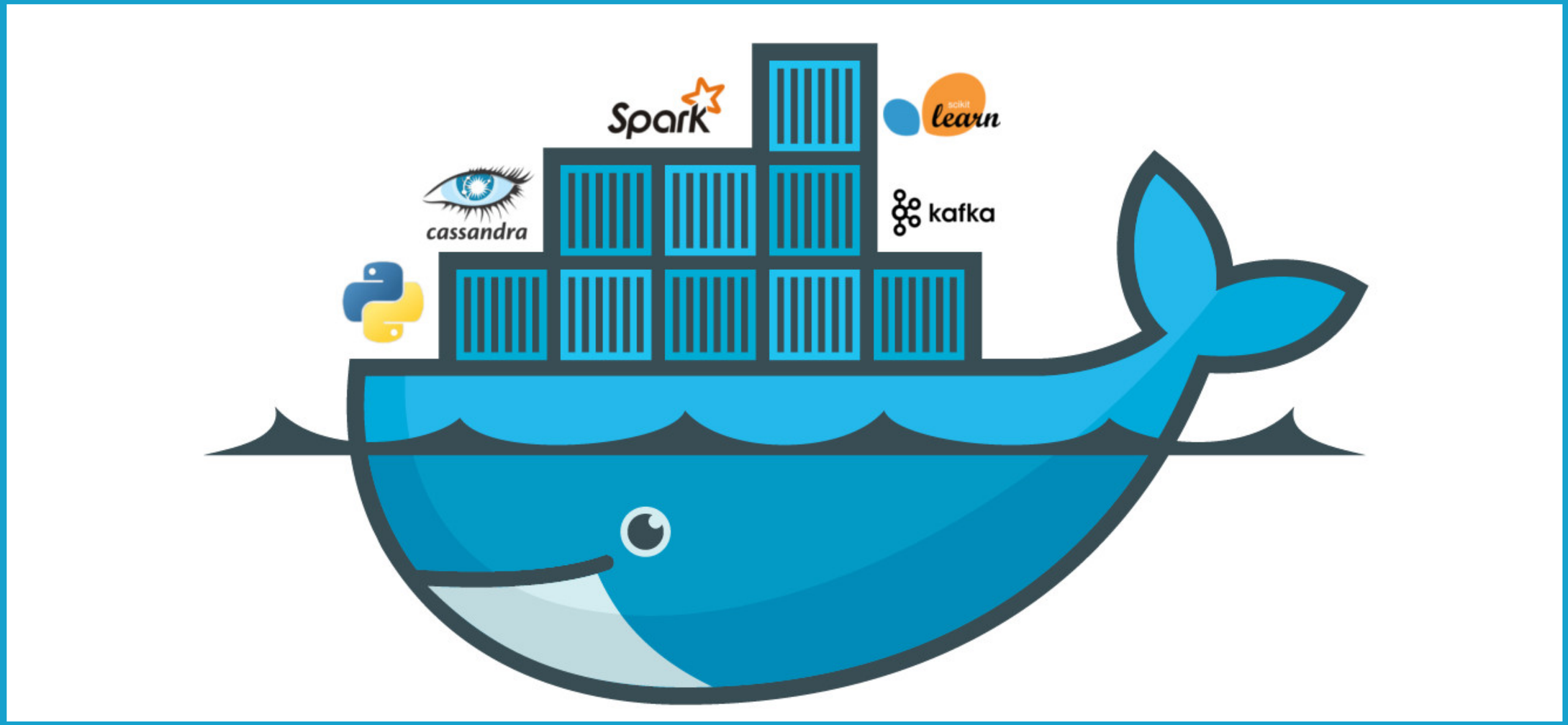
