Docker & Kubernetes

David Chiu

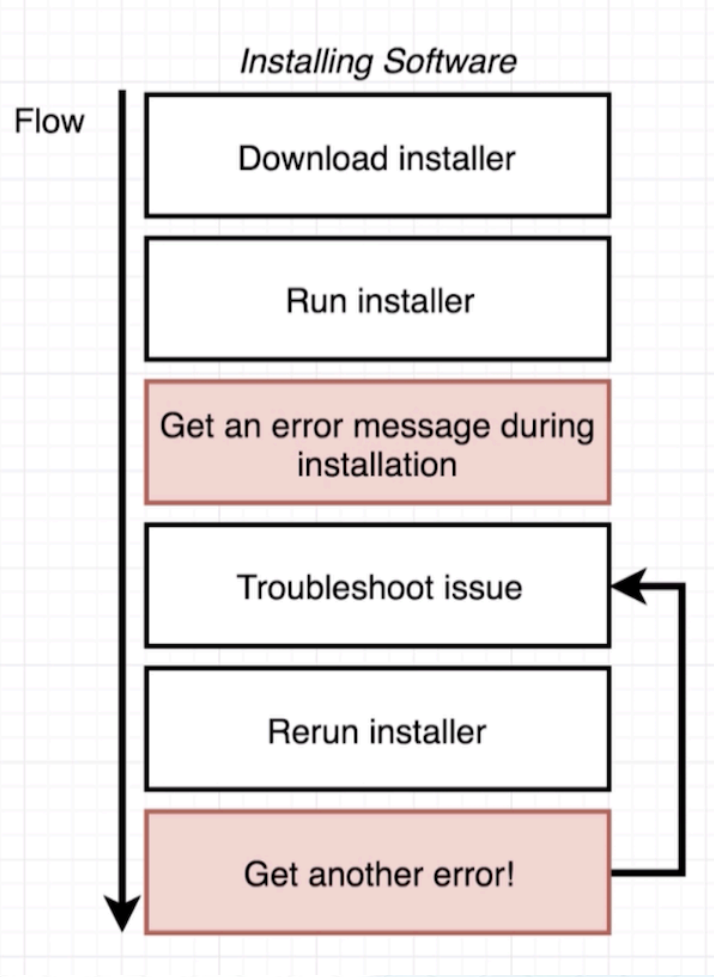
為什麼要使用 Docker

安裝軟體的痛點

- 我需要在一台伺服器上安裝不同環境測試這個案子
- 案子在開發上的時後需要的伺服器資源種類好多
- 我真的需要一個甚麼都安裝的環境來滿足每一個案子嗎？
- 如果這樣想到就安裝會不會有版本衝突而互相牽制的可能？

傳統的安裝流程會碰到環境依賴或套件不全等問題，導致安裝軟體會變得曠日費時

使用Docker 能夠讓啟用軟體服務變得相當簡單

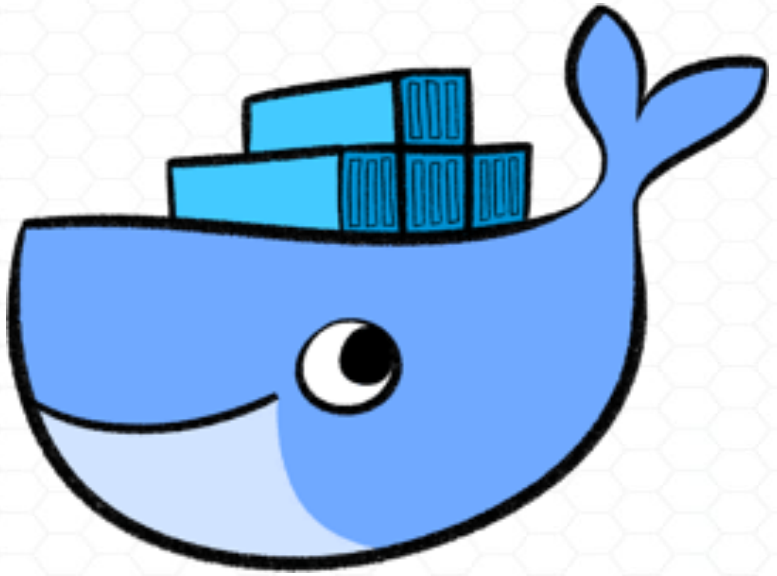


什麼是 Docker

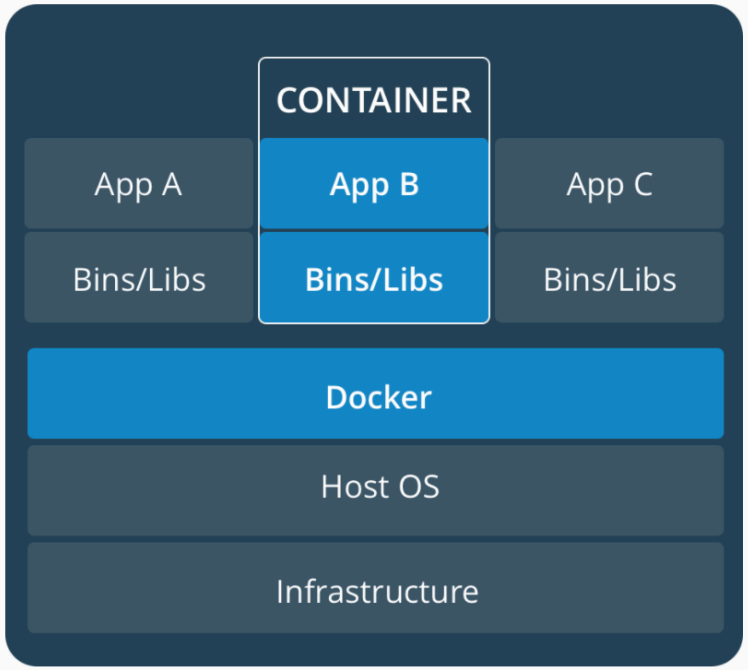
Docker 是一個開源專案，出現於 2013 年初，最初是 Dotcloud 公司內部的 Side-Project。

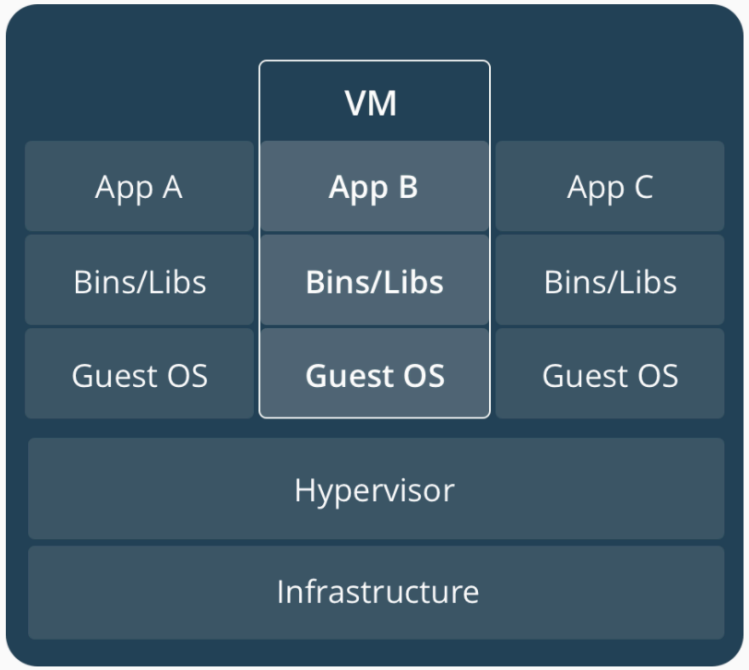
它基於 Google 公司推出的 Go 語言實作。（ Dotcloud 公司後來改名為 Docker ）

Docker 被視為輕量虛擬化技術



Docker 與 VM 的差別





CONTAINERS

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

VIRTUAL MACHINES

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

Docker 與 VM 的差別

特徵	Containers	Virtual Machines
啟動	秒	數分鐘
容量	MB	GB
效能	快	慢
支援數量	多 Containers	根據硬體限制
複製相同環境	快	慢

使用 Docker 的優點

- **更有效率的利用資源**
  - Docker 可以同時設定多個 Containers ，而且啟動速度快
- **統一環境**
  - 保持大家環境一致，可以快速建立與部署環境
- **DevOps**
  - 設定一次，可在任何地方都可以快速建立環境且正常執行

Docker 的概念

- ***Image***
  - 映像檔

- ***Container***
  - 容器，利用映像檔（ Image ）所創造出來的，一個 Image 可以創造出多個不同的 Container，
  - Container 也可以被啟動、開始、停止、刪除，並且互相分離。

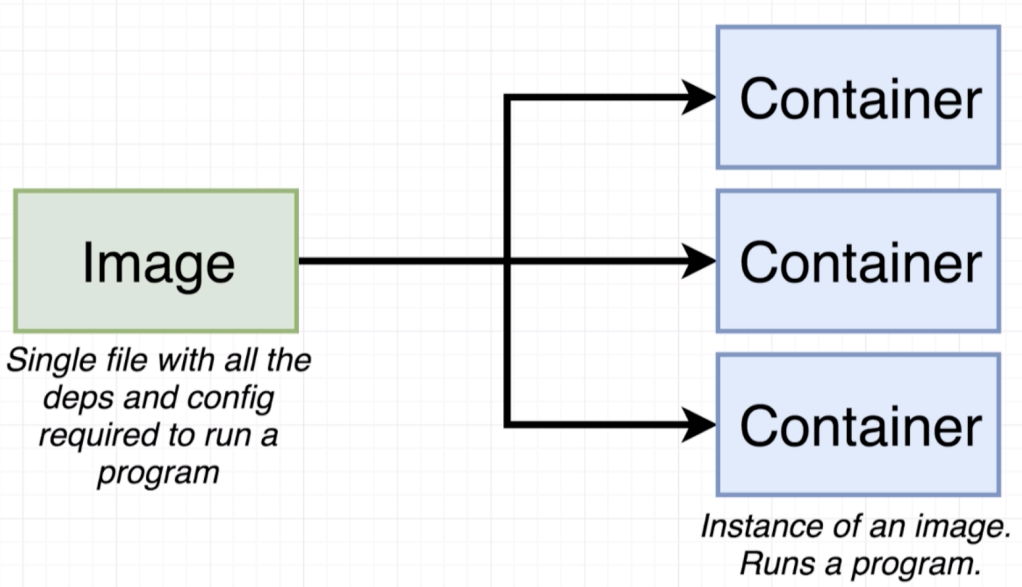
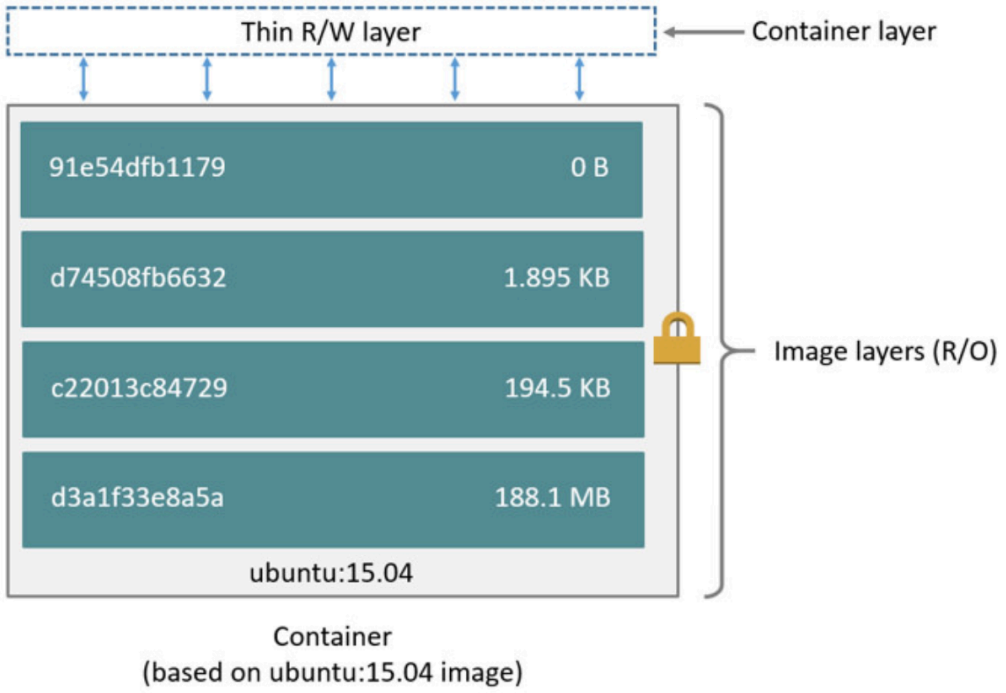


