

# 試想....

- APP 1 (JAVA 6)
- APP 2 (JAVA 8)
- APP 3 (python 2.7 + mysql 4)
- APP 4 (python 3 + mysql 5 + PHP 5)
- APP 5 (python 2.7 + mysql 5 + node.js)
- APP 6 (python 3 + mysql 6 + PHP 7)

# 你需要的軟體

- **VMware Player**

[https://my.vmware.com/en/web/vmware/free#desktop\\_end\\_user\\_computing/vmware\\_workstation\\_player/15\\_0](https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0)

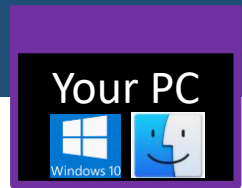
- **Ubuntu**

<https://www.ubuntu-tw.org/modules/tinyd0/>

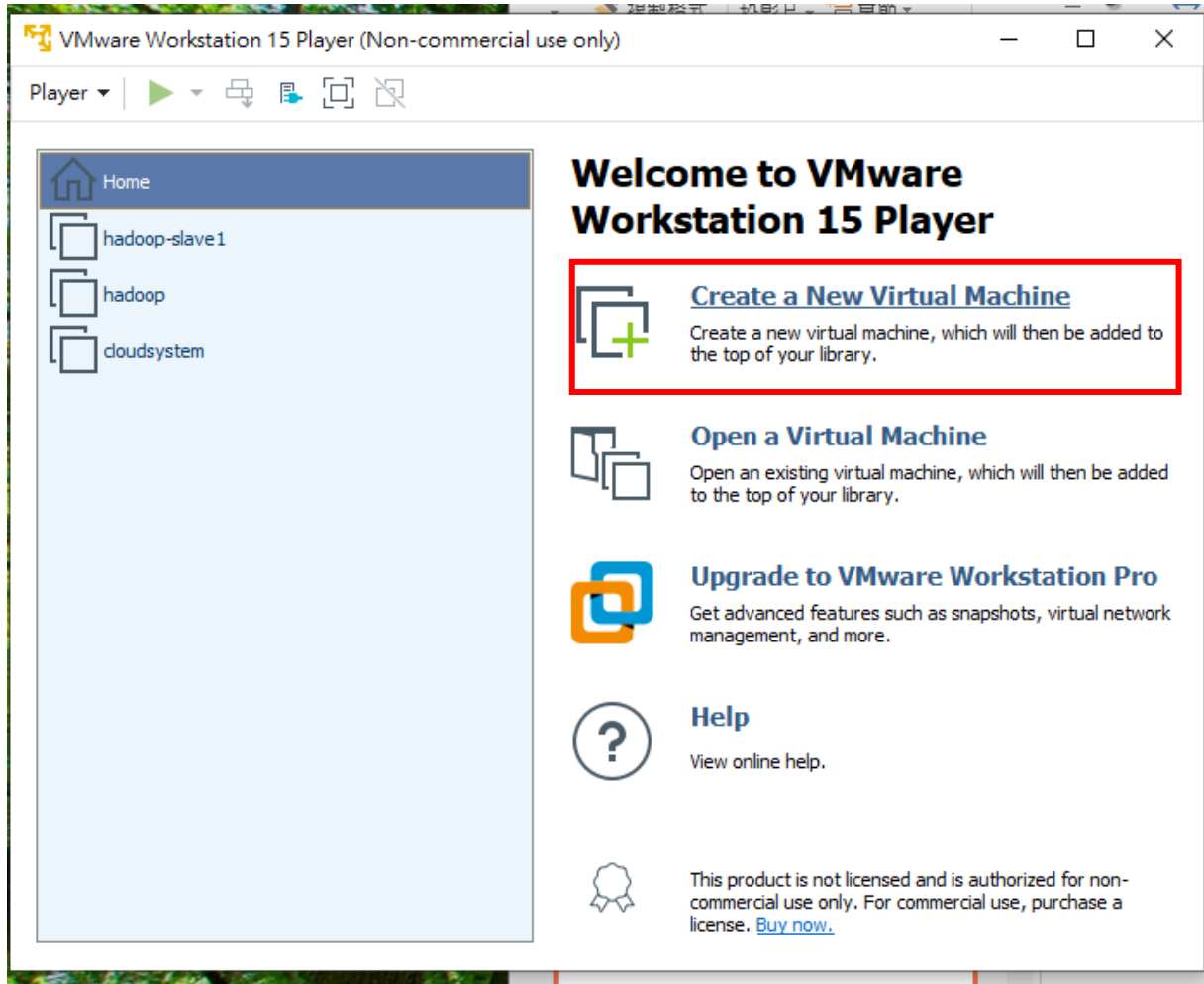
- **Pietty**

<https://sites.google.com/view/pietty-project/download?authuser=0>

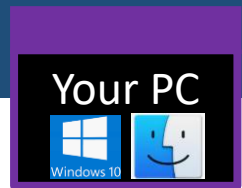
# 1. VMware Player 新建虛擬機



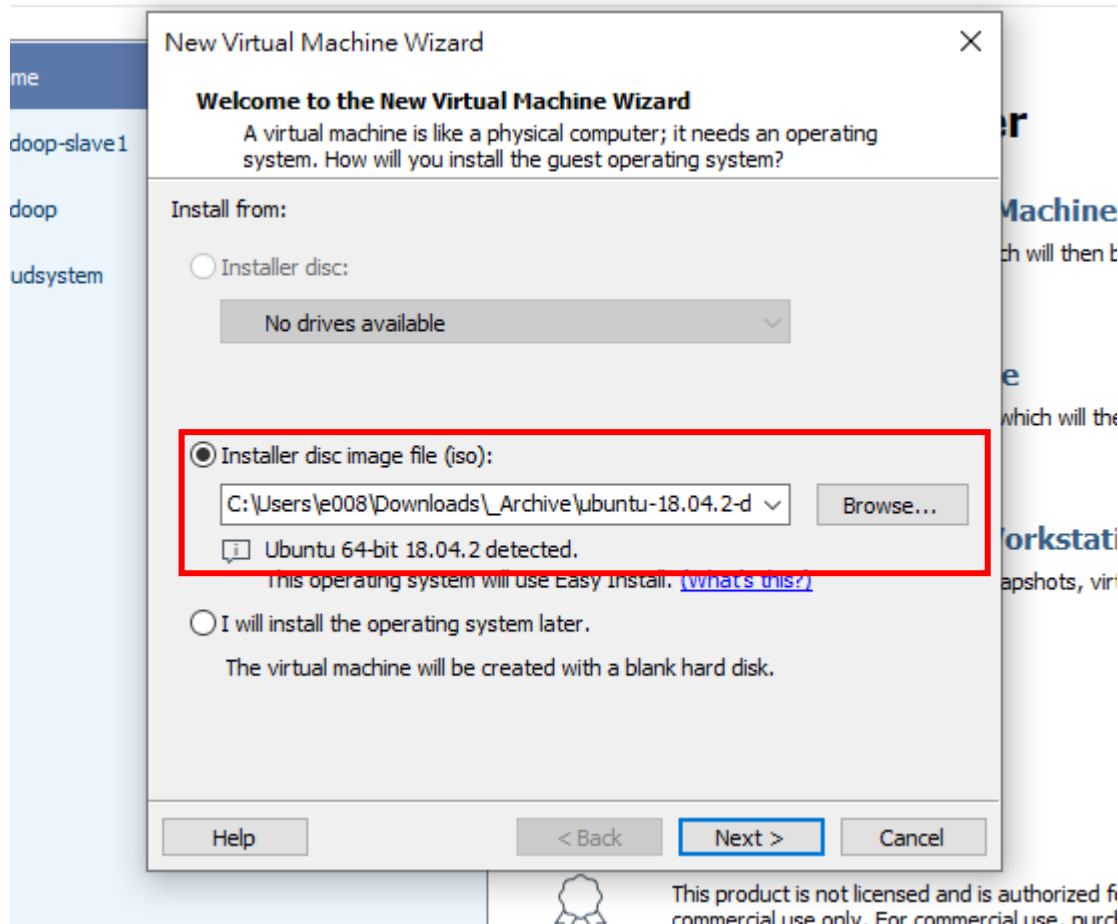
點選 Create a New Virtual Machine



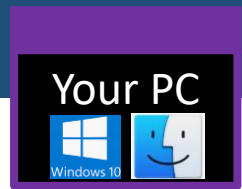
# 1. VMware Player 新建虛擬機



選擇下載好的ubuntu映像檔



# 1. VMware Player 新建虛擬機



輸入帳號資訊

A screenshot of the "New Virtual Machine Wizard" dialog box. The title bar says "New Virtual Machine Wizard" with a close button. The main heading is "Easy Install Information" with a subtitle "This is used to install Ubuntu 64-bit." Below this is a section titled "Personalize Linux" containing four input fields: "Full name:", "User name:", "Password:", and "Confirm:". At the bottom of the dialog are four buttons: "Help", "< Back", "Next >", and "Cancel".

