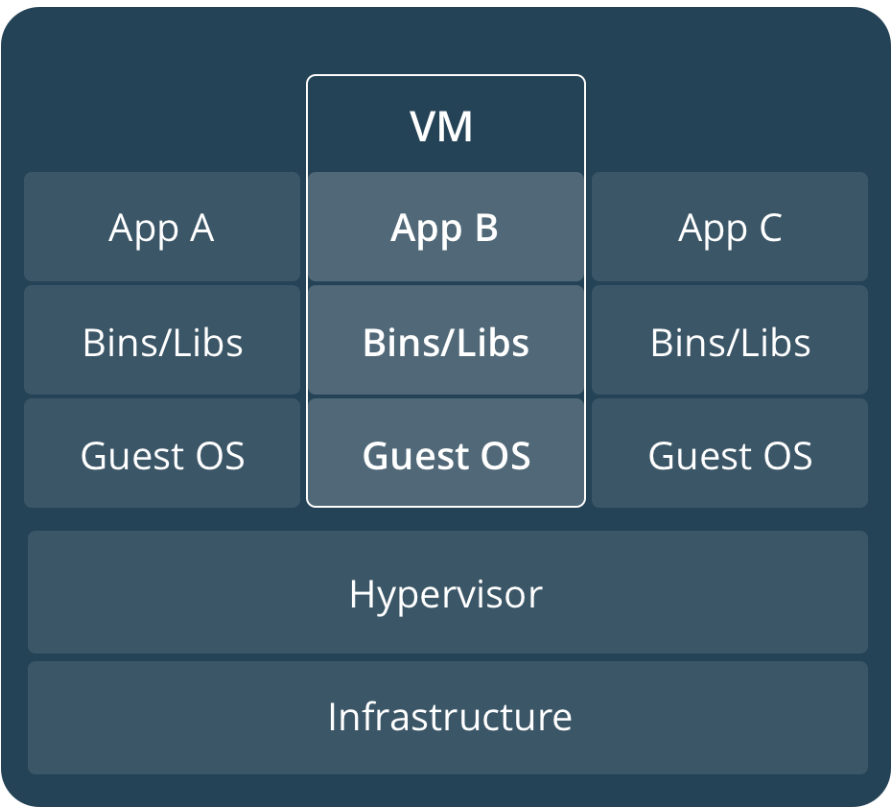
**DOCKER**

**DIFFERENTIATE BETWEEN VIRTUALIZATION VS CONTAINERIZATION**:

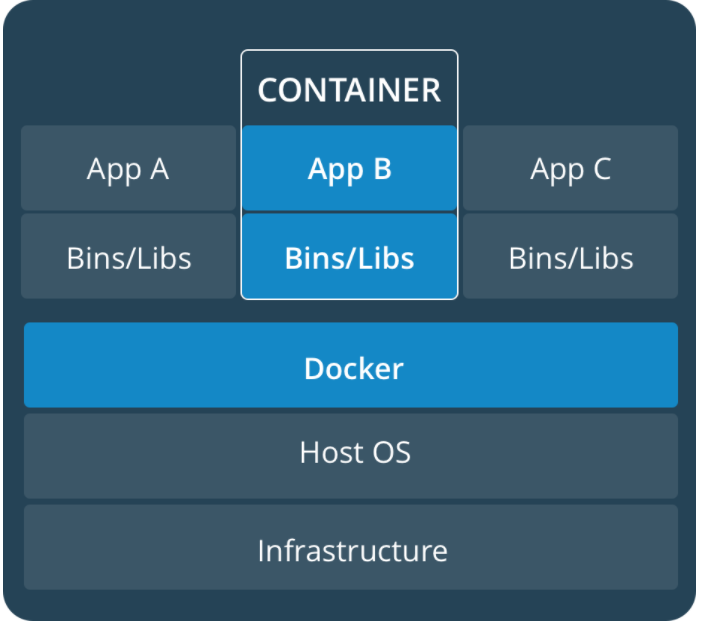
* **VIRTUALIZATION**:

Virtualization is the technique of importing a guest Operating System on top of Host Operating System. This technique was a revelation in the beginning because developers run multiple applications in different virtual machines all running on the same host this eliminated the need for extra hardware resource and enable backup allowing for easy recovery in case of failure condition thereby lowering the total cost of ownership but virtualization has some shortcomings running multiple VMs in the same host.



* **ADVANTAGES OF VIRTUALIZATION**:
  + - * Multiple OS in same machine
      * Easy maintenance and recovery
      * Lower cost of ownership
* **DISADVANTAGES OF VIRTUALIZATION**:
  + - * Multiple VMs lead to unstable performance
      * Hypervisiors are not as efficient as Host OS
      * Long Boot-up Process
* **CONTAINERIZATION**:

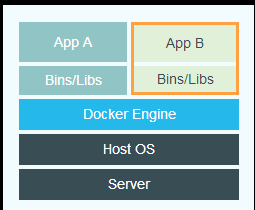
Containerization brings abstraction to the software. Containerization is more efficient because there is no guest OS. Binaries and Libraries of containers are on host kernel which makes processing and execution very fast. Booting up takes only fraction of second (Because all containers run on same host, light weight and faster than virtual machines)



* **ADVANTAGES OVER VIRTUALIZATION**:
* Containers on same OS kernel are Lighter and smaller
* Better resource utilization compared to VMs
* Short Boot-up Process

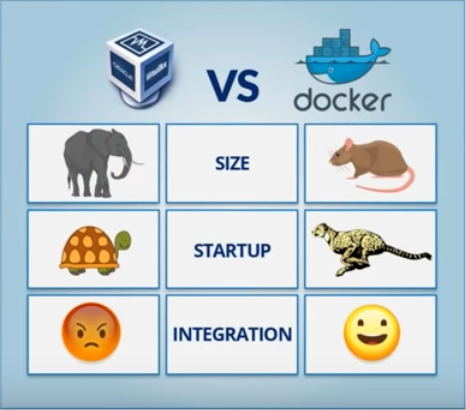
**WHAT IS DOCKER?**

Docker is a Containerization Platform which packages our application and all its dependencies together in the form of containers. To ensure that your application works seamlessly in any environment be it Development or Testing or Production.



