Docker เบื้องต้น

ก่อนจะไปใช้งาน Docker เรามาดู ที่มาที่ไปกันก่อนว่าทำไมถึงมีคนพัฒนา software แบบ Docker นี้ ขึ้นมา เจ้า Docker นี้ เป็น application ประเภทที่เรียกว่า software container ซึ่งมีวัตถุประสงค์หลักในการลดปัญหาใน เรื่องของ software version conflict และปัญหาความแตกต่างของ environment ระหว่างเครื่อง develop และ production

สมมติว่า เราต้องทำการพัฒนา application เพื่อทำงานบน Linux โดยเครื่อง production มีสภาพแวดล้อม คือใช้ MySQL + PHP 7.0 ในขณะที่เครื่อง development นั้นติดตั้งเพียง PHP 5.6 และไม่สามารถยกเลิกได้เนื่องจาก ยังมี application อื่นที่กำลังพัฒนาอยู่เช่นกัน ในสถานการณ์อย่างนี้ จะเกิดความยุ่งยากขึ้นมาทันที ดีไม่ดีก็ต้องหาเครื่อง development ใหม่ซึ่งคงขอมาไม่ได้ง่ายๆ

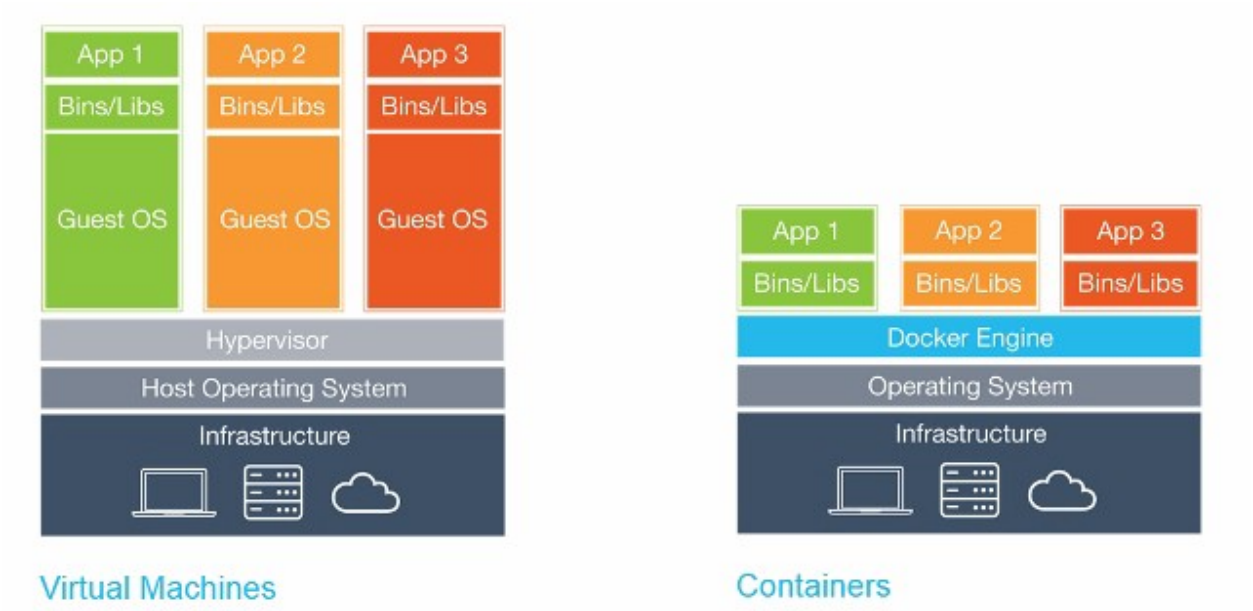
อีกตัวอย่างหนึ่ง สมมติว่าใน server ที่เราใช้งานอยู่นั้นมี application ทำงานอยู่หลายตัว และเราเกิดมีความ จำเป็นต้อง upgrade ตัวใดตัวหนึ่งในนั้น ซึ่งการ upgrade นี้ จะต้องใช้ library ที่มี version ไม่ตรงกับที่ใช้อยู่ในเครื่อง และไม่สามารถเปลี่ยนแปลงได้เนื่องจาก application อื่นๆก็ใช้งานอยู่ด้วย แบบนี้ก็ทำให้เราไม่สามารถ upgrade application เป็นบางตัวได้ คือ ถ้าจะทำก็ต้องทำทั้งหมด ทั้งที่ตัวที่เราจะ upgrade มีตัวเดียว เป็นต้น

นอกจากนี้ สิ่งที่ software container เข้ามาช่วยก็คือ ลดความยุ่งยากในการ deploy application และการ test ที่ซ้ำซ้อนระหว่างเครื่อง development และ production เนื่องจาก application ที่ทำงานอยู่ภายใน software container นั้น จะมีทุกอย่างครบในตัวอยู่แล้ว เมื่อต้องการนำไปใช้งานก็เพียงเอา container ไปติดตั้งบน server แล้วก็ run มันขึ้นมา โดยไม่จำเป็นต้อง install อะไรเพิ่มเติมอีกแล้ว ทั้งนี้ ยกเว้น application ที่มีความจำเป็นต้อง communicate กับ application ที่อยู่ใน container อื่นๆ ซึ่งก็จะต้องทำการ config และ test เหมือนเดิม

Software Container คืออะไร

Software Container เป็น concept ของการสร้างสภาพแวดล้อมเฉพาะให้ซอฟต์แวร์ทำงานได้โดยไม่กวนกับ ซอฟต์แวร์ตัวอื่นบนระบบปฏิบัติการเดียวกัน เราสามารถเอา Container ไปรันในคอมพิวเตอร์หรือ Server เครื่องไหนก็ ยังทำงานได้เหมือนเดิม โปรแกรมใน Container ยังทำงานได้ปกติไม่ผิดเพี้ยนจากเดิม ที่ผ่านมา Software Container มีการพูดถึงและมีการใช้งานกันมานานแล้ว อาทิ LXC (Linux Container), Solaris Containers, OpenVZ เป็นต้น แต่ ไม่เป็นที่แพร่หลายมากนัก เนื่องจากมีการใช้งานค่อนข้างยุ่งยาก

