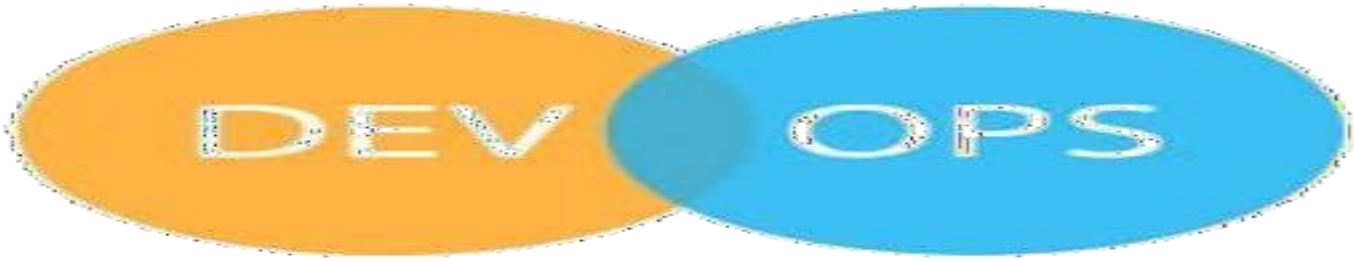
DEV
OPS

1. $ docker image pull alpine
2. $ docker image ls
3. $ docker container run alpine ls -l
4. $ docker container run alpine echo “help from alpine”
5. $ docker container run alpine /bin/sh
6. $ docker container run -it alpine /bin/sh
7. $ docker container ls
8. $ docker container ls -a
9. $ docker container run -it alpine /bin/ash - - - ❼ Container ISOLATION
   Echo “hello world” > hello.txt
   ls
   exit
10. $ docker container run alpine ls
11. This time we will start container which one we exited at point no. 9
    $ docker container start <container ID>
12. $ docker container exec <container ID> ls
13. $ docker run hello-world
14. $ docker start <container-name>
15. $ docker start --attach <container-name>
    Analysis Command 14 and 15 both.
16. $ docker run -it ubuntu bash
    Run the below command
    # ps -ef
    # ls -al
    In Another Terminal run below command
    $ docker top <container name>
17. $ docker run alpine:latest "echo" "Hello, World"
18. $ docker run -dit alpine sh
19. Comparison Alpine and Ubuntu/Debian size

a) time docker run --rm debian sh -c "apt-get update && apt-get install curl"

Output:

real 0m4.396s
user 0m0.029s
sys 0m0.021s

b) time docker run --rm alpine sh -c "apk update && apk add curl"

Analyze time to download the images and size the image.

20. Run Webserver Script from Docker run command

```
docker run -d --rm -p 8080:8080 --name webserver busybox \
    sh -c "while true; do { echo -e 'HTTP/1.1 200 OK\r\n'; \
    echo 'smallest http server'; } | nc -l -p 8080; done"
```

21.$ Create Dockerfile

```
FROM ubuntu

MAINTAINER Admin

RUN apt-get update

CMD ["echo", "Hello World"]
```

Run Build command:

```
$ docker build -t myfirstImage .

$ docker run --name test myfirstImage
```

22.$ docker run -it -p 80:80 nginx

23. $docker run -it -p 80:80 --name mynginx nginx

24.$ docker exec -it mynginx bash

Inside the container:

```
# cd usr/share/nginx/html
# cat index.html
```

Edit the file or remove the file and create new one:

```
# cat >> index.html
**********************************
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Welcome</title>
    <meta name="viewport" content="width=device-width, initial-
     scale=1">
</head>
<body>
    <h1>Welcome!</h1>
```

This file has been created on the host machine and has been mounted into the Nginx Docker Container!
</body>
</html>
Hit the browser and run http://localhost

25.Another WAY:
Create Dockerfile
FROM nginx
COPY html /usr/share/nginx/html

$ docker build -t mynginx_image .

$ docker run --name mynginx -p 80:80 -d mynginx_image

26.Difference between Exec and Attach

$ docker container run --name my_nginx -d -p 8080:80 nginx

$ docker container attach my_nginx
$ docker logs my_nginx
$ CTRL-p CTRL-q key combination to Dettach And Ctrl-c Stops the container

$ docker container run --name my_mysql -d mysql
$ docker container exec -it my_mysql ls /var
$ docker container exec -it my_mysql /bin/bash

Take Away:
The docker exec and docker attach commands allow you to connect to a running container. To get an interactive shell to a container, use the exec command to start a new shell session. The attach command attaches your terminal to a running container.

27.Daemonized Container
$ docker run --name my_daemonized -d ubuntu /bin/sh -c "while true; do echo my daemonized container; sleep 1; done "

$ docker logs