

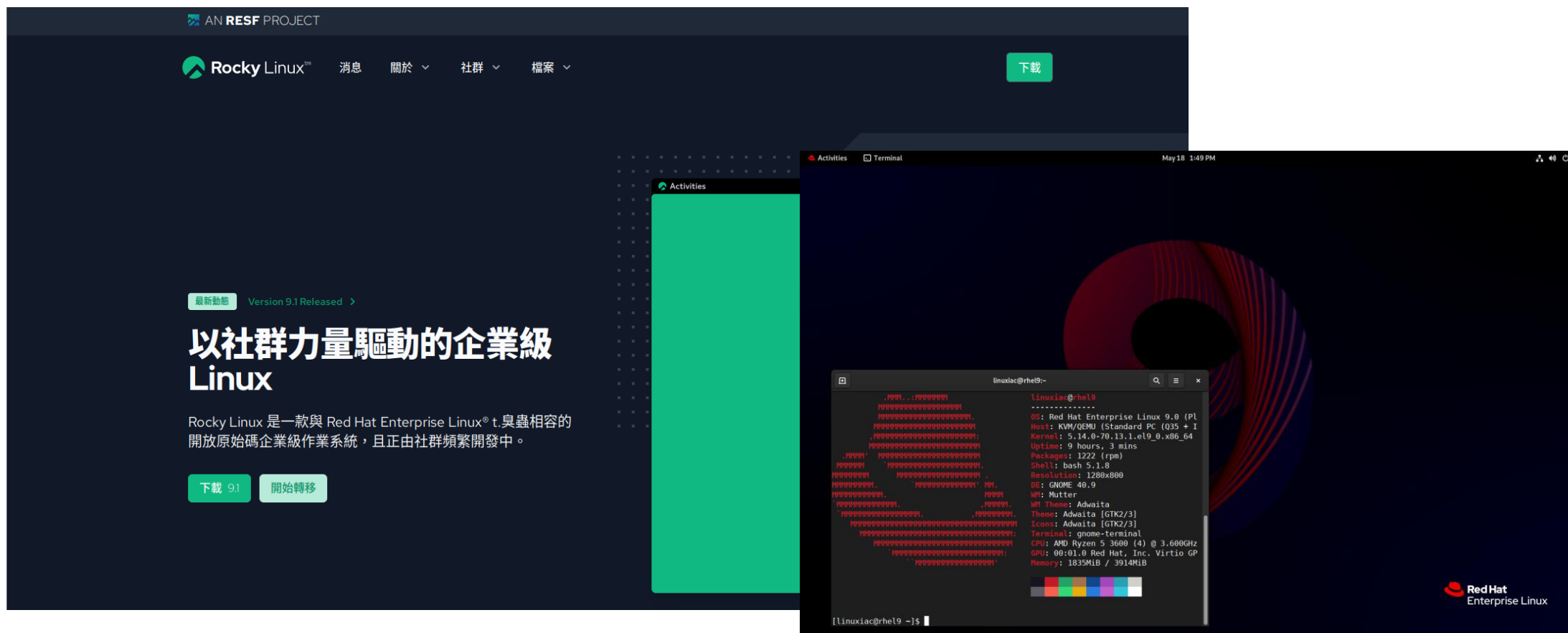
---

# 後端組成架構

- GitHub 倉庫：[https://github.com/neko0xff/2023\\_schoolResearch\\_Server-HW](https://github.com/neko0xff/2023_schoolResearch_Server-HW)
  - 維護者：[\[neko0xff\] https://github.com/neko0xff](https://github.com/neko0xff)
-

# HOST (宿主主機)

- OS: ROCKEY LINUX
  - 提供一個由社群支援且可用於正式營運的企業級作業系統
  - 基於紅帽企業Linux（RHEL）的源碼進行修改



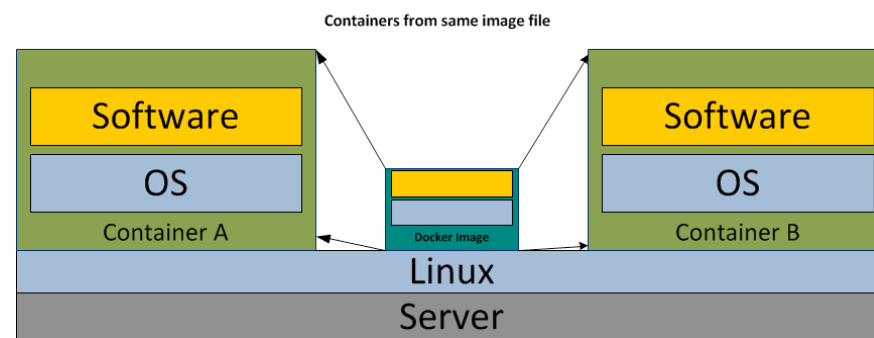
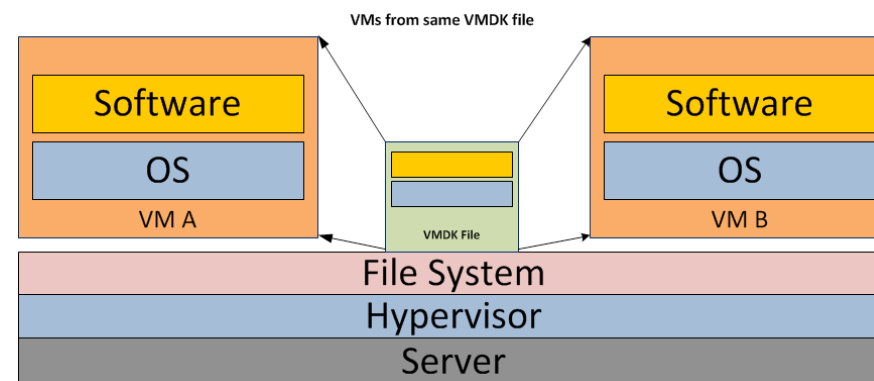
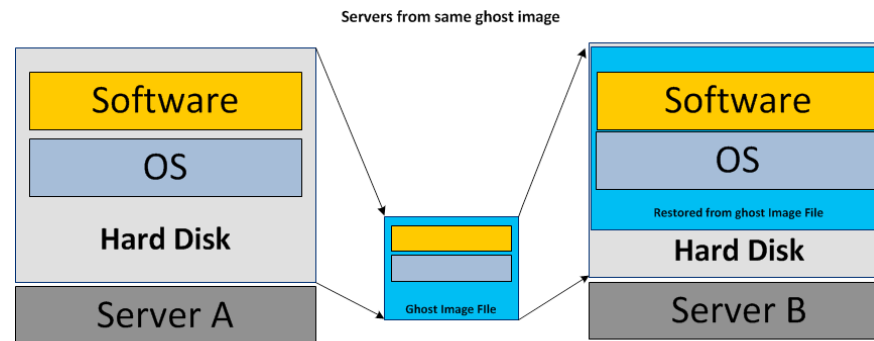
# 何為虛擬化？

