DevOps

Docker: Containers

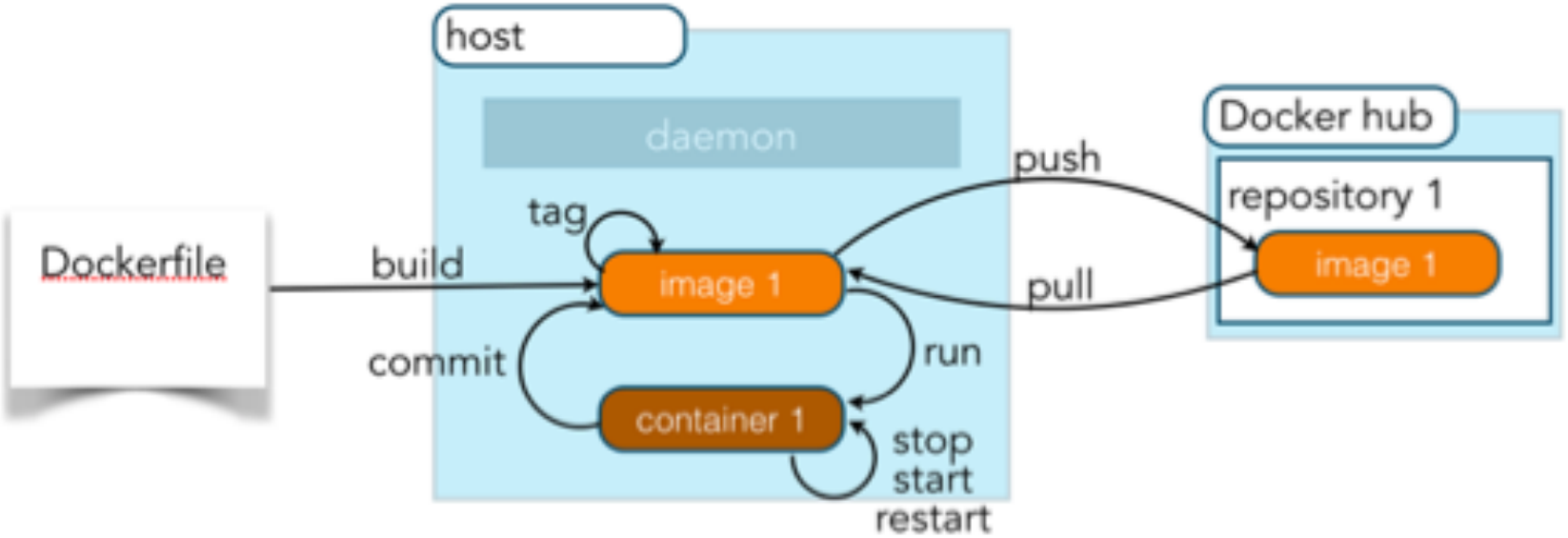
Trainer: Abhijith V G – AWS, eCommerce, Mobile & DevOps Architect

Magento Certified DEVELOPER PLUS

amazon web services | Certified

Developer - Associate

Docker: Docker Overview



Docker: Custom images - Creating a custom image

1. Using Dockerfile

A few Dockerfile instructions you should explore:
COPY, EXPOSE, CMD, ENTRYPOINT

```
Sree>cat Dockerfile
FROM ubuntu
MAINTAINER Sreeprakash Neelakantan <sree@example.com>

RUN apt-get update
RUN apt-get install -y git
```

```
Sree>docker build -t test-git .
```

```
Sree>docker images| grep git
test-git                    latest              d71b2fb410ec        10 seconds ago      249MB
```

Docker: Custom images - Running a container from the custom image

```
Sree>docker images| grep git
test-git                            latest              d71b2fb410ec        10 seconds ago      249MB
```

```
Sree>docker run -ti test-git
root@90cace1b3fb9:/# git version
git version 2.7.4
```