Image & Container

- Image 是唯讀（ R\O ）
- Container 是讀寫模式（ R\W ）

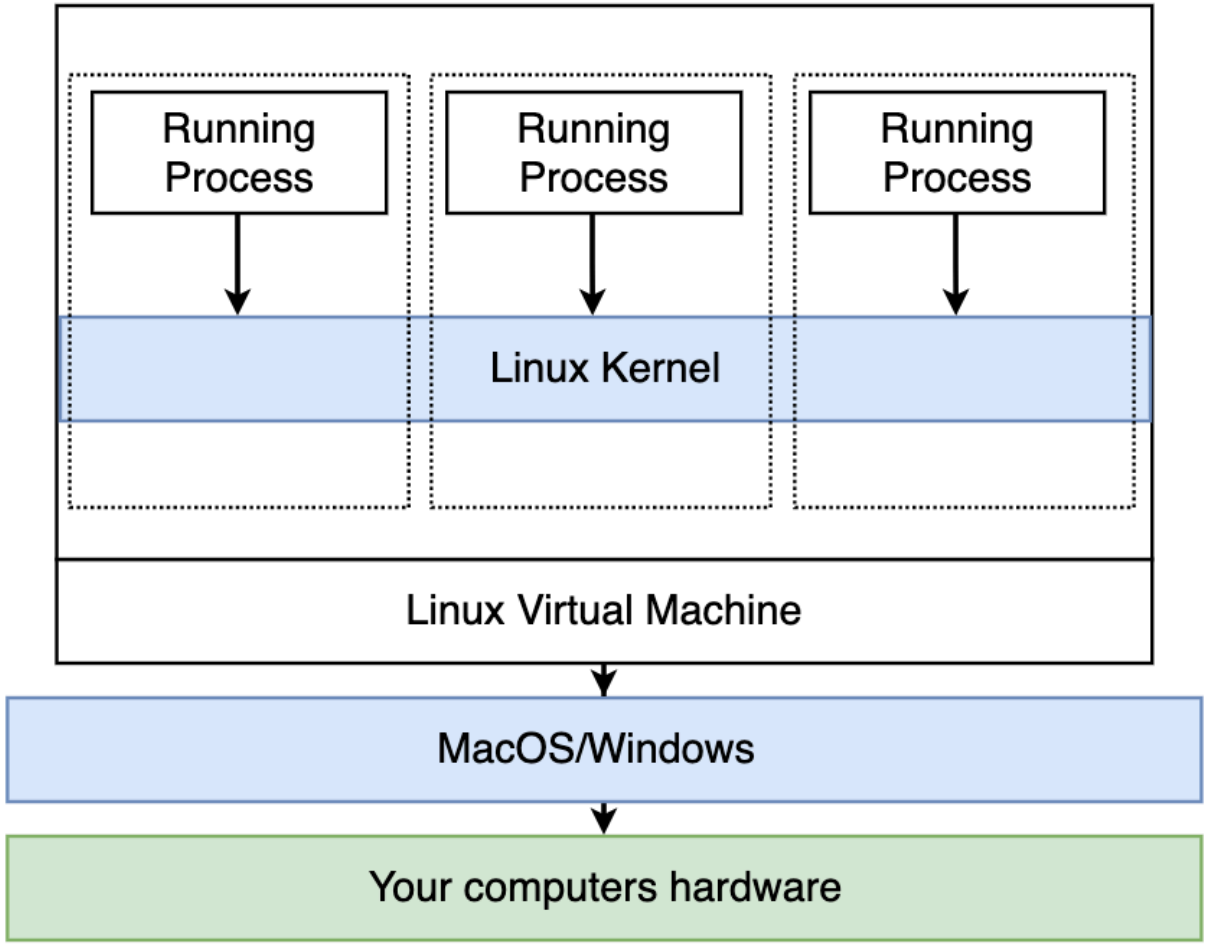


安装 Docker

下載 Docker Desktop (Windows & Mac)

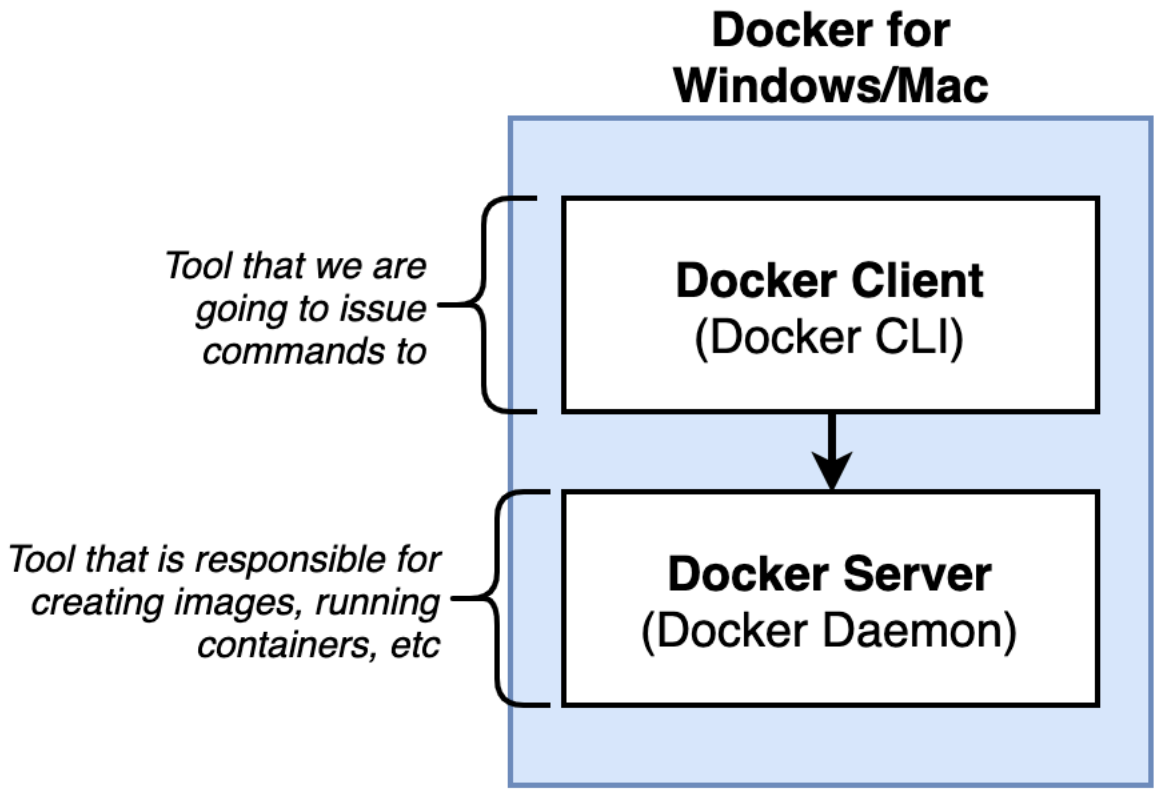
https://hub.docker.com/

Docker Desktop 的運作方式



Docker 運行方式

- Docker 分為 Client 與 Server
- Client 負責下指令
- Server 接受指令創建Images 與運行 Containers



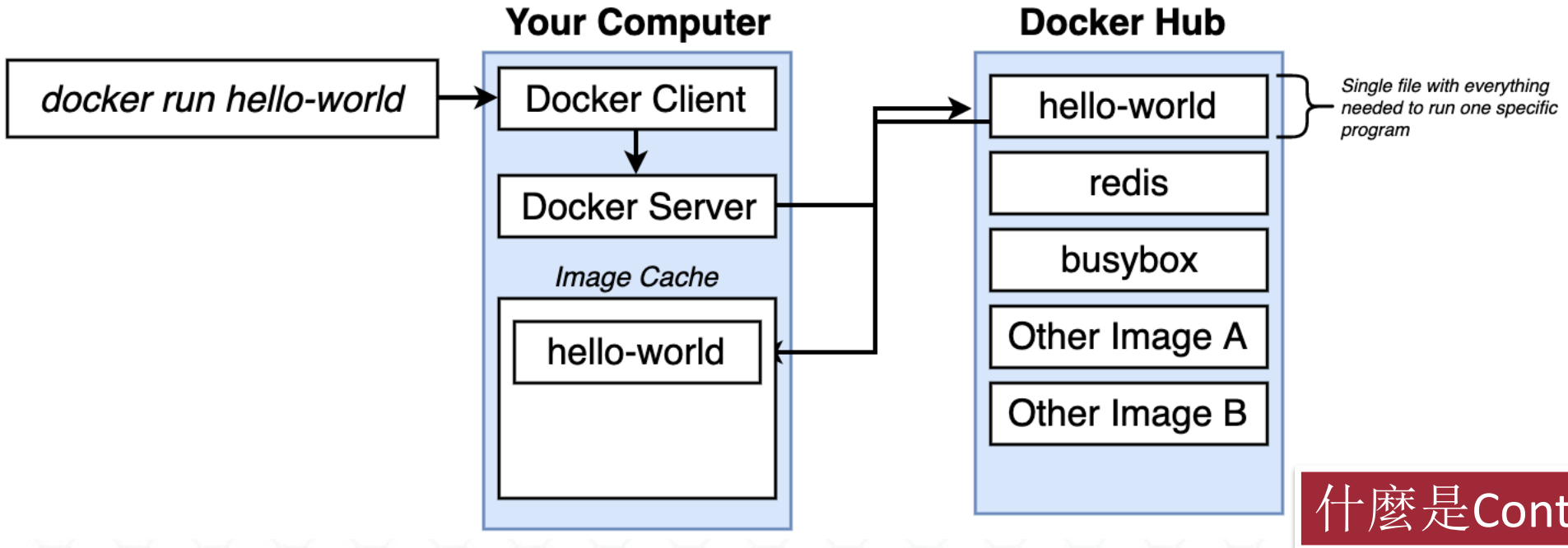
Docker 基本指令

- 查看目前版本
  - docker version
- 查看目前 images
  - docker images

如有權限問題
記得加上 sudo 指令

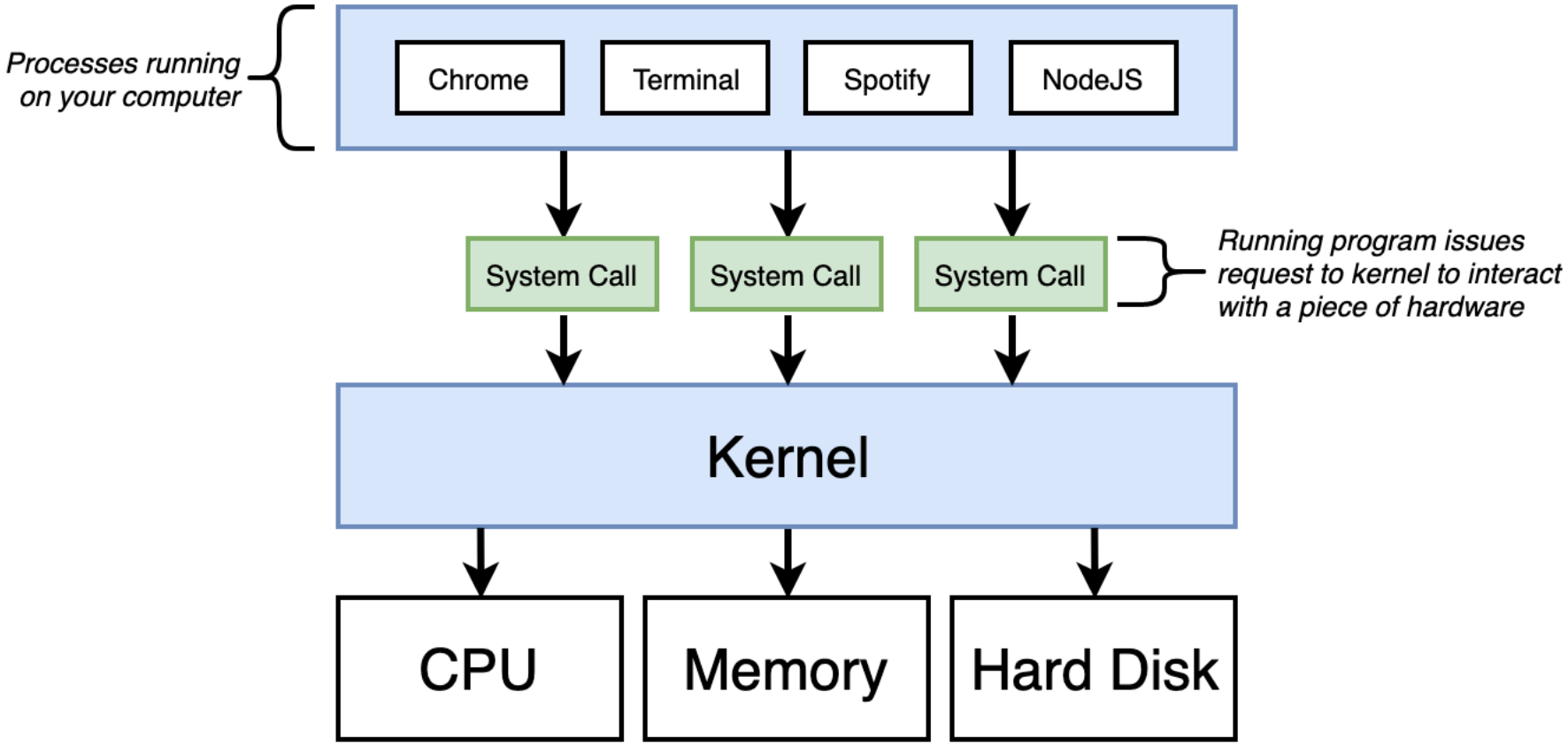
運行第一個 Docker Image

- 執行 hello-world
  - docker run hello-world

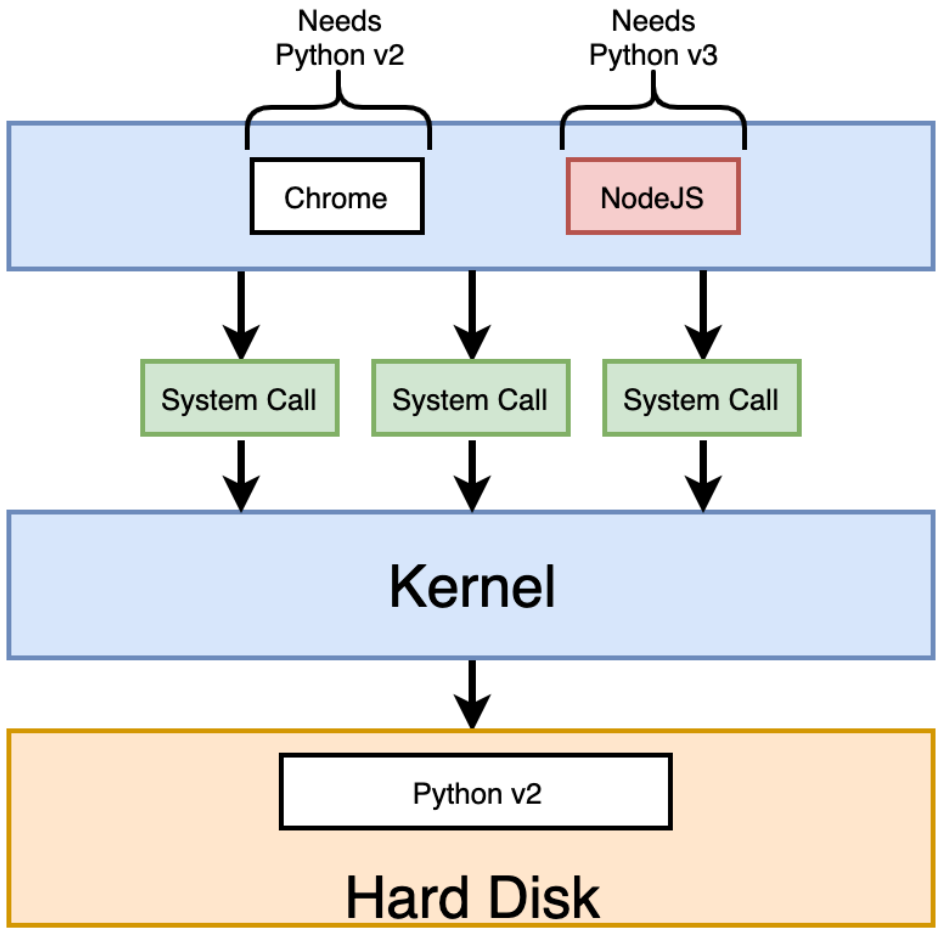


什麼是Containers?