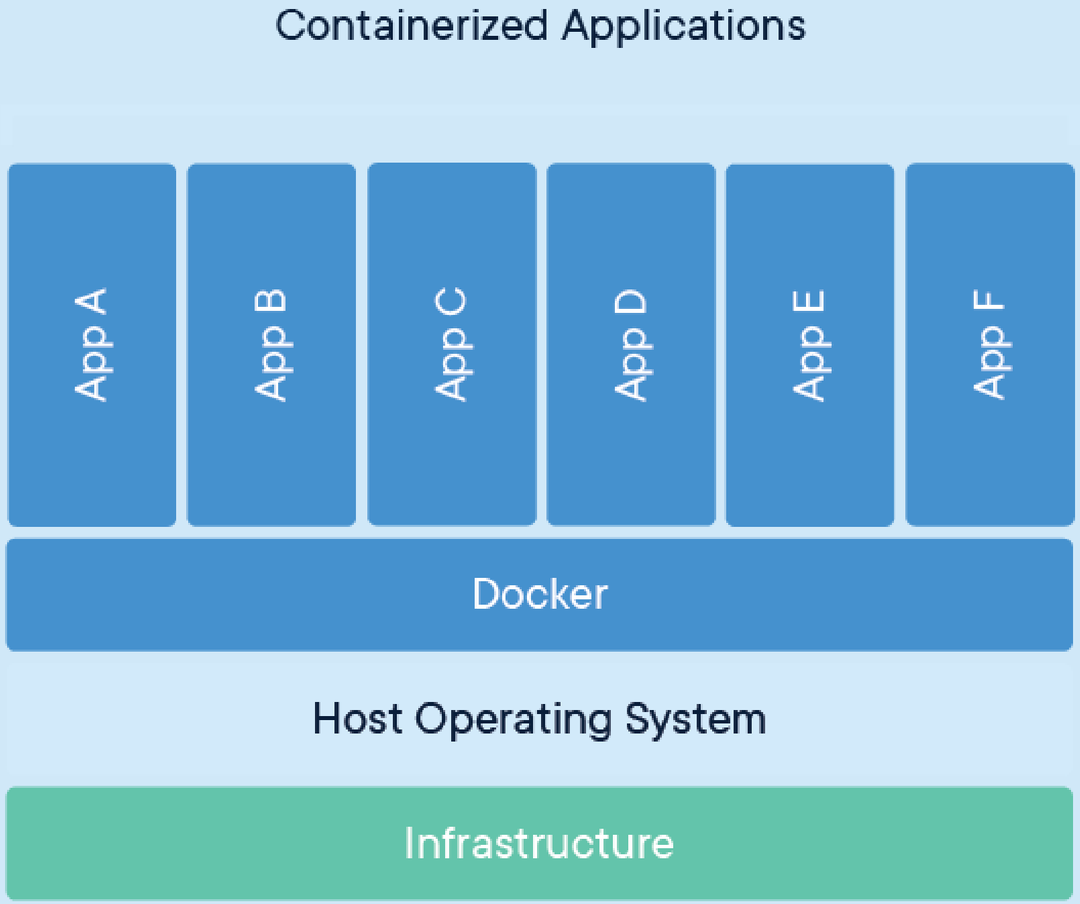
Docker

