Lesson 02 Demo 01

Create an Image for Java App

Objective: To create a Docker image for a Java application in the Eclipse IDE using OpenJDK as the base image to enable easy deployment and distribution of the application

Tool required: Eclipse IDE and Docker

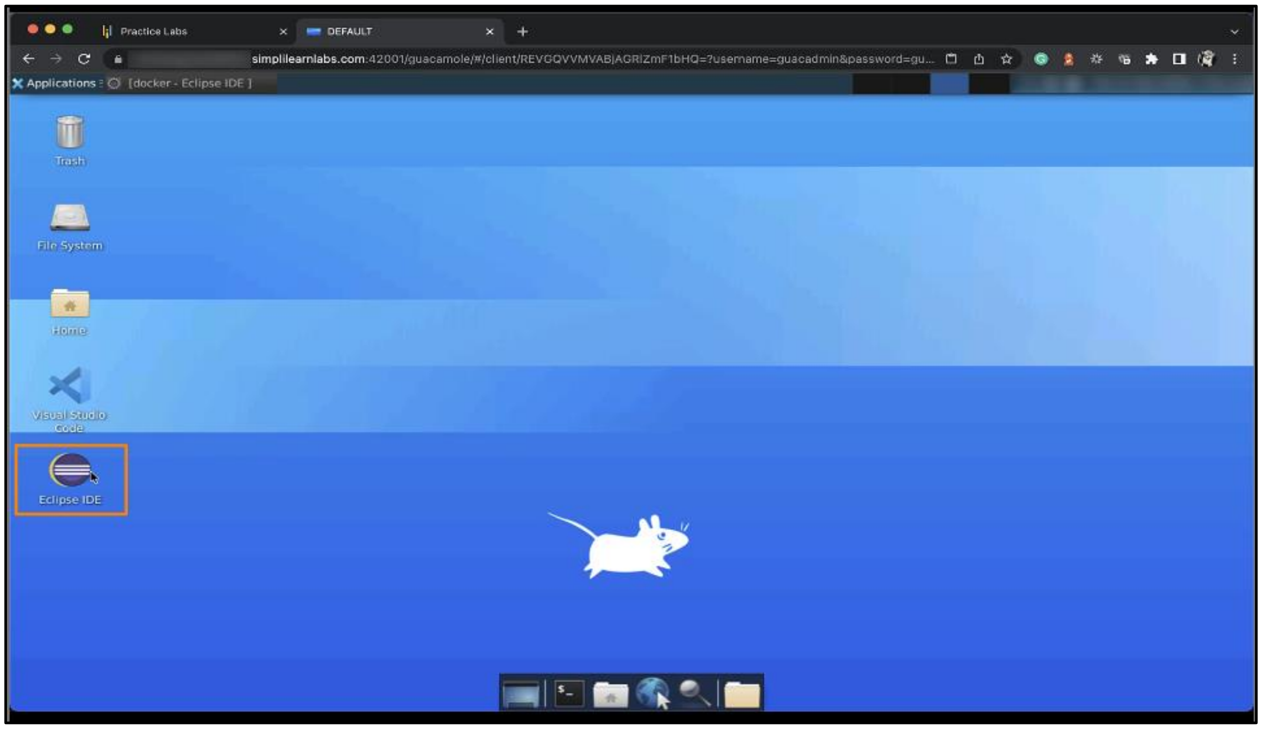
Prerequisites: None

Steps to be followed:

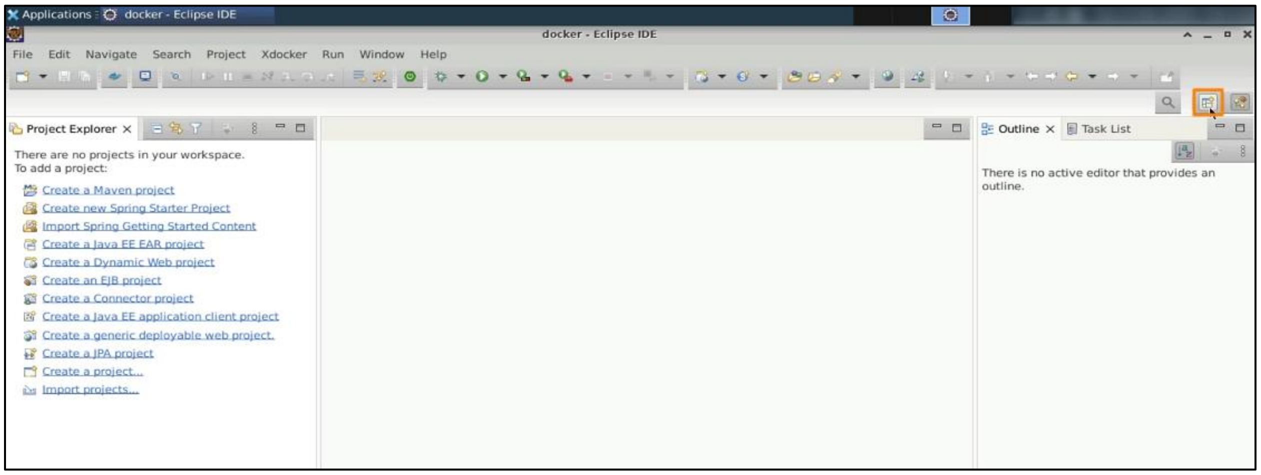
1. Creating a Java project
2. Creating a class
3. Creating a Dockerfile
4. Rebuilding the image

Step 1: Creating a Java project

1.1 Open **Eclipse IDE**


Practice Labs
DEFAULT
simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABJAGRIZmF1bHQ=?username=guacadmin&password=gu...
Applications : [docker - Eclipse IDE]
Trash
File System
Home
Visual Studio Code
Eclipse IDE

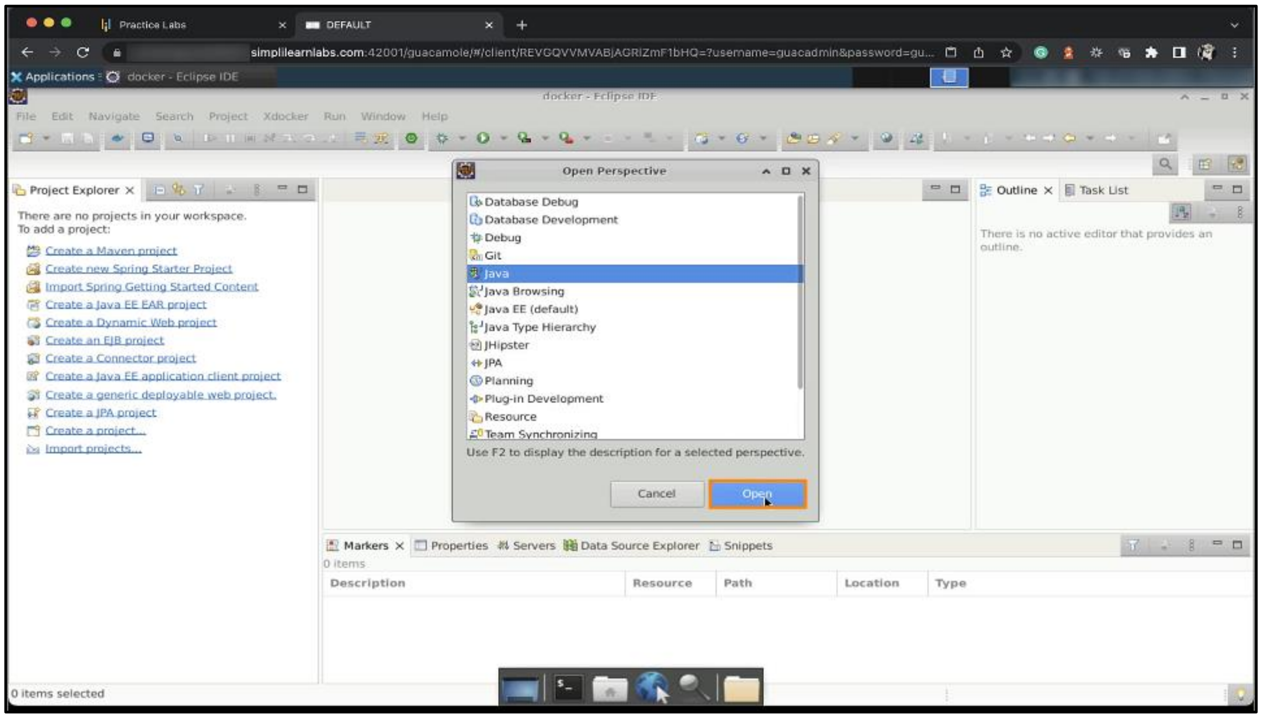
1.2 Click on the icon at the top-right corner, select **Java**, and click on the **Open** button