作業系統

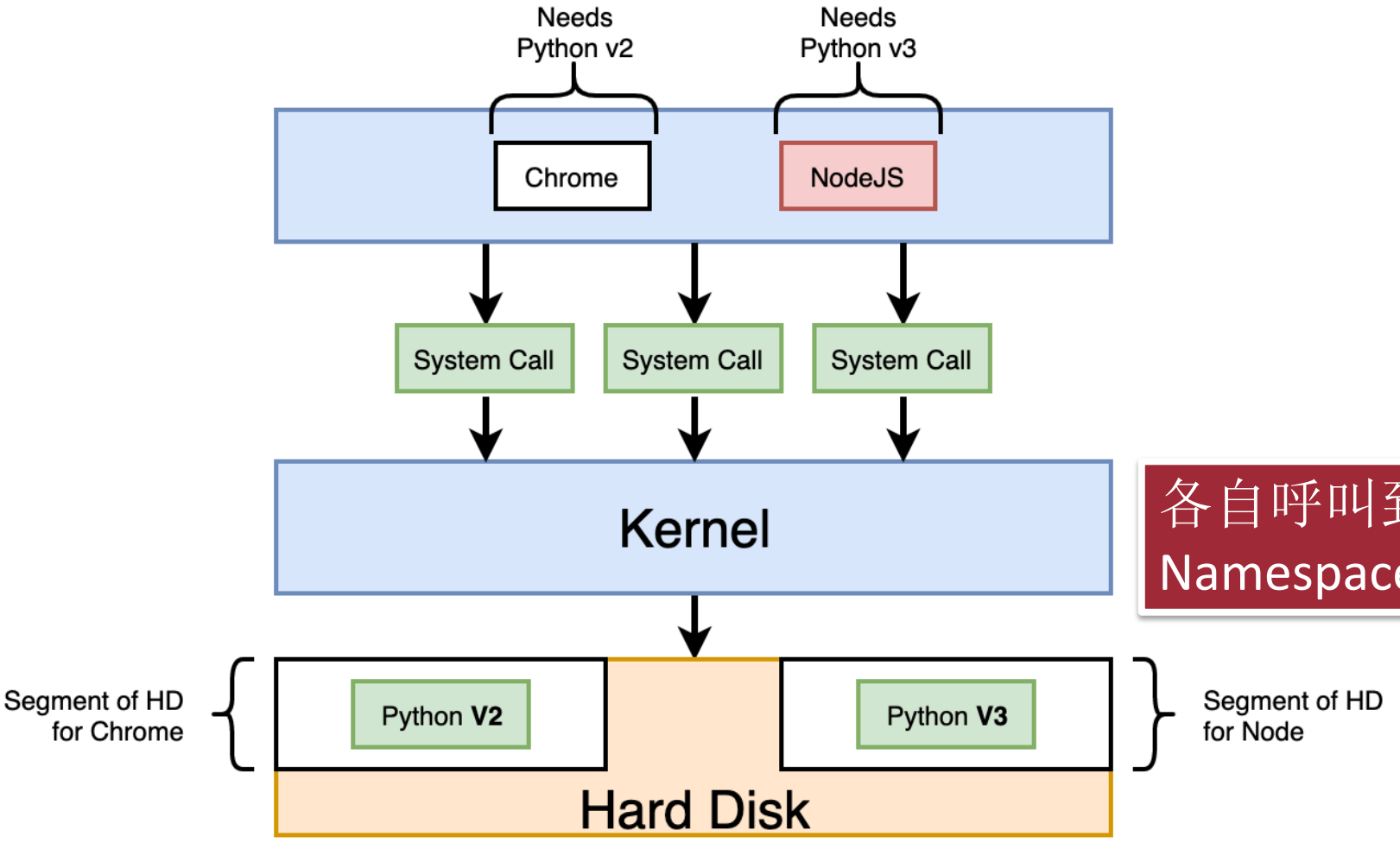

Processes running on your computer
Chrome
Terminal
Spotify
NodeJS
System Call
System Call
System Call
Running program issues request to kernel to interact with a piece of hardware
Kernel
CPU
Memory
Hard Disk

透過 Kernel 呼叫程式



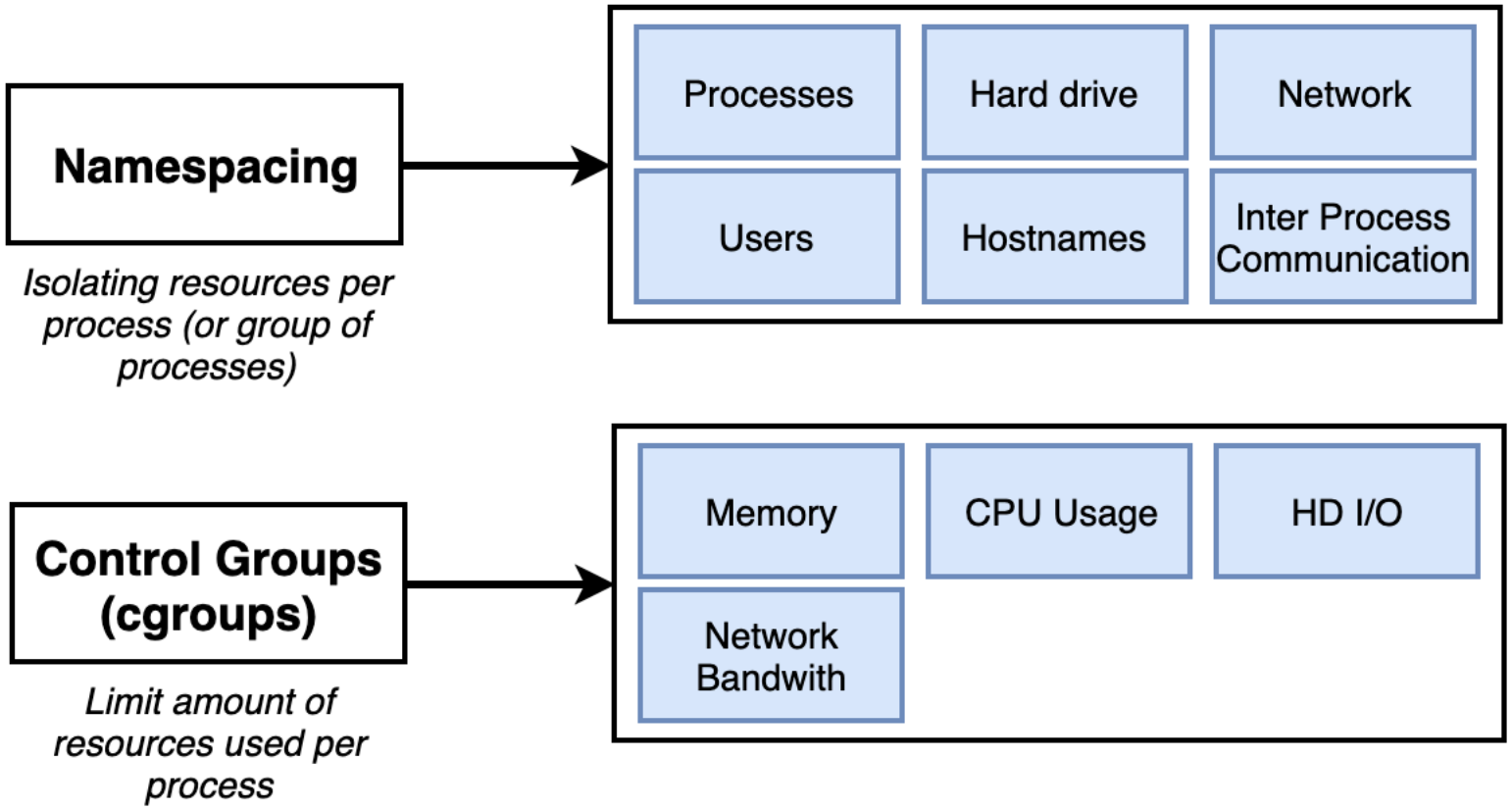
如果在硬碟只有裝Python2
NodeJS 無法呼叫Python3

透過 Kernel 呼叫程式

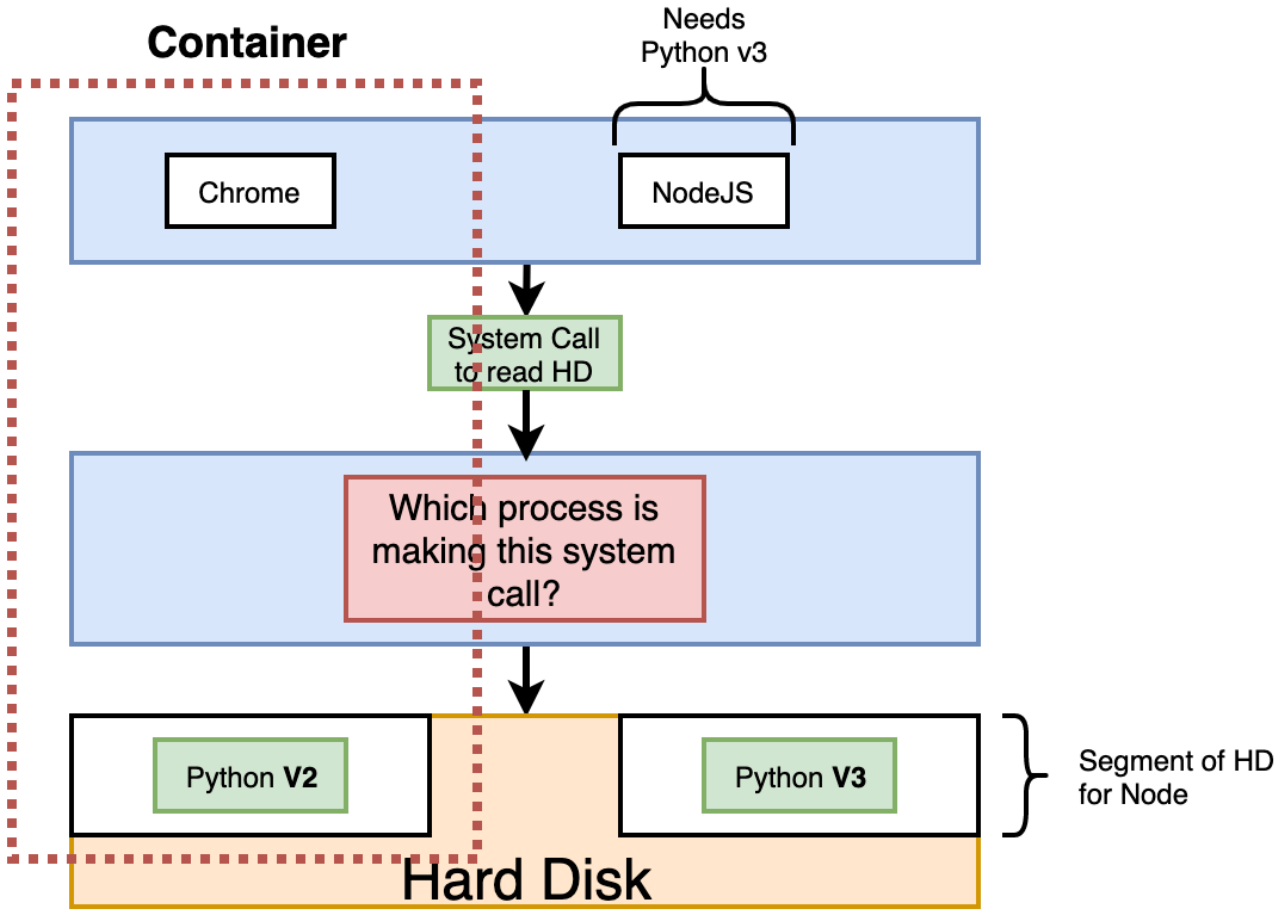


各自呼叫到各自
Namespace 下的程式

利用 Namespacing 與 control group 分割資源



Container


Container
Needs
Python v3
Chrome
NodeJS
System Call
to read HD
Which process is
making this system
call?
Python V2
Python V3
Segment of HD
for Node
Hard Disk

