**CFI101\_Docker**

把這邊當作是白板

~~上課時間:~~

~~早上下午晚上 09:00-11:00 / 12:00-19:00 (9hr)~~

~~下午晚上 12:00-18:00 (6hr)~~

~~早上下午 9:00-11:00 / 12:00-16:00 (6hr)~~

~~下課後就回家去複習~~

廷諺 Alvis

# 講義電子檔 - 下載

<https://drive.google.com/file/d/1zh5W23SqViUQWWImAz-KvW_5FQIu9U00/view?usp=sharing>

VMware Workstation:

<https://www.vmware.com/go/getworkstation-win>

# 知道如何有效學習與紀錄更重要!!!

**學習量化!!!**

## 範例: [AWS CodePipeline 測試實作流程(初探)](https://docs.google.com/document/d/1ilIlDiocO1Hj1ltLeb07lv5KrWd2keldFd4euqPOSck/edit?usp=sharing)

## 文件撰寫的架構如下

# 000\_XXXX技術初探研究 或 XXXX技術 實作

# 綱要/目標

這份文件主要是在做什麼?

# 素材

所需的書籍 軟體 套件 工具 等等的

# 預計流程

1.XXXX

2.XXXX

3.XXX

4.XXXXXX

# 實作流程

圖文並茂+操作指令+程式碼的實作流程

# 故障排除

有遇到問題的話，立即截圖放在這邊，處理流程也寫在這裡

# 參考資料

參考的所有文件/網站/書籍/論文

1.上課前的環境整備(包含虛擬機的安裝)

2.Linux基礎設置與安裝Docker

3.Docker基礎操作與指令

4.Docker基礎項目的實作

CentOS 7 (2009) 下載點 (交大):

<http://centos.cs.nctu.edu.tw/7.9.2009/isos/x86_64/CentOS-7-x86_64-DVD-2009.iso>

<http://ftp.stu.edu.tw/Linux/CentOS/7.9.2009/isos/x86_64/CentOS-7-x86_64-DVD-2009.iso>

<http://free.nchc.org.tw/centos/7.9.2009/isos/x86_64/CentOS-7-x86_64-DVD-2009.iso>

<http://mirror01.idc.hinet.net/centos/7.9.2009/isos/x86_64/CentOS-7-x86_64-DVD-2009.iso>

