TechMaster
Học là có việc

Docker Swarm

Nguyễn Hàn Duy
duy@techmaster.vn

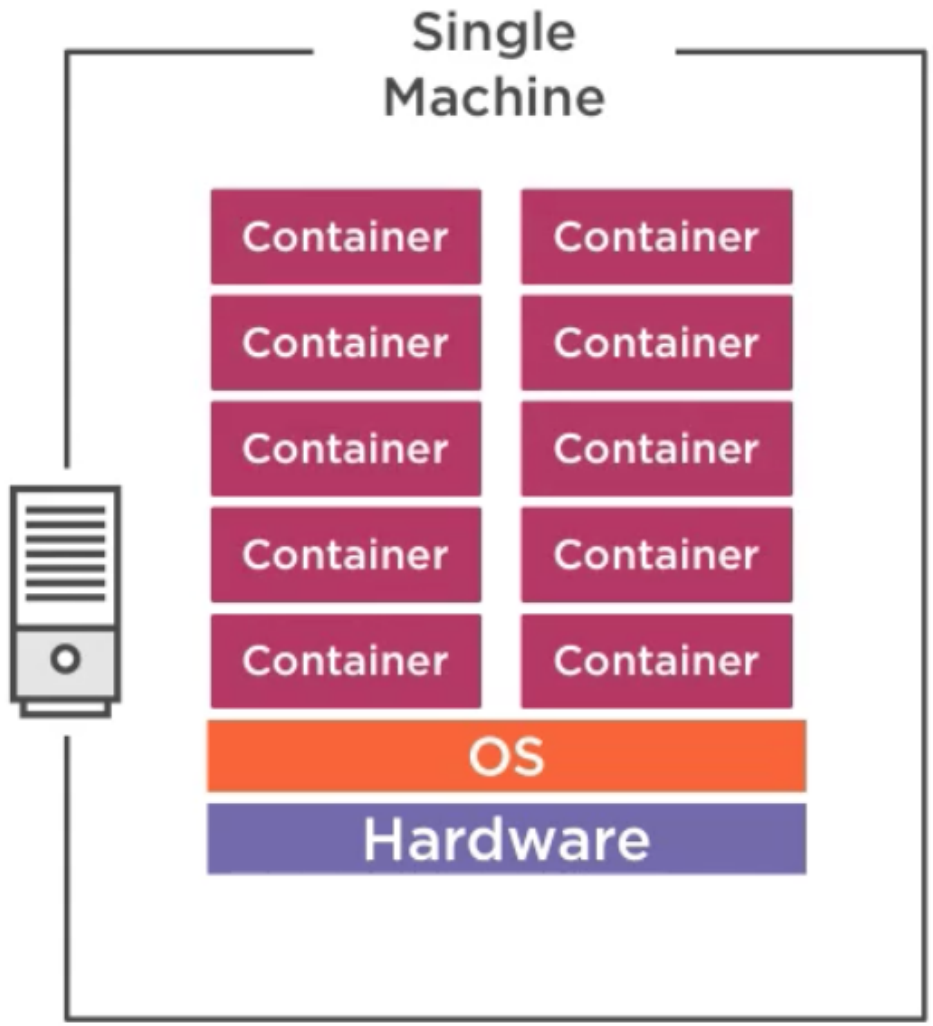
Nội dung

- Swarm cluster
- Service & tasks
- Ingress network
- Internal overlay network

Swarm cluster

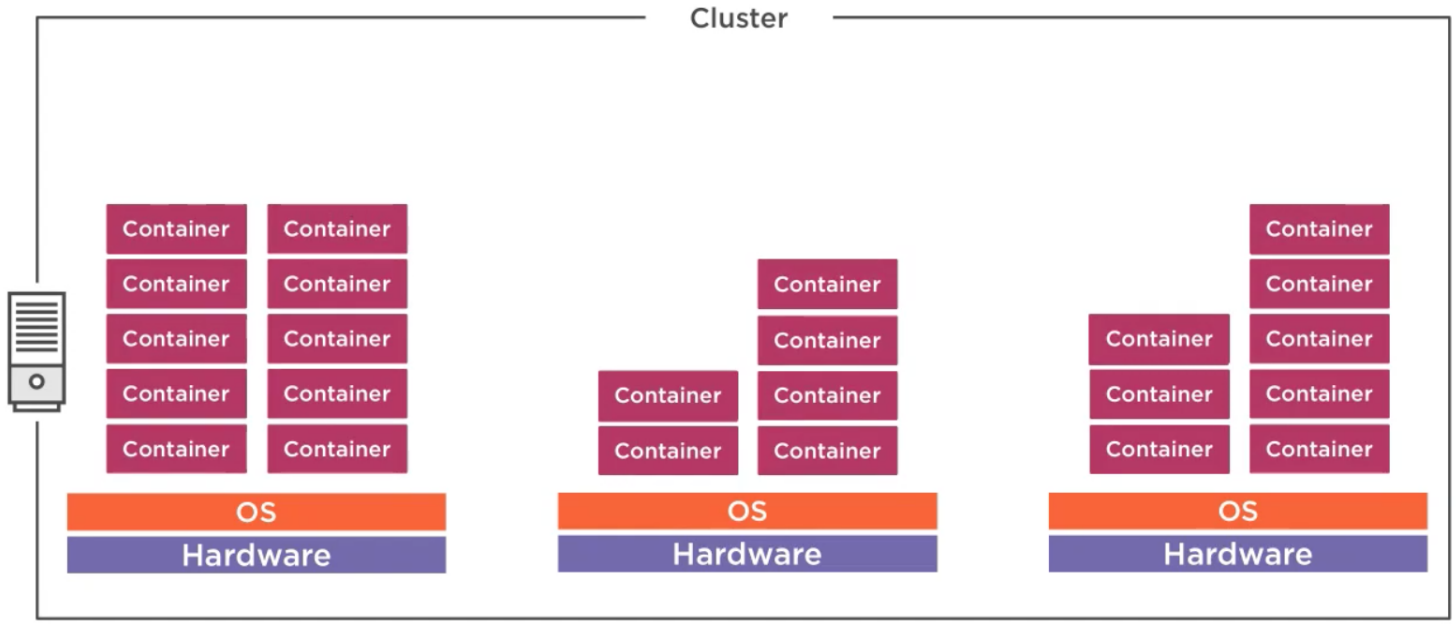
Standalone Docker host

- Nhiều container chạy trên 1 docker host
- Bị giới hạn computing resources
- Phù hợp với môi trường dev



Multiple Docker hosts

- Các container chạy trên nhiều docker host
- Các docker host được gom cụm (clustering) để trở thành 1 server khổng lồ
- Tận dụng computing resources của nhiều server
- Phù hợp trên môi trường production



Container orchestration

- Lên lịch, điều phối, quản lý trạng thái của các container
- Quản lý cụm các server

