Dockerization **07-03-2025**

Dockerization is the process of packaging an application and its dependencies into a **Docker container** to ensure that it runs consistently across diffe environments. This helps in eliminating issues related to environment compatibility and dependency management.

Key Concepts of Dockerization

1. **Docker Container** – A lightweight, standalone, and executable package that includes everything needed to run an application (code, runtime, libraries, and dependencies).
2. **Docker Image** – A blueprint for creating containers. It contains the application code and environment configuration.
3. **Dockerfile** – A script containing a set of instructions to create a Docker image.
4. **Docker Compose** – A tool for defining and running multi-container applications.
5. **Docker Hub** – A cloud-based repository to sto d share Docker images.

```
C:\Users\miniMiracle>docker images
REPOSITORY                TAG                        IMAGE ID       CREATED         SIZE
springboot-docker-demo    0.1.RELEASE                e264a36fd66c   21 hours ago    778MB
openzipkin/zipkin         latest                     d9316e7ff757   2 weeks ago     377MB
rabbitmq                  3.13.7-management-alpine   d759525efd68   5 months ago    279MB
```

Spring boot demo is exist in the local machine….we need to push into the docker hub(remotely)

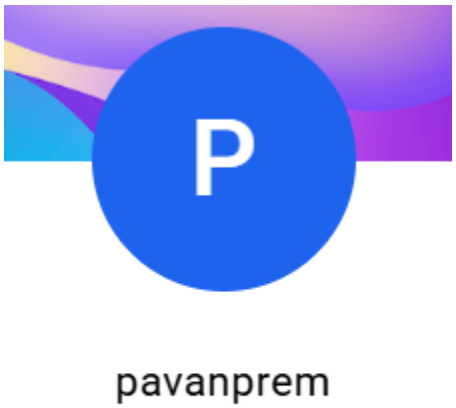
Now we need to login..before that our docker desktop should open ..

```
C:\Users\miniMiracle>docker login
Authenticating with existing credentials...
```

```
C:\Users\miniMiracle>docker images
REPOSITORY                TAG                        IMAGE ID       CREATED         SIZE
springboot-docker-demo    0.1.RELEASE                e264a36fd66c   22 hours ago    778MB
openzipkin/zipkin         latest                     d9316e7ff757   2 weeks ago     377MB
rabbitmq                  3.13.7-management-alpine   d759525efd68   5 months ago    279MB

C:\Users\miniMiracle>
```

```
how to push to docker image to docker hub:

docker push sailurams/springboot-docker-demo:0.1.RELEASE
```

replace sailurams with the dockerhub id



above one is my docker id

```
C:\Users\miniMiracle>docker tag springboot-docker-demo:0.1.RELEASE pavanprem/springboot-docker-demo:0.1.RELEASE
```

```
C:\Users\miniMiracle>docker images
REPOSITORY                          TAG                        IMAGE ID       CREATED        SIZE
springboot-docker-demo              0.1.RELEASE                e264a36fd66c   22 hours ago   778MB
pavanprem/springboot-docker-demo    0.1.RELEASE                e264a36fd66c   22 hours ago   778MB
openzipkin/zipkin                   latest                     d9316e7ff757   2 weeks ago    377MB
rabbitmq                            3.13.7-management-alpine   d759525efd68   5 months ago   279MB
```

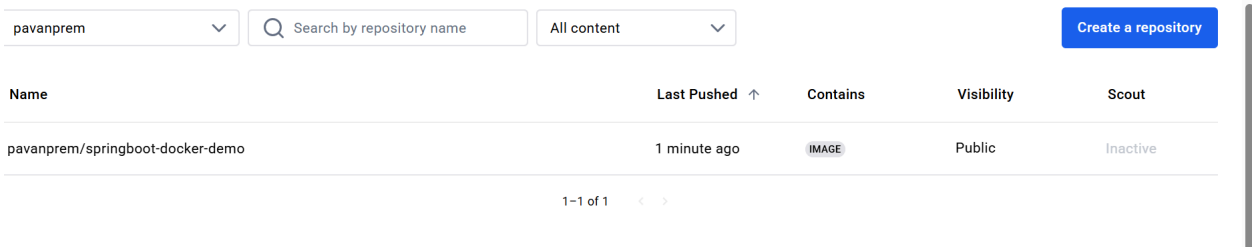
Now local image is associated with the docker id

Now we need to push it into the docker hub

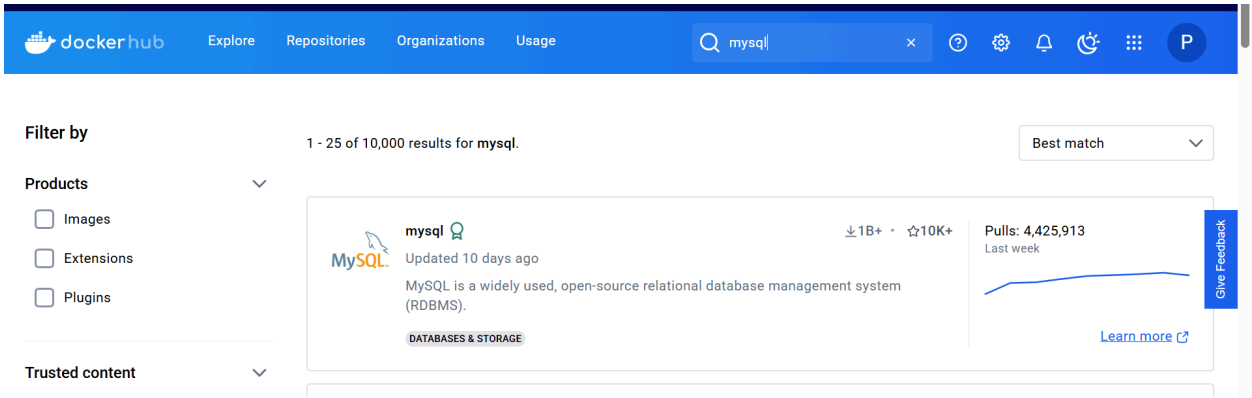
```
C:\Users\miniMiracle>docker push pavanprem/springboot-docker-demo:0.1.RELEASE
The push refers to repository [docker.io/pavanprem/springboot-docker-demo]
74ac377868f8: Pushing [======>                                            ]  3.146MB/30.43MB
b1f4e8047df7: Pushed
e9da8df62964: Pushed
5a9e900e010d: Pushing [===============>                                   ]  5.243MB/18.68MB
a182a611d05b: Pushing [==========>                                        ]  3.146MB/16.98MB
426329e266e4: Pushing [==>                                                ]  8.389MB/201.1MB
5f4059624ea0: Pushed
```

```
C:\Users\miniMiracle>docker push pavanprem/springboot-docker-demo:0.1.RELEASE
The push refers to repository [docker.io/pavanprem/springboot-docker-demo]
74ac377868f8: Pushed
b1f4e8047df7: Pushed
e9da8df62964: Pushed
5a9e900e010d: Pushed
a182a611d05b: Pushed
426329e266e4: Pushed
5f4059624ea0: Pushed
0.1.RELEASE: digest: sha256:e264a36fd66c23a09d6993b8a8c1137b1138481aa50bd00bdb8e28468617aa11 size: 856
```

Now check in the docker hub



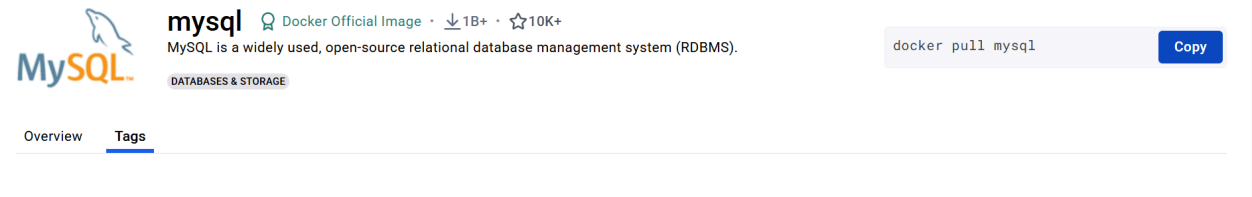
We can pull the docker image from remote to the local



