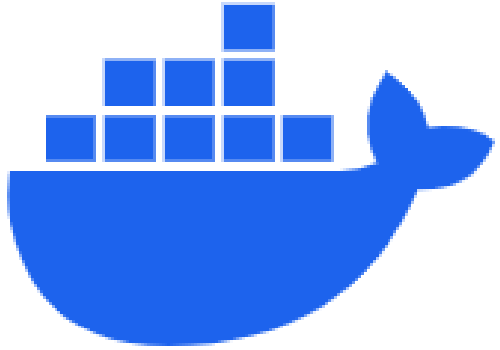
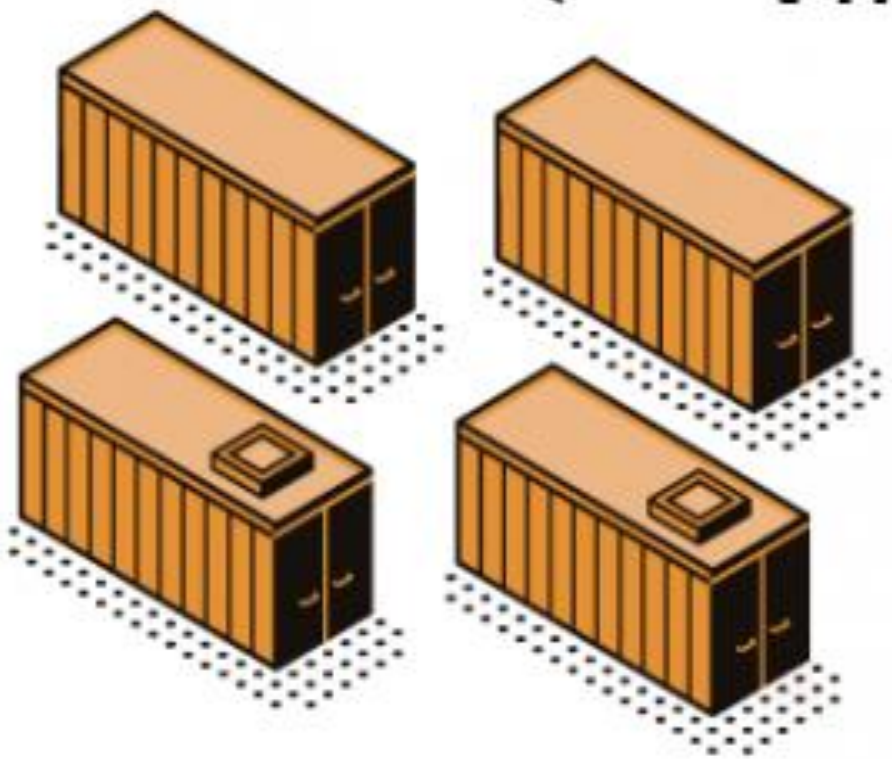

docker

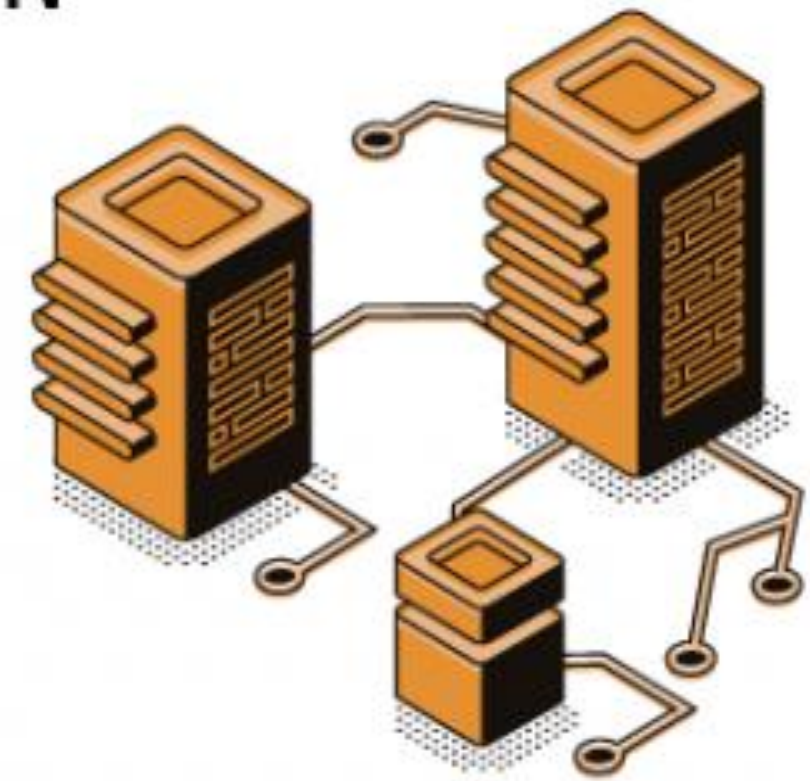
Nội Dung Thuyết Trình

- Giới thiệu về Containerization
- Container Runtime – Docker container runtime
- Docker Architecture – Components overview
- Details of main components
- Automation deployed with Docker-Compose



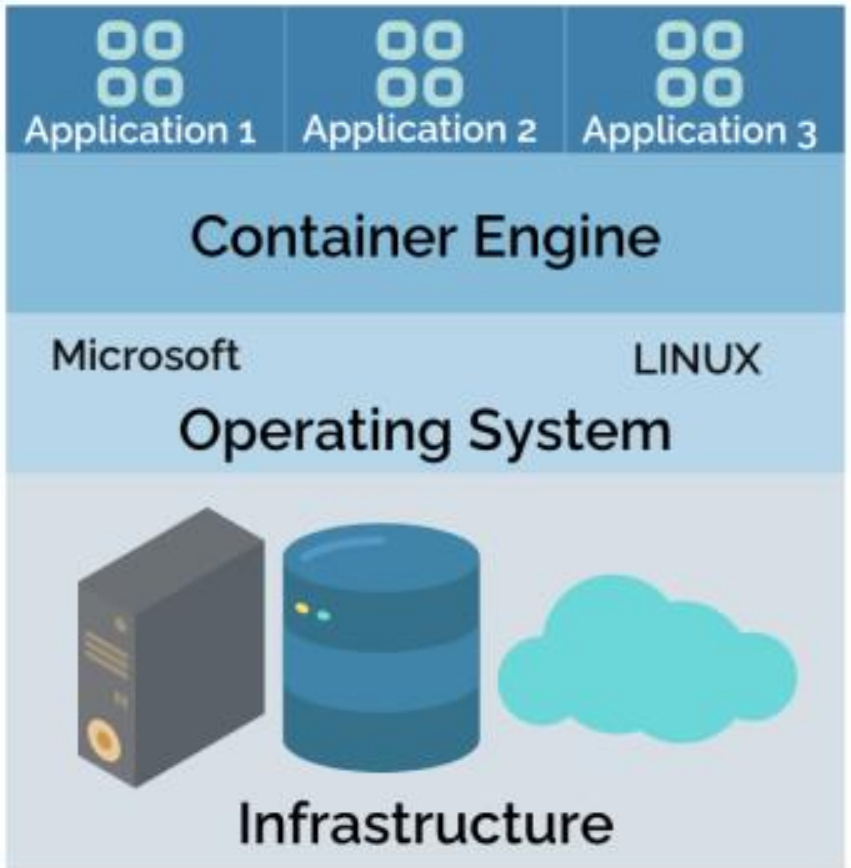
CONTAINERIZATION VS VIRTUALIZATION



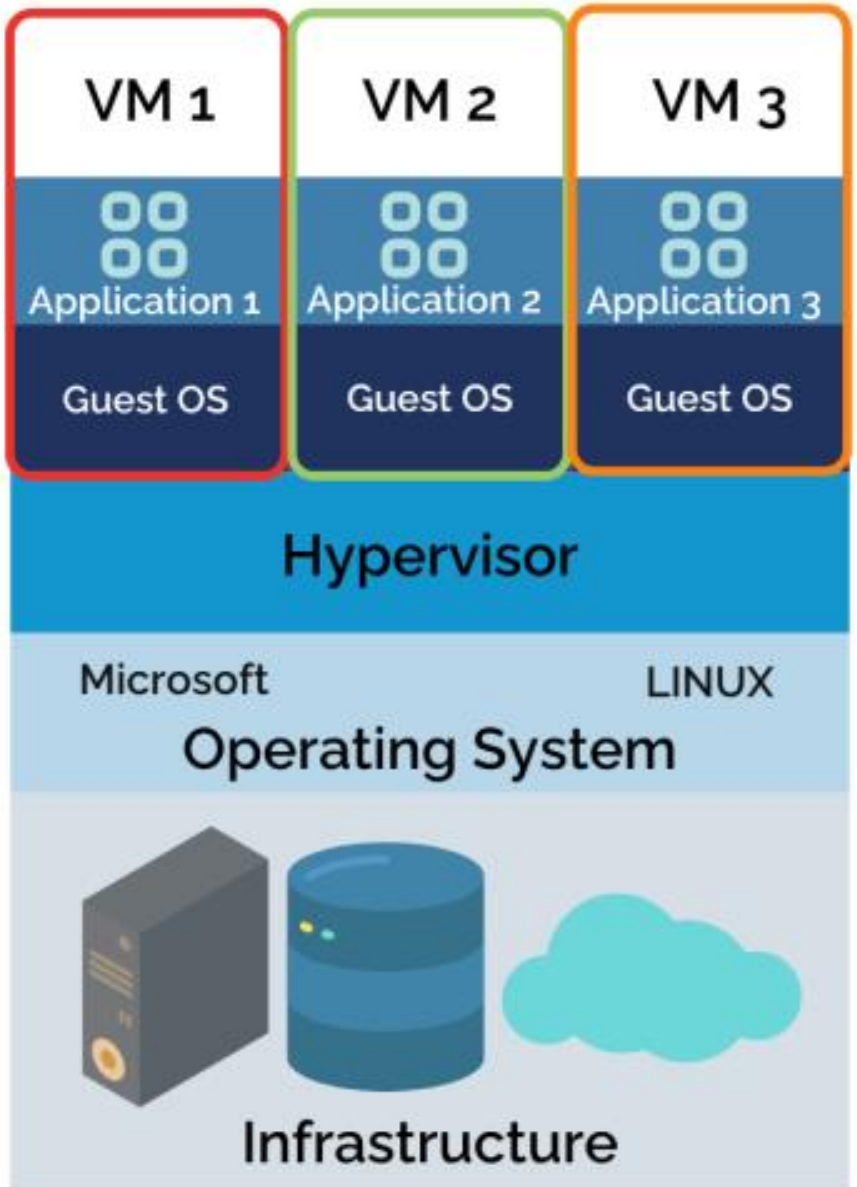


So sánh ảo hóa và container hóa

	Containerization	Virtualization
Khái niệm	Đóng gói ứng dụng và môi trường chạy của nó thành một container. Dùng chung OS với host.	Tạo ra các máy ảo độc lập với phần cứng thực bằng cách mô phỏng phần cứng máy tính.
Hiệu suất	Hiệu suất tốt hơn vì nó chia sẻ kernel với hệ thống chủ và không cần lớp giả mạo phần cứng.	Cần một lớp giả mạo phần cứng, do đó có thể có chi phí hiệu suất khiến môi trường ảo hóa chậm hơn so với chạy trực tiếp trên phần cứng.
Khởi tạo và triển khai	Nhanh chóng và yêu cầu ít tài nguyên hơn, vì chúng không cần tạo ra một hệ điều hành ảo đầy đủ.	Khởi tạo và triển khai máy ảo yêu cầu thời gian và tài nguyên lớn
Đóng gói	Mỗi container chứa một ứng dụng cụ thể và tất cả những gì cần thiết để chạy nó.	Máy ảo là một hệ điều hành đầy đủ và có thể chứa nhiều ứng dụng.
Kích thước và tài nguyên	Nhẹ nhàng vì chúng dùng chung kernel và một số thành phần của OS gốc	Tốn nhiều tài nguyên hơn vì mỗi máy ảo có kernel và hệ điều hành của riêng mình
Quản lý tài nguyên	Quản lý tài nguyên trên cấp độ container.	Quản lý tài nguyên trên cấp độ máy ảo.
Quản lý hệ thống	Đơn giản hóa quá trình triển khai và cập nhật ứng dụng, dễ quản lý hơn.	Đòi hỏi quản lý hệ thống và cập nhật giống như trên máy chủ vật lý.
Khả năng cô lập	Sử dụng chung kernel và một số thành phần, chỉ tách biệt về mặt ứng dụng, service.	Cô lập tuyệt đối giữa các ứng dụng và môi trường, sử dụng các phần cứng giả mạo riêng biệt và OS riêng



CONTAINERIZATION



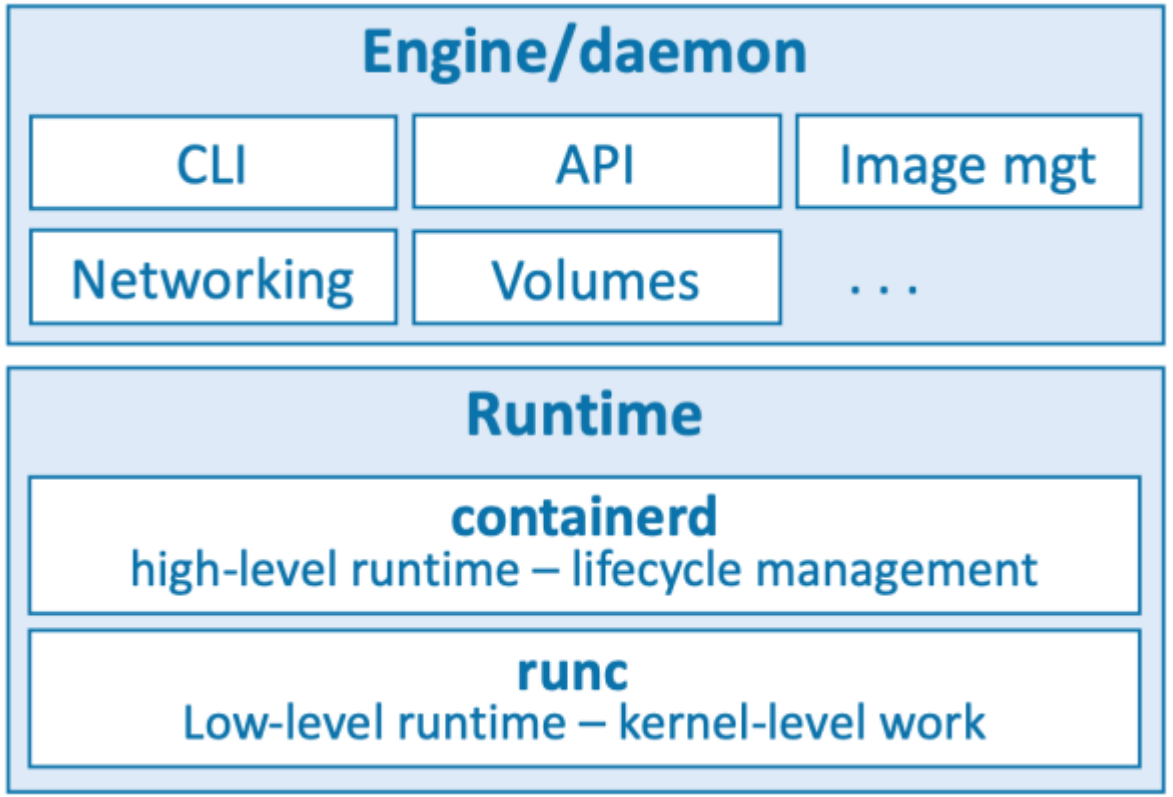
VIRTUALIZATION

Container Runtime

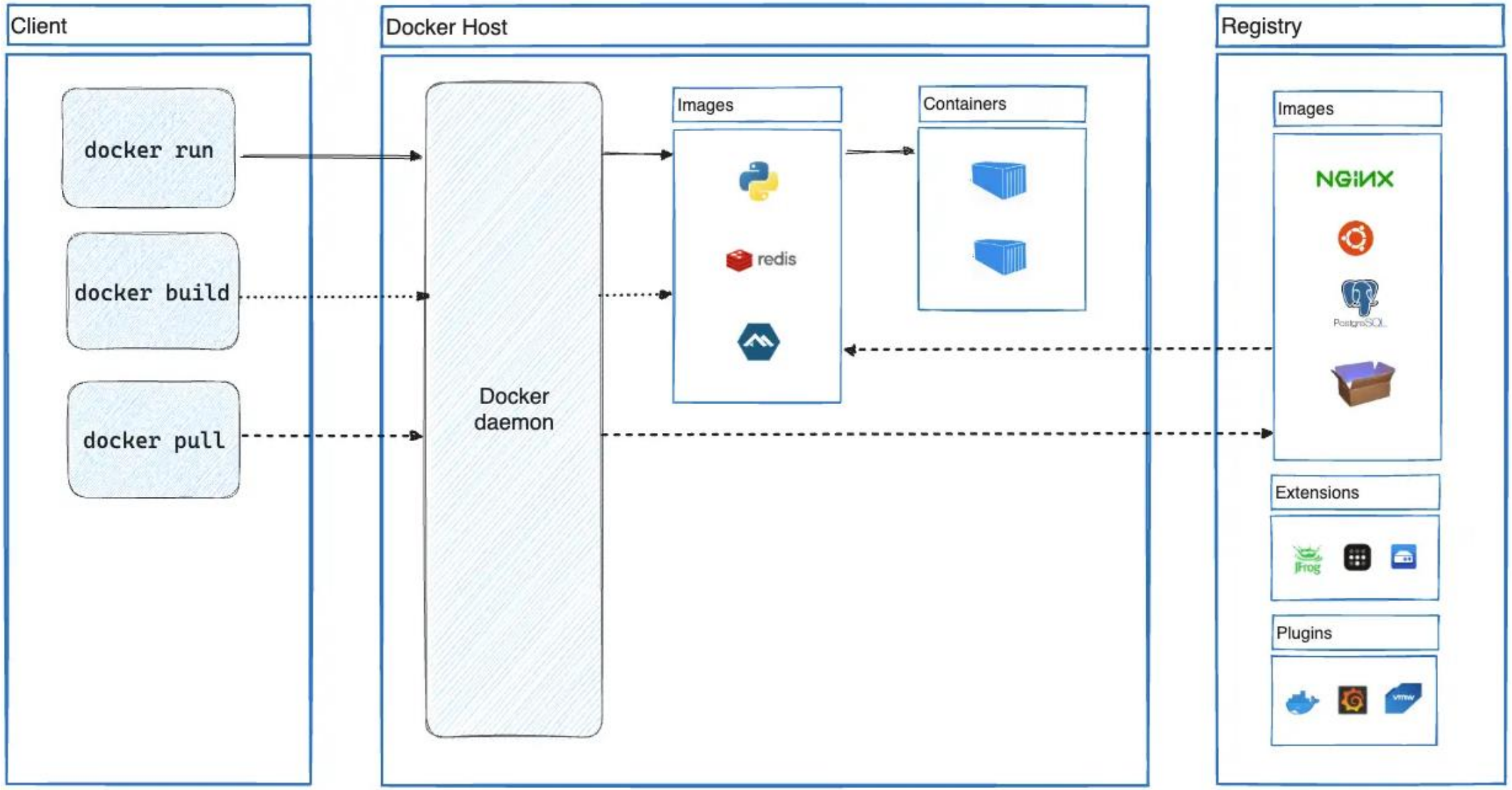
- Quản lý Container Life-cycle.
- Đảm bảo độ cô lập.
- Quản lý tài nguyên.
- Kết nối với Kernel của OS
- Quản lý File System
- Thực hiện đóng gói và giải gói container
- Giao tiếp với các thành phần khác

Docker Engine

- Format cho Container Image
- Phương pháp build container Image
- Quản lý container image
- Quản lý các Instance của container
- Chia sẻ container image
- Chạy container

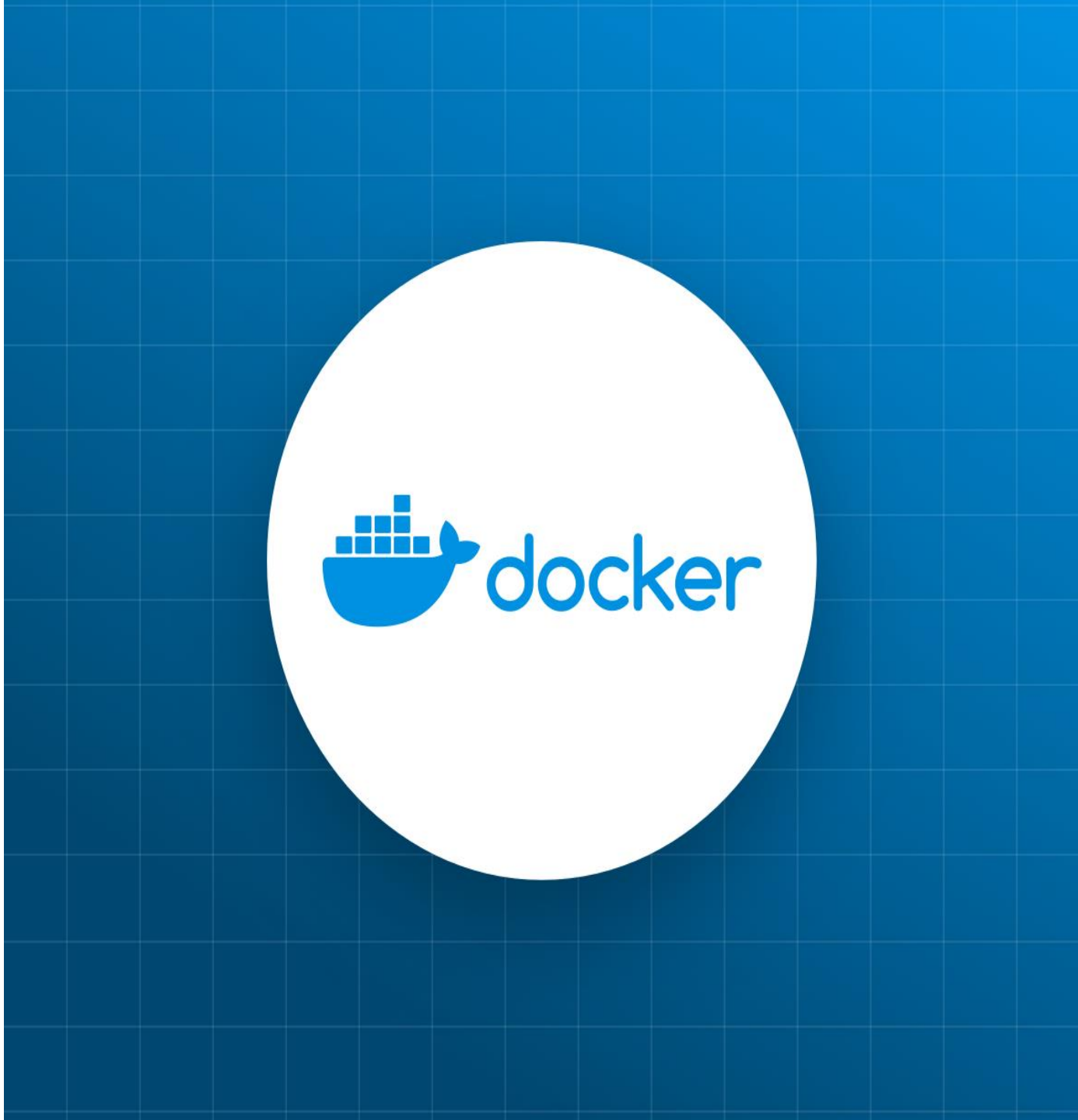


Docker Architecture



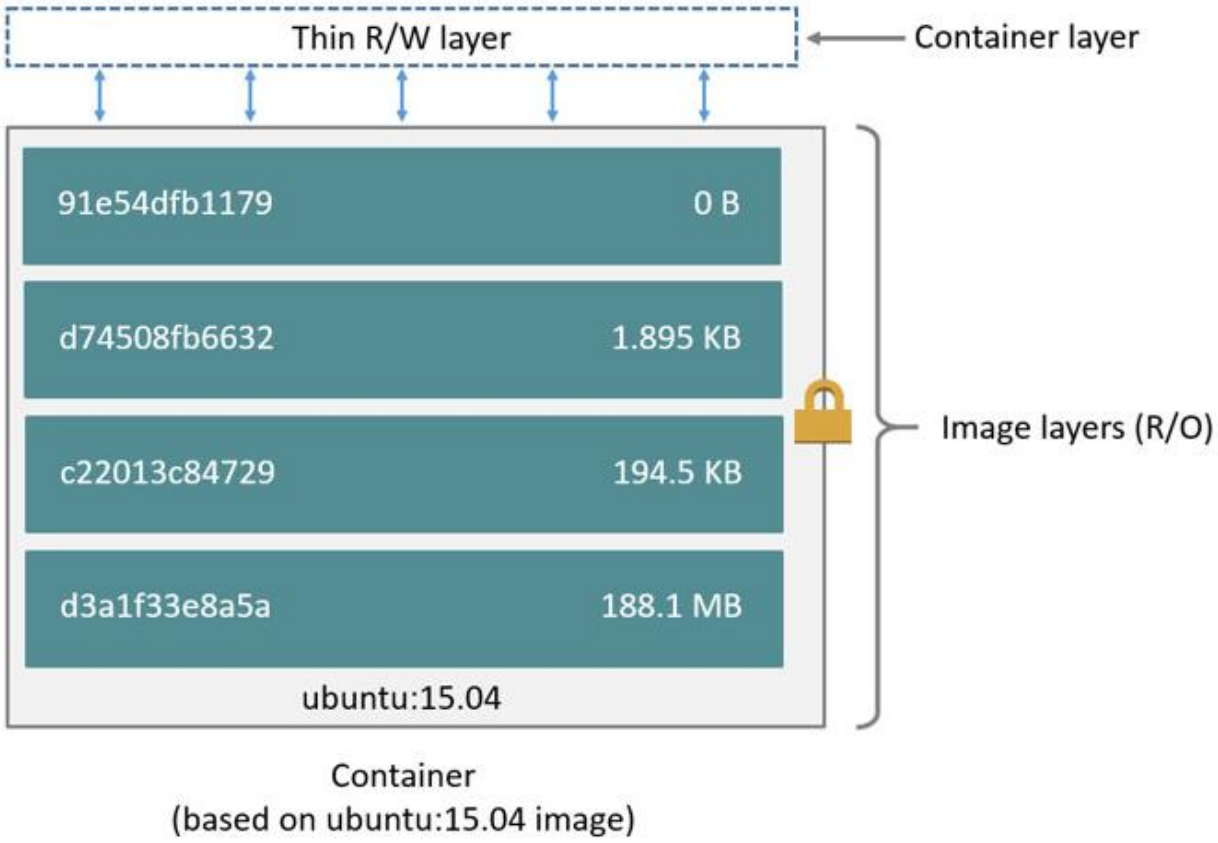
Docker Image

- Docker Image tương tự như VM Template.
- Image gồm 2 thành phần :
  - OS File system
  - Application (file và dependency)
- Có 3 cách để có image :
  - Tạo Image từ container đã dừng
  - Pull image từ các Registry
  - Build image từ Dockerfile



Images Contructs

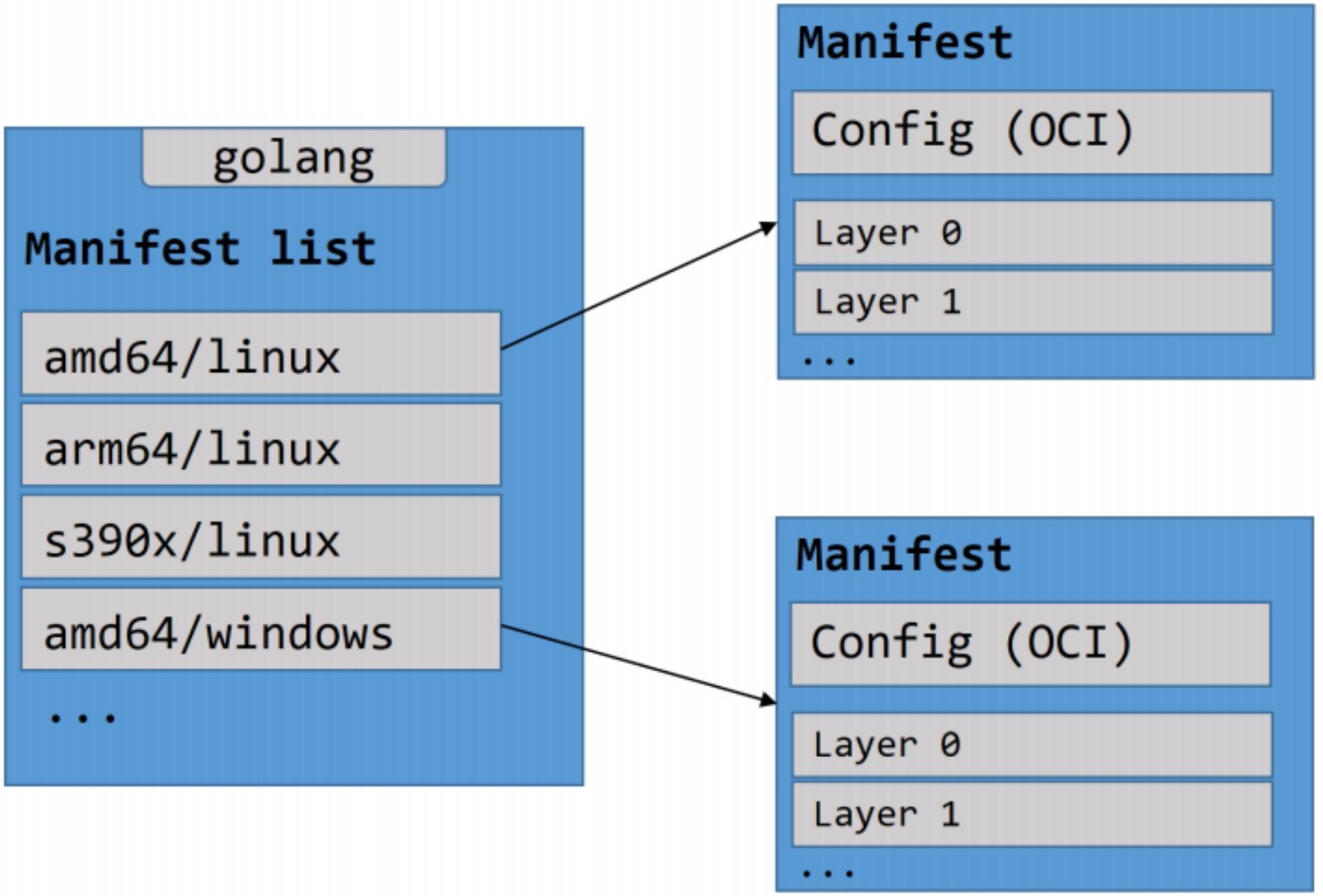
- Sử dụng OverlayFS để xếp chồng các layer.
- Layer dưới cùng là Base Image.
- Tối ưu lưu trữ bằng cách thêm layer dữ liệu.
- Mỗi layer trong image được đại diện bằng 1 mã hash.
- Container cũng là image – read/write.



Dockerfile

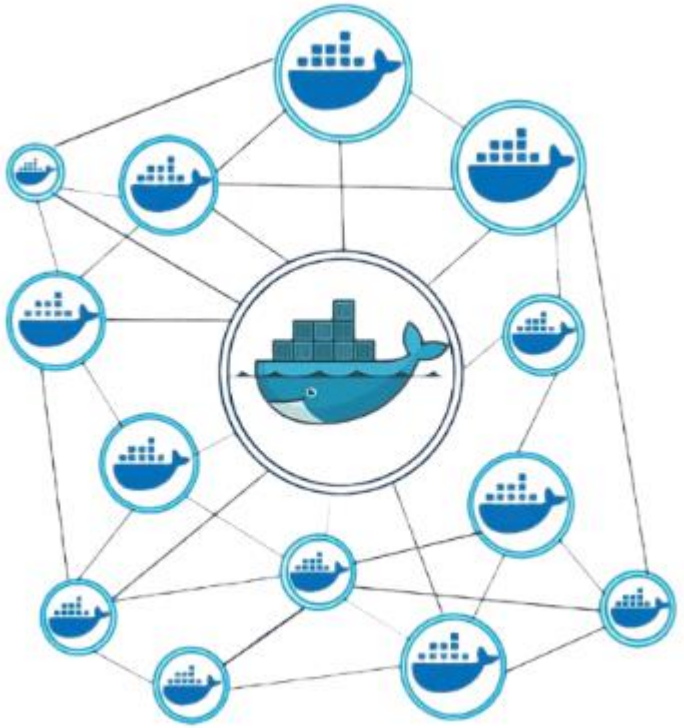
- Chỉ dẫn để docker build lên Image
- Gồm 2 thành phần chính :
  - Instruction : Loại chỉ dẫn
  - Argument : Các đối số đi kèm chỉ dẫn

Multi-Architecture Image



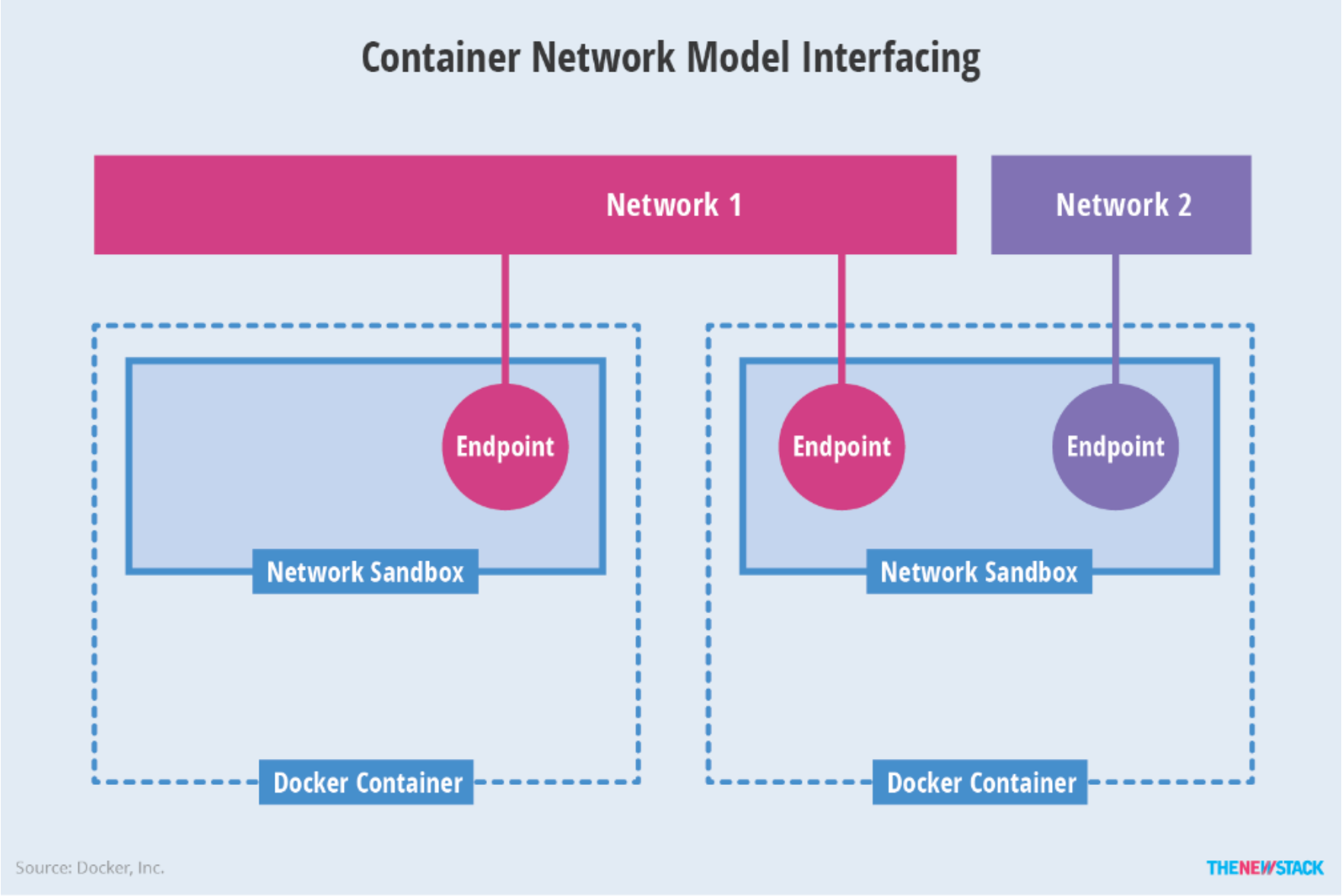
Docker Network

- Iptable rules
- Docker Container Network Model (CNM)
- CNM cung cấp tính linh động cho việc kết nối



CNM Constructs

- Sandbox
- Endpoint
- Network

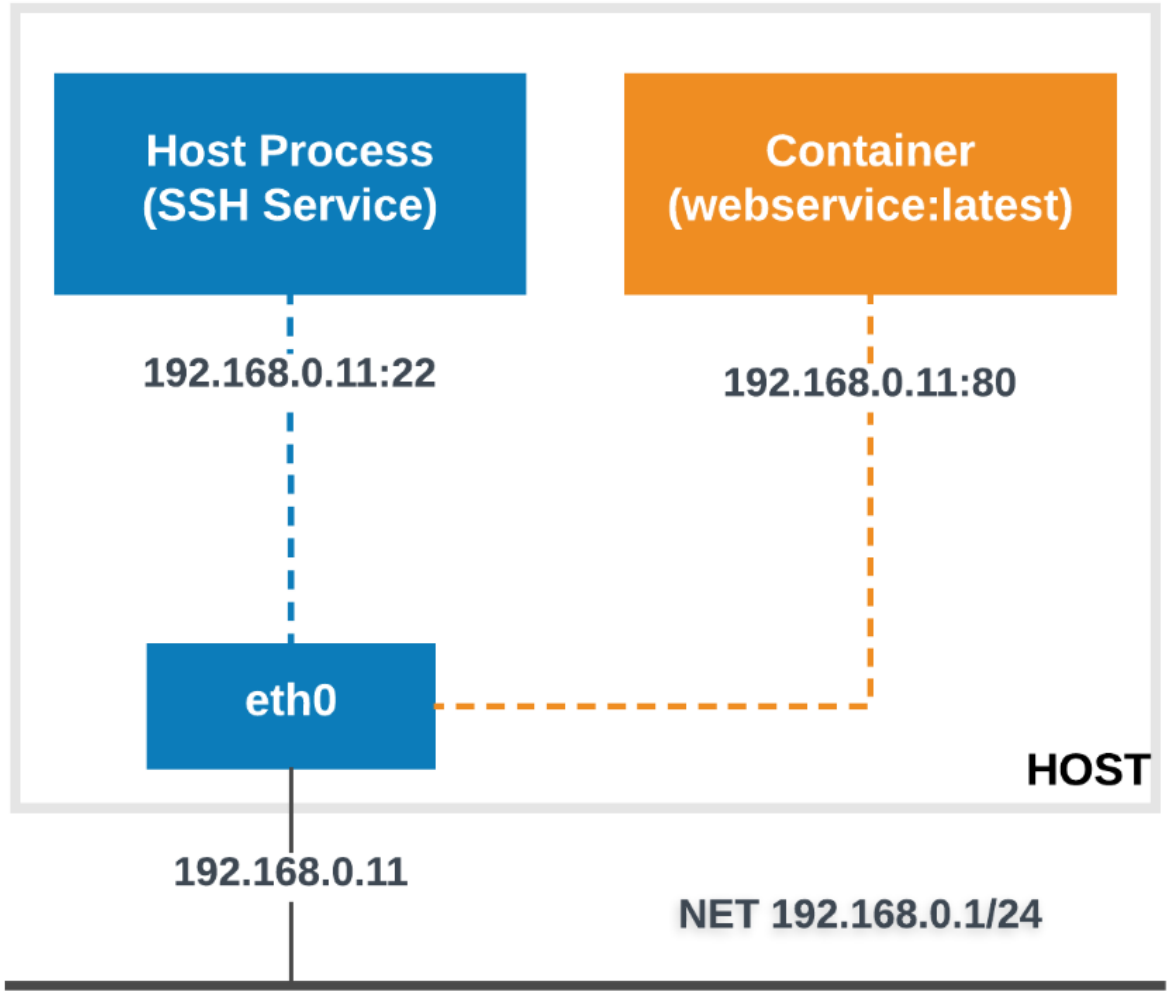


CNM Driver Interfaces

- Docker CNM cung cấp 2 Driver interface :
  - IPAM Drivers
  - Network Drivers
    - Native Network Drivers
    - Remote Network Drivers
- Network Scope :
  - Local
  - Swarm

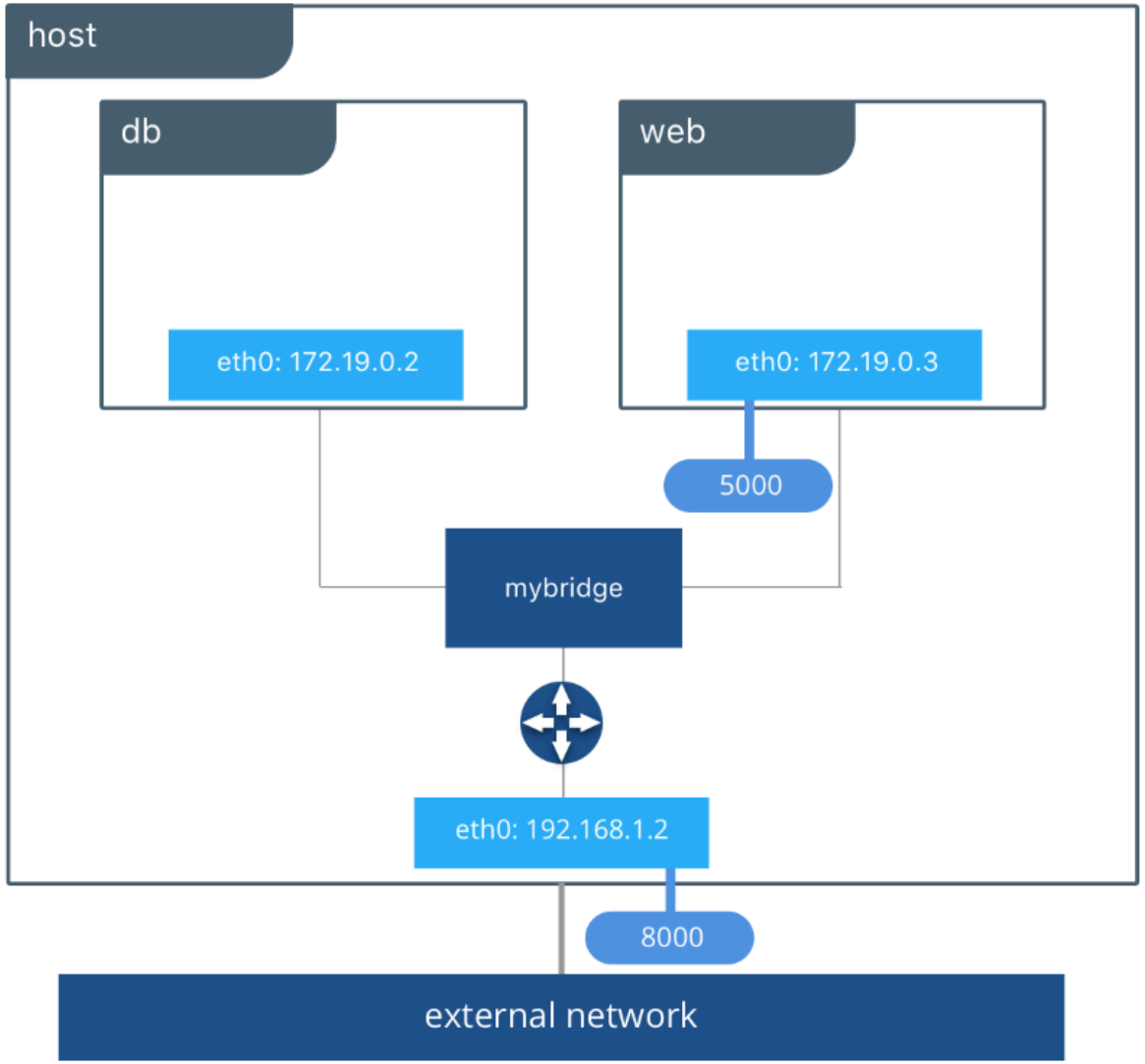
Docker Host network driver

- Sử dụng network thật của Host OS
- Không có sự cô lập
- Không thể bind vào cùng 1 port với container khác



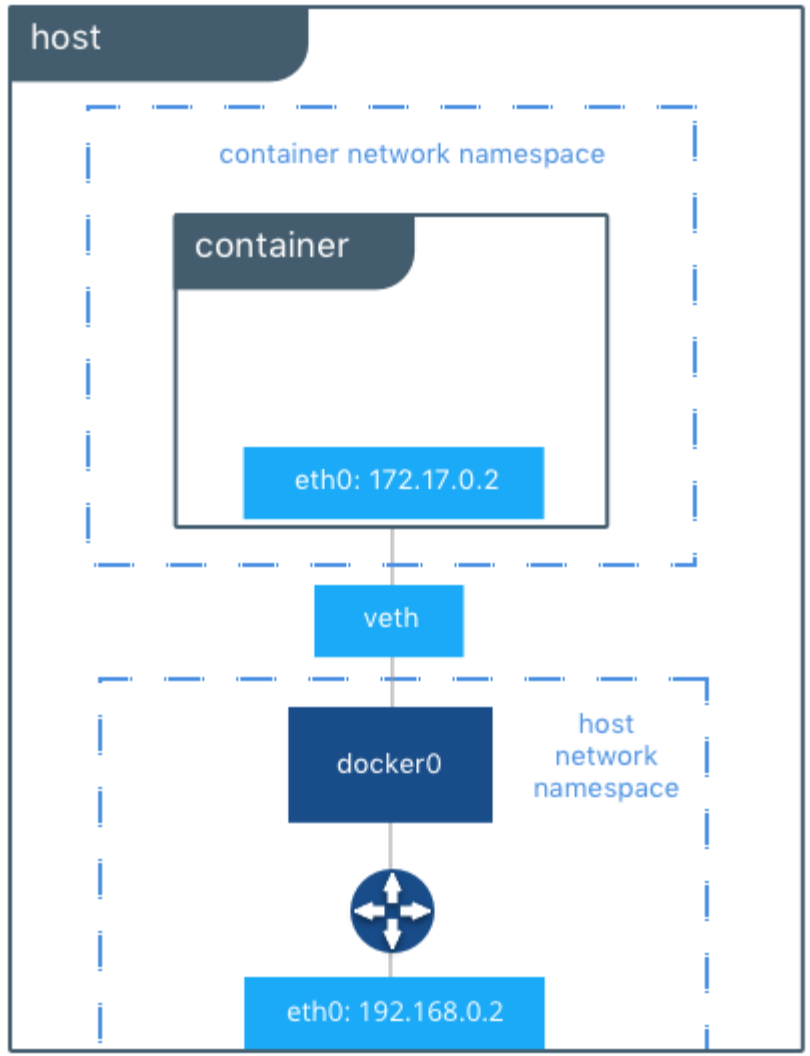
Docker Bridge network driver - Default

- Sử dụng Linux Bridge để tạo và quản lý bridge ảo.
- Linux bridge : Docker0

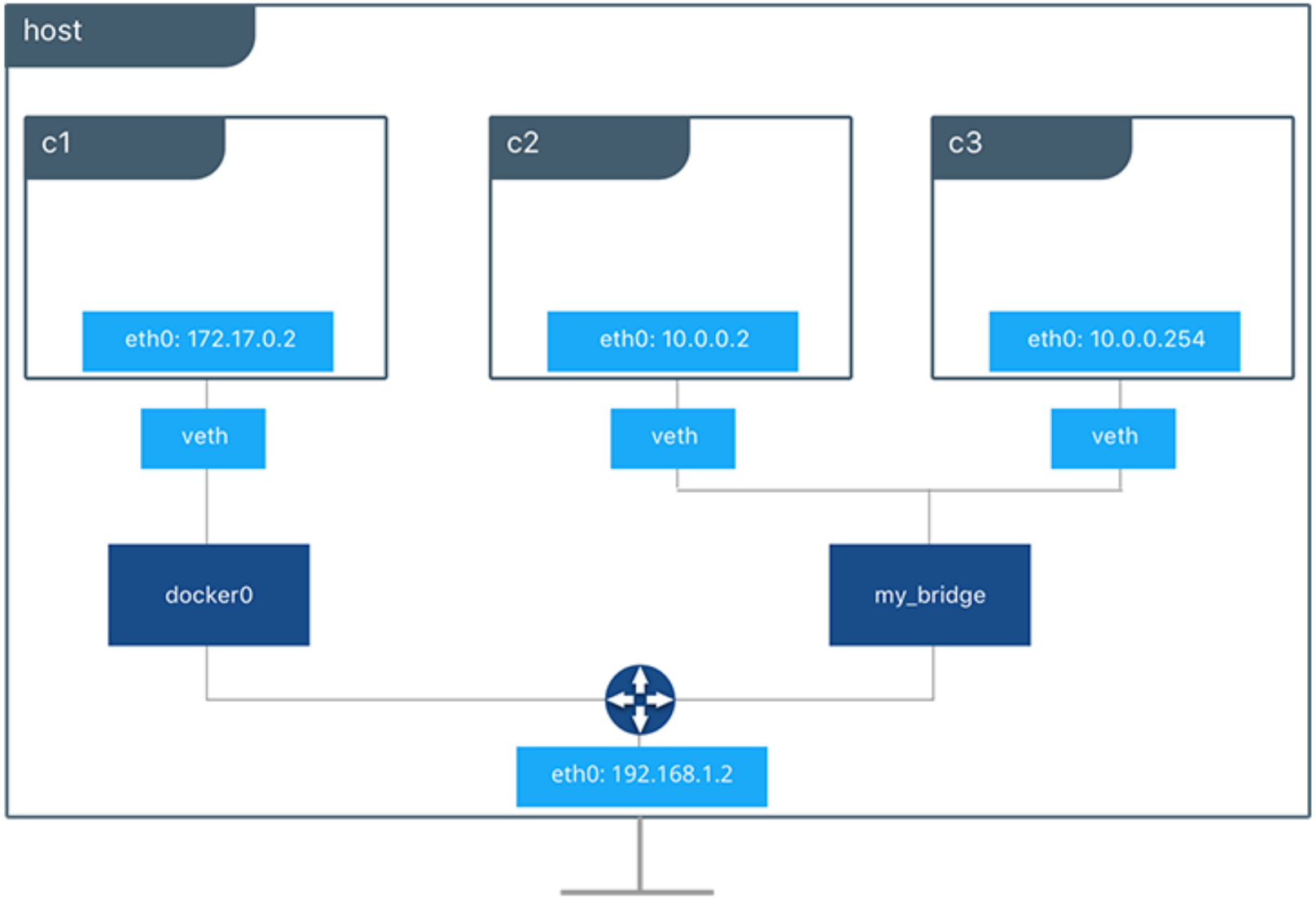


```
docker-mng@ubuntu-server:~$ docker run -it --name c1 busybox /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    valid_lft forever preferred_lft forever
148: eth0@if149: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
    valid_lft forever preferred_lft forever
```

```
docker-mng@ubuntu-server:~$ brctl show
bridge name     bridge id               STP enabled     interfaces
br-1553f9d11ab1         8000.0242a1667e41       no
br-43587b557ae6         8000.0242333b1ea0       no
br-c493b7f4d455         8000.0242578570bf       no
docker0         8000.024222772dbe       no              vethda8d66a
```

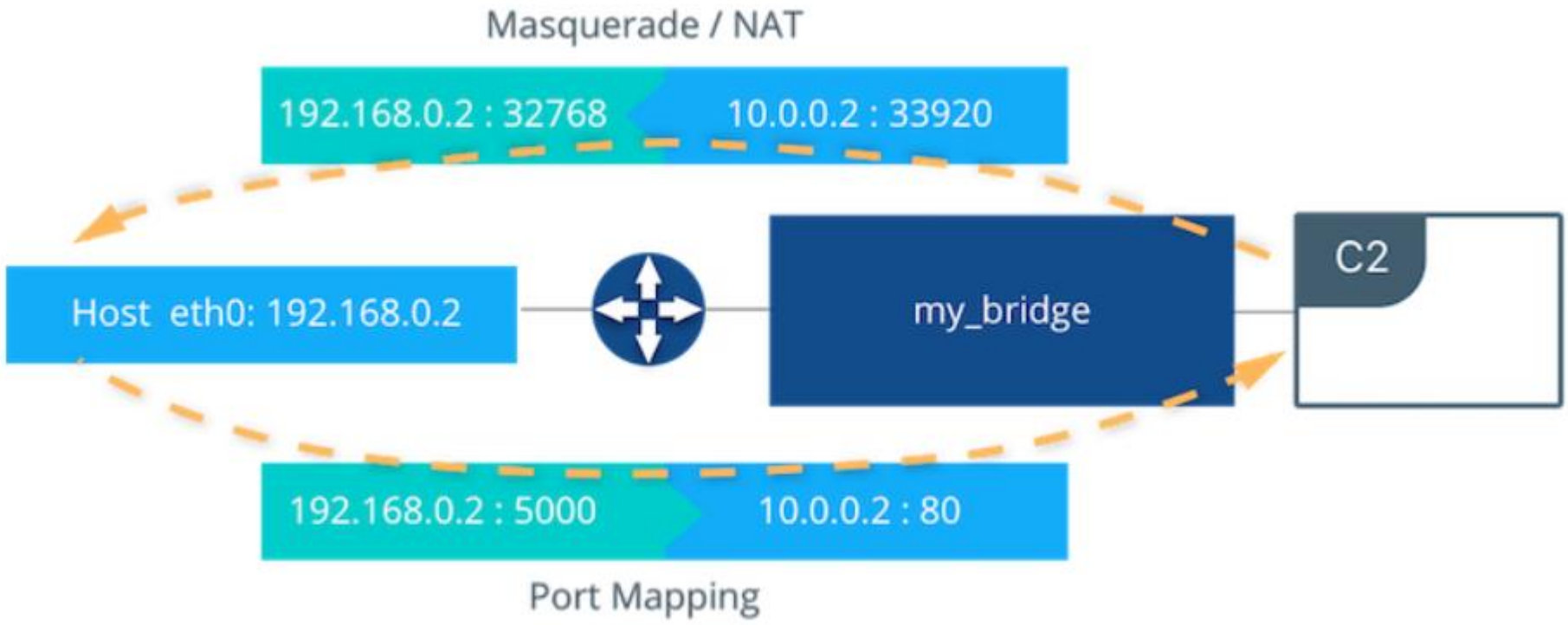


User-Defined Bridge Networks

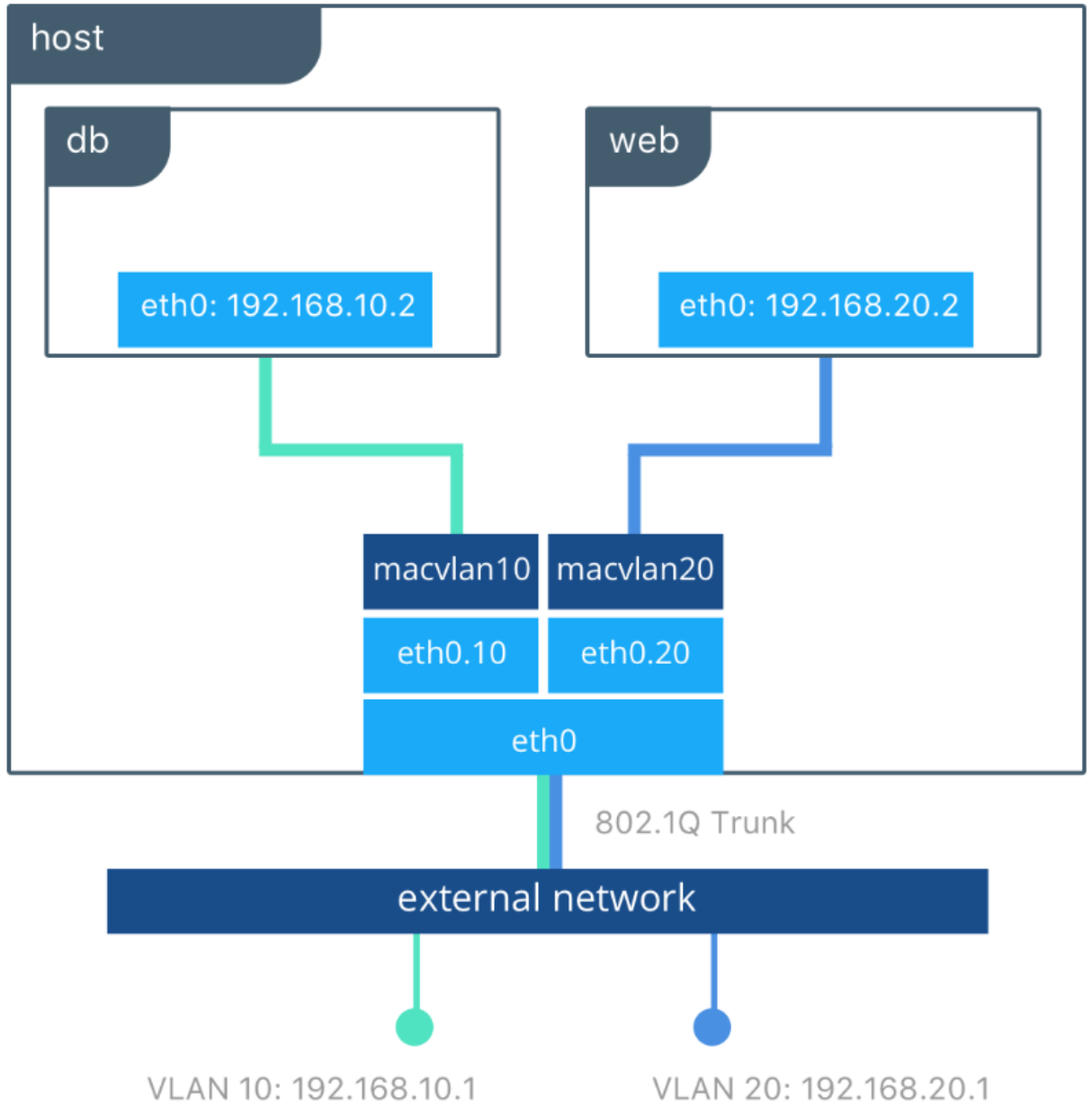


External Access to Container

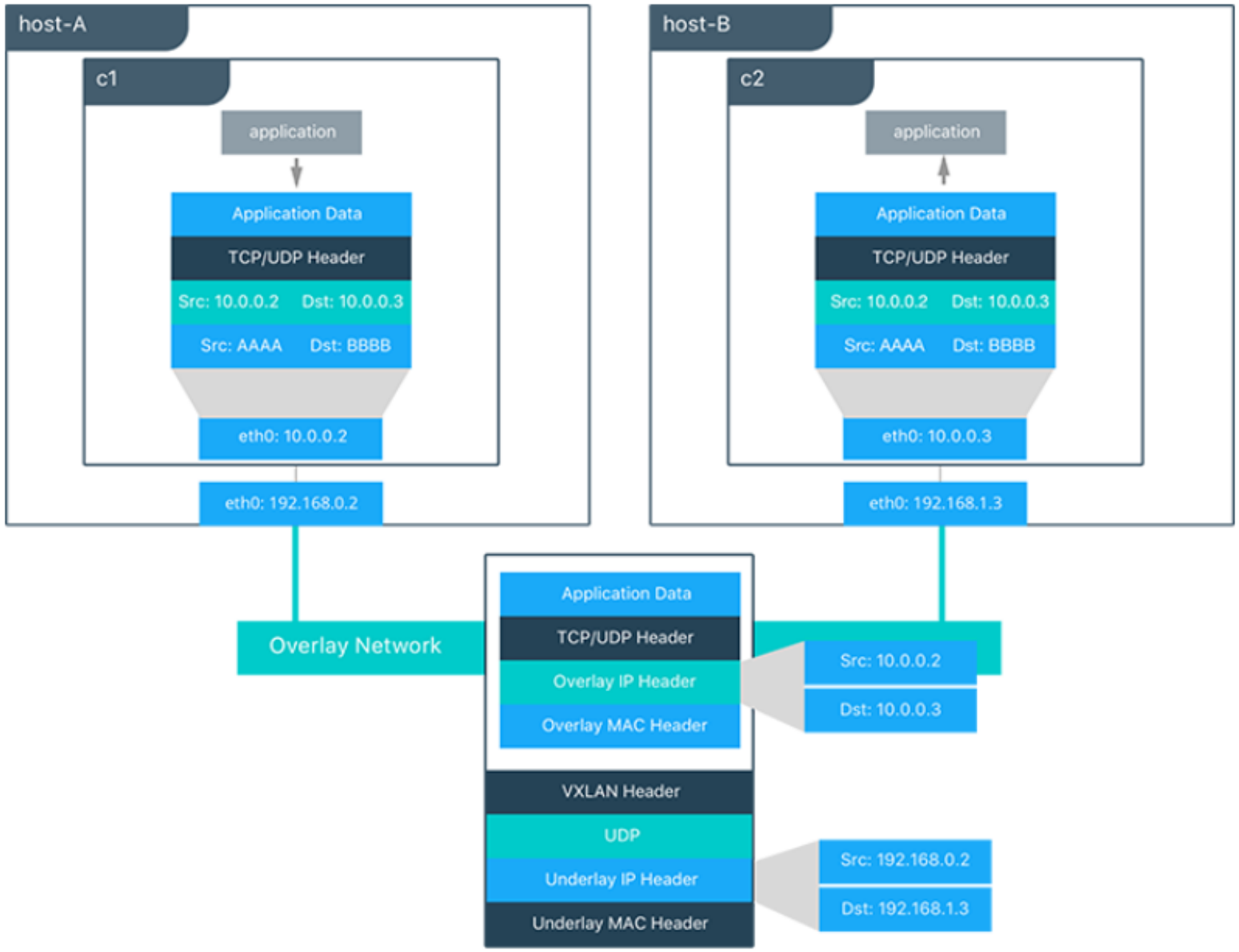
- Exgress – masqueraded/SNAT
- Ingress – port mapping



MACVLAN (Transparent)

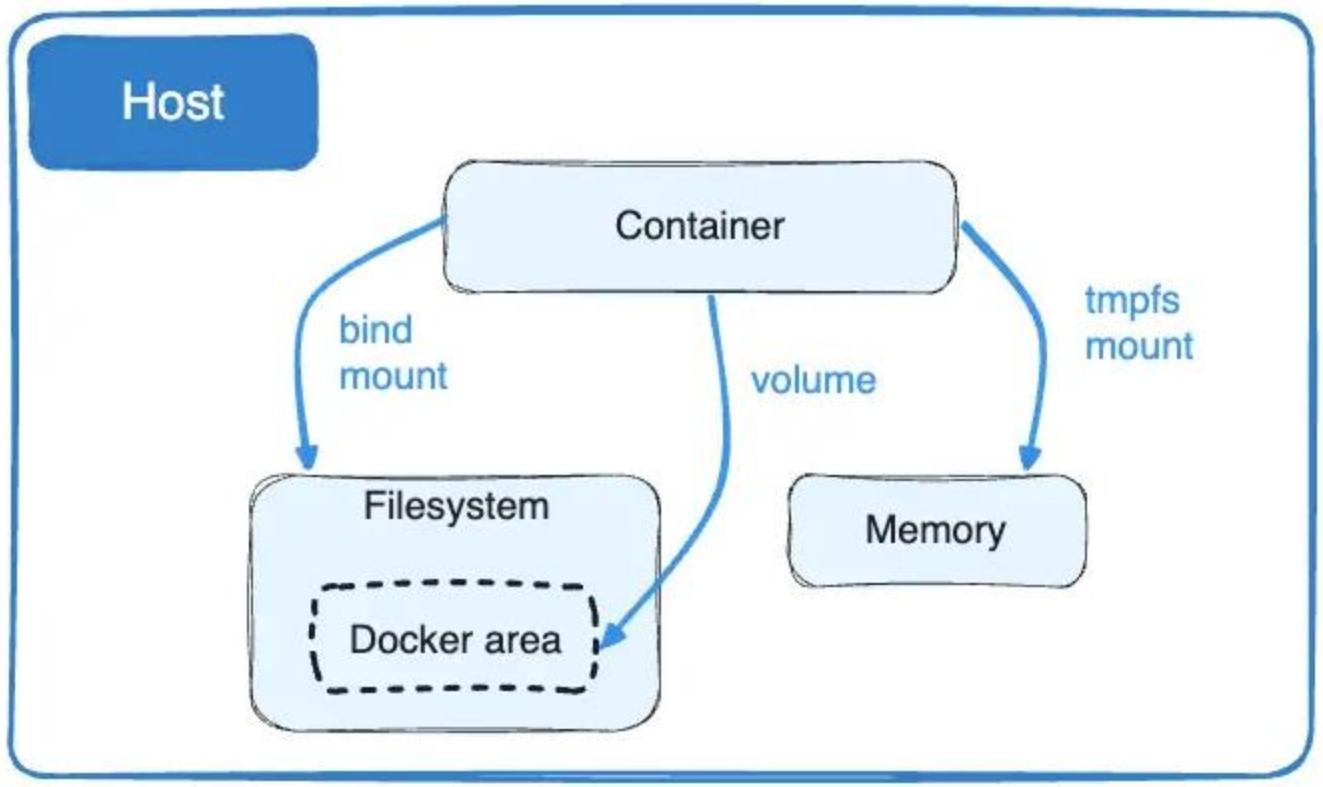


Overlay Driver Network Architecture



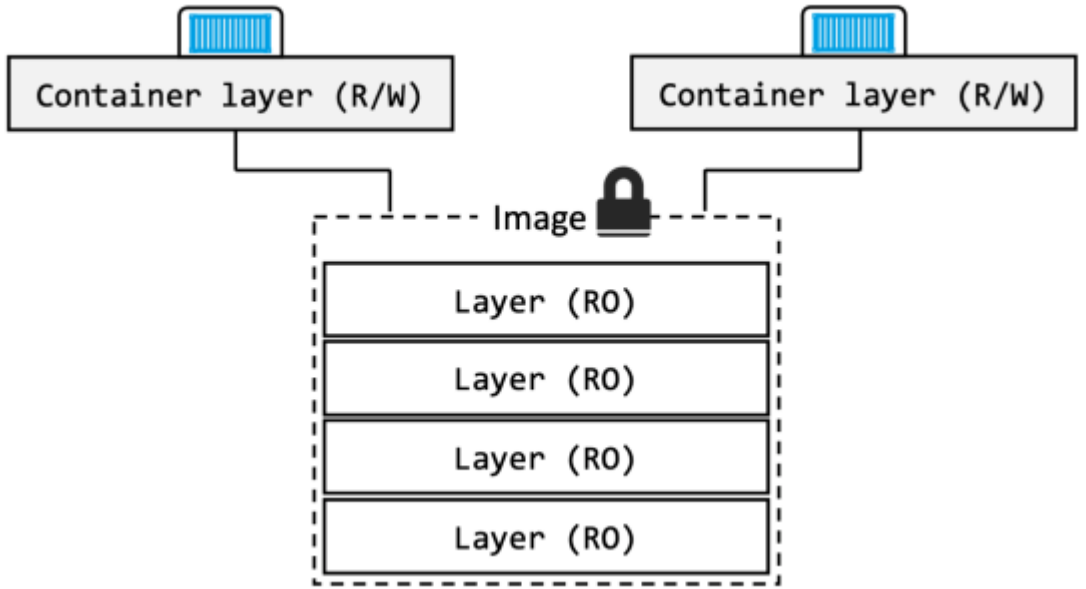
Docker Storage – on a Docker host

- Khái quát về 2 kiểu dữ liệu
  - Persistent data
  - Non-Persistent data



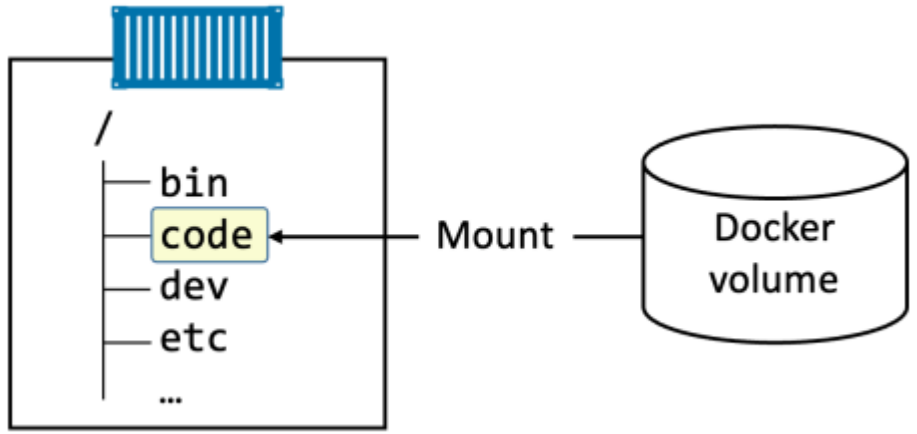
Non-Persistent data

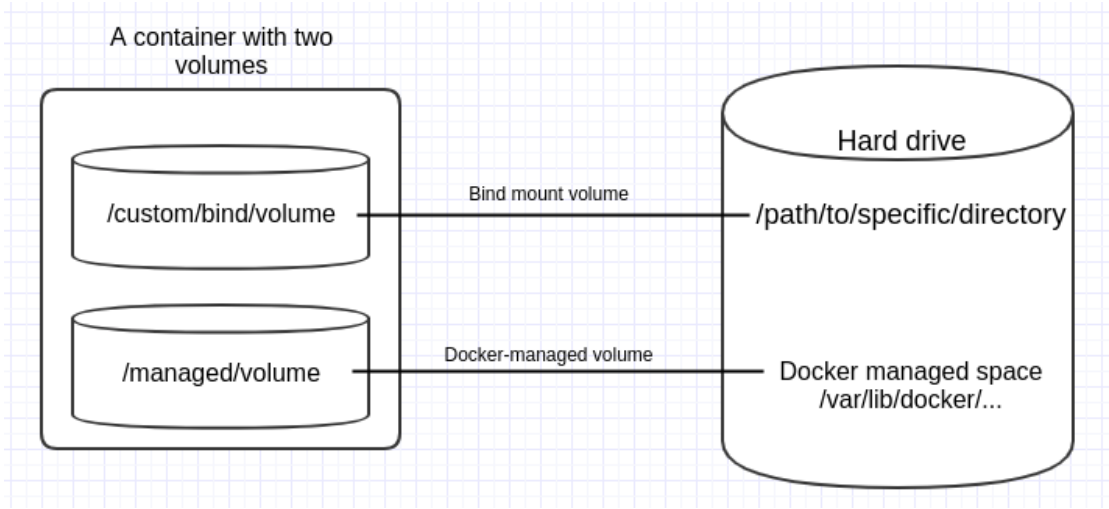
- Thiết kế bất biến – fix lỗi bằng re-deploy
- File system – read/write
- Non-persistent storage – life-cycle



Persistent Data

- Đối với Persistent :
  - Volume
  - Bind mounts





Automation deployed with Docker-Compose

- Tự động hóa quá trình triển khai trên Docker Host
- Tối ưu để triển khai các ứng dụng microservice
- Quản lý tập trung nhiều service bằng 1 file cấu hình
- Sử dụng file YAML hoặc JSON