New Virtual Machine Wizard

**Easy Install Information**  
This is used to install Ubuntu 64-bit.

Personalize Linux

Full name:

User name:

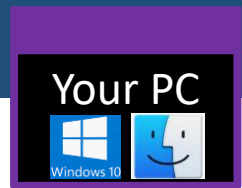
Password:

Confirm:

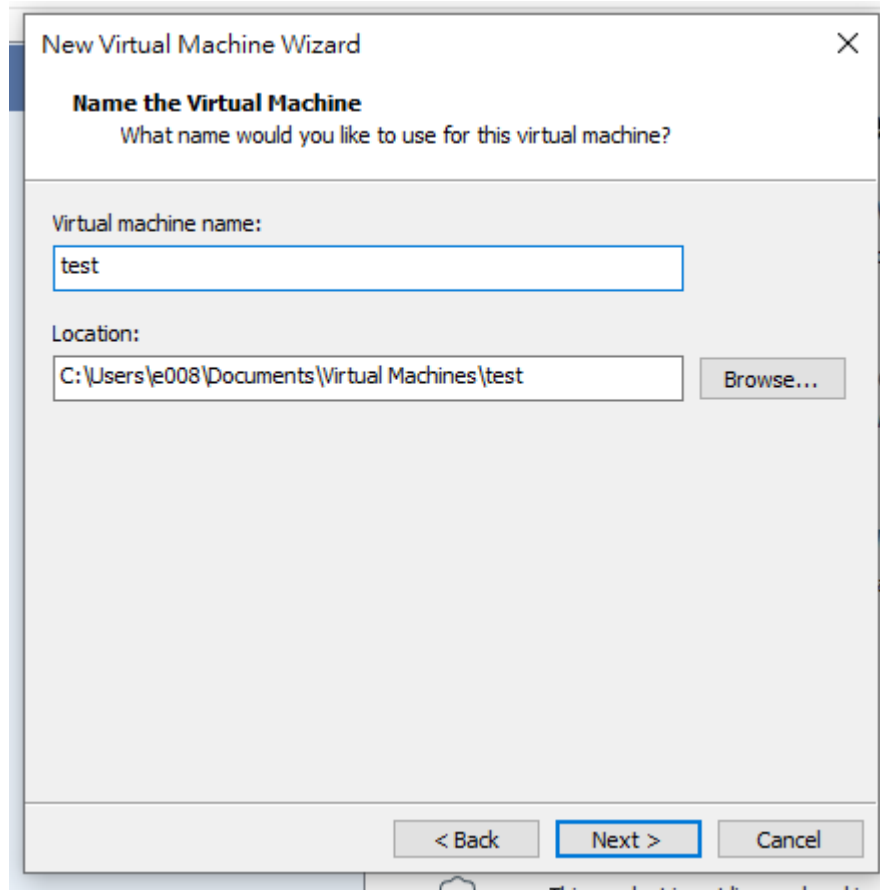
Help < Back Next > Cancel

This product is not licensed and is a

# 1. VMware Player 新建虛擬機



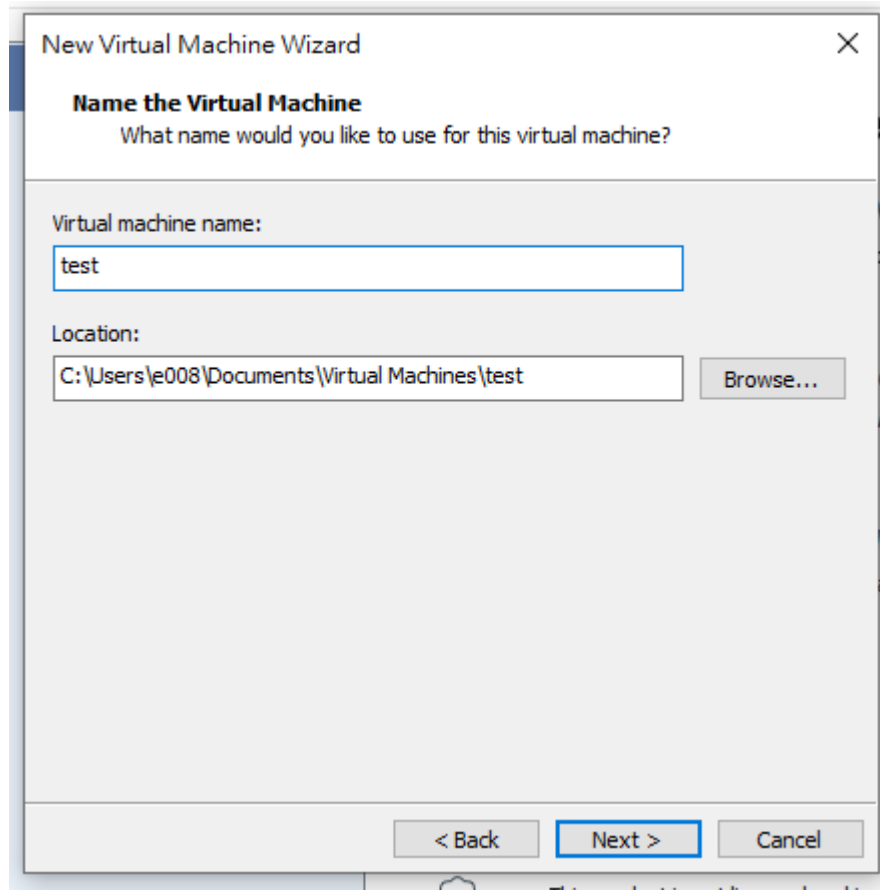
可以更改名稱



# 1. VMware Player 新建虛擬機



可以更改名稱，後面都用預設值，完成後開機等ubuntu安裝



# 安裝

```
sudo apt-get update
```

```
sudo apt-get install vim
```

```
sudo apt-get install openssh-server
```

```
sudo apt-get install net-tools
```

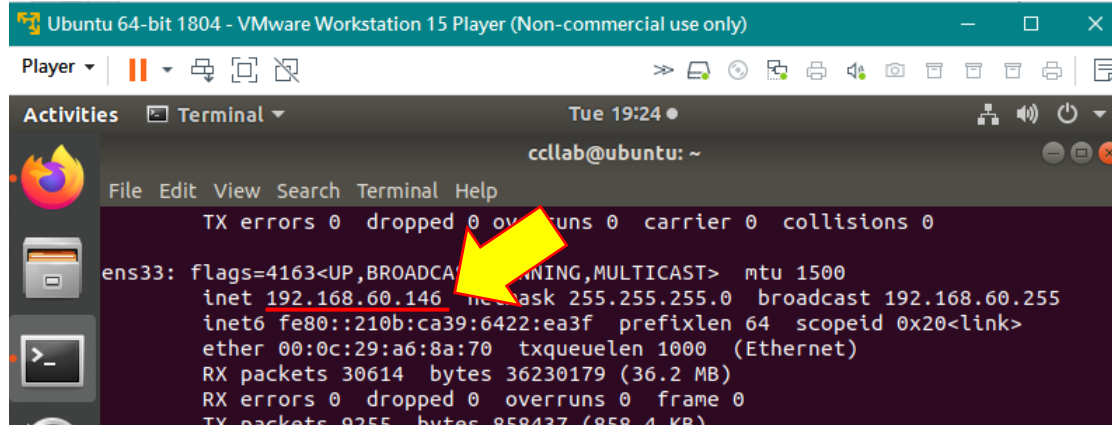


# 1. 在ubuntu上面安裝docker

Host



1. 打開 VMware 虛擬機畫面，登入進去，打開Ubuntu內的 Terminal 程式輸入 ifconfig 查看 虛擬Ubuntu 的IP



```
Ubuntu 64-bit 1804 - VMware Workstation 15 Player (Non-commercial use only)
Player | [Pause] [Full Screen] [Refresh] [Close]
Tue 19:24
ccllab@ubuntu: ~
File Edit View Search Terminal Help
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
ens33: flags=4163<UP,BROADCAST,LOOPBACK,NOARP,SMARTDRV,SYNCHRONOUS,
inet 192.168.60.146 netmask 255.255.255.0 broadcast 192.168.60.255
inet6 fe80::210b:ca39:6422:ea3f prefixlen 64 scopeid 0x20<link>
ether 00:0c:29:a6:8a:70 txqueuelen 1000 (Ethernet)
RX packets 30614 bytes 36230179 (36.2 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 9255 bytes 858437 (858.4 KB)
```

