Docker – First Image & Container

1. Let's start by checking whether the docker service is up and running.

```
[root@localhost ~]# service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; ve
   Active: active (running) since Sat 2019-08-10 15:20:01 EDT; 1min 25
     Docs: http://docs.docker.com
 Main PID: 1913 (dockerd-current)
   CGroup: /system.slice/docker.service
           ├─1913 /usr/bin/dockerd-current --add-runtime docker-runc=/
           └─1919 /usr/bin/docker-containerd-current -l unix:///var/ru

Aug 10 15:19:59 localhost.localdomain dockerd-current[1913]: time="201
```

2. Let's check the images in the docker.

$ docker image ls

```
[root@localhost ~]# docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
[root@localhost ~]#
```

Let's download the first image

```
[root@localhost ~]# docker pull hello-world
Using default tag: latest
Trying to pull repository docker.io/library/hello-world ...
latest: Pulling from docker.io/library/hello-world
1b930d010525: Pull complete
Digest: sha256:6540fc08ee6e6b7b63468dc3317e3303aae178cb8a45ed3123180328bcc1d20f
Status: Downloaded newer image for docker.io/hello-world:latest
[root@localhost ~]#
```

$ docker image ls

```
[root@localhost ~]# docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
docker.io/hello-world   latest              fce289e99eb9        7 months ago        1.84 kB
[root@localhost ~]#
```

To give a custom name/tag name to the image

$ docker tag hello-world:latest my-hello-world:latest

```
[root@localhost ~]# docker tag hello-world:latest my-hello-world:latest
[root@localhost ~]# docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
docker.io/hello-world   latest              fce289e99eb9        7 months ago        1.84 kB
my-hello-world          latest              fce289e99eb9        7 months ago        1.84 kB
[root@localhost ~]#
```

This would create another image linking to the source image.

If you observe the IMAGE ID, for both the images are same.

```
[root@localhost ~]# docker image rmi my-hello-world
Untagged: my-hello-world:latest
Untagged: docker.io/hello-world@sha256:6540fc08ee6e6b7b63468dc3317e3303aae178cb8a45ed3123180328bcc1d2
[root@localhost ~]# docker image ls
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
docker.io/hello-world   latest              fce289e99eb9        7 months ago        1.84 kB
[root@localhost ~]#
```

Note: -- New tag name that was created from the “hello-world” image does not consume any extra space on the disk.

It’s just a extra name to the same image.

A very use full feature to maintain the images in a structured way.

3. Run the First Container.

Let's first list all the containers

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
              NAMES
[root@localhost ~]#
```

$ docker ps

This will list all the running containers.

Note: -- Currently there are no Running containers

$ docker ps -a

This would list all the running and stopped containers

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
              NAMES
[root@localhost ~]#
```

Note: -- Currently there are no Running and Stopped containers

$ docker run hello-world

This will start the container from the image **<hello-world> and stop it after printing the content.**

As below.

```
[root@localhost ~]# docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
```

Let's check the container list again.

$ docker ps

```
[root@localhost ~]# docker ps
CONTAINER ID        IMAGE               COMMAND
              NAMES
[root@localhost ~]#
```

We still don't see any containers.

Let's now check all container list.

$ docker ps -a

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
  PORTS               NAMES
2ea73691568f        hello-world         "/hello"            3 minutes ago       Exited (0) 3 minutes ago
                      xenodochial_bohr
[root@localhost ~]#
```

The container ran from the image and stopped as there is no continuous process in it.

4. Delete the Container.

$ docker rm ***<container ID>***

```
[root@localhost ~]# docker container rm 2ea73691568f
2ea73691568f
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS
              NAMES
[root@localhost ~]#
```

To force the container deletion add "-f", if there is an error or warning.

$ docker rm ***<container ID>*** ***-f***

To delete all the containers that are stopped all at once, we can use "prune".

$ ***docker container prune***

```
[root@localhost ~]# docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
82b6a1ce85543e74a30fd91f05a07d0a50838d16b3a6ac71113f1828e1cf797d

Total reclaimed space: 0 B
[root@localhost ~]#
```