LAB Guide

Module1: Container

จะเป็นการแนะนำการใช้งาน docker command เช่นการโหลด image จาก registry, การ run container, port mapping และการ build image ด้วย Dockerfile

Step 1

ssh เข้าไปที่ master ด้วย command vagrant ssh master จากนั้น download file จาก github ด้วย command git clode https://github.com/chanwittt/kic-training จากนั้นเปลี่ยน Directory ไปที่ lab-module1 ด้วย command cd kic-training/lab-module1

ใช้คำสั่ง docker run -d –name=nginx -p8080:80 nginx เพื่อ run container

-d: run container in background

--name: ตั้งชื่อ container หากไม่ใส่ option นี้ docker จะทำการ auto generate ชื่อของ container ให้

-p: port mapping โดย port ตัวแรกหมายถึง port ของ host ใช้สำหรับ access application ส่วน port ตัวที่ 2 คือ port ของ app ที่ listen ใน container

-nginx: ชื่อของ image ที่ต้องการ run สามารถดูชื่อ image อื่นๆได้จาก https://hub.docker.com/ หรือ private registry ก็ได้เช่นกัน

```
vagrant@master:~/kic-training/lab-module2$ docker run -d --name=nginx -p8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
1f7ce2fa46ab: Pull complete
9b16c94bb686: Pull complete
9a59d19f9c5b: Pull complete
9ea27b074f71: Pull complete
c6edf33e2524: Pull complete
84b1ff10387b: Pull complete
517357831967: Pull complete
Digest: sha256:10d1f5b58f74683ad34eb29287e07dab1e90f10af243f151bb50aa5dbb4d62ee
Status: Downloaded newer image for nginx:latest
6b853e0b2cb4f6def871c33c152a1be846199025fd1f544e0c4b6e64237f056f
```

*หากติดปัญหา permission denied ให้ใส่ sudo ด้านหน้า command ที่จะ run หรือเพิ่มสิทธิ์ของ user โดยใช้ command

sudo usermod -aG docker $USER

newgrp docker

Step 2

ตรวจสอบ status ของ container ที่สั่ง run ไปด้วย command docker ps จะมี container ชื่อ nginx แสดงขึ้นมา

```
vagrant@master:~/kic-training/lab-module2$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                                     NAMES
91654415256e   nginx     "/docker-entrypoint.…"   5 seconds ago   Up 5 seconds   0.0.0.0:8081->80/tcp, :::8081->80/tcp     hardcore_meitner
6b853e0b2cb4   nginx     "/docker-entrypoint.…"   3 minutes ago   Up 3 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp     nginx
vagrant@master:~/kic-training/lab-module2$
```

สังเกตุ column IMAGE จะเป็นชื่อ Image ตามที่ระบุตอนใช้ command docker run หากต้องการ show container ทั้งหมดสามารถใช้ command docker ps -a

Step 3

ทดสอบ access nginx โดยใช้ command curl -I http://192.168.100.2:8080 จะได้ response ที่เป็น http 200 กลับมา

```
vagrant@master:~/kic-training/lab-module2$ curl -I http://192.168.100.2:8080
HTTP/1.1 200 OK
Server: nginx/1.25.3
Date: Mon, 11 Dec 2023 06:42:18 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Tue, 24 Oct 2023 13:46:47 GMT
Connection: keep-alive
ETag: "6537cac7-267"
Accept-Ranges: bytes
```

หรือทดสอบเข้าใช้งานผ่าน web browser ด้วย URL เดียวกัน จะแสดงหน้า html ของ nginx

Step 4

ใช้ docker exec เพื่อ command ใน container เช่นการแก้ไขไฟล์, restart service หรือ run bash

Run docker exec -it nginx sh -c "cat /etc/nginx/conf.d/default.conf" เพื่อ show nginx config ที่อยู่ใน container