# 上正課之前的環境整備

VMware安裝 → CentOS 7的虛擬機安裝 → Docker的安裝 → 操作Docker

**上課的鐵則：「非必要時，嚴禁使用Root帳戶」**

**ROOT非常危險!!!!!!!!!!**

**~~神(ROOT)~~ → 神之子(SUDOER)**

**查詢剛剛新增的網卡之IP**

| **ifconfig** |
| --- |

**我們會使用192.168.XXX.XXX的IP**

**打開命令提示字元**

**ping 你的VM ping 192.168.XXX.XXX**

**ssh你的VM ssh** [**admin@192.168.XXX.XXX**](mailto:admin@192.168.XXX.XXX)

# Mobaxterm終端機下載

<https://mobaxterm.mobatek.net/download.html>

# 設定CentOS 7 Linux的環境配置

WHEEL的群組

## sudo免密碼

| sudo visudo |
| --- |

打開sudoer的設定檔

| :set nu |
| --- |

顯示行號

| :107 |
| --- |

來到107行

註解107行

| #%wheel ALL=(ALL) ALL |
| --- |

解註解

| %wheel ALL=(ALL) NOPASSWD: ALL |
| --- |

## 設定LINUX IPV4網路轉送

打開設定檔

| sudo vi /etc/sysctl.conf |
| --- |

打入設定值

| net.ipv4.ip\_forward=1 |
| --- |

重啟網路服務

| sudo systemctl restart network |
| --- |

檢查是否設定正確

| sudo sysctl net.ipv4.ip\_forward |
| --- |

## 更改時區

查看日期與時間

| date |
| --- |

列出目前時區設定

| ls -l /etc/localtime |
| --- |

查閱有哪些時區名稱

| sudo timedatectl list-timezones |
| --- |

設定修改指定的時區

| sudo timedatectl set-timezone Asia/Taipei |
| --- |

## 網路時間同步

檢查是否透過NTP去進行時間同步

sudo systemctl status chronyd

**重新啟動時間同步程式**

sudo systemctl restart chronyd

# 安裝Docker(在Amazon Linux 2作業系統上)

[適用於Amazon ECS 的Docker 基本概念- Amazon Elastic Container Service](https://docs.aws.amazon.com/zh_tw/AmazonECS/latest/developerguide/docker-basics.html)

| **sudo yum update -y** |
| --- |

**更新已安裝的套裝服務，並在執行個體上封裝快取**

| **sudo amazon-linux-extras install docker -y** |
| --- |

**安裝最新的 Docker Engine 套件(Amazon Linux 2)**

| **sudo service docker start** |
| --- |

**啟動 Docker 服務**

| **sudo systemctl enable docker** |
| --- |

**開機自動啟動 Docker 服務**

| **sudo systemctl status docker** |
| --- |

**確認Docker安裝及啟用的狀態**

| **sudo usermod -a -G docker ec2-user** |
| --- |

**將 ec2-user 新增至 docker 群組，讓您可以在不使用 sudo 的情況下執行 Docker 命令。**

# 安裝Docker(在CENTOS作業系統上)

<https://docs.docker.com/engine/install/centos/>

| sudo yum remove docker \  docker-client \  docker-client-latest \  docker-common \  docker-latest \  docker-latest-logrotate \  docker-logrotate \  docker-engine |
| --- |

▲移除舊版本Docker

| sudo yum install -y yum-utils |
| --- |

▲安裝yum管理工具套件

| sudo yum-config-manager \  --add-repo \  <https://download.docker.com/linux/centos/docker-ce.repo> |
| --- |

▲透過yum管理工具導入docker的repo來源

| sudo yum install docker-ce docker-ce-cli containerd.io -y |
| --- |

▲安裝docker主程式及所需套件

| sudo systemctl start docker |
| --- |

▲啟動docker

| sudo systemctl enable docker |
| --- |

▲自動常駐/啟動docker

| sudo systemctl status docker |
| --- |

▲檢查docker是否啟用並常駐

| sudo usermod -aG docker $USER |
| --- |

▲將當前使用者加入docker群組

| exit |
| --- |

▲登出目前使用者

最後，再次登入即可使用docker

## 可以確認Docker運作是否正常的指令

查看docker版本

| docker version |
| --- |

執行第一個容器作為測試

| docker run hello-world |
| --- |

**雲端原生**

**Cloud Native**

[**https://youtu.be/nW7RkIYDhY8**](https://youtu.be/nW7RkIYDhY8)

**阿里巴巴的無影**

**共享經濟的概念**

**黑土**

**大家都共用資源**

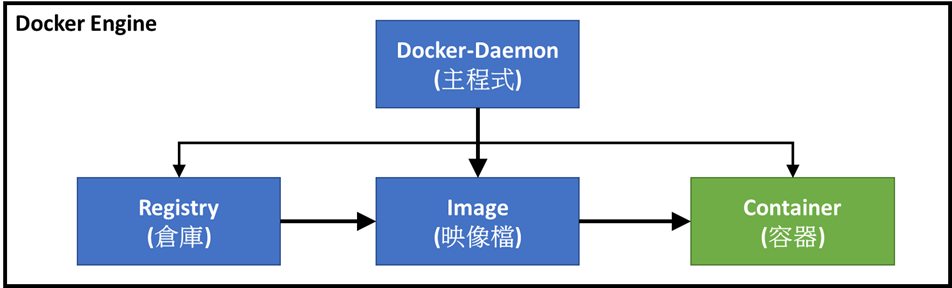
**以租代買**

**以訂閱取代一次性購買**

**基於DevOps的原則，為了讓開發及維運的協作環境一致，而順利達到軟體的開發/交付/佈署，簡言之就是解決環境不一致的問題，或是快速建構開發/測試環境。**

[AWS DevOps 簡介](https://docs.google.com/document/d/1_vEvmMNWYsDU4uRh5qbacAXPJKk1NSfi--UzPnQxHUI/edit?usp=sharing)

