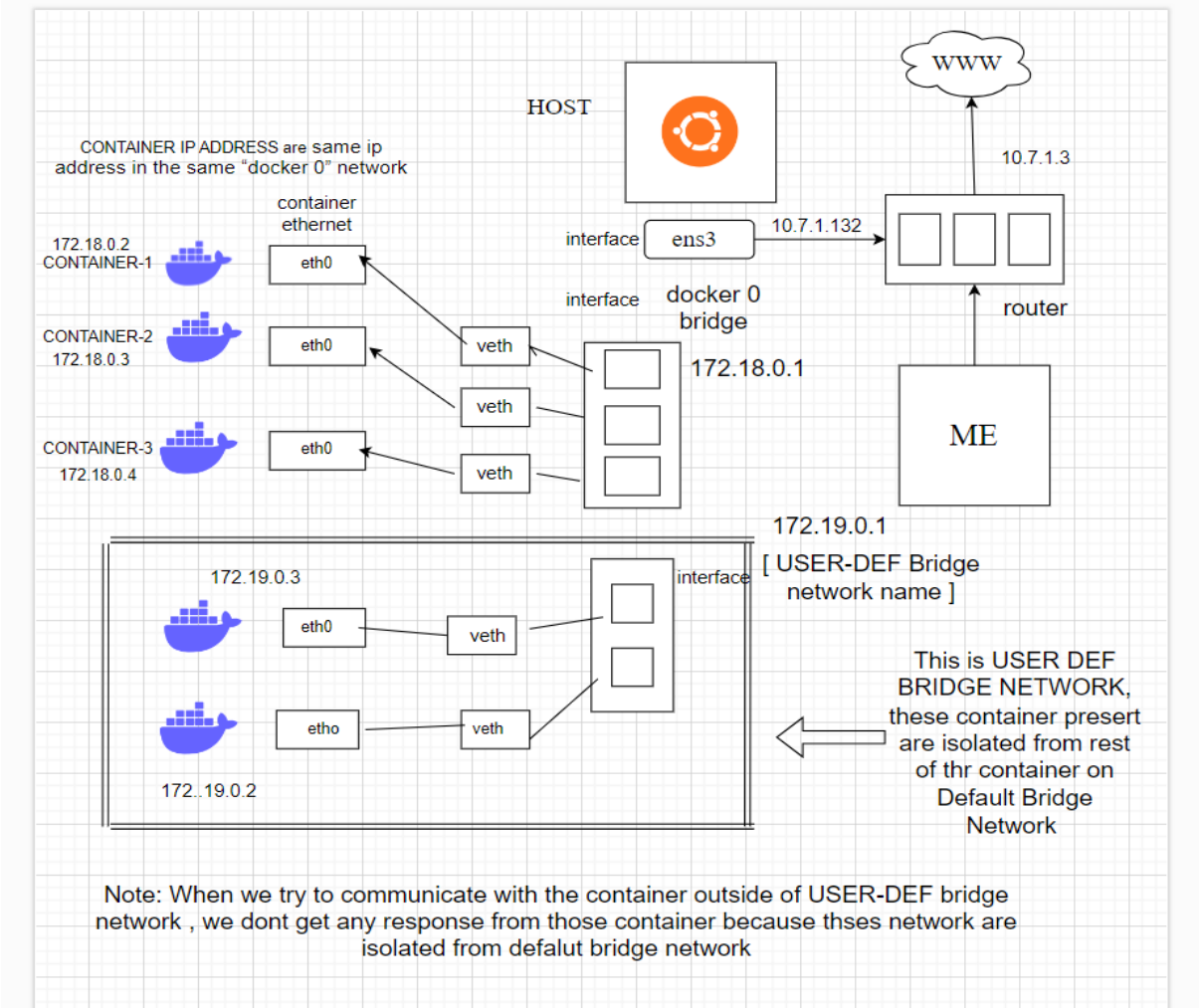
USER DEFINED BRIDGE NETWORK & HOST NETWORK

