

# Docker Assignment 7

Step 1: Launched an instances for our Docker Host:

The screenshot shows the AWS EC2 Instances page. On the left sidebar, under the 'Instances' section, there is a list of instance types, launch templates, spot requests, savings plans, reserved instances, dedicated hosts, capacity reservations, and capacity manager. Under the 'Images' section, there are AMIs and AMI Catalog. Under 'Elastic Block Store', there is a 'Hostname type' dropdown set to 'Automatic'. The main table lists three instances: D5 Linux, D6 Linux, and D7 Linux. D7 Linux is selected and highlighted with a red box. The table columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, and Availability Zone. Below the table, the details for the selected instance (i-09de41c0a4e55bf48) are displayed, including its Public IPv4 address (65.2.167.30), Private IP address (172.31.42.139), Instance state (Running), and Public DNS name (ec2-65-2-167-30.ap-south-1.compute.amazonaws.com).

Step 2: Installed Docker on the Docker Host and started it:

The screenshot shows a MobaXterm terminal window titled '13.233.0.185'. The terminal session is connected to the IP address 13.233.0.185. The user is root, as indicated by the '#'. The terminal shows the following commands being run:

```
[root@ip-172-31-38-198 ~]# docker -v
Docker version 25.0.14, build 0bab007
[root@ip-172-31-38-198 ~]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-38-198 ~]#
[root@ip-172-31-38-198 ~]#
```

The 'docker -v' command and the 'service docker start' command are highlighted with red boxes.

### Step 3: Created two different networks, ‘Network-A’ and ‘Network-B’ on the Docker Host:

The screenshot shows a terminal window with several tabs at the top: WhatsApp, what is the orchestrated utility use, Instances | EC2 | ap-south-1, EC2 Instance Connect | ap-south-1, and 65.2.167.30. The main area of the terminal shows a root shell session on an EC2 instance. The user runs the command `docker network create Network-A`, which outputs a long hex string. This command and its output are highlighted with a red box. The user then runs `docker network create Network-B`, which also outputs a long hex string, also highlighted with a red box. Both commands are preceded by the prompt `[root@ip-172-31-42-139 ~]#`.

```
[root@ip-172-31-42-139 ~]# docker network create Network-A
4d30c1bc1edc25c13cc26ca9d09e1dd2b91855117c32de1ee67a6aedf45c3648
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]# docker network create Network-B
3bfd1f0b832c09e094913f426326535290ba5fbe693407f9c2e1d73de8db03af
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
```

i-09de41c0a4e55bf48 (D7 Linux )

### Step 4: Created two containers, ‘Container-A’ with 80:80 port binding using HTTPD image and ‘Container-B’ by using CentOS:8 image respectively:

The screenshot shows a terminal window with several tabs at the top: WhatsApp, what is the orchestrated utility use, Instances | EC2 | ap-south-1, EC2 Instance Connect | ap-south-1, and 65.2.167.30. The main area of the terminal shows a root shell session. The user runs `docker run -itd --name Container-A --network Network-A httpd`, which outputs a long hex string. This command and its output are highlighted with a red box. The user then runs `docker run -itd --name Container-B --network Network-B centos:8 bash`, which also outputs a long hex string, also highlighted with a red box. Both commands are preceded by the prompt `[root@ip-172-31-42-139 ~]#`.

```
[root@ip-172-31-42-139 ~]# docker run -itd --name Container-A --network Network-A httpd
23b043fd1f955dc93f0ea2401d6bf68941c23da0ced1e00909d61b24d98c7703
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]# docker run -itd --name Container-B --network Network-B centos:8 bash
7f6aff2811be118d497a9e7ff302ac87a36606fd56458658739cfec5a5660fed
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
```

i-09de41c0a4e55bf48 (D7 Linux )

## Step 5: Established the connection between both the Containers and Networks:

```
[root@ip-172-31-42-139 ~]# docker network connect Network-A Container-B
[root@ip-172-31-42-139 ~]# docker network connect Network-B Container-A
[root@ip-172-31-42-139 ~]#
```

i-09de41c0a4e55bf48 (D7 Linux)

Basically, in **Network-A**, it added the **Container-B** with a different IP address of a its same subnet:

```
"Containers": {
    "01a696ba4d82244abc7d74cccf4bc1b58b52a4f7d89d2eb25725630b9147486c": {
        "Name": "Container-A",
        "EndpointID": "4f97bcafee5d921dd2570411269c19460b429cad852a88cc5cabafdd823ead63",
        "MacAddress": "02:42:ac:16:00:02",
        "IPv4Address": "172.22.0.2/16",
        "IPv6Address": ""
    },
    "404983cc1a6dc382921017c5e3099b4cdfcb7338321f94d401ed3eb6f01457cf": {
        "Name": "Container-B",
        "EndpointID": "3e3fcada8c4a50a0789105dc54d1d9f8d2552fd728c0f6dec5e74dd7c3f2c4a",
        "MacAddress": "02:42:ac:16:00:03",
        "IPv4Address": "172.22.0.3/16",
        "IPv6Address": ""
}
```

And, in **Network-B**, it added the **Container-A** with a different IP address of a its same subnet:

```
"Containers": {
    "01a696ba4d82244abc7d74cccf4bc1b58b52a4f7d89d2eb25725630b9147486c": {
        "Name": "Container-A",
        "EndpointID": "00f5215f6ad097a69a4b8e864c5cd3ee9709a371c7d63efa22374b03ad36937e",
        "MacAddress": "02:42:ac:17:00:03",
        "IPv4Address": "172.23.0.3/16",
        "IPv6Address": ""
    },
    "404983cc1a6dc382921017c5e3099b4cdfcb7338321f94d401ed3eb6f01457cf": {
        "Name": "Container-B",
        "EndpointID": "a03fa8b9c3edc9afa156d123d3501a7c67a3bb11ea6ccd27cbd2ae74e8a3d92",
        "MacAddress": "02:42:ac:17:00:02",
        "IPv4Address": "172.23.0.2/16",
        "IPv6Address": ""
}
```

Step 5: Created an ‘index.html’ file and copied it to the deployment folder of ‘Container-A’:

The screenshot shows a terminal session on an EC2 instance. The user has created an 'index.html' file containing three heading tags: an h1, an h2, and another h2. They then used the 'docker cp' command to copy this file into a Docker container named 'Container-A'. The terminal output indicates that the file was successfully copied.

