**Docker Network**

Default behaviour – bridged

Docker uses internal bridged network by default.

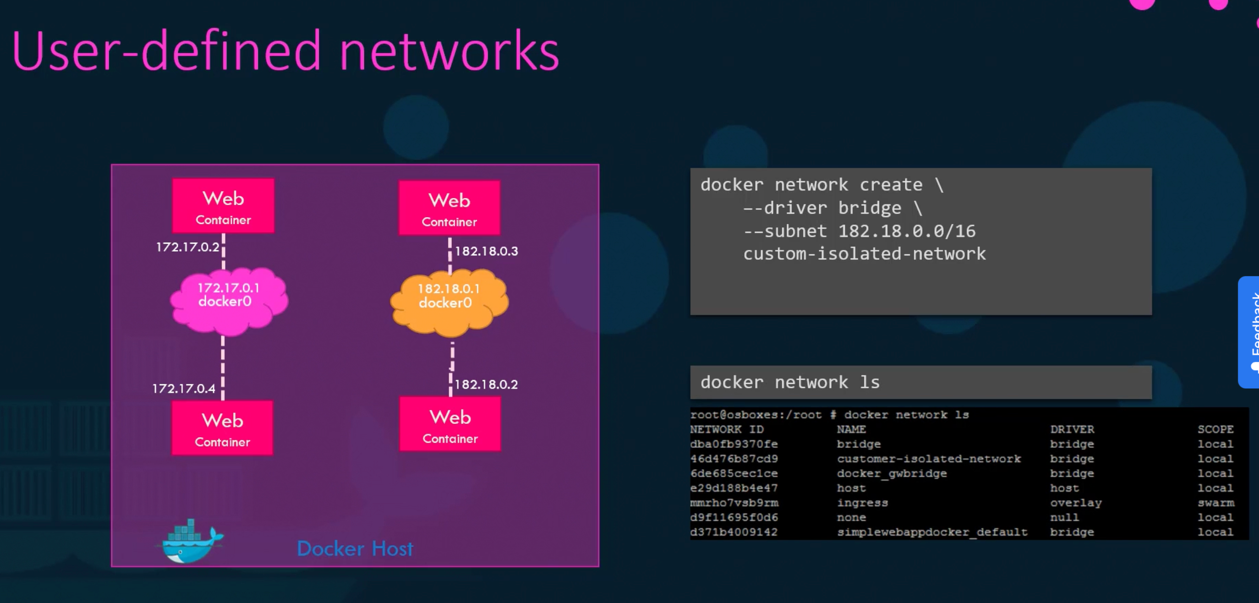
The default bridged network ip is 172.17.0.1

All the containers created on the system will be connected to this bridged network. So, all these containers will be able to access each other.

Eg:- if we have 4 containers each containers will get a different ip [172.17.0.2, 172.17.0.3, 172.17.0.4, 172.17.0.5]

Add additional networks

If one needs to isolate and group the containers, one can create new networks and map the network while creating the container as below.