- 可能會有的問題

每台電腦的軟體(使用的版本&函式庫)與硬體配置不盡相同，寫好的程式可能剛好跟開發者的電腦上的環境相容或者不相容。

- 運作方式

1. 先在宿主主機上模擬出一個接近實體的環境
2. 讓程式在不同硬體上執行時，都以為自己本身在同一個原始環境中執行



---

# 虛擬機(VM) VS 容器(CONTAINER)

## 虛擬機器（以作業系統為中心）

- 目標：將一個應用程式所需的執行環境打包起來，建立一個獨立環境，方便在不同的硬體中移動
- 在系統層上虛擬化: 在本機作業系統(Host OS)上再裝一個獨立運行於本機的作業系統(Guest OS)，然後讓兩個作業系統彼此不會因環境不同而不相容。



## 容器（以應用程式為中心）

- 目標：提供一個不需自己花時間調整就能開箱即用的環境
- 在作業系統層上虛擬化: 不需額外安裝作業系統 ( Guest OS )，而是透過容器管理工具直接將一個應用程式所需的程式碼和函式庫一同打包成容器，且同時建立資源控管機制直接隔離各個容器並分配宿主主機上的系統資源給容器使用。
- 同時建立容器時所需的系統資源&開機時間可大幅降低，進而改善虛擬機器因為需要裝 Guest OS ,而會有啟動慢、佔較多的系統資源的問題。

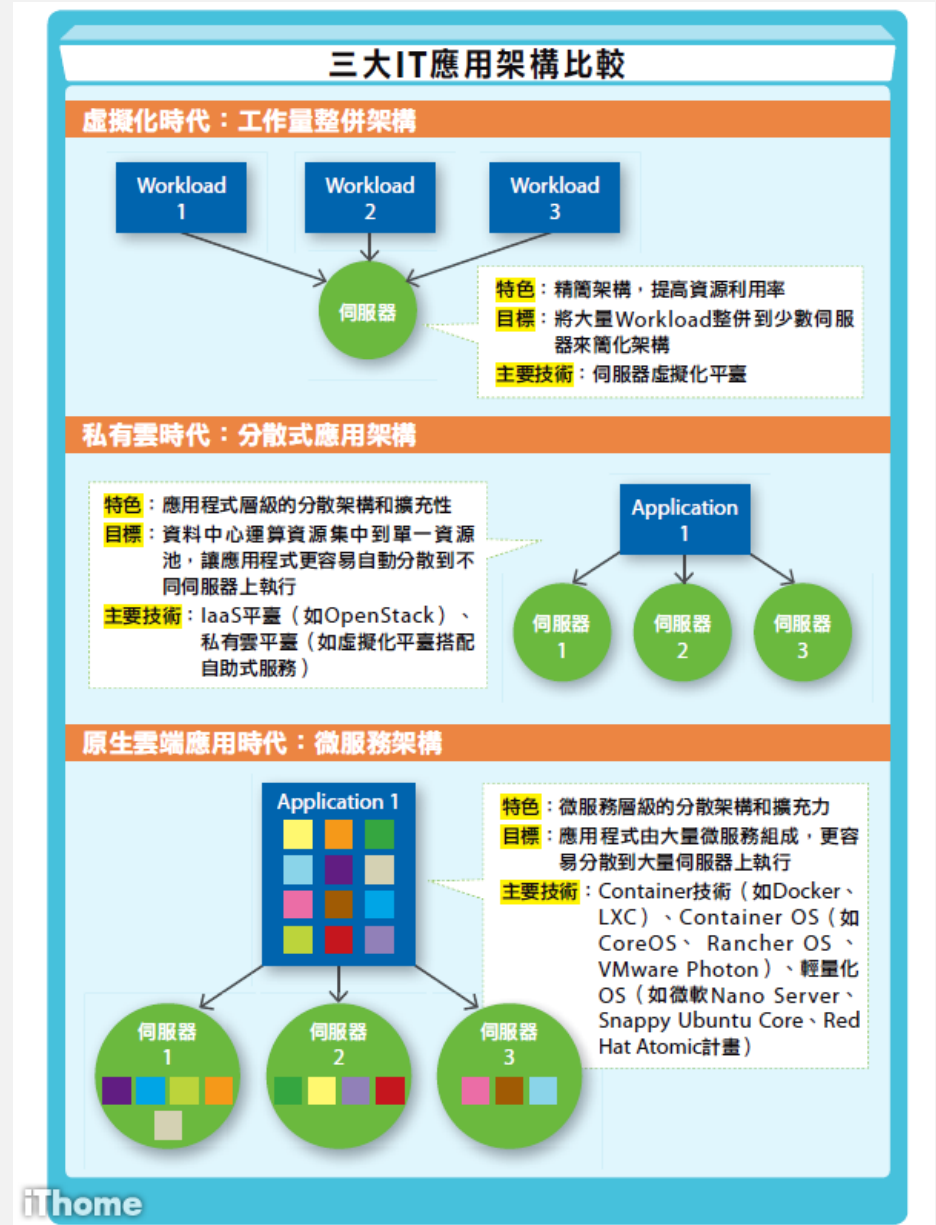


# 主要架構

## 微服務(Microservice)

是一種重新建構應用程式的方法。

將一個應用程式中不同的功能 (ex: 前端和後端部分) 切開隔離，變成彼此獨立運作的容器個體，使得其中一個出問題時整個系統不會受其影響。



# 微服務(MICROSERVICE)的定義

一群協同運作的小型自主服務(Autonomous Service)

- 小巧，專注，自主性
- 組合性&最佳可替換性
- 技術異質性
- 組織調教&彈性
- 容易佈署&擴展

- 來源：Andrew Wu. (2017, April 15). *架構師觀點 - 轉移到微服務架構的經驗分享 (Part 1)*. 安德魯的部落格. <https://columns.chicken-house.net/2017/04/15/microservice8-case-study/>

## 微服務(MICROSERVICE)的特色

- **來源：**施靜樺. (2019, January 27). *Container 概念筆記*. Medium.  
<https://medium.com/@jinghua.shih/container-%E6%A6%82%E5%BF%B5%E7%AD%86%E8%A8%98-b0963ae2d7c6>

