# DOCKER

## DOCKER NOTES:

1.
2. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.
3. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.
4. Group of VMs called as a **CLUSTER**.
5. hub.docker.com is official public repository for Docker.
6. Docker containers are very lightweight.
7. If we delete the container data will be gone ,it will not store in Image
8. On single host we can run multiple containers by using changing host port ip.
9. Docker Latest Version **17.09.1-ce(17x)** where 17 is Year 09 is Month.

## Dockerfile vs Docker Image vs Docker Container:

1. A Dockerfile is a recipe for creating Docker images
2. A Docker image gets built by running a Docker command (which uses that Dockerfile)
3. A Docker container is a running instance of a Docker image.

## DOCKER ORCHESTRATION TOOLS:

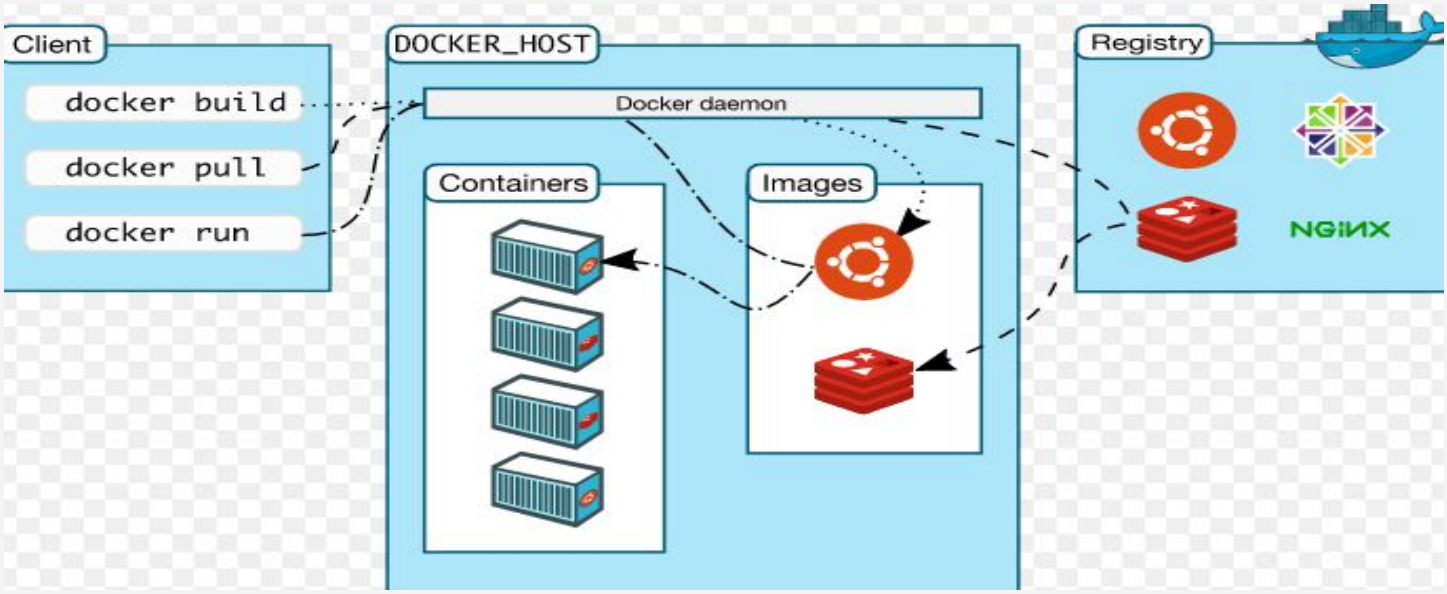
1. Docker Swarm
2. Kubernetes(Very High Demand in Market)
3. Apache Mesos
4. OpenShift → Kubernetes

## Docker Architecture:

Docker follows client-server architecture. Its architecture consists mainly three parts.

1. **Client:** Docker provides Command Line Interface (CLI) tools to client to interact with Docker daemon. Client can build, run and stop application. Client can also interact to Docker_Host remotely.
2. **Docker_Host:** It contains Containers, Images, and Docker daemon. It provides complete environment to execute and run your application.
3. **Registry:** It is global repository of images. You can access and use these images to run your application in Docker environment.

# DOCKER




==========

**Installing and Configuring DOCKER on Amazon Linux Machine:**

1. To run Docker on any Machine, the Machine must install with Docker
2. sudo yum install docker -y
3. sudo service docker start
4. sudo chkconfig docker on
5. docker --version

**Note:**

1. By default Docker Pulls Images from Docker-Hub,however we can change this default location
2. Docker Client and Docker Host both can be same machine or different machine.

