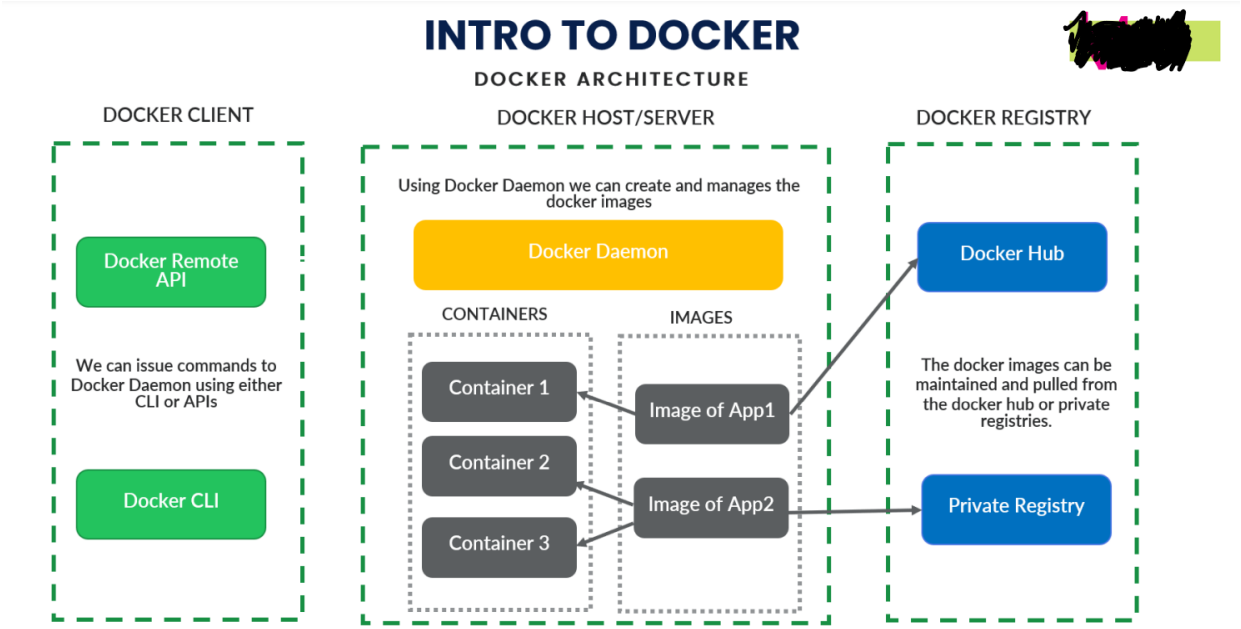
H] Docker:

H1] Docker Architecture:



1. Docker Client:

Docker client uses commands and REST APIs to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

2. Docker Server:

Using docker server we can build or manage images of our application. Here, Image contains **all the business logic that I have written and all the dependencies that are required to run that application**. And from that image we can create our containers, they are for example an image is like our java class and container is an object to that class.

From a single image we can create many containers as per our requirement

Also, once a docker image is been created we can then deploy that same image from our local in to dev/test/prod etc. environment

Without docker image we would have to manually install all the dependencies(lib's) to run that application

3. Docker Registery:

It's like the github repository where we share our code to different users, docker registery contains of **docker hub** where we can store our docker images as public copy and any person knowing that docker image name can pull that image and start running that image using docker command.

There is also an option of docker **private registery** where we have to pay some money to docker so as our docker image becomes private copy and not an public copy and would be accessed by only authorised persons.

H2] Docker Installation:

Windows installation url:

https://docs.docker.com/desktop/install/windows-install/

and after that it will automatically install docker in your windows 10 system and will restart windows10 system

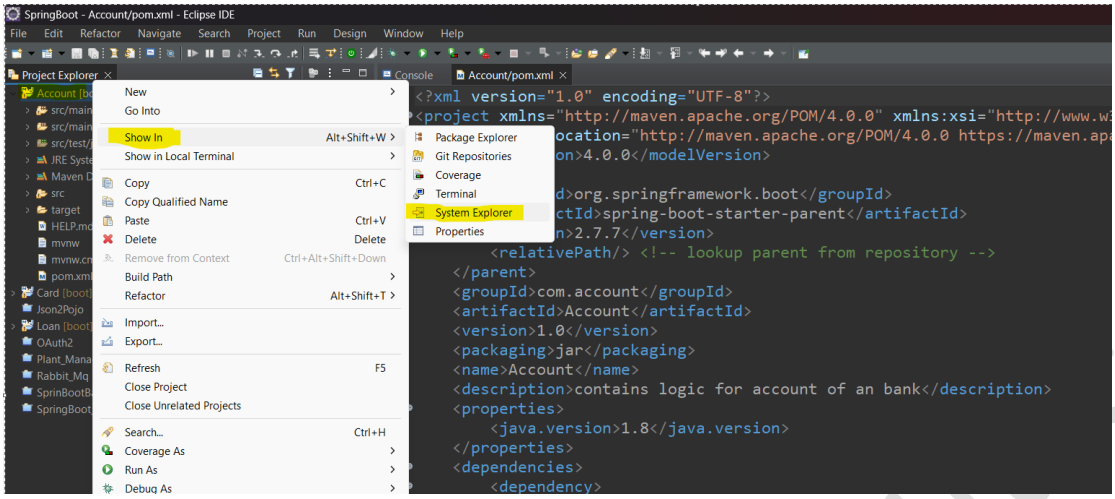
H3] Creating/Executing a Docker Image and Container:

H3.1] Create a jar/war file of your microservice as below:

S1] Inside pom.xml file in packaging tag type **jar/war** as highlited in below image