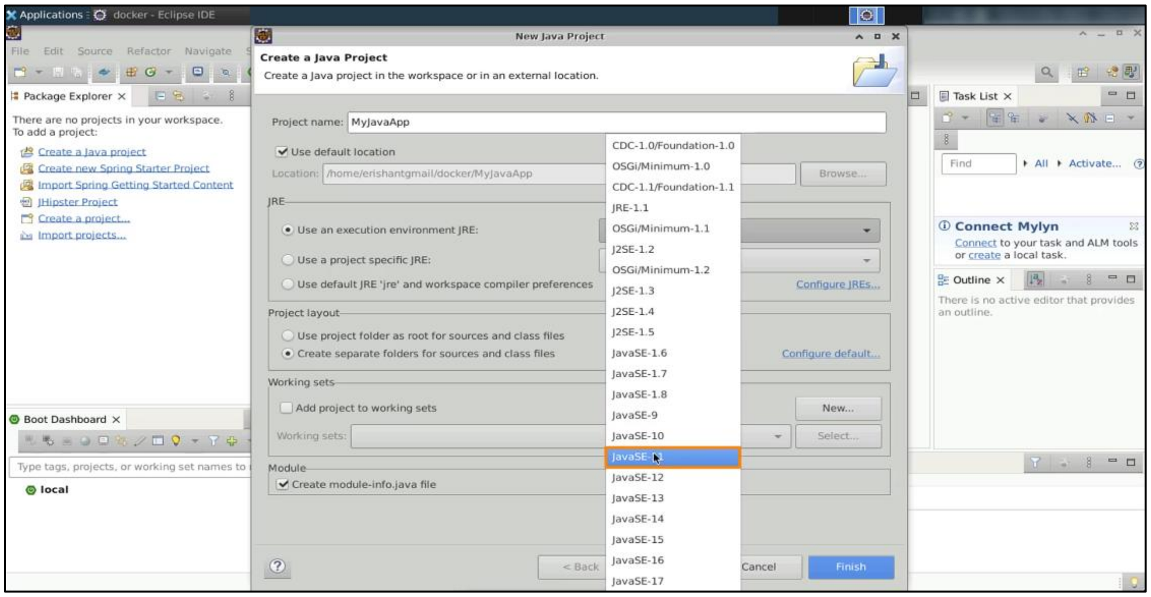
There are no projects in your workspace.
To add a project:

- Create a Maven project
- Create new Spring Starter Project
- Import Spring Getting Started Content
- Create a Java EE EAR project
- Create a Dynamic Web project
- Create an EJB project
- Create a Connector project
- Create a Java EE application client project
- Create a generic deployable web project.
- Create a JPA project
- Create a project...
- Import projects...

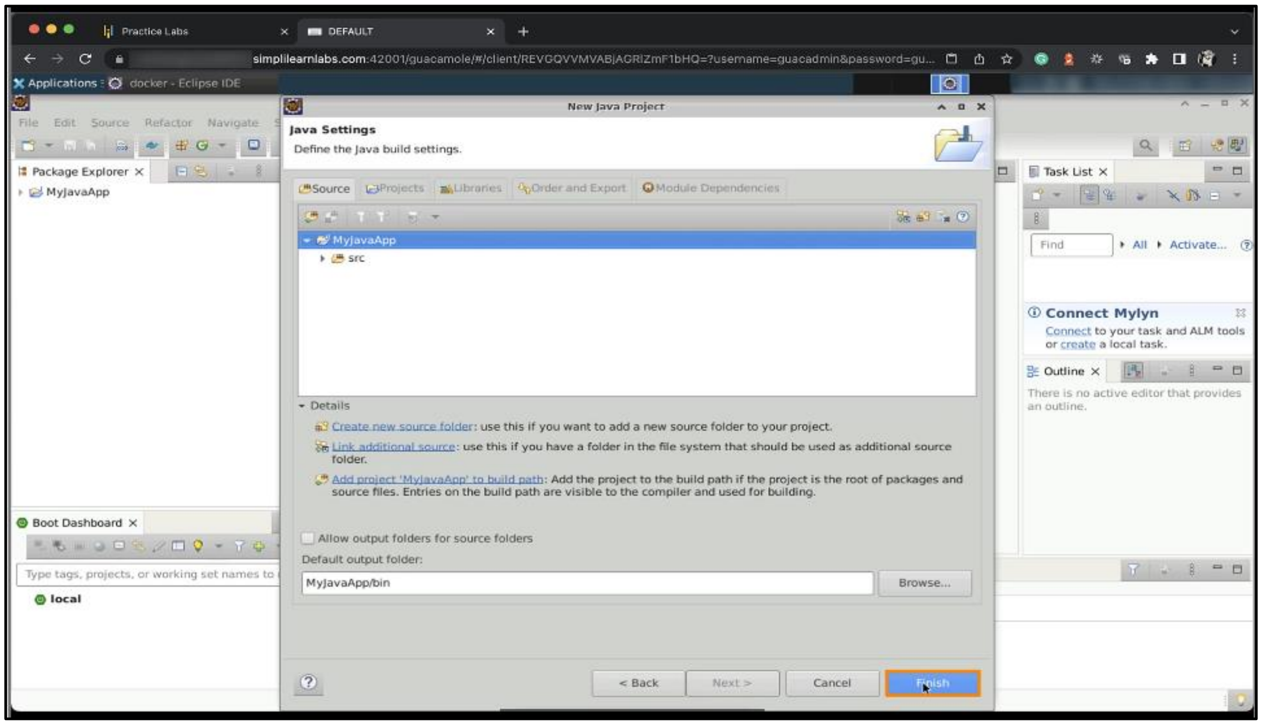
There is no active editor that provides an outline.



1.3 Name the project as **MyJavaApp** and specify the target runtime as **JavaSE-11**


New Java Project
Create a Java Project
Create a Java project in the workspace or in an external location.
Project name: MyJavaApp
Use default location
Location: /home/erishantgmail/docker/MyJavaApp
JRE
Use an execution environment JRE:
JavaSE-11
Finish

1.4 Click on **Next** and **Finish**


Practice Labs
DEFAULT
simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABJAGRlZmF1bHQ=?username=guacadmin&password=gu...
Applications
docker - Eclipse IDE
File Edit Source Refactor Navigate
Package Explorer
MyJavaApp
New Java Project
Java Settings
Define the Java build settings.
Source
Projects
Libraries
Order and Export
Module Dependencies
MyJavaApp
src
Details
Allow output folders for source folders
Default output folder:
MyJavaApp/bin
Browse...
< Back
Next >
Cancel
Finish
Boot Dashboard
Type tags, projects, or working set names to
local
Task List
Find
All
Activate...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
There is no active editor that provides an outline.

