**Docker :** Docker is a platform which help developer to package the application with their dependencies into a container. Docker container is light weighted container which help to run the docker image.

Container : run time environment.

Docker image : Docker image is ready only template file which is responsible to run the application with help of container.

DockerFile : DockerFile is a type of file which contains set of instruction which help to create the image. Inside this file we need to provide application details which we want run with their dependencies.

DockerHub : DockerHub is an open source repository for docker images. We can publish or push as well as pull user defined or pre defined images.

**Docker images**

docker --version

docker info

docker images : this command show all images present in local machine.

docker pull imageName : this command pull the image from Docker hub account(by default Docker hub repository)

docker pull hello-world : hello-world is pre defined image

to run the image

docker run imageName/imageId this command is use to run the image

**hello-world :** this image is responsible to run C program

**busybox :** this is OS image

**creating custom or user defined images**

1. Creating user defined image to display echo message.

Create the Dockerfile

**FROM busybox**

**CMD ["echo","Welcome to Docker image created by Akash kale"]**

Now you need to create the image

**docker build -t my-busybox . -f Dockerfile**

1. Creating image to run java program

public class Demo {

    public static void main(String args[]){

        System.out.println("Welcome to Java, Running through Docker");

    }

}

**javac Demo.java** compile the program

**java Demo**  run the program

**Dockerfile**

FROM openjdk:8

COPY Demo.java .

RUN javac Demo.java

CMD ["java","Demo"]

**Create the image**

**docker build -t my-java . -f Dockerfile**

1. Creating image to run spring boot application.

Create spring boot application with web as well as thymeleaf starter.

Create jar or war (ie build file using eclipse ide or maven command)

**mvn clean package**

running the spring boot project using jar file

jar file by default present inside target folder

**java -jar filename.jar**

now you need to create the Dockerfile

**FROM openjdk:17**

**COPY ./target/spring-boot-docker-0.0.1-SNAPSHOT.jar .**

**CMD ["java","-jar","spring-boot-docker-0.0.1-SNAPSHOT.jar"]**

Build the image

**docker build -t my-spring-boot . -f Dockerfile**

if image is responsible to run web application using any server

we need to run below command as

docker run -d -p portNumber:portNumber imageName

-d : detached mode

-p public port number

Before : port number expose port number. It can be same or different

After : port number must actual application port number.

**docker run -d -p 8080:8080 my-spring-boot**

**docker ps :** it show running container

**docker ps -a** : it show active as well as inactive container

**container related commands**

docker stop containerId/containerName

docker start containerId/containerName

docker rm containerId/containerName

docker rm containerId/containerName -f

docker rmi imageName/imageId

docker rmi imageName/imageId -f

**creating Image for React JS Application**

create the React JS project

**npx create-react-app react-with-docker**

after development task.

Please test code using **npm start**

We need to build the project

**npm run build**

to run frontend technologies like react js or angular or html/css/js. We use one of the open source server ie **nginx**

nginx run on default port number **80**