Kelly Technologies

Docker Session

The real use of docker is when running commands inside a docker container and that is where we will start.In this article we will start with using docker commands.

Run - The docker run command first creates a writeable container layer over the specified image, and then starts it using the specified command. This is the command that we use most while working with Docker.

```
[root@localhost ~]# docker run ubuntu echo "hello World"
hello World
```

When we run the above docker command , docker will download the ubuntu image if that is not available on the local system repository.Once downloaded the docker will run the container and once the container is started the command "echo hello World" is ran on the container and exits.

Let's see another example where we assign a name to the container.

```
[root@localhost ~]# docker run --name test -it ubuntu
root@d027bea9d123:/# exit
exit
```

The above example run a container named test using the ubuntu image. Since the image is already downloaded and available in our local system repository the docker run command will use that. "-it" command instructs docker to allocate a TTY connected to the container stdin which will create a interactive bash shell in the container.

The above example provides us with a terminal inside the container where we can edit the container. Once we enter the exit the container is exited and container is closed.

Below are some of the examples of the run command ,
docker run -it --rm ubuntu bash - Automatically remove the container when it exits
docker run -w /work -it ubuntu pwd - the command "-w" sets the working directory and run the provided command in that directory. If the path does not exist it is created inside the container.

There are 2 ways a container is ran,
Attached - When a container is ran in an attached mode , the container will provide the terminal inside the container. This is done by providing the "-it" with the docker run command

Detached - Run container in background and print container ID. THis is done by running the docker command as,

```
[root@localhost ~]# docker run -d --name testing -it ubuntu
344d7dc4b4630a6fe135909669258b3d42fc34cd681e29766d1e805bdfea225d
```

images - docker containers are based on the images. As in the above example we have seen that ubuntu image is downloaded when we run the docker container. The images command will show all the top level images , their repositories , tags along with their sizes.

```
[root@localhost ~]# docker images
REPOSITORY          TAG      IMAGE ID       CREATED       SIZE
jagadish1           latest   22434f5e23e0   2 days ago    226.5 MB
```

The images ones downloaded will be stored in local repository. If we run the container with the image , the container will try to use the image available in the local repository or it will try to download from online.

Docker image -a : will show all the docker images in the local repository

Rmi - Removes the images. This command will remove the image from the local repository.
[root@localhost ~]# docker rmi sample1
Untagged: sample1:latest
Deleted: sha256:9390d7143d8f53e78b9341efbd6740e3f32b82c1ced311f1ff3b7f5266947346

docker rmi -f `docker images` - Forcefully remove the images from local repository

Rm - Removing a container is done by using the "rm" command Or we can provide the "--rm" along with the docker run so that the container is removed when the container is exited.

[root@localhost ~]# docker rm 344d7dc4b463
344d7dc4b463

Ps - Much like linux ps command , the docker ps command will show running containers

[root@localhost ~]# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
344d7dc4b463 ubuntu "/bin/bash" 6 minutes ago Up 6 min testing

The ps command along with some argument will show both running and stopped containers

Stop - As the name says the command will stop one or more running conainers.
[root@localhost ~]# docker stop 344d7dc4b463
344d7dc4b463

Start - As the says the command starts the containers
[root@localhost ~]# docker start 344d7dc4b463
344d7dc4b463

Kill - kill command will kill the container.The main process inside the container will be send with a SIGKILL signal or any signal that is sent as argument with option --signal
[root@localhost ~]# docker kill 344d7dc4b463
344d7dc4b463

Pull - Pull an image or a repository from a registry. This will pull the image and saves in the local repository. When we use the docker run command, it not just pull the image , saves it and starts the container. In this pull case the image is downloaded by not started

[root@localhost ~]# docker pull busybox
Using default tag: latest
Trying to pull repository docker.io/library/busybox ...
latest: Pulling from docker.io/library/busybox

Digest: sha256:817a12c32a39bbe394944ba49de563e085f1d3c5266eb8e9723256bc4448680e

Now the image will be available in the local repository.