2. 打開 Pietty，登入 虛擬Ubuntu的IP，輸入

```
sudo apt-get update
```

```
sudo apt-get remove docker docker-engine docker.io
```

```
sudo apt install docker.io
```

```
sudo service docker start
```

接下來，所有的docker 相關指令  
都用root 權限操作

使用 `sudo -s` 指令切換到root權限

# 後面會用到的Docker 指令

- 查看建立的 container

`docker container ls -a`

- 刪除已經建立的 container

`docker container rm containerName`

- 重新啟動狀態為 EXIT 的 container

`docker restart containerName`

- 連入執行中的 container

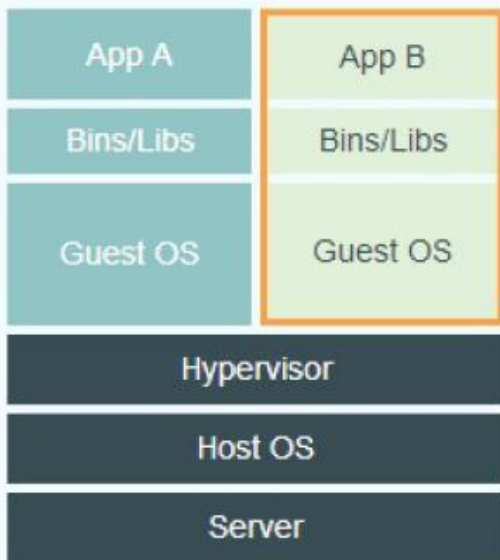
`docker attach containerName`

# Introduction to Docker

# Outline

- Introduction to Docker/Container
- Docker 基本指令
- Data Volumes
- Dockerfile

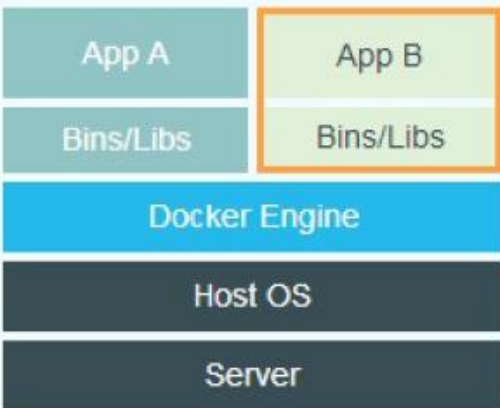
- Docker 是一個開源專案，誕生於 2013 年初，最初是 dotCloud 公司內部的一個業餘專案。它基於 Google 公司推出的 Go 語言實作。專案後來加入了 Linux 基金會，遵從了 Apache 2.0 協議，原始碼在 GitHub 上進行維護。
- Docker 專案的目標是實作輕量級的作業系統虛擬化解決方案。



## Virtual Machines

Each virtualized application includes not only the application - which may be only 10s of MB - and the necessary binaries and libraries, but also an entire guest operating system - which may weigh 10s of GB.

硬體層面實作

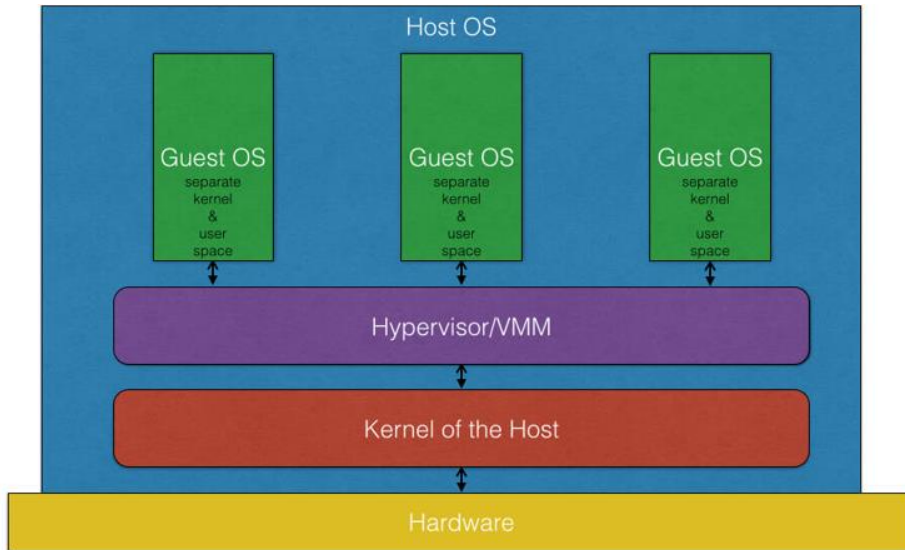


## Docker

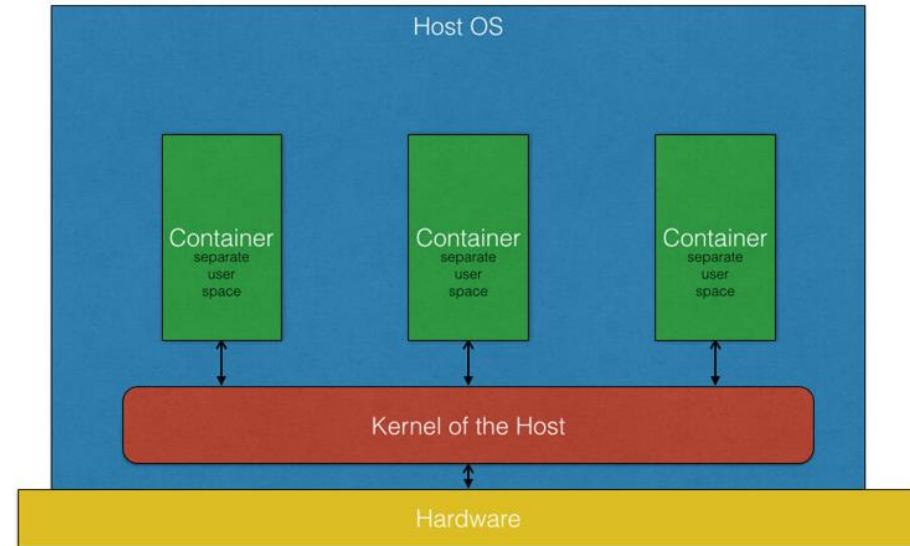
The Docker Engine container comprises just the application and its dependencies. It runs as an isolated process in userspace on the host operating system, sharing the kernel with other containers. Thus, it enjoys the resource isolation and allocation benefits of VMs but is much more portable and efficient.

