



docker

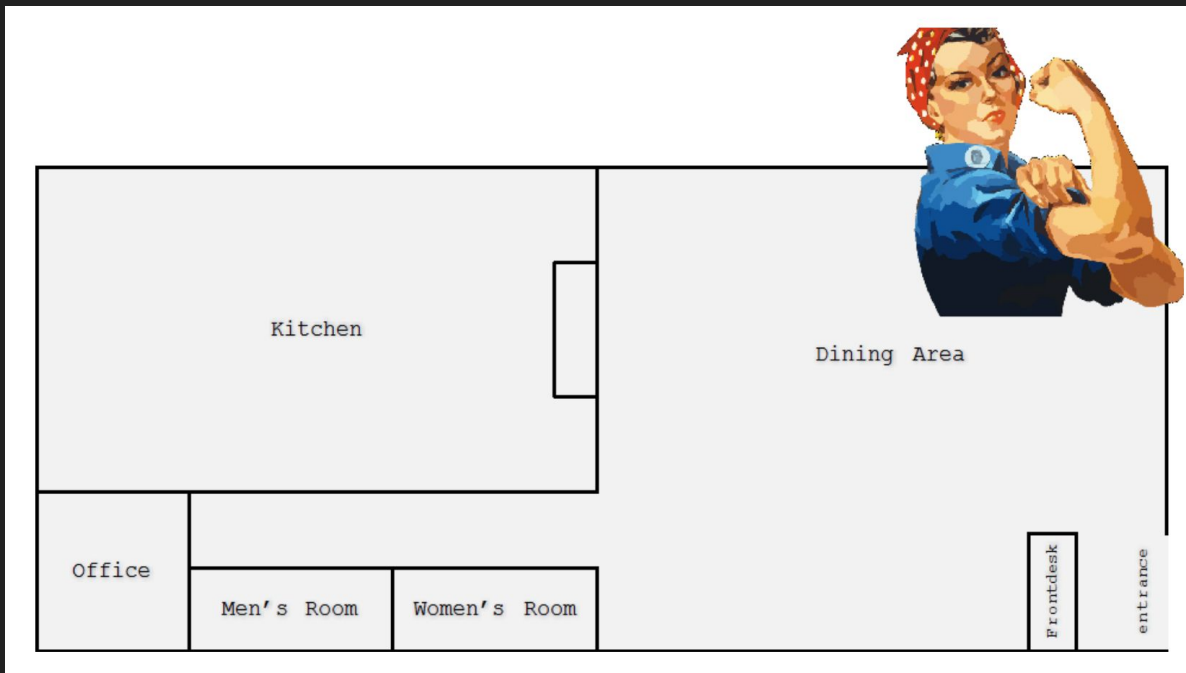
# 入門與實踐

蔡宗憲 (Andrew)

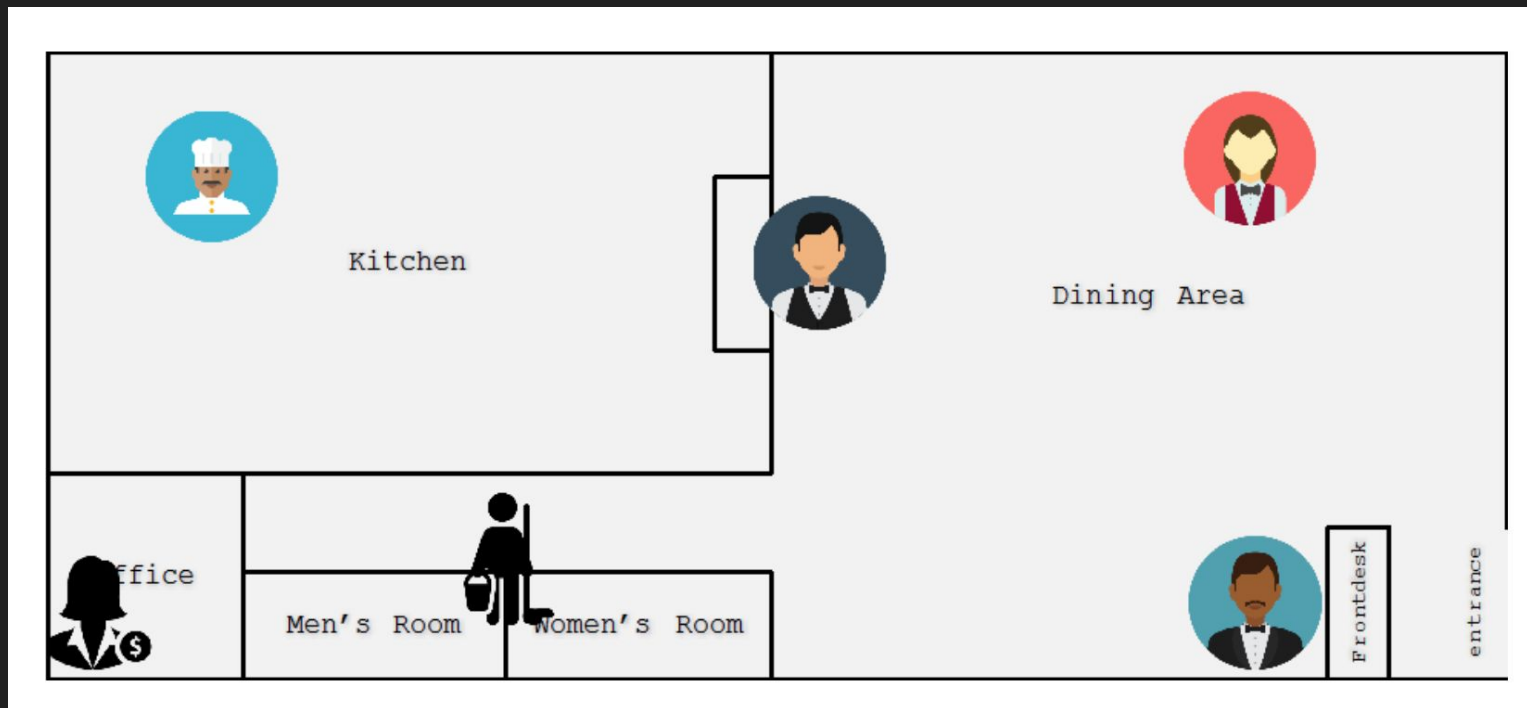
# 為什麼要教 Docker

- DevOps / 測試
- 微服務架構
- 跨平台
  - 程式碼編譯
  - 取得所有相關 library
  - 作業系統路徑

# 微服務架構 - 話說從頭，一個巨大的怪獸



# 微服務架構



# 微服務的優缺點

- 優點

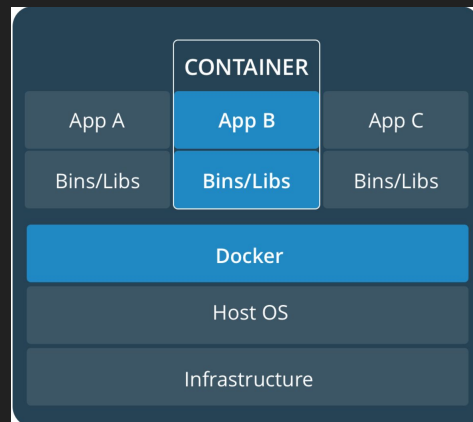
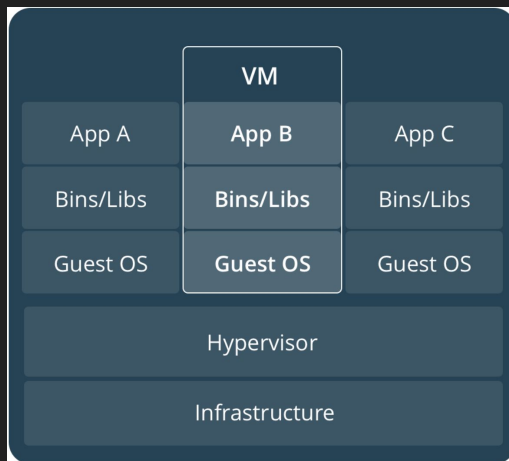
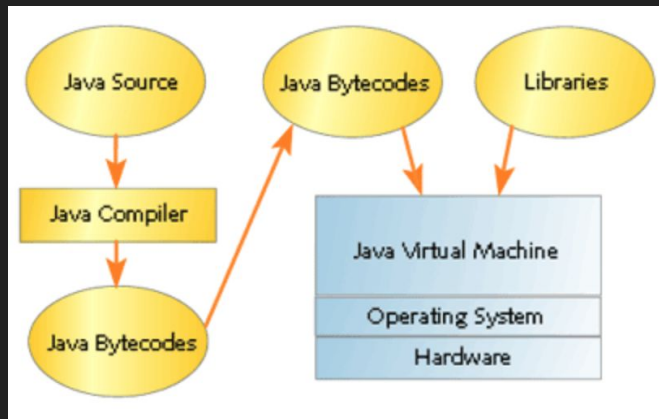
- 擴充性 / 負載平衡 / ...
- 簡化程式碼的複雜度 / 獨立開發
- 獨立部署
- 獨立運作 (簡少受其它程式干擾)

- 缺點 (燒錢)

- 系統複雜度與額外成本 (RPC)
- 跨月服務的協調
- 整合測試
- 部署