To check the list of docker networks

docker network ls

to inspect and get the network details

docker network inspect bridge

to manually assign a user-created network to the container

docker run -d --name=alpine-2 --network=none alpine

to create a network

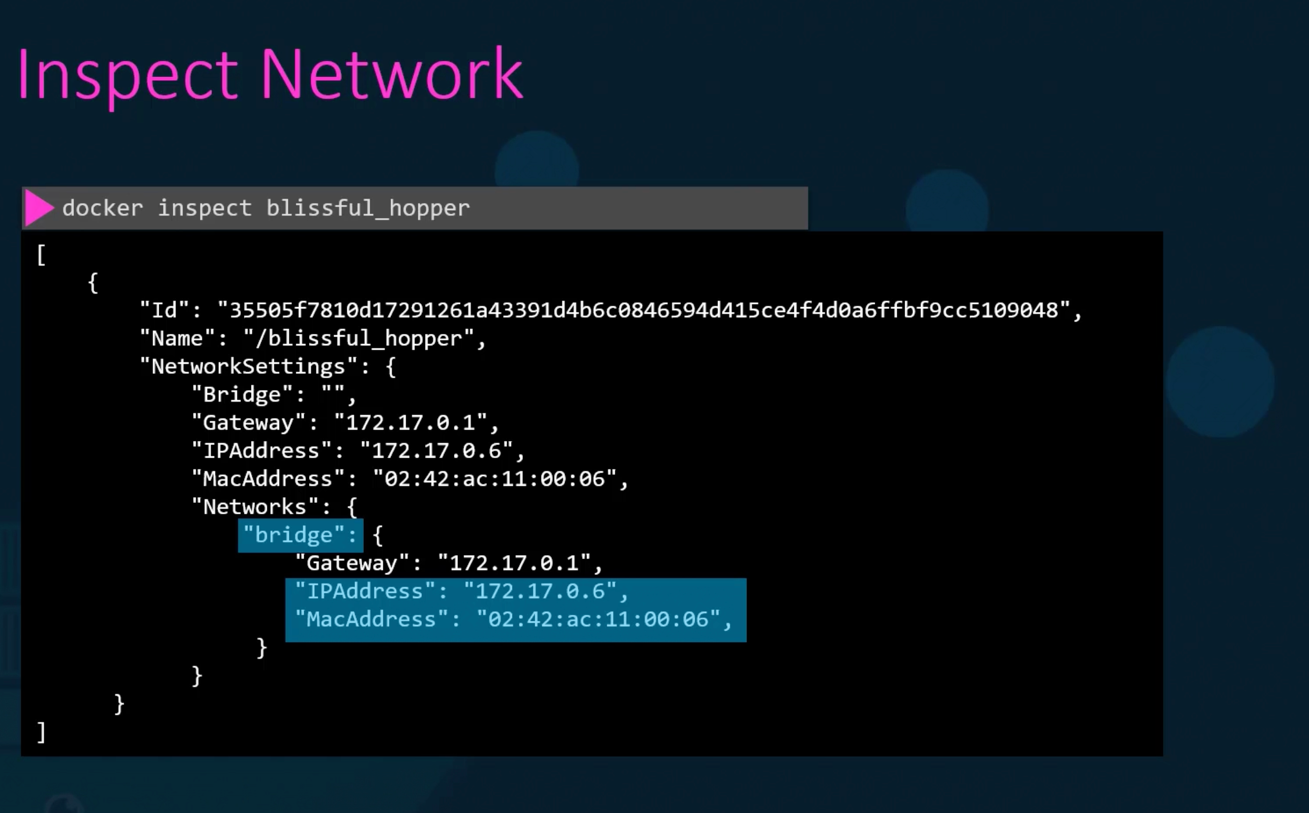
docker network create --driver bridge --subnet 182.18.0.1/24 --gateway 182.18.0.1 wp-mysql-network

assign a network

docker run -d --name=mysql-db --network=wp-mysql-network -e MYSQL\_ROOT\_PASSWORD=db\_pass123 mysql:5.6

