Name-Aayush Mishra

Sap Id-500107141

Roll no- R2142220385

Batch- B1 Devops (NH)

Experiment: 4

Working with Docker Networking

Step 1: Understanding Docker Default Networks

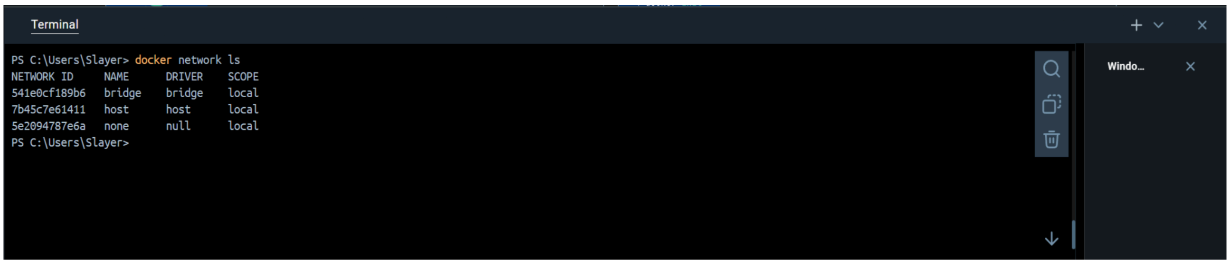
Docker provides three default networks:

- 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:

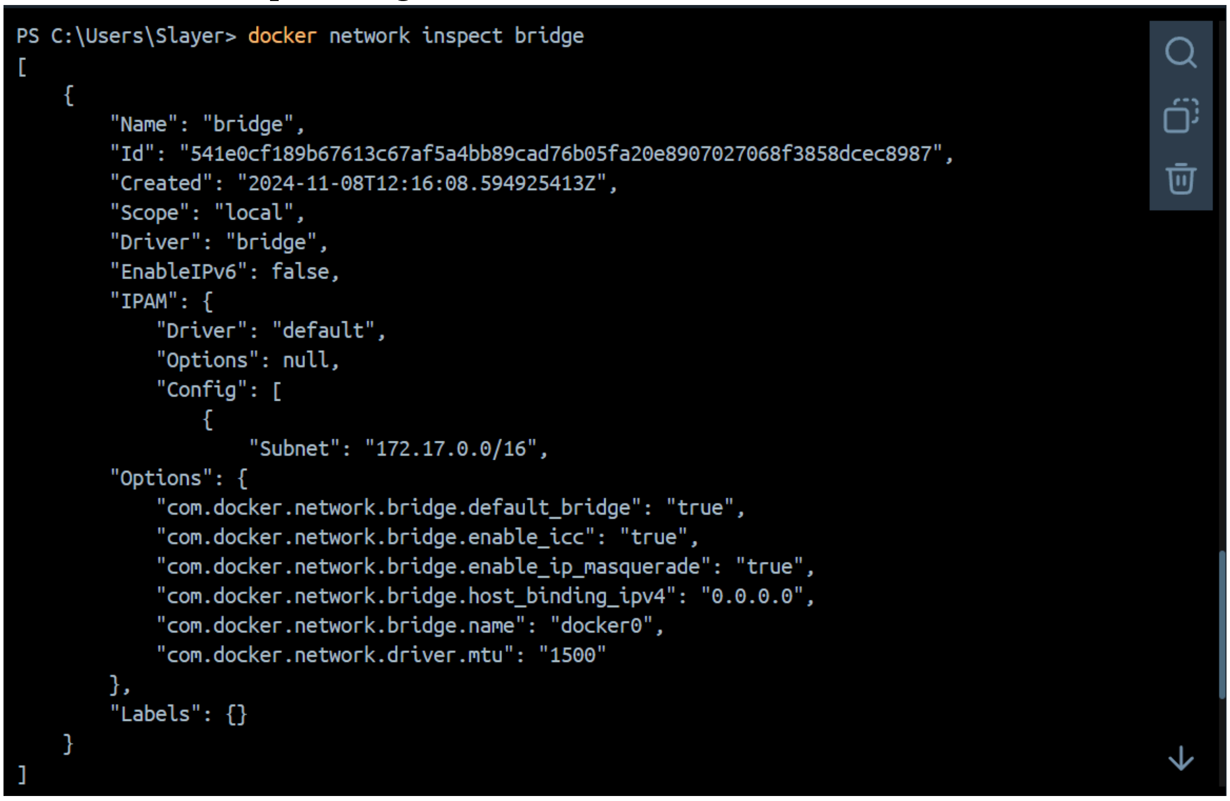
docker network ls



```
PS C:\Users\Slayer> docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
541e0cf189b6   bridge    bridge    local
7b45c7e61411   host      host      local
5e2094787e6a   none      null      local
PS C:\Users\Slayer>
```

1.2. Inspect the Bridge Network

docker network inspect bridge



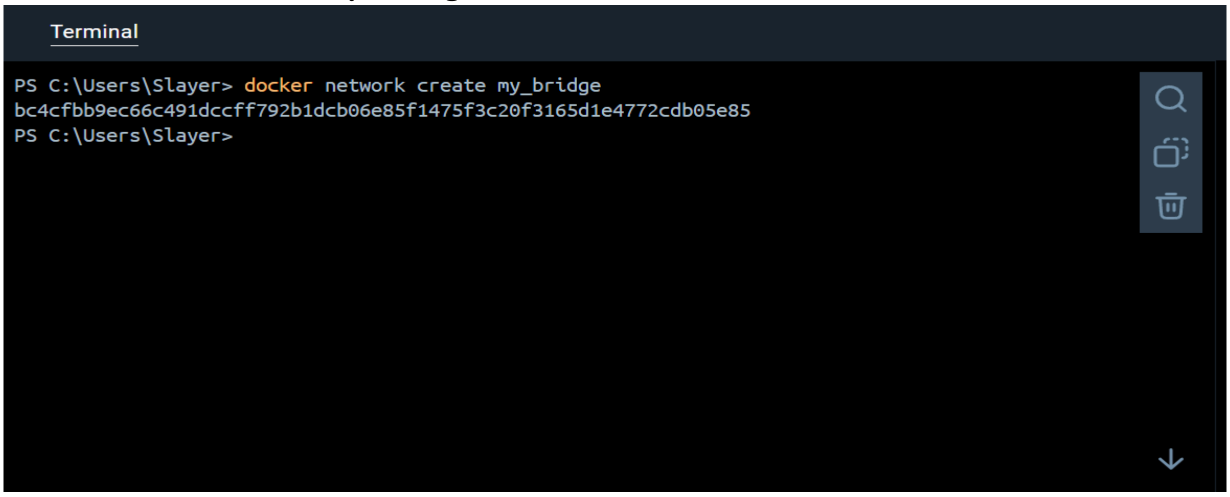
```
PS C:\Users\Slayer> docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "541e0cf189b67613c67af5a4bb89cad76b05fa20e8907027068f3858dcec8987",
        "Created": "2024-11-08T12:16:08.594925413Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
        "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": {}
    }
]
```

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

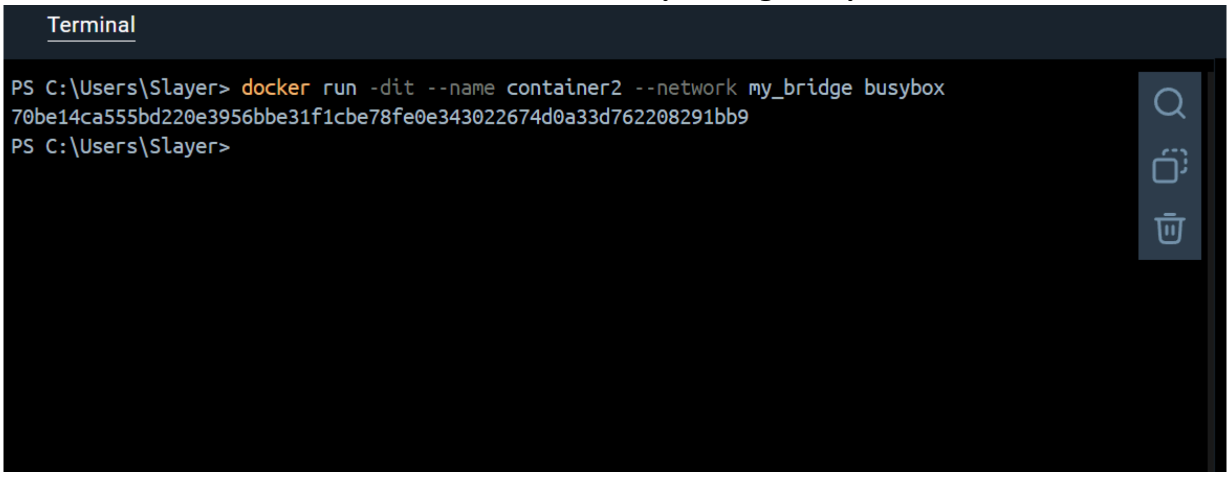


```
PS C:\Users\Slayer> docker network create my_bridge
bc4cfbb9ec66c491dccff792b1dcb06e85f1475f3c20f3165d1e4772cdb05e85
PS C:\Users\Slayer>
```

