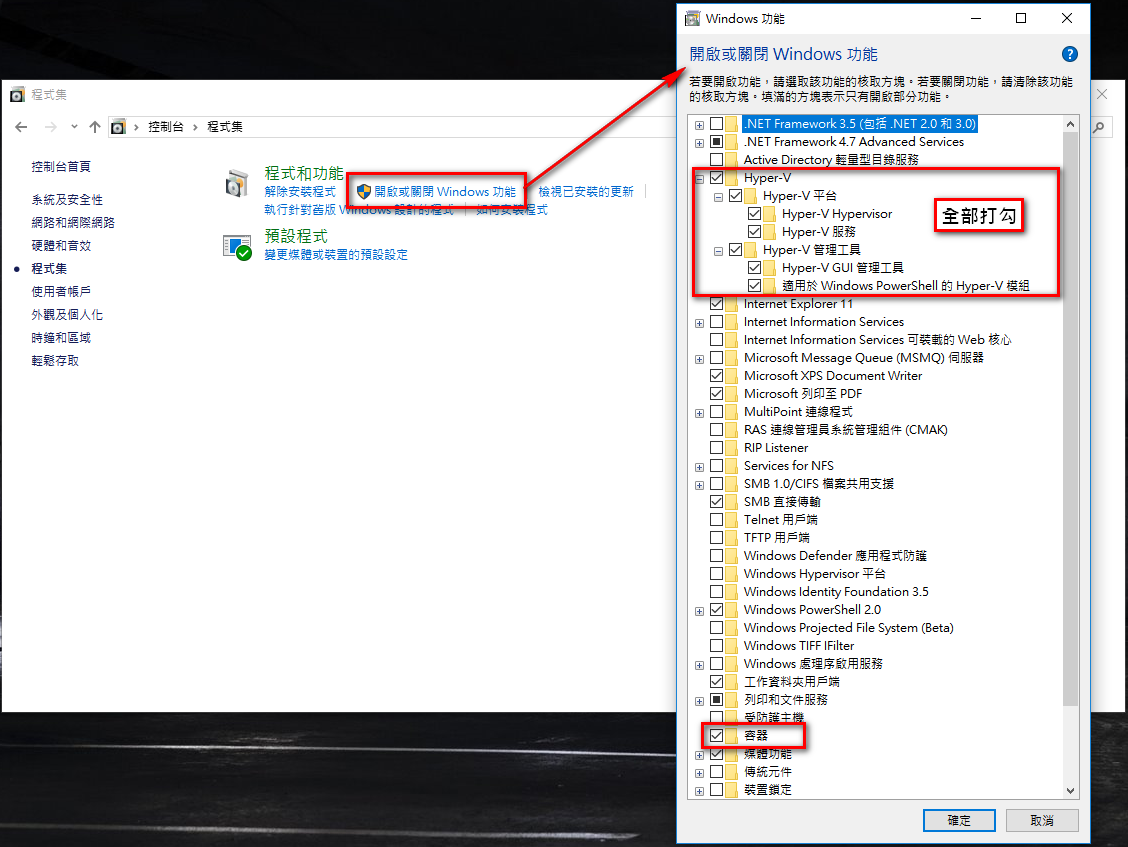
# Docker

研究員：Toby\_huang

1. Docker for Window

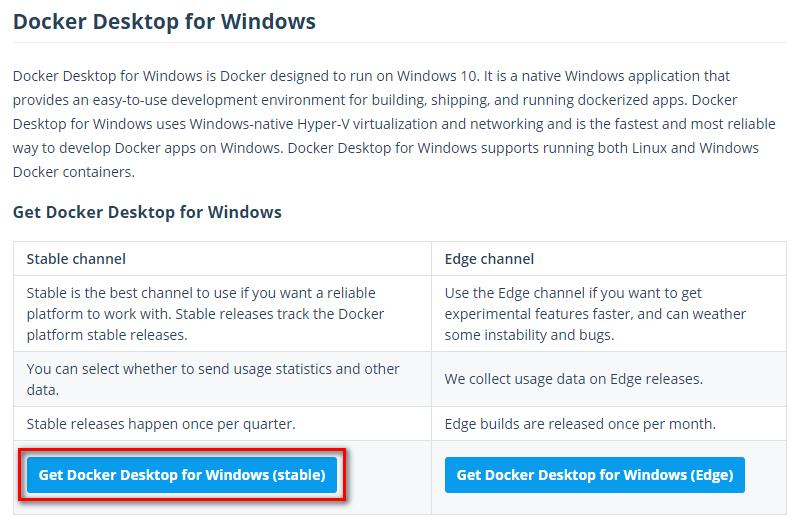
環境：

Docker Windows的所需條件，自然就是Windows10，且必須為Bulid10586以上版本而且必須啟用Hyper-V和容器功能：如下圖



下載：

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

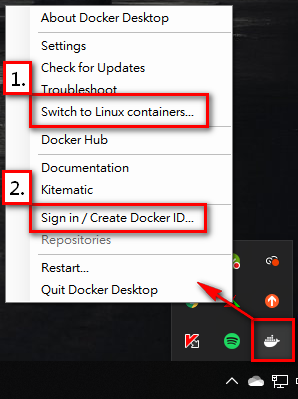


下載並安裝，再來要去<https://hub.docker.com/>申請一個DockerID

再來在安裝一個Kitematic，他是GUI介面，方便你使用Docker

你先開啟桌面上的Docker Desktop，在桌面的右下角可以看到一個屬於他的icon，對那個icon點右鍵，你就會在彈跳出來的視窗中找到Kitematic他會幫你導到下載的地方，也會跟你說要放置在docker資料夾裡的哪個地方。

使用：

開啟桌面上的Docker Desktop(如右圖)，在桌面的右下角可以看到一個海豚的標誌，點擊他會開啟一個小視窗(如右圖)

其中兩點：

1. Docker有些容器(container)是只能在linux環境下運行，有些則是在windows才能運行，這邊可以點擊切換。
2. 這邊登入你DockerID以後可以做push跟pull還有些相關指令。
3. Docker for Mac

環境：

基本上Mac不用管什麼環境，他本身就是linux，唯一要注意的應該就是看看你的mac有沒有Homebrew，他是mac系統常用的一個套件管理工具，我們等等也會用這個去安裝Docker。

打開終端機，輸入指令ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)"

安裝途中可能會需要你輸入一下你的登入密碼，應該有兩次。

安裝好以後就更新一下，

brew update

brew tap homebrew/dupes

下載：

打開終端機，輸入指令

brew cask install docker

1. 視窗化UI管理Docker  
   <https://github.com/docker/kitematic/releases>

有for Windows 以及 for Mac的