- **低耦合 ( loose coupling )**

服務與服務之間關聯低，不互相牽制，而相同功能應該要放在同一個服務之內。

- **高內聚力 ( high cohesion )**

如此達成的效果是想要改變一項功能只需要去一個地方修改，並且不會影響到其他服務。

- **資料管理去中心化 ( decentralized data management )**

每個服務都可以有自己的資料庫和存放點

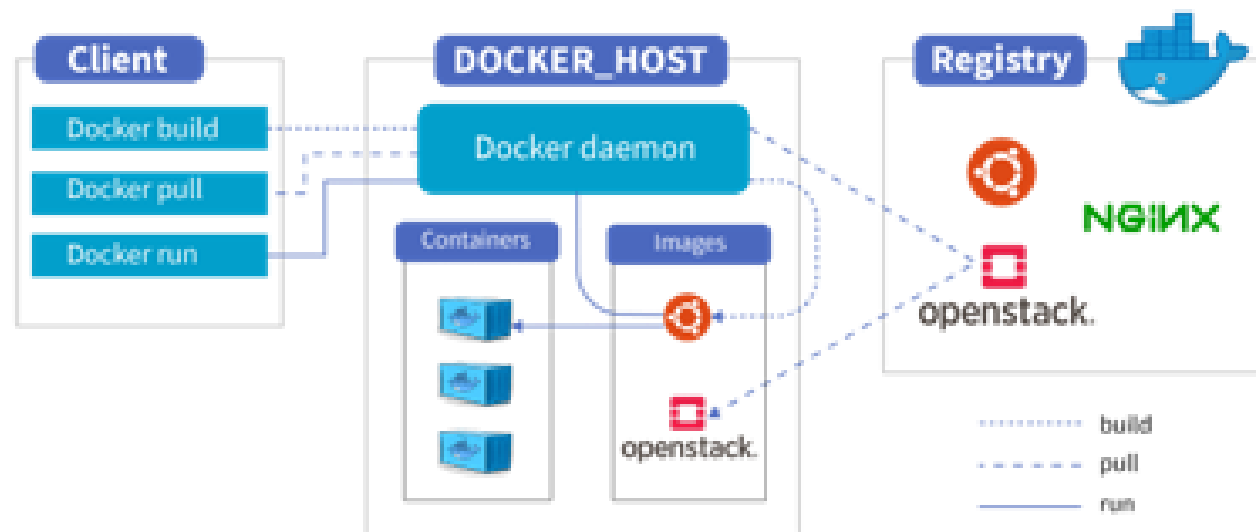
- **使用 API 溝通**

利用像是 HTTP 或者 MQTT 來聯絡每個服務

- **服務可以獨立部署**

團隊可以直接更新現有服務，而不需重新部署整個應用程式&環境

## DOCKER 介紹



- 翻成中文意思就是 “碼頭工人”
- 是以**容器為核心**的資訊技術 (Information Technology, IT) 交付與運行的標準化系統平台與生態體系。



---

# DOCKER介紹

與現實所對應的角色(ex: 以海運為例)

- 容器(Container) == 傳統運輸領域的貨櫃
- 雲端服務提供商 == 承載貨櫃的港口
- 所提供的基礎架構即服務(Infrastructure as a Service, IaaS) == 運送貨櫃的輪船



**來源：** 邱全成 . (n.d.). 雲報專欄：雲端運算下容器(Container)技術和應用. 台灣雲端物聯網產業協會.  
[http://www.twcloud.org.tw/files/file\\_pool/1/0i235396067708866326/5.pdf](http://www.twcloud.org.tw/files/file_pool/1/0i235396067708866326/5.pdf)

