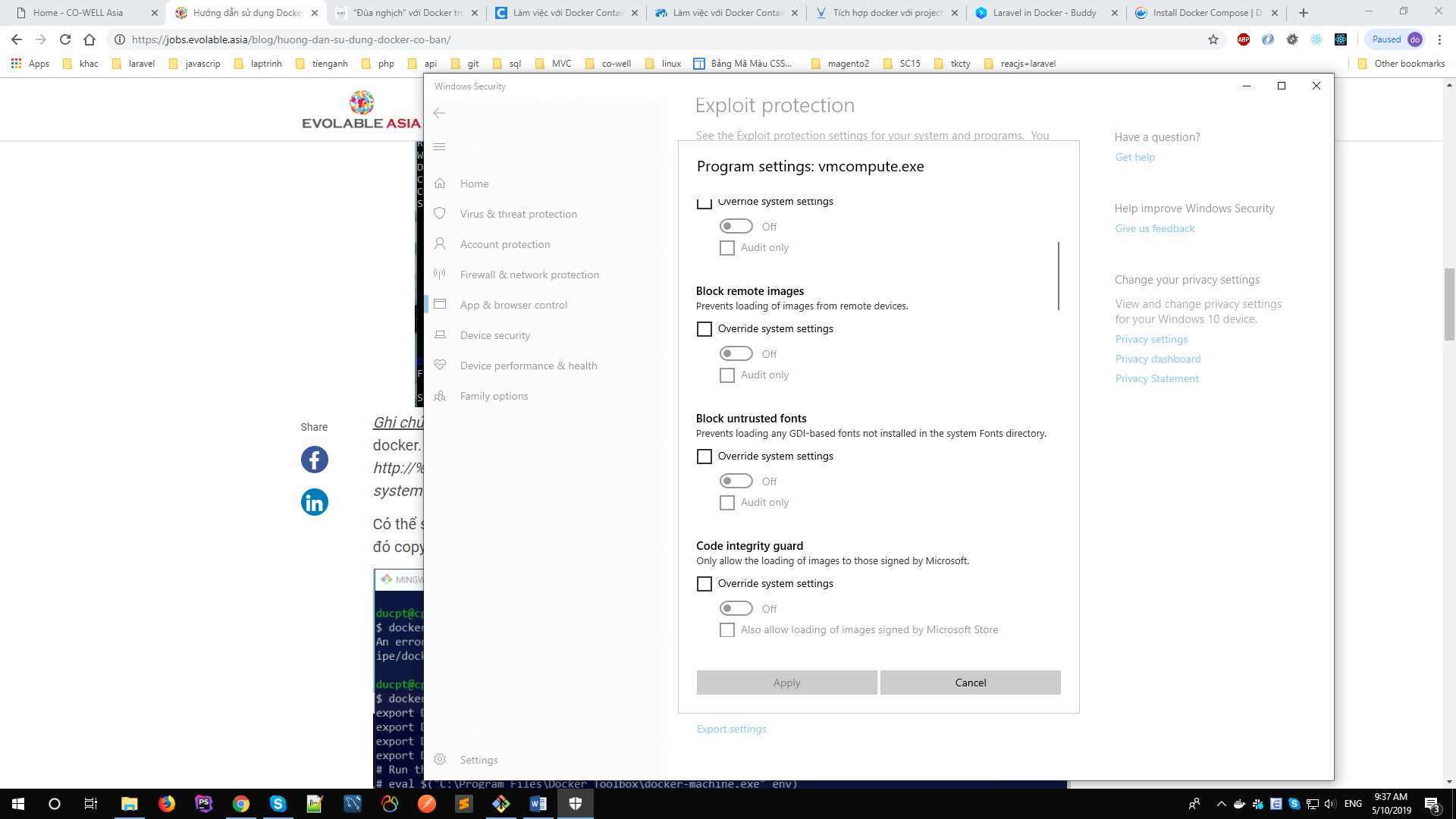
# Cài đặt



# Tạo Docker Images từ Dockerfile

để viết được một *Dockerfile* hoàn chỉnh, các bạn cần hiểu được các command của *Dockerfile*:

* **FROM**: Là base image để chúng ta tiến hành build một image mới. Command này phải được đặt trên cùng của Dockerfile
* **MAINTAINER**: Command này là tùy chọn, có thể có hoặc không. Nó chưa thông tin của người tiến hành xây dựng lên images.
* **RUN**: Sử dụng khi muốn thực thi một command trong quá trình build image
* **COPY**: Copy một file từ host machine tới docker image. Có thể sử dụng URL cho tệp tin cần copy, khi đó docker sẽ tiến hành tải tệp tin đó đến thư mục đích.
* **ENV**: Định nghĩa các biến môi trường
* **CMD**: Sử dụng khi muốn thực thi (execute) một command trong quá trình build một container mới từ docker image
* **ENTRYPOINT**: Định nghĩa những command mặc định, cái mà sẽ được chạy khi container running.
* **WORKDIR**: Định nghĩa directory cho **CMD**
* **USER**: Đặt user hoặc UID cho container được tạo bởi image
* **VOLUME**: Cho phép truy cập / liên kết thư mục giữa các container và máy chủ (host machine)

Bây giờ, hãy thực hành bằng việc build một image thao tác với các câu lệnh trên.

Trước khi tiến hành các bước sau, hãy đảm bảo rằng trên máy của bạn đã được cài đặt [Docker](https://docs.docker.com/engine/installation/).

**Bước 1**: Tạo *Dockerfile*

Trong bước này, chúng ta sẽ tạo một directory mới cho dockerfile và tiến hành định nghĩa những thứ chúng ta muốn làm với dockerfile này. Tạo một đường dẫn chứa Dockerfile mới theo các bước sau:

$ mkdir viblo

$ cd viblo

$ touch Dockerfile

cài đặt **Nginx** và **PHP-FPM 7** dựa trên image *Ubuntu16.04* của Docker. Ngoài ra, tồi cần thêm Supervisord cho phép khởi động Nginx và Php-fpm mỗi khi chúng ta start container trong cùng một command. Mở Dockerfile được tạo từ bước trên bằng editor quen thuộc của bạn và bắt đầu chỉnh sửa: Dòng đầu tiên của file, hãy thêm base image mà chúng ta muốn sử dụng bằng lệnh **FROM** ở đây là image Ubuntu16.04.

#Download base image ubuntu 16.04

FROM ubuntu:16.04

Tiến hành update Ubuntu software bên trong Dockerfile bằng lệnh **RUN**:

# Update Ubuntu Software repository

RUN apt-get update

Tiếp đến, tiền hành cài đặt Nginx, php-fpm, supervisord cho image của chúng ta bằng câu lệnh sau:

# Install nginx, php-fpm and supervisord from ubuntu repository

RUN apt-get install -y nginx php7.0-fpm supervisor && \

rm -rf /var/lib/apt/lists/\*

Ở giai đoạn này, tất cả các ứng dụng đã được cài đặt, việc chúng ta cần làm tiếp theo là tiến hành config cho chúng. Chúng ta sẽ tiến hành config Nginx để nó có thể sử lý các ứng dụng PHP bằng cách chỉnh sửa config của default virtual host. Chúng ta có thể thay thế mới hoàn toàn bằng file config của mình hoặc có thể chỉnh sửa với lệnh 'sed'.

#Define the ENV variable

ENV nginx\_vhost /etc/nginx/sites-available/default

ENV php\_conf /etc/php/7.0/fpm/php.ini

ENV nginx\_conf /etc/nginx/nginx.conf

ENV supervisor\_conf /etc/supervisor/supervisord.conf

# Enable php-fpm on nginx virtualhost configuration

COPY default ${nginx\_vhost}

RUN sed -i -e 's/;cgi.fix\_pathinfo=1/cgi.fix\_pathinfo=0/g' ${php\_conf} && \

echo "\ndaemon off;" >> ${nginx\_conf}

Tiếp theo, tiến hành config Supervisord cho Nginx và PHP-FPM. Chúng ta sẽ thay thế file config default của Supervisord bằng một file config mới sử dụng lệnh '**COPY** của Dockerfile

#Copy supervisor configuration

COPY supervisord.conf ${supervisor\_conf}

Bây giờ, tạo một đường đẫn mới để lưu php-fpm sock file và cấp quyền owner cho /var/www/html và PHP thành www-data

RUN mkdir -p /run/php && \

chown -R www-data:www-data /var/www/html && \

chown -R www-data:www-data /run/php

Tiếp theo, định nghĩa volumn để chúng ta có thể mount list directory vào host machine:

# Volume configuration

VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d", "/var/log/nginx", "/var/www/html"]

Cuối cùng, định nghĩa default command cho container và mở port cho http và https. Chúng ta sẽ tạo một file [start.sh](http://start.sh/) cho default **CMD** command khi mà container được khởi chạy.

# Configure Services and Port

COPY start.sh /start.sh

CMD ["./start.sh"]

EXPOSE 80 443

Sau khi hoàn tất các bước trên, bạn có được file Dockerfile như sau:

#Download base image ubuntu 16.04

FROM ubuntu:16.04

# Update Software repository

RUN apt-get update

# Install nginx, php-fpm and supervisord from ubuntu repository

RUN apt-get install -y nginx php7.0-fpm supervisor && \

rm -rf /var/lib/apt/lists/\*

#Define the ENV variable

ENV nginx\_vhost /etc/nginx/sites-available/default

ENV php\_conf /etc/php/7.0/fpm/php.ini

ENV nginx\_conf /etc/nginx/nginx.conf

ENV supervisor\_conf /etc/supervisor/supervisord.conf

# Enable php-fpm on nginx virtualhost configuration

COPY default ${nginx\_vhost}

RUN sed -i -e 's/;cgi.fix\_pathinfo=1/cgi.fix\_pathinfo=0/g' ${php\_conf} && \

echo "\ndaemon off;" >> ${nginx\_conf}

#Copy supervisor configuration

COPY supervisord.conf ${supervisor\_conf}

RUN mkdir -p /run/php && \

chown -R www-data:www-data /var/www/html && \

chown -R www-data:www-data /run/php

# Volume configuration

VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d", "/var/log/nginx", "/var/www/html"]

# Configure Services and Port

COPY start.sh /start.sh

CMD ["./start.sh"]

EXPOSE 80 443

Tiếp theo, tạo các file default, supervisord.conf, [start.sh](http://start.sh/) cùng directory với Dockerfile

$ touch default

$ touch supervisord.conf

$ touch start.sh

Trong file default, thêm nội dung sau:

server {

listen 80 default\_server;

listen [::]:80 default\_server;

root /var/www/html;

index index.html index.htm index.nginx-debian.html;

server\_name localhost;

error\_log /var/log/nginx/error.log;

access\_log /var/log/nginx/access.log;

location / {

try\_files $uri $uri/ =404;

}

location ~ \.php$ {

include snippets/fastcgi-php.conf;

fastcgi\_pass unix:/run/php/php7.0-fpm.sock;

}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

Trong file supervisord.conf, thêm nội dung sau:

[unix\_http\_server]

file=/dev/shm/supervisor.sock ; (the path to the socket file)

[supervisord]

logfile=/var/log/supervisord.log ; (main log file;default $CWD/supervisord.log)

logfile\_maxbytes=50MB ; (max main logfile bytes b4 rotation;default 50MB)

logfile\_backups=10 ; (num of main logfile rotation backups;default 10)

loglevel=info ; (log level;default info; others: debug,warn,trace)

pidfile=/tmp/supervisord.pid ; (supervisord pidfile;default supervisord.pid)

nodaemon=false ; (start in foreground if true;default false)

minfds=1024 ; (min. avail startup file descriptors;default 1024)

minprocs=200 ; (min. avail process descriptors;default 200)

user=root ;

; the below section must remain in the config file for RPC

; (supervisorctl/web interface) to work, additional interfaces may be

; added by defining them in separate rpcinterface: sections

[rpcinterface:supervisor]

supervisor.rpcinterface\_factory = supervisor.rpcinterface:make\_main\_rpcinterface

[supervisorctl]

serverurl=unix:///dev/shm/supervisor.sock ; use a unix:// URL for a unix socket

; The [include] section can just contain the "files" setting. This

; setting can list multiple files (separated by whitespace or

; newlines). It can also contain wildcards. The filenames are

; interpreted as relative to this file. Included files \*cannot\*

; include files themselves.