*nginx=ชื่อของ container สามารถใช้ container id ได้ด้วยเช่น สามารถดู container id ได้จาก step 2

```
vagrant@master:~/kic-training/lab-module2$ docker exec -it nginx sh -c "cat /etc/nginx/conf.d/default.conf"
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    #error_page  404              /404.html;
```

run docker exec -it nginx /bin/bash เพื่อเข้าไปใช้งาน bash *เหมือนการ ssh เข้าไปใน server การ run command อะไรก็แล้วแต่จะมีผลแค่ใน container เท่านั้น ไม่มีผลกับ host

```
vagrant@master:~/kic-training/lab-module2$ docker exec -it nginx /bin/bash
root@6b853e0b2cb4:/# hostname
6b853e0b2cb4
root@6b853e0b2cb4:/# cat /etc/nginx/conf.d/default.conf
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    #error_page  404              /404.html;
```

สามารถใช้ command exit เพื่อกลับออกจาก container

```
root@6b853e0b2cb4:/# exit
exit
vagrant@master:~/kic-training/lab-module2$
```

Step 5

ตรวจสอบ logs ของ container โดยใช้ command docker logs nginx

*nginx=ชื่อของ container สามารถใช้ container id ได้ด้วยเช่น สามารถดู container id ได้จาก step 2

```
vagrant@master:~/kic-training/lab-module2$ docker logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/12/11 06:30:28 [notice] 1#1: using the "epoll" event method
2023/12/11 06:30:28 [notice] 1#1: nginx/1.25.3
2023/12/11 06:30:28 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2023/12/11 06:30:28 [notice] 1#1: OS: Linux 5.4.0-164-generic
2023/12/11 06:30:28 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/12/11 06:30:28 [notice] 1#1: start worker processes
2023/12/11 06:30:28 [notice] 1#1: start worker process 29
```

Step 6

Copy file จาก host ไปยัง container โดยใช้ docker cp ./demo-index.html nginx:/usr/share/nginx/html เป็นการ copy file index.html ใน folder lab module1 ไปยัง container nginx

Update nginx config โดยใช้ docker exec -it nginx sh -c "rm /etc/nginx/conf.d/default.conf" และ docker cp ./app.conf nginx:/etc/nginx/conf.d

Restart nginx services ด้วย docker exec -it nginx sh -c "nginx -t && nginx -s reload"

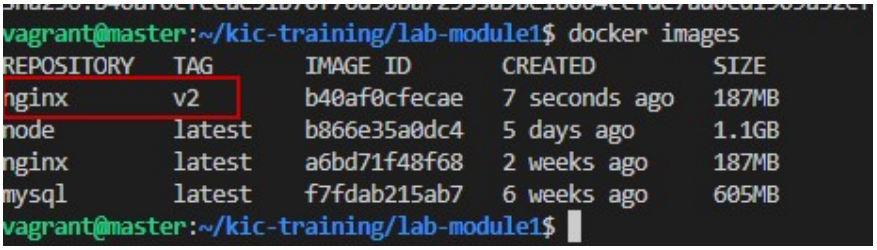
ทดสอบเข้า web browser ใส่ URL http://192.168.100.2:8080 หน้าเว็บจะมีการเปลี่ยนแปลงเป็นแบบใหม่ตามไฟล์ html ที่ update ไป

*ไฟล์ทั้งหมดที่มีการแก้ไขจะไม่ได้เก็บไว้ถาวร หากมีการลบ container และสร้างใหม่ ไฟล์ต่างๆกลับไปเหมือนตอนก่อนแก้ไข

สามารถใช้ command docker commit nginx nginx:v2 เพื่อ save เป็น image ใหม่เก็บไว้ใช้งานภายหลังได้

*nginx=ชื่อของ container สามารถใช้ container id ได้ด้วยเช่น สามารถดู container id ได้จาก step 2

ใช้ docker images เพื่อ show image ทั้งหมด จะมี images ที่ชื่อ nginx เพื่อเข้ามา แต่จะมี Tag เป็น v2 ตามที่ได้ตั้งไว้ตอน commit



Step 7

Build image โดยใช้ Dockerfile ด้วย command docker build -t mynginx:v1 .