==========

**DOCKER COMMANDS:**

1. sudo docker images → **Shows all Available Docker Images**
2. sudo docker ps → **Shows all Running Containers**
3. sudo docker ps -a → **Shows all Containers**
4. sudo docker ps -aq → **Shows all Container Id's**
5. docker run -d -p 8888:8080 tomcat:8.0
   **d = Detached mode or Demon Mode, p = Port Mapping**
6. sudo docker start container id → **To Start Docker Container**

**Address:** *MP Techno Solutions*, Above Titan ShowRoom, Marathahalli, Bangalore, Phone: 9606955170/1/2/3

# DOCKER

7. sudo docker stop container id → **To Stop Docker Container**
8. sudo docker rm container id → **To Remove Non-Running Docker Container**
   **Note:**Before removing the container, it must be in stopped state.
9. sudo docker rm -f container id → **To Remove Running Docker Container Forcefully**
10. sudo docker rmi image id → **To Remove Docker Image**
    **Note:** Docker does not delete Image that used by any container
11. sudo docker rmi -f image id → **To Remove Docker Image Forcefully**
12. sudo docker run -d -p 2017:8080 ImageId → **To Run Docker Container which Image Present Locally**
    **Ex: 2017 = Host Port, 8080 = Container Port**
13. sudo docker exec -it container id /bin/bash → **To Enter into Container**
    **exec = Execute, it = Interactive Terminal**
14. sudo docker build -t sureshkanna/myapp:0.1 . → **To Build Image from Dockerfile**
    **-t = Tagging, myapp= Project Name, 0.1 = Version, .= Dockerfile Location, sureshkanna = Docker Hub Username Example**
15. docker login → **To Login to Docker Hub Account**
16. sudo docker push sureshkanna/myapp:0.1 → **To Push Docker Image to DockerHub**
17. The stop command gracefully shuts down the process within the container and the kill command just stops it dead.
18. sudo docker rm $(docker ps -q -f status=exited) → **To remove all stopped containers**
    **Where f = filter**
19. **Ctrl+g** will use to go end page in vi editor
20. ssh-keygen -t rsa → **To generate public key and private key for password less authentication.**
21. sudo usermod -a -G docker ec2-user → **To give permissions to ec2-user to run docker commands directly**
22. sudo usermod -a -G docker jenkins → **To give permissions to jenkins to run docker commands directly.**

**Note:** After run above two commands we must reboot the server, then only it come to effect..

23. sudo reboot → **To Reboot the server**
24. docker images -f dangling=true **→List out dangling images**
25. sudo docker rmi $(docker images -f dangling=true -aq) → **To Remove dangling images.**

# DOCKER

==========

**Difference Between docker stop and docker kill:**

**docker stop:** Stop a running container (send **SIGTERM**, and then **SIGKILL** after grace period). The main process inside the container will receive **SIGTERM**, and after a grace period, **SIGKILL**.
**docker kill:** Kill a running container (send **SIGKILL**, or specified signal). The main process inside the container will be sent SIGKILL, or any signal specified with option --signal.
So stop attempts to trigger a graceful shutdown by sending the standard **POSIX** signal **SIGTERM**, whereas killjust kills the process by default (but also allows to send any other signal)

==========

**AUTOMATION USING JENKINS:**

**Sample code:**

**In Top Level Maven Targets:**

clean package

**In Execute Shell:**

```
docker rmi -f $(docker images -aq)
#docker rm -f $(docker ps -aq)
docker build -t sureshkanna/myapp:0.14 .
docker login -u docker-hub-username -p docker-hub-password
docker push sureshkanna/myapp:0.14

sudo ssh ec2-user@172.31.25.194 docker rm -f kanna
sudo ssh ec2-user@172.31.25.194 docker run -d -p 2020:8080 --name kanna
sureshkanna/myapp:0.14
```

==========

**Jenkins:**

1. vim /etc/sysconfig/jenkins → Change JENKINS User to ”ec2-user”
2. cd /var/lib/jenkins
3. chown -R ec2-user:ec2-user jenkins/
4. Cd /var/log

5. chown -R ec2-user:ec2-user jenkins/
6. cd /var/cache/
7. chown -R ec2-user:ec2-user jenkins/

==========

## Docker-maven-plugin:spotify:

---

**Commands**:

1. mvn clean package docker:build → **You can build an image with the above configurations by running this command.**
2. mvn clean package docker:build -DpushImage → **To push the image you just built to the registry, specify the pushImage flag.**
3. mvn clean package docker:build -DpushImageTag → **To push only specific tags of the image to the registry, specify the pushImageTag flag.**
4. -DskipDockerBuild **to skip image build**
5. -DskipDockerTag **to skip image tag**
6. -DskipDockerPush **to skip image push**
7. -DskipDocker **to skip any Docker goals**
8. mvn docker:removeImage -DimageName=foobar → **To remove the image named foobar**

## Authentication:

Authentication credentials can be explicitly configured in your pom.xml and in your Maven installation **settings.xml** file as part of the <servers></servers> block.

**<servers>**
**<server>**
**<id>docker-hub</id>**

```
    <username>docker-hub-username</username>
    <password>Docker-hub-password</password>
    <configuration>
      <email>example@gmail.com</email>
    </configuration>
  </server>
</servers>
```

==========