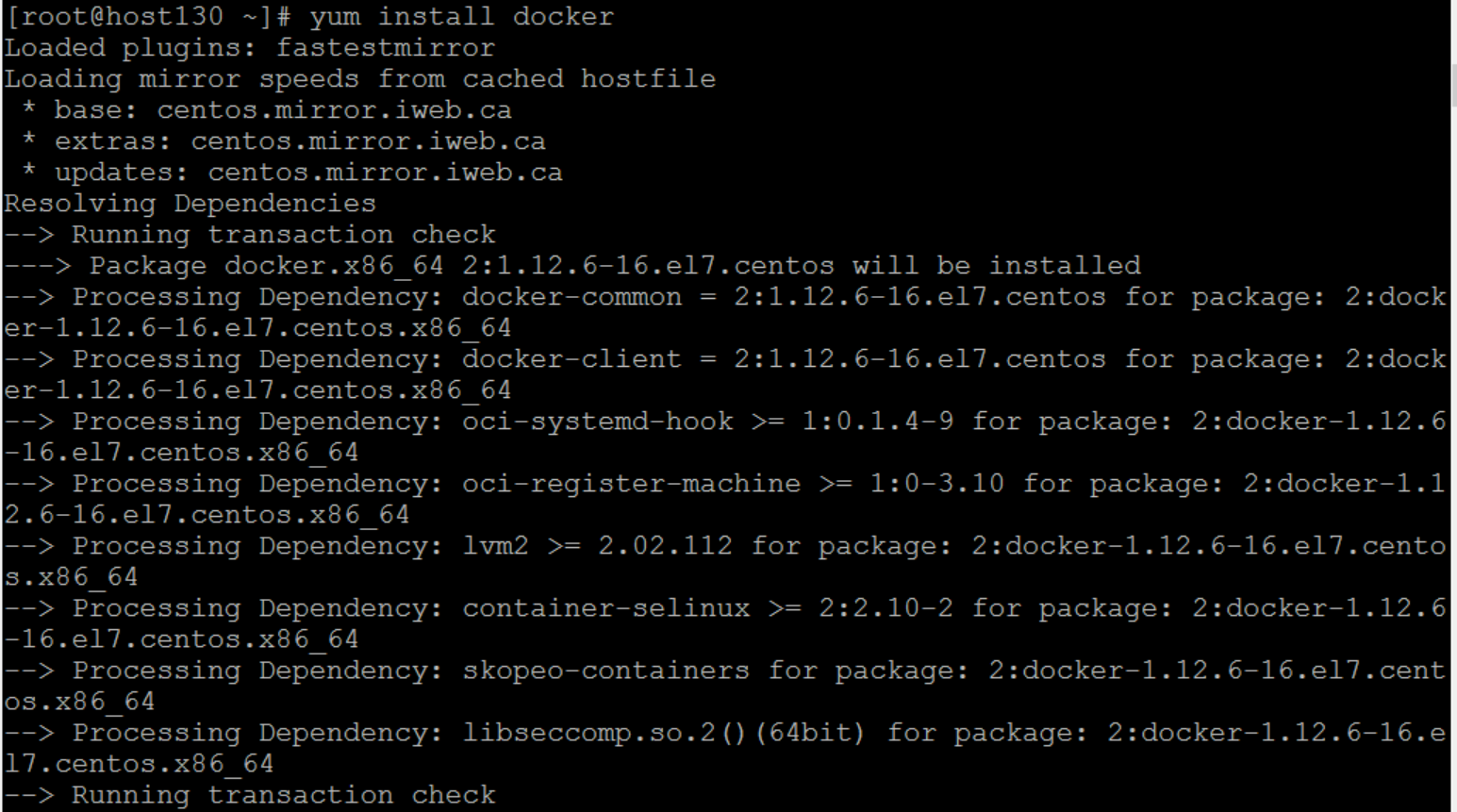
**ADVANTAGES OF DOCKER OVER VIRTUALIZATION :**

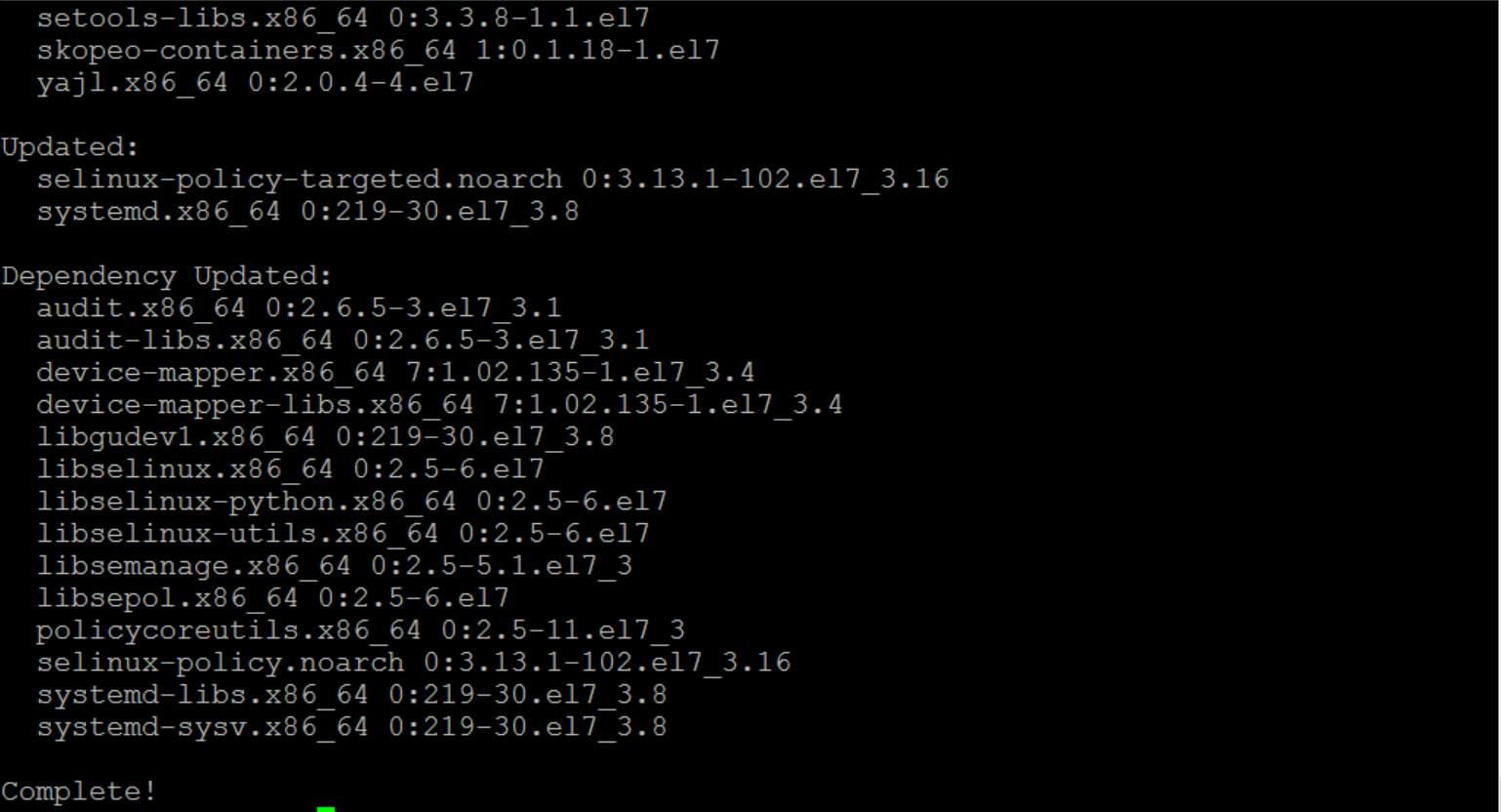
|  |  |  |
| --- | --- | --- |
| VIRTUALIZATION | DOCKER |  |
| BIG | SMALL | SIZE |
| SLOW | FAST | START-UP |
| POSSIBLE BUT HAS COMPLICATIONS | COMFORTABLE IN INTEGRATING WITH MANY DEVOPS TOOLS | INTEGRATION |



