Research & Development Team

DevOps

www.pnpsw.com

sommai.k@gmail.com

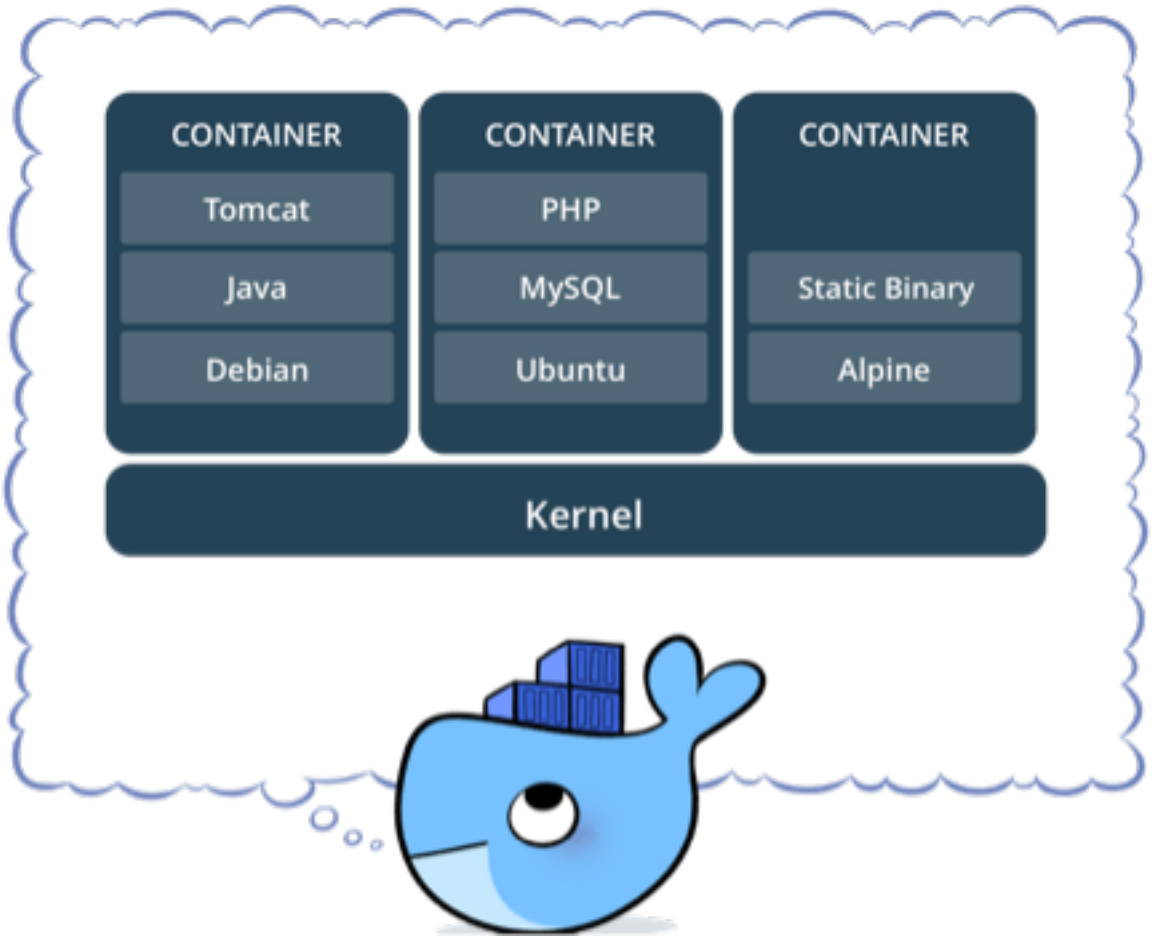
081-754-4663

Line Id : sommai.k