Docker: Custom images - Publishing the custom image

1. Create an account with Docker.com/docker.io
2. Execute ***docker login*** and use your docker.com/docker.io credentials.
3. Find the image ID
4. Tag your image with <repo account username>/<image name>:<version>
5. ***docker push <repo account username>/<image name>:<version>***

```
Sree>docker images| grep git|head -1
test-git                    latest              d71b2fb410ec        5 minutes ago       249MB
Sree>docker tag d71b2fb410ec schogini/test-git:latest
Sree>docker push schogini/test-git:latest
The push refers to a repository [docker.io/schogini/test-git]
33b27e4bac8a: Pushed
7a7d4a8aee0a: Pushing [==================>                                ]  16.18MB/38.87MB
a09947e71dc0: Mounted from library/ubuntu
9c42c2077cde: Mounted from library/ubuntu
625c7a2a783b: Mounted from library/ubuntu
25e0901a71b8: Mounted from library/ubuntu
8aa4fcad5eeb: Mounted from library/ubuntu
```

Docker: Docker Networking - Accessing containers

4 ways of running commands in container:

1. Using “docker run” to directly execute commands
Example: ***docker run ubuntu:trusty <COMMAND> eg: ls -l***

2. Using “-i” & “-t” options of “docker run” command. (-d option starts it in the background)
Example: ***docker run -ti ubuntu:trusty***

3. Using “docker exec” command for existing containers.
Example: ***docker exec <CONTAINER NAME/ID> ls –l***

4. Using “docker attach” command for existing containers.
Example: ***docker attach <CONTAINER NAME/ID>***

5. Access the exposed ports using HTTP or other protocols

Docker: Docker Networking - Exposing container ports (All)

Opening ports:

If you want containers to accept incoming requests, you will need to provide special options when invoking docker run. There are two approaches.

First, you can supply -P or --publish-all=true|false to docker run which is a blanket operation that identifies every port with an EXPOSE line in the image's Dockerfile

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>ls -l
Sree>echo Test123 > sree.txt
Sree>ls -l
total 8
-rw-r--r--  1 sree  staff  8 Aug 19 14:45 sree.txt
Sree>docker run -tid -v $PWD:/usr/share/nginx/html -P nginx
d89d38e7fa28e81c3f77648ed8547d68d45176903be8fd0da2e66a1ae38f8a1a
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                     NAMES
d89d38e7fa28        nginx               "nginx -g 'daemon ..."   4 seconds ago       Up 3 seconds        0.0.0.0:32779->80/tcp     cranky_engelbart
Sree>curl http://localhost:32779/sree.txt
Test123
```

Docker: Docker Networking - Exposing container ports (Specific)

Opening ports:
Second, you can supply -p to map specific ports with the hosts. The container ports used must be exposed within the container and the host port used must be free.

Below command maps container port 80 to host port 8080. This allows accessing (browsing in this case) port 8080 from the host to talk to the container process (nginx in this case)

```
SreeMacMin16GB:docker sree$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
SreeMacMin16GB:docker sree$ docker run -d -p 8080:80 nginx
23b46a08d90c95629f254a4e6bd8a158d195a3d26825625ea5e17df0dff4154d
SreeMacMin16GB:docker sree$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
 NAMES
23b46a08d90c        nginx               "nginx -g 'daemon ..."   3 seconds ago       Up 1 second         0.0.0.0:8080->80/tcp
 brave_wright
```

Docker: Docker Networking - Container Routing

When you install Docker, it creates three networks automatically. You can list these networks using the docker network ls command:

```
Sree>docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
315518bfd7ca        bridge              bridge              local
06380267a063        docker_gwbridge     bridge              local
a22b84f1c60a        host                host                local
sn6tn5pkh948        ingress             overlay             swarm
906f11bfc4e9        magento2_default    bridge              local
42d6ab5e52df        none                null                local
204c73bc3bfb        s941_default        bridge              local
```

These three networks are built into Docker. When you run a container, you can use the --network flag to specify which networks your container should connect to.

Docker: Docker Networking - Container Routing

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.

The none and host networks are not directly configurable in Docker. However, you can configure the default bridge network, as well as your own user-defined bridge networks.

These two have specific uses for the Docker installation.
Do not bind containers to these networks.

Docker: Docker Networking – Default Bridge Network

The default bridge network is present on all Docker hosts. If you do not specify a different network, new containers are automatically connected to the default bridge network.

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>docker run -tid --name box1 ubuntu:trusty
b12e612bfffd4839cacc5da693580304bfcf11d76feb32594a1747bde16d8a51
Sree>docker run -tid --name box2 ubuntu:trusty
2fca6a77df68e040a599d47a26606748ddd34d688764657375bfa5117f168778
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
2fca6a77df68        ubuntu:trusty       "/bin/bash"         4 seconds ago       Up 3 seconds                            box2
b12e612bfffd        ubuntu:trusty       "/bin/bash"         10 seconds ago      Up 8 seconds                            box1
Sree>docker inspect --format '{{.NetworkSettings.Networks.bridge.IPAddress}}' box1
172.17.0.2
Sree>docker inspect --format '{{.NetworkSettings.Networks.bridge.IPAddress}}' box2
172.17.0.3
Sree>docker attach box1
root@b12e612bfffd:/#
root@b12e612bfffd:/# netcat -l 2222
```

Docker: Docker Networking – Default Bridge Network

All containers within one bridge network and talk to each other (on all ports)

```
Sree2>docker attach box2
root@2fca6a77df68:/#
root@2fca6a77df68:/# netcat 172.17.0.2 2222
Hello from Box2
```

```
root@b12e612bfffd:/# netcat -l 2222
Hello from Box2
```

Docker: Docker Networking – User Defined (Bridge) Network

User defined network is preferred over the default bridge network!

```
Sree>docker network create --driver bridge sreenw1
74ac4386c7b1e71b3a99cdff4caa38ff01bc5cff29fbcc79ed08aa3a3aa0ddf5
Sree>docker run -tid --name box3 --network sreenw1 ubuntu:trusty
322613d0373fab1cbc91b2c9b0431e426122e775591ad047854c18b4b32b141b
Sree>docker run -tid --name box4 --network sreenw1 ubuntu:trusty
69eb8bf0eca6af22ecd1dd923a81772bfdfebfe579a57e3ae23897a86d4ed528
Sree>docker inspect --format '{{.NetworkSettings.Networks.sreenw1.IPAddress}}' box3
172.21.0.2
Sree>docker inspect --format '{{.NetworkSettings.Networks.sreenw1.IPAddress}}' box4
172.21.0.3
```

```
Sree>docker attach box3
root@322613d0373f:/#
root@322613d0373f:/# ping 172.21.0.3
PING 172.21.0.3 (172.21.0.3) 56(84) bytes of data.
64 bytes from 172.21.0.3: icmp_seq=1 ttl=64 time=0.202 ms
64 bytes from 172.21.0.3: icmp_seq=2 ttl=64 time=0.155 ms
```

Docker: Docker Networking – User Defined (Bridge) Network

One (custom) bridge (sreenw1) containers cannot communicate with another bridge containers!

