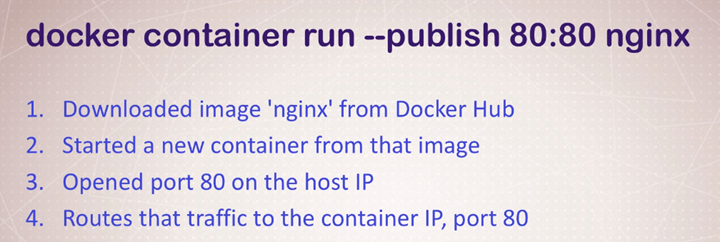
1. Docker version
2. docker info
3. docker – all commands
4. docker container run --publish 81:80 nginx [ **81: host; 80: container**]
5. docker container run --publish 81:80 *--detach* nginx [**HELPS Container to Run in Background**]
6. docker container ls [**list outs all the available containers**]
7. docker container stop <Container ID> [**Stops the Container**]
8. docker container run --publish 81:80 --detach *--name webhost* nginx [**Docker Container with name**]
9. docker container logs *webhost* [**prints the log by the container name**]
10. docker container rm <25e4 2a901 6296 94c4 7bb81 5d48> [**Delete Multiple containers at a time; but doesn’t remove the running container**]
11. docker container rm -f <25e4> [**Removes running container forcefully**]
12. docker run --name mongo -d *mongo* [**creates a process in docker**]
13. docker top *mongo* [**returns process id of the process running inside docker**]
14. docker stop *mongo* [ **kills the process resides in docker]**
15. docker image ls [**All the images in a docker container**]
16. docker container inspect *<webhost1>* [**Details of one container config**]
17. docker container stats –help
18. docker container stats –all [**Live Status of all containers**]

GETTING INSIDE THE Container -> Opening a BASH Without SSH; Different Linux distros…….

1. docker container run -it --name proxy nginx bash [**-i: works as SSH; -t: Holds the session**]: [ **Opens a bash**]
2. root@ecc1431cbf87:/# ls -al [ **login as root into the container: ls -al to see the directory]**
3. root@ecc1431cbf87:/# exit [ **brings you out of the bash: CONTAINER IS STOPPED -> docker container ls: No entry]**
4. PS C:\Users\soham> docker container run -it --name ubuntu ubuntu [ **Install Ubuntu Full]**
5. [**UPDATE**] **apt-get update**
6. [**Install**] apt-get install -y curl
7. docker container start -ai *ubuntu* OR docker start <e78f92e7a645> [ **Restart a closed container]**
8. *docker run --detach --name=****test-mysql*** *--env="MYSQL\_RANDOM\_ROOT\_PASSWORD=true" mysql*  docker container exec -it **test-mysql** bash [**Exactly same as run -it, but it starts an existing container, never creates a new one]**
9. install Another Distro:
10. docker pull alpine [ **installs Alpine as an image**]
11. [**Run alpine bash**] docker container run -it --name *alpineproxy* alpine bash [ **ERROR: as alpine is a small version; doesn’t have bash on the path**]
12. docker image ls -a

REPOSITORY TAG IMAGE ID CREATED SIZE

ubuntu latest 1e4467b07108 5 hours ago 73.9MB

mysql latest e3fcc9e1cc04 2 days ago 544MB

httpd latest 9d2a0c6e5b57 2 days ago 166MB

nginx latest 8cf1bfb43ff5 2 days ago 132MB

mongo latest 6d11486a97a7 2 weeks ago 388MB

alpine latest a24bb4013296 7 weeks ago 5.57MB

1. [ **Alpine has SH instead of Bash**] docker container run -it --name alpineproxy1 alpine sh