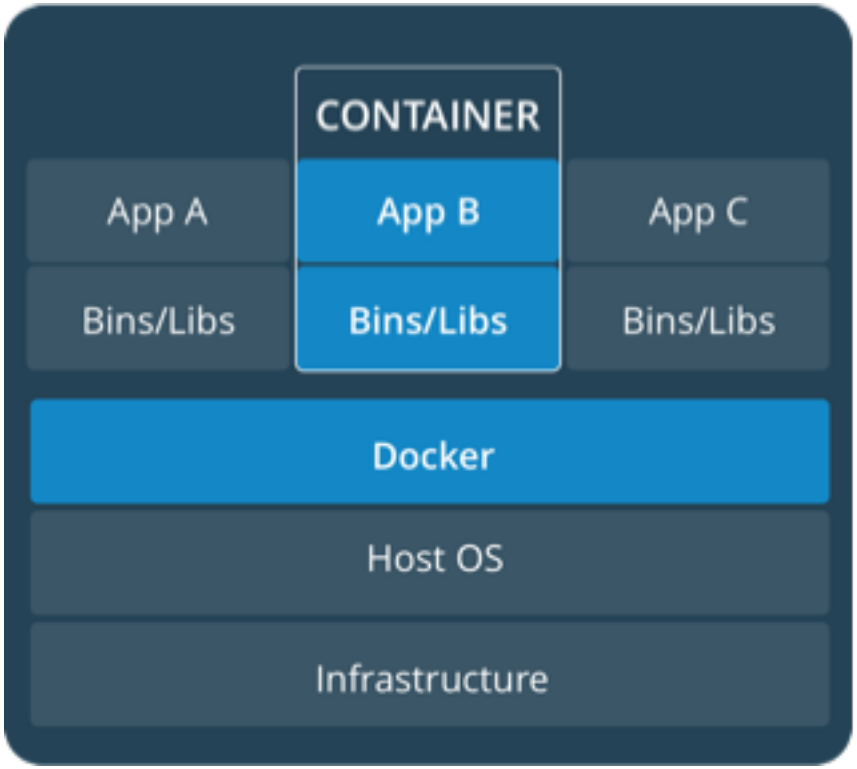
Containers virtualize with

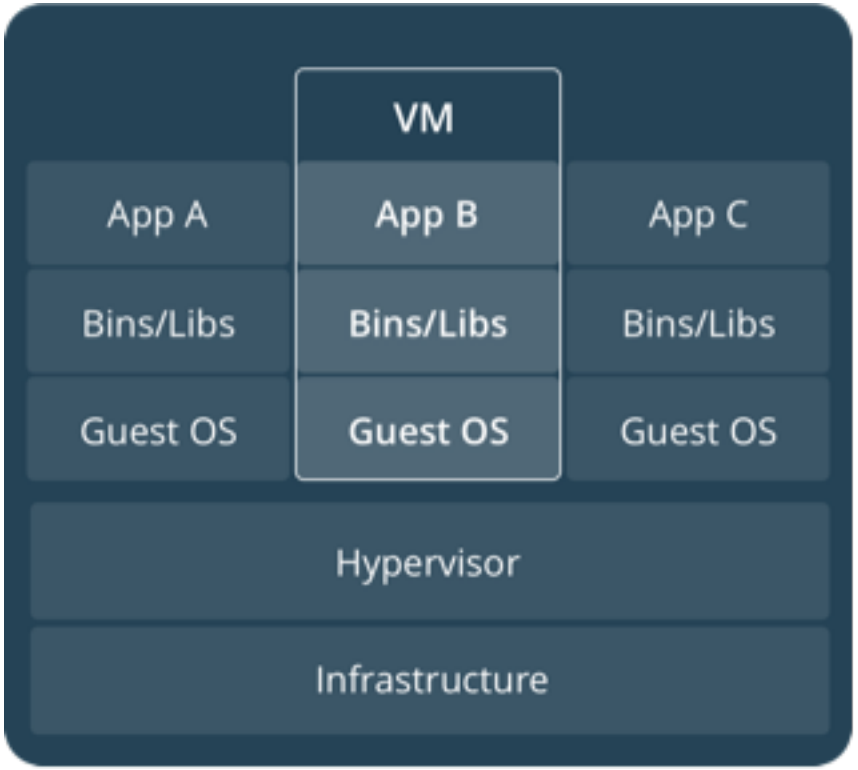
DOCKER

About Container

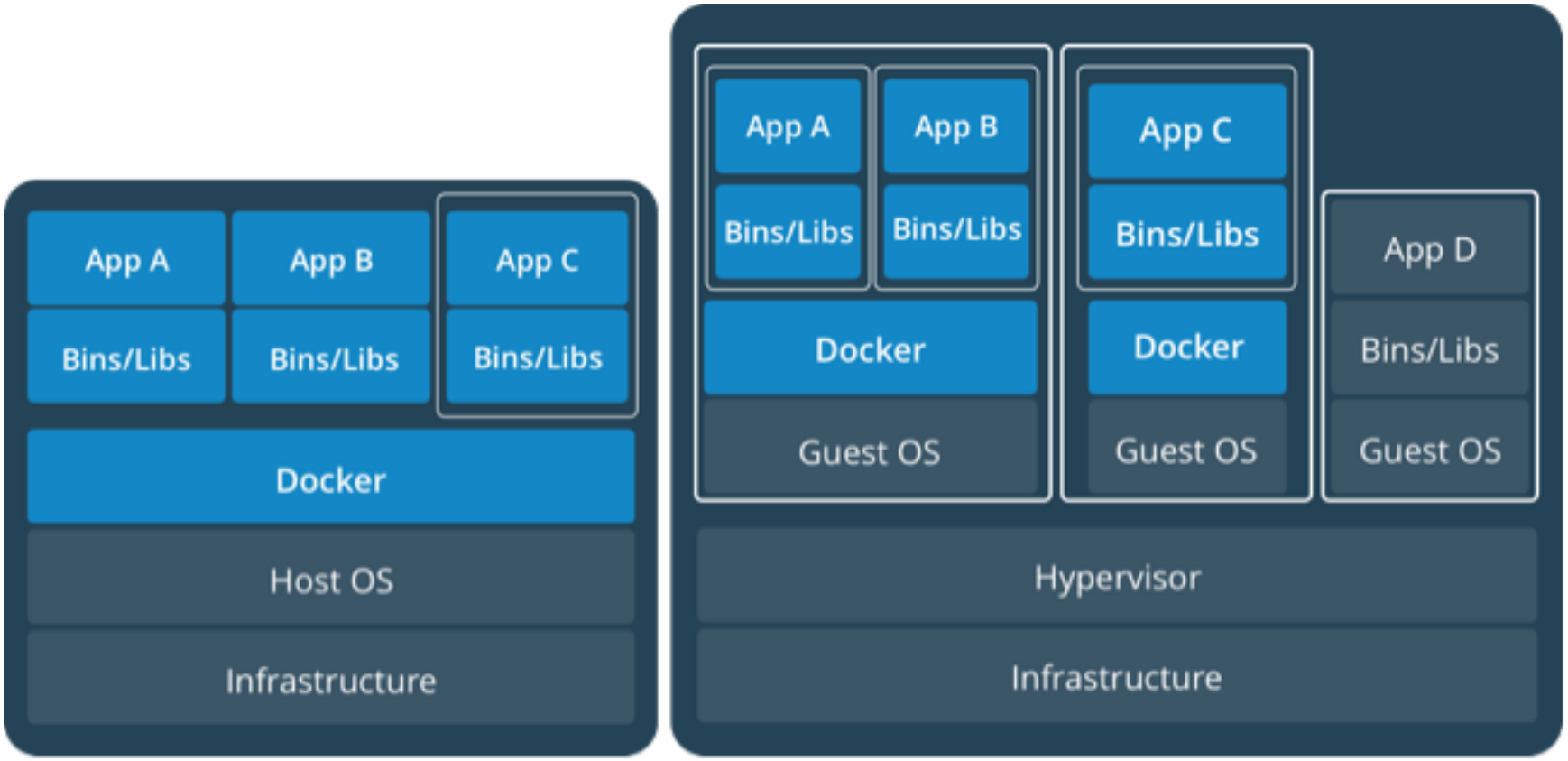


