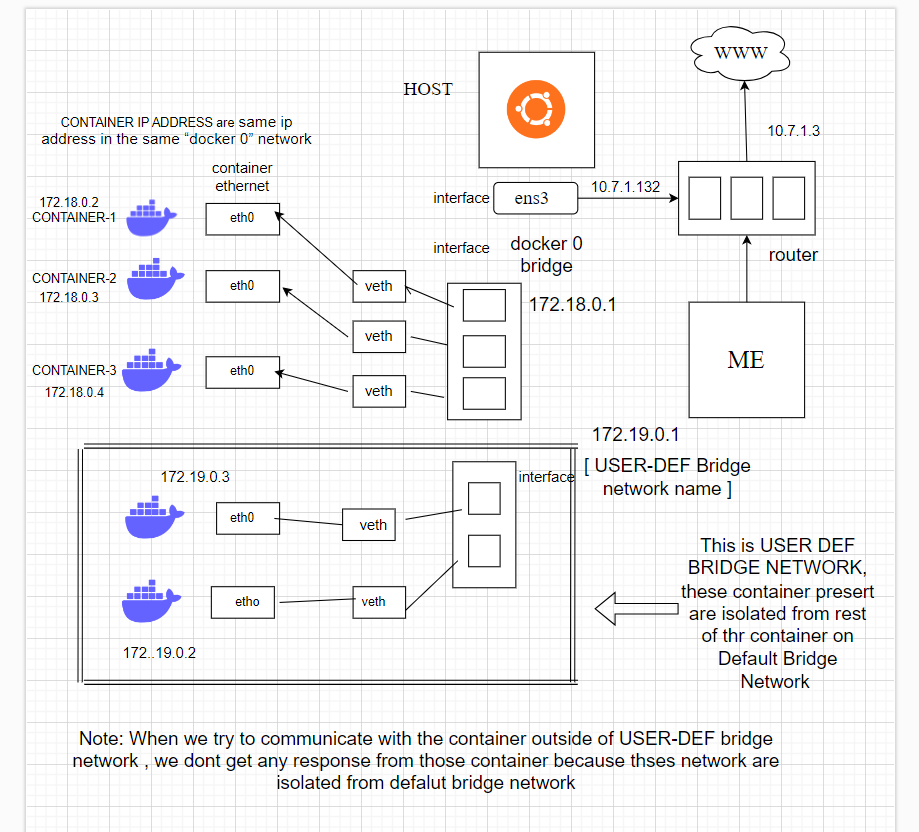
**USER DEFINED BRIDGE NETWORK**

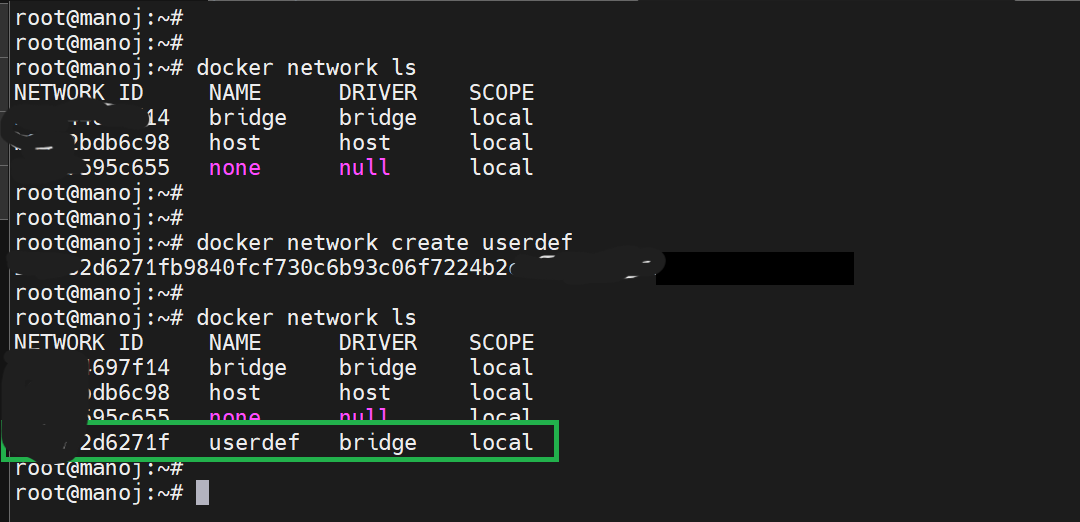
user-defined bridge network in Docker allows you to create custom bridge networks that provide enhanced features over the default bridge network. This custom network allows containers to communicate with each other using container names and provides better isolation and control over networking.