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.7</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.account</groupId>
    <artifactId>Account</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <name>Account</name>
    <description>contains logic for account of an bank</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
```

S2] Now go to the project directory where your project folder is located as beow:

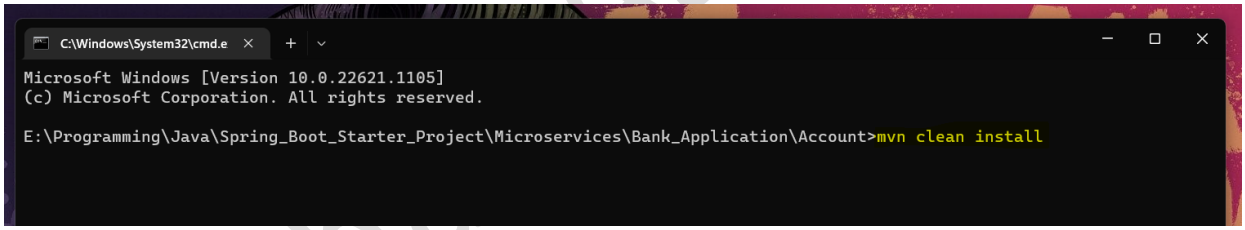


S3] Now open that folder path inside command prompt and run below command in it:

Command: **1]** **mvn clean install** **[This command will create a jar file]**

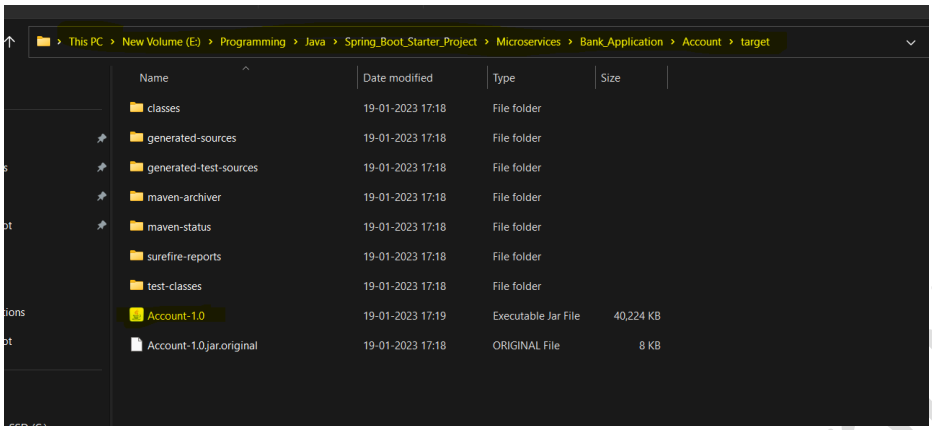
2] **mvn clean package** **[This commans will create executable jar file]**

Generally preffer to create the executable jar file using second command!!!



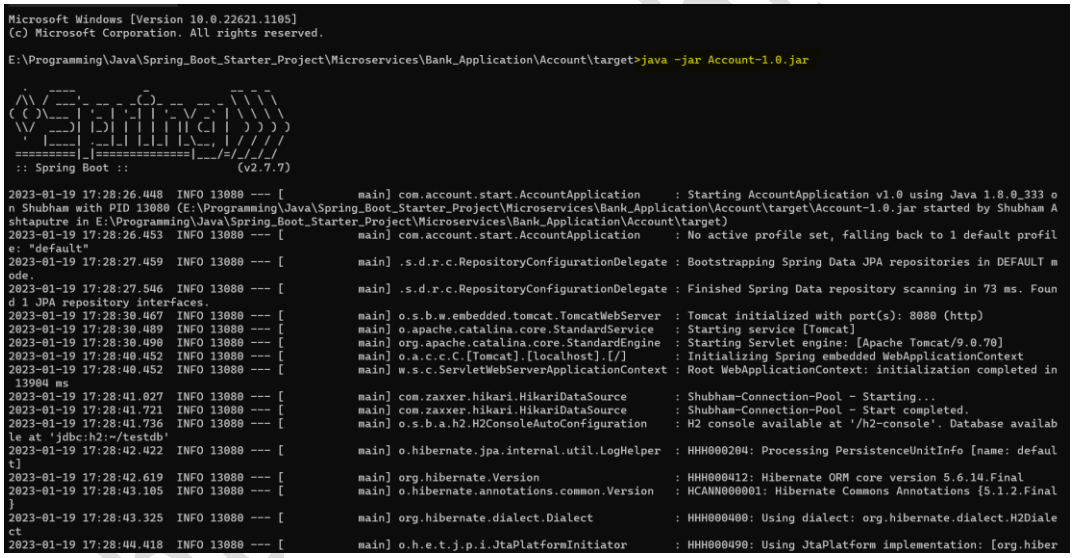
This command will create the jar file of our project and indicate once it completed building as below:

S4] Now you will get your deployed jar file in the target folder of your project as below:



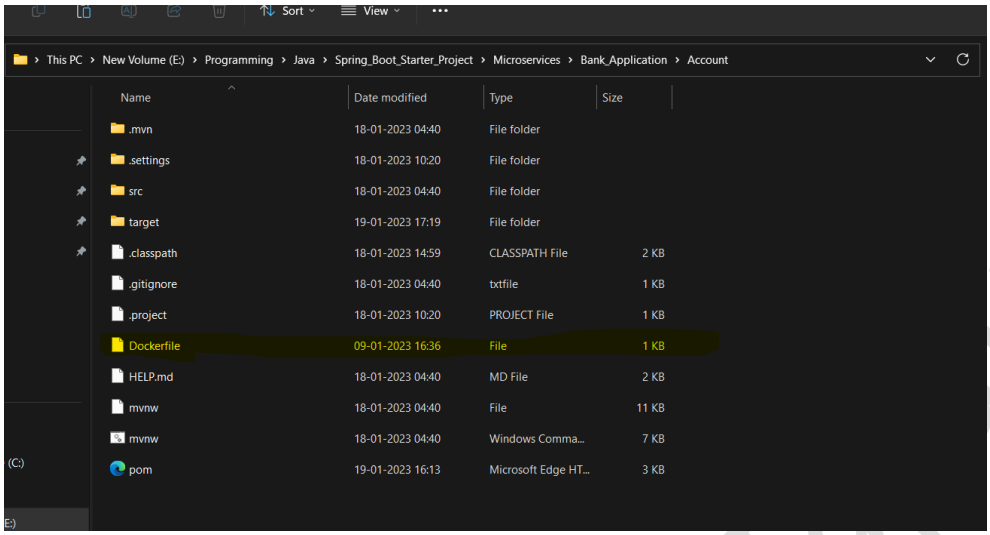
S5] Now you can check whether your jar file is running or not using below command:

java -jar Account-1.0.jar

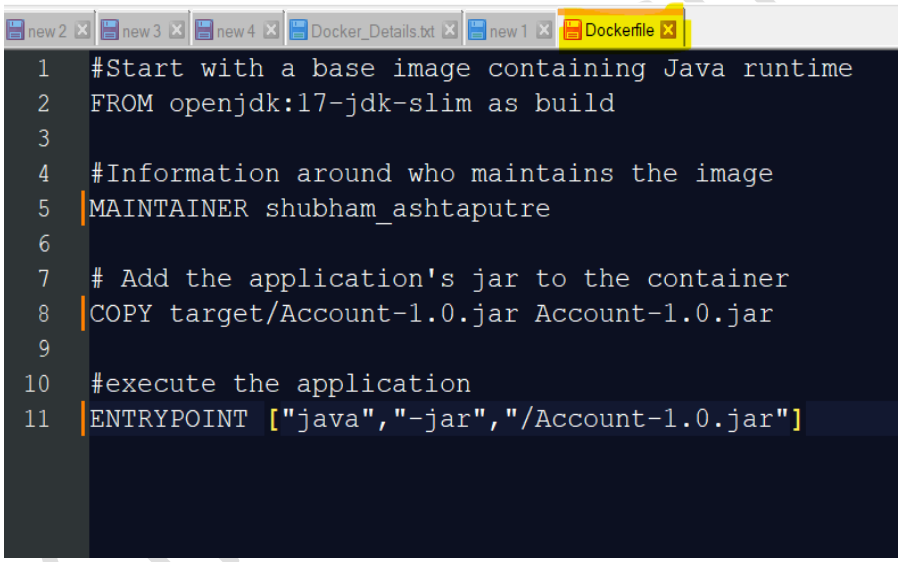


S6] press **ctrl+c** to close that running jar

H3.2] Create a docker file in your project folder:



Define the flowing lines in docker file as below:



FROM openjdk:17-jdk-slim as build

Here the '**FROM**' command indicates what is the base image on which I want to build my application, here in this case I want to build my application on open-jdk-17 but it is not installed on my local system so docker will download it from its own image repository and install it on my system

COPY target/Account-1.0.jar Account-1.0.jar

Here '**COPY**' command tells docker to copy the jar file from the defined location in out local system and then paste it into the docker filesystem

ENTRYPOINT ["java","-jar","/Account-1.0.jar"]

Here, the command entry point tells docker that whenever I want to start my container then start it with the **["java","-jar","/Account-1.0.jar"]** command

- `FROM` - Specifies the base image that the Dockerfile will use to build a new image. For this post, we are using phusion/baseimage as our base image because it is a minimal Ubuntu-based image modified for Docker friendliness.
- `MAINTAINER` - Specifies the Dockerfile Author Name and his/her email.
- `RUN` - Runs any UNIX command to build the image.
- `ENV` - Sets the environment variables. For this post, `JAVA_HOME` is the variable that is set.
- `CMD` - Provides the facility to run commands at the start of container. This can be overridden upon executing the `docker run` command.
- `ADD` - This instruction copies the new files, directories into the Docker container file system at specified destination.
- `EXPOSE` - This instruction exposes specified port to the host machine.

H3.3] Building a docker image:

Before building a docker image let's check is there any docker image present inside our project directory as below:

Cmd: docker images

```
C:\Windows\System32\cmd.e

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>
```

As you can see there is no docker image present inside our project directory

Now, run the below command to create the docker image based on your docker file configuration:

Cmd: docker build . -t bank/account

Here, '**.**'(**dot**) Indicates the docker file present inside my same project folder location, '**-t**' indicated the tag name that I want to give to my image i.e. '**bank/account**' because docker will by default give **some random number for the image/container** so just to given our own image/container name according to project requirment we use '**-t**' command

After image build is completed it shows as below:

```
C:\Windows\System32\cmd.e

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker build . -t bank/account
[+] Building 71.0s (7/7) FINISHED
 => [internal] load build definition from Dockerfile                                                    0.0s
 => => transferring dockerfile: 358B                                                                    0.0s
 => [internal] load .dockerignore                                                                       0.0s
 => => transferring context: 2B                                                                         0.0s
 => [internal] load metadata for docker.io/library/openjdk:11-jdk-slim                                  1.9s
 => [internal] load build context                                                                       2.0s
 => => transferring context: 41.20MB                                                                    2.0s
 => [1/2] FROM docker.io/library/openjdk:11-jdk-slim@sha256:868a4f2151d38ba6a09870cec584346a5edc8e9b71fde275eb2e  68.5s
 => => resolve docker.io/library/openjdk:11-jdk-slim@sha256:868a4f2151d38ba6a09870cec584346a5edc8e9b71fde275eb2e0  0.0s
 => => sha256:868a4f2151d38ba6a09870cec584346a5edc8e9b71fde275eb2e0625273e2fd8 549B / 549B                0.0s
 => => sha256:d2b6af2093e823ba0cdee4bcd45a905afe3fa054d08bde55b1d850515da69a08 1.16kB / 1.16kB            0.0s
 => => sha256:8e687a82603fe2a40c7c83684fc5b97b75dd4d51ce6b0e6250ef3fd64b7b9f66 5.59kB / 5.59kB            0.0s
 => => sha256:1efc276f4ff952c055dea726cfc96ec6a4fdb8b62d9eed816bd2b788f2860ad7 31.37MB / 31.37MB         10.7s
 => => sha256:a2f2f93da48276873890ac821b3c991d53a7e864791aaf82c39b7863c908b93b 1.58MB / 1.58MB            1.5s
 => => sha256:12cca292b13cb58fadde25af113ddc4ac3b0c5e39ab3f1290a6ba62ec8237afd 212B / 212B                0.6s
 => => sha256:69e15dccd787ba2cfe67f6abf5970ed88a5e019efbb499e499da3ab20b85fcc7 202.34MB / 202.34MB       66.3s
 => => extracting sha256:1efc276f4ff952c055dea726cfc96ec6a4fdb8b62d9eed816bd2b788f2860ad7                0.8s
 => => extracting sha256:a2f2f93da48276873890ac821b3c991d53a7e864791aaf82c39b7863c908b93b                0.1s
 => => extracting sha256:12cca292b13cb58fadde25af113ddc4ac3b0c5e39ab3f1290a6ba62ec8237afd                0.0s
 => => extracting sha256:69e15dccd787ba2cfe67f6abf5970ed88a5e019efbb499e499da3ab20b85fcc7                2.0s
 => [2/2] COPY target/Account-1.0.jar Account-1.0.jar                                                   0.3s
 => exporting to image                                                                                  0.2s
 => => exporting layers                                                                                 0.2s
 => => writing image sha256:fee66f03795903b2b2de196dedf2578e8c422cc0cf956a118f230f9594af8251            0.0s
 => => naming to docker.io/bank/account                                                                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

To verify run the below command:

```
C:\Windows\System32\cmd.e

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker images
REPOSITORY     TAG       IMAGE ID       CREATED              SIZE
bank/account   latest    fee66f037959   About a minute ago   465MB

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>
```

To inspect/ get details about the image run the below command:

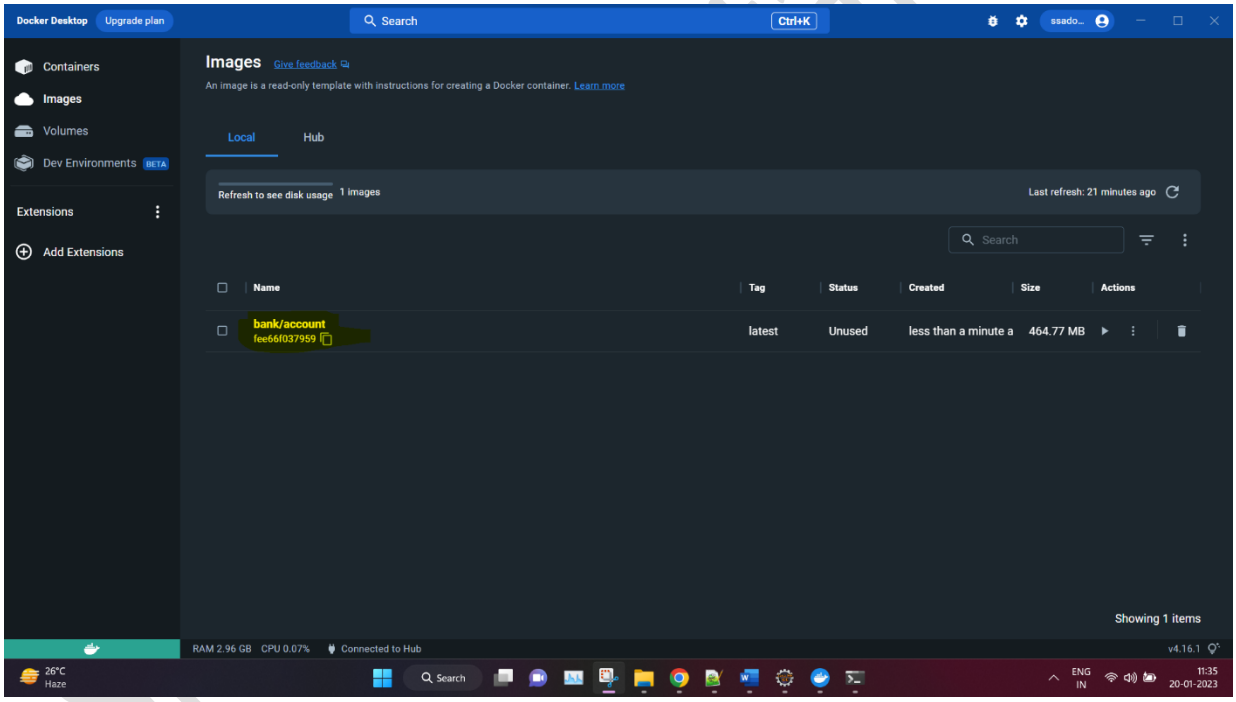
Cmd: docker image inspect <Image Id>

```
Display detailed information on one or more images

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker image inspect fee
```

Here you don't need to give full image Id name as docker will figure it out

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker image inspect fee
[
    {
        "Id": "sha256:fee66f03795903b2b2de196ded2578e8c422cc0cf956a118f230f9594af8251",
        "RepoTags": [
            "bank/account:latest"
        ],
        "RepoDigests": [],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2023-01-20T05:58:46.365888271Z",
        "Container": "",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": null,
            "Cmd": null,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "DockerVersion": "",
        "Author": "shubham_ashtaputre",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
```

Now inside your docker desktop application also you can see your created image:



H3.4] Start and Deploy microservice application on docker:

By default, the port on which the server will run is 8080 in docker if you want to change the port number then do changes in projects **application.yml** file as below:

```yaml
#setting configuration for H2 database
spring:
   datasource:
      url: jdbc:h2:~/testdb;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
      driverClassName: org.h2.Driver
      username: ssa
      password: ssa
      hikari:
         poolName: Shubham-Connection-Pool
   jpa:
      database-platform: org.hibernate.dialect.H2Dialect
      defer-datasource-initialization: true
      hibernate.ddl-auto: update
   h2:
      console:
         enabled: true
         path: /h2-console
         settings:
          web-allow-others: true
server:
  port: 8080
```

Run the blow command:

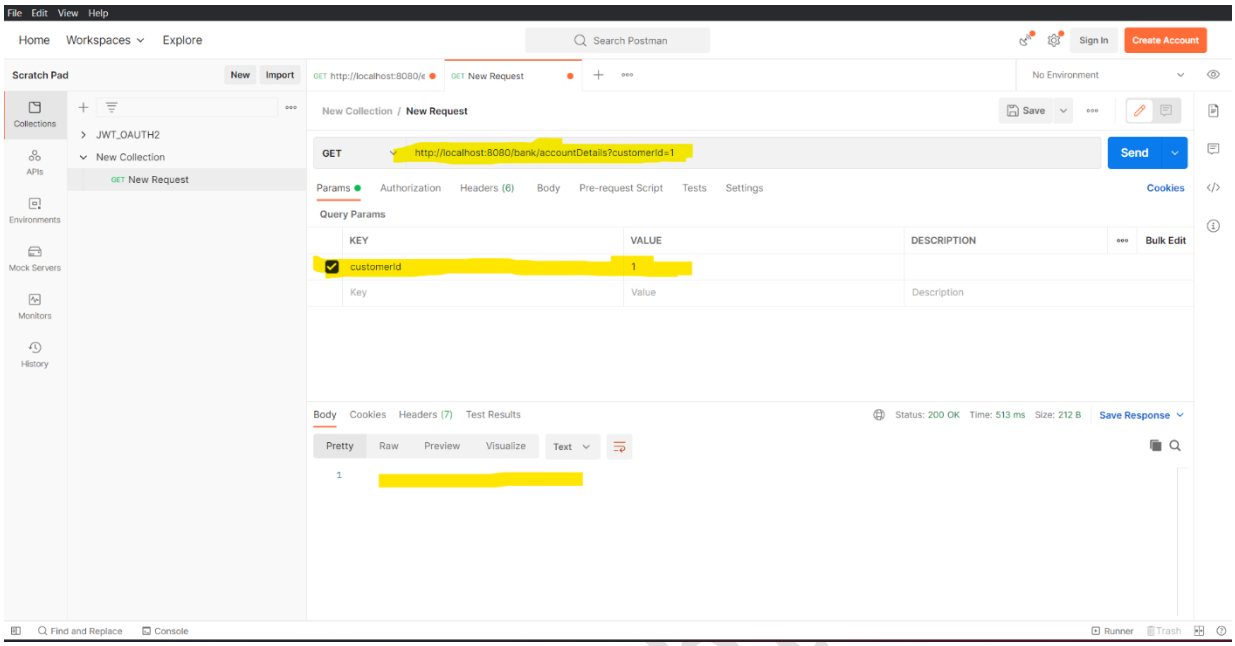
docker run -e DATABASE_SERVER=h2:tcp://localhost/~/testdb -dp 8080:8080 bank/account

here, **'-d'** means detach command it will run your container application in background rather than showing you the spring startup console and **'-p'** means port number on which the application will be hosted in the docker container and mapped to your local system port number as: **<your systems port number>:<docker container port number>**

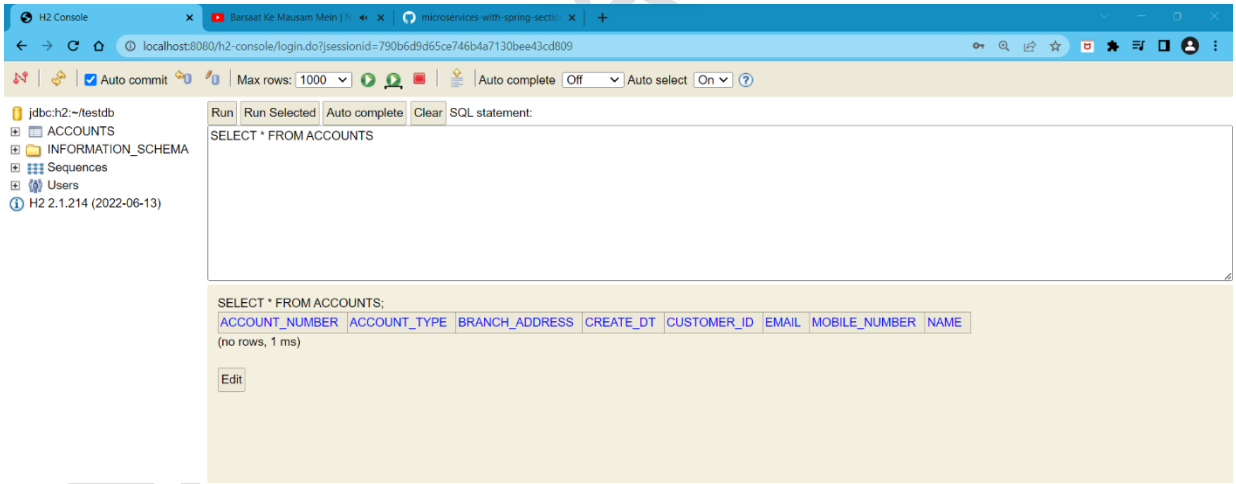
```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker run -e DATABASE_SERVER=h2:
tcp://localhost/~/testdb -dp 8080:8080 bank/account
1ebbe122d9af17460331bdddca7831ff16474b184d60d238304a9dec2568196a

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>
```

This command will create container for your docker image and start the h2 database server

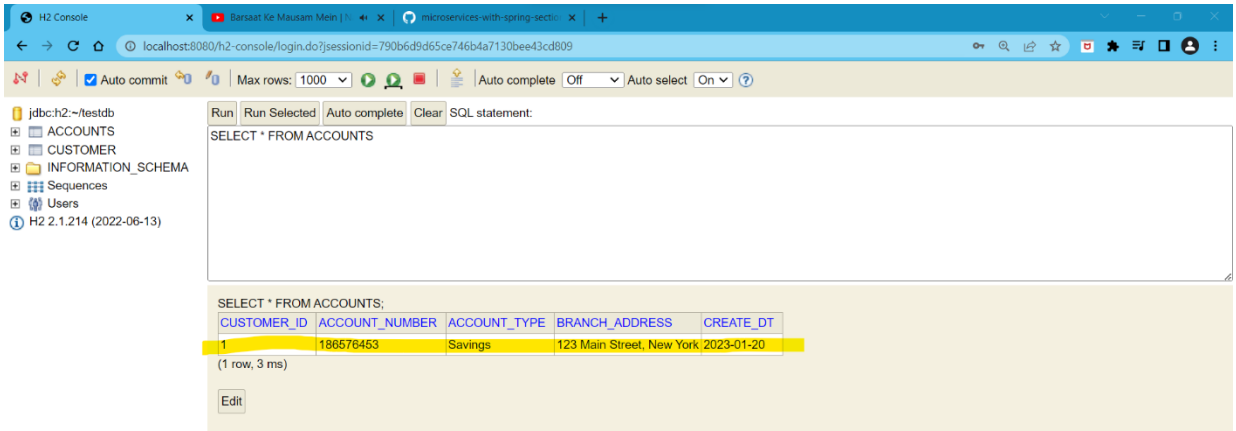
Now test your docker deployed API in postman:

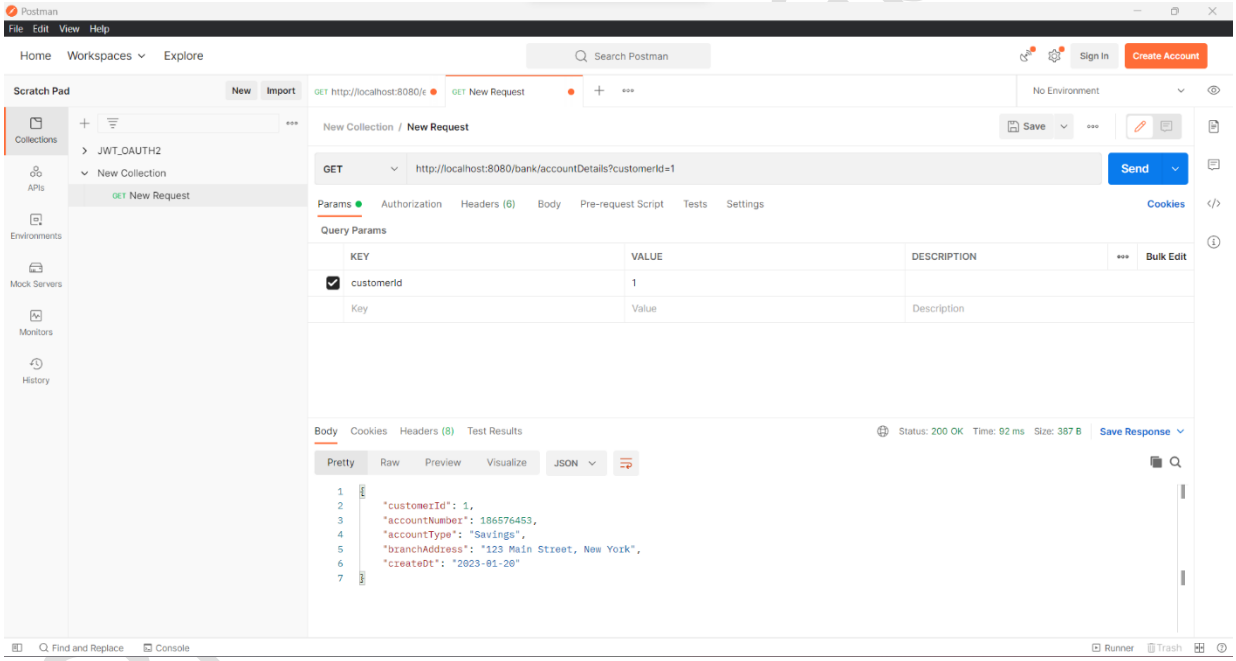


as seen even everything was fine, then also your API didn't give you any json response. The reason it happend was because in your docker virtual machine the H2 DB with Accounts table was created but with empty data



Now insert data into your H2 container for your Docker container application and now again check in POSTMAN application





H4] Docker Commands:

H4.1] Get the list of deployed docker containers:

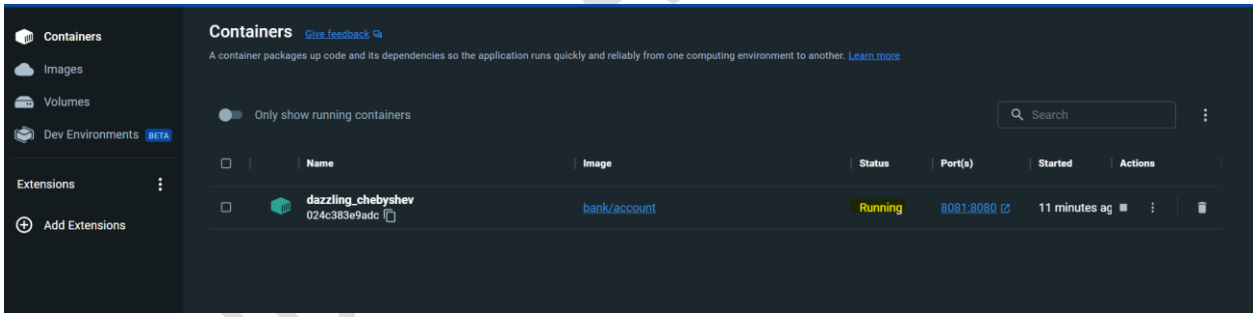
Cmd: 1] **docker container ls** [**see the list of running docker containers**]

2] **docker ps** [**see the list of running docker containers**]

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker run -dp 8081:8080 bank/account
024c383e9adccfae5ceb51f0c2dc27cbd329d5812ff4e19fa30dc56fca602ea8

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                    NAMES
024c383e9adc   bank/account   "java -jar /Account-…"   8 minutes ago   Up 8 minutes   0.0.0.0:8081->8080/tcp   dazzling_chebyshev

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>
```

