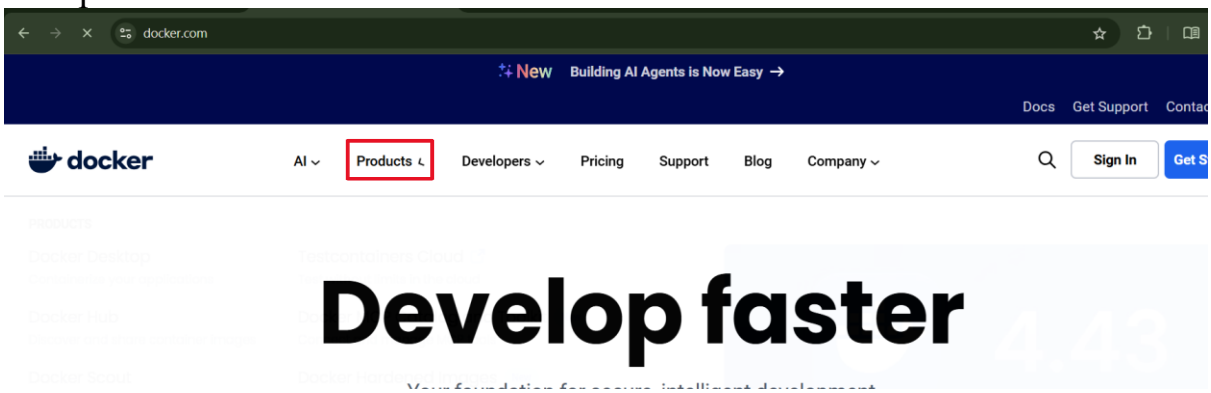
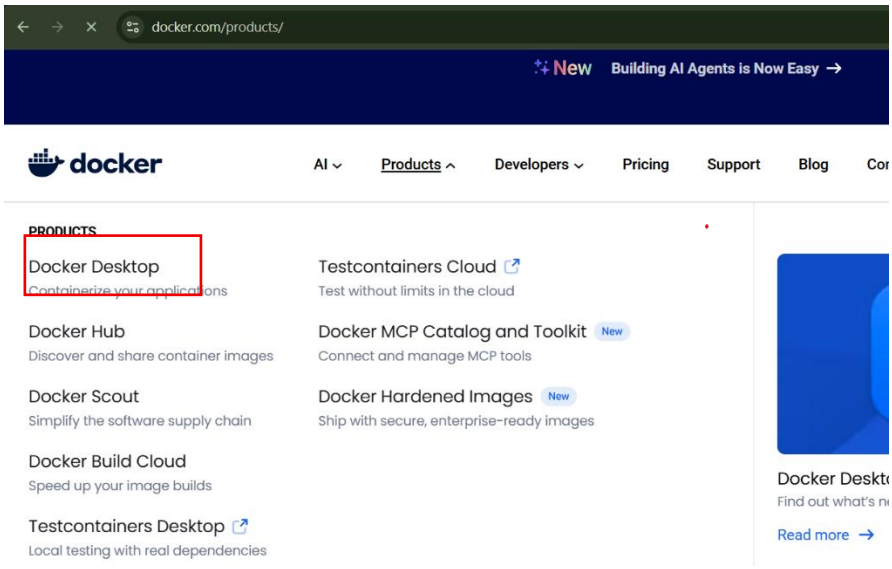
DOCKER - INSTALLATION

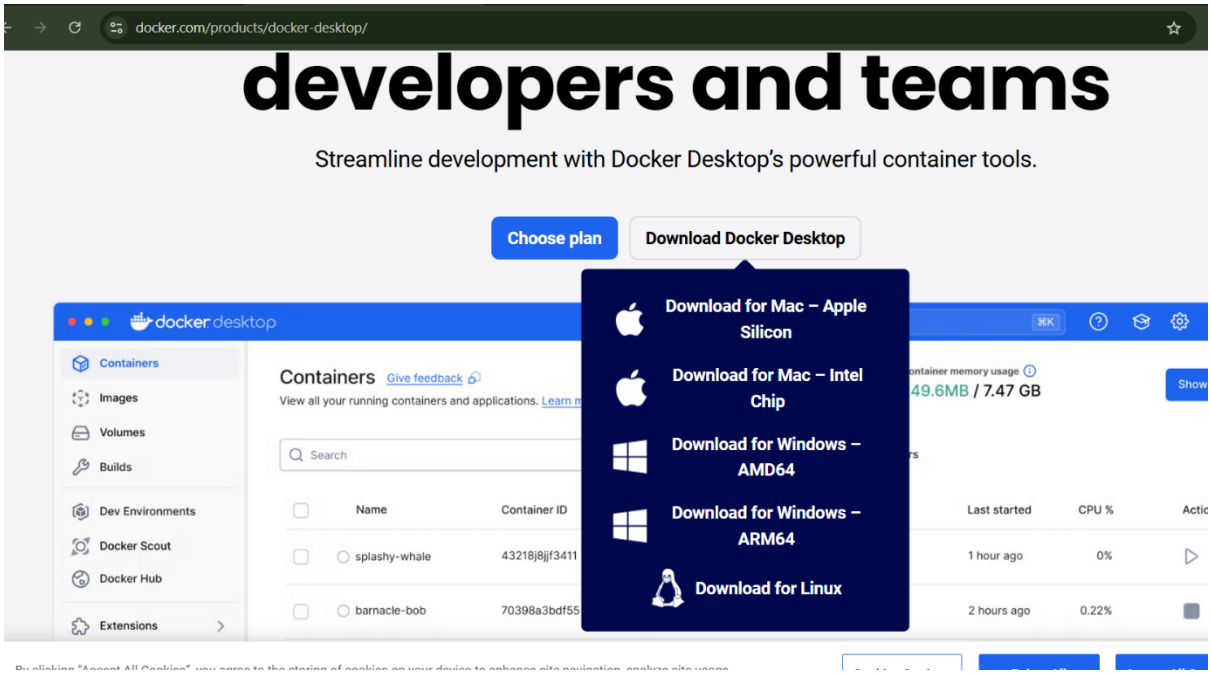
- Goto www.docker.com
- Open "Products" tab



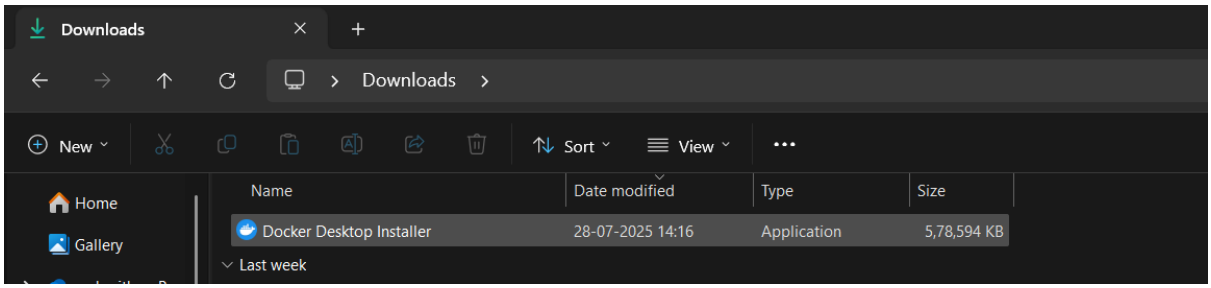
- Once you get into "Products" tab, you will find "Docker Desktop" as given below



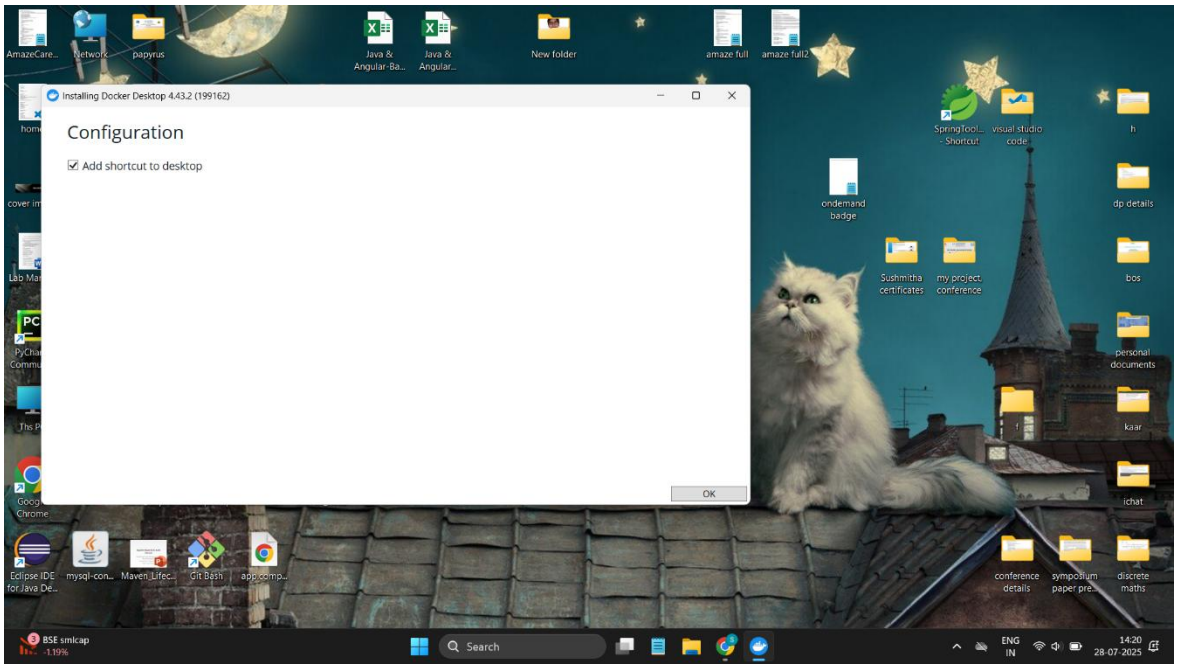
- Goto "Download Docker Desktop"
- Download the docker as per your OS.
- You will find two docker options for windows. To choose correct docker for your windows OS,,,
    1. Press Windows + R, type msinfo32, and press Enter
    2. Then Look for: System Type
    3. If you find system type as, x64-based PC → use AMD64
       If system type is ARM-based PC → use ARM64



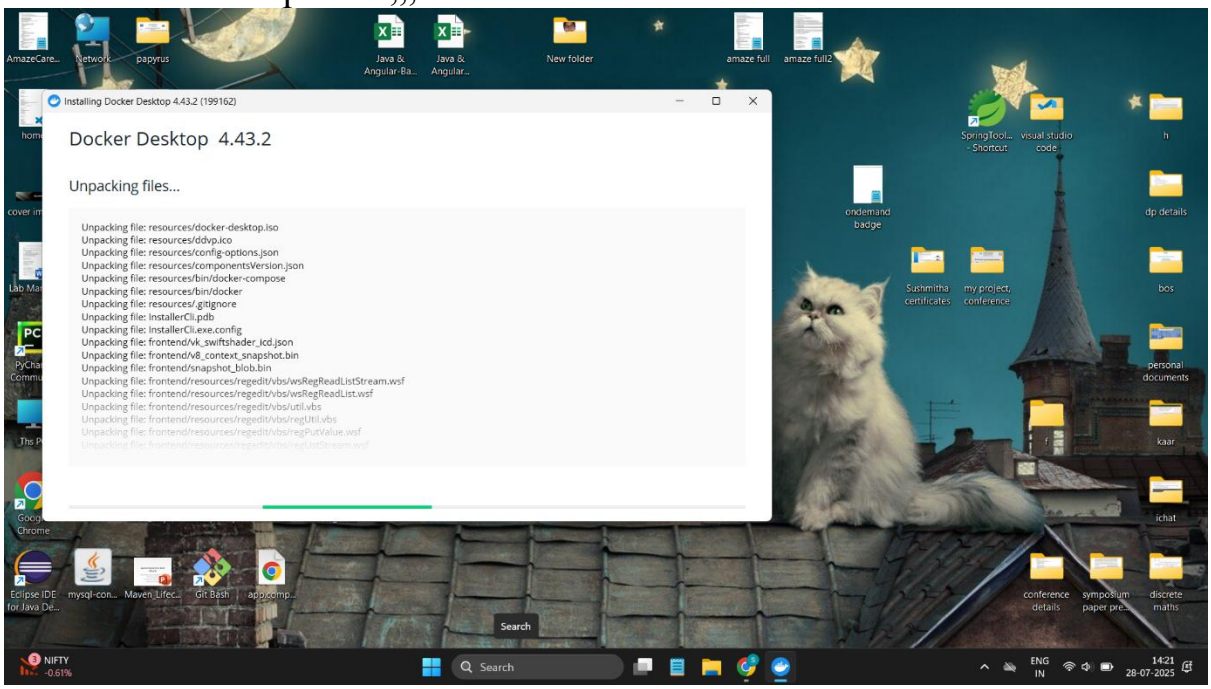
- After downloading,,, Goto downloads -> double click the "docker desktop installer"



