**DOCKER COMMANDS FOR CONTAINER INSTALL:**

**docker container run -d -p 3306:3306 --name db -MYSQL\_RANDOM\_ROOT\_PASSWORD=yes mysql**

**O/P:**

Unable to find image 'mysql:latest' locally

latest: Pulling from library/mysql

6d28e14ab8c8: Pull complete

dda15103a86a: Pull complete

55971d75ab8c: Pull complete

f1d4ea32020b: Pull complete

61420072af91: Pull complete

05c10e6ccca5: Pull complete

7e0306b13322: Pull complete

900b113c001e: Pull complete

06cd07c30bf4: Pull complete

df0d65aee5aa: Pull complete

108d207bdce2: Pull complete

b33faea3a1af: Pull complete

Digest: sha256:230d501a0c971221aef647661b331c56587fc5bd4a465dfa132c4d2b45835163

Status: Downloaded newer image for mysql:latest

07632beb265f8126e330d1520907d4a9115e60eaeb85229567865fea10b96503

**docker container log db**

**O/P:**

2020-03-04 10:55:13+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.19-1debian9 started.

2020-03-04 10:55:13+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'

2020-03-04 10:55:13+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 8.0.19-1debian9 started.

2020-03-04 10:55:14+00:00 [Note] [Entrypoint]: Initializing database files

2020-03-04T10:55:14.032579Z 0 [Warning] [MY-011070] [Server] 'Disabling symbolic links using --skip-symbolic-links (or equivalent) is the default. Consider not using this option as it' is deprecated and will be removed in a future release.

2020-03-04T10:55:14.032680Z 0 [System] [MY-013169] [Server] /usr/sbin/mysqld (mysqld 8.0.19) initializing of server in progress as process 46

2020-03-04T10:55:17.557626Z 5 [Warning] [MY-010453] [Server] root@localhost is created with an empty password ! Please consider switching off the --initialize-insecure option.

2020-03-04 10:55:21+00:00 [Note] [Entrypoint]: Database files initialized

2020-03-04 10:55:21+00:00 [Note] [Entrypoint]: Starting temporary server

2020-03-04T10:55:21.393425Z 0 [Warning] [MY-011070] [Server] 'Disabling symbolic links using --skip-symbolic-links (or equivalent) is the default. Consider not using this option as it' is deprecated and will be removed in a future release.

2020-03-04T10:55:21.393554Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.19) starting as process 96

2020-03-04T10:55:22.030602Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.

2020-03-04T10:55:22.037231Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.

2020-03-04T10:55:22.070981Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.19' socket: '/var/run/mysqld/mysqld.sock' port: 0 MySQL Community Server - GPL.

2020-03-04 10:55:22+00:00 [Note] [Entrypoint]: Temporary server started.

2020-03-04T10:55:22.101784Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: '/var/run/mysqld/mysqlx.sock'

Warning: Unable to load '/usr/share/zoneinfo/iso3166.tab' as time zone. Skipping it.

Warning: Unable to load '/usr/share/zoneinfo/leap-seconds.list' as time zone. Skipping it.

Warning: Unable to load '/usr/share/zoneinfo/zone.tab' as time zone. Skipping it.

Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.

2020-03-04 10:55:26+00:00 [Note] [Entrypoint]: GENERATED ROOT PASSWORD: IegohDoaBi0aeshongee0oujohkeenae

2020-03-04 10:55:26+00:00 [Note] [Entrypoint]: Stopping temporary server

2020-03-04T10:55:26.665647Z 10 [System] [MY-013172] [Server] Received SHUTDOWN from user root. Shutting down mysqld (Version: 8.0.19).

2020-03-04T10:55:28.141450Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.19) MySQL Community Server - GPL.

2020-03-04 10:55:28+00:00 [Note] [Entrypoint]: Temporary server stopped

2020-03-04 10:55:28+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.

2020-03-04T10:55:29.001319Z 0 [Warning] [MY-011070] [Server] 'Disabling symbolic links using --skip-symbolic-links (or equivalent) is the default. Consider not using this option as it' is deprecated and will be removed in a future release.

2020-03-04T10:55:29.001455Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.19) starting as process 1

2020-03-04T10:55:29.713761Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.

2020-03-04T10:55:29.718838Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.

2020-03-04T10:55:29.743152Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.19' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.

