何謂 Docker？

SECTION 01

手中拿著本書的讀者，可能對 Docker 抱有「好像很方便但不很瞭解」的印象。的確，Docker 難以簡單描述，但我們能夠概略地說明其要點。

Docker藏身於雲霧背後的真面目是？

「Docker」是什麼東西？

Docker 起初是伺服器工程師用於開發環境的軟體平台，但現在也廣泛用於正式環境、前端工程師的開發環境。因此，許多人都有「必須瞭解 Docker 才行！」的焦慮感，卻不曉得它是何方神聖。

即便向稍微熟悉的人請益，也僅是得到「容器技術……」、「鯨魚……」等回覆，沒有清楚描述相關細節，宛若藏身於雲霧的背後般，但我們想要知道的卻不是這樣的答案。

本書將在講解「何謂 Docker」的同時，說明其優勢與用法。當然，筆者會避免「鯨魚……」等藏身於雲霧背後般的說明，還請各位放心。

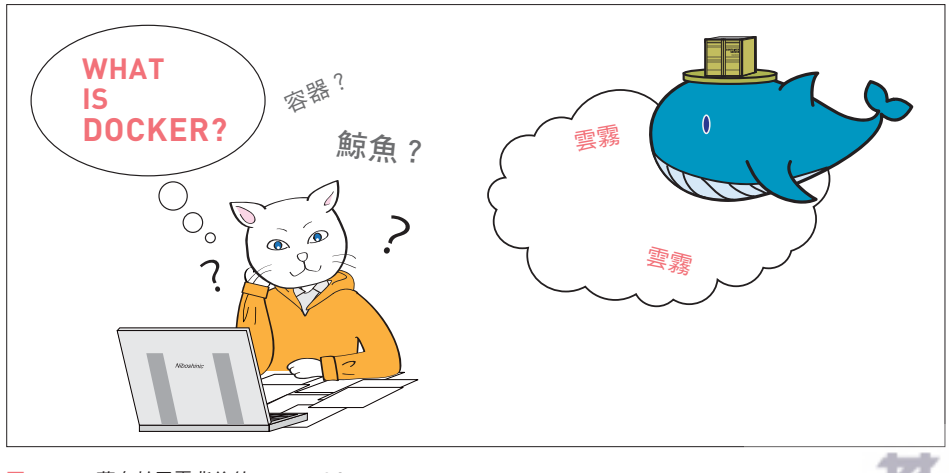

圖 1-1-1 藏身於雲霧背後的 Docker！？

Docker 是「可隔離資料、程式」的工具

簡單來說，Docker 是「可隔離資料、程式」的工具。它可用於伺服器、用戶端電腦，但現階段主要是用於伺服器。

電腦、伺服器可執行多個程式，各位在使用電腦時會同時執行 Word、Excel、郵件軟體。同理，伺服器也會同時執行 Apache※1、MySQL※2 等多個程式（軟體）。

