Docker 06-03-2025

Think of **Docker** like a **shipping container** for software.

Just like a shipping container holds everything needed for a product (clothes, electronics, food) and ensures it can be transported anywhere safely, **Docker** packs everything a software application needs (code, libraries, dependencies) into a **container**.

This container can then run **anywhere**—on your laptop, a cloud server, or someone else's computer—without worrying about compatibility issues. It ensures that **"It works on my machine"** also means **"It works everywhere."** 🚢📦

Using **Docker** has several benefits, especially for developers and DevOps teams. Here are the key advantages:

🚀 1. Consistency & Portability

- Docker ensures that an application runs **the same way** on any environment—whether it's your laptop, a testing server, or the cloud.
- No more **"It works on my machine"** issues!

⚡ 2. Faster Development & Deployment

- Containers start **with seconds**, much faster than traditional Virtual Machines (VMs).
- You can quickly build, test, and deploy applications in a **lightweight** and **isolated** environment.

📦 3. Lightweight & Efficient

- Unlike VMs, Docker **shares the host OS kernel**, making it **faster and less resource-intensive**.
- Containers use **less memory and CPU**, improving system performance.

🔧 4. Dependency Management

- Docker packages all required **libraries, configurations, and dependencies** into a single container.
- No need to install dependencies manually—everything is bundled inside the container.

🔹 **Bottom Line:** Docker makes software development **faster, more efficient, and more reliable** by eliminating environment-related issues. 🚢🔥

What is dockerization

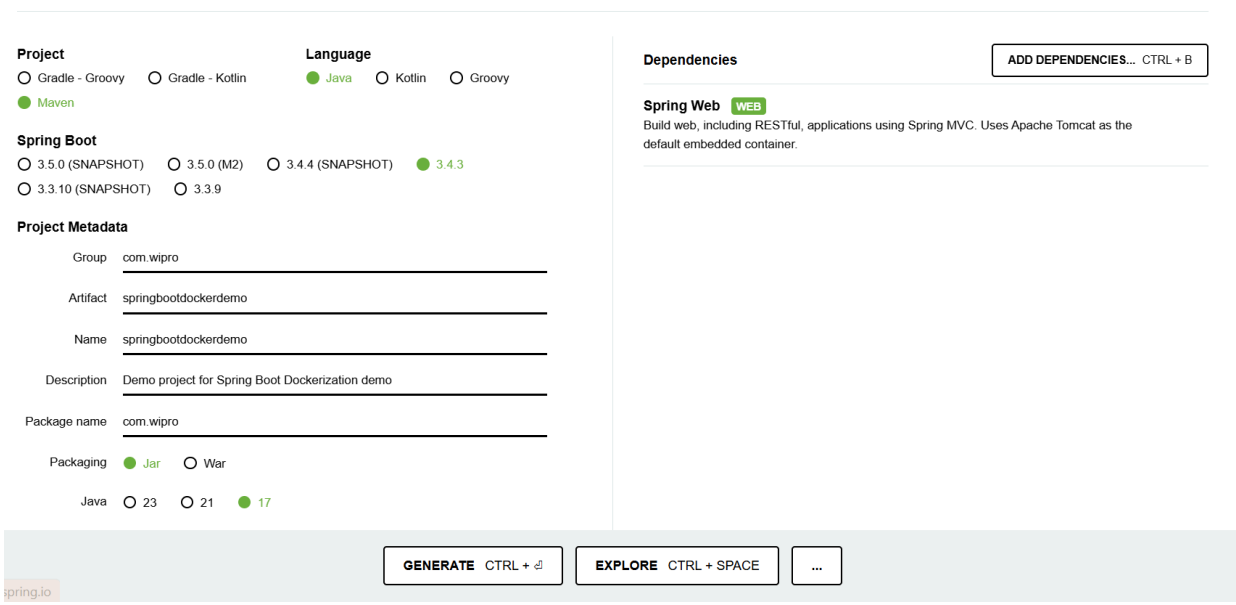
Dockerizing a Spring Boot Application 🚀 🐳

Dockerizing a Spring Boot application means packaging the application along with all its dependencies into a Docker container so that it can run anywhere without compatibility issues.