2020-03-04T10:55:29.848624Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: '/var/run/mysqld/mysqlx.sock' bind-address: '::' port: 33060

**docker container run -d --name proxy -p 80:80 nginx**

**docker container run -d --name webserver -p 8080:80 httpd**

**O/P:**

Unable to find image 'httpd:latest' locally

latest: Pulling from library/httpd

68ced04f60ab: Already exists

35d35f1e0dc9: Pull complete

8a918bf0ae55: Pull complete

d7b9f2dbc195: Pull complete

d56c468bde81: Pull completeDigest: sha256:946c54069130dbf136903fe658fe7d113bd8db8004de31282e20b262a3e106fb

Status: Downloaded newer image for httpd:latest

25283e3886e8cc4de778f47a8329b4c59cd7d6c77648dd17206d53cec8b6f5d5

**docker ps (as we can see internally ngnix and https runs on port 80 but in host they are published in port 80 and port 8080 respectively i.e left side is host port below)**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d88d3db82b45 nginx "nginx -g 'daemon of…" 13 hours ago Up 10 seconds 0.0.0.0:80->80/tcp proxy

25283e3886e8 httpd "httpd-foreground" 14 hours ago Up 10 minutes 0.0.0.0:8080->80/tcp webserver

07632beb265f mysql "docker-entrypoint.s…" 16 hours ago Up 2 hours 0.0.0.0:3306->3306/tcp, 33060/tcp db

**docker container ls**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**d88d3db82b45** nginx "nginx -g 'daemon of…" 13 hours ago Up 22 seconds 0.0.0.0:80->80/tcp proxy

**25283e3886e8** httpd "httpd-foreground" 14 hours ago Up 10 minutes 0.0.0.0:8080->80/tcp webserver

**07632beb265f** mysql "docker-entrypoint.s…" 16 hours ago Up 2 hours 0.0.0.0:3306->3306/tcp, 33060/tcp db

Now to check if nginix and apache running in ports properly:

**curl localhost**

<!DOCTYPE html>

<html>

<head>

<title>Welcome to nginx!</title>

<style>

body {

width: 35em;

margin: 0 auto;

font-family: Tahoma, Verdana, Arial, sans-serif;

}

</style>

</head>

<body>

<h1>Welcome to nginx!</h1>

<p>If you see this page, the nginx web server is successfully installed and

working. Further configuration is required.</p>

<p>For online documentation and support please refer to

<a href="http://nginx.org/">nginx.org</a>.<br/>

Commercial support is available at

<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>

</body>

</html>

**curl localhost:8080**

<html><body><h1>It works!</h1></body></html>

**Now stopping all of them to remove them.**

**docker container stop d88d3db82b45**

d88d3db82b45

**docker container stop 25283e3886e8**

25283e3886e8

**docker container stop 07632beb265f**

07632beb265f

**docker container ls -a**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

**d88d3db82b45** nginx "nginx -g 'daemon of…" 14 hours ago Exited (0) About a minute ago proxy

**25283e3886e8** httpd "httpd-foreground" 14 hours ago Exited (0) About a minute ago webserver

**07632beb265f**  mysql "docker-entrypoint.s…" 16 hours ago Exited (0) 44 seconds ago db

**REMOVING ALL THE CONTAINERS:-**

**docker container rm d88d3db82b45 25283e3886e8 07632beb265f**

d88d3db82b45

25283e3886e8

07632beb265f

**docker container top mysql**

PID USER TIME COMMAND

3871 999 0:01 mysqld

**docker container top nginx**

PID USER TIME COMMAND

3795 root 0:00 nginx: master process nginx -g daemon off;

3834 101 0:00 nginx: worker process

**docker container inspect mysql (this gives mysql container parameters )**

**docker container stats mysql (gives each second usage of container memory and cpu)**

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

1c79e0a10e18 mysql 1.25% 373.4MiB / 1.92GiB 18.99% 866B / 0B 5.04MB / 337MB 38

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

1c79e0a10e18 mysql 1.25% 373.4MiB / 1.92GiB 18.99% 866B / 0B 5.04MB / 337MB 38

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

1c79e0a10e18 mysql 1.17% 373.4MiB / 1.92GiB 18.99% 866B / 0B 5.04MB / 337MB 38

Below we login into ngnix container (**7d5626a19ba6**) with bash shell , when we run ls -al we actually run it inside the shell of nginx using -it (i means std/io and t means tty terminal):-

**docker container run -it --name proxy nginx bash**

root@**7d5626a19ba6**:/# ls -al

total 72

drwxr-xr-x 1 root root 4096 Mar 4 14:21 .

drwxr-xr-x 1 root root 4096 Mar 4 14:21 ..

