Docker Certified Associate (DCA)

Create by: **Hadi Tayanloo**

Phone : +98-912-8387233

Linkedin: linkedin. com/in/htayanloo/

Session 7

- Docker service
- Docker Swarm usefull Command
- Docker Service
- Docker Network

Docker Service Master

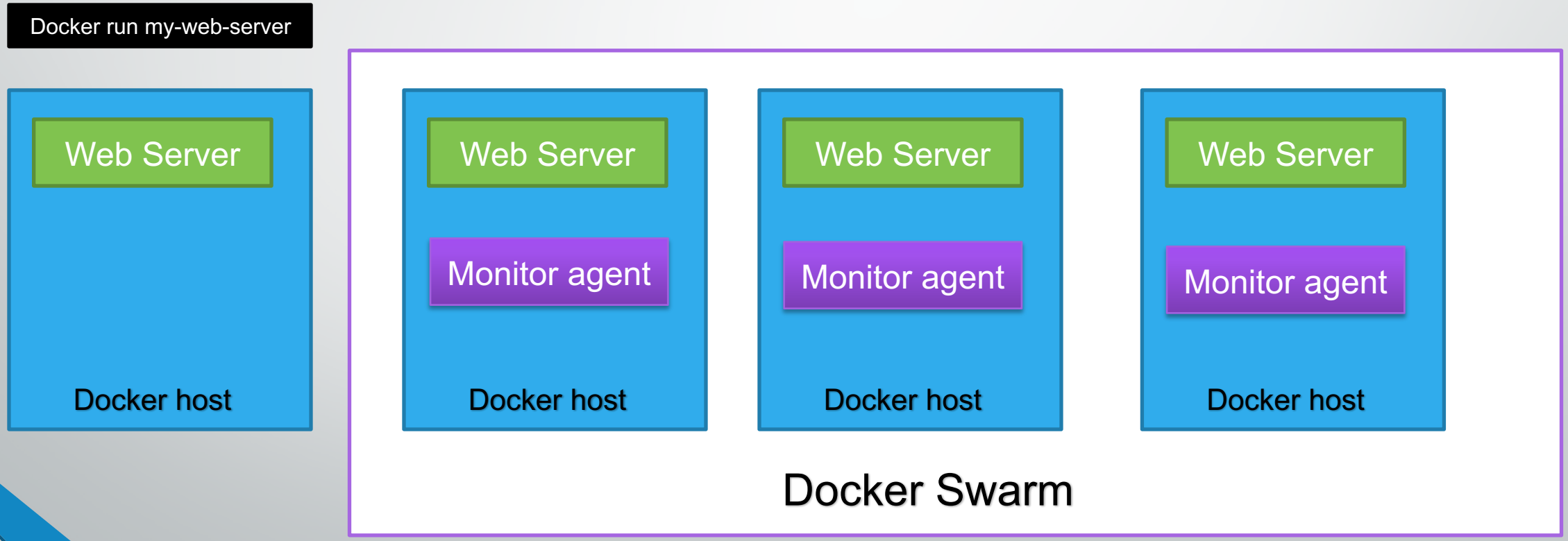
- **Service Visualizer**
  - docker service create \
    - --name=viz \
    - --publish=8080:8080/tcp \
    - --constraint=node.role==manager \
    - --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
      dockersamples/visualizer

Docker service

```
Docker service create -- replicas=3 my-web-server
```

```
Docker service create -- mode global my-web-server
```

Docker run my-web-server



Docker Service Master

- **Create service**
- Docker service create nginx
- Docker service create --replicas=3 nginx
- Docker service create --mode global nginx

Docker Service Master

- **Create service**

- docker service create --name dns-cache \
- --publish published=53,target=53 \
- dns-cache
- docker service create --name dns-cache \
- -p 53:53 \
- dns-cache

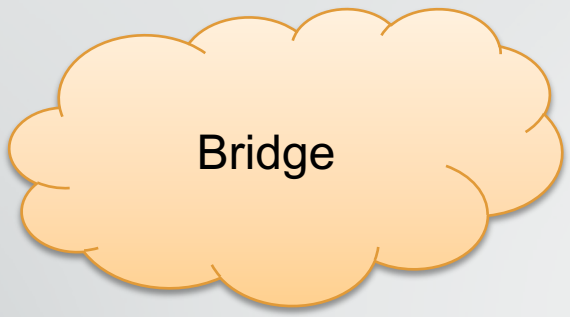
Docker Service Master

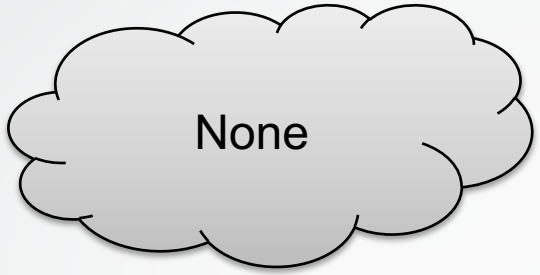
- **Service Monitor**
- Location of each container on nodes:
  - Docker service ls
  - Docker service ps (service_id or service_name)

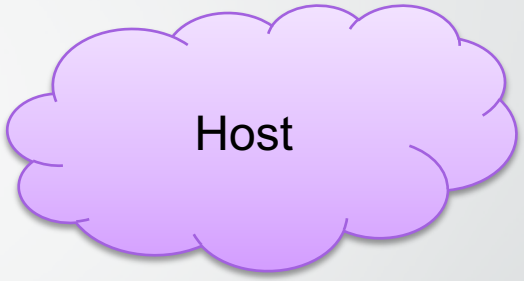
Docker Service Master

- **Service Update**
- Update Network port on service:
  - Docker service update (service_id or service_name) --publish-add 5000:80
  - Docker service ps (service_id or service_name)

