Docker Swarm Mode

Build your own Cluster

Swarm Mode

- Swarm เป็น feature ที่ใช้สำหรับทำ cluster หรือการขยายจำนวน instance ของ container เพื่อรับงานปริมาณมากๆ

- เดิม Swarm จะเป็นเครื่องมือแยกออกไปจาก Docker Engine แต่ใน version 1.12.0 Swarm จะถูกรวมเข้ามาใน Docker Engine โดยไม่ต้องติดตั้งเพิ่ม

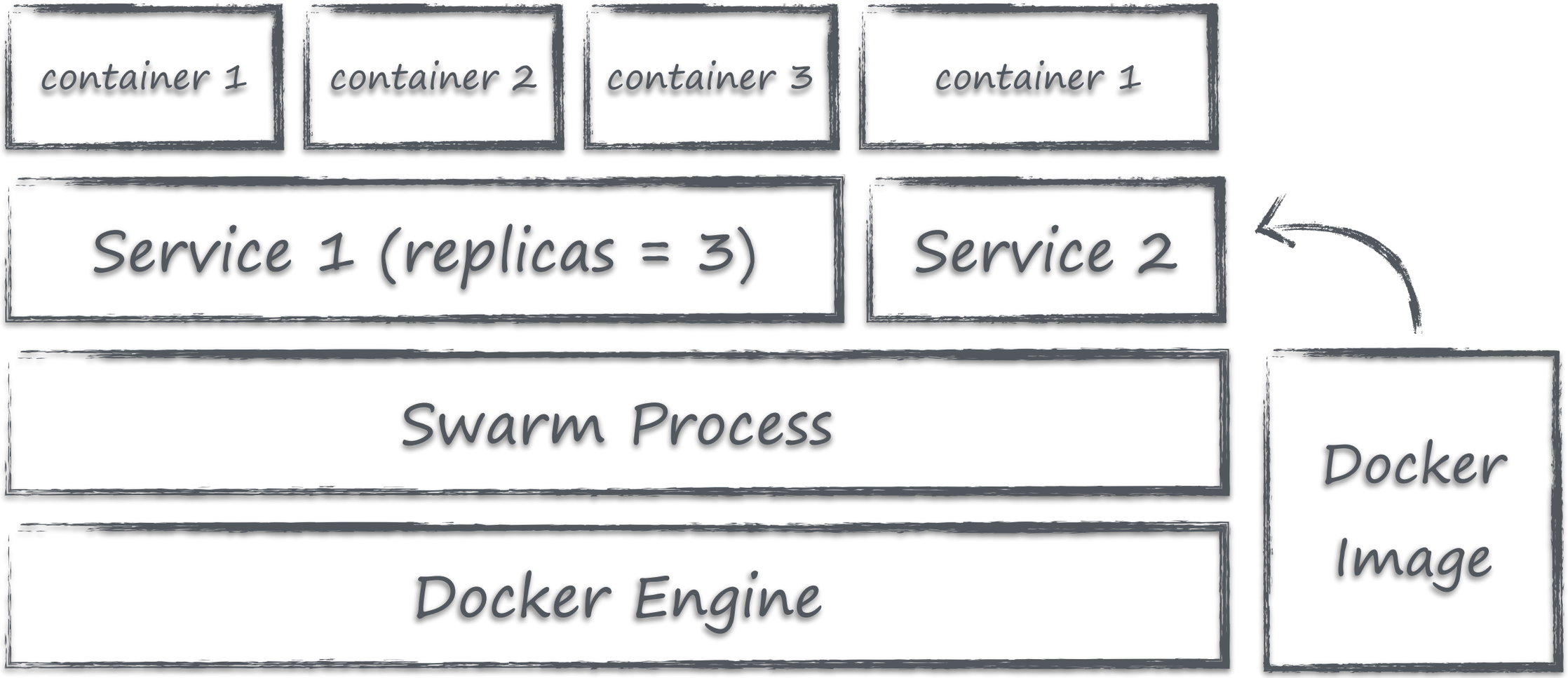
- เมื่อ Docker ทำงานแบบ Swarm จะเรียกว่า Swarm Mode

