->How containers communicate with each other with examples?

In This Tutorial we are taking about how two docker communicate with each other.

**Docker Network-**

There are five types of networking:-

1. Bridge Network

2) None Network

3) Host Network

4) Overlay Network

5) Macvlan Network

**Bridge Network:-**

Bridhe network is default network that comes with all other docker installation. It is also known as docker0 network. If not mentioned otherwise, all docker containers are created with docker0 network.

**None Network:-**

None network is generally known as container specific network. A container can be attached to none network. This is utilised for internal communication between containers being isolated to outside network.

**Host Network:-**

The host network adds a container on the host’s network stack.. As far as the network is concerned, there is no isolation between the host machine and the container. For instance, if you run a container that runs a web server on port 80 using host networking, the web server is available on port 80 of the host machine.

**Overlay Network:-**

Creates an internal private network that spans across all the nodes participating in the swarm cluster. So, Overlay networks facilitate communication between a swarm service and a standalone container, or between two standalone containers on different Docker Daemons.

**Macvlan Network:-**

Allows you to assign a MAC address to a container, making it appear as a physical device on your network. Then, the Docker daemon routes traffic to containers by their MAC addresses. Macvlan driver is the best choice when you are expected to be directly connected to the physical network, rather than routed through the Docker host’s network stack.

**Hands On:-**

Let us understand container linking and port exposure by an example.

We will link both apache2 and mysql-server containers using docker command ine tools.

Step:-1 First we should need the mysql server iamge

$ docker pull mysql/mysql-server:5.6

Text

Description automatically generated

**Step:-2**

Then we will launch a container from this image as

$ docker run -p 3306:3306 –-name mysql -e MYSQL\_ROOT\_PASSWORD=admin12345 -d mysql/mysql-server:5.6

In above command -p argument provides the host port number and –-name argument gives name We are mapping 3306 port of host with 3306 port inside the container. All traffic to and from mysql server will routed from these ports. Make sure ports on host are available to be used.

Text

Description automatically generated

**Step:-3**

After image and docker container are created them running or not

By using this command

$ docker ps

$ docker images

Text

Description automatically generated

**Step:-4**

Now lets pull apache2 server with php5 image as,

$ docker pull alexcheng/apache2-php5

Text

Description automatically generated

**Step:-5**

Now we should link the both mysql container and as well as apapche2-php5 containers

$ docker run -p 80:80 –name apache2 --link mysql docker pull alexcheng/apache2-php5

Text

Description automatically generated

In above command, **we have launched a container naming apache2 whose port 80 is mapped with 80 port of the host and this container is linked with our database container naming mysql by –link**

**Step:-6**

Check networks of both networks

$ docker inspect apache2

**Text

Description automatically generated**

$ docker inspect mysql

Text

Description automatically generated

**Step:-7**

We check this connectivity by,

$ docker exec -ti apache2 bash

$ ping mysql

Text

Description automatically generated

And,

$ docker exec -it mysql bash

$ ping apache2

If we get response for both the pings, then docker containers have been linked properly.