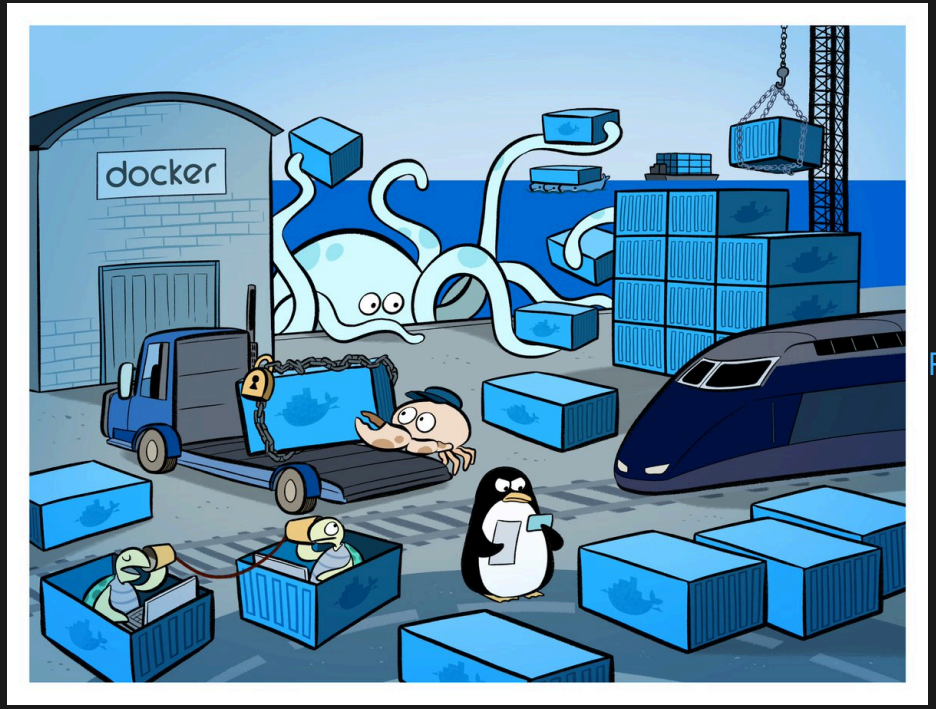
什麼是 Docker？

Docker 字面上的意思是「碼頭工人」，碼頭上會有打包、運送...等服務，碼頭工人 (Docker) 可快速的用貨櫃 (Container) 將貨物 (Application) 裝上船。

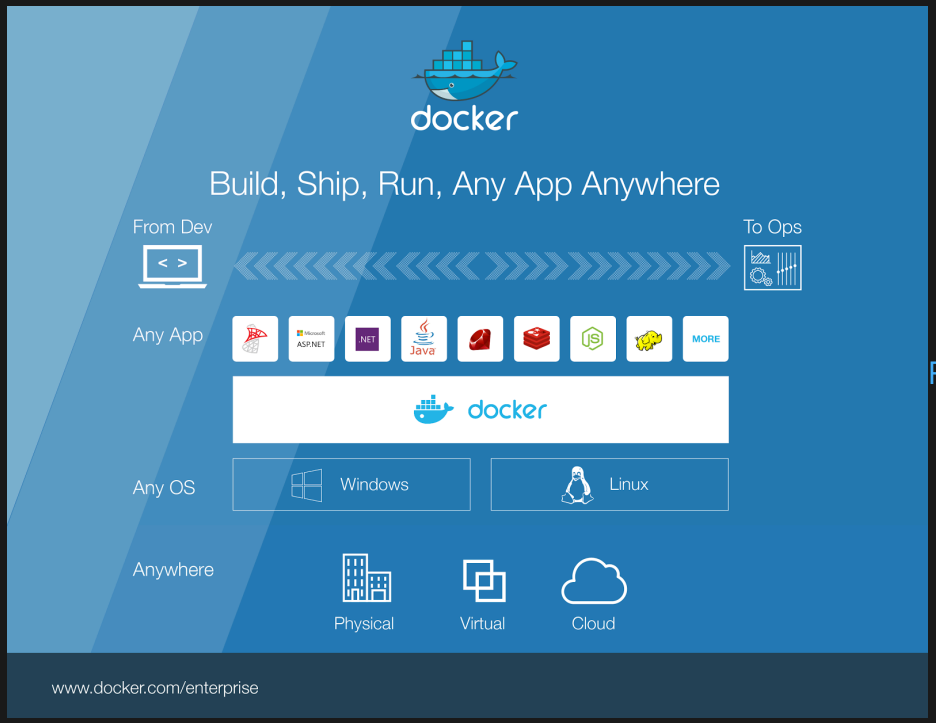


Ref

- Docker 是一種容器化技術 (輕量級的虛擬化技術)，可以把你的應用程式連同環境一起打包，部署的時候不用擔心環境的問題
- 原始碼在 GitHub 上進行維護，專案名稱為 Moby
- 使用 Go 語言開發
- 不需要用到 Hypervisor，直接利用硬體效能