-rwxr-xr-x 1 root root 0 Mar 4 14:21 .dockerenv

drwxr-xr-x 2 root root 4096 Feb 24 00:00 bin

drwxr-xr-x 2 root root 4096 Feb 1 17:09 boot

drwxr-xr-x 5 root root 360 Mar 4 14:21 dev

drwxr-xr-x 1 root root 4096 Mar 4 14:21 etc

drwxr-xr-x 2 root root 4096 Feb 1 17:09 home

drwxr-xr-x 1 root root 4096 Feb 26 20:02 lib

drwxr-xr-x 2 root root 4096 Feb 24 00:00 lib64

drwxr-xr-x 2 root root 4096 Feb 24 00:00 media

drwxr-xr-x 2 root root 4096 Feb 24 00:00 mnt

drwxr-xr-x 2 root root 4096 Feb 24 00:00 opt

dr-xr-xr-x 131 root root 0 Mar 4 14:21 proc

drwx------ 2 root root 4096 Feb 24 00:00 root

drwxr-xr-x 3 root root 4096 Feb 24 00:00 run

drwxr-xr-x 2 root root 4096 Feb 24 00:00 sbin

drwxr-xr-x 2 root root 4096 Feb 24 00:00 srv

dr-xr-xr-x 13 root root 0 Mar 4 14:21 sys

drwxrwxrwt 1 root root 4096 Feb 26 20:02 tmp

drwxr-xr-x 1 root root 4096 Feb 24 00:00 usr

drwxr-xr-x 1 root root 4096 Feb 24 00:00 var

**Log out of nginx shell:-**

root@**7d5626a19ba6**:/# exit

exit

Below the bash command line is changed for nginx-proxy due to our above commands:-

container ls -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

7d5626a19ba6 nginx "bash" 13 hours ago Exited (0) About a minute ago proxy

1c79e0a10e18 mysql "docker-entrypoint.s…" 14 hours ago Up 24 minutes 3306/tcp, 33060/tcp mysql

e0928cd95ac6 nginx "nginx -g 'daemon of…" 14 hours ago Up 25 minutes 80/tcp nginx

**docker container run -it --name ubuntu ubuntu**

Unable to find image 'ubuntu:latest' locally

latest: Pulling from library/ubuntu

423ae2b273f4: Pull complete

de83a2304fa1: Pull complete

f9a83bce3af0: Pull complete

b6b53be908de: Pull complete

Digest: sha256:04d48df82c938587820d7b6006f5071dbbffceb7ca01d2814f81857c631d44df

Status: Downloaded newer image for ubuntu:latest

**Below we are trying to install curl in our ubuntu container**

**root@375b53ff9275:/# apt-get install -y curl**

**Next within container we install below:-**

root@1c79e0a10e18:/# apt-get update

root@1c79e0a10e18:/# apt-get install -y procps

Next we check memory within container:-

**root@1c79e0a10e18:/# ps aux**

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

mysql 1 0.9 20.1 1835732 405104 ? Ssl 14:02 0:41 mysqld

root 199 0.0 0.1 18184 3244 pts/0 Ss 15:08 0:00 bash

root 523 0.0 0.1 36636 2848 pts/0 R+ 15:11 0:00 ps aux

**docker image ls**

REPOSITORY TAG IMAGE ID CREATED SIZE

mysql latest 7a3923452254 2 days ago 465MB

nginx latest a1523e859360 7 days ago 127MB

httpd latest c5a012f9cf45 7 days ago 165MB

ubuntu latest 72300a873c2c 12 days ago 64.2MB

alpine latest e7d92cdc71fe 6 weeks ago 5.59MB

**docker network inspect bridge**

[

{

"Name": "bridge",

"Id": "50e9a1d8c2274ec05a549789107388ba37a8e8b1f50924ff11a46cff543806f0",

"Created": "2020-03-04T03:52:15.737650047Z",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": null,