Comparing Containers and Virtual Machines

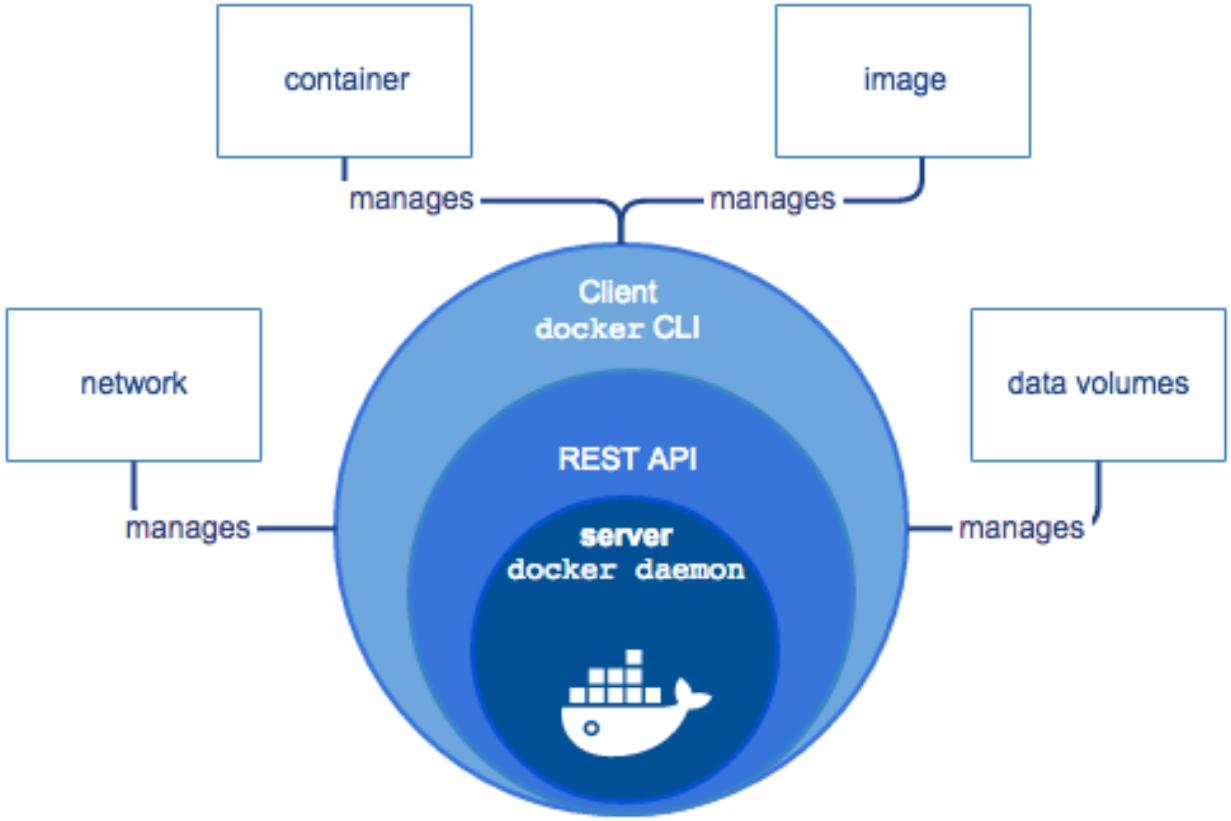




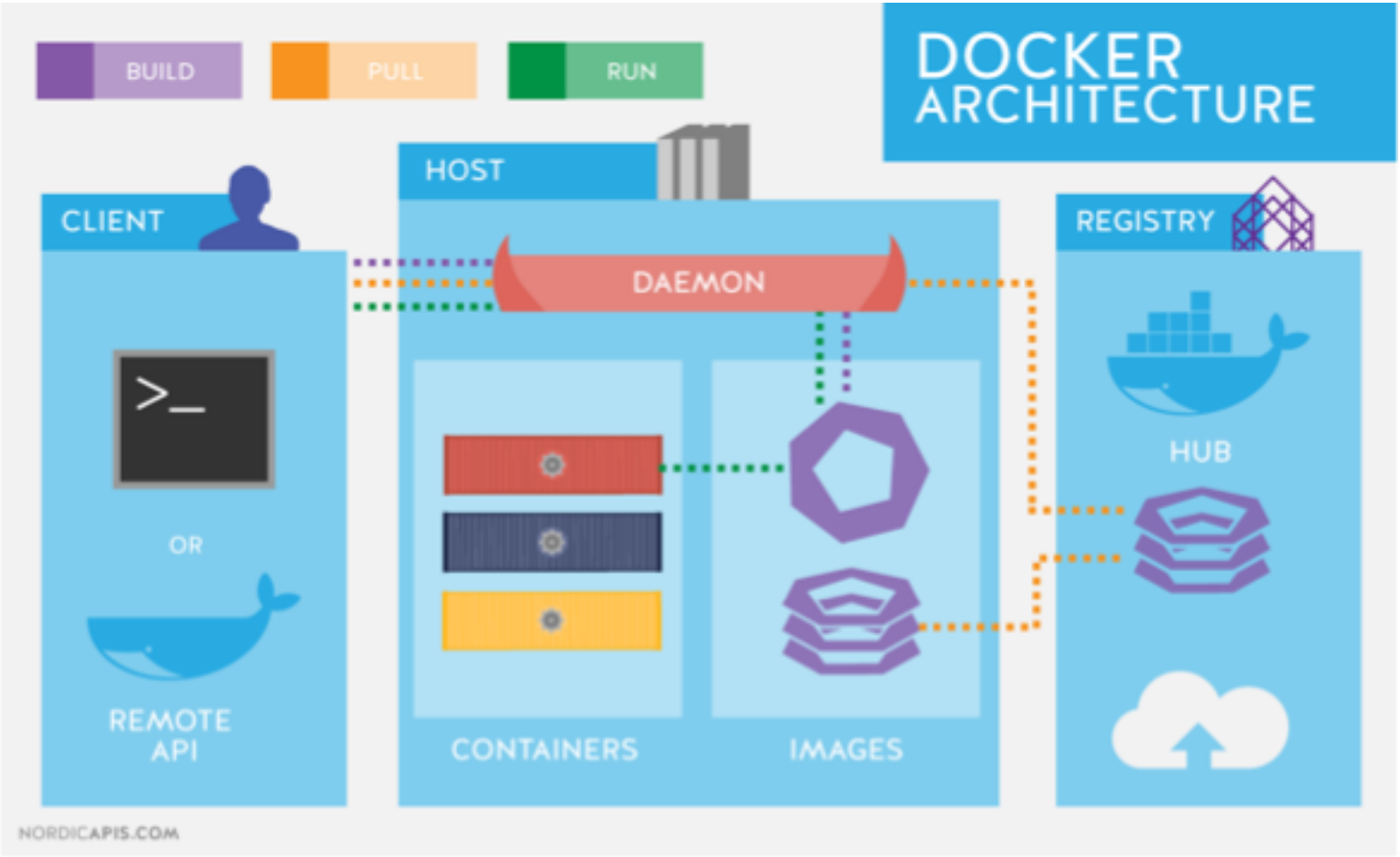
Containers and Virtual Machines Together