- ที่ผ่านมาเราใช้ container command ในการสั่งให้ container ทำงาน แต่เมื่อเป็น Swarm Mode เราจะสั่งผ่าน service ของ Swarm

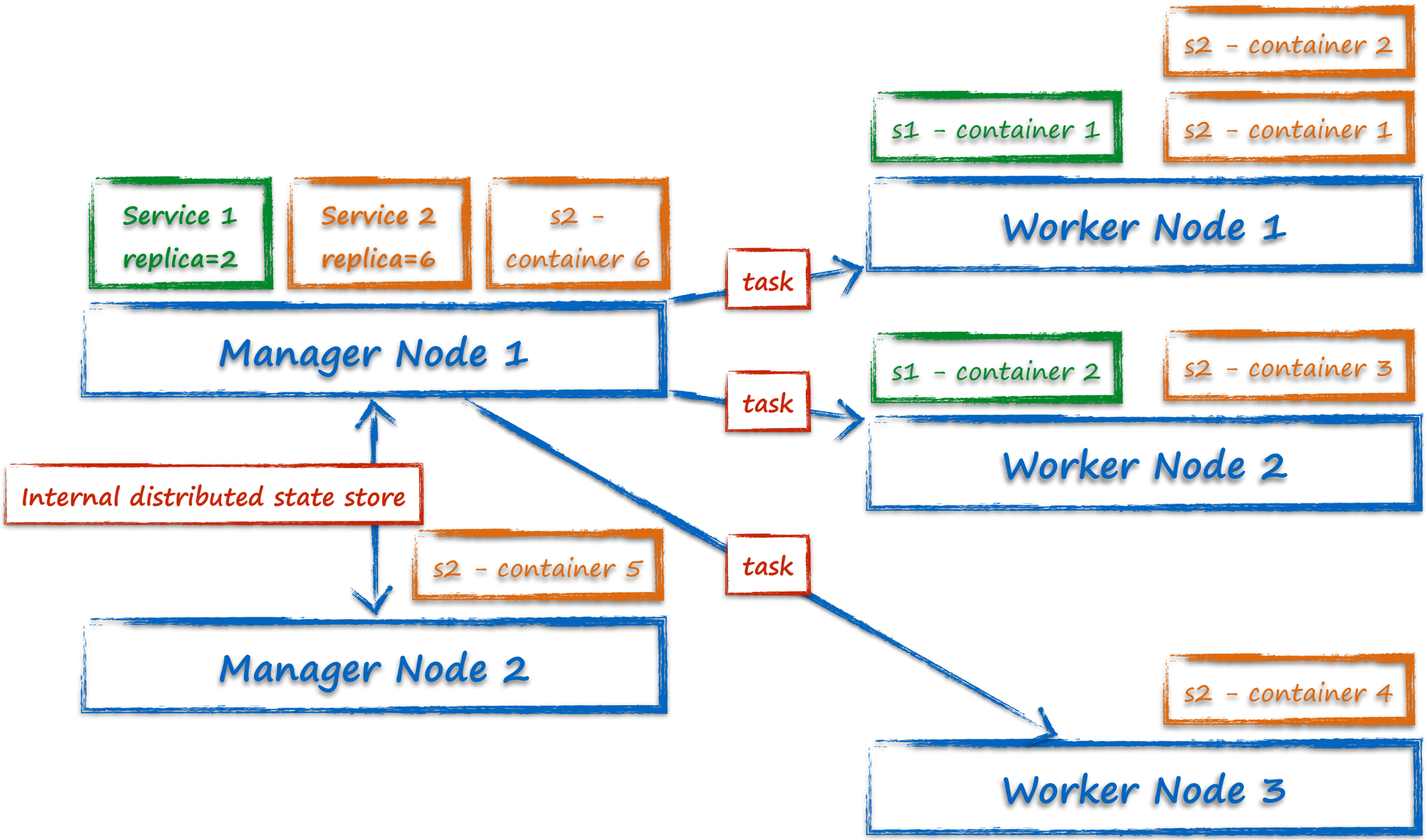
Docker Swarm Components

- Swarm
  - เป็น cluster management และ orchestration ใน docker engine
  - เราสามารถ deploy “service” ลงไปได้ โดยใช้ CLI (Command Line Interface) จัดการ เช่น เพิ่ม/ลด node, deploy service, start/stop
- Node
  - เป็น instance ของ docker engine ที่ทำงานร่วมกับ Swarm Cluster
  - มี Manager Node ทำหน้าที่รับ submit “Service” และมี Worker Node ทำหน้าที่รับ Task จาก manager node
  - Manager Node มีได้มากกว่า 1 node
- Services and Tasks
  - Service เป็น definition ของ Task ที่ถูก define ผ่าน manager node ไปยัง Worker node
  - เมื่อเราสร้าง service เราต้องกำหนด image เพื่อให้ Swarm นำไปสร้างเป็น container
  - Manager node จะทำการ replicate service ไปยัง worker node และสร้าง container ตามจำนวนที่เราระบุ เมื่อ task ถูก allocate ให้กับ worker node ใดแล้ว จะไม่สามารถย้าย task นั้นไปที่ node อื่นได้

Swarm Mode


container 1
container 2
container 3
container 1
Service 1 (replicas = 3)
Service 2
Swarm Process
Docker Engine
Docker Image

Swarm Node


Service 1
replica=2
Service 2
replica=6
s2 -
container 6
task
task
task
Manager Node 1
Manager Node 2
Internal distributed state store
s2 - container 5
s1 - container 1
s2 - container 2
s2 - container 1
Worker Node 1
s1 - container 2
s2 - container 3
Worker Node 2
s2 - container 4
Worker Node 3

Nodes

- Manager Node
  - maintaining cluster state
  - scheduling services
  - serving swarm mode HTTP API endpoints
  - recommends an odd number of nodes in H/A requirements
- Worker Node
  - execute containers
  - require manager node
