Lab Exercise 4- Working with Docker Networking

Name : Raman Boora

SapID: 500109408

Roll no: R2142221160

Step 1: Understanding Docker Default Networks

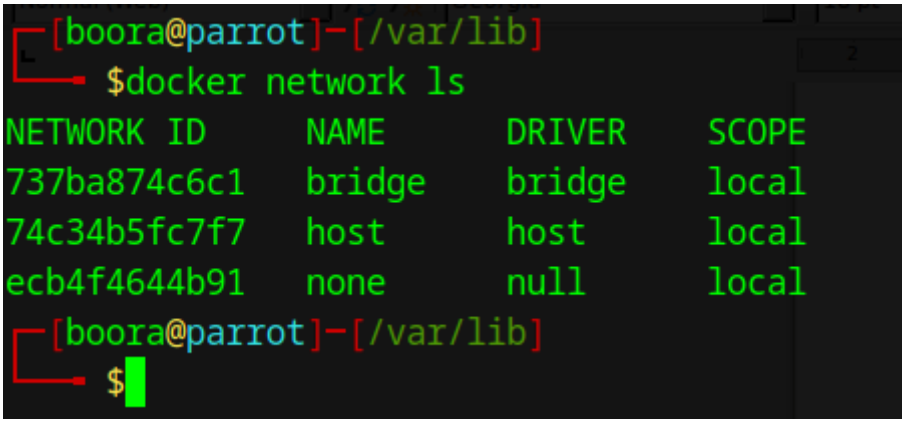
Docker provides three default networks:

- 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.

```
[boora@parrot]-[/var/lib]
└─$docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "737ba874c6c11e0ddec04f0d00809ffbf1d59acd2be925a0995824b54310f59f",
        "Created": "2024-11-01T13:53:43.09893837+05:30",
        "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"
```

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
```

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
```

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.

```
┌─[boora@parrot]─[/var/lib]
└──╼ $docker exec -it cont1 sh
/ # ping cont2
PING cont2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.108 ms
64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.186 ms
64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.126 ms
64 bytes from 172.18.0.2: seq=3 ttl=64 time=0.141 ms
64 bytes from 172.18.0.2: seq=4 ttl=64 time=0.179 ms
64 bytes from 172.18.0.2: seq=5 ttl=64 time=0.185 ms
^C
--- cont2 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.108/0.154/0.186 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
```

```
┌─[boora@parrot]─[/var/lib]
└──╼ $docker network disconnect --force net1 cont2
┌─[boora@parrot]─[/var/lib]
└──╼ $
```

4.2. Remove Networks

To remove the user-defined network:

```
docker network rm my_bridge
```

```
┌─[boora@parrot]─[/var/lib]
└──╼ $docker network disconnect --force net1 cont2
┌─[boora@parrot]─[/var/lib]
└──╼ $docker network remove net1
Error response from daemon: error while removing network: network net1 id 5f30055986e234d68604166d3f29bf474e87d61c83c6789cac0977fe321ff635 has active endpoints
┌─[✗]─[boora@parrot]─[/var/lib]
└──╼ $docker network disconnect --force net1 cont2
Error response from daemon: container 2bcada97c0d0795ee26cff690ba04f908d485812c2e01837aba8c384b5e2075a is not connected to network net1
┌─[✗]─[boora@parrot]─[/var/lib]
└──╼ $docker network disconnect --force net1 cont1
┌─[boora@parrot]─[/var/lib]
└──╼ $docker network remove net1
net1
┌─[boora@parrot]─[/var/lib]
└──╼ $
```

Step 4: Clean Up

Stop and remove all containers created during this exercise:

```
┌─[boora@parrot]─[/var/lib]
└──╼ $docker rm -f cont1 cont2
cont1
cont2
┌─[boora@parrot]─[/var/lib]
└──╼ $
```