虛擬機研究-docker

班級：資工二

姓名：鍾瑄

學號：110710520

[**一、** **docker是什麼？** 2](#_Toc43903317)

[**二、** **容器介紹** 3](#_Toc43903318)

[**三、** **docker與一般虛擬機之間的差別** 3](#_Toc43903319)

[**四、** **docker總架構圖** 4](#_Toc43903320)

[**（一）Docker Client** 5](#_Toc43903321)

[**（二）Docker Daemon** 5](#_Toc43903322)

[**Engine** 6](#_Toc43903323)

[**Job** 6](#_Toc43903324)

[**Docker Registry** 6](#_Toc43903325)

[**Graph Docker** 6](#_Toc43903326)

[**Driver** 6](#_Toc43903327)

[**Libcontainer** 8](#_Toc43903328)

[**Docker Container** 8](#_Toc43903329)

[五、 docker部分源碼解析 9](#_Toc43903330)

[（一）Docker 命令的 flag 参数解析 9](#_Toc43903331)

[（二）處理 flag 資訊並收集 Docker Client 的配置資訊 10](#_Toc43903332)

[**Main函數的源碼分析**： 10](#_Toc43903333)

[**六、** **docker安裝與使用** 11](#_Toc43903334)

[**(一)** **安裝須知** 11](#_Toc43903335)

[**(二)** **開始安裝 (此報告是以centos7 來實作)** 11](#_Toc43903336)

[七、參考資料： 1.什麼是 Docker？ 14](#_Toc43903337)

1. **docker是什麼？**

**(此部分是參考資料得出的內容)**

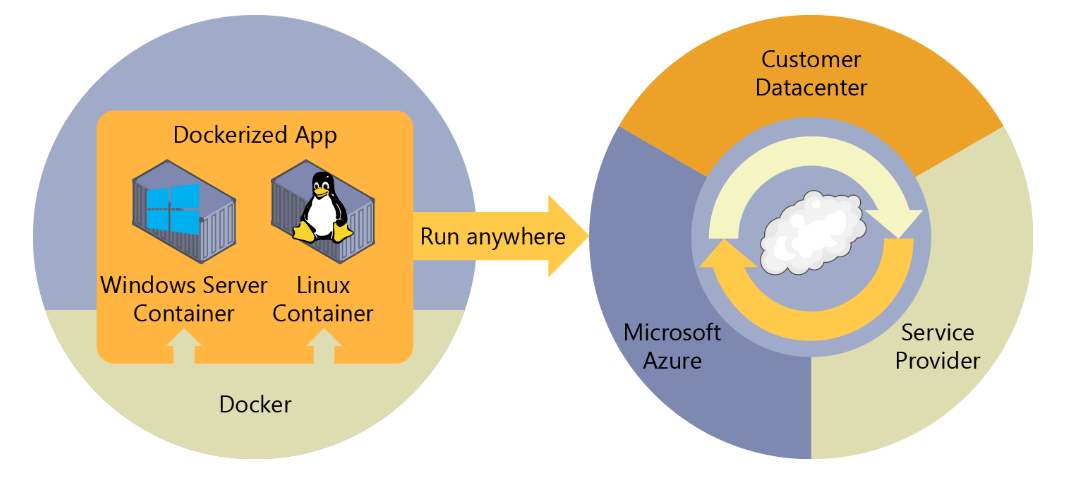
Docker 是一個開放原始碼軟體平台，用於開發、交付、執行應用。其允許用戶將基礎設備中的應用單獨分割出來，形成更小的顆粒（容器），從而提高速度。

 使用 Go 語言進行開發，基於 Linux內核的cgroup，namespace，以及AUFS類的UnionFS等技術，進行封裝隔離。

 能夠自動執行重複性任務，例如搭建和配置開發環境，讓開發人員能專注在真正重要的事情上。

 用戶可以方便地創建和使用容器，把自己的應用放入容器。容器還可以進行版本管理、複製、分享、修改，就像管理普通的代碼一樣。

 能將應用程式自動化部署為可攜式且可自足的容器，在雲端或內部部署上執行



[**圖：Docker 將容器部署在混合式雲端的所有圖層**](https://docs.microsoft.com/zh-tw/dotnet/architecture/containerized-lifecycle/what-is-docker)

***\* Go：****Google開發的程式語言*

***\* cgroup：****是Linux核心的一個功能，用來限制、控制與分離一個行程群組的資源（如CPU、記憶體、磁碟輸入輸出等）。*

***\* namespace：****是Linux內核的一項功能，它對內核資源進行分區，以使一組進程看到一組資源，而另一組進程看到另一組資源。*

***\* AUFS：****用於為Linux檔案系統實現聯合掛載*

***\* UnionFS：****用於​​Linux的文件系統服務，它為其他文件系統實現聯合掛載*

1. **容器介紹**

容器就是將軟體變成一個標準化單元，以用來開發、交付和部署。用簡單一點的說法來說的話，容器**就是一個存放東西的地方，就像書包可以裝文具、書本、水壺一樣。現在所說的容器存放的東西更偏向于應用比如網站、程式甚至是系統環境。**

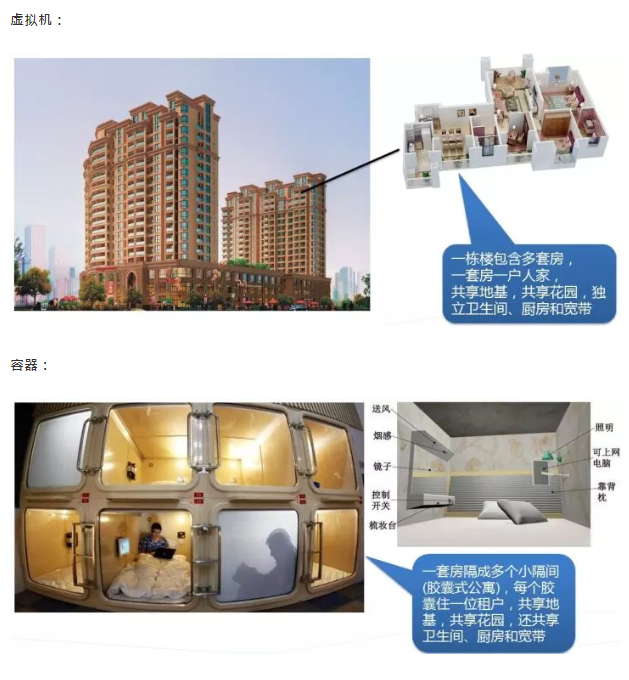
1. **docker與一般虛擬機之間的差別**

**(此部分是參考**<https://juejin.im/post/5b260ec26fb9a00e8e4b031a>**)**

Docker容器與虛擬機器類似，但原理上，容器是將作業系統層虛擬化，虛擬機器則是虛擬化硬體，所以容器更具有可攜式性且更能高效地利用伺服器。且多個容器可以在同一台機器上運行，共用作業系統內核，但各自獨立的在使用者空間中運行，而虛擬機則是將一台伺服器變成多台伺服器。允許多個虛擬機在一台機器上運行。每個虛擬機都包含一整套作業系統、一個或多個應用、必要的二進位檔案和資源，因此佔用的空間較容器來的大很多。

|  |  |  |
| --- | --- | --- |
| 特性 | 容器 | 虛擬機 |
| 啟動時間 | 秒 | 分鐘 |
| 硬碟使用 | MB | GB |
| 系統支持量 | 上千個 | 幾十個 |

[**表：容器與虛擬機的比較**](https://juejin.im/post/5b260ec26fb9a00e8e4b031a)