Click on the tags

Copy the tag

docker pull mysql



```
C:\Users\miniMiracle>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
893b018337e2: Download complete
23d22e42ea50: Download complete
f56a22f949f9: Download complete
d255dceb9ed5: Download complete
43759093d4f6: Downloading [==============================================>  ]  47.19MB/49.09MB
2be0d473cadf: Download complete
277ab5f6ddde: Downloading [==============================================>    ]  45.09MB/48.42MB
431b106548a3: Download complete
df1ba1ac457a: Download complete
```

```
C:\Users\miniMiracle>docker pull mysql
Using default tag: latest
latest: Pulling from library/mysql
893b018337e2: Download complete
23d22e42ea50: Download complete
f56a22f949f9: Download complete
d255dceb9ed5: Download complete
43759093d4f6: Download complete
2be0d473cadf: Download complete
277ab5f6ddde: Download complete
431b106548a3: Download complete
df1ba1ac457a: Download complete
cc9646b08259: Download complete
Digest: sha256:146682692a3aa409eae7b7dc6a30f637c6cb49b6ca901c2cd160becc81127d3b
Status: Downloaded newer image for mysql:latest
docker.io/library/mysql:latest
```

this command will pull the mysql image from docker hub
to local repository.

docker pull mysql
even though if we didn't mention tag name by default it will pull
latest tag because default tag is latest only.

to see all the existing container

docker ps

once we pull mysql image lets try to run this image by giving below
command

```
C:\Users\miniMiracle>docker images
REPOSITORY                          TAG                        IMAGE ID       CREATED        SIZE
pavanprem/springboot-docker-demo    0.1.RELEASE                e264a36fd66c   22 hours ago   778MB
springboot-docker-demo              0.1.RELEASE                e264a36fd66c   22 hours ago   778MB
openzipkin/zipkin                   latest                     d9316e7ff757   2 weeks ago    377MB
mysql                               latest                     146682692a3a   6 weeks ago    1.09GB
rabbitmq                            3.13.7-management-alpine   d759525efd68   5 months ago   279MB
```

Successfully pulled

Command to run the image

```
docker run -p 3307:3306 --name localhost
-e MYSQL_ROOT_PASSWORD=root(mandatory)
-e MYSQL_DATABASE=sys
-e MYSQL_USER =rk
-e MYSQL_PASSWORD=rk
-d
mysql:latest
```

```
C:\Users\miniMiracle>docker run -p 3307:3306 --name localhost3 -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=sys -e MYSQ
L_USER=pk -e MYSQL_PASSWORD=pk -d mysql:latest
5cdfbdf03061740099b0ef093d66ede08e334870763d6ccfaa8ddd570acca126
```

Now it is running and localhost 3 is the container name

User and pswd is our wish

Docker images

```
C:\Users\miniMiracle>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS
          NAMES
5cdfbdf03061   mysql:latest   "docker-entrypoint.s…"   About a minute ago   Up About a minute   33060/tcp, 0.0.0.0:3307-
>3306/tcp   localhost3
```

Docker images are running in the docker containers

```
after above command , try to connect with mysql through
the docker container whicih we created by usng below command


docker exec -it localhost bash
```

To connect with the mysql through the docker

```
C:\Users\miniMiracle>docker exec -it localhost bash
bash-5.1# mysql -u pk -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 9.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| sys                |
+--------------------+
```

As multiple images are running in the multiple containers……for suppose if the images need to communicate with each other from diff containers..we need to have the docker network

07-03-2025

Afternoon

What is Docker Network?

A **Docker network** is a way for Docker containers to communicate with each other or with external systems. It provides isolation, security, and connectivity between containers running on the same or different hosts.

Right now our spring boot application is running on one container and our mysql is running on other container

If these two want to communicate with each other then we go for the **docker network**

By default Bridge network is a type

How to check the containers running or not

docker ps

```
C:\Users\miniMiracle>docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED       STATUS              PORTS
          NAMES
5fd1c35fa685   mysql:latest       "docker-entrypoint.s…"   3 hours ago   Up 3 hours          33060/tcp, 0.0.0.0:3307-
>3306/tcp   localhost
746c75302260   openzipkin/zipkin  "start-zipkin"           2 days ago    Up 2 days (healthy)  9410/tcp, 0.0.0.0:9411->
9411/tcp    zipkin
```

Docker images are running in the docker containers

What is a Docker Container?