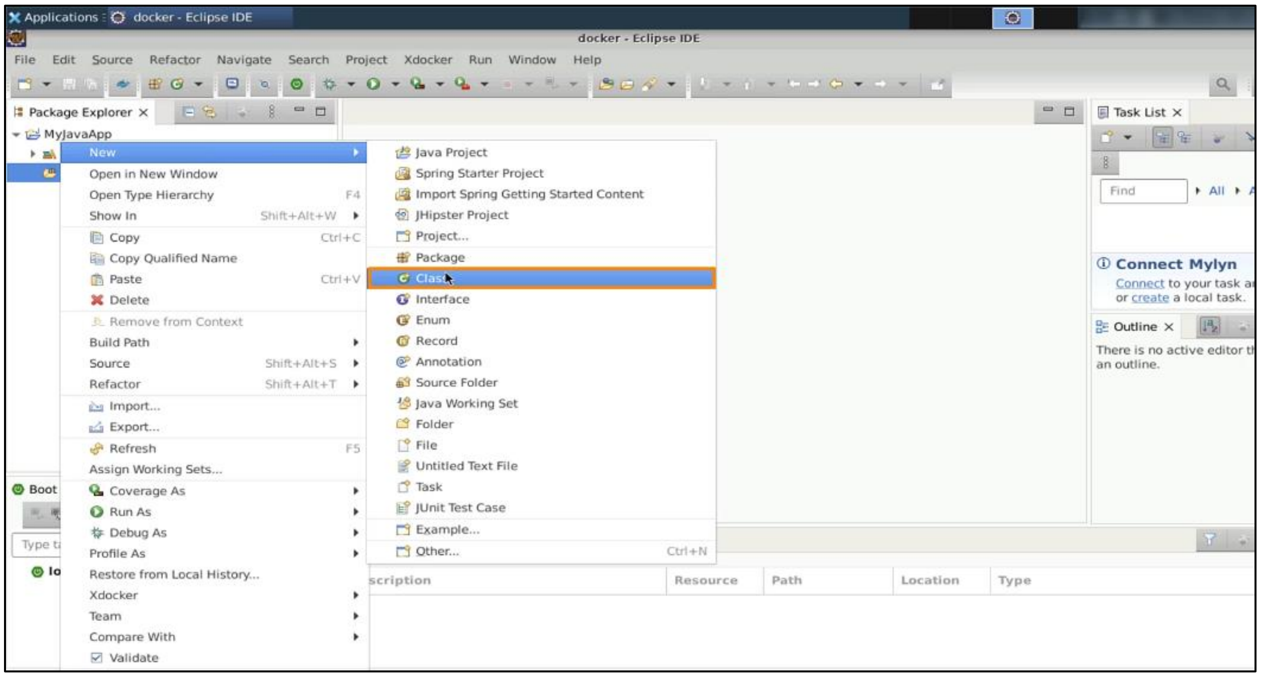
Create new source folder: use this if you want to add a new source folder to your project.

Link additional source: use this if you have a folder in the file system that should be used as additional source folder.

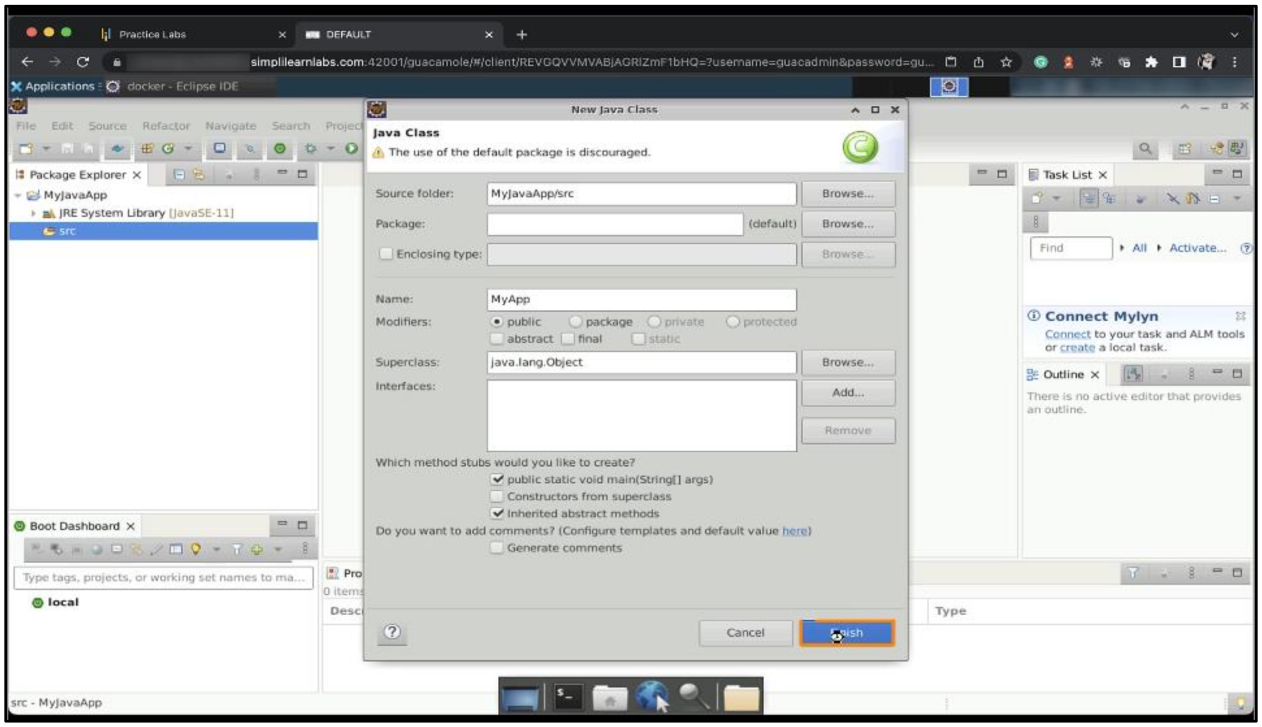
Add project 'MyJavaApp' to build path: Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

