🥰

高效開發好幫手：Docker

為什麼要使用Docker？

安裝環境很麻煩

- 若今天要部署後端，會需要安裝許多環境套件跟工具
  - 安裝流程非常繁瑣
  - 在安全的過程中如果跳出 Error，又要花很久爬 Stackoverflow

遷移主機也很麻煩

- 要重新寫部屬文件
- 不同作業系統，需要不同指令做安裝

學習K8s的根基

- 當Container數量變得很多，就會需要K8s協助做排程

後端工程師的必備技能

- 工作會用到

Docker 重要概念

Docker File

- 類似**食譜**

Docker Image

- 類似**模板、寶貝球**

Docker Container

- 類似**從寶貝球出來的寶可夢**

Docker 可以製造出標準化的Container，在任何環境跑

為什麼Docker可以讓部屬更方便？

- 假設今天有個筆電（Infrastructure）作為伺服器
  - 伺服器上會有作業系統（Operating System）
- 而我們有一個問題：同樣的一個App寫好後，怎麼在**不改動程式**的情況下部屬到不同平台？

> 🚀 解答：Container Engine 會協助把環境**虛擬化**

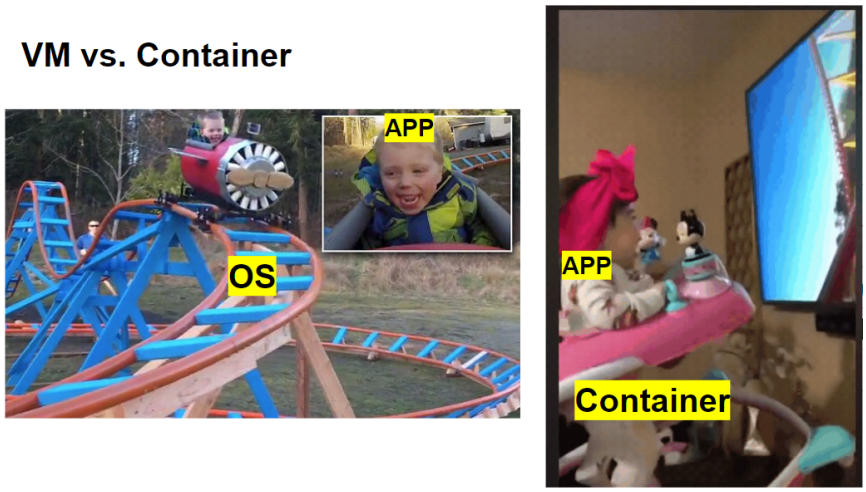
  - 應用程式不需要知道Container Engine底下有什麼

Virtual Machine vs Container

	VM	Container
虛擬化	機器	作業系統
需要灌	OS	docker image
Size	大 (GB)	小 (MB) ✅
啟動速度	慢	快 ✅

- Virtual Machine：把**機器虛擬化**
  - 但仍得要灌一個作業系統（如Ubuntu）
    - 導致運行上比較肥大、占空間
- Container：把**作業系統虛擬化**
  - 多個Container實際上用同一個OS運作

- 比較快，也比較有效率



Container就像右圖的遊戲機台一樣，雖然人不在遊樂園，但能提供你在遊樂園的感覺

可再回去搜尋：**那有沒有VM贏過Docker的地方？**

Docker File / Image

Docker File

- 是一個程式碼短短的檔案
- 執行後就可以build起來
  - Docker File build 起來後，就會獲得一個**映像檔（Image）**
  - **映像檔**跑起來後，就變成**容器（Container）**

Docker Volume

- 卡比獸若回去寶貝球，就會忘記他過去做過的事
- 對比Container：**重新打開後，上次的操作紀錄都會忘記**
  - 因此需要 Docker Volume：類似**外接USB**

- 即使Container被關掉了，還是可以記得儲存資訊

Docker Network

- 讓Docker可以建立自己的內網
  - 使Container間可以互相溝通

實作

- 目標：**部屬Express.JS + MogoDB**
  - 把Repo載下來進行操作
  - 再用Postman / Curl進行測試
- **Docker 社群 QRCode**