每個Container 擁有獨立的環境跟硬體資源

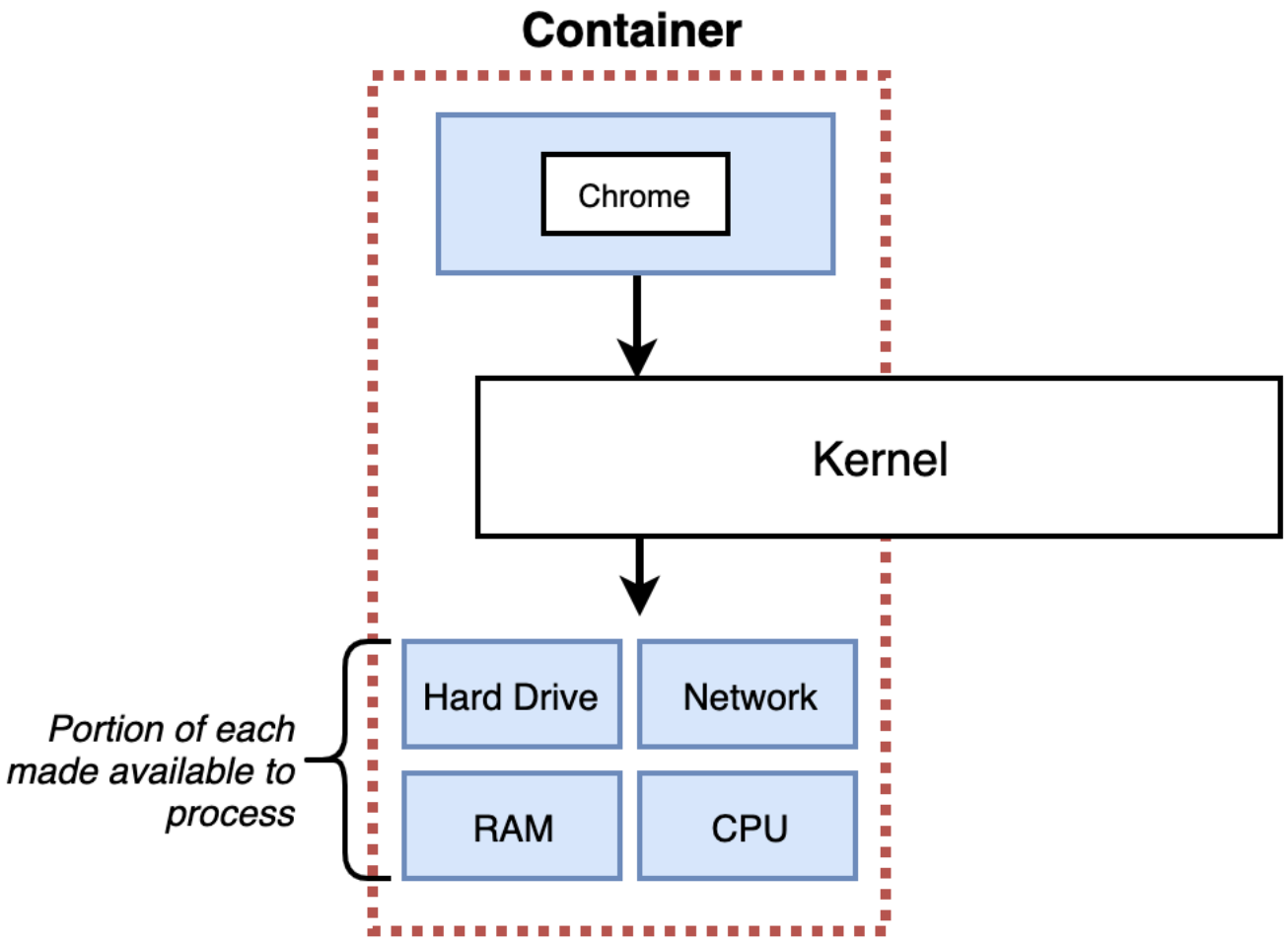
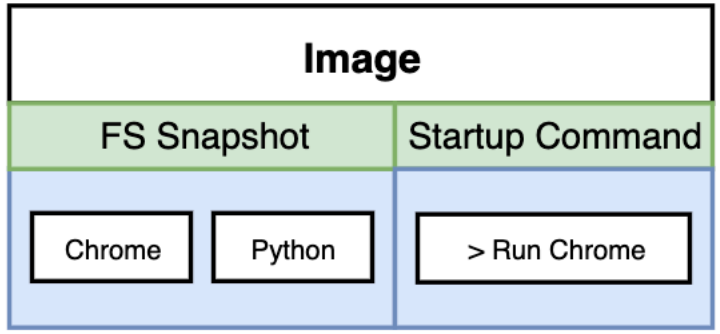
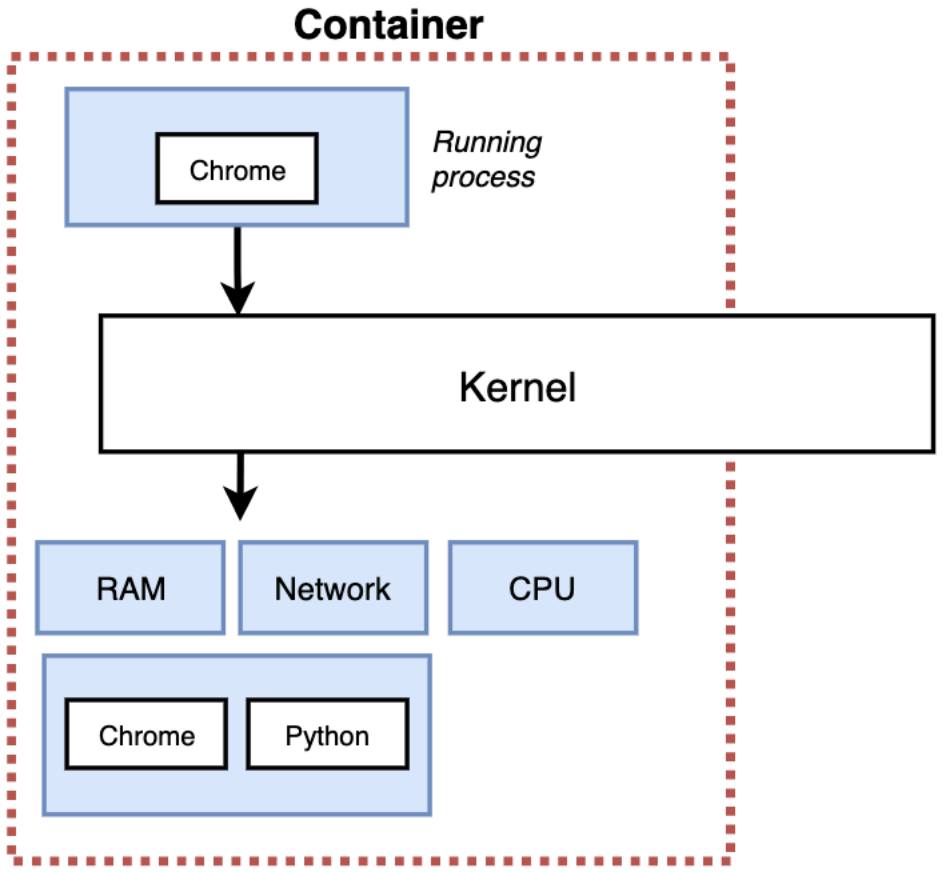