Docker คืออะไร?

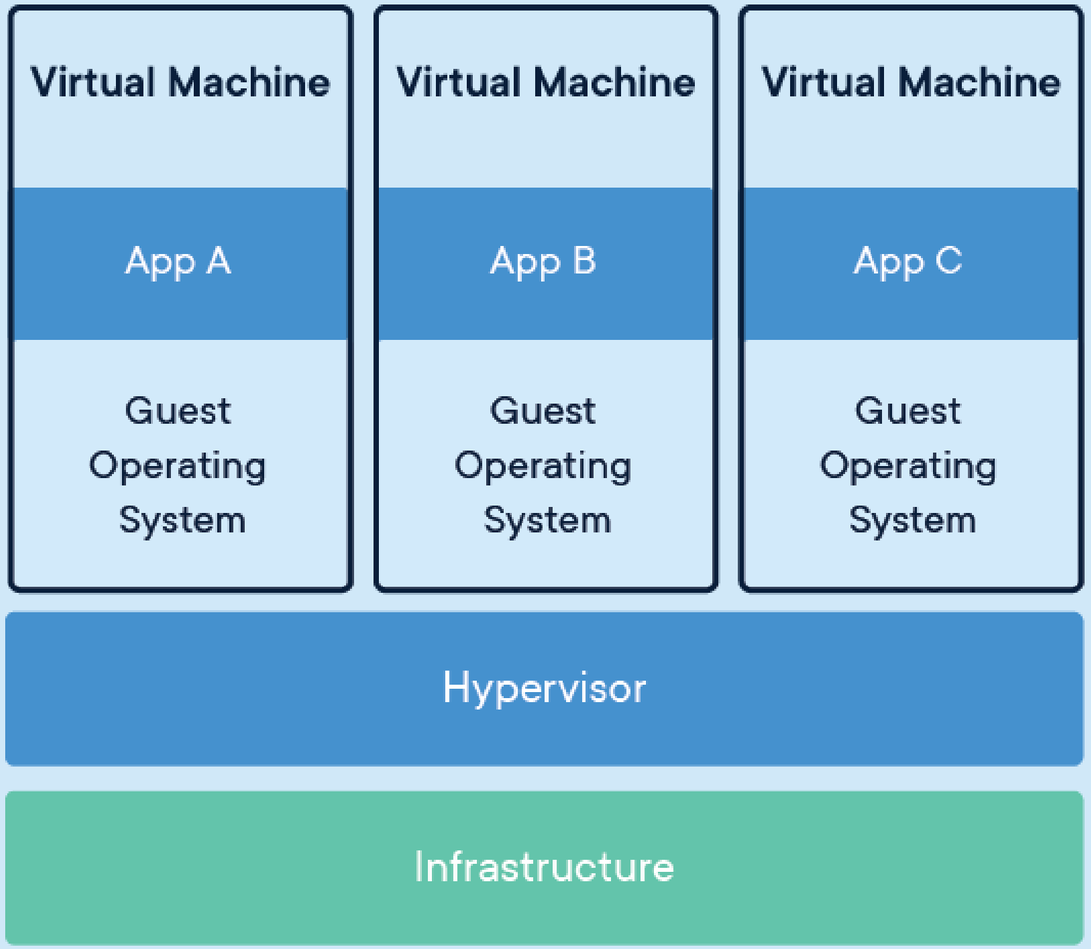


Docker คือ “Software Container”
ที่เป็นการสร้าง “สภาพแวดล้อมเฉพาะ” ให้กับซอฟต์แวร์ต่าง ๆ
และทำให้ซอฟต์แวร์เหล่านั้นทำงานบนเครื่องไหนก็ได้จะได้ผล
เหมือนกัน