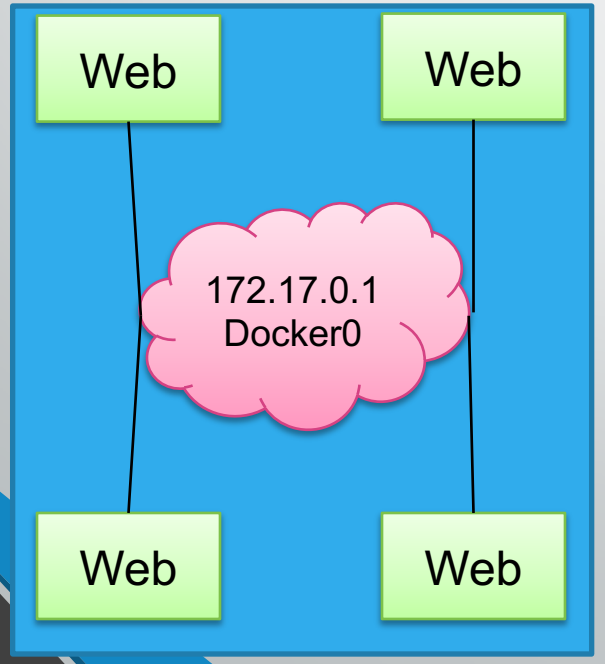
Docker Service Master



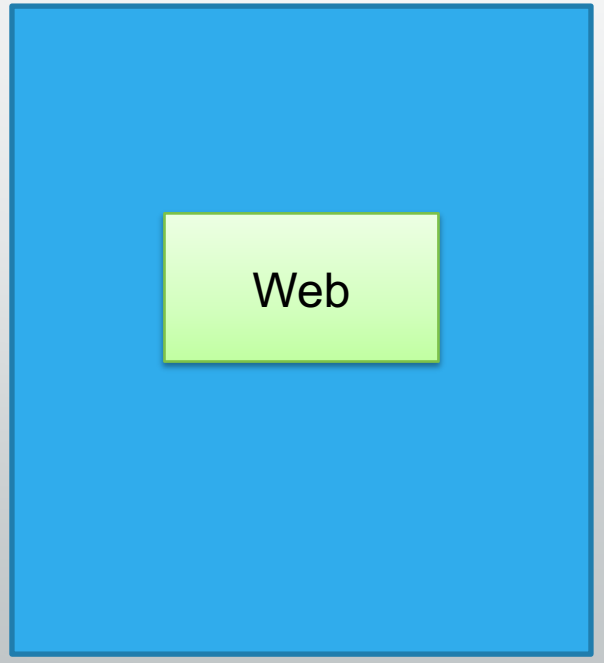




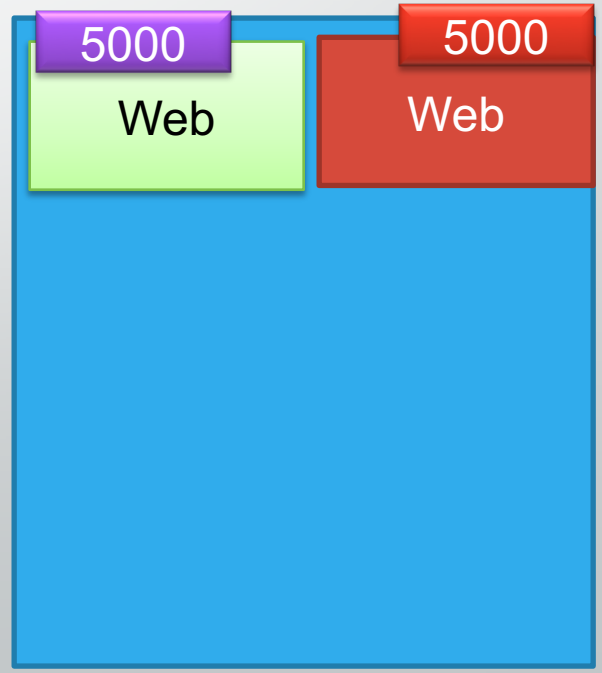
Docker run ubuntu



Docker run ubuntu –network=none



Docker run ubuntu –network=host



Docker Service Master

- **Docker Network**
- ip addr show
- $ docker network ls

- docker network inspect bridge
- Show container used this network default

Docker Service Master

- **Docker Network Create**

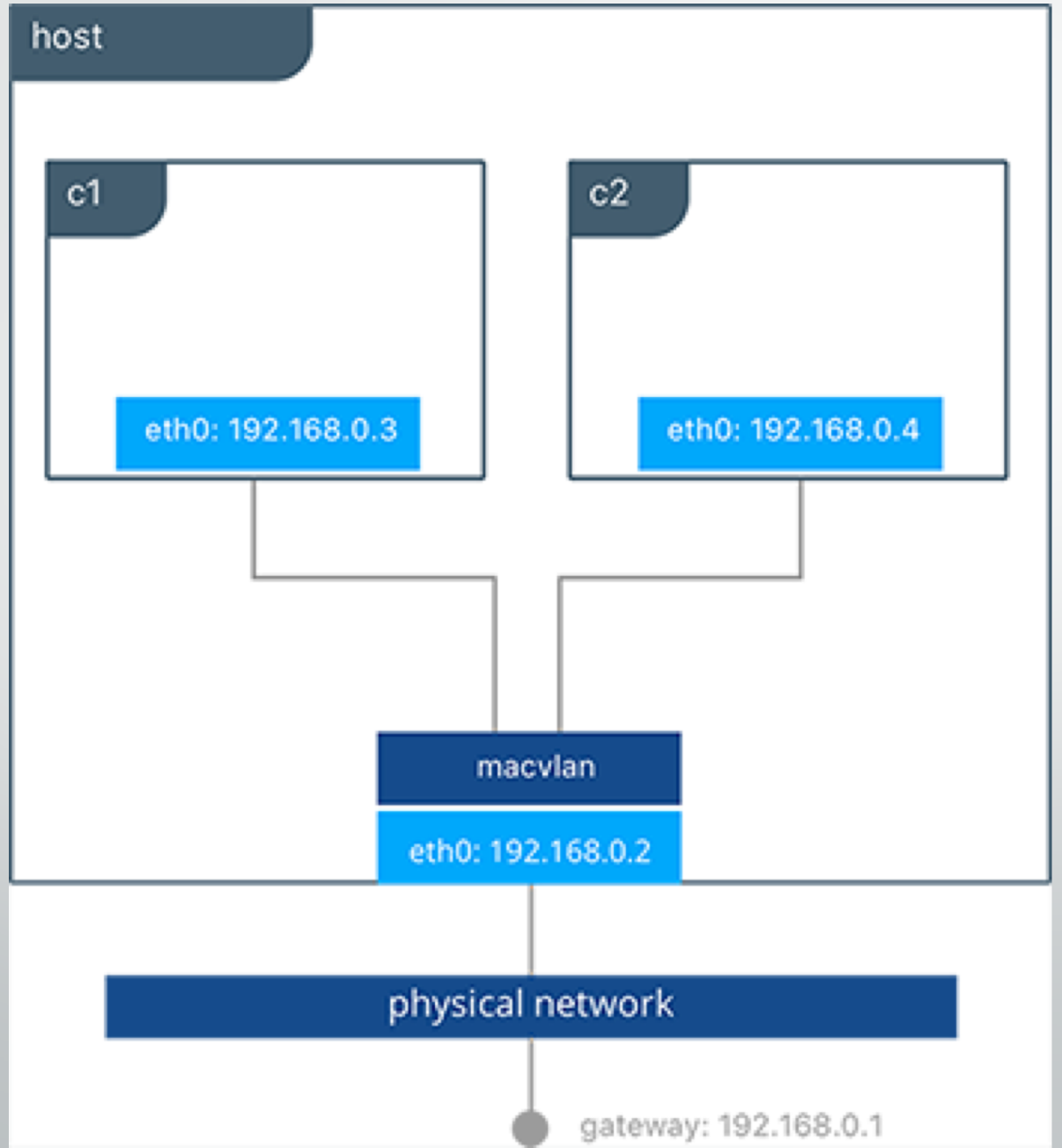
- docker network create --driver bridge isolated_nw
- $ docker network ls
- docker run --network=isolated_nw -itd --name=container3 busybox