[Git基礎實戰教學](https://docs.google.com/document/d/1g9EyM_q3HULHRnJnzBcWZ_-9B9TBqqsXcLPIm0UOcNY/edit?usp=sharing)

****

| docker pull centos |
| --- |

| docker pull ubuntu |
| --- |

| docker pull mysql |
| --- |

| docker pull alpine |
| --- |

## docker run 的部分

要快速啟用一個容器，最方便也最常用的指令

一行指令，就可以依照你的需求，秒開一個容器!!!!!

docker run \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ centos:latest XXX命令

docker run hello-world

基本上是你使用docker run 必備的3個參數

-i ：則讓容器的標準輸入保持打開

-t：讓Docker分配一個虛擬終端（pseudo-tty）並綁定到容器的標準輸入上

-d：背景執行

**容器本身就是一個程序**

1 PID

2 PID

3 PID

-e：設定環境變數(AAA=BBB)

用中文舉例: -e 班級代號=cfI101

通常會用在使用者還不確定的數值，但一定得設置，就可以加入環境變數

例如: 你今天要建一個資料庫，但是一定得設置Root密碼，資料庫才可以建立成功，但是你們覺得全世界的人資料庫ROOT密碼會一樣嗎??????? 一定不會

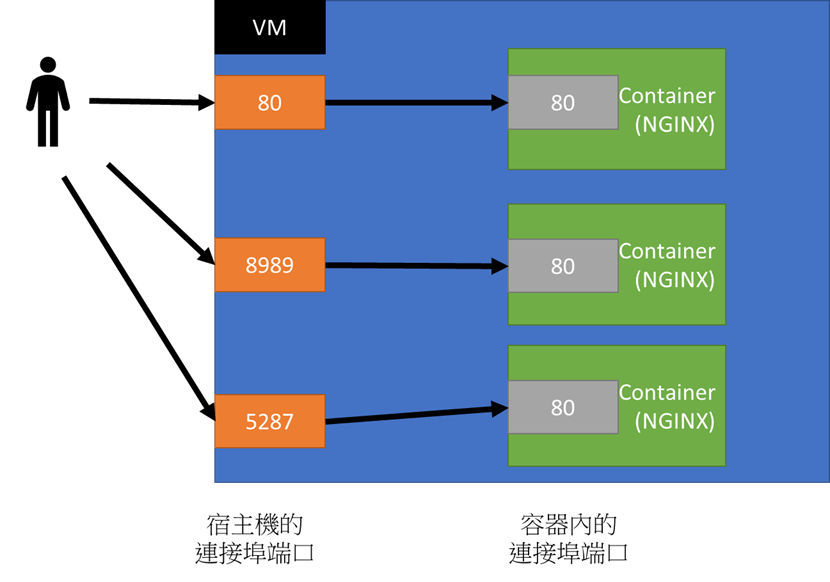
所以可以透過 環境變數 把你設定的密碼帶進去到容器裡面

-p (port)

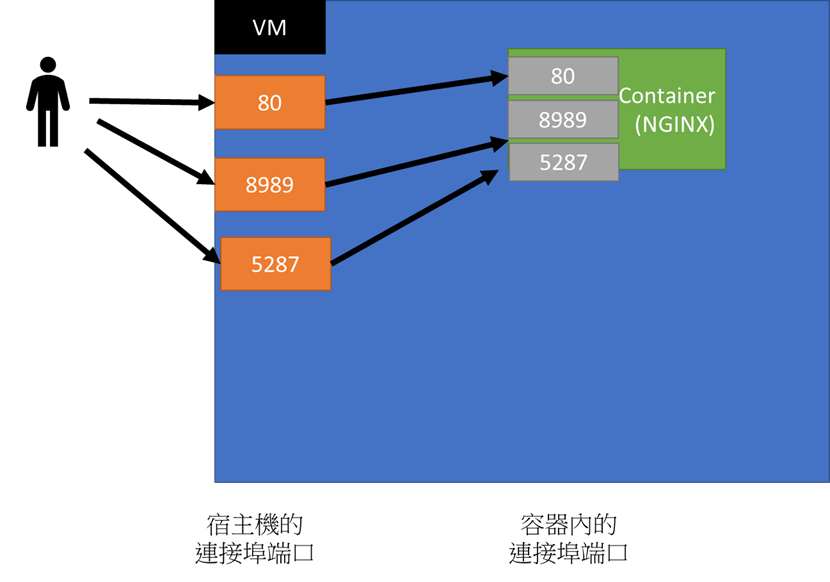
-p 80:80

-p 8989:80

-p 5287:80



-p 80:80 -p 8989:8989 -p 5287:5287



-v：資料對應(host folder:container folder)

(Volume)

**容器使用的概念，就像是免洗筷跟衛生紙一樣**

**不要的話就可以刪除**

**需要的時後再拿(建)一個新的**

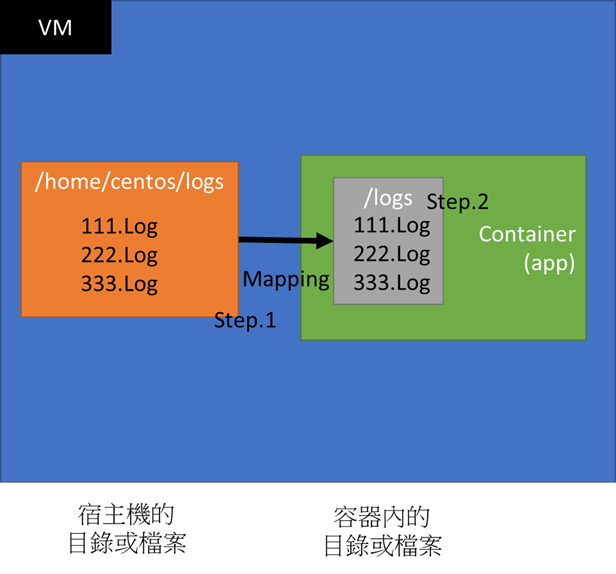
容器刪除的時候，裡面的所有資料也都會跟著被刪除!!!!!!

1. 資料映射(帶入)
2. 資料同步
3. 資料保留

-v $PWD/logs:/logs -v $PWD/logs2:/logs2

絕對路徑: /home/centos/logs

相對路徑: ~/logs $PWD/logs ./logs



容器與宿主機之間的資料同步與資料保留

--name 幫容器取一個名字

亦可當作內部使用的域名(Domain)

日後若有在內部互相連接的話，可以把名字取成像網址那樣

cfc102.studnet99.com

1.每做一個實例/LAB都會建立一個新的專案資料夾，所以每次操作要記得 確認自己當前的目錄在哪? 目錄下是否有建立指定的資料夾???

2.每次做完一個實作的lab完畢後要刪除

3.是為了確保不要和下一個lab衝突

# 實例1-快速建立jupyter開發環境

建立並切換目錄

| mkdir -p jupyter/app && cd jupyter |
| --- |

建立容器

| docker run \  --name jupyter \  -d \  -p 8888:8888 \  -p 5000:5000 \  -v $(pwd)/app:/home/jovyan/work \  jupyter/base-notebook \  start-notebook.sh --NotebookApp.token='' |
| --- |