H4.2] Close the deployed running microservice application container on docker:

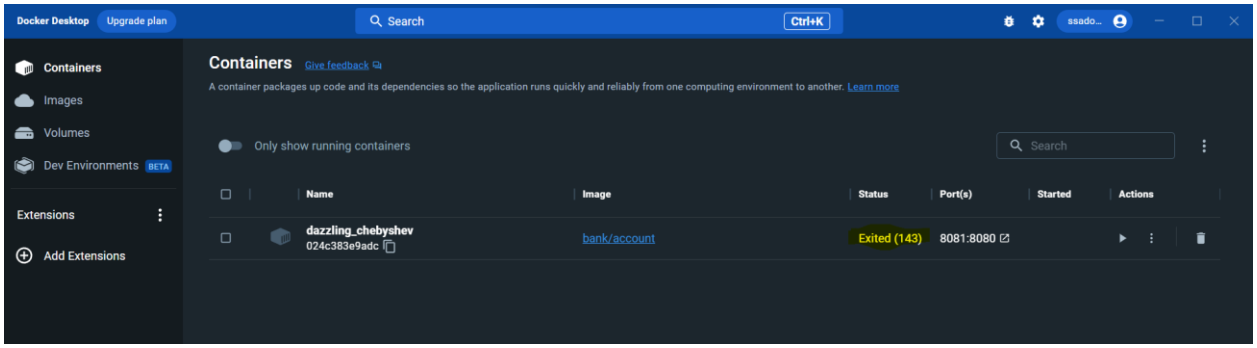
As we can see the docker container is still running:



Now we will close the running container using command as below:

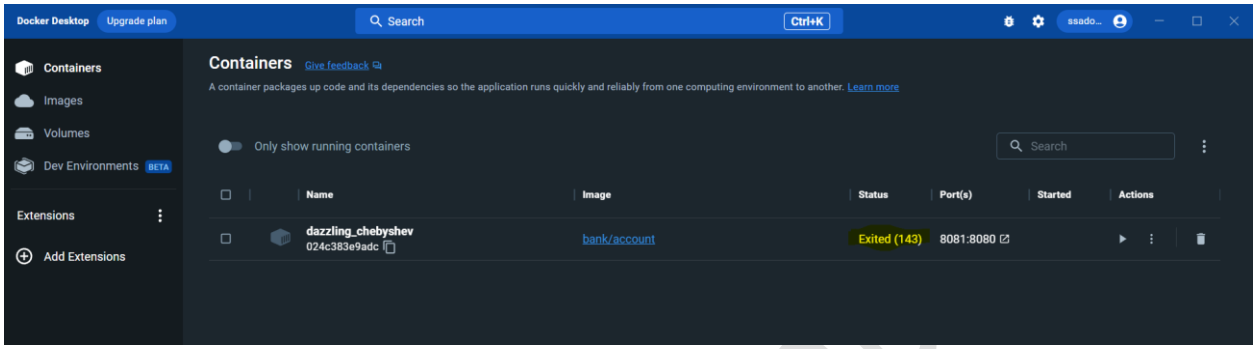
Cmd: **docker stop <Container-Id>**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker stop 024c383e9adc
024c383e9adc
```



H4.3] How to again start the existing docker container:

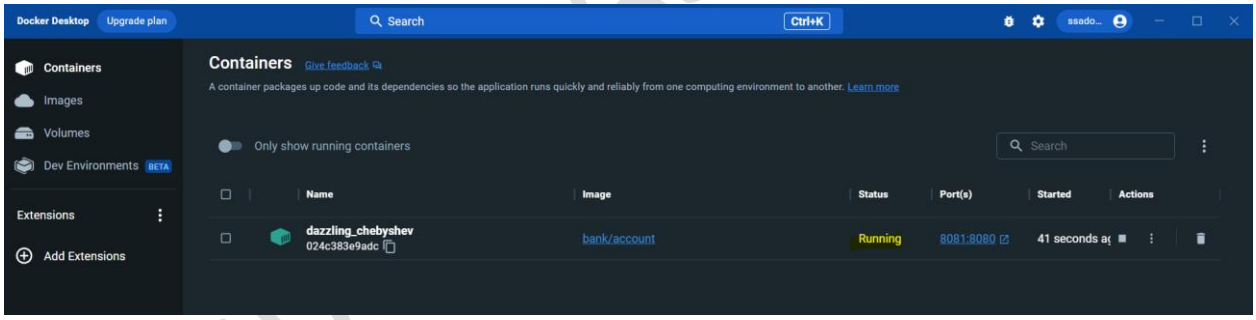
As we can see the existing docker container is now stopped



Now we will start the closed container using command as below:

Cmd: **docker start <Container-Id>**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker start 024c383e9adc
024c383e9adc
```



H4.4] How to get docker container logs:

1] Get static logs of the container:

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS         PORTS                    NAMES
024c383e9adc   bank/account   "java -jar /Account-…"   43 minutes ago   Up 4 seconds   0.0.0.0:8081->8080/tcp   dazzling_chebyshev

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker log 024c
docker: 'log' is not a docker command.
See 'docker --help'

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker logs 024c
```

Cmd: **docker logs <container id>**

2] Get dynamic logs of the container:

Here whenever there will be certain operations on our container or in container application the log values will start to change dynamically

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker logs -f 024c

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.7)

2023-02-27 11:28:34.865  INFO 1 --- [           main] com.account.start.AccountApplication     : Starting AccountApplication v1.0 using Java 1.8.0_212 on 02
4c383e9adc with PID 1 (/Account-1.0.jar started by root in /)
2023-02-27 11:28:34.869  INFO 1 --- [           main] com.account.start.AccountApplication     : No active profile set, falling back to 1 default profile: "
default"
```

Cmd: **docker logs -f <container id>**

-f : Here, ‘ **-f** ’ means follow the container, so log changes dynamically as per the operation on the container

H4.5] Docker get list of all container whether they are stopped or running:

Cmd: **docker ps -a**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS                      PORTS     NAMES
024c383e9adc   bank/account   "java -jar /Account-…"   45 hours ago   Exited (143) 23 minutes ago            dazzling_chebyshev
```

H4.6] Docker get latest container created details:

Cmd: **docker ps -a -l**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps -l
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS                      PORTS     NAMES
024c383e9adc   bank/account   "java -jar /Account-…"   45 hours ago   Exited (143) 24 minutes ago            dazzling_chebyshev
```