to inspect a container to see which network it uses along with all other details

docker inspect <container-name>

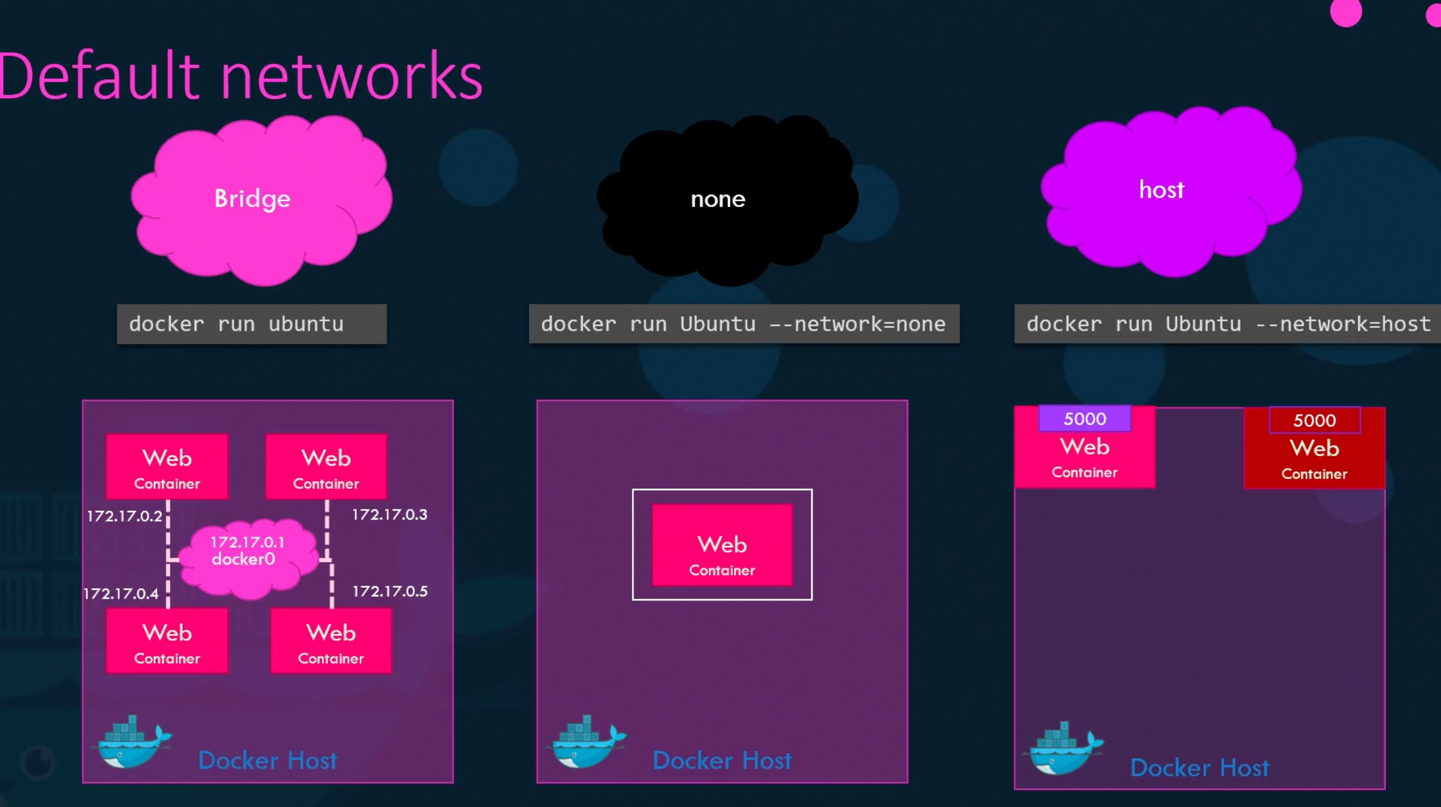


**QN:** Deploy a web application named webapp using the kodekloud/simple-webapp-mysql image. Expose the port to 38080 on the host.  
  
The application makes use of two environment variable:  
1: DB\_Host with the value mysql-db.  
2: DB\_Password with the value db\_pass123.  
Make sure to attach it to the newly created network called wp-mysql-network.

Also make sure to link the MySQL and the webapp container.

**ANS:** docker run -d --name=webapp --network=wp-mysql-network --link mysql -e MYSQL\_DB\_PASSWORD=db\_pass123 -e DB\_Host=mysql-db kodekloud/simple-webapp-mysql