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:**

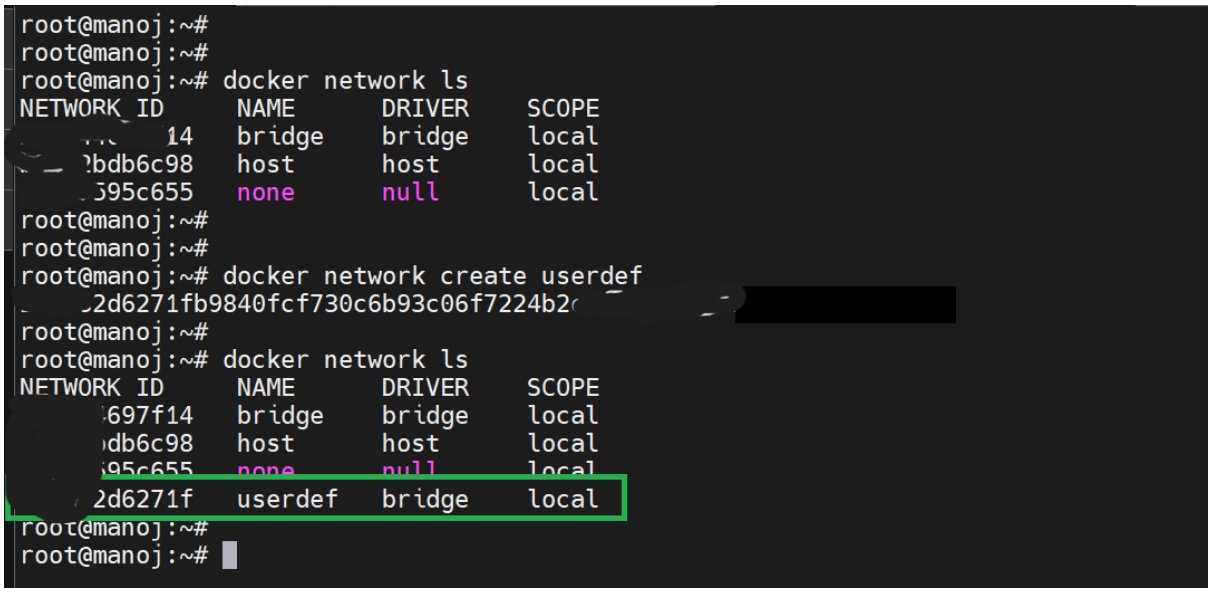
- Automatic DNS Resolution: Containers can resolve each other by name, making it easier to refer to other containers in your applications.
- Isolation: Containers on a user-defined bridge network are isolated from containers on other networks unless explicitly configured to communicate across networks.
- 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.

USER DEFINED BRIDGE NETWORK & HOST NETWORK

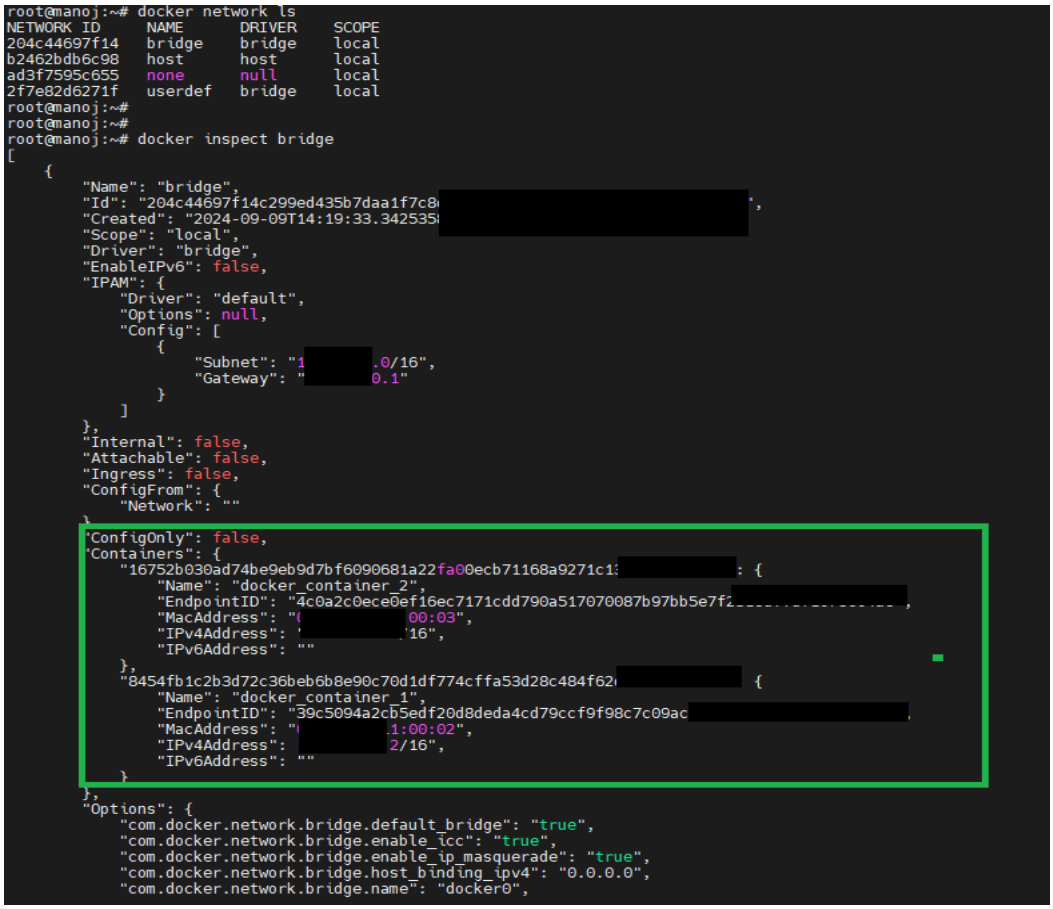
Creating the USER DEFINE BRIDGE NETWORK



Running the container inside the User-def Bridge Network

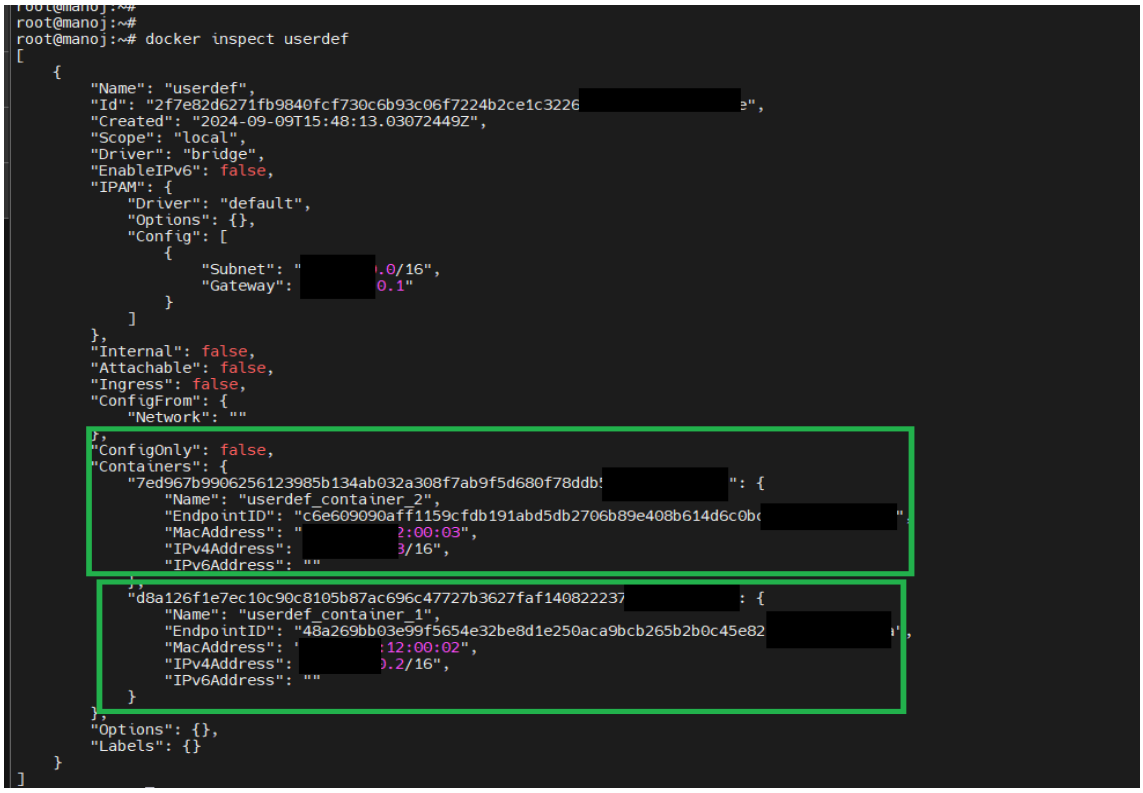
USER DEFINED BRIDGE NETWORK & HOST NETWORK

Default Bridge Network



```
root@manoj:~# docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
204c44697f14   bridge    bridge    local
b2462bdb6c98   host      host      local
ad3f7595c655   none      null      local
2f7e82d6271f   userdef   bridge    local
root@manoj:~#
root@manoj:~#
root@manoj:~# docker inspect bridge
[
    {
        "Name": "bridge",
        "Id": "204c44697f14c299ed435b7daa1f7c8[redacted]",
        "Created": "2024-09-09T14:19:33.342535[redacted]",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "1[redacted].0/16",
                    "Gateway": "[redacted]0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "16752b030ad74be9eb9d7bf6090681a22fa[redacted]": {
                "Name": "docker_container_2",
                "EndpointID": "4c0a2c0ece0ef16ec7171cdd790a517070087b97bb5e7f2[redacted]",
                "MacAddress": "([redacted]00:03",
                "IPv4Address": "[redacted]16",
                "IPv6Address": ""
            },
            "8454fb1c2b3d72c36beb6b8e90c70d1df774cffa53d28c484f62[redacted]": {
                "Name": "docker_container_1",
                "EndpointID": "39c5094a2cb5edf20d8deda4cd79ccf9f98c7c09ac[redacted]",
                "MacAddress": "[redacted]1:00:02",
                "IPv4Address": "[redacted]2/16",
                "IPv6Address": ""
            }
        },
        "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",
```

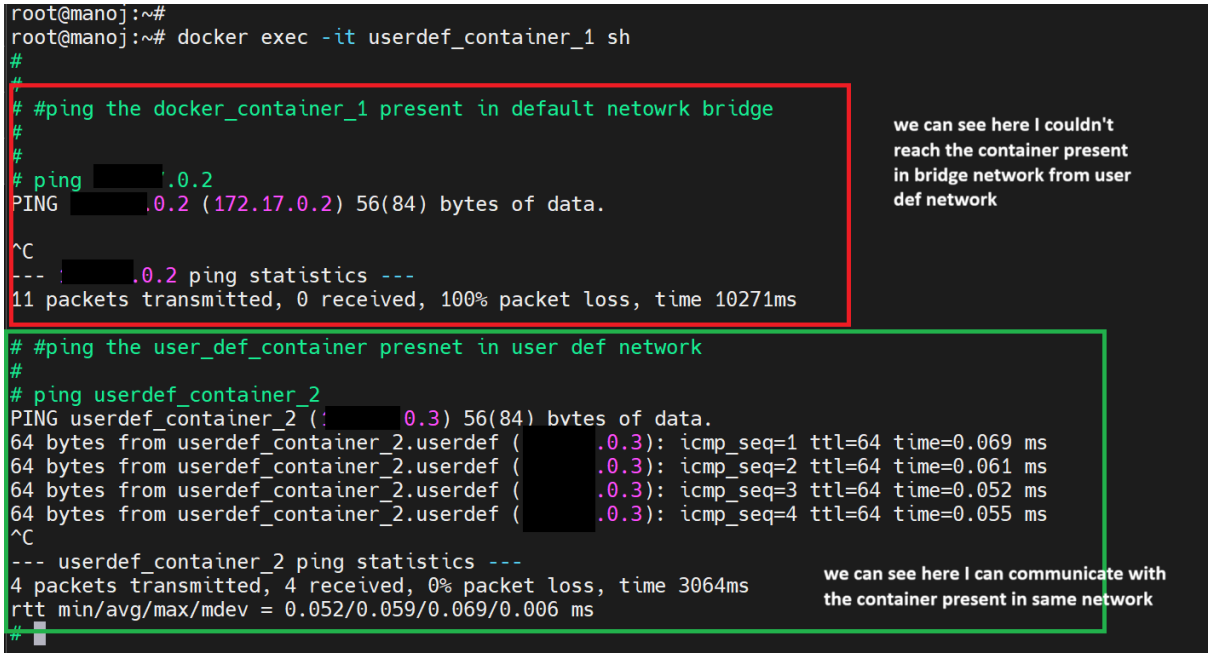
USER-DEFINE BRIDGE NETWORK



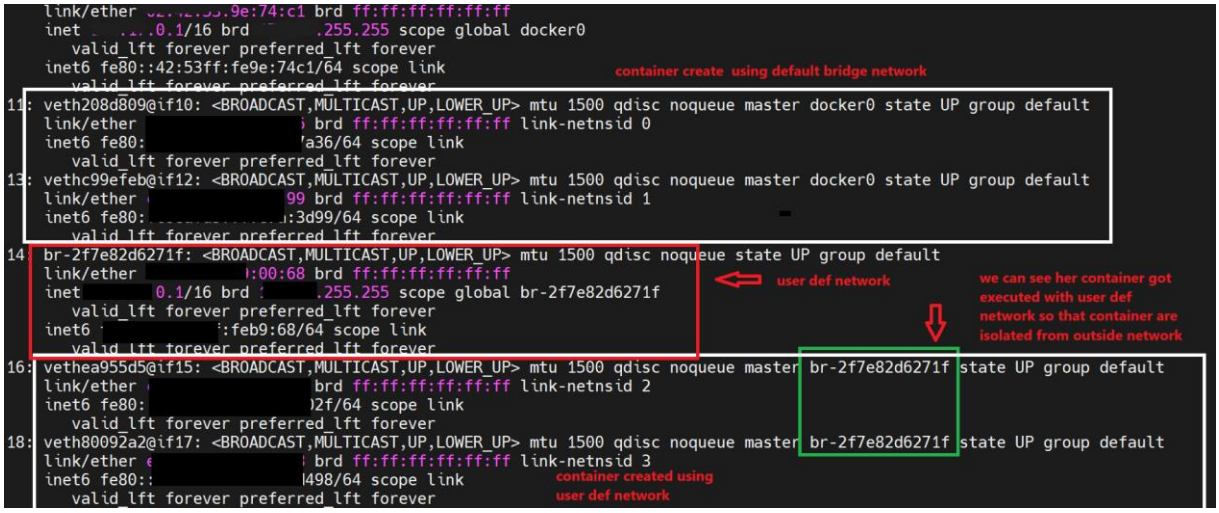
```
root@manoj:~#
root@manoj:~# docker inspect userdef
[
    {
        "Name": "userdef",
        "Id": "2f7e82d6271fb9840fcf730c6b93c06f7224b2ce1c3226[redacted]e",
        "Created": "2024-09-09T15:48:13.03072449Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "[redacted].0/16",
                    "Gateway": "[redacted]0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "7ed967b9906256123985b134ab032a308f7ab9f5d680f78ddb5[redacted]": {
                "Name": "userdef_container_2",
                "EndpointID": "c6e609090aff1159cfdb191abd5db2706b89e408b614d6c0b[redacted]",
                "MacAddress": "[redacted]2:00:03",
                "IPv4Address": "[redacted]3/16",
                "IPv6Address": ""
            },
            "d8a126f1e7ec10c90c8105b87ac696c47727b3627faf140822237[redacted]": {
                "Name": "userdef_container_1",
                "EndpointID": "48a269bb03e99f5654e32be8d1e250aca9bcb265b2b0c45e82[redacted]",
                "MacAddress": "[redacted]:12:00:02",
                "IPv4Address": "[redacted]0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

USER DEFINED BRIDGE NETWORK & HOST 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

USER DEFINED BRIDGE NETWORK & HOST NETWORK

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
```

2. Connect this container to your custom network:

```
docker network connect my_custom_network my_default_container
```

3. Run the custom bridge network container:

```
docker run -d --name my_custom_container --network my_custom_network nginx
```

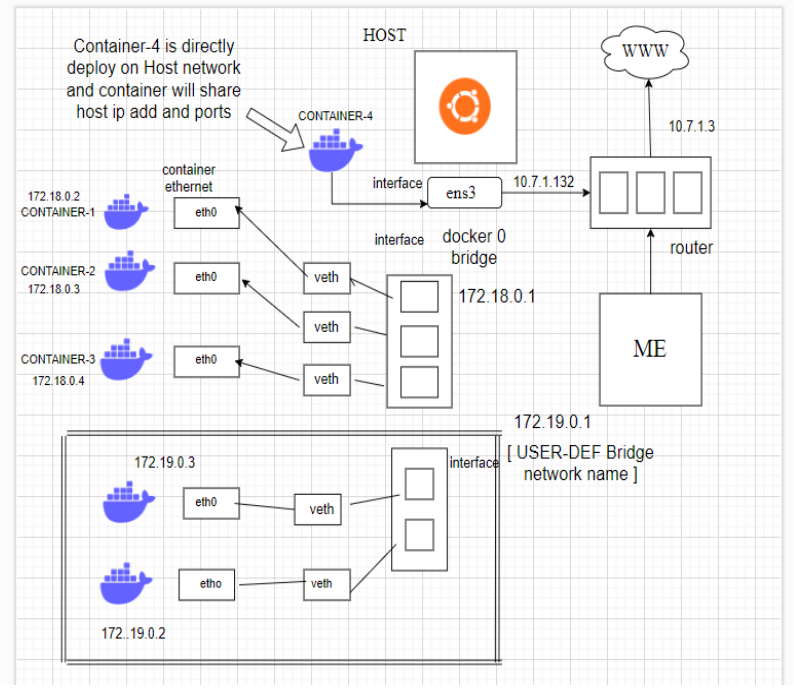
Now, both containers are connected to both networks and should be able to communicate with each other.

USER DEFINED BRIDGE NETWORK & HOST NETWORK

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

```
root@manoj:~#
root@manoj:~#
root@manoj:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED             STATUS             PORTS                                   NAMES
               busybox   "sh"                     32 minutes ago      Up 32 minutes                                              userde
               nginx     "/docker-entrypoint.…"   32 minutes ago      Up 32 minutes      80/tcp                                  userde
               busybox   "sh"                     About an hour ago   Up About an hour                                           docker
               nginx     "/docker-entrypoint.…"   About an hour ago   Up About an hour   0.0.0.0:80->80/tcp, :::80->80/tcp       docker
 container 1
root@manoj:~#
root@manoj:~# docker run -itd --rm --network host --name host_container nginx
5680d0de9264303e6e5cc43108d26e06d1ee47ebccbc8e4511b1
```

USER DEFINED BRIDGE NETWORK & 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.