Task #1: Tạo Swarm cluster

Các bước tiến hành

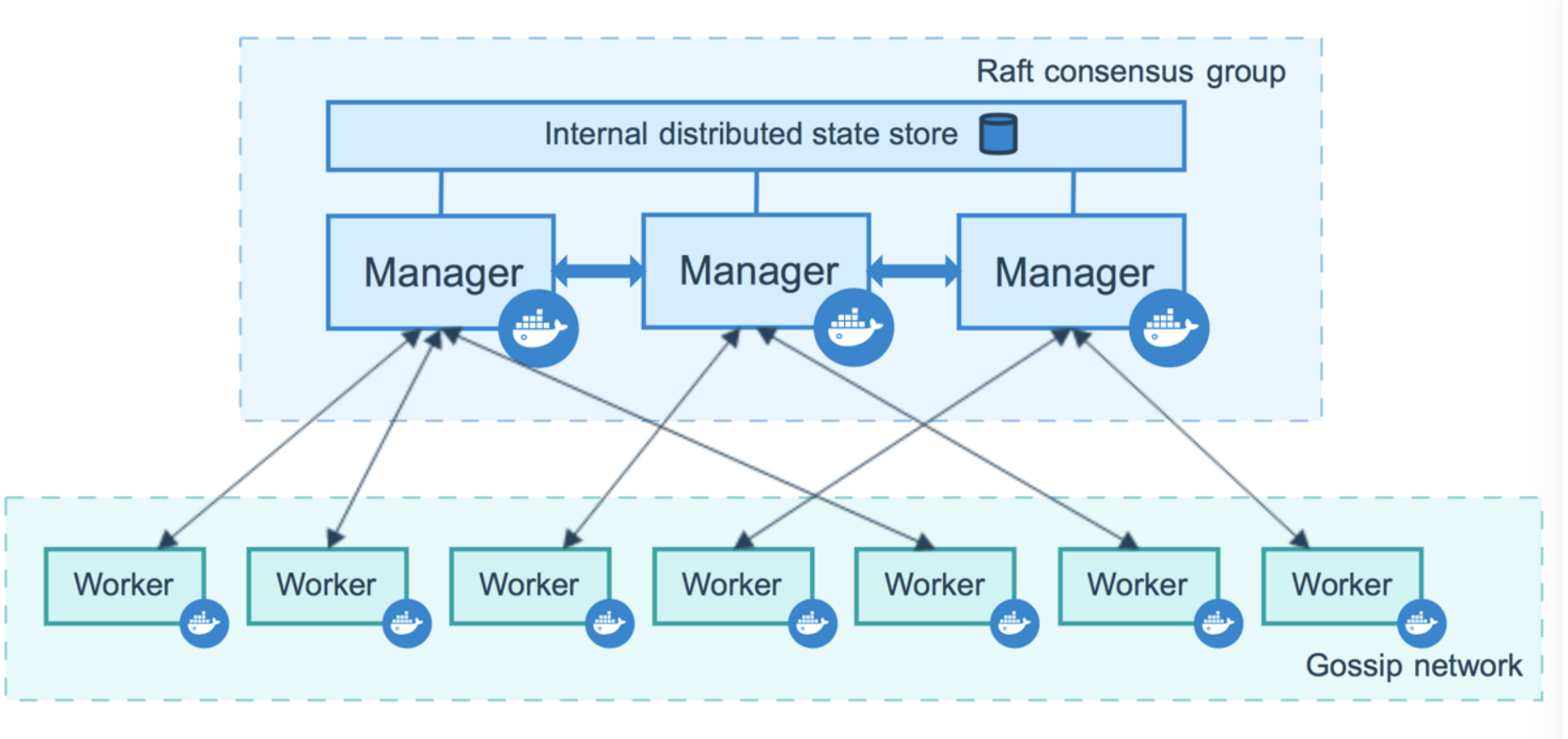
- Chuẩn bị 3 máy ảo đã cài Docker (có thể sử dụng https://labs.play-with-docker.com/)
- Trên máy 1, khởi tạo Swarm cluster bằng lệnh:

  docker swarm init --advertise-addr IP-của-máy-ảo
- Hệ thống tạo ra 1 đoạn code để thêm các worker node vào Swarm cluster. Copy paste đoạn code đó và chạy trên 2 node còn lại
- Trên cả 3 máy, chạy lệnh: **docker node ls**

Swarm nodes

- Mỗi node trong Swarm cluster ~ 1 docker host
- Có 2 loại node: manager và worker
- Manager:
  - Quản lý trạng thái của cluster
  - Điều phối các container
  - Chạy các container
- Worker:
  - Chạy các container do manager chỉ định

Manager & worker nodes



Service & tasks

Standalone container

- App được triển khai theo từng container
- Mỗi app là 1 contaner chạy trên 1 node

