\* 入門基礎

demo 1 在容器中架設簡易apache伺服器環境

學習目標:

1. 基礎Docker指令操作

2. 映象檔操作: 打包,刪除,匯出,匯入, 列出

3. 容器基本操作: 運行, 停止, 移除, 列出, 從容器製作映象檔

操作步驟:

step1. 轉換工作目錄到demo1，觀察demo1資料夾底下有什麼

cd demo1

step2. 製作映象檔

docker build -t apache-server:0.0.1 .

step3. 觀察本地端倉庫擁有的映象檔

docker images

step4. 運行映象檔( 確認本地端port80沒有其他程序佔用 )

docker run -p 80:80 -dit --name apache-app apache-server:0.0.1

step5. 到瀏覽器上方搜尋列輸入(正確的話會有一隻鯨魚)

localhost/docker.html

step6. 使用終端機進入到容器環境

docker exec -it apache-app /bin/sh

step7. 輸入ls後案enter看一下有什麼東西

ls

step8. 轉移工作目錄，並輸入ls觀察有什麼東西

cd htdocs

ls

cat index.html

cat docker.html

cat Dockerfile

step9. 輸入exit離開終端機

exit

step10. 觀察所有的容器狀態

docker ps -a

step11. 終止容器

docker stop apache-app

step12. 觀察所有的容器狀態

docker ps -a

step13. 移除容器

docker rm apache-app

step14. 儲存映象檔

sudo docker save -o apache-server.tar apache-server:0.0.1

step15. 映象檔更改名稱

docker tag apache-server:0.0.1 apache-server:0.0.2

step16. 刪除映像檔

docker rmi apache-server:0.0.1

step17. 載入映象檔

sudo docker load --input apache-server.tar

step18. 再次運行容器

docker run -p 80:80 -it --name apache-app apache-server:0.0.1 /bin/sh

step19. 製作一個txt檔,現在容器狀態已經被我們改變了

echo "hehehe" >> funny.txt

cat funny.txt

step20. 離開終端機

exit

step21. 觀察容器狀態

docker ps -a

step22. 重啟容器

docker start apache-app

step23. 進入容器並觀察目錄下的東西( 應該會發現funny.txt還在 )

docker exec -it apache-app /bin/sh

ls

step24. 將這個容器製作成映象檔

sudo docker commit -m "Add funny.txt" -a "Terry\_Huang" apache-app funny:v1

step25. 觀察倉庫裡的映象檔(應該會發現多了一個 funny:v1)

docker images

step26. 觀察剛剛製作的映象檔的訊息,作者

docker inspect funny:v1 | grep "Author"

docker inspect funny:v1 | grep "Comment"

step27. 從剛剛製作的映象檔運行一個容器,並觀察目錄底下有沒有funny.txt

docker run -it --name apache-app2 funny:v1 /bin/sh

ls

exit

step28. 從原本的映象檔運行一個容器,並觀察目錄底下有沒有funny.txt

docker run -it --name apache-app3 apache-server:0.0.1 /bin/sh

ls

exit

step29. 停止所有的容器

docker stop $(docker ps -aq)

step30. 移除所有的容器

docker rm $(docker ps -aq)

step31. 觀察容器

docker ps –a

\* 多容器應用

使用docker架設lamp環境

學習目標:

1. 多容器應用

2. 容器互連

3. volumn

操作步驟

step1. 轉移工作目錄到demo2/sever底下

cd demo2/server

step2. 觀察底下的Dockerfile

step3. 運行mysql伺服器容器

docker run --name mysql -e MYSQL\_ROOT\_PASSWORD=root -d mysql:5.5

step4. 製作映像檔

docker build -t apache-server .

step5. 運行有apache + php環境的容器，並連結mysql容器

docker run -itd -p 80:80 --link mysql:db -v "$PWD"/www/:/var/www/html --name apache apache-server

step6. 打開瀏覽器輸入

localhost/database.php

step7. 使用終端機進入容器

docker exec -it apache /bin/sh

step8. 觀察/etc/hosts檔案並使用ping指令

cat /etc/hosts

ping db

exit

step8. 停止所有容器

docker stop $(docker ps -aq)

step9. 移除所有容器

docker rm $(docker ps -aq)

\* docker-compose

docker\_compose\_demo 實作lamp環境

學習目標:

1. 了解docker\_compose.yml檔結構

2. 基礎docker-compose指令

step1. 轉移工作目錄到docker\_compose\_demo底下

cd docker\_compose\_demo

step2. 執行docker-compose指令，自動部署容器

docker-compose up -d

step3. 到瀏覽器上方搜尋列輸入(正確的話會顯示Successfully connect to the mysql)

localhost/index.php

step4. 嘗試在 docker\_compose\_demo/www 底下新增新的 html或php檔案

step5. 在瀏覽器上方搜尋列輸入

localhost/<yourfile>.html

localhost/<yourfile>.php

step6. 執行docker-compose ps指令觀察

docker-compose ps

docker ps

step7. 執行docker images觀察有哪些映象檔

docker images

step8. 執行docker-compose down指令終止服務

docker-compose down

step9. 輸入docker ps -a觀察一下

docker ps -a

step10. 輸入docker images觀察剛剛部署的應用所使用的映象檔是否還在

docker images

學習目標:

1. 了解如何使用laradock部署laravel所需環境

操作步驟

step1. 使用git將laradock專案複製回本地端

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

step2. 轉移工作目錄到laradock底下

cd laradock

step3. 複製env-example並更改名稱為.env

cp env-example .env

step4. 修改.env檔

# 在最頂端加入

DB\_HOST=mysql

# 搜尋 MYSQL\_VERSION，設定您的 DB 版本

# MYSQL\_VERSION=latest

# 改為

MYSQL\_VERSION=5.7

step5. 使用docker-compose指令部署laravel所需環境

docker-compose up -d apache2 mysql

step6. 進到容器內部

docker-compose exec --user=laradock workspace bash

step7. 使用composer下載laravel專案

composer create-project --prefer-dist laravel/laravel project "5.4.\*"

step8. 會發現本地端多了一個project的資料夾,裡面放的就是laravel專案

step9. 打開瀏覽器輸入

localhost/project/public/