

# 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

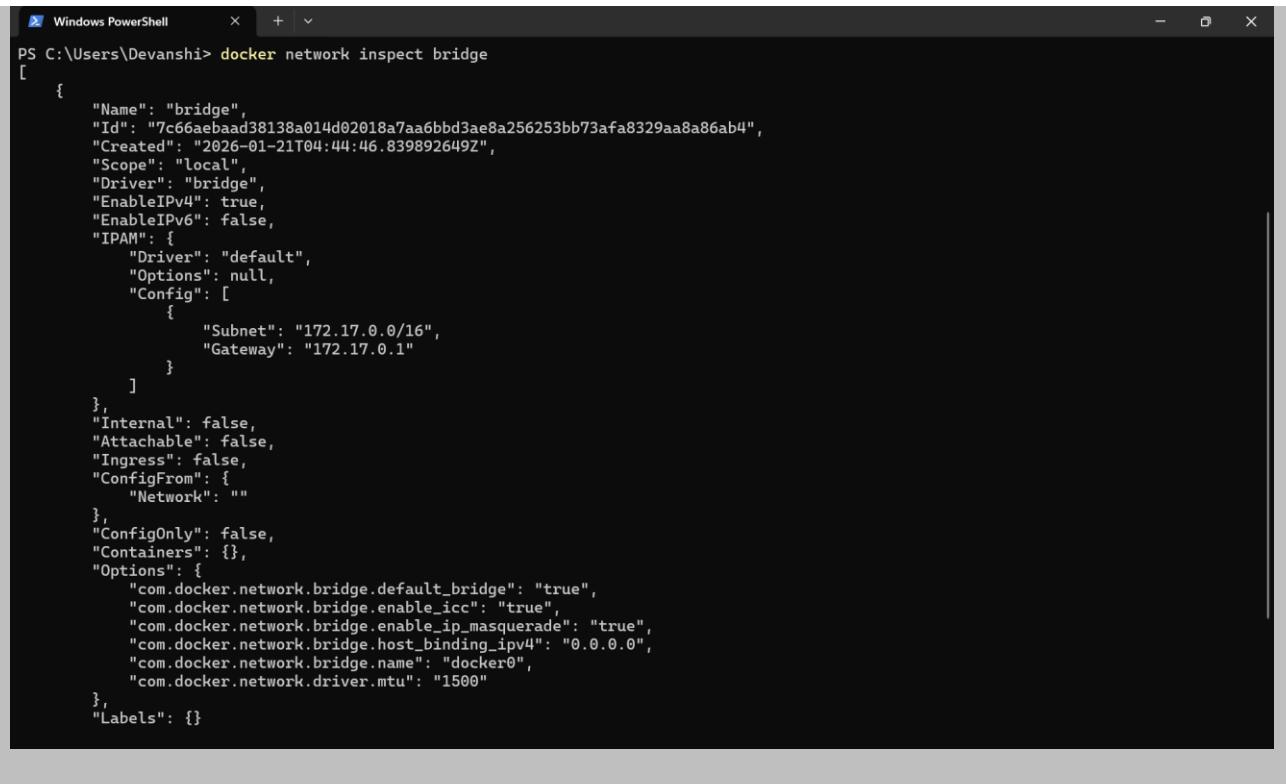
Check Docker's default networks using:

```
docker network ls
```

```
PS C:\Users\Devanshi> docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
7c66aebaad38  bridge    bridge      local
b3b3c46ff9a5   host      host       local
ca0f98de4dae  network1  bridge      local
8f0659be3044  none      null       local
```

### 1.2. Inspect the Bridge Network

```
docker network inspect bridge
```



```
PS C:\Users\Devanshi> docker network inspect bridge
[{"Name": "bridge", "Id": "7c6aeaa38138a014d02018a7aa6bb3ae8a256253bb73afa8329aa8a86ab4", "Created": "2026-01-21T04:44:46.839892649Z", "Scope": "local", "Driver": "bridge", "EnableIPv4": true, "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": {}}
```

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\Devanshi> docker network create djnetwork
406d63616bb3bb18c8745bf74535612affa1545e05120a8b5940217255bb5cfcc
```

### 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
```

```
PS C:\Users\Devanshi> docker run -dit --name container1 --network my_bridge busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
e59838ecfec5: Pull complete
Digest: sha256:2383baad1860bbe9d8a7a843775048fd07d8afe292b94bd876df64a69aae7cb1
Status: Downloaded newer image for busybox:latest
ed5c5eff9169a1039a552ee39ac01e5113921f0b8fe6b0a6b26e4d4d352441dc
```

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

```
PS C:\Users\Devanshi> docker run -dit --name container2 --network my_bridge busybox
448a447e291430a3e6628161c0768ffa83ac7f9ca5503778b5560bdbdcc2bb71
```

### 2.3. Test Container Communication

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

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

```
PS C:\Users\Devanshi> docker exec -it container1 ping container2
PING container2 (172.19.0.3): 56 data bytes
64 bytes from 172.19.0.3: seq=0 ttl=64 time=0.332 ms
64 bytes from 172.19.0.3: seq=1 ttl=64 time=0.121 ms
64 bytes from 172.19.0.3: seq=2 ttl=64 time=0.298 ms
64 bytes from 172.19.0.3: seq=3 ttl=64 time=0.171 ms
64 bytes from 172.19.0.3: seq=4 ttl=64 time=0.159 ms
64 bytes from 172.19.0.3: seq=5 ttl=64 time=0.196 ms
64 bytes from 172.19.0.3: seq=6 ttl=64 time=0.128 ms
64 bytes from 172.19.0.3: seq=7 ttl=64 time=0.098 ms
64 bytes from 172.19.0.3: seq=8 ttl=64 time=0.155 ms
```

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

## 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\Devanshi> docker network disconnect djnetwork container1
PS C:\Users\Devanshi>

{
    "Network": "",
},
"ConfigOnly": false,
"Containers": {
    "448a447e291430a3e6628161c0768ffa83ac7f9ca5503778b5560bdbdcc2bb71": {
        "Name": "container2",
        "EndpointID": "4a6e7f629764a127d92c5c0a9dca3ca887deb7bd14506c101957a5a651d8cf34",
        "MacAddress": "a2:8b:6e:e3:84:ca",
        "IPv4Address": "172.19.0.3/16",
        "IPv6Address": ""
    }
},
"Options": {
    "com.docker.network.enable_ipv4": "true",
    "com.docker.network.enable_ipv6": "false"
},
```

### 3.2. Remove Networks

To remove the user-defined network:

```
docker network rm my_bridge
```

```
PS C:\Users\Devanshi> docker stop container2
container2
PS C:\Users\Devanshi> docker network rm djnetwork
djnetwork
```

### Step 4: Clean Up

Stop and remove all containers created during this exercise:

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

```
PS C:\Users\Devanshi> docker rm -f container1 container2
container1
container2
PS C:\Users\Devanshi> docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
```