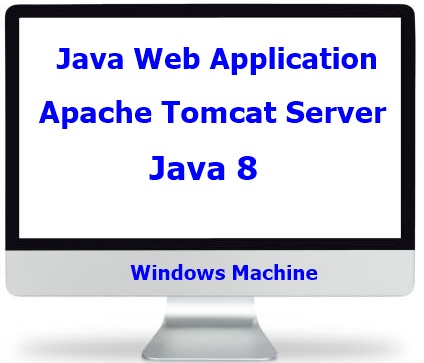


**What is docker**?

Docker is a tool which is used to automate the deployment of applications in lightweight containers so that applications can work efficiently in different environments. It is an open-source platform that is available for different OS like Windows, MAC, Linux. Docker enables us to separate our applications from our infrastructure so we can deliver software quickly. With the help of Docker, we can manage our infrastructure in the same ways we manage our applications and provide a consistent environment for application deployment. It was first started in 2013 and is developed by Docker, Inc an American Technology Company.

**Need of Docker**

**Scenario 1 Host same web applications on different machines or systems.**



**Scenario 2** code is working fine on developer machine but not working on tester machine due to difference in environments.

****

**Scenario 3 Setup five nodes with LB**

Traffic coming from internet

LB

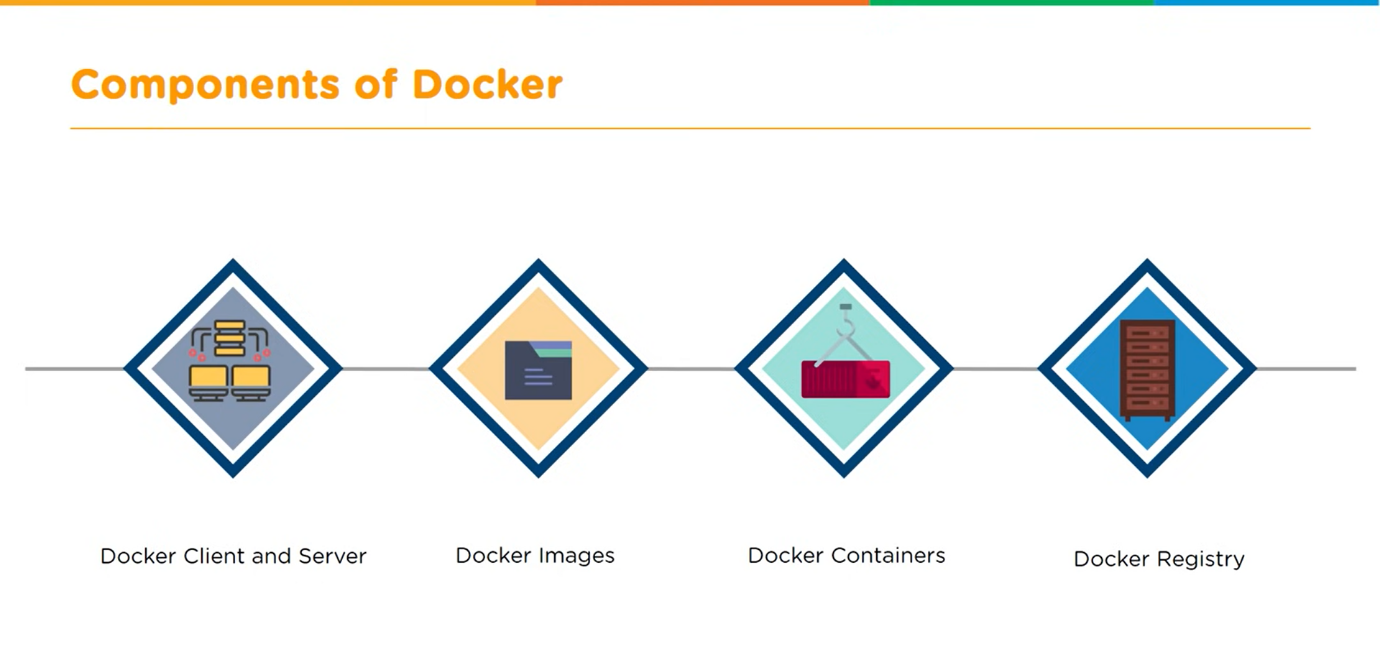
Node 5

Node 4

Node 2

Node 1

Node 3



Different components of docker

1. Docker Client and Server

2. Docker Images.

3. Docker Containers.

4. Docker Registry.

**1. Docker Client and Server**

Docker is the base engine installed on our machine to create containers. Docker uses a client server architecture. When we installed docker basically two components are installed.