Docker Service Master

- **Docker Network Customize**

- docker network create -d bridge --subnet 192.168.0.0/24 --gateway 192.168.0.1 -o parent=eth0 mvnet
- $ docker network ls
- docker run --network=isolated_nw -itd --name=container3 busybox

Docker Service Master

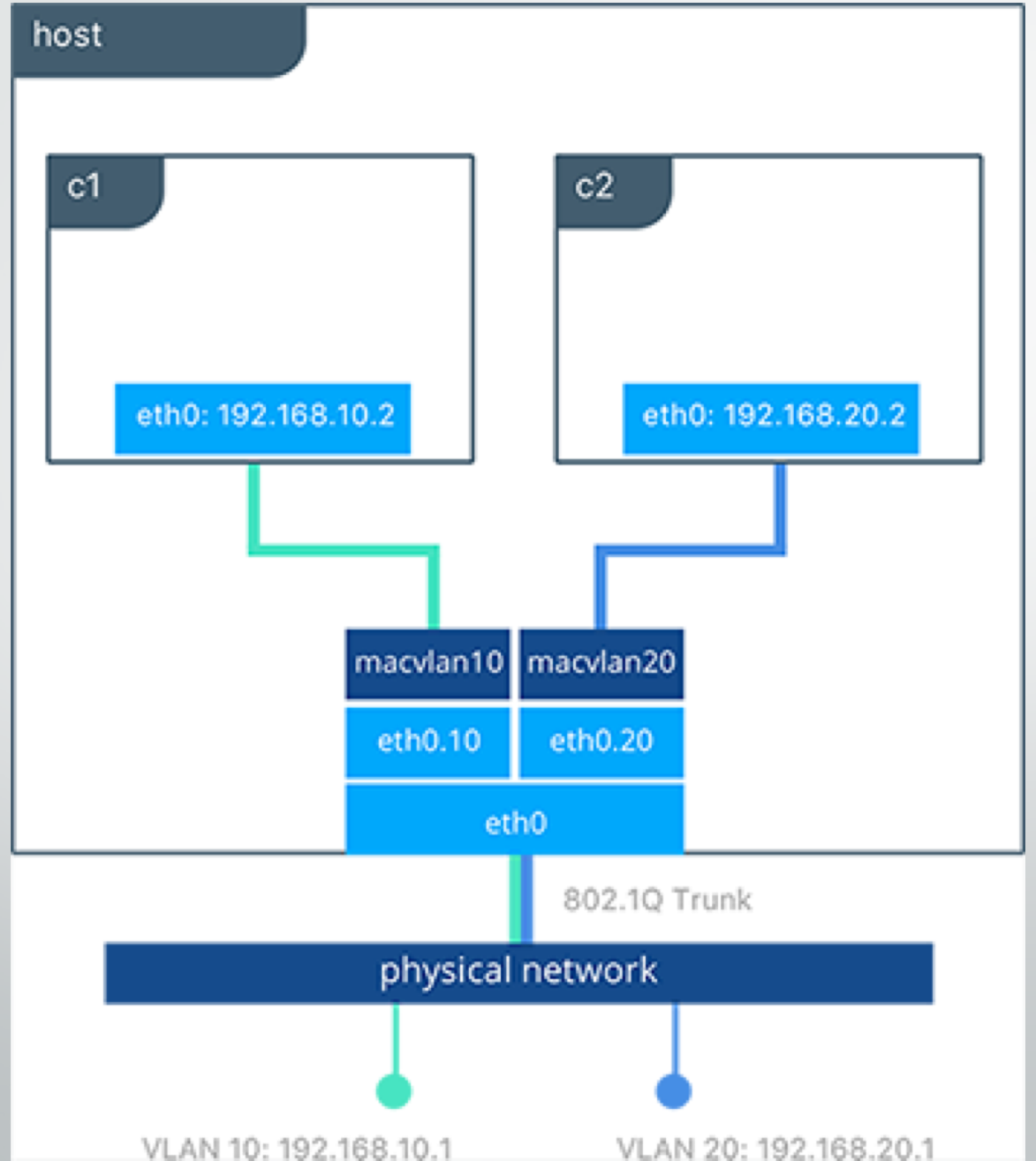


Docker Service Master

- **Docker Network MACVLAN**

- #Creation of MACVLAN network "mvnet" bound to eth0 on the host
- $ docker network create -d macvlan --subnet 192.168.0.0/24 --gateway 192.168.0.1 -o parent=eth0 mvnet
- #Creation of containers on the "mvnet" network
- $ docker run -itd --name c1 --net mvnet --ip 192.168.0.3 busybox sh
- $ docker run -it --name c2 --net mvnet --ip 192.168.0.4 busybox sh
- / # ping 192.168.0.3 PING 127.0.0.1 (127.0.0.1): 56 data bytes 64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.052 ms

