# LAB – 06

$ docker network ls
How many networks do you see in your environment? Briefly, explain their purpose.

A: 3 networks - null, host, bridge. Host is ec2 network that have it configured as docker host.

Docker installs linux bridge called docker0, then creates a network with docker0, that is called as bridge.

Null – this is a container with no interfaces. Its useful when we want to run a container with no network.

$ ifconfig
What is the Docker bridge interface called and state its IP address?

A - Ip: 172.17.0.1, docker0

$ docker container run -d --name webserver1 -p 80:80 nginx:latest

$ docker container inspect webserver1

Record the following network settings of your container.
Gateway: 172.17.0.1
IP Address: 172.17.0.3
MAC Address:  02:42:ac:11:00:03
Ports: 80, 80
NetworkID: efce27bdafc27a20080cfc9e2ee54c94e975ceeff57b1a6b23194a045f49f061

What Docker network this container lives in? How were you able to identify?

A:This lives in bridge network, am able to identify after inspecting the “networks” block.

docker container inspect webserver2

Gateway:172.17.0.1

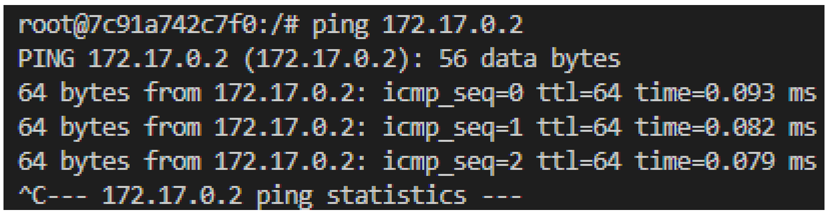
IP Address:172.17.0.2

MAC Address: 02:42:ac:11:00:02

Ports:8080

NetworkID:862353d5e24f1a3677ca1ff35168c1de39b83142b012a3cb896c2dc396cad0b6

Were you able to ping the webserver2? - yes



```
root@7c91a742c7f0:/# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: icmp_seq=0 ttl=64 time=0.093 ms
64 bytes from 172.17.0.2: icmp_seq=1 ttl=64 time=0.082 ms
64 bytes from 172.17.0.2: icmp_seq=2 ttl=64 time=0.079 ms
^C--- 172.17.0.2 ping statistics ---
```

$ docker network inspect dnsnet
The details of the dnsnet network should be displayed.
Are the Subnet and Gateway parameters being same that you had specified in earlier step?

Yes, they are same.

"Subnet": "192.168.54.0/24", "Gateway": "192.168.54.1"

What kind of network is this and specify its interface name?

A: interface name = br-6567ec392fe2 which is a bridge, name is dnsnet.

Write down the IP address and Gateway address of alpinedns1

Gateway: "192.168.54.1",

IPAddress: "192.168.54.2"

Write down the IP address and Gateway address of alpinedns2

Gateway: "192.168.54.1",

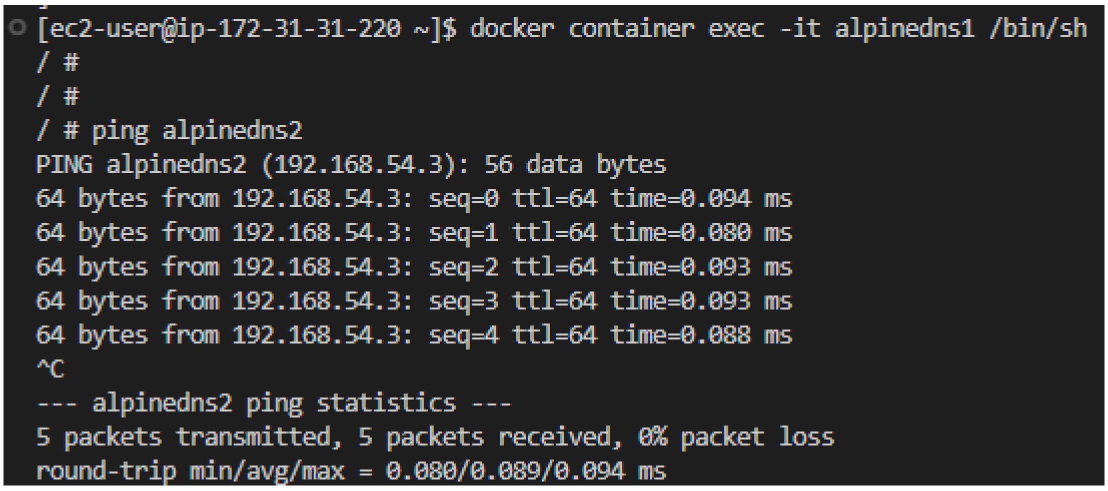
IPAddress: "192.168.54.3",

q. Are these both containers from the same subnetwork?

