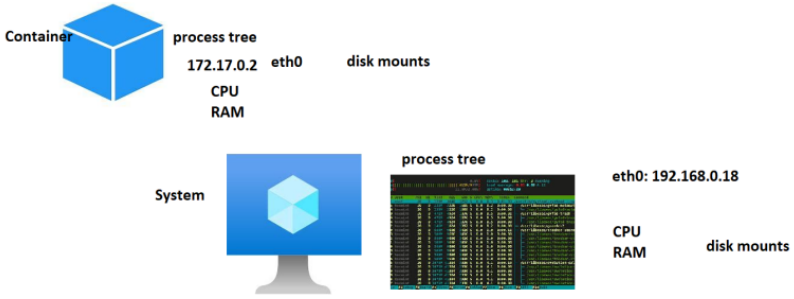
MENU ☰

APRIL 8, 2023

DevOps Classroomnotes 08/Apr/2023

How Isolations are created or How Containers Work

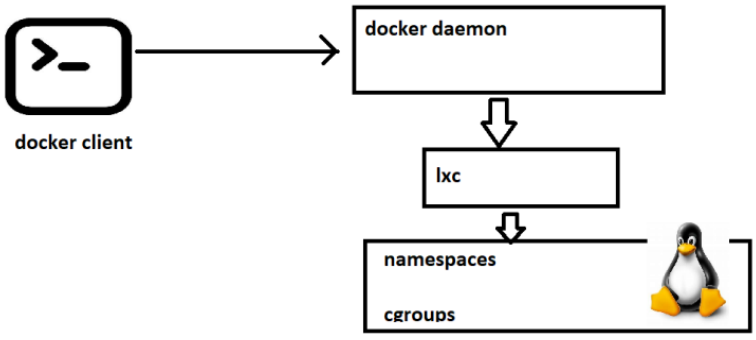
- Each container is getting a
  - new process tree
  - disk mounts
  - network (nic)
  - cpu/memory
  - users
- Refer Here for Docker Internals



Docker Architecture

Generation 1:

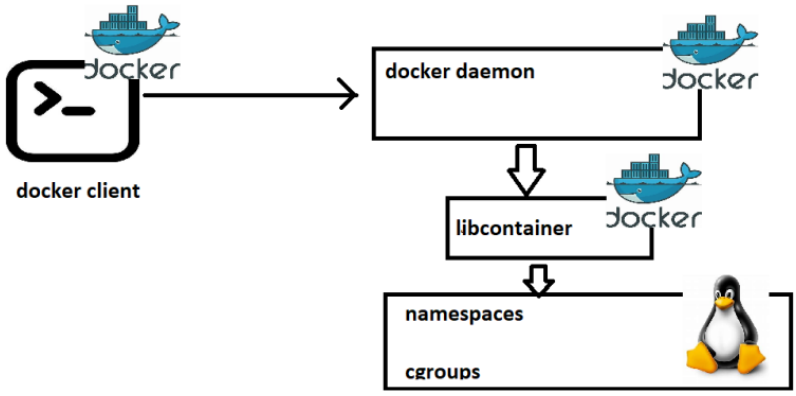
- This was first gen, Where docker daemon used lxc (a linux kernel feature) to create containers



Generation2:

- Since docker was relying on lxc which was kernel feature, updates to kernel frequently used to break containers created by docker.
- So docker has created its own component called libcontainer (libc) to create containers.

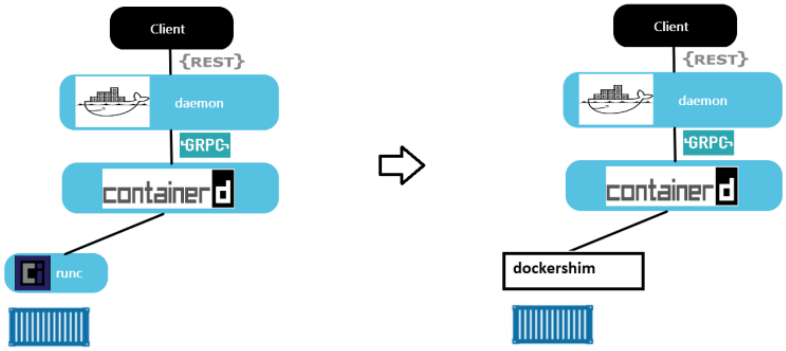
- Docker wanted containers to be multi os and lxc was definetly not the way forward.