作業系統層面上實作虛擬化，直接使用本地主機的作業系統

# Operating System Containers vs. Application Containers

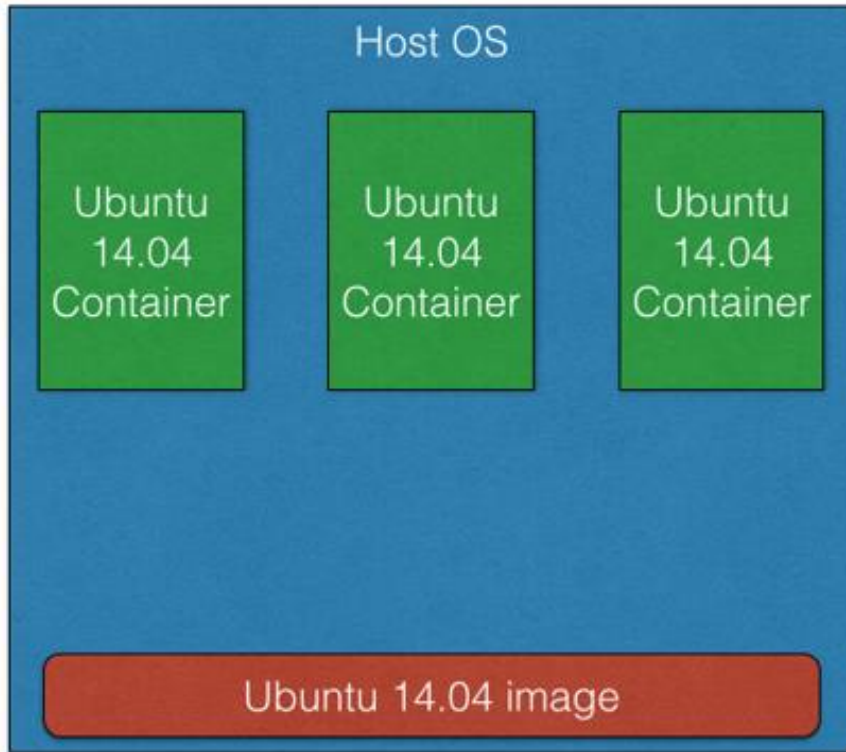


Hypervisor based Virtualization

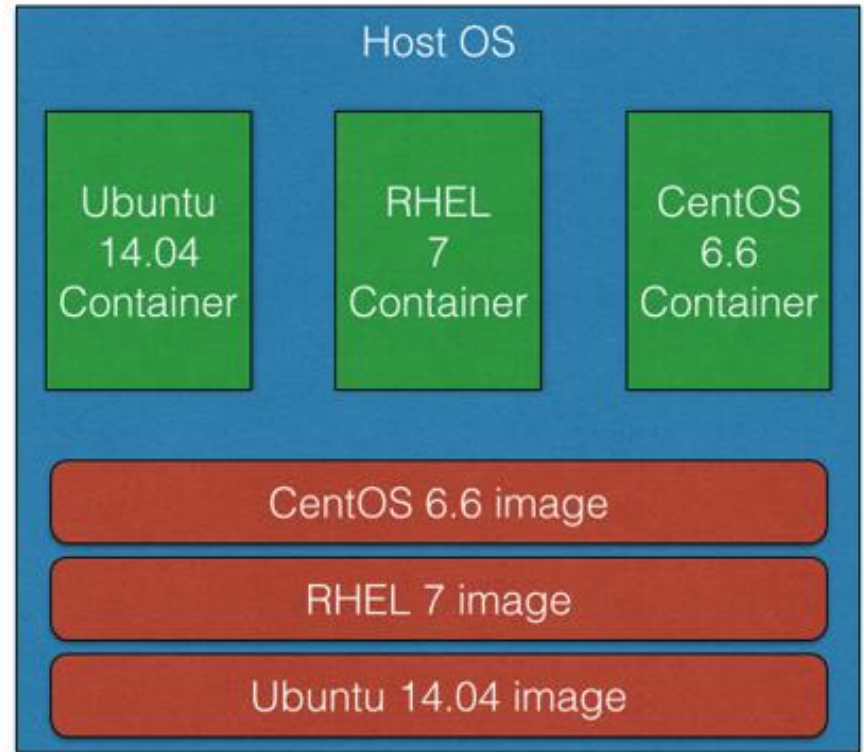


Operating System/Container Virtualization



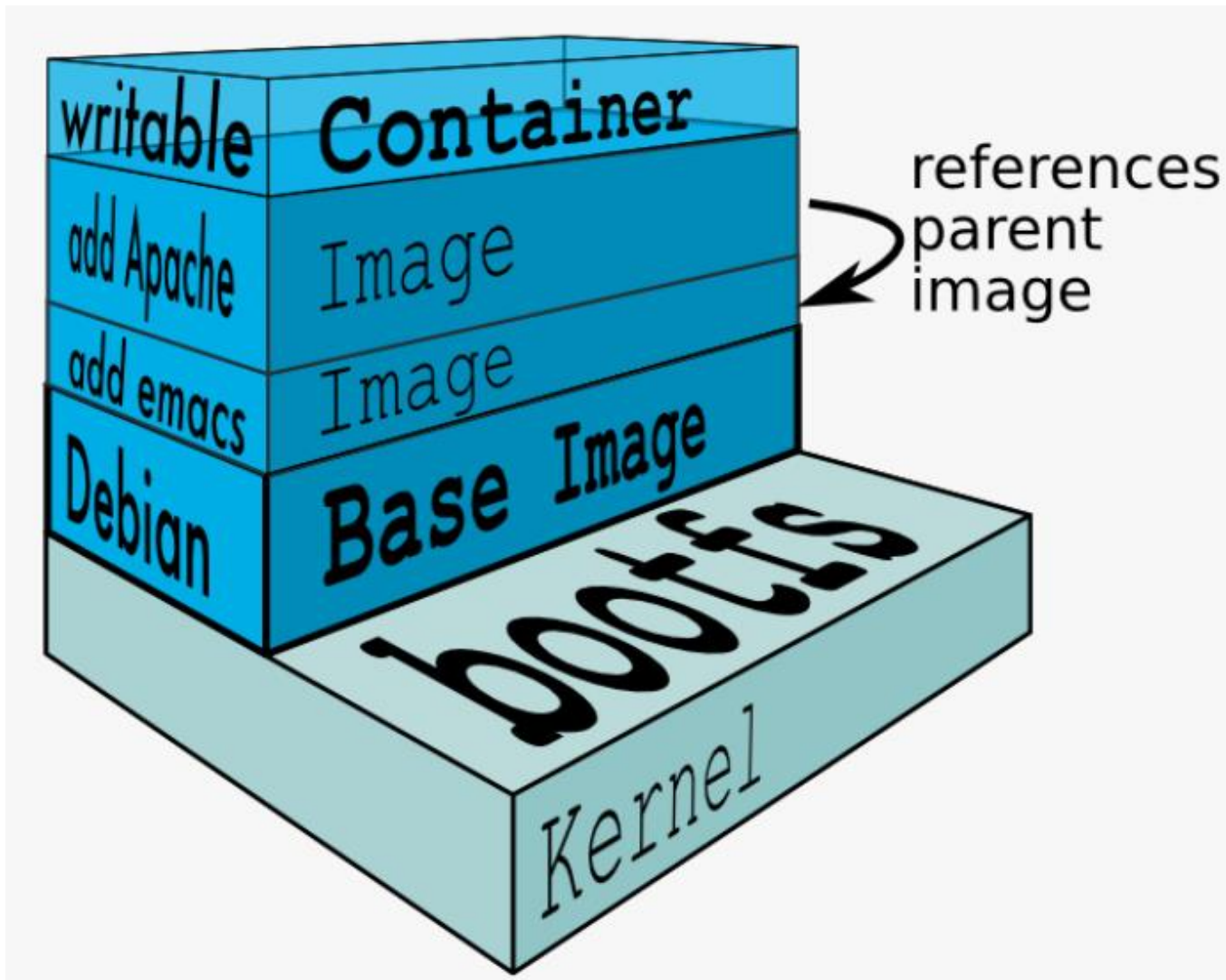


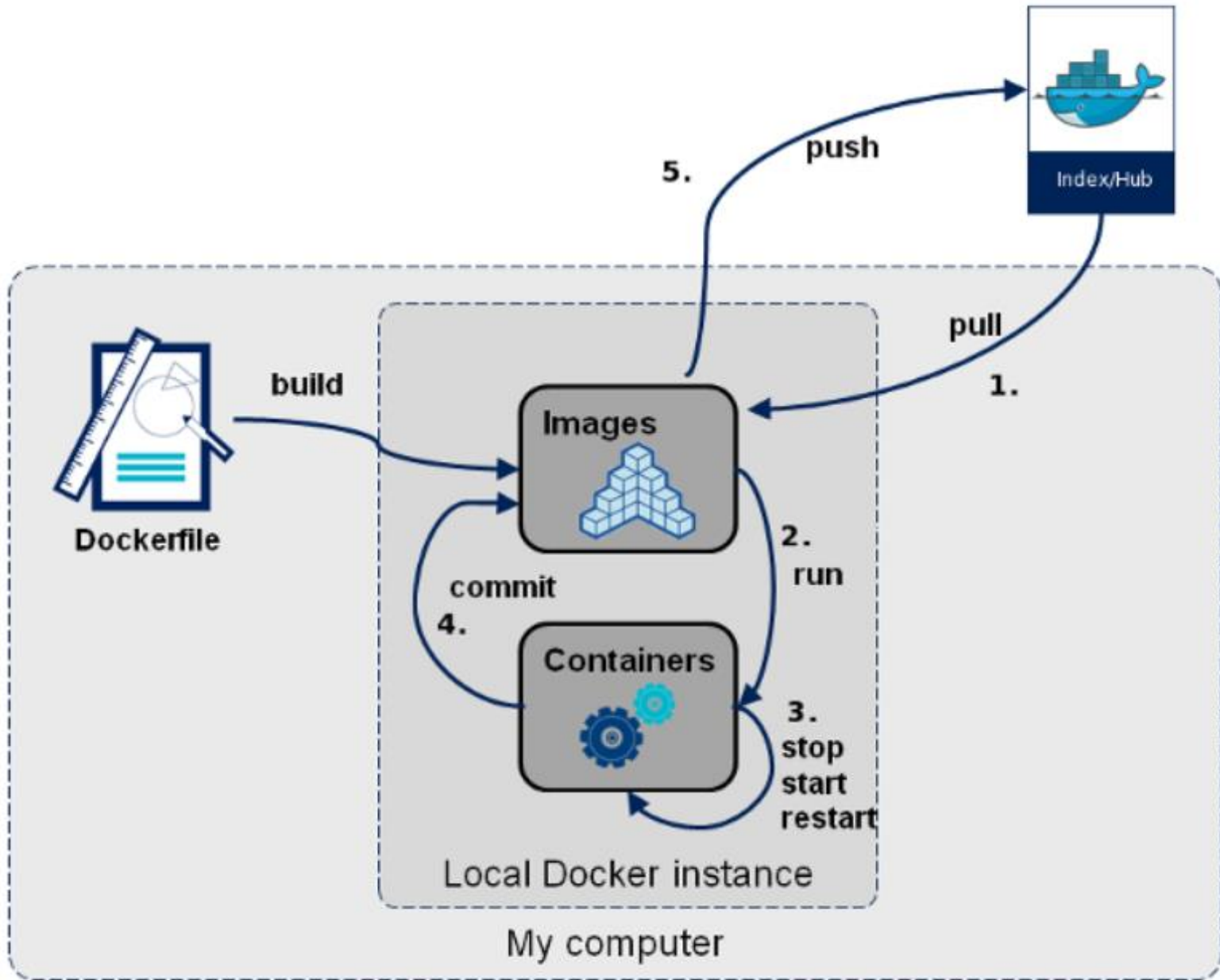
Identical OS containers

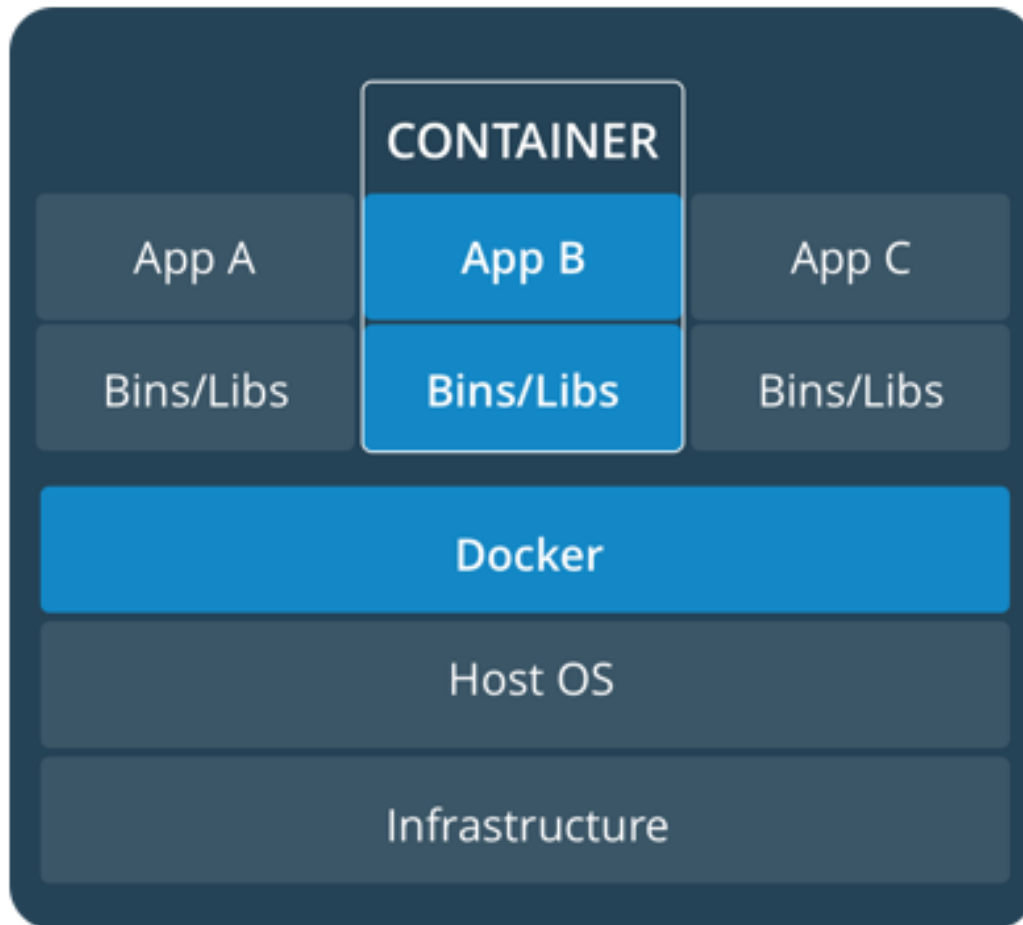


Different flavoured OS containers

# Layers of containers







- Docker 容器幾乎可以在任意的平台上執行，包括實體機器、虛擬機、公有雲、私有雲、個人電腦、伺服器
- 可以把一個應用程式從一個平台直接遷移到另外一個

特性	容器	虛擬機
啟動	秒級	分鐘級
硬碟容量	一般為 MB	一般為 GB
效能	接近原生	比較慢
系統支援量	單機支援上千個容器	一般幾十個

# 三個基本元素

- 映像檔 ( Image )
  - 唯讀的模板
  - 映像檔可以用來建立 Docker 容器

ex. 一個映像檔可以包含一個完整的 ubuntu 作業系統環境，裡面僅安裝了 Apache

- 容器 ( Container )
  - 從映像檔建立的執行實例
  - 每個容器都相互隔離
  - 映像檔是唯讀的，容器啟動的時候建立一層可寫層作為最上層
- 倉庫 ( Repository )
