Orchestration - Manage
containerization - Packaging

Page No.
Date 09/07/2024

Docker and kubernetes:

Docker ---> Containerization platform (application code + dependencies)
Kubernetes -> Orchestration platform (used to manage containers)

1. Application stack
2. Life without Docker
3. Life with Docker
4. Docker Introduction
5. Docker Architecture
6. Docker Installation (Linux VM - AWS)
7. Docker Commands
8. Docker File
9. Docker Network
10. Docker Volumes
11. Docker Compose
12. Docker Swarm (Orchestration platform)

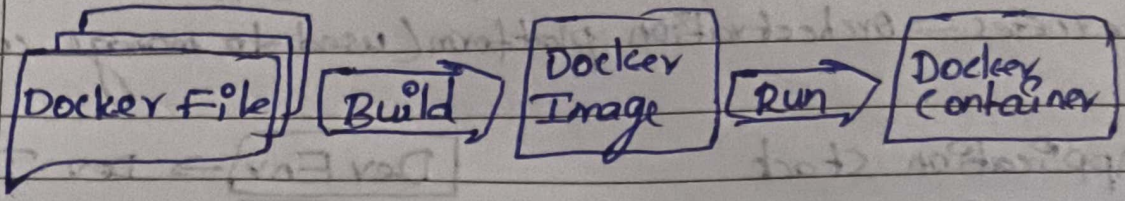
Dev Env -> Dev Testing
SIT ENV -> S/W Integration Test
UAT ENV -> User Acceptance Testing [Client Testing]
Pre-Prod -> Live Data Testing.
Prod Env -> Live Environment

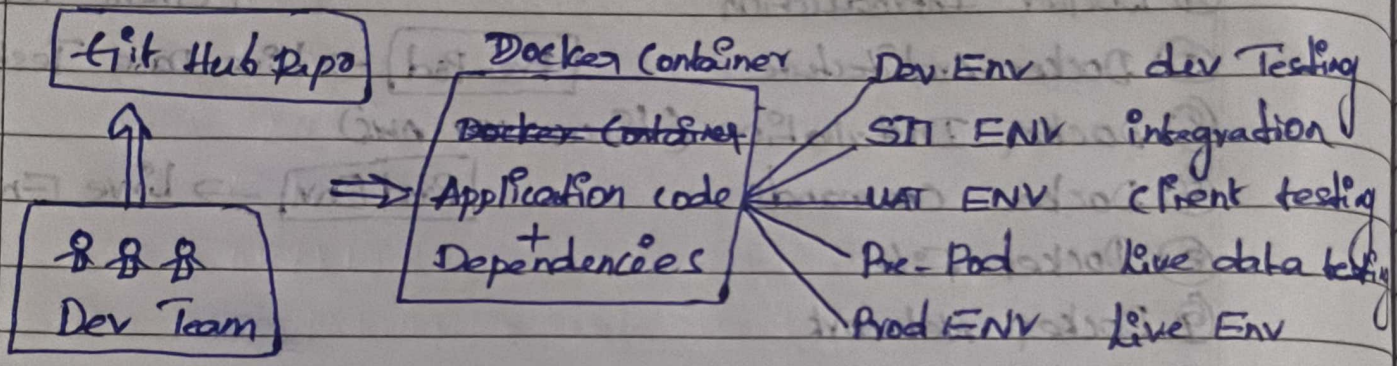
15 pa

Docker is a platform for packaging, deploying, and running applications.
Docker enables you to separate your applications from your infrastructure so you can deliver software quickly
Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code and runtime
By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay b/w writing code and running it in production

Page No.
Date / /

Docker Architecture :





Docker :

→ Docker is a containerization software
→ Docker is used to simplify our application deployment process
→ Docker will take care of required dependencies of our application
→ Using Docker we will run application as a container

What is Containerization

→ The process of packaging our application code + dependencies as single unit and executing inside container is called as Containerization.
→ Container is a Virtual Machine (Linux-VM)

