1

探索虛擬化

在現今競爭激烈且快速變化的世界中，要讓企業及組織處於領先地位的話，便需要依賴高可靠性、高成本效益以及可無限擴充的 IT 基礎架構，並且企業及組織必須要適應不斷變化的客戶需求，以便達到快速失敗、快速學習最後達成革命性的創新。同時，由於硬體成本不斷下降，企業及組織的重點投資項目已經轉移至充份利用現有基礎架構，或者是減少基礎架構的預算開銷為主，這表示必須在現有的 IT 環境上運作更多的應用程式或服務。

虛擬化基礎架構的出現，有效解決過往基礎架構的難題並且滿足現代化企業所有的 IT 需求。因為，在虛擬化基礎架構中，可以提供運算、儲存、網路等硬體資源及基礎架構管理平台，並且透過虛擬化技術有效活化硬體資源使用率、降低資料中心維運成本、有效節省資料中心內機櫃空間、達成應用程式高可用性、無限的可擴充性、容錯移轉及負載平衡等。

下列為虛擬化技術特色功能的優點項目列舉：

- 虛擬化管理程序是個軟體，當你在建構 IT 基礎架構時可以在單台實體伺服器上進行安裝，後續便可以在虛擬化平台上運作許多台**虛擬主機（Virtual Machine, VM）**。
- 虛擬化平台不僅可以運作多台 VM 虛擬主機，同時也將多個儲存設備的儲存資源彙整後，形成單一且巨大的虛擬儲存資源，並且這些資源可以依照需求提供給需要的 VM 虛擬主機。
- 透過虛擬化平台，可以讓你在 VM 虛擬主機當中運作不同類型及版本的**作業系統（Operating Systems, OS）**，為企業及組織提供混合運算的優勢。
- 企業及組織透過虛擬化技術，為資料中心建構集中式的主機及管理 IT 基礎架構。

了解虛擬化技術種類

目前，各種虛擬化技術已經發展成熟，接下來將要討論虛擬化技術種類，以及底層基礎架構如何提供各種隔離層級，最後演變成目前新興的容器化應用模式。

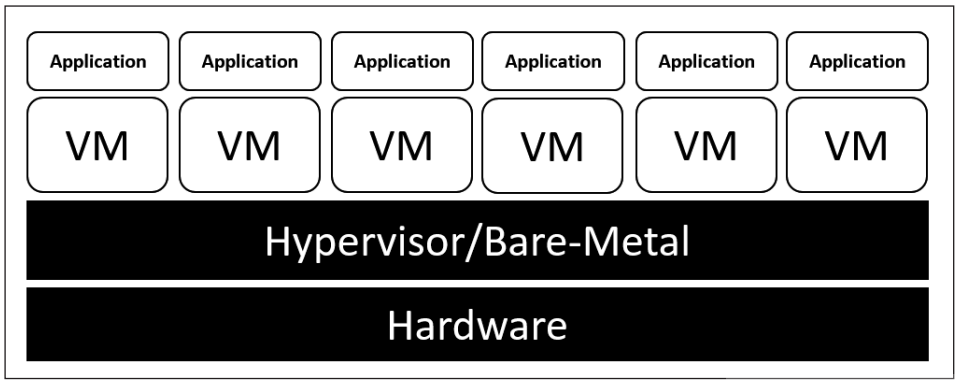
硬體 / 平台 / 伺服器虛擬化

在虛擬化技術浪潮風行之前，一台硬體伺服器只能承載一個作業系統並運作多個應用程式，然而企業及組織的營運環境中需要多租用戶及應用程式隔離機制，所以在一台硬體伺服器中便無法同時運作多個應用程式。虛擬化技術的出現，讓單台硬體伺服器透過伺服器虛擬化技術，能夠同時在單台硬體伺服器上運作多台 VM 虛擬主機及客體作業系統，並且每台 VM 虛擬主機之間可以完全互相隔離，同時每台 VM 虛擬主機的 CPU 及記憶體資源，可以依照企業及組織的營運需求進行調配。

