Docker Networking

CLARUSWAY©
WAY TO REINVENT YOURSELF

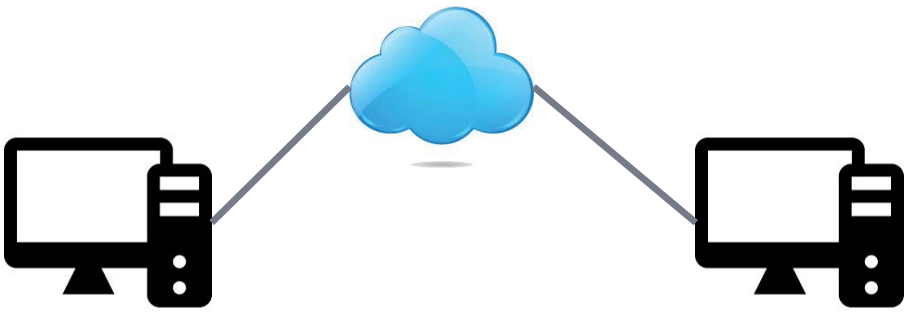
Table of Contents

CLARUSWAY©
WAY TO REINVENT YOURSELF

1 Networking overview

Networking overview

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

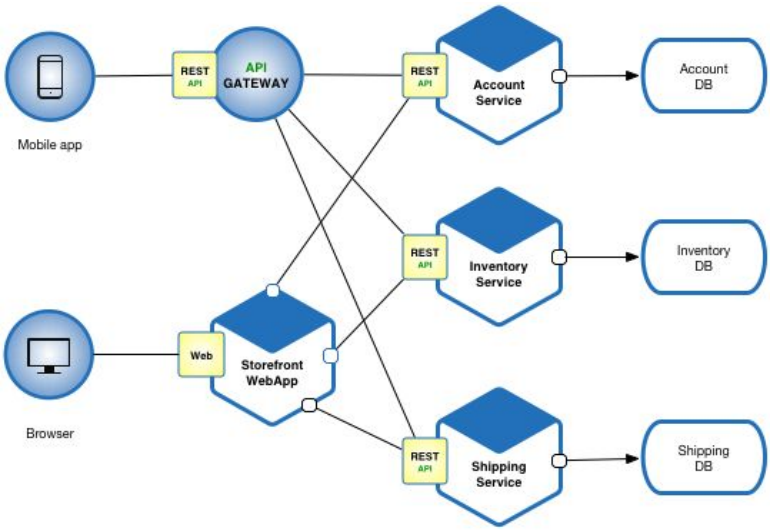


Networking overview

Network is usually involved in the outage in a production environment.

Art of networking is not easy, it is complex and takes a lot of experience and learning to master it.

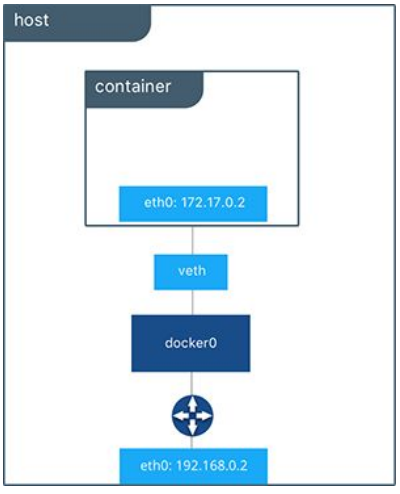
Correct configuration of network is crucial to establish an uninterrupted communication for microservice architecture.



Container Networking

Just like regular networking components, container networking involves container:

- private IP
- network interface
- routing table
- networking config file (resolv.conf)

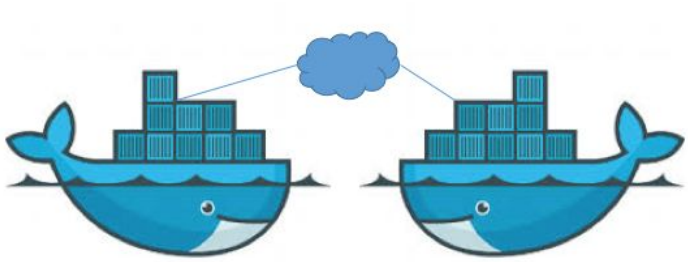


Tens or hundreds of containers running need a good IPAM and DNS.

Container networking is a namespace inside the container, which provides layer of isolation.

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

2 Network drivers

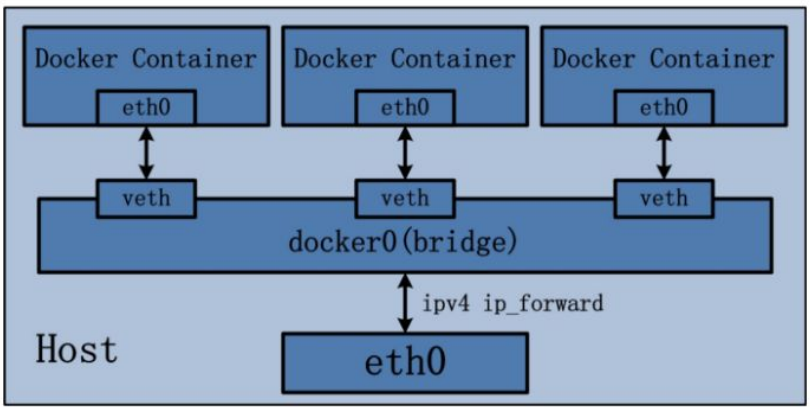
Network drivers

As default, docker has three network drivers.

- Bridge
- Host
- none

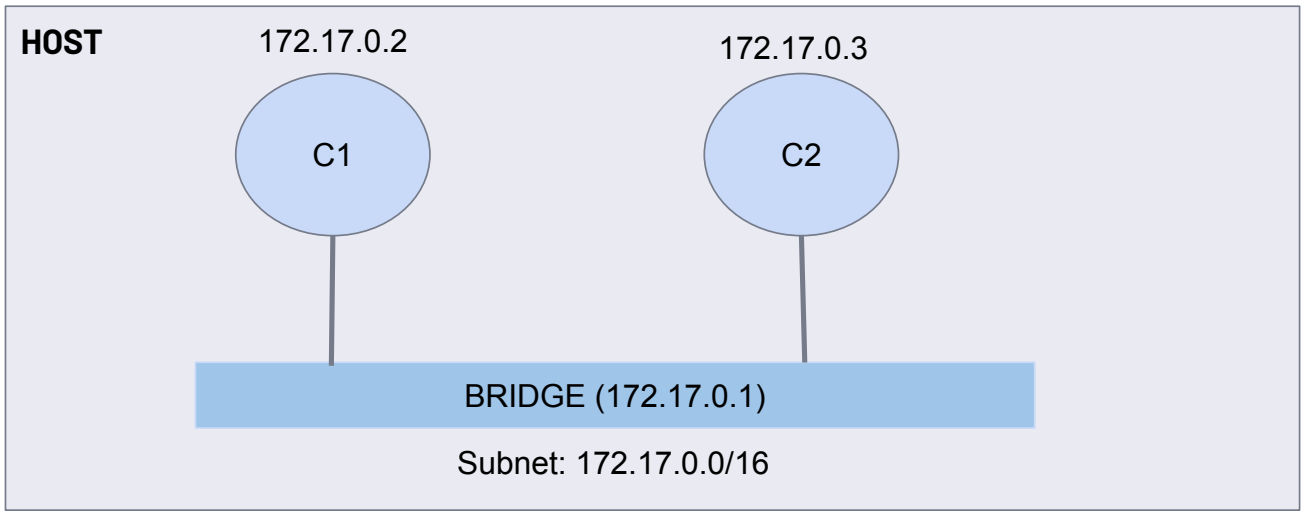
Network drivers

- **bridge** is the private 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.



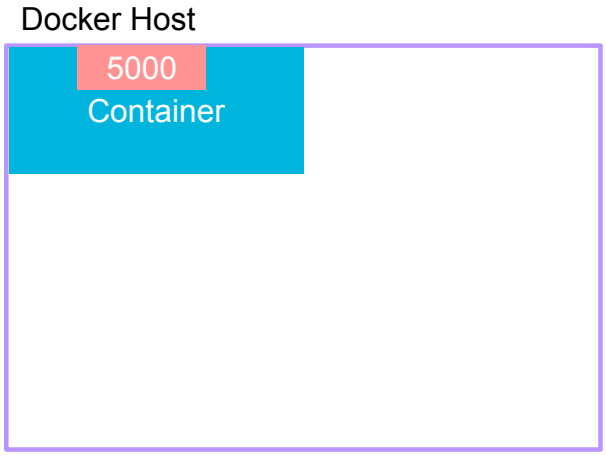
Network drivers

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



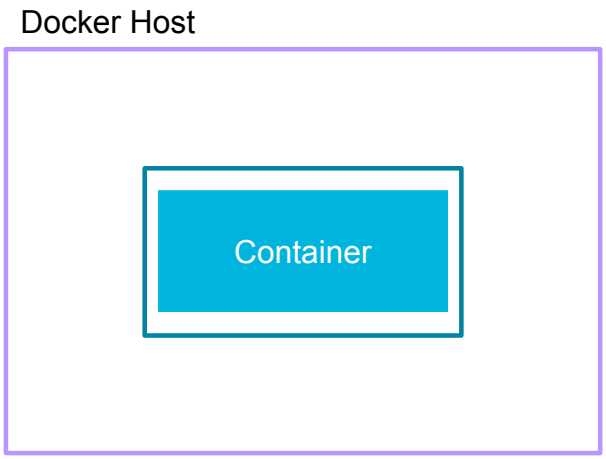
Network drivers

- **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

- **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.

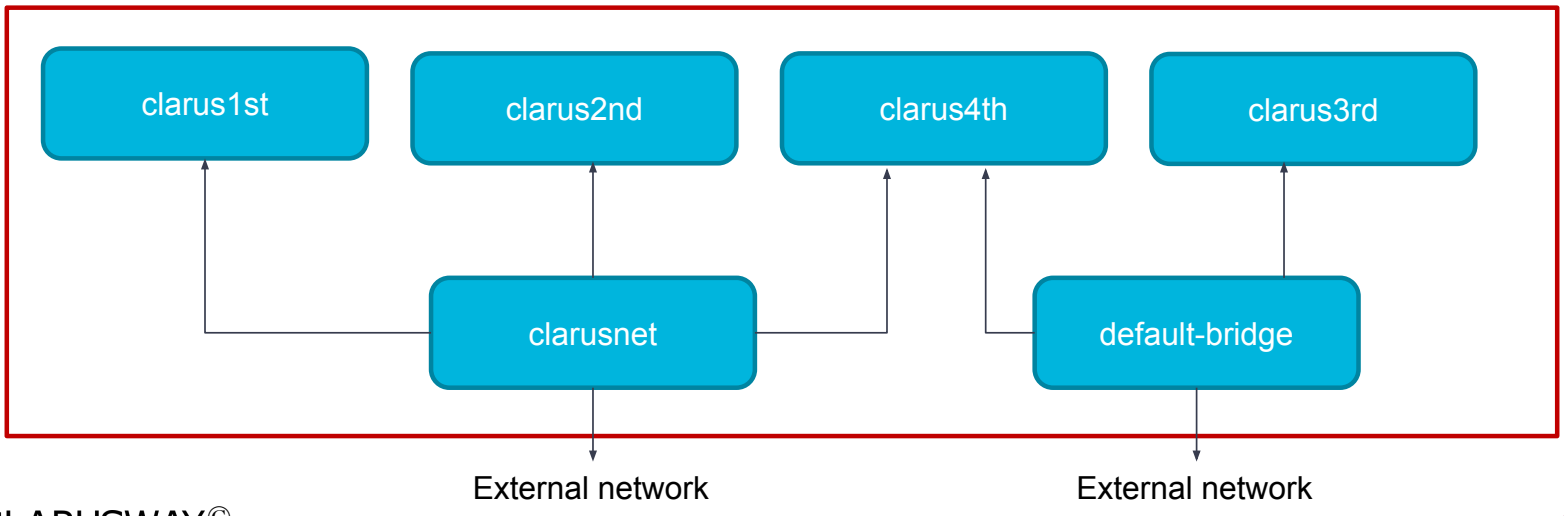


3 User-defined bridge networks

User-defined bridge networks

In addition to the default networks, users can create their own networks called **user-defined networks** of any network driver type.

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



4 Run - Port mappings

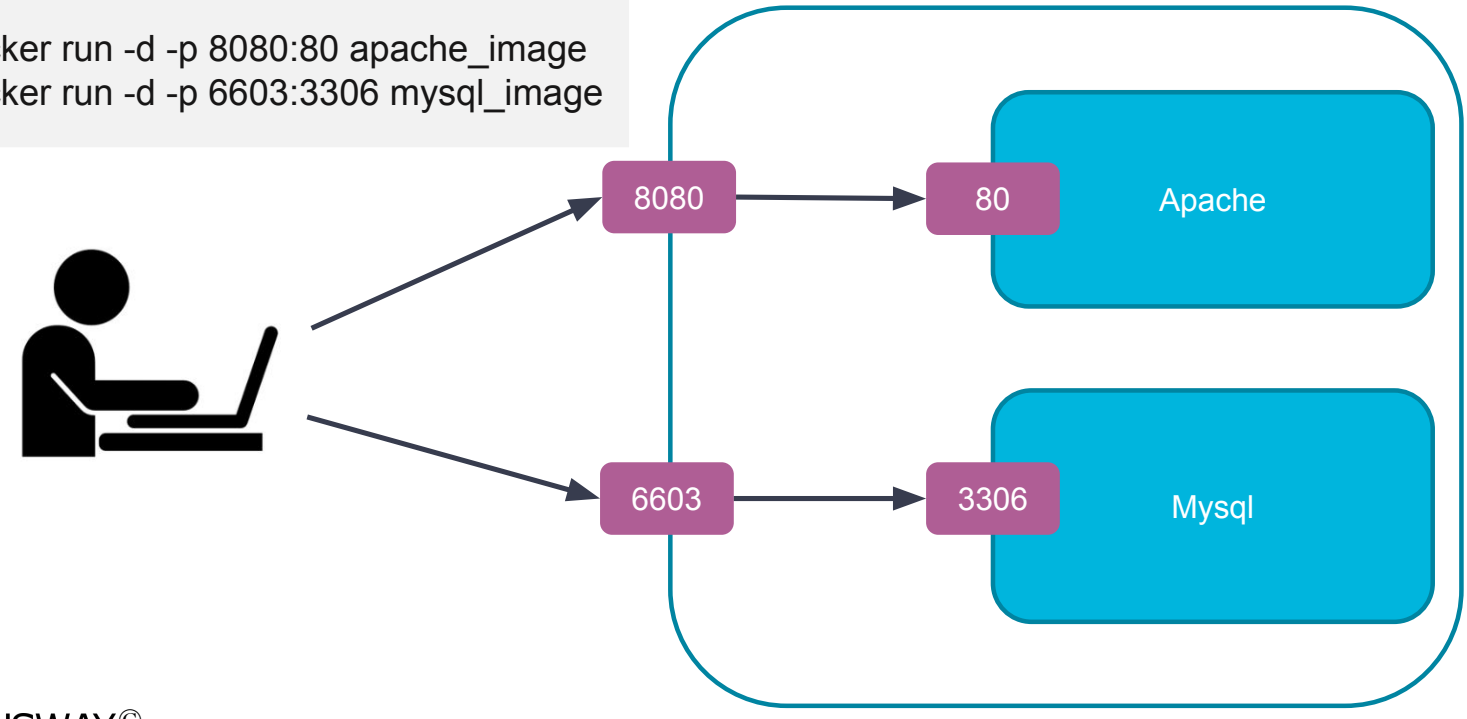
Run - Port mappings

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 apache_image
$ docker run -d -p 6603:3306 mysql_image
```

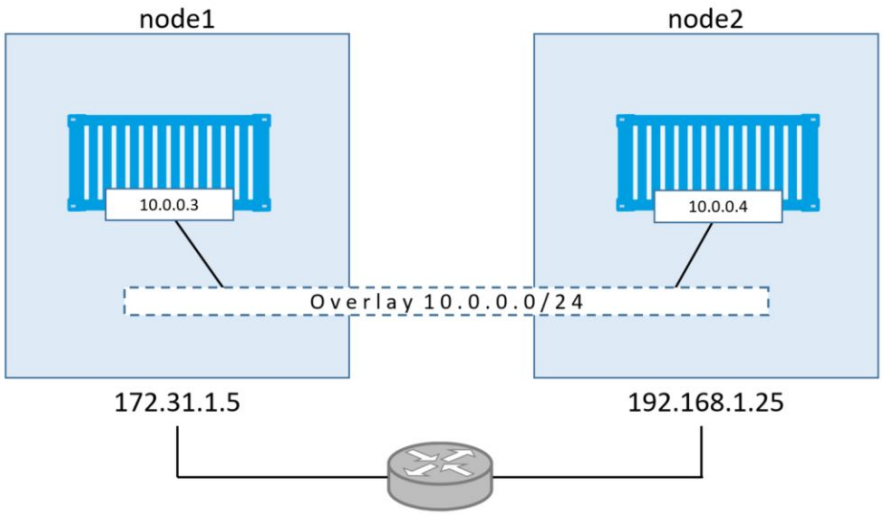


5 Other Network drivers

CLARUSWAY©
WAY TO REINVENT YOURSELF

Network drivers

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



CLARUSWAY©
WAY TO REINVENT YOURSELF

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

- **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.

6 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

THANKS!

Any questions?

You can find me at:

- tyler@clarusway.com

Network drivers

- **overlay:** Overlay networks connect multiple Docker daemons together and enable swarm services to communicate with each other. You can also use overlay networks to facilitate communication between a swarm service and a standalone container, or between two standalone containers on different Docker daemons. This strategy removes the need to do OS-level routing between these containers.

Network drivers

- **macvlan:** Macvlan networks allow you to assign a MAC address to a container, making it appear as a physical device on your network. The Docker daemon routes traffic to containers by their MAC addresses. 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 plugins:** You can install and use third-party network plugins with Docker. These plugins are available from Docker Hub or from third-party vendors.