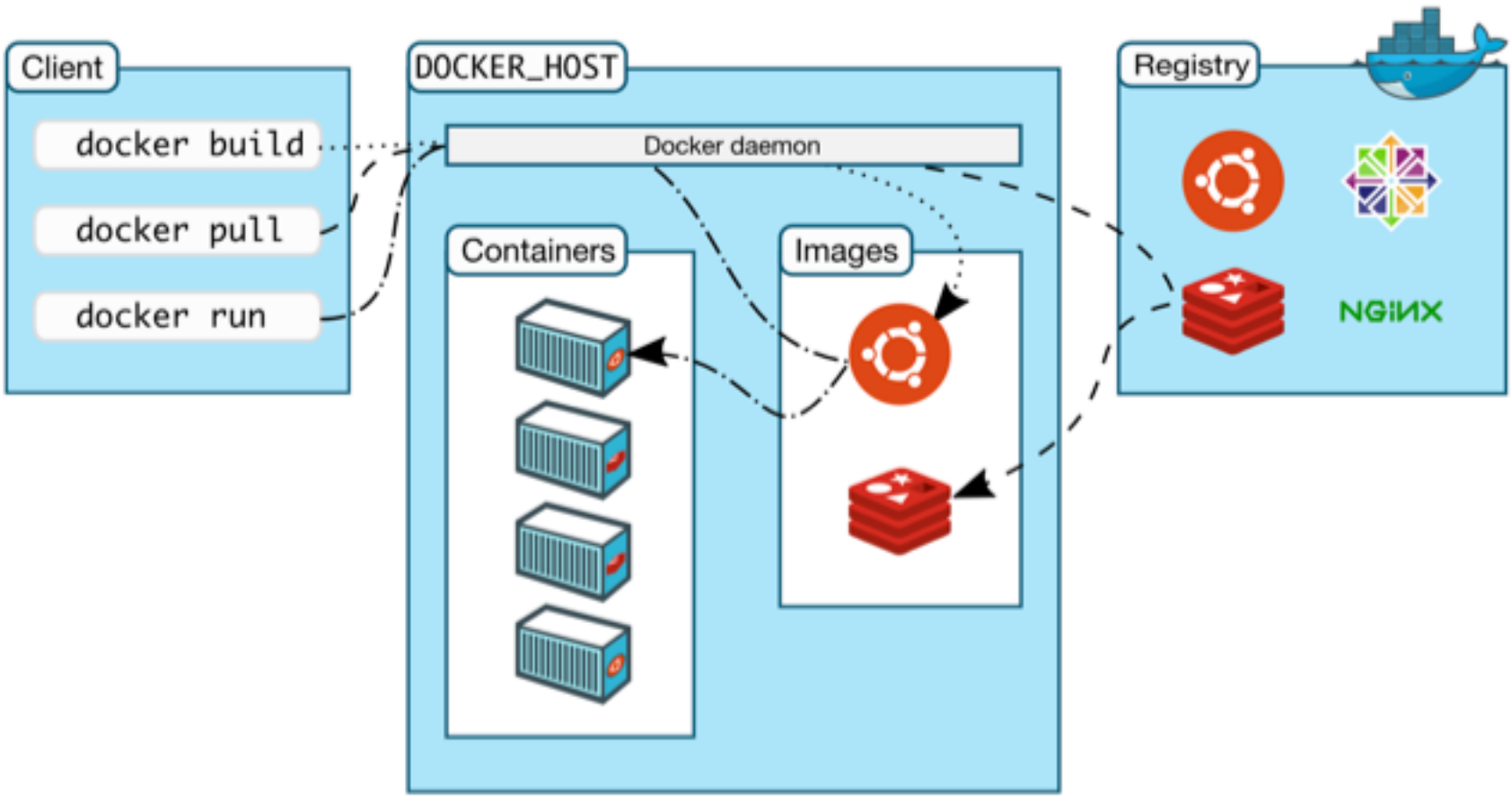
Docker engine



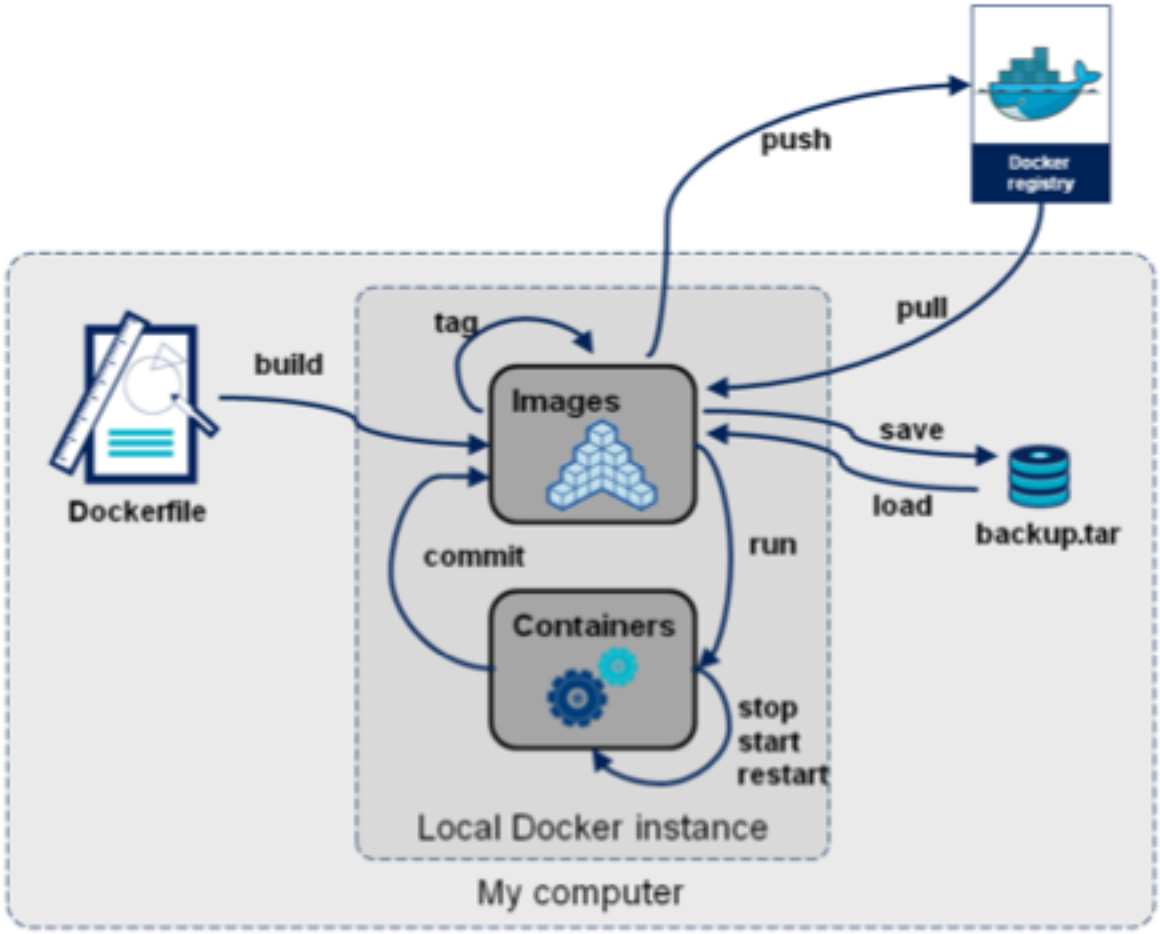
Docker Architecture



Docker Architecture#2



Docker Architecture#3



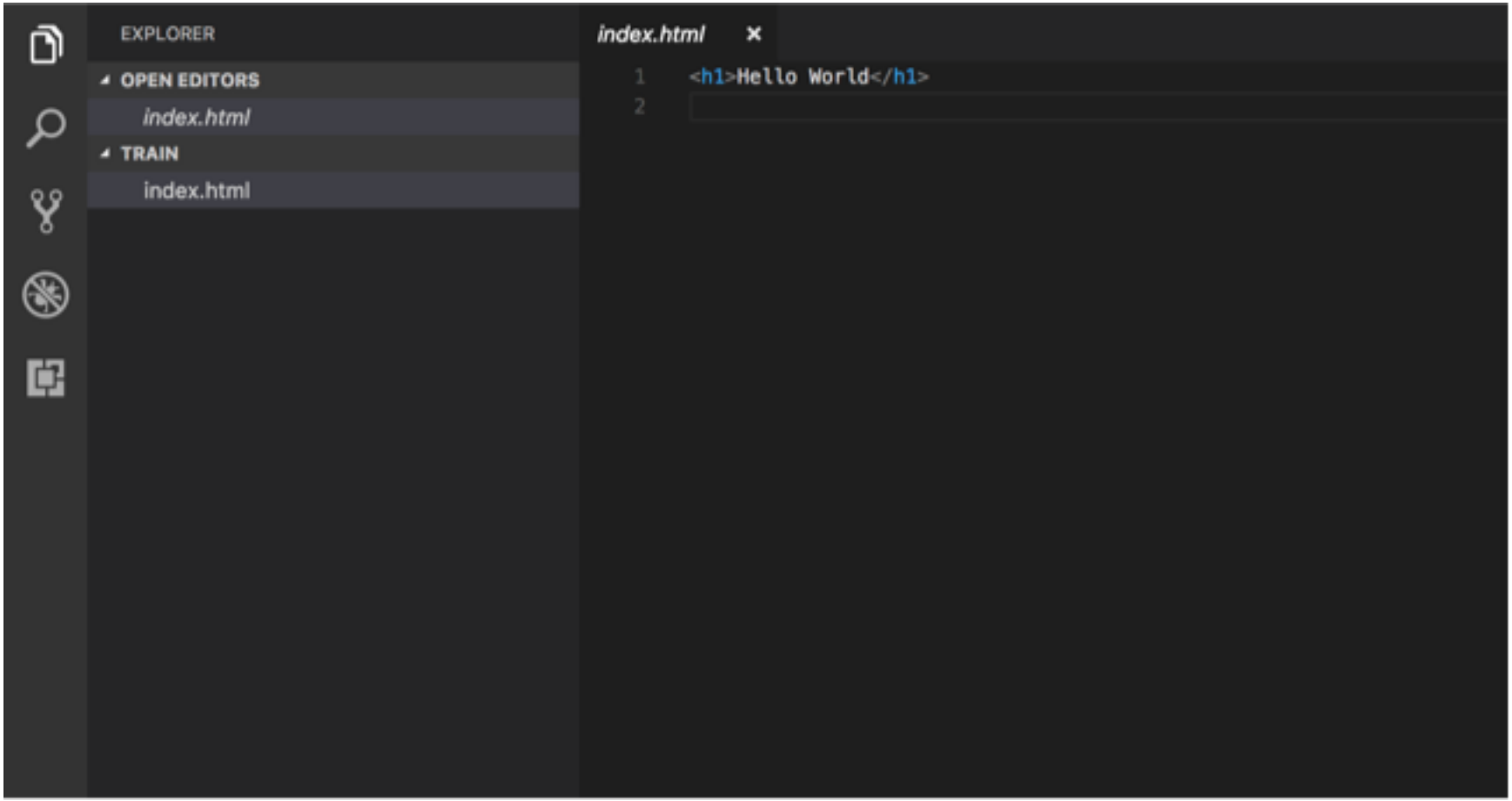
Hello World Docker

- เปิด cmd แล้วรันคำสั่งดังนี้

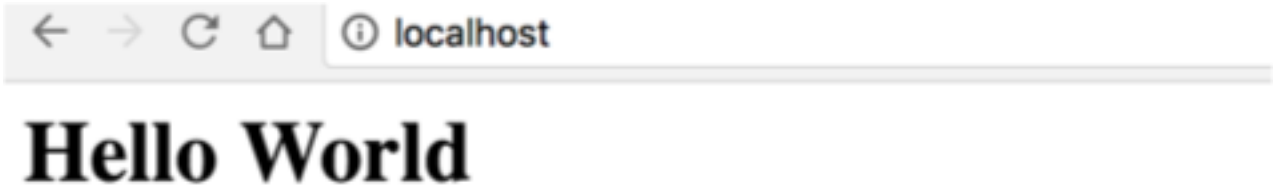
```
docker run --name some-nginx \
-v /your_path:/usr/share/nginx/html:ro \
-p 80:80 \
-d nginx
```

*** link สำหรับหา image https://hub.docker.com

สร้าง file index.html



เข้า url http://localhost



Docker Command

Login

- docker login
- docker login -u <user_name>
- docker login -u <user_name> -p <password>

Logout

- docker logout

List all image

- docker images
- docker image ls

Docker Command

Search image

- docker search <image name>

Pull image

- docker pull <image name>

Create container from image

- docker create <options> <image name>
  - --name
  - -v
  - -p

- docker create --name ubuntu14 -v /user/sommaik:/home ubuntu:14.04

Docker Command

Start Container

- docker start <container_id> or <container_name>

Stop Container

