DOCKER NETWORKING COMMANDS

TEAM: 121AD0012, 121AD0014, 121AD0019, 121AD0020

What is networking in Docker?

Networking is used for communication between two or more docker containers. It provides various network drivers to manage container communication efficiently. Docker's networking options make it highly flexible and capable of supporting simple local setups to complex multi-host or cloud environments.

Install Docker in Ubuntu

Use the following commands in the terminal to first install Docker.

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
sudo docker run hello-world
```

Get more information on Docker containers

sudo docker image inspect: get detailed information about an image.

sudo docker images: Lists all the Docker images available on your system.

sudo docker images -a: Lists all Docker images including intermediate images or dangling images

sudo docker image ls: Same as docker images, lists the images stored locally.

sudo docker container ls: Lists all **running** Docker containers.

sudo docker image ls -a: Lists all images, including intermediary layers that are not assigned to a repository.

sudo docker ps: Lists **running** Docker containers

sudo docker ps -a: Lists **all** containers, including those that are stopped.

sudo docker inspect --format '{{.Config.Image}}' ContainerName --> gives ImageName : Returns the image name associated with a specific container.

These commands are fundamental for managing Docker images and containers, allowing you to inspect, list, and monitor various aspects of your Docker environment.

Running a created container and opening its terminal or shell

```
sudo docker start c1
sudo exec -it c1 /bin/sh
sudo docker stop c1
sudo docker inspect c1
```

Default network in Docker

Create a container and run it. See that the default network is always set to ”bridge”.

```
sudo docker run -it --name c1 alpine
sudo docker network inspect bridge
```

Network Creation With No Masquerading

Creates a new Docker network (mynet) with IP masquerading disabled. Next “inspect” command shows detailed information about the newly created mynet network. “ls” lists all Docker networks on the system.

```
sudo docker network create --opt com.docker.network.bridge.enable_ ip_ masquerade=false
mynet
sudo docker network inspect mynet
sudo docker network ls
```

By disabling IP masquerading, containers connected to mynet won’t be able to access the internet but can still communicate within the network. This is useful for isolated or secure environments.

To connect a container to more than one network, we use the following command:

```
sudo docker network connect mynet ContainerName
```

Creating a container with custom network

```
sudo docker run -it --name c2 --network=mynet alpine
sudo docker start c1
#> ping c1 --> in c2
#> ping www.google.com --> in c2
#> exit --> in c2
sudo docker stop c1
```

This runs a new container named c2 based on the alpine image and attaches it to the custom mynet network. Second command starts an existing container named c1 (assuming it was stopped or created earlier). The "ping" command tests whether the two containers (connected to the same mynet network) can communicate with each other.
Internet connectivity (www.google.com) should not work, due to IP masquerading being disabled.

Network creation with no ICC:

Here we create a new Docker network named mynet2 with inter-container communication disabled. We now inspect its options. Then we run a new container and connect it to our network. Start the containers and check the connectivity. We find that the inter container connectivity is disabled while the internet connectivity is still intact.

```
sudo docker network create --opt com.docker.network.bridge.enable_ icc=false mynet2
sudo docker network inspect mynet2
sudo docker run -it --name c3 --network=mynet2 alpine
sudo docker start c1
#> ping c1 --> in c3
#> ping www.google.com --> in c3
#> exit --> in c3
sudo docker stop c1
```

Custom Subnet

Custom subnets in Docker allow you to define specific IP address ranges for your Docker networks. This can be useful for managing IP addresses and ensuring containers have predictable networking configurations. Ensure that the subnet you choose does not overlap with other networks (including the host's network) to avoid IP conflicts. Also be mindful of the Docker default bridge network (usually 172.17.0.0/16) to prevent any address conflicts.

```
sudo docker network create --driver=bridge --subnet=172.18.0.0/16 mynet_ updated1
sudo docker network create --driver=bridge --subnet=172.19.0.0/16 --opt
com.docker.network.bridge.enable_ ip_ masquerade=true mynet_ updated2
sudo docker network inspect mynet_ updated1
sudo docker network inspect mynet_ updated2
```

Custom Gateways

When running multiple services that need to communicate but also require external access through a specific IP or service, defining a custom gateway can streamline this process. If you're integrating Docker containers into an existing network infrastructure, using a custom gateway can ensure that traffic is routed correctly. Ensure the custom gateway address is within the specified subnet and not used by any other device to avoid conflicts. Be cautious with routing and firewall settings.

```
sudo docker network create --driver=bridge --subnet=172.18.0.0/16 --gateway=172.18.0.1 mynet_custom_gateway
sudo docker network inspect mynet_custom_gateway
```

Other options in Custom Networks

com.docker.network.bridge.name	*Interface name to use when creating the Linux bridge.*
com.docker.network.bridge.enable_ip_masquerade	*Enable IP masquerading.*
com.docker.network.bridge.enable_icc	*Enable or Disable inter-container connectivity.*
com.docker.network.bridge.host_binding_ipv4	*Default IP when binding container ports.*
com.docker.network.driver.mtu	*Set containers network Maximum Transmission Unit (MTU).*
com.docker.network.container_iface_prefix	*Set a custom prefix for container interfaces.*

NOTE:

When creating a new network, you use the "--driver" option to specify the network driver. The "--network" option, on the other hand, is used when you want to connect a container to an existing network.