Docker Service Master



Docker Service Master

- **Docker Network MACVLAN**

- #Creation of macvlan10 network in VLAN 10
- $ docker network create -d macvlan --subnet 192.168.10.0/24 --gateway 192.168.10.1 -o parent=eth0.10 macvlan10
- #Creation of macvlan20 network in VLAN 20
- $ docker network create -d macvlan --subnet 192.168.20.0/24 --gateway 192.168.20.1 -o parent=eth0.20 macvlan20
- #Creation of containers on separate MACVLAN networks
- $ docker run -itd --name c1--net macvlan10 --ip 192.168.10.2 busybox sh
- $ docker run -it --name c2--net macvlan20 --ip 192.168.20.2 busybox sh

Docker Service Master

- **Docker Network MACVLAN**

- docker network create -d macvlan \
  - --subnet=192.168.32.0/24 \
  - --ip-range=192.168.32.128/25 \
  - --gateway=192.168.32.254 \
  - --aux-address="my-router=192.168.32.129" \
  - -o parent=eth0 macnet32

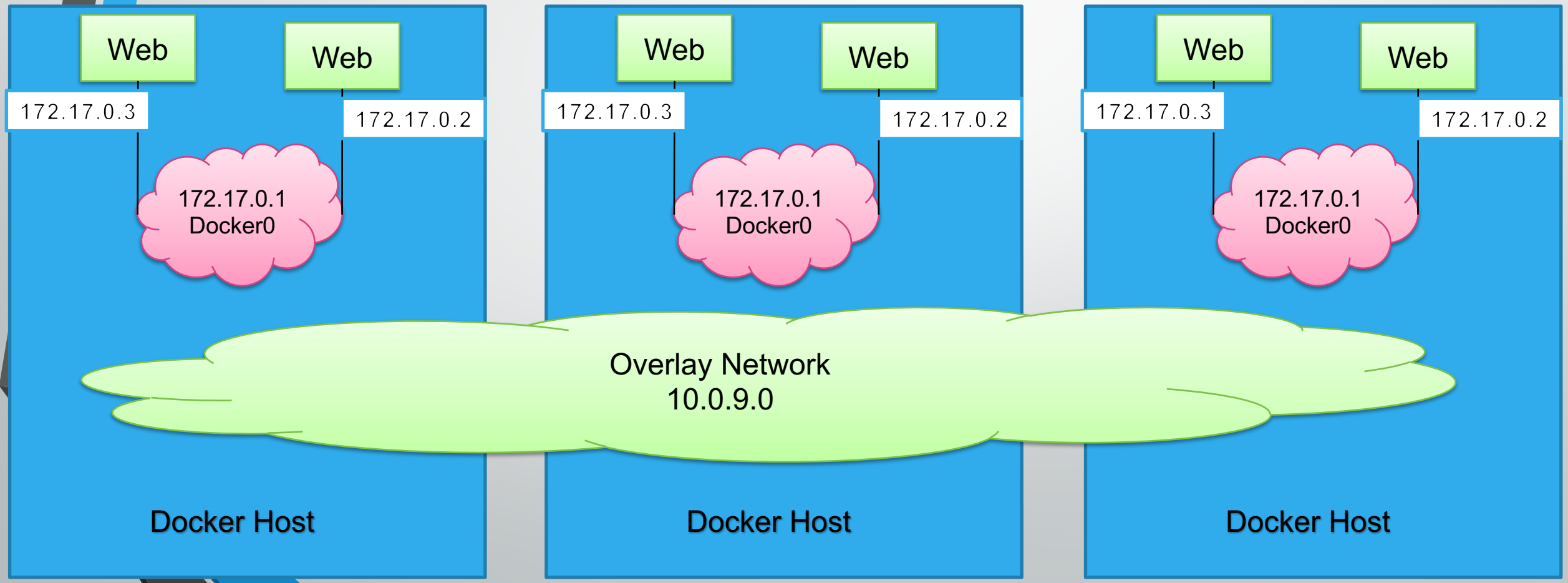
Docker Service Master

- **Docker Network with reserving**

- docker network create -d macvlan -o parent=eno1 \
  - --subnet 192.168.1.0/24 \
  - --gateway 192.168.1.1 \
  - --ip-range 192.168.1.192/27 \
  - mynet

Docker Service Master

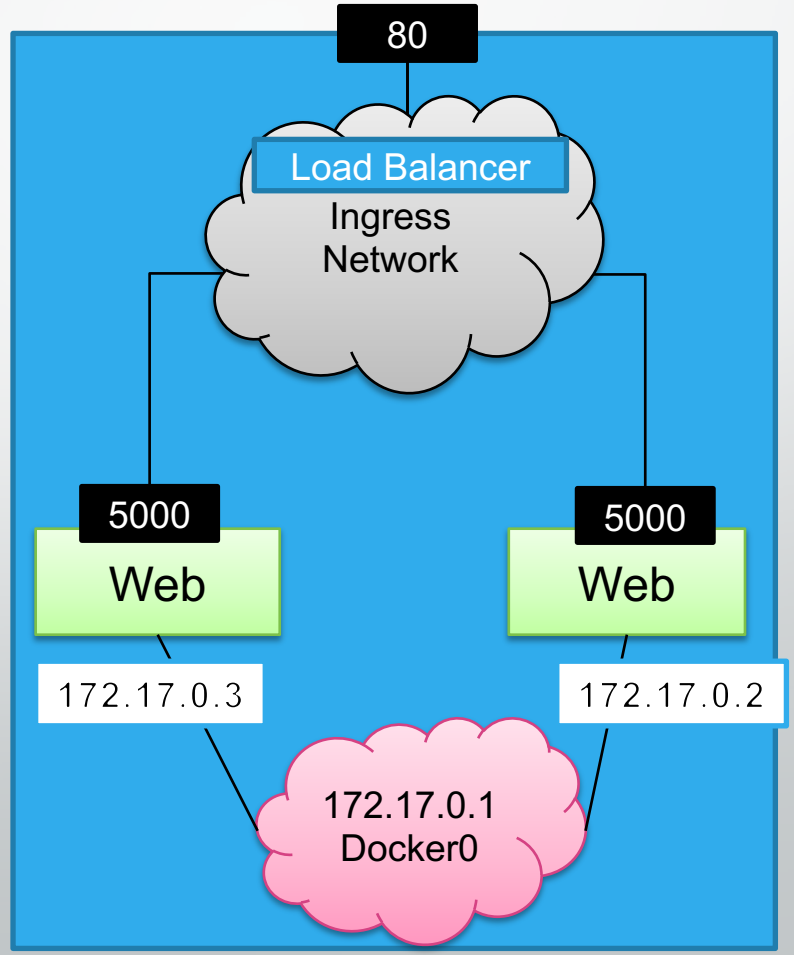
Docker network create –driver overlay –subnet 10.0.9.0/24 my-network



