Lab 7 : Docker

Usa Sammapun, Kasetsart University

Part 1: Installation

- installation on Mac OS

Download Docker Desktop จากเว็บด้านล่าง และ install ตามปกติ
https://docs.docker.com/docker-for-mac/install/

- installation on Windows

- ต้องเป็น Windows 10 หรือ 11 จึงจะใช้งาน Docker บน Windows ได้ โดยทำตามขั้นตอนต่อไปนี้
-

https://www.youtube.com/watch?v=naWNTWI_Bac

- หากไม่ต้องการ install บนเครื่องตนเอง หรือ Windows ไม่ใช่ Windows 10

- ให้ใช้เครื่องบน AWS ของอาจารย์ได้ โดยให้ download Putty (https://www.putty.org/)
- และ log on to AWS ด้วย Putty ดังนี้
  - IP: 34.211.144.67
  - username: kaset
  - password: Sawadee35813

Part 2: Running Docker Container

1) เปิด Terminal ใน Mac หรือ CMD ใน Windows หรือใช้ Putty

2) check if docker installed

```
$ docker
```

3) (download hello-world image if not exist) and run hello-world container to check whether docker can run

```
$ docker run hello-world
```

4) สองคำสั่งนี้เหมือนกันคือ ดูว่ามี docker image ใดบ้างอยู่ใน docker ของเรา

```
$ docker images
$ docker image ls
```

4) สองคำสั่งนี้เหมือนกันคือ ดูว่ามี docker container ใดบ้างอยู่ใน docker ของเรา

```
$ docker ps
$ docker container ls
```

5) (download if not exist) and run nginx image

```
$ docker run --name <your_name_nginx1> -d nginx
```

- nginx เป็น web server ยี่ห้อหนึ่ง
- ส่วน -d เป็น option ที่ให้ container ทำงานแบบ daemon คือ ทำงานใน background
- --name เป็น option ที่ให้เราตั้งชื่อ container ของเราได้ ปกติ docker จะตั้งชื่อแบบ random ให้ ในกรณีนี้ที่เราใช้เครื่องร่วมกันเพื่อเรียน docker จึงตั้งชื่อเอง จะได้รู้ว่า container ไหนเป็นของเรา

ลองรันแบบไม่ใช้ -d จะเห็นว่า shell จะไม่กลับมาให้เรารันคำสั่งอื่น (^C เพื่อ stop)

```
$ docker run --name <your_name_nginx2> nginx
```

6) ดูว่ามี container รันอยู่หรือไม่

```
$ docker ps
$ docker container ls
```

ทั้งสองคำสั่งทำงานเหมือนกัน แต่ docker ps เป็นคำสั่งแบบเก่า ส่วนคำสั่ง docker container ls เป็นคำสั่งแบบใหม่เพื่อแยกให้เห็นชัดเจนว่า คำสั่งนี้จัดการ container ไม่ใช่จัดการ image

7) run nginx at specific host port and 80 container port
กำหนดให้ port ของนิสิตเป็น 1 ต่อด้วยรหัสนิสิต 4 ตัวสุดท้าย เช่น
ถ้ารหัสนิสิตเป็น 6314400111 ให้ port เป็น 10111

```
$ docker run --name <your_name_nginx3> -p <your-port>:80 -d nginx
```

```
$ docker ps
```

8) ลองรันผ่าน browser โดยไปที่ url ต่อไปนี้ จะเห็นหน้าเว็บของ nginx
http://<IP ของ server ที่อาจารย์ให้>:<your-port>
http://localhost:<your-port>

9) docker จะเลือก port ให้อัตโนมัติ port จะได้ไม่ชนกัน เมื่อใช้ -P (ตัวใหญ่)

```
$ docker run --name <your_name_nginx4> -P -d nginx
$ docker run --name <your_name_nginx5> -P -d nginx
$ docker ps
```

ใช้ docker ps เพื่อดู port ที่ docker เลือกให้
จากนั้น ให้ลองเข้าไปดู nginx server ผ่าน browser

10) ลองรันเว็บ atm ของอาจารย์

```
$ docker run --name <your_name_atm> -P -d ladyusa/atm-usa
$ docker ps
```

http://<IP ของ server ที่อาจารย์ให้>:<your-port>/home
http://localhost:<your-port>/home

Part 3: start, stop container

11) หยุดการทำงานของ container
หาก run container แบบ daemon เมื่อจะหยุดการทำงาน จะใช้คำสั่ง docker stop โดยระบุ container ที่ต้องการด้วย id หรือชื่อ container ก็ได้

```
$ docker stop <id ของ container>
$ docker ps
```

```
$ docker stop <name ของ container>
$ docker ps
```

หรือทำผ่าน Docker Dashboard ได้

12) สามารถ start container ที่ stop ไปแล้วได้

```
$ docker start <id ของ container>
$ docker ps
```

```
$ docker start <name ของ container>
$ docker ps
```

13) ps with option -a show all containers (default shows just running)
โดยลอง stop container ของนิสิตตัวหนึ่งก่อนรัน command

```
$ docker stop <id>/<name>
$ docker ps
$ docker ps -a
```

