What is Docker?

Docker is an **open-source centralized platform designed** to create, deploy, and run applications.

Docker uses **container** on the host's operating system to run applications.

It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system

Containers ensure that our application works in any environment like development, test, or production.

Docker includes components such as **Docker client, Docker server, Docker machine, Docker hub, Docker composes,** etc.

### **Docker Containers**

Docker containers are the **lightweight** alternatives of the virtual machine. It allows developers to package up the application with all its libraries and dependencies, and ship it as a single package

The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications. It automatically generates storage and space according to the application requirement.

### **Virtual Machine**

A virtual machine is a software that allows us to install and use other operating systems (Windows, Linux, and Debian) simultaneously on our machine.

|  |  |
| --- | --- |
| **Containers** | **Virtual Machine** |
| Integration in a container is faster and cheap. | Integration in virtual is slow and costly. |
| No wastage of memory. | Wastage of memory. |
| It uses the same kernel, but different distribution. | It uses multiple independent operating systems. |

## Advantages of Docker

* It runs the container in seconds instead of minutes.
* It uses less memory.
* It provides lightweight virtualization.
* It does not a require full operating system to run applications.
* It uses application dependencies to reduce the risk.
* Docker allows you to use a remote repository to share your container with others.
* It provides continuous deployment and testing environment.

## Disadvantages of Docker

There are the following disadvantages of Docker -

* It increases complexity due to an additional layer.
* In Docker, it is difficult to manage large amount of containers.
* Some features such as container self -registration, containers self-inspects, copying files form host to the container, and more are missing in the Docker.
* Docker is not a good solution for applications that require rich graphical interface.
* Docker provides cross-platform compatibility means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

### **What is Docker daemon?**

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage. S

### **Docker architecture**

Docker follows Client-Server architecture, which includes the three main components that are **Docker Client**, **Docker Host**, and **Docker Registry**.

### **1. Docker Client**

Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

Docker Client uses Command Line Interface (CLI) to run the following commands -

docker pull

docker build

docker run

### **2. Docker Host**

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

### **3. Docker Registry**

Docker Registry manages and stores the Docker images.

**Pubic Registry -** Public Registry is also called as **Docker hub**.

**Private Registry -** It is used to share images within the enterprise

## Docker Objects

### **Docker Images**

Docker images are the **read-only binary templates** used to create Docker Containers

### **Docker Containers**

Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources.

net stop com.docker.service

net start com.docker.service

docker run -d -p 80:80 docker/getting-started

docker run -p 8082:8082 helloworld

netstat -ano | findstr :8080

taskkill /PID <PID> /F

**to push docker image**

docker tag sharukh6278/demo:javahello

docker push sharukh6278/demo:javahello

demo

docker pull hello-world ->pull images from repository

docker pull ubuntu:18.04 ->to pull ubanto image with tag or version

docker run hello-world -> check in localy else pull it form central repo

docker images -> to see all images in docker

docker rmi imageID -> to image form docker

docker rmi -f imageID ->to remove image from docker forcefully

docker images -q -> to have only ImageID

docker ps -a -> to see all container running

docker history ImageId ->show history when image uploaded latest

docker stop containerID ->to stop particular container

docker rm containerID ->to remove the particular container

docker start container id ->to start the container

docker kill ContainerId ->Kill the process running under th container

To upload local source code

FROM openjdk

COPY ./HelloWorld.jar /deployments/

EXPOSE 8083

CMD java -jar /deployments/HelloWorld.jar

To Just run simple java

FROM openjdk

COPY ./HelloWorld.java ./

EXPOSE 8083

RUN javac HelloWorld.java

CMD ["java", "HelloWorld"]

To build image

docker build -t myjenkins2:1.0 .

Ubuntu

Docker pull ubuntu

Docker run –name MyUbuntu23 -it ubuntu bash

Setting uP Jenkins in Docker

Docker pull Jenkins

docker run -p 8080:8080 -p 50000:50000 -v "W:\Training\Docker\tutorial\Jenkins docker\jenkins\_home" jenkins:2.60.3