* **Benefits of User-Defined Bridge Networks:**
* Automatic DNS Resolution: Containers can resolve each other by name, making it easier to refer to other containers in your applications.
* Isolation: Containers on a user-defined bridge network are isolated from containers on other networks unless explicitly configured to communicate across networks.
* Improved Flexibility: You can configure network options like subnet, gateway, and IP address ranges to suit your specific needs.
* Easier Management: User-defined bridge networks make managing multiple container communications simpler and more scalable.

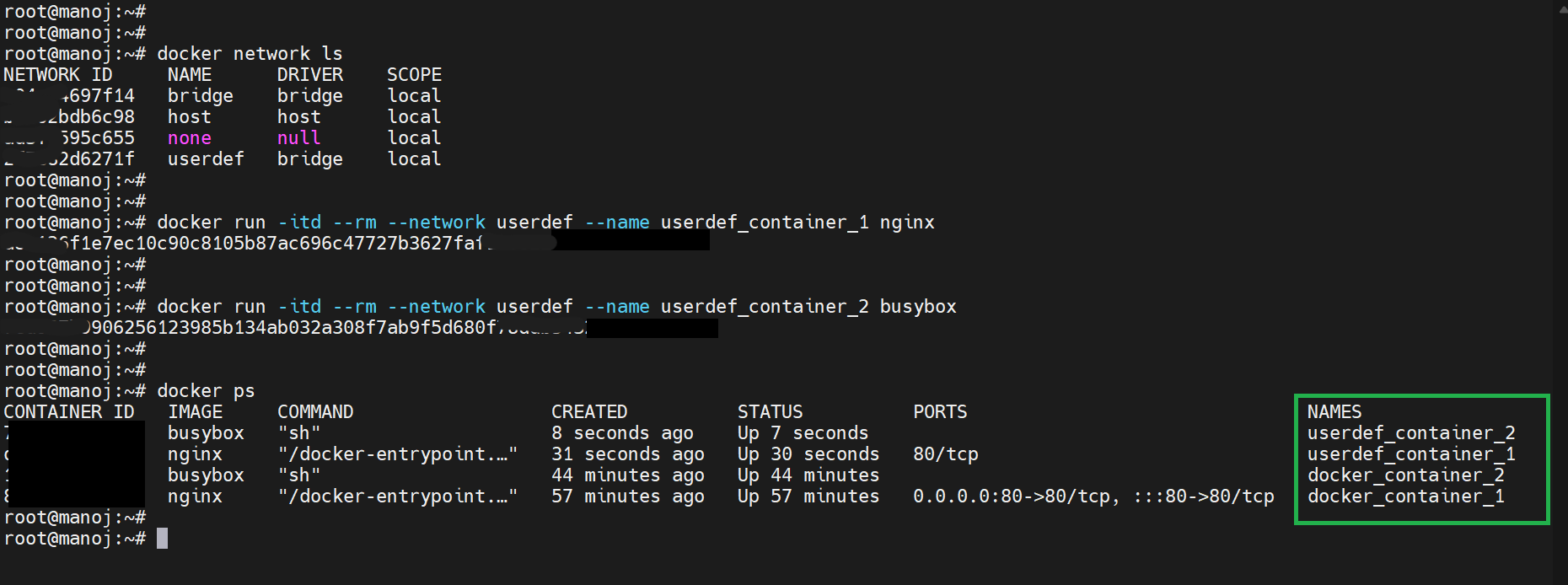


Note:- Can check below how to communicate with the containers present in default network and user-def network.