---

# DOCKER組成要素

由最重要的三個元素所組成

- **Image(映像檔):**

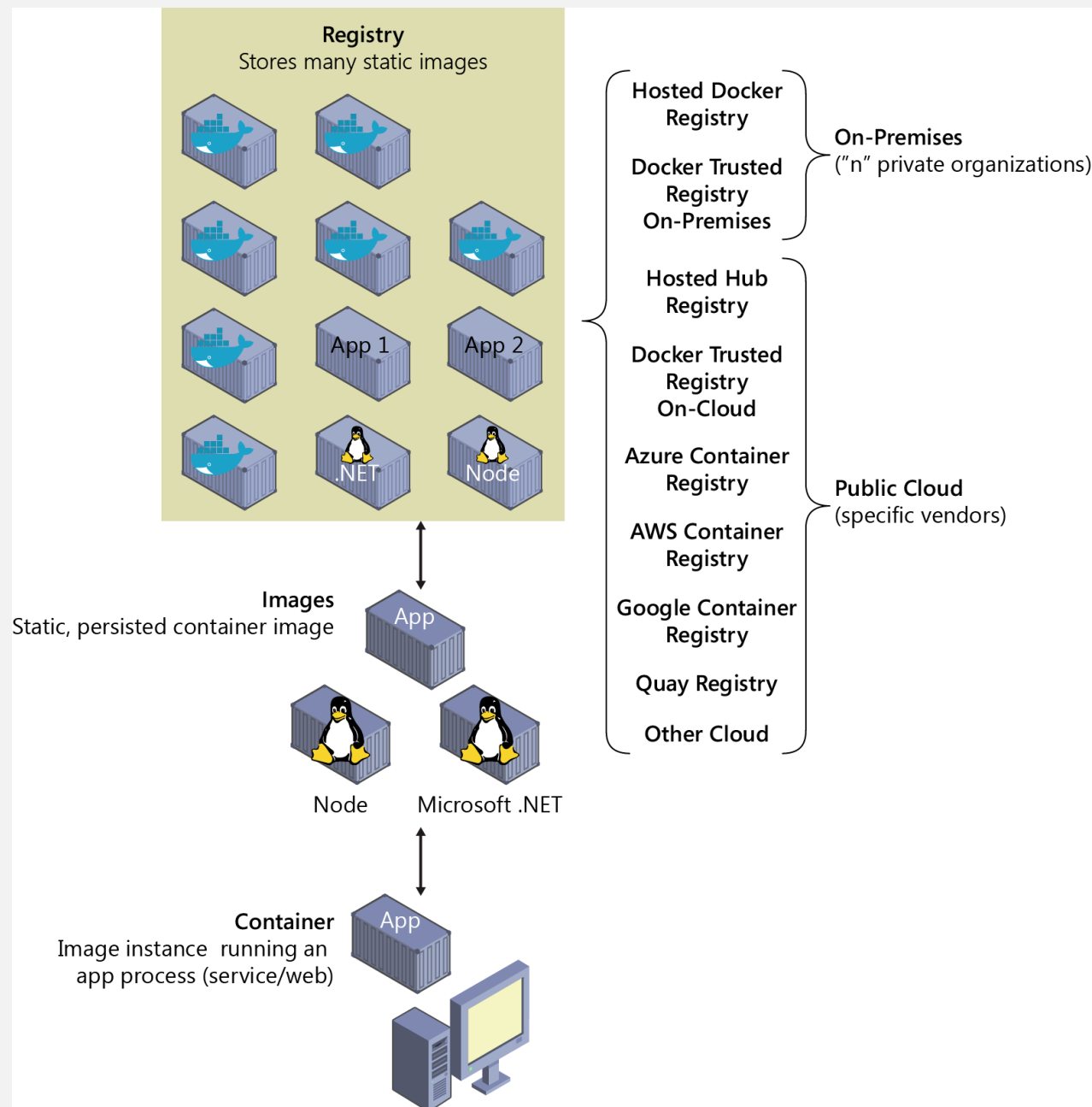
是一個最基礎的模板，用來重複產生容器實體

- **Container(容器):**

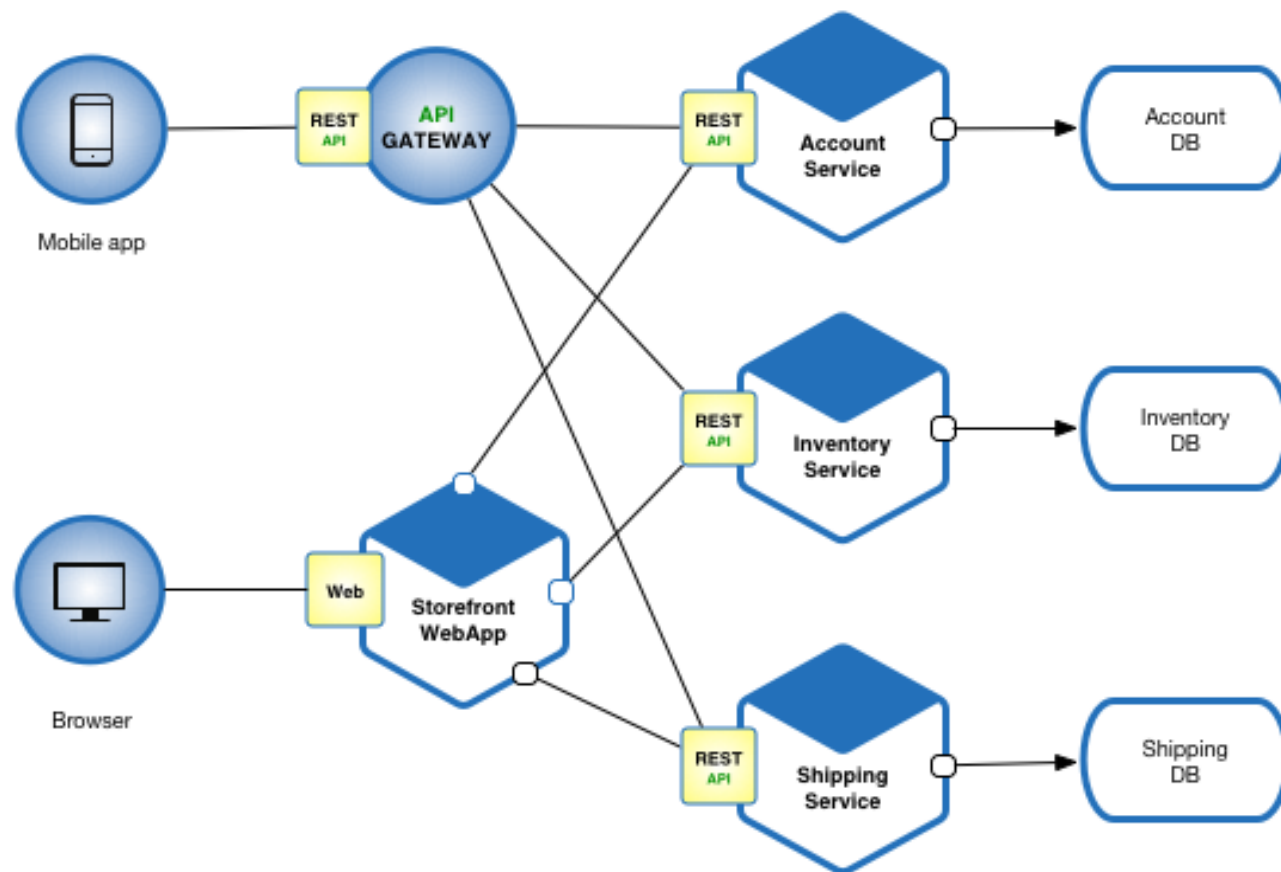
使用映像檔建立出來的執行實例

- **Registry(倉庫):**

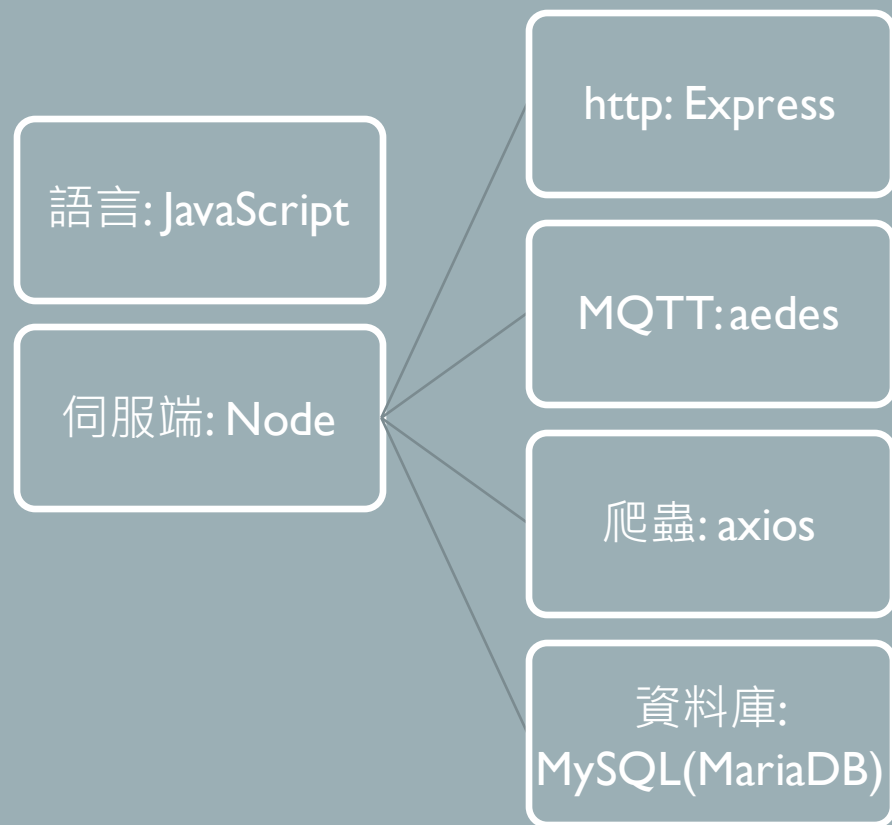
主要用來集中存放映像檔檔案的場所



# 後端 組成 架構




## 使用的程式語言&函式庫



express



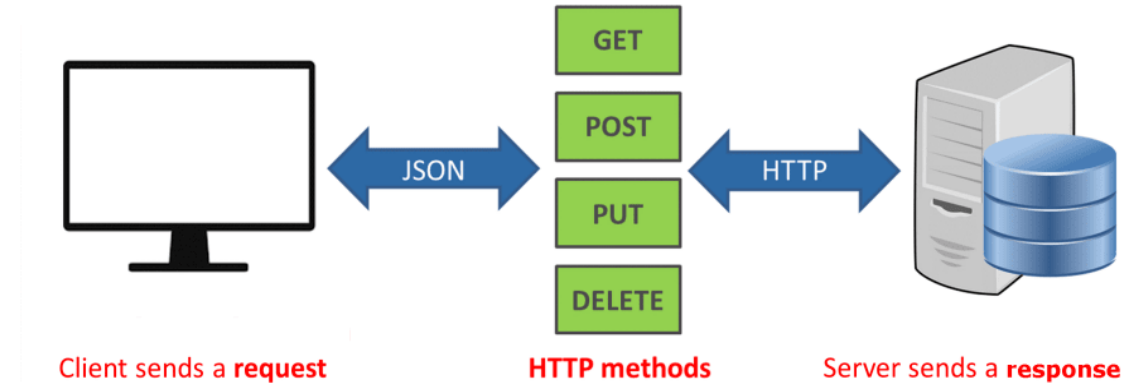