  - 集中存放映像檔檔案的場所
  - <https://hub.docker.com/>

功能表 | aws | 聯絡銷售人員 | 產品 | 解決方案 | 定價 | 入門 | 文件 | 更多 | 中文(繁體) | 我的帳戶 | 註冊

# 什麼是 Docker?

使用 Docker 可快速建立、測試和部署應用程式

開始使用 Docker

Docker 是一種軟體平台，可讓您快速地建立、測試和部署應用程式。Docker 將軟體封裝到名為容器的標準化單位，其中包含程式庫、系統工具、程式碼和執行時間等執行軟體所需的所有項目。使用 Docker，您可以將應用程式快速地部署到各種環境並加以擴展，而且知道程式碼可以執行。



在 AWS 上執行 Docker 可讓開發人員和管理員以高度可靠且低成本的方式建立、發佈和執行各種規模的分散式應用程式。AWS 支援兩種 Docker 授權模型：開放原始碼 Docker Community Edition (CE) 和訂閱型 Docker Enterprise Edition (EE)。

## Docker 的運作方式

Docker 透過提供執行程式碼的標準方法進行運作。Docker 是容器的作業系統。與**虛擬機器**虛擬化(免除直接管理的需要)伺服器硬體的方法相似，容器可**虛擬化**作業系統。安裝在每部伺服器上，並提供簡單的命令讓您使用以







# Azure 上的 Docker

保護及管理雲端中的企業容器應用程式

開始使用適用於 Azure 的 Docker >



## 現代化您的應用程式和基礎結構

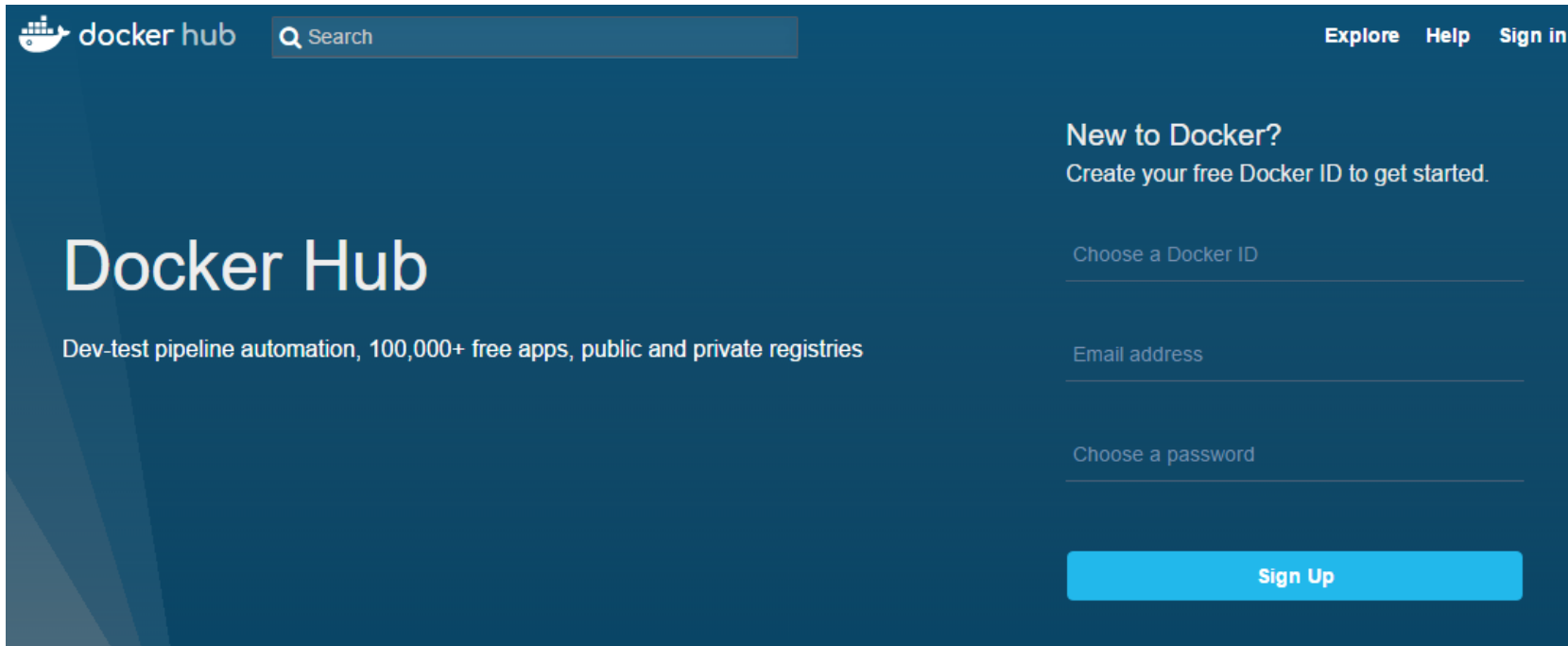
輕鬆快速地將您的應用程式移轉到 Azure，以提高安全性及現代化應用程式服務。在 Azure 上部署 Docker，您就能以企業級安全性、支援和規模，來執行現代化和傳統 Linux 或 Windows 應用程式。

