Docker Workshop for Complete Beginners

05 December 2021 08:38

# Docker Commands

docker -v : To check docker version docker info : To get info about docker

docker login : To login into docker hub account docker images : To check all docker images

docker pull <image-id> : To pull docker image from docker hub docker run <image-id> : To run docker image

docker rmi <image-id> : To remove docker image docker ps : To check docker containers running

# Running hello-world docker image

> docker run hello-world

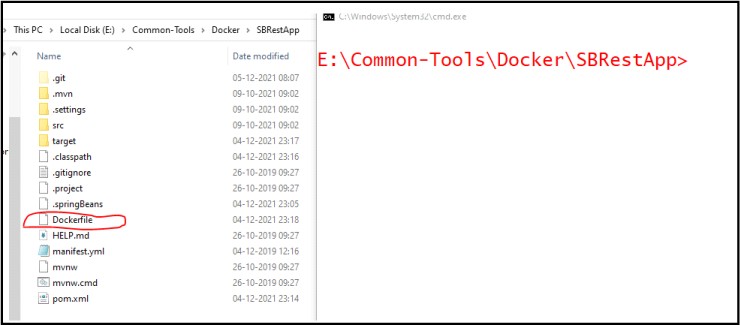
Note: When we run an image docker will search for that image in local, if it not available in local then it will download from docker hub and it will run that image.

Running Spring Boot Application Using Docker

-> Clone Spring Boot application from Git repo

https://github.com/Ashok-IT-School/spring-boot-docker-app.git

-> Open command prompt and navigate to spring-boot application till dockerfile location



-> Build Docker image with dockerfile

Syntax : docker build -t <image-name> . : To build docker image using Dockerfile

# Ex : docker build -t docker-image .

Note : After building check all images using docker images command

-> Run docker image

Syntax : docker run -p OS-HOST-PORT:APP-SERVER-PORT <image-name> : To run docker image

# Ex : docker run -p 9090:9090 spring-boot-rest-api

Note: We can run docker image in detached mode also

Syntax: docker run -d -p OS-HOST-PORT:APP-SERVER-PORT <image-name> : To run docker image in background (detached mode)

# Ex : docker run -d -p 5000:5555 spring-boot-rest-api

-> Tag docker image

Syntax: docker tag image-name username/repository:tag : To tag an image

# Ex : docker tag spring-boot-rest-api ashokit/spring-boot-rest-api:latest

-> Push docker image to docker hub Syntax : docker push username/image

# Ex: docker push ashokit/spring-boot-rest-api

**Add This Final Name in porm.xml file to generate jar**

**To right click on the project then run as configure after the build maven.**

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

<finalName>docker-hello-word</finalName>

</build>

**Docker file for Spring Boot Application**

# Start with a base image containing Java runtime

FROM java:8

# Make port 8080 available to the world outside this container

EXPOSE 8080

ADD target/spring-boot-docker-maven.jar spring-boot-docker-maven.jar

# Run the jar file

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

Other Way

<build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

            <plugin>

                <groupId>com.spotify</groupId>

                <artifactId>dockerfile-maven-plugin</artifactId>

                <version>1.4.0</version>

                <configuration>

                    <repository>javatechie/${project.artifactId}</repository>

                    <tag>${project.version}</tag>

                    <buildArgs>

                        <JAR\_FILE>target/${project.build.finalName}.jar</JAR\_FILE>

                    </buildArgs>

                </configuration>

                <executions>

                    <execution>

                        <id>default</id>

                        <phase>install</phase>

                        <goals>

                            <goal>build</goal>

                            <goal>push</goal>

                        </goals>

                    </execution>

                </executions>

            </plugin>

        </plugins>

        <finalName>spring-boot-docker-maven</finalName>

    </build>

# How to make an executable jar in Maven

**Add Final name jar in porm.xml 🡺**<finalName>docker-image</finalName>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