AXIOS



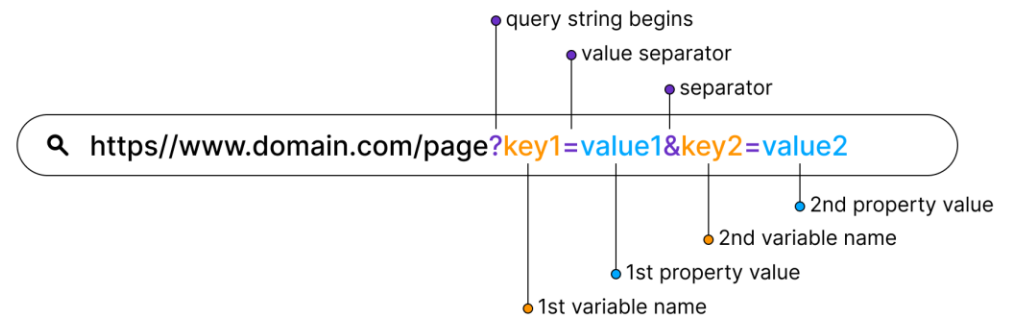
所使用的通訊協定

# HTTP REQUEST & RESPONSE

- 運行方式



- 使用Query & Param來讀取數值



# MQTT

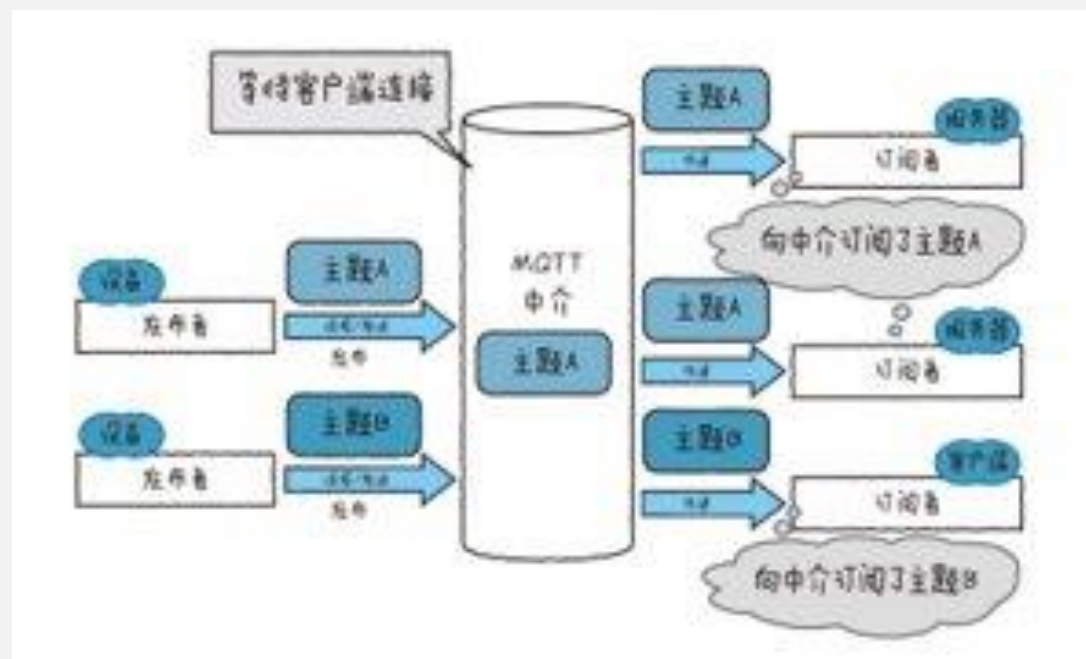
(MQ TELEMETRY TRANSPORT,  
消息隊列遙測傳輸)

是一種能實現一對多通信(發布/訂閱型)的協議

## 組成要素

- 伺服器端: 中介(BROKER)
- 客戶端
  - 發布者(PUBLISHER)  
調用資料庫的數值資料進行發布
  - 訂閱者(SUBSCRIBER)  
訂閱發布者發布的內容,使數值呈現於平台

