

# Lab Exercise 3- Working with Docker

## Networking

**Name:- Vansh Bhatt**

**Sap ID:- 500125395**

**Batch:- DevOps B1**

**To:- Hitesh Sharma Sir**

### **Step 1: Understanding Docker Default Networks**

Docker provides three default networks:

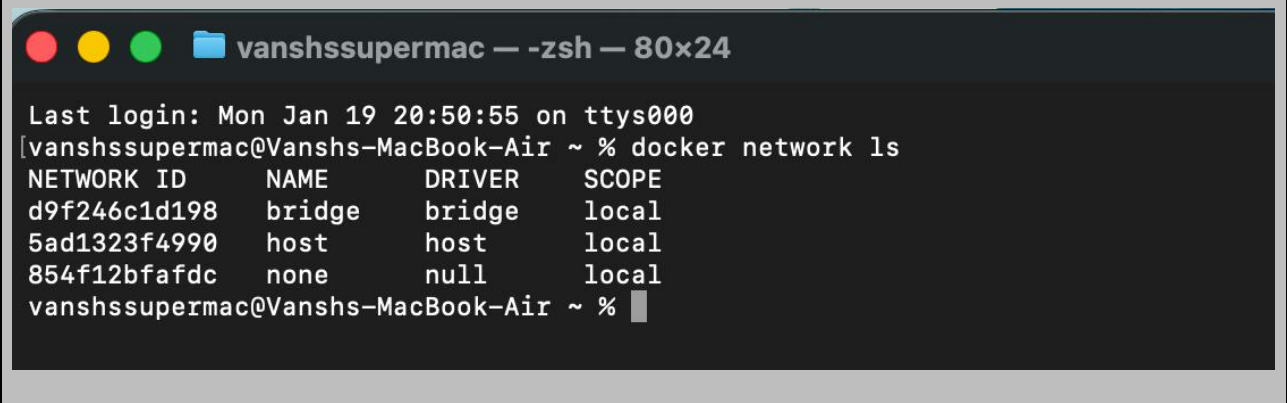
- bridge: The default network when a container starts.
- host: Bypasses Docker's network isolation and attaches the container directly to the host network.
- none: No networking is available for the container.

#### **1.1. Inspect Default Networks**

Check Docker's default networks using:

docker network

ls

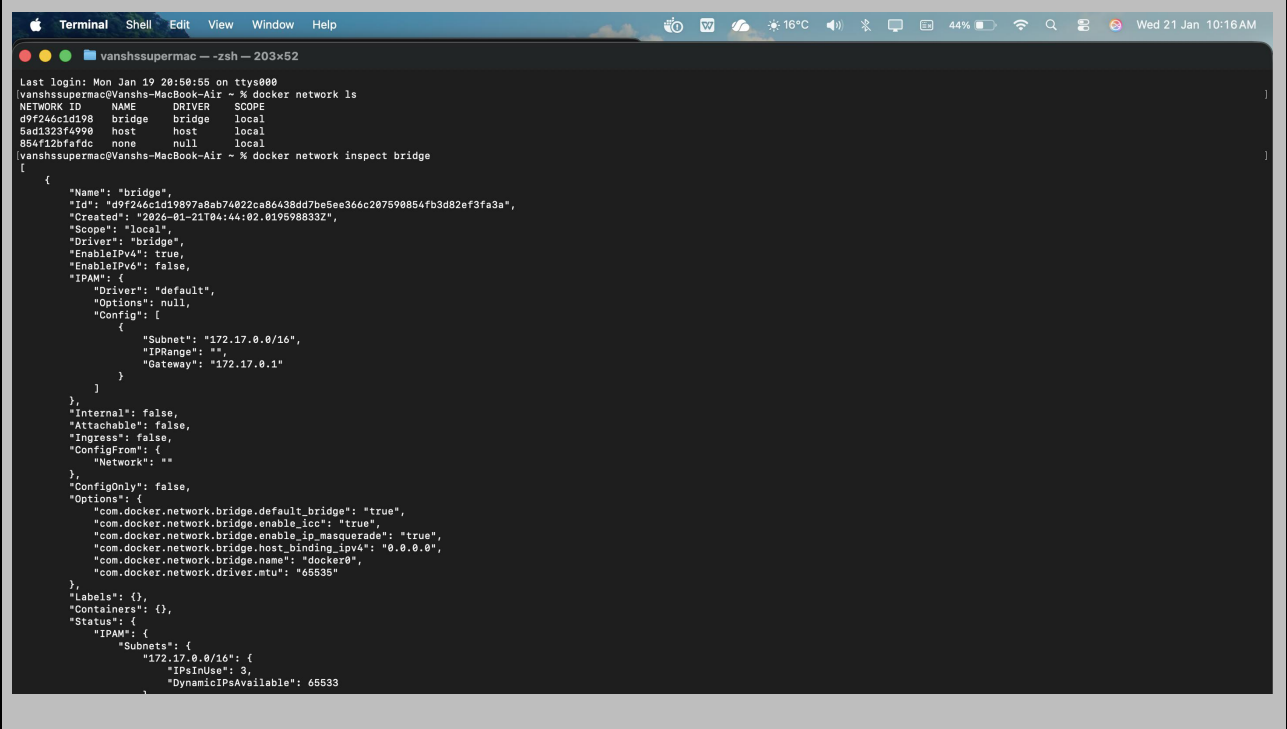


```
vanshssupermac — -zsh — 80x24

Last login: Mon Jan 19 20:50:55 on ttys000
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
d9f246c1d198        bridge             bridge             local
5ad1323f4990        host               host               local
854f12bfafdc        none              null               local
vanshssupermac@Vanshs-MacBook-Air ~ %
```