- 管理與監控 - Mesos, Kubernetes

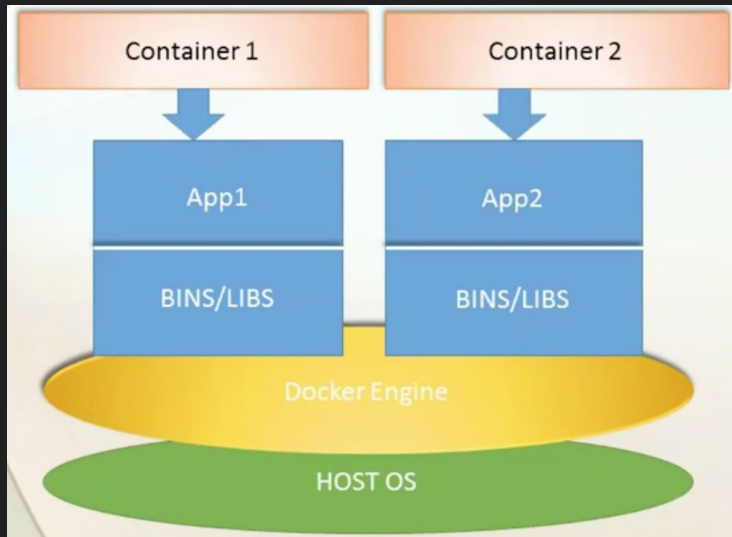
# 跨平台的解法

- Java Virtual Machine
- Virtual Machine
- Docker



# Docker 特點

- Build, Ship, and Run Any App, Anywhere
  - 不受程式語言限制 (相對 Java Virtual Machine 而言)
  - 輕量型 (相對於 Virtual Machine 而言)
  - 但是極輕量型 (手機) ?? 舊版的 windows ??
- 一包搞定
  - 設定檔位置固定 (不用處理權限問題)
  - 所需要的 library 都包在一起了
  - 環境乾淨, 測試方便
  - 容易清乾淨
  - DevOps: 大家都處於同一個環境
- 資源隔離 (port, disk, cpu, ...)