- Node can be promote and demote

Load Balancer

- Swarm Manager ใช้ Ingress Load Balancer เพื่อรับ load จากภายนอกผ่าน PublishedPort

- Swarm mode จะมี DNS ภายใน ทำให้แต่ละ container มองเห็นกัน

- Swarm Manager และจะ assign package ให้กับ container ผ่าน internal load balancer ที่อยู่ใน swarm cluster เองอัตโนมัติ

Features Highlight

- Cluster management integrated with Docker Engine
- Decentralized design
- Declarative service model
- Scaling
- Desired state reconciliation
- Multi-host networking
- Service discovery
- Load balancing
- Secure by default
- Rolling updates

References :- https://docs.docker.com/engine/swarm/

Lab1: Setup

1. เตรียม VM ไว้ 3 เครื่อง แต่ละเครื่องใช้ memory 512MB - 1GB

2. Setup network ของเครื่อง VM แต่ละเครื่องเป็นแบบ “bridge” หรือ “NAT” เพื่อให้แต่ละเครื่องมองเห็นกัน (ping กันเจอ และ ping เจอจากนอกเครื่อง host)

3. **เมื่อ Start Virtual Machine ครั้งแรก จะมี dialog ถามว่า “I Moved It” หรือ “I Copied It” ให้เลือก “I Copied it”**

4. เปลี่ยนชื่อ VM 1 เครื่องเป็น **“manager”**

   ```
   $ sudo nano /etc/hostname
   $ sudo reboot
   ```

5. ทำซ้ำข้อ 4. เพื่อเปลี่ยนชื่อ VM อีก 2 เครื่องเป็น **“worker1”** และ **“worker2”**

6. จด IP Address ของเครื่อง manager ไว้ ด้วยคำสั่ง ifconfig
   (ในเอกสาร เครื่อง manager จะใช้ IP “192.168.99.100”)

   ```
   $ ifconfig
   ```

7. (optional) ลบ image บน worker ออกทั้งหมด

Lab2: Create Swarm (1/2)

1. ssh ไปที่ manager
2. สร้าง swarm ใหม่ที่ manager node โดยพิมพ์คำสั่ง init

```
$ docker swarm init
```

note: docker swarm จะ print ผลลัพธ์ออกมาเป็นชุดคำสั่ง ซึ่งคำสั่งแรกให้ worker node ใช้เพื่อ join swarm ชุดนี้ ส่วนคำสั่งที่ 2 ใช้เพื่อให้ manager node join swarm เพื่อทำ H/A