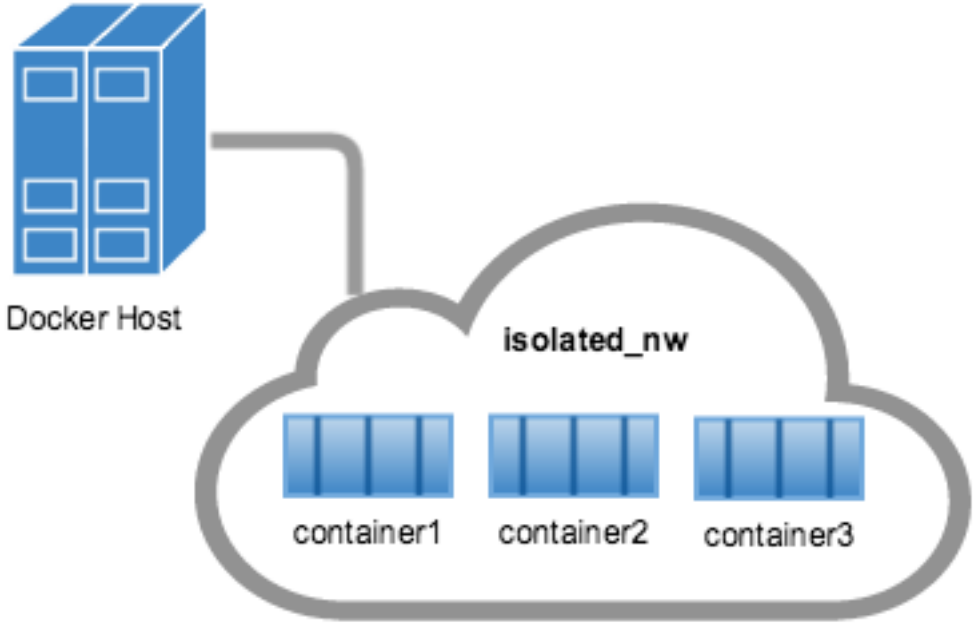
[you will need to have a 3rd container which is a member of the 2 networks.]

```
Sree>docker run -tid --name box5  ubuntu:trusty
b65559df9f614b2c214374088e2a423125e6bd27d6f36273d8b1937d87add53a
Sree>docker inspect --format '{{.NetworkSettings.Networks.bridge.IPAddress}}' box5
172.17.0.2
Sree>docker attach box3
root@322613d0373f:/#
root@322613d0373f:/# ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2) 56(84) bytes of data.
^C
--- 172.17.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2110ms
```

Docker: Docker Networking – User Defined (Bridge) Network

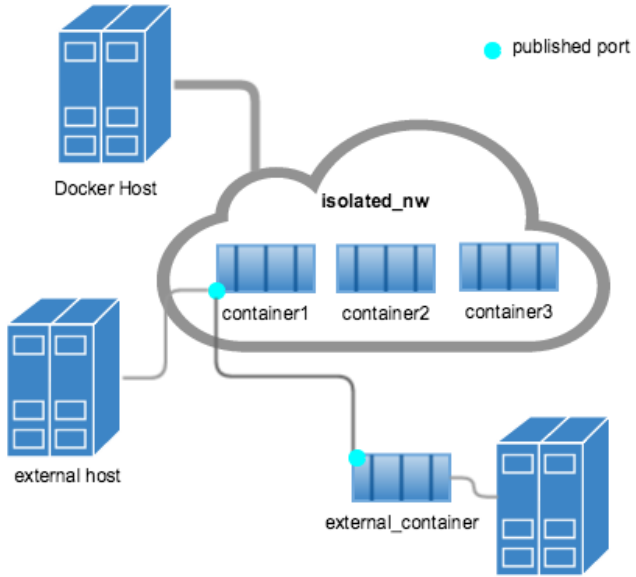
One (custom) bridge (sreenw1) containers cannot communicate with another bridge containers!

[you will need to have a 3rd container which is a member of the 2 networks.]



Docker: Docker Networking – User Defined (Bridge) Network

Need to publish(map) ports in order to allow communication between One (custom) bridge (sreenw1) container and another Container.



Docker: Docker Networking –Default Network Limitation

Containers within a default (bridge) network cannot discover others using the names, we need to use IP addresses.

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>docker run -tid --name box1 ubuntu:trusty
4af3602f0d7e456fcd6d72b38eacfe543b97a1ddbfdb3b59f62cbd429743052b
Sree>docker run -tid --name box2 ubuntu:trusty
b988bf8696ab69167e1d73ab4e819c3f6ff8f2d668256144eafb9e04e25dd012
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
b988bf8696ab        ubuntu:trusty       "/bin/bash"         3 seconds ago       Up 1 second                             box2
4af3602f0d7e        ubuntu:trusty       "/bin/bash"         9 seconds ago       Up 8 seconds                            box1
Sree>docker inspect --format '{{.NetworkSettings.Networks.bridge.IPAddress}}' box1
172.17.0.2
Sree>docker inspect --format '{{.NetworkSettings.Networks.bridge.IPAddress}}' box2
172.17.0.3
Sree>docker attach box1
root@4af3602f0d7e:/#
root@4af3602f0d7e:/# ping box2
ping: unknown host box2
root@4af3602f0d7e:/# ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3) 56(84) bytes of data.
64 bytes from 172.17.0.3: icmp_seq=1 ttl=64 time=0.154 ms
64 bytes from 172.17.0.3: icmp_seq=2 ttl=64 time=0.158 ms
```

Docker: Docker Networking –Default Network Limitation

Containers within a default (bridge) network if started using the link option can discover others using the names.

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>docker run -tid --name box1 ubuntu:trusty
c300ddebe7995169d000e23ee0b4c77c09d4c59361116275ca5bf984cad20609
Sree>docker run -tid --name box2 --link box1 ubuntu:trusty
2e9601d58bab067f54dfef7d86eed434d70f2e38f5bd7e7af42f3c96813eab09
Sree>docker attach box1
root@c300ddebe799:/#
root@c300ddebe799:/# ping box2
ping: unknown host box2
root@c300ddebe799:/# read escape sequence
Sree>docker attach box2
root@2e9601d58bab:/#
root@2e9601d58bab:/# ping box1
PING box1 (172.17.0.2) 56(84) bytes of data.
64 bytes from box1 (172.17.0.2): icmp_seq=1 ttl=64 time=0.227 ms
64 bytes from box1 (172.17.0.2): icmp_seq=2 ttl=64 time=0.111 ms
64 bytes from box1 (172.17.0.2): icmp_seq=3 ttl=64 time=0.095 ms
^C
--- box1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2070ms
rtt min/avg/max/mdev = 0.095/0.144/0.227/0.059 ms
```

Docker: Docker Networking –Default Network Limitation

A containers started with link option will add the name to the /etc/hosts file
See box1 entry in below

```
root@2e9601d58bab:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.17.0.2      box1 c300ddebe799
172.17.0.3      2e9601d58bab
```

Docker: Docker Networking –Custom Network Advantage

A containers started within a custom network can auto discover other containers within using their names.

```
Sree>docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
Sree>docker run -tid --name box3 --network sreenw1 ubuntu:trusty
a5e97cb7e1b16a7f078e6746a163331759735e730e0eca235da2061242204c58
Sree>docker run -tid --name box4 --network sreenw1 ubuntu:trusty
7f592532f91ed1308d01c7e3d69f98f3d75a600886b20ddcf4955adf3f75cfda
Sree>docker attach box3
root@a5e97cb7e1b1:/#
root@a5e97cb7e1b1:/# ping box4
PING box4 (172.21.0.3) 56(84) bytes of data.
64 bytes from box4.sreenw1 (172.21.0.3): icmp_seq=1 ttl=64 time=0.245 ms
64 bytes from box4.sreenw1 (172.21.0.3): icmp_seq=2 ttl=64 time=0.111 ms
^C
--- box4 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 0.111/0.178/0.245/0.067 ms
```

