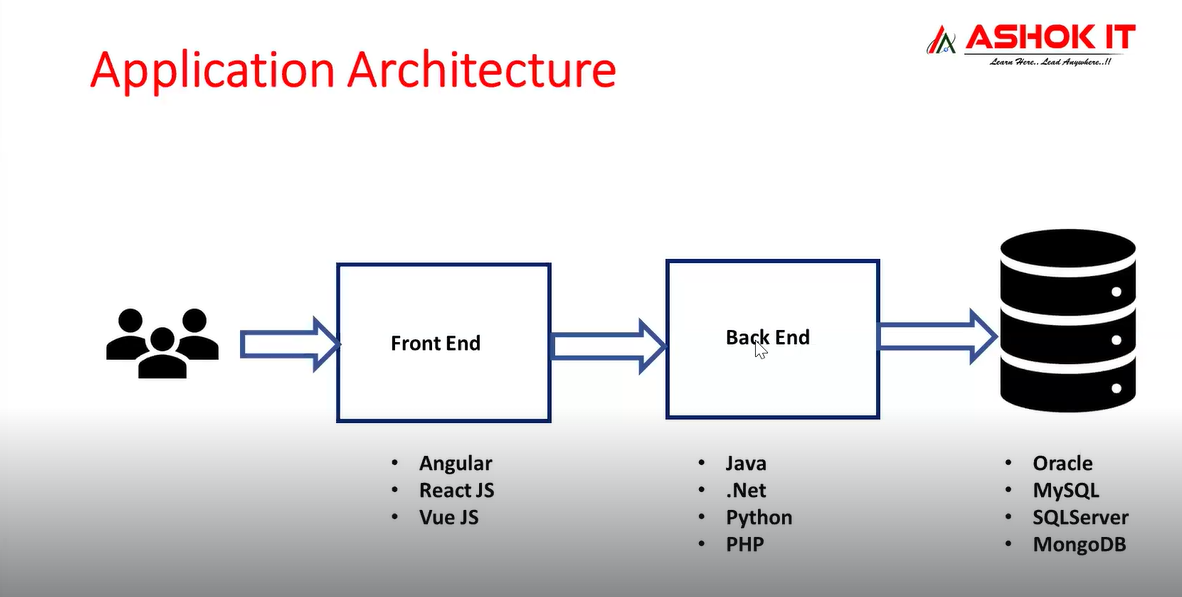
**01-28-Docker-11-Nov-24**

=====================================================================

**Application Architecture**

=====================================================================



1) Frontend : User Interface (UI)

2) Backend : Business logic

3) Database : Storage

=====================================================================

Tech Stack of Application

=====================================================================

Frontend : Angular 16v

Backend : Java 17v

Database : MySQL DB Server 8.5

Webserver : Tomcat 9.0

Project-1: Angular + Java + Oracle DB

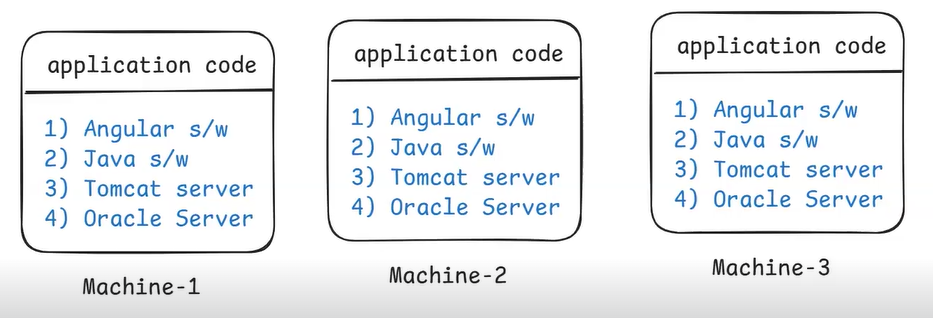
Project-2: React JS + Java + MySQL DB

Project-3: React JS + Node JS + Mongo DB

Project-4: Angular + .Net + SQL Server

Project-4: React JS + Python + MySQL DB

**Dependencies**



Note: If we want to run our application code, then we need to setup all required dependencies in the machine.