Docker Service Master

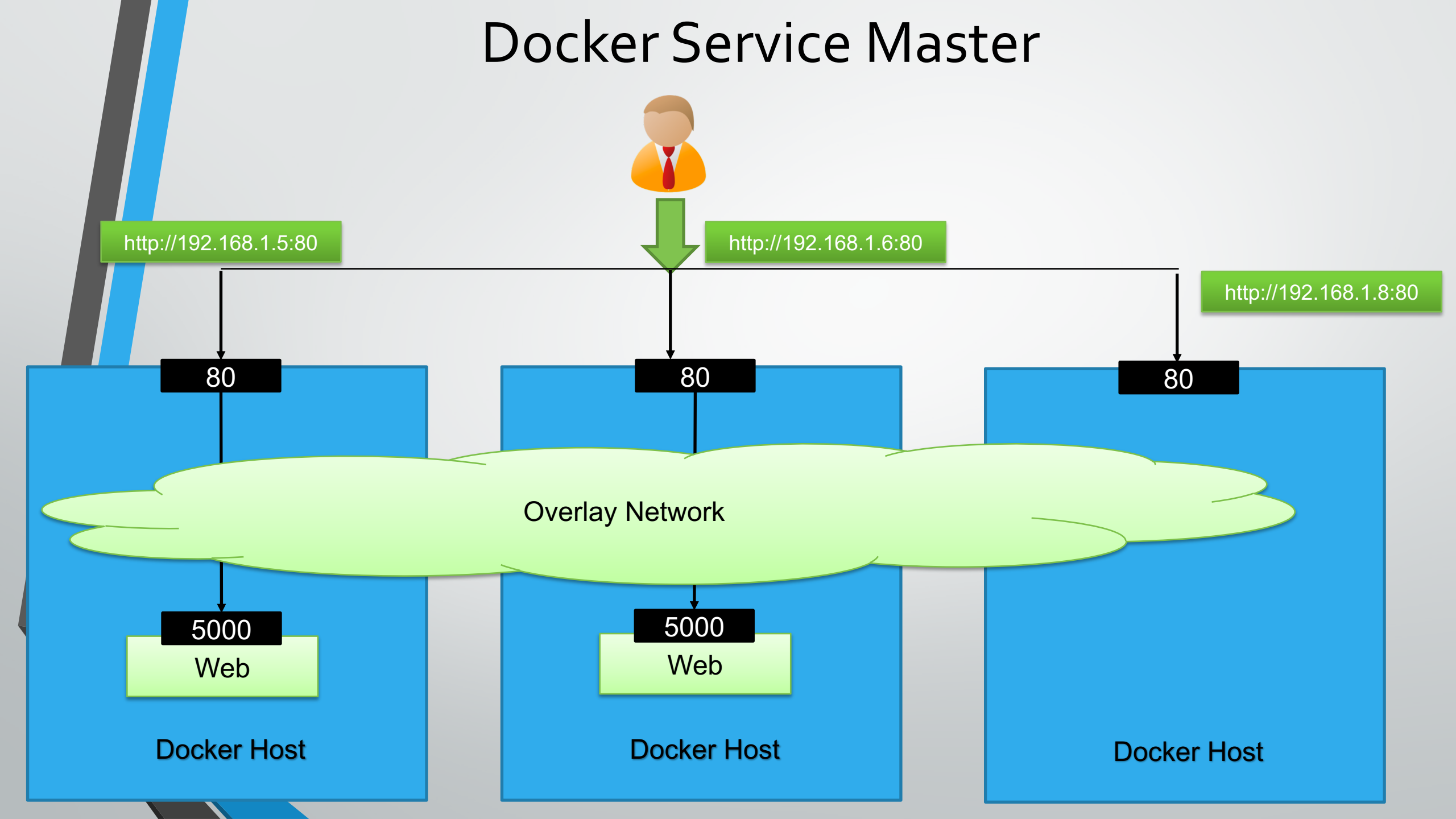
```
Docker run –p 80:5000 my-web-server
```