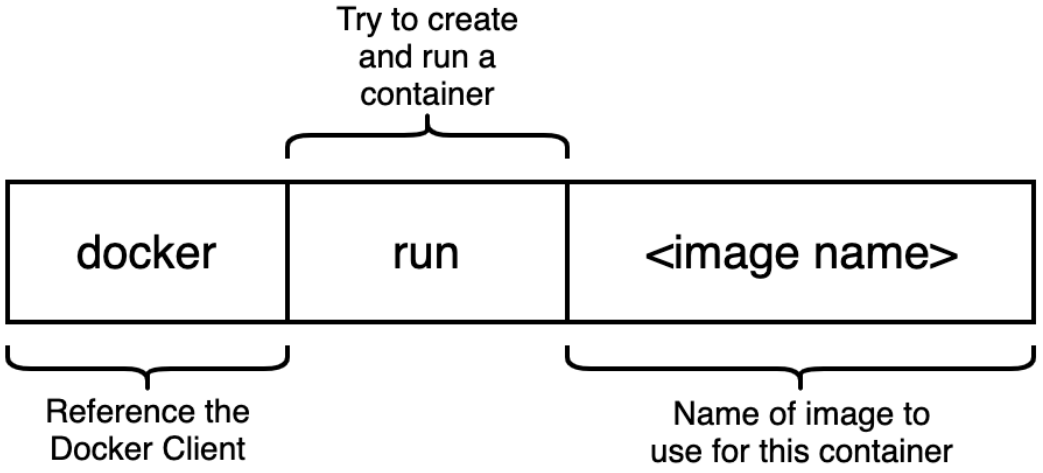
Image 與 Container 的關係



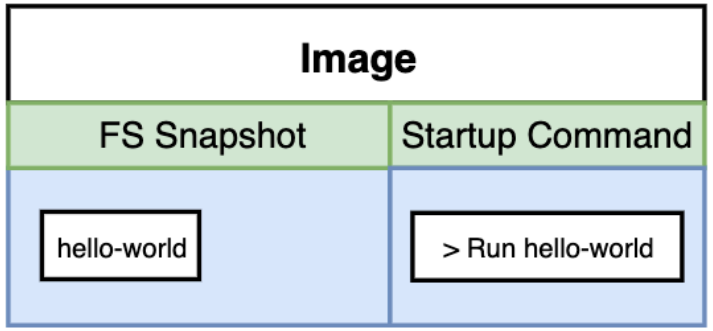


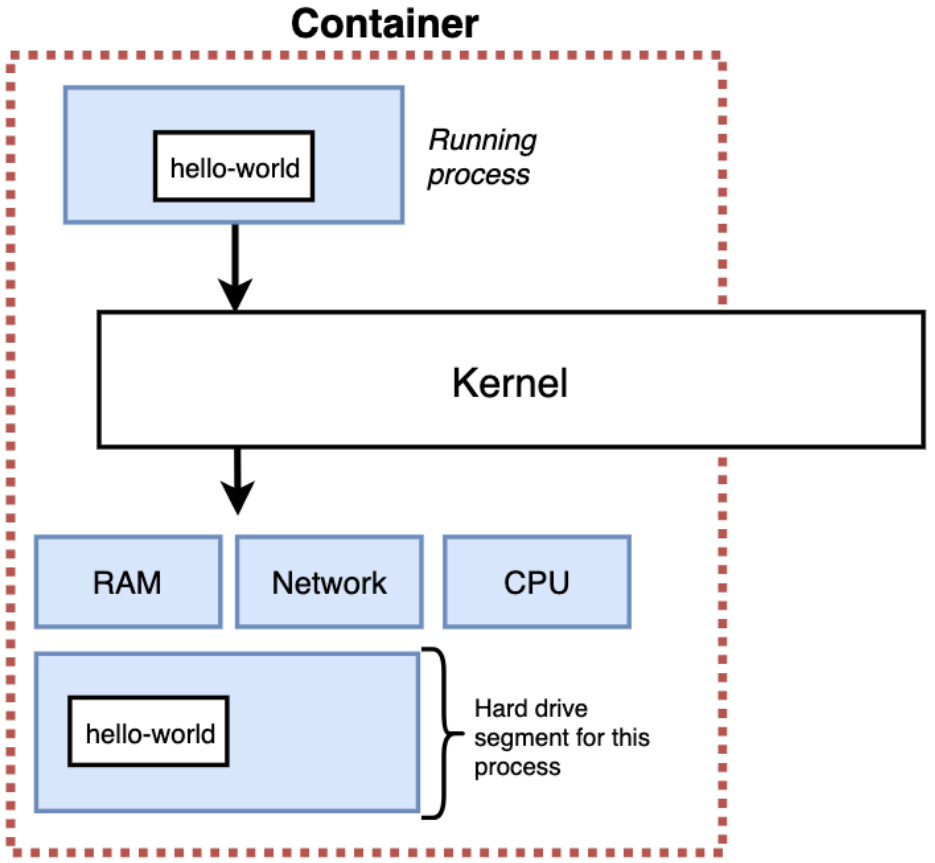
Docker Run

- docker run hello-world



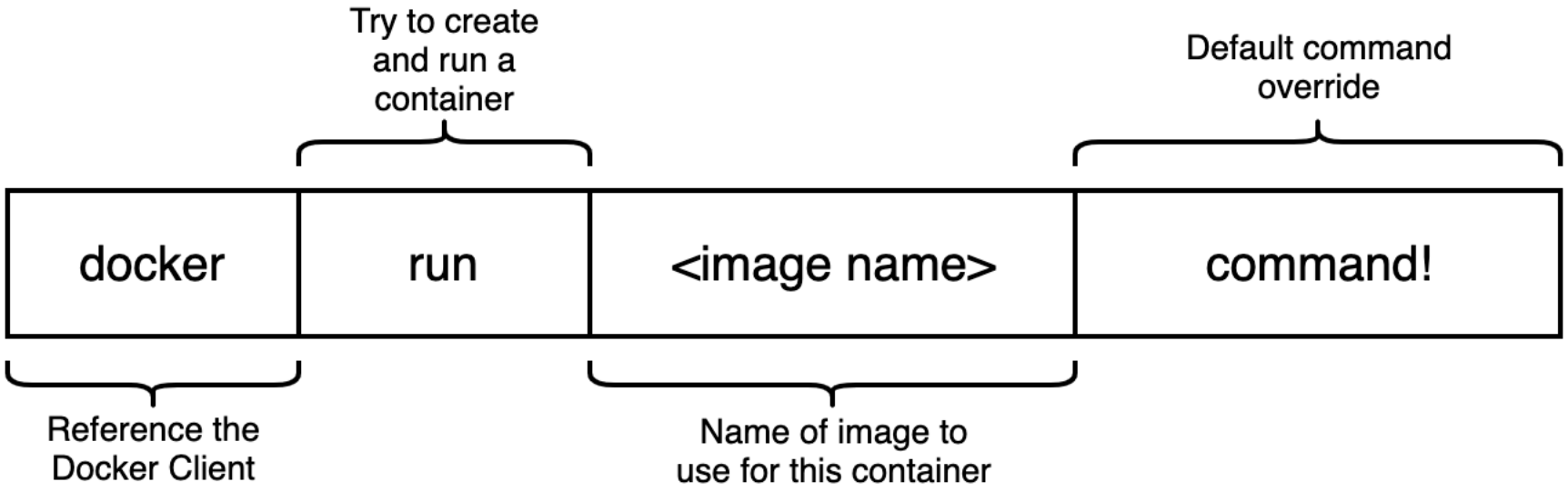
Docker Run 如何運行





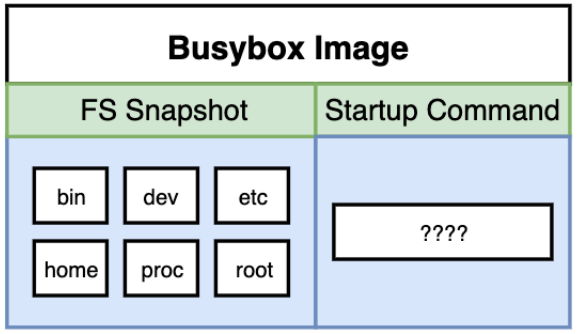
重載起始指令Docker Run

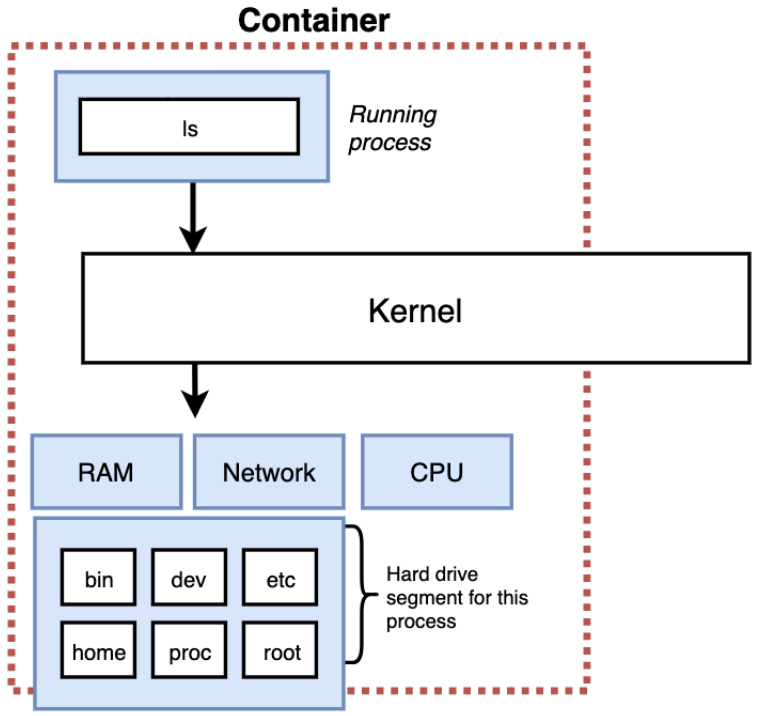
- docker run busybox echo hi there



重載起始指令Docker Run

- docker run busybox ls

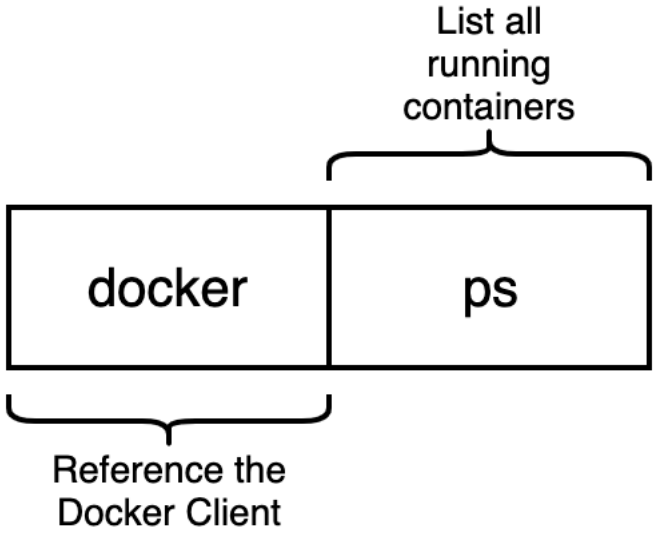




執行 docker run hello-world ls 呢?

列出運行中的 Docker

- docker ps



Docker PS

- 執行長時間工作
  - docker run busybox ping google.com
- 列出所有 Docker Process
  - docker ps

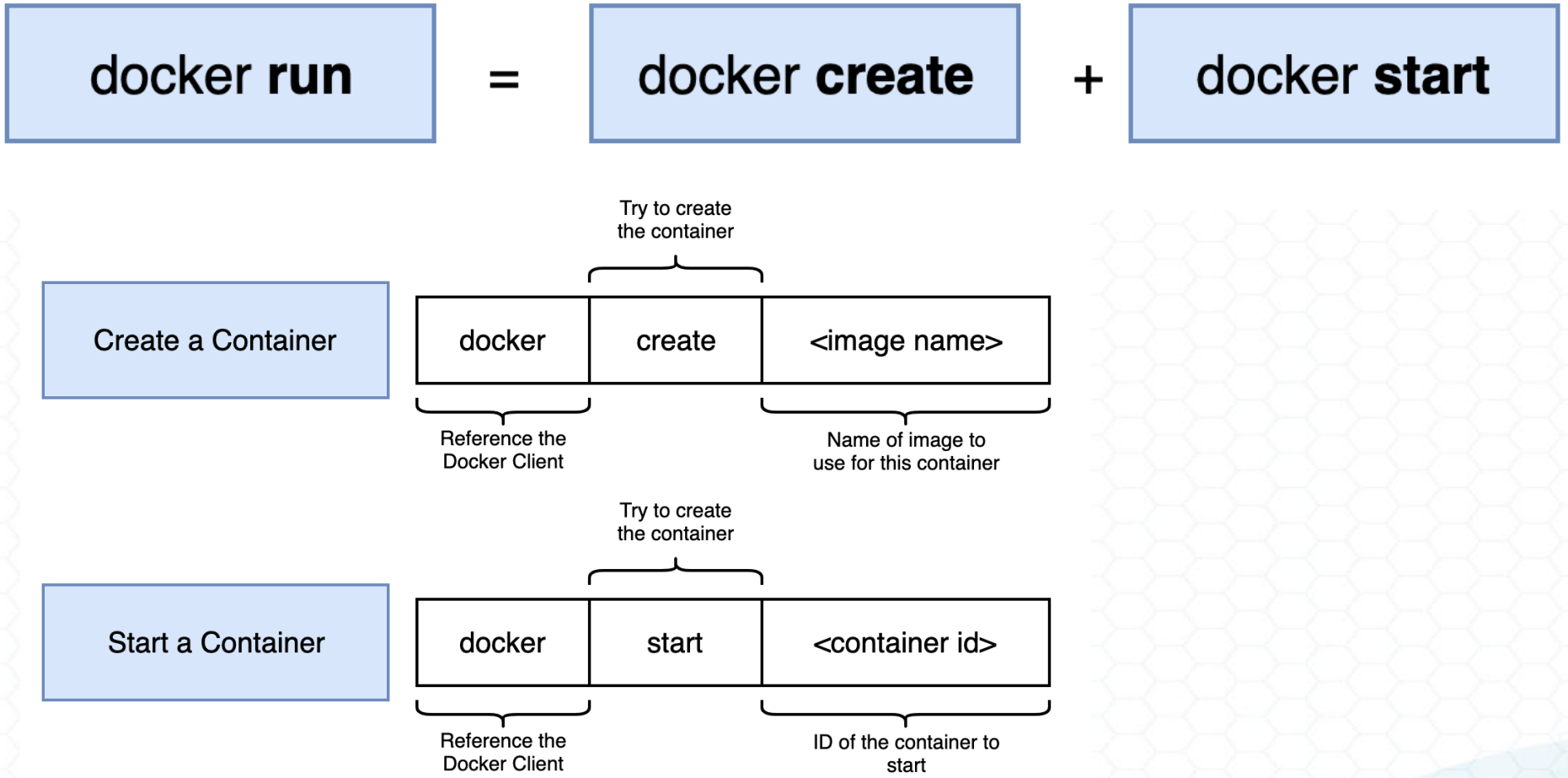
```
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS                    PORTS
                    NAMES
84fc0eaa906d        busybox                        "ping google.com"        54 seconds ago      Up 50 seconds
                    ecstatic_keller
9527decbeddf        gitlab/gitlab-ce:latest        "/assets/wrapper"        2 weeks ago         Up 2 weeks (healthy)      22/tcp, 443/tcp,
 0.0.0.0:8787->80/tcp   gitlab
```

Docker PS -all

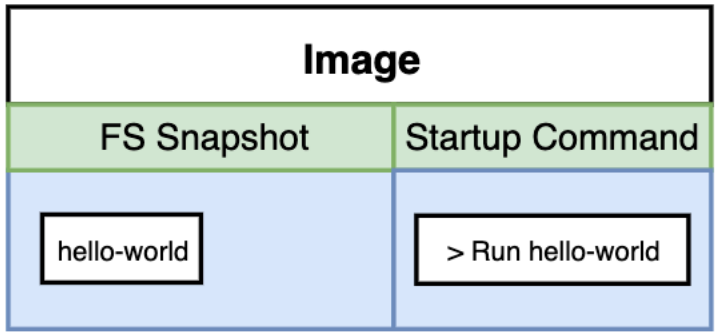
- sudo docker ps -all

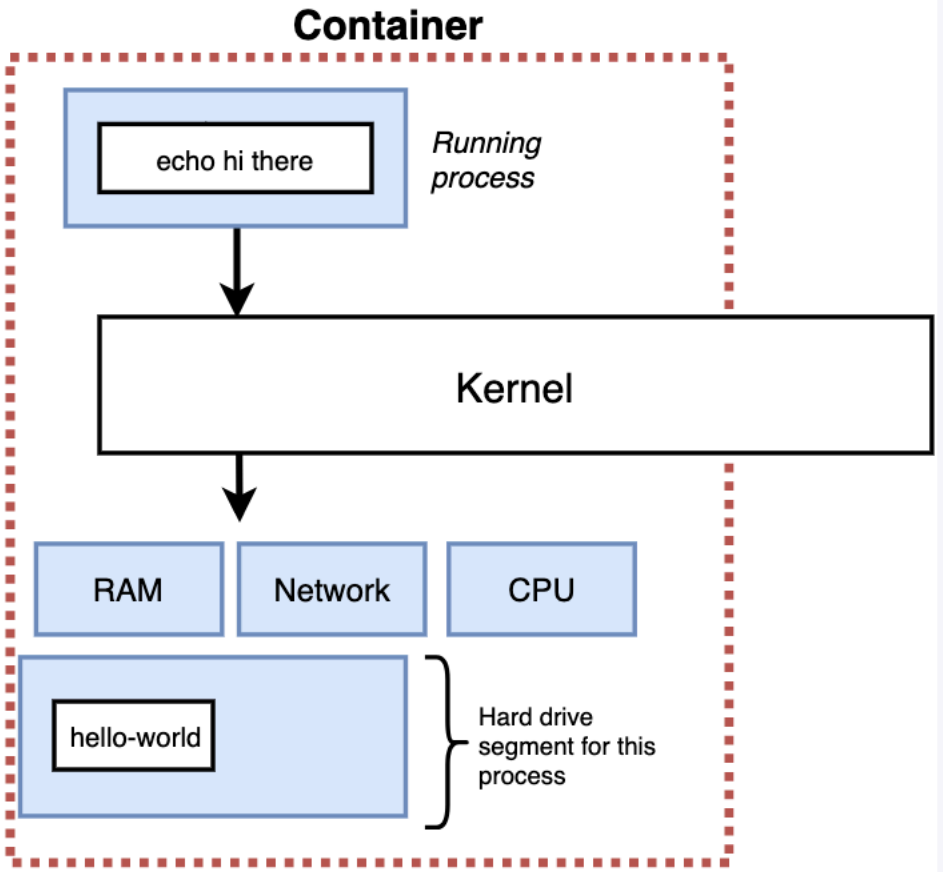
```
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS
  NAMES
84fc0eaa906d        busybox             "ping google.com"   3 minutes ago       Exited (0) 2 minutes ago
  ecstatic_keller
```

Docker run


docker run
=
docker create
+
docker start
Create a Container
Try to create the container
docker
create
<image name>
Reference the Docker Client
Name of image to use for this container
Start a Container
Try to create the container
docker
start
<container id>
Reference the Docker Client
ID of the container to start