A **Docker container** is a lightweight, portable, and isolated environment that runs an application along with all its dependencies. It allows applications to run consistently across different environments, whether it's a developer's laptop, a test server, or a production system

```
C:\Users\miniMiracle>docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
735e4119492b   bridge    bridge    local
23cf2a830909   host      host      local
e084de2e6290   none      null      local
```

Default one

```
C:\Users\miniMiracle>docker network create springboot-mysql-net
0b1f2e2a9f923accb57b799d515b839dbd531d97a3c1ad97c7238b2426ab6798
```

Command to create the docker network

```
C:\Users\miniMiracle>docker network ls
NETWORK ID     NAME                  DRIVER    SCOPE
735e4119492b   bridge                bridge    local
23cf2a830909   host                  host      local
e084de2e6290   none                  null      local
0b1f2e2a9f92   springboot-mysql-net   bridge    local
```

```
to run the docker mysql image in docker container using network:

docker run --name mysqldb --network springboot-mysql-net
-e MYSQL_ROOT_PASSWORD=root
-e MYSQL_DATABASE=sys
-e MYSQL_USER =rk
-e MYSQL_PASSWORD=rk
-d
mysql:latest
```

Prev we will run the docker image directly..now we r running vth the help of the network..so stop the prev docker image

```
C:\Users\miniMiracle>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
    NAMES
5cdfbdf03061   mysql:latest   "docker-entrypoint.s…"   26 minutes ago   Up 26 minutes   33060/tcp, 0.0.0.0:3307->3306/tc
p   localhost3

C:\Users\miniMiracle>docker stop 5cdf
5cdf

C:\Users\miniMiracle>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

```
C:\Users\miniMiracle>docker images
REPOSITORY                          TAG                        IMAGE ID       CREATED        SIZE
springboot-restful-webservices      0.1.RELEASE                95c34e56a74d   42 hours ago   855MB
springboot-docker-demo              0.1.RELEASE                e264a36fd66c   3 days ago     778MB
pavanprem/springboot-docker-demo    0.1.RELEASE                e264a36fd66c   3 days ago     778MB
openzipkin/zipkin                   latest                     d9316e7ff757   2 weeks ago    377MB
mysql                               latest                     146682692a3a   6 weeks ago    1.09GB
rabbitmq                            3.13.7-management-alpine   d759525efd68   5 months ago   279MB
```

Now we will run the mysql image by using the docker network

```
C:\Users\miniMiracle>docker run --name mysqldb --network springboot-mysql-net -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATAB
ASE=product_db -e MYSQL_USER=pk -e MYSQL_PASSWORD=pk -d mysql:latest
63c91b3956a18ff618414255fbf33ed19e1c54c257876e7132c57d58ce783ac9
```

```
docker exec -it [4digit container id] bash
```

```
C:\Users\miniMiracle>docker exec -it 63c9 bash
bash-5.1#  mysql -u pk -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 9.2.0 MySQL Community Server - GPL

