Docker

Library is a default docker repository which has the official images

Eg: library/centos

Why do you need docker ?

Graphical user interface

Description automatically generated

* Compatibilty/Dependency
* Long setup time
* Different Dev/Test/Prod environment

What can it do ?

Graphical user interface, application

Description automatically generated

* Containerize Applications
* Run each service with its own dependencies in separate containers

What are containers?

* Containers are running images and completely isolated environment as in they can have there own processes, Network, Mounts, Services just like virtual machine except they all share the same OS Kernel

A picture containing graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated with low confidence

Docker uses LXE containers.

Operating System:

OS Kernel - is responsible for interacting with underlying hardware while OS kernel remains the same which is Linux

It the Software above it makes this operating system different.

A screenshot of a computer

Description automatically generated with low confidence

Sharing the Kernel

Graphical user interface

Description automatically generated

Containers vs Virtual Machines

Graphical user interface, application

Description automatically generated

Containers & Virtual Machines

Graphical user interface

Description automatically generated

How is it done ?

Graphical user interface

Description automatically generated

Container vs image

Image - is a Package/Template. Just like a VM template. It is used to create one or more container.

Container - are running instances of images that are isolated and have there own environment and set of process

Diagram

Description automatically generated

Docker in Devops

A screenshot of a computer

Description automatically generated with medium confidence

Command:

Docker run - used to run/start a container from an image

Eg : docker run nginx

Docker pull nginx : only pulls the image and will not run the container

Eg: docker pull nginx

Docker ps - list all running containers

Docker ps -a - list all the containers

Graphical user interface

Description automatically generated

If the web service inside container stops then container will also exit

STOP - Stop a container

docker stop (NAME)



Rm - Remove a container (stopped or running conatiner permanently)



Images - List images

docker images



rmi - Remove images. (before removing the images , delete all dependent containers to remove image)

docker rmi nginx

Text

Description automatically generated

Exec - execute a command

To see the content of a file inside the running container

docker exec <container id> cat /etc/hosts

Run - attach and detach

docker run - runs on the foreground (attach mode)

Press ctrl+c to quit

docker run -d - (detach mode)

docker attach (docker id)

docker run -it centos bash

Bash - we specify because to make the process running or else container will exit

-it - to enter into the container

docker run -d centos sleep 20

Run - tag

Default tag: latest

docker run redis:4.0 -> tag

Run - STDIN

Docker runs on the non interactive mode

Run - PORT mapping

Docker run -p 80:5000 kodekloud/simple-webapp

80 - outside port

5000 - docker host port

Run - Volume mapping

On mysql data are stored /var/lib/mysql

docker run -v /opt/datadir:/var/lib/mysql mysql

External volume - /opt/datadir

Internal volume - /var/lib/mysql

To get the Additional details of container. It gives on json format

docker inspect <container id>

To find the ip address of an docker contianer

docker inspect <container id>

Container Logs

docker logs <container id>

How to create my own docker image ?

Dockerfile

FROM Ubuntu

RUN apt-get update & apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK\_APP=/opt/spurce-code/app.py flask run

Execute:

docker build . -f Dockerfile -t mmumshad/my-custom-app

docker push mmumshad/my-custom-app

Execution output: docker executes in a layer

Layer 1. Base Ubuntu Layer

Layer 2: Changes in apt packages

Layer 3: Changes in pip packages

Layer 4: Source code

Layer 5: Update Entrypoint with "flask" command

Container are not meant to host operating system they are meant to perform a specific task or a process

Once the task is complete container exist

Container leaves only as long as process inside it is alive. EG: If the webserver indise the contianer stops then container exists

FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]

Out : docker run ubuntu 10

Command at startup: sleep 10

Docker run --entrypoint sleep2.0 ubuntu 10

Command at start : sleep2.0 10

Docker Compose:

docker run mmumshad/simple-webapp

docker run mongodb

docker run redis:alpine

docker run ansible

docker-compose.yml

services:

web:

image: "mmumshad/simple-webapp"

database:

image: "mongodb"

messaging:

image: "redis:alpine"

orchestration:

image: "ansible"

docker-compose up -> to bring up all aplication stack

Diagram

Description automatically generated

Diagram

Description automatically generated

Graphical user interface

Description automatically generated

-- links -> To link 2 container

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated

Docker compose has to install separately

docker-compose.yml

version: '3.0'

services:

db:

environment:

POSTGRES\_PASSWORD: mysecretpassword

image: postgres

wordpress:

image: wordpress

links:

- db

ports:

- 8085:80

Docker Registry

Image: docker.io/nginx/nginx

| | |

Registry user/ image/

account repository

Private Registry

docker login private-registry.io

docker run private-registry.io/apps/internal-app

Deploy Private Registry

docker run -d -p 5000:5000 --name registry registry:2

docker image tag my-image localhost:5000/my-image

docker push localhost:5000/my-image

docker pull localhost:5000/my-image

or

docker pull 192.168.56.100:5000/my-image

Docker Engine

Docker File system

/var/lib/docker

|- aufs

|- containers

|- image

|- volumes

**Layered architecture**

**Dockerfile**

FROM Ubuntu

RUN agt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK\_APP= /opt/source-code/app.py flask run

docker build Dockerfile -t mmumshad/my-custom-app

|

| layer 1. Base Ubuntu Layer. 120 MB

| layer 2. Changes in apt packages 306 MB

| layer 3. Changes in pip packages 63MB

| layer 4. Source code. 229MB

| layer 5. Update Entrypoint. 0B

FROM Ubuntu

RUN agt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK\_APP= /opt/source-code/app2.py flask run

docker build Dockerfile -t mmumshad/my-custom-app

|

| layer 1. Base Ubuntu Layer. 0 MB

| layer 2. Changes in apt packages 0 MB

| layer 3. Changes in pip packages 0MB

| layer 4. Source code. 229MB

| layer 5. Update Entrypoint. 0B

Volumes : bounds from volume directory

docker volume create data\_volume

| /var/lib/docker

| volumes

| data\_volume

|

/var/lib/docker/volumes

docker run -v data\_volume:/var/lib/mysql mysql

docker run-v data\_volume2:/var/lib/mysql mysql

Bind mounting : bounds directory from any location on the docker host

docker run -v /data/mysql:/var/lib/mysql mysql

docker run \ --mount type=bind,source=/data/mysql,target=/var/lib/mysql mysql

Docker user storage drivers to enable layered architecture

Storage drivers :

1. AUFS. - Ubuntu & Debian
2. ZFS
3. BTRFS
4. Device Mapper
5. Overlay
6. Overlay2

Docker will choose best based on the operating system

Docker stores images, builder, plugins network, swarm, tmp, trust, volumes, containers,

aufs inside /var/lib/docker

docker info

ls -l /var/lib/docker/aufs/diff

docker pull hello-word

application source code will stored inside /var/lib/docker/aufs/diff

docker history. - list the layers

To see the actual spaces consumption on disk - Actual disk usage of docker

docker system df

it shows Images, Containers, Local\_Volumes

docker system df -v - This will split and show

**Networks**

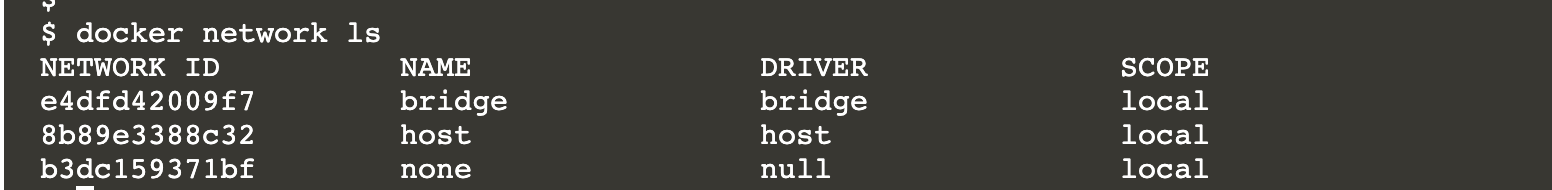
**Default networks**

Bridgenone host

|

docker run ubuntu docker run ubuntu --network=none docker run ubuntu network=host

Usually in the range 172.17.0……



**User-defined networks**

docker network create --driver bridge --subnet 182.18.0.0/16 custom-isolated-network

docker network ls

**Inspect Network**

docker inspect <container id>

Docker for windows

Windows containers

Base images:

1. windows Server Core

2. Nano Server

Support

1. Windows Server 2016

2. Nano Server

3. Windows 10 Professional and Enterprise (Hyper-V Isolated container)

Table

Description automatically generated

Table

Description automatically generated with medium confidence

**Container Orchestration**

Why Orchestrate? Helps us to scale up/down multiple instance in a single command

Container orchestration : helps us to deploy any number of application in single command

also helps us atomactically add additonal host

also provide support for advance networking between containers across different host.

also load balancing user request across different host

|  |
| --- |
| docker service create --replicas=100 nodejs |

Orchestration tool:

1. Docker swarm - from docker
2. Kubernetes - Google. (most popular, and support for all different cloud providers)
3. Mesos - from apache

**Docker Swarm**

With docker swarm we can combine multiple docker machine together into single cluster.

Setup Swarm

There should be a Multiple docker host.

1. Swarm Manager
2. Node/Worker
3. Node/Worker
4. Node/Worker

To make docker host as swarm manager run the below command

docker swarm init

For worker :

docker swarm join --token <token>

Docker Service : is a one or more instances of single application/instance across the nodes on swarm cluster

docker service create --replicas=3 my-web-server

Imp: docker service command should run on manager node and not on worker node

Kubernetes:

Kubernetes uses docker host to host application in the form of docker containers

It also support other container like (kiro, rocket)

Kubectl - is kubernetes CLI

kubectl run <container name> (to deploy an application)

eg: kubectl run my-web-app --image=my-web-app --replicas=1000

kubectl cluster-info

kubectl get nodes