

**SCHOOL OF COMPUTER SCIENCE**

**Container and Docker Security Lab**

**Submitted by**

|  |  |
| --- | --- |
| **Name** | **Vibhav Khaneja** |
| **Branch** | **BTech CSE(DevOps) B-1(NH)** |
| **Semester** | **5** |
| **SAPID** | **500105662** |
| **Roll no** | **R2142220297** |

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

**Step 1: Understanding Docker Default Networks**

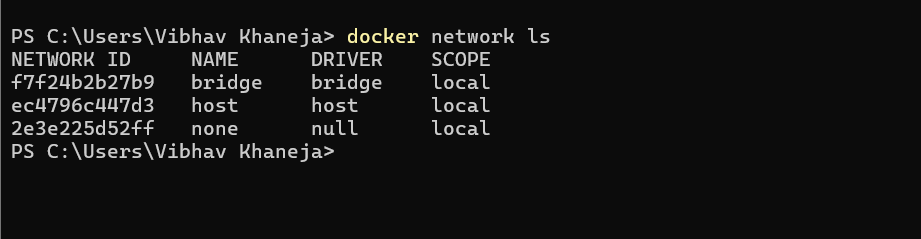
Docker provides three default networks:

* 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



**1.2. Inspect the Bridge Network**

docker network inspect bridge



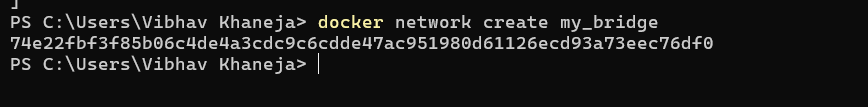
This command will show detailed information about the bridge network, including the connected containers and IP address ranges.

**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

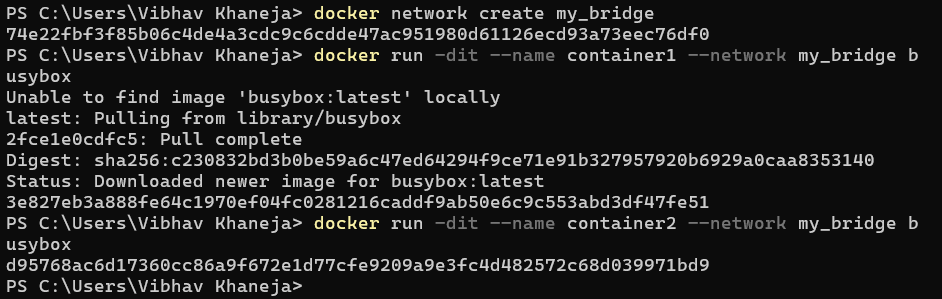


**2.2. Run Containers on the User-Defined Network**

Start two containers on the newly created my\_bridge network:

docker run -dit --name container1 --network my\_bridge busybox

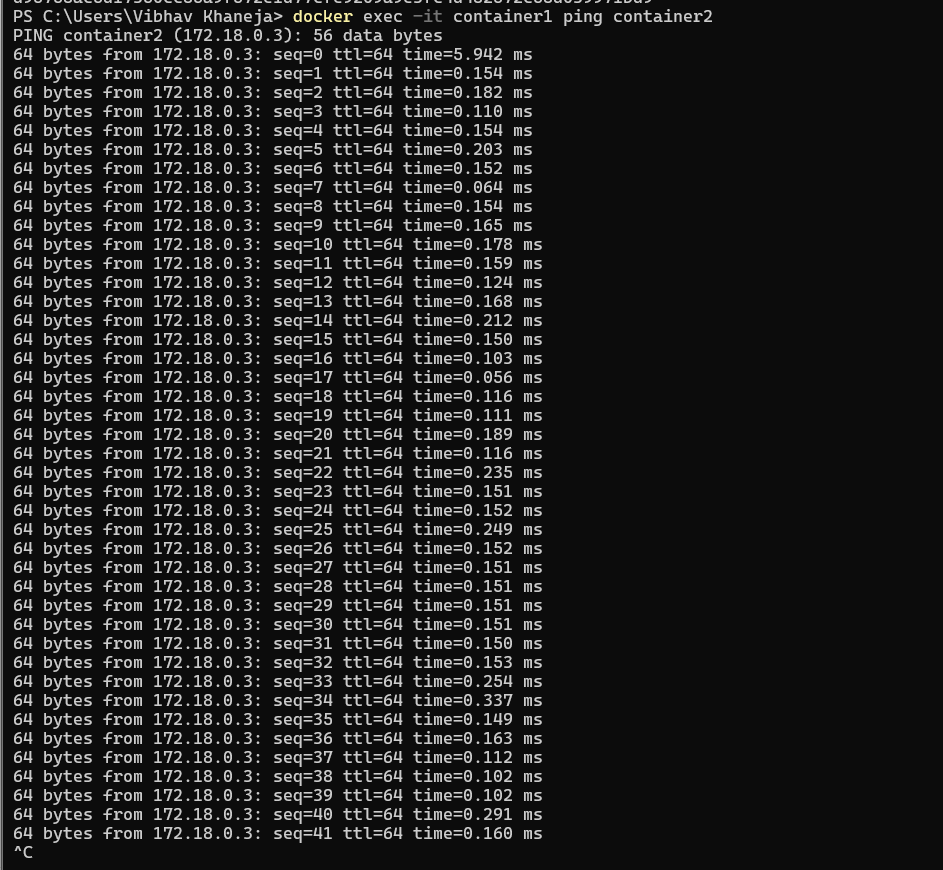
docker run -dit --name container2 --network my\_bridge busybox



