In this article we will explore the networking options provided by docker on a single host setup.

Docker provides the basic infrastructure for networking containers in a single host. When installing docker creates a virtual ethernet bridge called "docker0". Docker uses the network that is not in conflict with the host network. When we start the docker service , it will first check whether we have given a custom bridge option to the docker daemon.if not docker will create its own bridge called "docker0"

If we run the ifconfig command on host , we can see

```
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 0.0.0.0
        inet6 fe80::42:fcff:fed3:c709  prefixlen 64  scopeid 0x20<link>
        ether 02:42:fc:d3:c7:09  txqueuelen 0  (Ethernet)
        RX packets 50  bytes 3368 (3.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16  bytes 1978 (1.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

While creating this , it will first try to find the network range that is not being used by the host.It used the ip route command to find the networks that host needs to reach and then finds a range that does not conflict with any.

As said a virtual ethernet bridge called "docker0" is created.The job of the bridge is to connect 2 different networks and forwards packets from one connected network to the other. The "docker0" does the same job to all the network interfaces that are connected to this.

Docker provides 3 default networks

```
[puppet@root$:~]$  docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
064b2749f867        bridge              bridge              local
2a81b1db3008        host                host                local
192aee8753a7        none                null                local
```

The Bridge,host and None.

Bridge - The bridge network is provided by default and this is the network to which a container created on this host connects. This maps the "docker0" bridge on the host.

Lets start a Containers and once started check the container ID using

```
[puppet@root$:~]$  docker ps --format="{{.ID}} {{.Names}}"
2031be9aeaa6 stoic_leavitt
```

Now check the bridge network interface using the inspect command as below,

```
[puppet@root$:~]$  docker network inspect bridge
[
        {
        "Name": "bridge",
        "Id": "064b2749f8678f4429df796d84ae7547a7ccf3e5b9d0b7ea17584195e5628c7f",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
        "Driver": "default",
        "Options": null,
        "Config": [
                {
                "Subnet": "172.17.0.0/16",
                "Gateway": "172.17.0.1"
                }
        ]
        },
        "Internal": false,
        "Containers": {
        "2031be9aeaa6b47031318ae7c62aaf226d72bd008333e005c7d936fafb85d032": {
                "Name": "stoic_leavitt",
                "EndpointID":
"a7423df61658290c9e8e9f51fa9f2f357561252b5381616f526e4492b20a2400",
                "MacAddress": "02:42:ac:11:00:02",
                "IPv4Address": "172.17.0.2/16",
                "IPv6Address": ""
        }
        },
        "Options": {
        "com.docker.network.bridge.default_bridge": "true",
        "com.docker.network.bridge.enable_icc": "true",
        "com.docker.network.bridge.enable_ip_masquerade": "true",
        "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
        "com.docker.network.bridge.name": "docker0",
        "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
        }
]
```

In the above output we can see that the container we started uses the bridge network by default. The entry com.docker.network.bridge.name": "docker0" shows that the bridge is mapped to docker0. The containers that we create will be added to this network by default and will be assigned an IP address from the subset of the IPs reserved for bridge. In the above case the container will be given with IP address from 172.17.0.2/16 range. If we login to the container and use the ifconfig command like below,

```
[puppet@root$:~]$  docker exec -it 2031be9aeaa6 ifconfig
eth0    Link encap:Ethernet  HWaddr 02:42:ac:11:00:02
        inet addr:172.17.0.2  Bcast:0.0.0.0  Mask:255.255.0.0
        inet6 addr: fe80::42:acff:fe11:2/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:8 errors:0 dropped:0 overruns:0 frame:0
        TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:648 (648.0 B)  TX bytes:648 (648.0 B)
```

We can see that the IP address is assigned based on the Host.

None - This is used to attach a container to a no network.Containers attached to this network will not have a IP address and will be stand-alone. The none network is useful to create containers that need to be standalone batch jobs without any dependency on other networks.

Start a Container with the None network as

```
[puppet@root$:~]$  docker run -dit --net=none ifconfig-ubuntu /bin/bash
ab10aef79d5b3b387ede1197ae7ce63813be7341cf9e1109f57a5813827da76b
```

Now if we run the ifconfig command inside the container we can see no IP address,

```
[puppet@root$:~]$  docker exec -it ab10aef79d5b ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Host - When a container connects to the Host network , it gets the same configuration of the host. We can start a container with the host network and then inspect for details.

