**BACKEND(MDM-SERVICE)**

* Cloned the GitLab repository into IDE
* Download the extensions which is required to the project(ex:Gradle)
* Create a JAR file which is used to aggregate a collection of files into a single one.

Step-1:On right side click on Gradle icon ->Tasks ->Build ->clean -> bootJar

This creates a JAR file

* Create a Dockerfile
* FROM openjdk:17-oracle  
  VOLUME /tmp  
  COPY build/libs/mdm-service-0.0.1-SNAPSHOT.jar mdm-service.jar  
  ENTRYPOINT ["java","-jar","/mdm-service.jar"]
* In Terminal Build the docker file with command

Docker build -t <docker username>/<image name>

* Run the Dockerfile with adding port

docker run -p 8090:8090 sravyaj02/mdm-service:latest

* This creates a image and to check the image

Docker images

* To run the docker image

Docker run (image name or id)

* To push the image to docker hub

docker push sravyaj02/mdm-service

**Pushing the code from Local to GitLab:**

* Clone the running code from GitLab to respective IDE(vs code)
* Check the exact branch where you want to push change the branch from down left corner and check it by using **GIT BRANCH**
* Copy the code from local to cloned folder
* In terminal

**git status**

**git add .**

**git commit -m “msg”**

**git push**

* In GitLab add a new file ‘gitlab-ci.yml’ to specify instructions for GitLab CI/CD

**variables:**

**IMAGE\_OPENJDK\_GRADLE: gradle:7.3.3-jdk17-alpine**

**CI\_REGISTRY\_IMAGE: sravyaj02/mdm-service:latest**

**stages:**

**- build**

**# - test**

**- build-image**

**# - publish-image**

**build-job:**

**image: $IMAGE\_OPENJDK\_GRADLE**

**stage: build**

**script:**

**- echo "Compiling the code..."**

**- sh $CI\_PROJECT\_DIR/gradlew assemble**

**artifacts:**

**paths:**

**- build/libs/\*.jar**

**#test:**

**#image: $IMAGE\_OPENJDK\_GRADLE**

**#stage: test**

**#script:**

**# - echo "Running tests..."**

**#- sh $CI\_PROJECT\_DIR/gradlew test**

**build-image:**

**image: docker:latest**

**services:**

**- docker:dind**

**stage: build-image**

**script:**

**- echo "Building Docker Image..."**

**- docker build -t "$CI\_REGISTRY\_IMAGE" .**

**- docker login -u "$CI\_REGISTRY\_USER" -p $CI\_REGISTRY\_PASSWORD $CI\_REGISTRY**

**- docker push "$CI\_REGISTRY\_IMAGE"**

**# publish-image:**

**# image: docker:latest**

**# services:**

**# - docker:dind**

**# stage: publish-image**

**# script:**

**# - echo "Publishing Docker Image..."**

**# - docker login -u "$CI\_REGISTRY\_USER" -p $CI\_REGISTRY\_PASSWORD docker.io**

**# - docker push sravyaj02/mdm-service:latest**

* Click on Commit changes button which triggers the pipeline automatically.
* Check whether the stages of pipeline is is getting succeeded.
* Now to run the image or to check the portal is working on
* Pull the latest image from docker hub to local

Docker pull <image name>

* Run the image to check the portal

Docker run -p 8090:8090 <image name>

**To Automate this entire process:**

* To automate this we need to run the script where the script give the instructions or steps how to automate the manual process.
* Bash script :

#!/bin/bash

DOCKER\_USERNAME=sravyaj02

DOCKER\_PASSWORD=jonnalagadda

docker login -u $DOCKER\_USERNAME -p $DOCKER\_PASSWORD

IMAGE\_NAME=$1

if docker image inspect $IMAGE\_NAME > /dev/null; then

# The Docker image exists, so pull it

echo "Pulling Docker image..."

docker pull $IMAGE\_NAME

fi

echo "Running Docker image..."

winpty docker run -it $IMAGE\_NAME

* Save this file as deploy.sh and do gitbash from the path where it got saved.

chmod 755 deploy.sh

bash deploy.sh <image name>

This run the script and we can check the portal is working or not.