在現代化公有雲及私有雲平台中，伺服器虛擬化技術的 Hypervisor 虛擬化管理程序運作機制內，將會透過 **VMM（Virtual Machine Manager）**管理 VM 虛擬主機的虛擬硬體資源。每台實體伺服器都會安裝作業系統稱為 Host OS，在虛擬化平台上運作多台 VM 虛擬主機，每台 VM 虛擬主機安裝的作業系統稱為 Guest OS，在伺服器虛擬化平台上由 Host OS，管理底層 CPU 處理器及記憶體等硬體資源，並且將這些硬體資源調度給其上運作的 VM 虛擬主機使用。

此外，企業及組織透過伺服器虛擬化技術可以提供混合運算環境，也就是可以在 Hyper-V 虛擬化平台上，同時安裝及運作 Linux 和 Windows 作業系統（如 Windows 8.1 或 Windows 10）的 VM 虛擬主機，甚至是 Windows Server 作業系統。目前市場上，主流的伺服器虛擬化產品有 VMware vSphere ESXi、Citrix XenServer 及 Microsoft Hyper-V。

簡言之，伺服器虛擬化平台的運作架構如下圖所示：

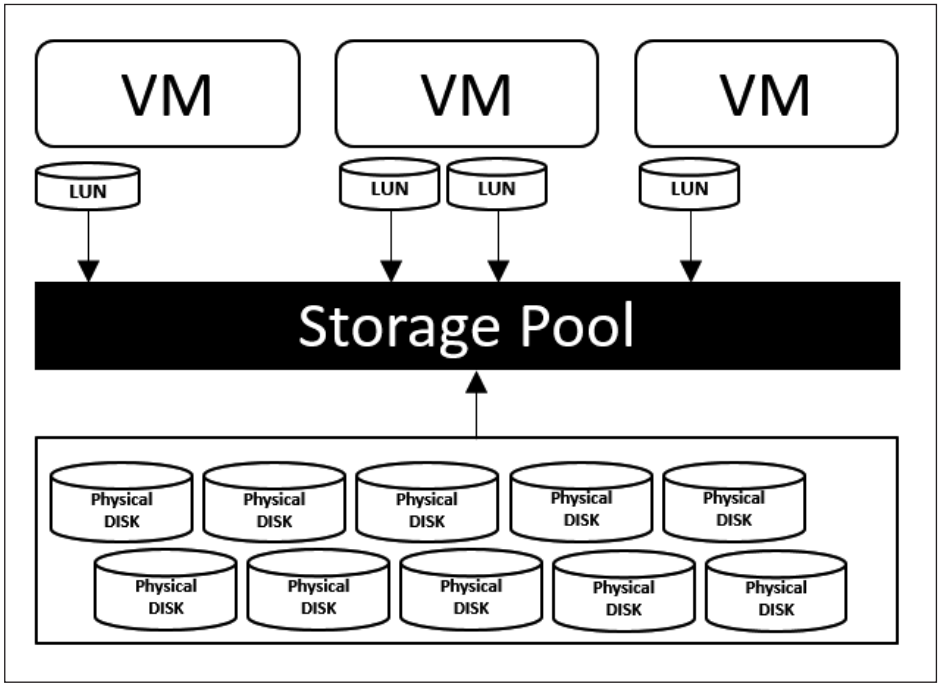


儲存虛擬化

儲存虛擬化（Storage Virtualization），將多個儲存資源彙整後提供單一大型的儲存空間，並且可以從單個控制台進行集中式管理。同時，儲存虛擬化也提供許多管理優勢，例如，管理備份、歸檔、依照需求自動分配儲存資源等。