操作(比較常用的指令)：

* docker run containerID ：執行容器
* docker stop containerID ：暫停容器
* docker stop $(docker ps -aq)：停止所有容器
* docker start containerID ：恢復容器
* docker inspect：檢視容器詳細資訊
* docker rm containerID：移除容器
* docker rm $(docker ps -aq)：移除所有容器
* docker ps：查看目前運行的container
* docker ps –a：查看全部的container(包括沒在運行的)
* docker images：查看目前images
* docker rmi imageID：移除映像檔
* docker rmi $(docker images -q)：移除所有映像檔
* docker --version：查詢相關版本
* docker-compose --version：查詢相關版本
* docker-machine --version：查詢相關版本
* docker run [options] IMAGE[:tag|@digest] [command] [arg…]

舉個例子

Docker run –d –p 80:80 –name my\_image nginx

-d 代表在Detached(背景)執行，如不加 –d ，預設會foreground(前景執行)

-p 代表將畚箕的80port的所有流量轉發到container中的80port

--name 設定container的名稱

再舉一個例子

Docker run –it –rm busybox

--rm 代表當 exit container時，會自動移除container

* docker start [options] container [container…]
* docker system prune --volumes –all：刪除docker所有的一切

實作一：

首先建立以後專門放專案的資料夾

mkdir ~/projects

安裝Laradock

cd ~/

git clone https://github.com/laradock/laradock.git Laradock

將我幫你準備好的資料夾裡面的.env檔移動到laradock資料夾裡(這邊window跟mac資料夾路徑會不一樣)