## 取得整合式管理、安全性並節省成本

針對 Docker 容器中的傳統和雲端應用程式，採用統一的作業模型和安全的供應鏈，以降低作業成本並提高效率。







建立自己的映像檔之後

使用 `push` 命令將它上傳到公有或者私有倉庫

另外一台機器上只需要從倉庫上 `pull` 下來就可以安裝一樣的 image

# Part 1. docker 基本指令

啟動 docker service

~~sudo /etc/init.d/docker start~~

sudo service docker start

sudo -s

docker ps -a

# docker search: 搜尋映像檔

```
root@YARNMaster:~# docker search ubuntu
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
ubuntu	Ubuntu is a Debian-based Linux operating s...	6090	[OK]	
rastasheep/ubuntu-sshd	Dockerized SSH service, built on top of of...	89		[OK]
ubuntu-upstart	Upstart is an event-based replacement for ...	73	[OK]	
ubuntu-debootstrap	debootstrap --variant=minbase --components...	30	[OK]	
torusware/speedus-ubuntu	Always updated official Ubuntu docker imag...	28		[OK]
nuagebec/ubuntu	Simple always updated Ubuntu docker images...	21		[OK]
nickistre/ubuntu-lamp	LAMP server on Ubuntu	18		[OK]
solita/ubuntu-systemd	Ubuntu + systemd	8		[OK]
nimmis/ubuntu	This is a docker images different LTS vers...	7		[OK]
darksheer/ubuntu	Base Ubuntu Image -- Updated hourly	3		[OK]
vcatechnology/ubuntu	A Ubuntu image that is updated daily	1		[OK]

automated build



```
root@YARNMaster:~# docker search CentOS
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
centos	The official build of CentOS.	3375	[OK]	
jdeathe/centos-ssh	CentOS-6 6.9 x86_64 / CentOS-7 7.3.1611 x8...	70		[OK]
nimmis/java-centos	This is docker images of CentOS 7 with dif...	26		[OK]
gluster/gluster-centos	Official GlusterFS Image [ CentOS-7 + Glu...	19		[OK]
million12/centos-supervisor	Base CentOS-7 with supervisord launcher, h...	16		[OK]

# docker pull: 從倉庫取得映像檔

sudo docker pull [registry.hub.docker.com/ubuntu:12.04](https://registry.hub.docker.com/ubuntu:12.04)

sudo docker pull ubuntu:12.04

可省略，或改成其他Docker hub網址

```
root@YARNMaster:~# sudo docker pull ubuntu:12.04
12.04: Pulling from library/ubuntu
8341f117fa20: Downloading 22.86 MB/39.1 MB
47c192d937be: Download complete
20bcec4513dd: Download complete
8b09610eae5c: Download complete
ddf16cf6c2ca: Download complete
d19993b6c2d5: Download complete
```

```
root@YARNMaster:~# sudo docker pull ubuntu:12.04
12.04: Pulling from library/ubuntu
8341f117fa20: Pull complete
47c192d937be: Pull complete
20bcec4513dd: Pull complete
8b09610eae5c: Pull complete
ddf16cf6c2ca: Pull complete
d19993b6c2d5: Pull complete
Digest: sha256:5ee5ef3baf8f551c7e430e196100fd9c7265129bb50cd9951b3d75931cb415fb
Status: Downloaded newer image for ubuntu:12.04
```

# docker create: 從映像檔(唯讀) , 建立容器(多了可寫入層)

```
docker create ubuntu:latest 指令
```

```
root@YARNMaster:~# docker create ubuntu:latest /bin/bash
fc4c1da5e4ed9f416995ffdf90f9d75f33026a32444b9e3b0f336e24633ec20d
```

產生一個執行 echo 'Hello World' 的 container , 但還不執行

```
docker create ubuntu:latest /bin/echo 'Hello World'
```

```
root@YARNMaster:~# docker create ubuntu:latest /bin/echo 'Hello World'
e19a02d1a4bece2b0a6c1924d95d9e7137b9c34f31e336670f3e507f2cca126b
```

```
root@YARNMaster:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
e19a02d1a4be	ubuntu:latest	"/bin/echo 'Hello Wor"	7 seconds ago	Created

# docker ps: 查看(執行中的)容器

-a : 顯示終止的容器

```
root@YARNMaster:~# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS          PORTS          NAMES
951b3a3f57e1  ubuntu:12.04  "/bin/bash"            About an hour ago  Exited (0) About an hour ago          desperate_turing
```

docker start <容器名稱/容器id>: 啟動容器

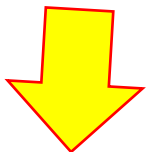
```
root@YARNMaster:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED
e19a02d1a4be       ubuntu:latest      "/bin/echo 'Hello Wor" 7 seconds ago
```

docker start e19a02d1a4be

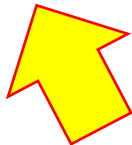
```
root@YARNMaster:~# docker start e19a02d1a4be
e19a02d1a4be
```

# 建立一個容器 並 執行 bash

sudo docker run -t -i --name 名子 ubuntu:12.04 /bin/bash



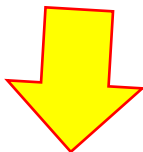
```
root@YARNMaster:~# sudo docker run -t -i ubuntu:12.04 /bin/bash
root@951b3a3f57e1:/#
```



-t pseudo-tty  
-i interactive

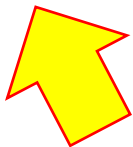
在Bash shell 進行輸入/輸出

-p 開啟特定 port



```
root@951b3a3f57e1:/home# exit
exit
root@YARNMaster:~#
```

離開容器





docker run = docker create + docker start

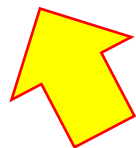
```
root@YARNMaster:~# docker run -t -i ubuntu:latest /bin/echo 'hello world'  
hello world
```

# docker images: 列出本機映像檔

ID 號 (唯一)      建立時間      映像檔大小

```
root@YARNMaster:~# docker images
REPOSITORY      TAG          IMAGE ID      CREATED      VIRTUAL SIZE
ubuntu          12.04       d19993b6c2d5 7 weeks ago 103.6 MB
```

```
$ sudo docker images
REPOSITORY      TAG          IMAGE ID      CREATED      VIRTUAL SIZE
ubuntu          12.04       74fe38d11401 4 weeks ago 209.6 MB
ubuntu          precise     74fe38d11401 4 weeks ago 209.6 MB
ubuntu          14.04       99ec81b80c55 4 weeks ago 266 MB
ubuntu          latest      99ec81b80c55 4 weeks ago 266 MB
ubuntu          trusty      99ec81b80c55 4 weeks ago 266 MB
```



ID 號一樣，代表都從同一個image下來的

TAG 用來標記來自同一個倉庫的不同映像檔。例如 ubuntu 倉庫中有多個映像檔，通過 TAG 來區分版本，例如 10.04、12.04、12.10、13.04、14.04 等。