1. 安裝Docker Desktop

- 把 Docker.app 安裝包拖進 Application 這個資料夾
  - 接著就能開啟程式
- 點擊 Dashboard 可以打開GUI介面

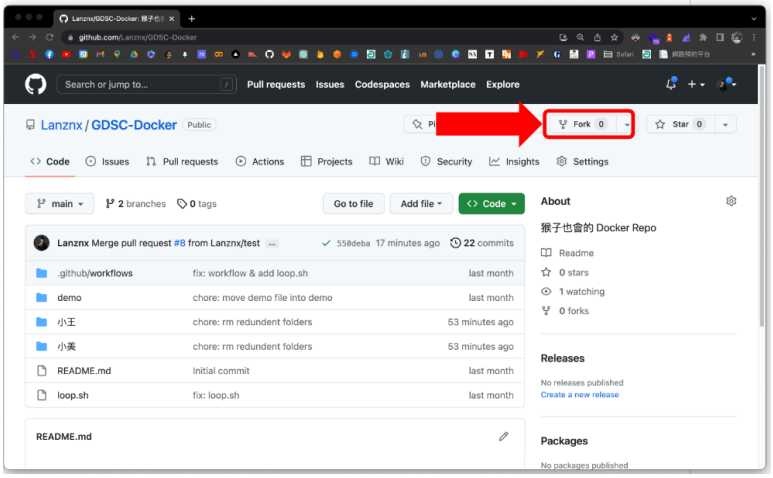
2. Fork Github Repo

Fork 是什麼

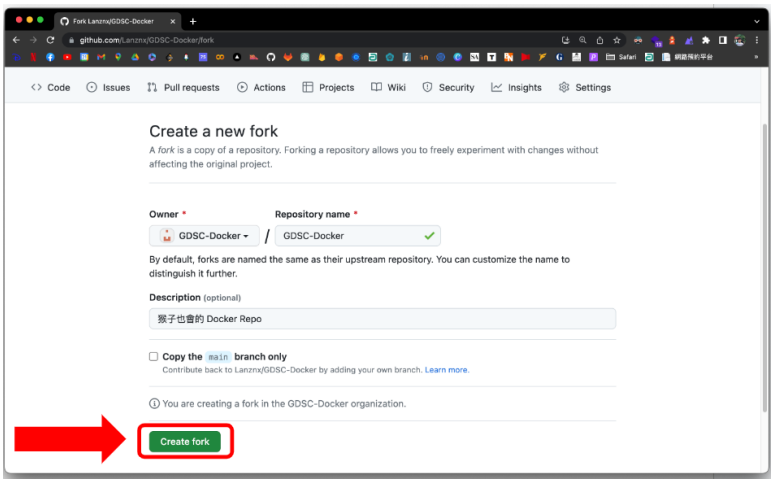
- 把對方的Repo **Fork** 過來，讓自己也有一份一模一樣的 Repo
  - 把別人的 repo **複製一份**的概念
  - 自己在改動程式碼時，只會改動到自己遠端的 Repo（即自己 Fork 過來的那份）
- 在本地端開發：須把遠端的 Repo **Clone** 下來
  - 讓自己本地端有一份一樣的程式碼

步驟

- 點擊GitHub上面的Fork按鈕



- 點擊Create Fork



- 接著點開綠色按鈕的**Code**，選取**HTTPS**
  - 複製網址下來，並在terminal欲新增專案的位置輸入**git clone+網址**
- Clone完後開啟資料夾，結束

專案架構

- 這份專案類似 MVC 架構
  - 但是不包含 V （View，即前端）
- index.js：程式進入點，C（Controller）
- M：Model，為一個資料夾
  - 資料庫存取層
- db 資料夾：資料庫連線設定
- test 資料夾：API測試
  - 最後會用裡面寫好的測試來檢驗環境是否設成功
- dockerfile
  - 點開後可看到不到十行的程式碼