Part 4: การ remove container และ image

14) remove container ได้ (แต่ image ยังคงอยู่)
โดย remove ได้เมื่อ container นั้น ไม่ได้รันอยู่

```
$ docker rm <id>/<name>
```

16) list all docker images

```
$ docker images
$ docker image ls
```

17) remove image (-f force remove even if it is running/has containers)
ต้องไม่มี container ที่ใช้ image นั้นอยู่เลย ไม่ว่าจะรันอยู่หรือไม่

```
$ docker rmi <image-name>
```

18) remove image ด้วย -f ถ้าต้องการลบ image แม้ว่าจะมี container อยู่

```
$ docker rmi -f <image-name>
```

19) pull image but does not run

```
$ docker pull <image-name>
```

III. การรัน MySQL ผ่าน Docker

1. Start Docker Desktop
2. สร้าง MySQL container ใน Docker ตามคำสั่งด้านล่าง
    - `--name=mysql_menu` เพื่อตั้งชื่อให้ container จะได้หาได้ง่าย
    - `-e MYSQL_ROOT_PASSWORD=abc123` เพื่อกำหนด password ในการเชื่อมมาที่ container เลือก password ตามต้องการ โดยต้องไปแก้ใน application.properties ให้ใช้ password ตรงกัน
    - `-e MYSQL_DATABASE=menu` เพื่อสร้าง database ชื่อ menu
    - `-p 3307:3306` เพื่อกำหนด port ในรูปแบบ port1:port2 โดย
        i. port1 ไว้เชื่อมกับโปรแกรมเรา ซึ่งต้องตรงกับไฟล์ application.properties
        ii. port2 ไว้เชื่อมกับ MySQL container ซึ่งเราต้องใช้ 3306 เสมอ
    -

```
$ docker run --name=mysql_menu -e MYSQL_ROOT_PASSWORD=abc123 -e
MYSQL_DATABASE=menu -p 3307:3306 -d mysql
```

1. ปรับแก้ใน application.properties ให้ตรงกับการรัน container

```
server.port = 8090

# Datasource
spring.datasource.url=jdbc:mysql://localhost:3307/menu
spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=abc123

# JPA
spring.jpa.show-sql=true
Spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

2. สามารถเข้าไปดูใน database ใน docker container ได้

```
$ docker exec -it mysql_menu bash
# /# mysql -u root -p
# mysql> show databases;
# mysql> use menu;
# mysql> show tables;
# mysql> select * from menu;
# mysql> exit
# /# exit
```

Part 5: Going Inside a Container

20) ตรวจสอบผลการทำงานผ่านคำสั่ง docker logs

```
$ docker logs <container-name>
```

21) เข้าใช้ shell ของ container

- การสั่งรันที่มี sh เป็นการสั่งให้ docker รันแบบใช้ shell command ได้ด้วย

```
$ docker run --name <container_name> -it -d nginx sh
$ docker ps
```

- เราสามารถเข้าไปใน container ด้วยคำสั่งต่อไปนี้ จะเห็นว่าเราจะได้ shell

```
$ docker exec -it <container_name> sh
```

- เราสามารถใช้คำสั่ง Unix ทั่วไปได้ เช่น ls เพื่อลิสต์ไฟล์ทั้งหมด

```
/ #  ls
```

- ใช้คำสั่ง Unix เพื่อออกจาก shell

```
/ #  exit
```

หมายเหตุ : ปกติ เราเข้าไปใน container ในลักษณะนี้เพื่อตรวจสอบการทำงานเท่านั้น แต่เราจะไม่เข้าไปเพื่อ install โปรแกรมใด ๆ เพิ่มเติม หากต้องการโปรแกรมเพิ่มเติม ให้เลือก docker image ที่มาพร้อมกับโปรแกรมที่ต้องการ install นั้น หรือสร้าง image ของตนเอง (จะเรียนในแลปถัด ๆ ไป)

Part 6: Volume

22) เราสามารถสร้าง disk เพื่อให้หลาย container เข้าใช้ร่วมกันได้

```
$ docker volume create <YOURNAME-VOL>
$ docker volume list
$ docker run -it -v <YOURNAME-VOL>:/data nginx sh
```

สร้างไฟล์ใน /data
เมื่อเข้าใน container แล้ว ให้ลองสร้างไฟล์ในโฟลเดอร์ data

```
/ #  cd data
/data #  cat > myfile.txt
Hello
^C
/data #  ls
/data #  exit
```

ลองสร้าง container อีกตัวมาเข้าถึงโฟลเดอร์ data ของเรา โดยสามารถให้ image อื่นเข้าได้ด้วย เช่น เราจะใช้ image alpine ซึ่งเป็น Linux OS ยี่ห้อหนึ่ง

```
$ docker run -it -v <YOURNAME-VOL>:/data alpine sh
/ #  cd data
/data #  ls
```

หมายเหตุ เราสามารถใช้ bash แทน sh ได้ด้วย