- Adoption of docker was drastically increased as it was stable.

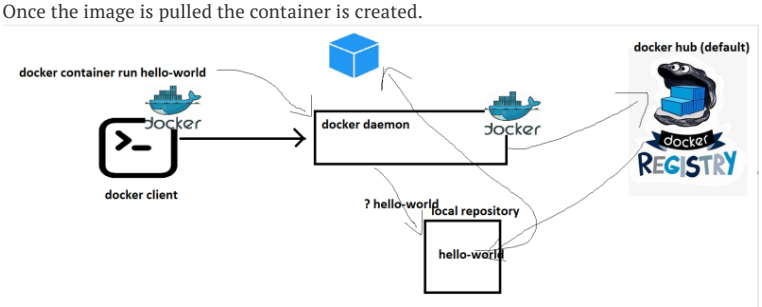
Generation 3:

- In this generation, docker engine was revamped from monolith to multi component architecture and the images and containers were according to OCI (open container initiative) image spec and runtime spec.
- In the latest architecture
- docker daemon exposes api's to listen requests from docker client.
- Passes the requests to containerd. This manages the lifecylcle of container
- containerd forks a runc process which creates container. once the container is created the parent of the container will be docker shim



Creating our first docker container

- docker container creation:
- To create container we need some image in this case lets take `hello-world`
- The command `docker container run hello-world` executed
- What happens
  - docker client will forward the request to docker daemon
  - docker daemon will check if the image exists locally. if yes creates the container by using image
  - if the image doesnot exist, then docker daemon tries to download the image from docker registry connected. The default docker registry is docker hub.
  - Downloading image into local repo from registy is called as pull.
  - Once the image is pulled the container is created.



- Registry is collection of docker images hosted for reuse.
- Docker hub Refer Here

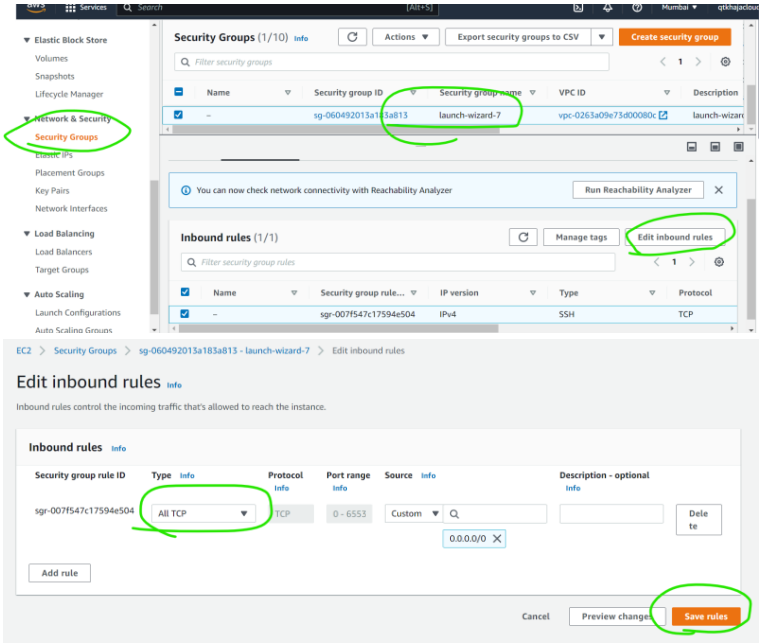
Playing with containers

- Create a new linux vm and install docker in it

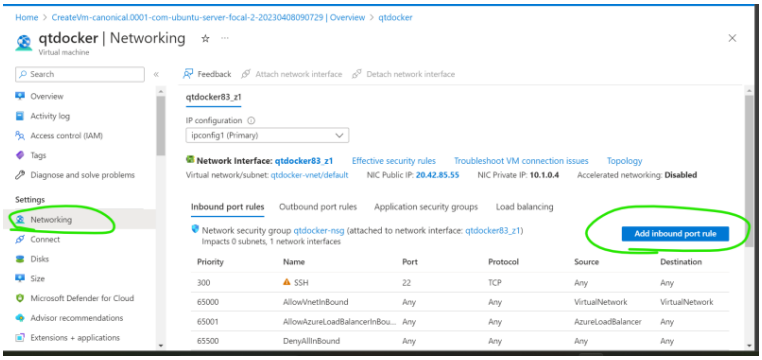
```
Azure  AWS

Last login: Sat Apr  8 03:44:09 2023 from 183.83.38.235
Dell@qtdocker:~$
Dell@qtdocker:~$ docker version
Client: Docker Engine - Community
 Version:           23.0.3
 API version:       1.42
 Go version:        go1.19.7
 Git commit:        3e7cbfd
 Built:             Tue Apr  4 22:06:10 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          23.0.3
  API version:      1.42 (minimum version 1.12)
  Go version:       go1.19.7
  Git commit:       59118bf
  Built:            Tue Apr  4 22:06:10 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
```