Docker 能做什麼？

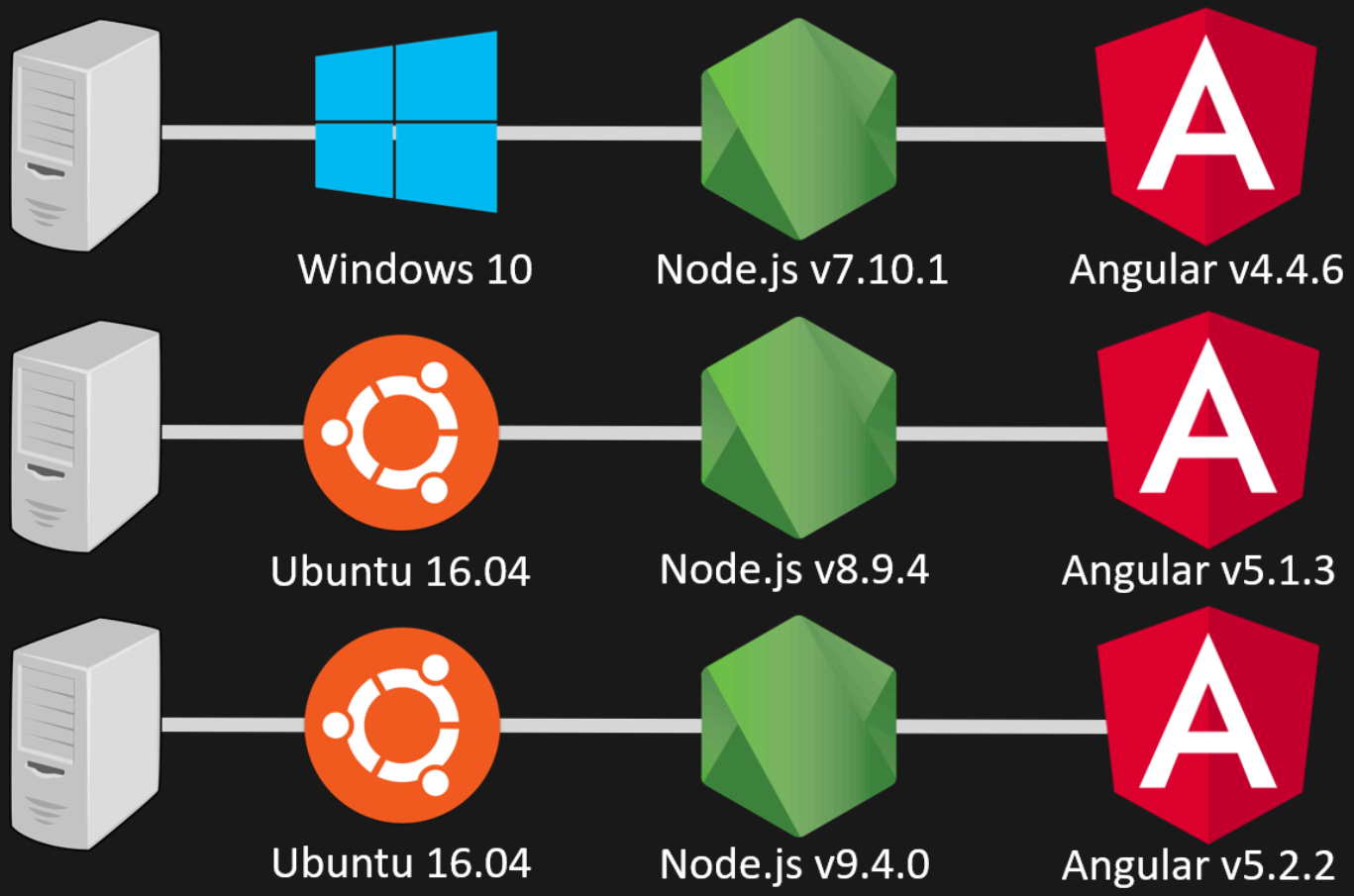
建置 image，然後使用 Registry 來管理 image，再使用 Docker Engine 將 container 和包含的 App 在任意平台 (不管是實體機、虛擬機還是雲端) 上執行



Ref

為什麼要使用 Docker？

以 Node.js 為例，每個專案都需要有開發環境



傳統架構缺陷

- 維護成本高 (機器硬體維修汰換)
- 建置機器複雜 (雖然有寫 Shell Script)
- 24 小時 on call (公司隨時都有人加班，有問題就要馬上修)
- 系統擴充困難