Step 2: Creating a class

2.1 Right-click on the **src** folder, select **New**, and click on **Class**



2.2 Name the class as **MyApp**, select **public static void main,** and click on **Finish**

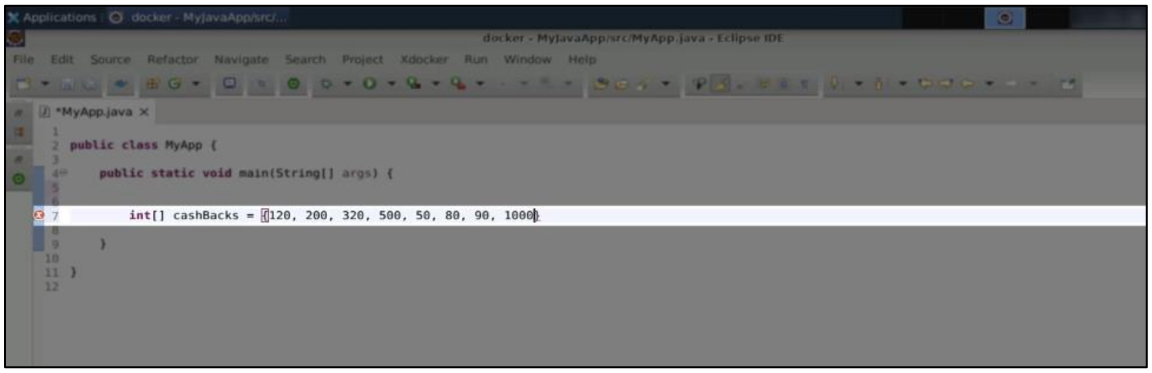


Java Class
The use of the default package is discouraged.

Source folder: MyJavaApp/src
Package: (default)
Enclosing type: [ ]
Name: MyApp
Modifiers: [✓ public] package private protected; abstract final static
Superclass: java.lang.Object
Interfaces:

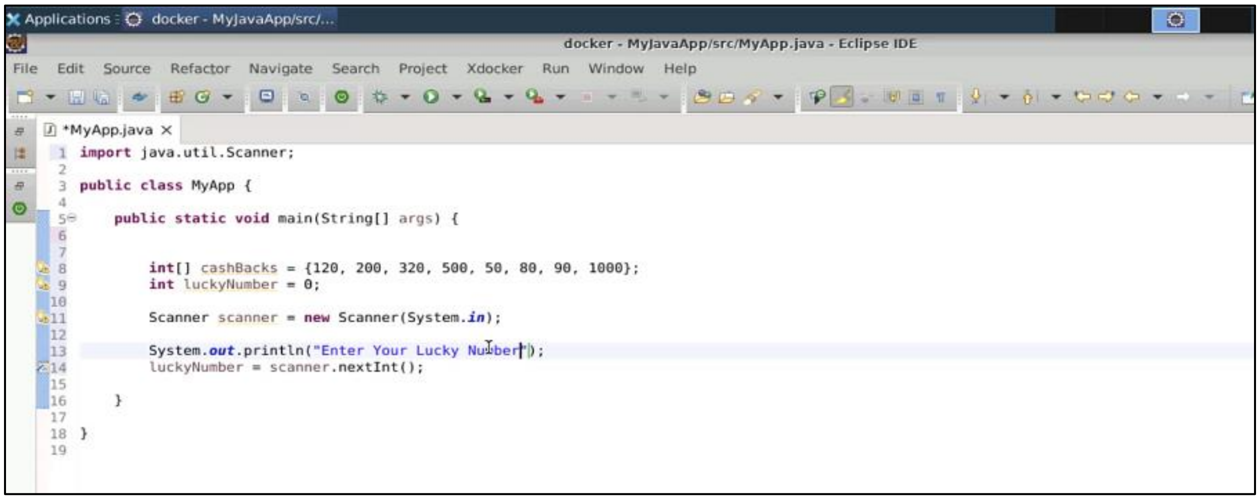
Which method stubs would you like to create?
[✓] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