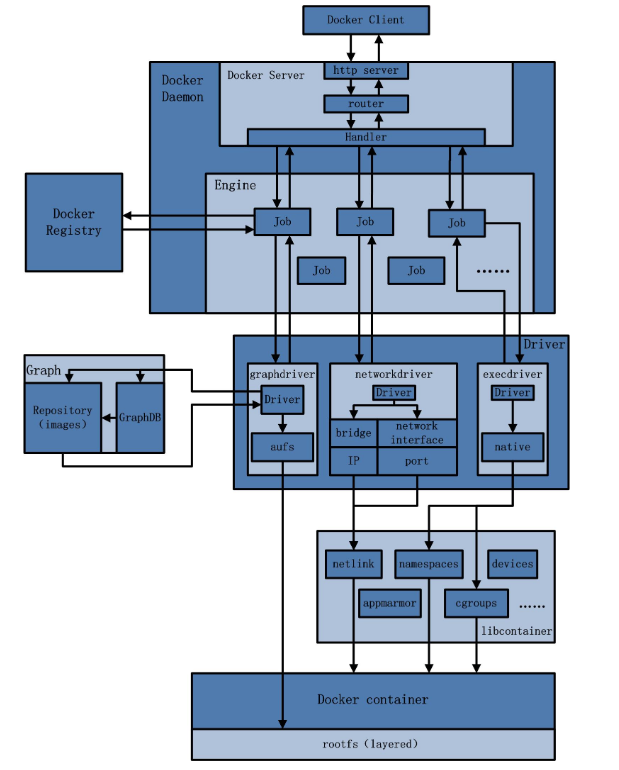


[**圖：虛擬機和容器比較的另類比喻**](https://juejin.im/post/5b260ec26fb9a00e8e4b031a)

※以下皆參考孫宏亮著的Docker 源碼分析，只有部分為自己統整的資料

1. **docker總架構圖**

Docker 並不像其他大型分散式系統那樣複雜。其主要的模組有: DockerC1ient、DockerDaemon、Docker Registry、Graph、Driver、libcontainer 以及 Docker Container。



**圖：docker總架構圖**

**（一）Docker Client**

Docker Client是與 Docker Daemon 建立通信的客戶端。用戶可以使用可執行檔 docker 作為Docker Client，發起眾多Docker容器的管理請求。Docker Client主要透過以下三種方式和 Docker Daemon 建立通信，分別為: tcp://host: port、unix://path \_ to \_ socket和fd://socketfd 。

**（二）Docker Daemon**

Docker Daemon是Docker架構中一個常駐在後臺的系統進程。所謂的"運行 Docker" ,即代表運行 Docker Daemon 。DockerDaemon 的作用主要有以下兩方面:

**1、接收並處理 Docker Client 發送的請求。**

**2、管理所有的 Docker 容器。**

Docker Daemon 運行時，會在後臺啟動一個 Server負責接收 Docker Client 發送

的請求;接收請求後，Server 通過路由與分發調度，找到相應的 Handler 來處理請求。其架構大致可以分為三部分: Docker Server、 Engine和 Job。

**Engine**

Engine是Docker架構中的運行引擎，同時也是 Docker 運行的核心模組。存儲著大量的容器資訊，同時管理著 Docker大部分Job的執行。所以Docker 中大部分任務的執行都需要 Engine 協助，並通過 Engine匹配相應的 Job完成執行。

**Job**

Job是Engine內部最基本的工作執行單元。 DockerDaemon完成的每一項工作都會呈現為一個 Job 。例如，在 Docker容器內部運行一個進程、創建一個新的容器、在網路上下載一個文檔，這都是一個 Job。

**Docker Registry**

Docker Registry是一個儲存容器鏡像( Docker Image) 的倉庫。容器鏡像是容器創建時用來初始化容器的檔案系統內容。其將大量的容器鏡像匯集在一起，並為分散的 Docker Daemon 提供鏡像服務。Docker 的運行過程中，有三種情況與 Docker Registry通信，分別為搜索鏡像、下載鏡像、上傳鏡像。這三種情況所對應的Job名稱分別為search、pull和push。

**Graph Docker**

Graph Docker是容器鏡像的保管者。Docker下載的鏡像和Docker構建的鏡像，都是由Graph 統一化管理。由於 Docker 支援多種不同的鏡像存儲方式，如 aufs、devicemapper、Btrfs等，故 Graph 對鏡像的儲存也會因以上種類而存在一些差異。

