**Lab Exercise 7- Understanding Docker Network Types**

**Lab Exercise: Understanding Docker Network Types**

**Objective:**

To explore and understand how different Docker network types (bridge, host, none, and user-defined networks) work by creating containers and observing their behavior on these networks.

**Prerequisites:**

* Docker installed on your machine
* Basic understanding of Docker commands

**Part 1: Overview of Docker Networks**

Docker provides several networking options:

1. **Bridge Network:**
   * The default network for containers if none is specified.
   * Containers on the same bridge network can communicate with each other.
2. **Host Network:**
   * Binds the container directly to the host’s network, removing network isolation.
3. **None Network:**
   * The container has no network interface (no internet access).
4. **User-Defined Bridge Network:**
   * Custom networks where containers can automatically discover each other via container names.

**Part 2: Exploring Bridge Network**

1. **Create and Run a Container on the Default Bridge Network:**

By default, Docker containers run on the bridge network if no network is specified. Let’s create a container and explore this:

docker run -d --name bridge-container --network bridge nginx

This starts an NGINX web server in a container on the default bridge network.

1. **Check the IP Address of the Container:**

You can inspect the container's network settings using:

docker inspect bridge-container | grep "IPAddress"

This shows the container’s IP address on the bridge network.

1. **Access the NGINX Server:**

Forward port 80 from the container to your host and access the NGINX server:

docker run -d -p 8080:80 --name bridge-nginx nginx

Now, open your browser and go to http://localhost:8080 to see the NGINX welcome page.

1. **Container-to-Container Communication:**

Launch another container and try to ping the first container:

docker run -it --network bridge busybox ping bridge-container

The containers on the same bridge network can communicate using their IP addresses or container names.

**Part 3: Exploring Host Network**

1. **Run a Container on the Host Network:**

Let’s create a container that uses the host network mode:

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

This container will use the host machine’s network stack, meaning there’s no network isolation.

1. **Access NGINX via Host Network:**

Since the container is bound directly to the host’s network, you can access NGINX by going to http://localhost without any port mapping.

1. **Observe Differences with Bridge Network:**

Compare this with the bridge network. In the host mode, the container does not have its own private IP address because it shares the host's networking stack.

**Part 4: Exploring None Network**

1. **Run a Container with No Network:**

Launch a container in the none network mode:

docker run -d --network none --name none-container nginx

This container has no network interface and cannot access the outside world.

1. **Check the IP Address:**

Inspect the container's network settings:

docker inspect none-container | grep "IPAddress"

You will see that there is no IP address associated with this container because it has no network interface.

1. **Verify the Isolation:**

Try to ping an external service from inside the container:

docker exec -it none-container ping google.com

The command will fail because the container is isolated from the network.