ความแตกต่างระหว่าง Virtual Machine กับ Container



- Container จะเป็นการเพื่อจำลองและควบคุมสภาพแวดล้อมสำหรับการรันเฉพาะบาง Service เช่น Contianer ที่รัน nginx ใน ubuntu ก็จะบรรจุ Environment เหล่านี้ไว้เป็น 1 Contianer และรัน service เท่าที่จำเป็นต้องใช้เท่านั้น ทำให้ใช้ทรัพยากรน้อยกว่า Virtual Machine
- Virtual Machine จะเป็นการจำลอง Environment มาทั้ง OS รันขึ้นมาเป็นเครื่อง Server 1 เครื่อง และมีการรัน service หลายๆ service ใน VM เดียวกัน ทำให้แต่ละ VM ต้องใช้ทรัพยากรจำนวนมาก

  อย่างไรก็ดี ด้วยสถาปัตยกรรมนี้ ทำให้ container ไม่สามารถทำงานบน Host ที่มี Kernel ต่างไปจาก Host ที่ทำการพัฒนา application นั้นมา เช่น application ที่พัฒนามาบน Linux เมื่อนำไปใส่ไว้ใน container ก็ไม่สามารถนำ container นั้นไปทำงานบน Windows Native ได้ เป็นต้น

Docker คืออะไร

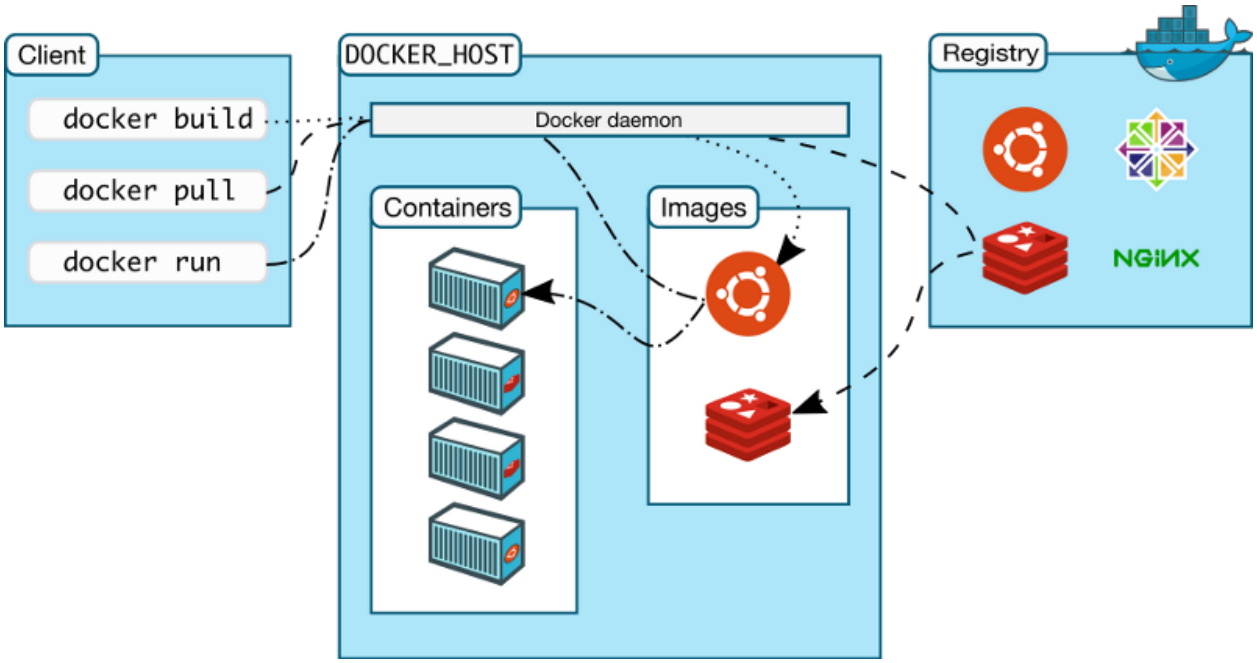
Docker ก็เป็น Software Container ที่ถูกพัฒนาขึ้นมาให้สามารถการจัดการ Container ได้ง่าย Image มีขนาดเล็ก แยกเป็นชั้นๆ สร้างแนวคิด build, ship, run ในการ deploy application

Docker มีชุดซอฟต์แวร์ให้ใช้งานดังนี้

- Docker Engine เป็น Core หลักในการทำงาน
- Docker Machine เป็นซอฟต์แวร์สำหรับสร้าง หรือ เซ็ตอัพเครื่องให้พร้อมสำหรับ container
- Docker Swarm เป็นซอฟต์แวร์ที่เอาไว้เชื่อมต่อ Docker Engine หลายๆเครื่องมาเชื่อมต่อรวมกันให้เป็น cluster

- Docker Compose เป็นซอฟต์แวร์สำหรับอำนวยความสะดวกในการรัน Container โดยสามารถสั่งรันหลายๆ Container ได้พร้อมๆกัน ตั้งค่าต่างๆ รวมไปถึงเชื่อมโยง Container ให้สามารถทำงานร่วมกันได้
- Kitematic คือ เครื่องมือ GUI ที่รวบรวมคำสั่งของ Docker Command ต่าง ๆ ในรูปแบบ UI ให้สามารถใช้งานได้ง่าย ๆ ยิ่งขึ้น

องค์ประกอบต่าง ๆของ Docker



Docker image

คือต้นแบบของ Container ข้างในจะเป็น Linux ที่มีการติดตั้ง Application และ มีการ Configuration เอาไว้แล้ว ซึ่งเกิดมาจากการ build ไฟล์ Dockerfile ขึ้นมาเป็น image

Docker container

container จะถูกสร้างมาจาก Docker Image ที่เป็นต้นแบบ เกิดเป็น container จะได้ Service หรือ Application ที่สามารถเรียกใช้งานได้ทันที

Docker registry

เราสามารถสร้าง Docker Image แล้วนำไปเก็บรวบรวมไว้บน server (ลักษณะเดียวกับการเก็บ Source Code ไว้บน Github) โดย Docker registry ณ ปัจจุบันก็มีให้เลือกใช้งานได้หลากหลายโดยมี Docker Hub เป็น Docker registry หลักในการเรียกใช้(pull) Docker Image และนอกจากนี้ยังมีผู้ให้บริการ docker registry เจ้าอื่นๆด้วย เช่น Gitlab, Quay.io, Google Cloud เป็นต้น