***\* Btrfs：****一種支援寫入時複製的檔案系統。*

***\* devicemapper****：設備映射器是Linux內核提供的框架，用於將物理塊設備映射到更高級別的虛擬塊設備。*

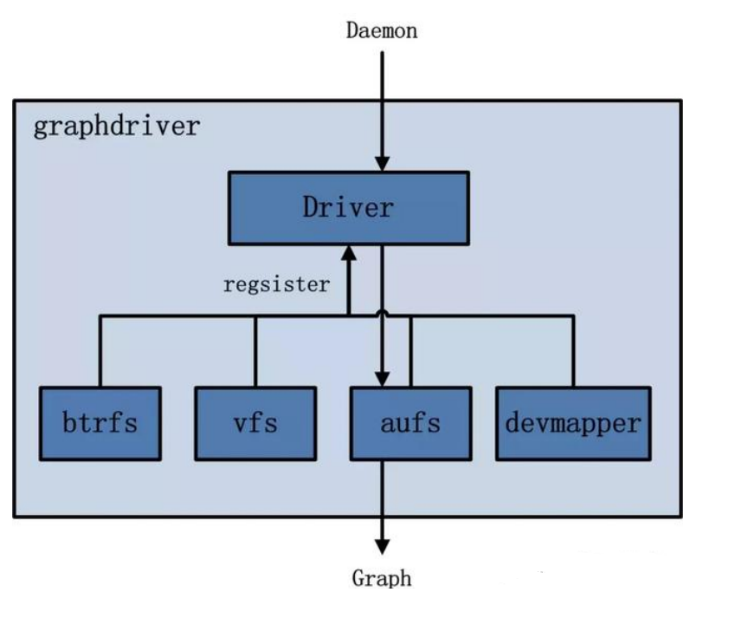
**Driver**

Driver Docker 架構中的驅動模組。通過 Driver 驅動，可以實現對容器運行環境的定制，定制的維度主要有網路環境、存儲方式以及容器執行方式。其實現可以分為以下三類驅動: graphdriver、networkdriver和execdriver。

1. **graphdriver：**主要用於容器鏡像的管理。其包括：

從遠端 Docker Registry上下載鏡像並進行儲存。

本地構建完鏡像後的儲存。

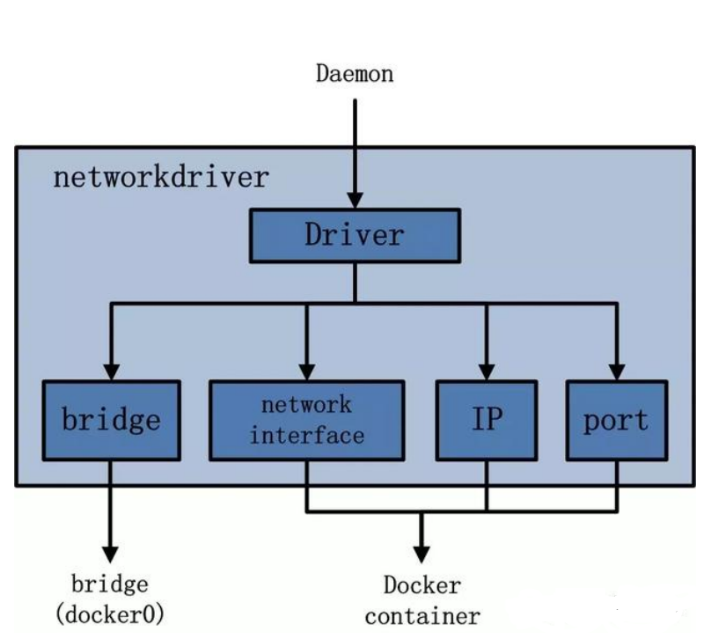
包含了四種系統(btrfs, vfs, aufs, devmapper)驅動Drive在Docker Danmon下完成註冊。

**圖：graphdriver架構圖**

1. **networkdriver：**完成容器網路環境的配置。其中包括:

 Docker 啟動時為 Docker 環境創建網橋。

 Docker 容器創建時為其創建專屬虛擬網卡設備。

 Docker 容器分配IP、埠並與宿主機做埠映射，設置容器防火牆策略等。

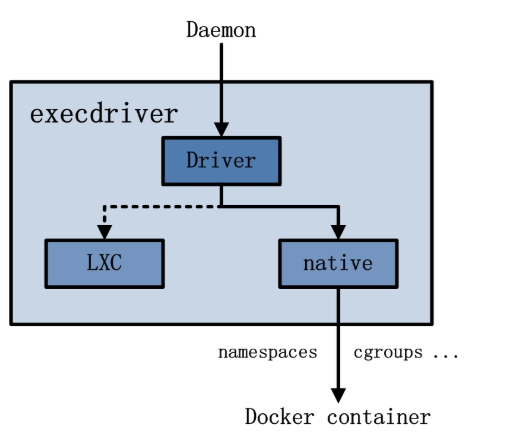
**圖：networkdriver架構圖**

1. **Execdriver：**為Docker容器的執行驅動。其負責：

創建容器運行命名空間。

容器資源使用的統計與限制。

器內部進程的真正運行等。



**圖：Execdriver架構圖**

**Libcontainer**

Libcontainer 是Docker架構中一個使用 Go 語言設計的庫，設計初衷是希望該庫可以不依靠任何依賴，可以直接訪問內核中與容器相關的 API。Docker 可以直接調用 Libcontainer 來操縱容器的 Namespace、Cgroups、Apparmor、網絡設備以及防火牆規則等。且Libcontainer 提供了一整套標準的接口來滿足上層對容器管理的需求。或者說 Libcontainer屏蔽了Docker上層對容器的直接管理。

**Docker Container**

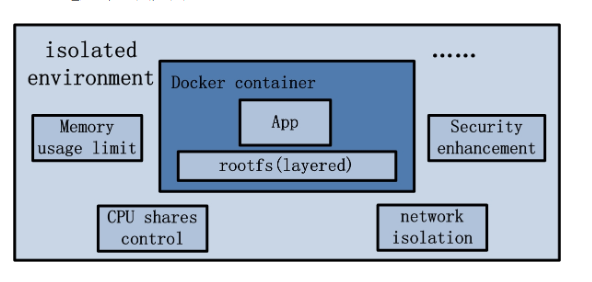
Docker Container是 Docker 架構中服務交付的最終體現形式。Docker 按照用戶的需求與指令，訂製相應的 Docker 容器：

用戶通過指定容器鏡像，使得 Docker 容器可以自定義 rootfs 等文件系統。

用戶通過指定計算資源的配額，使得 Docker 容器使用指定的計算資源。

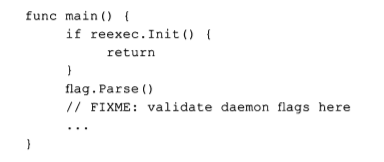
用戶通過配置網絡及其安全策略，使得 Docker 容器擁有獨立且安全的網絡環境。

用戶通過指定運行的命令，使得 Docker 容器執行指定的工作。

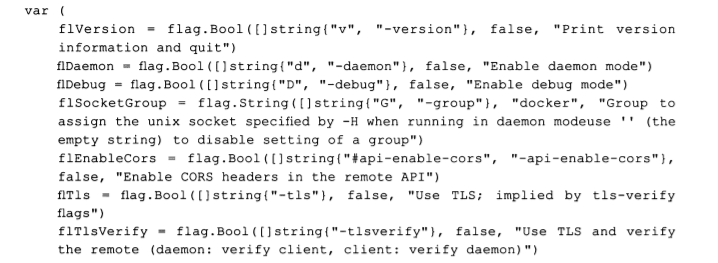
**圖：Docker Container架構圖**

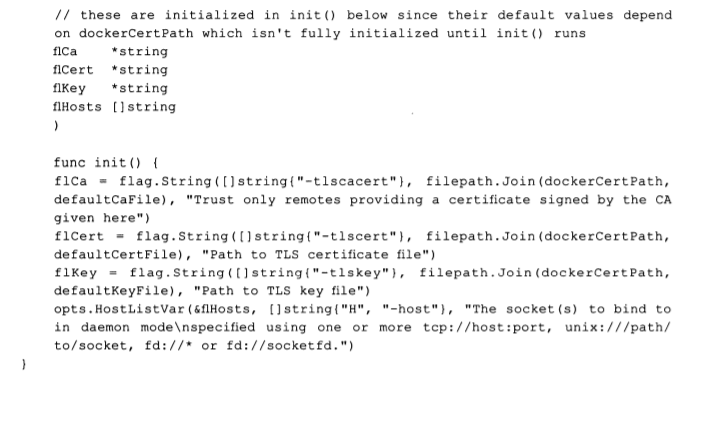
1. docker部分源碼解析

（一）Docker 命令的 flag 参数解析



首先判斷 reexec.Init( )，若為true，則直接退出運行，否則將繼續執行。由於在docker 運行之前沒有任何 Initializer 註冊，故該程式碼片段執行的返回值為false。reexec的作用是協調execdriver 與容器創建時 dockerinit 這兩者的關係。





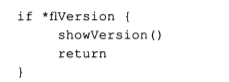
以上源碼展示了 Docker 如何定義 flag 參數，以及在 init 函數中實現部分 flag 參數的初始化。 Docker main 函數執行前，這些變數創建以及初始化工作已經全部完成。這裡涉及了Go 語言的一個特性，即 init 函數的執行。

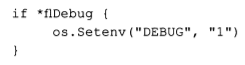
（二）處理 flag 資訊並收集 Docker Client 的配置資訊

處理的 flag 參數有: flVersion、flDebug、flDaemon、fltTlsVerify 以及 fltTls。

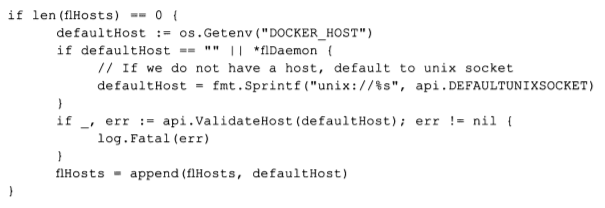
Docker Client 收集的配置資訊有: protoAddrParts (通過 fltHosts 參數獲得，作用是提供 Docker Client和Docker Server 的通信協議以及通信地址)、tlsConfig (通過一系列flag參數獲得，如 \*fltTls、\*flTlsVerify，作用是提供安全傳輸層協議的保障)。

**Main函數的源碼分析**：



此源碼代表著，若 Docker 發現flag參數 flVersion 為真，則說明Docker 使用者希望查看 Docker 的版本資訊。此時 Docker 調用showVersion( )顯示版本資訊，並從 main 函數退出;否則的話，繼續往下執行。

flDebug 參數為真的話，Docker通過os包中的Setenv函數創建一個名為 DEBUG環境變數，並將其值設為 "1"; 繼續往下執行。

flHosts的作用是為 Docker Client 提供所要連接的host物件，也就是為 Docker Sever 提供所要監聽的物件。首先判斷flHosts變數是否長度為0。如果是的話，則說明用戶並沒有傳入地址，此時 Docker 的策略為選用預設值。Docker 通過 os 包獲取名為 DOCKER HOST環境變數的值，將其賦值於defaultHost。若 defaultHost為空或者flDaemon為真，說明目前還沒有一個定義的 host 物件，則將其預設設置為unix socket，值為 api.DEFAULTUNIXSOCKET

該常量位於叢./docker/api/common.go ，值為 "/var/run!docker.sock"，故defaultHost 為"unix:///var/runldocker.sock"。驗證該 defaultHost 的合法性之後，將defaultHost 的值追加至flHost 的末尾，繼續往下執行。當然若 flHost 的長度不為0，則說明用戶已經指定位址，同樣繼續往下執行。

1. **docker安裝與使用**
2. **安裝須知**

以下是參考：<https://reurl.cc/NjmDL6>

以及docker官網：<https://docs.docker.com/engine/install/centos/>進行安裝作業

指令參考：<https://dwatow.github.io/2017/10-26-docker-commands/>

1. 需要有一台 64bit 的伺服器資源，大部分的image都是使用64位元的作業系統環境開始，也是 Docker 預設支援。

2. 選用 Linux 為 Host(註) 是因為這種情境下才是 Docker 最原始標準配備，其它環境(Mac, Windows)都是透過作業系統容器(VM)來模擬。

3. Linux kernel 的版本需大於 3.10 。

＊Image： Docker 在執行時需要一個基底的環境，與作業系統安裝時的映像檔有著差不多的概念，而由image建立起的容器稱為Container。

1. **開始安裝 (此報告是以centos7 來實作)**

1. 檢查環境是否滿足 Docker 的需求

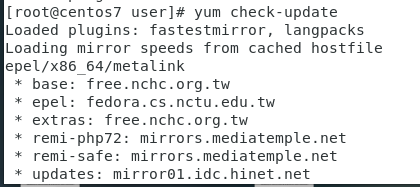


＊如果有大於3.10就可以繼續

2. 檢測有無安裝需要的工具： curl



3. 如果無顯示結果，執行yum install curl，如果有進行更新



4. 輸入curl -fsSL https://get.docker.com/ | sh 安裝docker

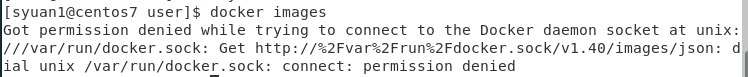
5. 如果想以非root用戶身份使用Docker，可以使用 sudo usermod -aG docker ‘your user’指令指派



這樣在user裡也可以使用docker



在非指定使用者則不能使用



6. 輸入 docker -v 確認正確執行



7. 開啟docker



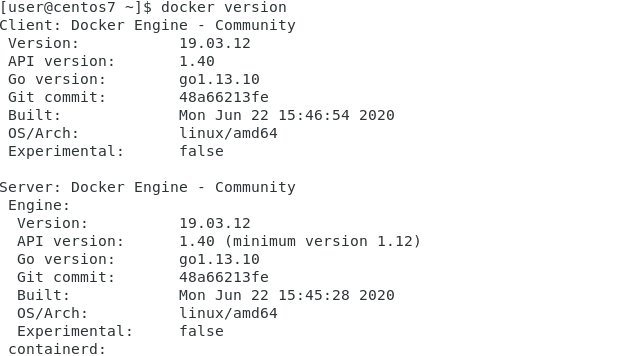
8. 創造名為Hello-World的image



9. docker images列出所有images



10. docker version 查看docker版本



七、參考資料：  
1.[什麼是 Docker？](https://docs.microsoft.com/zh-tw/dotnet/architecture/containerized-lifecycle/what-is-docker)

<https://docs.microsoft.com/zh-tw/dotnet/architecture/containerized-lifecycle/what-is-docker>  
2.[可能是把Docker的概念讲的最清楚的一篇文章https://juejin.im/post/5b260ec26fb9a00e8e4b031a](https://juejin.im/post/5b260ec26fb9a00e8e4b031a)  
3.[如何通俗解释Docker是什么？](https://www.zhihu.com/question/28300645)

<https://www.zhihu.com/question/28300645>  
4.[Docker原始碼分析（一）：Docker架構](https://www.itread01.com/content/1548105503.html)

<https://www.itread01.com/content/1548105503.html>  
5.[圖解 Docker 架構](https://kknews.cc/code/mp469e6.html)

<https://kknews.cc/code/mp469e6.html>  
6.[Docker 源碼分析](https://www.tenlong.com.tw/products/9787111510727)

<https://www.tenlong.com.tw/products/9787111510727>

7. 教你一次學會安裝 Docker 開始玩轉 Container 容器世界

<https://reurl.cc/NjmDL6>

8. 在CentOS上安裝Docker Engine

<https://docs.docker.com/engine/install/centos/>