1. Docker client which is a CLI tool to run docker commands.

2. Docker server. Docker server take request from docker client and interact with OS to create and manage Containers.

Docker client and server interact with each other using REST APIs.

**2. Docker Image**

Docker image is a template with instructions which is used for creating docker containers.

Docker image can be build using docker file or running containers can also be saved as image.

It provides a convenient way to package up applications and preconfigured server environments, which we can use for our own private use or share publicly with other Docker users.

**3. Docker Containers**

Container is a software package that consists of all the dependencies required to run the application.

Multiple containers can run on the same hardware.

Containers are maintained in an isolated environment.

Quick and easy to setup.

**4. Docker Registry**

Docker Registry is an open-source server-side service used for hosting and distributing images.

Docker also has its own default registry called Docker Hub.

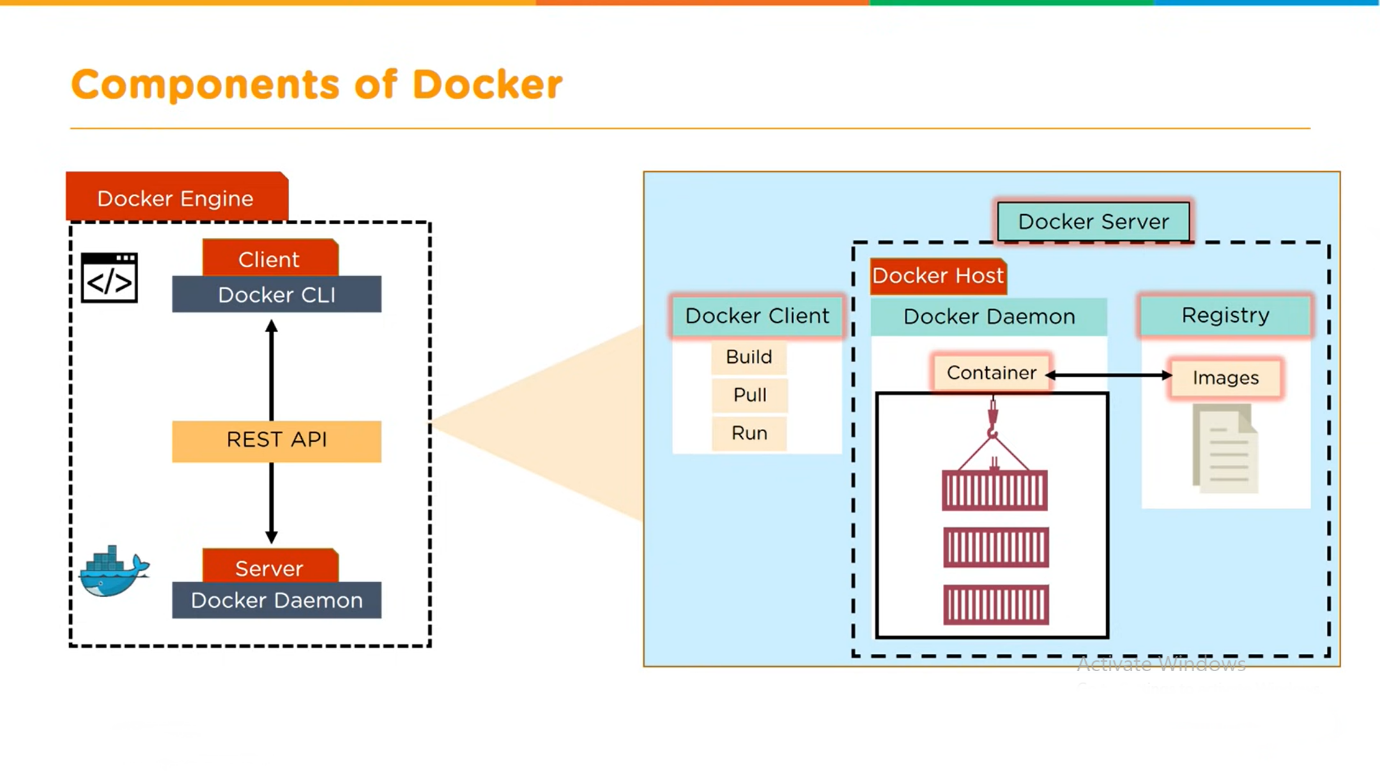
Images can be stored in either public and private repositories.

Pull and push are the commands used to interact with a Docker Registry.

**For docker install please visit below link.**

**https://docs.docker.com/desktop/install/windows-install/**

**How Docker Works.**

****

**Docker Basic Commands**

**To check docker version installed.**

1. docker -v or docker --version

**To Pull an image from docker Hub**

2. docker pull imagename e.g docker pull ubuntu

**To see the list of images.**

3. docker images

**To Remove an Image**

4. docker rmi imagename/imageid e.g docker rim ubuntu

**To create an ubuntu container in detach mode.**

5. docker run -id ubuntu

**To get the list of all Container**

6.docker ps -a

**To create an ubuntu container in interactive mode**

7. docker run -it ubuntu

**To create an ubuntu container in detach mode with name**

8. docker run -dt --name myfirstcontainer ubuntu

**To rename a docker container**

9. Docker rename old-name new-name to rename a container.

**To start a docker container**

10. docker start containerid or name

**To stop a docker container**

11. docker stop containerid or name

**To restart a docker container**

12. docker restart containerid or name

**To remove a container**

13. docker rm containerid or name e.g docker rm myfirstcontainer

**To execute a command inside docker container**

14 docker exec -it container name/id command to execute

e.g docker exec -it mycontainer apt-get install vim

**To save a container as image**

15. docker commit contained or name e.g docker commit myfirstcontainer

**To tag an image**

16. docker tag imageid Reponame:tagname e.g docker tag 6ad0f48e8b54 ubuntu\_java8\_tomcat-9:latest

**To expose a container port and map it with host port.**

17. docker run -id -p system-port: container-port image name e.g docker run -id -p 8080:8080 ubuntu\_java8\_tomcat-9:latest

-p flag is used to expose container port and map it with host machine.

**To copy file from host to container and vice versa**

18. docker cp filename with path on host machine containerid: filename with path

Example: docker cp ok.txt myfirstcontainer:/ok.txt

**To login a docker account**

19. docker login

**To save an image to docker hub**

20. docker push javadev92786/ubuntu\_java8\_tomcat8

**To logout from the account**

21. docker logout

**To save an image as tar file**

22. docker save -o file.tar image-reponame: tag-name

**To load an image from tar file.**

23. docker load -i file.tar

**Build a Docker Image with Docker File.**

Docker File

A Docker file is a text document that contains all the commands a user can use on the command line to assemble an image. A docker file do not have any extension and saved with name “Dockerfile” .

Syntax to build an image from docker file.

Docker build -t tag-name directory path

Example

Different Instructions USED in Dockerfile

1. **FROM**: This instruction is used to pull an image from Docker hub to use as a base image to customize.

Syntax FROM image-name: tag-name

Example FROM ubuntu: latest

1. **COPY**: This is used to copy file from host machine to image. It takes two arguments. First is source file name from host machine and second is destination path inside the image.

Syntax COPY source file destination path

COPY app.war /usr/local/tomcat/webapps/

1. **ADD**: This command is similar to COPY but have some addition features like it can copy file from URL, extract a compressed file to destination if we try to copy a compressed file. It also takes 2 arguments.

Syntax ADD source file destination path

Example ADD a.txt /tmp/

1. **LABEL**: Label command is used to provide some meta information about the image. It takes arguments in the form of key=”value” pair. We can specify multiple labels in single label command separated by space.

Syntax LABEL key=”value” key=”value”

Example: LABEL java.version=”1.8” tomcat.version=”9.0”

1. **RUN**: A RUN instruction is used to run specified commands. We can use multiple RUN instructions to run different commands inside a docker file.

Syntax RUN command to run

Example: RUN apt-get -y install vim

RUN apt-get -y install wget

1. **CMD**: If we want to run a default command every time a container is created from an image or started then we can specify that command in the CMD instruction.

Syntax: CMD command to run

Example: CMD echo "Welcome to docker"

1. **WORKDIR**: we can specify our working directory inside the container using the WORKDIR command.

Syntax WORKDIR directory path

Example: WORKDIR /usr/src/app

1. **USER**: to set user that runs the commands in docker container.

