**What is Docker ?**

Docker is the world’s leading software container platform

Docker makes the process of application deployment very easy and efficient and resolves a lot of issues related to deploying applications.

Docker is a tool designed to make it easier to deploy and run applications by using containers.

Docker gives you a standard way of packaging your application with all its dependencies in a container.

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.

------------------------------------------------------------

**Important factors:**

Docker file

Docker images

Docker Containers

Docker Hub / Registry

Docker client

Docker server / daemon

Docker engine

Stydied from : <https://automationstepbystep.com/> 🡪 free course 🡪 Dockers

**install Dockers on Windows :**

Prerequisites

OS - 64 bit

Windows 7 or higher

STEP 1 : Install Docker

https://docs.docker.com/toolbox/toolb...

https://docs.docker.com/docker-for-wi...

-------------------------------------------------------------

**Basic:**

: docker version

: docker -v

: docker info

: docker --help

Eg: dockers images --help

dockers run --help

: docker login

————————————

Images

: docker images

Eg: docker images -a

(shows all the image)

docker images -q

(shows only id of image)

: docker pull

Eg: docker pull ubuntu

: docker rmi

Eg: docker rmi 74435f89ab78

(the image will deleted)

————————————

Containers

: docker ps

: docker run

Eg: docker run -it ubuntu

(go inside the container)

: docker start

Eg: docker start a7a619b013b8

: docker stop

Eg: docker stop a7a619b013b8

————————————

System

: docker stats

: docker system df

: docker system prune

(container deleted)

Eg: docker system prune -a

(image and container deleted)

**What are Images:**

Docker Images are templates used to create Docker containers

Container is a running instance of image

Where are Images Stored

Registries (e.g. docker hub)

Can be stored locally or remote

: docker images --help

: docker pull image

: docker images

: docker images -f “dangling=false”

: docker images -f “dangling=false” -q

: docker run image

Eg: docker run --name myubuntu1 -it ubuntu bash

(myubuntu1 is continer name)

: docker rmi image

: docker rmi -f image

: docker inspect

: docker history imageName

**What are Containers:**

Containers are running instances of Docker Images

COMMANDS

: docker ps

: docker run ImageName

Eg: docker run --name myubuntu1 -it ubuntu

: docker start ContainerName/ID

Eg: docker start myubuntu1

: docker stop ContainerName/ID

: docker pause ContainerName/ID

: docker unpause ContainerName/ID

: docker top ContainerName/ID

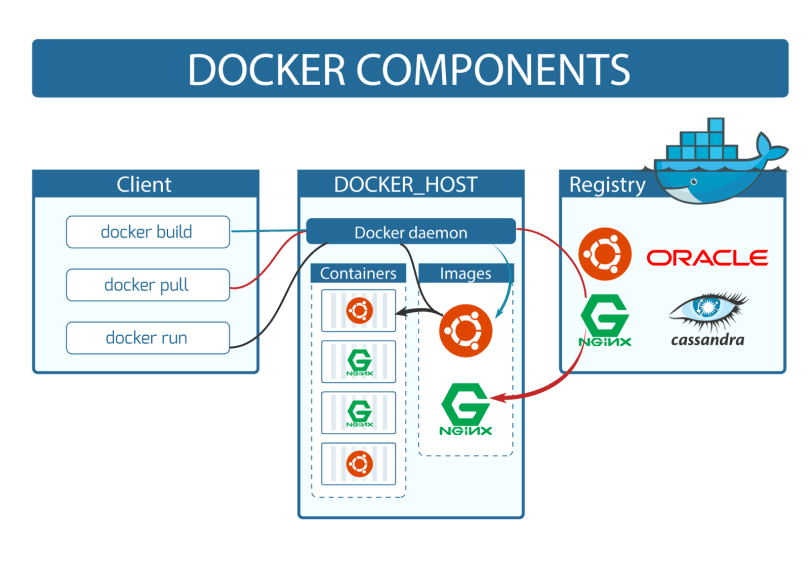
: docker stats ContainerName/ID

: docker attach ContainerName/ID

: docker kill ContainerName/ID

: docker rm ContainerName/ID

: docker history ImageName/ID

-

**How to run Jenkins on Docker container | How to create Jenkins Volumes on Docker**

1. How to start Jenkins on Docker Container

2. Start and Stop Jenkins Container

3. How to set Jenkins home on Docker Volume and Host Machine

: docker pull jenkins

: docker run -p 8080:8080 -p 50000:50000 jenkins

: docker run --name MyJenkins -p 8080:8080 -p 50000:50000 -v /Users/Deepu/Desktop/Jenkins\_Home:/var/jenkins\_home jenkins

( error : touch: cannot touch '/var/jenkins\_home/copy\_reference\_file.log': Permission denied

Can not write to /var/jenkins\_home/copy\_reference\_file.log. Wrong volume permissions?)