Note: dependencies nothing but the softwares which are required to run our application.

Ex: java 17 + Angular 16 + MySQL 8.5 + Tomcat server 9.0

Note: If we want to run same application in 100 machines then it is hectic task to setup dependencies and there is a chance of human mistakes.

=> To overcome above problem we will use Docker tool.

If we want to run application in any machine then we have to install all the required software’s with proper version compatibility.

Ex: Angular 17v, Java 11v, Tomcat 9.0v, MySQL 8.5v

Note: If we do any mistake in s/w installations then application can't execute.

=====================================================================

**What is Docker ?**

=====================================================================

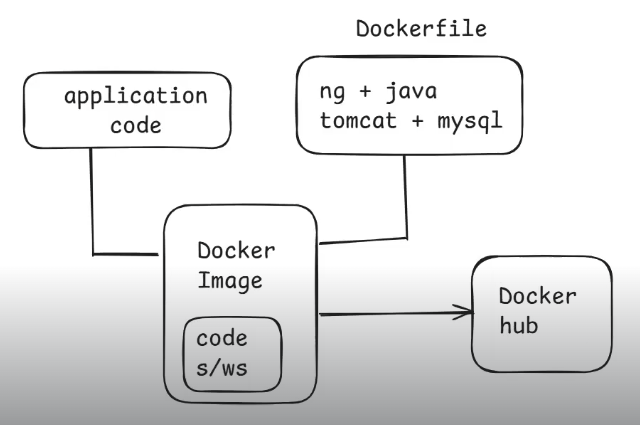
* Docker is a free & open source software.
* Docker is used for containerization.

Note: Containerization means packaging application code + application dependencies as single unit for execution.

* With the help of docker, we can run our application in any machine.
* Docker will take care of dependencies installation required for application execution.
* We can make our application portable using Docker.

Note: Docker is platform independent. We can use docker in windows, linux and mac also.

Docker Container = application code + application dependencies



Hey docker , in order to run our application we need angular,java,mysql installation what docker will do is it will take care of s/w installation and application code and it will create a docker image.

Once docker image is created that docker image we are going to store into one docker registry known as Docker Hub.

Docker Hub used to store docker images.

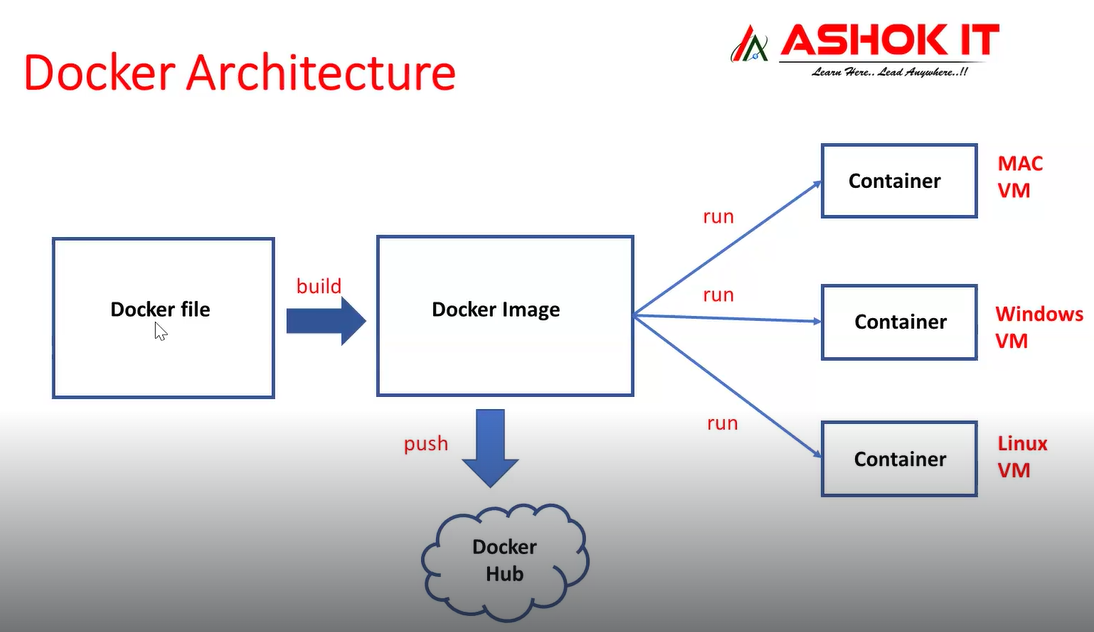
Once docker image is stored in docker hub we can download that image from docker hub and we can run that docker image.

