LAB 6 – Virtualization

Q1 - `docker network ls`
- How many networks do you see in your environment? Briefly, explain their purpose.

A1 – I see 3 networks configured by default in my environment which are –

1. Bridge –This is used to make connection to the outside internet (www/external networks)
2. Host – This is used to make connection to the host(host-to-container), for managing and other tasks.
3. None –This network is used to keep the network private, used to secure the environment from any outside threats or access.

Q2 - What is the Docker bridge interface called and state its IP address?
A2 – Docker bridge interface is called `docker0` and its ip is 172.17.0.1

Q3 - What Docker network this container lives in? How were you able to identify?
A3 – This container lives in `bridge` network inside docker. We can identify from `NetworkSettings.Networks.bridge` parameter in docker inspect.

Gateway: 172.17.0.1
IPAddress: 172.17.0.2
Mac Address: 02:42:ac:11:00:02
Ports: 80
Network ID: a0e72da0c990667c016691be833e4bfbdd192aca21e829f7ed111d759a199ed6

Q4 – Were you able to ping the webserver2?
A4 – Yes

Q5 - Are the Subnet and Gateway parameters being same that you had specified in earlier step?
A5 – Yes

Q6 - What kind of network is this and specify its interface name?
A6 – It is a bridge network & `br-6a3669ad61d2` this is its interface name.

Q7 – Are these both containers from the same subnetwork?
A7 –
Alpinedns1:
IP: 192.168.54.2
Gateway: 192.168.54.1
Alpinedns2:
IP: 192.168.54.3
Gateway: 192.168.54.1

- Yes both are from same subnetwork

Q8 - Are you able to ping using DNS name?
A8 – Yes, I was able to ping alpinedns1

Q9 – How would you prove that containers re using true DNS when communicating with each other?
A9 -
Login to alpinedns1 > run `wget -O – ifconfig.net` (Prints out public dns)
Login to alpinedns2 > run `wget -O – ifconfig.net` (“)
Both showed their public dns to be `54.174.145.242`
Also both are on same network, using same subnet with which they can ping each other.

Q10 - Can we share container network between Docker hosts?
A10 – No, unless a specific feature is used(swarms) or vpn to connect hosts.

Q11 - Which interface is this network tied to on your Docker host?
A11 – Bridge is tied to `docker0`

Q12 - What can you conclude from the output of host network configuration?
A12 – That there is no information about things like gateway, ip, subnets, etc since `host` driver directly uses the host network stack and do no have any specific private network created by docker.

Q13 - Why is subnets, interfaces or other metadata not defined in this network?
A13 - The command `docker network inspect host` doesn't show information about subnets, interfaces, or other metadata because the host network is a special network that connects directly to the host's network stack, without any extra configuration. Because it's not created or managed by Docker, it doesn't have a subnet or interface defined in the same way as other Docker networks, so this information isn't displayed in the output of the command.

Q14 – Why is none network mostly empty?
A14 – Because `none` driver is essentially having no network connection i.e no network stack allocated to this type by docker.

Q15 - Based on your above observation, explain the differences between the none and host networks?
Q15 - In Docker, the "none" and "host" networks are different because "none" means the container has no network connectivity, while "host" means the container shares the host network stack. When you use "none", the container can't access the network, and when you use "host", the container can access the same network interfaces as the host. Therefore, "none" and "host" are opposite in terms of network access.

Q16 - Based on the output, why there is no IP address of gateway to the container? Can this container
A16 – Since it uses the host network stack so that’s why the docker network inspect command shows no ip, subnet or gateway since docker did not allocate any

Q17 - Can this container communicate with other containers or the Internet?
A17 – Yes, containers running in host network mode can communicate with other containers and the Internet, as they use the host's networking stack directly.