開啟瀏覽器，並輸入本地IP(虛擬機IP)及port號

192.168.XXX.XXX:8888

輸入並執行程式碼

| !pip install Flask !pip install requests |
| --- |

| from flask import Flask, request, abort app = Flask(\_\_name\_\_) @app.route('/',methods=['GET']) def hello\_world():  return '網頁測試成功～～～～～' if \_\_name\_\_ == "\_\_main\_\_":  app.run(host='0.0.0.0',port=5000) |
| --- |

執行程式碼，開啟瀏覽器，並輸入本地IP及port號

192.168.XXX.XXX:5000

# 實例2-將jupyter與MYSQL串接(--Network)

建立目錄

| mkdir -p jupyter\_db/sqldb && cd jupyter\_db && mkdir app |
| --- |

創建自定義網路

| docker network create test-network |
| --- |

檢查創建的網路

| docker network ls |
| --- |

建立MYSQL容器

| docker run \  --network test-network \  --name mysql \  -e MYSQL\_ROOT\_PASSWORD=123456 \  -v $(pwd)/sqldb:/var/lib/mysql \  -d \  -p 3306:3306 \  mysql:latest |
| --- |

建立jupyter容器

| docker run \  --network test-network \  --name jupyter \  -d \  -p 8888:8888 \  -p 5000:5000 \  -v $(pwd)/app:/home/jovyan/work \  jupyter/base-notebook \  start-notebook.sh --NotebookApp.token='' |
| --- |

輸入並執行程式碼

| !pip install pymysql import pymysql # 開啟資料庫連線 db = pymysql.connect(user="root", password="123456", host="mysql") # 使用 cursor() 方法建立一個遊標物件 cursor cursor = db.cursor() # 使用 execute() 方法執行 SQL 查詢 cursor.execute("SELECT VERSION()") # 使用 fetchone() 方法獲取單條資料. data = cursor.fetchone() print ("Database version : %s " % data) # 關閉資料庫連線 db.close() |
| --- |

# 實例3-快速建立nginx Web Server

建立目錄

| mkdir -p nginx/web && cd nginx |
| --- |

建立容器

| docker run \  -itd \  -v $(pwd)/web:/usr/share/nginx/html \  -p 80:80 \  -p 443:443 \  nginx |
| --- |

80 http

443 https (預開，但不會用到) TLS/SSL

下載網頁模版

<https://www.free-css.com/free-css-templates>

將下載好的網頁，放置nignx的web目錄

開啟瀏覽器，並輸入本地IP(虛擬機IP)

192.168.XXX.XXX

安裝git後，下載教材

| sudo yum install git -y  git clone <https://github.com/aeon33system/docker_Advanced_ngrok.git>  cd docker\_Advanced\_ngrok |
| --- |

~~設定vim環境~~

| ~~sudo vim /etc/vimrc~~ |
| --- |

~~加在最下面~~

| ~~set smartindent set tabstop=2 set shiftwidth=2 set expandtab set number~~ |
| --- |