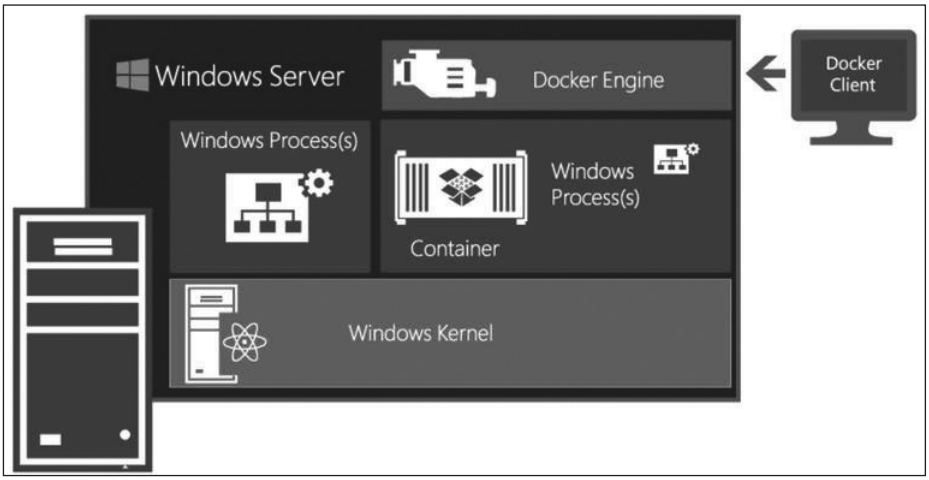
舉例來說，預設情況下 Microsoft Azure VM 虛擬主機運作環境中，將會包含兩個用於儲存用途的硬碟，但是可以根據企業及組織的需求，在幾分鐘之內（僅限於 VM 虛擬主機層級）將任意數量的硬碟加入至 VM 虛擬主機中。因此，能夠即時為 VM 虛擬主機帶來可擴充性及更好的資源使用率，並且只要根據資源的使用情況進行付費即可。

儲存虛擬化運作架構如下圖所示：



簡介 Windows Server Container

在新版 Windows Server 2016 雲端作業系統中，微軟推出作業系統虛擬化解決方案 Windows Server Container，它可以建立輕量級的執行程序並且互相隔離，這個輕量級執行程序具備作業系統、執行程序、檔案系統、網路功能等。從技術上的角度來看，Windows Server Container 與 Linux Container 非常類似，唯一的差別在於處理容器的底層核心及工作負載。如下圖所示，為 Windows Server Container 運作架構示意圖：



技術背景

下列項目為 Windows Server Container 的技術背景簡述：

- 微軟研究團隊發起 **Drawbridge** 專案計畫，將容器技術帶入到 Windows Server 生態系統當中。
- 在 2014 年時，微軟開始與 Docker 合作將 Docker 技術導入 Windows 當中。
- 在 2015 年 7 月時，微軟加入 **OCI（Open Container Initiative）**組織，以便擴大對容器的標準、格式、運作架構等的影響力。
- 在 2015 年時，Microsoft Azure 技術長 Mark Russinovich，在議程中發表 Windows 運作 Docker 容器技術。
- 在 2015 年 10 月時，微軟宣佈在 Windows Server 2016 TP3 技術預覽版本，以及在 Windows 10 作業系統版本中支援 Docker 容器技術。

4. 接著，請在容器中啟動 **Redis Server**。預設情況下，啟動 Redis Server 將會使用 6379 連接埠並等待連接，管理人員可以使用「**Ctrl + C**」組合鍵來停止 Redis Server：

```
cd redis
.\redis-server.exe
```

上述指令執行後，指令執行結果輸出顯示如下：

```
PS C:\> cd .\redis\
PS C:\redis> .\redis-server.exe
[4392] 01 Sep 03:20:13.026 # Warning: no config file specified, using the default config. In order to specify a config f
ile use C:\redis\redis-server.exe /path/to/redis.conf

                Redis 3.0.503 (00000000/0) 64 bit

                Running in standalone mode
                Port: 6379
                PID: 4392

                      http://redis.io

[4392] 01 Sep 03:20:13.030 # Server started, Redis version 3.0.503
[4392] 01 Sep 03:20:13.030 * The server is now ready to accept connections on port 6379
```

現在，我們已經完成在運作中的容器執行 Redis Server 的目的。因為，在 Docker 的運作架構中，並不允許將運作中的容器進行轉換並產生容器映像檔的動作，所以我們必須要把運作中的容器「停止」運作，才能夠順利執行轉換並產生出一個新容器映像檔的動作。請按下「**Ctrl + C**」組合鍵停止 Redis Server，此時將會回到容器的 PowerShell 指令模式。

5. 請執行 exit 指令。一旦管理人員在互動模式的容器中，執行 exit 指令後容器便會離開互動模式並停止運作：

```
exit
```

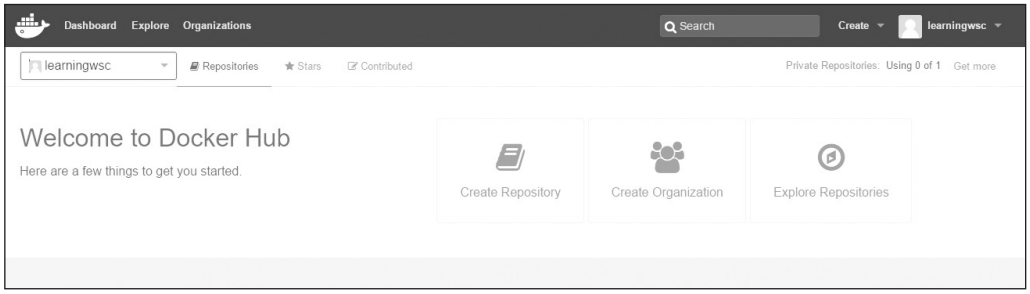
6. 接著，請執行 docker ps 指令，列出 Windows 容器主機上所有狀態為「運作中」的容器，由於我們已經將剛才運作的容器停止，所以應該不會看到才對：

```
docker ps
```

7. 管理人員只要執行 docker ps -a 指令，即可列出 Windows 容器主機上「所有」運作狀態的容器資訊：

```
docker ps -a
```

當你填入相關申請資料後按下【**Sign Up**】鈕，Docker Hub 網站便會將使用者帳戶啟用資訊，透過 E-Mail 傳送至你填寫申請資料所留下的 E-Mail 位址，請點選使用者帳戶啟用 E-Mail 內容中的啟動連結，將會觸發開啟瀏覽器然後導向至 Docker Hub 網站登入頁面，接著鍵入申請資料所設定的使用者帳戶名稱及密碼後，便可以按下【**Login**】鈕進行登入 Docker Hub 網站的動作，成功登入後便會導向至 Docker Hub 內的個人帳戶主頁：



請執行下列指令，建立新的 Windows Container 容器映像檔。當然，你也可以自行建立其他容器映像檔，但建議採用 `dockerid/imagename` 命名格式較佳：

```
docker commit --change='WORKDIR /redis' --change='CMD
powershell .\redis-server.exe' -c "EXPOSE 6379" aa636dc079e3
learningwsc/mycacheserver:latest
```

在本書實作環境中，我們將剛才建立且正確運作的 **Redis Cache Server 容器**，產生出另一個新的容器映像檔，並且採用 `learningwsc/mycacheserver:latest` 命名格式，其中 `learningwsc` 便是剛才註冊的 Docker Hub 使用者帳號（Docker ID），而 `mycacheserver` 則是容器映像檔名稱標籤為 `:latest`。

Docker 預設採用的標籤便是 Latest，所以上述的容器映像檔標籤設定雖然是多餘的，但這是建議推送容器映像檔至存放庫的良好使用習慣。

順利產生新的容器映像檔之後，便可以準備把容器映像檔推送到 Docker Hub 公開容器映像檔存放庫上。請執行下列指令，透過剛才所註冊的 Docker Hub 使用者帳號（Docker ID），以及使用者密碼登入 Docker Hub 網站：

```
docker login --username learningwsc --password ********
```

Docker 磁碟區

Docker 磁碟區（Docker Volumes）並非在 UnionFS 檔案系統當中，而是在容器主機上以資料夾的方式存在，因此實際上使用的是容器主機的檔案系統。同時，我們還可以讓容器同時使用多個容器主機中的資料夾，如此一來，容器便可以依據儲存需求進行不同路徑的存放，因此當我們希望讓容器運作關聯式資料庫服務（如 SQL Server 或 MySQL）時，只要確保容器能夠運作關聯式資料庫服務即可，而無須把資料量巨大的資料庫倒入至容器當中，是讓容器透過儲存磁碟區的方式，直接讀取容器主機中存放的資料庫檔案即可。本節將說明如何建立及管理 Docker 磁碟區運作機制。

在 Docker 運作環境中，提供 Docker 指令 `docker volume` 以便管理人員能夠建立及管理 Docker 磁碟區。在開始進行實作演練之前，我們可以先透過 `docker volume` 指令列出所有可用的參數資訊：