การติดตั้ง Docker

สามารถ Download มาติดตั้งตาม OS ที่เราจะใช้งานได้เลยตาม link ด้านล่าง
นี้ https://www.docker.com/products/docker-desktop

สำหรับ Docker for Windows นั้นต้องใช้ฟีเจอร์ Hyper-V ด้วย ซึ่งใน Windows Home version จะไม่มี Hyper-V ก็จะไมสามารถติดตั้งได้ โดย Docker ในเวอร์ชัน Windows และ Mac จะเรียกว่า Docker Desktop ส่วนใน Linux based จะเรียกว่า Docker Engine ตัวฟรีทั้งหมดสามารถโหลดได้จาก link ด้านล่างนี้ (ปกติ การติดตั้งใน Linux จะใช้ผ่านระบบ installation ของ Linux นั้นๆมากกว่าการ download เอง)

https://hub.docker.com/search/?type=edition&offering=community

สำหรับ Docker Desktop นั้น เมื่อติดตั้งเสร็จแล้ว จะมี icon ปรากฏบนหน้าจอ (ยกเว้นไปเอา check box ออก ก็ไปรันจาก start menu เอา) เมื่อ run โปรแกรม มันจะเป็น System service ซึ่งจะเห็นไป icon อยู่ใน System Tray ด้านล่างขวา เมื่อโปรแกรม start แล้ว จะปรากฏหน้าต่างแนะนำ และให้เราทำการ login เข้าระบบของ Docker (ก่อนที่เราจะ download ได้ มันก็จะบังคับให้ลงทะเบียนไว้แล้ว) ในการติดตั้ง โปรแกรมจะถามว่าจะใช้ Windows based container แทน Linux หรือไม่ ตรงนี้ให้ใช้เป็น Linux based ไปก่อน ก็ไม่ต้องไปเลือกอะไร

เมื่อทำการ login แล้ว ให้ลอง right click ที่ icon จะปรากฏเมนูขึ้นมา เลือก Settings เพื่อมากำหนดค่าต่างๆ ก่อน ในส่วนของ General โปรแกรมจะเลือก Start Docker Desktop when you login เอาไว้เลย ทำให้มันทำงานทันทีเมื่อเรา login ซึ่งหากเราแค่จะใช้เพื่อทดสอบ ก็แนะนำให้เอาออก ส่วน option อื่นๆก็ตามชอบ หากต้องการเป็นส่วนหนึ่งในการพัฒนา Docker ก็เลือก Send usage statistic ไว้ แต่ถ้าไม่อยากเปลือง net ก็เอาออก ส่วนอันสุดท้ายไม่จำเป็นก็ไม่ต้องไปเลือก

ต่อไปให้ไปที่หัวข้อ Shared Drives ซึ่งจะเป็นการกำหนดว่าจะใช้ Drive ไหนในการทำงานกับ Docker ถ้าไม่เลือกไว้ก็จะมีปัญหาในการรันคำสั่ง Docker เลือกแล้วอย่าลืม Apply

ในส่วน Advanced จะเป็นการกำหนด resources ให้กับ Docker ทั้ง vCPU, Memory และ พื้นที่เก็บ Image

ส่วนของ Network จะเป็น virtual network สำหรับใช้ระหว่าง container ภายในของเครื่อง

ส่วนของ Proxies Daemon และ Kubernetes ให้ใช้ default setting

ถ้าต้องการ reset ค่ากลับเหมือนเดิม ก็เลือกเมนู Reset

สำหรับการใช้งาน Docker นั้นจะต้องใช้คำสั่งในการสั่งงาน ผ่าน Command Line เมื่อติดตั้งเสร็จแล้ว ให้เปิดโปรแกรม Terminal หรือ cmd ในแบบ “Run as Administrator” เพื่อป้องกัน permission error

ตัวอย่างคำสั่งแรก ลองใช้คำสั่งตรวจสอบ version ดังนี้

docker --version หรือ docker -v

ถ้ามี Output ที่ไม่ใช่ Error ก็แปลว่าการติดตั้งเสร็จสมบูรณ์

การทำงานกับ Docker

ในเอกสารชุดนี้ จะนำเสนอการใช้งานแบบเบื้องต้นเท่านั้น ซึ่งยังอาจไม่สามารถนำไปประยุกต์ใช้กับการทำงานจริงได้ แต่จะนำเสนอให้เห็นภาพกว้างๆของวิธีและแนวทางในการใช้งาน

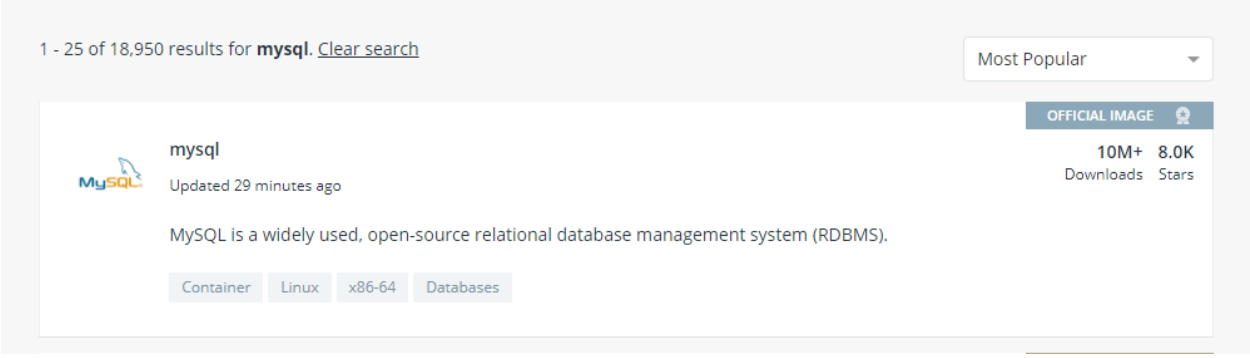
Docker Image

Docker Image ก็เหมือนกับบะหมี่กึ่งสำเร็จรูป ที่เราสามารถนำมาปรับแต่งแล้วก็ใช้งานได้เกือบจะในทันที โดย image เหล่านี้สามารถหาโหลดได้จาก Docker Hub ตาม link นี้ https://hub.docker.com/