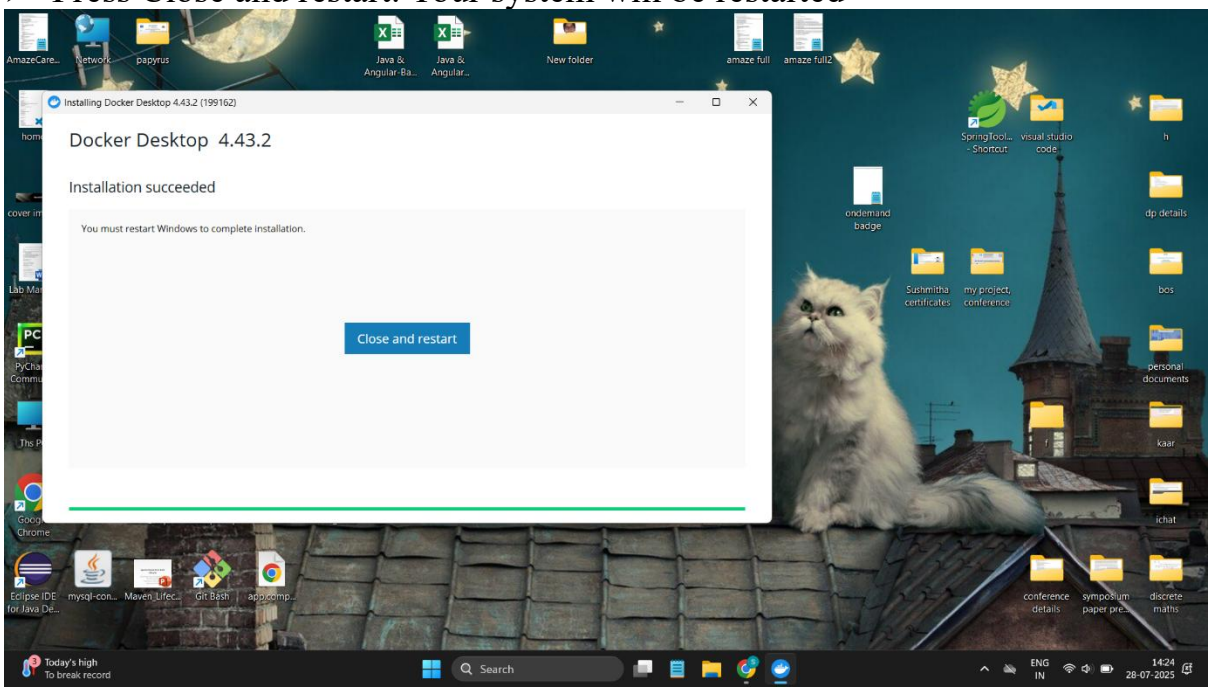
- Check the box -> Press OK



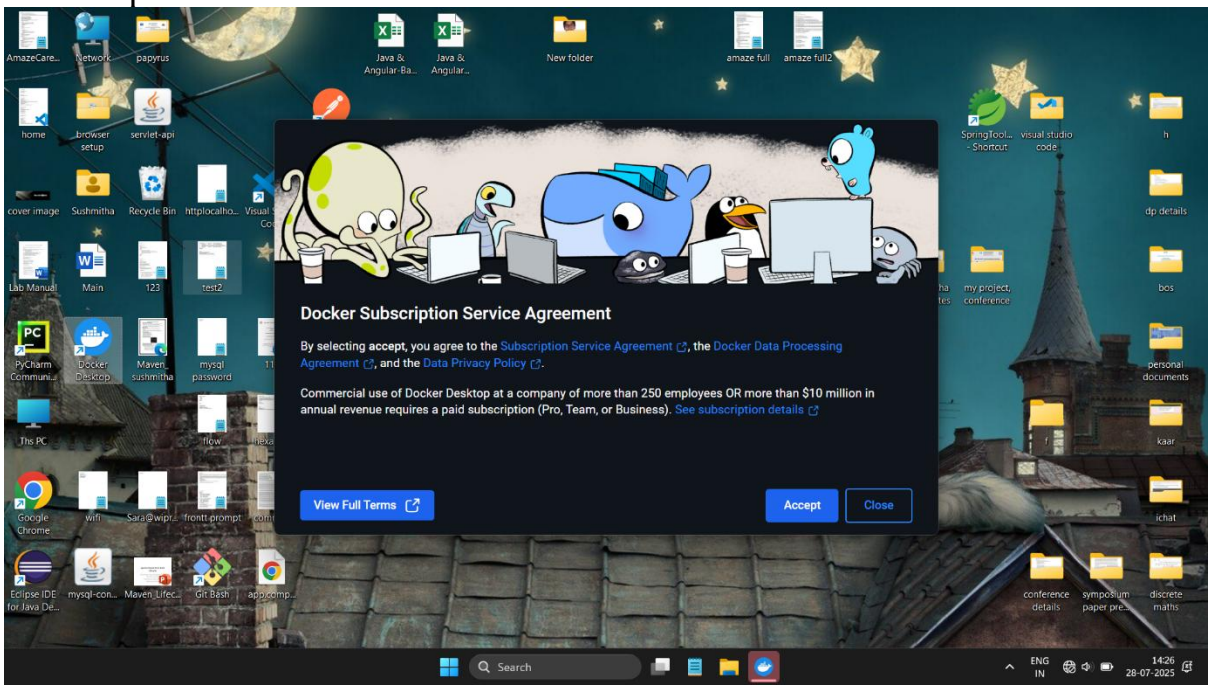
➢ Files will be unpacked,,, wait until it is finished



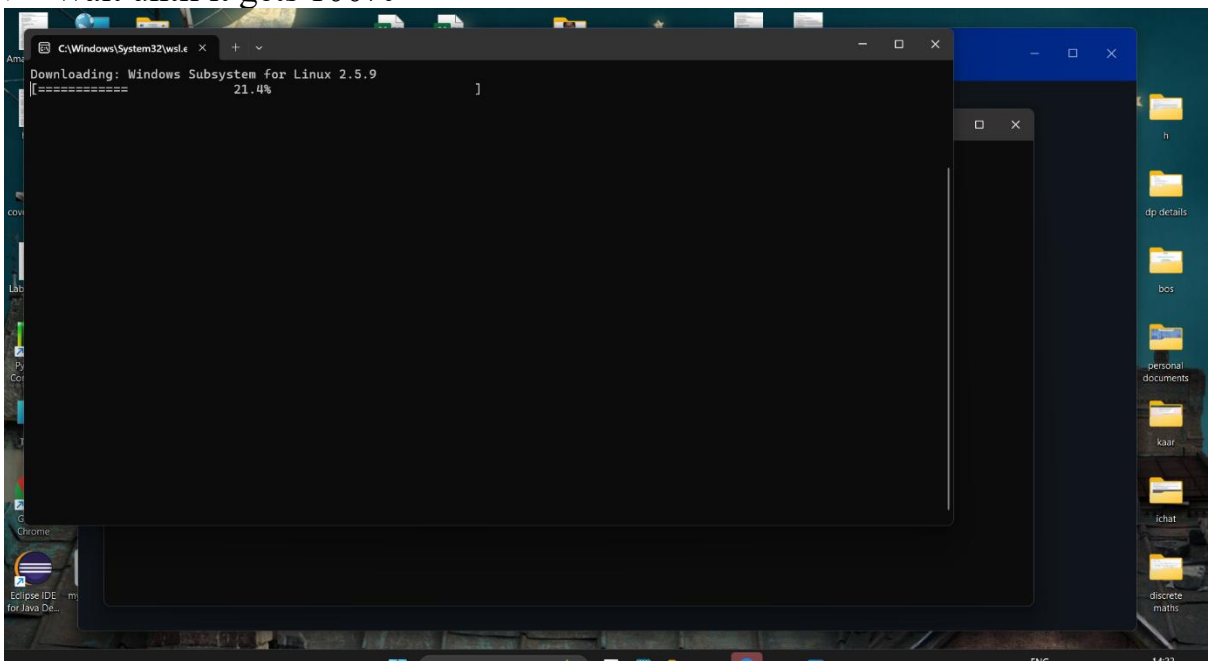
➢ Press Close and restart. Your system will be restarted



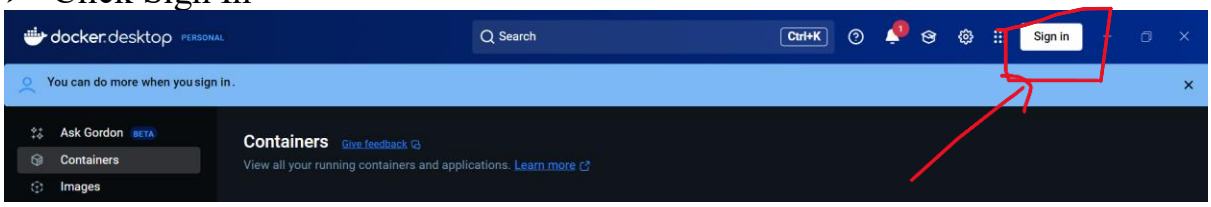
- Click the Docker Desktop app
- Accept the terms



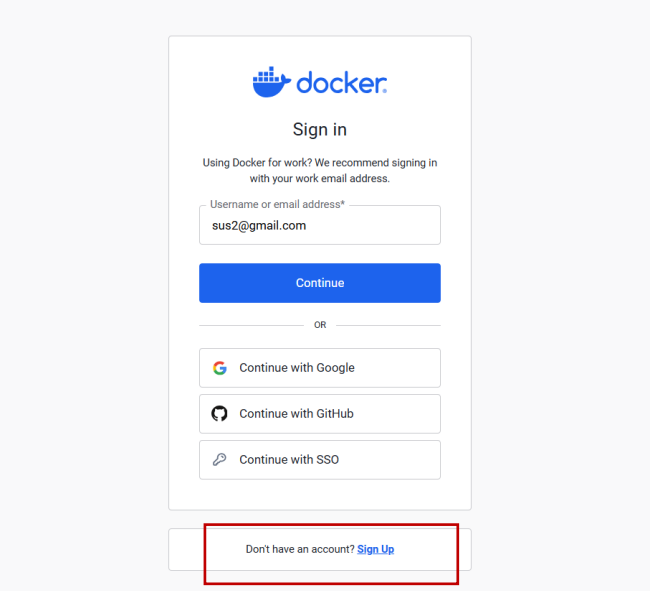
- Once you accept the terms, cmd will be automatically opened.
- Press Enter
- WSL(Windows Subsystem for Linux) will get to download
- Wait until it gets 100%



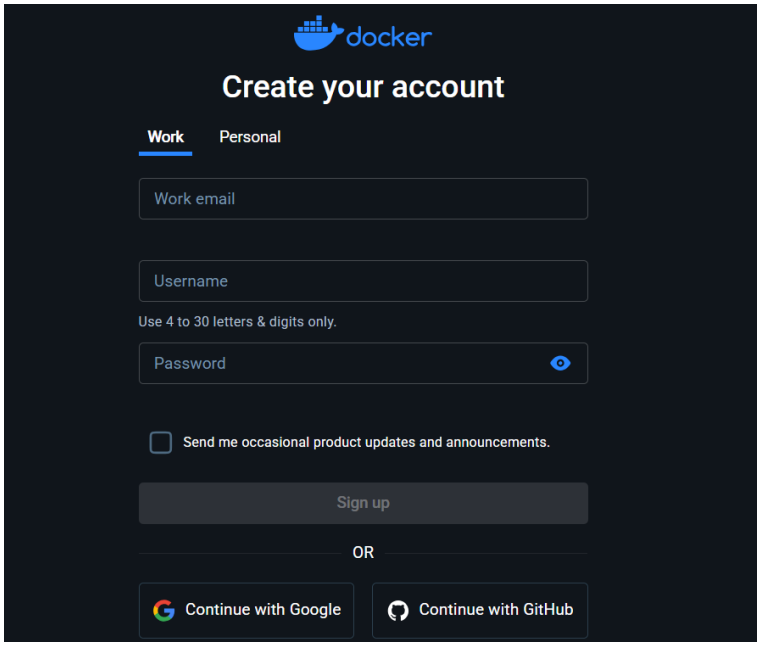
- Once WSL is downloaded, close everything and open Docker Desktop app again
- Click Sign In



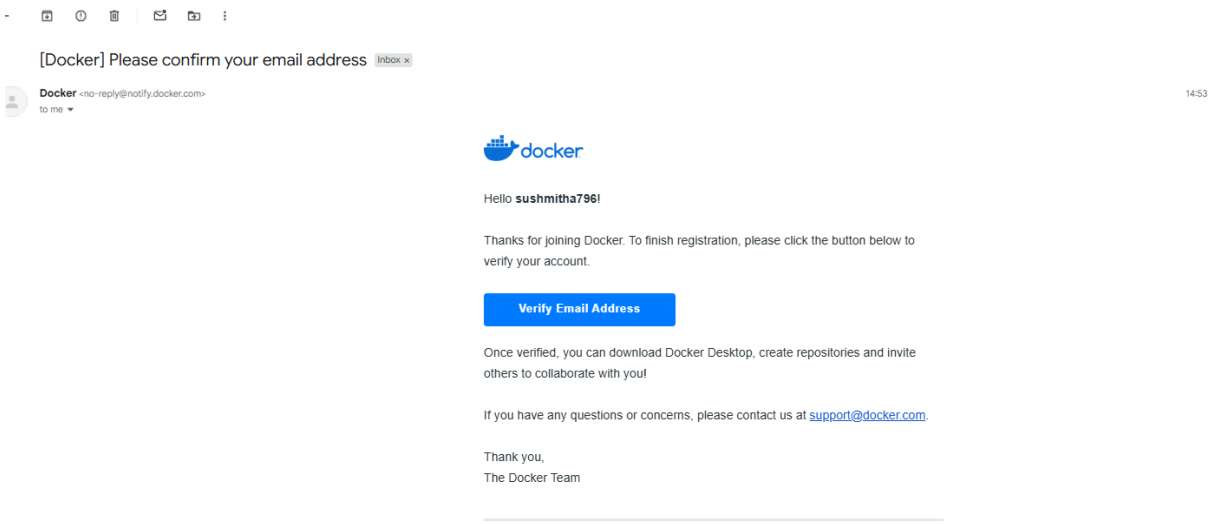
- If you don't have account -> Press SignUp