```
sudo docker run -t -i ubuntu:14.04 /bin/bash
```

如果沒有指定 TAG，預設使用 latest

# docker attach: 連接容器

```
root@YARNMaster:~# docker run -t -i ubuntu:latest /bin/echo "Hello world"
Hello world
root@YARNMaster:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
dffdbc217d78	ubuntu:latest	"/bin/echo 'Hello wor"	8 seconds ago	Exited (0) 7 seconds ago

```
root@YARNMaster:~# docker attach dffdbc217d78
You cannot attach to a stopped container, start it first
```

容器必須要執行中，才能 attach

# exit 或 Ctrl + D 退出容器

```
root@YARNMaster:~# docker run -t -i ubuntu:latest /bin/bash
root@1fb84c327cef:/# exit
```

容器退出(exit)後，就自動停止執行

# docker run -d 可在背景執行

```
root@YARNMaster:~# docker run -t -i -d ubuntu:latest /bin/bash
f32feca10cc55e64778bdd99fe0c820a56d4425e2932de62bf18e754de02240a
root@YARNMaster:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              UP
f32feca10cc5        ubuntu:latest      "/bin/bash"        6 seconds ago      Up 6 seconds
root@YARNMaster:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              UP
f32feca10cc5        ubuntu:latest      "/bin/bash"        12 seconds ago     Up 12 seconds
root@YARNMaster:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              UP
f32feca10cc5        ubuntu:latest      "/bin/bash"        16 seconds ago     Up 16 seconds
```

# 不停止容器下退出

依次按: Ctrl+P Ctrl+Q

# docker logs: 查看容器輸出

```
root@YARNMaster:~# docker create ubuntu:latest /bin/echo 'Hello World'  
125dd0519103515ebda5f8ffd913596489781202ac043f4222bb32b0ff7f0160
```

```
root@YARNMaster:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
125dd0519103	ubuntu:latest	"/bin/echo 'Hello Wor"	6 seconds ago	Created

```
root@YARNMaster:~# docker start 125dd0519103  
125dd0519103
```

```
root@YARNMaster:~# docker logs 125dd0519103  
Hello World
```

docker kill:強迫關閉容器

# Exercise 1: 使用容器背景執行下列script 並輸出結果

```
for (( c=1; c<=100; c++ ))
do
    echo "Hi $c."
    sleep 1
done
```

`docker run -t -i ubuntu /bin/echo 'Hello world'`

`/bin/bash -c 'for (( c=1; c<=100; c++ )); do echo "Hi $c."; sleep 1; done'`

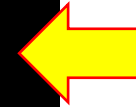
```
root@YARNMaster:~#   
Hi 1.  
Hi 2.  
Hi 3.  
Hi 4.  
Hi 5.  
Hi 6.  
Hi 7.  
Hi 8.  
Hi 9.  
Hi 10.  
Hi 11.  
Hi 12.
```



# docker export: 匯出容器

```
root@YARNMaster:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS             
73f6b57b81d7       ubuntu:latest      "/bin/bash -c 'for (" 8 minutes ago      Exited (0) 6 minutes ago
root@YARNMaster:~# docker export 73f6b57b81d7 > myContainter.tar
```

```
root@YARNMaster:~# ls -lh
total 300M
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Desktop
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Documents
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Downloads
-rw-r--r-- 1 ubuntu ubuntu 8.8K Mar 29 05:42 examples.desktop
-rw-rw-r-- 1 ubuntu ubuntu 203M Jan 25 2016 hadoop-2.7.2.tar.gz
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Music
-rw-r--r-- 1 root root 97M Jun 5 11:00 myContainter.tar
drwxrwxr-x 3 ubuntu ubuntu 4.0K Apr 13 02:10 nn
drwxrwxr-x 3 ubuntu ubuntu 4.0K Apr 12 05:48 nn.old
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Pictures
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Public
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Templates
drwxr-xr-x 2 ubuntu ubuntu 4.0K Mar 29 05:59 Videos
```



# 倉庫 ( Repository )

- 架設私有倉庫
    - image不需保存在公有雲
    - 分享image給其他電腦
  - 從 docker hub 下載並建立registry私有倉庫 (也是一個docker image)
- docker run -d -p 5000:5000 registry

```
root@YARNMaster:~# docker run -d -p 5000:5000 registry
Unable to find image 'registry:latest' locally
latest: Pulling from library/registry

fe3ee2fb752c: Pull complete
aabcd9ac8e7a: Pull complete
67a6196d0825: Pull complete
2dc2ec96fd0d: Pull complete
e85e8e163f99: Pull complete
e96e889b5692: Pull complete
d79f67d0c556: Pull complete
9b3ffb299729: Pull complete
5e9cdf0621bd: Pull complete
f5297d6e649d: Pull complete
Digest: sha256:1fd7f060074f8279ad001d5b24b612167a89c5eab42998eac7490d7b4ab3418a
Status: Downloaded newer image for registry:latest
d71d145b7499e1113a530ad72e70a715cacc7463422a0950eb1b51b458b5f527
root@YARNMaster:~# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
d71d145b7499	registry	"/entrypoint.sh /etc/"	About a minute ago	Up About a minute

# 把 image push 到私有倉庫

```
root@YARNMaster:/opt/data/registry# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
971f60a18127       registry           "/entrypoint.sh /etc/" 17 minutes ago     Up 17 minutes
73f6b57b81d7       ubuntu:latest     "/bin/bash -c 'for (" 10 hours ago       Exited (0) 10 hours ago
```

```
docker tag ubuntu:latest localhost:5000/hello:0.1
```

```
docker push localhost:5000/hello:0.1
```

```
root@YARNMaster:/opt/data/registry# docker tag ubuntu:latest localhost:5000/hello:0.1
root@YARNMaster:/opt/data/registry# docker push localhost:5000/hello:0.1
The push refers to a repository [localhost:5000/hello] (len: 1)
522985afde44: Pushed
ca30f2d5c4b9: Pushed
3033132e39d2: Pushed
9fb44af9e6e7: Pushed
b4fec67fb7bc: Pushed
c600c2e44538: Pushing [=====>] 52.52 MB/118.3 MB
```

# docker images 查看有多少images

```
root@YARNMaster:/opt/data/registry# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu              latest             522985afde44       3 days ago        118.3 MB
localhost:5000/hello 0.1                522985afde44       3 days ago        118.3 MB
registry            latest             f5297d6e649d       3 weeks ago       33.2 MB
ubuntu              12.04             d19993b6c2d5       7 weeks ago       103.6 MB
```

## 從私有倉庫的image建立 容器

```
docker run -i -t localhost:5000/hello:0.1
```

```
docker run -i -t localhost:5000/hello:0.1 /bin/echo 'Hello'
```

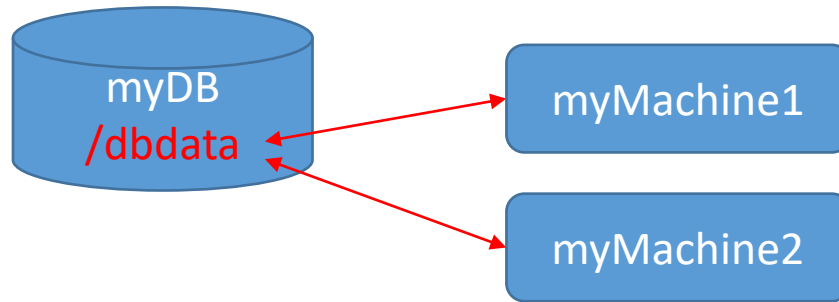
```
root@YARNMaster:/opt/data/registry# docker run -i -t localhost:5000/hello:0.1 /bin/echo 'Hello'
Hello
```

```
root@YARNMaster:/opt/data/registry# docker ps -a
CONTAINER ID   IMAGE                 COMMAND              CREATED        STATUS                    PORTS
ea8555c5832d   localhost:5000/hello:0.1  "/bin/echo Hello"   24 seconds ago  Exited (0) 23 seconds ago
971f60a18127   registry              "/entrypoint.sh /etc/" 33 minutes ago  Up 33 minutes
```

## Part 2. 共用空間管理

- 建立容器之間可以存取的共用空間容器 (Data Volume)
- 容器產出的資料存放在 **Data Volume 容器**
- 其他容器可以讀寫

# 實作:



- 建立三個容器 (Ubuntu)
- myDB 容器 提供一個 data volume (/dbdata)
- myMachine1、myMachine2 共同掛載
- 由myMachine1 寫入 檔案 test.txt
- myMachine2 讀出檔案