- *docker run* … image

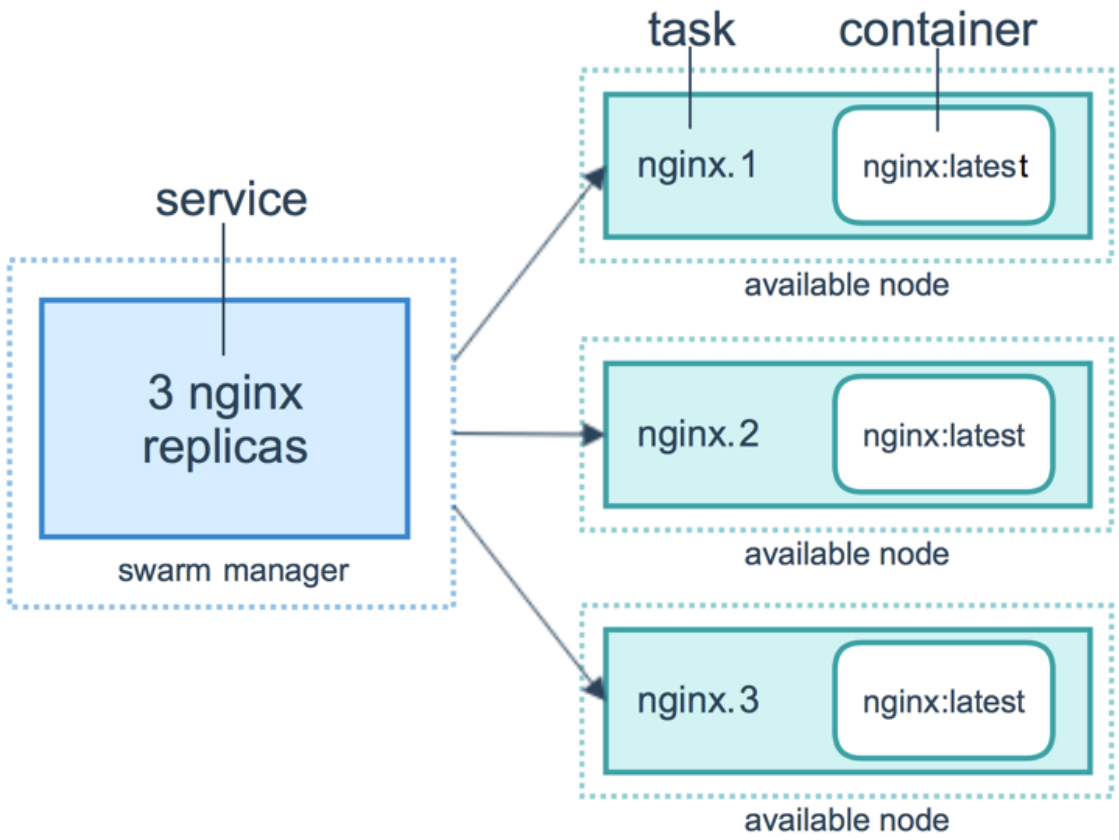
Swarm service

- App được triển khai dưới dạng service
- Mỗi service gồm 1 hoặc nhiều task chạy trên cluster
- Mỗi task khởi tạo 1 container
- ***docker service create*** … image

Standalone container

Nginx Container

Swarm service



Task #2: Triển khai 1 service chạy image Nginx trên Swarm cluster

Các bước tiến hành

Trên node *manager*:

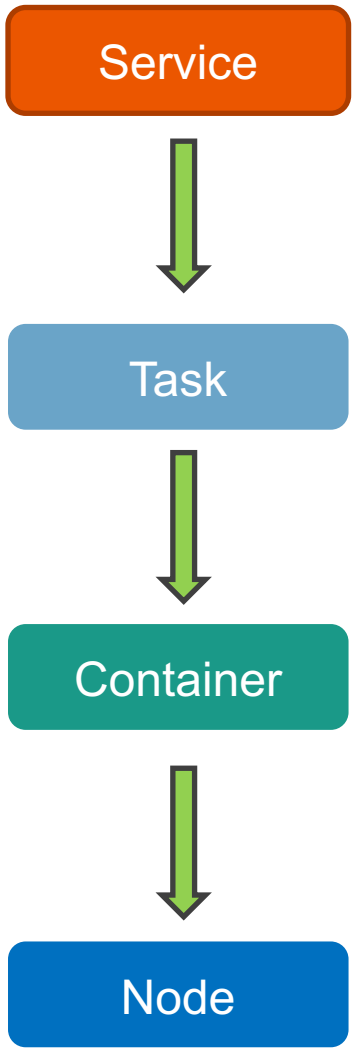
- Sử dụng **docker service create** với image *nginx:alpine*, expose ra cổng *8080*, đặt tên service là *nginx-app*
- Liệt kê danh sách service: **docker service ls**
- Liệt kê các task có trong service nginx-app: **docker service ps** nginx-app

Trên tất cả các node:

- Kiểm tra port 8080 có được open:

sudo netstat -tulpn | grep LISTEN | grep 8080

Service mô tả desired state của app



Desired state của service nginx-app:

- Sử dụng image **nginx:latest**
- Expose ra cổng **8080** của cluster
- Số lượng task: 1
- Chạy trên 1 node bất kỳ

Service mô tả trạng thái chúng ta mong muốn (***What***), còn làm thế nào (***How***) để duy trì trạng thái đó là nhiệm vụ của Swarm managers

Task #3: Cập nhật desired state cho app nginx

Yêu cầu

Cập nhật desired state mới cho service nginx-app:

- Số lượng task: 6 (**--replicas**)
- Không publish ra cổng 8080 của cluster nữa (**--publish-rm 8080:80**)
- Thay vào đó publish ra cổng 8081 (**--publish-add 8081:80**)
- Kiểm tra trạng thái service nginx-app