## 1.2. Inspect the Bridge Network

docker network inspect bridge



```
Terminal  Shell  Edit  View  Window  Help
vanshssupermac — -zsh — 203x52

Last login: Mon Jan 19 20:50:55 on ttys000
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network ls
NETWORK ID          NAME                DRIVER             SCOPE
d9f246c1d198        bridge             bridge             local
5ad1323f4990        host               host               local
854f12bfafdc        none              null               local
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "d9f246c1d1987a8ab74022ca86438dd7be5ee366c207590854fb3d82ef3fa3a",
    "Created": "2026-01-21T04:44:02.019598833Z",
    "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
            }
          }
        ]
      }
    }
  }
]
```

This command will show detailed information about the bridge network, including the connected containers and IP address ranges.

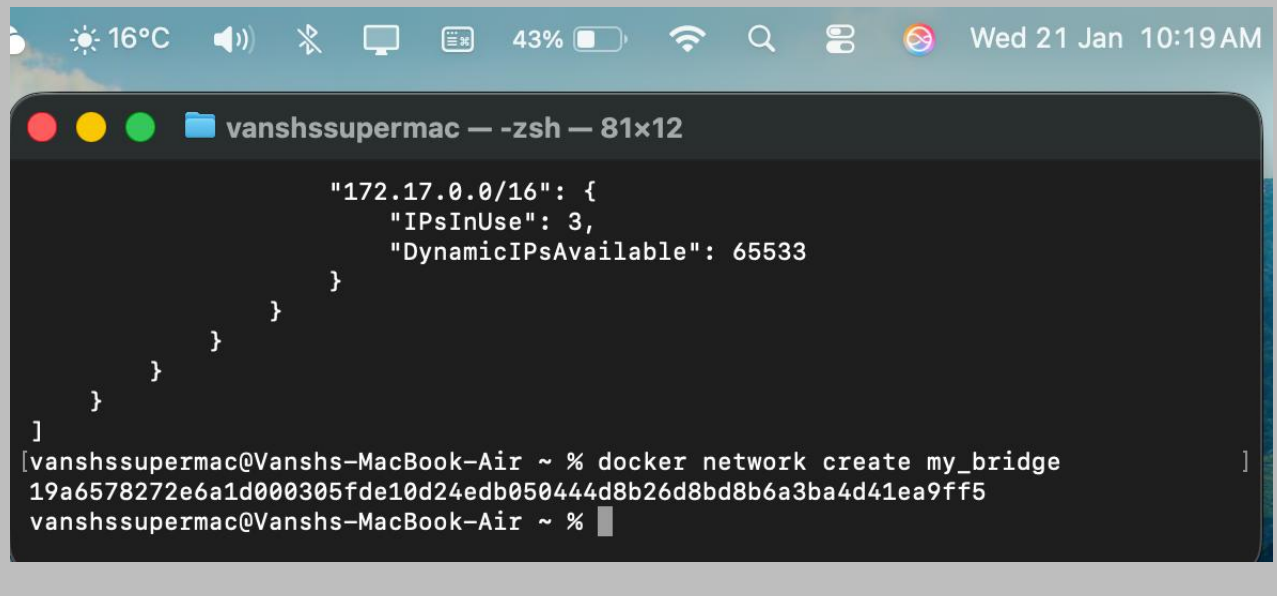
## Step 2: Create and Use a Bridge Network