Workshop

- ค้นหา MySQL image บน docker hub

- ระบบจะแสดงผลของ Image ที่เป็น MySQL ทั้งหมด ซึ่งมีเยอะมาก (18,950) ทั้งนี้ image ในระบบของ Docker Hub จะมี 2 ประเภทหลักๆคือ Official Image กับ Shared Image ก็คือ Image ที่มีคนสร้างไว้แล้วนำมา shared ลงใน Docker Hub การจะพิจารณาว่าจะใช้ image ไหนก็ให้พิจารณาจากจำนวนการ download และ จำนวน stars ที่ image นั้นได้รับ ใน workshop นี้ จะใช้ Official Image ตามรูป



- ทำการ pull image ด้วยคำสั่ง docker image pull mysql ดังรูป

```
PS C:\WINDOWS\system32> docker image pull mysql
Using default tag: latest
latest: Pulling from library/mysql
27833a3ba0a5: Pull complete
864c283b3c4b: Pull complete
cea281b2278b: Pull complete
8f856c14f5af: Pull complete
9c4f38c23b6f: Pull complete
1b810e1751b3: Pull complete
5479aaef3d30: Pull complete
ded8fa2e1614: Pull complete
636033ba4d2e: Pull complete
902e6010661d: Pull complete
dbe44d2bf055: Pull complete
e906385f419d: Pull complete
Digest: sha256:a7cf659a764732a27963429a87eccc8457e6d4af0ee9d5140a3b56e74986eed7
Status: Downloaded newer image for mysql:latest
PS C:\WINDOWS\system32>
```

Docker ก็จะทำการ download image มาเก็บไว้ในเครื่อง ทีนี้ จะมีส่วนรายละเอียดอีกเล็กน้อยดังนี้

- Tag – เปรียบเสมือน version ของ image หากเราไม่ระบุ ก็จะได้ latest มาเป็น default หากจะระบุ tag ให้ใส่ ":" ต่อท้ายชื่อ image แล้วตามด้วย tag label เช่น
  docker image pull mysql:8
- Layers – ในสถาปัตยกรรมของ Docker จะแบ่ง image ออกเป็นหลายๆส่วน แต่ละส่วนเรียกว่า Layer แต่ละ Layer ก็จะมี ID กำกับ อย่างในกรณีของ mysql จะเห็นว่ามี Layer ใน image อยู่ทั้งหมด 12 Layer ข้อดีของการใช้ Layer ก็คือ หากมีการ update ในตัว image ระบบจะ download เฉพาะ Layer ที่เปลี่ยนมาใช้เท่านั้น

- ทดลองเรียกดูว่าในขณะนี้มี image ในระบบของเราทั้งหมดเท่าไหร่ด้วยคำสั่ง
  docker image ls

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> docker image ls
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
mysql                     latest              7bb2586065cd        31 hours ago        477MB
docker4w/nsenter-dockerd  latest              2f1c802f322f        5 months ago        187kB
PS C:\WINDOWS\system32>
```

จะเห็นว่ามี image อยู่ในระบบ 2 ตัว คือ mysql และ docker4w/nsenter-dockerd ก็คือตัว Docker Desktop นั่นเอง

Further Commands

- หากต้องการดูว่ามี command / option อะไรบ้างในสวนของ docker image ให้ใช้คำสั่ง
  docker image –help
- หากต้องการลบ image ที่ไม่ใช้ สามารถลบได้ดังนี้
  docker image rm <image:tag>
- จำไว้ว่า การลบด้วย rm นั้น ระบบจะไม่ทำการ ลบ ออกไปจาก hard disk จริงๆ ก็เหมือนกับการ delete บน Windows ซึ่งหมายความว่า มันจะยังคงใช้เนื้อที่อยู่นั่นเอง หากต้องการลบ image เหล่านี้ออกอย่างถาวรให้ใช้คำสั่ง
  docker image prune
  จะเป็นการ purge image ทั้งหมดที่เคยถูกลบไปแล้วออกจากเครื่องอย่างถาวร

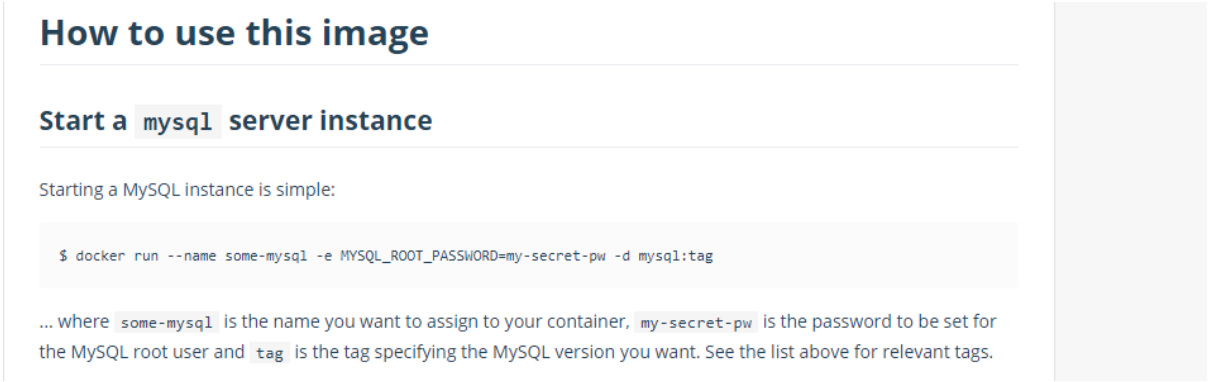
- หากต้องการดูรายละเอียดของ image สามารถใช้คำสั่งดังนี้
  docker image inspect <image_name>

Docker Container

เมื่อเราได้ docker image มาแล้ว หากต้องการนำมาใช้งาน เราก็ต้องนำเอา image นั้น มาสร้างให้เป็น container นึกสภาพก็คล้ายกับ เรามีของที่จะบรรจุตู้ container แล้ว ก็ต้องหา container มาใส่ ถ้าเราใส่ของเหมือนๆกันอยู่ภายใน container หลายๆตู้ ก็เหมือนกับเราสร้าง container ของ docker image ที่เหมือนกันหลายๆอัน ซึ่งนี่เป็นจุดเด่นในเรื่องของการ deploy โปรแกรมที่ทำ image ไว้แล้วได้อย่างรวดเร็ว

Workshop

- ดูวิธีการ run ตัว mysql จาก docker hub จะเห็นคำแนะนำเกี่ยวกับการใช้คำสั่ง ตามรูปด้านล่าง



How to use this image

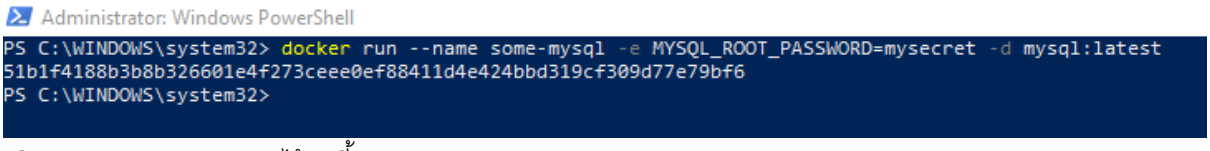
Start a `mysql` server instance

Starting a MySQL instance is simple:

```
$ docker run --name some-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag
```

... where `some-mysql` is the name you want to assign to your container, `my-secret-pw` is the password to be set for the MySQL root user and `tag` is the tag specifying the MySQL version you want. See the list above for relevant tags.

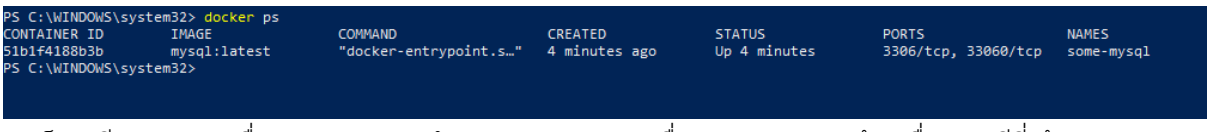
- ทดลองรันตามตัวอย่างดังนี้



อธิบาย option แบบคร่าวๆได้ดังนี้

  - --name คือการตั้งชื่อให้กับ container ที่กำลังรัน หากไม่ตั้งชื่อ เราจะได้ชื่อเป็น random id ยาวๆ ใช้งานลำบาก
  - -d คือออพชันที่บอกให้ container นี้ทำงานแบบ daemon หรือทำงานเป็นพื้นหลัง
  - -e เป็นการกำหนดค่าให้กับ Environment Variable ที่ container จะเอาไปใช้แทนที่ค่านั้นๆตามที่กำหนด ตามตัวอย่างนั้น mysql ต้องการ MYSQL_ROOT_PASSWORD ในการสร้าง container เราจึงต้องกำหนดค่านี้ หากไม่กำหนดก็จะไม่สามารถสร้าง container ได้ และจะมี error เกิดขึ้น

- ตรวจสอบการทำงานของ container ด้วยคำสั่ง docker ps ดังนี้



จะเห็นว่ามี container ชื่อ some-mysql ทำงานอยู่ จาก image ชื่อ mysql:latest สร้างเมื่อ 4 นาทีที่แล้ว ทำงานที่พอร์ต 3306

- ทดลองเชื่อมต่อไปยังพอร์ต 3306 บนเครื่อง localhost

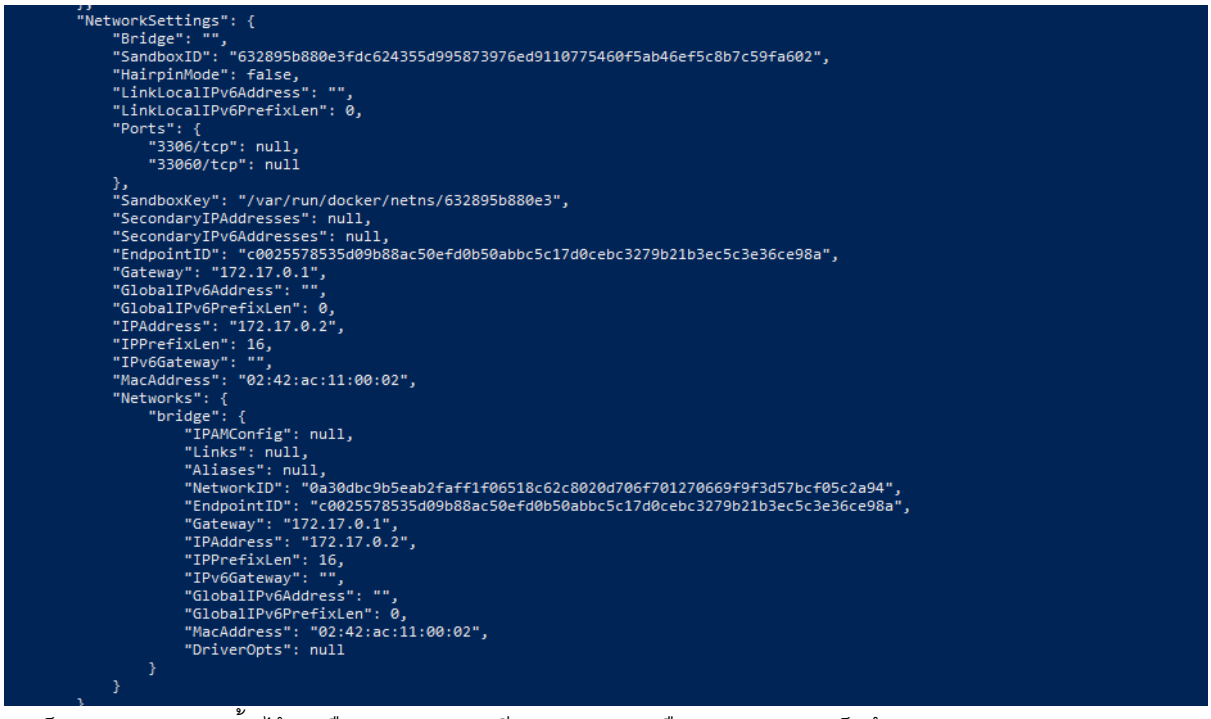
```
PS C:\WINDOWS\system32> telnet localhost 3306
Connecting To localhost...Could not open connection to the host, on port 3306: Connect failed
PS C:\WINDOWS\system32>
```

จะเห็นว่า ไม่สามารถต่อเชื่อมได้ ทั้งนี้เนื่องจาก container นั้นเปรียบเสมือน VM Server ตัวหนึ่ง มันจะมี IP เป็นของมันเอง ถ้าเราไม่ได้ทำการ map port ออกมาที่ host แล้ว ก็ไม่สามารถเข้าถึง service นั้นๆได้

- ดูรายละเอียดของ container ด้วยคำสั่ง

  docker container inspect <container-name>

ด้วยคำสั่งนี้ เราจะเห็นรายละเอียดทั้งหมดของ container หากดูไล่ลงมาเรื่อยๆก็จะเห็นรายละเอียดเกี่ยวกับ Network Setting ดังนี้



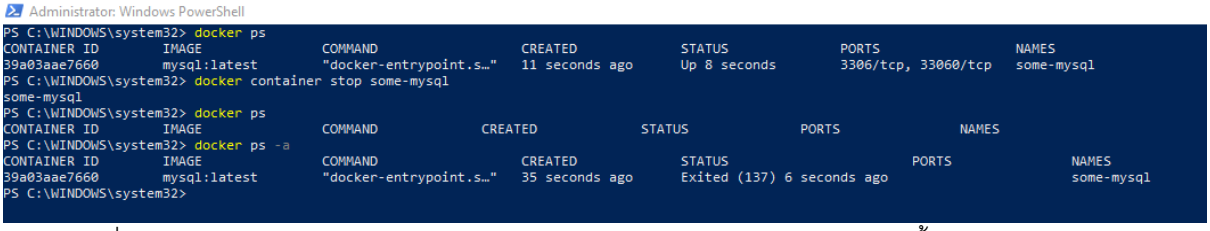
จะเห็นว่า container นั้น ได้ IP คือ 172.17.0.2 มี Gateway IP คือ 172.17.0.1 เป็นต้น

- หยุดการทำงานของ container ด้วยคำสั่ง

  docker container stop <container-name>

  จากนั้นลองเช็คด้วย

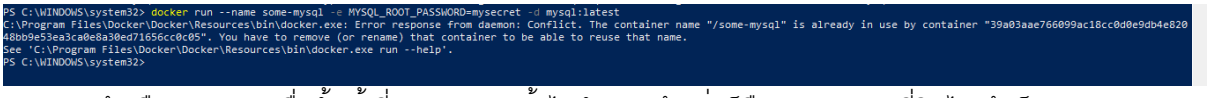
docker ps และ docker ps -a ดังรูป



จะเห็นว่าเมื่อทำการ stop container แล้ว หากใช้ docker ps ตามปกติ จะไม่มีอะไรแสดงขึ้นมาเลย แต่หากใช้ docker ps -a แล้ว ก็จะเห็นว่ามี container 1 ตัว สถานะเป็น Exited เมื่อ 6 วินาทีที่แล้ว ทั้งนี้ option “-a” จะเป็นการแสดงผล container ทั้งหมด ที่ทำงานและไม่ได้ทำงาน

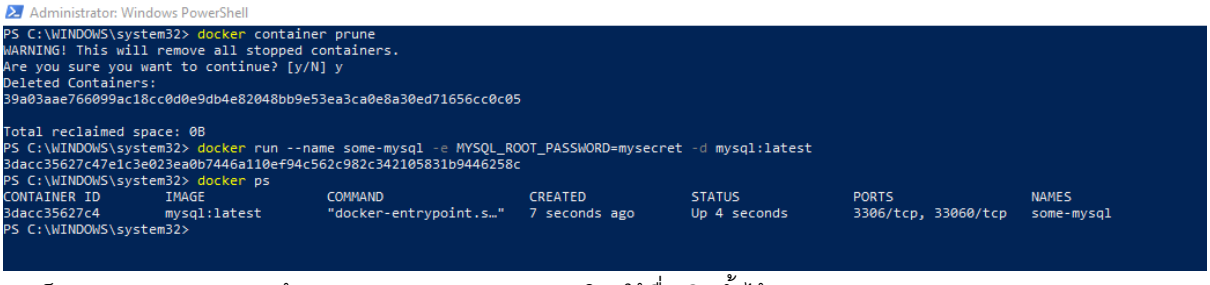
NOTE: เราไม่สามารถ restart container ได้ ไม่ว่ากรณีใด ทำได้เพียงสร้างขึ้นมาใหม่เท่านั้น

- ทดลอง run container แบบเดิมอีกครั้ง



ระบบจะแจ้งเตือน error ว่าชื่อซ้ำ ทั้งที่ container นั้นไม่ทำงานแล้ว นั่นก็คือ container ที่ปิดไปแล้วก็ยังคงค้างอยู่ในระบบแบบเดียวกับ image ทั้งนี้ storage path ที่ถูกกำหนดให้ container จะยังคงอยู่ ซึ่งเราสามารถไป copy ข้อมูลที่สำคัญมาก่อนได้ เราสามารถลบ container เป็นการถาวรได้ด้วยคำสั่ง docker container prune เหมือนกับการลบ image ซึ่งเมื่อรันแล้ว storage path จะถูกลบทิ้งไปด้วยแบบถาวรและเราจะไม่สามารถกู้ข้อมูลใดๆกลับมาได้อีก

- ทดลองลบ container ที่ปิดแล้วและลองรันใหม่



จะเห็นว่าหลังจาก prune แล้ว เราสามารถรัน container โดยใช้ชื่อเดิมซ้ำได้

NOTE: Docker for Windows จะเก็บ storage ใน Virtual HD ดังนั้นเราจะไม่สามารถเห็น storage นั้นได้โดยตรง

- โดย concept ของ container แล้ว ตัว container ไม่ควรจะมีขนาดที่ใหญ่ขึ้น แล้วถ้าเป็น container ของ MySQL ที่เป็น database แล้วจะทำอย่างไร? คำตอบคือ เราควรต้องกำหนด External Volume ที่อยู่บน host แยกออกมาต่างหาก ข้อดีก็คือ External Volume นี้ จะไม่ถูกลบแม้ว่า container จะถูกลบไป ทำให้เราสามารถรัน container ขึ้นมาใหม่โดยใช้ External Volume เดิมได้ ก็จะสามารถใช้ข้อมูลเดิมได้ ก่อนจะทำการ map volume เข้าไปใน container เราต้องรู้ก่อนว่า มี path อะไรที่จะสามารถ map เข้าไปได้บ้าง เราสามารถสังเกตุได้จากการใช้คำสั่ง docker image inspect ได้ เช่นในกรณีของ MySQL เราจะเห็นข้อมูลดังนี้