http://192.168.1.5:80

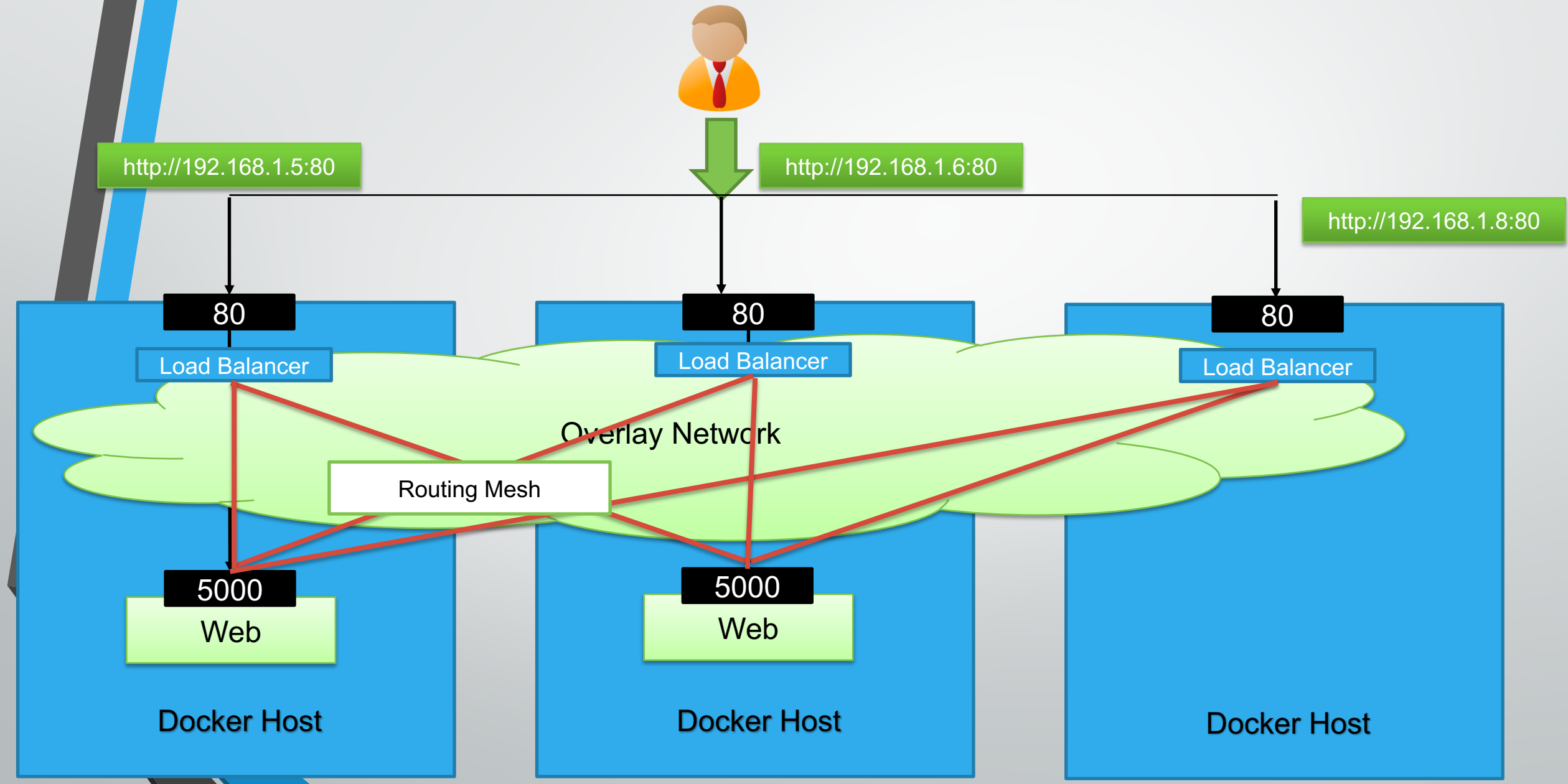



```
Docker service create \
      --replicas=2 \
      -p 80:5000 \
      My-web-server
```