Docker

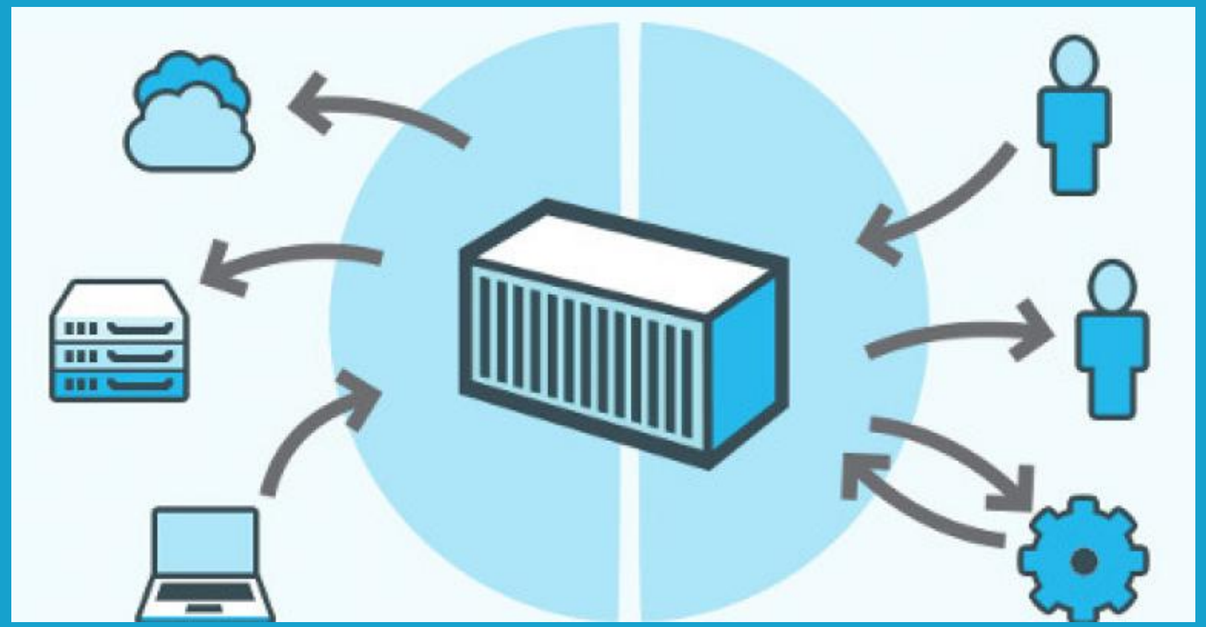

Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

VM


Virtual Machine
App A
Guest Operating System
Virtual Machine
App B
Guest Operating System
Virtual Machine
App C
Guest Operating System
Hypervisor
Infrastructure

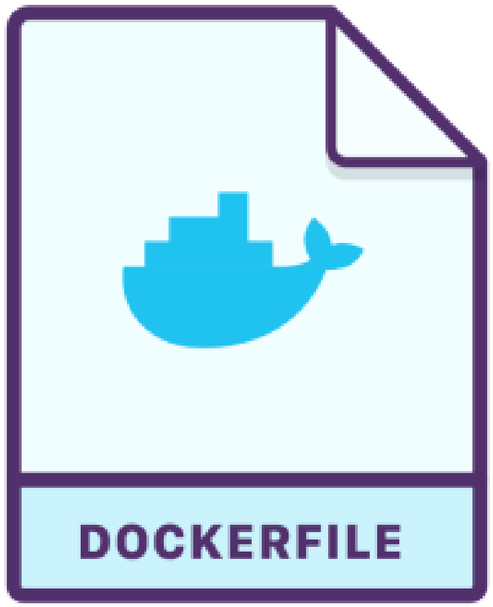
Docker container คืออะไร

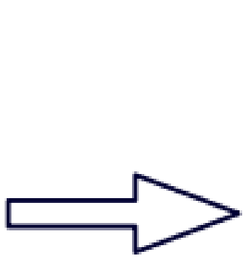
Docker container เปรียบเสมือนกล่อง ซึ่งนำ docker image มาติดตั้ง เพื่อให้สามารถใช้งาน service ที่ต้องการ จาก image นั้นๆ ได้

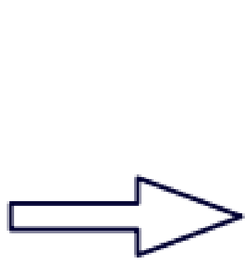


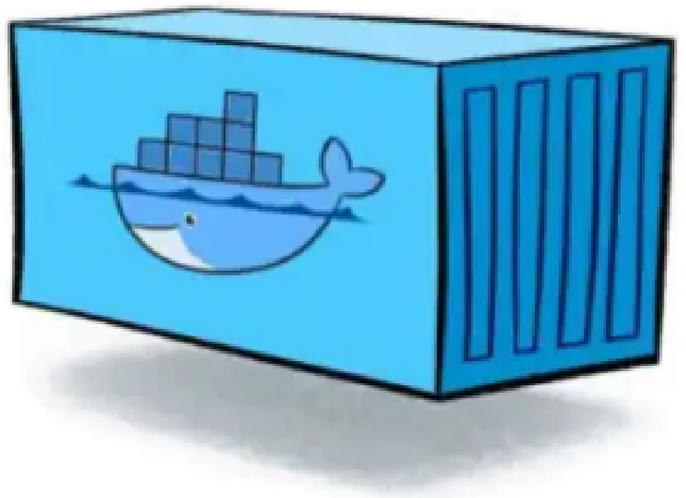
Docker image คืออะไร

Docker image เป็นตัวต้นแบบของ container ซึ่งภายในจะประกอบด้วย application ต่างๆ ที่มีการติดตั้งไว้เพื่อใช้งานสำหรับ service นั้นๆ รวมทั้งมีการ config ค่าต่างๆ ไว้เรียบร้อยแล้ว









Docker file

Docker Image

Docker Container

Dockerfile

```
FROM python:3.6-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
~
~
~
```

