DOCKER LAB

NAME: ARYANBANSAL

ROLLNO: R2142220237

SAPID:500101700

SPECIALIZATION: BTECH DEVOPS B1(NON-HONS.).

SUBMITTED TO:PROF.HITESH KUMAR SHARMA

Lab Exercise 4- Working with Docker Networking

Step 1: Understanding Docker Default Networks

Docker provides three default networks:

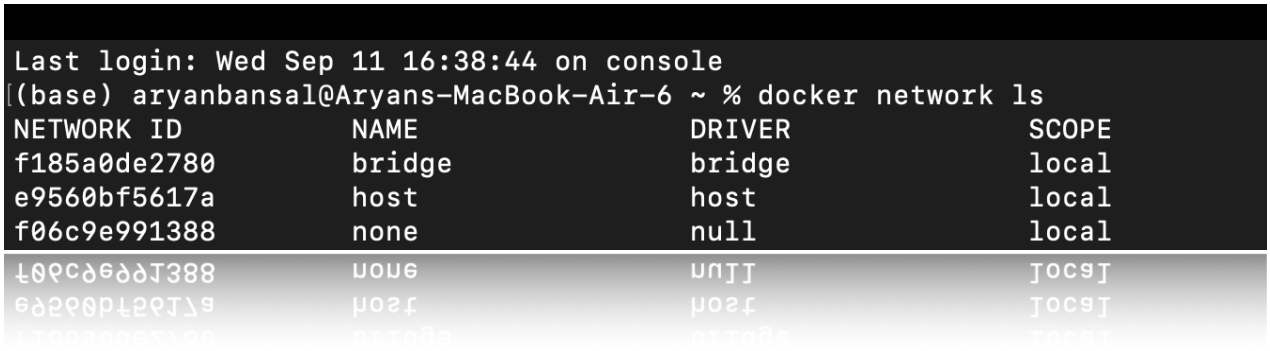
- bridge: The default network when a container starts.
- host: Bypasses Docker's network isolation and attaches the container directly to the host network.
- none: No networking is available for the container.

1.1. Inspect Default Networks

Check Docker's default networks using:

```
docker network ls
```

1.2. Inspect the Bridge Network




```
docker network inspect bridge
```

```
(base) aryanbansal@Aryans-MacBook-Air-6 ~ % docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "f185a0de27806853c3ba92ede7a9eb1fad5236e85bbc81da6cbaab4c74eba9ca",
        "Created": "2024-09-13T05:27:14.344343542Z",
        "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": "65535"
        },
        "Labels": {}
    }
]
(base) aryanbansal@Aryans-MacBook-Air-6 ~ %
```

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
```

```
(base) aryanbansal@Aryans-MacBook-Air-6 ~ % docker network create my_bridgearyan
749955a2b93e35cd62bc9cbd24958a61679bb50f4817c5c3e393f4d520b59535
```

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
```

```
(base) aryanbansal@Aryans-MacBook-Air-6 ~ % docker run -dit --name container1 --network my_bridgearyan busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
835f85a6d665: Pull complete
Digest: sha256:c230832bd3b0be59a6c47ed64294f9ce71e91b327957920b6929a0caa8353140
Status: Downloaded newer image for busybox:latest
fa0c6ff951c4983e5762d4aa621c42bde6cac95a92c4aa4f8670e8ca4e102fdd
(base) aryanbansal@Aryans-MacBook-Air-6 ~ %
```

```
(base) aryanbansal@Aryans-MacBook-Air-6 ~ % docker run -dit --name container2 --network my_bridgearyan busybox
b60bb0dd56b3efaa96086bf618f22248b9e06187feec4c34970a3d82501b5bfd
```

2.3. Test Container Communication

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

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

```
(base) aryanbansal@Aryans-MacBook-Air-7 ~ % 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=0.369 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.244 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.233 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.450 ms
^C64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.387 ms

--- container2 ping statistics ---
6 packets transmitted, 5 packets received, 16% packet loss
round-trip min/avg/max = 0.233/0.336/0.450 ms
(base) aryanbansal@Aryans-MacBook-Air-7 ~ %
```

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

Step 4: Disconnect and Remove Networks

4.1. Disconnect Containers from Networks

To disconnect container1 from my_bridge:

```
docker network disconnect my_bridge container1
```

```
(base) aryanbansal@Aryans-MacBook-Air-7 ~ % docker network disconnect my_bridgearyan container1
```

4.2. Remove Networks

To remove the user-defined network:

```
docker network rm my_bridge
```

```
(base) aryanbansal@Aryans-MacBook-Air-7 ~ % docker network rm my_bridgearyan
Error response from daemon: error while removing network: network my_bridgearyan id 749955a2b93e35cd62bc9cbd24958a61679bb50f4817c5c3
e393f4d520b59535 has active endpoints
```

Step 5: Clean Up

Stop and remove all containers created during this exercise:

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

```
(base) aryanbansal@Aryans-MacBook-Air-7 ~ % docker rm -f container1 container2 host_network_container
container1
container2
host_network_container
(base) aryanbansal@Aryans-MacBook-Air-7 ~ %
```