讀懂 Docker File

`FROM node:16-alpine`

- 設定image要跑在什麼樣的環境
  - node.js 16版
- alpine：把image不要的東西都不放進來

`WORKDIR /workspace`

- 切換Docker目錄

`COPY package*.json ./`

- 丟 package.json 進去 container 內部空間
- `package*.json`：本機空間
- `./`：Container內部

`RUN npm install`

- 下載Node modules

`COPY . .`

- 複製專案其他檔案進去

`EXPOSE 3000`

- 對外開放Port 3000

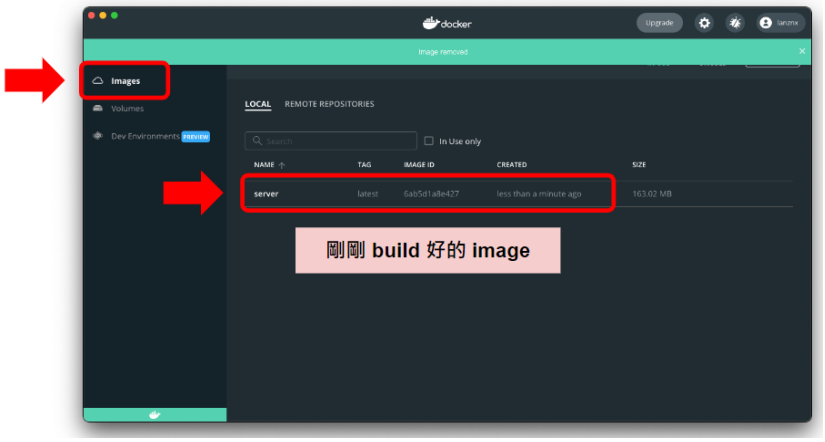
`CMD [ “node”,”index.js”]`

- 把程式進入點放進去
- 將server / container跑起來

3. Build Docker Image

- 輸入 ls 確定當前目錄有Dockerfile
- 輸入 `docker build . -t server` 把當前目錄下的 dockerfile build 起來
  - -t = --tag：幫這個image取名
- build起來後，去Docker Desktop可以看到剛剛build好的image

Build Docker Image



4. Run Docker Image

- **GUI版本**
  - 點Run將Container跑起來
  - 將Container Name命名為Monkey
  - 讓Local Host開在3000，跟Container port保持一致
  - 按右下角的Run → 成功跑起來了
- **指令版本**

```
docker run -p 3000:3000 --name monkey server
```

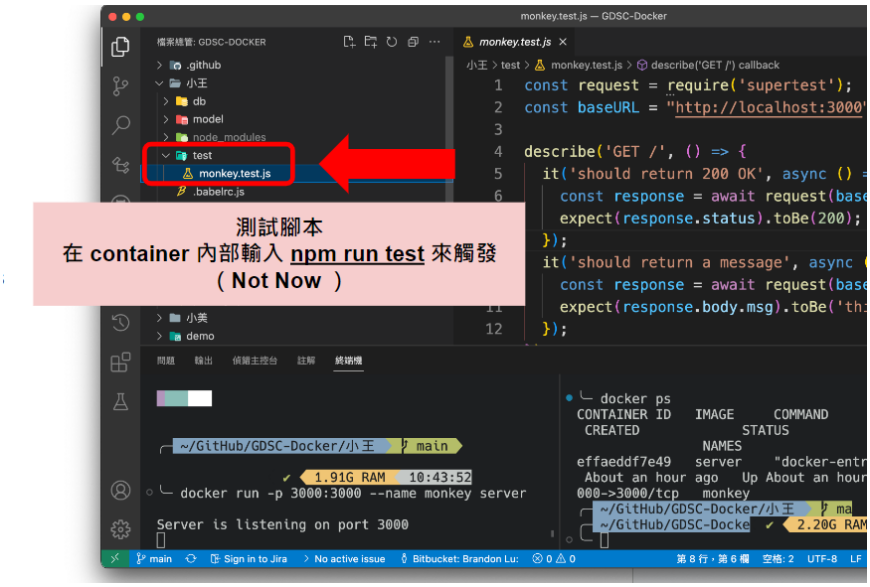
1. 註：把 server 這個 image 給跑起來
2. `-p 3000:3000` 是把機器上的 port 對應到 container 的 port
   a. 進階：思考為什麼要做 port mapping？
3. `--name monkey` 是命名 container 成 monkey

5. 測試環境是否建立成功

目標：負責發 API 去指定的 url

1. **Postman 測試**
    - GET → http://localhost:3000/
        - 回傳 json：”msg”:”this is working” ⇒ 成功
        - 若看到 Error: connect ECONNREFUSED：可能Container還沒建立起來，或沒啟動成功
    - POST → http://localhost:3000/
        - 過很久後卻回傳 internal server error
        - 因為剛剛只有部屬Express Server，還沒建立起資料庫
        - 所以現在要把MongoDB架起來
2. **curl 測試**
    - POST
        - curl -X POST -H "Content-Type: application/json" -d '{"name":"Docker"}' http://localhost:3000/monkey
3. **Supertest 自動化測試**
    - 進入 container 執行 monkey.test.js