## 通信方式



測試工具



**MQTT X**

MQTT: MQTTX



**POSTMAN**

HTTP: Postman



# 對於後端的其它部分

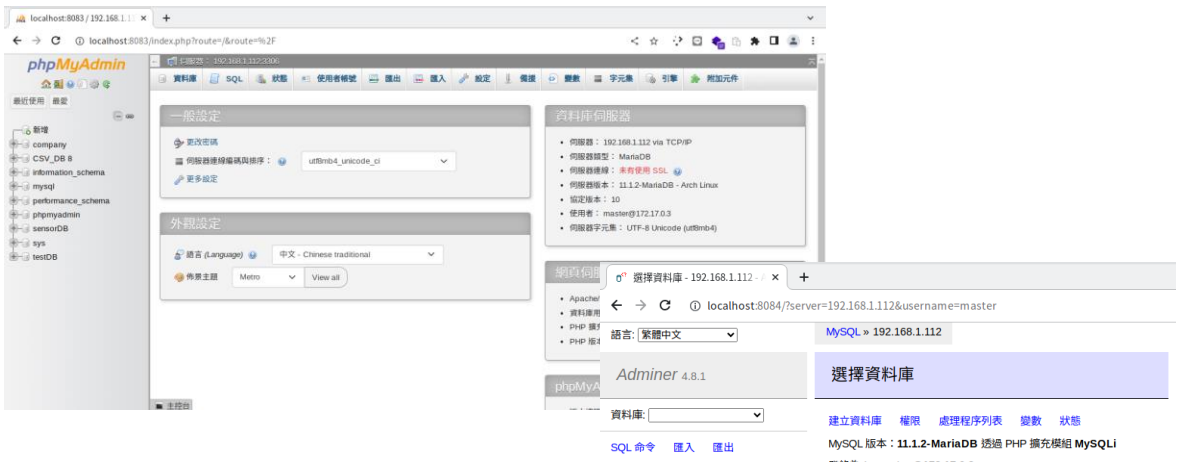
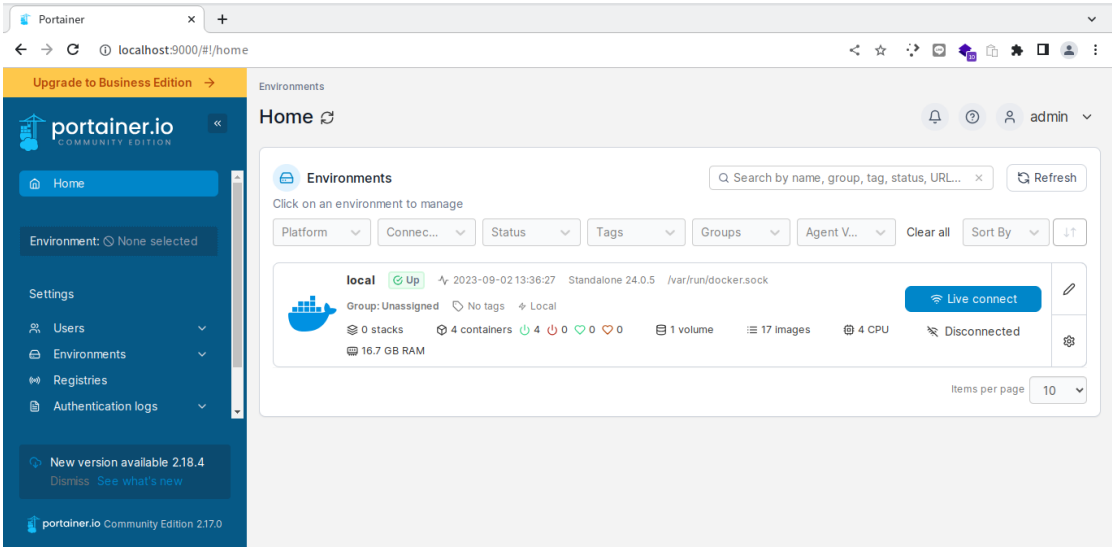
例如：建置&管理容器和API

# 使用管理介面來管理容器&資料庫

容器: Portainer



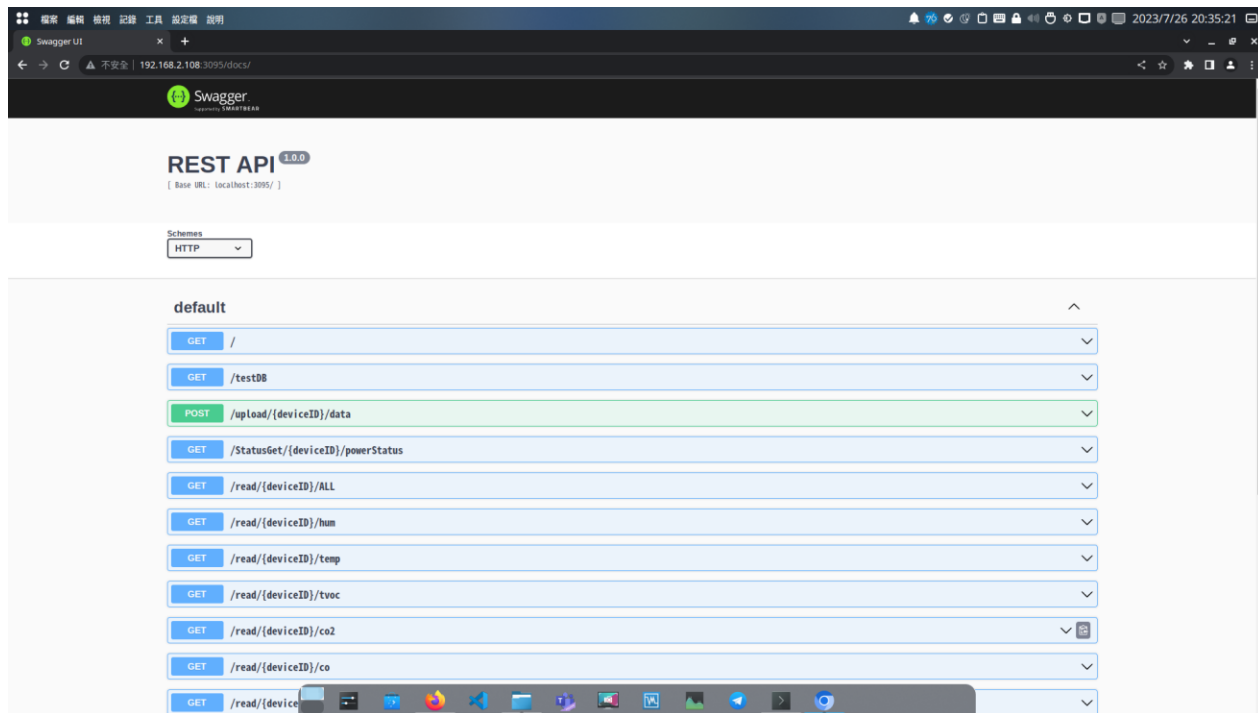
資料庫: phpmyadmin & Adminer



資料庫 - 重新載入		校對	資料庫	大小	計算
<input type="checkbox"/>	company	utf8mb4_unicode_ci	?	?	?
<input type="checkbox"/>	CSV_DB 8	utf8mb3_general_ci	?	?	?
<input type="checkbox"/>	information_schema	utf8mb3_general_ci	?	?	?
<input type="checkbox"/>	mysql	utf8mb4_unicode_ci	?	?	?
<input type="checkbox"/>	performance_schema	utf8mb3_general_ci	?	?	?
<input type="checkbox"/>	phpmyadmin	utf8mb4_unicode_ci	?	?	?
<input type="checkbox"/>	sensorDB	utf8mb4_unicode_ci	?	?	?
<input type="checkbox"/>	sys	utf8mb3_general_ci	?	?	?
<input type="checkbox"/>	testDB	utf8mb4_unicode_ci	?	?	?

