DevOps Assignment-2

1.Pull any image from the docker hub, create its container, and execute it showing the output.

Ans:

Docker:

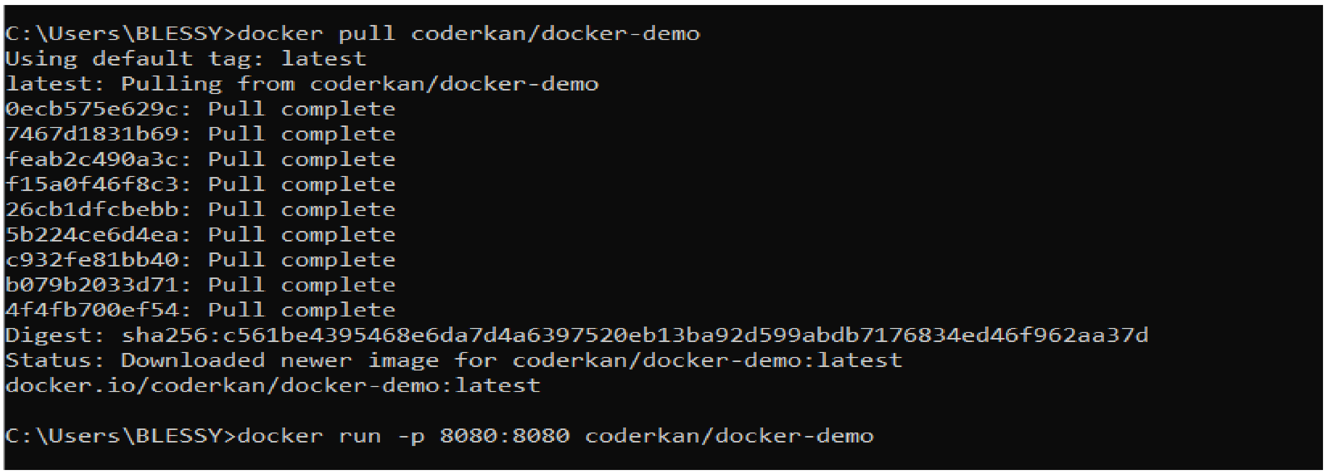
- ➢ Docker is a tool of DevOps.
- ➢ Docker is an open-source centralized platform designed to create,deploy and run
- ➢ applications.
- ➢ Docker uses container on the host's OS to run applications.
- ➢ It allows applications to use the same linux kernel as a system on the host computer
- ➢ rather than creating a whole virtual operating system.
- ➢ Containers ensure that our application works in any environment like development,test or production.
- ➢ Docker includes componnts such as Docker client,Docker server,Docker Machine,Docker Hub,Docker Composes etc

docker pull image:

- docker pull command used to download the images from any registry,it might be a public registry or private registry.
- **Syntax:**

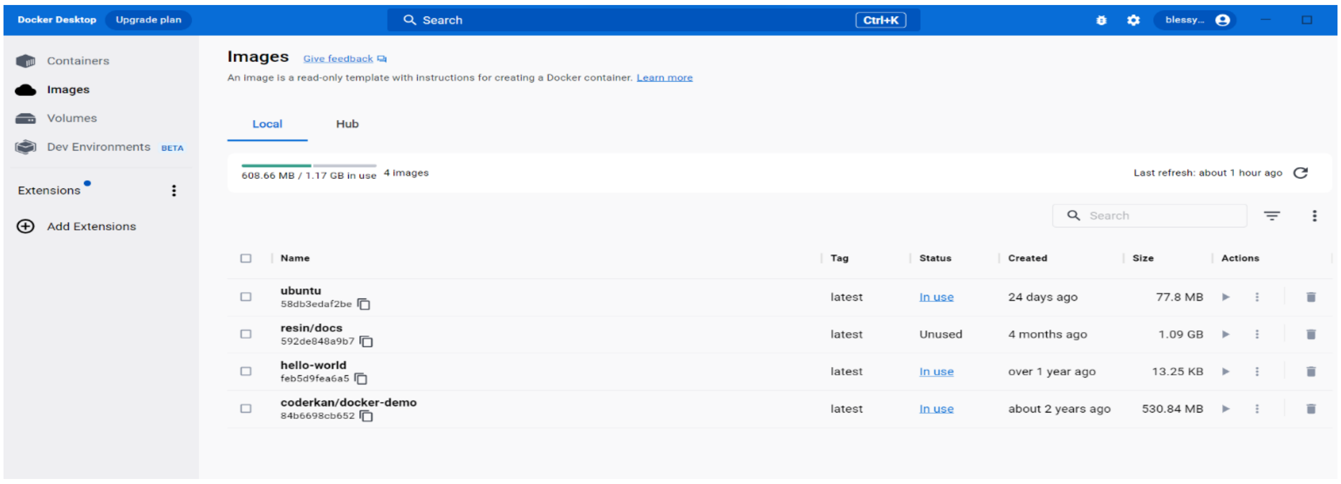
  docker pull <image name>

When we run the pull command from the command line, it first checks locally or on the host for the images and if the image does not exist locally then the Docker daemon connects to the public registry 'hub.docker.com' if there is no private registry mentioned in the 'daemon.json' file and pulls the Docker image mentioned in the command and if it finds the image locally then it checks for the updates and downloads newer version of the image.



```
C:\Users\BLESSY>docker pull coderkan/docker-demo
Using default tag: latest
latest: Pulling from coderkan/docker-demo
0ecb575e629c: Pull complete
7467d1831b69: Pull complete
feab2c490a3c: Pull complete
f15a0f46f8c3: Pull complete
26cb1dfcbebb: Pull complete
5b224ce6d4ea: Pull complete
c932fe81bb40: Pull complete
b079b2033d71: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:c561be4395468e6da7d4a6397520eb13ba92d599abdb7176834ed46f962aa37d
Status: Downloaded newer image for coderkan/docker-demo:latest
docker.io/coderkan/docker-demo:latest

C:\Users\BLESSY>docker run -p 8080:8080 coderkan/docker-demo
```



Images Give feedback

An image is a read-only template with instructions for creating a Docker container. Learn more