After running docker image docker container will be created.

**Advantage Of Docker**

* Application can be run on any machine without bothering about which software is available on which machine we no need to check that.
* We can run application on any machine if the docker file is available in that machine.

=====================================================================



=====================================================================

Docker Architecture

=====================================================================

1) Docker file

2) Docker Image

3) Docker Registry /nexus /jfrog / AWS ECR

4) Docker Container

**1) Docker file**

* Docker file Contains Set of instructions to build docker image.
* Dockerfile is used to specify where is app code and what dependencies are required for our application execution.

Note: Using dockerfile we will build docker image.

* Developer and Devops engineer both are write docker file.
* Docker file is a text file , will specify what softwares required for application
* Docker image will download that softwares and keep that.

**2) Docker Image**

* Docker image is a package which contains app code and app dependencies.
* Docker Registry is used to store docker images.
* Docker images are public and private , it depends on us.
* We cant see the docker image .
* Docker image is called as artifact that is store in nexus / Jfrog.

=====================================================================

**Docker Container**

=====================================================================

* Note: When we run docker image then docker container will be created.
* Every Docker container is a linux virtual machine.
* Every different container should be mapped with different port number.
* Docker Conatiner is a Linux machine.
* We can run docker image in any machine , we no need to install softwares.

Download docker s/w 🡪 Download docker image 🡪 run docker image in any machine 🡪

Application will run in any machine. This process is called containerization.

We are packaging our code and dependencies for easy execution.

* Inside Docker Container our application will be executed.
* Inside the docker container our application wil be executed.

=====================================================================

**Install Docker in Linux VM**

=====================================================================

Step-1 : Create EC2 VM (amazon linux) & connect with that vm using ssh client

Step-2 : Execute below commands

# Install Docker

sudo yum update -y

sudo yum install docker -y

sudo service docker start

# Add ec2-user user to docker group

sudo usermod -aG docker ec2-user

# Exit from terminal and Connect again

exit

# Verify Docker installation

docker -v

=====================================================================

Create account in Docker Hub

Url - <https://hub.docker.com/>

And download Docker hub.

Ashokit Docker repositories

Url-https://hub.docker.com/search?q=ashokit

Download repo from ashokit

url -docker pull ashokit/spring-boot-rest-api

Always we are going to run docker container in detached mode to execute other commands in terminal otherwise terminal will be blocked.

Command – docker rum –d < Image Id >

**03-28-Docker-13-Nov-24**

=====================================================================

**Docker commands**

=====================================================================

docker images : To display docker images available in our system.

* Default docker image is Hello-world
* Command – docker pull hello-world

docker pull <image-id/name> : To download docker image from docker hub.

docker rmi <image-id/name> : To delete docker image.

* If the container is running we cant delete images directly first we need to delete container then we can delete the images.

docker run <image-id/name> : TO create/run docker container.

docker ps : To display running docker containers.

docker ps -a : To display running + stopped containers.

docker stop <container-id> : To stop running docker container.

docker start <container-id> : To start docker container which is in stopped state.

docker rm <container-id> : To delete docker container.

* Only container will be deleted not the images.

# delete stopped containers + unused images + build cache

docker system prune -a

docker build -t <tag-name> . : To build docker image , -t represent tag name

docker login : To login into docker hub account

docker push <img-name> : To push docker img into docker hub

docker logs < Container-id > : To see container logs.

**Date: 13-Nov-2024 Topic : Docker Containers**

=====================================================================

**QUE - 1) What is detached mode** ?

ANS - It is used to created in background. It will allow us to run commands in terminal.

Ex : docker run -d ashokit/spring-boot-rest-api

**QUE -2) What is port mapping ?**

* It is used to map container port number to host machine port number
* Once we perform port mapping then we can access our application which is running inside the container by using host machine public address.

Syntax : docker run -d -p <host-port:container-port> <image-name>

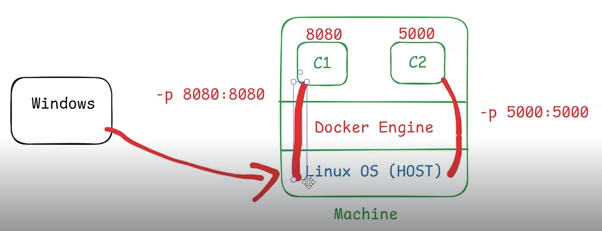
Port no Host id

Ex : docker run -d -p 9090:9090 ashokit/spring-boot-rest-api

Note: To access our application we need to enable HOST PoRT in security group inbound rules.

App URL : http://public-ip:host-port/welcome/raja

Note: If we are using docker in windows machine instead of public we can use localhost in the url.



* If you want to access the application which is running in the coantainer the you need to do port mapping .
* for that container port number we need to map with the host post number ,Then only then we can access the application.

Host is nothing but the machine in which we have installed the docker.

**Interview Question**

QUE -Can run two containers on same port number ?

Ans -Yes , you can run two containers on same port number but when you map port number with host id then host id should be different otherwise we wont able to run two application on same port

=====================================================================

**Running Real-world applications using docker images**

=====================================================================

docker pull ashokit/spring-boot-rest-api

docker run ashokit/spring-boot-rest-api

docker run -d ashokit/spring-boot-rest-api

docker ps

docker logs <container-id>

docker run -d -p host-port:container-port ashokit/spring-boot-rest-api

Ex: docker run -d -p 9090:9090 ashokit/spring-boot-rest-api

########### Java App URL : http://public-ip:host-port/welcome/{name}

docker pull ashokit/python-flask-app

docker run -d ashokit/python-flask-app

docker run -d -p 5000:5000 ashokit/python-flask-app

########### Python App URL : http://public-ip:host-port/

Note: Here -d represents detached mode.

Note: Here -p represents port mapping. (host-port:container-port)

Note: host port and container port no need to be same.

Note: Host port number we need to enable in ec2-vm security group inbound rules to allow the traffic.

How to create a dockerfile and docker img ?

* command – docker build –t < img-name > . ( Dot refers the current working directory in which our docker file is going to execute)
* You can specify other directories instead of . if your Dockerfile or build context is located elsewhere. For example:
* docker build -t image-name /path/to/context  
  Here, /path/to/context is used as the build contex

=====================================================================

Dockerfile

=====================================================================

=> Dockerfile contains set of instructions to build docker image.

file name : Dockerfile

Note: We will keep Dockerfile inside project directory.

=> To write dockerfile we will use below keywords

1) FROM

2) MAINTAINER

3) RUN

4) CMD

5) COPY

6) ADD

7) WORKDIR

8) EXPOSE

9) ENTRYPOINT

10) USER

**04-28-Docker-14-Nov-24**

=====================================================================

**FROM**

=====================================================================

* It is used specify base image to create our docker image.
* Which software is required to run your application that will specify in dockerfile.
* Base images we can find in docker hub only

Ex:

FROM tomcat:9.0

FROM openjdk:17

FROM python:3.3

FROM node:19

FROM mysql:8.5

=====================================================================

**MAINTAINER**

=====================================================================

=> To specify author of Dockerfile (who created/modifed Dockerfile)

Ex:

MAINTAINER Ashok<ashok.b@oracle.com>

**Note: It is optional.**

========

**RUN**

========

=> RUN keyword is used to specify instructions (commands) which are required to execute at the time of docker image creation.