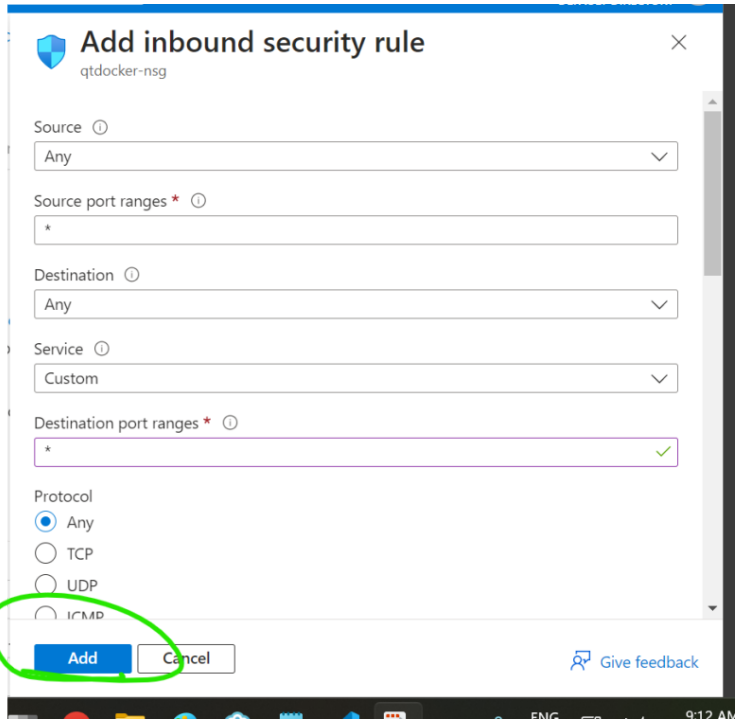
```
Azure  AWS

ubuntu@ip-172-31-35-3:~$
ubuntu@ip-172-31-35-3:~$ docker version
Client: Docker Engine - Community
 Version:           23.0.3
 API version:       1.42
 Go version:        go1.19.7
 Git commit:        3e7cbfd
 Built:             Tue Apr  4 22:05:48 2023
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          23.0.3
  API version:      1.42 (minimum version 1.12)
  Go version:       go1.19.7
  Git commit:       59118bf
  Built:            Tue Apr  4 22:05:48 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
```

- Open all the ports
  - AWS

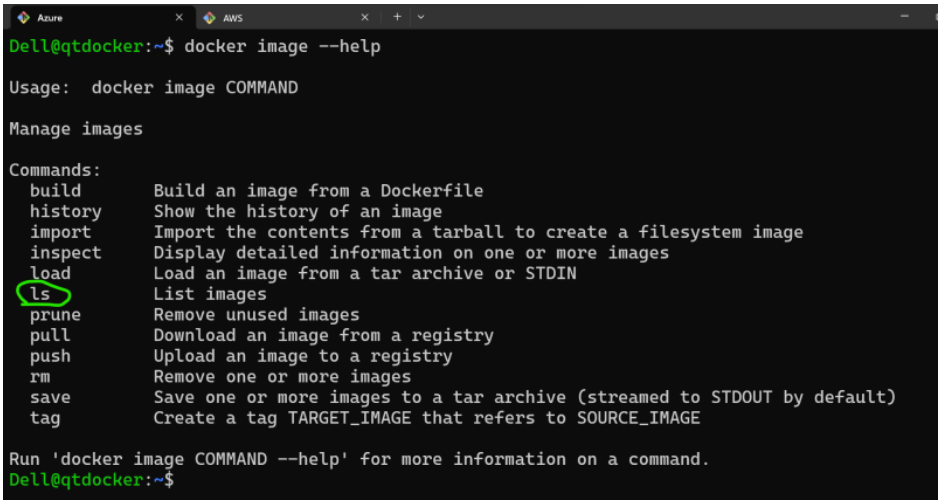


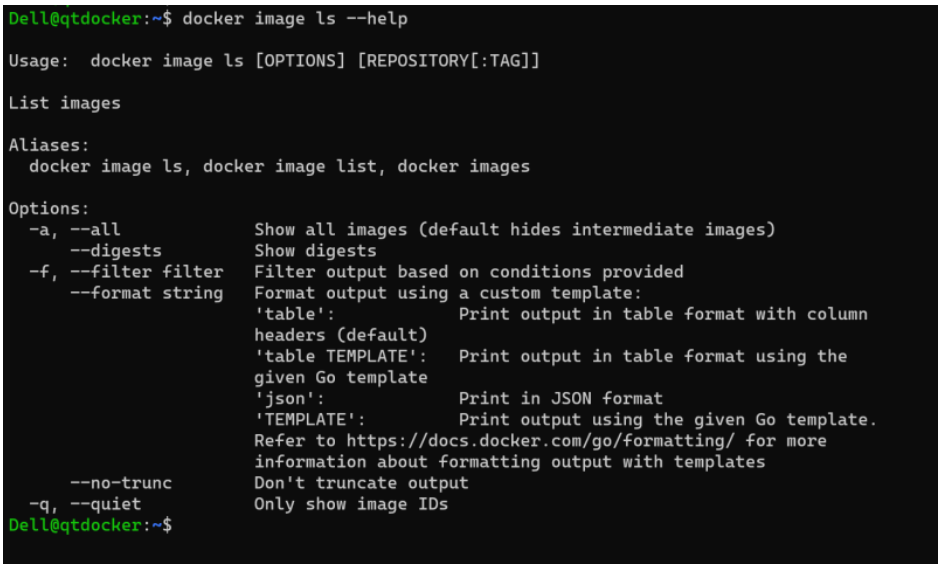
- Azure

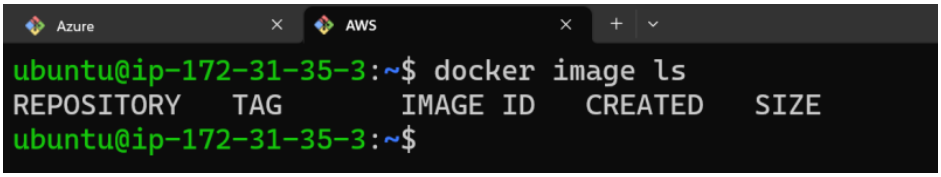




Check docker images in the host







pull the images from docker hub

- image naming convention

```
[username]/[repository]:[<tag>]
shaikkhajaibrahim/myspc:1.0.1
username => shaikkhajaibrahi
repository => what image => myspc
tag => version => 1.0.1
```

- default tag is latest

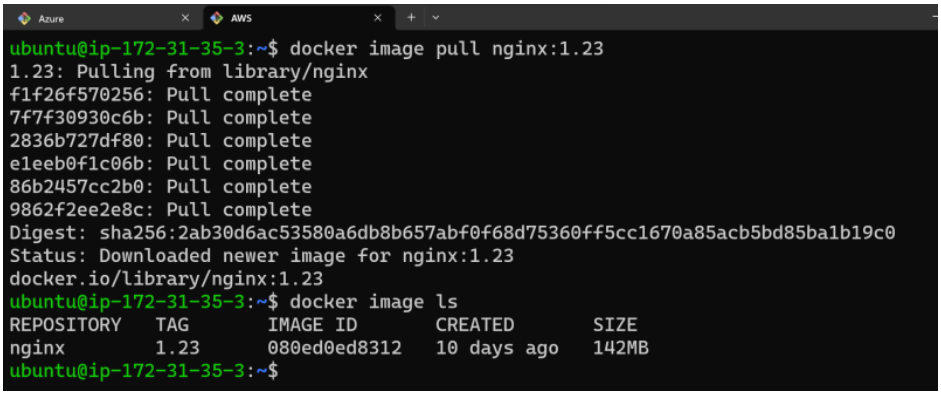
```
nginx
nginx:latest
```

- official images dont have username

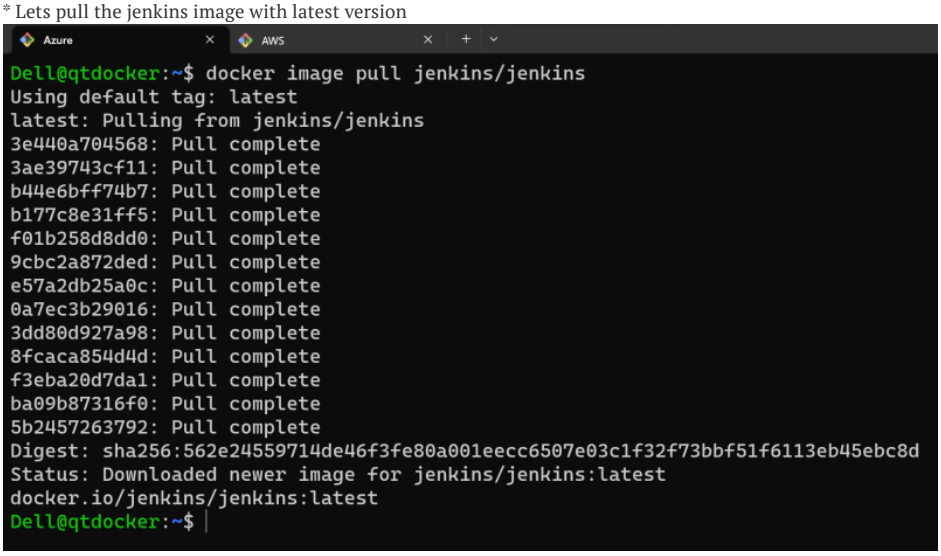
```
nginx
ubuntu
alpine
shaikkhajaibrahim/myspc
```

- Lets pull the image nginx with tag `1.23`

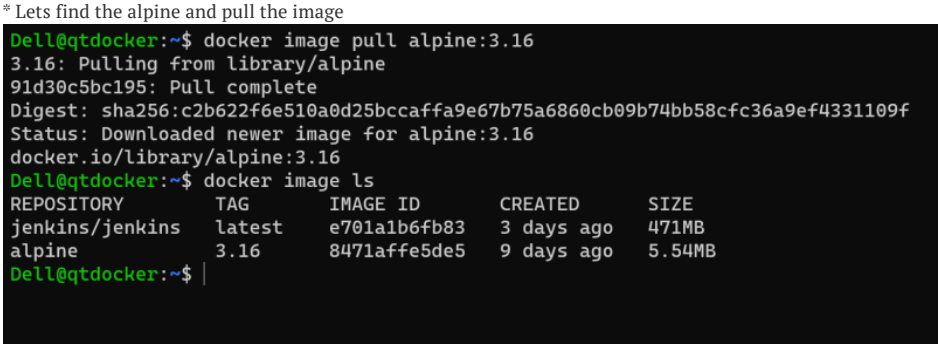
```
docker image pull nginx:1.23
docker image ls
```



* Lets pull the jenkins image with latest version

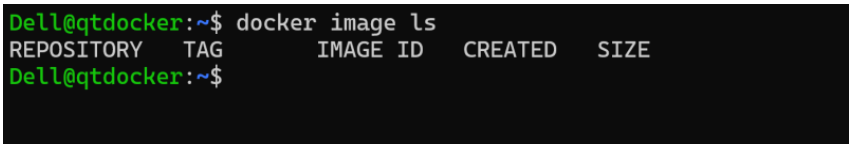


* Lets find the alpine and pull the image



Remove images from local

- Every image will have unique image id and image name
- We can delete individually `docker image rm alpine:3.17`
- if i have to delete all the images `docker image rm $(docker image ls -q)



Create a container with nginx

- To create and start the container we use run command

```
Dell@qtdocker:~$ docker container run -d nginx:1.22
Unable to find image 'nginx:1.22' locally
1.22: Pulling from library/nginx
f1f26f570256: Already exists
fd03b214f774: Pull complete
ef2fc869b944: Pull complete
ac713a9ef2cc: Pull complete
fd071922d543: Pull complete
2a9f38700bb5: Pull complete
Digest: sha256:fc5f5fb7574755c306aaf88456ebfbe0b006420a184d52b923d2f0197108f6b7
Status: Downloaded newer image for nginx:1.22
340e9bca8bf6610c1061250071bcb825737beba63bad2a6c74d9a0a88924c7dc
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         P
ORTS      NAMES
340e9bca8bf6   nginx:1.22   "/docker-entrypoint.…"   6 seconds ago   Up 5 seconds   8
0/tcp     mystifying_napier
Dell@qtdocker:~$
```