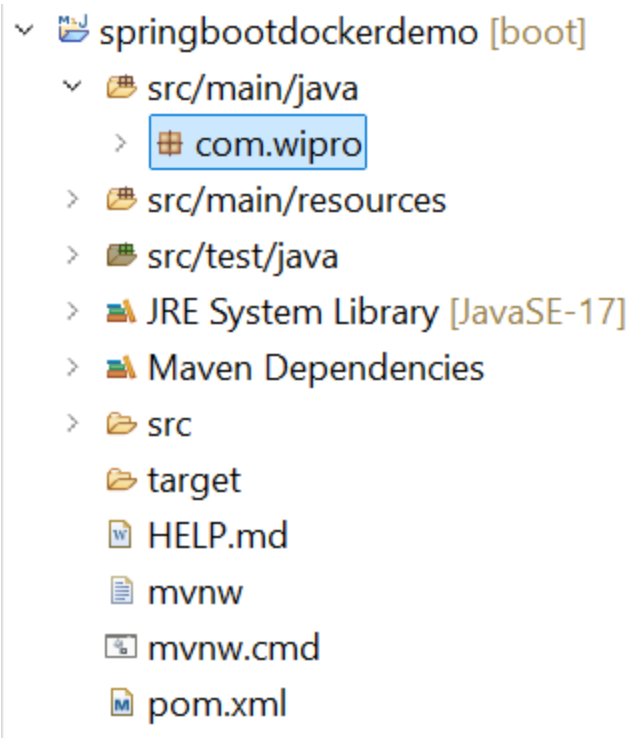
Why Dockerize Spring Boot? 🤔

- ✅ Runs **on any system** without environment conflicts
- ✅ **No need to install Java** or dependencies manually
- ✅ Easily **scalable & deployable** on cloud platforms
- ✅ Works with **Docker Compose & Kubernetes** for microservices

Now create one basic spring boot application



Generate >>extract>import

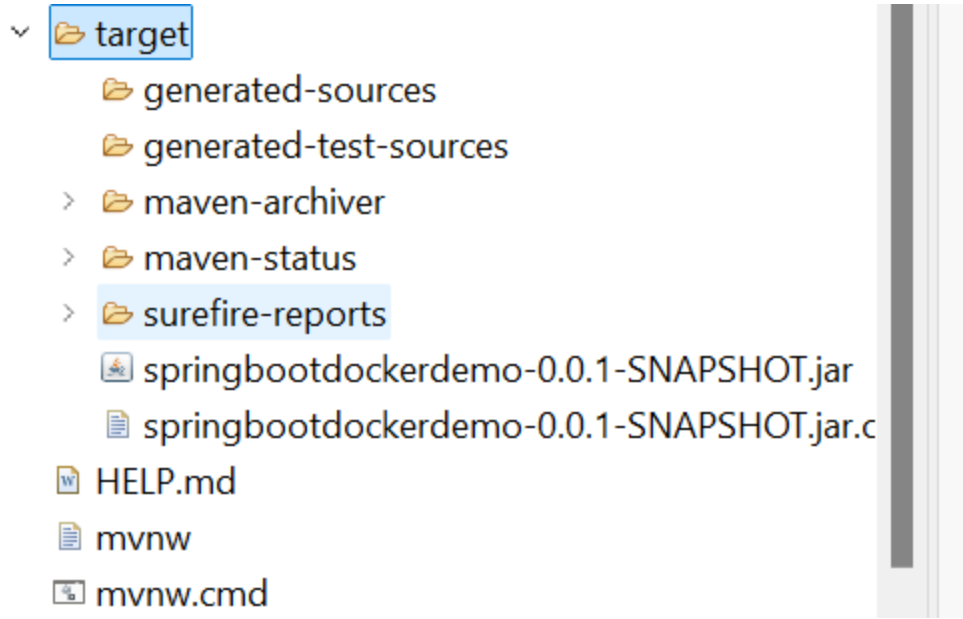


1.Now create the docker file(where it contains all the instructions to create the docker image

