Experiment 7: Develop a simple containerized application using Docker.

```
P c@DESKTOP-TEKQI08 MINGW64 ~ (master)
$ cd e:

P c@DESKTOP-TEKQI08 MINGW64 /e
$ mkdir javadocker

P c@DESKTOP-TEKQI08 MINGW64 /e
$ cd javadocker

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker
$ git init
Initialized empty Git repository in E:/javadocker/.git/

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ git config user.email "gujjuladeepak@gmail.com"

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ git config user.name "Deepak"

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ vim Sample.java
```

```
public class Sample
{
    public static void main(String args[])
    {
        System.out.println("Gujjula Deepak");
    }
}
```

```
public class Sample
{
        public static void main(String args[])
        {
                System.out.println("Gujjula Deepak");
        }
}
```

```
P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ touch Dockerfile

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ vim Dockerfile
```

Dockerfile should contain the following code

Use an official OpenJDK runtime as a parent image
FROM openjdk:11-jdk-slim
Set the working directory inside the container
WORKDIR /app
Copy the current directory contents into the container at /app
COPY . .
Compile the Java program
RUN javac Sample.java
Command to run the program
CMD ["java", "Sample"]

```
# Use an official OpenJDK runtime as a parent image
FROM openjdk:11-jdk-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . .

# Compile the Java program
RUN javac Sample.java

# Command to run the program
CMD ["java", "Sample"]
```

```
P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ git add .
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'Sample.java', LF will be replaced by CRLF the next time Git touches it

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ git commit -m "java Docker test"
[master (root-commit) c31db75] java Docker test
 2 files changed, 21 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 Sample.java
```

```
P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ git remote add origin https://github.com/deepak574/javadockertest.git

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (master)
$ git branch -M main

P c@DESKTOP-TEKQI08 MINGW64 /e/javadocker (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 569 bytes | 189.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/deepak574/javadockertest.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Open docker desktop

Open cmd

```
C:\Users\P c>docker --version
Docker version 27.2.0, build 3ab4256
```

docker build -t javadockerapp https://github.com/deepak574/javadockertest.git

```
C:\Users\P c>docker build -t javadockerapp https://github.com/deepak574/javadockertest.git
[+] Building 10.9s (8/8) FINISHED                                                    docker:desktop-linux
 => [internal] load git source https://github.com/deepak574/javadockertest.git                       4.2s
 => [internal] load metadata for docker.io/library/openjdk:11-jdk-slim                               2.5s
 => [auth] library/openjdk:pull token for registry-1.docker.io                                       0.0s
 => [1/4] FROM docker.io/library/openjdk:11-jdk-slim@sha256:868a4f2151d38ba6a09870cec584346a5edc8e9b  0.1s
 => => resolve docker.io/library/openjdk:11-jdk-slim@sha256:868a4f2151d38ba6a09870cec584346a5edc8e9b  0.1s
 => CACHED [2/4] WORKDIR /app                                                                        0.0s
 => [3/4] COPY . .                                                                                   0.1s
 => [4/4] RUN javac Sample.java                                                                      3.1s
 => exporting to image                                                                               0.5s
 => => exporting layers                                                                              0.2s
 => => exporting manifest sha256:d92ceeebe6280e3c3bf967ced854107a45551c115376ddf0b30579bbb8c7d78b    0.0s
 => => exporting config sha256:15a0acb0e4454f2b533184ab28ebbda437867441aaf94d1de1b78a4f65bac217      0.0s
 => => exporting attestation manifest sha256:1e3cc2a1416e22789c266de6aa53663f32c3d68d86f25fd16a83874  0.0s
 => => exporting manifest list sha256:97aa58d4f5b453978920de3d5e9bb16e095db27bf009cc2942667ac1aadb24  0.0s
 => => naming to docker.io/library/javadockerapp:latest                                              0.0s
 => => unpacking to docker.io/library/javadockerapp:latest                                           0.1s

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

docker images

```
C:\Users\P c>docker images
REPOSITORY      TAG       IMAGE ID       CREATED         SIZE
javadockerapp   latest    97aa58d4f5b4   9 seconds ago   673MB
```

Docker run javadockerapp

```
C:\Users\P c>docker run javadockerapp
Gujjula Deepak
```

Open browser and login to docker hub
Push the Docker Image to Docker Hub

Login to Docker Hub from the Command Line

First, you need to log in to your Docker Hub account from your terminal.

```
C:\Users\P c>docker login
Authenticating with existing credentials...
Login Succeeded
```