Docker 可將多個程式、資料，隔離至各自獨立的環境。而且，除了程式、資料外，Docker 也可隔離（類）作業系統。

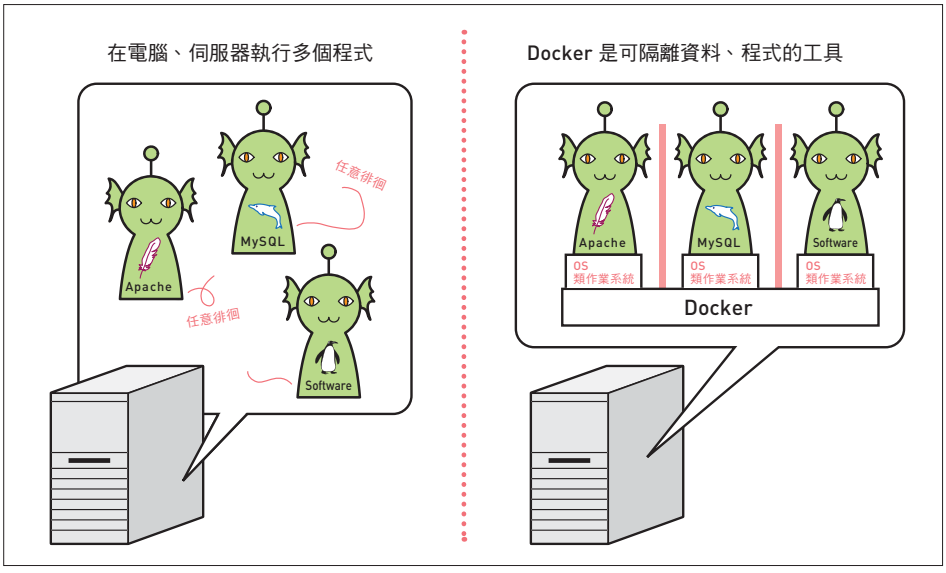

圖 1-1-2　Docker 是「可隔離資料、程式」的工具

容器（container）與 Docker Engine

具體而言，Docker 好比將電腦、伺服器上的環境，如 INABA、YODOKO 的置物櫃般分成多個空間，在獨立的置物空間中放入資料、程式。這個置物櫃就稱為容器（container），而使用容器的工具就是 Docker 。

在使用 Docker 之前，需要安裝 Docker 軟體（Docker Engine），安裝後便可建立、運行容器。

※1　提供網路伺服器功能的軟體，是具代表性的伺服器軟體。
※2　提供資料庫功能的軟體，其他知名的 DBMS 還有 PostgreSQL。

容器裡頭搭載類作業系統！

那麼，容器中是裝入什麼東西呢？

「容器中裝入什麼」的描述感覺好像是把什麼裝進空的容器，但「基本上不會使用」什麼都沒有裝載的「真・空容器」[※2]。雖然不是不存在，但我們幾乎不會接觸到真・空容器。

那麼，容器裡頭究竟如何呢？容器一定會搭載「類 Linux 系統」。如同居酒屋尚未點菜就會先出開胃小菜，搭載「類 Linux 系統」的容器通常是最小的容器。雖然稱之為「空容器」，裡頭其實還是有裝載東西。

不過，「類作業系統」這種拐彎抹角的說法，可能會令人覺得莫名其妙。由於這是近似 OS 的系統，但又不是完整的作業系統，所以才說得如此不乾脆。

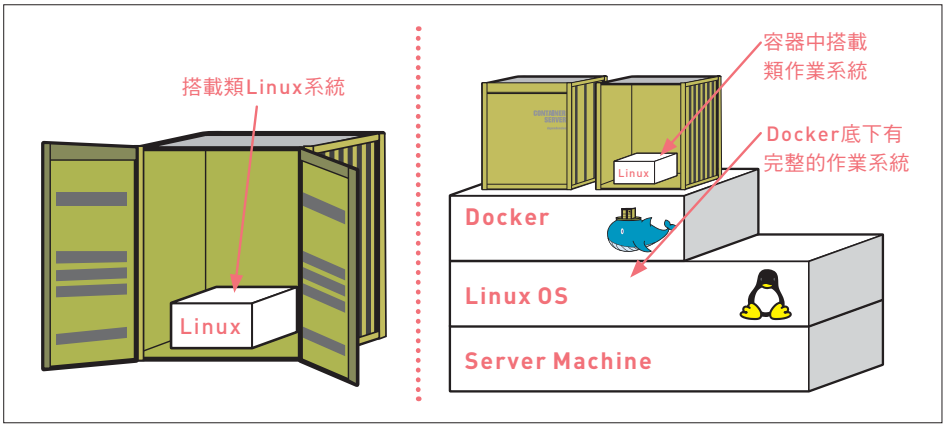

圖 2-1-2　容器中搭載類 Linux 系統

OS 究竟有什麼作用？

若要進一步描述的話，OS 扮演著將軟體等程式的命令傳達給硬體的角色。人類看來覺得縝密複雜的程式，對硬體（主機）來說卻是粗略籠統的命令。人類能夠從一句話判斷多個動作，但硬體缺乏自主判斷、視情況變通的能力，僅會聽命行事而已，一舉一動都需要人類的指示。

例如，看著桌子上的橘子說道：「請吃橘子。」人類會毫不猶豫地行動，但對象是硬體的話，「請拿起桌子右上方的橘子，剝開外層的果皮後，食用裡頭的果肉。」必須下達具體的指示才行。程式編寫的內容只有「請吃橘子」，後續需要 OS 轉成硬體容易理解的指示。

※2　可用 scratch 映像檔建立空容器，但新手通常不會用到。

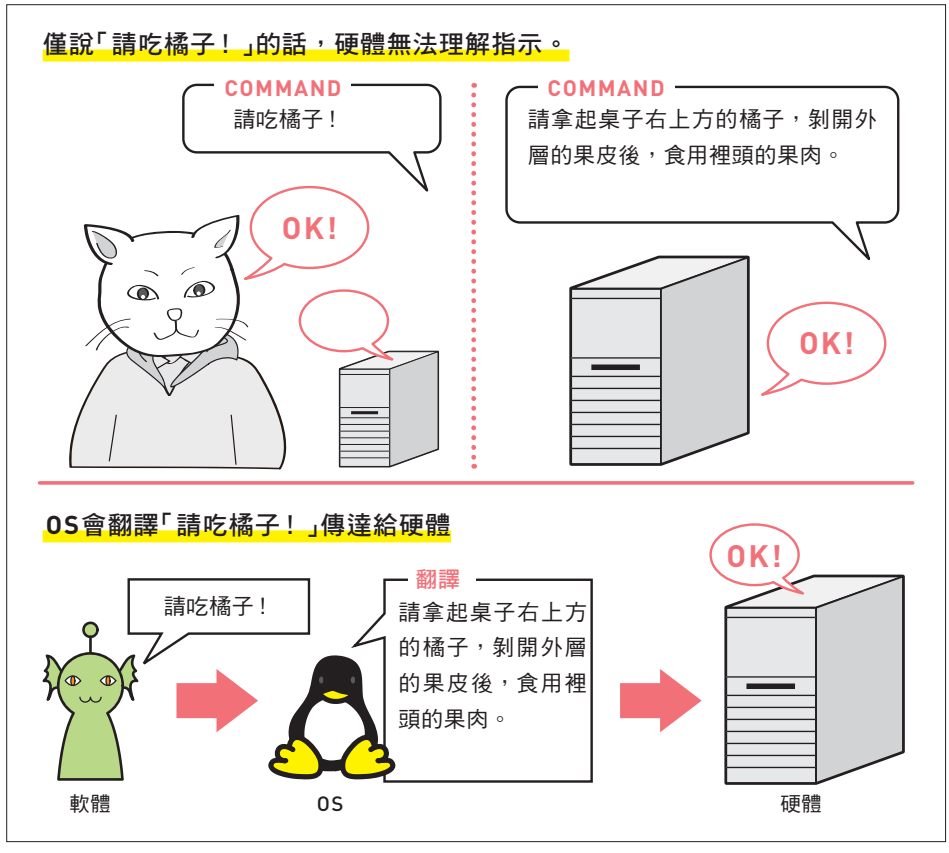

圖 2-1-3　OS 的作用

瞭解 OS 的作用後，回頭繼續討論容器。

請見**圖 2-1-2**，明明 Docker Engine 已經有底層的 Linux 系統，容器中卻又搭載（類）Linux 系統，呈現奇妙的構成。

然而，這是 Docker 的重要特徵之一。

OS 本來就是由「核心部分（kernel）」與「外殼部分（shell）」[※3] 所構成。外殼部分將來自程式的訊息傳達給核心部分，再由核心部分操作硬體。

而 Docker 會完全隔離容器，底層 Linux 系統的外殼部分無法接收容器中的程式命令。因此，容器得搭載 OS 的外殼部分，接收程式的命令後再傳給底層的核心部分。

※3　外殼部分可接收程式的命令，並向程式傳達核心部分的執行結果。例如，接收鍵盤輸入的內容，於螢幕顯示繕打的文字。「外殼部分」的套裝軟體即為所謂的發行版（distribution），如常見的 Red Hat、CentOS、Ubuntu 等。一般幾乎不會只用到 Linux 核心，而是會與發行版搭配使用，所以「Linux 系統使用 Red Hat」等，描述時會直接稱呼發行版的名稱。