H4.7] Docker display container size:

Cmd: **docker ps -a -s**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps -a -s
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS                      PORTS     NAMES                SIZE
024c383e9adc   bank/account   "java -jar /Account-…"   45 hours ago   Exited (143) 24 minutes ago            dazzling_chebyshev   34.9kB (virtual 148MB)
```

H4.8] Docker display only container Id’s:

Cmd: **docker ps -a -q**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps -a -q
024c383e9adc
```

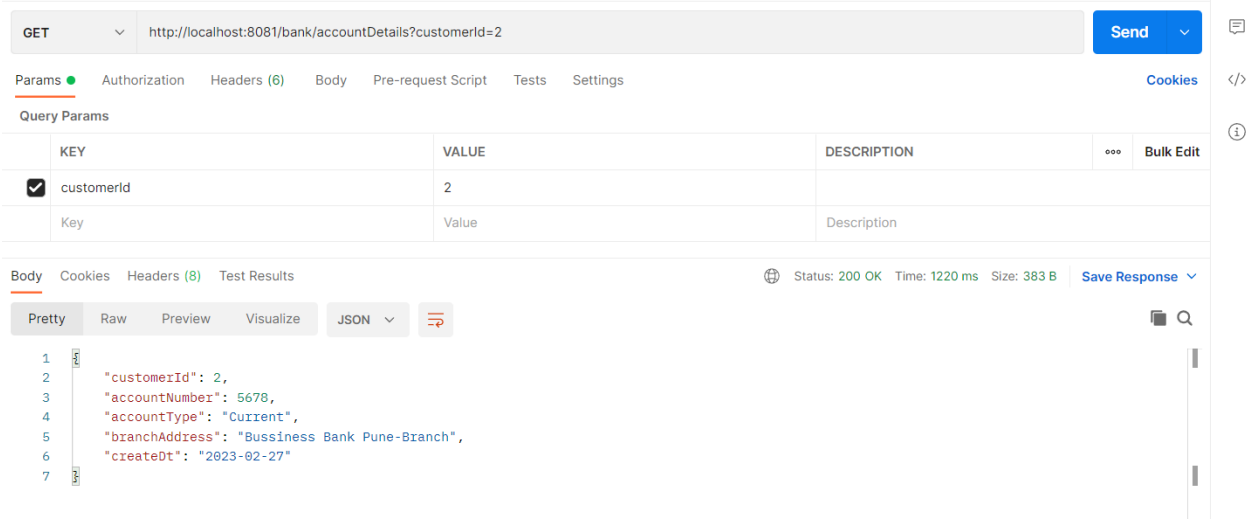
H4.9] Docker start multiple containers at single time:

Cmd: **docker start <container-id-1> <container-id-2>**

```
C:\Users\madan>docker start fc ee
fc
ee

C:\Users\madan>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS         PORTS                    NAMES
[redacted]     [redacted]/accounts    "java -jar /accounts…"   50 minutes ago  Up 5 seconds   0.0.0.0:8081->8080/tcp   elated_shtern
[redacted]     [redacted]/accounts    "java -jar /accounts…"   52 minutes ago  Up 5 seconds   0.0.0.0:8080->8080/tcp   optimistic_ptolem
y
```

H4.10] How to stop docker container to accept any request without stopping docker container?

Here, my docker container is accepting and sending me back response to my request

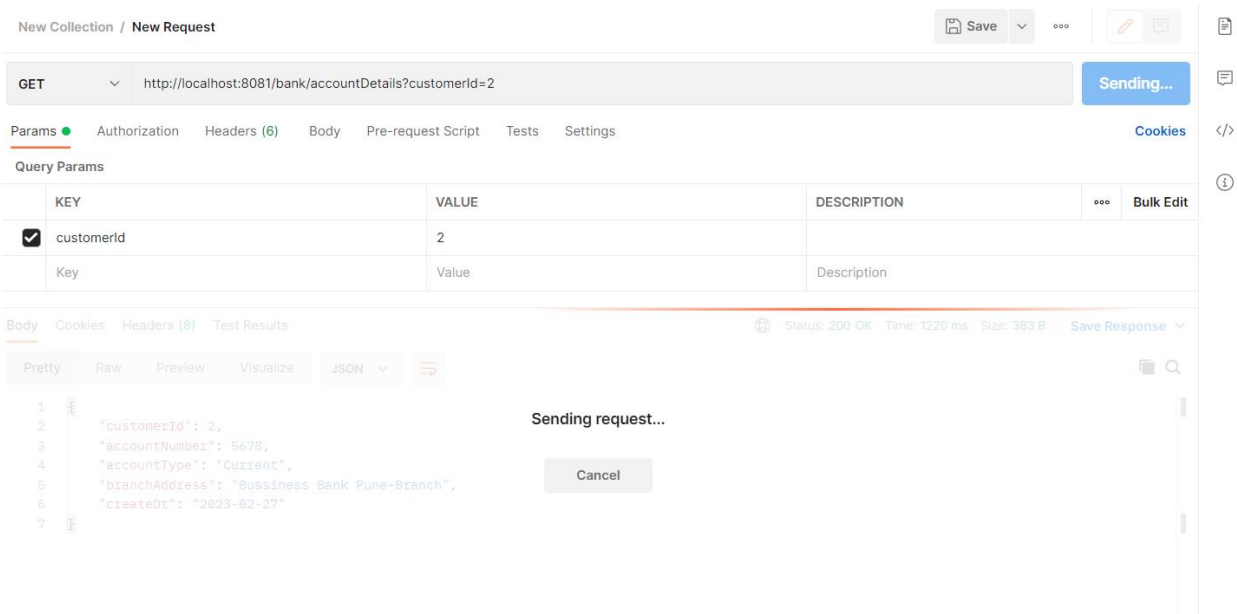


But, now if I want to pause all request's contained in this container use below command:

Cmd: **docker pause <container-id-1>**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker pause 024c383e9adc
024c383e9adc

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS                 PORTS                    NAMES
024c383e9adc   bank/account   "java -jar /Account-…"   45 hours ago   Up 3 minutes (Paused)  0.0.0.0:8081->8080/tcp   dazzling_chebyshev
```

It is not accepting the request



Now to make it work again use the below command:

Cmd: **docker unpause <container-id-1>**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker unpause 024c383e9adc
024c383e9adc

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS         PORTS                    NAMES
024c383e9adc   bank/account   "java -jar /Account-…"   45 hours ago   Up 6 minutes   0.0.0.0:8081->8080/tcp   dazzling_chebyshev
```

H4.11] To get all details of your container use below command:

Cmd: **docker container inspect <container-id-1>**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker container inspect 024c383e9adc
[
    {
        "Id": "024c383e9adccfae5ceb51f0c2dc27cbd329d5812ff4e19fa30dc56fca602ea8",
        "Created": "2023-02-27T11:28:32.568126981Z",
        "Path": "java",
        "Args": [
            "-jar",
            "/Account-1.0.jar"
        ],
        "State": {
            "Status": "running"
```

H4.12] Difference between docker kill and stop?

Docker kill <container-id> will instantly kill your running docker image without properly shutting it down

H4.13] How to check our system resource consumption made by all running docker containers?

Cmd: **docker stats**