**Other types of networks**

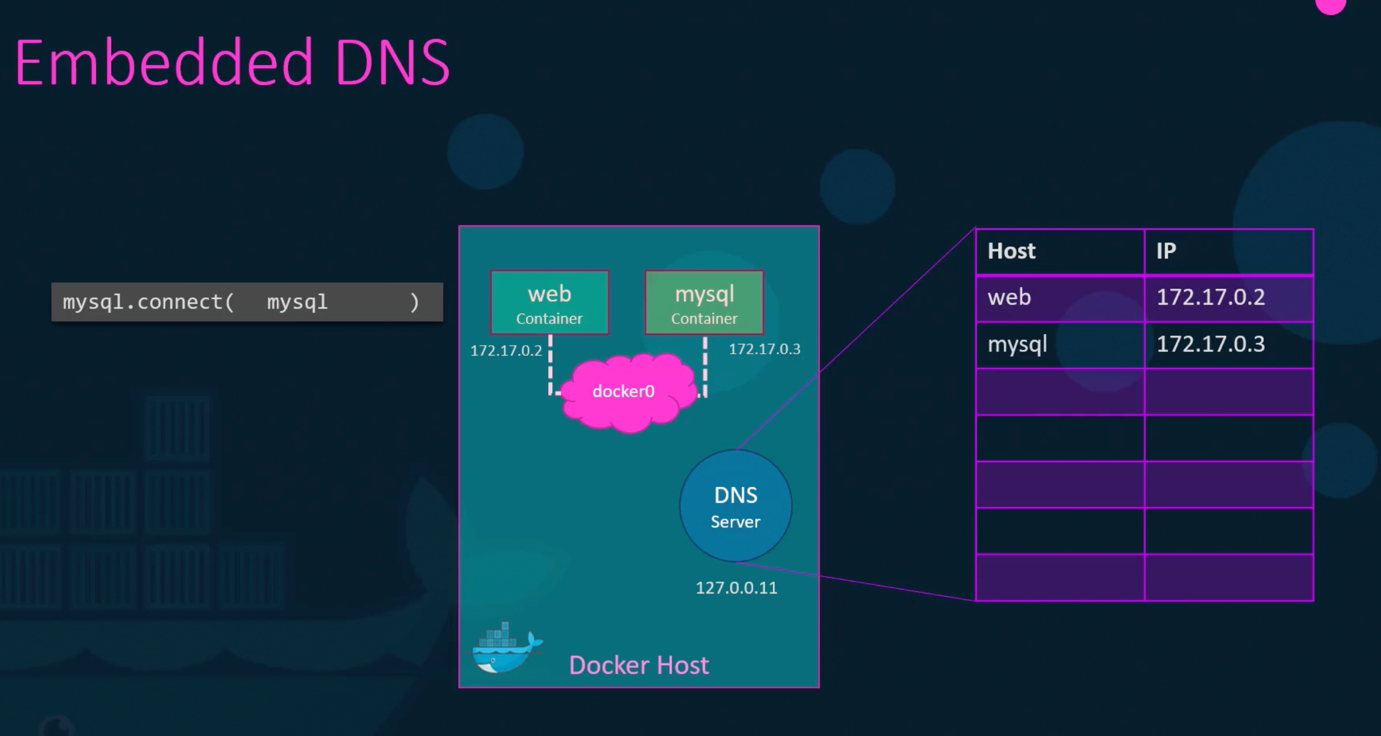


Bridged network – default one

Host network – connect to the post in the host. Once if a container utilizes a port, other containers ill not be able to utilize it.

None – the container is not attached to any network. It runs in an isolated network and cannot connect with other networks.

Embedded DNS



Docker resolves itself using the container name. So, instead of ip to connect to diff. container use the name instead of ip.

Docker maintains a dns server at 127.0.0.11. it’ll have a list of containers and the associated Ips to resolve itself.