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| performance_schema |
| product_db         |
+--------------------+
3 rows in set (0.04 sec)
```

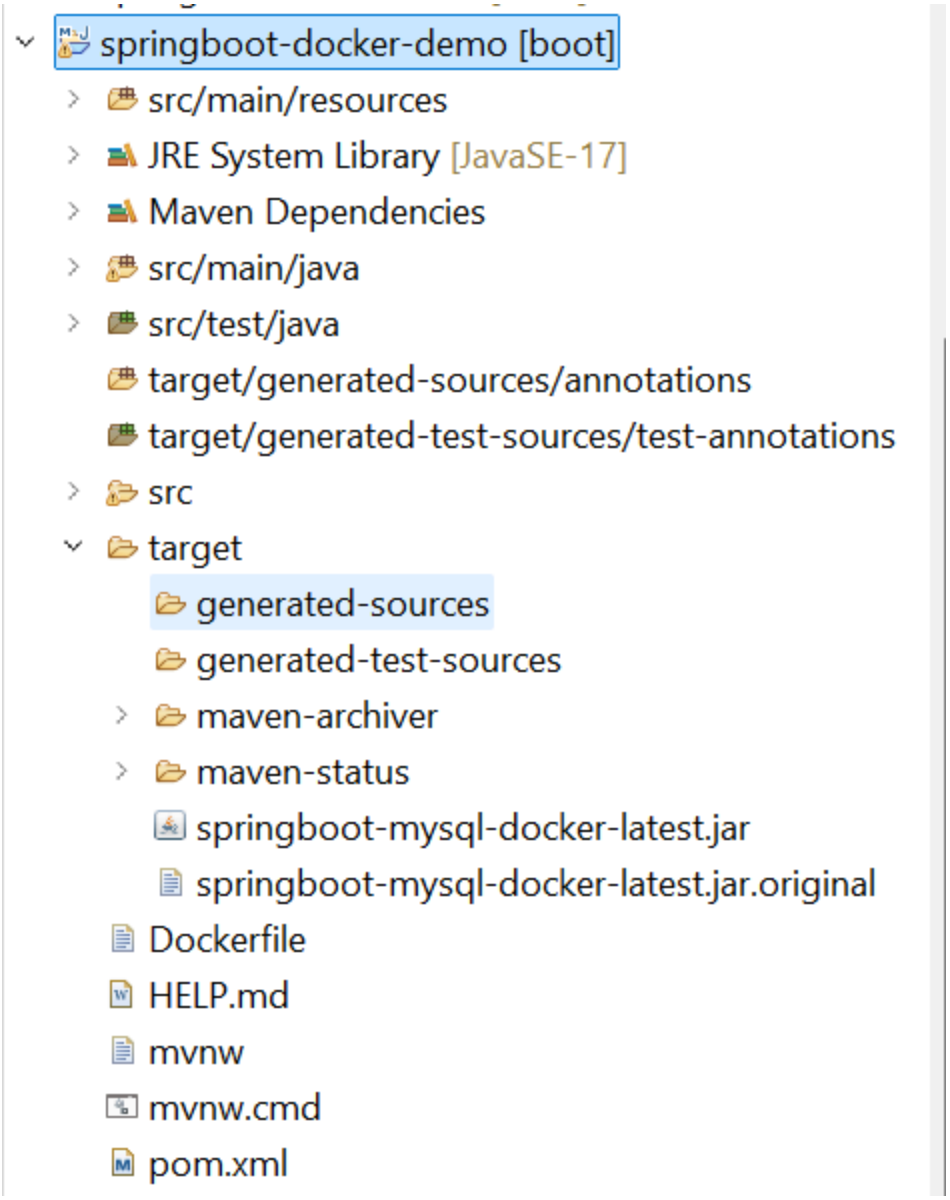
```
note:
before building your actual spring boot application give the database
properties according to your mysql image docker container credentails.
like below inside your src/main/resources
```

```
<finalName>springboot-mysql-docker-latest</finalName>
ild>
```

With this name only we r going to create the jar file

Runas >>maven build>>clean package

```
now create the image for your spring boot restfull application
(before build the image for your application build the springboot application and be
ready with jar file)

note:
while you are creating the jar file
right click -> run as -> maven build... -> clean package -> skip tests check in

and then run and apply

so then without errors jar file gets created
```



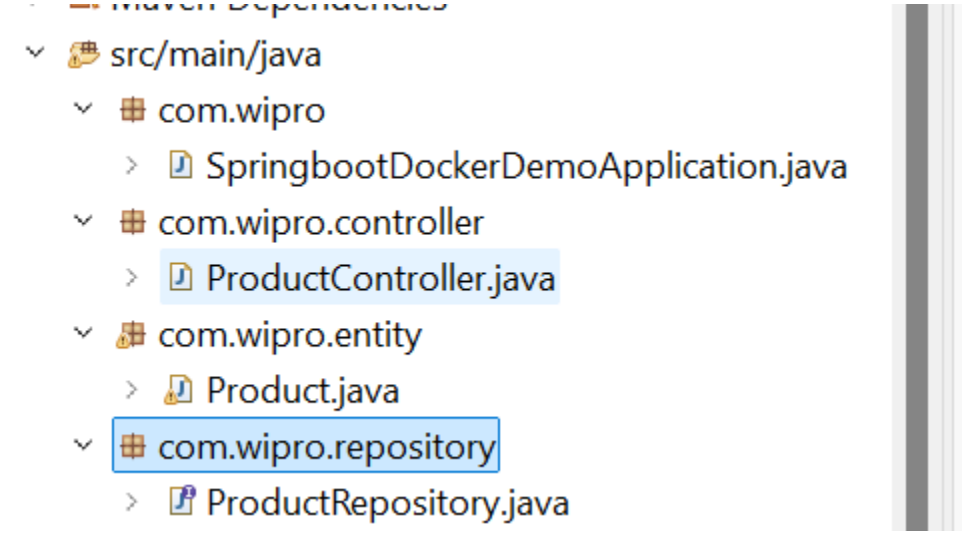
Dockerfile