Docker Network Commands - Here are the summary of the network commands provided by docker.

Docker network connect|disconnect <Network Name> <Container> : Connects or disconnects the containers to the network specified

Docker network inspect <network Name> : inspects the network and provides the details in the JSON format

Docker network ls : list all the available docker networks

Docker network rm <network Name> : remove the docker network

Docker network create [--driver <driver Name>] <network Name> : creates a network with the specified driver name taking the driver specified

The driver are generally bridge ,none and host. The driver is a network plugin that adapts the networking features of docker to the host networking support.

```
[puppet@root$:~]$  docker network create  --driver bridge sample-1
87af01944848c312d4bcf51564bf26a0830b40617074cf3922b9e74673c2cb29

[puppet@root$:~]$  docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
064b2749f867        bridge              bridge              local
2a81b1db3008        host                host                local
192aee8753a7        none                null                local
87af01944848        sample-1            bridge              local

[puppet@root$:~]$  docker run -dit ifconfig-ubuntu /bin/bash
b624040483f41c47cb36252dc943953ff461fdfe78790e48760f95158534d637
[puppet@root$:~]$  docker run -dit ifconfig-ubuntu /bin/bash
7f99932627942a352b529e8c98dd8d20a95b8a424298875cb1423e626b32d35e
[puppet@root$:~]$  docker ps --format="{{.ID}} {{.Names}}"
7f9993262794 furious_mccarthy
b624040483f4 gigantic_williams
[puppet@root$:~]$  docker network connect sample-1 furious_mccarthy
[puppet@root$:~]$  docker network connect sample-1 gigantic_williams
[puppet@root$:~]$  docker network inspect sample-1
[
        {
        "Name": "sample-1",
        "Id": "87af01944848c312d4bcf51564bf26a0830b40617074cf3922b9e74673c2cb29",
```

```
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
        "Driver": "default",
        "Options": {},
        "Config": [
                {
                "Subnet": "172.18.0.0/16",
                "Gateway": "172.18.0.1/16"
                }
        ]
        },
        "Internal": false,
        "Containers": {
        "7f99932627942a352b529e8c98dd8d20a95b8a424298875cb1423e626b32d35e": {
                "Name": "furious_mccarthy",
                "EndpointID":
"6c5d2810768493e605365bba36633d107a066ea9eba243d0832c92934cec40ea",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
        },
        "b624040483f41c47cb36252dc943953ff461fdfe78790e48760f95158534d637": {
                "Name": "gigantic_williams",
                "EndpointID":
"359b1c392f9345724f44ed07e3067b3f80ee51e269d8f8e172fe72c4c41a3e42",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
        }
        },
        }
]
[puppet@root$:~]$  docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
        PORTS               NAMES
7f9993262794        ifconfig-ubuntu     "/bin/bash"         About a minute ago  Up About a
minute              furious_mccarthy
b624040483f4        ifconfig-ubuntu     "/bin/bash"         About a minute ago  Up About a
minute              gigantic_williams
```

```
[puppet@root$:~]$  docker exec -it 7f9993262794 ping -c 4 b624040483f4
PING b624040483f4 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: icmp_seq=0 ttl=64 time=0.047 ms
64 bytes from 172.18.0.3: icmp_seq=1 ttl=64 time=0.056 ms
64 bytes from 172.18.0.3: icmp_seq=2 ttl=64 time=0.052 ms
64 bytes from 172.18.0.3: icmp_seq=3 ttl=64 time=0.049 ms
--- b624040483f4 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.047/0.051/0.056/0.000 ms
```

In the above example ,we have created a network called "sample-1" . we also created 2 containers and attached them to the sample-1. In order to check whether they are in same network , we pinged the second container from the first container.