Lab Exercise 4- Working with Docker Networking

Name – Sujal Bhandari

SAP_ID – 500106865

Roll_No – R2142220181

Step 1: Understanding Docker Default Networks

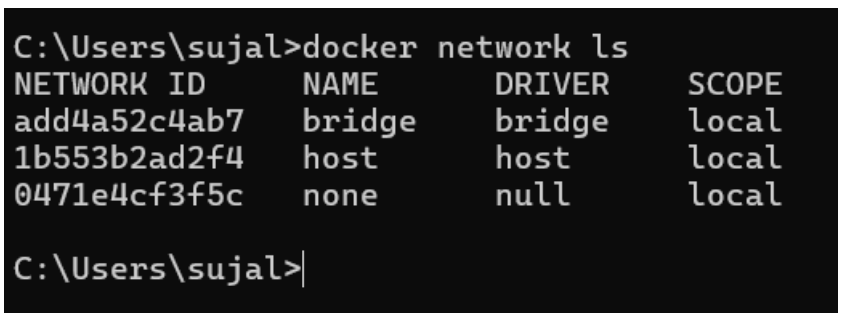
Docker provides three default networks:

- bridge: The default network when a container starts.
- host: Bypasses Docker's network isolation and attaches the container directly to the host network.
- none: No networking is available for the container.

1.1. Inspect Default Networks

Check Docker's default networks using:

```
docker network ls
```



1.2. Inspect the Bridge Network

```
docker network inspect bridge
```

```
C:\Users\sujal>docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "add4a52c4ab7eaeadde2572a7b3802925b2668a14ea6702be2fc956aaf3fa328",
        "Created": "2024-09-13T05:28:18.575790175Z",
        "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,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": 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": {}
    }
]

C:\Users\sujal>
```

This command will show detailed information about the bridge network, including the connected containers and IP address ranges.

Step 2: Create and Use a Bridge Network

2.1. Create a User-Defined Bridge Network

A user-defined bridge network allows containers to communicate by name instead of IP.

```
docker network create my_bridge
```

```
C:\Users\sujal>docker network create my_bridge
d10eb7e47e684807e2cbd6c34d2c8ee110ab2a4293d926c14fafbdc8e4b6f40b

C:\Users\sujal>
```

2.2. Run Containers on the User-Defined Network

Start two containers on the newly created my_bridge network:

```
docker run -dit --name container1 --network my_bridge busybox
```

```
C:\Users\sujal>docker run -dit --name container1 --network my_bridge busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
2fce1e0cdfc5: Pull complete
Digest: sha256:c230832bd3b0be59a6c47ed64294f9ce71e91b327957920b6929a0caa8353140
Status: Downloaded newer image for busybox:latest
8e3e1f7db6d861790bcdac4ab6c9daa46bc40239cfb9aca6ebcb7a92bd5ab224

C:\Users\sujal>
```

```
docker run -dit --name container2 --network my_bridge busybox
```

```
C:\Users\sujal>docker run -dit --name container2 --network my_bridge busybox
85f08e3df3d871d8a48904a283b2c37281ae9ea0e03863ee3eb903bd24cd0121

C:\Users\sujal>
```

2.3. Test Container Communication

Execute a ping command from container1 to container2 using container names:

```
docker exec -it container1 ping container2
```

```
C:\Users\sujal>docker exec -it container1 ping container2
PING container2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.095 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.060 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.070 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.060 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.103 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.061 ms
^C
--- container2 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.060/0.074/0.103 ms

C:\Users\sujal>
```

The containers should be able to communicate since they are on the same network.

Step 3: Disconnect and Remove Networks

3.1. Disconnect Containers from Networks

To disconnect container1 from my_bridge:

```
docker network disconnect my_bridge container1
```

```
C:\Users\sujal>docker network disconnect my_bridge container1

C:\Users\sujal>
```

4.2. Remove Networks

To remove the user-defined network:

```
docker network rm my_bridge
```

```
C:\Users\sujal>docker network rm my_bridge
my_bridge

C:\Users\sujal>
```

Step 4: Clean Up

Stop and remove all containers created during this exercise:

docker rm -f container1 container2

```
C:\Users\sujal>docker rm -f container1 container2
container1
container2

C:\Users\sujal>
```