Windows： mv ~/Desktop/dockerSettingFiles/.env ~/Laradock

Mac： mv ~/dockerSettingFiles/.env ~/Laradock

將我幫你準備好的資料夾裡面的php7.2.ini檔移動到laradock/php-fpm資料夾裡(這邊window跟mac資料夾路徑會不一樣)

Windows： mv ~/Desktop/dockerSettingFiles/php7.2.ini ~/Laradock/php-fpm

Mac： mv ~/dockerSettingFiles/php7.2.ini ~/Laradock/php-fpm

將我幫你準備好的資料夾裡面的default.conf檔移動到laradock/nginx/sites資料夾裡(這邊window跟mac資料夾路徑會不一樣)

Windows： mv ~/Desktop/dockerSettingFiles/default.conf ~/Laradock/nginx/sites

Mac： mv ~/dockerSettingFiles/default.conf ~/Laradock/nginx/sites

然後啟動 Laradock 服務

docker-compose up -d nginx mysql phpmyadmin

這樣就已經架好虛擬機了，在網頁連結輸入localhost或是127.0.0.1就可以看到畫面了。

之後你可以把專案都丟在一開始我們創建的projects資料夾裡。

到這邊為止是for不需要資料庫的同仁，接下來我們要繼續設定資料庫的部分。

將我幫你準備好的資料夾裡面的my.cnf檔移動到laradock/mysql資料夾裡(這邊window跟mac資料夾路徑會不一樣)

Windows： mv ~/Desktop/dockerSettingFiles/my.cnf ~/Laradock/mysql

Mac： mv ~/dockerSettingFiles/my.cnf ~/Laradock/mysql

Windows

mv ~/Desktop/dockerSettingFiles/iconkit.back.conf ~/Laradock/nginx/sites

mv ~/Desktop/dockerSettingFiles/iconkit.front.conf ~/Laradock/nginx/sites

mv ~/Desktop/dockerSettingFiles/project.management.conf ~/Laradock/nginx/sites

mv ~/Desktop/dockerSettingFiles/webbbin.back.conf ~/Laradock/nginx/sites

mv ~/Desktop/dockerSettingFiles/webbbin.front.conf ~/Laradock/nginx/sites

Mac

mv ~/dockerSettingFiles/iconkit.back.conf ~/Laradock/nginx/sites

mv ~/dockerSettingFiles/iconkit.front.conf ~/Laradock/nginx/sites

mv ~/dockerSettingFiles/project.management.conf ~/Laradock/nginx/sites

mv ~/dockerSettingFiles/webbbin.back.conf ~/Laradock/nginx/sites

mv ~/dockerSettingFiles/webbbin.front.conf ~/Laradock/nginx/sites

實作二：

Laradock是一個Docker File，是一個用Docker建置的PHP環境。

首先建立Laravel專案資料夾（依個人習慣，這邊是在家目錄底下建立projects資料夾）

mkdir ~/projects

安裝Laradock（依個人習慣，這邊是在家目錄底下建立Laradock）

cd ~/

git clone https://github.com/laradock/laradock.git Laradock

製作 Laradock 環境檔（複製 Laradock 底下的 env-example 模版，命名為 .env，Laradock 將吃 .env 的設定）

cd Laradock

cp env-example .env

進入 .env 環境檔，修改相關設定

vim .env

（於 .env 內）

站點路徑：

APP\_CODE\_PATH\_HOST=../projects

php版本：

PHP\_VERSION=7.2

mysql版本：

MYSQL\_VERSION=5.7

工作區的安裝軟件 按照需要修改為true：

WORKSPACE\_INSTALL\_NPM\_BOWER=true

WORKSPACE\_INSTALL\_WORKSPACE\_SSH=true

WORKSPACE\_INSTALL\_XDEBUG=true

WORKSPACE\_INSTALL\_MONGO=true

時區：

WORKSPACE\_TIMEZONE=Asia/Taipei

PHP

Php-fpm資料夾裡找到php7.2.ini修改設定  
 修改上傳文件大小限制

memory\_limit = 256M

post\_max\_size = 30M

upload\_max\_filesize = 20M

啟動 Laradock 服務

docker-compose up -d nginx mysql

進入 Workspace 容器

docker-compose exec workspace bash

因為剛剛已經將 Laradock 的預設根目錄指向 ../projects/laravel-test所以進來 Workspace 這時候的 /var/www/ 應該是在 laravel-test 資料夾內了

