

## Docker - Networking

Docker takes care of the networking aspects so that the containers can communicate with other containers and also with the Docker Host. If you do an **ifconfig** on the Docker Host, you will see the Docker Ethernet adapter. This adapter is created when Docker is installed on the Docker Host.

```
demo@ubuntudemo:~$ sudo ifconfig
docker0    Link encap:Ethernet  HWaddr 02:42:b4:a4:43:59
            inet addr:172.17.0.1  Bcast:0.0.0.0  Mask:255.255.0.0
            inet6 addr: fe80::42:b4ff:fea4:4359/64 Scope:Link
            UP BROADCAST MULTICAST  MTU:1500  Metric:1
            RX packets:55 errors:0 dropped:0 overruns:0 frame:0
            TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:3448 (3.4 KB)  TX bytes:2576 (2.5 KB)

eth0       Link encap:Ethernet  HWaddr 08:00:27:f5:15:76
            inet addr:192.168.137.200  Bcast:192.168.137.255  Mask:255.255.255.0
            inet6 addr: fe80::a00:27ff:fe5:1576/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:199 errors:0 dropped:0 overruns:0 frame:0
            TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:13734 (13.7 KB)  TX bytes:5238 (5.2 KB)

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:40 errors:0 dropped:0 overruns:0 frame:0
            TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:3184 (3.1 KB)  TX bytes:3184 (3.1 KB)

demo@ubuntudemo:~$
```

This is a bridge between the Docker Host and the Linux Host. Now let's look at some commands associated with networking in Docker.

### Listing All Docker Networks

This command can be used to list all the networks associated with Docker on the host.

### Syntax

```
docker network ls
```

### Options

None

## Return Value

The command will output all the networks on the Docker Host.

## Example

```
sudo docker network ls
```

## Output

The output of the above command is shown below

```
demo@ubuntudemo:~$ sudo docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
f07aad6ccadf        bridge             bridge              local
faae6bf679ea        host               host                local
54a2d37e7e00        none              null                local
demo@ubuntudemo:~$
```

## Inspecting a Docker network

If you want to see more details on the network associated with Docker, you can use the Docker **network inspect** command.

## Syntax

```
docker network inspect networkname
```

## Options

- **networkname** – This is the name of the network you need to inspect.

## Return Value

The command will output all the details about the network.

## Example

```
sudo docker network inspect bridge
```

## Output

The output of the above command is shown below –

```
"Name": "bridge",
"Id": "f07aad6ccadf388082ccf9ad37db43f78adec85fb96ae0b2e9e8390c6d674242"
,
"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": {},
"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": {}
}
demo@ubuntudemo:~$
```

Now let's run a container and see what happens when we inspect the network again. Let's spin up an Ubuntu container with the following command –

```
sudo docker run -it ubuntu:latest /bin/bash
```

```
demo@ubuntudemo:~$ sudo docker run -it ubuntu:latest /bin/bash
```

Now if we inspect our network name via the following command, you will now see that the container is attached to the bridge.

```
sudo docker network inspect bridge
```

```

        {
            "Subnet": "172.17.0.0/16",
            "Gateway": "172.17.0.1"
        }
    ],
    "Internal": false,
    "Containers": {
        "8e7b9a6dc121ba1c9a9fe48542db0149ee87b5efe031f518fb15751741ea0447": {
            "Name": "suspicious_blackwell",
            "EndpointID": "d30971d663e91ec2439355bb43c99613d500e35fbbae1957e7f74cb650f40723",
            "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": {}
}
lemo@ubuntudemo:~$

```

## Creating Your Own New Network

One can create a network in Docker before launching containers. This can be done with the following command –

### Syntax

```
docker network create --driver drivername name
```

### Options

- **drivername** – This is the name used for the network driver.
- **name** – This is the name given to the network.

### Return Value

The command will output the long ID for the new network.

### Example

```
sudo docker network create --driver bridge new_nw
```

## Output

The output of the above command is shown below –

```
demo@ubuntudemo:~$ sudo docker network create --driver bridge new_nw
f01b64dc09425cc4906e20b5e17765e3248ea727068e0e2172bfc4aec42586fe
demo@ubuntudemo:~$ _
```

You can now attach the new network when launching the container. So let's spin up an Ubuntu container with the following command –

```
sudo docker run -it --network=new_nw ubuntu:latest /bin/bash
```

```
demo@ubuntudemo:~$ sudo docker run -it --network=new_nw ubuntu:latest /bin/bash
```

And now when you inspect the network via the following command, you will see the container attached to the network.

```
sudo docker network inspect new_nw
```

```

    "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": {
        "38604fc42bcb5f78d42a8f40f34fa245301b2020a84c9e602786d2103ca6b847": {
            "Name": "boring_dubinsky",
            "EndpointID": "74d6b14a6393bf3081d5d9ec012b5b76b2ead49e85a5f664c
a621761a9e69612",
            "MacAddress": "02:42:ac:12:00:02",
            "IPv4Address": "172.18.0.2/16",
            "IPv6Address": ""
        }
    },
    "Options": {},
    "Labels": {}
}
]
leno@ubuntudemo:~$

```