- docker stop <container_id> or <container_name>

Stop all container

- docker stop $(docker ps -a -q)

List all container

- docker ps <options>

Docker Command

Pause Container

- docker pause <container_id> or <container_name>

Unpause Container

- docker unpause <container_id> or <container_name>

Exec Container

- docker exec -it <container_id> bash

Inspect Container

- docker inspect <container_id>

Docker Command

Logs container

- docker logs

Commit Container

- docker commit <container_id> <new_image_name>
- docker commit 2x5t aloha

Push Image

- docker push <account>/<image name>

Tag

- docker tag ubuntu ubuntu-x

Docker Command

Export container

- docker export <container_id> **>** <to_path>

Import container

- docker import - <from_path>

Save Image

- docker save <image name> **>** <to path>
- docker save <image name>:<tag> **>** <to path>

Load Image

- docker load **<** <from path>

Docker Command

Remove container

- docker rm <container_id>

Remove all stop container

- docker rm $(docker ps -a -q)

Remove Image

- docker rmi <image_id>

Docker Network

- docker network ls
- docker network create <network_name> default bridge
- docker network create --subnet 10.0.0.1/24 <network_name>
- docker network inspect <network_name> or <container_id>
- docker network create my-net (create images networks)
- docker run --network <network_name> <image_name>
- docker run -it --name <container_name> --net--alias alias2 --network <network_name> <image_name>

Docker parameter

Run in the background

- -d

Create name to container is running

- --name

Port mapping

- -p (local_port:container_port)

Container host name

- -h

Docker parameter

Environment

- -e

Map volume paths

- -v

Keep STDIN open even if not attached

- -i

Allocate a pseudo-TTY

- -t

Dockerfile

FROM

- FROM <image>[:<tag>]
- FROM ubuntu:14.04

RUN

- RUN <command>
- RUN echo “Hello World”

EXPOSE

- EXPOSE <port>
- EXPOSE 8080

Dockerfile

ENV

- ENV <key> <value>

CMD

- CMD command param1 param2

WORKDIR

- WORKDIR /path/to/workdir

VOLUME

- VOLUME /path

Dockerfile

Build

- docker build <option> <path>
  - -t tag name

Example

- docker build -t first .