Create + Start



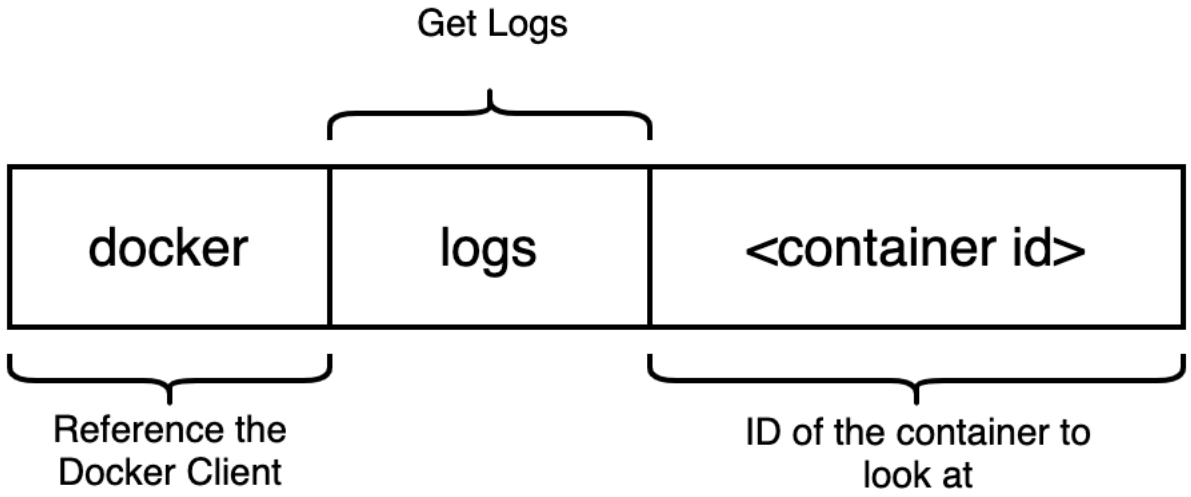


Create + Start

```
david@peopleminer:~$ sudo docker create hello-world
c801301a2f6622f78b14e5291ea9ad93cd8812a3311f5b9aaee8c0de4674bbde
david@peopleminer:~$ sudo docker start  -a c801301a2f6622f78b14e5291ea9ad93cd8812a3311f5b9aaee8c0de4674bbde

Hello from Docker!
This message shows that your installation appears to be working correctly.

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 Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

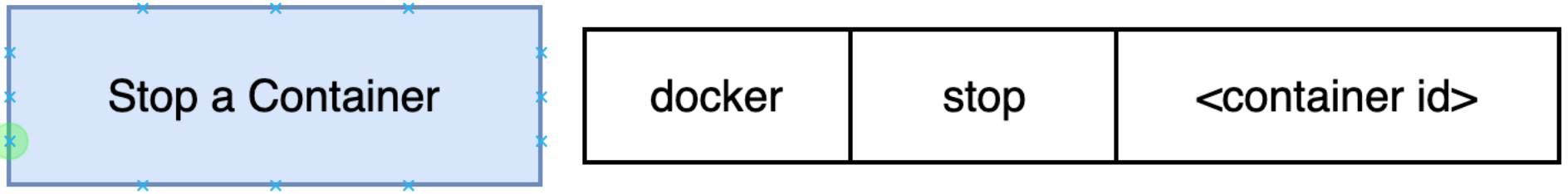
Docker Logs

- docker logs

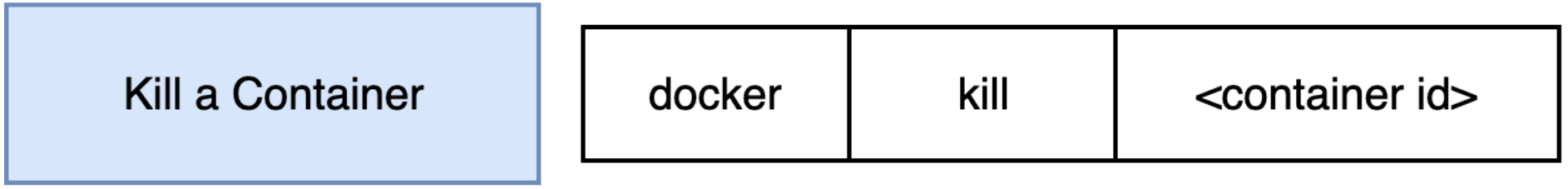


Stop Containers

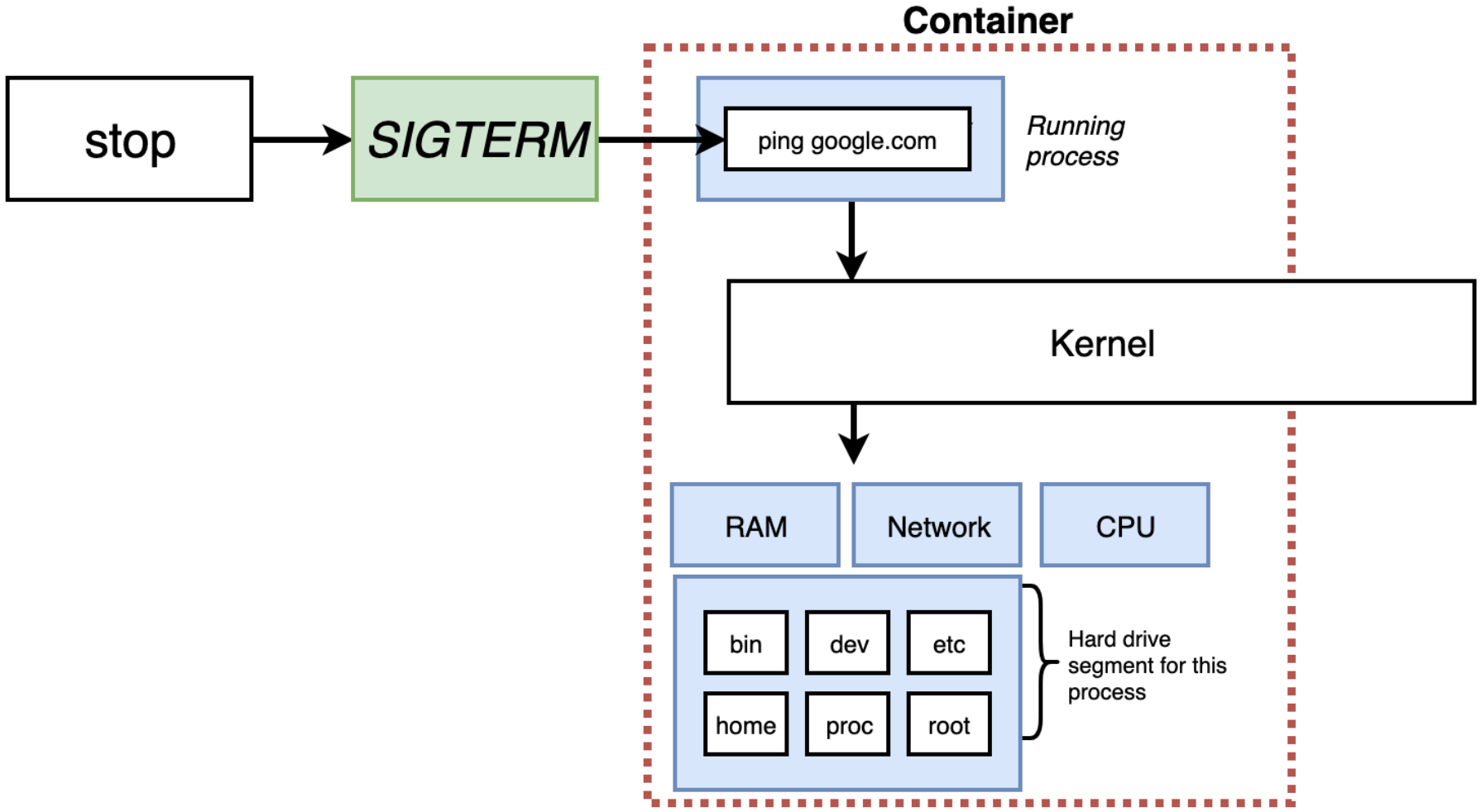
- docker stop



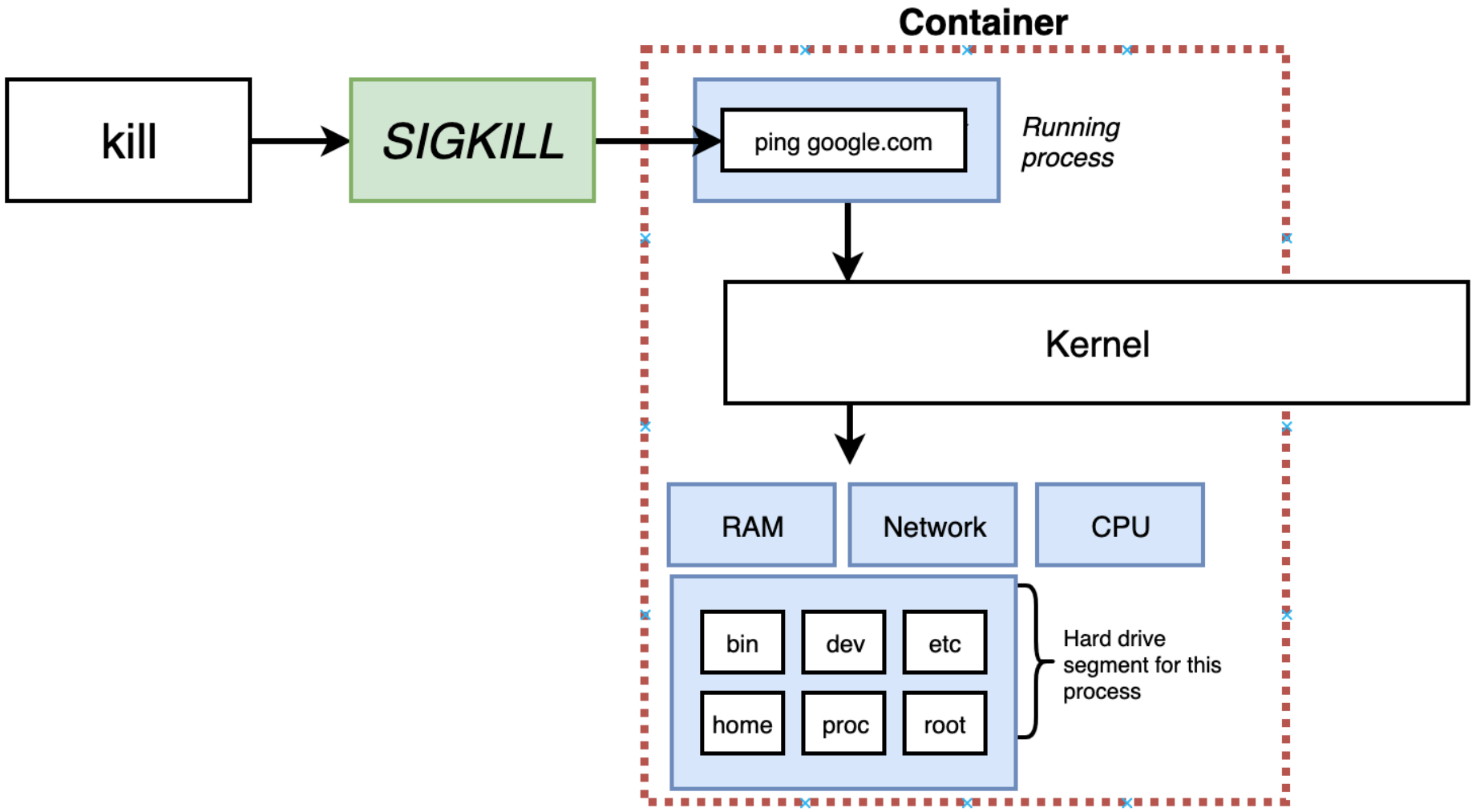
- docker kill



如何停止container

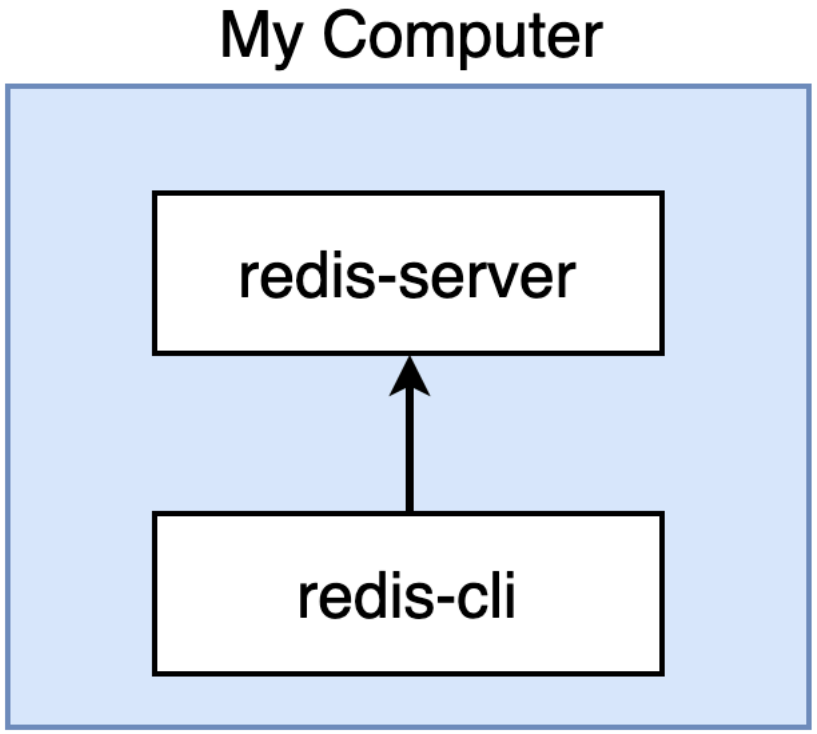


如何停止container

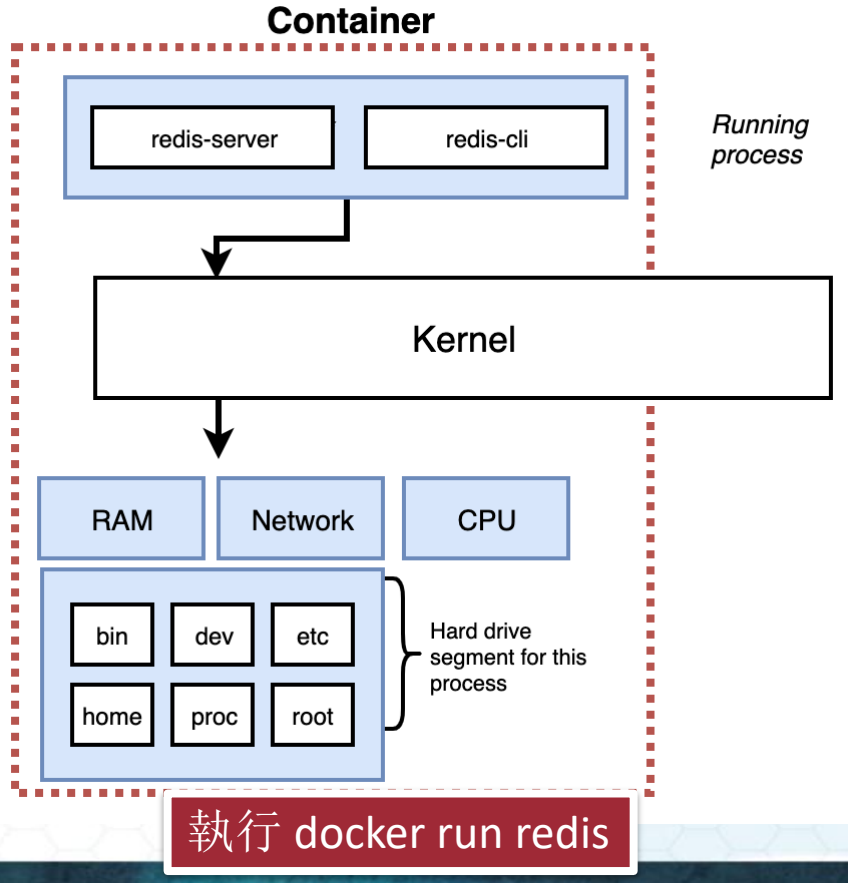


啟用 Redis

在自己的電腦

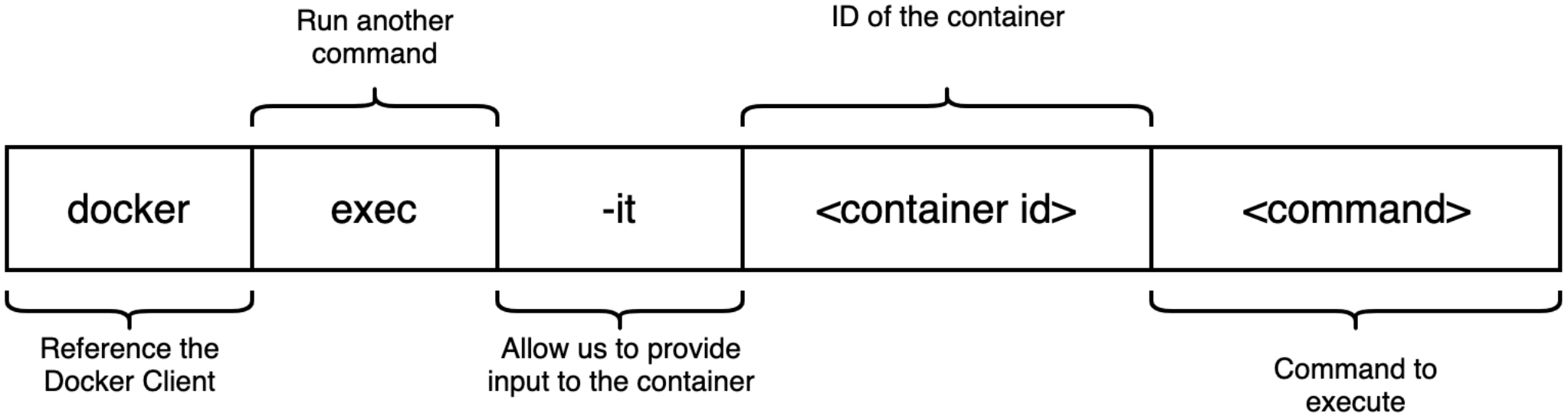


在Container中