進一步裝載軟體（程式）後，搭配組合無窮無盡。如**圖 2-2-10** 所示，光是裝載官方提供的容器映像檔 Apache（網路伺服器軟體）、MySQL（資料庫管理軟體），就存在多種類作業系統 ※10 與版本的搭配組合。

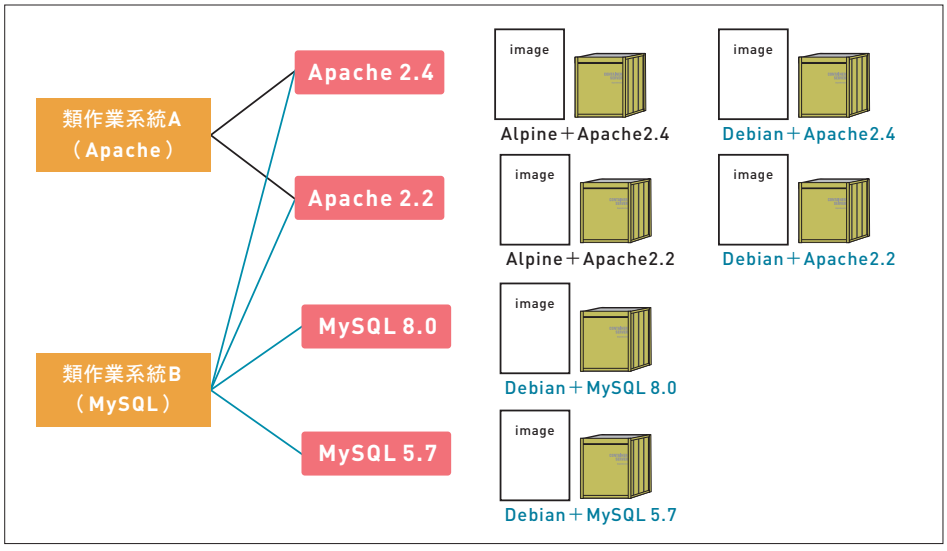

圖 2-2-10　類作業系統與軟體存在不同版本的搭配組合

除了 Apache、MySQL 外，nginx（網路伺服器軟體）、Sendmail（郵件軟體）、PostgreSQL（資料庫管理軟體）等開源軟體，Docker Hub 也提供了大部分的映像檔。

如何選擇安全的容器映像檔

存在如此眾多的種類，反而不曉得該選擇哪種容器映像檔。如前所述，任誰都能夠上傳至 Docker Hub 公開，當中也可能有不安全的映像檔。

下面就來討論選擇時的重點。

使用官方提供的映像檔

首先，容器映像檔有許多官方釋出的檔案，Docker 官網、建立管理軟體的企業團體都有提供映像檔。若有自己想要使用的映像檔，選擇官方檔案是既安全又方便。

不過，有些軟體會限定容器中的（類）作業系統使用特定的 OS、版本，無法自由地選擇映像檔。使用時，需要自行多加留意。

※10　Alpine、Debian 也是具代表性的 Linux 發行版，占用的電腦資源皆小，Apache、MySQL 的官方軟體也可搭配這兩種發行版。

SECTION 03

Docker 容器的生命週期與資料儲存

本節要說明 Docker 容器的使用流程。其實，容器不是「長期愛護使用」而是「建立用完即捨棄」。是否理解這件事，將會影響自己能否熟練容器技術。

Docker 容器是建立用完即捨棄

討論容器技術的時候，肯定會談到「容器的壽命與生命週期」。這是因為容器具有「建立用完即捨棄」的特性。「建立」姑且不論，「捨棄」可能會讓人感到不對勁。

如前所述，Docker 可簡單建立裝載軟體的容器。因此，並非不斷更新、愛護使用同一個容器，而會另外建立裝載新版軟體的新容器。

換言之，我們會不斷更換新的容器。

容器的使用一般是預設同時執行多個容器，所以不可能一一更新眾多的容器。容器具有簡單建立的特性，維護時卻採取一一更新的方式，只會讓工作事倍功半而已。比起直接重新建立，一一維護更新反而缺乏效率。