Ex:

RUN 'git clone <repo-url>'

RUN 'mvn clean package'

Note: We can specify multiple RUN instructions in Dockerfile and all those will execute in sequential manner.

**========**

**CMD**

**========**

=> CMD keyword is used to specify instructions (commands) which are required to execute at the time of docker container creation.

Ex:

CMD "java -jar <jar-file-name>"

CMD "python app.py"

Note: If we write multiple CMD instructions in dockerfile, docker will execute only last CMD instruction.

**=====**

**COPY**

**=====**

=> COPY instruction is used to copy the files from source to destination.

Note: It is used to copy application code from host machine to container machine.

Source : HOST Machine

Destination : Container machine

EX:

COPY target/app.jar /usr/app/

COPY target/webapp.war /usr/app/

COPY app.py /usr/app/

**=====**

**ADD**

**=====**

=> ADD instruction is used to copy the files from source to destination.

* **File copying:** Similar to COPY.
* **Remote URLs:** Can download files from a URL and place them in the image.
* **Tar extraction:** Automatically extracts compressed .tar archives into the image.

EX:

ADD target/app.jar /usr/app/

ADD <file-url> /usr/app/

**========**

**WORKDIR**

**========**

=> WORKDIR instruction is used to set / change working directory in container machine.

Ex:

COPY target/app.jar /usr/app/

WORKDIR /usr/app/

CMD "java -jar app.jar"

**========**

**EXPOSE**

**========**

=> EXPOSE instruction is used to specify application is running on which PORT number.

Ex:

EXPOSE 8080

**===========**

**ENTRYPOINT**

**===========**

=> It is used to execute instruction when container is getting created.

Note: ENTRYPOINT is used as alternate for 'CMD' instructions.

CMD "java -jar app.jar"

ENTRYPOINT ["java", "jar", "app.jar"]

================================================

What is the diff between 'CMD' & 'ENTRYPOINT' ?

=================================================

CMD instructions we can override.

ENTRYPOINT instructions we can't override.

===========

USER

===========

=> It is used to set user account to execute dockerfile commands

USER 'ashokit'

RUN echo 'hi'

===================================

Dockerizing SpringBoot application

===================================

# App Git Repo : https://github.com/ashokitschool/spring-boot-docker-app.git

=> Spring Boot is a java framework which is used to develop java based applications.

=> Spring Boot applications will be packaged as a jar file.

=> To run the jar file we will use below command

Ex: java -jar app.jar

Note: When we run springboot application jar file, internally springboot will use tomcat server as "embedded container" with default port number 8080.

================ Java SpringBoot App Dockerfile ===========

FROM openjdk:17

MAINTAINER "Ashok"

COPY target/sb-app.jar /usr/app/

WORKDIR /usr/app/

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "sb-app.jar"]

=============================================================

1) Clone git repo

git clone https://github.com/ashokitschool/spring-boot-docker-app.git

2) Go inside project directory and perform maven build

cd spring-boot-docker-app

mvn clean package

3) Create docker image

docker build -t ashokit/sb-app .

docker images

4) Create docker container

docker run -d -p 8080:8080 ashokit/sb-app

docker ps

docker logs <container-id>

5) Access application URL in browser

windows : http://localhost:8080/

Linux : http://public-ip:8080/

===================================================

Dockerizing Java Web application (no springboot)

===================================================

## App Git Repo : https://github.com/ashokitschool/maven-web-app.git

=> Normal java web apps will be packaged as war file.

Note: war file will be created inside project target directory.