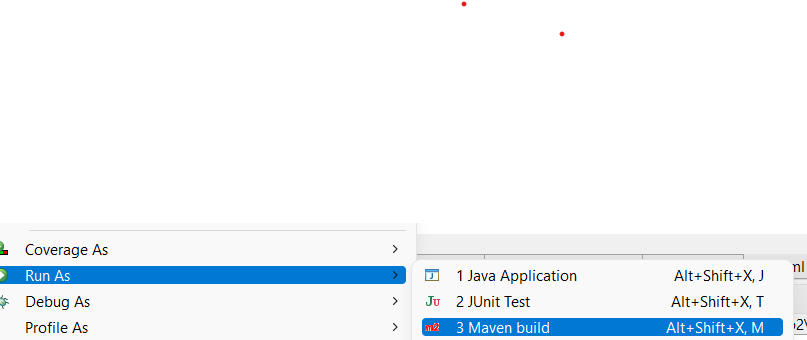
</plugin>

</plugins>

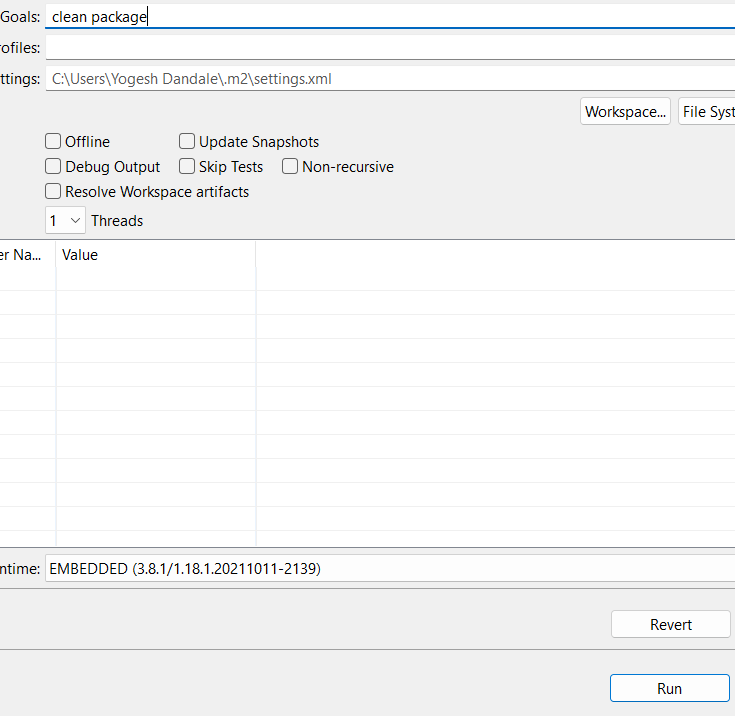
<finalName>docker-image</finalName>

</build>

Then click right on the project Run as built maven

****

Then write in Goals clean package then click run option



After creating jar then add Docker file in project

**Docker file for Spring Boot Application**

# Start with a base image containing Java runtime

FROM java:8

# Make port 8080 available to the world outside this container

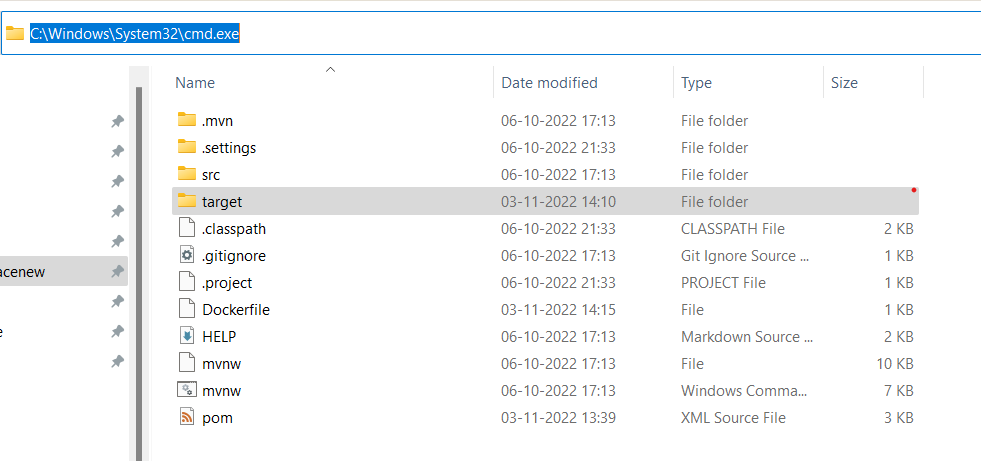
EXPOSE 8080

ADD target/spring-boot-docker-maven.jar spring-boot-docker-maven.jar

# Run the jar file

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

**Then Go to project structure into Dockerfile and open camand line**

****

Write command on cmd docker build -t docker iamge name and dot

docker build -t <docker-image> .

run image

docker-image

push to docker hup

>docker tag <docker image name> yogeshdandale1/<docker image name> :latest

# Ex : docker tag spring-boot-rest-api ashokit/spring-boot-rest-api:latest

>docker push yogeshdandale1/<image name>

# Ex: docker push ashokit/spring-boot-rest-api

# Kubernate Explaning link

# 1)Command prompt

# <https://medium.com/@javatechie/kubernetes-tutorial-setup-kubernetes-in-windows-run-spring-boot-application-on-k8s-cluster-c6cab8f7de5a>

# 2)yml file

# https://medium.com/@javatechie/kubernetes-tutorial-run-deploy-spring-boot-application-in-k8s-cluster-using-yaml-configuration-3b079154d232