[include]

files = /etc/supervisor/conf.d/\*.conf

[program:php-fpm7.0]

command=/usr/sbin/php-fpm7.0 -F

numprocs=1

autostart=true

autorestart=true

[program:nginx]

command=/usr/sbin/nginx

numprocs=1

autostart=true

autorestart=true

Trong file [start.sh](http://start.sh/), thêm nội dung:

#!/bin/sh

/usr/bin/supervisord -n -c /etc/supervisor/supervisord.conf

Thêm câu lệnh sau vào file Dockerfile để có thể file [start.sh](http://start.sh/) có thể thực hiện được.

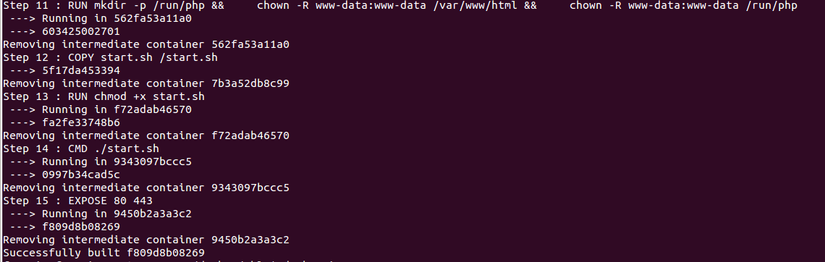
RUN chmod +x start.sh

Đến đây, việc tạo Dockerfile coi như hoàn tất.

**Bước 2**: Build Docker image và tạo container mới trên image vừa tạo

Chạy command sau để build image từ Dockerfile đã được tạo ở trên:

docker build -t nginx\_image .

Câu lệnh trên sẽ tạo ra 1 image mới có tên là nginx\_image. Sau khi câu lệnh hoàn tất, trên màn hình của bạn sẽ hiển thị.

Tiến hành kiểm tra image bằng lệnh

$ docker images

Kết quả thu được:https://viblo.asia/uploads/f0cc61fa-677e-47d1-a13b-d72c3fb15db2.png

Như vậy là chúng ta đã build thành công 1 Docker image. Bây giờ hãy build một container mới dựa trên image đó. Để tiện cho việc test, hãy tạo thêm thư mục webroot.

mkdir -p /webroot

Bây giờ, chạy lệnh sau để tạo container mới từ nginx\_image ở trên:

docker run -d -v /webroot:/var/www/html -p 9000:80 --name viblo\_test nginx\_image

https://viblo.asia/uploads/85c08a37-0157-49e5-8e23-bfabfbac5cdc.png

**Note**:

* --name viblo\_test nginx\_image :Tạo mới container với tên 'viblo\_test', dựa trên docker image 'nginx\_images'.
* -p 9000:80 : 'viblo\_test' container chạy trên cổng 9000 của host machine
* -v /webroot:/var/www/html: Thư mục /webroot ở trên host machine sẽ được rewrite vào thư mục /var/www/html ở trong container.

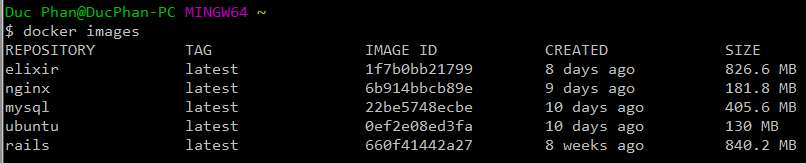
**Bước 3**: Testing Nginx and PHP-FPM in the Container

Tạo file index.html trong thư mục /webroot với nội dung sau:

<h1>Nginx and PHP-FPM 7 inside Docker Container</h1>

# **IV. Các lệnh cơ bản thường dùng:**

## **1. Hiển thị danh sách các images :**

.png” alt=”” width=”806″ height=”163″ />docker images

## **2. Tải image về local**

docker pull <name\_image:tag>

(phần :tag là options, nếu để trống  thì mặc định download bản latest )

Ví dụ:  docker pull ubuntu   => download ubuntu latest

docker pull ubuntu:14.04 => download ubuntu version 14.04

Truy cập trang <https://hub.docker.com/> , nơi lưu trữ các images tập trung để tìm images cần dùng.

## **3. Chạy một image**

docker run –name <tên\_container> -v <thư mục trên máy tính>:<thư mục trong container> -p<port\_máy tính>:<port\_container>  <image name> bash

Ví dụ: docker run –name  eva\_nginx  -p 80:80 -d nginx

* docker run : lệnh chạy của docker
* –name: đặt tên cho container ở đây là  eva\_nginx . Name này là duy nhất, không thể tạo trùng, nếu không đặt thì docker tự genate.
* -p mở port container ra ngoài IP public  **192.168.99.100**
* -d bật chế độ chạy background
* nginx: tên images

## **4. Liệt kê các container**

docker ps -a (liệt kê tất cả các container)

docker ps  ( chỉ liệt kê các container đang chạy background)

## **5. Dừng container đang chạy**

docker stop <container\_id hoặc name\_container>

docker stop $(docker ps –a –q) (Dừng tất cả các docker đang dùng)

## **6. Khởi động lại container đã dừng**

docker start <container\_id hoặc name\_container>

## **7. Xóa container không còn sử dụng**

docker rm <container\_id hoặc name\_container>

docker rm $(docker ps -a -q)   (Xóa tất cả các docker)

## **8. Truy cập vào 1 container đang chạy**

docker exec -it <container\_id hoặc name\_container> bash

## **9. Export bản container**

docker export <container\_id hoặc name\_container> | gzip > file\_export.tar.gz

## **10. Import container => image**

zcat file\_export.tar.gz | docker <new\_name\_image>Sau khi chạy xong. Chạy lệnh docker images để kiểm tra lại trong danh sách list images.

Ngoài ra còn nhiều lệnh khác. Sử dụng lệnh docker -h  để xem chi tiết.

# Câu lệnh cơ bản với laravel

1. Để build hai container là "webserver" và "db" vừa định nghĩa ở file "docker-compose.yml" bằng cách chạy lệnh

docker-compose build

Lệnh này sẽ tải image ở dockerhub nên mất một vài phút (tùy thuộc vào mạng nhé)

1. Sau khi chạy xong lệnh trên mà không báo lỗi lầm gì là đã hành công 96,69% rồi nhé. Tiếp tục chạy lệnh

Docker-compose up

lệnh này để khởi động các container mà đã định nghĩa ở file docker-compose.yml Chạy lệnh

docker ps

để kiểm tra các container vừa build có khởi chạy chưa.

1. Kết nối với db. Chạy lệnh

docker inspect laravel\_db\_1

với 'laravel\_db\_1' là tên của container mysql:5.7. Các bạn kéo xuống dưới và để ý dòng : IpAdress : 172.0.0.4 (có thể khác nhau) đó là địa chỉ host của mysql dùng để kết nối. còn thông tin user và password các bạn điền như ở file docker-compose.yml

1. Đừng quên chạy lệnh

chmod -R 777 /storage bootstrap/cache

1. Để lấy địa chỉ ip của host web các bạn chạy lệnh

docker inspect laravel\_webserver\_1

Với 'laravel\_webserver\_1' là tên sever web mà chúng ta đã định nghĩa ở file docker-compose.yml. Các bạn chú ý đến dòng IpAdress : 172.0.0.2 (có thể khác nhau) copy địa chỉ ip past vào trình duyệt để xem kết quả nhé.

1. Để làm việc trong máy ảo các bạn chạy lệnh sau

docker exec -it laravel\_webserver\_1 bash

Thử chạy lệnh

php artisan make:auth

1. Docker down type

Usage: down [options]

Options:

--rmi type Remove images. Type must be one of:

'all': Remove all images used by any service.

'local': Remove only images that don't have a

custom tag set by the `image` field.

-v, --volumes Remove named volumes declared in the `volumes`

section of the Compose file and anonymous volumes

attached to containers.

--remove-orphans Remove containers for services not defined in the

Compose file

-t, --timeout TIMEOUT Specify a shutdown timeout in seconds.

(default: 10)