=> To execute that java web application we need to deploy that war file in tomcat server.

=> Inside tomcat server we will have "webapps" folder. It is called as deployment folder.

=> To run war file we need to keep war file inside tomcat/webapps folder.

================ Dockerfile for Java web application ===============

FROM tomcat:latest

MAINTAINER "Ashok<797979>"

EXPOSE 8080

COPY target/maven-web-app.war /usr/local/tomcat/webapps/

====================================================================

1) Clone git repo

git clone https://github.com/ashokitschool/maven-web-app.git

2) Go inside project directory and perform maven build

cd maven-web-app

mvn clean package

3) Create docker image

docker build -t ashokit/maven-web-app .

docker images

4) Create docker container

docker run -d -p 8080:8080 ashokit/maven-web-app

docker ps

docker logs <container-id>

5) Access application URL in browser

windows : http://localhost:8080/maven-web-app

Linux : http://public-ip:8080/maven-web-app

=================================

Dockerizing Python application

=================================

=> Python is a general purpose language

Note: It is also called as scripting language.

=> We don't need any build tool for python applications.

=> We can run python application code directley like below

ex: python app.py

=> If we need any libraries for python (Ex: Flask) application development then we will mention them in "requirements.txt" file

Note: We will use "python pip" s/w to download libraries configured in requirements.txt file.

================== Python Flask App Dockerfile ==================

FROM python:3.6

COPY . /app/

WORKDIR /app/

EXPOSE 5000

RUN pip install -r requirements.txt

ENTRYPOINT ["python", "app.py"]

===================================================================

1) Clone git repo

git clone https://github.com/ashokitschool/python-flask-docker-app.git

2) Go inside project directory and Create docker image

cd python-flask-docker-app

docker build -t ashokit/py-app .

docker images

3) Create docker container

docker run -d -p 5000:5000 ashokit/py-app

docker ps

docker logs <container-id>

4) Access application URL in browser

windows : http://localhost:5000/

Linux : http://public-ip:5000/

================================

How to access docker container

================================

# display docker containers which are in running mode

docker ps

# go inside docker container from linux host

docker exec -it <container-id> /bin/bash

# go inside docker container from windows host

docker exec -it <container-id> sh

========================

Assignments for today

========================

1) Setup Jenkins Server as Docker Container

2) Setup MySQL DB as Docker Container

3) Dockerizing Angular & React application

1) Docker Network

2) Docker Volumes

3) Docker Compose

4) Docker Swarm

================

Docker Network

================

=> Network is all about communication

=> Docker network is used to provide isolated network for containers

=> If we run 2 containers under same network then one contianer can communicate with another container.

=> By default we have 3 networks in Docker

1) bridge

2) host

3) none

=> Bridge network is used to run standalone containers. It will assign one IP for container. It is the default network for docker container.

=> Host network is also used to run standalone containers. This will not assign any ip for our container.

=> None means no network will be available.

# display docker networks

$ docker network ls

# create docker network

$ docker network create ashokit-nw

# inspect docker network

$ docker network inspect ashokit-nw

# run docker container with custom network

$ docker run -d -p 8080:8080 --network ashokit-nw ashokit/maven-web-app

# delete docker network

$ docker network rm ashokit-nw

===============

Docker Compose

===============

=> Earlier ppl developed projects using Monolithic Architecture

(everthing in single app)

Ex :

MakeMyTrip app ------> One docker image ----> One Conainer

=> Now a days projects are developing based on Microservices architecture.

=> Microservices means multiple backend apis will be avialable.

Ex:

1) hotels-api

2) flights-api

3) trains-api

4) cabs-api

=> For every API we need to create seperate container.

Note: When we have multiple containers like this, management will become very

difficult (create / stop / start)

=> To overcome these problems we will use Docker Compose.

=> Docker Compose is used to manage Multi - Container Based applications.

=> In docker-compose, using single command we can create / stop / start multiple containers at a time.