如果把架構換成 VM 呢？

雖然較省錢，但每個專案都需維護 VM 檔案，以 VMware 為例，包括 虛擬磁碟 (.vmdk)、VM 的記憶體 (.vmem)、虛擬 BIOS (.nvram)、設定檔 (.vmx)、補充設定檔 (.vmxf)、紀錄檔 (.log)、快照 (.vmsd)、快照狀態資訊 (.vmsn)、紀錄 VM 暫停狀態 (.vmss) ... 等檔案

```
D:\VM\Ubuntu $ ls -al
total 30558838
drwxr-xr-x 1 Titan 197609          0 一月 10 01:42  .
drwxr-xr-x 1 Titan 197609          0 八月 10 11:21  ..
-rw-r--r-- 1 Titan 197609       8684 一月 10 01:42 'Ubuntu.n
-rw-r--r-- 1 Titan 197609 27544518656 一月 10 01:42 'Ubuntu.v
-rw-r--r-- 1 Titan 197609          0 八月  6 21:37 'Ubuntu.v
-rw-r--r-- 1 Titan 197609       2994 一月 10 01:42 'Ubuntu.v
-rw-r--r-- 1 Titan 197609        383 八月  6 23:17 'Ubuntu.v
-rw-r--r-- 1 Titan 197609 3741319168 一月 10 00:59 'Ubuntu-f
-rw-r--r-- 1 Titan 197609    6130557 一月 10 01:42 'Ubuntu-f
-rw-r--r-- 1 Titan 197609     270160 一月 10 01:42  vmware.l
...
```

VM 的缺點

- 吃系統記憶體資源
- 切換多重專案浪費時間
- 浪費主機硬碟空間
- ...etc

改用 Docker 後

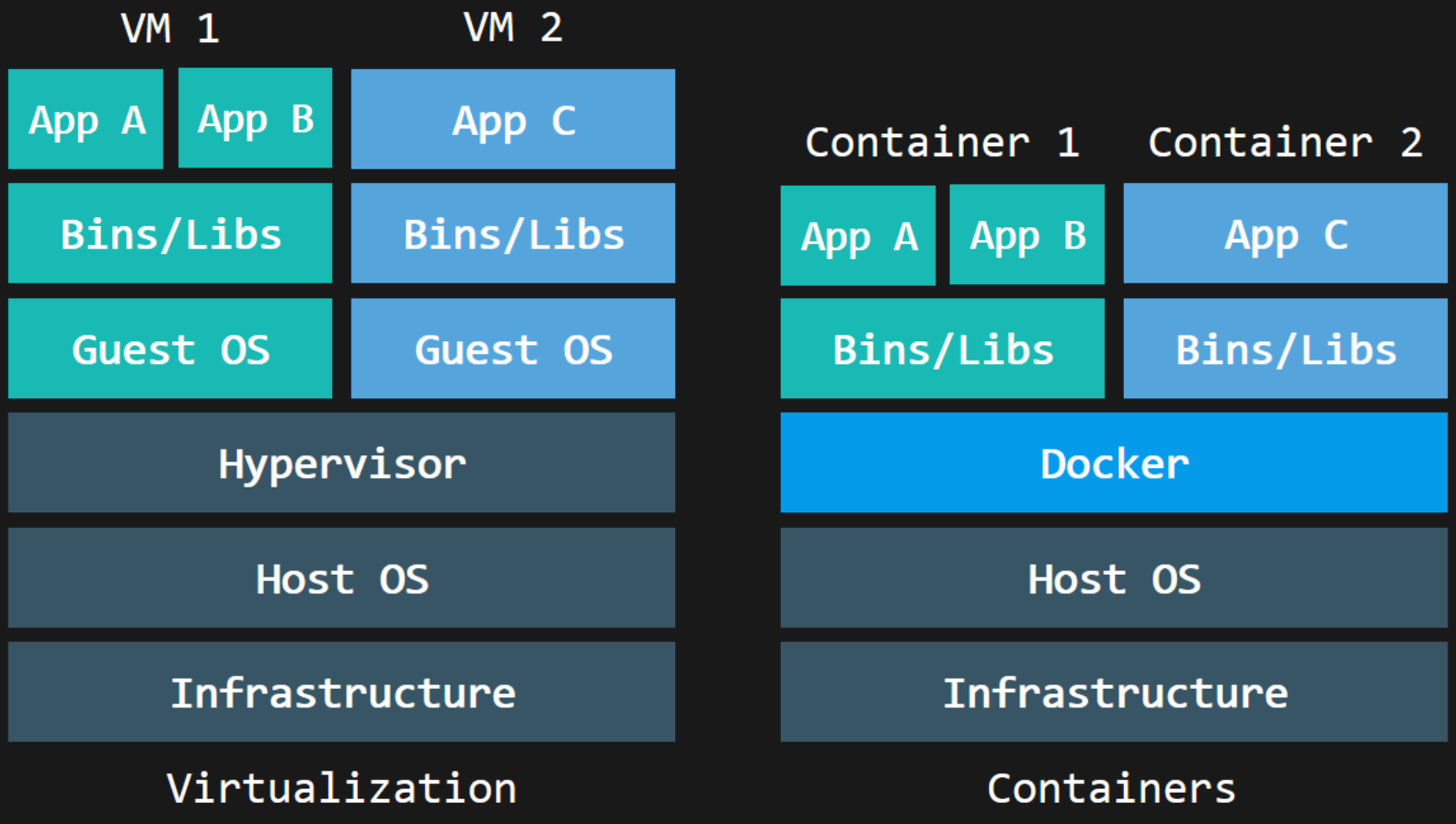
- 不需要額外的機器及人力維護成本
- 各專案可透過 Dockerfile 將環境進行版本控制
  - 可以利用 Dockerfile 把環境寫成一個檔案，在透過 Dockerfile 快速建立環境
- 任何環境都可執行：不管是實體機、虛擬機、雲端，不管系統是 Windows、Linux 都可以

