - Là tập lệnh hướng dẫn Docker đóng gói ứng dụng thành 1 tệp image chứa tất cả mọi thứ để đảm bảo ứng dụng có thể hoạt động
  - tệp Image
    - Là một tệp hoạt động không cần đến OS
    - Đồng thời là một môi trường thực thi
    - Chứa các tệp của ứng dụng
    - Chứa các thư viện bên thứ 3
    - Chứa các biến môi trường
  - Sau khi kết xuất ra được tệp Image, Docker sẽ thực thi một container để sử dụng tệp Image đó
  - Container chứa ứng dụng đang thực thi trên tập Image khi này trở thành một process đặc biệt trong OS, giúp ứng dụng có thể hoạt động trên các development machine
    - Nghĩa là, thay vì chúng ta chạy ứng dụng trực tiếp thông qua gọi tên ứng dụng trên terminal thì, ta sẽ gọi kèm khai báo `docker run my-app` để ứng dụng chạy trên container của Docker
  - Để đảm bảo có thể phân phối được các image này thì từ Dev sẽ gửi các image này lên Docker-Hub để Test/Prod có thể tải về
  - Cho phép giảm thiểu việc bảo trì các tài liệu dài và phức tạp về việc xây dựng tệp Image của ứng dụng trên Docker
- Cách đóng gói thành tệp Image
  - Giả sử, chúng ta có một file `cpp` có tên là `helloworld`
  - File `cpp` này nằm trong thư mục `Docker Test`
  - Trong thư mục `Docker Test` , chúng ta tạo thêm 1 file tên là `Dockerfile` (lưu ý phải đúng theo tên này) mà không có đuôi tệp
  - Trong tệp Dockerfile, ta thực hiện trình tự như sau:
    - Chỉ định nền image (nền image là một khối file có sẵn có thể tìm thấy trên Dockerhub) với từ khóa `FROM`
      - Đối với tệp `cpp` , chúng ta không có nền image với tên `g++` nên có thể dùng thay thế một base image khác là `frolvlad/alpine-gxx`
      - Để minh bạch hơn đối với nền image có nhiều version, ta có thể thêm 1 dấu `:` và kèm tag ở sau như `latest` , `40` , ...
        - Ví dụ : `FROM frolvlad/alpine-gxx:latest`
    - Copy nội dung trong thư mục hiện tại vào 1 directory khác
      - Thực hiện với từ khóa `COPY`
        - Ví dụ : `COPY . /dircopypath`
          - Trong đó, `.` là chỉ định vị trí của thư mục hiện tại
    - Chỉ định thư mục hoạt động khi chạy Container
      - Thực hiện với từ khóa `WORKDIR`
        - Ví dụ : `WORKDIR ./dircopypath`
          - Trong đó, `.` là chỉ định vị trí của thư mục hiện tại

- Phân biệt giữa `RUN` và `CMD` trong Dockerfile
  - `RUN`
    - Lệnh cho phép tải các ứng dụng và package cho Container.
    - Giúp thực thi bất kỳ lệnh nào trên image hiện tại và tạo ra một layer mới trên image bằng cách xác nhận kết quả.
    - Cho phép chạy nhiều lệnh `RUN` trên 1 Dockerfile
  - `CMD`
    - Giúp cài đặt câu lệnh thực thi mặc định
    - Chỉ thực thi khi gọi `docker run ...` mà không có chỉ định lệnh
    - Chỉ cho phép chạy lệnh `CMD` cuối cùng trong Dockerfile. Các trước đó sẽ bị bỏ qua.
  - Sau khi phân biệt được giữa `RUN` và `CMD` , để đảm bảo đóng gói và thực thi được file `cpp` . Ta thực thi lệnh `RUN` để biên dịch file `cpp`
    - Lệnh thực thi `RUN g++ helloworld.cpp -o helloworld`
  - Sau khi thực thi xong lệnh `RUN` , ta gọi đến lệnh `CMD` để thực thi lệnh cuối cùng để chạy file `cpp`
    - Lệnh thực thi `CMD ./helloworld`
  - Sau khi hoàn tất Dockerfile
    - Lúc này ta bật Terminal và truy cập tới vị trí chứa Dockerfile của hệ thống.
    - Để đảm bảo Docker có thể hoạt động, ta thực thi liên tục các lệnh theo thứ tự sau:
      - `sudo su`
      - `systemctl start docker`
      - `systemctl enable docker`
      - `systemctl restart docker`
    - Thực thi xây dựng tệp image bằng câu lệnh sau
      - `docker build -t <name> .`
      - Trong đó
        - Tag `-t` giúp gọi dễ dàng gọi tên container
        - `.` chỉ định ví trị hiện tại khi thực thi lệnh `docker ...`
- Cách để push container lên Dockerhub
  - Trước hết, ta thực hiện việc đăng nhập trên Docker và Dockerhub
  - Sau đó, ta thực hiện đăng nhập trên Terminal thông qua lệnh sau
    - `docker login` hoặc `docker login -u <ur-username`
  - Tiếp đến thực hiện thay đổi tag của tệp Image trước khi push lên Dockerhub bằng lệnh sau
    - `docker tag <image-tag> <ur-Dockerhub-name>/<image-name>`
    - Ví dụ:
      - Ta đang có một container được khởi tạo với tên `hello-b` và ta cần đẩy container này lên Dockerhub.