- Click either Work or Personal -> Fill the details -> Press Sign Up

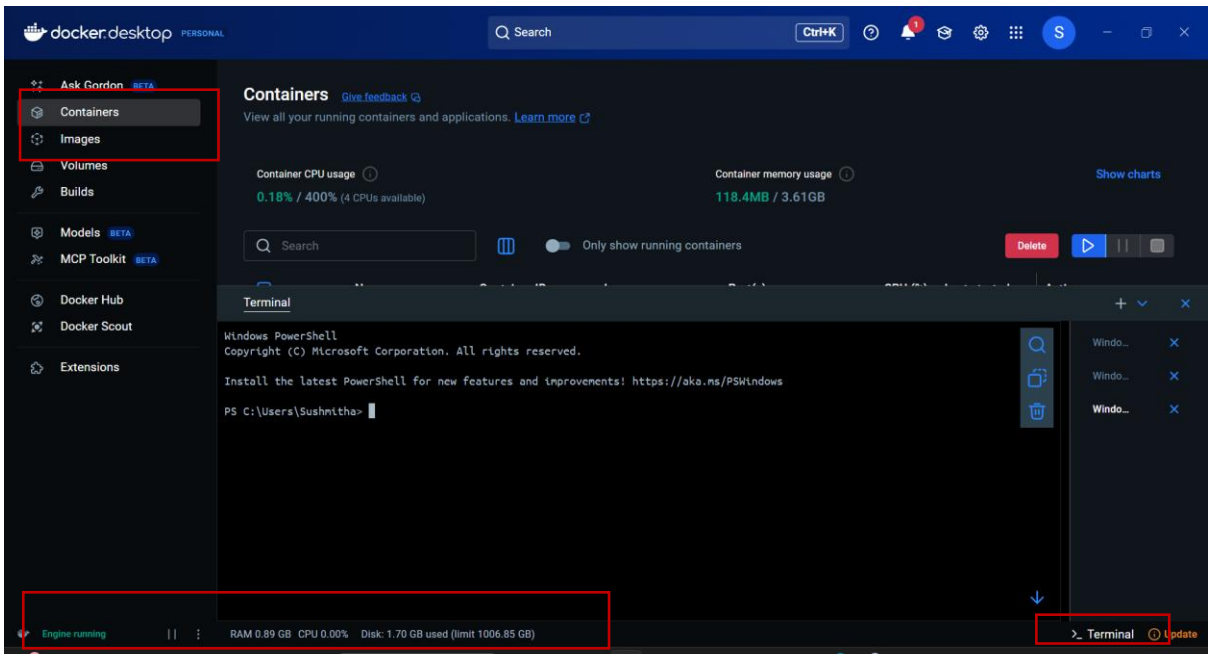


- Once you fill your details and Sign Up, Mail will be triggered to confirm your given mail id.
- Goto mail -> Verify email address



- Once finished, Login using your username and password

OPTIONS IN DOCKER DESKTOP APP



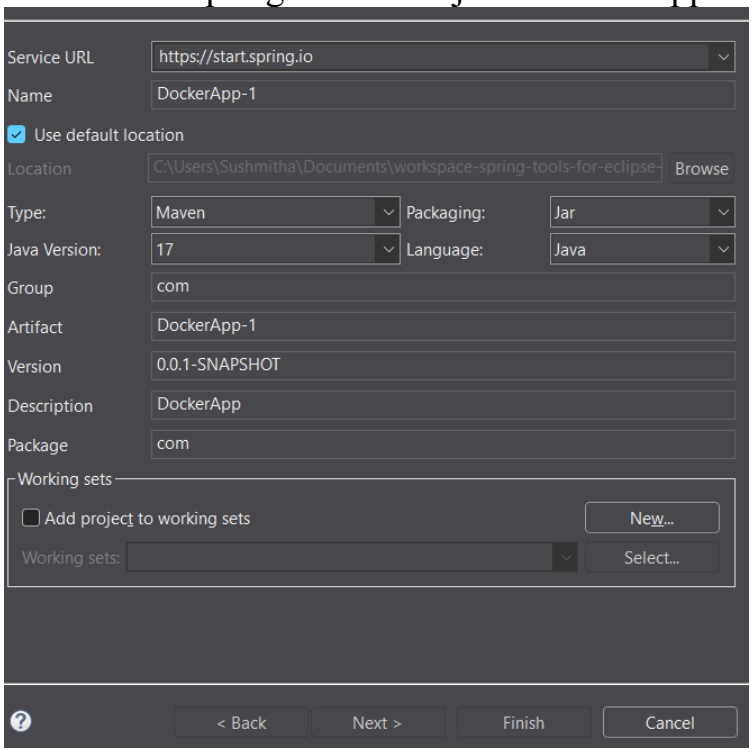
- Navigate to “Images” to see the list of images created
- Navigate to “Containers” to see the list of images created
- Bottom Right Corner holds “Terminal” option and this is inbuilt terminal
- Check whether Docker Desktop app is running or not, by “Engine running” message in the bottom. If not running, it willnot show Engine running message
- You can find the amount RAM, CPU and Disk memory holded by Docker app in the bottom

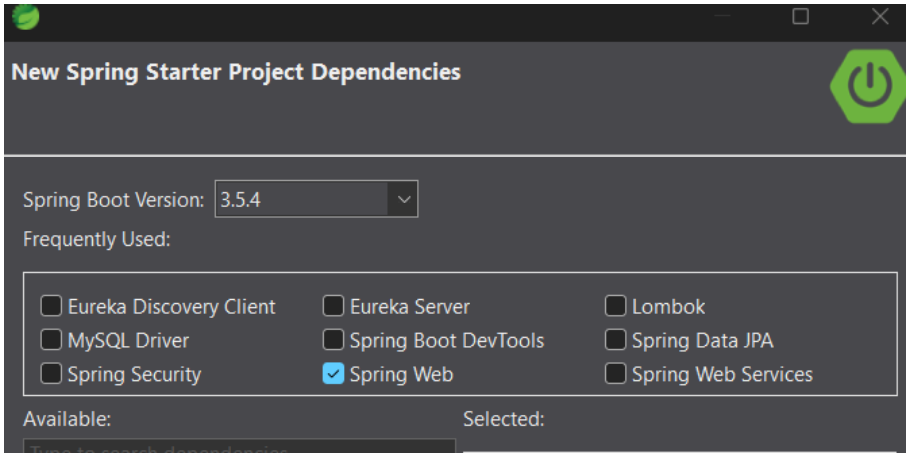
FLOW – TO BE FOLLOWED –
TO CONVERT A SPRINGBOOT APPLICATION INTO A CONTAINER

1. Create a springboot application
2. Create Dockerfile to create an Image for that springboot application
3. Run the image and then Container will be generated
4. Run the Container to see output

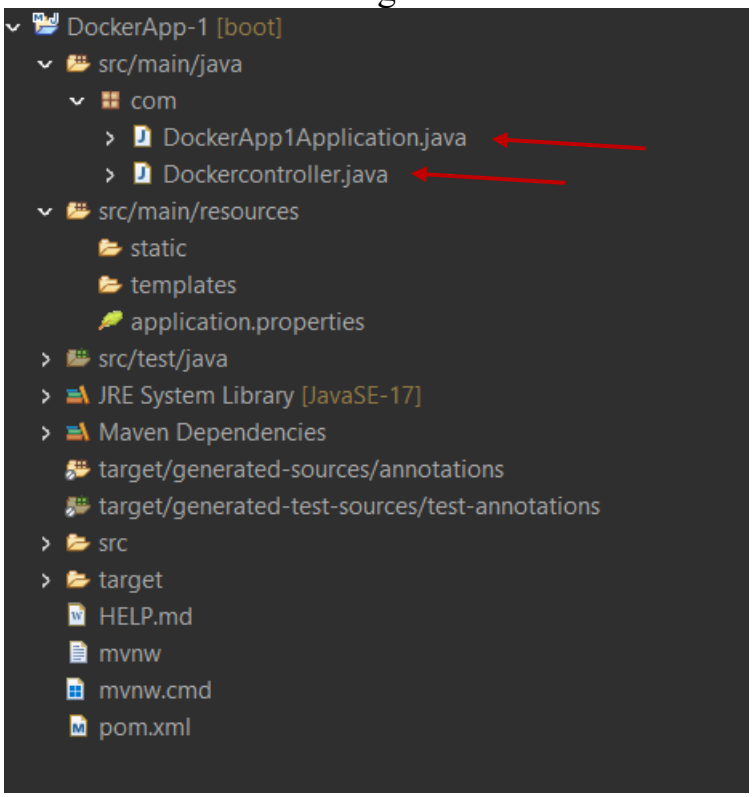
Step 1: Creating simple springboot application

➢ Create a Spring Strater Project “DockerApp-1”





- Create two classes as given below



- **Copy paste the below given code in respective files and Save the files**

DockerApp1Application.java

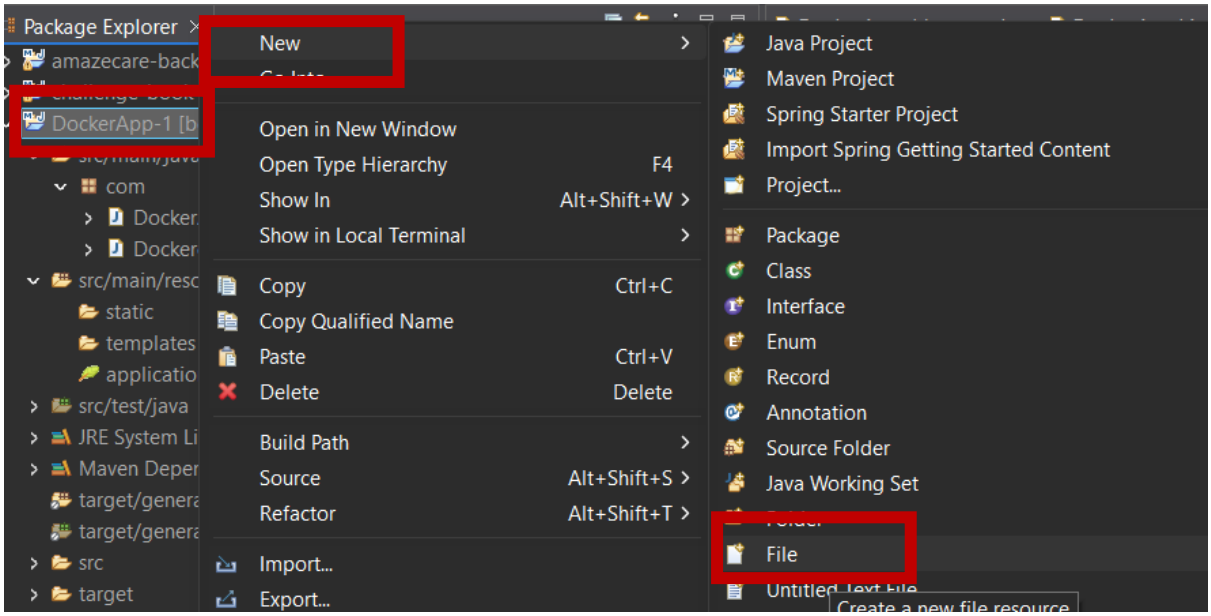
```
package com;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class DockerApp1Application {
public static void main(String[] args) {
SpringApplication.run(DockerApp1Application.class, args);}}
```

Dockercontroller.java

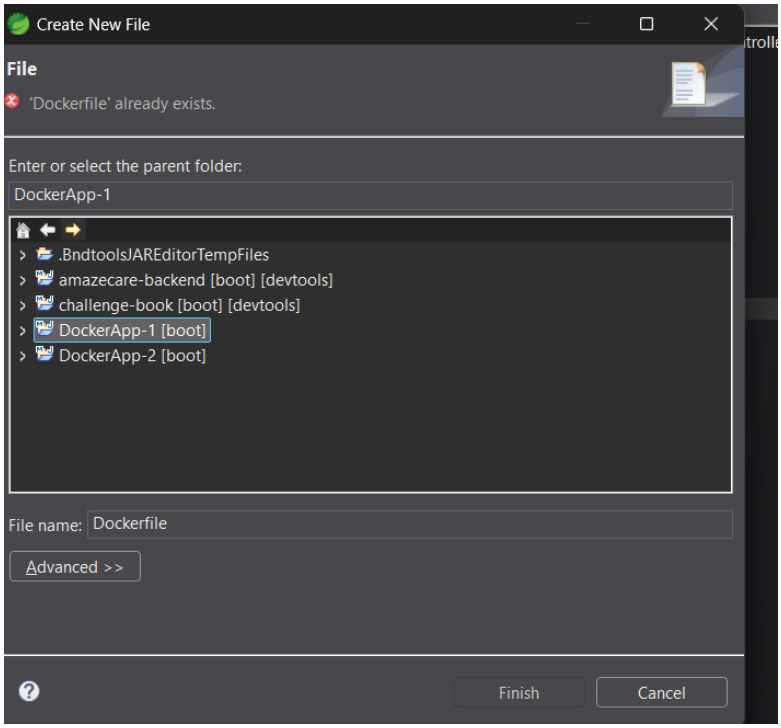
```
package com;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class Dockercontroller{
@GetMapping("/data")
public String getData() {
return "Hello Friend";
}
}
```

Step 2: Creating Dockerfile and .jar

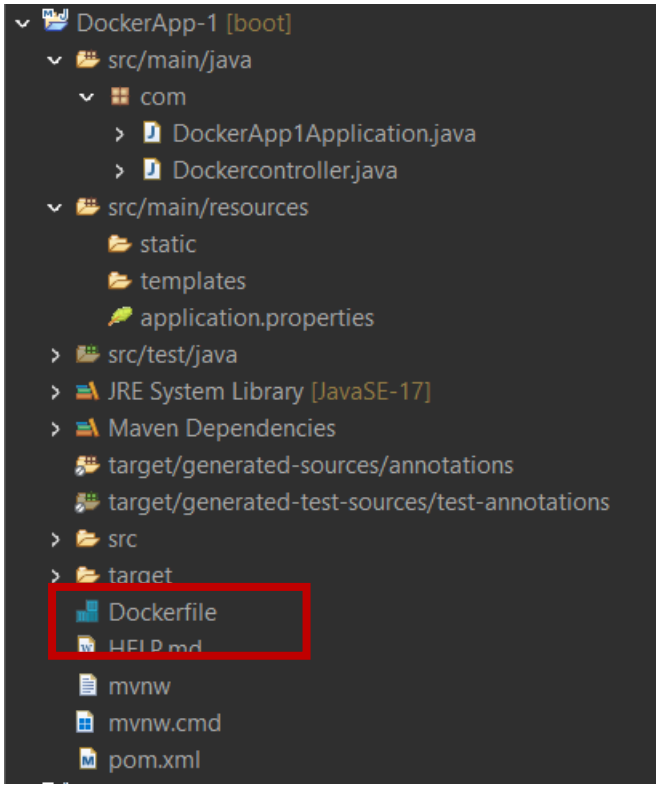
- Right click on “DockerApp-1”
- Goto New -> File



- Give file name as “Dockerfile” (should be same as “Dockerfile”)
- Click Finish

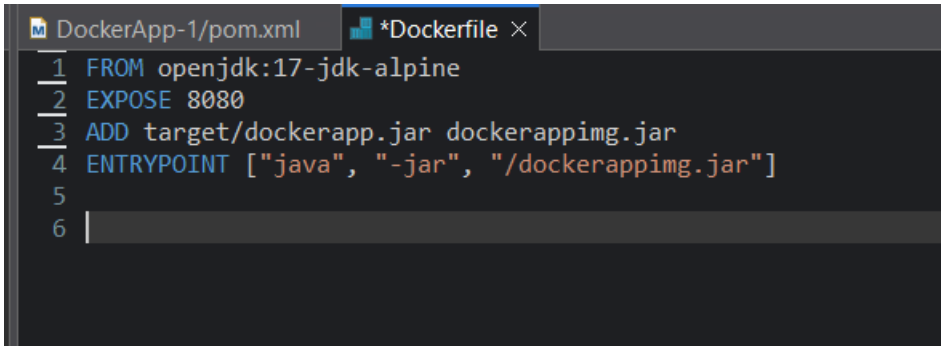


- Dockerfile will be created and it will be visible after target folder

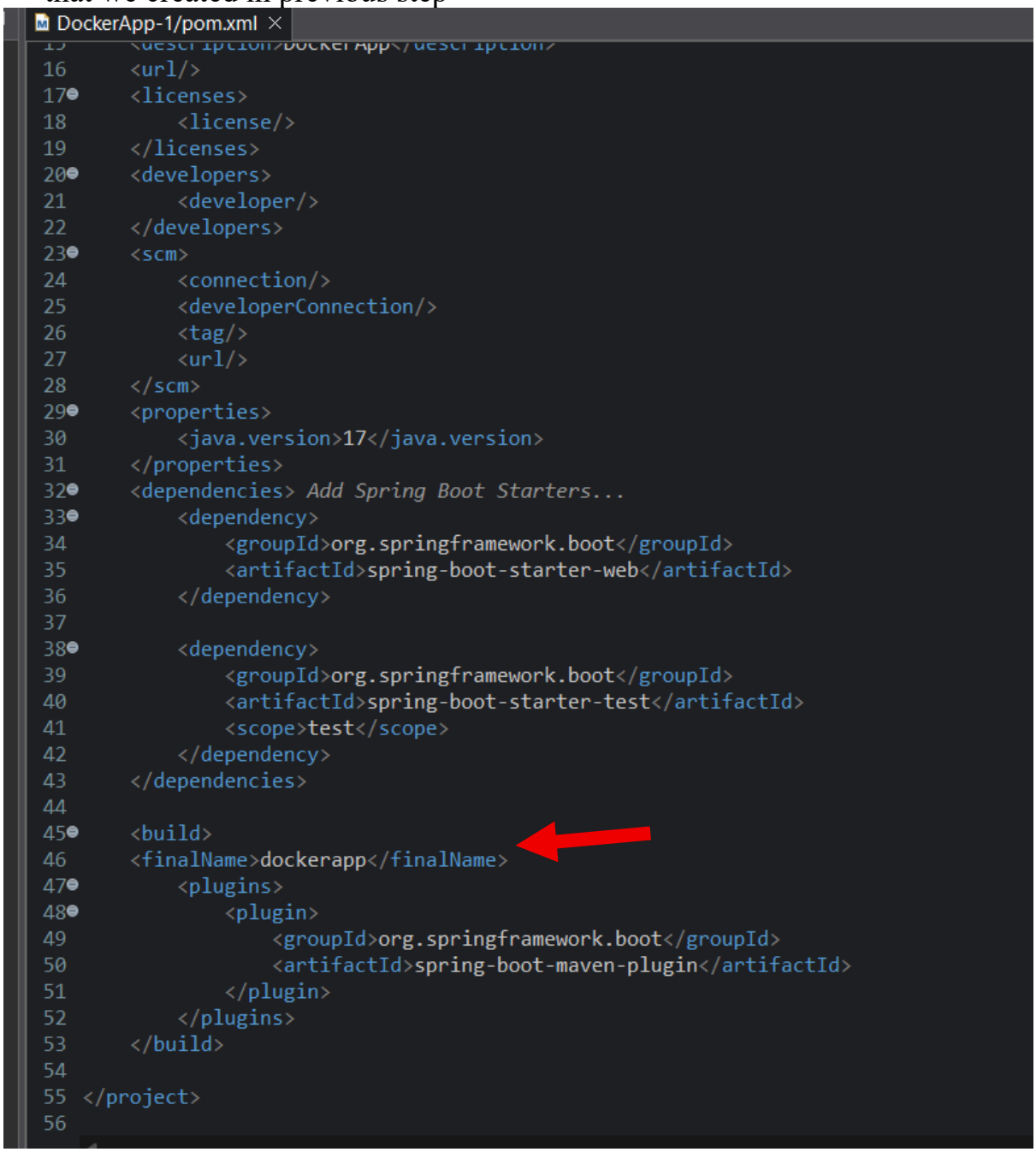


- **Copy Paste the code in Dockerfile, as given below in the screenshot and Save the file**

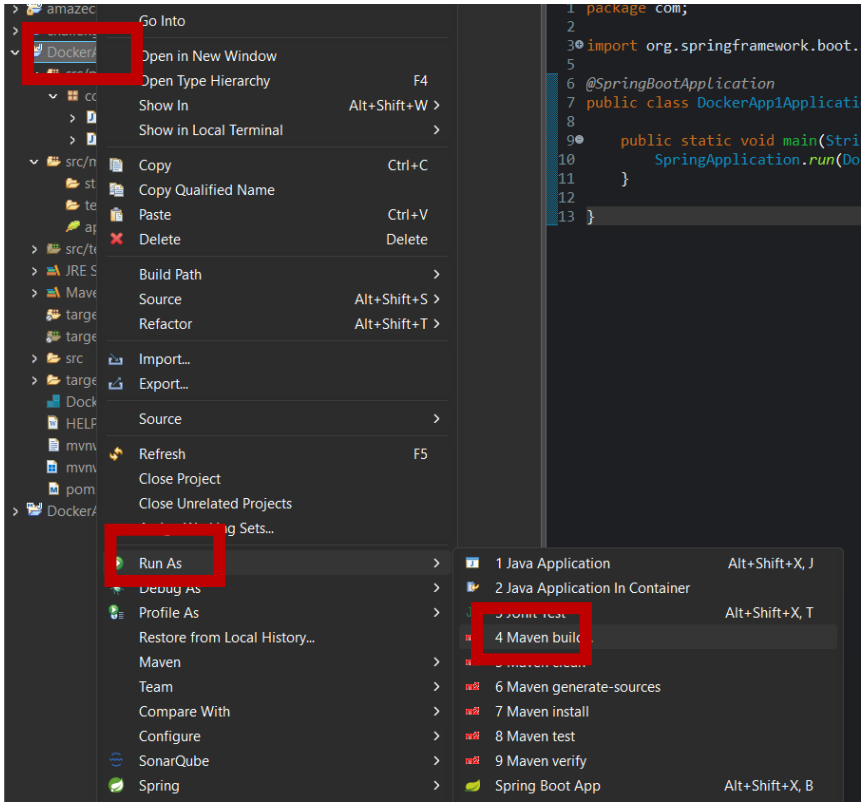
```
FROM openjdk:17-jdk-alpine
EXPOSE 8080
ADD target/dockerapp.jar dockerappimg.jar
ENTRYPOINT ["java", "-jar", "/dockerappimg.jar"]
```



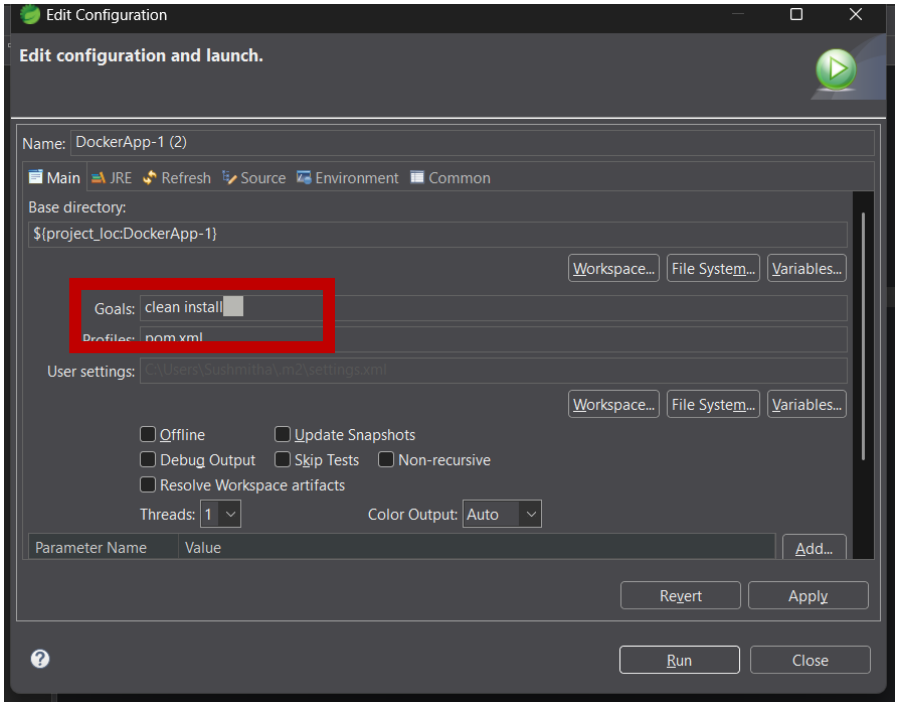
- Goto pom.xml file
- Add <finalName>dockerapp</finalName>
- Add this tag within <build> …. </build>
- Finalname tag fixes the given name as the jar file name (Example: dockerapp.jar)
- You can Give your own name within the finalname tag. If you change the finalname then Make sure to add the same name in the 3rd line of Dockerfile that we created in previous step



- Right click on DockerApp-1
- Goto Run As -> Maven build



- Set Goals as “clean install”
- Click Run

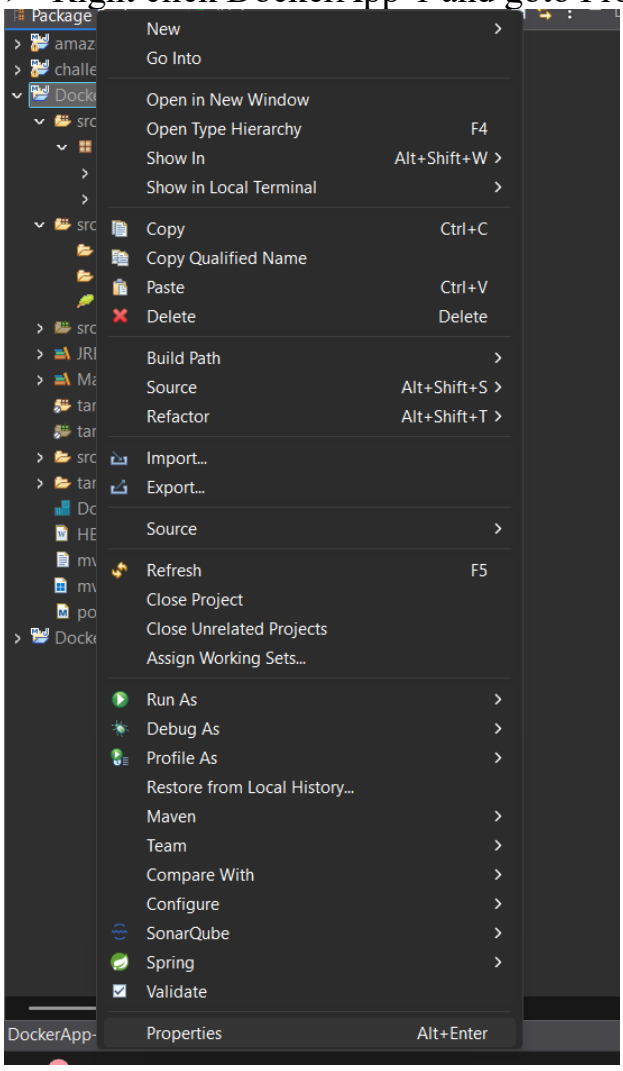


➢ You will get Buil success message in the console

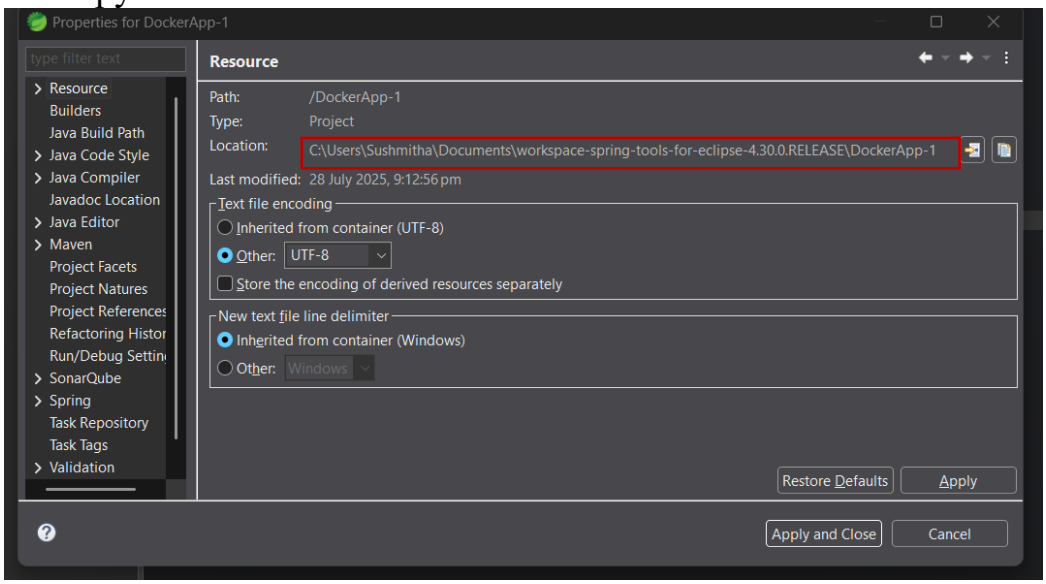
```
[INFO]
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ demo ---
[INFO] Building jar: C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\DockerApp-1\t
[INFO]
[INFO] --- spring-boot:3.5.4:repackage (repackage) @ demo ---
[INFO] Replacing main artifact C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\Doc
[INFO] The original artifact has been renamed to C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-
[INFO]
[INFO] --- install:3.1.4:install (default-install) @ demo ---
[INFO] Installing C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\DockerApp-1\pom.
[INFO] Installing C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\DockerApp-1\targ
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  13.478 s
[INFO] Finished at: 2025-07-28T21:13:07+05:30
[INFO] ------------------------------------------------------------------------
[WARNING] The requested profile "pom.xml" could not be activated because it does not exist.
```