608.66 MB / 1.17 GB in use 4 images

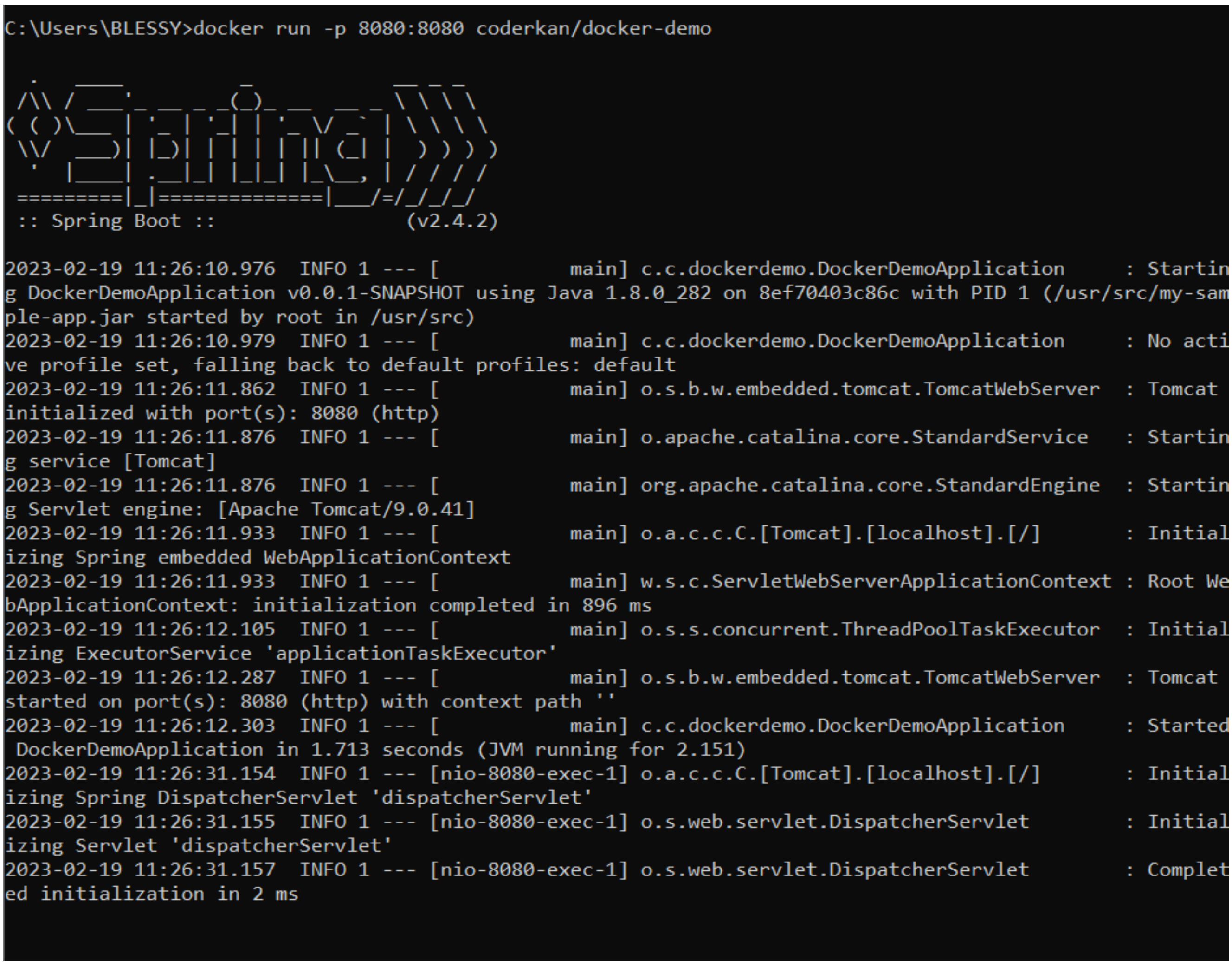
Last refresh: about 1 hour ago

Name	Tag	Status	Created	Size	Actions
ubuntu 58db3edaf2be	latest	In use	24 days ago	77.8 MB	
resin/docs 592de848a9b7	latest	Unused	4 months ago	1.09 GB	
hello-world feb5d9fea6a5	latest	In use	over 1 year ago	13.25 KB	
coderkan/docker-demo 84b6698cb652	latest	In use	about 2 years ago	530.84 MB	

Docker run:

Docker run command is used to create a container and execute or run the container in by using the command **docker run image.**