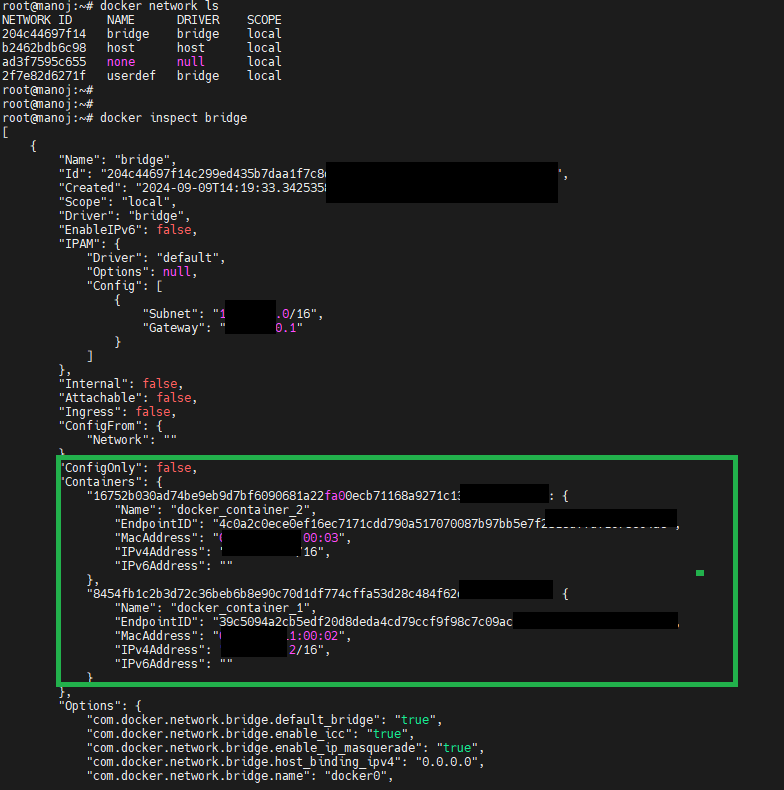
Creating the USER DEFINE BRIDGE NETWORK

****

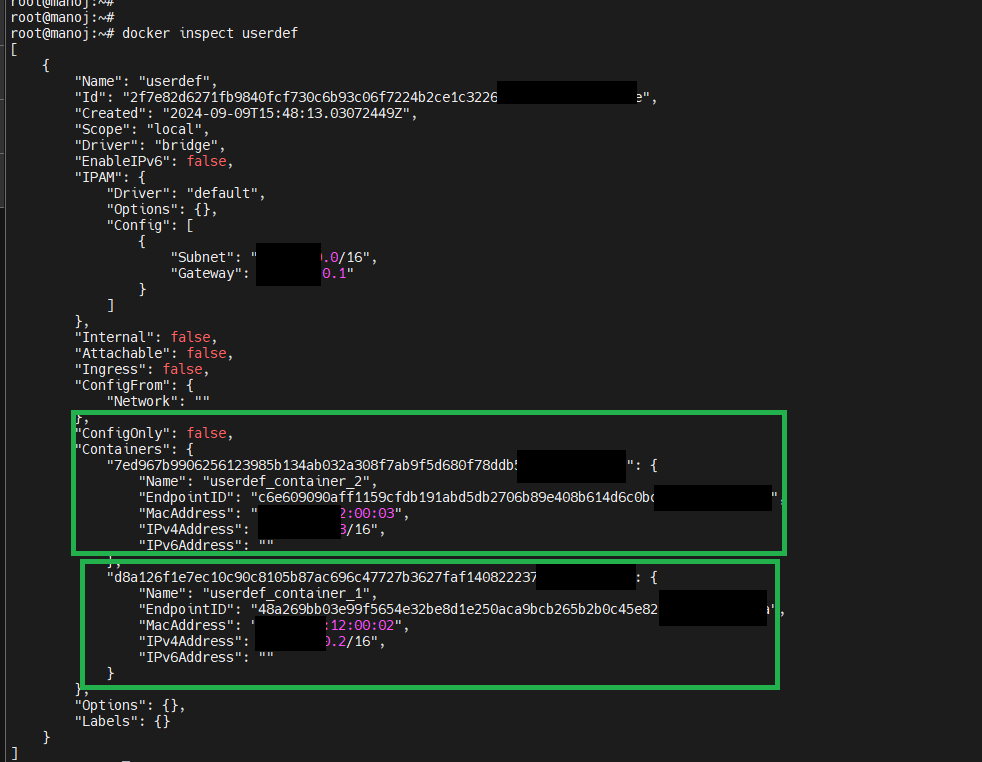
Running the container inside the User-def Bridge Network