# ***安裝docker-compose***

### 安裝的官方文件:

### <https://docs.docker.com/compose/install/>

### Linux安裝方式:

### 依序輸入下方指令

| sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose |
| --- |

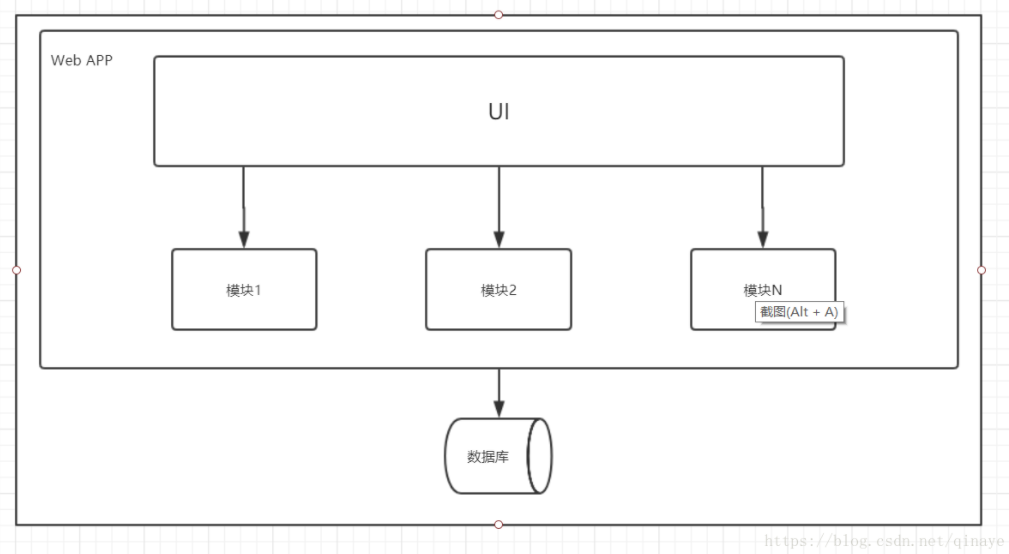
| sudo chmod +x /usr/local/bin/docker-compose |
| --- |

| sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose |
| --- |

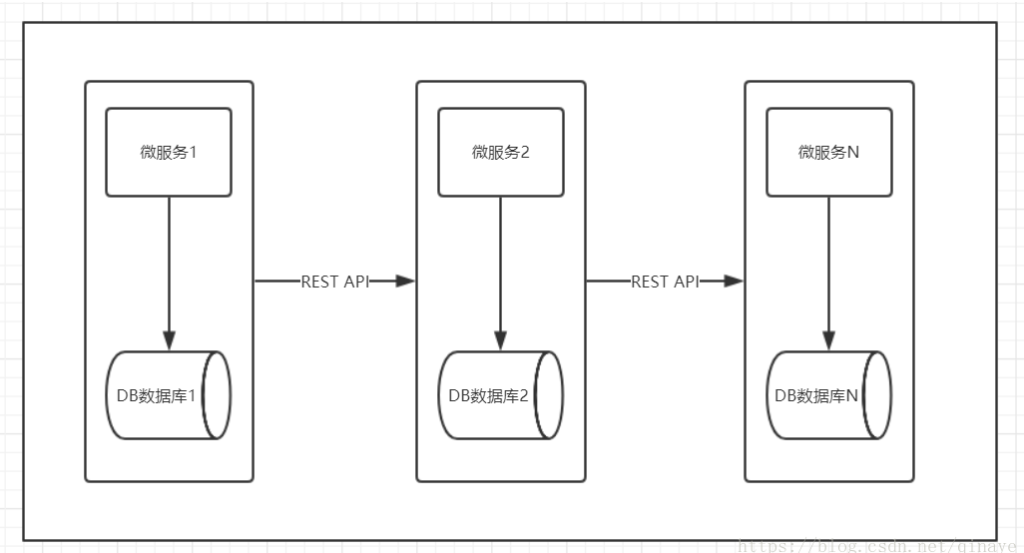
| docker-compose --version |
| --- |

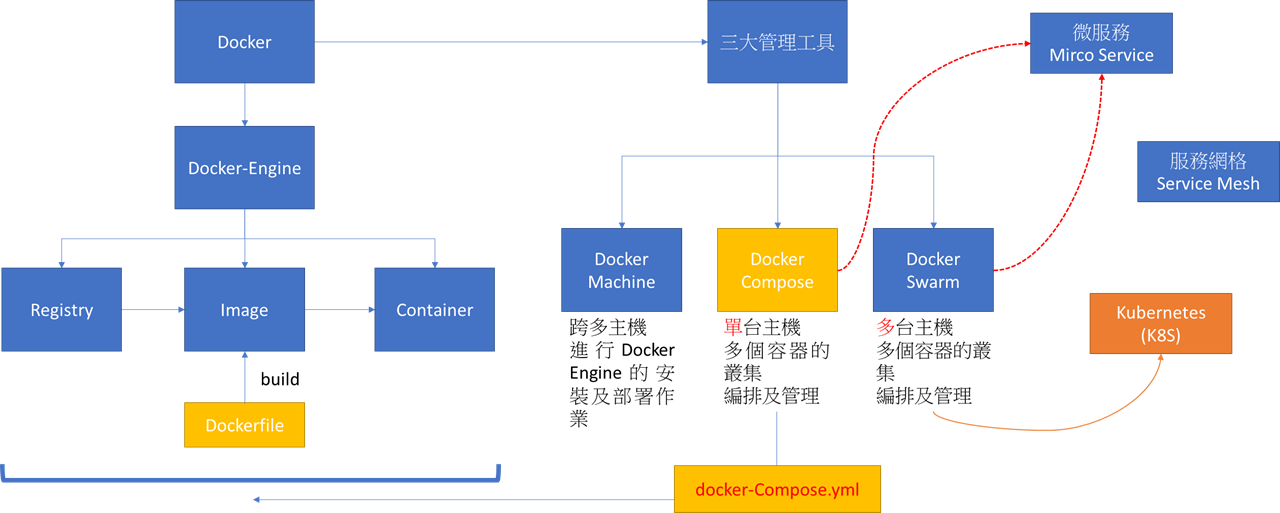
微服務的樣貌與比較

單體式軟體架構



微服務軟體架構





Dockerfile

EXPOSE這個語句，是宣告容器對外的埠號

什麼意思呢?

大家好，這是一片Nginx的光碟，如果你今天想要用這片光碟的話，請你務必要開啟80 Port才能給外面的人存取，我這邊只是提醒你而已，我不會幫你開唷~~~~~你要自己下 -p 參數去開啟

# Lab1-初步客製化jupyter開發環境(1) 失敗的案例(不用做)

開啟並編輯dockerfile

| vim $(pwd)/lab1/1\_1/dockerfile |
| --- |

依照指定的dockerfile建立Image

| docker build -t aiclass/custom-jupyter:v1 ./lab1/1\_1/ |
| --- |

使用客製化的Image，建立Container

| docker run --name lab1-container -d -p 8888:8888 -v $(pwd)/lab1:/home/jovyan/work aiclass/custom-jupyter:v1 start-notebook.sh --NotebookApp.token='' |
| --- |

開啟瀏覽器，輸入127.0.0.1:8888執行Jupyter

開啟lab1.ipynb，執行程式

**※沒有安裝此套件，所以無法導入**

# Lab1-初步客製化jupyter開發環境(2) 成功的案例

~~先移除剛剛建立的第一個容器~~

| ~~docker rm -f XXXXXXXXXX~~ |
| --- |

開啟並編輯dockerfile

| vim $(pwd)/lab1/1\_2/dockerfile |
| --- |

依照指定的dockerfile建立Image

| docker build -t aiclass/custom-jupyter:v2 ./lab1/1\_2/ |
| --- |

使用客製化的Image，建立Container

| docker run --name lab1-container -d -p 8888:8888 -v $(pwd)/lab1/:/home/jovyan/work aiclass/custom-jupyter:v2 start-notebook.sh --NotebookApp.token='' |
| --- |