-t : tag เป็นการตั้งชื่อ image

.: path ที่ Dockerfile อยู่ . หมายถึง path ที่อยู่ปัจจุบัน

ตัวอย่าง Dockerfile

```
1   FROM nginx
2   RUN rm /etc/nginx/conf.d/default.conf
3   COPY ./app.conf /etc/nginx/conf.d
4   COPY ./demo-index.html /usr/share/nginx/html
```

FROM: เป็นระบุว่าจะ build image โดยใช้ base จาก image ตัวไหน เช่น nginx, ubuntu, alpine

RUN: เป็นการ run command ใน image จากตัวอย่างจะเป็นการลบไฟล์ nginx config เดิมทิ้ง

COPY: copy file จากเครื่อง host เข้าไปใน image

```
vagrant@master:~/kic-training/lab-module1$ docker build -t mynginx:v1 .
[+] Building 0.3s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 167B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [1/4] FROM docker.io/library/nginx
 => [internal] load build context
 => => transferring context: 9.58kB
 => [2/4] RUN rm /etc/nginx/conf.d/default.conf
 => [3/4] COPY ./app.conf /etc/nginx/conf.d
 => [4/4] COPY ./demo-index.html /usr/share/nginx/html
 => exporting to image
 => => exporting layers
 => => writing image sha256:0e60114bd79d96e7f2335c453b24dcac205038b430500d8165ae83e0ea65a4b8
 => => naming to docker.io/library/mynginx:v1
vagrant@master:~/kic-training/lab-module1$
```

ลองใช้ docker images เพื่อ show image จะมี image ที่ชื่อว่า mynginx เพิ่มขึ้นมา

```
vagrant@master:~/kic-training/lab-module1$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
mynginx      v1        0e60114bd79d   51 seconds ago   187MB
nginx        v2        b40af0cfecae   18 minutes ago   187MB
node         latest    b866e35a0dc4   5 days ago       1.1GB
nginx        latest    a6bd71f48f68   2 weeks ago      187MB
mysql        latest    f7fdab215ab7   6 weeks ago      605MB
```

ลองนำ images ที่ build ใหม่ไป run ด้วย command docker run -d –name my-nginx -p8081:80 mynginx:v1

ใช้ docker ps เพื่อตรวจสอบ status จะมี container ใหม่ที่ใช้ image mynginx:v1 show ขึ้นมา

```
vagrant@master:~/kic-training/lab-module1$ docker run -d --name my-nginx -p8081:80 mynginx:v1
0c3528b7e25b441fa2d6f2583004aea6f2c5bab2d36d26e5449b82e36063fa3e
vagrant@master:~/kic-training/lab-module1$ docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED             STATUS             PORTS                                   NAMES
0c3528b7e25b   mynginx:v1    "/docker-entrypoint.…"   7 seconds ago       Up 6 seconds       0.0.0.0:8081->80/tcp, :::8081->80/tcp   my-nginx
6b853e0b2cb4   nginx         "/docker-entrypoint.…"   About an hour ago   Up About an hour   0.0.0.0:8080->80/tcp, :::8080->80/tcp   nginx
vagrant@master:~/kic-training/lab-module1$
```

จากนั้นทดสอบโดยใช้ web browser ใส่ URL http://192.168.100.2:8081 หน้าตาของตัว web จะแตกต่างจากหน้า web default ของ image nginx

ลบ container ที่สร้างไว้โดยใช้ command

docker stop nginx

docker rm nginx

docker stop my-nginx

docker rm my-nginx

Command อื่นๆ

docker start {container id หรือ container name}: start container ที่มีการสร้างไว้แล้ว

docker images: show images ที่มีอยู่บนเครื่อง

docker rmi {image id หรือ image name} ลบ image