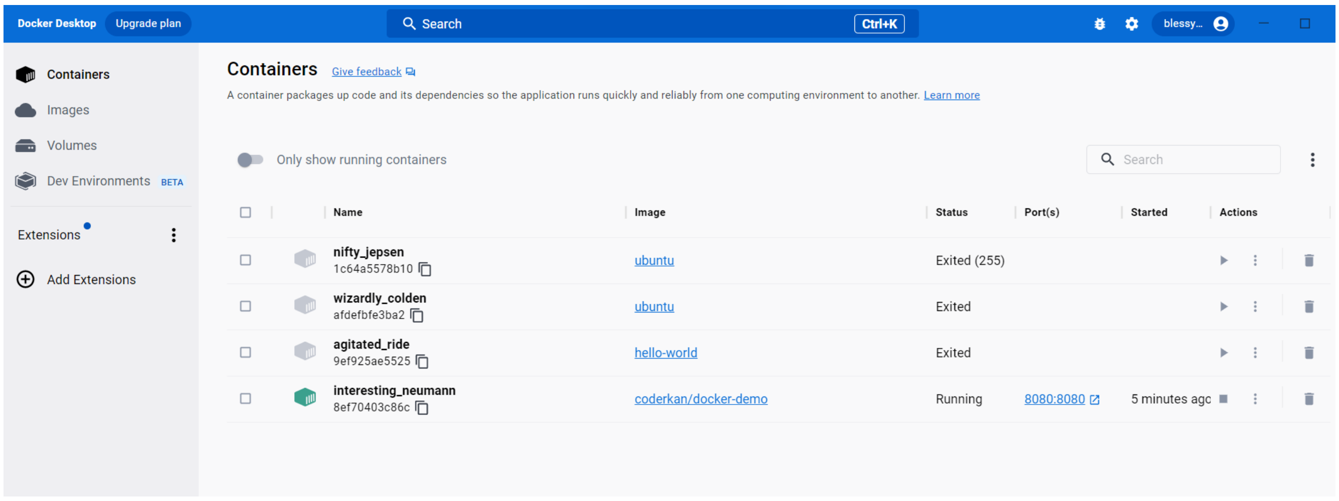
The docker run command first creates a writeable container layer over the specified image, and then starts it using the specified command.



```
C:\Users\BLESSY>docker run -p 8080:8080 coderkan/docker-demo

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.2)

2023-02-19 11:26:10.976  INFO 1 --- [           main] c.c.dockerdemo.DockerDemoApplication     : Startin
g DockerDemoApplication v0.0.1-SNAPSHOT using Java 1.8.0_282 on 8ef70403c86c with PID 1 (/usr/src/my-sam
ple-app.jar started by root in /usr/src)
2023-02-19 11:26:10.979  INFO 1 --- [           main] c.c.dockerdemo.DockerDemoApplication     : No acti
ve profile set, falling back to default profiles: default
2023-02-19 11:26:11.862  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat
initialized with port(s): 8080 (http)
2023-02-19 11:26:11.876  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Startin
g service [Tomcat]
2023-02-19 11:26:11.876  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Startin
g Servlet engine: [Apache Tomcat/9.0.41]
2023-02-19 11:26:11.933  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initial
izing Spring embedded WebApplicationContext
2023-02-19 11:26:11.933  INFO 1 --- [           main] w.s.c.ServletWebServerApplicationContext : Root We
bApplicationContext: initialization completed in 896 ms
2023-02-19 11:26:12.105  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initial
izing ExecutorService 'applicationTaskExecutor'
2023-02-19 11:26:12.287  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat
started on port(s): 8080 (http) with context path ''
2023-02-19 11:26:12.303  INFO 1 --- [           main] c.c.dockerdemo.DockerDemoApplication     : Started
 DockerDemoApplication in 1.713 seconds (JVM running for 2.151)
2023-02-19 11:26:31.154  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initial
izing Spring DispatcherServlet 'dispatcherServlet'
2023-02-19 11:26:31.155  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initial
izing Servlet 'dispatcherServlet'
2023-02-19 11:26:31.157  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Complet
ed initialization in 2 ms
```

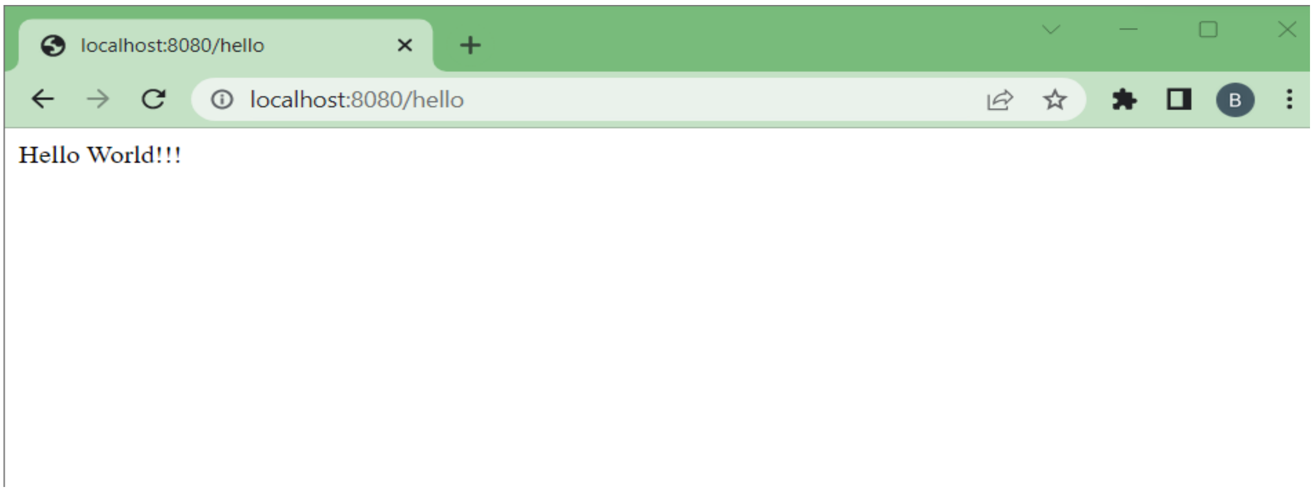


Containers Give feedback

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

Only show running containers

Name	Image	Status	Port(s)	Started	Actions
nifty_jepsen 1c64a5578b10	ubuntu	Exited (255)			
wizardly_colden afdefbfe3ba2	ubuntu	Exited			
agitated_ride 9ef925ae5525	hello-world	Exited			
interesting_neumann 8ef70403c86c	coderkan/docker-demo	Running	8080:8080	5 minutes agc	



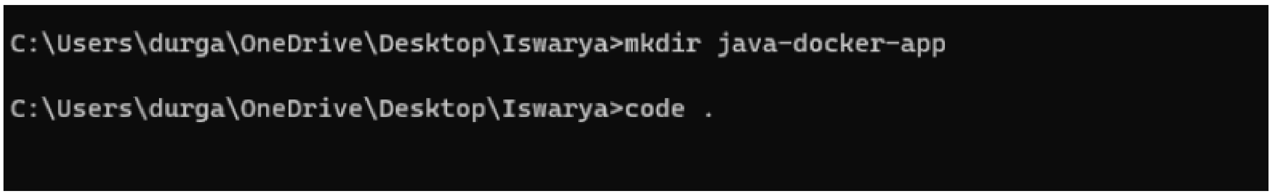
```
Hello World!!!
```

2.Create the basic java application, generate its image with necessary files, and execute it with docker.

Ans:

Step -1:

First create directory using command mkdir. It is used to organize the files. To open visual studio code execute code .



```
C:\Users\durga\OneDrive\Desktop\Iswarya>mkdir java-docker-app

C:\Users\durga\OneDrive\Desktop\Iswarya>code .
```

Step-2:

Create a Hello.java and Dockerfile with the code

<u>Hello.java:</u>

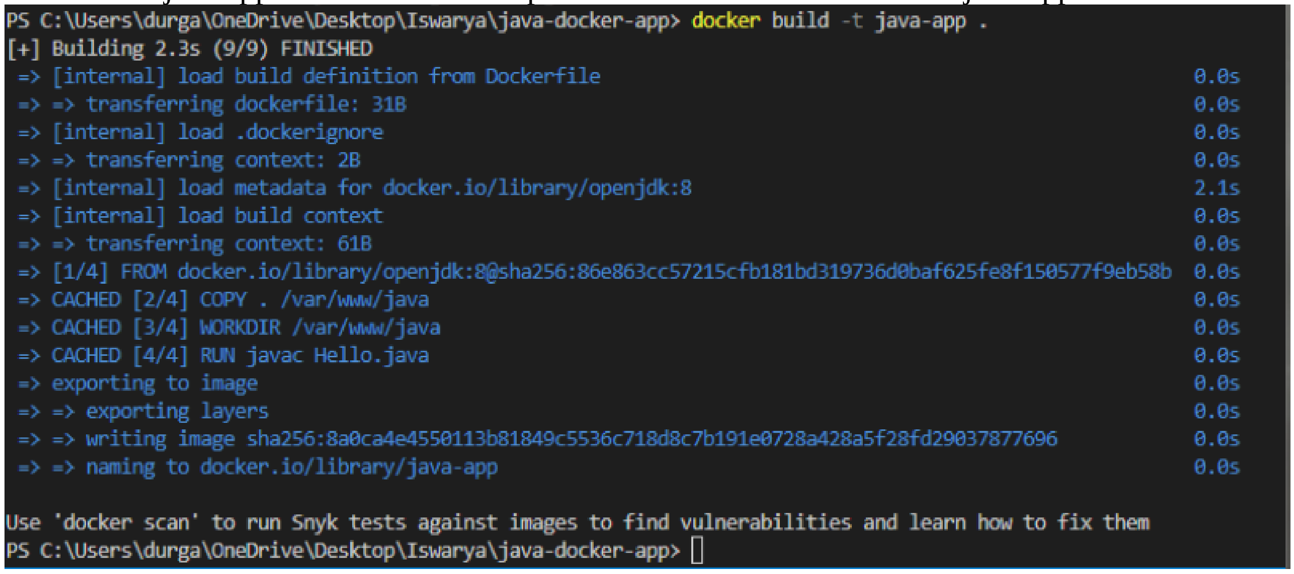
```
 class Hello{
 public static void main(String[] args){
System.out.println("This is java app \n by using Docker");
 }
 }
```

<u>Dockerfile:</u>

```
 FROM openjdk:8
 COPY . /var/www/java
 WORKDIR /var/www/java
 RUN javac Hello.java
CMD ["java", "Hello"]
```

Step3:

Now build a java application with the help of the command docker build -t java-app .

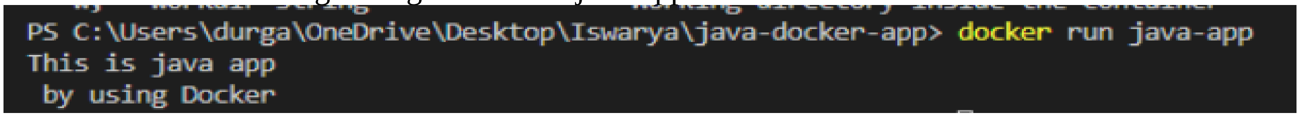


```
PS C:\Users\durga\OneDrive\Desktop\Iswarya\java-docker-app> docker build -t java-app .
[+] Building 2.3s (9/9) FINISHED
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 31B                                                           0.0s
 => [internal] load .dockerignore                                                             0.0s
 => => transferring context: 2B                                                               0.0s
 => [internal] load metadata for docker.io/library/openjdk:8                                  2.1s
 => [internal] load build context                                                             0.0s
 => => transferring context: 61B                                                              0.0s
 => [1/4] FROM docker.io/library/openjdk:8@sha256:86e863cc57215cfb181bd319736d0baf625fe8f150577f9eb58b  0.0s
 => CACHED [2/4] COPY . /var/www/java                                                         0.0s
 => CACHED [3/4] WORKDIR /var/www/java                                                        0.0s
 => CACHED [4/4] RUN javac Hello.java                                                         0.0s
 => exporting to image                                                                        0.0s
 => => exporting layers                                                                       0.0s
 => => writing image sha256:8a0ca4e4550113b81849c5536c718d8c7b191e0728a428a5f28fd29037877696  0.0s
 => => naming to docker.io/library/java-app                                                   0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\durga\OneDrive\Desktop\Iswarya\java-docker-app>
```

Step-4:

Now run the docker image using docker run java-app.



```
PS C:\Users\durga\OneDrive\Desktop\Iswarya\java-docker-app> docker run java-app
This is java app
 by using Docker
```