Docker Service Master

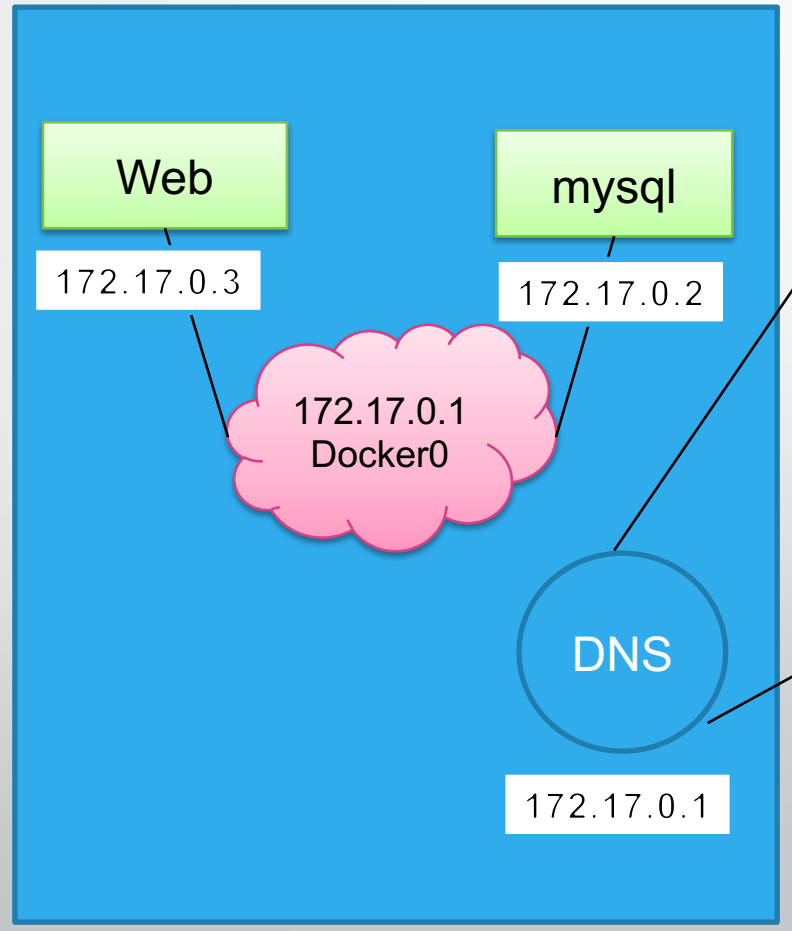


Docker Service Master



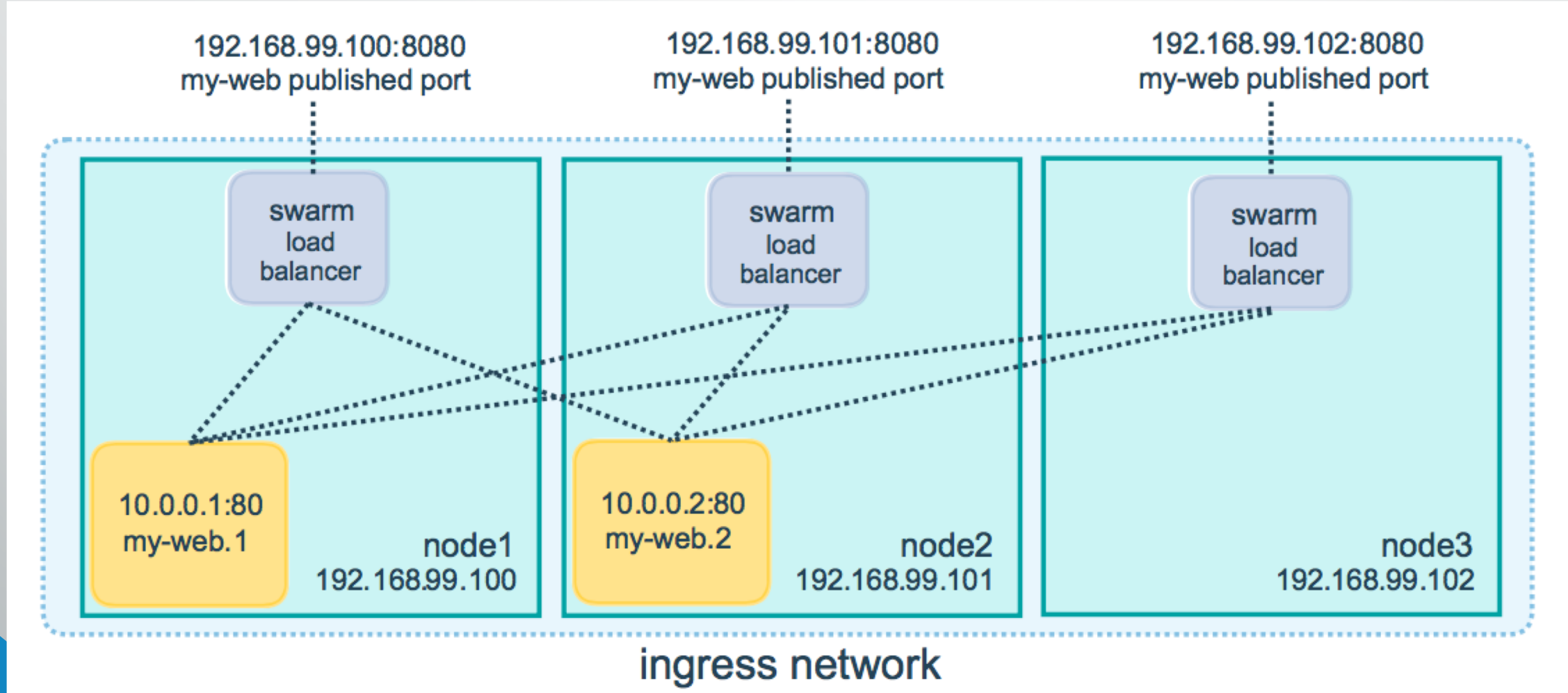
Docker Service Master

```
Mysql.connect(  mysql )
```

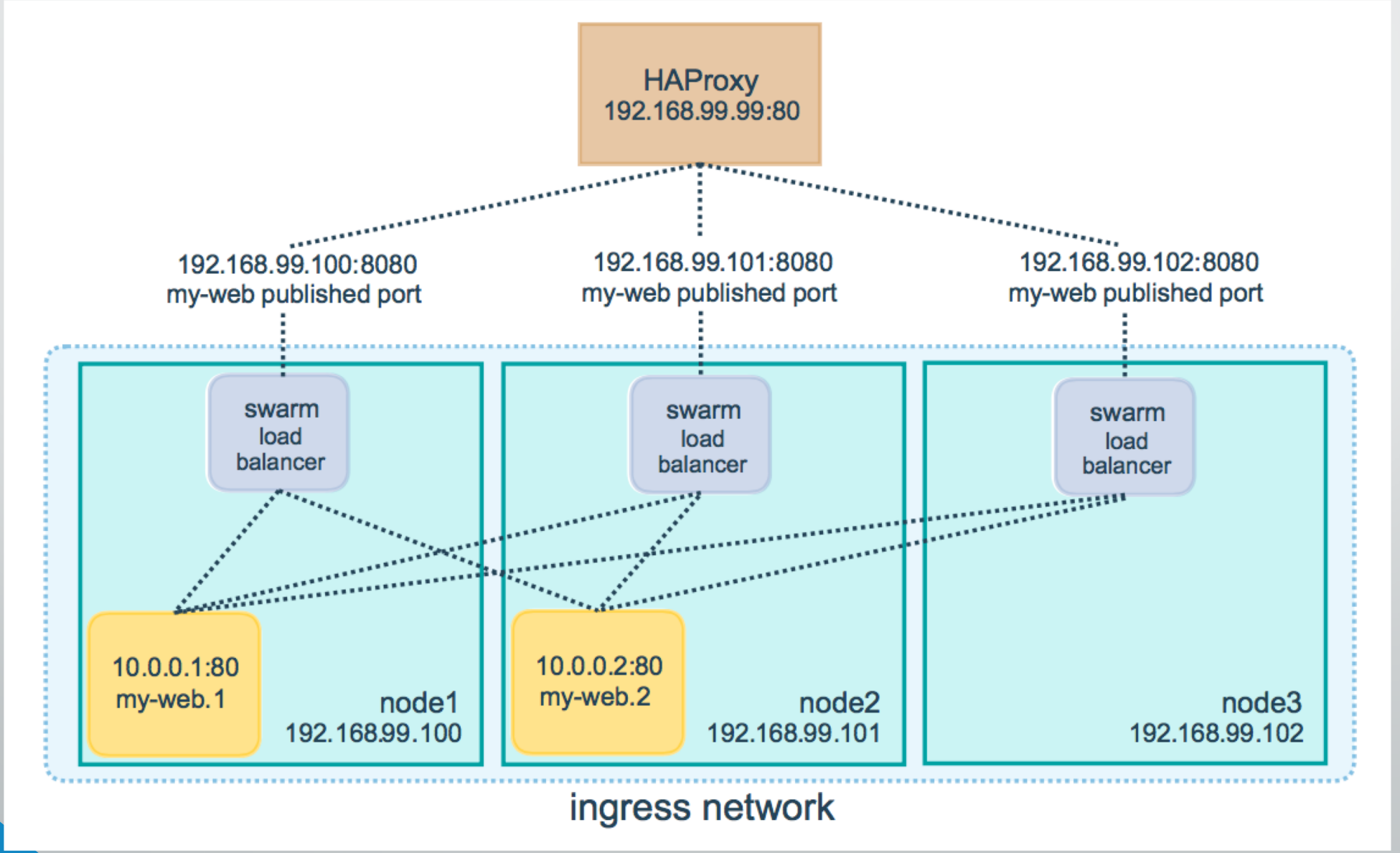


Host	IP
Web	172.17.0.2
Mysql	172.17.0.3

Docker Service Master



Docker Service Master



Docker Service Master

- **Docker Service Port Need**

Open protocols and ports between the hosts
The following ports must be available. On some systems, these ports are open by default.
TCP port 2377 for cluster management communications
TCP and **UDP port 7946** for communication among nodes
UDP port 4789 for overlay network traffic
If you plan on creating an overlay network with encryption (--opt encrypted), you also need to ensure **ip protocol 50 (ESP)** traffic is allowed.