ตัวอย่าง

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

    docker swarm join \
    --token SWMTKN-1-6ctms3evwe4v61ovto98lczfx1z0chmj7j808yc2hafwtp4stg-cvzbcqtxhjx9brfrurbwgptns \
    192.168.99.100:2377

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

note2: ใช้ ip address ของเครื่อง manager แทน 192.168.99.100

Lab2: Create Swarm (2/2)

3. copy คำสั่ง docker swarm join สำหรับ worker node ที่ได้จากข้อ 2 ไปสั่งที่ worker1

```
$ docker swarm join \ --token SWMTKN-1-6ctms3evwe4v61ovto98lczfx1z0chmj7j808yc2hafwtp4stg-
cvzbcqtxhjx9brfrurbwgptns \
192.168.99.100:2377
```

4. ทำซ้ำข้อ 4 บน worker1 และ worker2

5. ที่ manager พิมพ์คำสั่ง docker node ls เพื่อดูรายชื่อของ swarm node ซึ่งเครื่องหมาย * อยู่ด้านหลัง ID จะเป็นตัวบอกว่า ID นี้คือเครื่องปัจจุบัน (node ls ทำงานบน manager node เท่านั้น)

```
$ docker node ls
ID                            HOSTNAME  STATUS  AVAILABILITY  MANAGER STATUS
5my12u0fcyv9pjq9f5nf4rawa     worker1   Ready   Active
6dx279ticqyh95sbbfv9pm36b     worker2   Ready   Active
91zd0ld5251i4pmxqlppgq0ph *   manager   Ready   Active        Leader
```

6. ดู node ที่ทำงานอยู่ใน cluster ด้วยคำสั่ง node ls

   $ docker node ls

7. ดูรายละเอียดของ Swarm ทั้งหมดด้วยคำสั่ง docker info

Join Swarm

- ในกรณีที่เราไม่ได้เก็บ token ไว้ตอนที่สั่ง swarm init เราสามารถสั่งให้ docker พิมพ์ token ใหม่ได้ ด้วยคำสั่ง swarm join-token

- join token มี 2 แบบ คือ token สำหรับ join เพื่อเป็น worker node และเพื่อเป็น manager node

- คำสั่งสำหรับ join เพื่อเป็น worker node

```
$ docker swarm join-token worker
```

- คำสั่งสำหรับ join เพื่อเป็น manager node

```
$ docker swarm join-token manager
```

Lab3: Prepare Dockerfile (1/3)

1. ssh ไปที่เครื่อง manager1 แล้ว cd ไปที่ home folder

   ```
   $ cd ~
   ```

2. สร้าง text file ตั้งชื่อว่า “Dockerfile”

   ```
   $ nano Dockerfile
   ```

3. เพิ่ม code ดังนี้

   ```
   FROM node
   MAINTAINER admin

   RUN git clone https://github.com/olarn/node-getip.git
   WORKDIR /node-getip
   RUN npm install
   EXPOSE 3000

   CMD ["node", "index.js"]
   ```

Lab3: Prepare Dockerfile (2/3)

4. Save Dockerfile

   - press ctrl+x
   - press y
   - press enter

5. Build docker image ด้วยคำสั่ง

   ```
   $ docker build -t admin/getip .
   ```

   note: มีจุด (.) ที่ตอนท้ายของ command ด้วย

6. ทดลอง list images ดู จะเห็นว่ามี image ชื่อ admin/getip” อยู่ในรายการ

   ```
   $ docker images
   ```

7. ทดสอบ docker image โดยการสร้าง container

   ```
   $ docker run -d --name test1 -p 3000:3000 admin/getip
   $ curl http://localhost:3000

      Hello, world! My IP = 172.17.0.3
   ```

8. ลบ container สำหรับทดสอบ image ออกไป (เก็บไว้เฉพาะ image)

   ```
   $ docker rm -f getip
   ```

Lab3: Distribute Docker Image (3/3)