- 在container 輸入 npm run test 觸發
    - 看東西有沒有照你的意思成功執行

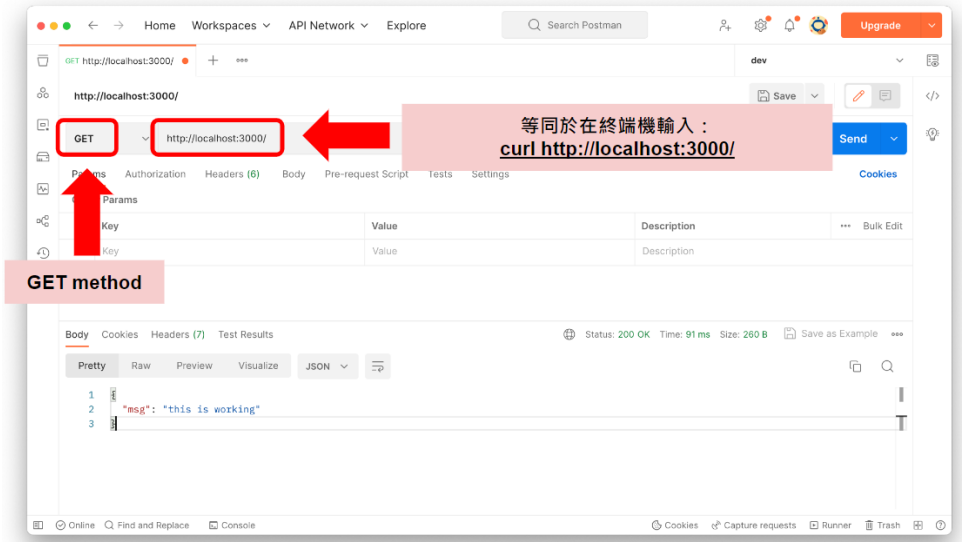


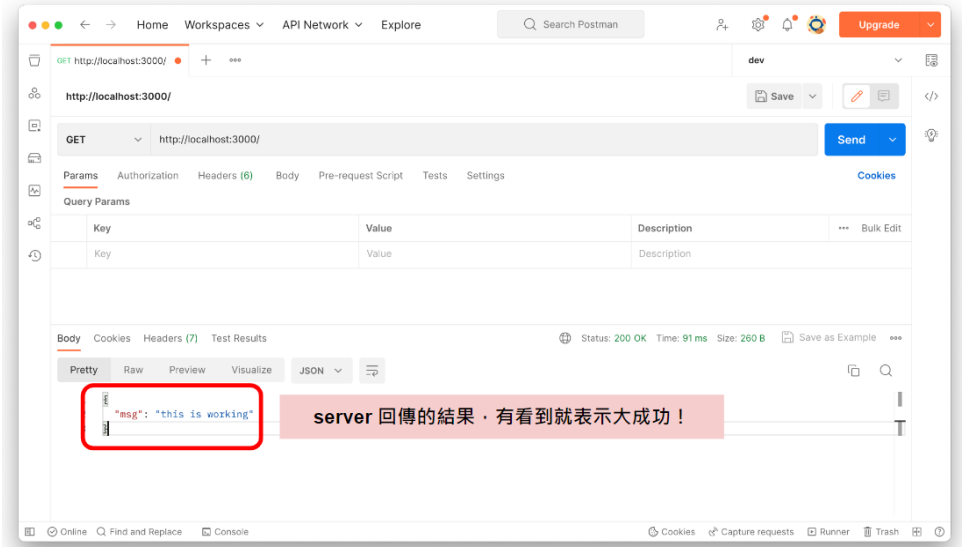
Curl 版本

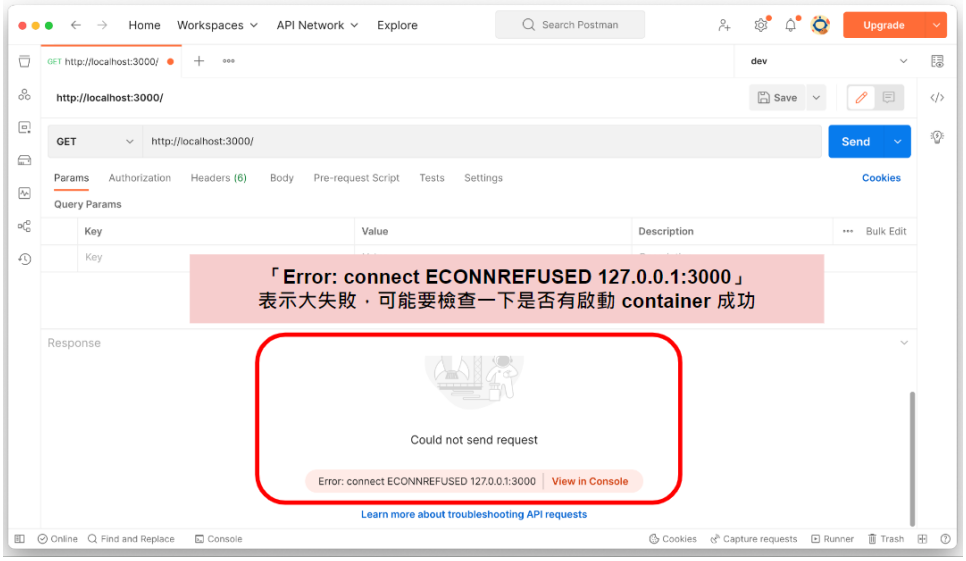
- # test connection
  - `curl http://localhost:3000/`
- # Get monkey with name K8s
  - `curl http://localhost:3000/monkey?name=K8s`
- # Get all monkeys
  - `curl http://localhost:3000/monkey?name=all`
- # Get monkey with name NOMONKEY (expect 404)