**INSTALLATION OF DOCKER**:

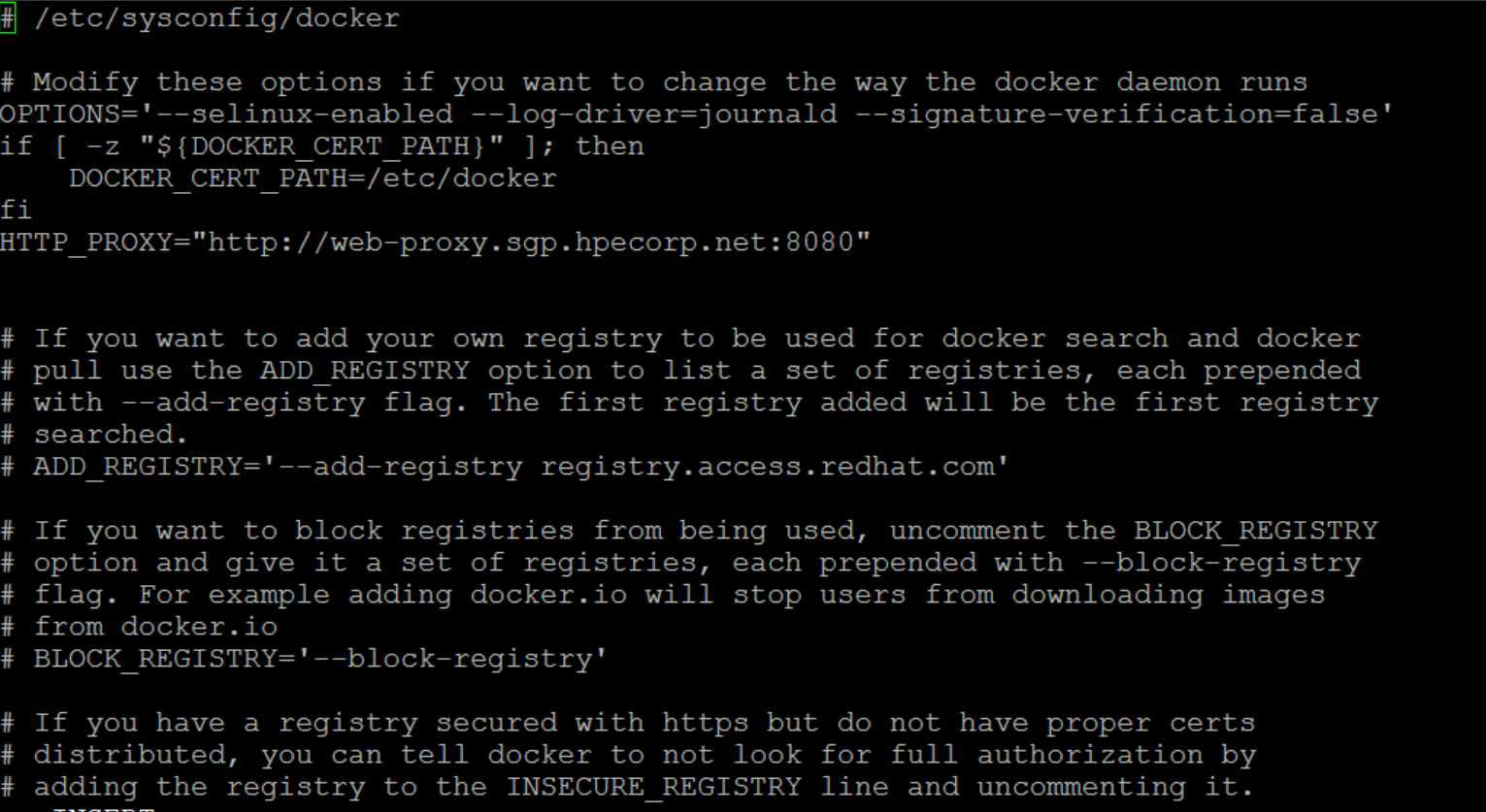
#Yum install docker





**Adding Proxy**:

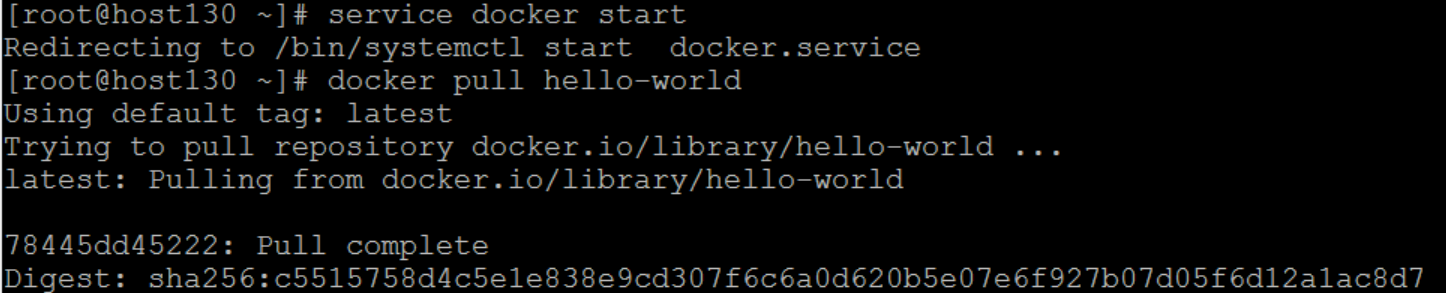
#Vi /etc/sysconfig/docker

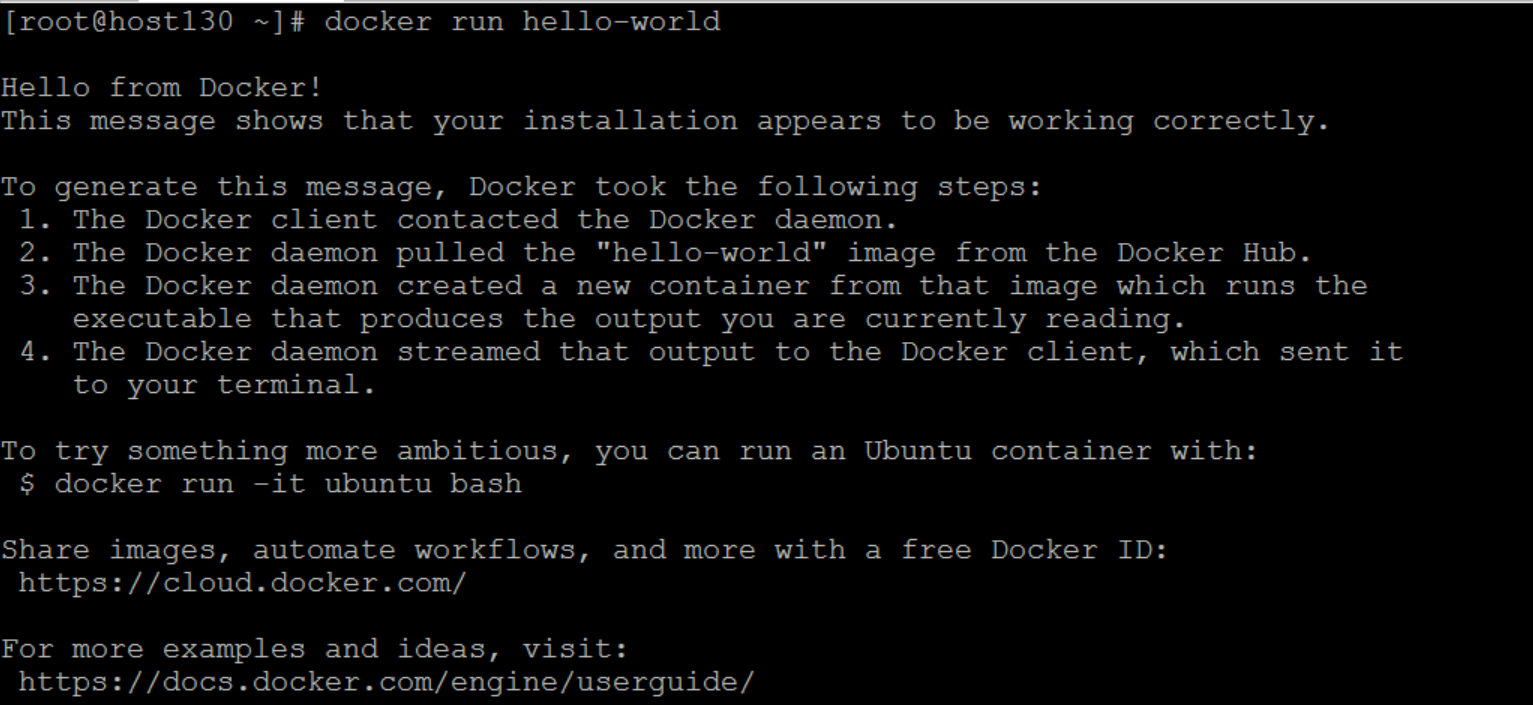
HTTP\_PROXY=”http://web-proxy.sgp.hpecorp.net:8080”  


**Example**:

Let check if the docker is working or not. Run te command

#docker pull hello-world





**HOW DOCKER IS USED IN DEVOPS**:

Docker is used 90% of time in continuous testing phase and 10% of time in continuous deployment time, because docker sets up the testing environment. It is also used to deploy code to production.

In Centos docker root directory is