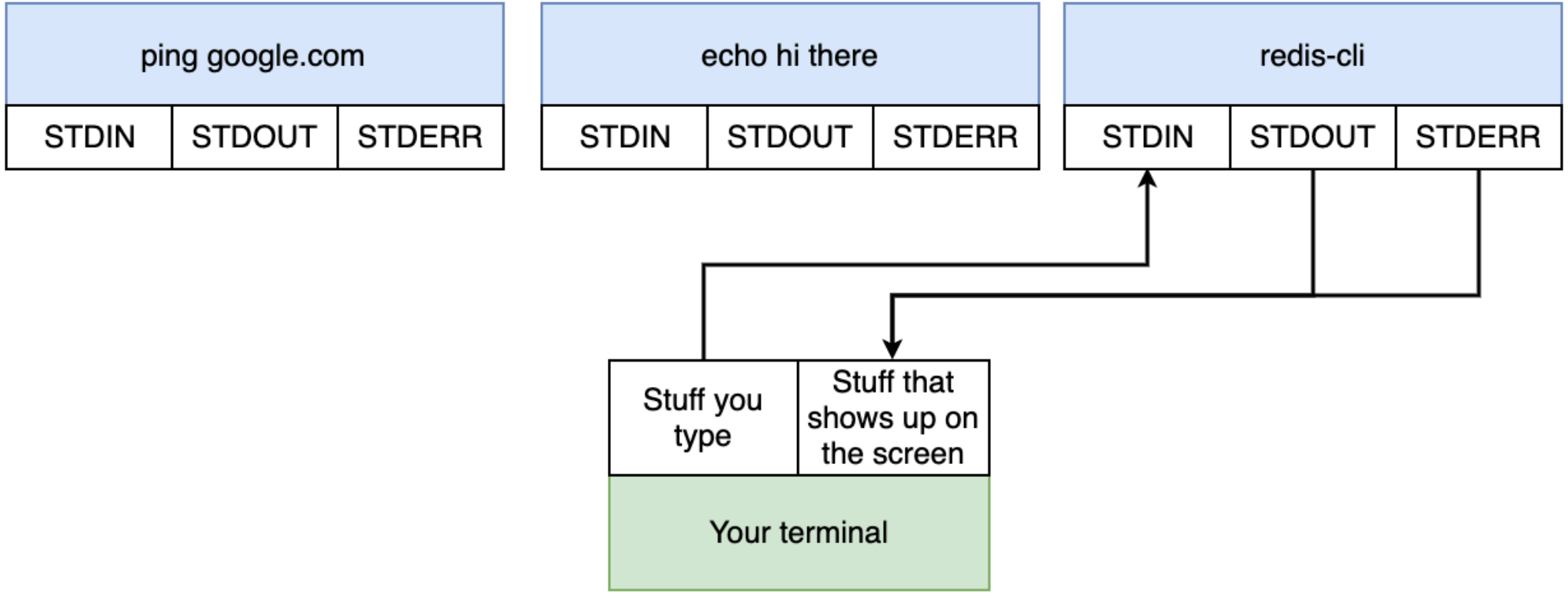


It flag

- 輸入至Container 中
  - docker exec –it container_id redis-cli

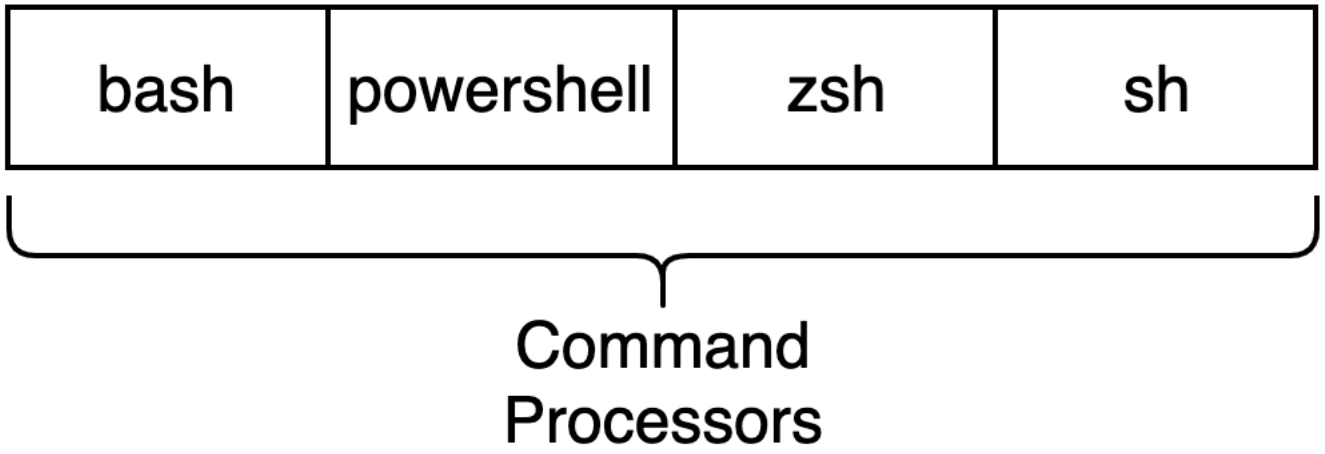


Container 操作流程

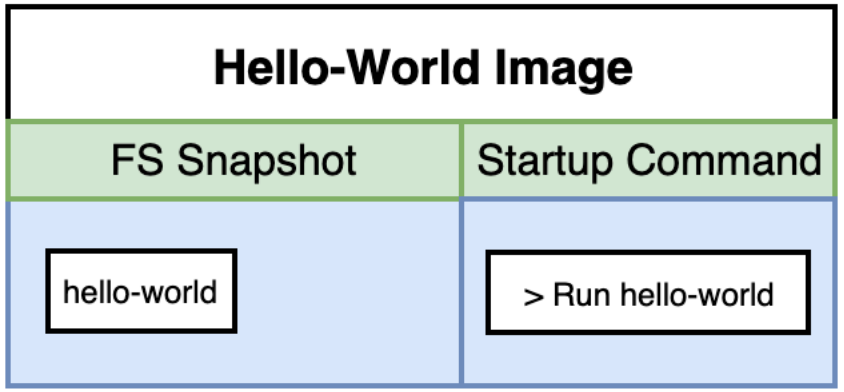


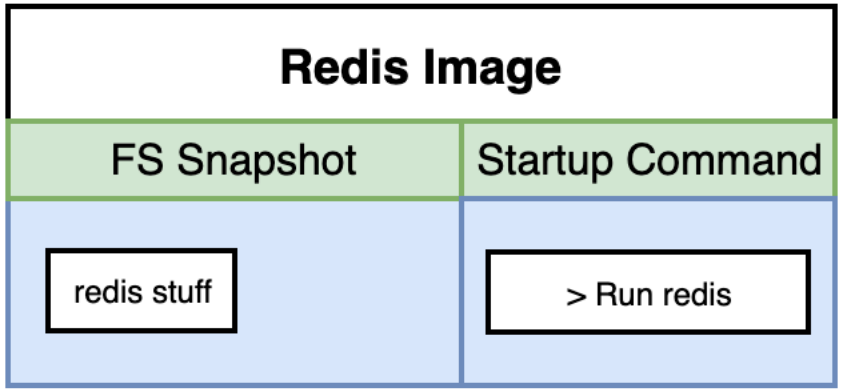
取得 command prompt

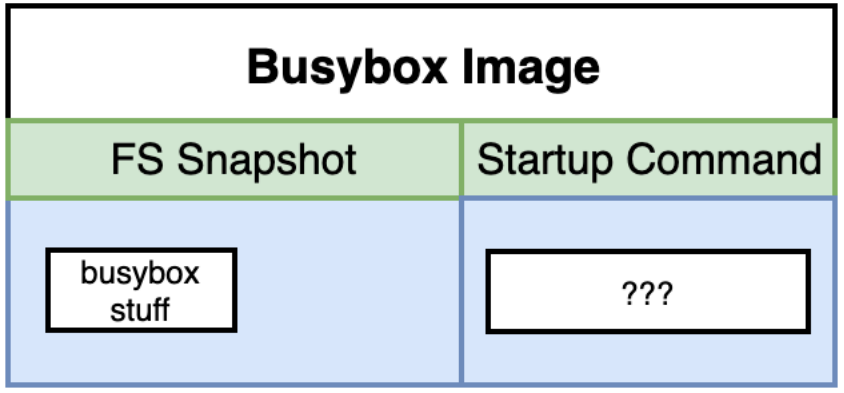
- docker exec –it container_id sh



建立 Docker

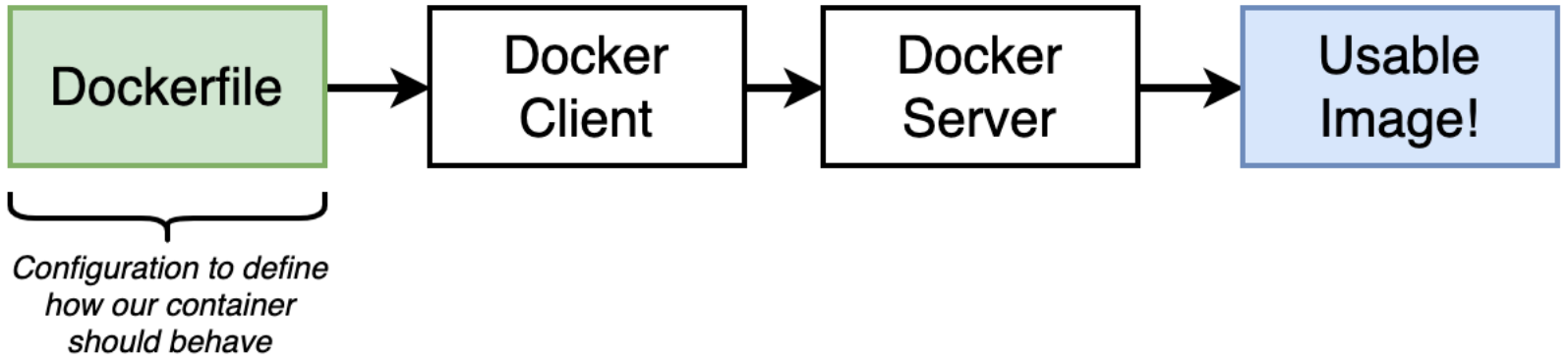

Hello-World Image
FS Snapshot
Startup Command
hello-world
> Run hello-world


Redis Image
FS Snapshot
Startup Command
redis stuff
> Run redis

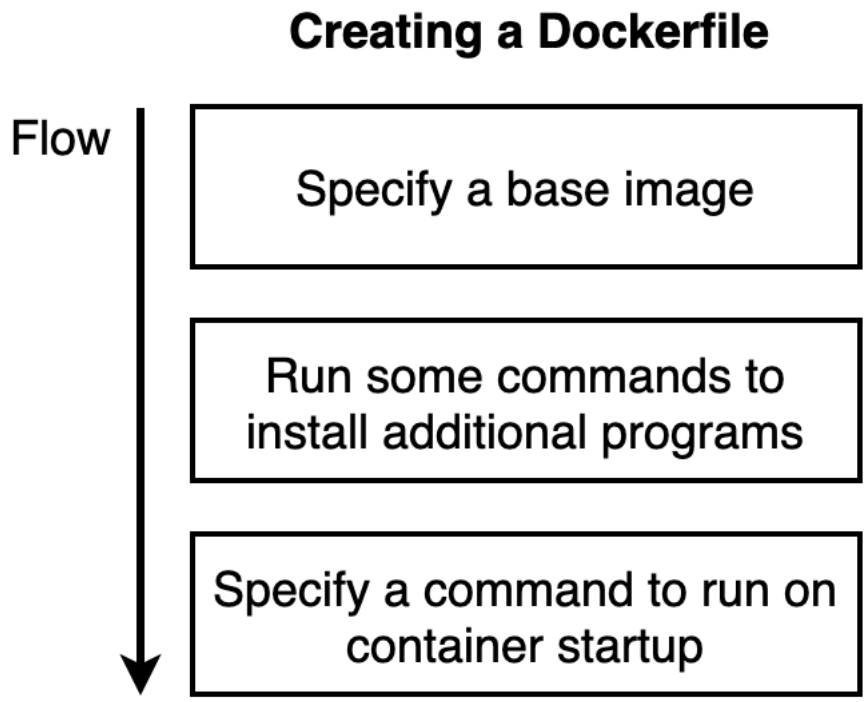

Busybox Image
FS Snapshot
Startup Command
busybox stuff
???

Docker File

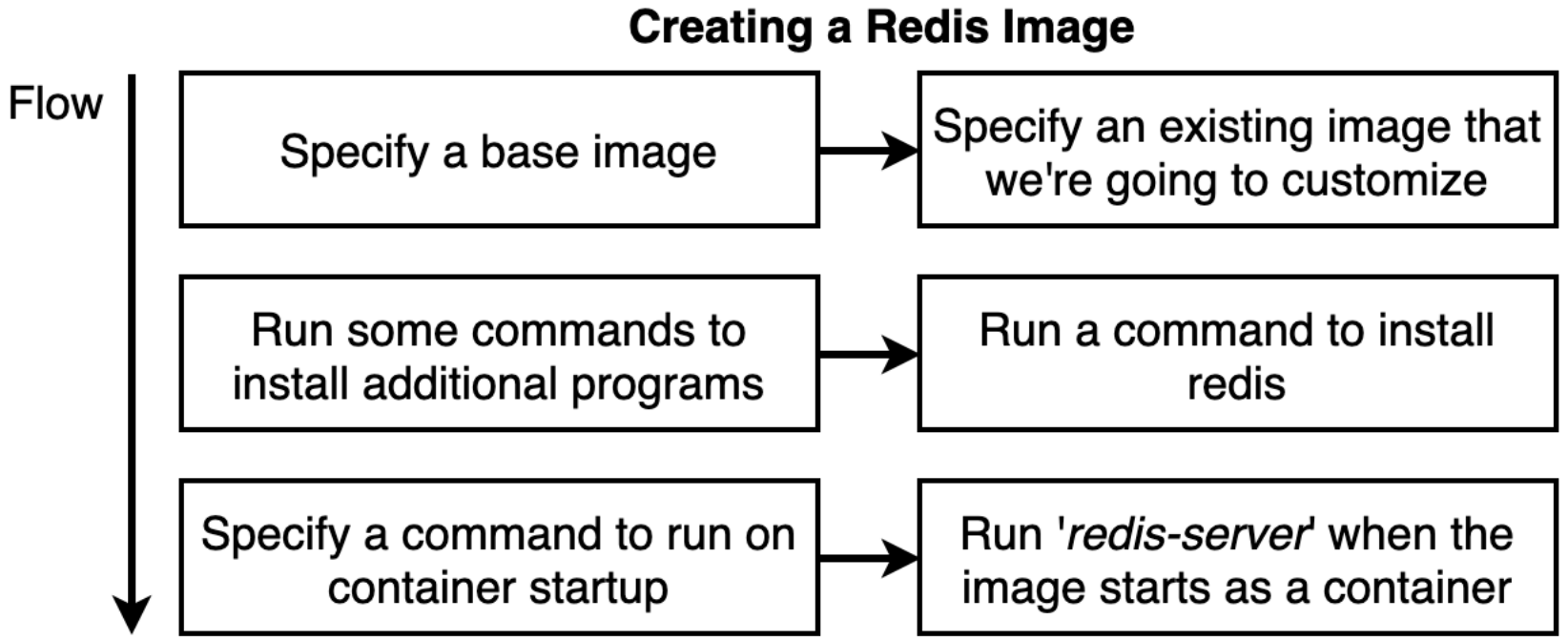
- 利用 Docker File 建立 Image



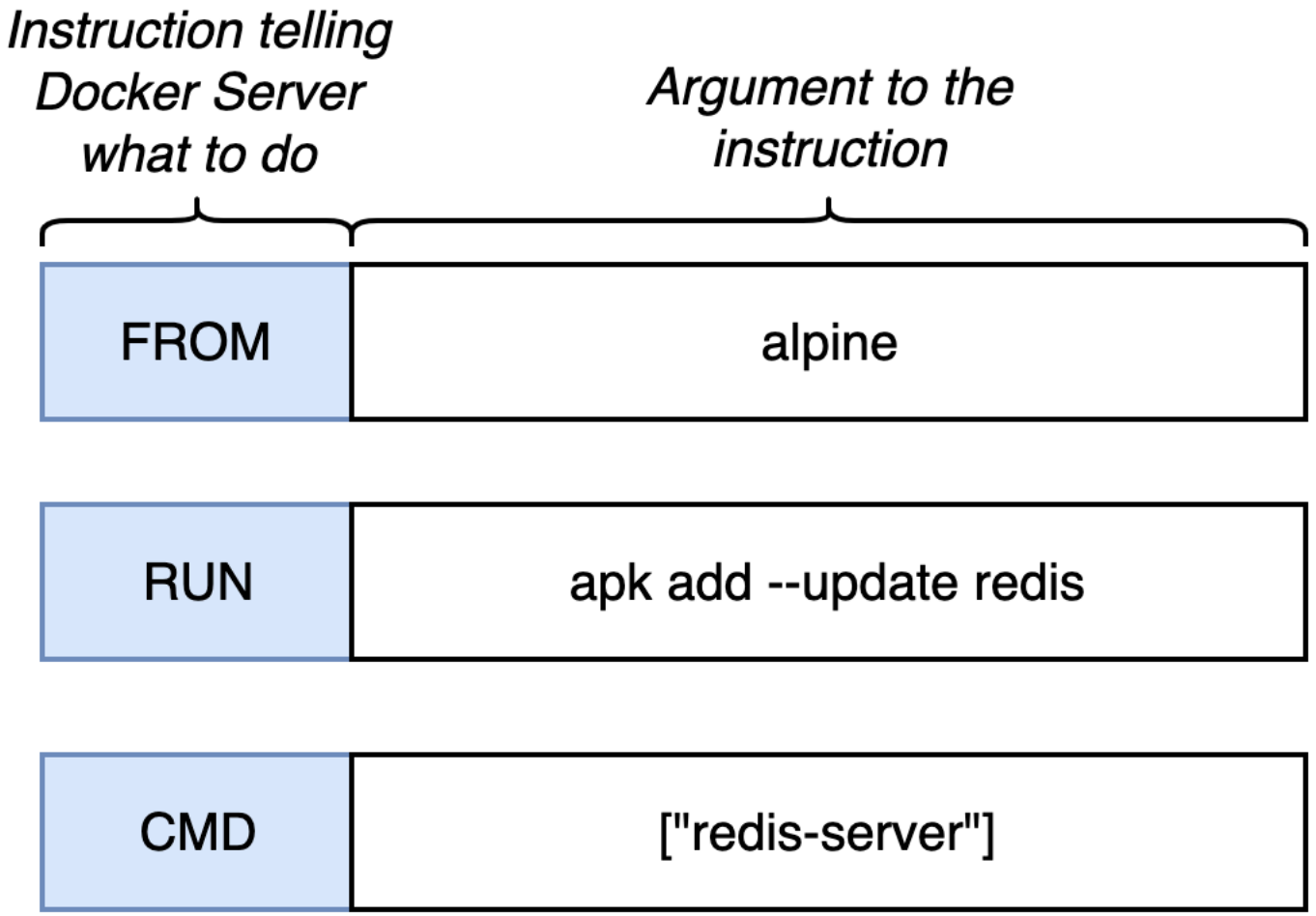
Dockerfile 建立流程



建立 Redis Image

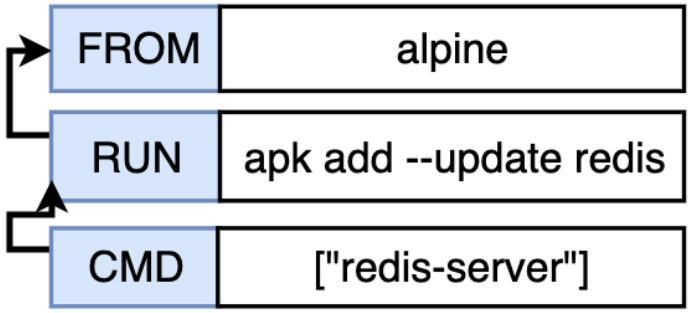


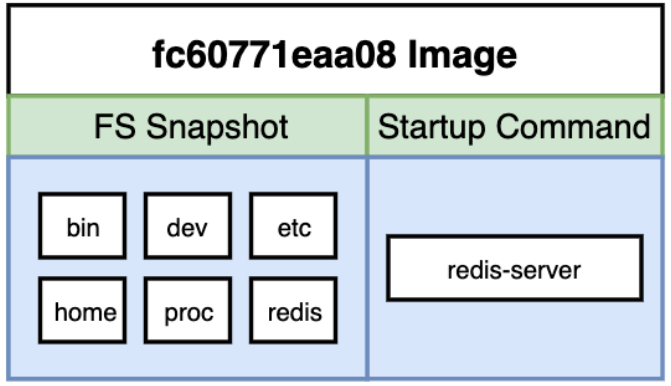
Docker File 範例

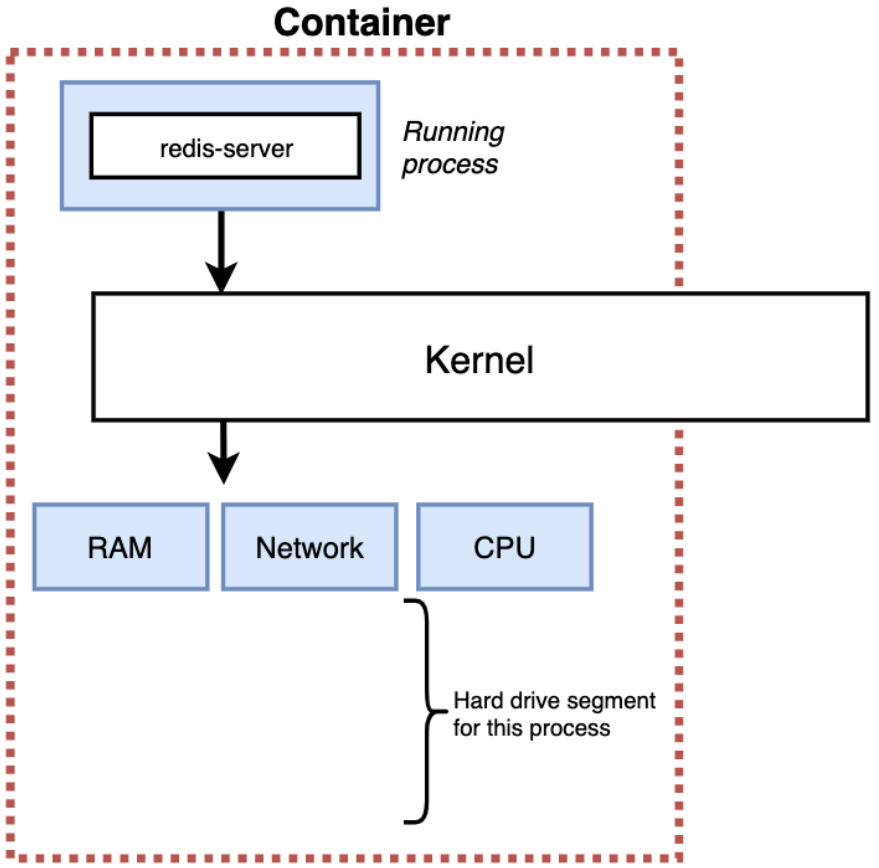


執行 docker build .

