NAME: PARTHIB HALDER

SAP ID: 500126084

BATCH: DEVOPS B2

Lab Exercise 3- 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

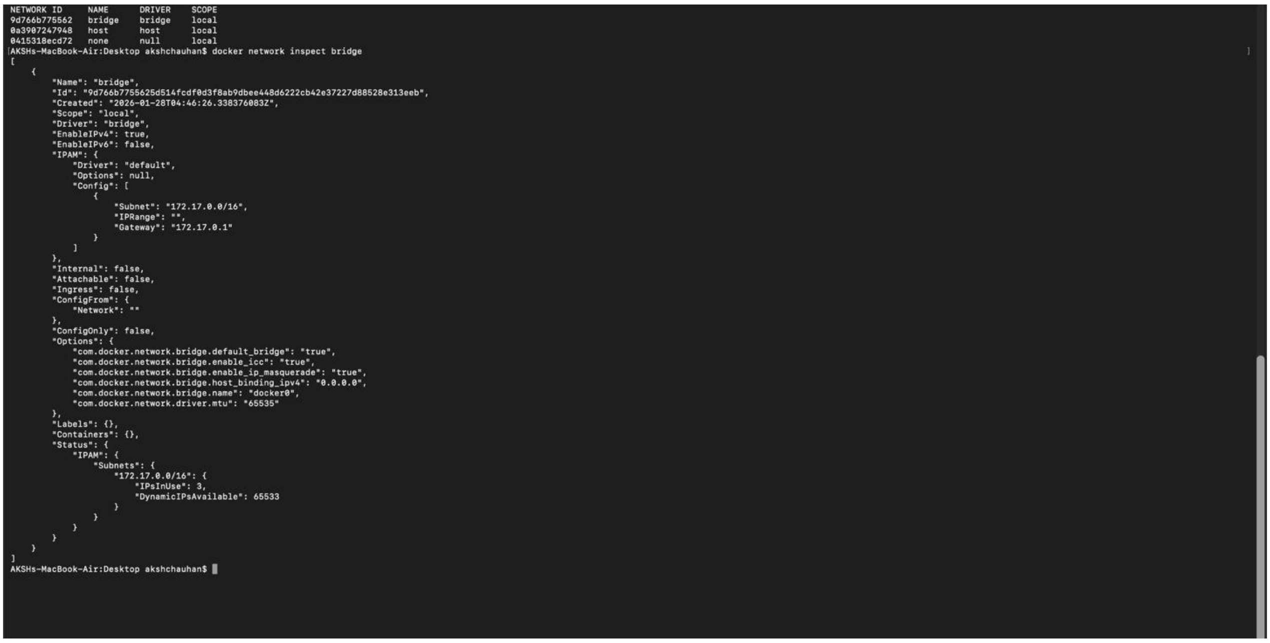
Check Docker's default networks using:

```
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.



```
NETWORK ID     NAME      DRIVER    SCOPE
9d766b775562   bridge    bridge    local
0a3907247948   host      host      local
0415318ecd72   none      null      local
AKSHs-MacBook-Air:Desktop akshchauhan$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "9d766b7755625d514fcdf0d3f8ab9dbee448d6222cb42e37227d88528e313eeb",
        "Created": "2026-01-28T04:46:26.338376083Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "IPRange": "",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "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": {},
        "Containers": {},
        "Status": {
            "IPAM": {
                "Subnets": {
                    "172.17.0.0/16": {
                        "IPsInUse": 3,
                        "DynamicIPsAvailable": 65533
                    }
                }
            }
        }
    }
]
AKSHs-MacBook-Air:Desktop akshchauhan$ 
```

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:

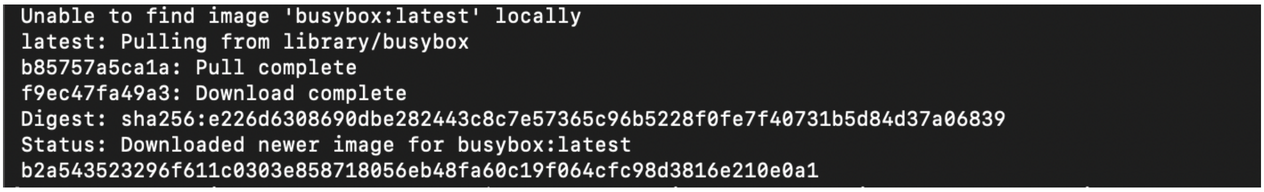
```
docker run -dit --name container1  --network my_bridge busybox

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

2.3. Test Container Communication

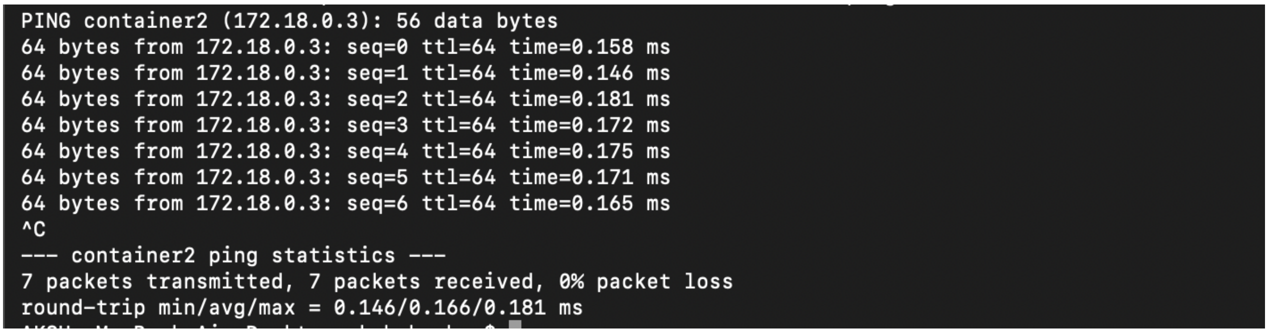
Execute a ping command from container1 to container2 using container names:

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

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



```
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
b85757a5ca1a: Pull complete
f9ec47fa49a3: Download complete
Digest: sha256:e226d6308690dbe282443c8c7e57365c96b5228f0fe7f40731b5d84d37a06839
Status: Downloaded newer image for busybox:latest
b2a543523296f611c0303e858718056eb48fa60c19f064cfc98d3816e210e0a1
```



```
PING container2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.158 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.146 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.181 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.172 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.175 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.171 ms
64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.165 ms
^C
--- container2 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.146/0.166/0.181 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
```

3.2. Remove Networks

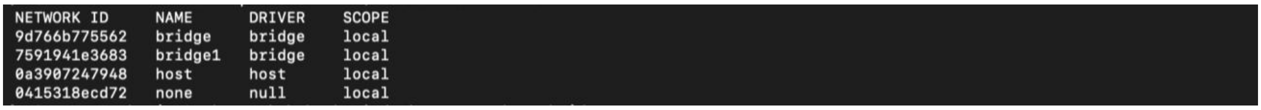
To remove the user-defined network:

```
docker network rm my_bridge
```

Step 4: Clean Up

Stop and remove all containers created during this exercise:

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



```
NETWORK ID     NAME      DRIVER    SCOPE
9d766b775562   bridge    bridge    local
7591941e3683   bridge1   bridge    local
0a3907247948   host      host      local
0415318ecd72   none      null      local
```