# 使用 Docker 的公司



[Learn More](#)



[Learn More](#)



[Learn More](#)



[Learn More](#)



[Learn More](#)



[Learn More](#)



## 公司介紹

產業類別：電腦系統整合服務業

產業描述：電腦系統整合服務業

員 工：暫不提供

資 本 額：暫不提供

聯 絡 人：曾小姐


公司地址：台北市大安區忠孝東路4段  地圖

電 話：暫不提供

傳 真：暫不提供

公司網址：<http://www.touchcloud.com.tw>

相關連結：[更多](#) ▼

[儲存公司](#) [相似公司](#)

## 公司簡介

- \* We are virtualization experts + IoT gurus
- \* Funded in 2016, Jan.
- \* Founded by four stock listed companies
- \* Focus on cloud virtualization technology and target for IoT market
- \* Think Globally! Act Locally!

## 主要商品 / 服務項目

- 1) Big Data Analysis with Machine Learning
- 2) Computer Vision with Deep Learning
- 3) Openstack implementation
- 4) **Docker** cluster solutions
- 5) IoT total solutions

# DevOps/Docker微服務工程師

歐立威科技股份有限公司    本公司其他工作

## ■ 工作內容

1. 協助建置Docker環境、CI/CD相關流程與自動化測試
2. 建置與管理自動化佈署及監控機制
3. 熟悉網路、資安、Linux環境及各項服務，能夠提供產品開發上的建議
4. 熟Java、Python等程式開發
5. 有DevOps運營或開發經驗者佳
6. 有Docker或Kubernetes經驗者佳
7. 良好溝通協調能力

# 安裝 Docker for Windows

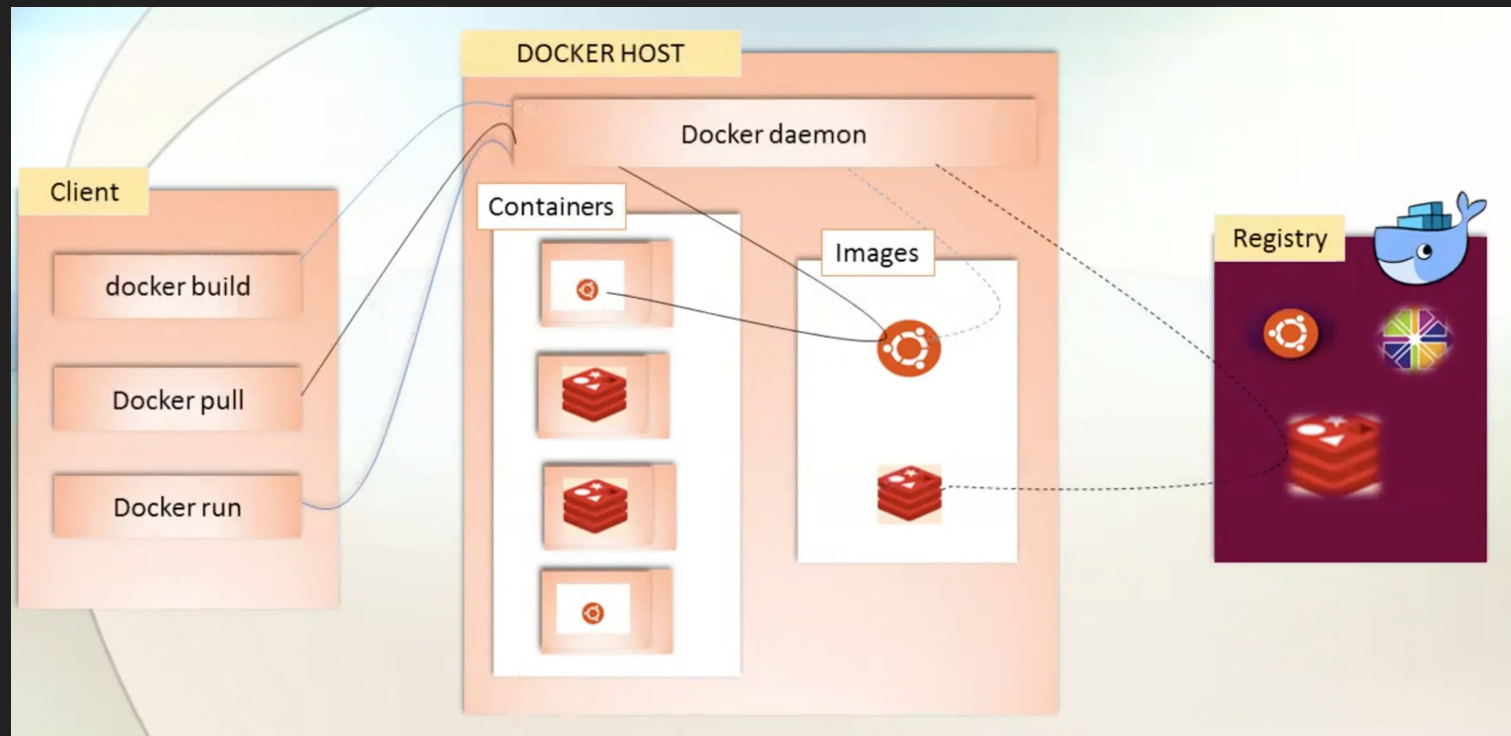
- [.net framework v4.0.30319](#)
- [Docker toolbox on Windows](#)
- [Docker for Windows](#)

