Docker Networking

Agenda

In this session, you will learn about:

- Networking Overview
- Understanding CNM
- Network Drivers

Scope of this section

- This section does not go into **OS-specific** details about how Docker networking works.
- Example:
  - Linux Bridge
  - iptables
  - Packet Encapsulation/Decapsulation

Networking overview

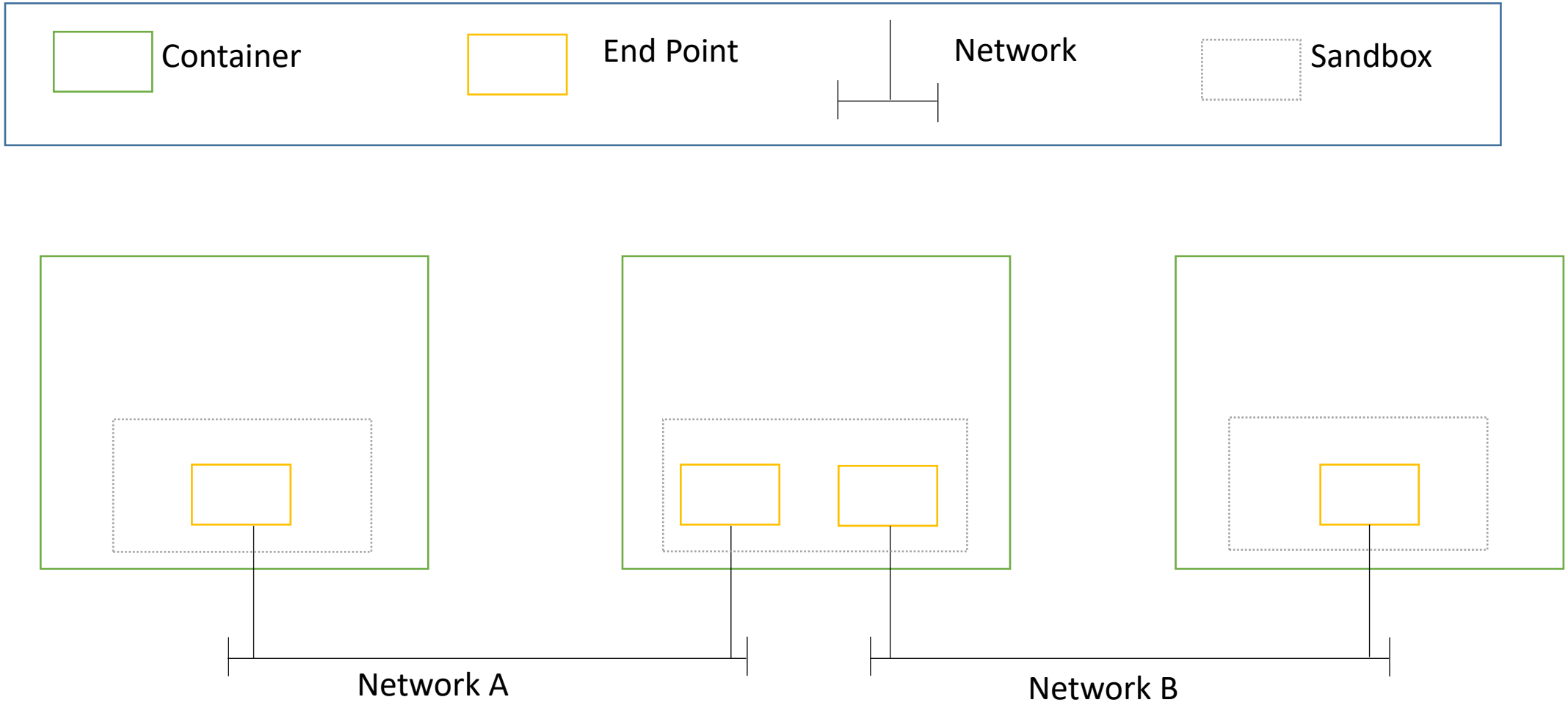
- Docker containers uses **Linux Bridge** feature by default to communicate.

- Whether your Docker hosts run Linux, Windows, or a mix of the two, you can use Docker to manage them in a platform-agnostic way.

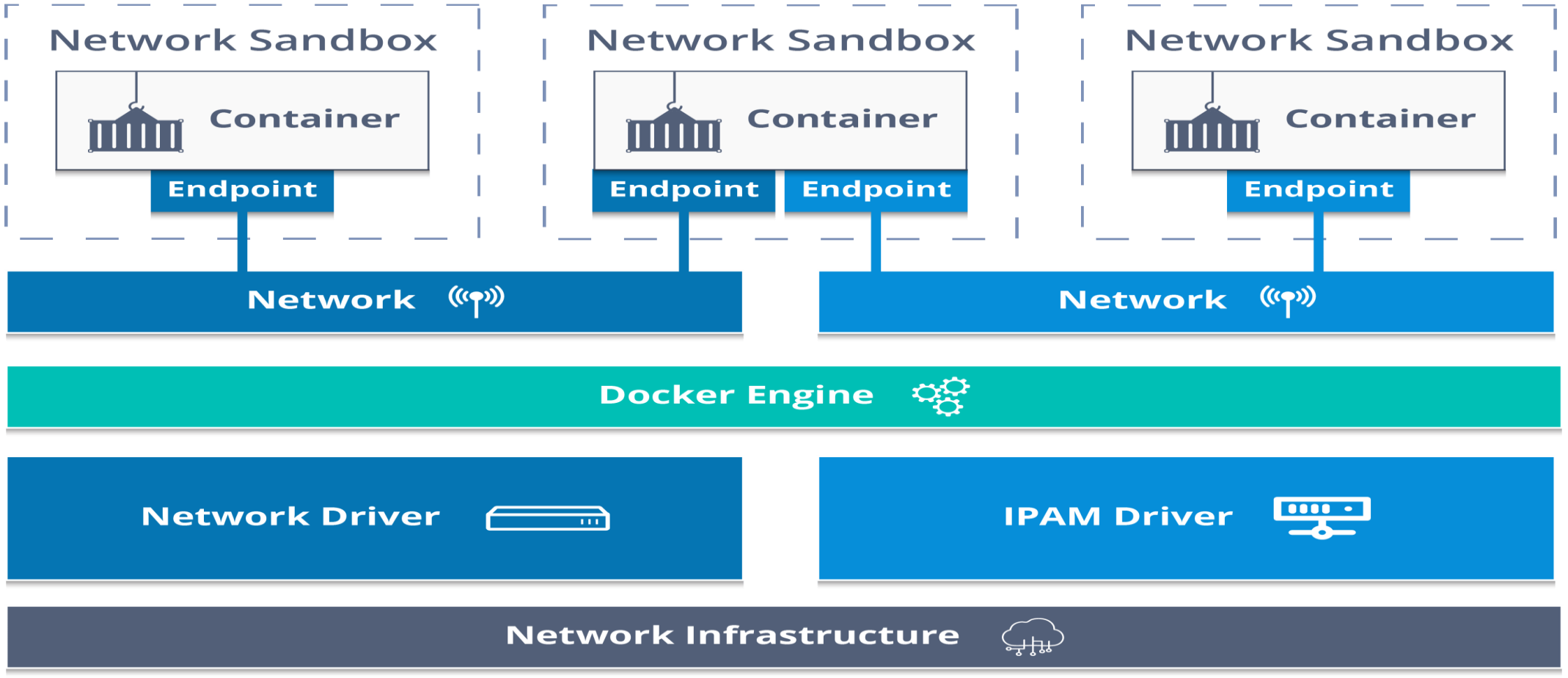
CNM – Container Network Model

- **CNM** is an open-source networking specification for containers.
- **CNM** defines Networks, Endpoints, Sandboxes.
- **Libnetwork** is Docker's implementation of CNM.
- **Libnetwork** is extensible via pluggable drivers which allows various network technologies.

Containers and CNM

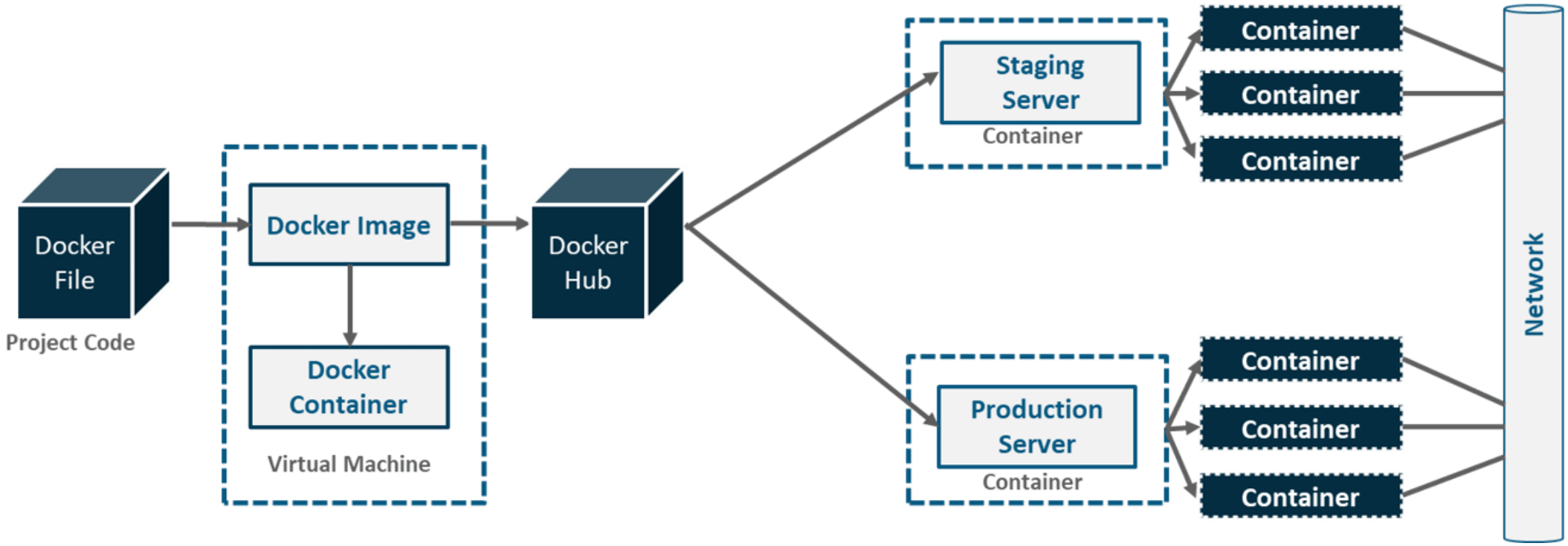


Architecture of Container Network Model



Architecture of Container Networking Model

Networking - Workflow



CNM built on 5 Objects

- A CNM has mainly built on 5 objects:
  - Network Controller
  - Driver
  - Network
  - Endpoint
  - Sandbox

Network Controller

Provides the entry-point into Libnetwork that exposes simple APIs for Docker Engine to allocate and manage networks.

Driver

- Owns the Network
- Responsible for managing the Network
- Supports multiple drivers to satisfy various use-cases and deployment scenarios.

Network

- Provides connectivity between a group of endpoints that belong to the same network and isolate from the rest.
- Whenever a network is created or updated, the corresponding Driver will be notified of the event.

Endpoint

- Provides the connectivity for container in a Network.

Sandbox

- Created when users request to create an endpoint on a network.
- A Sandbox can have multiple endpoints attached to different networks representing container’s network configuration such as:
  - IP-Address
  - MAC-Address
  - Routes
  - DNS

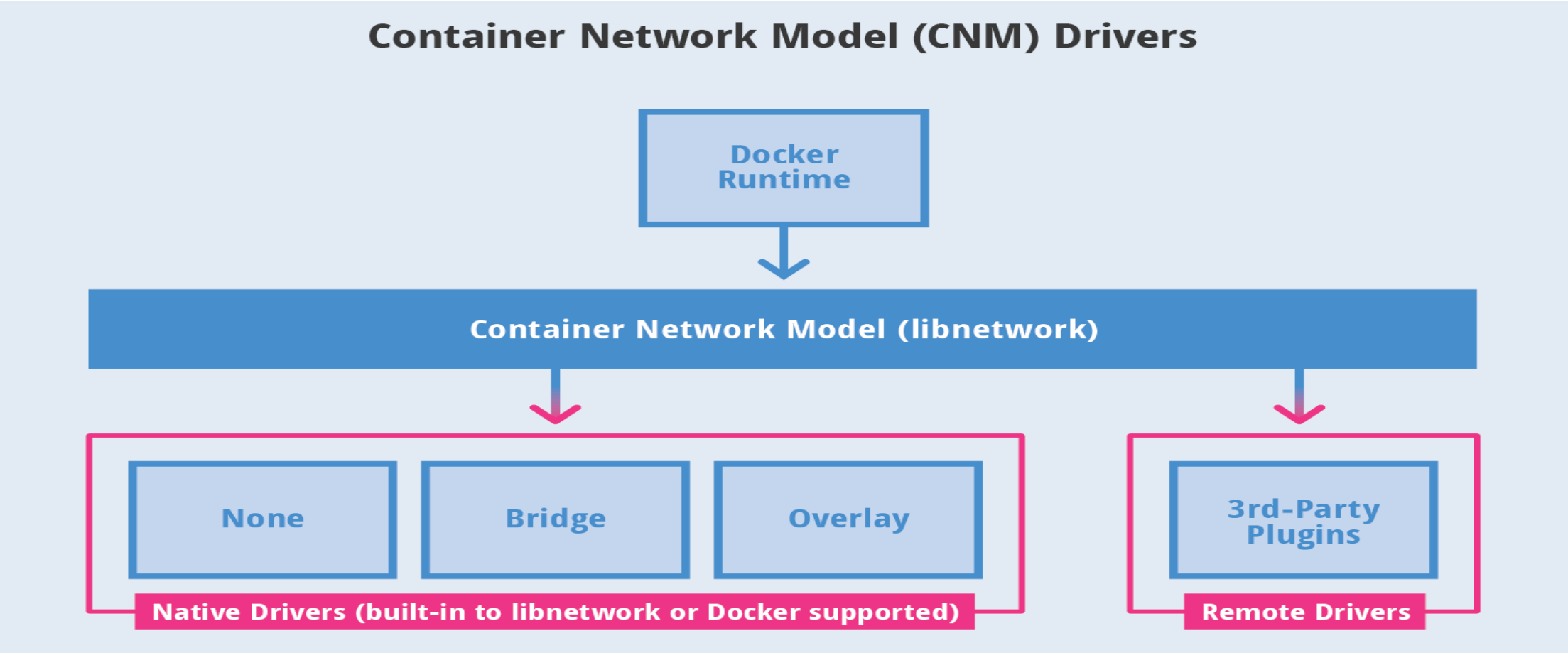
Network Drivers

Network Drivers

- bridge
- host
- overlay
- macvlan
- none

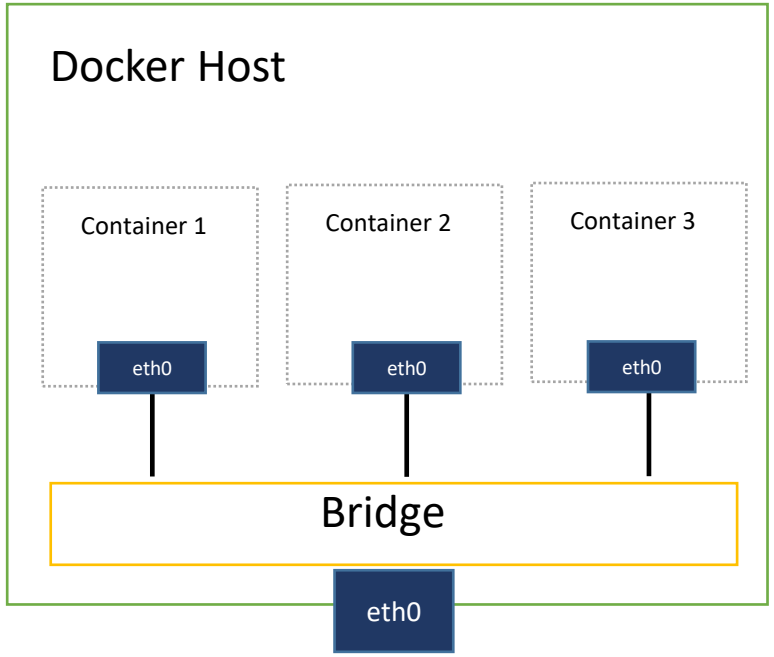
```
$ docker info

<snip>
…
Plugins:
 Volume: local
 Network: bridge host macvlan none overlay
…
<snip>
```

Network Drivers



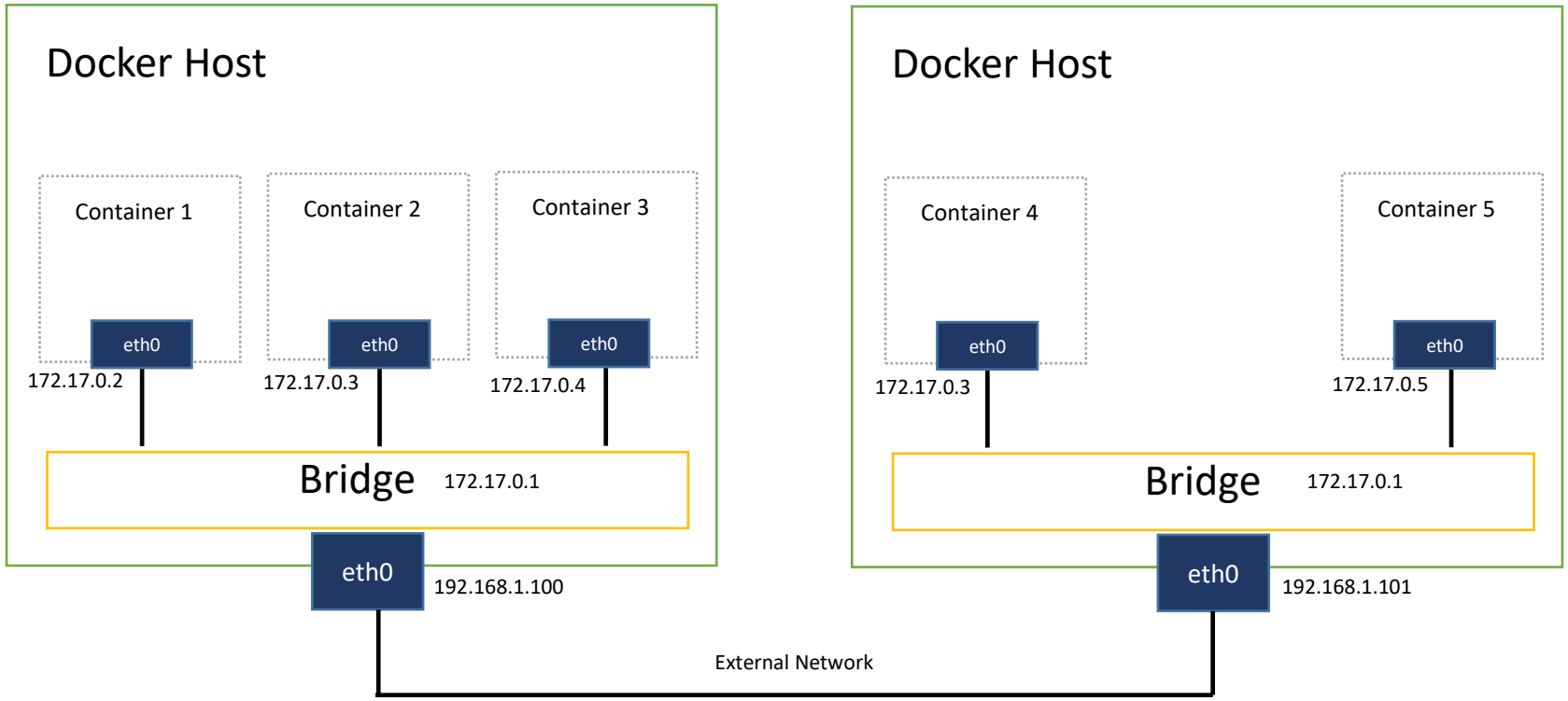
Bridge Driver

- Default driver when a container is run
- Additional bridge networks can be created
- Bridge networks apply to containers running on the same host
- Has its own internal IP address allocation mechanism
- Example : 172.17.0.0/16 (Default)
- Very robust



Bridge Driver

- Bridge networks are usually used when your applications run in standalone containers that need to communicate.
- Container to external network done via PAT (iptables)

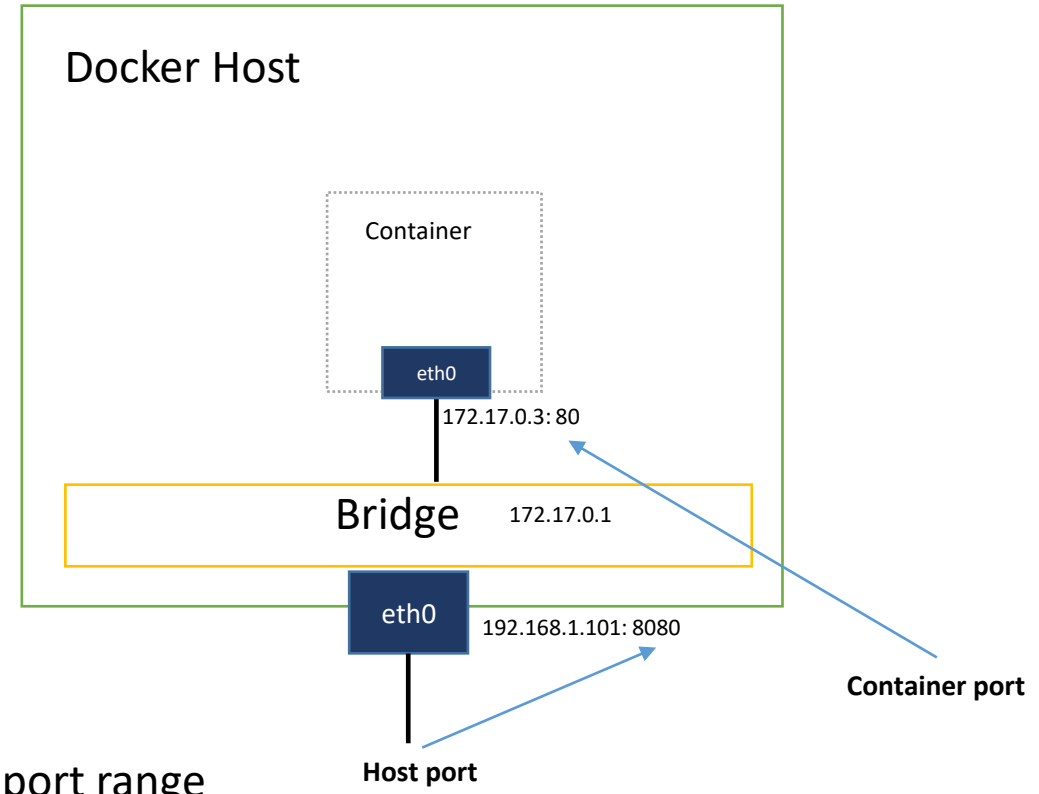


Bridge Networking and Port Mapping

- Access to containers via port mapping
- Map container port to host port

docker container run -p 8080:80 nginx

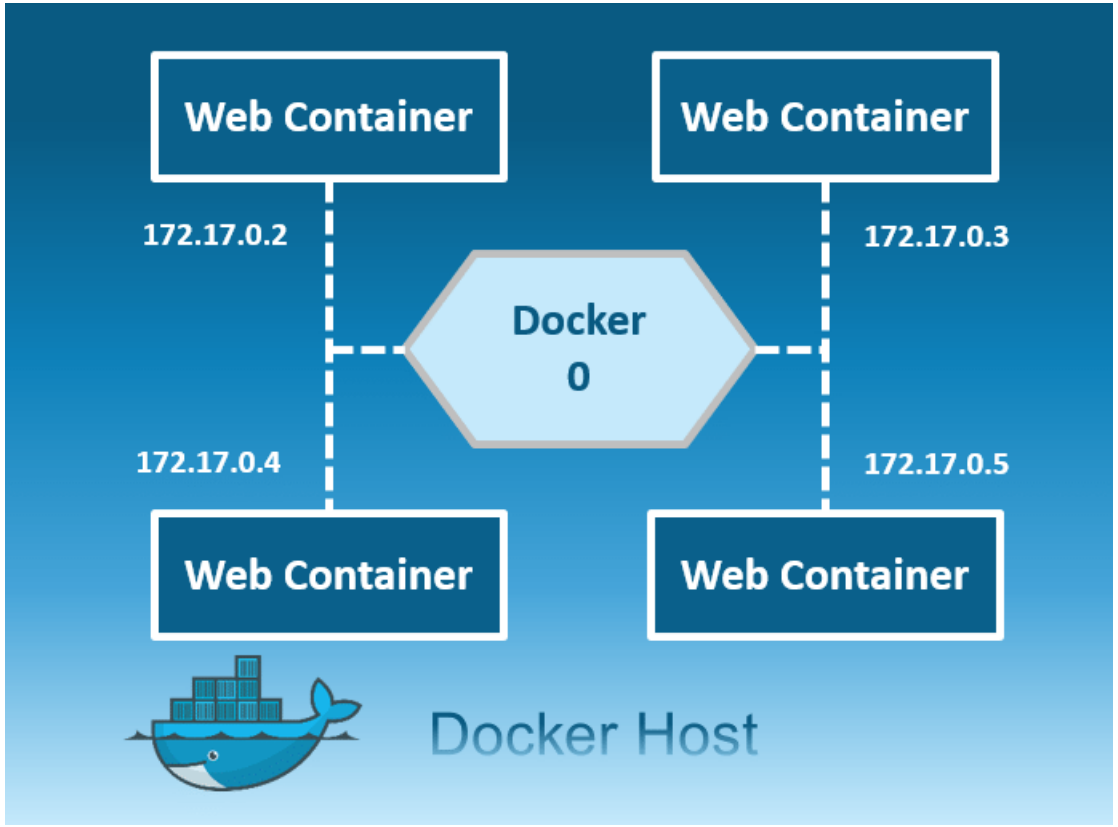
Host Port – 8080
Container Port - 80



Dynamic port range

cat /proc/sys/net/ipv4/ip_local_port_range
32768 to **60999**

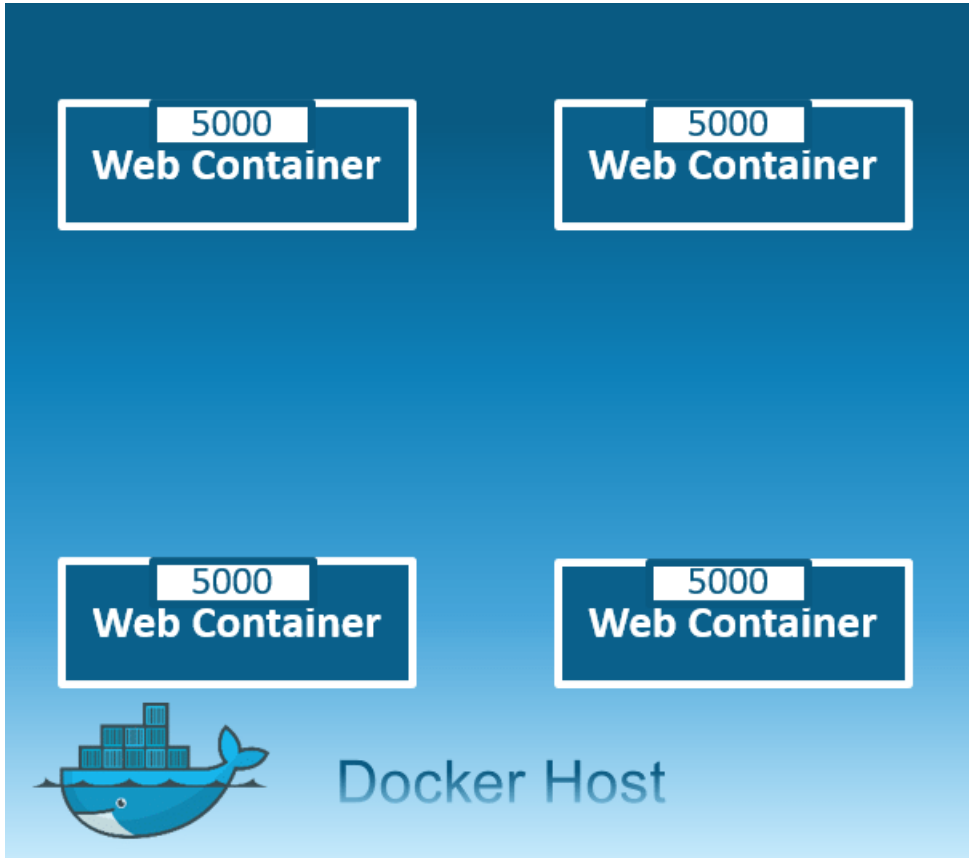
Example – Bridge driver



Host

- Driver removes the network isolation between the docker host and the docker containers to use the host's networking directly.
- Will not be able to run multiple web containers on the same host, on the same port as the port is now common to all containers in the host network.

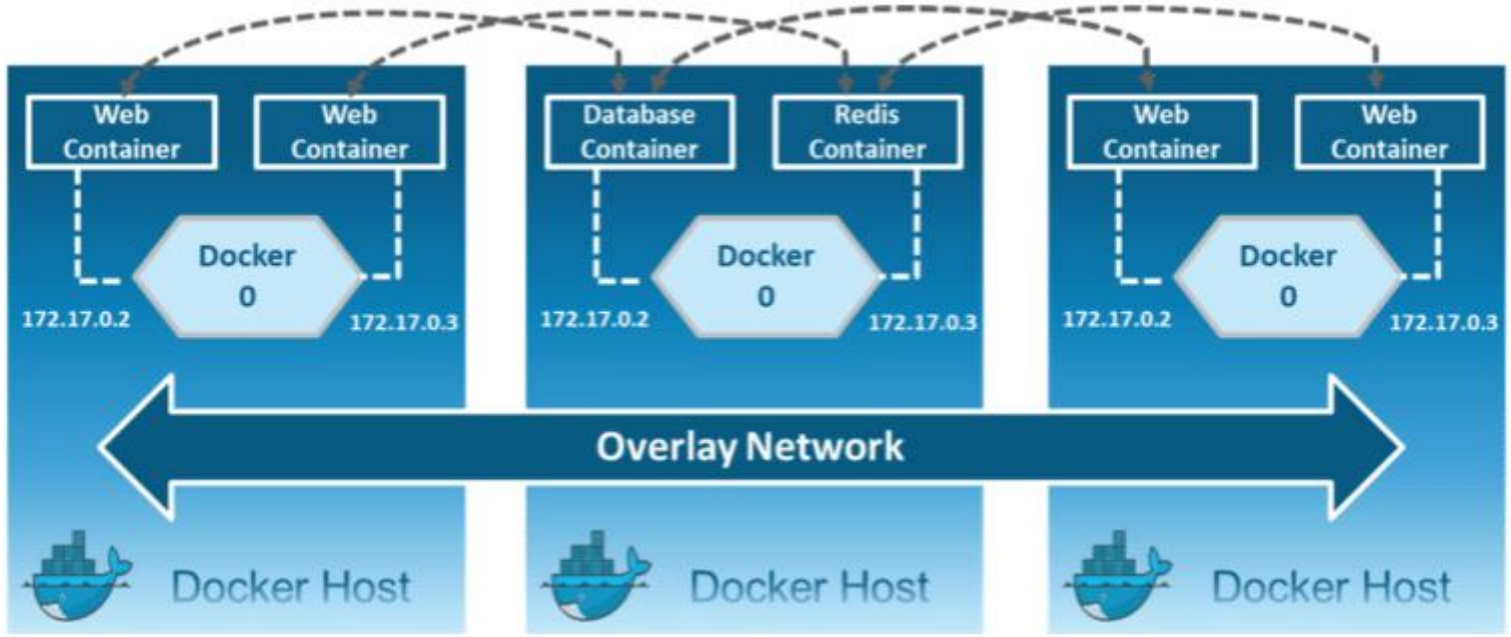
Example – Host driver



Overlay

- Creates an internal private network that spans across all the nodes participating in the swarm cluster.
- So, Overlay networks facilitate communication between a swarm service and a standalone container, or between two standalone containers on different Docker Daemons.

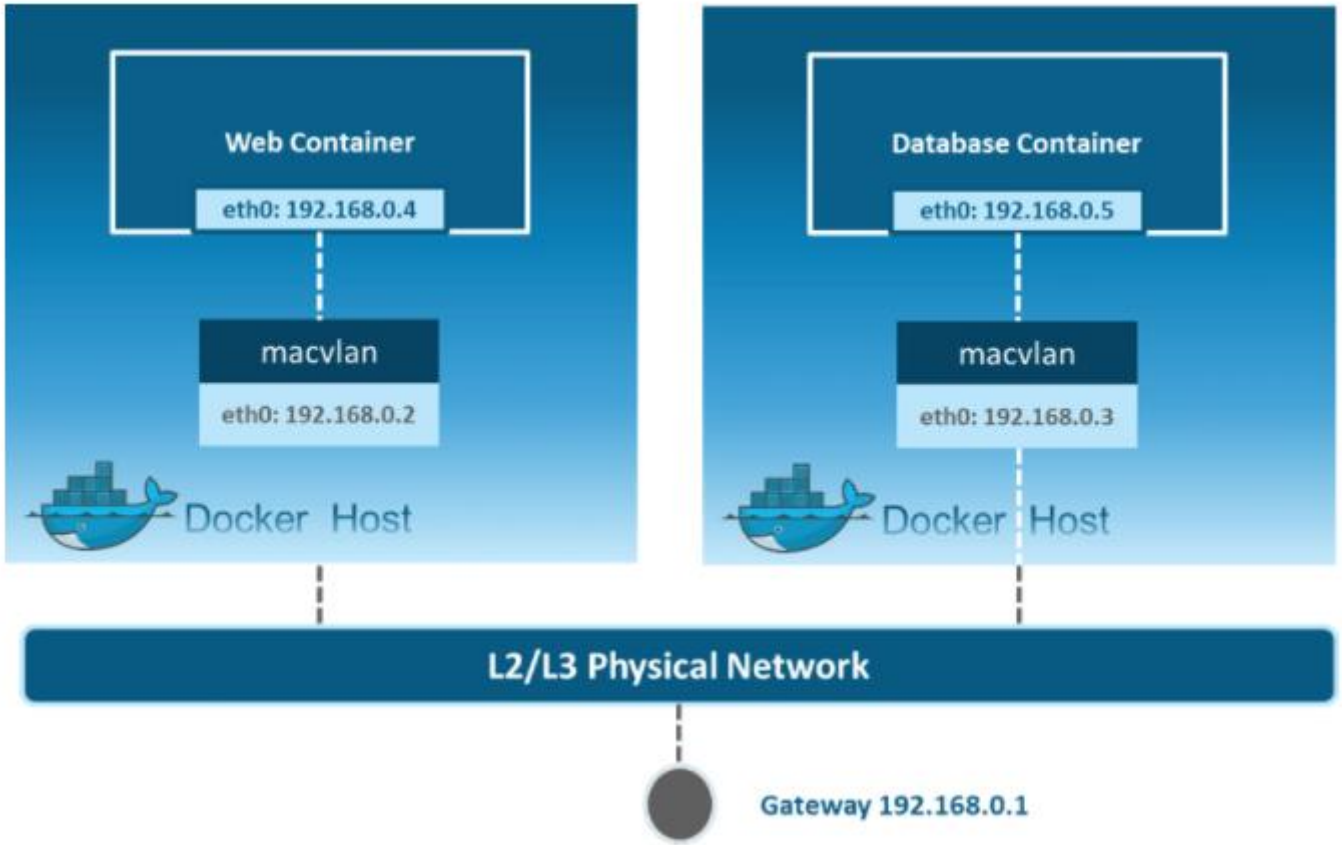
Example – Overlay driver



Macvlan

- Allows you to assign a MAC address to a container, making it appear as a physical device on your network.
- Docker daemon routes traffic to containers by their MAC addresses.
- Macvlan driver is the best choice when you are expected to be directly connected to the physical network, rather than routed through the Docker host’s network stack.

Example – Macvlan driver



Disable Networking for Container

- Use the **--network none** flag when starting the container to disable networking
- Within the container, only the loopback device is created

Networking Drivers Summary

- **Bridge networks -** Best when you need multiple containers to communicate on the same Docker host.
- **host:** Uses the host's networking directly.
- **none:** Disables all networking.
- **Overlay networks** - Best when you need containers running on different Docker hosts to communicate, or when multiple applications work together using swarm services.
- **Macvlan networks** - Best when you are migrating from a VM setup or need your containers to look like physical hosts on your network, each with a unique MAC address.
- **Third-party network plugins** - allow you to integrate Docker with specialized network stacks.

Basic Docker Network Commands

- `docker network ls`
- `docker network create`
- `docker network inspect`
- `docker network rm`

Lab – Docker Networking