ondia

Docker Networking

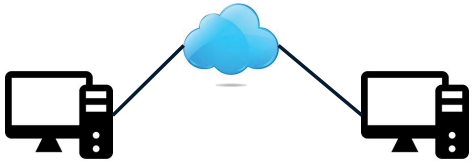
AGENDA

- Networking overview
- Network drivers
- User-defined bridge networks
- Run - Port mappings
- Other Network drivers
- docker network Commands

Networking overview

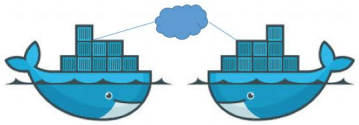
Networking overview

A **network** is two or more computer systems linked together by some form of the transmission medium.



Networking overview

- One of the reasons Docker containers and services are so powerful is that you can connect them together, or connect them to non-Docker workloads.
- 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.

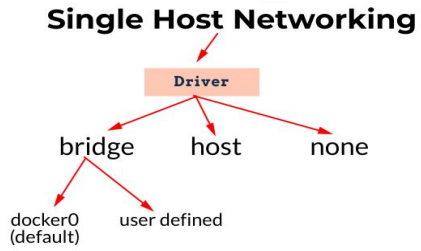


Docker networks

Network drivers

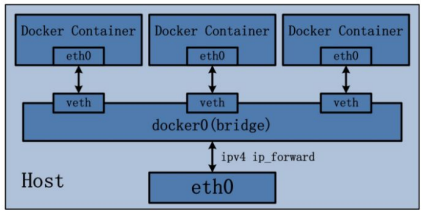
Network drivers

As default, docker has three network drivers.



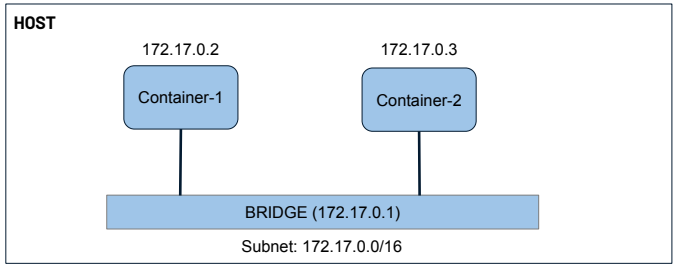
Network drivers

- **bridge** is the default network driver. If we don't specify a driver, this is the type of network we are creating.
- When we install the docker, the Docker daemon creates virtual ethernet bridge **docker0** that performs the operation by automatically delivering packets among various network interfaces.



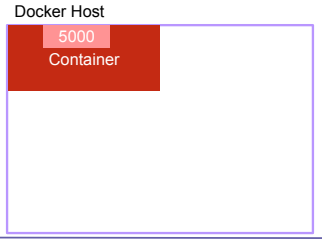
Network drivers

- When we create containers, it will automatically attach to the bridge driver.



Network drivers

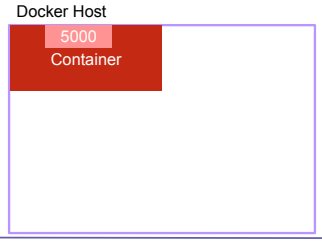
- **Host** removes network isolation between the docker host and docker containers. It uses the host's networking directly.
- Host networks are best when the network stack should not be isolated from the Docker host, but we want other aspects of the container to be isolated.



Network drivers

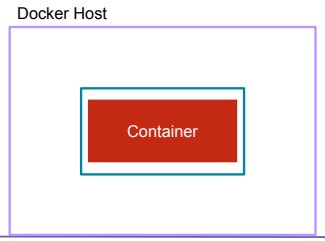
Host mode networking can be useful for the following use case:

- High-Performance Networking
  - Eliminate the network translation layer (no NAT), reducing latency.
  - Suitable for low-latency or high-throughput network applications.



Network drivers

- **None** network driver disables all networking of containers.
- **None** network driver will not configure any IP for the container and doesn't have any access to the external network as well as for other containers.
- It is used when a user wants to block the networking access to a container.