因此，我們會直接捨棄舊的容器，重新以映像檔建立新的容器。這樣的做法比較省時省力。

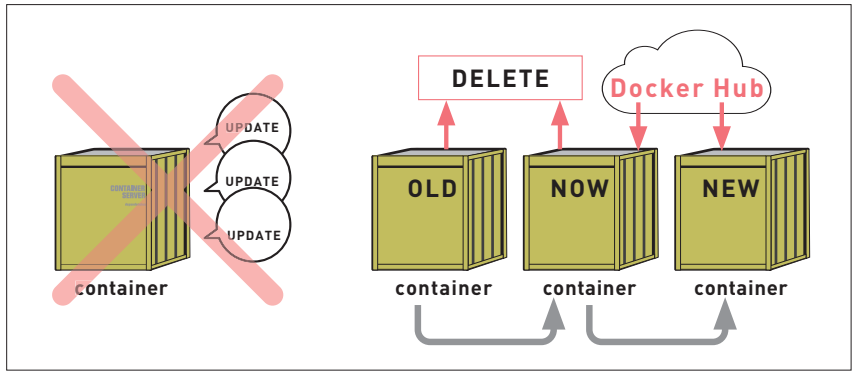

圖 2-3-1 容器是建立用完即捨棄

如上所述，容器的「建立」➡「啟動」➡「停用」➡「捨棄」➡「建立」……，這一連串的流程稱為「容器的生命週期」。

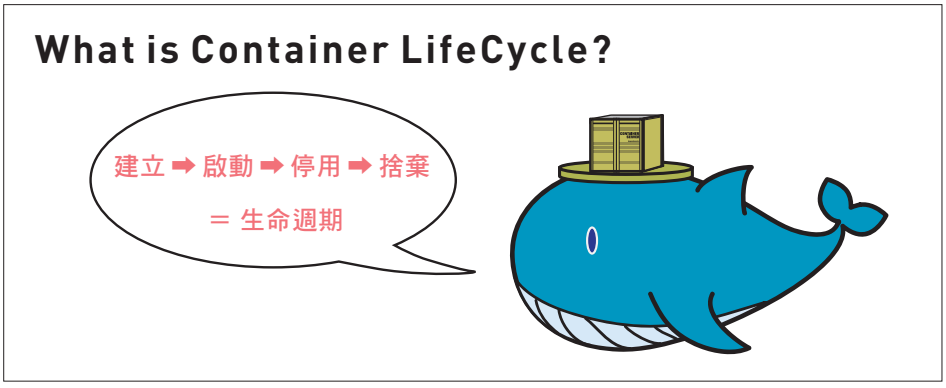

圖 2-3-2　容器的生命週期

資料儲存

容器捨棄後，裡頭的資料會如何呢？

容器捨棄後，因為檔案存於容器裡頭，資料理所當然會跟著消失。資料消失就麻煩了。因此，大多會掛載 Docker 實體主機（電腦）的硬碟，將資料存放於該硬碟。

掛載（mounting）是指「可讀寫的連接狀態」，如同在平常使用的用戶端電腦插入外接 USB、HDD，Docker 的容器也可連接實體主機的磁碟來讀寫資料。

如此一來，即便捨棄容器，資料會存於安全的外部，不會跟著消失不見。即便 Docker 本身發生問題，資料也能夠安然無恙。為了預防電腦發生故障的情況，我們會將資料存於外接硬碟，這樣來想掛載會比較容易理解。