Sử dụng lệnh: ***docker service update [OPTIONS]*** tên-service

Tham khảo các option:

https://docs.docker.com/engine/reference/commandline/service_update/

Task #4: Gỡ 1 node ra khỏi cluster

Yêu cầu

- Trên 1 worker node có các task đang chạy, dùng lệnh sau để gỡ node ra khỏi cluster: **docker swarm leave**
- Kiểm tra trạng thái của service nginx-app

Task #5: Thêm node vào cluster

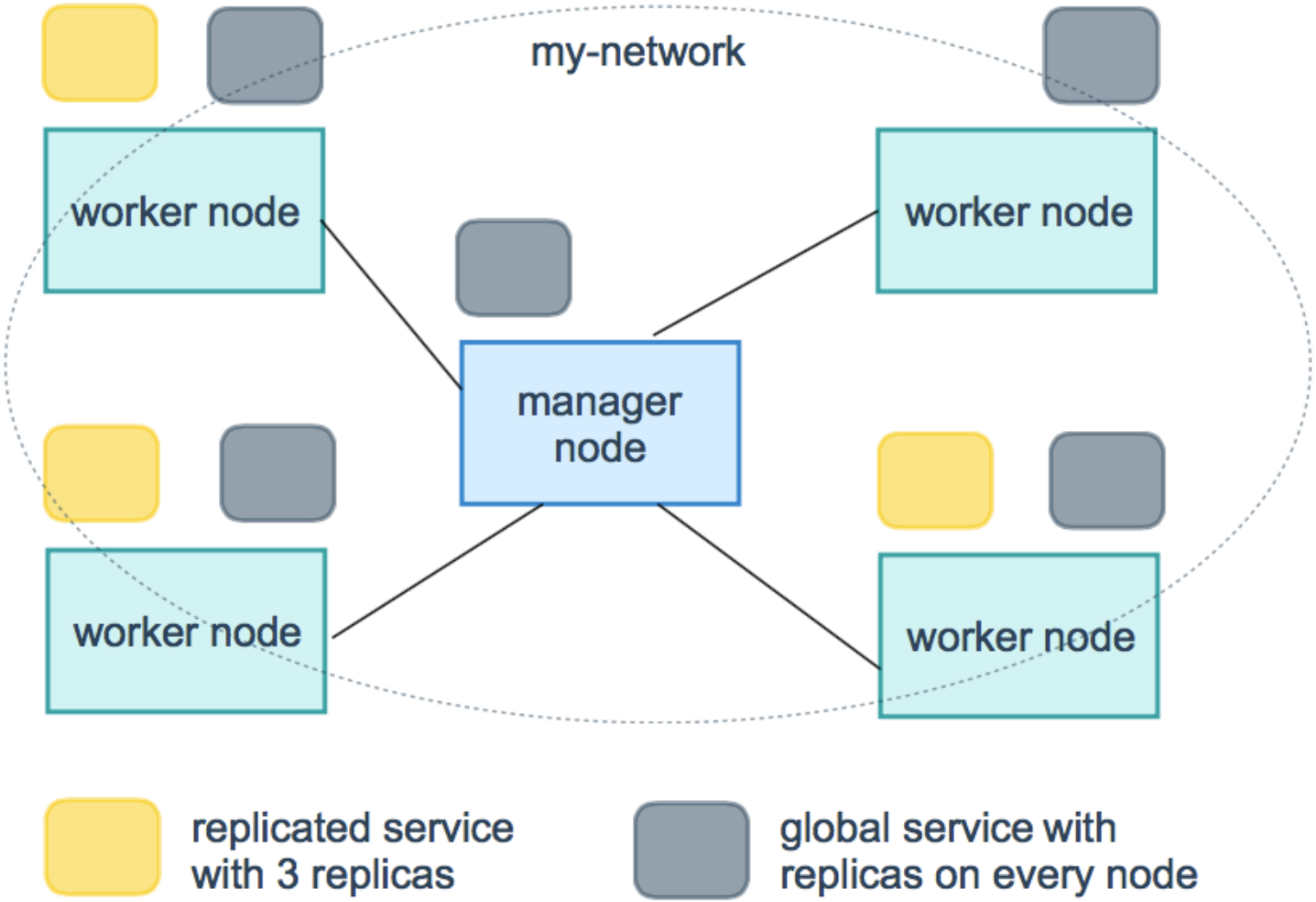
Yêu cầu

- Tạo thêm Docker host instance
- Trên *node manager*, chạy lệnh sau để lấy token join vào Swarm:

docker swarm join-token worker

- Copy lệnh mà node manager tạo ra để chạy trên instance mới
- Kiểm tra trạng thái service nginx-app