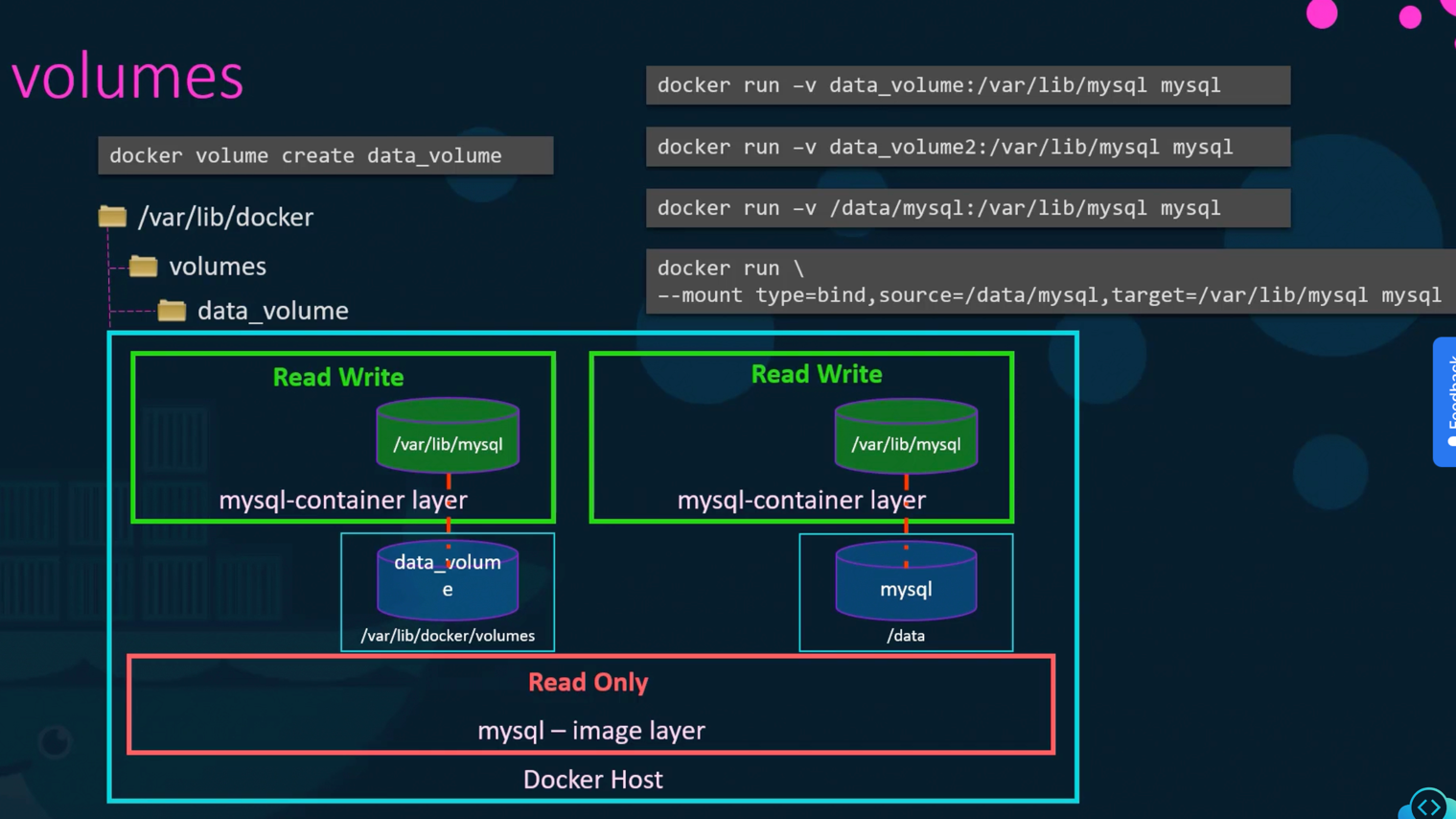
So, mysql.connect(mysql) from web server will know which mysql ip to connect within the docker.

**Docker Storage**

Volume mount:

Within docker image we can’t update source code and re-run the container again.

But it you try to update the source code inside docker image, docker creates a copy of the source code in read-write mode inside the container. The main docker image remains unaffected until a docker build runs.



mounts the host to docker container

use the –mount command line arg with key-value options.

/var/lib/mysql -> this is where the mysql is created inside the docker container.

To connect to mysql and execute a command in docker

docker exec mysql-db mysql -pdb\_pass123 -e 'use foo; select \* from myTable'

when you do a mount to a directory inside the host, the data remains persisted in the host instead of the container. So even if the container is deleted the data is not lost and can be retrieved back when a new container is created.

Bind Mount

