Lab Exercise 4- Working with Docker Networking

Step 1: Understanding Docker Default Networks

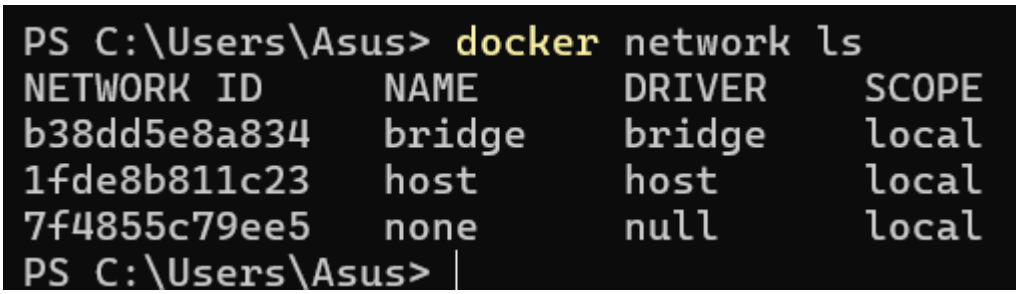
Docker provides three default networks:

- 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
```

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

```
PS C:\Users\Asus> docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "b38dd5e8a834442f38835c180e7c2ce91aeb89716a4a4d961860ec71a7b14488",
        "Created": "2024-11-10T20:17:03.690111554Z",
        "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": {}
    }
]
PS C:\Users\Asus>
```

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
```

```
PS C:\Users\Asus> docker network create my_bridge
12afae7c071c5cafe8188910dc0fd16d70ade132e3ab55ba5cc211e28ff38a6e
PS C:\Users\Asus>
```

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

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

```
PS C:\Users\Asus> docker run -dit --name container1 --network my_bridge busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
a46fbb00284b: Pull complete
Digest: sha256:768e5c6f5cb6db0794eec98dc7a967f40631746c32232b78a3105fb946f3ab83
Status: Downloaded newer image for busybox:latest
c907e9cedd9ae983d96e85d34951584cc9f3f4436e4cf42f2688ec7fb64008ba
```

```
PS C:\Users\Asus> docker run -dit --name container2 --network my_bridge busybox
bec3f81c4c80a049c17b6d5e4a06bd819d59c30cf91d881223c18ac0a38b22f2
PS C:\Users\Asus>
```

2.3. Test Container Communication

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

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

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

```
PS C:\Users\Asus> 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=334.609 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.067 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.159 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.075 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.166 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.180 ms
64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.078 ms
64 bytes from 172.18.0.3: seq=7 ttl=64 time=0.249 ms
64 bytes from 172.18.0.3: seq=8 ttl=64 time=0.131 ms
64 bytes from 172.18.0.3: seq=9 ttl=64 time=0.072 ms
64 bytes from 172.18.0.3: seq=10 ttl=64 time=0.098 ms
64 bytes from 172.18.0.3: seq=11 ttl=64 time=0.134 ms
64 bytes from 172.18.0.3: seq=12 ttl=64 time=0.094 ms
64 bytes from 172.18.0.3: seq=13 ttl=64 time=0.068 ms
```

Step 3: Disconnect and Remove Networks

3.1. Disconnect Containers from Networks

To disconnect container1 from my_bridge:

```
docker network disconnect my_bridge container1
```

```
PS C:\Users\Asus> docker network disconnect my_bridge container1
PS C:\Users\Asus>
```

4.2. Remove Networks

To remove the user-defined network:

```
docker network rm my_bridge
```

```
PS C:\Users\Asus> docker network rm my_bridge
my_bridge
PS C:\Users\Asus>
```

Step 4: Clean Up

Stop and remove all containers created during this exercise:

```
docker rm -f container1 container2
```