===================================

What is docker-compose.yml file ?

===================================

=> docker-compose.yml file is used to specify containers information.

=> The default file name is docker-compose.yml (we can change it).

=> docker-compose.yml file contains below 4 sections.

version : It represents compose yml version.

services : It represents containers info (image-name, port-mapping).

networks : It represents network to run our containers.

volumes : Represents containers storage location.

======================

Docker Compose Setup

======================

# install docker compose

sudo curl -L "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

# Check docker compose is installed or not

$ docker-compose --version

================================================

Spring Boot with MySQL DB using Docker-Compose

================================================

=> I want to deploy springboot application with MySQl DB

a) MySQL DB will run in one container

b) SpringBoot-App will run in another container

Note-1: SpringBoot application should connect with MySQl DB. That means app container depends on DB Container.

Note-2: As we need containers communication those 2 containers should run on same network.

=> Below is the "docker-compose.yml" to manage springboot-app with mysql-db deployment.

---

version: "3"

services:

application:

image: spring-boot-mysql-app

ports:

- 8080:8080

networks:

- springboot-db-net

depends\_on:

- mysqldb

volumes:

- /data/springboot-app

mysqldb:

image: mysql:5.7

networks:

- springboot-db-net

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=sbms

volumes:

- /data/mysql

networks:

springboot-db-net:

...

================================

Application Execution Process

================================

# clone git repo

git clone https://github.com/ashokitschool/spring-boot-mysql-docker-compose.git

# go inside project directory

cd spring-boot-mysql-docker-compose

# build project using maven

mvn clean package

# build docker image

docker build -t spring-boot-mysql-app .

# check docker images

docker images

# create docker containers using docker-compose

docker-compose up -d

# check docker containers running

docker-compose ps

# stop docker containers running

docker-compose stop

# start docker containers which are stopped

docker-compose start

# delete docker containers using docker-compose

docker-compose down

====================================

Stateful Vs Stateless Containers

====================================

Stateless Container : Data will be deleted after container deletion.

Statefull Container : Data will be available permanently even after container deletion.

Note: Docker containers are stateless by default.

Note: In spring-boot-mysql app, we are using "mysqldb" as docker container to store application data. When we re-create containers db also got recreated hence we lost data (this is not accepted in realtime).

=> To maintain data permanently, we need to make docker container as statefull.

=> To make container as statefull, we need to use Docker volumes concept.

================

Docker Volumes

================

=> Volumes are used to persist data which is generated by docker container.

=> Volumes are used to avoid data loss.

=> Using volumes we can make container as statefull.

=================================

Making docker container stateful

=================================

---

version: "3"

services:

application:

image: spring-boot-mysql-app

ports:

- "8080:8080"

networks:

- springboot-db-net

depends\_on:

- mysqldb

volumes:

- /data/springboot-app

mysqldb:

image: mysql:5.7

networks:

- springboot-db-net

environment:

- MYSQL\_ROOT\_PASSWORD=root

- MYSQL\_DATABASE=sbms

volumes:

- .app:/var/lib/mysql

networks:

springboot-db-net:

...

=> Once we execute above docker-compose yml it will create containers and it will map MySQl DB container data to .app directory.

=> We can check container data in .app directory

ls -la .app

=============

Docker Swarm

=============

=> It is an Orchestration platform.

=> Orchestration means Managing the process (containers).

=> Using Docker swarm we will setup Docker cluster

Note: Cluster means group of servers.

============================

Docker Swarm Cluster Setup

============================

-> Create 3 EC2 instances (ubuntu) & install docker in all 3 instances using below 2 commands

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

-> Out of 3 machines we will consider

1 - Master Node

2 - Worker Nodes

Note: Enable 2377 port in security group for Swarm Cluster Communications

-> Connect to Master Machine and execute below command

# Initialize docker swarm cluster

sudo docker swarm init --advertise-addr <private-ip-of-master-node>

Ex : sudo docker swarm init --advertise-addr 172.31.5.176

# Get Join token from master (this token is used by workers to join with master)

sudo docker swarm join-token worker.

