|  |
| --- |
| Docker |

**Updating microservices to make them Docker compatible**

**Description:** This repository has three maven projects with the names **accounts, loans, cards** which are continuation from the **section4** repository. These three projects are updated, so that Docker images can be generated for the three microservices. Below are the key steps that are followed inside this section where we focused on docker images generation and setup of docker-compose file for all the three microservices.

**Key steps:**

* Go to <https://www.docker.com/> and create an account in it. Make sure to install docker desktop based on the OS that you are working like we discussed in the course.
* Once docker installation is successfull, please verify the same by running a **"docker --version"** command to confirm if the docker installation is successful or not.
* Make sure you are logged in to your docker desktop with the account details that you have created in step 1.
* Make sure maven is installed inside your system by running a command **"mvn --version"**
* Go to the folder location where **accounts** microservice **pom.xml** is present and run the command **"mvn clean install"**. This command will generate accounts microservice jar **"accounts-0.0.1-SNAPSHOT.jar"** under the **target** folder of **accounts** microservice.
* Validate if the jar is generated as expected in the above step by running the command **"mvn spring-boot:run"** from the folder location where **accounts** microservice **pom.xml** is present. This will start the accounts microservice at port 8080. We can validate the same by invoking the REST API <http://localhost:8080/myAccount> through Postman by passing the below request in JSON format. You should get the response from the corresponding microservices. Post validating the response stop the accounts microservice.

{

"customerId": 1

}

* We can start and verify accounts microservice response by running another command **"java -jar target/accounts-0.0.1-SNAPSHOT.jar"** from the command line tool. Post validating the response stop the accounts microservice.
* Now lets create the **Dockerfile** inside **accounts** microservice root folder with the following content. This **Dockerfile** will be used to generate a docker image by using the basic and lengthy approach that we discussed in the course.

#Start with a base image containing Java runtime

FROM openjdk:17-jdk-slim as build

#Information around who maintains the image

MAINTAINER eazybytes.com

# Add the application's jar to the container

COPY target/accounts-0.0.1-SNAPSHOT.jar accounts-0.0.1-SNAPSHOT.jar

#execute the application

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

* Make sure docker server is up and running in your machine. Now run the docker command **"docker build . -t eazybytes/accounts"** from the location where **Dockerfile** is created in the above step.
* The above step should have generated **accounts** microservice docker image and the same can be validated by running the docker command **"docker images"**
* Now create the **accounts** microservice container from the docker image generated in the above step by running the docker command **"docker run -p 8080:8080 eazybytes/accounts"**. We can validate the docker container created by invoking the REST API <http://localhost:8080/myAccount> through Postman by passing the below request in JSON format. You should get the response from the accounts microservice docker container created in this step.

{

"customerId": 1

}

* Now create one more container/instance of accounts microservice by running the docker command **"docker run -p 8081:8080 eazybytes/accounts"**. We can validate the docker container created with exposed port 8081 by invoking the REST API <http://localhost:8081/myAccount> through Postman by passing the below request in JSON format. You should get the response from the accounts microservice docker container created in this step.

{

"customerId": 1

}

* Run the docker command **"docker ps"** to see the 2 running containers of **accounts** microservice inside the docker server.
* Run the docker command **"docker stop [container-id]"** for both the 2 running containers of **accounts** microservice to stop them.
* Now in order to generate docker images for **loans** and **cards** microservices using **Buildpacks**, update the respective **pom.xml** files with the below build plugin details. This will allow us to create docker images with out the need of **Dockerfile** like we created for **accounts** microservice.

**loans\pom.xml, cards\pom.xml**

<build>

<plugins>

<plugin>

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

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

<configuration>

<image>

<name>eazybytes/${project.artifactId}</name>

</image>

</configuration>

</plugin>

</plugins>

</build>

* Now run the maven command **"mvn spring-boot:build-image"** for both **loans** & **cards** microservices using command line tool from the folder location where pom.xml is present. This will generate the docker images for both **loans** & **cards** microservices using **Buildpacks**. The same can be validated by running the docker command **"docker images"** which will list all the docker images.
* Now create the **loans** microservice container from the docker image generated in the above step by running the docker command **"docker run -p 8090:8090 eazybytes/loans"**. We can validate the docker container created by invoking the REST API <http://localhost:8090/myLoans> through Postman by passing the below request in JSON format. You should get the response from the loans microservice docker container created in this step.