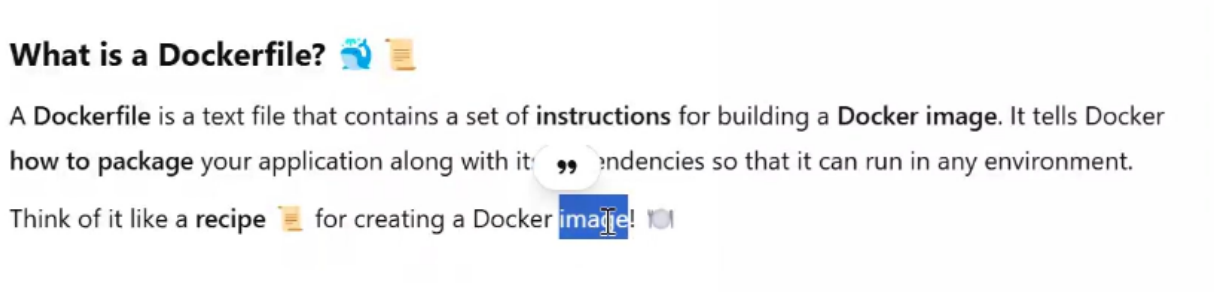
2.after importing the project we need to create one controller package and create the class

```
package com.wipro.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class DockerController {

    @GetMapping("/docker")
    public String getData()
    {
        return "welcome to dockerization";
    }
}
```

Now we need to create the jar file
Rightclick on project>>runas >maven build>clean package