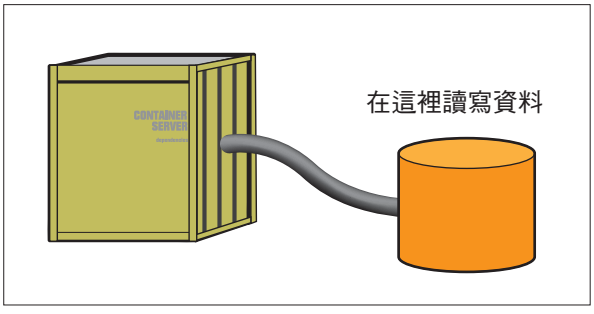

圖 2-3-3
掛載至實體主機儲存資料

執行結果

```
apa000ex2
```

Step 5 以「rm」指令刪除「apa000ex2」容器

執行「rm」指令，刪除「apa000ex2」容器。

輸入的內容

```
docker rm apa000ex2
```

執行結果

```
apa000ex2
```

Step 6 以指令加上參數，確認容器的刪除情況

以「ps」指令加上「-a」參數來執行，確認「apa000ex2」容器的刪除情況。若清單未顯示「apa000ex1」的內容，表示容器正常刪除。

輸入的內容

```
docker ps -a
```

執行結果

```
CONTAINER ID  IMAGE  COMMAND  CREATED  STATUS  PORTS  NAMES
```

本節的動手實作就到這邊。在 Step 3 有順利以瀏覽器訪問嗎？

若本身有架設網頁的經驗，不妨建置簡單的 HTML 檔，建置方式會在第 6 章詳述。此外，在桌面版 Docker 的畫面，也可確認容器、映像檔的狀態，確認方式請見附錄。

SECTION 05

熟練建立容器

經過前面兩節的練習，你應該已經大致瞭解了建立容器的流程。為了更加熟練建立容器，本節就來練習啟動各種類型的容器。

各種類型的容器

容器存在許多類型，堪稱有多少種軟體就有多少種容器。除了 Apache 的容器外，本節也會練習建立其他的容器。雖說如此，也不需要想得太困難，總之就是反覆建立、捨棄。已經相當熟練的人，可直接跳到 4-6 節繼續閱讀。

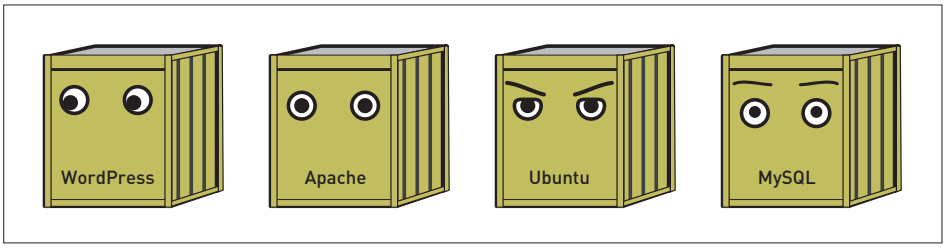

圖 4-5-1 容器存在許多類型

除了 Apache 外，還有其他具代表性的常見容器。後面介紹的映像檔，全部都是官網的檔案。這裡以 nginx 和 MySQL 來練習，若讀完本書後還有餘裕，不妨挑戰其他類型的容器。

在 Docker 做的事情與在容器內部做的事情宛若親子關係

「在容器內部做的事情」與在 Docker 做的事情，讀者可能難以理解其中的差異，下面就來簡單整理不同之處。

在 Docker 執行的操作與在容器內部執行的操作

在母親 Docker Engine 執行的操作，包含開始結束 Docker Engine 本身、設定網路或者磁碟、顯示運行中容器的清單等，是有關管理整個容器的操作。

如同前面的操作，建立或者啟動停用容器、下載建立容器用的映像檔等，使用 docker 指令下達命令。另一方面，在容器內部執行的操作，包含對容器新增軟體、執行停止內部軟體、變更設定、容器內部間複製移轉檔案、刪除檔案等。