開啟瀏覽器，輸入127.0.0.1:8888執行Jupyter

開啟lab1.ipynb，執行程式

※因為在build Image的時候

已經安裝好所需的套件

所以可以直接導入

# **Lab2-打包Nginx的網頁程式碼**

開啟並編輯dockerfile

| vim $(pwd)/lab2/1\_1/dockerfile |
| --- |

依照指定的dockerfile建立Image

| docker build -t aiclass/custom-nginx:v1 ./lab2/1\_1/ |
| --- |

使用客製化的Image，建立Container

| docker run -itd -p 80:80 --name nginx aiclass/custom-nginx:v1 |
| --- |

開啟瀏覽器，輸入127.0.0.1

即可直接看到打包進去的網頁

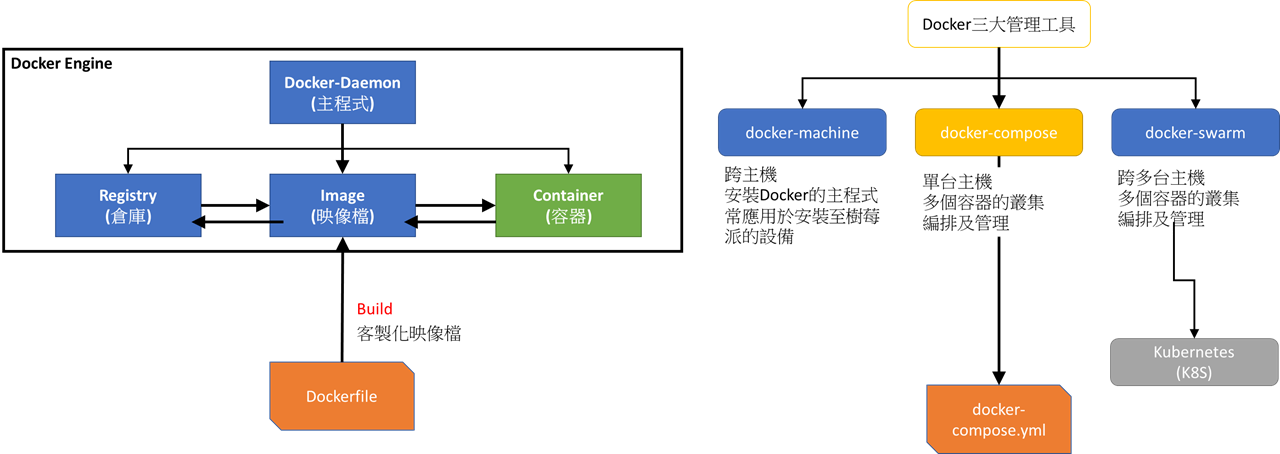
※直接將網頁打包進去Image

不必特地將容器與本機掛載資料

1.我要用python:3.7的映像檔

2.我需要安裝XXX套件 XXX工具 XXXX素材

3.複製本地的資料夾內的程式碼到XXXXX路徑去.....



# Lab3-docker-compose打包程式碼並執行程式

開啟並編輯docker-compose.yml

| vim $(pwd)/lab3/docker-compose.yml |
| --- |

開啟並編輯dockerfile

| vim $(pwd)/lab3/dockerfile |
| --- |

啟動docker-compose

| docker-compose -f $(pwd)/lab3/docker-compose.yml up -d |
| --- |

開啟瀏覽器，輸入127.0.0.1

即可直接看到打包進去程式碼顯示的結果

# Lab4-docker-compose執行LINE機器人開發及執行環境

開啟並編輯docker-compose.yml

| vim $(pwd)/lab4/docker-compose.yml |
| --- |

開啟並編輯dockerfile

| vim $(pwd)/lab4/dockerfile/dockerfile-jupyter |
| --- |

啟動docker-compose

| docker-compose -f $(pwd)/lab4/docker-compose.yml up -d |
| --- |

| docker-compose up -d |
| --- |

開啟瀏覽器，輸入192.168.XXX.XXX:8888 執行Jupyter

設定LINE BOT憑證資訊

開啟app.py.ipynb，執行程式

※成功後，可以直接使用手機，與line機器人對話

-region ap

拿到Ngrok生成的https網址

| curl $(docker port chatbot\_ngrok 4040)/api/tunnels |
| --- |

# 補充給你們的

[line/line-bot-sdk-python: LINE Messaging API SDK for Python](https://github.com/line/line-bot-sdk-python)

Docker官方參考文件(有各種初階至進階的說明及教學文件)

[Reference documentation](https://docs.docker.com/reference/)