2.2. Run Containers on the User-Defined Network Start

two containers on the newly created my_bridge network:

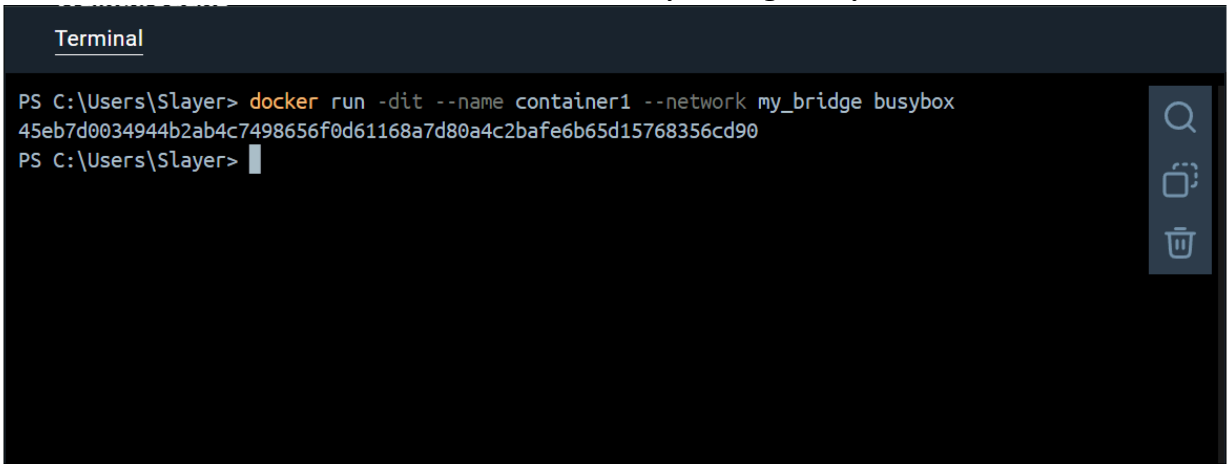
docker run -dit --name container1 --network my_bridge busybox



Terminal

```
PS C:\Users\Slayer> docker run -dit --name container2 --network my_bridge busybox
70be14ca555bd220e3956bbe31f1cbe78fe0e343022674d0a33d762208291bb9
PS C:\Users\Slayer>
```

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



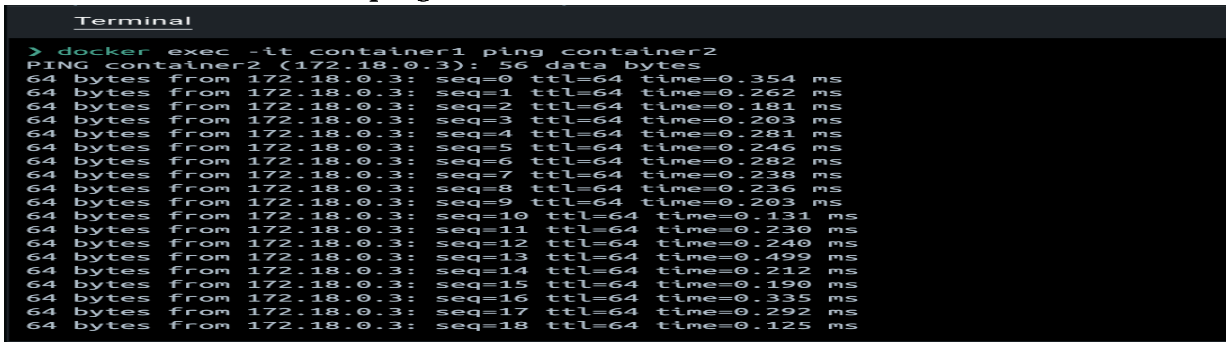
Terminal

```
PS C:\Users\Slayer> docker run -dit --name container1 --network my_bridge busybox
45eb7d0034944b2ab4c7498656f0d61168a7d80a4c2bafe6b65d15768356cd90
PS C:\Users\Slayer>
```