{

"customerId": 1

}

* Now create the **cards** microservice container from the docker image generated previously by running the docker command **"docker run -p 9000:9000 eazybytes/cards"**. We can validate the docker container created by invoking the REST API <http://localhost:9000/myCards> through Postman by passing the below request in JSON format. You should get the response from the cards microservice docker container created in this step.

{

"customerId": 1

}

* Run the docker command **"docker ps"** to see the 2 running containers of **loans & cards** microservices inside the docker server.
* Run the docker command **"docker stop [container-id]"** for both the 2 running containers of **loans & cards** microservice to stop them.
* Push all your docker images of **accounts, loans, cards** into docker hub by running the command **"docker push docker.io/[image-name]"**. Please make sure you logged into your docker account inside your docker desktop server, otherwise you will get authentication related issues while trying to push the docker images into docker hub. Validate if the docker images successfully pushed or not by loging into <https://hub.docker.com/> website.
* In order to start all our docker images using a single docker compose command, install Docker compose by following the instruction mentioned in the <https://docs.docker.com/compose/install/> website. More details can be found in the course.
* Post installation of docker compose, validate the same by running the docker compose command **"docker-compose --version"**
* Now write a **docker-compose.yml** file inside accounts project with the following content,

version: "3.8"

services:

accounts:

image: eazybytes/accounts:latest

mem\_limit: 700m

ports:

- "8080:8080"

networks:

- eazybank-network

loans:

image: eazybytes/loans:latest

mem\_limit: 700m

ports:

- "8090:8090"

networks:

- eazybank-network

cards:

image: eazybytes/cards:latest

mem\_limit: 700m

ports:

- "9000:9000"

networks:

- eazybank-network

networks:

eazybank-network:

* Open the command line tool where the **docker-compose.yml** is present and run the docker compose command **"docker-compose up"** to start all the microservices containers with a single command. All the running containers can be validated by running a docker command **"docker ps"**.
* Invoke the REST APIs <http://localhost:8080/myAccount>, <http://localhost:8090/myLoans>, <http://localhost:9000/myCards> through Postman by passing the below request in JSON format. You should get the response from the corresponding microservices.

{

"customerId": 1

}

* Stop the three running containers by executing the docker compose command **"docker-compose down"** from the location where **docker-compose.yml** is present.

**Docker Commands**

| **Docker Command** | **Description** |
| --- | --- |
| "docker build . -t eazybytes/accounts" | To generate a docker image based on a Dockerfile |
| "docker run -p 8081:8080 eazybytes/accounts" | To start a docker container based on a given image |
| "docker images" | To list all the docker images present in the Docker server |
| "docker image inspect image-id" | To display detailed image information for a given image id |
| "docker image rm image-id" | To remove one or more images for a given image ids |
| "docker image push docker.io/eazybytes/accounts" | To push an image or a repository to a registry |
| "docker image pull docker.io/eazybytes/accounts" | To pull an image or a repository from a registry |
| "docker ps" | To show all running containers |
| "docker ps -a" | To show all containers including running and stopped |
| "docker container start container-id" | To start one or more stopped containers |
| "docker container pause container-id" | To pause all processes within one or more containers |
| "docker container unpause container-id" | To unpause all processes within one or more containers |
| "docker container stop container-id" | To stop one or more running containers |
| "docker container kill container-id" | To kill one or more running containers instantly |
| "docker container restart container-id" | To restart one or more containers |
| "docker container inspect container-id" | To inspect all the details for a given container id |
| "docker container logs container-id" | To fetch the logs of a given container id |
| "docker container logs -f container-id" | To follow log output of a given container id |
| "docker container rm container-id" | To remove one or more containers based on container ids |
| "docker container prune" | To remove all stopped containers |
| "docker compose up" | To create and start containers based on given docker compose file |
| "docker compose stop" | To stop services |