"Config": [

{

"Subnet": "172.17.0.0/16",

"Gateway": "172.17.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"1c79e0a10e1893c826fb1f0e2913b46491ffce0ea86dce5231d1d087264fdedc": {

"Name": "mysql",

"EndpointID": "7df2424a2e3a75a278b12f65008d0bf40d33e2db962baae0318dc0ad6c1aff1c",

"MacAddress": "",

"IPv4Address": "172.17.0.3/16",

"IPv6Address": ""

},

"e0928cd95ac679c76c2542580cf5362a4458e41dc51575304c2347c44cc81ead": {

"Name": "nginx",

"EndpointID": "24ca84032f91b65cc70b9756f544c9d167110daf9f845afd8c1471444110fd11",

"MacAddress": "",

"IPv4Address": "172.17.0.2/16",

"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": {}

}

]

**Below bridge is the default docker virtual network which is NAT’d behind host IP.**

**Host network is used to connect the container directly to the host network of system.**

**None below is not connected to any network**

**docker network ls**

**NETWORK ID NAME DRIVER SCOPE**

50e9a1d8c227 bridge bridge local

de697d9aa6cc host host local

a19bb64bdc32 none null local

**SPAWNS NEW VIRTUAL NETWORK TO ATTACH CONTAINERS:-**

**docker network create my\_app\_net**

860dd687aec36bc7024a7aab2d4f3252fd537642ccb36a6263de85f674d3f248

NEXT I CREATE A NEW CONTAINER **new\_nginx** and add it to **VIRTUAL NETWORK my\_app\_net**

**docker container run -d --name new\_nginx --network my\_app\_net nginx**

effc6e01f4fbb5167f44fc26113dc70979bc847c1dda778d8f173a1db1c5f268

**docker network ls (Below is new bridge network created)**

**NETWORK ID NAME DRIVER SCOPE**

50e9a1d8c227 bridge bridge local

de697d9aa6cc host host local

860dd687aec3 my\_app\_net bridge local

a19bb64bdc32 none null local

Next I check and confirm that new container named **new\_nginx** is in new virtual network **my\_app\_net**

**docker network inspect my\_app\_net**

[

{

"Name": "my\_app\_net",

"Id": "860dd687aec36bc7024a7aab2d4f3252fd537642ccb36a6263de85f674d3f248",

"Created": "2020-03-04T23:00:48.563402502Z",

"Scope": "local",

"Driver": "bridge",

"EnableIPv6": false,

"IPAM": {

"Driver": "default",

"Options": {},

"Config": [

{

"Subnet": "172.18.0.0/16",

"Gateway": "172.18.0.1"

}

]

},

"Internal": false,

"Attachable": false,

"Ingress": false,

"ConfigFrom": {

"Network": ""

},

"ConfigOnly": false,

"Containers": {

"effc6e01f4fbb5167f44fc26113dc70979bc847c1dda778d8f173a1db1c5f268": {

"Name": "new\_nginx",

"EndpointID": "aa66793037e1c3fa34586260e3e712b620b95678cdaaca2d7778b6bb5f4328f5",

"MacAddress": "",

"IPv4Address": "172.18.0.2/16",

"IPv6Address": ""

}

},

"Options": {},

"Labels": {}

}

Above we created a network and connected our container **new\_nginx** to the newly created bridge network **my\_app\_net** but we can disconnect existing container from one network and connect them to other.

**docker network ls**

**NETWORK ID NAME DRIVER SCOPE**

50e9a1d8c227 bridge bridge local

de697d9aa6cc host host local

860dd687aec3 my\_app\_net bridge local

a19bb64bdc32 none null local

**docker container ls**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

effc6e01f4fb nginx "nginx -g 'daemon of…" 27 hours ago Up 2 hours 80/tcp new\_nginx

1c79e0a10e18 mysql "docker-entrypoint.s…" 38 hours ago Up 13 hours 3306/tcp, 33060/tcp mysql

e0928cd95ac6 nginx "nginx -g 'daemon of…" 38 hours ago Up 13 hours 80/tcp nginx

Now we connect another container existing to newly created network i.e network (860dd687aec3)

**docker network connect 860dd687aec3 e0928cd95ac6**

Now we inspect if existing nginx has been connected to new network which we created recently:-

**Below we can see the old container existing nginx has been connected to new network**

**docker container inspect e0928cd95ac6 along with old default**

………………some logs there …………..