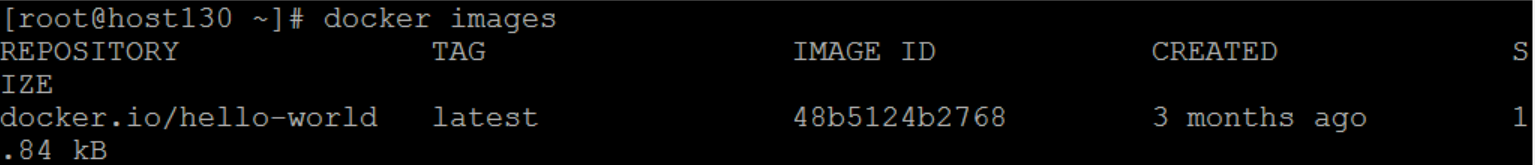
# cd /var/lib/docker

**HOW DOCKER WORKS**:

* Docker engine is the heart of the docker system.
* It works like a client-server application which uses a server called daemon-process and command line interface, the client.
* The REST API with combination of socket IO and TCP/IP connection is used for communication between the command line interface and docker daemon.
* In linux operating system, there is a docker client which can be accessed from terminal and docker host which runs docker daemon.
* We build docker images and run the docker containers by passing commands from command line interface to docker daemon.