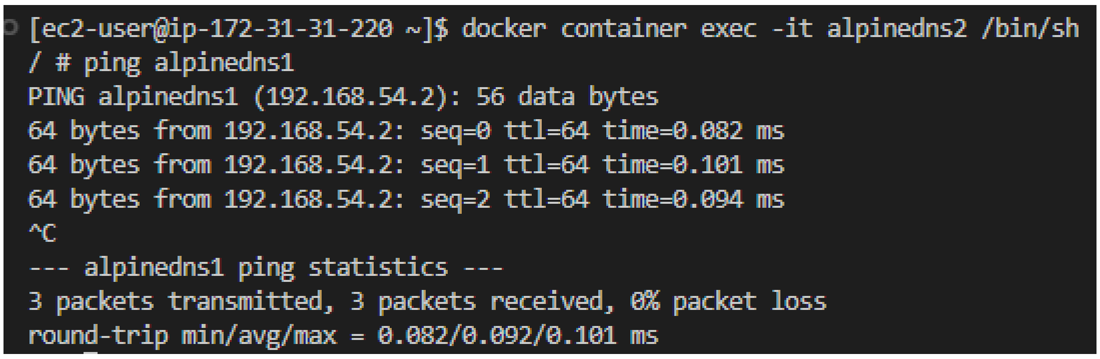
Ans: Yes, both container 'alpinedns1' and alpinedns2' belongs to same subnetwork

Are you able to ping using DNS name?

A: Yes. ping was successful.



```
[ec2-user@ip-172-31-31-220 ~]$ docker container exec -it alpinedns1 /bin/sh
/ #
/ #
/ # ping alpinedns2
PING alpinedns2 (192.168.54.3): 56 data bytes
64 bytes from 192.168.54.3: seq=0 ttl=64 time=0.094 ms
64 bytes from 192.168.54.3: seq=1 ttl=64 time=0.080 ms
64 bytes from 192.168.54.3: seq=2 ttl=64 time=0.093 ms
64 bytes from 192.168.54.3: seq=3 ttl=64 time=0.093 ms
64 bytes from 192.168.54.3: seq=4 ttl=64 time=0.088 ms
^C
--- alpinedns2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.080/0.089/0.094 ms
```



```
[ec2-user@ip-172-31-31-220 ~]$ docker container exec -it alpinedns2 /bin/sh
/ # ping alpinedns1
PING alpinedns1 (192.168.54.2): 56 data bytes
64 bytes from 192.168.54.2: seq=0 ttl=64 time=0.082 ms
64 bytes from 192.168.54.2: seq=1 ttl=64 time=0.101 ms
64 bytes from 192.168.54.2: seq=2 ttl=64 time=0.094 ms
^C
--- alpinedns1 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.082/0.092/0.101 ms
```

As we are successfully able to ping containers by their hostname from each other, we can confirm that true DNS is being used for communica�on between the containers

Q: Can we share container network between Docker hosts

A: Sure, that is possible if we use the host network rather than the bridge network.

Q: Which interface is this network tied to on your Docker host?

A: No network tied to host

What can you conclude from the output of host network configuration?
Why is subnets, interfaces or other metadata not defined in this network?

A: containers running in the "host" network mode are not affected by this information. Instead, containers operating in this mode share the same IP addresses and network interfaces as the host, enabling them to speak with each other and the host's network interfaces directly.

```
$ docker network inspect none
```
Q: Why is none network mostly empty?

A: When we examine the network configuration, it is largely empty because the "none" network mode disables networking for the container.

Q: Based on your above observation, explain the differences between the none and host networks?

A: Host - Networking is within host network , Direct access to host’s network stack, Uses IP Address of host, Low security

None -Networking is disabled, Network access is isolated, No IP Address, High security as container is completely isolated

Q:Based on the output, why there is no IP address of gateway to the container?
A: The host network is used directly by the container because it was created using the host network. Because of this, it lacks a unique IP address or gateway there.

Q:Can this container communicate with other containers or the Internet?

A: Indeed, it can communicate with other containers and the internet using the host's network, just like it can with other hosts.

```
$ docker container exec -it hostnet1 /bin/sh

/ # ifconfig
```

Q: What can you conclude about the output of the preceding command?

A: This will display any host machine network interfaces that are available. The host network interfaces are displayed because the container was started utilising the host network.

Q: why is hostnet3 container exited and is currently in a stopped state? Justify your answer.

A: Because hostnet3 attempted to launch the Nginx web server on ports already in use by hostnet2, which led to a conflict. The Nginx web server could not be started by the container, which promptly terminated.