      - Về mặc định, khi xây dựng tệp Image bằng lệnh `docker build -t ...` , ta sẽ tạo ra một container với tên và có tag của container mặc định là `latest`
      - Do đó, để thay đổi tag của container từ `latest` thành `test` thì ta thực hiện như sau:
        - `docker tag test shanghuang1811/hello-b`
        - Trong đó:
          - shanghuang1811 là tên Dockerhub giả sử
  - Cuối cùng ta thực hiện push tệp Image lên bằng câu lệnh
    - `docker push <ur-Dockerhub-name>/<image-name>:<tag>`
    - Ví dụ
      - `docker push shanghuang1811/hello-b:test`
- Các lệnh trong Docker Terminal
  - Lưu ý 1: Đối với các lệnh có sử dụng `&lt;container-id&gt;` thì chỉ cần đề cập đến các ký tự đầu của `&lt;container-id&gt;` sao cho có thể phân biệt được .
  - Lưu ý 2: Đối với các tag trên lệnh, có thể gộp các tag lại với nhau thành 1 tag duy nhất .
    - Ví dụ : `docker run -i -t ...` có thể viết thành `docker run -it`.
  - `docker run` : dùng để thực thi container
    - Nếu không có tệp Image ở thiết bị gọi lệnh thì sẽ tìm trên Dockerhub và pull về trước khi chạy
    - Về mặc định, khi thực thi lệnh sẽ chạy ở chế độ `attach` (nghĩa là thiết bị thực thi lệnh chỉ hoạt động trên console hoặc đầu ra của container đang thực thi chứ không thể làm gì khác)
    - Cho phép thực thi lệnh đi kèm
      - Ví dụ : `docker run <image-name> sleep 5`
    - Về mặc định, khi chạy các image thì Docker sẽ sử dụng tag mặc định là `latest`
    - Để có thể thực thi lệnh theo version mong muốn, ta thực hiện như sau:
      - `docker run <image-name>:<version>`
      - Để tra cứu các tag, tìm kiếm tên của các base image trên Dockerhub
    - Các option tag thực thi lệnh
      - `-d` : giúp thực thi lệnh ở chế độ `detach` (nghĩa là khi thực thi lệnh, thiết bị thực thi sẽ chạy container ở phần nền hệ thống mà không rằng buộc giới hạn ở chế độ `attach` , do đó người dùng có thể thực thi tiếp các lệnh hoặc làm việc khác ngoài console)
      - `-i` : giúp cho phép nhập liệu đối với các container có sử dụng nhập liệu đầu vào. Tuy nhiên, chúng sẽ làm mất đi các thiết lập Terminal trong dòng lệnh gọi đến xử lý nhập liệu
      - `-t` : giúp gọi đến Terminal giả lập để thực thi lệnh trên Terminal giả
      - `-p` : giúp mở một cổng liên kết đến một cổng được mở trên image
        - Ví dụ: khi một image `a_mi` chạy trên một URL là `http://111.23.4.5:500/` với địa chỉ IP là `111.23.4.5` và cổng là `500` . để giúp người sử dụng có thể sử dụng một cổng khác (giả sử là cổng `41`) mà vẫn có thể chạy tới URL hiện hữu trên image `a_mi` thì ta gọi lệnh như sau:
        - `docker run -p 41:500 a_mi`
        - Tuy nhiên, không thể mở 1 cổng để liên kết 1 cổng khác cho nhiều image khác nhau
      - `-v` : giúp liên kết dữ liệu sinh ra trong khi chạy container đến 1 địa chỉ bên trong bộ nhớ thực của thiết bị
        - Syntax lệnh: `docker run -v <path-directory>:<path-to-link> <image-id |image-name>`
          - Ở *Linux*, ta có thể dùng `$(pwd)` để chạy lệnh trên chính vị trí hiện tại
          - Ở *Windows*, ta phải viết đủ đường dẫn theo định dạng theo ví dụ: `C:\User\...:/data`
      - `-e`: dùng để thực thi các biến môi trường chỉ định cho các biến môi trường có trong image.
        - Syntax lệnh: `docker run -e <environment-specific> <image-id |image-name>`
        - Để kiểm tra các biến môi trường nào được phép thay đổi, thực hiện lệnh `docker inspect <container-id | container-name>` trên container đang chạy image hiện hữu và kiểm tra phần `"CONFIG"` .

- `docker attach <container-id | container-name>` : giúp thực thi chế độ `attach` trên container đang chạy.
- `docker ps` : dùng để liệt kê các container đang thực thi
  - `-a` : dùng để liệt kê các container đã thực thi trong quá khứ
- `docker stop <container-id | container-name>` : dùng để dừng thực thi một container
- `docker rm <container-id | container-name>` : dùng để xóa một container
  - Nếu sau khi chạy câu lệnh có trả về `<container-id | container-name>` nghĩa là xóa thành công
- `docker images | docker image ls` : dùng để liệt kê các image đang có trên thiết bị gọi lệnh
- `docker rmi <image-id | image-name>` : dùng để xóa một tệp Image
  - Để đảm bảo có thể xóa hoàn toàn, bắt buộc phải dừng mọi container | dependency đang sử image đó
- `docker pull <image-name>` : dùng để pull 1 tệp Image từ Dockerhub về
- `docker exec <container-id | container-name> <command>` : dùng để thực thi khối lệnh `<command>` khi container có `<container-id | container-name>` đang thực thi
- `docker inspect <container-id | container-name>` : dùng để liệt kê chi tiết các thông tin về 1 container dưới định dạng JSON
- `docker logs <container-id | container-name>` : dùng để hiển thị nội dung đầu ra của container