```
1 FROM eclipse-temurin:19
2 LABEL maintainer="pavan@wipro.com"
3 WORKDIR /app
4 COPY target/springboot-mysql-docker-latest.jar /app/springboot-docker-demo.jar
5 ENTRYPOINT ["java","-jar","springboot-mysql.jar"]
6
```

Dockerfile | application.properties

```
#server.port=9090

spring.datasource.url=jdbc:mysql://localhost:3306/sys
spring.datasource.username=root
spring.datasource.password=PavanPrem

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.ddl-auto=create

spring.profiles.active=docker
#to read the data  from the application-docker.properties

```

```
spring.datasource.url=jdbc:mysql://mysqldb:3306/product_db
spring.datasource.username=pk
spring.datasource.password=pk
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update

```



Dockerfile | application.properties | application-docker.properties | SpringbootDockerDemoApp

```java
package com.wipro;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class SpringbootDockerDemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringbootDockerDemoApplication.class, args);
    }
}
```

```java
package com.wipro.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("/api")
public class ProductController {

    @Autowired
    private ProductRepository repository;

    @GetMapping("/display")
    public String getMessage() {
        return "Welcome to Spring Boot Docker Demo";
    }

    @PostMapping("/product/create")
    public ResponseEntity<Product> createProduct(@RequestBody Product product) {
        return new ResponseEntity<>(repository.save(product), HttpStatus.CREATED);
    }
}
```

```java
// Entity
package com.wipro.entity;

import jakarta.persistence.Entity;

@Entity
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int productId;

    private String productName;
    private Double productPrice;
}
```

Dockerfile | application.prope... | application-docke... | SpringbootDockerD... | P

```java
package com.wipro.repository;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface ProductRepository extends JpaRepository<Product, Integer> {
}
```

now go to the dockerfile existed location and give the below command

```
docker build -t springboot-restful-webservices:0.1.RELEASE .
```

Once the image gets created now we need to run the image

```
C:\Users\miniMiracle\eclipse-workspace\Docker\springboot-docker-demo\springboot-docker-demo>docker build -t springboot-restful-webservices:0.1.RELEASE .
[+] Building 8.7s (9/9) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.1s
 => => transferring dockerfile: 322B                                                                                0.0s
 => [internal] load metadata for docker.io/library/eclipse-temurin:19                                               2.6s
 => [auth] library/eclipse-temurin:pull token for registry-1.docker.io                                              0.0s
```

As 2 images are running in the same network so that the containers can communicate with each other

```
C:\Users\miniMiracle\eclipse-workspace\Docker\springboot-docker-demo\springboot-docker-demo>docker images
REPOSITORY                          TAG                       IMAGE ID       CREATED         SIZE
springboot-restful-webservices      0.1.RELEASE               f71027d27a51   2 minutes ago   855MB
springboot-docker-demo              0.1.RELEASE               e264a36fd66c   3 days ago      778MB
pavanprem/springboot-docker-demo    0.1.RELEASE               e264a36fd66c   3 days ago      778MB
openzipkin/zipkin                   latest                    d9316e7ff757   2 weeks ago     377MB
mysql                               latest                    146682692a3a   6 weeks ago     1.09GB
rabbitmq                            3.13.7-management-alpine  d759525efd68   5 months ago    279MB
```

now run the image then only both the images can communicate with each other from diff containers within the same network

```
C:\Users\miniMiracle\eclipse-workspace\Docker\springboot-docker-demo\springboot-docker-demo>docker run --network springb
oot-mysql-net --name springboot-mysql-container-new -p 8081:8080 -d springboot-restful-webservices:0.1.RELEASE
2e41e583b6f7750dfb035f4c9ce1825058aef5a31ba197df6a28181b5c41154c
```

with the above command it started to run the springboot application with mysql docker container credentials.
means table also gets created
in the mysql docker container only

now open postman
send the postman post request to insert the data

```
C:\Users\miniMiracle\eclipse-workspace\Docker\springboot-docker-demo\springboot-docker-demo>docker run --network springb
oot-mysql-net --name springboot--mysql-container-v1 -p 8081:8080 -d springboot-restful-webservices:0.1.RELEASE
5e77e56cebb43efe7f9e3f6585db96aa264c82004a0546e7d40fb3b236e5f714

C:\Users\miniMiracle\eclipse-workspace\Docker\springboot-docker-demo\springboot-docker-demo>
```

Run the image