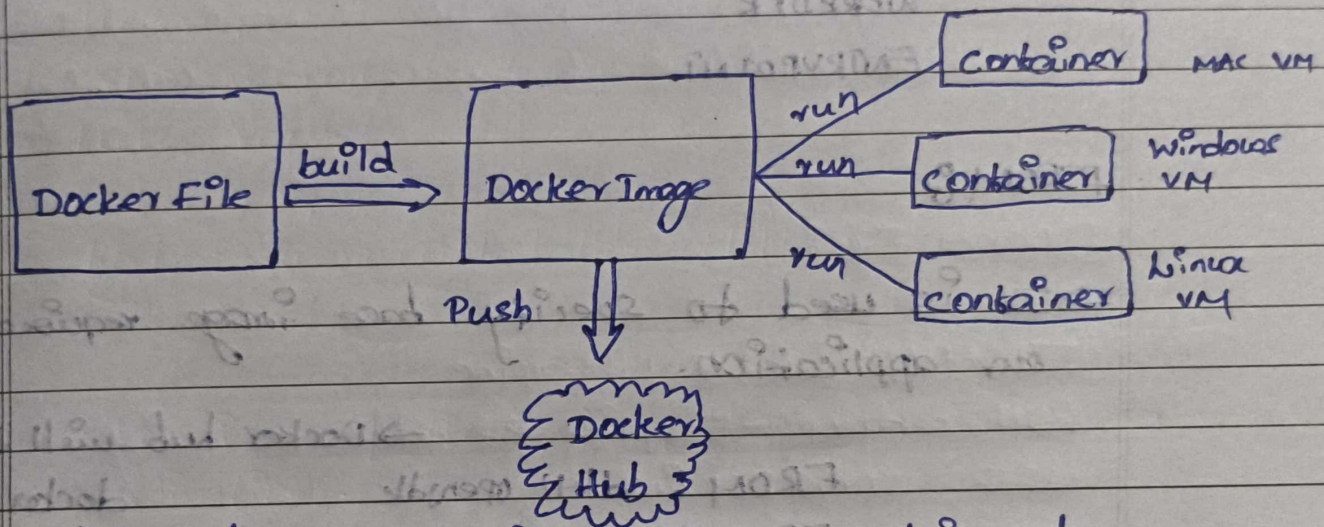
Page No.
Date 18 / 07 / 2024

Docker is an open-source platform for developing, shipping and running applications in containers.

Containers are lightweight, isolated environments that package applications and their dependencies.

Benifits of using Dockers
Portability, scalability, consistency, and resource efficiency.

Docker Architecture



→ docker logs <container-id> : To see container logs.
→ docker images : to display available docker images
→ docker pull <image-name> : download docker image
→ docker pull Mahesh/Java-project (We can run also docker run)
→ docker ps : display running docker containers
→ docker ps -a : display running + stopped containers
→ docker rmi <img-id> : To delete docker image
→ docker rm <container-id> : To delete stopped docker container
→ docker stop <container-id> : To stop running container
→ docker start <container-id> : To start container
→ docker system prune -a : to delete un-used images + stopped container
→ docker run -d -p 9090:9090 Mahesh/Java project [Create container]
-d represents detached mode
-p represents port Mapping

Page No.

Date 11 / 07 / 2024

Docker File:

It contains instructions to build image. We will specify application dependencies here

Docker File keywords

FROM
MAINTAINER
COPY
RUN
CMD
EXPOSE
WORKDIR
ENTRYPOINT

FROM:

→ It is used to specify base image required for our application.

→ Docker hub will find docker image

FROM: ~~tdkll~~ openjdk
FROM: tomcat8.5
FROM: mysql8.5
FROM: python-3.1
FROM: node-19

MAINTAINER

It is used to specify author of Dockerfile.

MAINTAINER <mahedr@gmail.com>

COPY

It is used to copy the files from host machine to container machine.

<SRC> <DEST>

COPY target/app.war /usr/app/tomcat/webapp.war

Page No.
Date / /

RUN :-

It is used to execute instructions while creating docker image

```
RUN 'sudo yum install git'
RUN 'sudo apt install maven'
RUN 'git clone <repo>'
```

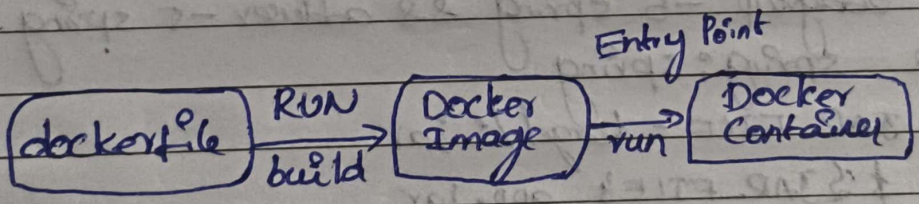
Note: We can run write multiple RUN instructions in dockerfile and they will be processed from top to bottom.

CMD :

It is used to execute instructions while creating docker container

```
CMD 'java -jar jar-file>
```

Note: If we write multiple CMD instructions docker will process only last CMD instructions



EXPOSE

It is used to specify container port number → just for documentation purpose

```
EXPOSE 8080
```

WORKDIR

It is used to specify working directory. (path change)

```
WORKDIR
```

Page No.
Date / /

Linux – Docker image creation:

```
FROM ubuntu
MAINTAINER <Mahesh>
RUN 'echo run msg-1'
RUN 'echo run msg-2'
CMD 'echo cmd msg-1'
CMD 'echo cmd msg-2'
```

{ – image creation RUN instructions should be executed }

{ container creation CMD will execute }

```
$ vi Dockerfile
$ cat Dockerfile
$ ls -l
-rw-r--r--. 1 ec2-user 128 July 22 03:56 Dockerfile
$ docker build -t app1.
```

(-t) → tag

Docker File:

①
```
FROM openjdk:8-jdk-alpine
RUN addgroup -S spring && adduser -S spring -G spring
USER spring:spring
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

②
```
FROM openjdk:11.0.3-jdk-slim
RUN mkdir /usr/myapp
COPY target/java-kubernates-0.0.1-SNAPSHOT.jar /usr/myapp/app.jar
WORKDIR /usr/myapp
Expose 8080
CMD ["java","-Xms128m","-Xmx256m","-jar","app.jar"]
```