2.3. Test Container Communication

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

docker exec -it container1 ping container2



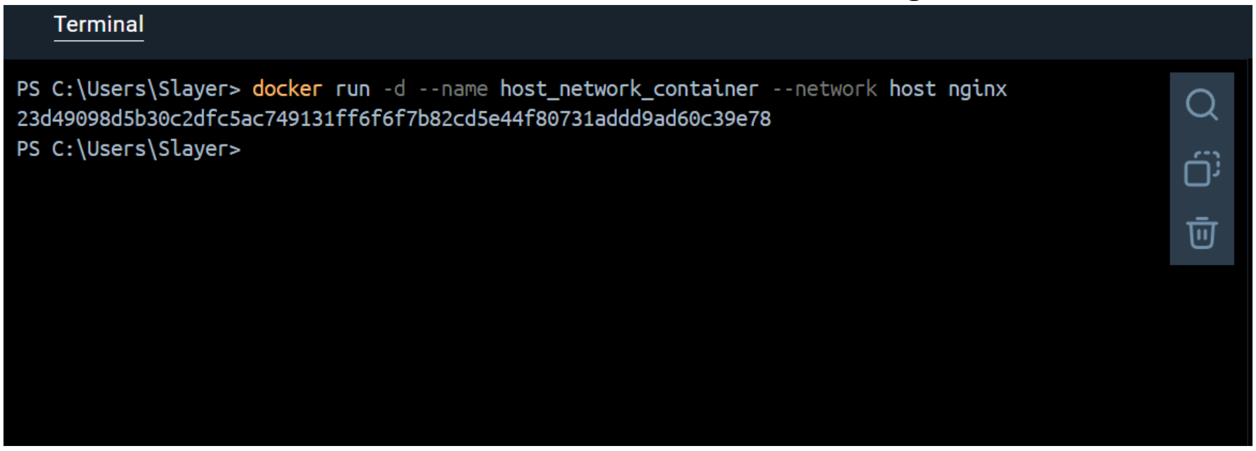
Terminal

```
> 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=0.354 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.262 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=0.203 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.281 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.246 ms
64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.282 ms
64 bytes from 172.18.0.3: seq=7 ttl=64 time=0.238 ms
64 bytes from 172.18.0.3: seq=8 ttl=64 time=0.236 ms
64 bytes from 172.18.0.3: seq=9 ttl=64 time=0.203 ms
64 bytes from 172.18.0.3: seq=10 ttl=64 time=0.131 ms
64 bytes from 172.18.0.3: seq=11 ttl=64 time=0.230 ms
64 bytes from 172.18.0.3: seq=12 ttl=64 time=0.240 ms
64 bytes from 172.18.0.3: seq=13 ttl=64 time=0.499 ms
64 bytes from 172.18.0.3: seq=14 ttl=64 time=0.212 ms
64 bytes from 172.18.0.3: seq=15 ttl=64 time=0.190 ms
64 bytes from 172.18.0.3: seq=16 ttl=64 time=0.335 ms
64 bytes from 172.18.0.3: seq=17 ttl=64 time=0.292 ms
64 bytes from 172.18.0.3: seq=18 ttl=64 time=0.125 ms
```

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

Step 3: Create and Use a Host Network 3.1. Run a Container Using the Host Network
The host network allows the container to use the host machine's networking stack:

docker run -d --name host_network_container --network host nginx



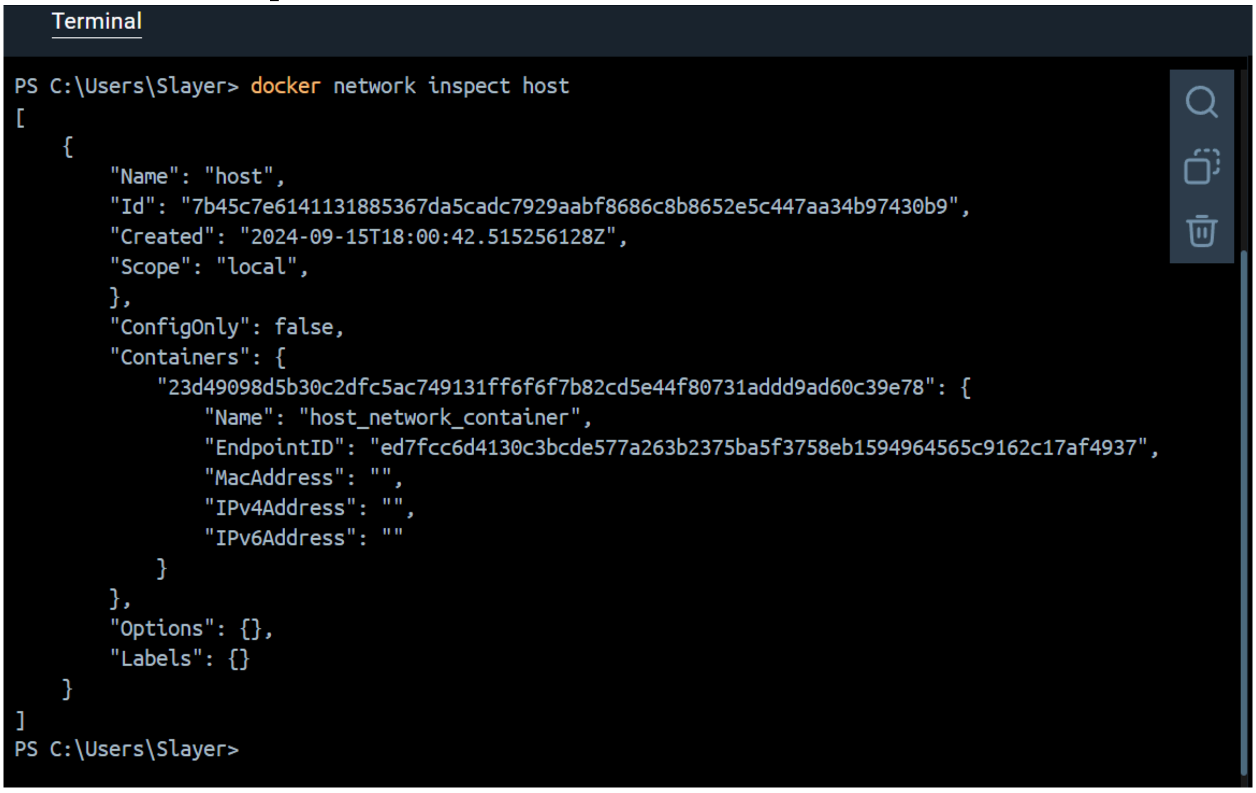
```
PS C:\Users\Slayer> docker run -d --name host_network_container --network host nginx
23d49098d5b30c2dfc5ac749131ff6f6f7b82cd5e44f80731addd9ad60c39e78
PS C:\Users\Slayer>
```

Access the NGINX server via localhost:80 in your browser to verify the container is

using the host network.

3.2. Check Network

docker network inspect host



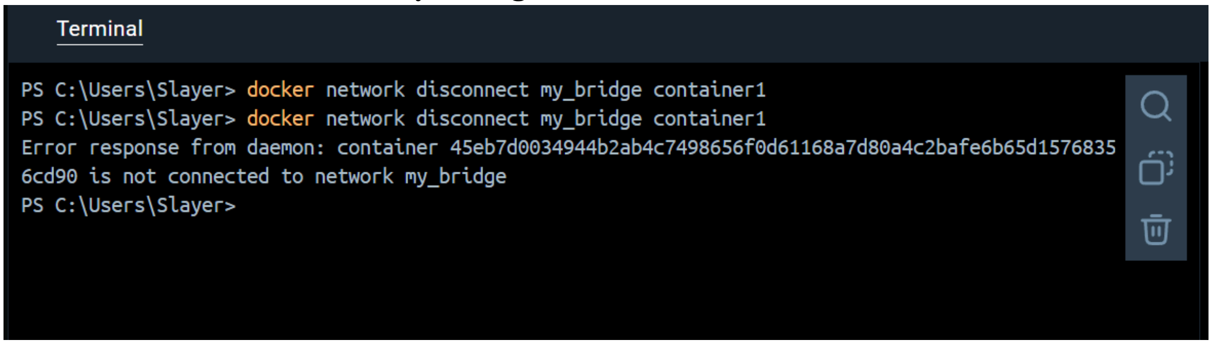
```
PS C:\Users\Slayer> docker network inspect host
[
    {
        "Name": "host",
        "Id": "7b45c7e6141131885367da5cadc7929aabf8686c8b8652e5c447aa34b97430b9",
        "Created": "2024-09-15T18:00:42.515256128Z",
        "Scope": "local",
        },
        "ConfigOnly": false,
        "Containers": {
            "23d49098d5b30c2dfc5ac749131ff6f6f7b82cd5e44f80731addd9ad60c39e78": {
                "Name": "host_network_container",
                "EndpointID": "ed7fcc6d4130c3bcde577a263b2375ba5f3758eb1594964565c9162c17af4937",
                "MacAddress": "",
                "IPv4Address": "",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
PS C:\Users\Slayer>
```