```
        ],
        "ArgsEscaped": true,
        "Image": "sha256:fb28e3c8be89ca675d11ac80ea54ca99a2d57ecb2684c8ea8152e75e7a2010db",
        "Volumes": {
            "/var/lib/mysql": {}
        },
        "WorkingDir": "",
        "Entrypoint": [
            "docker-entrypoint.sh"
        ],
        "OnBuild": null,
        "Labels": {}
    },
    "DockerVersion": "18.06.1-ce",
```


จะเห็นว่ามี Volumes ที่กำหนดอยู่คือ /var/lib/mysql เราจสามารถกำหนด External Path ให้ไป map กับ volume นี้ได้

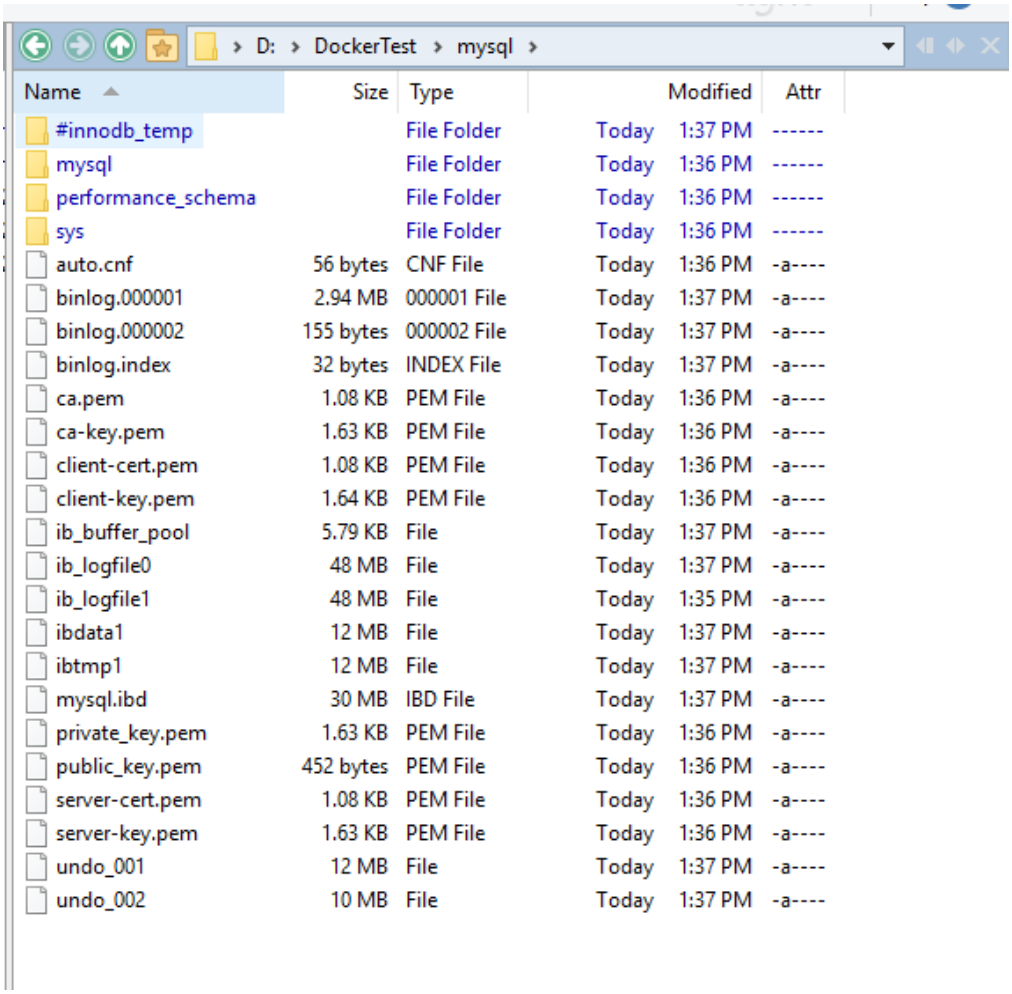
- ทดลองสร้าง container แบบกำหนด mapping port และ external path ดังนี้

  docker run --name mysql-01 -e MYSQL_ROOT_PASSWORD=mysecret -d -p 3306:3306 -v d:/DockerTest/mysql:/var/lib/mysql mysql:latest