## 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
```



A terminal window titled 'vanshssupermac — -zsh — 81x12' displays the command `docker network create my_bridge`. The output shows a JSON representation of the network configuration, including the IP range '172.17.0.0/16', the number of IPs in use (3), and the number of dynamically available IPs (65533). The terminal also shows the command being executed again, resulting in a long alphanumeric ID for the network.

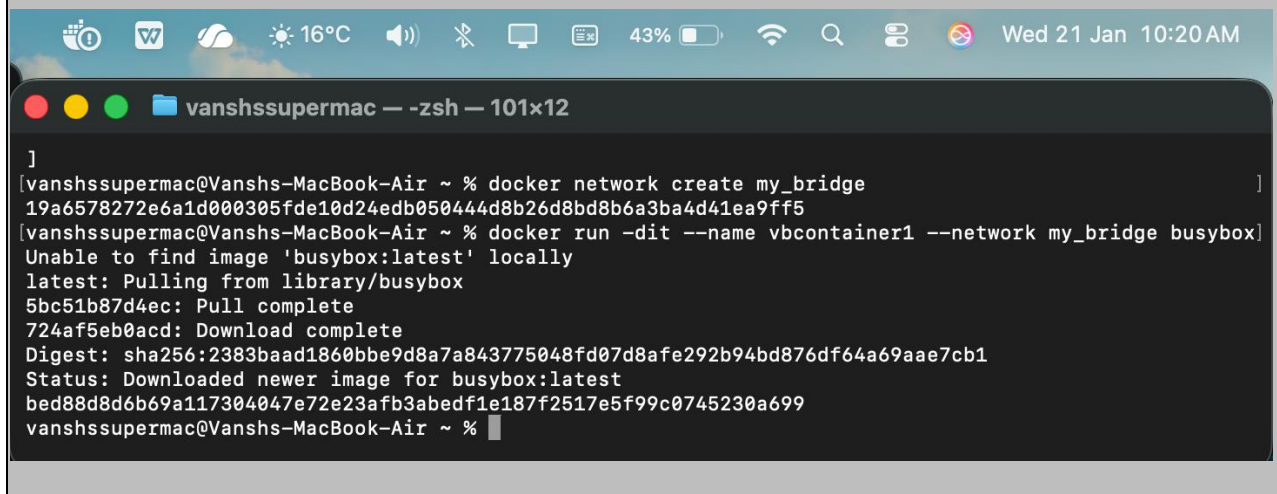
```
    "172.17.0.0/16": {
      "IPsInUse": 3,
      "DynamicIPsAvailable": 65533
    }
  }
}
]
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network create my_bridge
19a6578272e6a1d000305fde10d24edb050444d8b26d8bd8b6a3ba4d41ea9ff5
vanshssupermac@Vanshs-MacBook-Air ~ % ]
```

## 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
```