  - `curl http://localhost:3000/monkey?name=NOMONKEY`
- # Get monkey without name (expect 400)
  - `curl http://localhost:3000/monkey`
- # Create monkey with name Docker
  - `curl -X POST -H "Content-Type: application/json" -d '{"name":"Docker"}' http://localhost:3000/monkey`
- # Create monkey without name (expect 400)

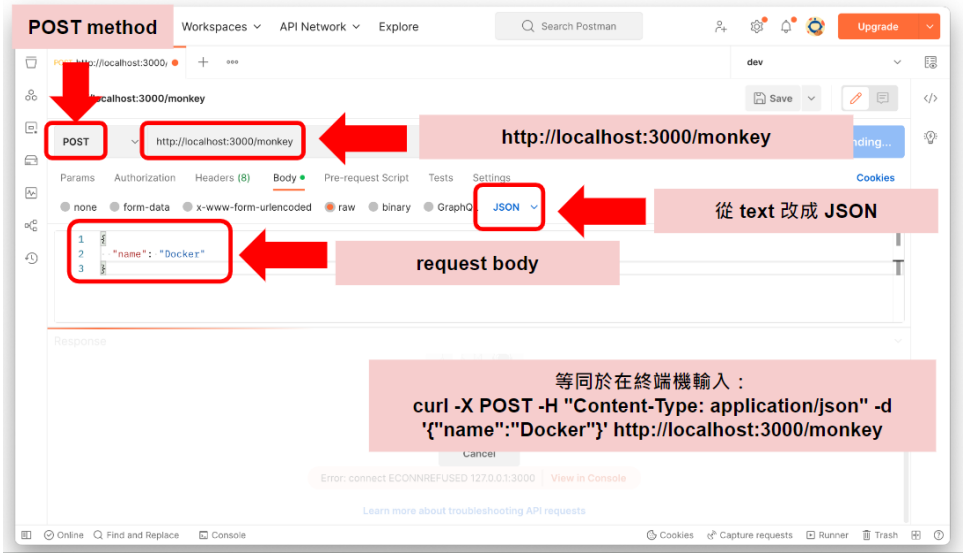
- `curl -X POST -H "Content-Type: application/json" http://localhost:3000/monkey`

