# **Lab Exercise 4- Working with Docker**

# **Networking**

**Name: Aditya Tomar** 

SAP: 500106015

E.NO: R2142221060

**Batch: B-2(DevOps)** 

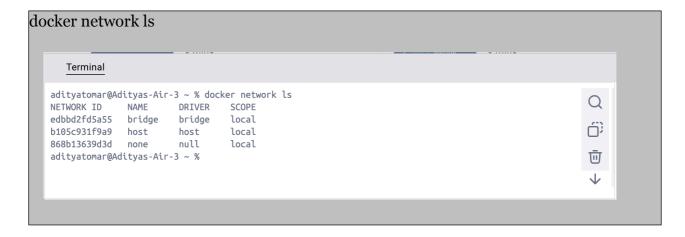
# Step 1: Understanding Docker Default Networks

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:



### 1.2. Inspect the Bridge Network

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

adityatomar@Adityas-Air-3 ~ % docker network create my_bridge
a497b1b9209a3d7877095b37689c2512db32df44fbb766ca762d332b09186bc3
adityatomar@Adityas-Air-3 ~ %
```

### 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

```
adityatomar@Adityas-Air-3 ~ % docker run -dit --name container1 --network my_bridge busybox Unable to find image 'busybox:latest' locally latest: Pulling from library/busybox 75e8ca8f509f: Pull complete Digest: sha256:34b191d63fbc93e25e275bfccf1b5365664e5ac28f06d974e8d50090fbb49f41 Status: Downloaded newer image for busybox:latest 7231f0b3b3adcae870021c996ef8e1e5f62a9591cf6b474890222cd025bbb83a
```

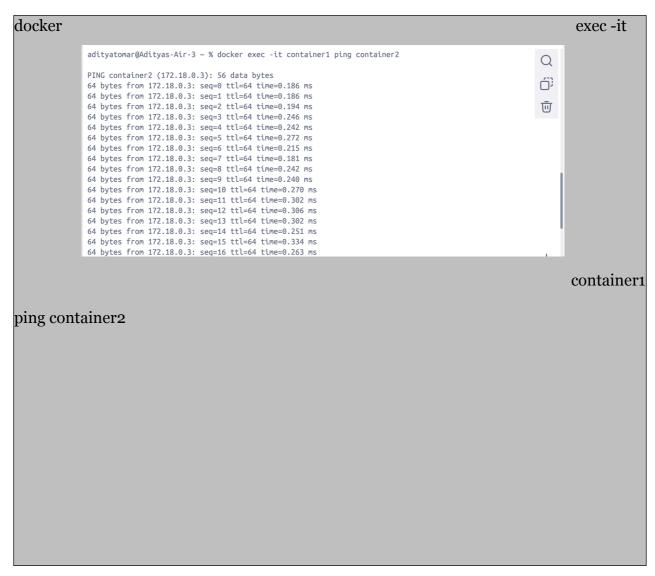
```
docker run -dit --name container2 --network my_bridge busybox

adityatomar@Adityas-Air-3 ~ % docker run -dit --name container2 --network my_bridge busybox

d5a5bc3fa683e5fb867feae5609ffa68bf301a080ced97ea858d032c39a8c20a
```

### 2.3. Test Container Communication

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



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
 adityatomar@Adityas-Air-3 ~ % docker run -d --name host_network_container --network host nginx
                                                                                                       Q
 Unable to find image 'nginx:latest' locally
                                                                                                       □;
 latest: Pulling from library/nginx
 92c3b3500be6: Pull complete
                                                                                                       回
 ee57511b3c68: Pull complete
 33791ce134bf: Pull complete
 cc4f24efc205: Pull complete
 3cad04a21c99: Pull complete
 486c5264d3ad: Pull complete
 b3fd15a82525: Pull complete
 Digest: sha256:04ba374043ccd2fc5c593885c0eacddebabd5ca375f9323666f28dfd5a9710e3
 Status: Downloaded newer image for nginx:latest
 0b54ba7a4ddaf22a060dd2d24801f5aa8fc4038ccec8148ed9006121b5bae499
```

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
               adityatomar@Adityas-Air-3 ~ % docker network inspect host
                                                                                                                               Q
               [
                                                                                                                               \bigcap_{i}
                        "Name": "host"
                        "Id": "b105c931f9a9ab99e6bf5558adf79f0db1e894bde08d92368230bdecce2e93e8",
                        "Created": "2024-08-11T14:26:36.057801291Z",
                                                                                                                               回
                        "Scope": "local",
"Driver": "host",
                        "EnableIPv6": false,
                        "IPAM": {
                            "Driver": "default",
                            "Options": null,
                            "Config": null
                        "Internal": false,
                        "Attachable": false,
                        "Ingress": false,
                        "ConfigFrom": {
    "Network": ""
                        "ConfigOnly": false,
```

**Step 4: Disconnect and Remove Networks** 

### 4.1. Disconnect Containers from Networks

To disconnect container1 from my\_bridge:



### 4.2. Remove Networks

To remove the user-defined network:

```
docker network rm my_bridge

adityatomar@Adityas-Air-3 ~ % docker network rm my_bridge

my_bridge
```

## Step 5: Clean Up

Stop and remove all containers created during this exercise:

