SELECTING A DOCKER STORAGE DRIVER

Docker volumes are used to write data.

Docker uses storage drivers that makes data porting much easier for container and image related data

How to know what is your current storage driver is?

```
[root@ip-172-31-44-133 Dockerfiles]# docker info |grep Storage  ----->[ note:  S capital  ]
Storage Driver: overlay2
```

```
cd /etc/docker [sudo su ]
```

```
[root@ip-172-31-44-133 docker]#  ls
key.json
```

we shall override this json with daemon.json which will be looked upon while restarting docker

```
vi daemon.json
{
"storage-driver":"devicemapper"
}
```

```
systemctl stop docker
```

```
systemctl start docker
```

```
[root@ip-172-31-44-133 docker]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE ---> all images gone
```

```
docker pull centos:6
```

```
root@ip-172-31-44-133 devicemapper]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
centos              6               70b5d81549ec        3 months ago        195MB
```

```
oot@ip-172-31-44-133 devicemapper]# docker info | grep Storage
Storage Driver: devicemapper
WARNING: devicemapper: usage of loopback devices is strongly discouraged for production use.
         Use `--storage-opt dm.thinpooldev` to specify a custom block storage device.
```

PREPARE FOR A DOCKER SECURE REGISTRY

from home (/home/ec2-user) We shall use two new directories -> certs & auth. Auth will hold user authentication and certs shall hold the certificates that we are creating now.

```
mkdir auth
mkdir certs
```

Note: download the openssl incase if we dont have in our system

```
yum install openssl
```

```
cd /home/ec2-user [ home directory ]
```

```
[root@ip-172-31-44-133 ec2-user]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout certs/dockerrepo.key -x509 -days 365 -out certs/dockerrepo.crt -subj /CN=myregistrydomain.com
Generating a 4096 bit RSA private key
.....................++
..................................
```

```
[root@ip-172-31-44-133 ec2-user]# ls -la certs
total 8
```

to enable this, lets do an ifconfig

```
[root@ip-172-31-44-133 ec2-user]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet6 fe80::42:c2ff:fe0c:b7cc  prefixlen 64  scopeid 0x20<link>
        ether 02:42:c2:0c:b7:cc  txqueuelen 0  (Ethernet)
        RX packets 122817  bytes 6678000 (6.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 177646  bytes 520089075 (495.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.31.44.133  netmask 255.255.240.0  broadcast 172.31.47.255
        inet6 fe80::8fb:8fff:fe0c:7a22  prefixlen 64  scopeid 0x20<link>
        ether 0a:fb:8f:0c:7a:22  txqueuelen 1000  (Ethernet)
        RX packets 1099961  bytes 1134132076 (1.0 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 555552  bytes 73025256 (69.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

we need to put this onto our host file

```
[root@ip-172-31-44-133 ec2-user]# vim /etc/hosts [1]
```

with /etc/hosts file, just paste the ip address and domain name that we created, at last line

```
172.31.19.24 myregistrydomain.com
```

```
[root@ip-172-31-44-133 ec2-user]# mkdir -p /etc/docker/certs.d/myregistrydomain.com:5000
```

```
[root@ip-172-31-44-133 ec2-user]#  cd /etc/docker/certs.d/myregistrydomain.com:5000
```

```
[root@ip-172-31-44-133 myregistrydomain.com:5000]# cp /home/ec2-user/certs/dockerrepo.crt ca.crt
```

Now pull the registry image from the docker hub

```
[root@ip-172-31-44-133 myregistrydomain.com:5000]# docker pull registry:2
```

□□□ This has to be done ONLY at the /home/ec2-user path and NOT anywhere else.

```
[root@ip-172-31-44-133 ec2-user]# docker run --entrypoint htpasswd registry:2 -Bbn test password >auth/htpasswd
```

```
[root@ip-172-31-44-133 auth]# cat /home/ec2-user/auth/htpasswd
```

IMAGE MANIPULATION IN THE REGISTRY

```
[root@ip-172-31-44-133 ec2-user]# docker run -d \
  -p 5000:5000 \
  --restart=always \
  --name myregistry6 \
  -v `pwd`/auth:/auth \
  -e "REGISTRY_AUTH=htpasswd" \
  -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" \
  -e REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd \
  -v `pwd`/certs:/certs \
  -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/dockerrepo.crt \
  -e REGISTRY_HTTP_TLS_KEY=/certs/dockerrepo.key \
  registry:2

acf572033d96e9c3ee4c56a85f990415f8b664a5611380f478903c50ae1d3c59
```

□□□□DONT COPY PASTE !   BE VERY CAREFUL IN GIVING THIS LENGTHY COMMAND.
IF given wrongly, the container either wont be created or will not be able to run and shall exit

```
[root@ip-172-31-44-133 ec2-user]# docker ps
```

```
[root@ip-172-31-44-133 ec2-user]# docker pull busybox
```

```
[root@ip-172-31-44-133 ec2-user]# docker images
```

```
[root@ip-172-31-44-133 ec2-user]# docker tag busybox myregistrydomain.com:5000/my-busybox
```

```
[root@ip-172-31-44-133 ec2-user]# docker push myregistrydomain.com:5000/my-busybox
```

```
[root@ip-172-31-44-133 docker]# docker push myregistrydomain.com:5000/my-busybox
```

```
[root@ip-172-31-44-133 docker]# docker login myregistrydomain.com:5000
```

```
[root@ip-172-31-44-133 docker]# docker push myregistrydomain.com:5000/my-busybox
```

```
[root@ip-172-31-44-133 docker]# docker rmi busybox
```

```
[root@ip-172-31-44-133 docker]# docker pull myregistrydomain.com:5000/my-busybox
```

```
[root@ip-172-31-44-133 docker]# docker images
REPOSITORY                          TAG             IMAGE ID         CREATED           SIZE
myregistrydomain.com:5000/my-busybox   latest          e1ddd7948a1c       17 hours ago        1.16MB
□□□□
```

[1] WHY HOST FILE??: (/etc/host)

Domain Name Servers (DNS) bind the internet together. They match machine-usable IPv4 and IPv6 addresses with human-readable domain names. It's the internet's equivalent of the phone book. But, while DNS is utterly essential to the internet, it's not the only way to connect IP addresses with domain names. You can also use Hosts files to connect mysterious IP addresses with meaningful domain names.

A Hosts file is a file that almost all computers and operating systems can use to map a connection between an IP address and domain names.

This file is an ASCII text file. It contains IP addresses separated by a space and then a domain name. Each address gets its own line. For example, 64.30.228.118 is the IP address for CBS Interactive, ZDNet's parent company.

To place this in a Host file you'd enter the following line with a text editor, such as Windows' notepad or Linux's vi.

```
64.30.228.118 cbsinteractive.com
```