|  |  |
| --- | --- |
| **Sec 6** | **Profile creation for microservice.** |
| **configserver** | **Building config server for microservice.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**In terminal**

mvn clean install **– used to compile maven project, create jar on the basic of artifact id and version as mentioned in pom.xml and generate fat jar**

mvn spring-boot:run**- it will start the Springboot project**

or

java -jar Accounts-0.0.1-SNAPSHOT.jar

ctrl +c – stop the project running

**Docker**

**docker version – for start Docker**

* **Image**: Think of it as a template or blueprint containing all the dependencies and instructions needed to create a container. It's a snapshot of a filesystem and parameters needed to run a container.
* **Container**: An instance of an image that is running as a process. It's a lightweight, standalone, and executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and configuration files.

Email – [kandk7543@gmail.com](mailto:kandk7543@gmail.com)

Username - kanhaiyadocker

**Generate a docker file in Springboot project(name = Dockerfile)**

And paste –

#Start with a base image containing Java runtime

FROM openjdk:17-jdk-slim

#Information around who maintains the image

MAINTAINER eazybytes.com

# Add the application's jar to the image

COPY target/accounts-0.0.1-SNAPSHOT.jar accounts-0.0.1-SNAPSHOT.jar – verify with generated jar in target

# execute the application

ENTRYPOINT ["java", "-jar", "accounts-0.0.1-SNAPSHOT.jar"]

**docker build . -t kanhaiyadocker/accounts:**s4 ----For this command , we tells the docker to generate a docker image. Before running this command check your docker server is running. To run the docker server-

docker images - it will show the all images.

docker inspect image (image id- initial 3 or 4 char of image id)- it will show the all details of image

docker run -p 8080:8080 kanhaiyadocker/accounts:s4 **– it will start outer world+ docker(2nd port) for docker**

ctrl + c**= shoutdown**

docker run -d -p 8080:8080 kanhaiyadocker/accounts:s4 **= it will start in detached mode means not show logs or copy container id from docker destop and write**

docker start (copied container id) **and same for stop it**

docker stop (copied container id)

docker ps**- it will show all the running container**

docker ps –a **= it will show all the stop container**

docker stop (container id)-

**we can run many command just changing 1st port name ex –**

docker run -d -p 808**1**:8080 kanhaiyadocker/accounts:s4

**2. Using build pack – using we can transfer our application source code into Docker image, so we can run it on any cloud(best)**

**Springboot version -2.6.0**

**Step 1-**

changes in pom

<packaing > jar

And

<configuration>

<image>

<name>kanhaiyadocker/${project.artifactId}:s4</name>

</image>

**Step 2**

mvn spring-boot:build-image **in file location terminal**

docker run -d -p 8090:8090 kanhaiyadocker/loans:s5

**3. docker jib (only for java) for this project we will use jib**

**a.** <plugin>

<groupId>com.google.cloud.tools</groupId>

<artifactId>jib-maven-plugin</artifactId>

<version>3.4.1</version>

<configuration>

<to>

<image>kanhaiyadocker/${project.artifactId}:s4</image>

</to>

</configuration>

</plugin>

**mvn compile jib:dockerBuild**

docker run -d -p 9000:9000 kanhaiyadocker/cards:s4

**docker push command**

docker image push docker.io/kanhaiyadocker/accounts:s4

**docker pull command**

docker pull kanhaiyadocker/cards:s4

**docker compose is used to run/start multiple images simultaneously.- docker compose version**

**With the help of yml file – docker-compose.yml (name format)**

services:

accounts:

image: "kanhaiyadocker/accounts:s4"

container\_name: accounts-ms

ports:

- "8080:8080"

deploy:

resources:

limits:

memory: 700m

networks:

- rajebank

loans:

image: "kanhaiyadocker/loans:s5"

container\_name: loans-ms

ports:

- "8090:8090"

deploy:

resources:

limits:

memory: 700m

networks:

- rajebank

cards:

image: "kanhaiyadocker/cards:s4"

container\_name: cards-ms

ports:

- "9000:9000"

deploy:

resources:

limits:

memory: 700m

networks:

- rajebank

networks:

rajebank:

driver: "bridge"

open the project in terminal in which **docker-compose.yml** is present

docker compose up –d –create and start all the ms container

docker compose start - start all the ms container

docker ps – it will show all the running container

docker compose down- delete all the ms container

docker compose stop – stop the all running container

**sec 6**

Make a config server

Dependency – config server-

+ actuator

And add @EnableConfigServer in main class

Set port , create all ms profile under resources