**01. Network types supported by Docker**

Topics going to be discussed are: -

1. Overview on Bridge Network
2. Overview on Host Network
3. Overview on None Network
4. Overview on Overlay Network

Different types of networks supported by docker are: -

* Bridge
* Host
* None
* Overlay

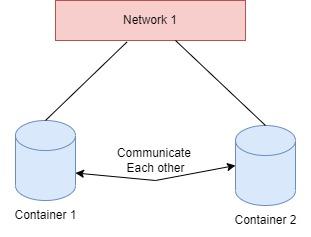
These networks are called as docker network drivers.

**Bridge Network**: -

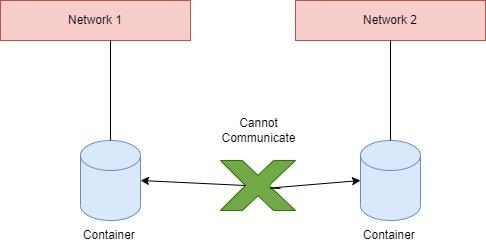
1. It is a default network driver for containers
2. Docker Engine supports default and user-defined network
3. It is a software bridge created by docker
4. Containers in same bridge can communicate with each other, where as other bridge network is blocked
5. Docker bridge network creates some rules (Iptables) on the host machine.

**Detail explanation of above points**: -

1. It is the default network driver used by docker for containers. Which means whenever we create a container without passing - -**network** parameter by default the docker engine will use the docker defined bridge network which is also known as default bridge network.
2. Docker engine supports default bridge network or we can also define our own bridge network with our own subnet which is known as user-defined bridge network.



1. Containers created in the same bridge network can communicate with each other, whereas containers created in different bridge networks cannot communicate with each other because Iptables firewall between these two networks will block the communication by default. Still, we can establish communication between containers created in different networks by manipulating the Iptables.



**Host Network**: -

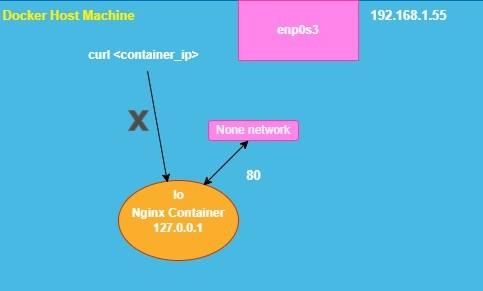
1. Host network driver will assign a container with host network as your Docker host machine uses.
2. We need to ensure the at application running on host and containers with host network driver do not use same port number
3. Host networks only works on Linux, not on Docker Desktop for Mac, Windows, or Docker EE on Window Server
4. It is not recommended to use host network driver for applications, only if it Dynamic ports

**Detail explanation of above points**: -

1. The host network driver will assign a container with host network Ip address of docker host machine. Which means whatever the network used by the docker host; the same network will be assigned to the container created with host network driver. Here we need to remember that only network stack of host machine is used by the container and the rest of them are still isolated.
2. While dealing with host network we need to ensure that if applications is running on both host machine and container then they both cannot use same port number because in a same network two process ids cannot listen on same port number.
3. Host network only works on Linux operating system and doesn’t work on Docker Desktop for Mac, Windows, or Docker EE on Windows Server as the host network feature is only available in Linux OS.
4. Even though we have the feature to assign host network to a container it is not recommended as we are allowing the container to manipulate the networks on the host which will affect all the host networks.
5. Host network for containers is only recommended when for an application the port number is not constant I.e., whenever we restart a container the port number for the application inside the container changes then instead of accessing the application by changing port number, we can provide host network to the container such that we can access the container every time with host network itself.

**None Network**: -

1. If containers should not be attached to any network stack in Docker, we have to use none
2. With this driver, containers will be able to connect to outside world as well
3. Application inside container, will be using loopback interface

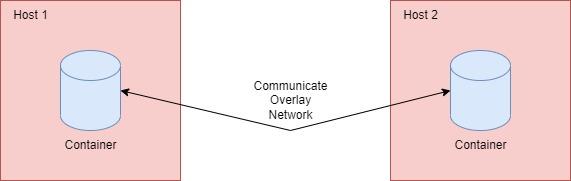


**Detail explanation of above points**: -

1. None network means if we want to create container without any network like bridge, host etc. These types of containers are called stand-alone containers, even the host network cannot communicate with these containers and the application inside the container will be using loop back interface which is 127.0.0.1 this IP address will be available for all the operating systems.
2. So, the docker host cannot communicate with container using container loop back interface because even the docker host machine will have its loop back interface. As the container wont able to communicate with the host machine, similarly the containers with this driver will not able to connect to outside world as well. If we want to run the application, we need to be inside the container.
3. The purpose of using none network for container is for development and testing application.

**Overlay Network**: -

1. Overlay network will ensure containers sitting on two different Docker Host machines can communicate with each other.
2. This feature is available by default on Docker Swarm.



**Detail explanation of above points**: -

Using overlay network containers sitting on two different docker host machine can communicate with each other. The overlay network feature is by default available on docker swarm but to implement overlay network without using docker swarm, we have to make use of Consul or etcd key-value storage.