Step 3: Creating image

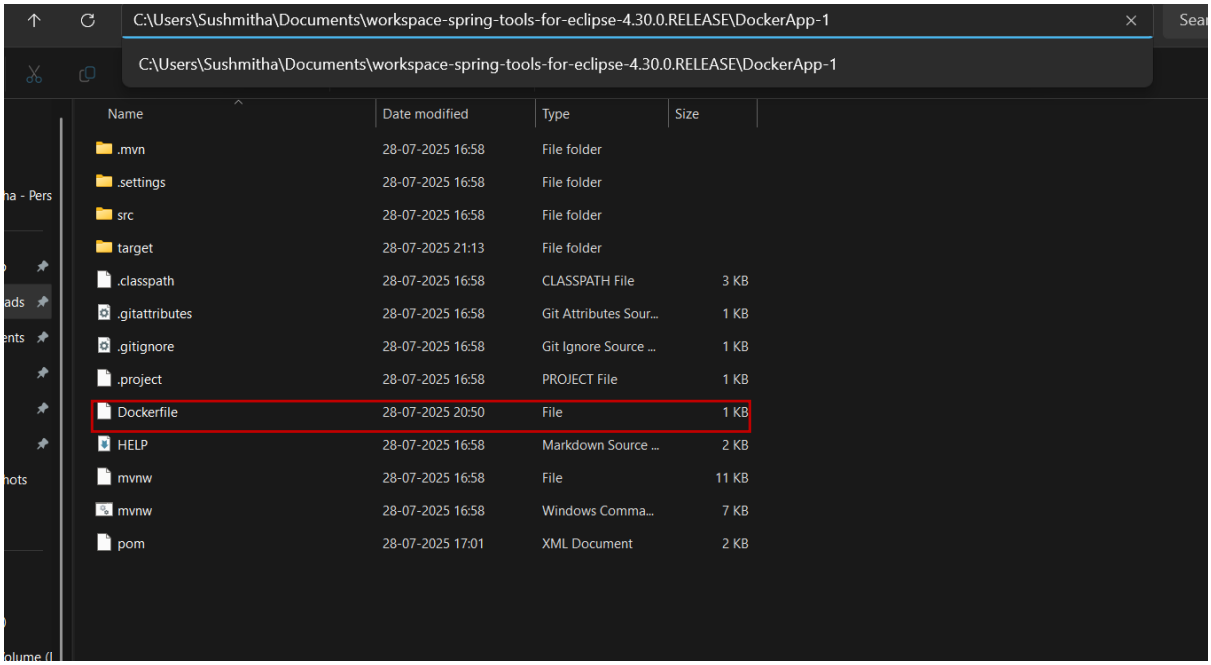
➢ Right click DockerApp-1 and goto Properties



- Copy the location



- Check if Dockerfile is visible in that location



- Open Command Prompt
- Use **docker build <path of file>** command to build or create the image as shown below

```
C:\Users\Sushmitha>docker build C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\DockerApp-1
[+] Building 7.4s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 308B
 => [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine
 => [auth] library/openjdk:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => CACHED [1/2] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9dbd9784383d81
 => [internal] load build context
 => => transferring context: 20.98MB
 => [2/2] ADD target/dockerapp.jar dockerappimg.jar
 => exporting to image
 => => exporting layers
 => => writing image sha256:8d7b0d252f48ce370b184874340acf816ab8d34b8d0f6274fa19e34c7e9cedf6
```

- Use **docker images** command to list out all images created by our local system
- Given below is the only image created by this local system so only one is shown

```
C:\Users\Sushmitha>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
<none>       <none>    8d7b0d252f48   6 seconds ago    347MB
```

- Copy the image ID from the output we got for **docker images** command
- Use **docker tag <image ID> yourImagename:latest** to rename Repository name and Tag name (it replaced <none> value as in the previous screenshot)
- *Example: docker tag 8d7b0d252f48 myimage:latest*
- Again use **docker images** command to check whether the name is changed

```
C:\Users\Sushmitha>docker tag 8d7b0d252f48 myimage:latest

C:\Users\Sushmitha>docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
myimage      latest    8d7b0d252f48   5 minutes ago    347MB
```

- Use docker run <yourImagename> command to run the image

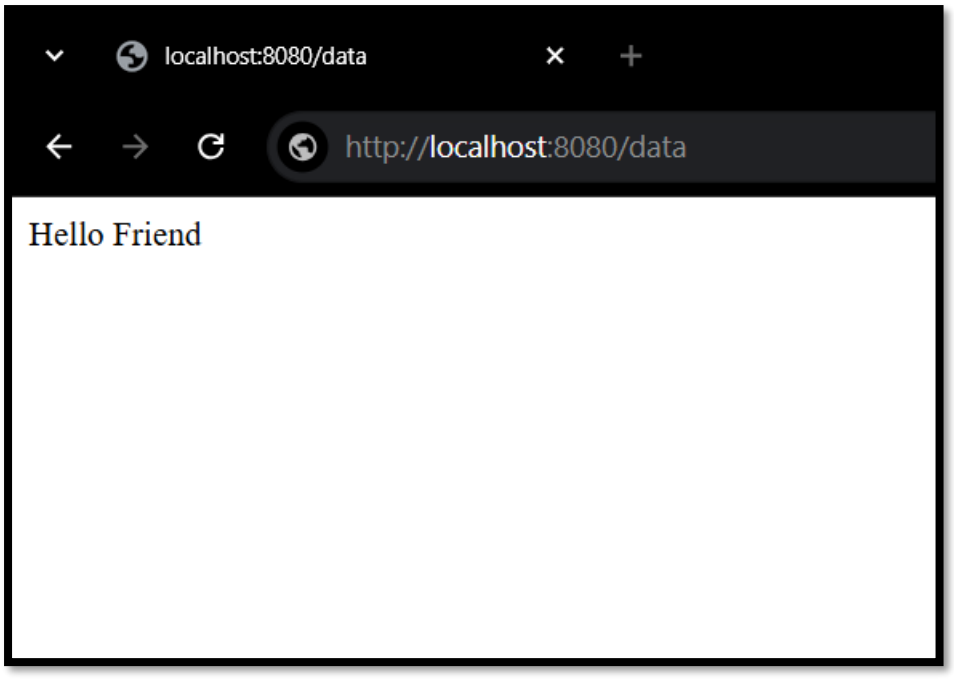
Example: docker run myimage

```
C:\Users\Sushmitha>docker run myimage

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

 :: Spring Boot ::                (v3.5.4)

2025-07-28T16:26:16.291Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : Starting DockerApp1Application v0.0.1-SNAPSH
OT using Java 17-ea with PID 1 (/dockerappimg.jar started by root in /)
2025-07-28T16:26:16.295Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : No active profile set, falling back to 1 def
ault profile: "default"
2025-07-28T16:26:18.097Z  INFO 1 --- [DockerApp-1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-28T16:26:18.134Z  INFO 1 --- [DockerApp-1] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-28T16:26:18.134Z  INFO 1 --- [DockerApp-1] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1
.43]
2025-07-28T16:26:18.245Z  INFO 1 --- [DockerApp-1] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationC
ontext
2025-07-28T16:26:18.246Z  INFO 1 --- [DockerApp-1] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization c
ompleted in 1826 ms
2025-07-28T16:26:18.915Z  INFO 1 --- [DockerApp-1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with cont
ext path '/'
2025-07-28T16:26:18.953Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : Started DockerApp1Application in 3.445 secon
ds (process running for 4.64)
```

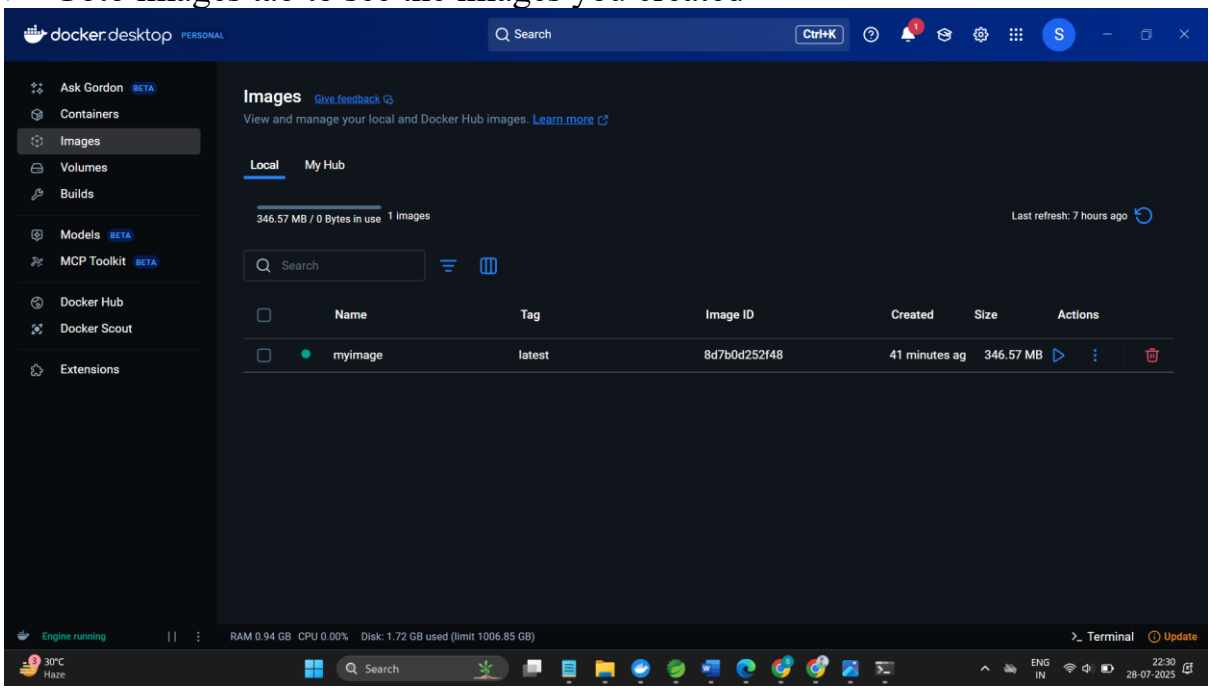
- Use docker run -p <host_port>:<container_port> <imageId>
  Example 1: docke run -p 8080:8080 8d7b0d252f48
  Or
  Example 2: docke run -p 3000:8080 8d7b0d252f48
- -p 8080:3000. This means any request to localhost:8080 will be forwarded to port 3000 inside the container. You can also map the same port on both sides using -p 8080:8080, which is helpful when you want consistent port access. If you don't use -p, the container's ports will remain isolated and not accessible from outside

```
C:\Users\Sushmitha>docker run -p 8080:8080 8d7b0d252f48

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

 :: Spring Boot ::                (v3.5.4)

2025-07-28T16:27:16.933Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : Starting DockerApp1Application v0.0.1-SNAPSH
OT using Java 17-ea with PID 1 (/dockerappimg.jar started by root in /)
2025-07-28T16:27:16.939Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : No active profile set, falling back to 1 def
ault profile: "default"
2025-07-28T16:27:18.503Z  INFO 1 --- [DockerApp-1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-28T16:27:18.527Z  INFO 1 --- [DockerApp-1] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-28T16:27:18.527Z  INFO 1 --- [DockerApp-1] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1
.43]
2025-07-28T16:27:18.575Z  INFO 1 --- [DockerApp-1] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationC
ontext
2025-07-28T16:27:18.578Z  INFO 1 --- [DockerApp-1] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization c
ompleted in 1544 ms
2025-07-28T16:27:19.207Z  INFO 1 --- [DockerApp-1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with cont
ext path '/'
2025-07-28T16:27:19.227Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : Started DockerApp1Application in 3.223 secon
ds (process running for 3.893)
2025-07-28T16:27:56.699Z  INFO 1 --- [DockerApp-1] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispa
tcherServlet'
2025-07-28T16:27:56.700Z  INFO 1 --- [DockerApp-1] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2025-07-28T16:27:56.713Z  INFO 1 --- [DockerApp-1] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 13 ms
```