****

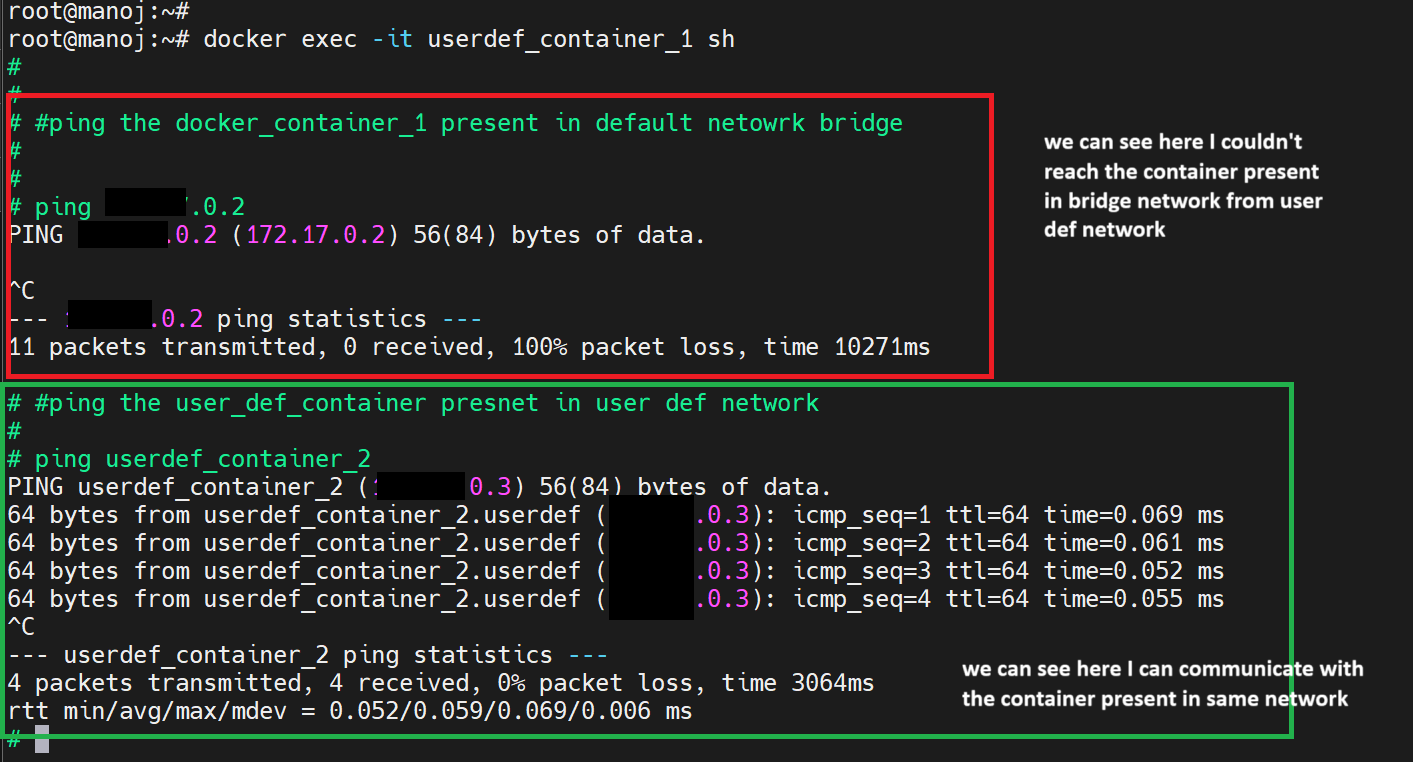
**Default Bridge Network**

****

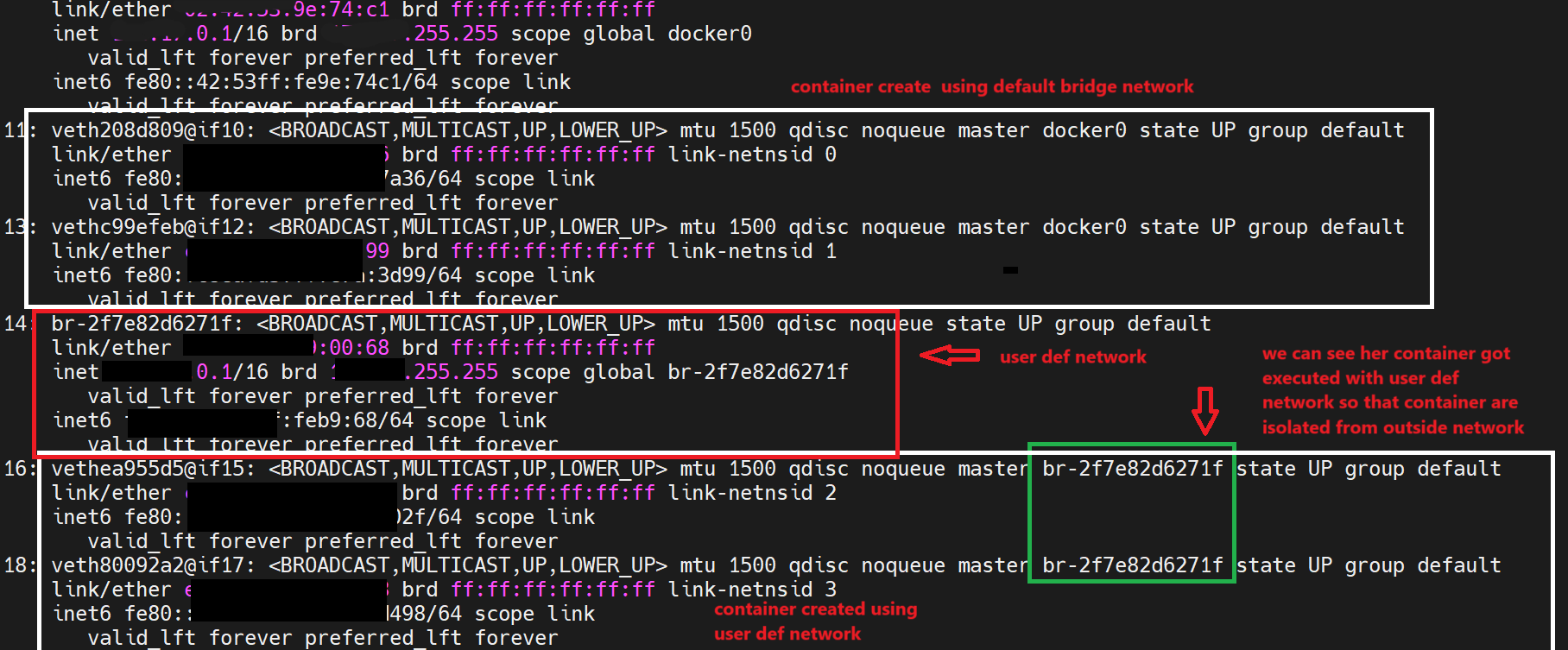
**USER-DEFINE BRIDGE NETWORK**

****

We can see here, Container got isolated from default bridge network. When we try to ping the container that present in default bridge network it is not communicating with the container present in User-def Bridge network.

****

We can see that container are running on User-def Bridge Network

****

* **Enabling Communication**

By default, containers on different networks cannot communicate directly. To enable communication, you have two main options:

**Option A: Use Host Network**

Run one of the containers using the host network. This allows it to communicate with any other containers on the host, but it might not be suitable for all use cases.

docker run -d --name my\_host\_container --network host nginx

**Option B: Connect Containers to Both Networks**

You can connect containers to both the default bridge network and your custom network. Here’s how you can do it:

1. Run the default bridge network container:

docker run -d --name my\_default\_container --network bridge nginx

1. Connect this container to your custom network:

docker network connect my\_custom\_network my\_default\_container

1. Run the custom bridge network container:

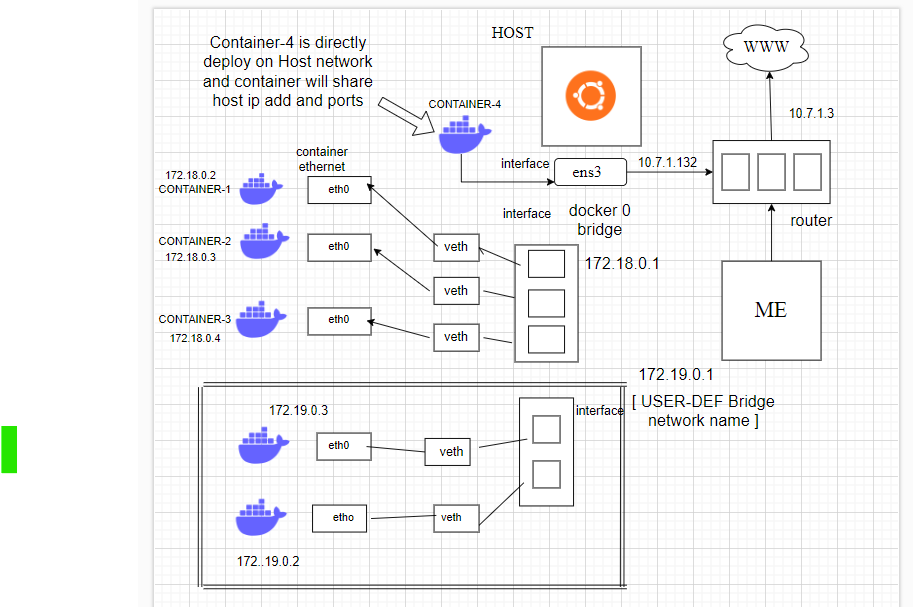
docker run -d --name my\_custom\_container --network my\_custom\_network nginx

Now, both containers are connected to both networks and should be able to communicate with each other.

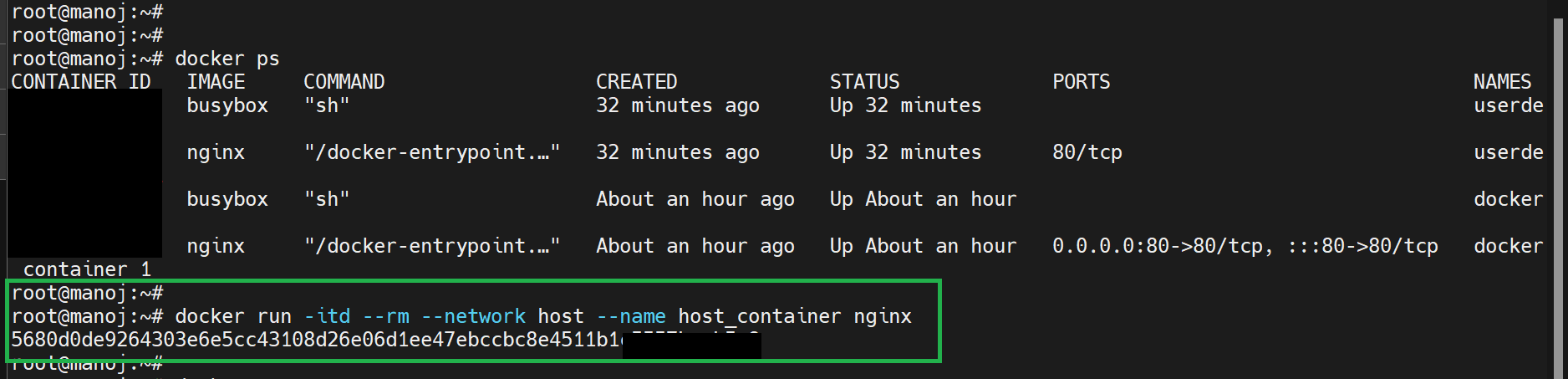
**Host Network**

The container shares the network stack with the host, meaning it uses the host's IP address. There's no network isolation between the container and the host.

* **Features**:
* No network overhead (since the container shares the host's network stack).
* Suitable for applications where performance and direct access to the host’s network are critical.
* **Use case**: High-performance applications or cases where containers need to access low-level networking interfaces of the host.



Creating a Docker Container on HOST Network



When we create a container on HOST Network, we can access the container directly without provide any port because the ports and IP address are assigned from HOST network only.