Step 4: Disconnect and Remove Networks

4.1. Disconnect Containers from Networks

To disconnect container1 from my_bridge:

docker network disconnect my_bridge container1

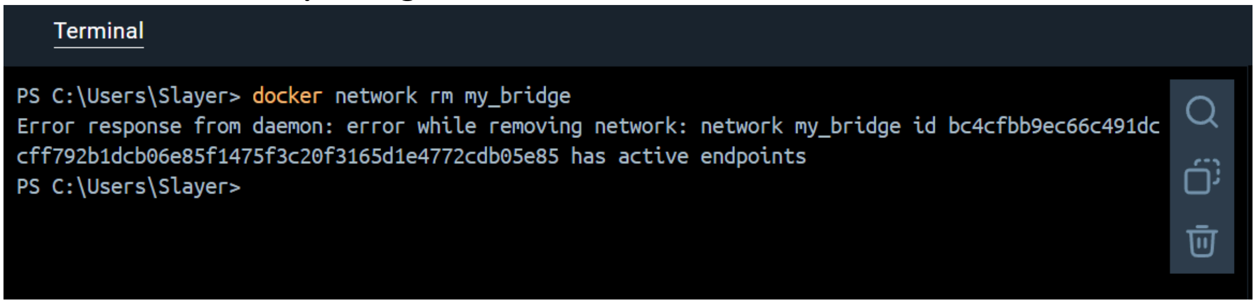


```
PS C:\Users\Slayer> docker network disconnect my_bridge container1
PS C:\Users\Slayer> docker network disconnect my_bridge container1
Error response from daemon: container 45eb7d0034944b2ab4c7498656f0d61168a7d80a4c2bafe6b65d1576835
6cd90 is not connected to network my_bridge
PS C:\Users\Slayer>
```

4.2. Remove Networks

To remove the user-defined network:

docker network rm my_bridge

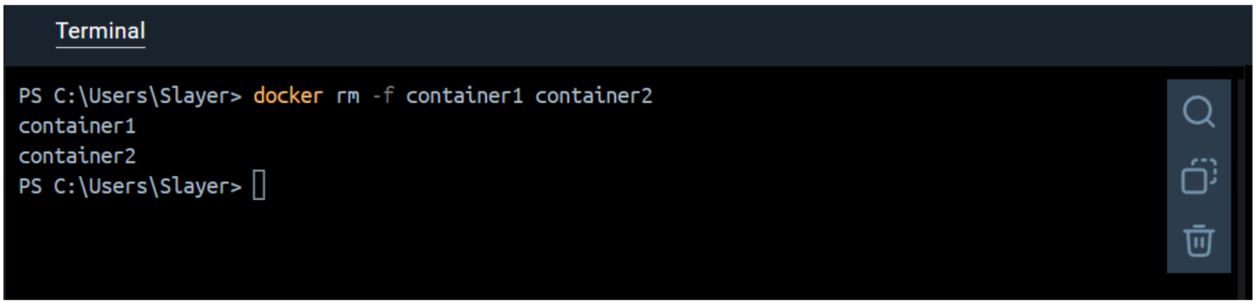


```
PS C:\Users\Slayer> docker network rm my_bridge
Error response from daemon: error while removing network: network my_bridge id bc4cfbb9ec66c491dc
cff792b1dcb06e85f1475f3c20f3165d1e4772cdb05e85 has active endpoints
PS C:\Users\Slayer>
```

Step 4: Clean Up

Stop and remove all containers created during this exercise:

docker rm -f container1 container2



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