A terminal window titled 'vanshssupermac — -zsh — 101x12' shows the command `docker run -dit --name container1 --network my_bridge busybox`. The output indicates that the 'busybox:latest' image was not found locally and was pulled from the Docker library. It shows the pull progress, the image ID, and the digest.

```
]
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network create my_bridge
19a6578272e6a1d000305fde10d24edb050444d8b26d8bd8b6a3ba4d41ea9ff5
[vanshssupermac@Vanshs-MacBook-Air ~ % docker run -dit --name container1 --network my_bridge busybox]
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
5bc51b87d4ec: Pull complete
724af5eb0acd: Download complete
Digest: sha256:2383baad1860bbe9d8a7a843775048fd07d8afe292b94bd876df64a69aae7cb1
Status: Downloaded newer image for busybox:latest
bed88d8d6b69a117304047e72e23afb3abedf1e187f2517e5f99c0745230a699
vanshssupermac@Vanshs-MacBook-Air ~ % ]
```

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

```
busybox
```

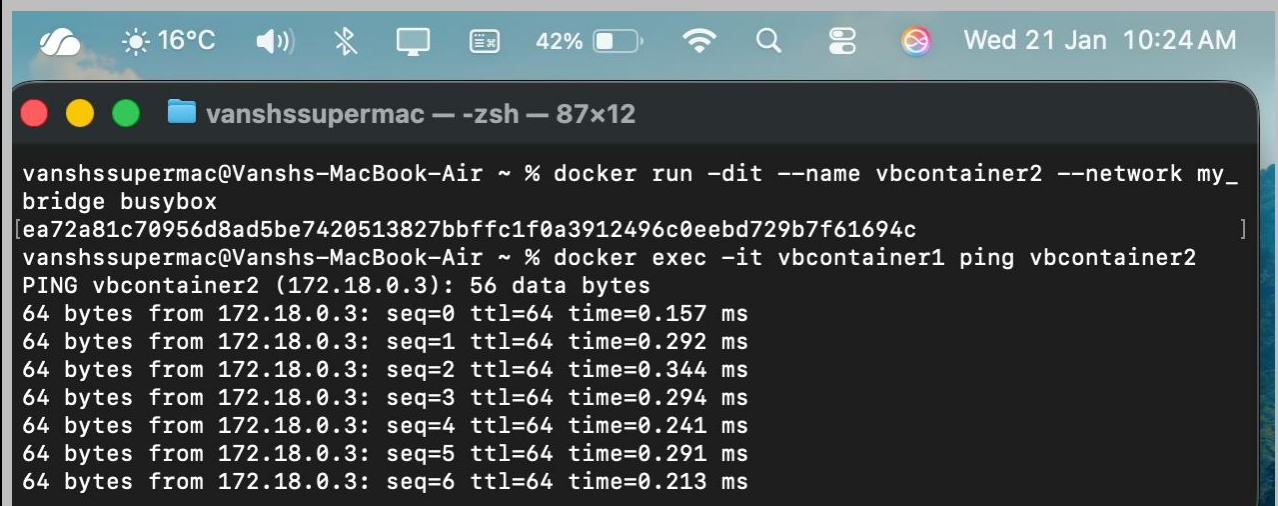
```
[vanshssupermac@Vanshs-MacBook-Air ~ % docker run -dit --name vbcontainer2 --network my_bridge busybox]  
ea72a81c70956d8ad5be7420513827bbffc1f0a3912496c0eebd729b7f61694c  
vanshssupermac@Vanshs-MacBook-Air ~ %
```

## 2.3. Test Container Communication

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

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

```
container2
```



The screenshot shows a macOS desktop environment with a terminal window titled "vanshssupermac — -zsh — 87x12". The terminal displays the following commands and output:

```
vanshssupermac@Vanshs-MacBook-Air ~ % docker run -dit --name vbcontainer2 --network my_bridge busybox  
[ea72a81c70956d8ad5be7420513827bbffc1f0a3912496c0eebd729b7f61694c]  
vanshssupermac@Vanshs-MacBook-Air ~ % docker exec -it vbcontainer1 ping vbcontainer2  
PING vbcontainer2 (172.18.0.3): 56 data bytes  
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.157 ms  
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.292 ms  
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.344 ms  
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.294 ms  
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.241 ms  
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.291 ms  
64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.213 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
```

```
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network disconnect my_bridge vbcontainer1 ]  
vanshssupermac@Vanshs-MacBook-Air ~ % █
```