(it will create a folder in desktop) (copy the admin password 🡪 search localhost:8080 🡪start use of Jenkins 🡪 Jenkins 🡪New Item 🡪 job\_test1 ) (remove a container)

(create a new container as shown below)

: docker run --name MyJenkins2 -p 9090:8080 -p 50000:50000 -v /Users/raghav/Desktop/Jenkins\_Home:/var/jenkins\_home jenkins

(search localhost:9090 🡪 took us to login page 🡪 the job created earlier in available here(job\_test1))

: docker volume create myjenkins

: docker volume ls

: docker volume inspect myjenkins

: docker run --name MyJenkins3 -p 9090:8080 -p 50000:50000 -v myjenkins:/var/jenkins\_home jenkins

: docker inspect MyJenkins3 In case you face issues like installing plugins on this Jenkins, can setup jenkins with this command:

$ docker run -u root --rm -p 8080:8080 -v /srv/jenkins-data:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock --name jenkins jenkinsci/blueocean

**Dockerfile**

FROM RUN CMD

Step 1 : Create a file named Dockerfile

C:\Users\Deepu\Desktop>mkdir DockerFiles

C:\Users\Deepu\Desktop>cd DockerFiles

Step 2 : Add instructions in Dockerfile

Eg: FROM ubuntu

MAINTAINER deepak <deepaksri728@gmail.com>

RUN apt-get update

CMD ["echo", "Hello World...! from my first docker image"]

Step 3 : Build dockerfile to create image

Step 4 : Run image to create container

COMMANDS

: docker build

: docker build -t ImageName :Tag directoryOfDocekrfile

Eg: docker build -t myimage1:1.0 .

: docker run imageid



**Docker compose:**

tool for defining & running multi-container docker applications

: use yaml files to configure application services (docker-compose.yml)