Tag Your Docker Image for Docker Hub

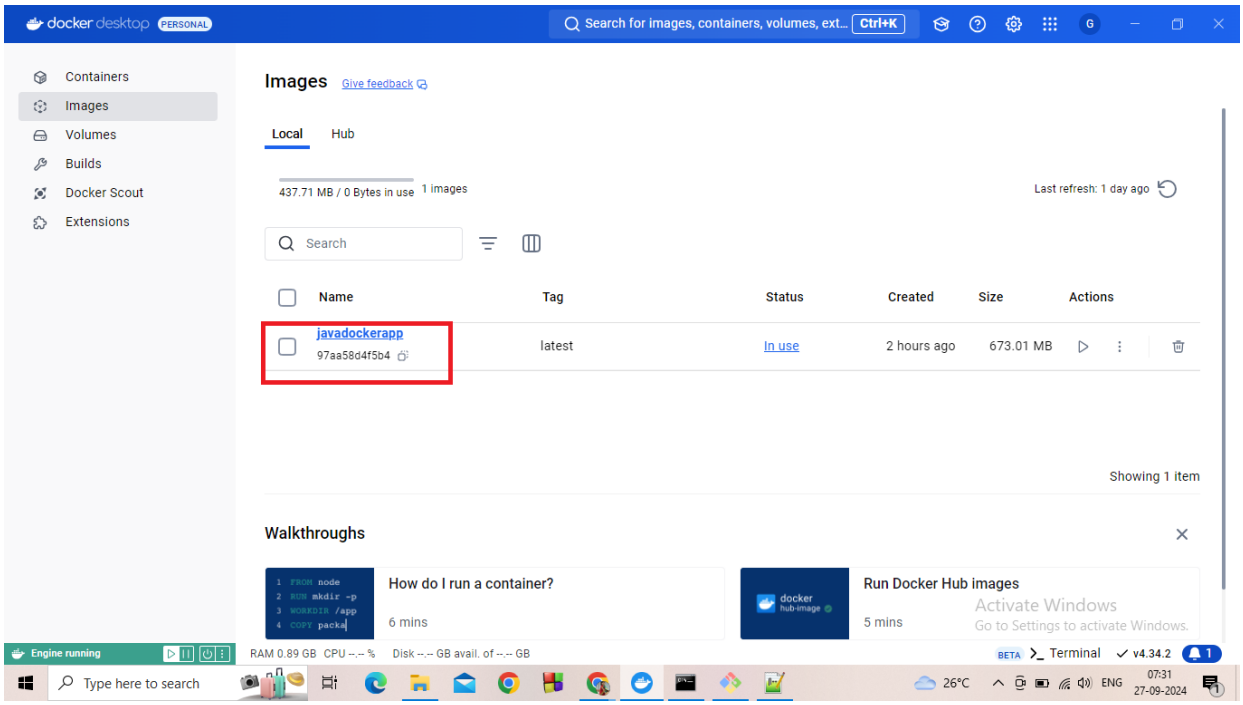
Docker images need to be tagged with your Docker Hub username and the repository name you will use.

```
C:\Users\P c>docker tag javadockerapp gujjuladeepak/javadockerapp:latest
```

Push the Docker Image to Docker Hub

After tagging the image, push it to Docker Hub using the following command:

```
C:\Users\P c>docker push gujjuladeepak/javadockerapp:latest
The push refers to repository [docker.io/gujjuladeepak/javadockerapp]
e4dc918eef66: Pushed
69e15dccd787: Pushed
1efc276f4ff9: Pushed
9421bb337be6: Pushed
a2f2f93da482: Pushed
c1a41d8e9e7a: Pushed
9e58bcf4d106: Pushed
12cca292b13c: Pushed
latest: digest: sha256:97aa58d4f5b453978920de3d5e9bb16e095db27bf009cc2942667ac1aadb2431 size: 856
```



Step 5: Pull the Docker Image from Docker Hub

Now that the image is on Docker Hub, you can pull it from anywhere, such as another server, your local machine, or a cloud environment.

1. Pull the Docker Image

Use the docker pull command to pull the image from Docker Hub:

Example

```
docker pull gujjuladeepak/javadockerapp:latest
```

Docker will download the image from Docker Hub to your local machine or server.

2. Run the Docker Container

Once the image is pulled, you can run it just like any other Docker container:

```
docker run gujjuladeepak/javadockerapp:latest
```

This will execute the Java program inside the container, and you should see the output: