

## Lesson 05 Demo 03

### Creating a Host Network

**Objective:** To create a host network for a standalone container and enable direct binding to the Docker host's network interface for seamless integration and optimal network performance

**Tools required:** Ubuntu OS and Docker

**Prerequisites:** None

Steps to be followed:

1. Initialize Docker swarm and create a standalone container
2. Inspect and verify container networking

#### Step 1: Initialize Docker swarm and create a standalone container

1.1 Initialize Docker swarm using the following command:

**sudo docker swarm init**

A terminal window screenshot with a title bar 'labsuser@ip-172-31-8-102: ~'. The terminal shows the command 'sudo docker swarm init' being executed. The output indicates the swarm is initialized and the current node is a manager. It provides a token and a command to add workers. The prompt returns to the user.

```
labsuser@ip-172-31-8-102:~$ sudo docker swarm init
Swarm initialized: current node (x5dcc9yt7f49saa2fl15ac87n) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-3x5spc2pywde0q5avqjkgmb8yhzmd5ckdo9vs4ww7vnh3eu5hc-3j0fnxozakj49rx82qus8jqnc 172.31.8.102:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
labsuser@ip-172-31-8-102:~$
```

- 1.2 Run the following command to create and start a container as a detached process using the host networking driver:

```
sudo docker run --rm -d --network host --name nginx_container1 nginx
```

```
labsuser@ip-172-31-8-102:~$ sudo docker run --rm -d --network host --name nginx_container1 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
8a1e25ce7c4f: Pull complete
e78b137be355: Pull complete
39fc875bd2b2: Pull complete
035788421403: Pull complete
87c3fb37cbf2: Pull complete
c5cdd1ce752d: Pull complete
33952c599532: Pull complete
Digest: sha256:6db391d1c0cfb30588ba0bf72ea999404f2764feb0f1f196acd5867ac7efa7e
Status: Downloaded newer image for nginx:latest
658236f62c9f7cdc6fa1622f85ff850acd15f5514e98a953badb294965417832
labsuser@ip-172-31-8-102:~$
```

**Note:** The host networking driver works only on Linux hosts and is not supported on Docker Desktop for Mac or Windows Server.

- 1.3 Open the browser and navigate to <http://localhost:80/>

## Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

## Step 2: Inspect and verify container networking

- 2.1 Inspect the container to check the **NetworkMode** under the **HostConfig** using the following command:

```
sudo docker container inspect nginx_container1
```

```
labsuser@ip-172-31-8-102:~$ sudo docker container inspect nginx_container1
[
  {
    "Id": "76637ca340f65e039f693a6952732d3e3642b1363c821f5f3dec29369522355c",
    "Created": "2024-03-17T13:00:02.375855751Z",
    "Path": "/docker-entrypoint.sh",
    "Args": [
      "nginx",
      "-g",
      "daemon off;"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
```



2.3 Examine all the network interfaces using the following command:  
**ip addr show**

```
labuser@ip-172-31-8-102: ~  
labuser@ip-172-31-8-102:~$ sudo netstat -tulpn | grep :80  
tcp        0      0 0.0.0.0:80          0.0.0.0:*           LISTEN      2900/nginx: master  
tcp6       0      0 :::8080            :::*                 LISTEN      454/java  
tcp6       0      0 :::80              :::*                 LISTEN      2900/nginx: master  
labuser@ip-172-31-8-102:~$ ip addr show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000  
    link/ether 0a:ef:b4:13:31:fd brd ff:ff:ff:ff:ff:ff  
    inet 172.31.8.102/20 metric 100 brd 172.31.15.255 scope global dynamic ens5  
        valid_lft 2381sec preferred_lft 2381sec  
    inet6 fe80::8ef:b4ff:fe13:31fd/64 scope link  
        valid_lft forever preferred_lft forever  
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default  
    link/ether 02:42:17:ca:11:79 brd ff:ff:ff:ff:ff:ff  
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0  
        valid_lft forever preferred_lft forever  
labuser@ip-172-31-8-102:~$
```

2.4 Stop the container using the following command:  
**sudo docker container stop nginx\_container1**

```
labuser@ip-172-31-8-102:~$ ip addr show  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
    inet6 ::1/128 scope host  
        valid_lft forever preferred_lft forever  
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000  
    link/ether 0a:ef:b4:13:31:fd brd ff:ff:ff:ff:ff:ff  
    inet 172.31.8.102/20 metric 100 brd 172.31.15.255 scope global dynamic ens5  
        valid_lft 2381sec preferred_lft 2381sec  
    inet6 fe80::8ef:b4ff:fe13:31fd/64 scope link  
        valid_lft forever preferred_lft forever  
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default  
    link/ether 02:42:17:ca:11:79 brd ff:ff:ff:ff:ff:ff  
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0  
        valid_lft forever preferred_lft forever  
labuser@ip-172-31-8-102:~$ sudo docker container stop nginx_container1  
nginx_container1  
labuser@ip-172-31-8-102:~$
```

By following these steps, you have successfully created a host network for a standalone container to facilitate direct binding to the Docker host's network interface and ensure seamless integration and optimal network performance.