Dockerfile

FROM (image_name)	เป็นคำสั่งที่กำหนดว่าจะใช้ base image ตัวไหน
WORKDIR (Container path)	กำหนด working directory
COPY Current path / Container path	คำสั่งนี้ใช้ copy ไฟล์จาก directory ข้างนอก ต่างกับ ADD ตรงที่ใช้ได้เฉพาะไฟล์ที่อยู่ใน local เป็น remote url ไม่ได้
RUN (command)	คำสั่งนี้ใช้สำหรับบอกว่า จะให้ทำอะไรก่อนจะเริ่มรัน container
ENTRYPOINT ["echo","Hello"]	คำสั่งนี้คล้ายๆ CMD แต่ CMD เป็นเหมือนคำสั่ง default สามารถโดนเขียนทับได้ตอนที่สั่ง docker run แต่ ENTRYPOINT เป็นเหมือนคำสั่งที่รอรับ parameter จากคำสั่ง docker run
CMD ["Docker"]	คำสั่งนี้ใช้ run ตอนที่ build image เสร็จแล้ว ทั้ง ENTRYPOINT และ CMD สามารถใช้ร่วมกันได้

Dockerfile

EXPOSE (port)	คำสั่งเพื่อกำหนด port ให้ Docker
VOLUME /app	คำสั่งเพื่อกำหนด directory path สำหรับเก็บไฟล์ที่เราต้องการ
ENV USER = root	คำสั่งเพื่อกำหนดค่าตัวแปรสำหรับสภาพแวดล้อมในการทำงาน
ADD someFile.js /mydir/	คำสั่งเพื่อเพิ่มไฟล์ที่ต้องการเข้ามาในโฟลเดอร์ที่กำหนด

Dockerfile

CMD VS ENTRYPOINT

CMD

```
FROM centos:8.1.1911
CMD ["echo", "Hello Docker"]
```

```
$ docker run <image-id>
Hello Docker
$ docker run <image-id> hostname
 # hostname is exec to override CMD
244af....
```

ENTRYPOINT

```
FROM centos:8.1.1911
CMD ["echo", "Hello Docker"]
```

```
$ docker run <image-id>
Hello Docker
$ docker run <image-id> hostname
 # hostname as parameter to exec
Hello Docker hostname
```

Dockerfile

CMD + ENTRYPOINT

```
FROM centos:8.1.1911
ENTRYPOINT ["echo", "Hello"]
CMD ["Docker"]
```

docker run <image-id>

Hello Docker

docker run <image-id> Ben

Hello Ben

คำสั่งที่เกี่ยวกับ Docker

docker เป็นการแสดงคำสั่งทั้งหมด

docker version เป็นการแสดง version ของ docker

docker ps / docker container ls ใช้ดู container ที่กำลังทำงานอยู่

- ใช้ docker ps -a / docker container ls -a เพื่อดู container ทั้งหมด

docker inspect เป็นการแสดงรายละเอียดของ container

docker build เป็นการสร้าง docker image จาก Dockerfile

- docker build . หรือ docker build -t (tag ที่เราต้องการตั้ง) .

docker logs ใช้ดู log ของ docker container

- docker logs (ชื่อ container หรือ container id)

คำสั่งที่เกี่ยวกับ Docker

docker create สร้าง container จาก image

docker start ใช้ start docker container

- docker start (ชื่อ container หรือ container id)

docker stop ใช้ stop container

docker restart ใช้ restart container

docker rm ใช้ลบ docker container (ต้อง stop ก่อน)

- ลบโดยไม่ต้องทำการ stop ก่อน ให้ใช้คำสั่ง
docker rm -f (ชื่อ container หรือ container id)

คำสั่งที่เกี่ยวกับ Docker

docker pull เป็นการดึง docker image จาก Docker Hub Repository

- docker pull (ชื่อ image)

docker push เป็นการส่ง docker image ขึ้นไปเก็บไว้บน Docker Hub Repository

- ตั้ง tag ให้กับ image ของเราก่อน

docker tag (ชื่อ image ของเรา) ชื่อที่ต้องการตั้ง tag (ต้องขึ้นต้นด้วย docker id ของเรา / ชื่อ image ของเรา)