現在我們可以建立 laravel 專案

composer create-project laravel/laravel --prefer-dist

這邊你也可以git clone任何專案進來

接下來我們重啟 Laradock，在瀏覽器網址列輸入 localhost 應該就可以看到 Laravel 專案了

docker-compose down

docker-compose up -d nginx mysql

設定nginx基本conf

cd ~/Laradock/nginx/sites  
 vim default.conf

（於 default.conf 內）

server\_name localhost;

root /var/www;

index index.php index.html index.htm;

location / {

autoindex\_localtime on;

autoindex\_exact\_size off;

autoindex on;

try\_files $uri $uri/ /index.php$is\_args$args;

}

---------------------------------到這為止就已經可以連上專案了---------------------------------

建立站台（選用）

將專案設定運行在 Nginx 的虛擬站台

Laradock 本身已提供 Nginx 虛擬站台的設定檔模板，複製一份即可

cd ~/Laradock/nginx/sites

cp laravel.conf.example laravel.test.conf

打開 laravel.test.conf，修改根目錄laravel.test.conf

vim

與建立 laravel 專案時一樣，因為一開始在 .env 已經將 Laradock 的預設根目錄指向 ../projects/laravel-test，所以 laravel.test 的根目錄也是從 ../projects/laravel-test 開始，故 /var/www/專案名稱/public = ../projects/laravel-test/專案名稱/public

（於 laravel.test.conf 內）

server\_name laravel.test;

root /var/www/laravel/public

設定 Host

我們設定的 Nginx 虛擬站台名稱 laravel.test(基本上跟你的專案名稱有點相關比較好認)，但因為它實際上不存在，所以我們需要在本機上設定 DNS 對應

打開 /etc/hosts 檔案

sudo vim /etc/hosts

在最下方新增一行

127.0.0.1 laravel.test

重新啟動 Laradock

cd ~/Laradock/

docker-compose down

docker-compose up -d nginx mysql

接下來在瀏覽器網址列輸入 laravel.test，應該就會被導向 laravel-test/public 資料夾，與 localhost 呈現的頁面一樣

設定MySQL

預設使用最後一版，目前是 MySQL 8.0，預設帳密都是 root，這可以在 Laradock 中的 .env 搜尋 “MYSQL\_ROOT\_PASSWORD” 找到

vim .env

/MYSQL\_ROOT\_PASSWORD

以下是預設值↓

MYSQL\_VERSION=latest

MYSQL\_DATABASE=default

MYSQL\_USER=default

MYSQL\_PASSWORD=secret

MYSQL\_PORT=3306

MYSQL\_ROOT\_PASSWORD=root

MYSQL\_ENTRYPOINT\_INITDB=./mysql/docker-entrypoint-initdb.d

我們先離開，修改密碼驗證的模式

sudo vim ~/laradock/mysql/my.cnf

[mysqld]

sql-mode="STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION"

character-set-server=utf8

default\_authentication\_plugin=mysql\_native\_password <----加入這行，改用 mysql\_native\_password

我們重啟 Laradock

sudo docker-compose down

sudo rm -rf ~/.laradock/data <----務必刪除

sudo docker-compose up -d nginx mysql phpmyadmin

因為是採用 Docker 映像檔，若進入 mysql 要使用 docker-compose 指令

sudo docker-compose exec mysql bash

然後修改密碼，我們使用 root 登入 mysql，修改 Laradock 預設時新增的使用者叫做 default 的密碼

mysql -uroot –proot

> ALTER USER 'default'@'%' IDENTIFIED WITH mysql\_native\_password BY 'a\_A@123456';

> GRANT ALL PRIVILEGES ON \* . \* TO 'default'@'%';

> select user, host, plugin from mysql.user;

> FLUSH PRIVILEGES;

前往 http://localhost:8080 進入 phpmyadmin，Server 輸入 mysql，Username 與 Password 輸入剛剛修改後的：default/a\_A@123456。

sudo docker-compose up -d mysql phpmyadmin

打開 http://localhost:8080 就可進入。若要改 port ，可以到 Laradock 的 .env 修改 PMA\_PORT=8080