# 基本概念

# 基本名詞

- Image (映像檔)
- Container (容器)
- Repository (倉庫)
- 在 ubuntu 底下, 所有 docker 相關指令要加 sudo
  - 每個指令 sudo docker ...
  - 先 sudo su, 之後就不用再加 sudo

# Docker 架構



指令

# Docker 常駐程式指令

- `service docker start`
- `service docker stop`
- `service docker restart`



# 查詢指令

- docker version
- docker info
- --help 很好用

# 映像檔指令

- 取得映像檔: `docker pull <image>`
- 列出所有映像檔: `docker images`
- 移除映像檔: `docker rmi <image>`
- 存出: `docker save <image> -o <tar archive file>`
- 載入: `docker load -i <tar archive file>`
- 標示來源: `docker tag <image> <remote image>`
- 將映像檔放回 registry: `docker push <remote image>`

# 練習1: 取得 hello-world 映像檔

- docker pull hello-world

```
ca4f61b1923c: Pull complete  
error pulling image configuration: Get https://registry-1.docker.io/v2/library/hello-world/blobs/sha256:f2a91732366c0332ccd7afd2a5c4ff2b9af81f549370f7a19acd460f87686bc7: net/http: TLS handshake timeout
```

- service docker start
- service docker status
- docker pull hello-world
- docker images
- docker rmi hello-world
- 試試看: ubuntu:14.04

# 容器指令(1)

- 新建並啟動容器: `docker run <image>`
  - `-d`: 長駐背景執行
  - `-t`: 虛擬終端
  - `-i`: 標準輸入模式保持打開
  - `--name`: 容器名稱
  - `-v`: 加入的空間 (volume) `<外部空間:內部空間>`
  - `-p`: 連接埠 (port) 的對應 `<外部連接埠:內部連接埠>`
- 查詢容器狀態: `docker ps` (目前執行中的容器)
  - `-a`: 所有現存的容器
- 改變容器狀態
  - 啟動容器: `docker start <container>`
  - 終止容器: `docker stop <container>`
  - 重新啟動容器: `docker restart <container>`

## 練習2: 印出一個 hello world

- `docker run ubuntu:14.04 echo "hello world"`
- `docker ps`
- `docker ps -a`

## 練習3: 進入容器中觀察

- `docker run -it ubuntu:14.04 bash`
- `docker ps`
- `cd /home/`
- `ls -lrt`
- 開新終端機 & `docker ps`
- `exit`
- `docker ps`
- `docker ps -a`