**DOCKER IMAGES**:

* + - Docker images are used to create Docker containers. Docker provides a simple way to build new images or update existing images. Docker images are the **build** component of docker.
    - Docker images are related to executable file. It is a building block of docker container and it created with build command.



**DOCKER CONTAINER**:

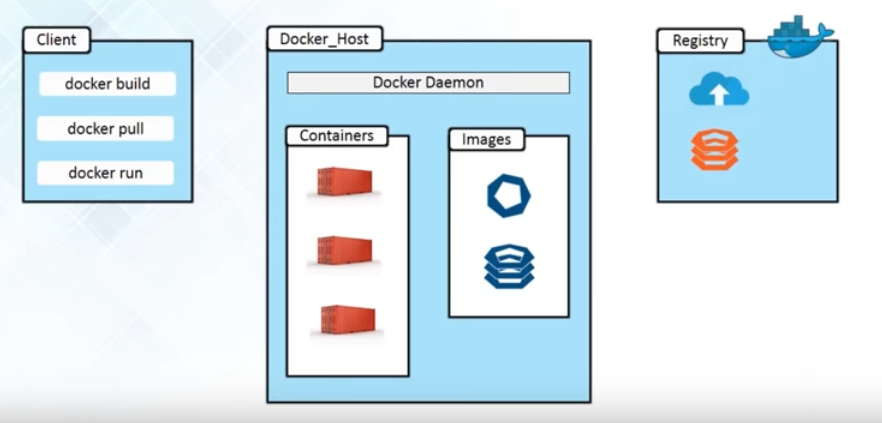
* + - Docker containers are created from docker images.They hold everything that is needed for an application to run. Each container is an isolated and secure applicaton platform. Docker containers are the **run** component of docker.
    - Read only templates used to create container using **run** command.
    - Containers are instance of images and hold the entire package that is needed to run the application.

**DOCKER REGISTRY**:

* + - Docker registry is where all images are stored.
    - The registry can be either a user’s local repository or a public repository so that multiple users can collaborate and build an application.
    - Example : Multiple teams within same organization can share containers by uploading them to docker hub.

**DOCKER ARCHITECTURE**:

* + - Docker client
    - Docker host
    - Registry



* **DOCKER CLIENT**:

It is the primary user interface to docker. It accepts commands from the user and communicates back and forth with a docker daemon.

* **DOCKER DAEMON**:

Daemons run on host machine. Daemons create and manage docker objects, images, containers, networks, volumes, data etc,. The user does not directly interact with the daemon but instead through the docker client.

* **DOCKER REGISTRY**:

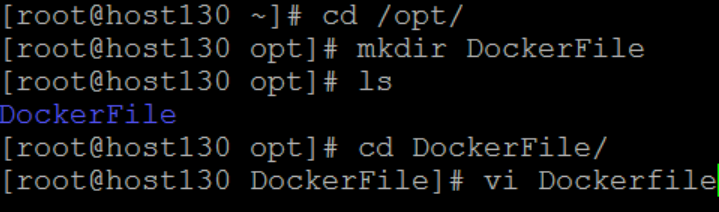
Registry store images. These may be private or public from which we can upload or download images. This can be done on docker hub which is docker’s version of github. Docker registry are the distribution component of docker.

**ARCHITECTURE**:

* Docker daemon is responsible for containers and images.
* We can build docker image through command line interface(client) using build command to docker daemon.
* Daemon build image based on imports and saved in registry which can be either docker hub or local repository.
* We can also pull images from docker hub which would have been built by a different user and contributed to the docker hub.
* To run the instance of any docker image, run command is given from command line interface which creates a container.