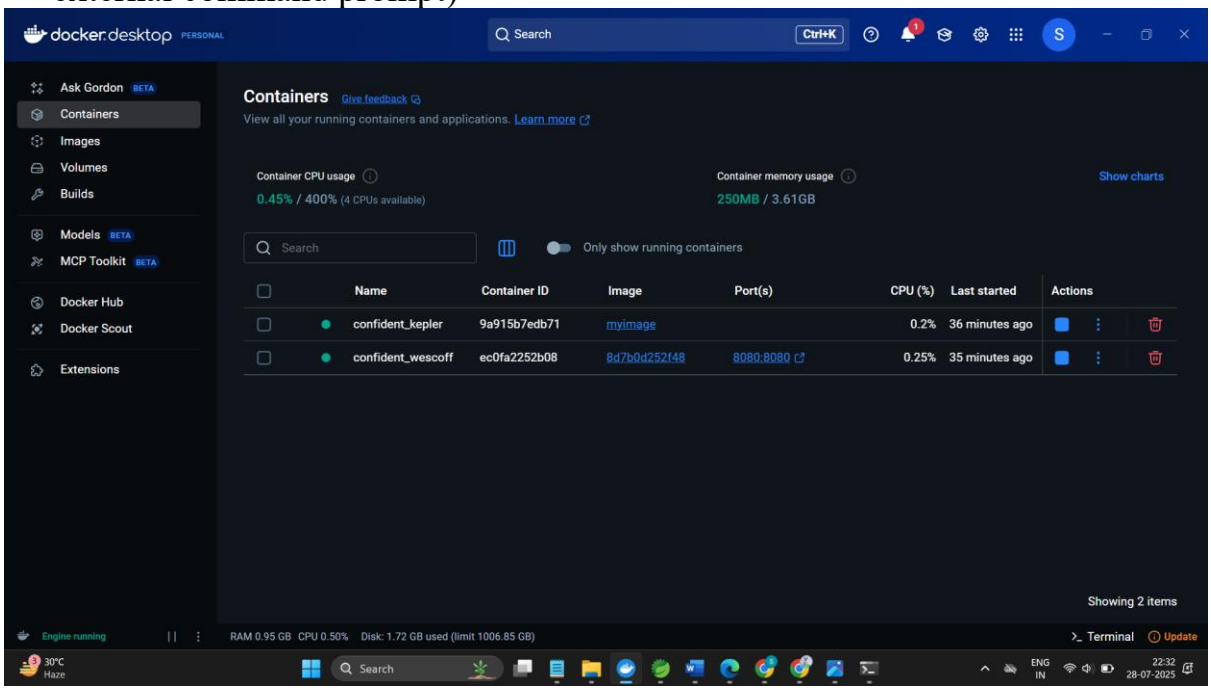
- Goto to browser
- Paste this http://localhost:8080/data (if you use -p 8080:8080 in previous step) in url box and press enter
  (Or)
- Paste this http://localhost:3000/data (if you use -p 3000:8080 in previous step) in url box and press enter



- Open Docker Desktop app
- Goto images tab to see the images you created



- Goto Containers tab to see the Containers you created
- You can also use the inbuilt terminal to run the commands (instead of external command prompt)



PUSHING AN IMAGE TO DOCKER HUB

- Logging to the Docker desktop app is mandatory
- Open command prompt
- Use *docker images* command to list the images we have.
- We are going to push this "myimage" in to docker hub

```
C:\Users\Sushmitha>docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
myimage      latest    8d7b0d252f48   12 hours ago   347MB
```

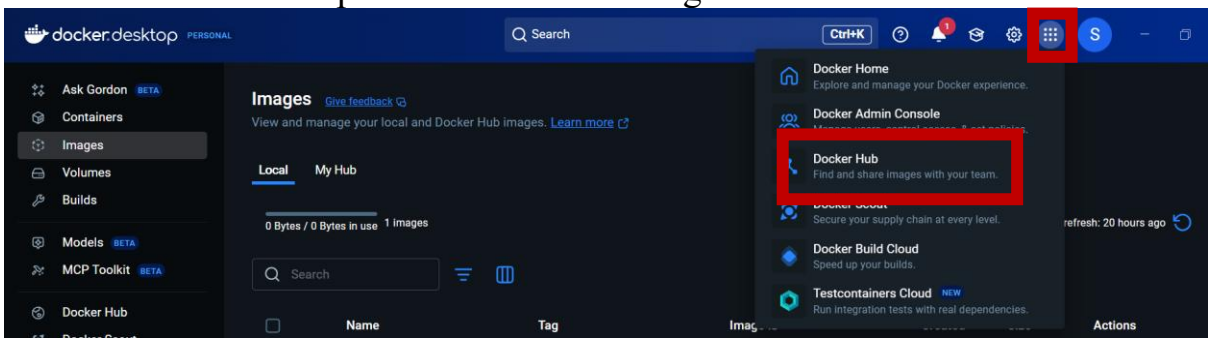
- You have to tag your image with your username and repository name.
- Use command
  docker tag <your image name> <your-username>/<repo-name>:<tag>
  to tag your local image.
- Give the name of your image in <your image name> and give your correct user name in < you-username>
- Name your repository as you wish in <repo-name>
- Give the exact tag name <tag> as you see in the output of docker images command
  E*xample: docker tag myimage sushmitha796/myimage:latest*
  Here I have given repository name as same as image name, you can give different names for respository.

```
C:\Users\Sushmitha>docker tag myimage sushmitha796/myimage:latest

C:\Users\Sushmitha>docker images
REPOSITORY             TAG       IMAGE ID       CREATED        SIZE
myimage                latest    8d7b0d252f48   12 hours ago   347MB
sushmitha796/myimage   latest    8d7b0d252f48   12 hours ago   347MB
```

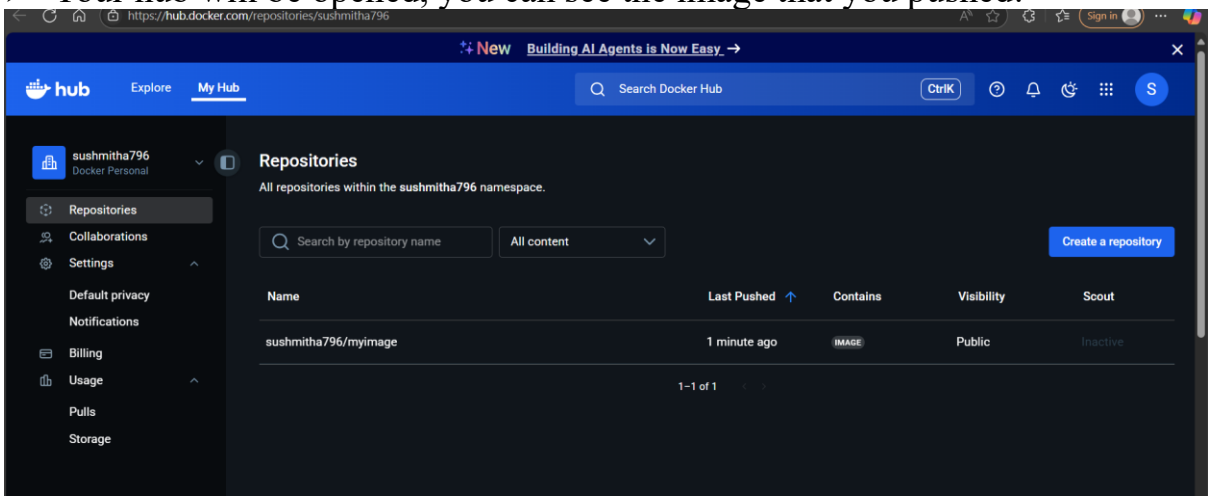
- To push the image in docker hub, use
  docker push <dockerhub-username>/<repository-name>:<tag>
- This uploads the image to Docker Hub under your account.
  Example: docker push sushmitha796/myimage:latest

```
C:\Users\Sushmitha>docker push sushmitha796/myimage:latest
The push refers to repository [docker.io/sushmitha796/myimage]
2cf3f3ec2619: Pushed
34f7184834b2: Mounted from library/openjdk
5836ece05bfd: Mounted from library/openjdk
72e830a4dff5: Mounted from library/openjdk
Put "https://registry-1.docker.io/v2/sushmitha796/myimage/manifests/latest": EOF
```

- Lets check whether the image is pushed to the repository in our account
- Goto Docker Desktop app
- Click the marked option in the screenshot given below

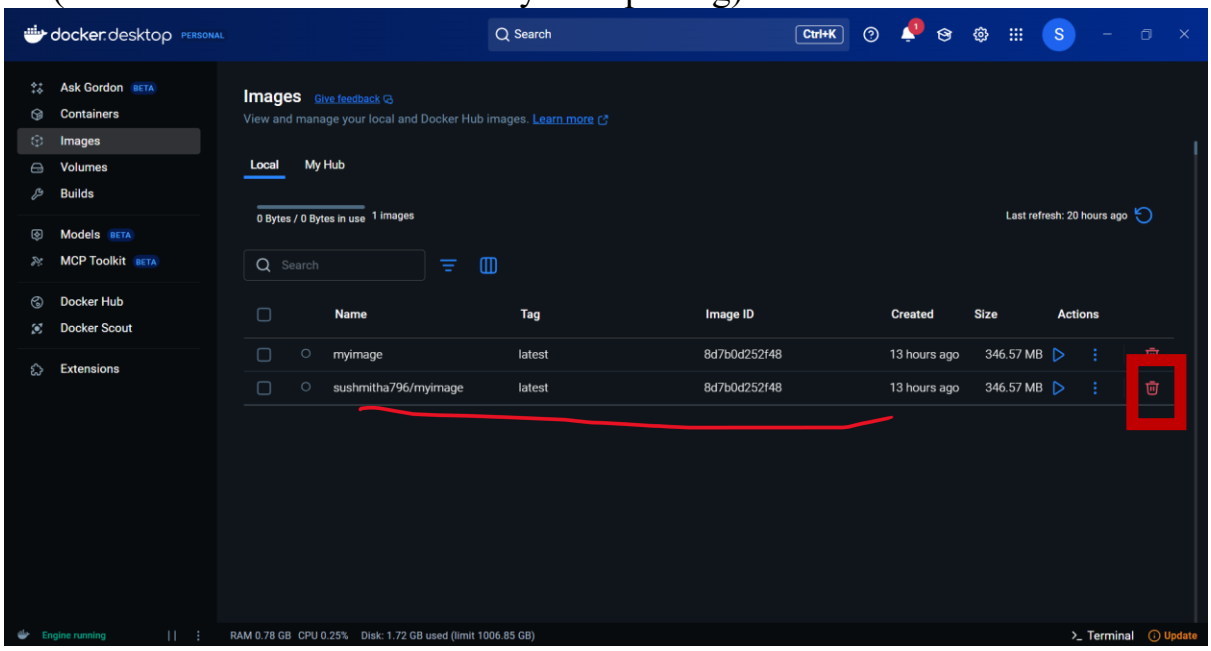


- Your hub will be opened, you can see the image that you pushed.

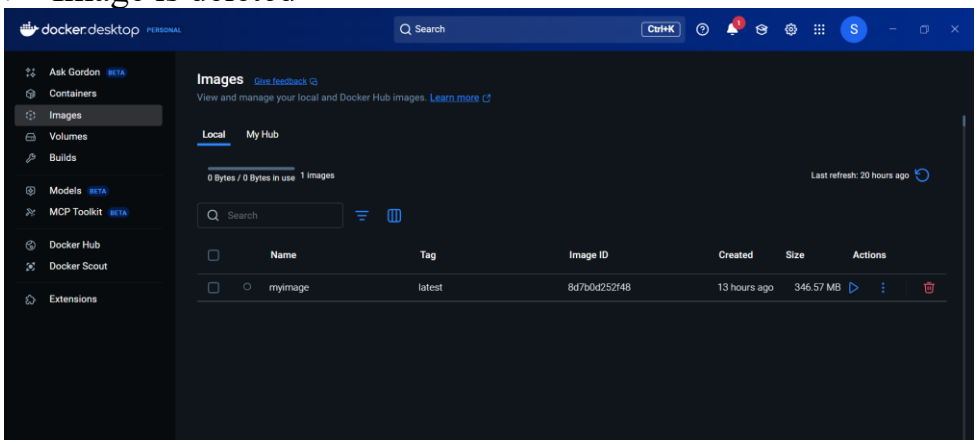


PULL AN IMAGE FROM DOCKER HUB

- Docker desktop app -> images -> delete the image which was created during pushing
  (so that it will be easier to verify after pulling)



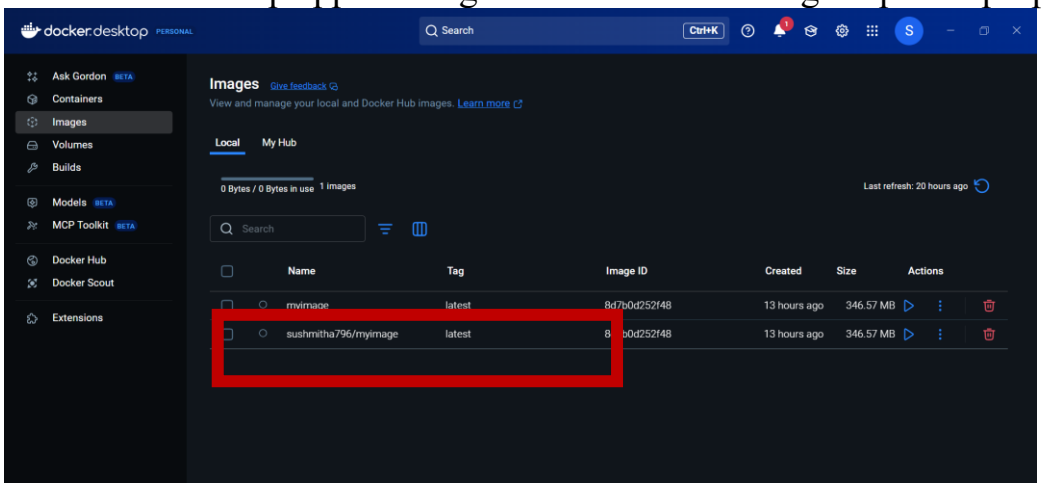
- Image is deleted



- Goto command prompt
- Use the command,
- docker pull <dockerhub-username>/<repository-name>:<tag> to pull the image from docker hub
  Example: docker pull sushmitha796/myimage:latest

```
C:\Users\Sushmitha>docker pull sushmitha796/myimage:latest
latest: Pulling from sushmitha796/myimage
Digest: sha256:b440e193a1d15044f813cf48ca622cb7c0b9ce3b16ad2d4e1942c5a534f1be34
Status: Downloaded newer image for sushmitha796/myimage:latest
docker.io/sushmitha796/myimage:latest
```

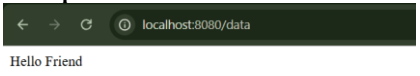
- Docker desktop app -> images -> check if the image is pulled properly