所以為什麼要使用 Docker？

- 容易部署
- 快速建置環境，可自動建立開發環境
- 開發與正式環境一致
- 環境 (資源) 隔離
- 資源有效利用 (直接使用系統資源)
- 降低維運成本
- 比虛擬化技術更輕量

Virtualization vs Containerization

(虛擬化技術 vs 容器化技術)



名詞解釋

- Host OS：實體電腦的 OS。
- Guest OS：從 Host OS 裡面啟動的 VM，上面所跑的 OS 稱為 Guest OS。
- Hypervisor：Host OS 裡面負責管理 VM 的 Apps 稱為 Hypervisor 或 Virtual Machine Monitor (VMM，虛擬機器監視器)，Hypervisor 是在硬體層虛擬化。

虛擬化技術 (Virtualization)

常見：VirtualBox、VMWare

以往的虛擬化技術是在 Host OS 上建立虛擬環境，透過 Hypervisor 模擬一套完整的硬體環境資源 (Guest OS)，目標是建立一個可以用來執行整套 OS 的沙箱獨立執行環境，所以 VM 可以建立多個獨立的環境，使用者就能 Guest OS 上安裝其他應用程式

容器化技術 (Containerization)

而容器化技術則是透過在 Host OS 上執行 Container Engine 來建立各個 Container (虛擬執行環境，直接使用 Host OS 的系統資源)，每個 Container 也都是彼此獨立，但 Container 是共用相同的核心，共享系統資源

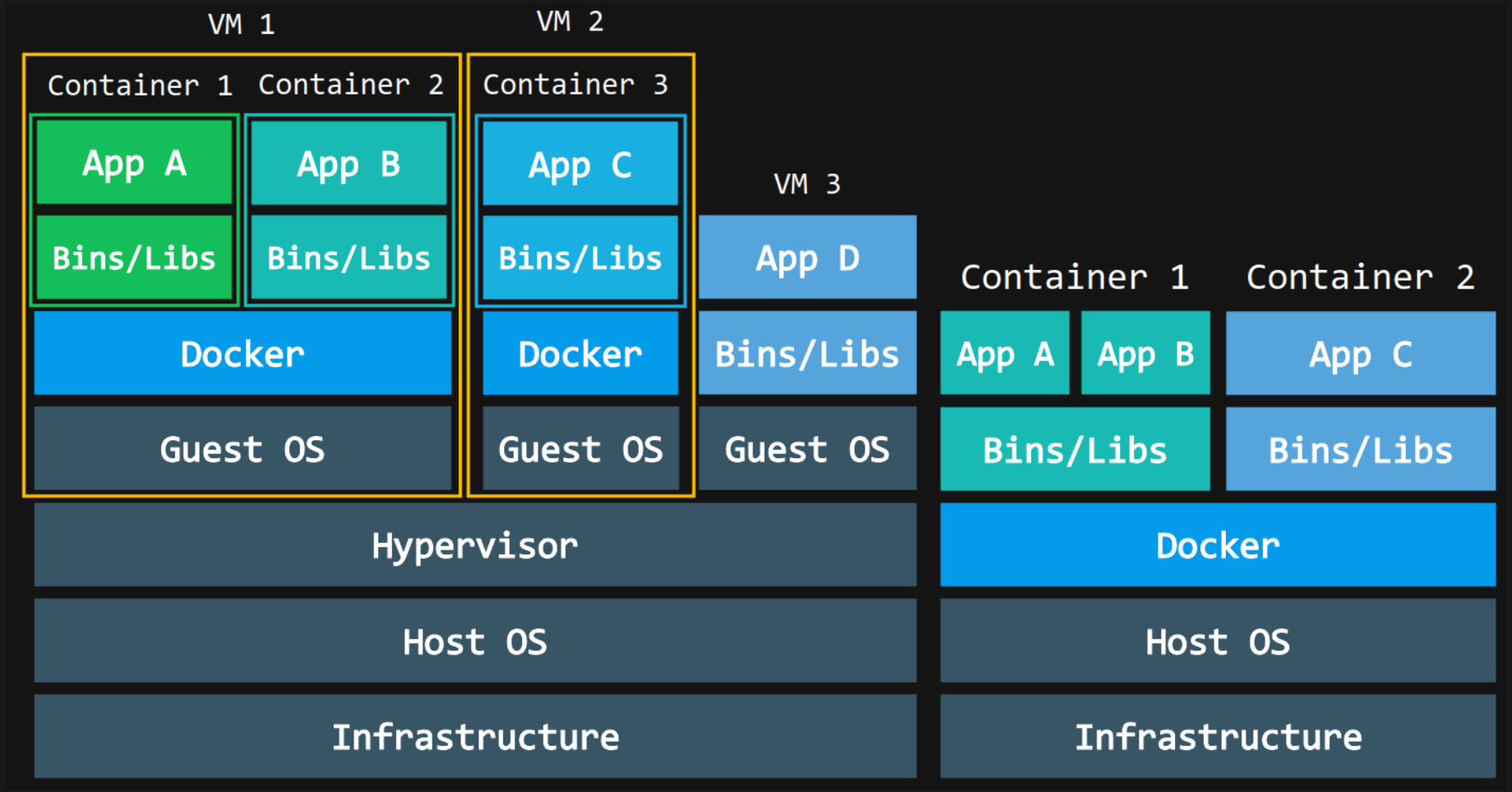
Docker 為什麼會比傳統 VM 輕量？
因為少跑了 N 個完整的 Guest OS

其實 container (容器) 和 VM 的目標是相似的，都是為了將 APP 和其依賴分離成可以在任何環境上獨立執行。另外，container 和 VM 減少了對於實體硬體的需求，無論是在成本和效益上，都可以更加有效的利用運算資源。

Docker 為什麼會比傳統 VM 輕量？

	Virtualization	Containerization
啟動	慢 (最快也要分鐘)	快 (秒開)
容量	大 (GB)	小 (MB)
效能	慢	快
host 可支撐數量	數個 ~ 數十個	數個 ~ 數百個
複製相同環境	超慢	快

Docker with VM



- Docker 和 VM 不只可以單獨的使用，也可以搭配使用。
- 假如要確保整體系統完整的虛擬化，就要先用 VM 安裝 OS，然後在 VM 中的 OS 上使用 Docker 啟動需要執行的 container。
- 可以按照使用者的使用情境來決定要如何安排使用 VM 和 Docker。

Docker 部署在實體伺服器 vs 部署在 VM 中

原生硬體的效能遠遠超過 VM 的模擬硬體效能，將 Docker 直接安裝在實體伺服器的 Linux 中，一定比先安裝 VM Linux，再在此 VM 中執行 Docker 要快上許多。但企業考量的重點除了效能之外更有部署的便利性、現存系統的穩定度、整個系統的高可用性等重點，筆書建議還是將 Docker 部署在 VM 中，犧牲一點點效能，換來的是更方便的部署、更穩定的環境以及更彈性的設定，這是絕對值得的。

本頁摘至最完整的 Docker 聖經一文

Docker 三個基本概念

Image (映像檔)

- 唯讀模式 (R\O)
- 用來建立 Container 的模板
- 包含軟體執行所需的所有內容：code, runtime, system tools, system libraries, settings
- 可從 Repository 下載到別人做好的 image 來直接使用

Container (容器)

- 讀寫模式 (R\W)
- 也是貨櫃的意思
- 可將軟體執行所需的所有資源打包到一個隔離的容器中
- 可解決環境不一致的問題 (例如：雲端、OS、套件版本...等環境)
- 一個 image 可以創造出多個不同的 container
- Docker 利用 Container 來執行應用

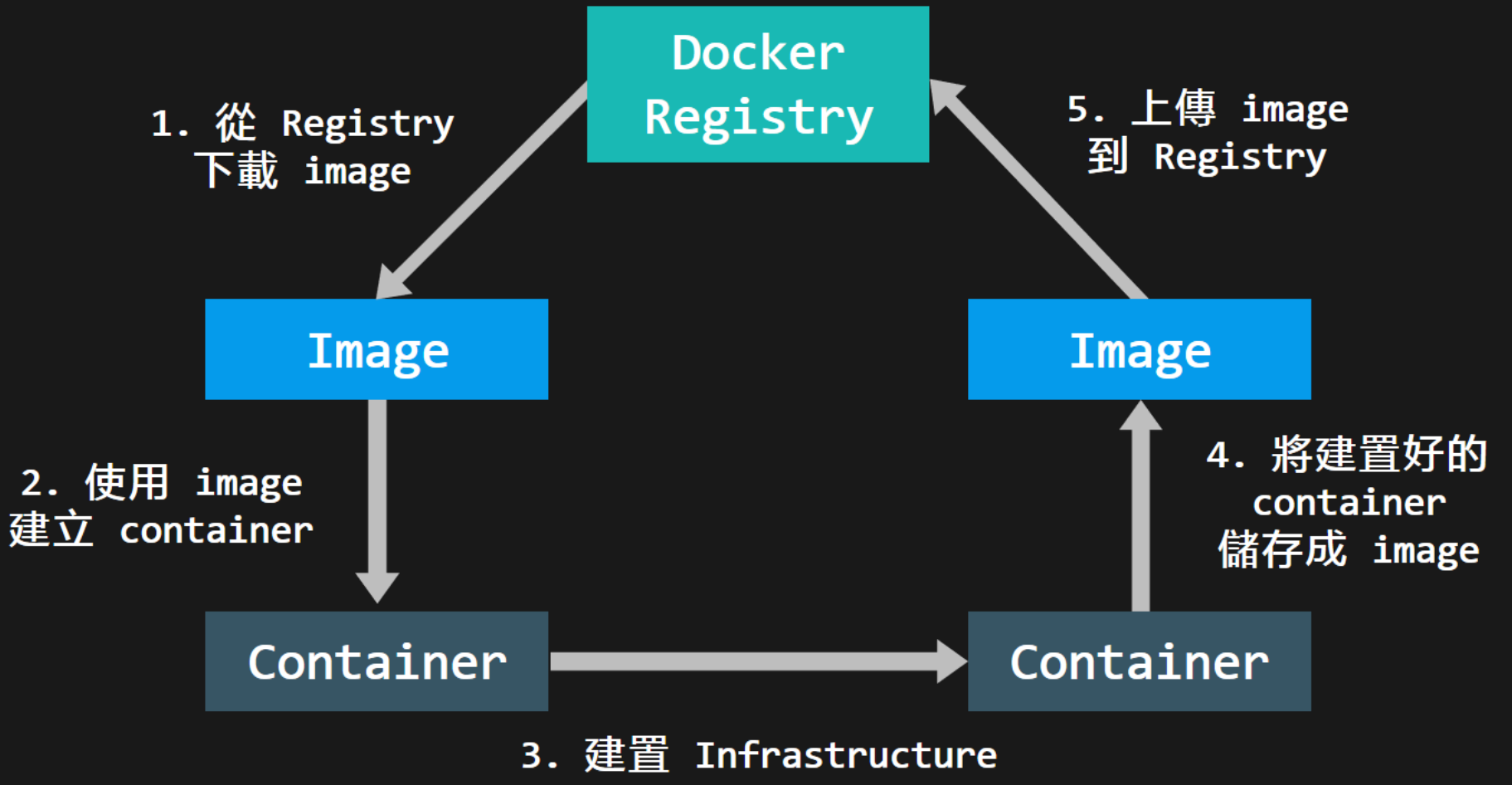
Registry (倉庫)

- 集中存放 image 的地方
- Repository (倉庫，儲存庫) vs Registry (倉庫註冊伺服器)
  - Registry：儲存和託管 image 的服務，Docker 預設的 Registry 是 Docker Hub
  - Repository：提供不同版本 (tag) 的相同應用程式或服務的集合

Registry (倉庫)

- Registry 上存放著多個 Repository，每個 Repository 中又包含了多個 image，每個 image 有多個不同的標籤 (tag)，標籤會以數字或英文的方式命名 (舉例)
- Registry 分為公開 (Public) 和私有 (Private) 兩種形式
  - 最大的公開 Registry 是 Docker Hub，存放了大量的 image 供使用者下載
  - 使用者也可以在本地端內建立一個私有 Registry

Docker 是怎麼運作的？



安裝 Docker

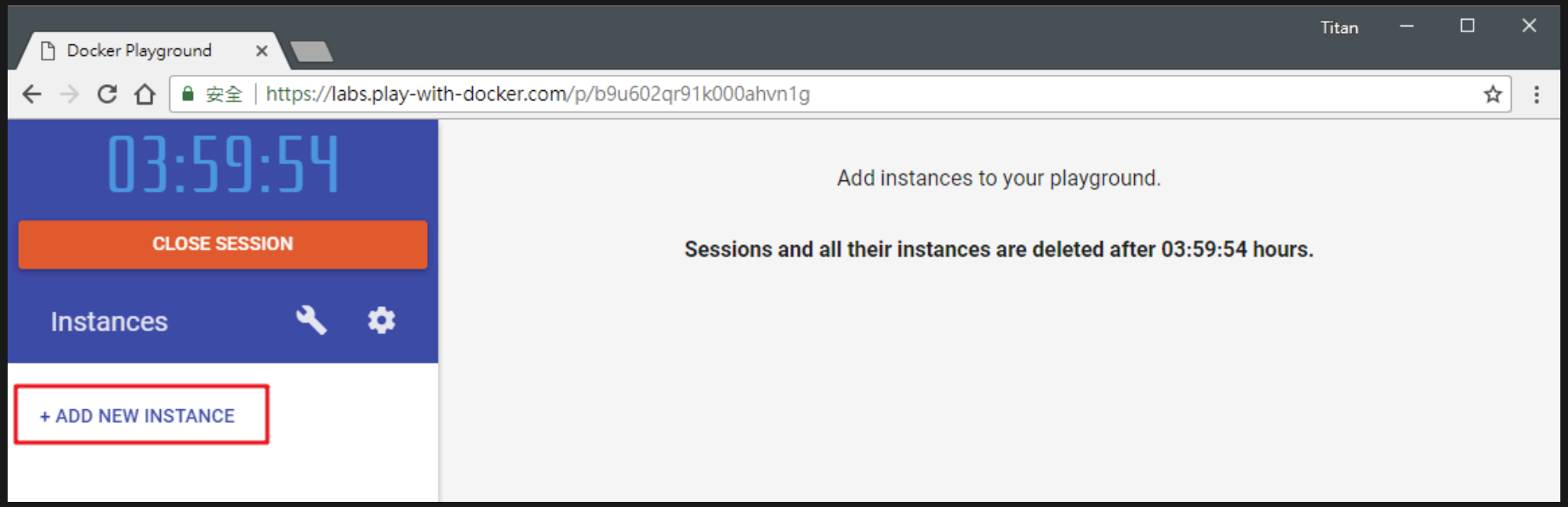
詳情請參考官方的 Install Docker 文件

```
# 檢查版本
$ docker -v
Docker version 17.12.0-ce, build c97c6d6
```

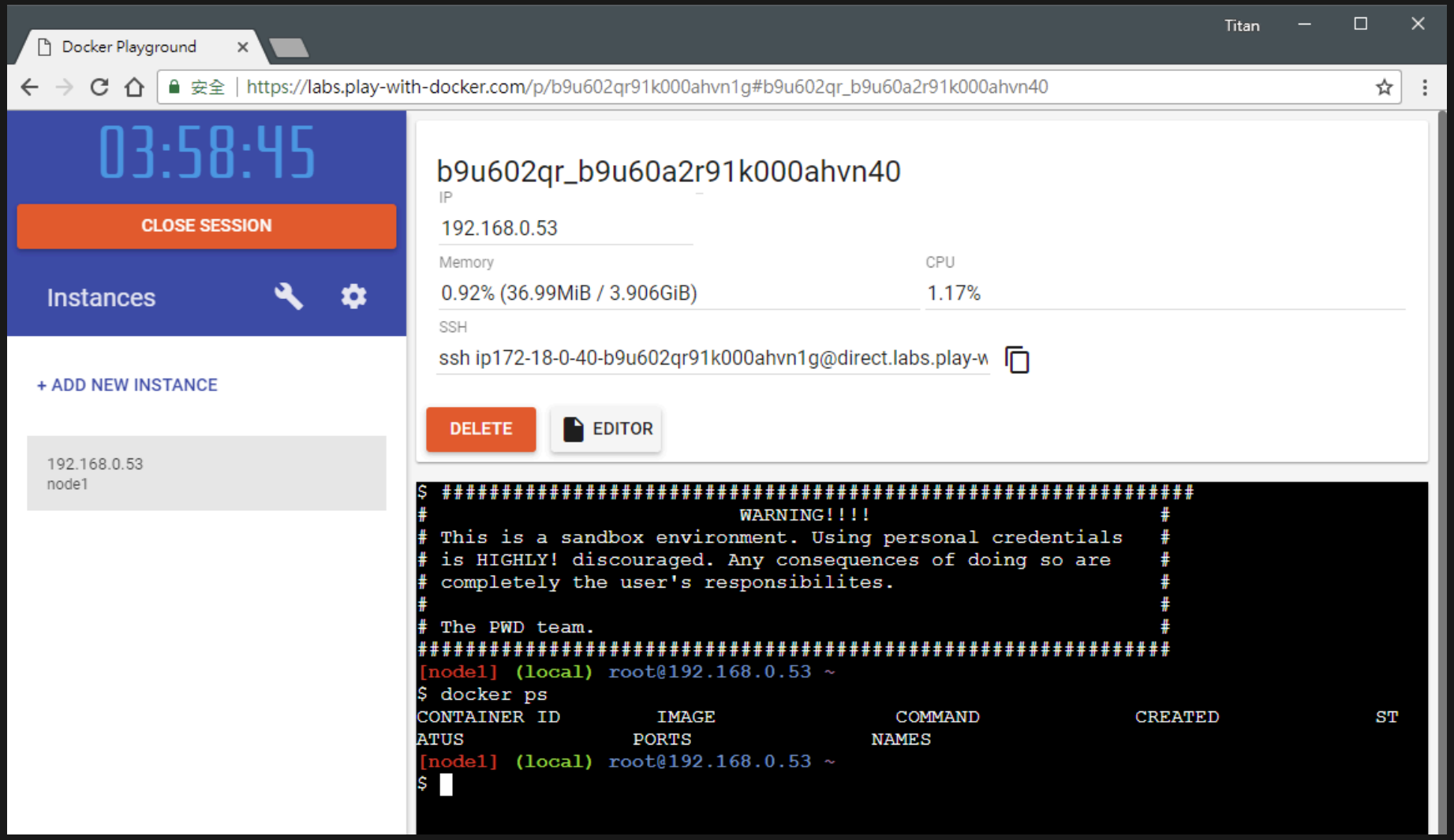
Play with Docker

如果你不想安裝 Docker，有另一種方式可以練習 Docker，那就是 Docker 官方提供的 Play with Docker (PWD)

進去後會看到一個倒數的時間，因為每次 PWD 會提供 4 小時的使用時間。接著只要按「+ ADD NEW INSTANCE」就可以建立一台新的 VM。



看到 shell 就可以開始練習 Docker



Hello Docker

執行 `docker run hello-world` 指令後，會看到下面一大串輸出：

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:66ef312bbac49c39a89aa9bcc3cb4f3c9e7de3788c9441
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be work