Service: global vs replicated



Task #6: Tạo global service monitoring các node

Tạo global service monitoring các node

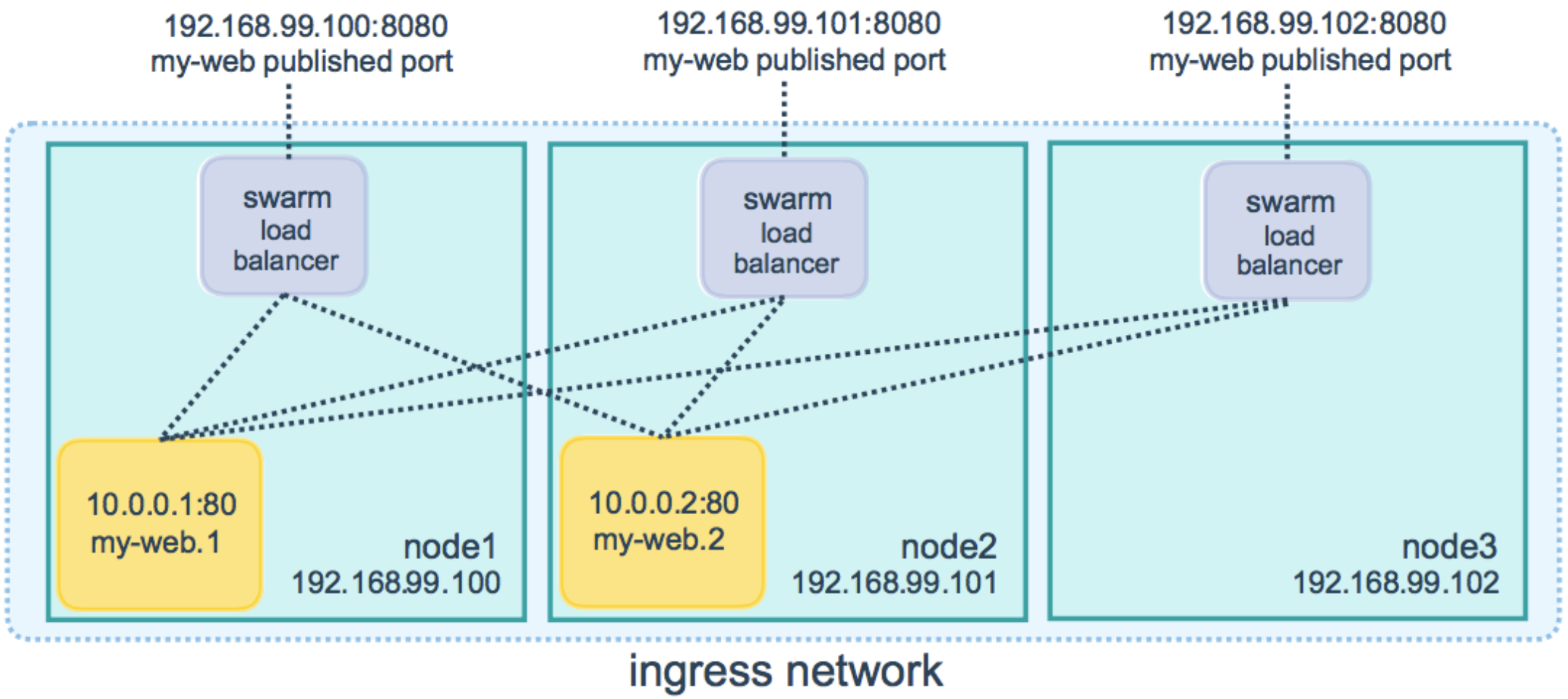
- Tạo global service cho image google/cadvisor:latest
- Kiểm tra trên các node
- Sau khi tạo global service, add thêm 1 worker node vào cluster

Tạo service có tên monitor, mode global, expose ra cổng 9005

```
docker service create \
--mount type=bind,source=/,destination=/rootfs:ro \
--mount type=bind,source=/var/run,destination=/var/run:rw \
--mount type=bind,source=/var/run/docker.sock,destination=/var/run/docker.sock \
--mount type=bind,source=/sys,destination=/sys:ro \
--mount type=bind,source=/var/lib/docker/,destination=/var/lib/docker:ro \
--mode global \
--name monitor \
-p 9005:8080 \
google/cadvisor:latest
```

Ingress network

Ingress network



Task #7: Triển khai service jwilder/whoami

Các bước tiến hành

Trên node manager:

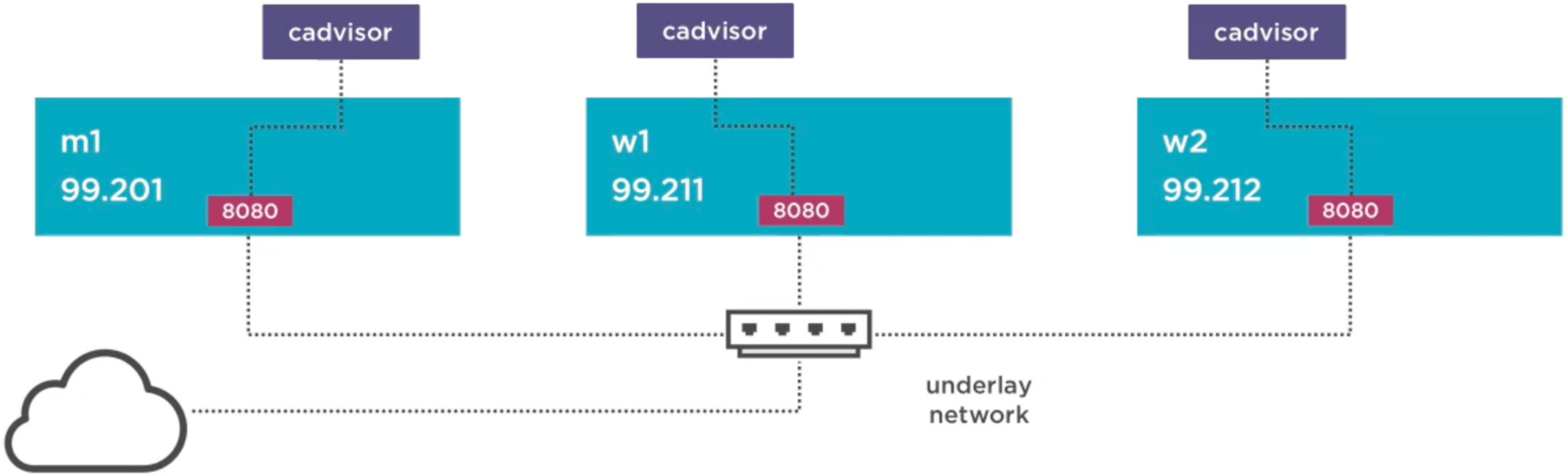
- Tạo service từ image **jwilder/whoami** gồm 2 tasks
- Expose cổng 8000 của image ra cổng 8085 của cluster
- Đặt tên service là whoami-app
- Chỉ chạy các tasks trên các node worker

Gọi vào cổng 8085 từ một node bất kỳ trong Swarm

Tham khảo cách đặt constraint cho service:

https://docs.docker.com/engine/reference/commandline/service_create/#specify-service-constraints---constraint

Host mode published port



Task #8: Host-mode published port cho global service

Các bước tiến hành

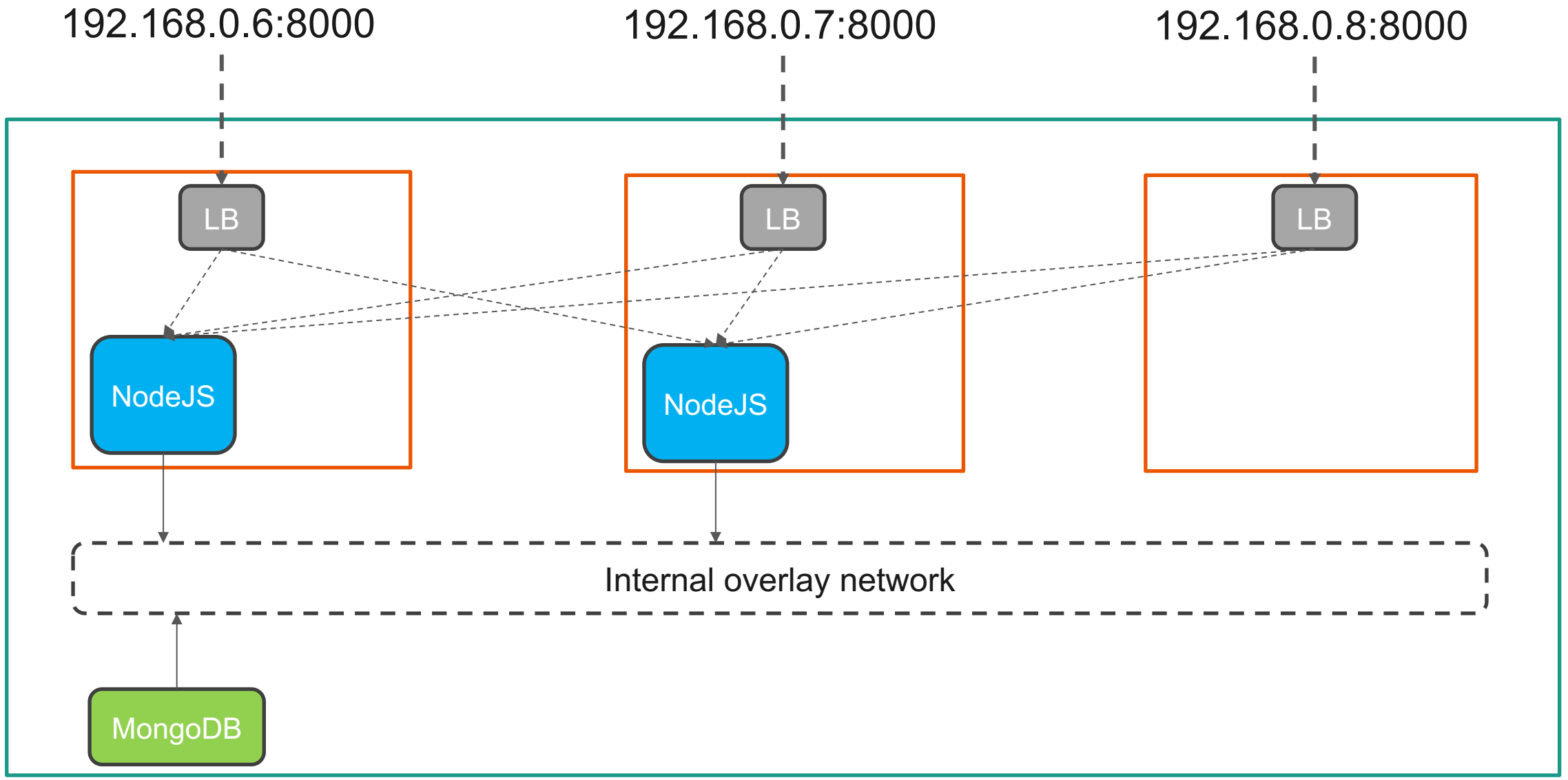
- Remove published port:

  docker service update --publish-rm 9005:8080 monitor
- Kiểm tra trạng thái service: **docker service ps monitor**
- Add host-mode published port:

  docker service update --publish-add mode=host,published=9006,target=8080 monitor
- Kiểm tra lại trạng thái service monitor
- Truy cập cổng 9006 từ các node trong cluster

Internal overlay network

Internal overlay network



Task #9: Triển khai ứng dụng NodeJS + MongoDB

Các bước tiến hành

- Viết Dockerfile cho source code NodeJS: https://github.com/handuy/nodejs-mongodb
- Build thành docker image có tên <docker-hub-repo>/demo-service
- Push image lên Docker Hub

Các bước tiến hành

Tạo 1 internal overlay network: **docker network create -d overlay myapp**

Với service MongoDB:

- Sử dụng image **mongo:latest**
- Tên service là **mongodb**
- Mount thư mục **/data/db** ra 1 volume có tên dbdata
- Chạy ở 1 node cố định
- Gắn vào network myapp

Với service NodeJS:

- Chạy sau service mongodb
- Scale thành 3 tasks
- Sử dụng image demo-service vừa build ở slide trước
- Expose cổng 3000 của ứng dụng ra cổng 8000 của cluster
- Định nghĩa 2 biến môi trường: MONGODB_URI=mongodb://mongodb:27017/demo, PORT=3000
- Gắn vào network myapp