Compose file

file name

- docker-compose.yaml

Example

```yaml
version: '3'
services:
  jenkins:
    container_name: jenkins
    image: jenkins
    volumes:
    - ./jenkins:/var/jenkins_home
    ports:
    - 8080:8080
    - 5000:5000
  ubuntu:
    container_name: ubuntu14
    image: "ubuntu:14.04"
```

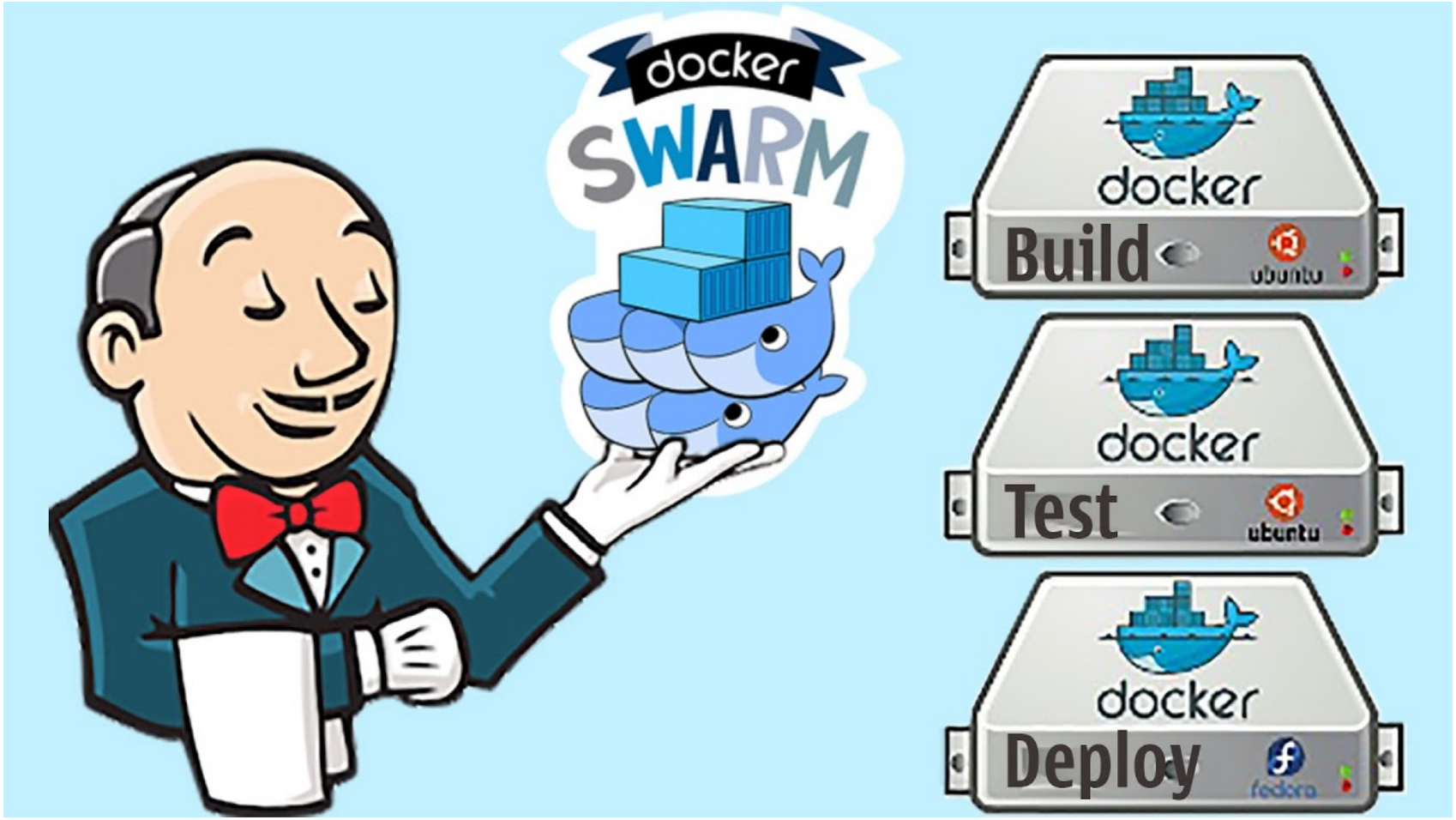
Docker Compose

- docker-compose up -d –build
- docker-compose up --force-recreate
- docker-compose ps
- docker-compose scale web=5
- docker-compose stop
- docker-compose rm

ORCRESTRATION WITH

DOCKER SWARM

Jenkins With docker swarm



Docker Swarm command

- docker swarm init : สำหรับเริ่มต้น docker swarm mode (manager)
- docker swarm join : สำหรับให้เครื่อง worker join เข้า manager
- docker swarm join-token : สำหรับสร้าง key เพื่อให้เข้ามา join
- docker service create : สร้าง service เพื่อให้บริการ
- docker service inspect : ตรวจสอบ service
- docker service ls : ดู service ทั้งหมด
- docker service rm : ลบ service
- docker service scale : เพิ่ม / ลด จำนวน node ที่ รัน service
- docker service ps : ดูสถานะ service
- docker service update : ปรับปรุง service
- docker node ls : ดู node ทั้งหมด

เริ่มสร้าง docker swarm (manager)