Syntax: USER username

Example: USER tomcat

1. **ENV**: to set an Environment variable we can use ENV command.

Syntax: ENV variable name=value.

Example: ENV JAVA\_HOME=””

1. **EXPOSE:** To expose a container port on which a service is listening or running.

Syntax: EXPOSE port number

Example EXPOSE 8080

1. **ENTRYPOINT:** toconfigure a default command or script that will run whenever a container is created from the image. It is similar to CMD command but we can’t override the command given in ENTRYPOINT.

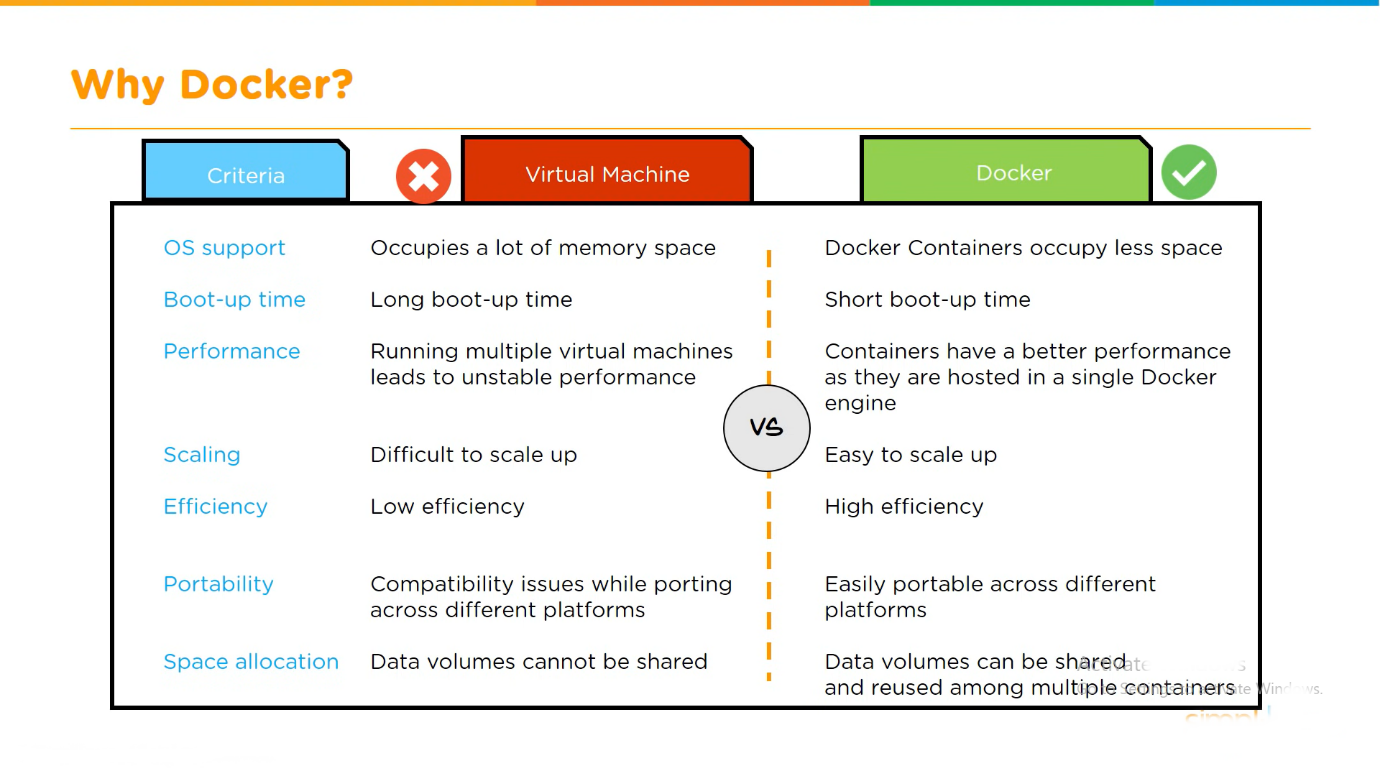
Syntax ENTRYPOINT command to run

Example: /usr/local/tomcat/bin/catalina.sh run

**Advantages of Docker.**

1. Time Saving
2. Less Hardware cost
3. Isolation
4. Portable.
5. Common platform for Development and Testing.
6. Development playground for Developers.
7. Easy Sharing
8. Compatible with Micro Service Architecture.

**Docker vs VM**

****