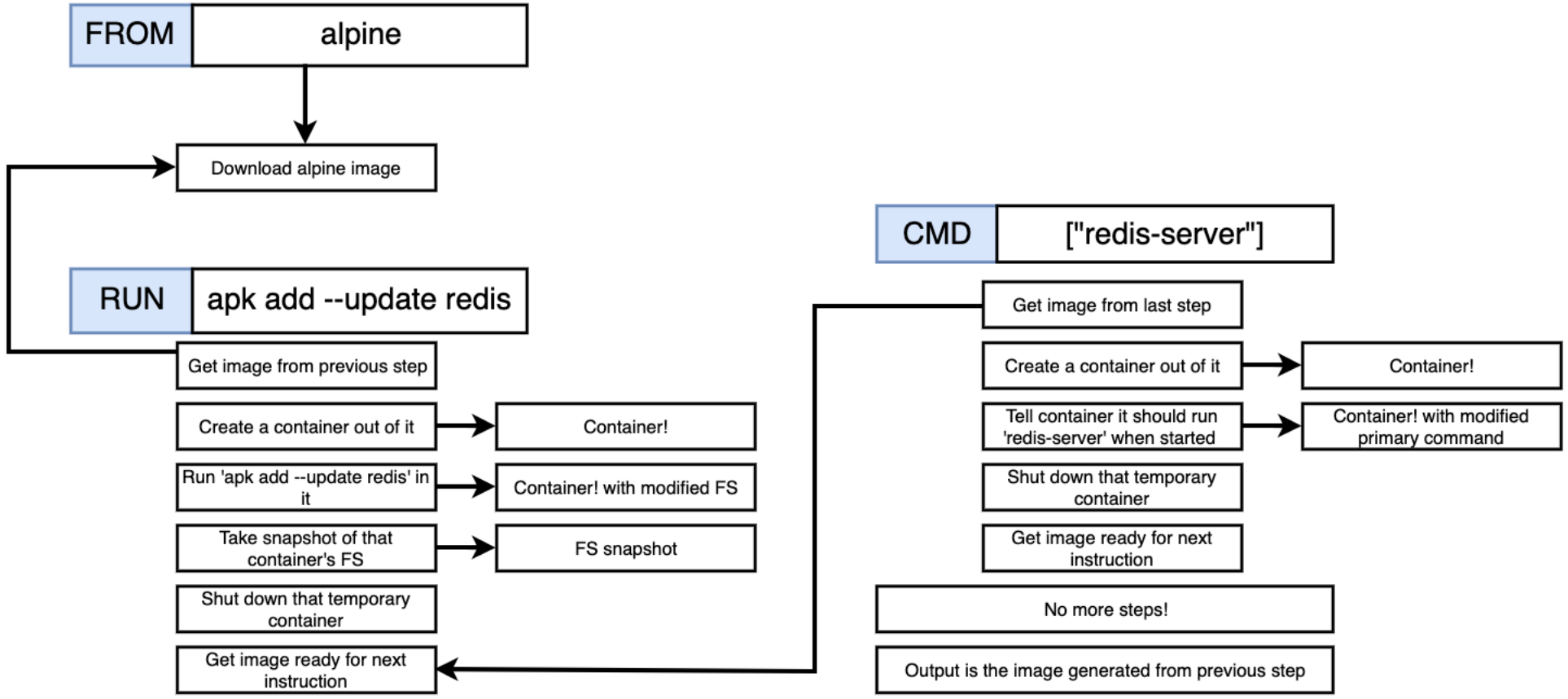
建立Image 流程


FROM
alpine
RUN
apk add --update redis
CMD
["redis-server"]


fc60771eaa08 Image
FS Snapshot
Startup Command
bin
dev
etc
home
proc
redis
redis-server


Container
redis-server
Running process
Kernel
RAM
Network
CPU
Hard drive segment for this process

流程總覽


FROM
alpine
Download alpine image
RUN
apk add --update redis
Get image from previous step
Create a container out of it
Container!
Run 'apk add --update redis' in it
Container! with modified FS
Take snapshot of that container's FS
FS snapshot
Shut down that temporary container
Get image ready for next instruction
CMD
["redis-server"]
Get image from last step
Create a container out of it
Container!
Tell container it should run 'redis-server' when started
Container! with modified primary command
Shut down that temporary container
Get image ready for next instruction
No more steps!
Output is the image generated from previous step

THANK YOU