## 4.2. Remove Networks

To remove the user-defined network:

```
docker network rm
```

```
my_bridge
```

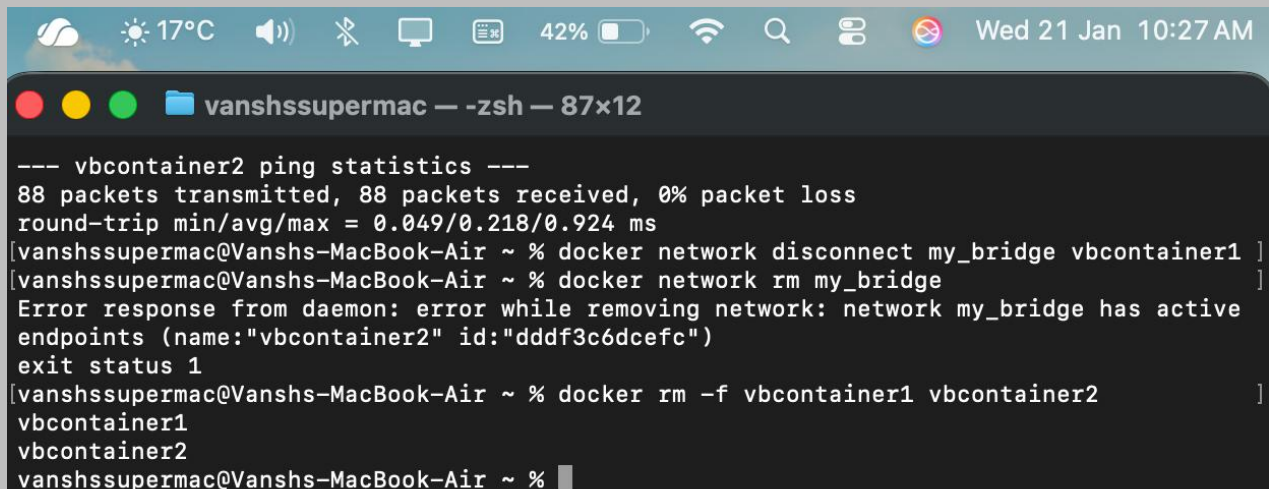
```
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network rm my_bridge  
Error response from daemon: error while removing network: network my_bridge has active  
endpoints (name:"vbcontainer2" id:"dddf3c6dcefc")  
exit status 1  
vanshssupermac@Vanshs-MacBook-Air ~ % █
```

## Step 4: Clean Up

Stop and remove all containers created during this exercise:

```
docker rm -f container1
```

```
container2
```



The screenshot shows a macOS terminal window with a title bar indicating the user is 'vanshssupermac' in a 'zsh' shell. The terminal output includes ping statistics for 'vbcontainer2', followed by the same sequence of Docker commands as in the previous block: disconnecting 'vbcontainer1' from 'my\_bridge', attempting to remove 'my\_bridge' (which fails due to 'vbcontainer2' being active), and finally removing both 'vbcontainer1' and 'vbcontainer2' with the '-f' flag. The system status bar at the top shows a temperature of 17°C, 42% battery, and the date 'Wed 21 Jan 10:27 AM'.

```
--- vbcontainer2 ping statistics ---  
88 packets transmitted, 88 packets received, 0% packet loss  
round-trip min/avg/max = 0.049/0.218/0.924 ms  
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network disconnect my_bridge vbcontainer1 ]  
[vanshssupermac@Vanshs-MacBook-Air ~ % docker network rm my_bridge  
Error response from daemon: error while removing network: network my_bridge has active  
endpoints (name:"vbcontainer2" id:"dddf3c6dcefc")  
exit status 1  
[vanshssupermac@Vanshs-MacBook-Air ~ % docker rm -f vbcontainer1 vbcontainer2  
vbcontainer1  
vbcontainer2  
vanshssupermac@Vanshs-MacBook-Air ~ % █
```