"Mounts": [],

"Config": {

"Hostname": "e0928cd95ac6",

"Domainname": "",

"User": "",

"AttachStdin": false,

"AttachStdout": false,

"AttachStderr": false,

"ExposedPorts": {

"80/tcp": {}

},

"Tty": false,

"OpenStdin": false,

"StdinOnce": false,

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",

"NGINX\_VERSION=1.17.8",

"NJS\_VERSION=0.3.8",

"PKG\_RELEASE=1~buster"

],

"Cmd": [

"nginx",

"-g",

"daemon off;"

],

"Image": "nginx",

"Volumes": null,

"WorkingDir": "",

"Entrypoint": null,

"OnBuild": null,

"Labels": {

"maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"

},

"StopSignal": "SIGTERM"

},

"NetworkSettings": {

"Bridge": "",

"SandboxID": "18bf75cadd0db96f267128edf0b76deb657789ab20bcb55843aa1cf72aaf7c1c",

"HairpinMode": false,

"LinkLocalIPv6Address": "",

"LinkLocalIPv6PrefixLen": 0,

"Ports": {

"80/tcp": null

},

"SandboxKey": "/var/run/docker/netns/18bf75cadd0d",

"SecondaryIPAddresses": null,

"SecondaryIPv6Addresses": null,

"EndpointID": "24ca84032f91b65cc70b9756f544c9d167110daf9f845afd8c1471444110fd11",

"Gateway": "172.17.0.1",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"IPAddress": "172.17.0.2",

"IPPrefixLen": 16,

"IPv6Gateway": "",

"MacAddress": " ",

"Networks": {

"bridge": {

"IPAMConfig": null,

"Links": null,

"Aliases": null,

"NetworkID": "50e9a1d8c2274ec05a549789107388ba37a8e8b1f50924ff11a46cff543806f0",

"EndpointID": "24ca84032f91b65cc70b9756f544c9d167110daf9f845afd8c1471444110fd11",

"Gateway": "172.17.0.1",

"IPAddress": "172.17.0.2",

"IPPrefixLen": 16,

"IPv6Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"MacAddress": "",

"DriverOpts": null

},

"my\_app\_net": {

"IPAMConfig": {},

"Links": null,

"Aliases": [

"e0928cd95ac6"

],

"NetworkID": "860dd687aec36bc7024a7aab2d4f3252fd537642ccb36a6263de85f674d3f248",

"EndpointID": "9aa91ee511867d9af6cb1177c107cf0fc027c0f2b3543fade31ca8a1255823d5",

"Gateway": "172.18.0.1",

"IPAddress": "172.18.0.3",

"IPPrefixLen": 16,

"IPv6Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"MacAddress": "",

"DriverOpts": {}

}

}

}

}

]

Now we disconnect the container **e0928cd95ac6** from the network **860dd687aec3**:-

**docker network disconnect 860dd687aec3 e0928cd95ac6**

After disconnecting not there is only default network connected to the container **e0928cd95ac6**

**docker container inspect e0928cd95ac6**

…………………………………some log here …………………………….

"Mounts": [],

"Config": {

"Hostname": "e0928cd95ac6",

"Domainname": "",

"User": "",

"AttachStdin": false,

"AttachStdout": false,

"AttachStderr": false,

"ExposedPorts": {

"80/tcp": {}

},

"Tty": false,

"OpenStdin": false,

"StdinOnce": false,

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",

"NGINX\_VERSION=1.17.8",

"NJS\_VERSION=0.3.8",

"PKG\_RELEASE=1~buster"

],

"Cmd": [

"nginx",

"-g",

"daemon off;"

],

"Image": "nginx",

"Volumes": null,

"WorkingDir": "",

"Entrypoint": null,

"OnBuild": null,

"Labels": {

"maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"

},

"StopSignal": "SIGTERM"

},

"NetworkSettings": {

"Bridge": "",

"SandboxID": "18bf75cadd0db96f267128edf0b76deb657789ab20bcb55843aa1cf72aaf7c1c",

"HairpinMode": false,

"LinkLocalIPv6Address": "",

"LinkLocalIPv6PrefixLen": 0,

"Ports": {

"80/tcp": null

},

"SandboxKey": "/var/run/docker/netns/18bf75cadd0d",

"SecondaryIPAddresses": null,

"SecondaryIPv6Addresses": null,

"EndpointID": "24ca84032f91b65cc70b9756f544c9d167110daf9f845afd8c1471444110fd11",