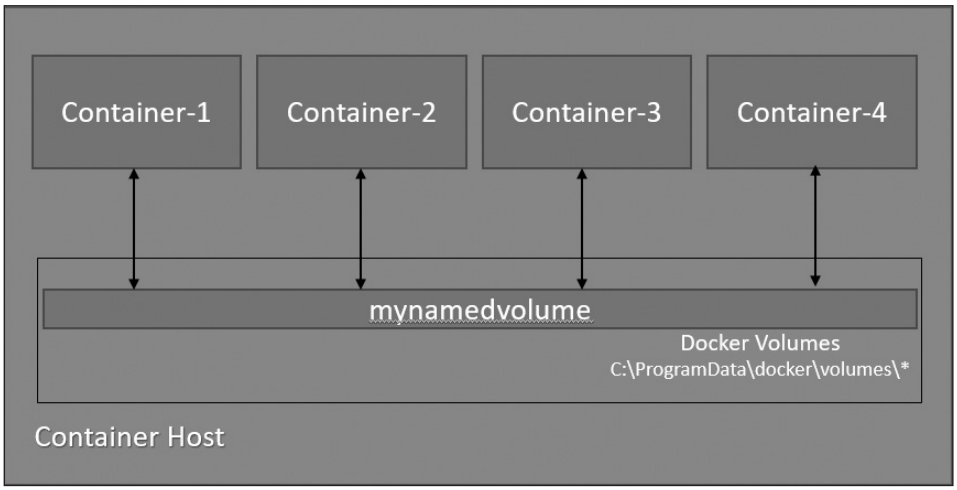
```
PS C:\> docker volume

Usage:  docker volume COMMAND

Manage volumes

Options:
      --help   Print usage

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
```

管理人員若希望列出所有容器使用的磁碟區資訊時，請執行下列指令：

```
docker volume ls
```

上述指令執行後，指令執行結果輸出顯示如下：

```
PS C:\> docker volume ls
DRIVER              VOLUME NAME
local               52f7c01262685d69ae19b47cb17a2e31ebf1765cfbf11c4f1c3a2f532449bf6a
local               5347e0fc517f4861a71c86106709ae666ac906630288ba6a5e0c53a1d1bbe20e
local               5971e14f63e5c012251a931268021d12b76e41cf17eb53611e11fd5cc6069d28
local               72899fadef700c3920c5ca0a348e1809904b710809a0802819d7a75632c44e7c
local               9fff346d788cd308c2c7cac258ca7f12eba08a6c6e5a5e3825f708acc387d931
local               f4edc81013114c1bc7a4e0367470c72e1e426f72e8fda616829cde6048bdf433
```



當管理人員使用 Docker 指令搭配 -v 參數建立及掛載磁碟區後，其他容器便可以使用 --volumes-from 參數掛載同一個磁碟區。值得注意的是，使用 --volumes-from 參數的容器必須處於「運作中」（Running）的狀態才行，舉例來說，在下列 Docker 指令中使用 --volumes-from 參數搭配指定的容器 ID，讓兩個容器同時掛載使用同一個磁碟區，因此不管哪個容器對磁碟區進行修改或更新的動作，另一個容器也都看得見所有的變更。

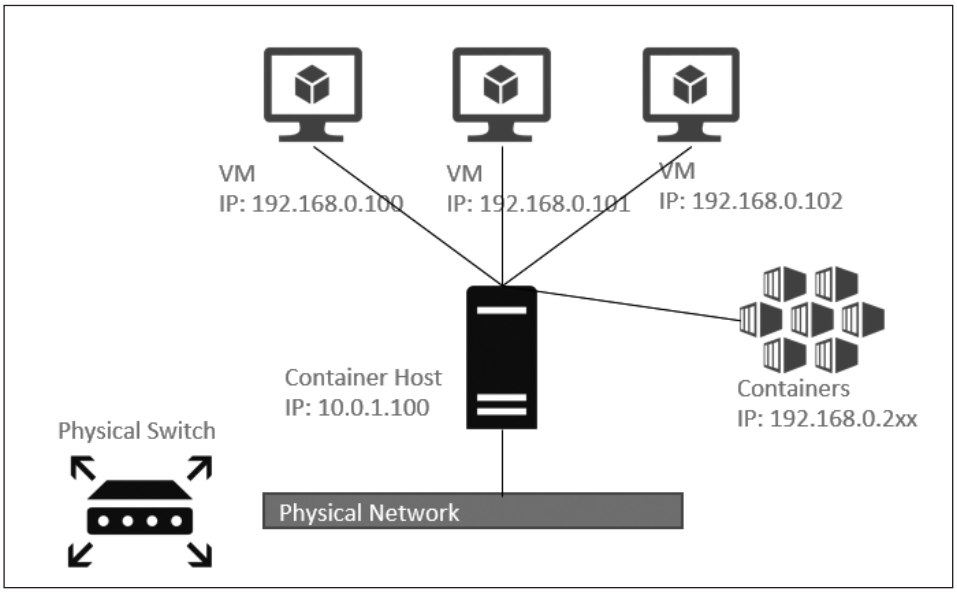
```
docker run -it --volumes-from cd8f3b00ccc5 microsoft/windowsservercore
```

將 Music Store 資料儲存至磁碟區

在本書實作環境中，Music Store 音樂網站可以讓使用者建立新的音樂專輯相簿，並且使用者可以上傳音樂專輯封面圖檔。然而，我們尚未幫 Music Store 音樂網站建立持續性儲存資源，來儲存使用者所上傳的音樂專輯封面圖檔。

現在，讓我們幫 Music Store 音樂網站，建立專用的持續性儲存資源來儲存使用者所上傳的音樂專輯封面圖檔。本節實作演練的範例檔案，可以至 GitHub 網站取得本書完整範例程式碼：https://github.com/vishwanathsrikanth/learningwsc/tree/master/chapter6/musicstore-volumes

因此，無論管理人員採用 Windows Server 容器或 Hyper-V 容器運作環境，虛擬網路的運作機制和使用方式與傳統 VM 虛擬主機是相同的（如下圖所示）：



事實上，當管理人員建立及運作容器時，預設便會使用 `--network nat` 參數為容器指定使用 NAT 虛擬網路環境。因此，即便管理人員在建立及運作容器時未指定虛擬網路類型，容器便會直接使用預設的 NAT 虛擬網路環境：

```
docker run -it --network nat microsoft/windowsservercore cmd
```

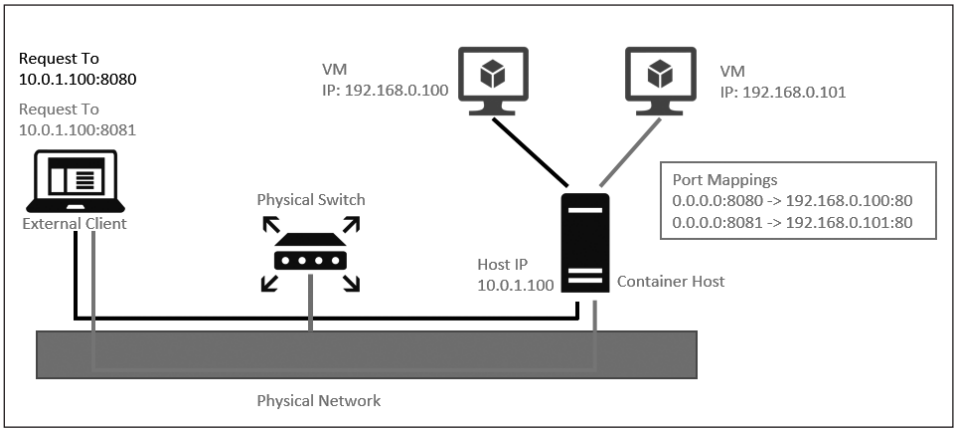
在下列 Docker 指令中，我們將會建立一個新的容器虛擬網路環境，其中子網路為 `172.20.81.0/24` 且預設閘道為 `172.20.81.1`，名稱為 `mynetwork`：

```
docker network create -d nat --subnet=172.20.81.0/24
--gateway=172.20.81.1 mynetwork
```

在 Windows Server 容器運作環境中，每個容器都會透過容器主機的網路堆疊，並採用已經隔離的虛擬網路環境與外部網路通訊，而 Hyper-V 容器運作環境，每個 Hyper-V 容器已經封裝在最佳化的 VM 虛擬主機當中，透過 vmNIC 及 Hyper-V 容器本身的網路堆疊與外部網路通訊。在上述 Docker 指令中，指定的預設閘道被分配給 WinNAT 及容器主機的 TCP/IP 網路堆疊，後續建立及運作的容器再透過這兩項機制與外部網路通訊（如下圖所示）：