Logs - The docker logs command batch-retrieves logs present at the time of execution.

```
[puppet@root$:/work/roles/npm]$  docker logs eb649e927b2c
root@eb649e927b2c:/# apt-install ping
bash: apt-install: command not found
root@eb649e927b2c:/# apt-update
bash: apt-update: command not found
root@eb649e927b2c:/# apt
apt 1.2.12 (amd64)
Usage: apt [options] command
```

Inspect - Returns low level information of the running docker container. The low level details include details like state, image details , network details ,volume details etc

```
[root@localhost ~]# docker inspect bafbdbd7b831
[
        {
        "Id": "bafbdbd7b8314127d809f38c33d7e35333a5db64e859bb3627320d96fe4d5767",
        "Created": "2017-02-19T08:20:09.079913276Z",
        "Path": "/bin/bash",
        "Args": [],
        "State": {
        "Status": "running",
        "Running": true,
        "Paused": false,
        "Restarting": false,
        "OOMKilled": false,
        "Dead": false,
        "Pid": 6713,
        "ExitCode": 0,
        "Error": "",
        "StartedAt": "2017-02-19T08:20:10.837330372Z",
        "FinishedAt": "0001-01-01T00:00:00Z"
```

},

****

****

Attach - As we discussed earlier a container can be started either attached and detached mode. A Container can be started in detached mode using the "-d" argument to the run command. Once the container is started in detached mode , the container run in the background and displays the container ID. in order to attach to the detached container we can use the docker attach command as

[root@localhost ~]# docker attach bafbdbd7b831

root@bafbdbd7b831:/#

Once we exit the container , the container is exit completely. In order to come out of the container without exiting , we can press the ^P +^Q so that we come out of the container without killing that. We need to make sure to attach correctly else we can lock up our terminal

Info - displays the docker system details

[puppet@root$:~]$ docker info

Containers: 70

Running: 1

Paused: 0

Stopped: 69

Images: 3

Server Version: 1.12.5

Top - display the running process of the container

[root@localhost ~]# docker top d3a2df0aab3a

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	7056	7040	0	13:57	pts/1	00:00:00	/bin/bash

Stats- Display a live stream of container(s) resource usage statistics

[root@localhost ~]# docker stats d3a2df0aab3a

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
d3a2df0aab3a	0.00%	3.82 MiB / 5.555 GiB	0.07%	648 B / 648 B	3.81MB/0B	0

pause/unpause - pause or unpasue all running process inside the container.

[root@localhost ~]# docker pause d3a2df0aab3a

D3a2df0aab3a

Once the container is paused, the status of the container changes to paused when we use the docker ps command

Diff - Inspect changes to files or directories on a container's filesystem
[root@localhost ~]# docker diff d3a2df0aab3a
C /run
A /run/secrets

Cp - Copy files/folders between a container and the local filesystem
[root@localhost ~]# docker cp $PWD/anaconda-ks.cfg d3a2df0aab3a:/tmp
Now check the container using diff command.
[root@localhost ~]# docker diff d3a2df0aab3a
C /run
A /run/secrets
C /tmp
A /tmp/anaconda-ks.cfg

Exec- run a command inside a container. The exec command can be used to run a command in already started container. It can be a shell or some other process.
[root@localhost ~]# docker exec 55d648b02396 touch /tmp/tooMuch
Now check the container using diff command.
[root@localhost ~]# docker diff 55d648b02396
C /tmp
A /tmp/tooMuch
C /run
A /run/secrets

Export - Exports the container file system to a tar file.
[root@localhost ~]# docker export 55d648b02396 -o ./hello.tar

Import - imports the contents from local tar file to create a file system
[root@localhost ~]# tar -c hello.tar | docker import - exampleimagedir
sha256:5975b25de3511eab088d461959e107b6313da00e2e9fd3d9d6a61241ce1b0991

History - shows the history of the image
[root@localhost ~]# docker history ping-ubuntu:v0.1

IMAGE	CREATED	CREATED BY	SIZE	COMMENT
afe9768b31f0	4 days ago	/bin/bash	45.02 MB	
f49eec89601e	4 weeks ago	/bin/sh -c #(nop)	CMD ["/bin/bash"]	

****

Rename - Renames a container
[root@localhost ~]# docker rename testing menting
[root@localhost ~]# docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
55d648b02396	ubuntu	"/bin/bash"	18 minutes ago	Up 18

PORTS NAMES
minutes menting

Some of the use full docker commands are,
docker kill $(docker ps -q) : Kill all containers
docker rm $(docker ps -a -q) : delete all stopped containers
docker rmi $(docker images -q -f dangling=true) : Delete all the NONE images
docker rmi $(docker images -q) : Delete all images
docker ps -n 1 : show last 1
docker rmi -f `docker images` - Forcefully remove the images from local repository

docker images -a : Show all docker iamges in local repository

Docker ps -aq | xargs docker rm -f : will delete all containers running and non running