Once the jar file got created we need to prepare the docker file

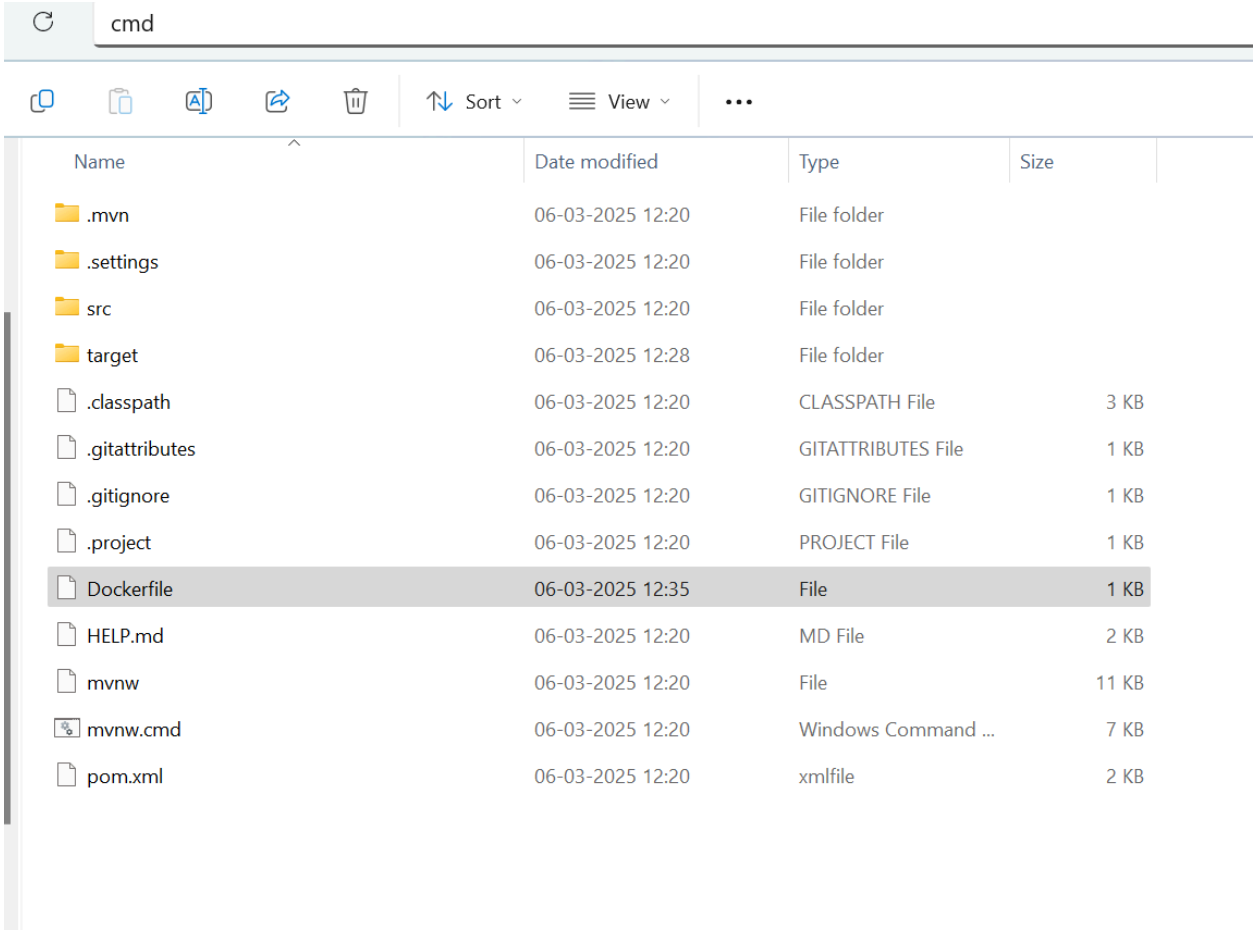


Now create the docker file
Right click on project>new>file>Dockerfile

```
FROM eclipse-temurin:19
LABEL maintainer="pavan@wipro.com"
WORKDIR /app
COPY target/springbootdockerdemo-0.0.1-SNAPSHOT.jar /app/springboot-docker-demo.jar
ENTRYPOINT ["java","-jar","springboot-docker-demo.jar"]
```

With this docker file we need to create the docker image

Go to the docker demo folder location
And open the cmd and pass one command



It opens the command prompt then run the command

```
docker build -t springboot-docker-demo:0.1.RELEASE .
```

Part	Meaning
`docker build`	Comm build a Docker image.
`-t`	Tagging option to name the image.
`springboot-docker-demo`	The **name** of the Docker image being created.
`:0.1.RELEASE`	The **version tag** of the image (you can use `latest` if you don't specify one).

. means the current directory

What Happens When You Run This Command?

1. **Docker reads the** `Dockerfile` in the current directory (`.`).
2. **It follows the instructions** inside the `Dockerfile` to create a **Docker image**.
3. **The image is tagged as** `springboot-docker-demo:0.1.RELEASE`, so you can easily refer to it later.

```
C:\Users\miniMiracle\eclipse-workspace\springbootdockerdemo\springbootdockerdemo>docker build -t springboot-docker-demo:
0.1.RELEASE .
[+] Building 22.6s (5/8)                                                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 256B                                                                                    0.0s
 => [internal] load metadata for docker.io/library/eclipse-temurin:19                                                   3.8s
 => [auth] library/eclipse-temurin:pull token for registry-1.docker.io                                                  0.0s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [1/3] FROM docker.io/library/eclipse-temurin:19@sha256:f3fbf1ad599d4b5dbdd7ceb55708d10cb9fafb08e094ef91e92aa  18.7s
 => => resolve docker.io/library/eclipse-temurin:19@sha256:f3fbf1ad599d4b5dbdd7ceb55708d10cb9fafb08e094ef91e92aa6  0.0s
 => => sha256:a182a611d05b01879f065473c49fb968b8d30312f931350ea07af1c46aa8b4f9 16.98MB / 16.98MB                    2.6s
 => => sha256:5f4059624ea044152b112a63a260d905e9e04009d3d4bdc57dd873c36824574d 174B / 174B                          0.7s
 => => sha256:426329e266e460bc5ce7cd2ad6d6841e68381cc09efc53c243182eb221cce102 201.12MB / 201.12MB                 17.5s
 => => sha256:74ac377868f863e123f24c409f79709f7563fa464557c36a09cf6f85c8b92b7f 30.43MB / 30.43MB                    6.4s
 => => extracting sha256:74ac377868f863e123f24c409f79709f7563fa464557c36a09cf6f85c8b92b7f                            1.5s
 => => extracting sha256:a182a611d05b01879f065473c49fb968b8d30312f931350ea07af1c46aa8b4f9                            1.0s
 => => extracting sha256:426329e266e460bc5ce7cd2ad6d6841e68381cc09efc53c243182eb221cce102                            0.9s
 => [internal] load build context                                                                                       1.3s
 => => transferring context: 20.72MB                                                                                    1.3s
```

```
C:\Users\miniMiracle>docker images
REPOSITORY                TAG                       IMAGE ID       CREATED              SIZE
springboot-docker-demo    0.1.RELEASE               bba0ac8ef00b   About a minute ago   778MB
openzipkin/zipkin         latest                    d9316e7ff757   2 weeks ago          377MB
rabbitmq                  3.13.7-management-alpine  d759525efd68   5 months ago         279MB

C:\Users\miniMiracle>
```

docker run -p 8081:8080 springboot-docker-demo meaning of this command

```
docker run -p 8081:8080 springboot-docker-demo
```

Copy

runs a Docker container from the `springboot-docker-demo` image and maps ports between the container and the host machine.

`docker run`	Runs a new **container** from a Docker image.
`-p 8081:8080`	Maps **port 8080** inside the container to **port 8081** on the host machine.
`springboot-docker-demo`	The **name of the Docker image** to run.