```
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT   MEM %     NET I/O         BLOCK I/O   PIDS
024c383e9adc   dazzling_chebyshev  0.39%     402MiB / 3.519GiB   11.16%    3.4kB / 2.3kB   0B / 0B     52
```

Then press **ctrl+c** to exit from it

H4.14] how to delete/remove the existing docker container?

Cmd: **docker rm <container-id>**

H4.15] how to delete a docker image?

Cmd: **docker rmi <image-id>**

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Loan>docker rmi f0eb
Untagged: loan:0.0.1-SNAPSHOT
Deleted: sha256:f0ebbee30aacf65efe22e4ef4d41bce9c2b05e22cc905609639b360b9d307dbe
Deleted: sha256:db063c4a2dbc83c2efd7c187fab62b6e620765d40aed766d1424537e3be704f0
Deleted: sha256:ea5f1761d11137c5bacf48b898901f74400a4bb7e81aeddcfe9f3a902b185be2
Deleted: sha256:4aa6a47a69319e8829c1df8fb614783e630b699ae62ce1b29de14c84db22014f
Deleted: sha256:e33fad7398966a23a47da03b1eeb8ac0ab1b3ff8ffa72d8df10fa4c376d804a2
Deleted: sha256:5af8bd774f32f0b7bd0bf348ccaf5b1d2ced80ad4137419d654eb34c9534b463
Deleted: sha256:b5ff3f9ae64550bd608a4c1d543160f2a305632986df1098a20e0a4c55105953
Deleted: sha256:fa6dde9a03b87c8a6aa9af53e3db1a26e2b2881b137cbebbb0b8ea7dff045c56
Deleted: sha256:b36d706c446bd44a2fe596a87a56b274abc116068183a6554ca867b9c92e027b
Deleted: sha256:b61be94a55b07611e57bbabcca340c84a871e3fadb7b3aa842c9cf24dcf644c2
Deleted: sha256:7f58c475e16af4e72d87e9eb619a16bda6805aa5629d46509b428c2994ad9f31
Deleted: sha256:87df318b4be1a38ecc6b281e5a5d7c0e12de5aba256da8cb1a5ed699dafb455f
Deleted: sha256:9182f1f0eef652235026f230c65fcfa7b8bd053024b2fdfde037847707bed14e
Deleted: sha256:dd5aae16c9fdf2ad6427ddd0e995ad7e79826536e700d9e1b970ddfec8892d1d
Deleted: sha256:549011ee534e62329fea04f6f6aab35d41d9927667591502fde34a7f3f8eb355
Deleted: sha256:8b7ac3edd291e93dd678297435ac9091b7991c63eb527b86914990dddbbd0b39
Deleted: sha256:378c12170fb24435e86672f603fb8fee7a1aab7c82f8898c847267f6a849aac9
Deleted: sha256:474207cc06adec3790ab791e89b6df8ec8d1dc33a4c33e467bd30000bc8e701e
```

H5] Docker Buildpacks:

H5.1] What are BuildPacks?

H5.1.1] A Buildpack is a program that turns source code into a runnable container image without precisely defining the steps, as in the case of Dockerfiles. Instead, it detects the language and converts the source code into a runnable container image. There are buildpacks for Ruby, Go, Node. js, Java, Python, and more.

H5.1.2] In short the steps that we have performed in **"H3] Creating/Executing a Docker Image and Container"** to manually create docker image is been automated using the Buildpacks

H5.1.3] And this Buildpack is a concept like microservices

H5.2] Which buildpack do springboot internally uses?

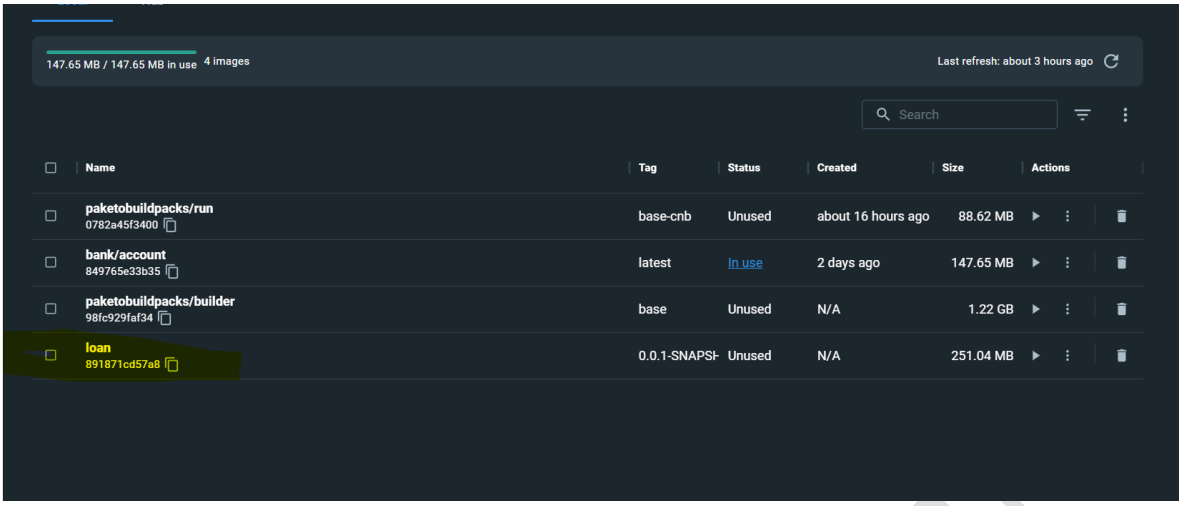
It uses **PACKETO-BUILDPACK**

H5.3] Command to create docker image using buildpack:

Cmd: **mvn spring-boot:build-image**

```
[INFO]     [creator]     Adding cache layer 'buildpacksio/lifecycle:cache.sbom'
[INFO]
[INFO] Successfully built image 'docker.io/library/loan:0.0.1-SNAPSHOT'
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:11 min
[INFO] Finished at: 2023-03-01T16:29:31+05:30
[INFO] ------------------------------------------------------------------------

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Loan>
```

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Loan>docker images
REPOSITORY                TAG              IMAGE ID       CREATED        SIZE
paketobuildpacks/run      base-cnb         0782a45f3400   16 hours ago   88.6MB
bank/account              latest           849765e33b35   2 days ago     148MB
loan                      0.0.1-SNAPSHOT   891871cd57a8   43 years ago   251MB
paketobuildpacks/builder  base             98fc929faf34   43 years ago   1.22GB

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Loan>
```



H6] Pushing Docker images from our Local repository to Docker hub repository

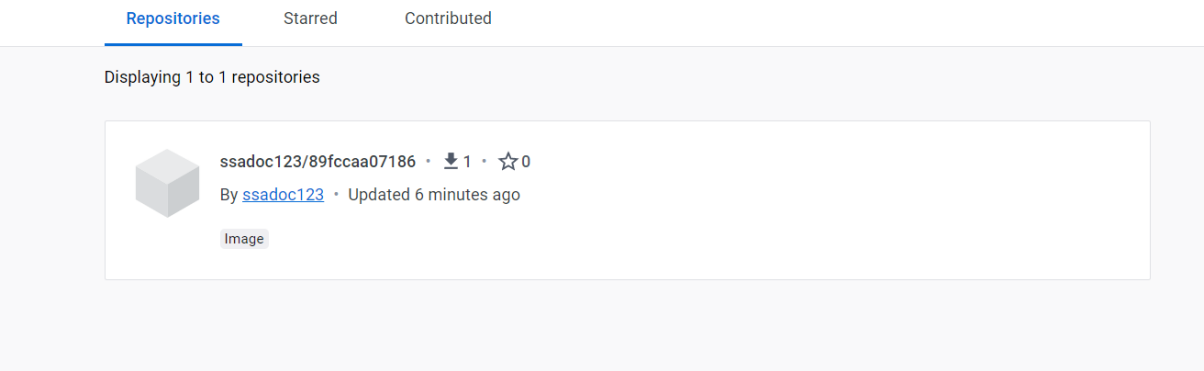
Cmd:

1] docker login docker.io

2] docker tag <image-id> **YOUR_DOCKERHUB_NAME**/<image-id>

3] docker push **YOUR_DOCKERHUB_NAME**/<image-id>

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application>docker tag 89fccaa07186 ssadoc123/89fccaa07186

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application>docker push ssadoc123/89fccaa07186
Using default tag: latest
The push refers to repository [docker.io/ssadoc123/89fccaa07186]
1dc94a70dbaa: Pushed
c2069f7acfbe: Pushed
1e7663fc17ad: Pushed
5f70bf18a086: Pushed
f80d7dca8c61: Pushed
a258d3332031: Pushed
e1b15112baf2: Pushed
fa8f59e21cf5: Pushed
6763b99e3792: Pushed
30a03cb7ff3f: Pushed
bdffd30dc8fd: Pushed
7fbc97c38fad: Pushed
eb80257f15f5: Pushed
ec0381c8f321: Pushed
b22e542bc827: Pushed
0b0f6513f974: Pushed
b39f850bc48b: Pushed
060d8f75a7a2: Pushed
748d218824a2: Pushed
a32f9b55358f: Pushed
de4d4f8ab58b: Pushed
a9d782af4b6e: Pushed
52c5ca3e9f3b: Pushed
latest: digest: sha256:730b49eba128e349439ec334b6bf967ae06a18ebc1de7993dd2f9e95849f7fc1 size: 5325
```



H7] Docker Compose

H7.1] Check if docker compose exists or not:

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application>docker-compose version
Docker Compose version v2.15.1

E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application>docker-compose --version
Docker Compose version v2.15.1
```

H7.2] Check for docker version

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application>docker --version
Docker version 20.10.22, build 3a2c30b
```

Compose and Docker compatibility matrix

There are several versions of the Compose file format – 1, 2, 2.x, and 3.x. The table below is a quick look. For full details on what each version includes and how to upgrade, see About versions and upgrading.