- To run the pulled image, use
  docker run -p <host-port>:<container-port> <username>/<image-name>:<tag>
  Example 1: docker run -p 8080:8080 sushmitha796/myimage:latest
  Or
  Example 2: docker run -p 3000:8080 sushmitha796/myimage:latest
  This means any request to localhost:8080 will be forwarded to port 3000 inside the container. You can also map the same port on both sides using -p 8080:8080, which is helpful when you want consistent port access. If you don't use -p, the container's ports will remain isolated and not accessible from outside

```
C:\Users\Sushmitha>docker run -p 8080:8080 sushmitha796/myimage:latest

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

 :: Spring Boot ::                (v3.5.4)

2025-07-29T06:08:53.098Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : Starting DockerApp1Application v0.0.1-SNAPSH
OT using Java 17-ea with PID 1 (/dockerappimg.jar started by root in /)
2025-07-29T06:08:53.104Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : No active profile set, falling back to 1 def
ault profile: "default"
2025-07-29T06:08:55.139Z  INFO 1 --- [DockerApp-1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2025-07-29T06:08:55.174Z  INFO 1 --- [DockerApp-1] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2025-07-29T06:08:55.175Z  INFO 1 --- [DockerApp-1] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.1
.43]
2025-07-29T06:08:55.234Z  INFO 1 --- [DockerApp-1] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationC
ontext
2025-07-29T06:08:55.239Z  INFO 1 --- [DockerApp-1] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization c
ompleted in 1924 ms
2025-07-29T06:08:55.989Z  INFO 1 --- [DockerApp-1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with cont
ext path '/'
2025-07-29T06:08:56.028Z  INFO 1 --- [DockerApp-1] [           main] com.DockerApp1Application               : Started DockerApp1Application in 3.823 secon
ds (process running for 4.813)
```

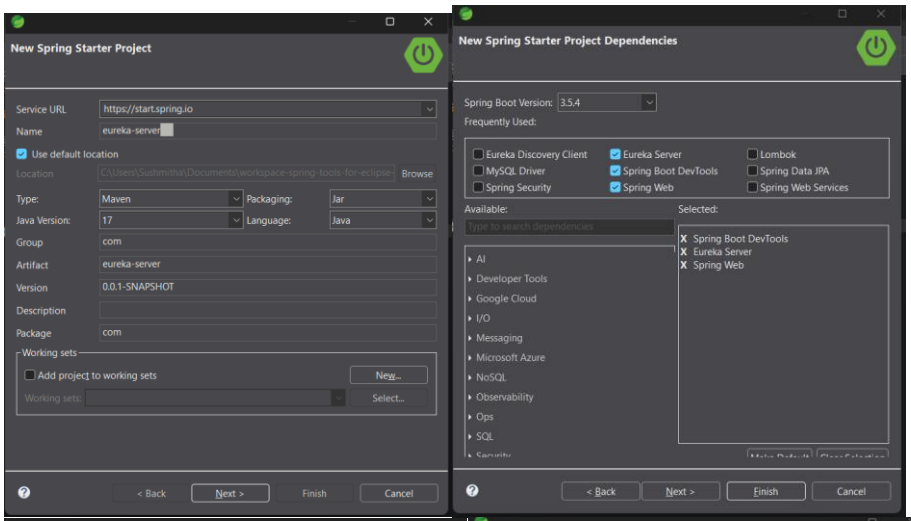
- Goto to browser
- Paste this http://localhost:8080/data (if you use -p 8080:8080 in previous step) in url box and press enter
  (Or)
- Paste this http://localhost:3000/data (if you use -p 3000:8080 in previous step) in url box and press enter

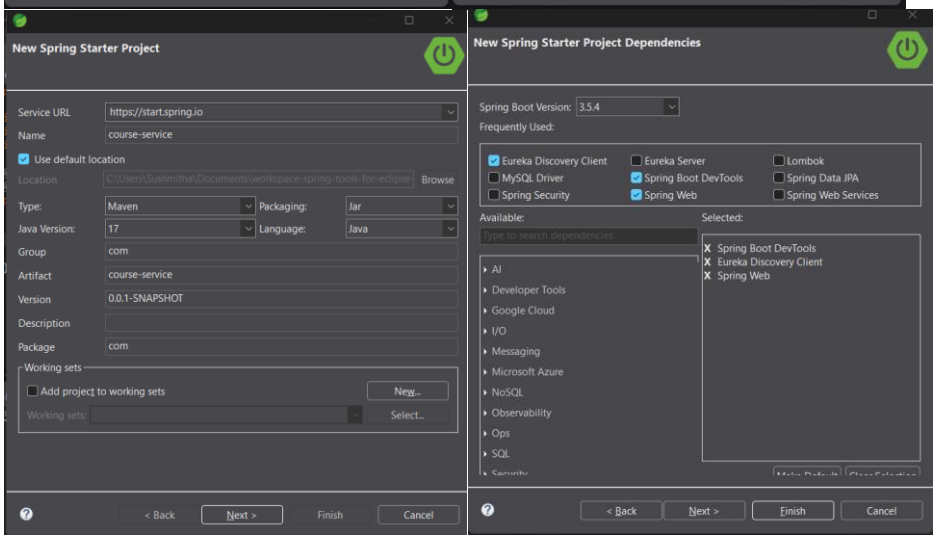


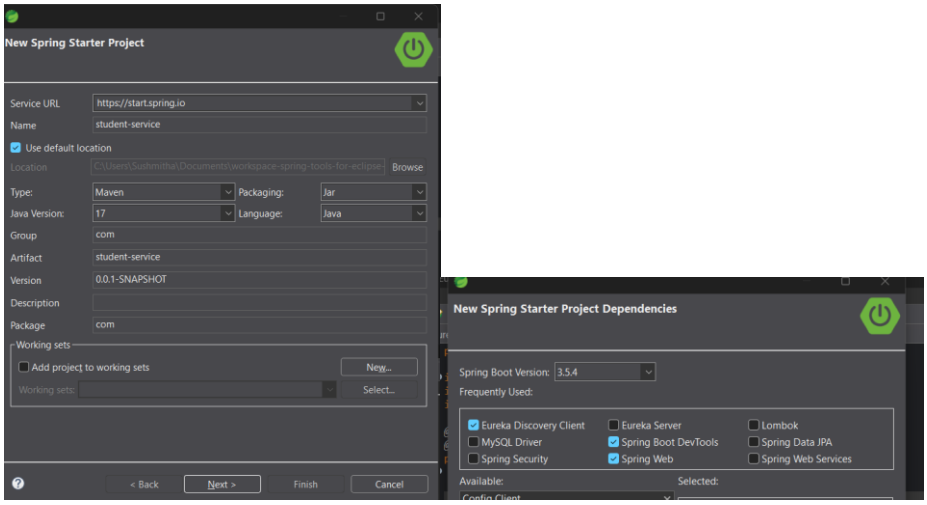
Hello Friend

MICROSERVICES

- Create three Spring strater projects
  1. eureka-server
  2. student-service
  3. course-service

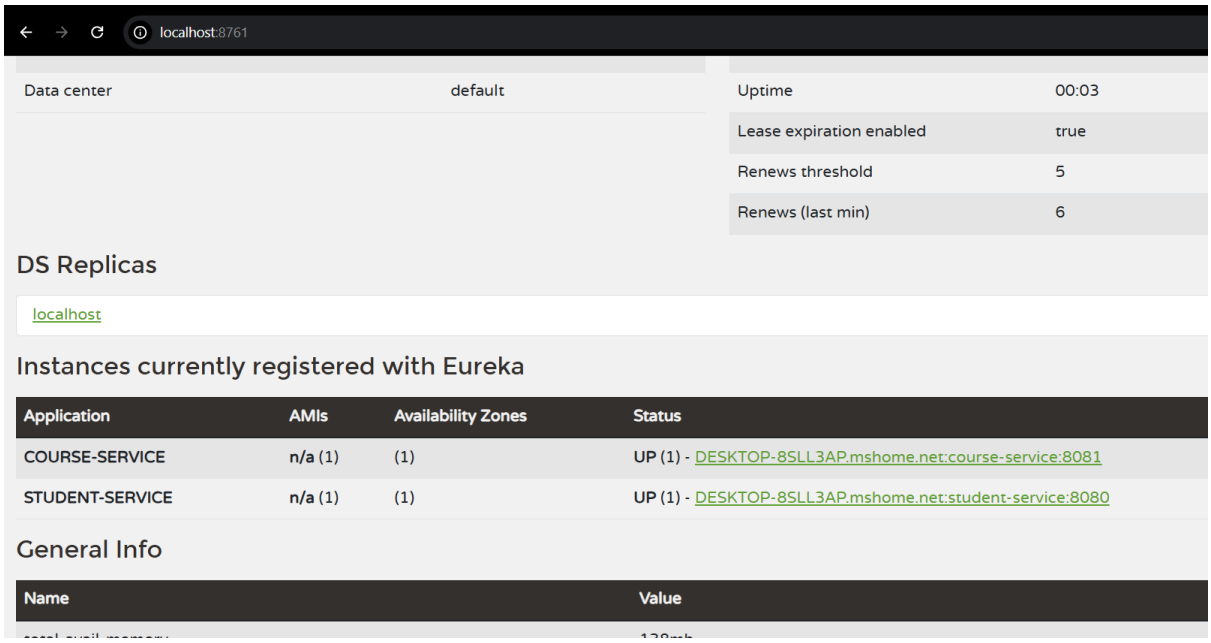




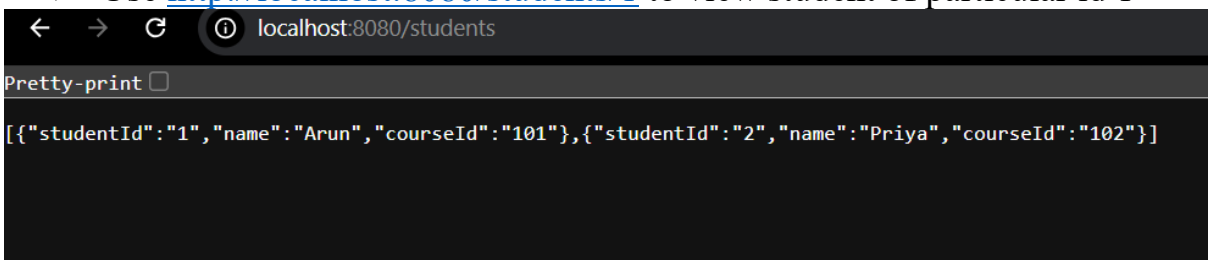


Get the code for eureka-server, course-service, student-service from the github link:
https://github.com/SushmithaSaravanan27/microservice

- Run the projects in the order given below:
  1. eureka-server
  2. course-service
  3. student-service
- Goto browser, http://localhost:8761 run this to view the Eureka server dashboard
- After running course-service, student-service, it will be visible in eureka dashboard

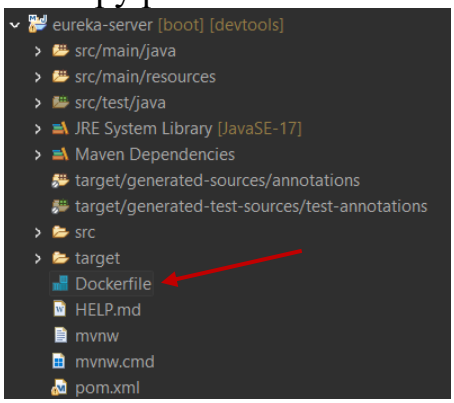


- Goto postman and insert values
- Use http://localhost:8080/students to view all students
- Use http://localhost:8080/students/1 to view student of particular id 1



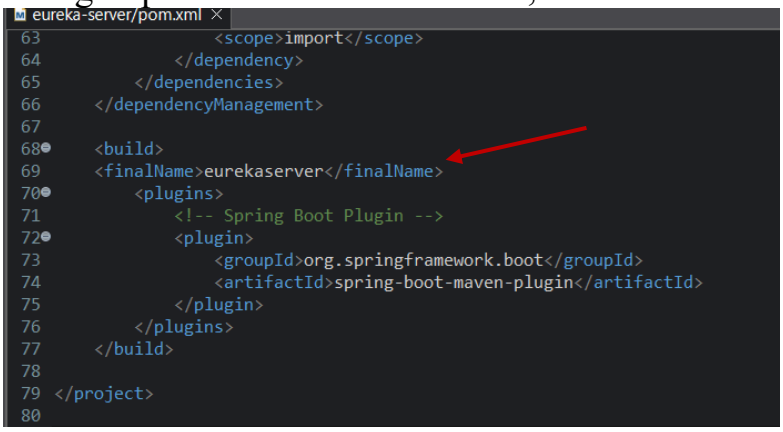
DOCKERIZING MICROSERVICES

- Goto **eureka-server** ->create a file and name it as Dockerfile
- Copy paste and save the code given below in the Dockerfile

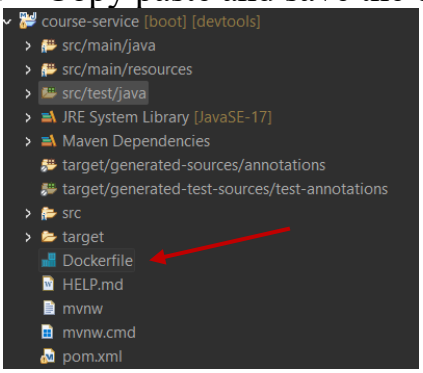



```
FROM openjdk:17-jdk-alpine
WORKDIR /app
COPY target/eurekaserver.jar eureka-server.jar
EXPOSE 8761
ENTRYPOINT ["java", "-jar", "eureka-server.jar"]
```

- Add this tag in pom.xml of eureka-server, inside <build>..</build> tag

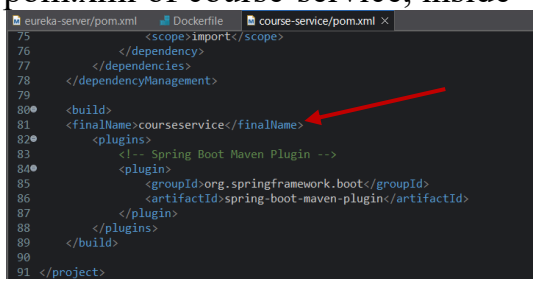


- Goto **course-service** ->create a file and name it as Dockerfile
- Copy paste and save the code given below in the Dockerfile

