Container Linking

Prepared by:
Ms. Avita Katal
Assistant Professor (SG)
School of Computer Science
UPES, Dehradun

Container Linking

There are times during the development of our application when we need two containers to be able to communicate with each other. It might be possible that the services of both containers are dependent on each other. This can be done with the help of **Container Linking.**

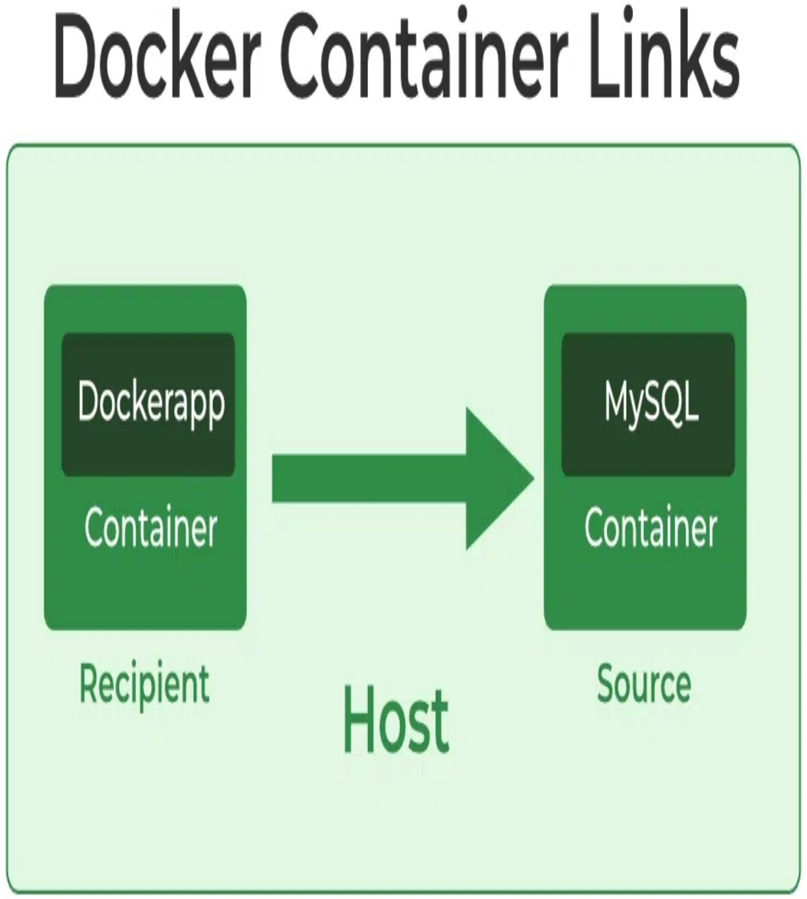
Previously the containers were used by using the "–link" flag but that has now become deprecated and is considered a legacy command

Connect with the Linking System

There are two ways of linking the containers

- The default way
- User-defined way

To understand the formation of a custom network between two containers we need to understand how docker assigns the network automatically.



Default Way

Once we install docker and create a container a default bridged network is assigned to docker, by the name of Docker0. The IP is in the range of 172.17.0.0/16 (where 172.17.0.1 is assigned to the interface)

Now the containers that we will create will get their IPs in the range of 172.17.0.2/16.

Step 1: Create two new containers, webcon, and dbcon

```
$ docker run -it --name webcon -d httpd
$ docker run -it --name dbcon  -e MYSQL_ROOT_PASSWORD=1234 -d mysql
```

You can use any image, we'll be using **MySQL** and HTTPD images in our case.

Step 2: Check the IPs of the new containers.

```
$ docker network inspect bridge
```

With the help of these IPs, the docker host establishes a connection with the containers.

Step 3: Get inside the webcon container and try to ping the dbcon container, if you get a response back this means that the default connection is established.

```
$ docker container exec -it webcon /bin/bash
```

(to get into the webcon container)

```
$ ping "172.17.0.3"
```

(ping the dbcon container)

User Defined Way

Step 1: Create a custom bridge network.

```
$ docker network create <bridge_name>
```

(This will create a bridge with custom subnet and gateway)

We can also give our own subnet and gateway.

```
$ docker network create --subnet <your_subnet> --gateway <Your_gateway> bridgename
```

Step 2: Verify if your network has been created or not.

```
$ docker network ls
```

Step 3: Associate or link the two containers on the network that you just created by using the "–net" flag.

```
$ docker run --name <container_name>
--net=<custom_net>
-d <image_name>
```

We have used httpd and Alpine images for our containers.

Step 4: Get inside the webnew container(IP- 10.7.0.10) and ping the alpine container(IP- 10.7.0.2)

```
$ docker exec -it webnew /bin/bash
$ ping "10.7.0.2" (inside the webnew container)
```

If you start receiving the packets from the Alpine container then you have successfully established a connection between both containers using your own OUR-NET network. So this is how you can create your own custom bridged network which allows you to establish a connection between your container.