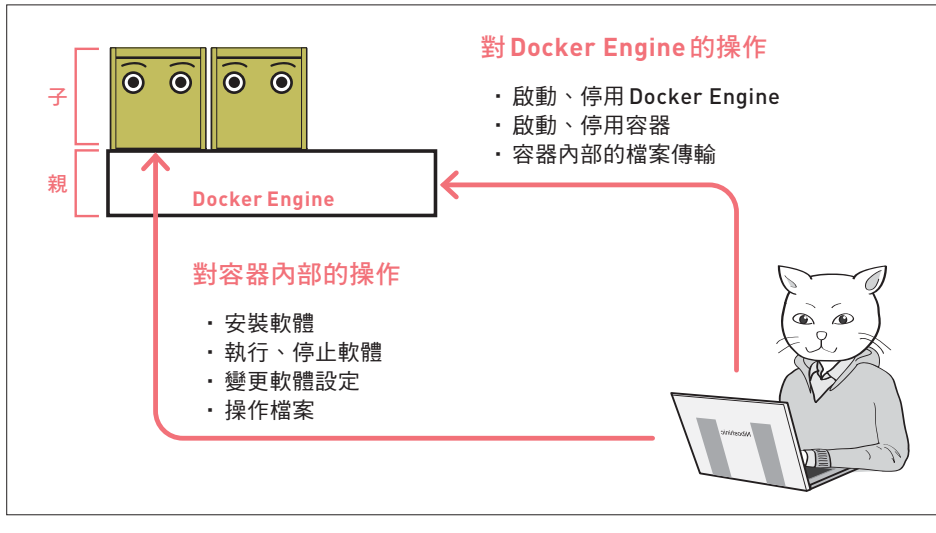

圖 6-5-4　Docker Engine 與容器內部的操作

Docker 與容器可能要用不同的語法（語言）

如前所述，Docker 和容器宛若親子卻是不一樣的存在，根據情況可能要用不同的語法（語言）。

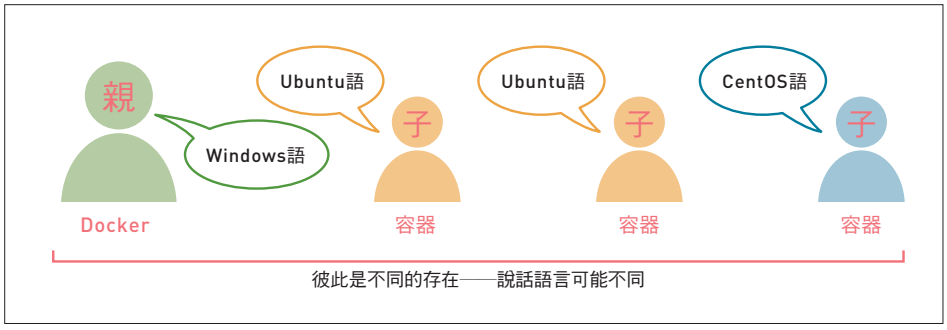

圖 6-5-5　Docker 與容器的說話語言可能不同

使用前面的桌面版 Docker※24 時，是以 Windows、Mac 的語法（語言）對母親 Docker Engine 下達命令。由於兩系統的 Docker 指令彼此通用，有些人可能訝異：Windows、Mac 使用的指令竟然一樣！

在安裝桌面版 Docker 的時候，由於只需要啟動安裝程式，並不曉得內部是怎麼下達命令，所以可能不好理解，但 Widnows 和 Linux 其實是以不同的語法來安裝 Docker。而且，Linux 有 CentOS、Ubunu 等不同的發行版※25，各發行版的語法又不盡相同。尤其，安裝指令※26 等幾個命令，在 Red Hat 系列※27 與 Devian 系列※28 的語法截然不同。

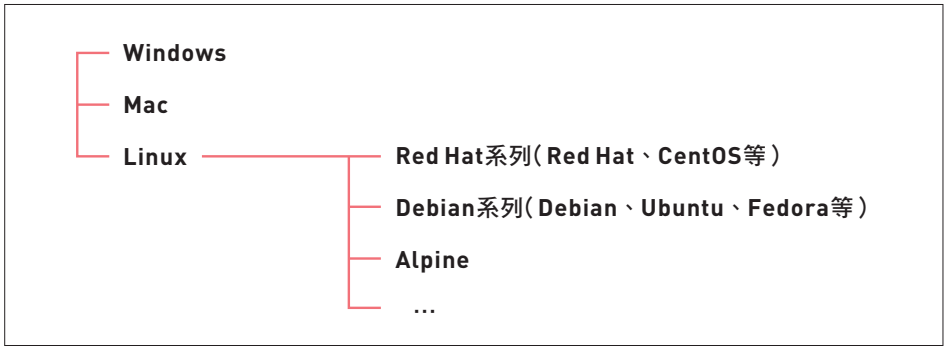

圖 6-5-6　語法（語言）的種類

※24　若是桌面版的前身 Toolbox 版，需要 Docker 官方提供的虛擬環境，命令得採用 Linux 的語法，且不在命令提示字元、終端機操作。
※25　請見 3-1 節的 P.50 專欄。
※26　因為軟體套件管理系統不同。
※27　Red Hat Enterprise Linux、CentOS 等。
※28　Debian、Ubuntu、Fedora 等。