Note: Copy the token and execute in all worker nodes with sudo permission

Ex: sudo docker swarm join --token SWMTKN-1-4pkn4fiwm09haue0v633s6snitq693p1h7d1774c8y0hfl9yz9-8l7vptikm0x29shtkhn0ki8wz 172.31.37.100:2377

Note: With above steps docker swarm cluster is ready.

-> In Docker swarm we need to deploy our application as a service.

sudo docker service create --name maven-web-app -p <host-port:container-port> <imageName>

Ex : sudo docker service create --name java-web-app -p 8080:8080 ashokit/javawebapp

Note: By default 1 replica will be created

Note: We can access our application using below URL pattern

URL : http://master-node-public-ip:8080/java-web-app/

# check the services created

$ sudo docker service ls

# we can scale docker service

$ docker service scale <serviceName>=<no.of.replicas>

# inspect docker service

$ sudo docker service inspect --pretty <service-name>

# see service details

$ sudo docker service ps <service-name>

# Remove one node from swarm cluster

$ sudo docker swarm leave

# remove docker service

$ sudo docker service rm <service-name>

=========================================

Deploy Angular Application using Docker

========================================

=> Angular is a framework which is used to develop frontend of application (UI)

=> We will use NODE software to build and run Angular application.

=> Angular app libraries will be configured in "package.json" file

=> To download librarires configured we will use 'npm install' command.

=> To build angular application we will use below command

npm run build --prod

Note: When we run above command 'dist' folder will be generated for deployment.

=> To deploy angular application we will use NGINX webserver.

==================== Dockerfile of Angular App ================================

FROM node:18 as build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build --prod

FROM nginx:alpine

COPY --from=build /app/dist/<your-project-name> /usr/share/nginx/html

EXPOSE 80

========================================================================

## Step-1 : Clone git repo

git clone https://github.com/ashokitschool/angular\_docker\_app.git

## Step-2 : Go inside project directory and build docker image

cd angular\_docker\_app

docker build -t ng-app .

## Step-3 : Run docker container

docker images

docker run -d -p 80:80 ng-app

## Step-4 : Access angular app in browser

Windows : http://localhost:80/

Linux : http://public-ip:80/

=========================================

Deploy React Application using Docker

========================================

## Step-1 : Clone git repo

git clone https://github.com/ashokitschool/ReactJS\_Docker\_App.git

## Step-2 : Go inside project directory and build docker image

cd ReactJS\_Docker\_App

docker build -t react-app .

## Step-3 : Run docker container

docker images

docker run -d -p 80:80 react-app

## Step-4 : Access react-app in browser

Windows : http://localhost:80/

Linux : http://public-ip:80/

===========================================

Build + deploy Springboot app using docker

===========================================

============ Step-1 : Create Dockerfile with below content ===============

# Stage 1: Build the application

FROM maven:latest AS build

# Set the working directory

WORKDIR /app

# Clone the Git repository

RUN git clone https://github.com/ashokitschool/spring-boot-docker-app.git .

# Build the application

RUN mvn clean package

# Stage 2: Run the application

FROM openjdk:18-jdk-alpine

# Set the working directory

WORKDIR /app

# Copy the built jar file from the build stage

COPY --from=build /app/target/spring-boot-docker-app.jar /app/spring-boot-docker-app.jar

# Expose the port the app runs on

EXPOSE 8080

# Command to run the application

ENTRYPOINT ["java", "-jar", "/app/spring-boot-docker-app.jar"]

============ Step-2 : Build Docker Image ===============

docker build -t sb-app .

============ Step-3 : Run Docker Image ===============

docker run -d -p 8080:8080 sb-app

============ Step-4 : Access Application in browser ===============

Windows URL : http://localhost:8080/

Linux URL : http://public-ip:8080/

===================================================================