Network drivers

None Network Driver – Use Cases

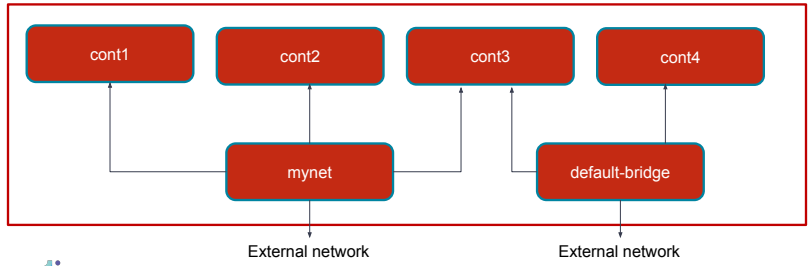
- Complete Network Isolation
  - Run containers with no network connectivity for security or containment.
- Offline Application Testing
  - Simulate how an application behaves when the network is unavailable.
- Security-Sensitive Workloads
  - Ensure sensitive tools or scripts cannot reach external networks.
- Manual Network Configuration
  - Use Linux tools (ip netns, veth, etc.) to create custom networking setups.
- Resource or Performance Testing
  - Measure CPU, memory, or disk performance without any network interference.

User-defined bridge networks

User-defined bridge networks

Users can also create their own networks called **user-defined networks** of any network driver type.

```
$ docker network create --driver bridge mynet
```



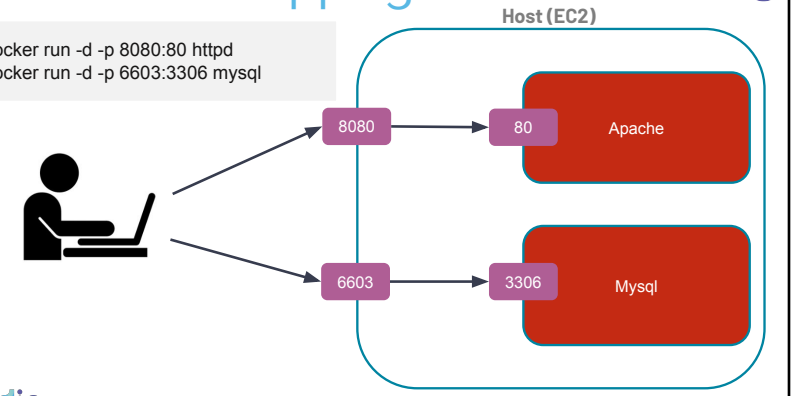
Run - Port mappings

By default, when you create a container, it does not publish any of its ports to the outside world. To make a port available to services outside of Docker, or to Docker containers which are not connected to the container's network, use the **--publish** or **-p** flag.

```
-p  host_port : container_port
```

Run - Port mappings

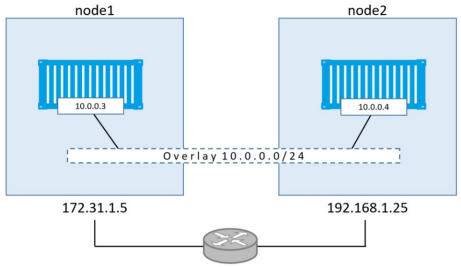
```
$ docker run -d -p 8080:80 httpd
$ docker run -d -p 6603:3306 mysql
```



Other Network drivers

Network drivers

- **Overlay** networks connect multiple Docker daemons together and enable swarm services to communicate with each other.



Network drivers

- **Macvlan** networks allow you to assign a MAC address to a container, making it appear as a physical device on your network.
- Using the macvlan driver is sometimes the best choice when dealing with legacy applications that expect to be directly connected to the physical network, rather than routed through the Docker host's network stack.

Network drivers

Macvlan Network Driver – Use Cases:

- Direct IP Assignment from Physical Network
  - Give a container its own IP address on the same subnet as the host.
- Bypass Docker Host Network Stack
  - Useful when avoiding NAT, firewall rules, or port conflicts on the host.
- Legacy or Network-Sensitive Applications
  - Run apps that require direct Layer 2 access or unique MAC addresses.
- Container-to-Physical Device Communication
  - Allow containers to communicate directly with devices on the local LAN (e.g., printers, IoT devices).
- Advanced Networking Labs or Simulations
  - Simulate physical machines or create networking test environments using containers.

Network drivers

- **Network plugins:** We can install and use third-party network plugins with Docker. These plugins are available from Docker Hub or from third-party vendors.
- Third-party network plugins allow us to integrate Docker with specialized network stacks.

Docker Network Commands

docker network Commands

Command	Description
docker network connect	Connect a container to a network
docker network create	Create a network
docker network disconnect	Disconnect a container from a network
docker network inspect	Display detailed information on one or more networks
docker network ls	List networks
docker network prune	Remove all unused networks
docker network rm	Remove one or more networks

ondia

THANKS!

Any questions?

Powered by Clarusway