**Docker Networking (-p: expose port)**

1. docker container run -p 82:80 --name webhost -d nginx

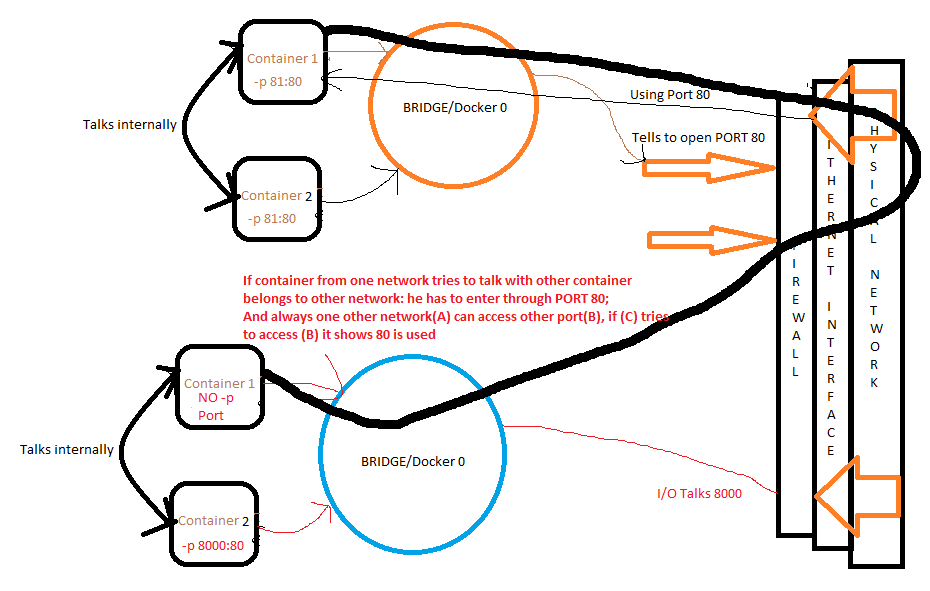
docker container port webhost [ **Available ports: 80/tcp -> 0.0.0.0:82**]

1. DEFAULT container doesn’t have same IP as Host (Home Network)

**[BATTRIES INCLUDED, BUT REMOVABLE]**

docker container inspect --format '{{ .NetworkSettings.IPAddress }}' webhost

[**container IP: 172.17.0.3; home IP: 192.168.0.104**]

1. Port inter communication 
2. [**List all networks**] docker network ls

NETWORK ID NAME DRIVER SCOPE

43417b85d5de bridge bridge local (**CONNECTED WITH CONTAINERS)**

6fd9e0430513 host host local (**LOCAL MACHINE**)

091b669dfdb5 none null local (**A NETWORK DEVICE THAT IS NOT ATTACHED TO NETWORK**)

1. [**inspect bridge & connected containers**] docker network inspect bridge

[

"Containers": {

"63279293e34a7cb2738414e60de14b4a6adcb30ce4d5a0ac13b26c688c15ec4a": {

"Name": "test-mysql",

"EndpointID": "e6d01d24b1306ef57e2a25ed66d564a272f49ae78c986fbf74ec8f05316a9023",

"MacAddress": "02:42:ac:11:00:02",

"IPv4Address": "172.17.0.2/16",

"IPv6Address": ""

},

"6797d5f03fb79ec7b02f4dba542ff631052a323551083b733175b5e3cacd768b": {

"Name": "webhost",

"EndpointID": "a12daf838aa2e19015afa89adbd6c349631dedbd4b94dd54a0ad76036dd69a47",

"MacAddress": "02:42:ac:11:00:03",

"IPv4Address": "172.17.0.3/16",

"IPv6Address": ""

}

},

"Options": {

"com.docker.network.bridge.default\_bridge": "true",

"com.docker.network.bridge.enable\_icc": "true",

"com.docker.network.bridge.enable\_ip\_masquerade": "true",

"com.docker.network.bridge.host\_binding\_ipv4": "0.0.0.0",

"com.docker.network.bridge.name": "docker0",

"com.docker.network.driver.mtu": "1500"

},

"Labels": {}

}

]

1. [**CREATE a new docker network/Bridge 0**] docker network create *NEW\_Bridge*

NETWORK ID NAME DRIVER SCOPE

bfd85baf685e NEW\_Bridge bridge local

43417b85d5de bridge bridge local

6fd9e0430513 host host local

091b669dfdb5 none null local

1. [**Create a container and assign it to newly created Bridge 0]**

docker container run -d --name new\_nginx --network *NEW\_Bridge* nginx

1. [ **If we do not use default bridge it has some special feature; where we can directly communicate with 2 containers despite of IP**]

docker container run -d --name my\_ngnix --network *NEW\_Bridge* nginx

1. [**ping the new container]**

docker container exec -it webhost20 ping NEW\_Bridge

1. [**Default bridge doesn’t have a build in DNS in it, we have to use --link list**]