- docker tag hello-world mo0303/hello-world

- docker push mo0303/hello-world

คำสั่งที่เกี่ยวกับ Docker

docker save เป็นการ save docker image ให้อยู่ในรูปแบบไฟล์

- docker save -o redis.tar.gz redis

docker load เป็นการ extract file เป็น docker image

- docker load -i (path file image)
docker load -i /tmp/redis.tar.gz

คำสั่งที่เกี่ยวกับการ Transfer file

scp (Secure Copy) เป็นคำสั่งคัดลอก file ผ่านการเชื่อมต่อ ssh

- scp <file> <user>@<IP/Domain>:<file>

scp redis.tar.gz root@192.xxx.xxx.xxx:/tmp

rsync เป็นเครื่องมือที่ใช้ในการ sync ไฟล์ระหว่าง 2 directories บน Unix

rsync -Pav /root/MM/mo.html root@192.xxx.xxx.xxx:/tmp

-a คือ archive ทำให้ ไฟล์ตันฉบับกับไฟล์ปลายทางเหมือนกันทุกอย่าง

-v คือ verbose อธิบายรายละเอียดระหว่างทำงาน

-P คือ progress บอกความคืบหน้า

คำสั่งที่เกี่ยวกับ Docker

docker run เป็นคำสั่ง run container

- docker run --name myweb -p 8080:80 -v /myweb:/var/www/html -d nginx:latest
–name กำหนดชื่อ container name ถ้าไม่ระบุมันจะสุ่มตั้งชื่อมาให้เราเอง

-d เป็นการสั่ง container ให้รันแบบ background

-p เป็นการ map port ระหว่าง container เพื่อใช้สื่อสารกัน เช่น container ของ nginx จะใช้ port 80 เป็น default หากต้องการให้ใช้ port 8080 เราก็กำหนดให้เป็น -p 8080:80

-v คือการ mount volume เพื่อให้สามารถเรียกใช้ไฟล์ร่วมกันได้ ระหว่าง container กับเครื่องเรา อย่างเช่น /myweb:/var/www/html (ซ้ายคือ path ในเครื่องของเรา และด้านขวาคือ path ใน container)

คำสั่งที่เกี่ยวกับ Docker

docker exec -it เป็นคำสั่งเพื่อเข้าไปใน container

- docker exec -it CONTAINER_ID COMMAND

-i เป็นการเชื่อมต่อกับ STDIN ใน container

-t เป็นการแนบ shell ของเราเข้าไปใน terminal ของ container

docker run -it image /bin/bash

- สร้าง container และรันจาก image

จากนั้นรัน bash แสดงผลบนหน้าจอ และเข้าไปใน container

docker rename ใช้เปลี่ยนชื่อของ docker container

- docker rename container_id new_name

คำสั่งที่เกี่ยวกับ Docker

docker run -it –name container-name -v localhost-path:container-path image_name /bin/bash

สร้าง container โดย map disk local host ไปยัง container ตาม path ที่กำหนดไว้ ข้อดีคือสามารถ access data ได้จาก local host โดยไม่ต้องเข้าไปใน container และ share data path ระหว่าง container ได้ด้วย

docker run -it –name nettest1 –net mynetwork –ip 10.1.4.100 image_name /bin/bash

run container โดย assign network ที่สร้างขึ้น

คำสั่งที่เกี่ยวกับ Docker image

ส่วนใหญ่ใช้คำสั่งคล้ายกับ Docker container

docker images / docker image ls แสดง image ทั้งหมดที่มีของเรา

docker rmi (image name) ลบ image

docker image inspect ใช้ดูรายละเอียดของ image

docker image build ใช้สร้าง image จาก Dockerfile

คำสั่งที่เกี่ยวกับ Docker Network

docker network create เป็นคำสั่งเพื่อสร้าง network ใน container

- docker network create --driver=bridge --subnet=172.28.0.0/16 --ip-range=172.28.5.0/24 --gateway=172.28.5.254 mo-net