這與容器有什麼關聯呢？容器搭載了類作業系統，根據安裝哪種 Linux 類作業系統，容器內部使用的指令會有若干差異。換言之，若容器 A 搭載 Debian 系列、容器 B 搭載 Red Hat 系列的話，即便兩個容器承載於相同 Docker，命令容器內部的語法仍會不一樣。

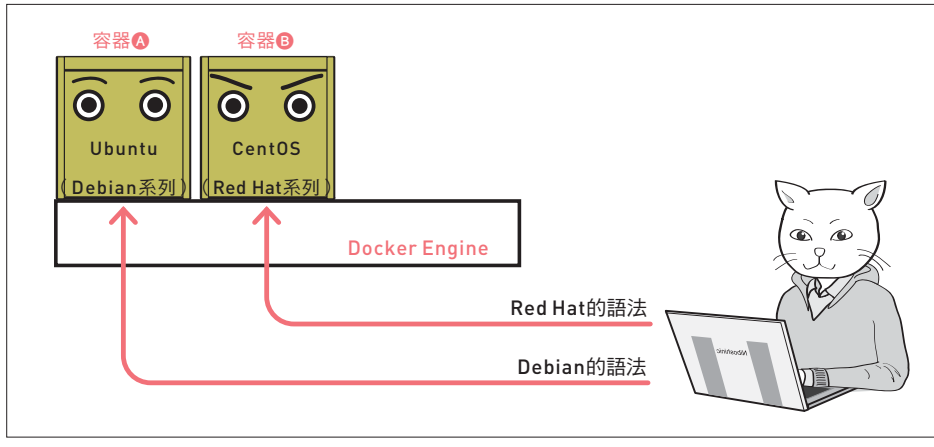

圖 6-5-7　命令容器內部的語法會因作業系統而異。

更複雜的是，若安裝 Linux 版 Docker Engine 的底層電腦是 CentOS（Red Hat 系列），而容器 A 是 Ubuntu（Debian 系列）、容器 B 是 Alpine 的話，親子之間的語法各自不同，整個非常混亂。

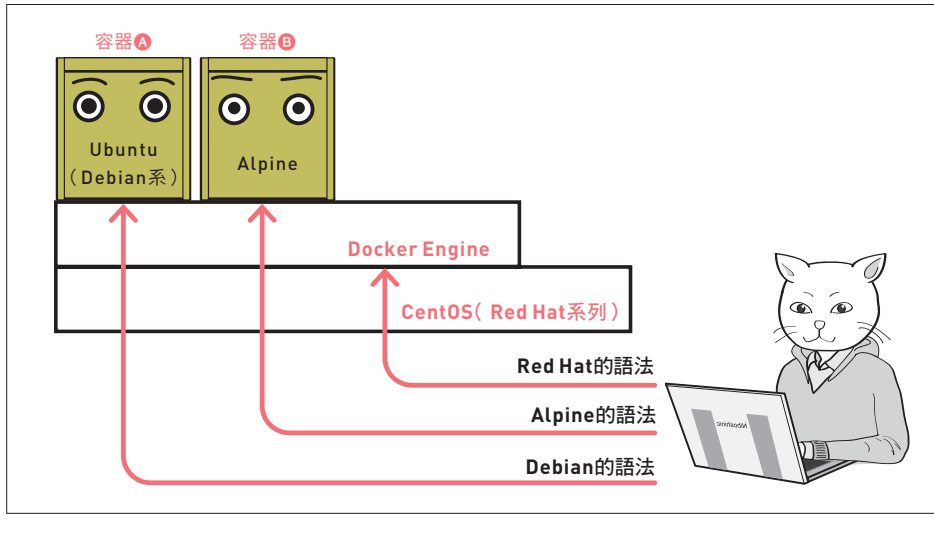

圖 6-5-8　底層作業系統的語法也有可能不同。