Note: โดยปกติ docker engine ในเครื่อง worker จะค้นหา image จาก docker registry โดยไม่ต้อง copy ไปเอง ถ้าเรามี account ของ remote docker repository เราสามารถ push image ไปเก็บไว้ได้ เช่นที่ Docker Hub หรือจะใช้ private repository เช่น VMWare Harbor ก็ได้

9. Backup image

```
$ docker save -o ~/getip.tar admin/getip
```

10. copy image ไปยัง worker1 และ worker2 (สมมติว่า IP ของ worker เป็น 192.168.99.101 และ 102)

```
$ scp getip.tar docker@192.168.99.101:/home/docker/getip.tar
$ scp getip.tar docker@192.168.99.102:/home/docker/getip.tar
```

11. ssh ไปที่เครื่อง worker1 ทำการ restore image

```
$ cd ~
$ docker load -i ./getip.tar
```

12. ทำซ้ำข้อ 11 กับเครื่อง worker2

Lab4: Deploy the Service (1/2)

1. ที่ manager พิมพ์คำสั่ง

   ```
   $ docker service create --replicas 1 --name getip -p 3000:3000 admin/getip
   ```

2. เช็คการทำงานของ service โดยพิมพ์คำสั่ง

   ```
   $ docker service ls
   $ docker service ps getip
   ```

3. เมื่อ service เริ่มทำงาน docker จะสร้าง container ขึ้นมารองรับงานนั้น เช็คการทำงานของ container ด้วยคำสั่ง ps

   ```
   $ docker ps
   ```

4. ที่เครื่อง host (นอก VM) ทดลอง browse ไปที่ node เพื่อดูว่าสามารถติดต่อกับ node ได้หรือไม่ โดยเปิด browser แล้ว browse ไปที่ IP ของเครื่อง VM manager

   http://192.168.99.100:3000

   ควรจะได้ผลลัพธ์เป็น

   ```
   Hello, world! My IP = 10.255.0.6
   ```

   โดย IP ที่ได้ จะเป็น IP ที่อยู่ใน docker network ที่ถูกสร้างจาก docker swarm

Lab4: Deploy the Service (2/2)

5. เพิ่มจำนวน service ด้วยคำสั่ง scale

   ```
   $ docker service scale getip=5
   ```

6. เช็คว่า service ถูก scale ไปที่ node ไหนบ้างด้วยคำสั่ง ps จะเห็นว่า service ถูกกระจายไปยัง node ต่างๆ

   ```
   $ docker service ls
   $ docker service ps getip
   ```

7. ทดลองใช้ browser ทำการ browse ไปที่เครื่อง worker node ว่าสามารถเรียกใช้ web ได้หรือไม่

8. ดูรายละเอียดของ service ที่เครื่อง manager ด้วยคำสั่ง inspect

   ```
   $ docker service inspect getip
   $ docker service inspect --pretty getip
   ```

Lab5: Clean Up (1/2)

1. ลบ service ออกด้วยคำสั่ง rm

   ```
   $ docker service rm getip
   ```

2. ตรวจสอบว่า service หยุดทำงานแล้วด้วยคำสั่ง ls, ps และ inspect

   ```
   $ docker service ls
   $ docker service ps getip
   $ docker service inspect getip
   ```

3. ลบ worker ออกจาก swarm cluster โดย

    1. ssh ไปที่เครื่อง worker2 แล้วพิมพ์คำสั่ง

       ```
       $ docker swarm leave
       ```

    2. ssh ไปที่ manager node แล้วพิมพ์คำสั่ง

       ```
       $ docker node rm worker2
       ```

    3. เช็คดูว่า node ถูกลบออกไปแล้วด้วยคำสั่ง node ls

       ```
       $ docker node ls
       ```

Lab5: Clean Up (2/2)

4. ทำซ้ำข้อ 3 เพื่อลบ worker1 ออกจาก swarm

5. ลบ swarm ออกโดย ssh ไปที่เครื่อง manager แล้วพิมพ์คำสั่ง

```
$ docker swarm leave --force
```

References

- Docker Swarm Mode :- https://docs.docker.com/engine/swarm/

- Tutorial :- https://docs.docker.com/engine/swarm/swarm-tutorial/

- Using Compose with Swarm :- https://docs.docker.com/compose/swarm/