docker network ls แสดง network ที่มีทั้งหมด

docker network inspect (Network_name) แสดงรายละเอียดของ network

docker network inspect bridge

docker network rm (Network_name) ลบ network ที่กำหนด

คำสั่งที่เกี่ยวกับ Docker Volume

docker volume create เป็นคำสั่งเพื่อสร้าง volume ใน container

- docker volume create hello

docker volume ls แสดง volume ที่มีทั้งหมด

docker volume inspect (Volume_name) แสดงรายละเอียดของ volume

- docker volume inspect myvol

docker volume rm (Volume_name) ลบ volume ที่กำหนด

คำสั่งที่เกี่ยวกับ Docker prune

docker image prune ลบ image ที่ไม่ได้ใช้งานทั้งหมด

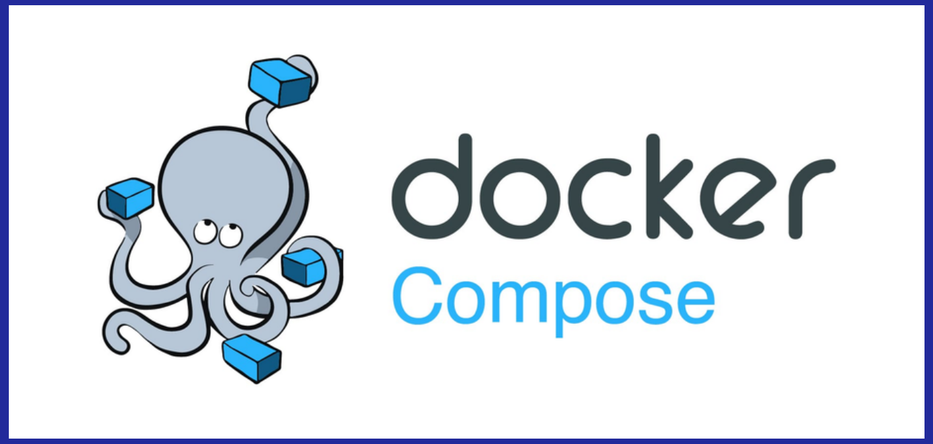
docker container prune ลบ container ที่หยุดทำงานทั้งหมด

docker network prune ลบ network ที่ไม่ได้ใช้งานทั้งหมด

docker volume prune ลบ volume ที่ไม่ได้ใช้งานทั้งหมด

Docker Compose คืออะไร?

Docker compose คือ script คำสั่ง ที่เอาไว้สร้าง container หลายๆอันขึ้นมาพร้อมกัน
โดยการสร้างไฟล์ที่ชื่อว่า docker-compose.yml เพื่อใช้ในการควบคุมการสร้าง container และกำหนด option ต่างๆที่ใช้ในการรัน container แต่ละตัว



Docker Compose

```yaml
version: "3.9"
services:
  show:
    build: .
    depends_on:
        web:
           condition: service_started
    image: "localhost:5000/mov3"
    networks:
        - mo-net
  web:
    image: "mo:v4"
    restart: always
    ports:
       - "80:80"
    healthcheck:
        test: ["CMD", "curl", "192.168.10.117:80"]
        interval : 5s
        timeout : 5s
        retries : 5
    volumes:
       - ../myweb/eiei.html:/usr/share/nginx/html/index.html
    networks:
        - mo-net
networks:
    mo-net:
```

Docker Compose

version	เป็นการระบุว่าเราจะใช้ Compose file เวอร์ชั่นไหน
services	เป็นการระบุ container ที่จะต้องใช้
image	เป็นการเรียกใช้ Image จาก Docker Hub Registry
ports	เป็นการทำ port mapping ระหว่าง host กับ container
volumes	การสร้าง volumes มี 2 แบบ ซึ่งถ้าสร้างอยู่ภายในชื่อ service แต่ละตัวก็คือการเชื่อม volume แต่ถ้าอยู่ในระดับเดียวกับ services: จะเป็นการสร้าง volume
build	การบอกว่าให้ใช้ image ที่สร้างจาก Dockerfile
restart: alway	เป็นการกำหนดให้ service นั้น restart ตัวเองอัตโนมัติเมื่อเกิดข้อผิดพลาด หรือสั่งให้เริ่มต้นทำงานอัตโนมัติเมื่อเปิดเครื่องขึ้นมาใหม่
network	เป็นการใช้เพื่อสร้างเส้นทางสื่อสารกันระหว่าง container
depens_on	สั่งให้ service นั้นเริ่มทำงานหลังจาก service ที่ depens_on อยู่เริ่มต้นทำงานเสร็จแล้ว