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the
    (amd64)
```

它會告訴你執行此指令都做了哪些事：

1. Docker client 直接跟 Docker daemon 溝通。
2. Docker daemon 從 Docker Hub pull `hello-world` 此 image 回來。
3. Docker daemon 從該 image 建立一個新的 container，該 container 執行執行檔來產生下面這堆輸出。
4. Docker daemon 將輸出串流傳輸到 Docker client，並將其發送到終端。

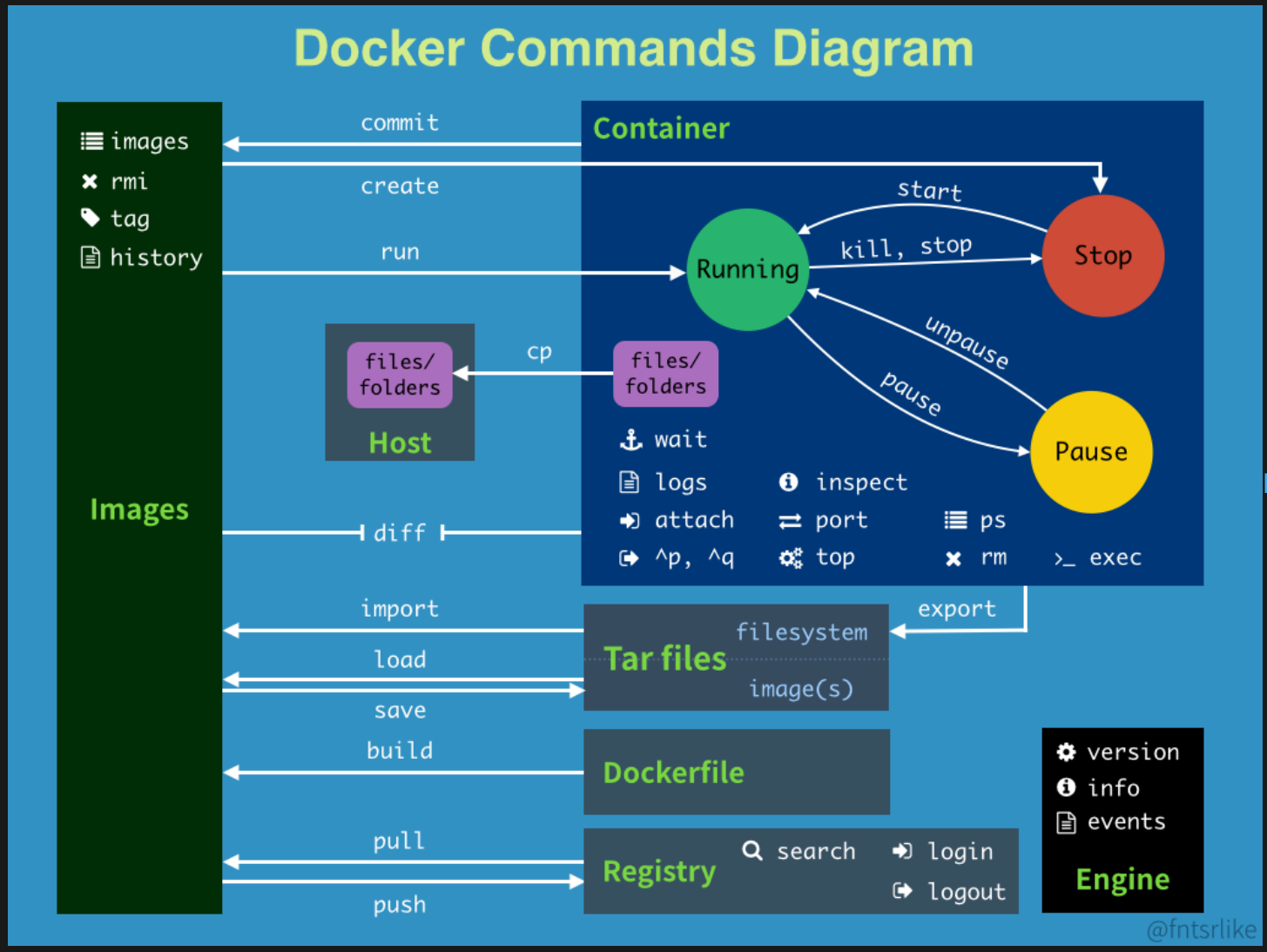
可以到 Docker Hub 看 `hello-world` 這個 image，裡面有官方提供產生該 image 的 `Dockerfile`：

```
FROM scratch
COPY hello /
CMD ["/hello"]
```

- `FROM scratch`：使用哪個 image 作為基底 (Base Image)，`scratch` 是官方提供的空白 image
- `COPY hello /`：將 `hello` 執行檔複製到 `/` 根目錄
- `CMD ["/hello"]`：執行 `hello` 此執行檔

簡單來說就是在 container 裡面執行 `hello` 這個執行檔來列印出這堆訊息。

Docker 常用指令



Ref