"Gateway": "172.17.0.1",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"IPAddress": "172.17.0.2",

"IPPrefixLen": 16,

"IPv6Gateway": "",

"MacAddress": "",

"Networks": {

"bridge": {

"IPAMConfig": null,

"Links": null,

"Aliases": null,

"NetworkID": "50e9a1d8c2274ec05a549789107388ba37a8e8b1f50924ff11a46cff543806f0",

"EndpointID": "24ca84032f91b65cc70b9756f544c9d167110daf9f845afd8c1471444110fd11",

"Gateway": "172.17.0.1",

"IPAddress": "172.17.0.2",

"IPPrefixLen": 16,

"IPv6Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"MacAddress": "",

"DriverOpts": null

}

}

}

}

]

**We move along to see how container talk with each other, containers and talk with container in same virtual network using container names :-**

**So my\_alpine is in my\_app\_net**

**docker container run -d --name my\_alpine --network my\_app\_net nginx:alpine**

Unable to find image 'nginx:alpine' locally

alpine: Pulling from library/nginx

4167d3e14976: Pull complete

bb292c78f105: Pull complete

Digest: sha256:abe5ce652eb78d9c793df34453fddde12bb4d93d9fbf2c363d0992726e4d2cad

Status: Downloaded newer image for nginx:alpine

**8dfe2c236063357b6c27eb5177c632d1eaa9ad1fc5916009066fc3be3ff09735**

**8dfe2c2360633** is in network **my\_app\_net**

**docker container inspect 8dfe2c2360633**

………………………………..some log here ……………………………………..

"Mounts": [],

"Config": {

"Hostname": "8dfe2c236063",

"Domainname": "",

"User": "",

"AttachStdin": false,

"AttachStdout": false,

"AttachStderr": false,

"ExposedPorts": {

"80/tcp": {}

},

"Tty": false,

"OpenStdin": false,

"StdinOnce": false,

"Env": [

"PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",

"NGINX\_VERSION=1.17.9",

"NJS\_VERSION=0.3.9",

"PKG\_RELEASE=1"

],

"Cmd": [

"nginx",

"-g",

"daemon off;"

],

"Image": "nginx:alpine",

"Volumes": null,

"WorkingDir": "",

"Entrypoint": null,

"OnBuild": null,

"Labels": {

"maintainer": "NGINX Docker Maintainers <docker-maint@nginx.com>"

},

"StopSignal": "SIGTERM"

},

"NetworkSettings": {

"Bridge": "",

"SandboxID": "b86731b65628cb86d6122b7de3b9a66214d124b908f2b98ac20270a927e69721",

"HairpinMode": false,

"LinkLocalIPv6Address": "",

"LinkLocalIPv6PrefixLen": 0,

"Ports": {

"80/tcp": null

},

"SandboxKey": "/var/run/docker/netns/b86731b65628",

"SecondaryIPAddresses": null,

"SecondaryIPv6Addresses": null,

"EndpointID": "",

"Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"IPAddress": "",

"IPPrefixLen": 0,

"IPv6Gateway": "",

"MacAddress": "",

"Networks": {

"my\_app\_net": {

"IPAMConfig": {},

"Links": null,

"Aliases": [

"8dfe2c236063"

],

"NetworkID": "860dd687aec36bc7024a7aab2d4f3252fd537642ccb36a6263de85f674d3f248",

"EndpointID": "dede1568dd29d38ff94760b3cc03cce94066cb2780f53b531ac59dbe1cab0c73",

"Gateway": "172.18.0.1",

"IPAddress": "172.18.0.3",

"IPPrefixLen": 16,

"IPv6Gateway": "",

"GlobalIPv6Address": "",

"GlobalIPv6PrefixLen": 0,

"MacAddress": "",

"DriverOpts": {}

}

}

}

}

]

Next I include the **my\_alpine** in the network **my\_app\_net**

**docker network connect 860dd687aec3 8dfe2c236063**

**docker container exec -it my\_alpine ping new\_nginx**

PING new\_nginx (172.18.0.2): 56 data bytes

64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.143 ms

64 bytes from 172.18.0.2: seq=1 ttl=64 time=0.229 ms

64 bytes from 172.18.0.2: seq=2 ttl=64 time=0.205 ms

**So within a network we can ping using name of container which is dns**

**The default bridge network driver allow containers to communicate with each other when running on the same docker host.**