Docker Service Master

- **Docker Network**

Open protocols and ports between the hosts
The following ports must be available. On some systems, these ports are open by default.
TCP port 2377 for cluster management communications
TCP and **UDP port 7946** for communication among nodes
UDP port 4789 for overlay network traffic
If you plan on creating an overlay network with encryption (--opt encrypted), you also need to ensure **ip protocol 50 (ESP)** traffic is allowed.

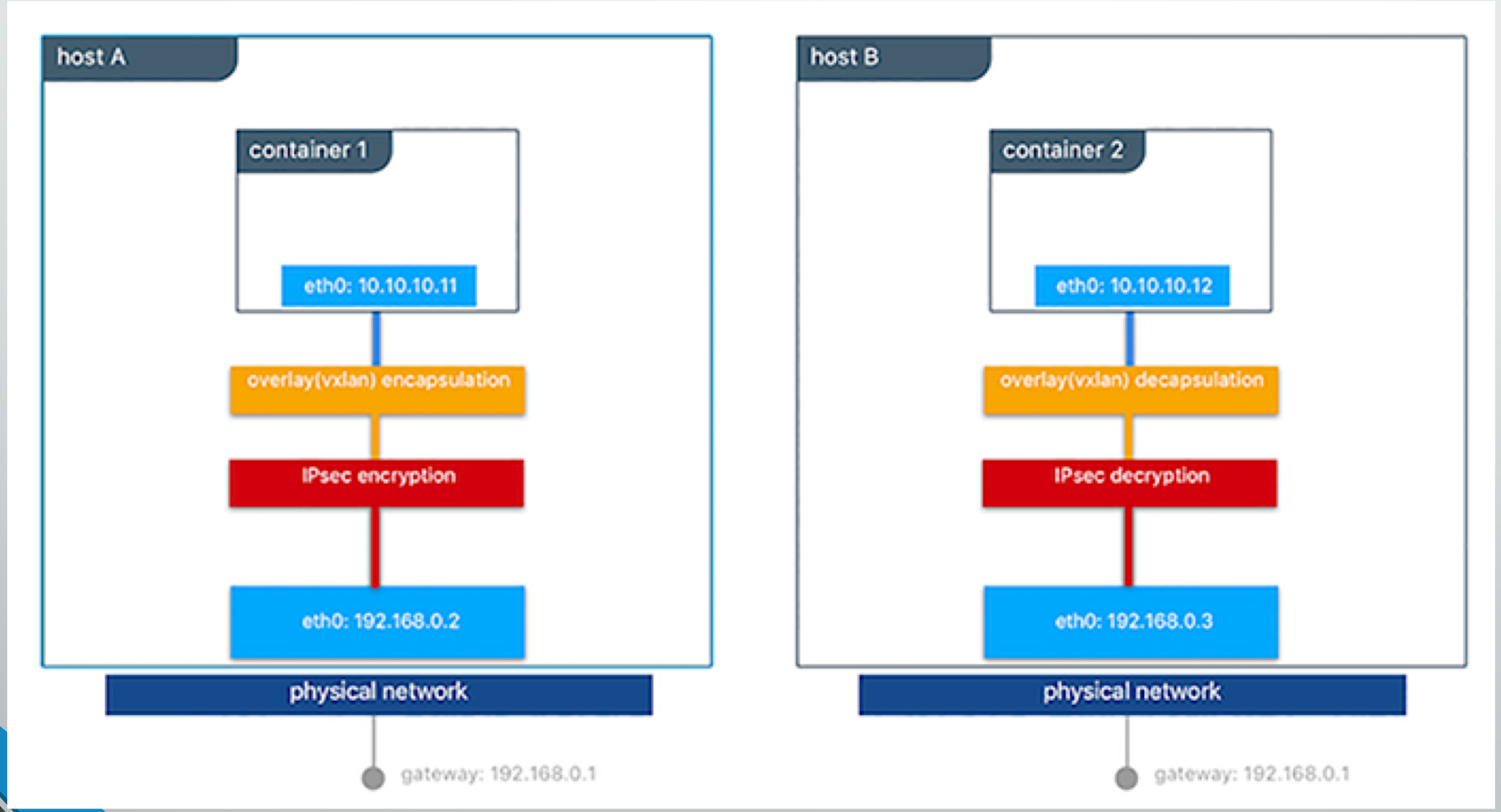
Docker Service Master

- **Docker Network Encryption**

- docker network create --opt encrypted --driver overlay my-multi-host-network

Docker Service Master

- **Docker Network Encryption**



Docker Service Master

- **Service Scale**
  - $ docker service scale **frontend**=50
  - $ docker service update --replicas=50 **frontend**
  - docker service scale backend=3 frontend=5

Docker Service Master

- **Service Maintenance Mode**
- **Step1**:
  - docker service create --replicas 3 --name redis --update-delay 10s redis:3.0.6
- **Step2**:
  - docker service ps redis
- **Step3**:
  - docker node update --availability drain node-1
- **Step** 4:
  - docker node inspect --pretty node-1
- **Step5**:
  - docker service ps redis
- **Step6**:
  - docker node update --availability active worker1

Docker Service Master

- **Service Update Image**

- $ docker service update -d --image repo/image:tag service_name

Docker Service Master

- **Service Container Placement by Hostname**

docker service create --name my-nginx --replicas=2 --constraint node.hostname==node-1 nginx

Docker Service Master

- **Service node tag**

docker node update --label-add php=true node-2

Docker Service Master

- **Service Container Placement by Label**

docker service create --name my-nginx --replicas=2 --constraint node.labels.php==true nginx

Docker Service Master

- **Docker Stack Compose**

docker stack deploy --compose-file docker-compose.yml vossibility