固定連接埠對應

解決容器與外部網路環境通訊的問題後，接著便是外部網路如何與容器主機上運作的容器互相通訊。當管理人員建立及運作容器時，應該在容器主機與容器之間建立連接埠對應的動作，也就是整合容器主機的**連接埠位址轉譯（Port Address Translation, PAT）**機制，將外部網路存取容器的 TCP/UDP 連接埠進行轉換作業，以便外部網路順利存取容器所提供的服務（如下圖所示）：



當管理人員鍵入下列指令執行建立及運作容器的動作時，在容器順利運作後便會將容器主機的連接埠 80，對應到容器連接埠 80 的動作，舉例來說，該容器運作 IIS 網頁伺服器服務，所以當外部網路嘗試存取容器主機的連接埠 80 時，不管是採用 IP 位址或 DNS 名稱解析的方式，都會透過 PAT 連接埠位址轉譯機制導向給容器：

```
docker run -it -p 80:80 microsoft/windowsservercore cmd
```

容器主機及容器的連接埠對應關係可以不同，例如，下列指令便會將容器主機的連接埠 8082，對應至容器的連接埠 80：

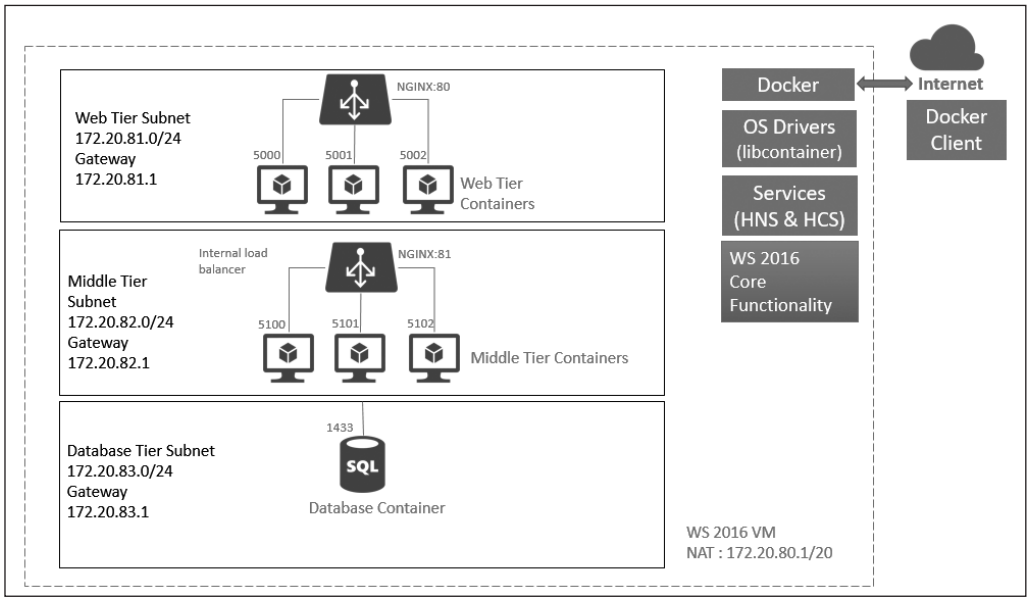
```
docker run -it -p 8082:80 microsoft/windowsservercore cmd
```

管理人員在執行 Docker 指令時，可以透過 -p 參數來指定容器主機所要對應的連接埠，或者是在 Dockerfile 檔案中使用 EXPOSE 指令進行連接埠的對應作業，倘若未指定容器主機所要對應的連接埠時，那麼系統將會自動採用隨機連接埠的方式進行對應。管理人員可以隨時透過 docker ps 指令，確認容器主機及容器的連接埠對應資訊（如下圖所示）：

```
docker run -itd -p 80 microsoft/windowsservercore cmd
```

如下圖所示，為 Windows Server 2016 NAT 容器虛擬網路環境中，Music Store 音樂網站的佈署模式及運作架構示意圖。

容器子網路的部分，取決於容器主機的 NAT 虛擬網路環境子網路，如下圖所示的 NAT 虛擬網路環境子網路為本書實作環境，至於實務應用上則取決於容器主機 NAT 虛擬網路環境的 `InternalIPAddressPrefix` 子網路。