```
PS C:\WINDOWS\system32> docker run --name mysql-01 -e MYSQL_ROOT_PASSWORD=mysecret -d -p 3306:3306 -v d:/DockerTest/mysql:/var/lib/mysql mysql:latest
4e74faa61180e025a8a0c674ac996510706b0d1a0b7d76df5344d8a020a70cb5
PS C:\WINDOWS\system32>
```

เมื่อรันแล้ว จะมี pop-up ถามการอนุญาตเรื่อง network port ก็ให้ตอบ Yes / OK

จากนั้น หากไปดูใน D:\DockerTest\mysql จะเห็นว่ามีไฟล์ของ mysql อยู่



option “-“ นั้น ไว้สั่งให้ docker ทำการ map port จาก container มายัง host โดยเลขที่อยู่ข้างหน้า “:” คือ port บน host ที่ต้องการจะเปิด ส่วนเลขที่อยู่ข้างหลังคือ port ของ container

option “-v” ไว้สั่งให้ docker ทำการ map volume จาก host path ไปยัง container path

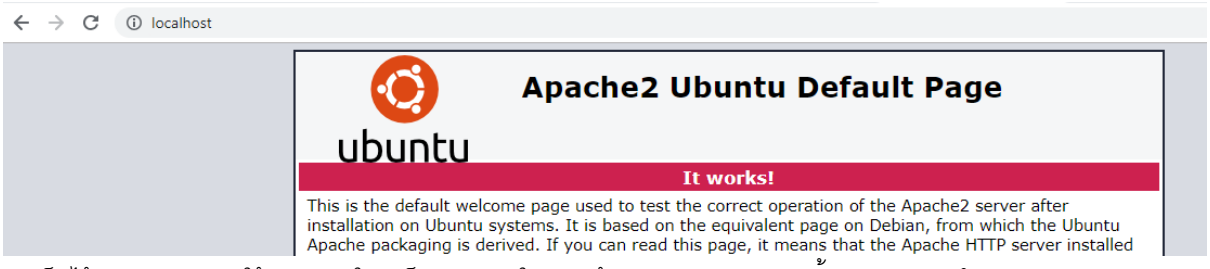
- ทดลองต่อเข้าพอร์ต 3306 ผ่าน localhost ด้วย telnet หรือ mysql connector ใดๆ เช่น telnet localhost 3306 จะพบว่าสามารถเชื่อมต่อเข้าไปได้

Dockerfile

นอกจากการเอา image ที่มีอยู่แล้วมาสร้างเป็น container นั้น เราสามารถสร้าง image ขึ้นมาเองได้ การสร้าง image จะมีด้วยกัน 2 วิธี คือ การ build image จาก container ที่กำลังทำงานอยู่ หรือ การใช้ Dockerfile ในการช่วยสร้าง image

Workshop - สร้าง custom image ผ่าน container

- โหลด ubuntu มาด้วยคำสั่ง docker image pull ubuntu
- รัน container ด้วยคำสั่ง docker container run -it --name=ubuntu-01 ubuntu
  ด้วย option -it จะทำให้เรา access เข้าไปยัง container ที่รันขึ้นมา ก็จะเห็นเป็น Ubuntu prompt
- รันคำสั่งใน ubuntu ดังนี้
  apt-get update - เพื่อ update ระบบ และ repository
  apt-get install apache2 - เพื่อติดตั้ง apache2
- ออกจาก container โดยไม่ terminate ด้วยปุ่ม <CTRL-P> ตามด้วย <CTRL-Q>
- ทำการสร้าง image จาก container ubuntu-01 ดังนี้
  docker commit ubuntu-01
- ตรวจสอบ image ด้วยคำสั่ง docker image ls จะเห็น image ที่ไม่มีชื่อ ให้จำ ID หรือ copy ไว้
- ทำการเปลี่ยน Tag ของ image เพื่อให้ง่ายต่อการใช้ด้วยคำสั่ง
  docker tag <image id> my-ubuntu:1.0
  เราจะอ้างอิง image นี้ว่าเป็น my-ubuntu เวอร์ชัน 1.0
- ลองรัน container ขึ้นมาใหม่พร้อม map port 80 ดังนี้
  docker run -it --name ubuntu-01 -p 80:80 my-ubuntu:1.0
- เมื่อเข้ามาใน Ubuntu prompt ทำการรัน apache2 service ดังนี้
  service apache2 start
  จะเห็นได้ว่า package ของ Apache2 ได้ถูกติดตั้งไว้ใน container เรียบร้อยแล้ว ซึ่งแตกต่างจาก ubuntu ตัวหลักที่เราโหลดมาเป็นต้นแบบ
- ลองใช้ browser บน host เปิดไปยัง http://localhost ควรจะเห็น default page ของ Apache2



จะเห็นได้ว่า เราสามารถใช้ image ใดๆเป็น แม่แบบในการสร้าง container จากนั้นเราสามารถทำการ update container นั้นๆแล้วนำมาสร้างเป็น image แบบที่ต้องการได้

Workshop – สร้าง custom image ด้วย Dockerfile

- สร้าง folder สำหรับเก็บ Dockerfile ไว้ที่ตรงไหนก็ได้
- สร้างไฟล์ชื่อ Dockerfile (ตัวเล็ก ตัวใหญ่ ตามนี้ และไม่มีนามสกุลไฟล์) ไว้ใน folder จากข้อที่แล้วด้วย content ดังนี้

```
FROM ubuntu:18.04
RUN apt-get update && apt-get install -yq apache2
EXPOSE 80
CMD apachectl -D FOREGROUND
```

รายละเอียดสามารถอธิบายได้คร่าวๆดังนี้

  - FROM ไว้ระบุ image ที่จะเป็นต้นแบบ ควรใส่ Tag ที่ชัดเจน ปกติจะไม่ใช้ latest
  - RUN เสมือนการรันคำสั่งจากภายใน container คล้ายกับที่เรารันคำสั่งผ่าน Ubuntu prompt
  - EXPOSE ไว้ระบุว่า container นี้ จะมี port อะไรบ้าง
  - CMD เพื่อบอกว่า หลังจาก container start แล้ว จะให้ทำคำสั่งอะไร จะมีความแตกต่างจาก RUN ที่จะเป็นการทำงานในระหว่างการสร้าง image เท่านั้น

- ใน windows prompt ให้ cd ไปยัง drive และ folder ที่มี Dockerfile แล้วรันคำสั่ง

  docker build -t my-ubuntu:2.0 .

  (อย่าลืม '.' ที่อยู่หลังสุด!!!)

```
PS D:\Documents\ABAC\Docker Tutorial\ubuntu-apache> docker build -t my-ubuntu:2.0 .
```

- หลังจากสร้างแล้ว ลองตรวจสอบด้วย docker image ls ดู จะต้องมี image ชื่อ my-ubuntu Tag 2.0 อยู่ในรายการ

```
PS D:\Documents\ABAC\Docker Tutorial\ubuntu-apache> docker image ls
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
my-ubuntu                 2.0                 450fa6c2c24f        17 seconds ago      210MB
my-ubuntu                 1.0                 e03935d2041a        16 minutes ago      210MB
mysql                     latest              7bb2586065cd        8 days ago          477MB
ubuntu                    18.04               94e814e2efa8        3 weeks ago         88.9MB
ubuntu                    latest              94e814e2efa8        3 weeks ago         88.9MB
docker4w/nsenter-dockerd  latest              2f1c802f322f        5 months ago        187kB
```

- ทำการรัน container ด้วยคำสั่ง

  docker container run -d -- name ubuntu-01 -p 80:80 my-ubuntu:2.0

PS1 : อย่าลืม stop container ตัวที่แล้วก่อน เพราะจะใช้ชื่อและ port เดียวกัน ถ้าจะทดลองคนละชื่อและคนละ port ก็ไม่ต้อง stop

PS2 : ถ้าลบแล้ว อย่าลืมใช้ docker container prune ไม่อย่างนั้นแล้วก็จะใช้ชื่อ ubuntu-01 ไม่ได้

- ทดสอบด้วยการเปิด browser เช่นเดียวกับ Workshop ก่อนหน้านี้