How It Works

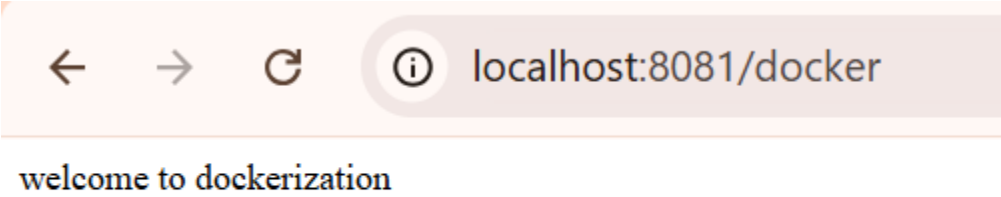
- Inside the **Docker container**, the Spring Boot app runs on **port 8080** (as defined in `application.properties` or `Dockerfile`).
- The `-p 8081:8080` flag **maps** this internal port (`8080`) to `8081` on the host (your computer).
- Now, instead of accessing the app at `http://localhost:8080`, you **must use** `http://localhost:8081`.

```
E:\wipro_training_2025_jan_22nd\springboot\springbootdockerdemo>docker run -p 8081:8080 springboot-docker-demo:0.1.RELEA
SE
```

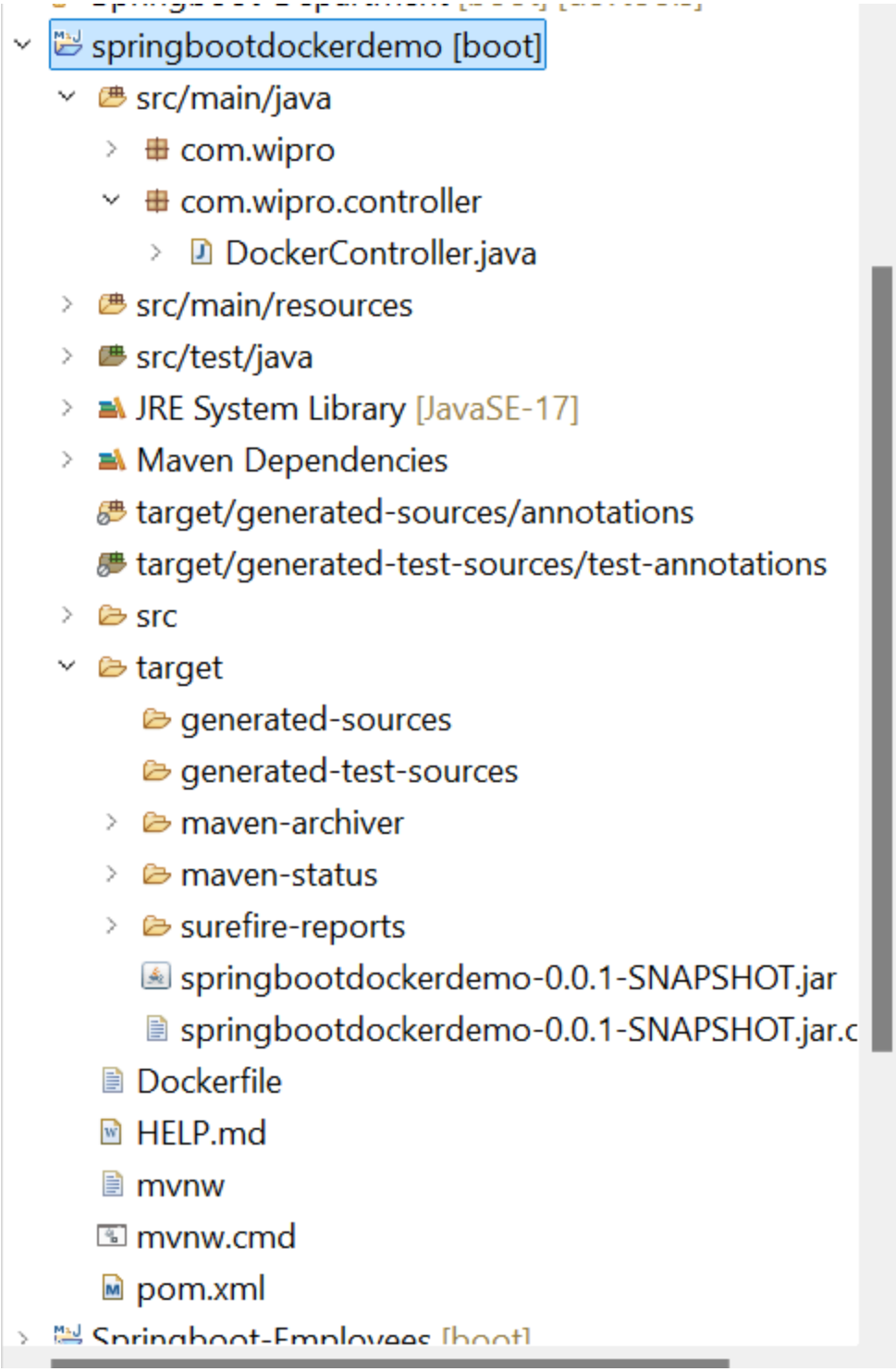
```
 => => exporting manifest list sha256:bba0ac8ef00bff3f28d3dd91a0
11617b70e187906fc5340cdd500f7295c61e57                    0.0s
 => => naming to docker.io/library/springboot-docker-demo:0.1.RE
LEASE                                                     0.0s
 => => unpacking to docker.io/library/springboot-docker-demo:0.1
.RELEASE                                                  0.3s

C:\Users\miniMiracle\eclipse-workspace\springbootdockerdemo\spri
ngbootdockerdemo>docker run -p 8080 springboot-docker-demo:9.1.R
ELEASE
```