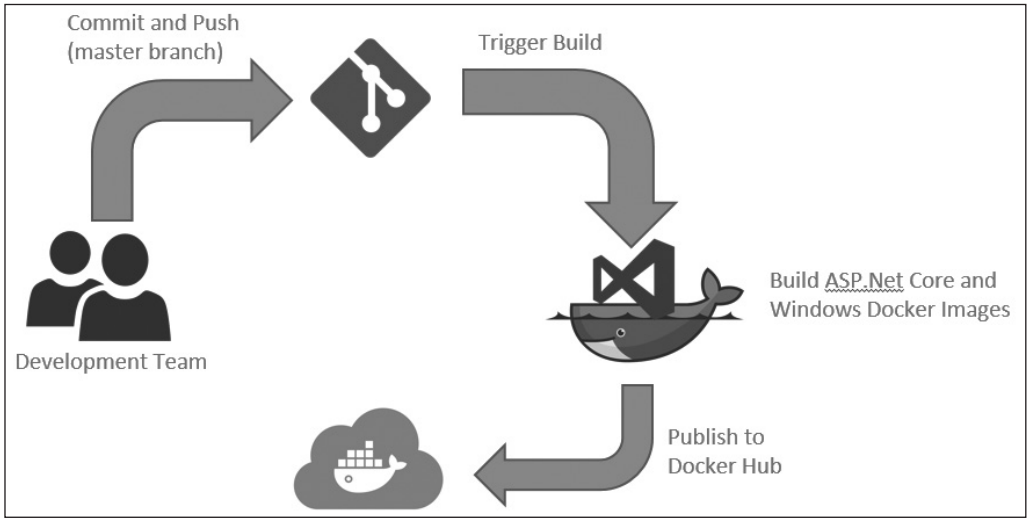


請依照下列操作步驟，開始為 Music Store 音樂網站依照上述運作架構開始進行佈署作業：

1. 請在 Visual Studio 開發工具中，開啟 Music Store 音樂網站解決方案以便建構 Music Store Web 及 API 容器映像檔。
2. 請點選 `MusicStore.API` 這個 Web 專案檔，並在右鍵選單中選擇【**Build**】項目，與前面幾章實作演練操作步驟一樣，建構客製化容器映像檔的方式是相同的，並且在建構容器映像檔之前確保 `project.json` 檔案內容如下：

```
"scripts": {
    "prepublish": [ "bower install", "dotnet bundle" ],
    "postcompile": [ "powershell -executionpolicy
                     bypass ./docker/dockertask.ps1 -build
```

至此，我們已經完成為 Music Store 音樂網站應用程式，自動化建構程序的工作流程。因此，從現在開始開發團隊只要針對**主要分支（Master Branch）**，進行任何程式碼提交的動作之後，便會觸發執行自動化建構程序，然後將建構完成的 Docker 容器映像檔發佈到指定的 Docker Hub 帳號。如下圖所示，為整個自動化建構程序的運作流程示意圖：



建構佇列

除了 CI 持續整合機制之外，VSTS 平台還允許管理人員手動觸發建構程序，並且搭配排程作業在夜間（離峰）時間進行建構程序，以便確保每天都能自動觸發建構程序，並在建構程序發生失敗時通知開發團隊，以確保開發團隊交付程式碼後都能夠自動產生應用程式。

Git 分支原則，在這個自動化程序中將扮演成功的關鍵原因，因為 Git 分支機制能夠允許同時運作多個版本的程式碼，其中主要分支便是佈署至正式營運環境的程式碼，所以開發團隊中的所有開發人員可以從各個分支的名稱（如 Develop），來判斷要將程式碼提交至哪個所屬分支，當這些開發分支測試完成並佈署到正式營運環境後，便會被合併後併入 Master 當中，這也是佈署多個分支的目的之一。

我們也可以在 VSTS 平台上，建構多個分支並組態設定觸發建構程序，由於 VSTS 平台也包含工作項目管理機制，可以選擇把工作項目與失敗的建構項目，或者是針對失敗的測試項目進行關聯，然後分配或指派給相關團隊或人員，同時在建構系統中顯示整體的運作資訊，以便達到應用程式生命週期儀表板的目的。