**DOCKER NETWORK**

Docker networking enables a user to link a Docker container to as many networks.

Note: A user can add containers to more than one network.

**Network Drivers**

Docker supports networking for its containers via network drivers. These drivers have several network drivers.

* Bridge
* Host
* None
* Overlay
* Macvlan

**Bridge**

* It is a private default network created on the host
* Containers linked to this network have an internal IP address through which they communicate with each other easily

### **Host**

* It is a public network
* It utilizes the host’s IP address and TCP port space to display the services running inside the container
* It effectively disables network isolation between the docker host and the docker containers, which means using this network driver a user will be unable to run multiple containers on the same host

### **None**

* In this network driver, the Docker containers will neither have any access to external networks nor will it be able to communicate with other containers
* This option is used when a user wants to disable the networking access to a container

### **Overlay**

* It is an important network driver in Docker networking. It helps in providing the interaction between the stand-alone container and the Docker swarm service

### **Macvlan**

* It simplifies the communication process between containers
* This network assigns a MAC address to the Docker container. With this Mac address, the Docker server (daemon) routes the network traffic to a router
* Note: Docker Daemon is a server which interacts with the operating system and performs all kind of services
* It is suitable when a user wants to directly connect the container to the physical network rather than the Docker host

## Basic Docker Networking Commands

### **List down the Networks associated with Docker**

docker network ls

### **Connect a Running Container to a Network**

$ docker network connect multi-host-network container

In the command shown above, You can also use the docker network option to start a container and immediately connect it to multiple host networks.

### **Specify the IP Address that you want to assign to the Container**

$ docker network connect --IP 10.10.36.122 multi-host-network container

### **Disconnect a Container from a Network**

$ docker network disconnect multi-host-network container1

In the above command, the disconnect option is used to stop the running docker containers on multiple host network

### **Remove a Network**

$ docker network rm network\_name

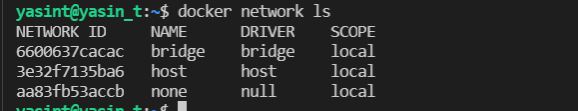
### **Remove all Unused Networks**

$ docker network prune

The above ‘prune’ command can be used when a user wants to remove all unused networks at a time

Hands-on

* Docker container create three default network driver.



* Container can connect to each other by bridge network driver by ID however they can connect each other by user defined bridge network driver.

docker network inspect bridge

* Run two `alpine` containers with interactive shell, in detached mode, name the container as `cont1st` and `cont2nd`, and add command to run alpine shell.

docker container run -dit --name cont1st alpine ash

docker container run -dit --name cont2nd alpine ash

* Get the IP of `clarus2nd` container. (172.17.0.3)

docker container inspect cont2nd | grep IPAddress

* Connect to the `clarus1st` container.

docker exec -it clarus1st sh

-Ping to google.com and ‘cont2nd’ container by its IP four times to show the connection.

ping -c 4 google.com

ping -c 4 172.17.0.3

```

- Try to ping `cont2nd `container by its name, should face with bad address.

ping -c 4 clarus2nd

**User-defined Network Bridge in Docker:**

* Create a bridge network `testnet`.

docker network create --driver bridge testnet

* List the networks