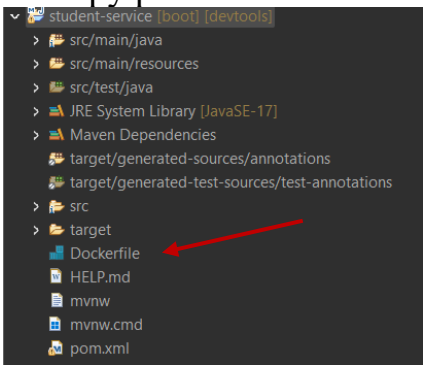



```
FROM openjdk:17-jdk-alpine
WORKDIR /app
COPY target/courseservice.jar course-service.jar
EXPOSE 8081
ENTRYPOINT ["java", "-jar", "course-service.jar"]
```

- Add this tag in pom.xml of course-service, inside <build>..</build> tag



- Goto **student-service** ->create a file and name it as Dockerfile
- Copy paste and save the code given below in the Dockerfile

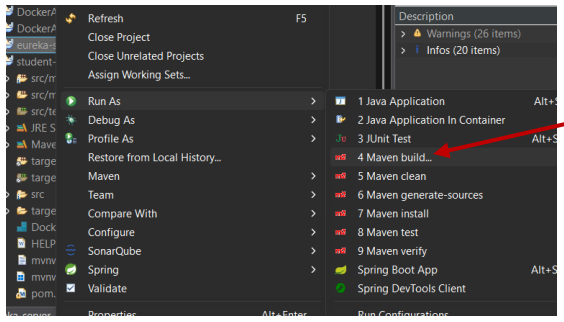


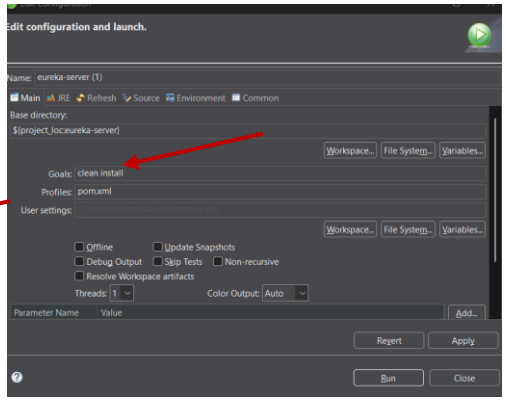

```
FROM openjdk:17-jdk-alpine
WORKDIR /app
COPY target/studentservice.jar student-service.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "student-service.jar"]
```

- Add this tag in pom.xml of student-service, inside <build>..</build> tag

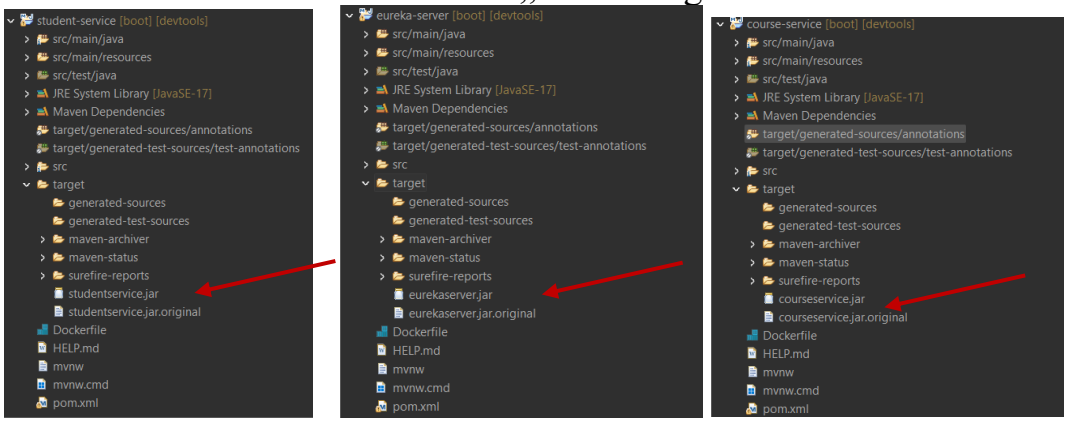
```
81                  <scope>import</scope>
82              </dependency>
83          </dependencies>
84      </dependencyManagement>
85
86      <build>
87      <finalName>studentservice</finalName>
88          <plugins>
89              <!-- Spring Boot Plugin -->
90              <plugin>
91                  <groupId>org.springframework.boot</groupId>
92                  <artifactId>spring-boot-maven-plugin</artifactI
93              </plugin>
94          </plugins>
95      </build>
96
97  </project>
98
```

- Right click on the eureka-server
- Goto RunAs -> maven build
- Give “clean install” in Goals
- Do the same maven build for student-service and course-service

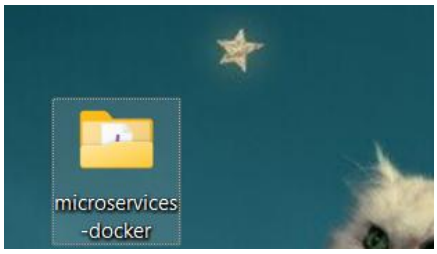


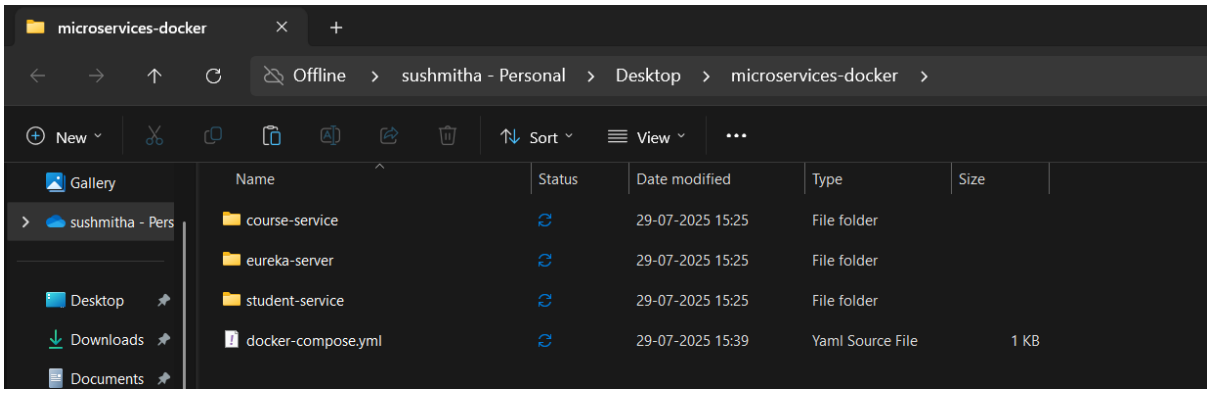


- After mave build step,, you will notice the jar files created in eureka-server, student-service and course-service,, inside target folder



- Go to your desired location (e.g., Desktop).
- Right-click → New → Folder.
- Name the folder as *microservices-docker*.
- Goto the location where the eureka-server, student-service, course-service projects are actually located in your system.
- Copy all the three folders and paste inside *microservices-docker* folder
- Inside the *microservices-docker* folder:
  - → Right-click → New → Text Document.
  - → Rename it to: docker-compose.yml
    (Make sure it doesn't remain as .txt or textdocument)
  - → Copy, paste and save the code given below, in docker-compose.yml file





docker-compose.yml

```
version: '3.8'

services:
  eureka-server:
    build: ./eureka-server
    ports:
      - "8761:8761"
    container_name: eureka-server
    networks:
      - spring-cloud-net

  course-service:
    build: ./course-service
    ports:
      - "8082:8080"    # HostPort:ContainerPort
    container_name: course-service
    depends_on:
      - eureka-server
    networks:
      - spring-cloud-net

  student-service:
    build: ./student-service
    ports:
      - "8083:8080"    # HostPort:ContainerPort
    container_name: student-service
    depends_on:
      - eureka-server
    networks:
      - spring-cloud-net

networks:
  spring-cloud-net:
    driver: bridge
```

- Goto command prompt
- To create image for eureka-server, first get into the path where eureka-server exists. Use the command *cd path of eureka-server*
  Example: cd C:\Users\Sushmitha\Documents\workspace\erueka-server
- To create image, Use the command *docker build -t anyimagename*
  Example: docker build -t eureka-server-img .

```
C:\Users\Sushmitha>cd C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\eureka-server

C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\eureka-server>docker build -t eureka-server-img .
[+] Building 7.6s (9/9) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                               0.1s
 => => transferring dockerfile: 348B                                                                                               0.0s
 => [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine                                                           4.2s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                                                     0.0s
 => [internal] load .dockerignore                                                                                                  0.0s
 => => transferring context: 2B                                                                                                    0.0s
 => [internal] load build context                                                                                                  2.6s
 => => transferring context: 59.00MB                                                                                               2.6s
 => CACHED [1/3] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c966a748515423890 2f2f25e9dbd9784383d81  0.0s
```

- To create image for student-service, first get into the path where student-service exists. Use the command *cd path of student-service*
  Example: cd C:\Users\Sushmitha\Documents\workspace\student-service
- To create image, Use the command *docker build -t anyimagename*
  Example: docker build -t student-service-img .

```
C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\course-service>cd C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\student-service

C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\student-service>docker build -t student-service-img .
[+] Building 5.6s (8/8) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                               0.0s
 => => transferring dockerfile: 354B                                                                                               0.0s
 => [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine                                                           1.7s
 => [internal] load .dockerignore                                                                                                  0.0s
 => => transferring context: 2B                                                                                                    0.0s
```

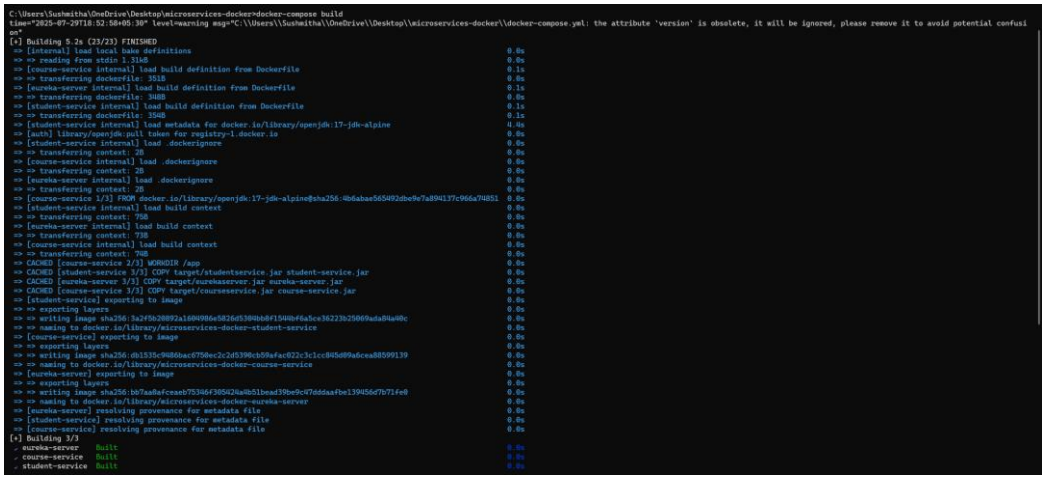
- To create image for course-service, first get into the path where course-service exists. Use the command *cd path of course-service*
  Example: cd C:\Users\Sushmitha\Documents\workspace\course-service
- To create image, Use the command *docker build -t anyimagename*
  Example: docker build -t course-service-img .

```
C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\eureka-server>cd C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE

C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\course-service>docker build -t course-service-img .
[+] Building 5.3s (8/8) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                              0.0s
 => => transferring dockerfile: 351B                                                                                              0.0s
 => [internal] load metadata for docker.io/library/openjdk:17-jdk-alpine                                                          1.2s
 => [internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 2B                                                                                                   0.0s
 => [1/3] FROM docker.io/library/openjdk:17-jdk-alpine@sha256:4b6abae565492dbe9e7a894137c966a7485154238902f2f25e9dbd9784383d81    0.0s
 => CACHED [2/3] WORKDIR /app                                                                                                     0.0s
 => [internal] load build context                                                                                                 3.4s
 => => transferring context: 74.38MB                                                                                              3.4s
 => [3/3] COPY target/courseservice.jar course-service.jar                                                                        0.3s
```

- Use *docker network create <networkname>*
  Example: docker network create ourmicroservices
- This command creates a custom Docker network named ourmicroservices.
- It allows multiple containers (like Eureka, Student Service, Course Service) to communicate internally

```
C:\Users\Sushmitha\Documents\workspace-spring-tools-for-eclipse-4.30.0.RELEASE\student-service>docker network create ourmicroservices
f3097b36aad70b40829f65d82563f1646d669076f2999cc347b085c8f41c3f5b
```

- Use the command *docker-compose build* to build Docker images for all services



- *Use docker-compose up -d* to start your containers in the background (detached mode)

- Use *docker ps* command to show all currently running containers on your system.

```
C:\Users\Sushmitha\OneDrive\Desktop\microservices-docker>docker ps
CONTAINER ID   IMAGE                                  COMMAND                  CREATED       STATUS          PORTS                                              NAMES
83ec4f74b3327  microservices-docker-course-service    "java -jar course-se…"   3 hours ago   Up 25 seconds   0.0.0.0:8082->8080/tcp, [::]:8082->8080/tcp        course-service
d8bd6b37bee0   microservices-docker-student-service   "java -jar student-s…"   3 hours ago   Up 25 seconds   0.0.0.0:8083->8080/tcp, [::]:8083->8080/tcp        student-service
c7c4bd05591b   microservices-docker-eureka-server     "java -jar eureka-se…"   3 hours ago   Up 25 seconds   0.0.0.0:8761->8761/tcp, [::]:8761->8761/tcp        eureka-server
```