**NOTE : Do not pull ngnix container only that doesn’t have ping utility installed by default , in that case log into nginx. So better install nginx:aphine**

**Otherwise you might get error:-**

[Docker, **\“ping\”**: executable file not found in **$PATH":** unknown](https://stackoverflow.com/questions/49463719/docker-ping-executable-file-not-found-in-path-unknown)

**DOCKER LAYERS:**

**docker history nginx:latest**

**IMAGE CREATED CREATED BY SIZE COMMENT**

a1523e859360 8 days ago /bin/sh -c #(nop) CMD ["nginx" "-g" "daemon… 0B

<missing> 8 days ago /bin/sh -c #(nop) STOPSIGNAL SIGTERM 0B

<missing> 8 days ago /bin/sh -c #(nop) EXPOSE 80 0B

<missing> 8 days ago /bin/sh -c ln -sf /dev/stdout /var/log/nginx… 22B

<missing> 8 days ago /bin/sh -c set -x && addgroup --system -… 57.5MB

<missing> 8 days ago /bin/sh -c #(nop) ENV PKG\_RELEASE=1~buster 0B

<missing> 8 days ago /bin/sh -c #(nop) ENV NJS\_VERSION=0.3.8 0B

<missing> 8 days ago /bin/sh -c #(nop) ENV NGINX\_VERSION=1.17.8 0B

<missing> 8 days ago /bin/sh -c #(nop) LABEL maintainer=NGINX Do… 0B

<missing> 9 days ago /bin/sh -c #(nop) CMD ["bash"] 0B

<missing> 9 days ago /bin/sh -c #(nop) ADD file:e5a364615e0f69616… 69.2MB

**If you change docker file then you need to build it again using below file:-**

docker image build -t **customngnix (my customized ngnix)**

**Note:** In docker the unique data like say database data are stored in different location and not in UFS (Union File System) .In Union File System there is only metdata.

If you want to find where the path , then kindly open **DockerFile** and check **VOLUME** keyword in it.

We can see the path once we execute inspect command:-

**docker container inspect mysql**

**OUTPUT (Volume section shows path where its in container):-**

"Volumes": {

"/var/lib/mysql": {}

},

Below Mounts Section shows

1. **"Source":** Physical location in host
2. **"Destination":** Location in container

"Mounts": [

{

"Type": "volume",

"Name": "39ad9874f11724823fc2de9cd8aac6625804a45599d38043a97ff10e9ef18f3e",

"Source": "/var/lib/docker/volumes/39ad9874f11724823fc2de9cd8aac6625804a45599d38043a97ff10e9ef18f3e/\_data",

"Destination": "/var/lib/mysql",

"Driver": "local",

"Mode": "",

"RW": true,

"Propagation": ""

}

],

So when you remove container this location data doesn’t get removed it need to be removed physically.

Checking volumes there in the system:-

**docker volume ls**

**DRIVER VOLUME NAME**

local 39ad9874f11724823fc2de9cd8aac6625804a45599d38043a97ff10e9ef18f3e

local ce2a9a3e240b82d25ebd26912858ba439b087a54769ed218286fe9cc2d4d0846

**Note: Even if we remove the container the volume will remain**

As viewing the volume we done get any idea which volume contains whats so the concept of **Named Volumes** have come into being.

**docker volume inspect 39ad9874f11724823fc2de9cd8aac6625804a45599d38043a97ff10e9ef18f3e**

**Below we create mysql with named volume:-**

docker container run -d --name mysql2 -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=True -v mysql-db:/var/lib/mysql mysql

074b816cb0b030e25ea91a72cbfd2c79b1c8b32787dee3168c204376a2715051

**docker volume ls (mysql-db is the named volume )**

**OUTPUT:**

DRIVER VOLUME NAME

local 39ad9874f11724823fc2de9cd8aac6625804a45599d38043a97ff10e9ef18f3e

local ce2a9a3e240b82d25ebd26912858ba439b087a54769ed218286fe9cc2d4d0846

local mysql-db

Bind Mounts is also another very important concept.