**2.3. Test Container Communication**

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

docker exec -it container1 ping container2



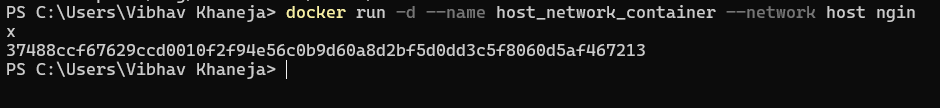
The containers should be able to communicate since they are on the same network.

**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:

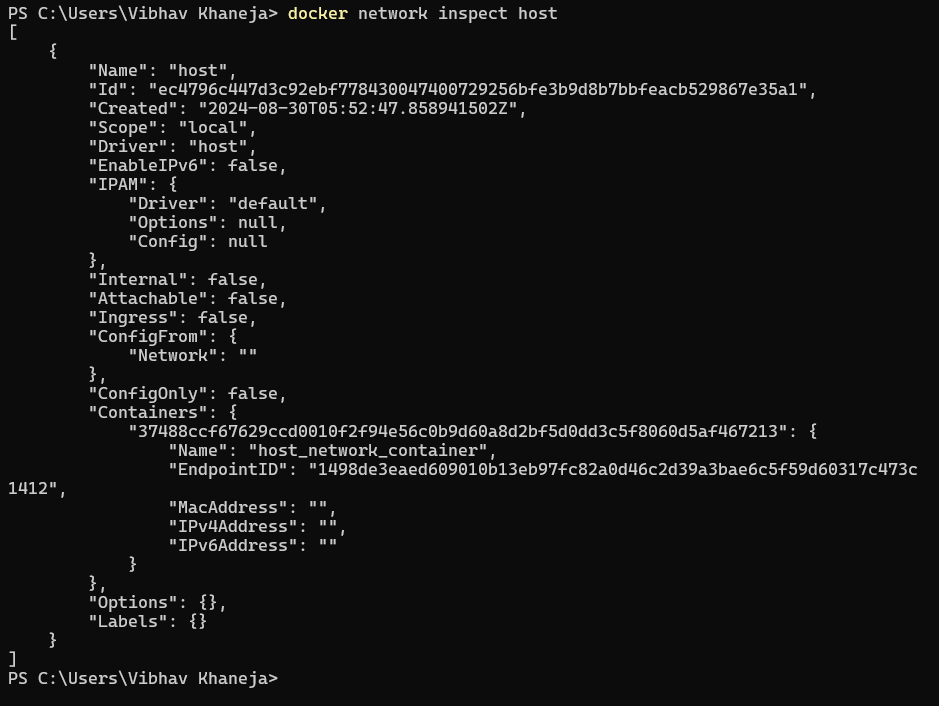
docker run -d --name host\_network\_container --network host nginx



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



**Step 4: Disconnect and Remove Networks**

**4.1. Disconnect Containers from Networks**

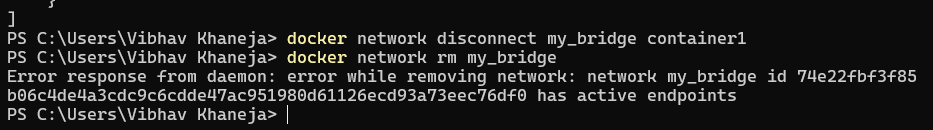
To disconnect container1 from my\_bridge:

docker network disconnect my\_bridge container1

**4.2. Remove Networks**

To remove the user-defined network:

docker network rm my\_bridge



**Step 5: Clean Up**

Stop and remove all containers created during this exercise:

docker rm -f container1 container2 host\_network\_container