This table shows which Compose file versions support specific Docker releases.

Compose file format	Docker Engine release
Compose specification	19.03.0+
3.8	19.03.0+
3.7	18.06.0+

H7.3] Supported docker file name:

docker-compose.yml,

docker-compose.yaml,

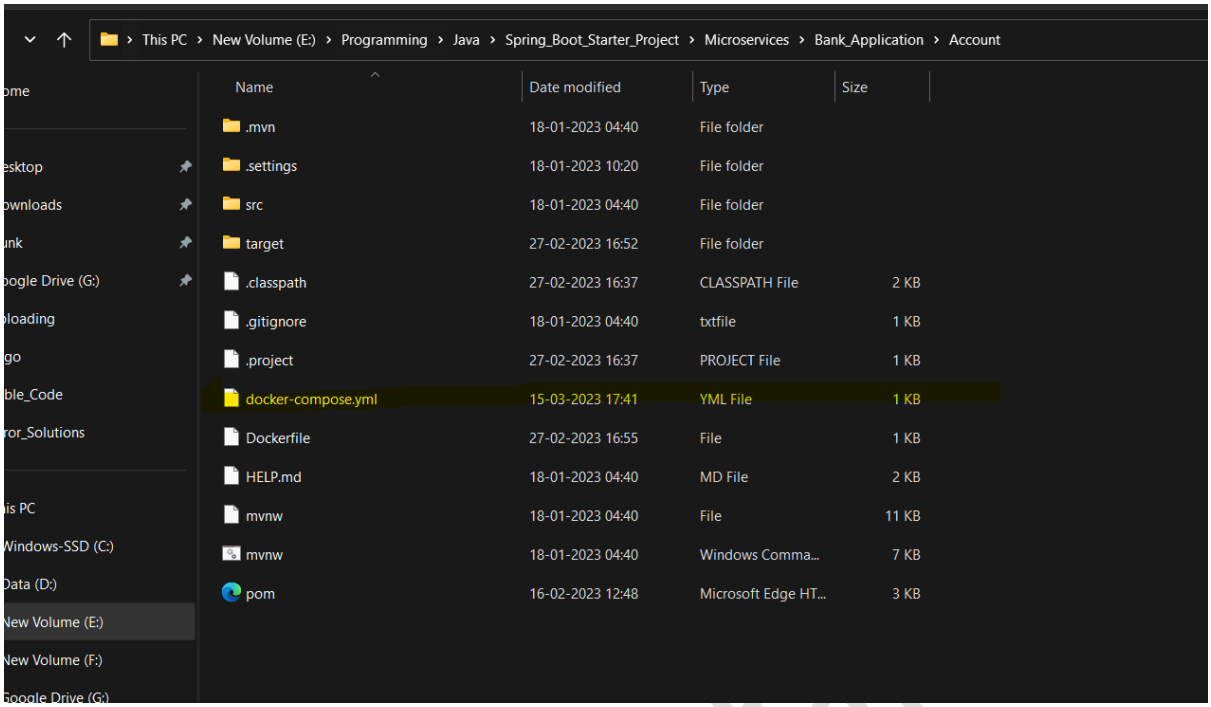
compose.yml,

compose.yaml

H7.4] Create a docker compose file as **docker-compose.yml**

```yaml
version: "3.8"

services:

    accounts:
        image: ssadoc123/849765e33b35
        mem_limit: 700m
        ports:
          - "8083:8083"
        networks:
          - ssa-network

    loans:
        image: ssadoc123/89fccaa07186
        mem_limit: 700m
        ports:
          - "8084:8084"
        networks:
          - ssa-network

    cards:
        image: ssadoc123/f39d079b81fb
        mem_limit: 700m
        ports:
          - "8085:8085"
        networks:
          - ssa-network

networks:
    ssa-network: {}
```

H7.5] Place that compose file in any on project folder as below:



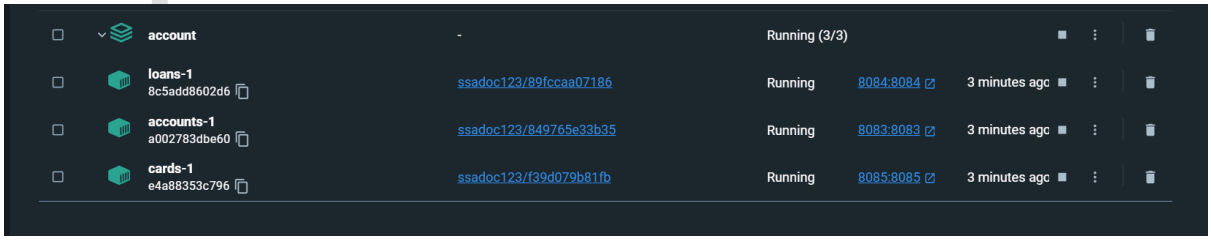
H7.6] Run the below command:

Cmd:

Docker-compose up

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker-compose up
[+] Running 3/3
 - cards Pulled                                                                                    3.8s
 - accounts Pulled                                                                                 3.8s
 - loans Pulled                                                                                    3.8s
[+] Running 4/4
 - Network account_ssa-network   Created                                                           0.1s
 - Container account-cards-1     Created                                                           0.1s
 - Container account-accounts-1  Created                                                           0.1s
 - Container account-loans-1     Created                                                           0.1s
Attaching to account-accounts-1, account-cards-1, account-loans-1
account-loans-1     | Setting Active Processor Count to 16
account-cards-1     | Setting Active Processor Count to 16
account-loans-1     | Calculated JVM Memory Configuration: -XX:MaxDirectMemorySize=10M -Xmx94170K -XX:MaxMetaspaceSize=110629K -XX:ReservedCodeCacheSize=240
M -Xss1M (Total Memory: 700M, Thread Count: 250, Loaded Class Count: 17118, Headroom: 0%)
account-loans-1     | Enabling Java Native Memory Tracking
account-loans-1     | Adding 124 container CA certificates to JVM truststore
account-loans-1     | Spring Cloud Bindings Enabled
account-cards-1     | Calculated JVM Memory Configuration: -XX:MaxDirectMemorySize=10M -Xmx94170K -XX:MaxMetaspaceSize=110629K -XX:ReservedCodeCacheSize=240
M -Xss1M (Total Memory: 700M, Thread Count: 250, Loaded Class Count: 17118, Headroom: 0%)
account-loans-1     | Picked up JAVA_TOOL_OPTIONS: -Djava.security.properties=/layers/paketo-buildpacks_bellsoft-liberica/java-security-properties/java-secu
rity.properties -XX:+ExitOnOutOfMemoryError -XX:ActiveProcessorCount=16 -XX:MaxDirectMemorySize=10M -Xmx94170K -XX:MaxMetaspaceSize=110629K -XX:ReservedCode
CacheSize=240M -Xss1M -XX:+UnlockDiagnosticVMOptions -XX:NativeMemoryTracking=summary -XX:+PrintNMTStatistics -Dorg.springframework.cloud.bindings.boot.enab
le=true
```

Your containers got created and running as below:



H7.7] Stop a docker container:

Cmd:

docker-compose stop

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker-compose stop
[+] Running 3/3
 - Container account-accounts-1  Stopped                                                                    2.6s
 - Container account-cards-1     Stopped                                                                    2.5s
 - Container account-loans-1     Stopped                                                                    2.4s
```

H7.8] Start a docker container:

Cmd:

docker-compose start

```
E:\Programming\Java\Spring_Boot_Starter_Project\Microservices\Bank_Application\Account>docker-compose start
[+] Running 3/3
 - Container account-loans-1     Started                                                                    1.8s
 - Container account-cards-1     Started                                                                    1.9s
 - Container account-accounts-1  Started                                                                    1.9s
```