Command to run:
docker run -p 8080:8080 springboot-docker-demo:0.1.RELEASE

```
C:\Users\miniMiracle\eclipse-workspace\springbootdockerdemo\springbootdockerdemo>docker run -p 8080:8080 springboot-docker-demo:0.1.R
ELEASE

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.4.3)

2025-03-06T07:43:16.576Z  INFO 1 --- [springbootdockerdemo] [           main] c.wipro.SpringbootdockerdemoApplication  : Starting Spr
ingbootdockerdemoApplication v0.0.1-SNAPSHOT using Java 19.0.2 with PID 1 (/app/springboot-docker-demo.jar started by root in /app)
2025-03-06T07:43:16.580Z  INFO 1 --- [springbootdockerdemo] [           main] c.wipro.SpringbootdockerdemoApplication  : No active pr
ofile set, falling back to 1 default profile: "default"
2025-03-06T07:43:17.563Z  INFO 1 --- [springbootdockerdemo] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initi
alized with port 8080 (http)
2025-03-06T07:43:17.579Z  INFO 1 --- [springbootdockerdemo] [           main] o.apache.catalina.core.StandardService   : Starting ser
vice [Tomcat]
2025-03-06T07:43:17.580Z  INFO 1 --- [springbootdockerdemo] [           main] o.apache.catalina.core.StandardEngine    : Starting Ser
vlet engine: [Apache Tomcat/10.1.36]
2025-03-06T07:43:17.606Z  INFO 1 --- [springbootdockerdemo] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing
 Spring embedded WebApplicationContext
2025-03-06T07:43:17.608Z  INFO 1 --- [springbootdockerdemo] [           main] w.s.c.ServletWebServerApplicationContext : Root WebAppl
icationContext: initialization completed in 951 ms
2025-03-06T07:43:18.017Z  INFO 1 --- [springbootdockerdemo] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat start
ed on port 8080 (http) with context path '/'
2025-03-06T07:43:18.050Z  INFO 1 --- [springbootdockerdemo] [           main] c.wipro.SpringbootdockerdemoApplication  : Started Spri
ngbootdockerdemoApplication in 2.019 seconds (process running for 2.912)
```



welcome to dockerization


springbootdockerdemo [boot]
src/main/java
com.wipro
com.wipro.controller
DockerController.java
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
src
target
generated-sources
generated-test-sources
maven-archiver
maven-status
surefire-reports
springbootdockerdemo-0.0.1-SNAPSHOT.jar
springbootdockerdemo-0.0.1-SNAPSHOT.jar.c
Dockerfile
HELP.md
mvnw
mvnw.cmd
pom.xml

DockerController.java | SpringbootdockerdemoApplication.java | Dockerfile

```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.web.bind.annotation.GetMapping;

@RestController
public class DockerController {

    @GetMapping("/docker")
    public String getData()
    {
        return "welcome to dockerization";
    }
}
```

Again checking

```
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
```

```
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
```

Localhost3 is the container name

In the docker containers