## 容器指令 (2)

- 從容器建立新的映像檔: `docker commit -m <message> -a <author> <container> <repository>:<tag>`
- 對一個執行中的容器執行指令: `docker exec <container>`
  - i
  - t
  - d

## 練習4: 修改容器, 建立一個具備 Git 的映像檔

- 再次進入容器
- `apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys E1DD270288B4E6030699E45FA1715D88E1DF1F24`
- `su -c "echo 'deb http://ppa.launchpad.net/git-core/ppa/ubuntu trusty main' > /etc/apt/sources.list.d/git.list"`
- `apt-get update`
- `apt-get install git`
- `docker commit -m "ubuntu with git" -a "andrew" <container id>`  
`git-ubuntu:14.04`
- `docker images`
- `docker run -it git-ubuntu:14.04 bash`



## 練習5: 映像檔的存出和載入

- `docker save git-ubuntu:14.04 -o git-ubuntu.tar`
- `docker rmi git-ubuntu:14.04`
- `docker load -i git-ubuntu.tar`

## 練習6: 容器執行命令

- `docker run --name demo -td ubuntu:14.04 bash`
- `docker ps`
- `docker exec -it <container id> touch allo`
- `ls`
- `docker exec -it demo bash`
- `ls`
- `exit`

## 練習7: 容器啟動、停止、重啟、刪除

- `docker stop <container>`
- `docker ps`
- `docker start <container>`
- `docker ps`
- `docker exec -it <container> ls`
- `docker restart <container>`
- `docker ps -a`
- `docker rm <container> ???`
- `docker stop <container>`
- `docker rm <container>`
- `docker ps -a | grep <container>`

## 練習8: nginx (共享空間, 連接埠對應)

- `docker run --name nginx-without-mount -p 8081:80 -d nginx`
- 打開瀏覽器: <http://localhost:8081>
- `mkdir ~/nginx/`
- 進入容器中複製檔案內容: `/etc/nginx/conf.d/default.conf` 至 `~/nginx/conf/demo.conf`
- 打開 `demo.conf`, 將 `access log` 前的 `#` 去除 (去掉註解)
- `docker run --name nginx-with-mount -p 8082:80 -v ~/nginx/log:/var/log/nginx -v ~/nginx/conf:/etc/nginx/conf.d -d nginx`
- `tail -f ~/nginx/log/host.access.log`
- 打開瀏覽器: <http://localhost:8082/test>
- 觀看終端機的變化

## 練習9: docker registry

- `docker run -d -p 5000:5000 --restart=always -v ~/docker/registry:/var/lib/registry --name registry registry:2`
- 確認 `git-ubuntu:14.04` image 仍然存在於 `docker images` 中
- `docker tag git-ubuntu:14.04 localhost:5000/demo/git-ubuntu:14.04`
- `docker push localhost:5000/demo/git-ubuntu:14.04`
- 問題: 和 “`docker push git-ubuntu:14.04`” 的差別 ??

# Dockerfile

# 為什麼要有 Dockerfile

- 透明化 (隨便一個 docker image 不知道裡面有什麼東西)
- 流程版本控管

# Dockerfile 的內容

- 格式
  - INSTRUCTION 參數
  - # 註解
- 基本內容
  - 基底映像檔
  - 關於這個映像檔的描述
  - 操作指令
  - 容器啟動時執行指令



# 基底映像檔

FROM ubuntu:14.04

# 關於這個映像檔的描述

- LABEL <key1>=<value1> <key2>=<value2>
  - LABEL maintainer="[apfols@gmail.com](mailto:apfols@gmail.com)"

# 操作指令

- RUN
  - RUN <command>
    - RUN /bin/bash -c “echo hello”
  - RUN [ <executable>, <param1>, ..]
    - RUN [“/bin/bash”, “-c”, “echo hello”]
- ENV <key1>=<value2> <key2>=<value2>
- COPY <src> <dest>
- ENTRYPOINT (nginx)
  - ENTRYPOINT <command>
  - ENTRYPOINT [ <executable>, <param1>, ..]
- EXPOSE <port>

# 容器啟動時執行指令

- CMD
  - RUN <command>
  - RUN [ <executable>, <param1>, ...]
  - RUN [<param1>, <param2>, ...] (with entrypoint)

## 練習10: 具有 git 的 ubuntu docker file

docker-compose

# 基本介紹

- .env file
- 指令
  - docker-compose up ⇒ nohup docker-compose up &
  - docker-compose stop

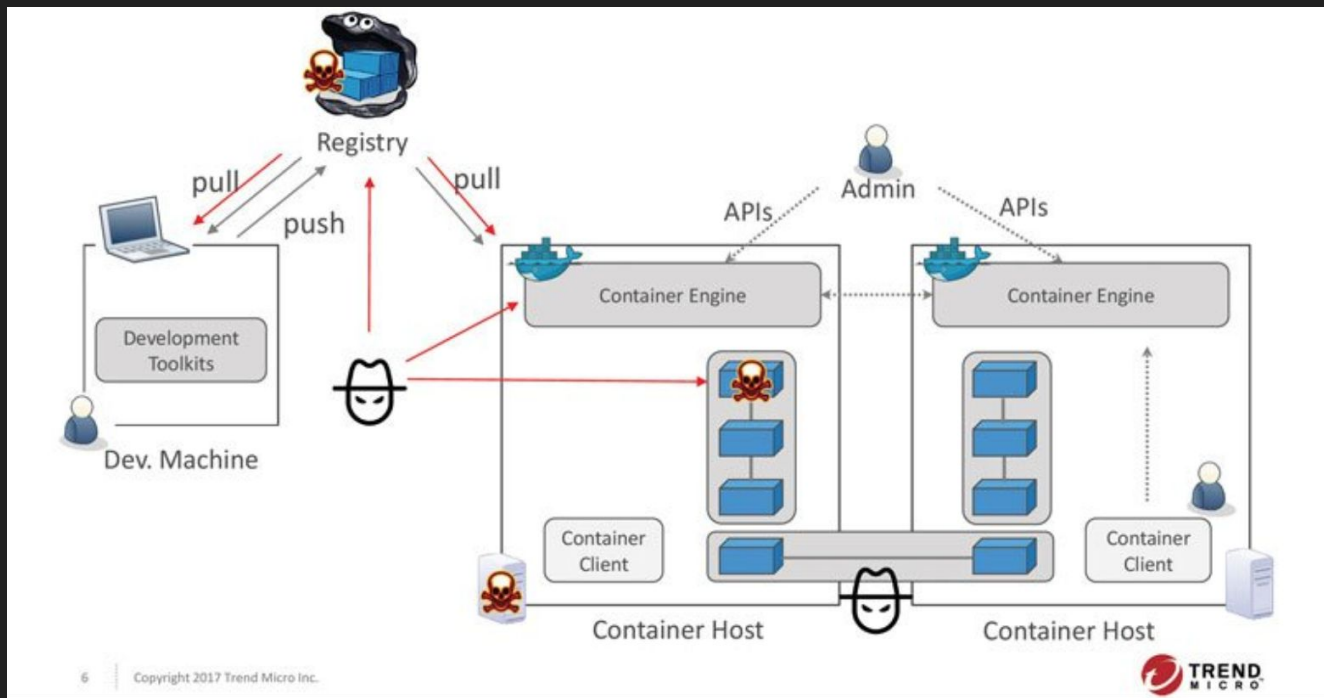
潛在問題



# 容器的管理

- Kubernetes
- Mesos

# Docker 的安全漏洞



# Reference

- [Docker -- 從入門到實踐](#)
- [微服務介紹](#)
- [簡單認識「微服務」概念](#)
- [Docker 容器常見攻擊手法與防護對策](#)
- [Dockerfile簡單介紹](#)
-