Docker Compose

HealthCheck เป็นการตรวจเช็คสถานะของ container
ว่าสามารถทำงานได้ปกติอย่างที่ต้องการได้หรือไม่

test	เป็นการระบุคำสั่งที่จะดำเนินการและเป็นการ healthcheck ของ container
interval	เป็นการระบุเวลาระหว่างการ health check ในแต่ละครั้ง
timeout	เป็นการระบุจำนวนวินาทีที่รอให้คำสั่ง health check return exit code ก่อนจะประกาศว่าล้มเหลว
retries	เป็นการระบุจำนวนความล้มเหลวจากการ health check ก่อนประกาศว่า container as unhealthy
start-period	เป็นการระบุจำนวนวินาทีที่ container จำเป็นต้องเริ่มต้นก่อนการ health check

คำสั่งที่เกี่ยวกับ Docker Compose

docker-compose build เป็นคำสั่งเพื่อสร้าง docker-compose file

docker-compose up เป็นคำสั่งเพื่อ create และ run docker-compose file

docker-compose down เป็นคำสั่งเพื่อ stop และ remove container

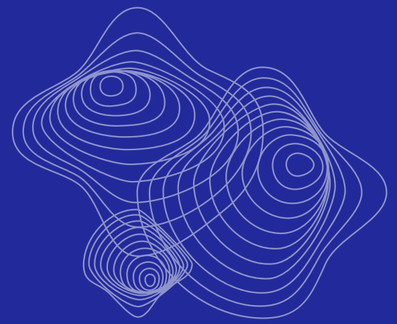
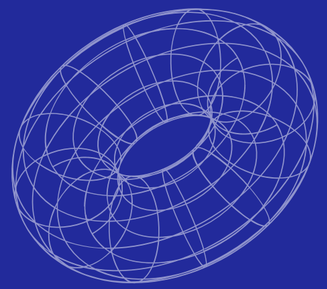
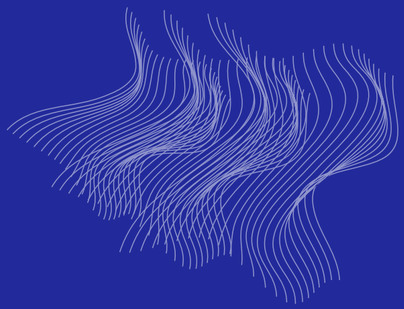
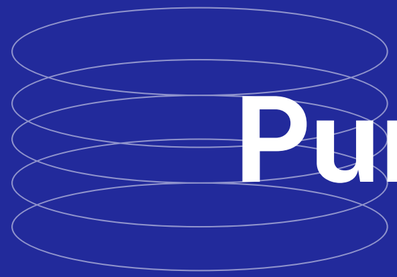
docker-compose ps เป็นคำสั่งเพื่อดูรายชื่อและสถานะของ container จาก docker compose

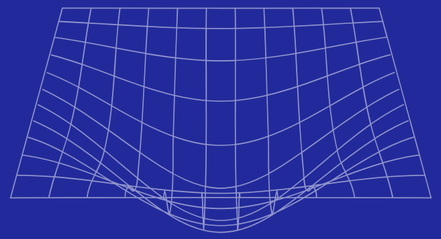
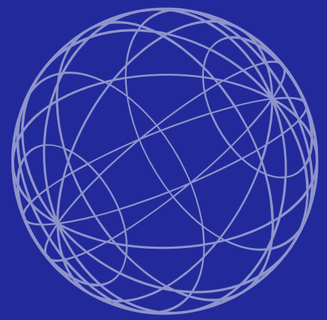
docker-compose images เป็นคำสั่งเพื่อดู รายชื่อของ image ใน docker compose

docker-compose rm ลบ container ที่หยุดการทำงาน

docker-compose run เป็นคำสั่งเพื่อ run 1 service ใน docker compose file

- docker compose run web bash


Puripat Rattanayuenyong