- note: i will be using -d for some time and we will discuss importance of this in next session
- every container gets an id and a name. name can be passed while creating container, if not docker will give random name

```
Dell@qtdocker:~$ docker container run --name mynginx1 -d nginx:1.22
6531f97fd9c2e4e77607e13c57684c4f14c9c4f5106736a1908c463ec10bc228
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS
 NAMES
6531f97fd9c2   nginx:1.22   "/docker-entrypoint.…"   8 seconds ago   Up 8 seconds   80/tcp
 mynginx1
340e9bca8bf6   nginx:1.22   "/docker-entrypoint.…"   2 minutes ago   Up 2 minutes   80/tcp
 mystifying_napier
Dell@qtdocker:~$
```

- Remove all the running containers `docker container rm -f $(docker container ls -q )`

```
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS          PORTS
   NAMES
7d7f907d8dfa   nginx:1.22   "/docker-entrypoint.…"   5 seconds ago    Up 4 seconds    80/tcp
   mynginx3
27e4c7416ecc   nginx:1.22   "/docker-entrypoint.…"   9 seconds ago    Up 7 seconds    80/tcp
   mynginx2
ef62979992ee   nginx:1.22   "/docker-entrypoint.…"   14 seconds ago   Up 13 seconds   80/tcp
   mynginx1
340e9bca8bf6   nginx:1.22   "/docker-entrypoint.…"   5 minutes ago    Up 5 minutes    80/tcp
   mystifying_napier
Dell@qtdocker:~$ docker container ls -q
7d7f907d8dfa
27e4c7416ecc
ef62979992ee
340e9bca8bf6
Dell@qtdocker:~$ docker container rm -f $(docker container ls -q)
7d7f907d8dfa
27e4c7416ecc
ef62979992ee
340e9bca8bf6
Dell@qtdocker:~$
```

```
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
Dell@qtdocker:~$ docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
Dell@qtdocker:~$
```

- Remove specific container

```
Dell@qtdocker:~$ docker container rm mynginx1
Error response from daemon: You cannot remove a running container 6531f97fd9c2e4e77607e13c576
84c4f14c9c4f5106736a1908c463ec10bc228. Stop the container before attempting removal or force
remove
Dell@qtdocker:~$ docker container rm -f mynginx1
mynginx1
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS
 NAMES
340e9bca8bf6   nginx:1.22   "/docker-entrypoint.…"   4 minutes ago   Up 4 minutes   80/tcp
 mystifying_napier
Dell@qtdocker:~$
```

- Remove all containers `docker container rm -f $(docker container ls -a -q )`

```
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS          PORTS     NAMES
d983e8b31c2b   nginx:1.22   "/docker-entrypoint.…"   7 seconds ago    Up 6 seconds    80/tcp    mynginx3
8401c1b26a08   nginx:1.22   "/docker-entrypoint.…"   14 seconds ago   Up 13 seconds   80/tcp    mynginx2
778c25cd3830   nginx:1.22   "/docker-entrypoint.…"   20 seconds ago   Up 19 seconds   80/tcp    mynginx1
Dell@qtdocker:~$ docker container stop mynginx2
mynginx2
Dell@qtdocker:~$ docker container ls
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS          PORTS     NAMES
d983e8b31c2b   nginx:1.22   "/docker-entrypoint.…"   28 seconds ago   Up 28 seconds   80/tcp    mynginx3
778c25cd3830   nginx:1.22   "/docker-entrypoint.…"   41 seconds ago   Up 41 seconds   80/tcp    mynginx1
Dell@qtdocker:~$ docker container ls -a
CONTAINER ID   IMAGE        COMMAND                  CREATED          STATUS
     PORTS     NAMES
d983e8b31c2b   nginx:1.22   "/docker-entrypoint.…"   43 seconds ago   Up 42 seconds
     80/tcp    mynginx3
Dell@qtdocker:~$
778c25cd3830   nginx:1.22   "/docker-entrypoint.…"   56 seconds ago   Up 55 seconds
     80/tcp    mynginx1
Dell@qtdocker:~$
```

- Exercise: Start and stop containers

Leave a Reply

Enter your comment here...

This site uses Akismet to reduce spam. Learn how your comment data is processed.

About continuous learner

devops & cloud enthusiastic learner

VIEW ALL POSTS

PREVIOUS POST

AWS Classroomnotes 07/Apr/2023

NEXT POST

Terraform Book

POWERED BY WORDPRESS.COM.