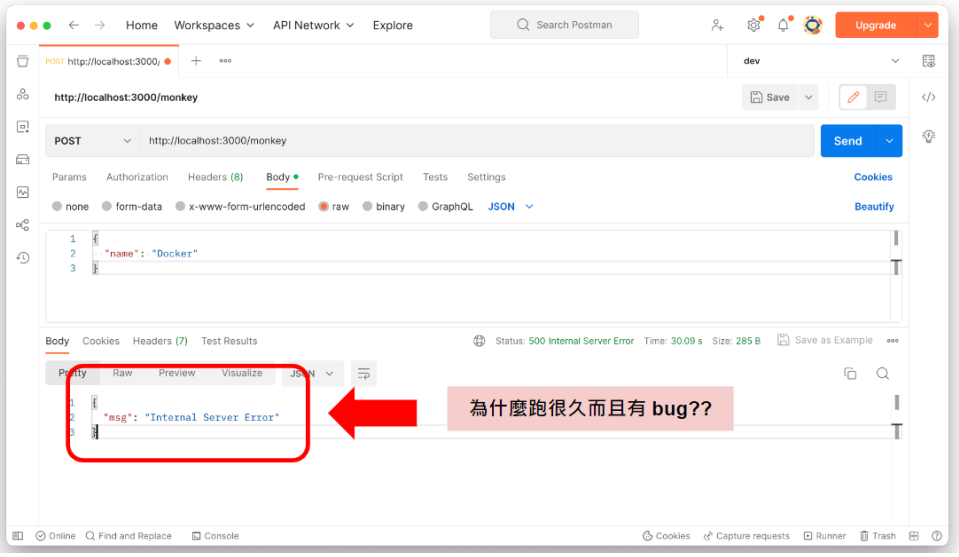
Postman 版本






Home
Workspaces
API Network
Explore
Search Postman
Upgrade
http://localhost:3000/
GET
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
「Error: connect ECONNREFUSED 127.0.0.1:3000」
表示大失敗，可能要檢查一下是否有啟動 container 成功
Response
Could not send request
Error: connect ECONNREFUSED 127.0.0.1:3000
View in Console
Learn more about troubleshooting API requests


POST method
Workspaces
API Network
Explore
Search Postman
Upgrade
POST
http://localhost:3000/monkey
http://localhost:3000/monkey
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
從 text 改成 JSON
1 {
2 "name": "Docker"
3 }
request body
Response
等同於在終端機輸入：
curl -X POST -H "Content-Type: application/json" -d
'{"name":"Docker"}' http://localhost:3000/monkey
Cancel
Error: connect ECONNREFUSED 127.0.0.1:3000
View in Console
Learn more about troubleshooting API requests



利用 Docker Compose 管理容器

- Docker超級方便的秘訣
  - 業界常用方式：把**yaml檔**寫好，再用**compose**指令來完成
  - 而不是慢慢的build好再run起來

試做 Docker Compose：指令版

1. 先把剛剛建的container刪掉
   - 輸入 `docker remove monkey -f`
2. 輸入 `docker compose up`
   - compose 之後，就有兩種不同顏色的訊息
   - 會根據docker-compose.yaml的設定把mongoDB pull下來

Docker Compose 做了甚麼

- 它會讀取Docker-compose.yaml裡面寫了甚麼

- 兩個Container的設定會互相隔離，那要怎麼溝通？
  - Docker是怎麼讓mongoDB跟Express Server做連結？
    - 答案：透過Docker Network連線
- 將chaewon打開，即可看到MongoDB & Express Server小程式
  - 另外有個自動化腳本

測試環境是否建立成功

- 可在Monkey這個container選取右邊選項 (GUI)
  - 選擇進入container（第二個按鈕）
- 按了Run in terminal後，會自動執行docker exec，系統自動幫你跑指令進去container
  - 後面的sh代表會進去裡面用shell做操作
- 可以輸入 ls 看看container長怎樣
- npm run test
  - 觸發自動化API測試
- 有跑到那些API = **舒服(整片都是綠色的)**

Docker Compose

- 寫好Yaml檔，然後Docker compose up ⇒ 就全部都部署好了!
- Compose 會把 dockerfile build 起來，container run 起來
  - 不用再安裝 node , npm …
- DockerHub：類似GitHub的地方，擺放大家的Docker Image
  - 不用自己寫 Docker File
  - 只要Pull ，然後Docker Compose 就好了

AutoGPT

1. 閱讀官方文件
2. 去官網申請OpenAI API Key，並放在.env
3. 下載最新版本

Key takeaways

Why learning docker？

- docker compose up 可取代滿滿的 sudo install nodejs, install npm, update, nginx…
    - 少處理很多麻煩指令
    - 而且CentOS社群較小，較難找到問題解方

今天教的東西

- 認識Docker file/ image / container
- 大致Docker 原理
- Docker vs Virtual Machine
- Docker 實際使用
- Docker Compose 概覽

課後複習

面試容易被問

- Docker 真的比較好嗎？
    - 和 VM 相比的優缺有什麼？
- 自己寫一個Compose設定