已選中 (0)

刪除



# 建立可視化的API文件

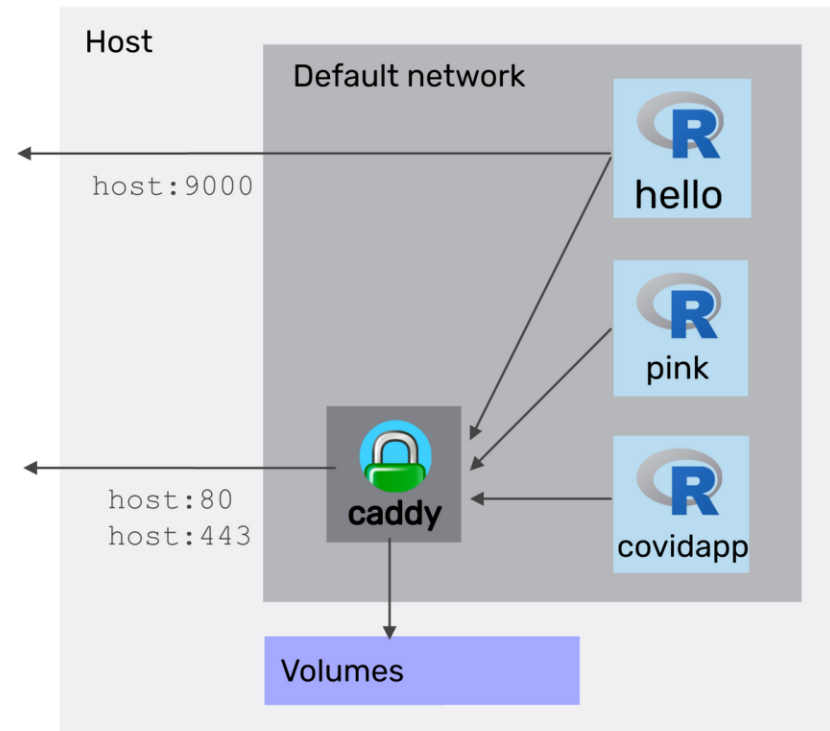
- 相關套件: swagger
- 實做過程:

<https://hackmd.io/@zangmenhsu/By5wQawuh>





© Analythium



## 自動建置部分

- 打包方式：手動打包+執行 => docker compose
- 官方文件：<https://docs.docker.com/compose/>
- 相關影片：

[https://www.youtube.com/watch?v=DM65\\_JyGxCo](https://www.youtube.com/watch?v=DM65_JyGxCo)

# DOCKERFILE

# 基於的映像檔

FROM alpine:latest

# 建立工作目錄

WORKDIR /usr/src/app

# 時區

ENV TZ=Asia/Taipei

RUN echo "\${TZ}" > /etc/timezone

RUN ln -sf /usr/share/zoneinfo/\${TZ} /etc/localtime

# 安裝&更新所需的套件

RUN apk update

RUN apk upgrade --no-cache

RUN apk add --no-cache nodejs npm icu-data-full tzdata

RUN rm /var/cache/apk/\*

# 把目錄下的程式碼直接復制到容器中

COPY package\*.json ./

RUN npm install

COPY . .

# 指定使用的端口

EXPOSE 3095

EXPOSE 3094

# 容器啟動時

CMD [ "npm", "start" ]

Command	Overview
FROM	Specify base image
RUN	Execute specified command
ENTRYPOINT	Specify the command to execute the container
CMD	Specify the command at the time of container execution (can be overwritten)
COPY	Simple copy of files / directories from host machine to container image
ADD	COPY + unzip / download from URL ( <b>not recommended</b> )
ENV	Add environment variables
EXPOSE	Open designated port
WORKDIR	Change current directory
MAINTAINER	<b>deprecated</b> now LABEL maintainer="maintainer@example.com" should be specified as

## 指令部分(DOCKERFILE ONLY )

- 01 打包成一個image

```
image="neko_0xff/iotgateway_server"
```

```
NetworkMode="host"
```

```
docker build . -t $image --network=$NetworkMode
```

- 02 運行Container

```
port1="3095:3095"
```

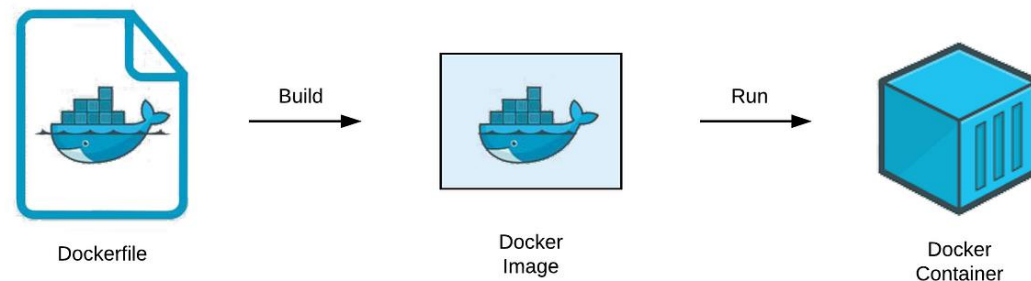
```
port2="3094:3094"
```

```
image="neko_0xff/iotgateway_server"
```

```
NetworkMode="host"
```

```
ContainerName="IoTGateway_Server"
```

```
docker run --network=$NetworkMode --name=$ContainerName --restart=always -p $port1 -p $port2 -d $image
```



## 指令部分(DOCKER COMPOSE )

### 01 Docker-compose.yml: 設置編譯時的配置

version: '3'

services:

container\_server:

restart: always # 跟系統服務一起重啟

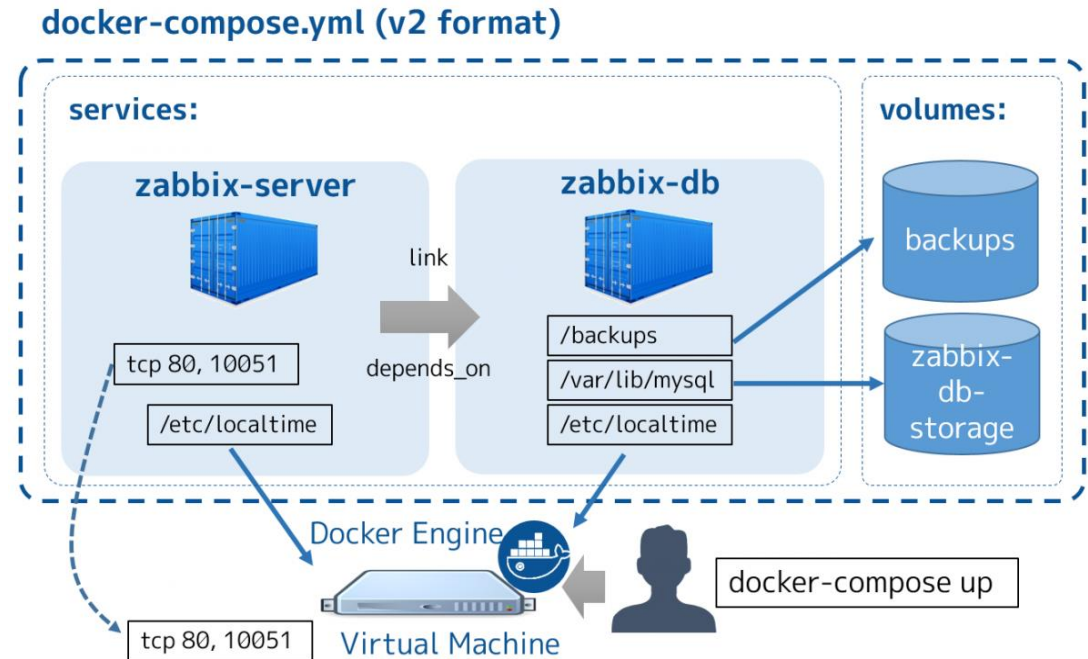
network\_mode: host # 網路: 使用實體機

# 編譯時的設置

build:

context: .

dockerfile: Dockerfile.env



### 02 同時打包成image+打包完後直接運行

\$ docker compose up --build -d

---

# 使用DOCKER COMPOSE 的特點

01

可在**docker-compose.yml**中定義  
所需運行的服務，同時  
提升使用者的可讀性

02

所有服務&容器的設置  
參數都可在同一份檔案  
中一目瞭然

03

可透過運行一個指令來  
啟動/停止所有的服務

---



---

## 使用MAKE 來管理容器的部分

- 如右圖所示，這是`Makefile`的內容
    - 其中定義了所有容器&映像檔的
      1. **build**: 開始**建置**映像檔&容器
      2. **up**: 開始**運行**相對應的容器
      3. **logs**: 檢視容器的**運行記錄**
      4. **stop**: **停止**運行相對應的容器
      5. **clean**: **停止且刪除**已建置映像檔&容器
- 

### M Makefile

```
1 CC1:=docker compose
2 IMAGE:=crawlerdata
3
4 .PHONY: build up logs stop clean
5
6 all: build
7
```

```
7
8 build:
9     @$(CC1) up --build -d
10
11 up:
12     @$(CC1) up -d
13
14 logs:
15     @$(CC1) logs --tail=100 -f
16
17 stop:
18     @$(CC1) stop
19
20 clean:
21     @$(CC1) down
22
```

---

# 使用MAKE來管理容器的部分

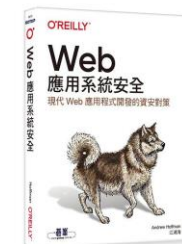
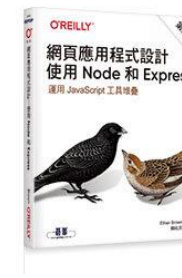
只要輸入一行`make`，則會馬上執行打包&停止容器等部分

```
zangmenhsu@Host-02 ~/文件/GitHub/2023_schoolResearch_ClientApp/src/viewapp_v1_1 <main*>
└─ sudo make
[sudo] zangmenhsu 的密碼：
[+] Building 212.5s (27/27) FINISHED                                docker:default
=> [web internal] load .dockerignore                                0.4s
=> => transferring context: 742B                                     0.0s
=> [web internal] load build definition from Dockerfile.env         0.3s
=> => transferring dockerfile: 1.30kB                                0.0s
=> [web internal] load metadata for docker.io/library/nginx:alpine  4.2s
=> [web internal] load metadata for docker.io/library/ubuntu:latest 4.0s
=> CACHED [web stage-1 1/4] FROM docker.io/library/nginx:alpine@sha256:16164a43b5faec40adb521e98272edc52 0.0s
=> [web build-env 1/17] FROM docker.io/library/ubuntu:latest@sha256:ec050c32e4a6085b423d36ecd025c0d3ff0 0.0s
=> [web internal] load build context                                8.9s
=> => transferring context: 57.78MB                                  8.3s
=> CACHED [web build-env 2/17] RUN echo "Asia/Taipei" > /etc/timezone 0.0s
=> CACHED [web build-env 3/17] RUN ln -sf /usr/share/zoneinfo/Asia/Taipei /etc/localtime 0.0s
=> CACHED [web build-env 4/17] RUN apt-get update                  0.0s
=> CACHED [web build-env 5/17] RUN apt-get install -y curl git wget unzip libgconf-2-4 gdb libstdc++6 l 0.0s
=> CACHED [web build-env 6/17] RUN apt-get install -y clang cmake ninja-build pkg-config libgtk-3-dev l 0.0s
=> CACHED [web build-env 7/17] RUN apt-get clean                   0.0s
=> CACHED [web build-env 8/17] RUN git clone https://github.com/flutter/flutter.git /usr/local/flutter 0.0s
```

---

# 相關書籍

1. 小笠原種高 . (2021). *圖解 Docker & Kubernetes 的知識與使用方法*(衛宮紘, Trans.). 碁峰資訊.
2. Brown Ethan . (2020). *使用 Node 和 Express, 2/e (Web Development with Node and Express, 2/e)* (賴屹民, Trans.). 歐萊禮.
3. Hoffman Andrew (2021). *Web 應用系統安全 / 現代 Web 應用程式開發的資安對策 (Web Application Security)* (江湖海, Trans.). 歐萊禮.
4. 吳瑞北. (2020). 物聯網ABC = Internet of things : AI/Big data/cloud.國立臺灣大學-出版中心
  - Cource: [http://cc.ee.ntu.edu.tw/~rbwu/pages/course.html#IoT\\_Intro](http://cc.ee.ntu.edu.tw/~rbwu/pages/course.html#IoT_Intro)



---

# 網路來源引用

- 王宏仁. (2015, May 15). *Docker* 掀起微服務革命，廠商搶進Container OS競賽. IThome.  
<https://www.ithome.com.tw/news/95752>
  - Andrew Wu, (2017, April 15). 架構師觀點 - 轉移到微服務架構的經驗分享 (Part 1). 安德魯的部落格.  
<https://columns.chicken-house.net/2017/04/15/microservice8-case-study/>
  - 邱全成 . (n.d.). 雲報專欄：雲端運算下容器(Container)技術和應用. 台灣雲端物聯網產業協會.  
[http://www.twcloud.org.tw/files/file\\_pool/1/0i235396067708866326/5.pdf](http://www.twcloud.org.tw/files/file_pool/1/0i235396067708866326/5.pdf)
-

---

報告結束

---