- runคำสั่ง ใน terminal ดังนี้

docker swarm init --advertise-addr=192.168.33.12

```
vagrant@mgr1:~$ docker swarm init --advertise-addr=192.168.33.12
Swarm initialized: current node (izupnbin1ra7cv2a9ur4k95ve) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3dip03ya9iixhhsk1aegfs3ptd30pwvuzpk7413iookg6nca99-egmkmkx8hvhh8fok2d8zfppmx 192.168.33.12:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

หลังจากที่รันคำสั่งเสร็จจะได้ผลลัพธ์เป็น เลข token เพื่อเอาไป run ที่เครื่อง worker ต่อไป

Join เข้ากลุ่ม swarm

- หลังจากที่ได้สร้างเครื่อง manager ไปแล้วก็ต้องนำคำสั่งที่ได้จาก console ของเครื่อง manager มารันในเครื่อง worker ดังตัวอย่าง

```
vagrant@node1:~$ docker swarm join --token SWMTKN-1-3dip03yo9iixhhsk1oegfs3ptd30pwvuzpk7413iookg6nco99-egmkmkx8hvhh8fok2d8zfppmx 192.168.33.12:2377
This node joined a swarm as a worker.
```

ตรวจสอบจำนวน node ทั้งหมด

- runคำสั่ง ใน terminal ดังนี้

docker node list

```
vagrant@mgr1:~$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
izupnbin1ra7cv2a9ur4k95ve *   mgr1                Ready               Active              Leader              18.03.1-ce
mdi3i3cjpknx8d58ujrfc9ujz     node1               Ready               Active                                  18.03.1-ce
```

สร้าง service บนเครื่อง manager

- runคำสั่ง ใน terminal ดังนี้

docker service create <<option>> nginx

** option ที่ควรมี

--replicas 2
--name web_server
--constraint "node.role != manager"
--publish 8080:80

replicas คือ จำนวน node ที่ต้องการสร้าง
name ชื่อที่เอาไว้อ้างอิง
constraint เงื่อนไขในการเลือกเครื่อง
publish คือ port ที่จะเปิดให้บริการจากภายนอก

ดู service ทั้งหมด

- runคำสั่ง ใน terminal ดังนี้

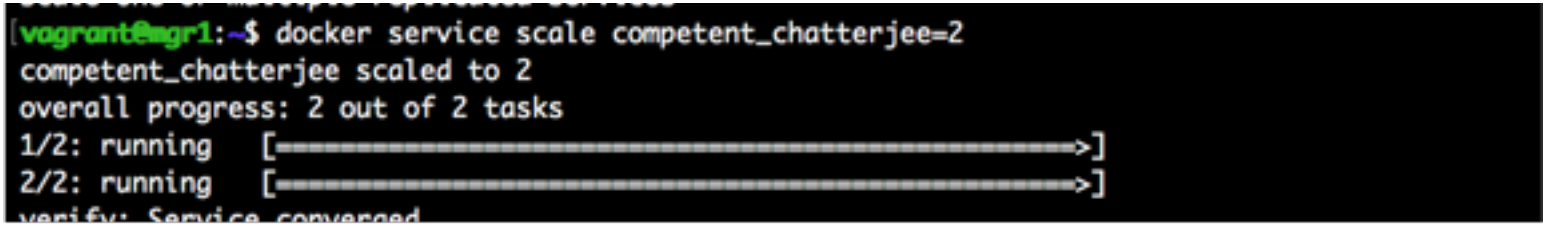
docker service ls

```
verify: Service converged
vagrant@mgr1:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE               PORTS
99nj4lwci1sp        competent_chatterjee   replicated          1/1                 nginx:latest
```

เพิ่มลดจำนวน node

- runคำสั่ง ใน terminal ดังนี้

docker service scale <service_name>=2



ตรวจสอบ service

- runคำสั่ง ใน terminal ดังนี้

docker service ps <service_name>

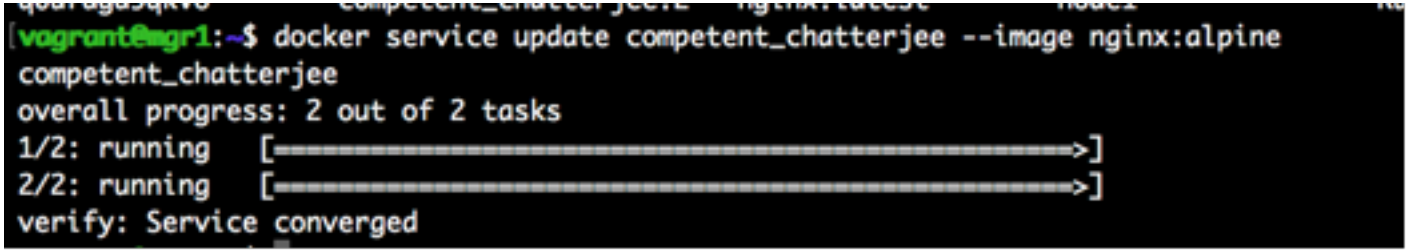
```
vagrant@mgr1:~$ docker service ps competent_chatterjee
ID                  NAME                    IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERROR               PORTS
txebpk4l9vsb        competent_chatterjee.1  nginx:latest        mgr1                Running             Running 6 minutes ago
q8u7dgd9qkvo        competent_chatterjee.2  nginx:latest        node1               Running             Running 37 seconds ago
```

จะแสดงข้อมูลของ service ว่าทำงานอยู่บน node ไหนตามจำนวน replicas ที่ได้ตั้งเอาไว้

update service

- runคำสั่ง ใน terminal ดังนี้

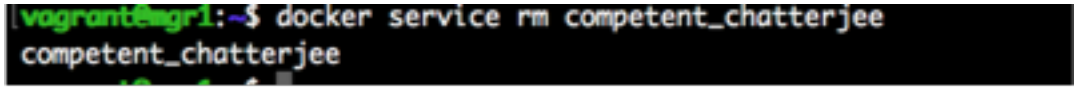
docker service update <service_name> --image nginx:alpine



ลบ service

- runคำสั่ง ใน terminal ดังนี้

docker service rm <service_name>



Any questions?