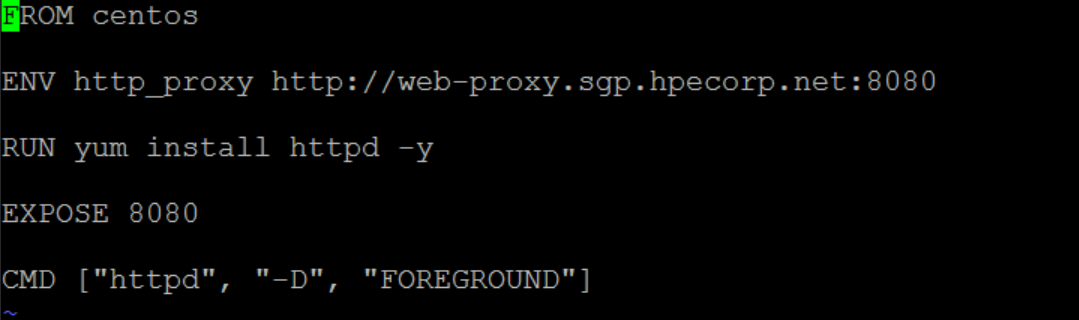
**DOCKER FILE**:

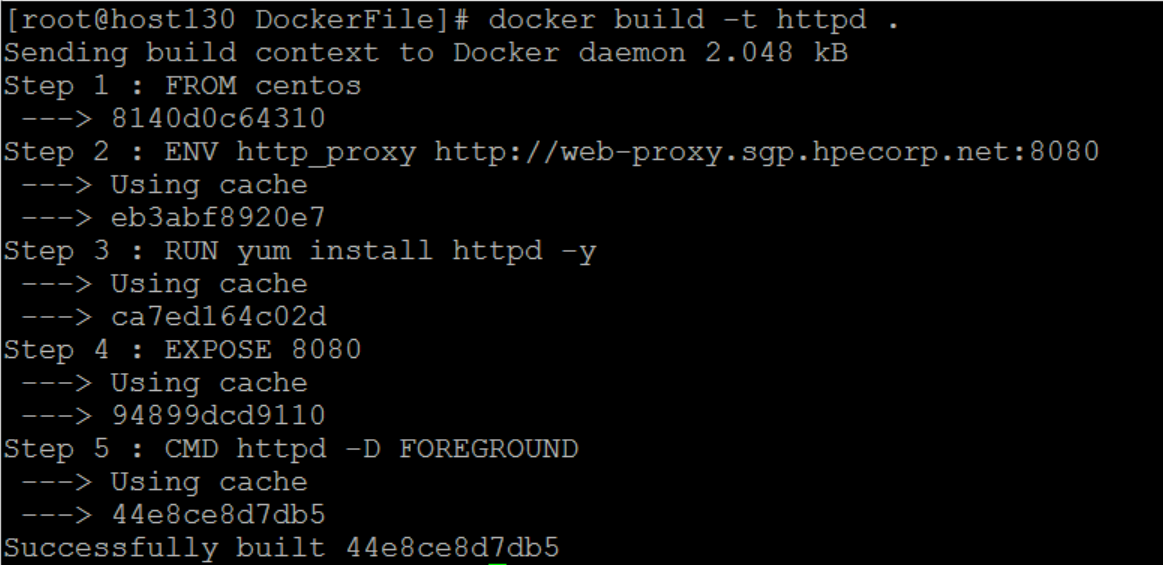
Docker can build images automatically by reading the instructions from a **Dockerfile**. A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image. Using **docker build** users can create an automated build that executes several command-line instructions in succession.



If you are behind proxy server in docker file also we need to add a proxy

# http\_proxy http://web-proxy.sgp.hpecorp.net:8080







The port number exposed by the container is 80. Hence with the –p command, we are mapping the same port number to the 80 port number on our localhost. The –d option is used to run the container in detached mode. This is so that the container can run in the background.

DOCKER COMMANDS:

#yum install epel-release

#yum install net-tools mlocate

#yum install docker-io

#systemctl start docker

#systemctl status docker

#systemctl enable docker

Check if docker is working fine:

#docker run hello-world

#docker info

#docker --version

#docker ps -a, -l

#docker run -it id =To run a container in an interactive mode

#docker rmi imageID

#docker history ImageID

By

N.Nivetha Rani (60116545)

J.Janani Booma(60116575)