: can start all services with a single command

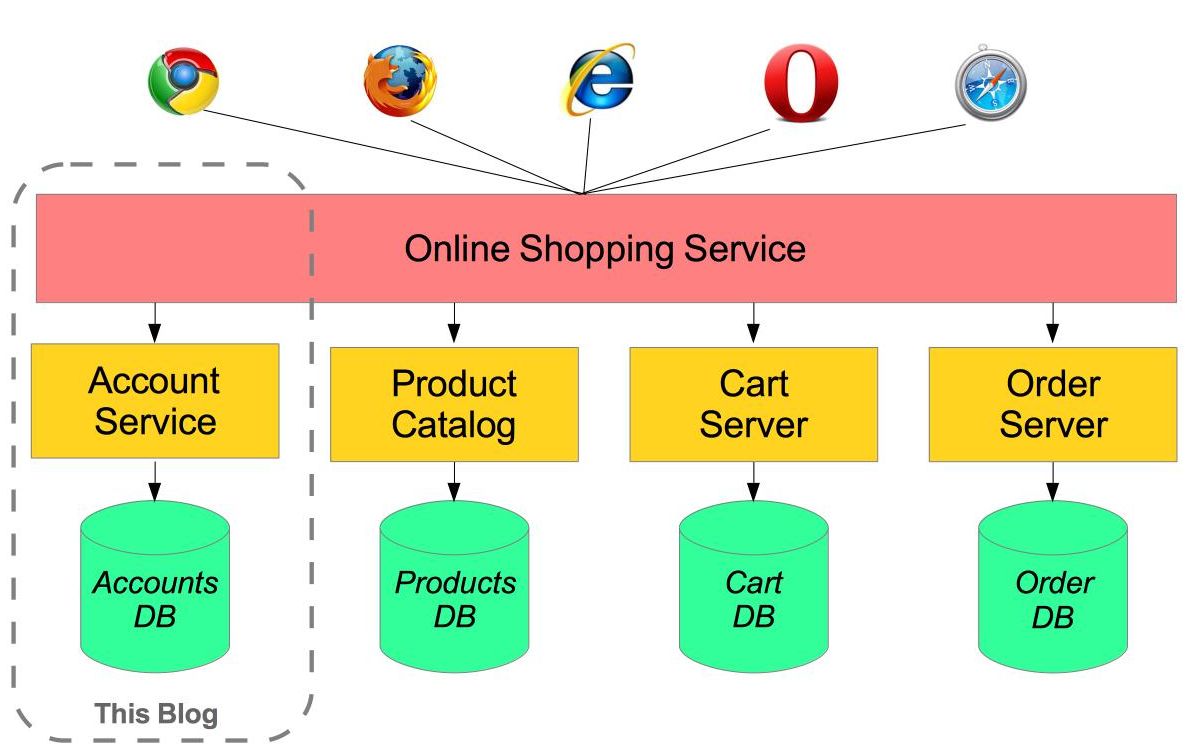
: docker compose up

: can stop all services with a single command

: docker compose down

: can scale up selected services when required

Eg:



Step 1 : install docker compose

(already installed on windows and mac with docker)

docker-compose -v

2 Ways

1. https://github.com/docker/compose/rel...

2. Using PIP

pip install -U docker-compose

Step 2 : Create docker compose file at any location on your system

docker-compose.yml

Step 3 : Check the validity of file by command

docker-compose config

Eg: services:

database:

image: radis

web:

image: nginx

version: '3.1'

Step 4 : Run docker-compose.yml file by command

docker-compose up -d

Steps 5 : Bring down application by command

docker-compose down

C:\Users\Deepu\Desktop\DockerComposeFile>docker-compose up -d --scale database=4

Eg: CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f3a2eab1578c nginx "/docker-entrypoint.…" 15 seconds ago Up 11 seconds 0.0.0.0:8080->80/tcp dockercomposefile\_web\_1

dec759a308e1 redis "docker-entrypoint.s…" 15 seconds ago Up 11 seconds 6379/tcp dockercomposefile\_database\_4

834770df1c4a redis "docker-entrypoint.s…" 15 seconds ago Up 11 seconds 6379/tcp dockercomposefile\_database\_1

b2cc413b1e7f redis "docker-entrypoint.s…" 15 seconds ago Up 11 seconds 6379/tcp dockercomposefile\_database\_2

5fa27b9e8436 redis "docker-entrypoint.s…" 15 seconds ago Up 11 seconds 6379/tcp dockercomposefile\_database\_3

**Volumes are the preferred mechanism for persisting data generated by and used by Docker containers**

: docker volume //get information

: docker volume create

Eg: docker volume create myvol1

: docker volume ls

: docker volume inspect

Eg: docker volume inspect myvol1

: docker volume rm

: docker volume prune

Instead of deleting containers one by one of docker ps -a , we can use docker container prune. and for docker ps (running containers) we can use docker rm $(ps -aq)

**Use of Volumes**

===========

Decoupling container from storage

Share volume (storage/data) among different containers

Attach volume to container

On deleting container volume does not delete

Commands

docker run --name MyJenkins1 -v myvol1:/var/jenkins\_home -p 8080:8080 -p 50000:50000 jenkins

docker run --name MyJenkins2 -v myvol1:/var/jenkins\_home -p 9090:8080 -p 60000:50000 jenkins

docker run --name MyJenkins3 -v /Users/raghav/Desktop/Jenkins\_Home:/var/jenkins\_home -p 9191:8080 -p 40000:50000 jenkins

NOTES

By default all files created inside a container are stored on a writable container layer

The data doesn’t persist when that container is no longer running

A container’s writable layer is tightly coupled to the host machine where the container is running. You can’t easily move the data somewhere else.

Docker has two options for containers to store files in the host machine

so that the files are persisted even after the container stops

VOLUMES and BIND MOUNTS

Volumes are stored in a part of the host filesystem which is managed by Docker

Non-Docker processes should not modify this part of the filesystem

Bind mounts may be stored anywhere on the host system

Non-Docker processes on the Docker host or a Docker container can modify them at any time

In Bind Mounts, the file or directory is referenced by its full path on the host machine.

Volumes are the best way to persist data in Docker

volumes are managed by Docker and are isolated from the core functionality of the host machine

A given volume can be mounted into multiple containers simultaneously.

When no running container is using a volume, the volume is still available to Docker and is not removed automatically. You can remove unused volumes using docker volume prune.

When you mount a volume, it may be named or anonymous.

Anonymous volumes are not given an explicit name when they are first mounted into a container

Volumes also support the use of volume drivers, which allow you to store your data on remote hosts or cloud providers, among other possibilities.

Additional reference:

$ docker-machine ip default

192.168.99.100

Next plan

* Docker swarm
* Referring real time applications used dockers