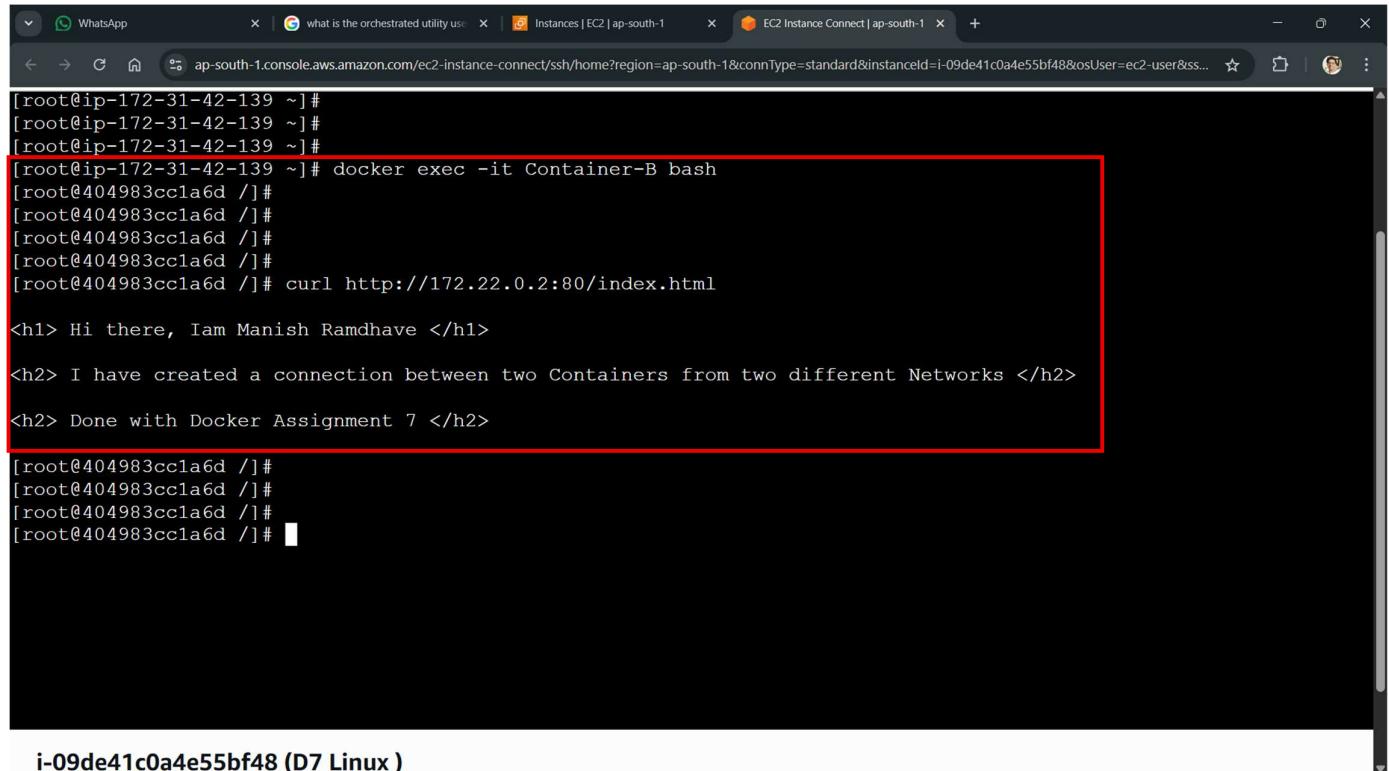
```
[root@ip-172-31-42-139 ~]# vi index.html
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]# cat index.html
<h1> Hi there, Iam Manish Ramdhave </h1>
<h2> I have created a connection between two Containers from two different Networks </h2>
<h2> Done with Docker Assignment 7 </h2>

[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]# docker cp index.html Container-A:/usr/local/apache2/htdocs
Successfully copied 2.05kB to Container-A:/usr/local/apache2/htdocs
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
```

i-09de41c0a4e55bf48 (D7 Linux )

## Results:

1. We have logged in to the **Container-B** run the ‘**index.html**’ file of the **Container-A’s** on **Port No.80** by using the **Apache HTTPD**:



```
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]#
[root@ip-172-31-42-139 ~]# docker exec -it Container-B bash
[root@404983cc1a6d /]#
[root@404983cc1a6d /]#
[root@404983cc1a6d /]#
[root@404983cc1a6d /]#
[root@404983cc1a6d /]# curl http://172.22.0.2:80/index.html

<h1> Hi there, Iam Manish Ramdhav </h1>
<h2> I have created a connection between two Containers from two different Networks </h2>
<h2> Done with Docker Assignment 7 </h2>

[root@404983cc1a6d /]#
[root@404983cc1a6d /]#
[root@404983cc1a6d /]#
[root@404983cc1a6d /]#
```

2. We have run the **Container-A index.html** file on **Port No.80** from the browser also:

