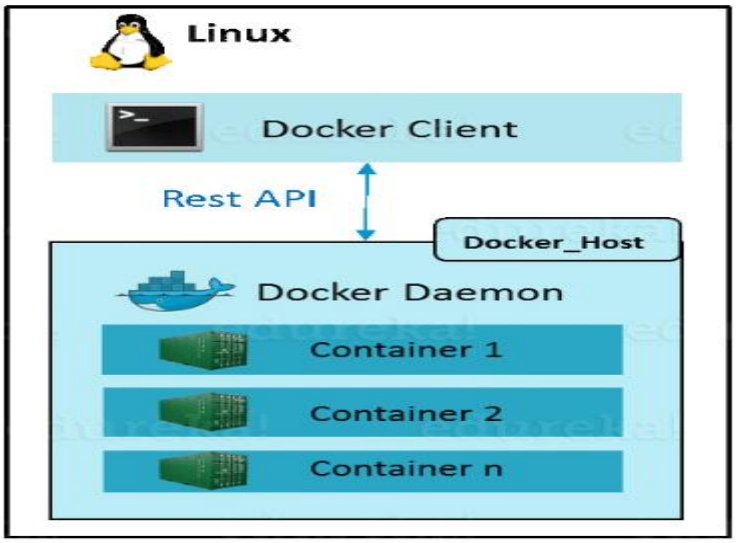
What is Docker?

-Docker is a containerization platform that packages your application and all its dependencies together in the form of a docker container to ensure that your application works seamlessly in any environment.

What is Docker Engine?

Now I will take you through Docker Engine which is the heart of the Docker system. Docker Engine is simply the docker application that is installed on your host machine. It works like a client-server application which uses:

- A **server** which is a type of long-running program called a daemon process
- A command line interface (CLI) **client**
- REST API is used for communication between the CLI client and Docker Daemon

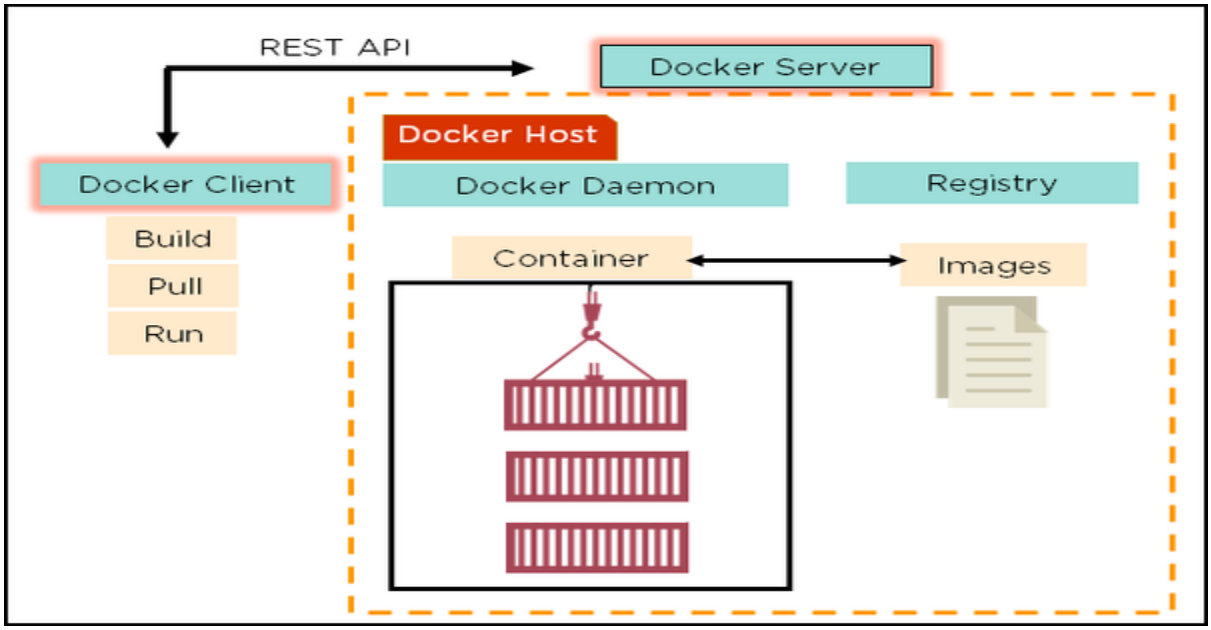


As per the above image, in a Linux Operating system, there is a Docker client which can be accessed from the terminal and a Docker Host which runs the Docker Daemon. We build our Docker images and run Docker containers by passing commands from the CLI client to the Docker Daemon.

What is Docker Architecture?

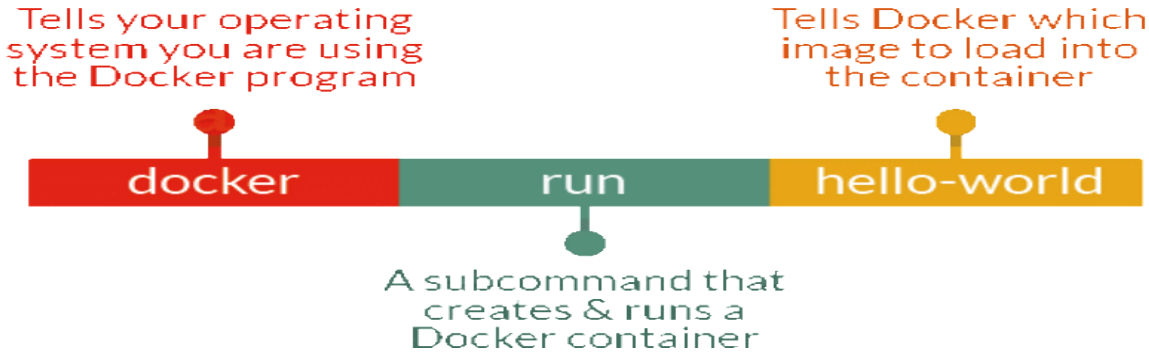
Docker Architecture includes a Docker client – used to trigger Docker commands, a Docker Host – running the Docker Daemon and a Docker Registry – storing Docker Images. The Docker Daemon running within Docker Host is responsible for the images and containers.

- To build a Docker Image, we can use the CLI (client) to issue a build command to the Docker Daemon (running on Docker_Host). The Docker Daemon will then build an image based on our inputs and save it in the Registry, which can be either Docker hub or a local repository
- If we do not want to create an image, then we can just pull an image from the Docker hub, which would have been built by a different user
- Finally, if we have to create a running instance of my Docker image, we can issue a run command from the CLI, which will create a Docker Container.



What is Docker Image?

Docker Image can be compared to a template which is used to create Docker Containers. They are the building blocks of a Docker Container. These Docker Images are created using the build command. These Read only templates are used for creating containers by using the run command.



Docker lets people (or companies) create and share software through Docker images. Also, you don't have to worry about whether your computer can run the software in a Docker image — a Docker container *can always run it*.

We can either use a ready-made docker image from docker-hub or create a new image as per my requirement.

Commands:

$ docker images (List of installed images)

$ docker run hello-world (To create image of hello world)

$ docker run namespace/image (Pull from docker hub)

$ docker rmi imageid (To delete image)

$ docker rmi -f imageid (For forcefully delete image)

$ docker rmi -f $(docker images) (To delete all images)

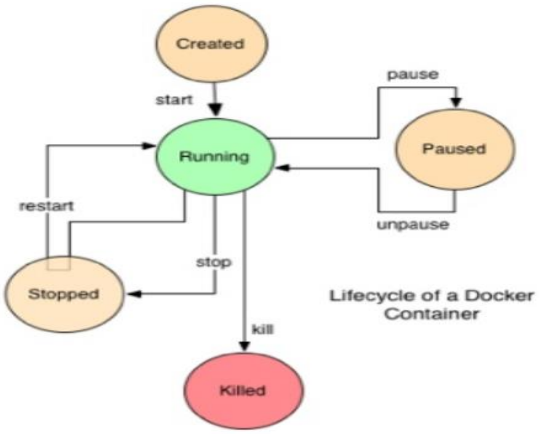
$ docker rm $(docker ps -a -q) (To remove all the containers over images)

$ docker rm $(docker ps -a -q -f) (For forceful delete)

What is Docker Container?

Containers are the ready applications created from Docker Images or you can say a Docker Container is a running instance of a Docker Image and they hold the entire package needed to run the application. This happens to be the ultimate utility of Docker.

Life Cycle of Container:



$ docker ps (Containers list present in local system)

$ docker ps -a (To check all containers running or not)

$ docker ps -q -a (To get all container id's on local machine)

$ docker top containerid (To see the top precesses)

$ docker stop containerid (To stop the Container)

$ docker rm containerid (To remove the container)

$ docker rm -f containerid (For forceful remove of container)

$ docker stats containerid (To offer the statistics of a running container)

$ docker attach containerid (In order to attach to a running container)

$ docker pause containerId (To pause the processes in a running container)

$ docker unpause containerId (To pause the processes in a running container)

$ docker kill containerId (To kill the processes in a running container)

$ docker logs containerid (To see logs of the container)

$ docker logs –tail 5 dockerid (To see the last 5 logs)

What is Docker Registry?

Finally, Docker Registry is where the Docker Images are stored. The Registry can be either a user's local repository or a public repository like a Docker Hub allowing multiple users to collaborate in building an application. Even with multiple teams within the same organization can exchange or share containers by uploading them to the Docker Hub. Docker Hub is Docker's very own cloud repository similar to GitHub.

Commands:

$ docker login containerregistryurl -u username -p password //Login to registry

$ docker push containerregistry.dockerhub.io/ projectname/microservicename:v1.0.0 //push image to registry)

$ docker pull containerregistry.dockerhub.io/ projectname/microservicename:v1.0.0 //pull image from registry

How to install docker in Linux:

Step-1:

$ uname -r (Gives the information about operating system and kernel version) above 3.10

Go to browser and check for Docker Manuals, To find the link:

https://docs.docker.com/v17.09/manuals/

Click on above link to find information of how to install Docker in any operating system

Click on linux, We will get info about that

Step-2:

First we need to update all the packages

$ sudo yum -y update

Next Run the command to install Docker

$ sudo yum install -y docker

$ docker (To see the commands of Docker)

$ docker - -version (To check the version of docker)

$ docker info (It gives the information of docker running or not)

Step-3: Start Docker

$ sudo service docker start

$ sudo service docker status

$ docker info (Get the all information about docker running on your system)

Add user to docker service group

$ sudo usermod -a -G docker username

Step-4: Stop Docker

$ sudo service docker stop

Step-5: Uninstall Docker

$ sudo yum remove docker -y

For Help:

$ docker (To see docker install or not)

$ docker help (To see the help page)

Build & Tagging:

$ docker build -t projectname/microservicename:v1.0.0 (To Build new image)

$ docker tag imagename taggingname/imagename //Tagging with container registry

(eg: docker tag projectname/microservicename:v1.0.0 containerregistry.dockerhub.io/ projectname/microservicename:v1.0.0)

$ docker login containerregistryurl -u username -p password //Login to registry

$ docker push containerregistry.dockerhub.io/ projectname/microservicename:v1.0.0 //push image to registry)

$ docker pull containerregistry.dockerhub.io/ projectname/microservicename:v1.0.0 //pull image from registry