# step 1. 產生 myDB & /dbdata

```
docker run -i -t -v /dbdata --name myDB ubuntu:latest
```

跳出 (Ctrl+P Ctrl+Q)

```
root@YARNMaster:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
7bbc3a177bd7       ubuntu:latest      "/bin/bash"        4 minutes ago      Up About a minute
```

```
root@YARNMaster:~# docker attach myDB
root@eac07053a1ac:/#
root@eac07053a1ac:/# ls
bin boot dbdata dev etc home lib lib64 media mnt opt proc root run sbin srv sys t
```

```
docker attach myDB
```

寫入 檔案 test.txt

內容: hello world

```
echo 'hello world' > /dbdata/test.txt
```

## step2. 掛載 myDB 內的 /dbdata

```
docker run -it --volumes-from myDB --name myMachine1 ubuntu
```

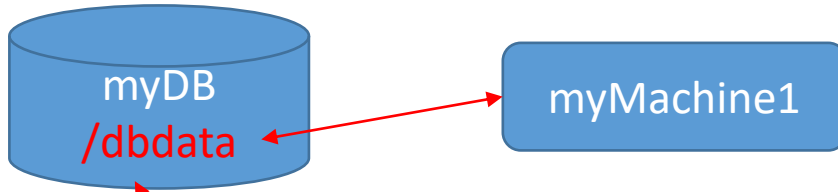
```
docker run -it --volumes-from myDB --name myMachine2 ubuntu
```

```
docker run -it --volumes-from myMachine1 --name myMachine2 ubuntu
```

==> myMachine2 也可以透過掛載 myMachine1 來共用空間

myDB 、 myMachine1 都 不需在執行狀態





```
cd /dbdata  
for (( c=1; c<=100; c++ )); do echo "Hi $c." > output.txt ; sleep 1; done
```

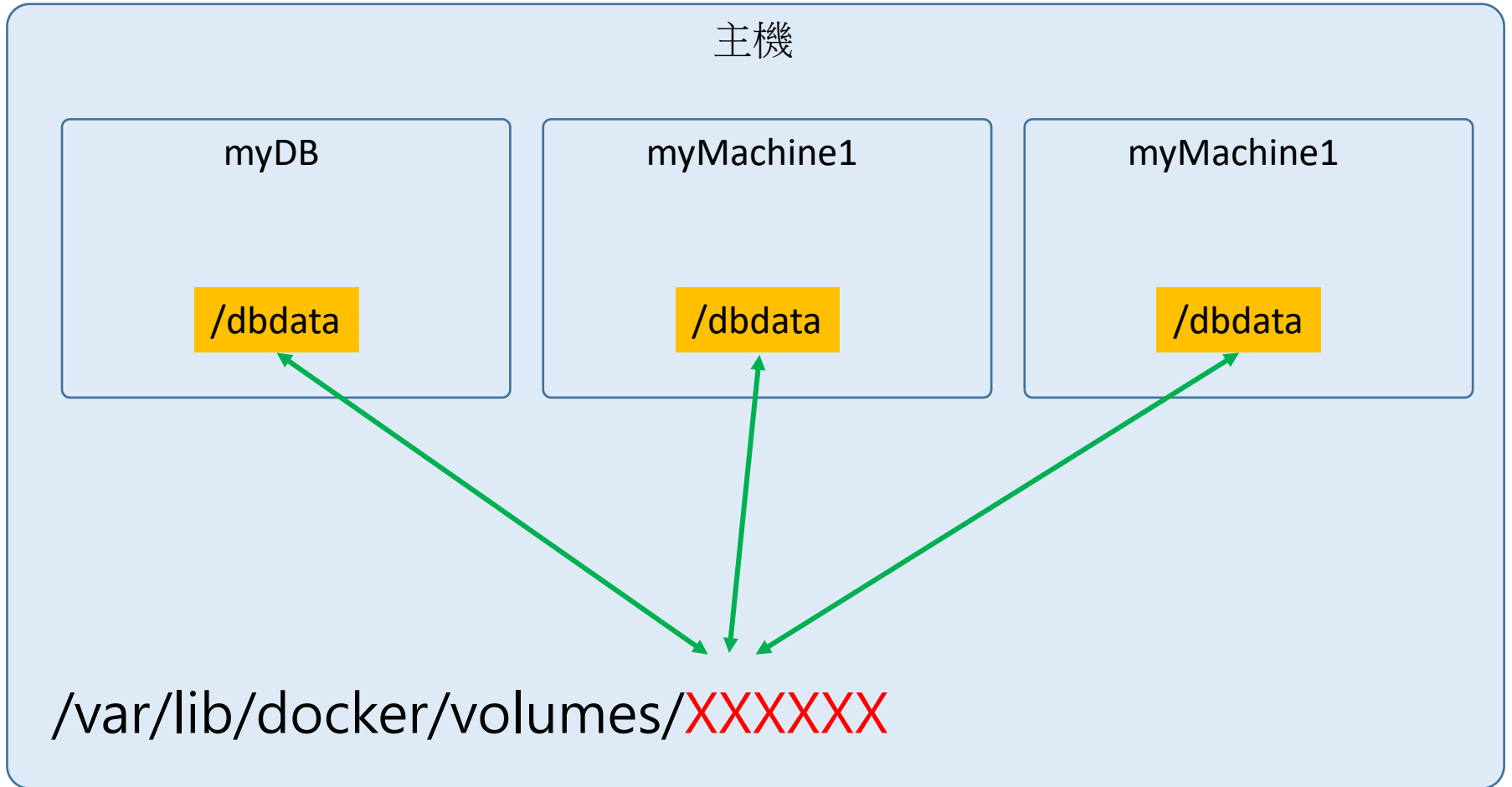


## step3. 刪除 /dbdata 空間

```
root@YARNMaster:~# docker rm myDB
myDB
root@YARNMaster:~# docker rm myMachine1
myMachine1
root@YARNMaster:~# docker rm -v myMachine2
```

最後一個掛載該空間的 容器 要加上 **-v** 才會真的刪除 /dbdata

# Data Volume



```
root@YARNMaster:~# ll /var/lib/docker/volumes/
total 36
drwx-----  9 root root 4096 Jun  6 00:19 ./
drwx----- 10 root root 4096 Jun  6 01:00 ../
drwxr-xr-x   3 root root 4096 Jun  6 00:07 140196d3a107df11170c4ac794348f66a3c1068b082a23617fe6fb9cbef1804a/
drwxr-xr-x   3 root root 4096 Jun  5 20:55 3cddd32c5044fd186988c1459d810467911895f9d11da40cb32ab86469694c5b/
drwxr-xr-x   3 root root 4096 Jun  5 21:02 4320a8d12f5aa96e8249bc441e3cf52d9b9204b1e8296513f02aa33e6721a85d/
drwxr-xr-x   3 root root 4096 Jun  6 00:06 ae98aea57af49362c786355799b5d26035bb4a783686a84b70b65a9d0938d31e/
drwxr-xr-x   3 root root 4096 Jun  5 23:54 b76ae49d8d18f45044b5b8d46fd0cb7d7bb51545fc68c57363a7890943be8b5d/
drwxr-xr-x   3 root root 4096 Jun  5 21:03 f72ac54a619641302f67d9bb09c4142ead4d71059969752d913001b17d97a16c/
drwxr-xr-x   3 root root 4096 Jun  6 00:19 fd526b4981d5bd3539343bf33bdff20a8a4304fd33de56107c3806cdcf00871a/
```

誰是 /dbdata ?

**docker inspect** myMachine2

查看 myMachine2 的設定檔案 (JSON 格式)

```
"Mounts": [
  {
    "Name": "fd526b4981d5bd3539343bf33bdff20a8a4304fd33de56107c3806cdcf00871a",
    "Source": "/var/lib/docker/volumes/fd526b4981d5bd3539343bf33bdff20a8a4304fd33de56107c3806cdcf00871a/_data",
    "Destination": "/dbdata",
    "Driver": "local",
    "Mode": "",
    "RW": true
  }
],
```

```
docker inspect myMachine2 | grep -A5 -B5 'dbdata'
```

取出'dbdata'字串的上下5行

```
root@YARNMaster:~# docker inspect myMachine2 | grep -A5 -B5 'dbdata'
},
"Mounts": [
  {
    "Name": "fd526b4981d5bd3539343bf33bdf20a8a4304fd33de56107c3806cdcf00871a",
    "Source": "/var/lib/docker/volumes/fd526b4981d5bd3539343bf33bdf20a8a4304fd33de56107c3806cdcf00871a/_data",
    "Destination": "/dbdata",
    "Driver": "local",
    "Mode": "",
    "RW": true
  }
],
```



```
/var/lib/docker/volumes/fd526b4981...../_data
```

主機上的路徑