Docker: Docker Networking –Custom Network Advantage

A containers started within a custom network can auto discover other containers within using their names, even though no entry gets added to /etc/hosts.

```
Sree>docker run -tid --name box3 --network sreenw1 ubuntu:trusty
ff545e7189ccd572c0e2f227d39bab522db717713579a07c2ff701de2e94597b
Sree>docker run -tid --name box4 --network sreenw1 ubuntu:trusty
777ca15cad8711765645a561361f45011c11f32bb329622def84c04c5da7345e
Sree>docker attach box3
root@ff545e7189cc:/#
root@ff545e7189cc:/# cat /etc/hosts
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
172.21.0.2      ff545e7189cc
root@ff545e7189cc:/#
```

Docker: Docker Networking –Default vs Custom Network

Note: Any link between containers created with legacy link is static in nature and hard-binds the container with the alias. It does not tolerate linked container restarts.

The new link functionality in user defined networks supports dynamic links between containers, and tolerates restarts and IP address changes in the linked container.

Docker: HW-2

1. Create a new custom (bridge) network in your name
2. Launch two ubuntu:trusty containers in this new custom network. Remember to give names to these containers
3. Find the IP addresses of these containers
4. Get inside the container and check the connectivity using IP address as well as name (use ping and netcat)
5. Launch two ubuntu:trusty containers using the default bridge network and confirm if you are able to communicate between them!
6. Repeat step 5 by linking containers at launch
7. **Advanced Exercise Two Docker Container Application**
    1. Create a custom network and launch a mysql server container in it
    2. Launch another container in the same network using image richarvey/nginx-php-fpm
    3. Test a PHP script to connect to the mysql server and increment a counter when browsed. Ensure that the document root /var/www/html is mapped to host!
    4. The entire infrastructure and application should be bash scripted!

Docker: HW-2 Solution using Bash Script

https://gist.github.com/schogini/fd1f9bf52bddbbb5fec2580d42ef79ff

```
#!/bin/sh
# Abhijith V G abhijithvg@example.com

mkdir -p html

cat > html/index.php <<EOT
<?php
\$dbhost = 'abhidb1';
\$dbuser = 'root';
\$dbpass = 'abhi1234';
\$dbname = 'countdb';

\$c = mysqli_connect(\$dbhost, \$dbuser, \$dbpass, \$dbname) or die("Unable to Connect to db: " . mysqli_connect_error() . ' (' . mysqli_connect_errno() . ')');

if (mysqli_query(\$c,'INSERT INTO counter SET data = "dummy"') === TRUE) {
} else {
  echo "Error: " . mysqli_error(\$c);
}
echo "Counter: ". mysqli_insert_id(\$c);
mysqli_close(\$c);
EOT

# Delete the containers if present
docker rm -f abhidb1 abhiweb1

# Delete the network if present
docker network rm abhinet1

# Creating the network fresh
docker network create abhinet1

# Creating the containers fresh
docker run -tid --network abhinet1 --name abhiweb1 -p 8181:80 -v $PWD/html:/var/www/html richarvey/nginx-php-fpm

docker run -d --network abhinet1 --name abhidb1 -e MYSQL_ROOT_PASSWORD=abhi1234 mysql/mysql-server

# Wait for MySQL to start
sleep 60

docker exec -ti abhidb1 mysql -h localhost -u root -pabhi1234 -e "create database countdb;"
docker exec -ti abhidb1 mysql -h localhost -u root -pabhi1234 -e "GRANT ALL ON *.* to root@'%' IDENTIFIED BY 'abhi1234';flush privileges;"
docker exec -ti abhidb1 mysql -h localhost -u root -pabhi1234 -e "use countdb;CREATE TABLE counter (id INT AUTO_INCREMENT PRIMARY KEY, data VARCHAR(255))"
```

Docker: HW-2 Questions

Identify the flaws in the solution
Modify this solution to work without custom bridge network…

IMPORTANT POINTS YOU SHOULD UNDERSTAND

- Difference between attach and exec
- Difference between ENTRYPOINT and CMD
- Exposing ports
- Mapping Volumes
- Networking
- Difference between default and custom bridge


Cognixia
THANK YOU
average 45%