2.3 In the **MyApp.java** file, create an integer array called **cashBacks** to store the user's lucky number



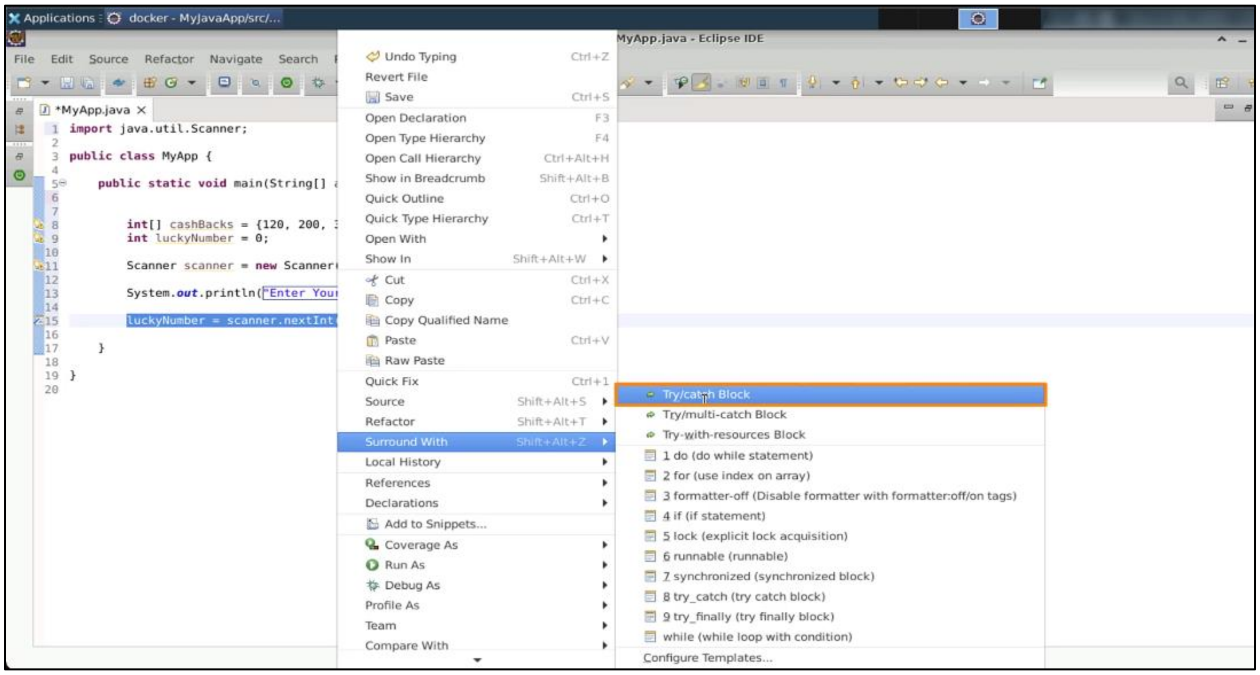
```

public class MyApp {
    public static void main(String[] args) {


        int[] cashBacks = {120, 200, 320, 500, 50, 80, 90, 1000}

    }

}

```

2.4 Use the Scanner class to read the user's input for the lucky number using the **nextInt()** function


Applications
docker - MyJavaApp/src/...
docker - MyJavaApp/src/MyApp.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Xdocker Run Window Help
*MyApp.java

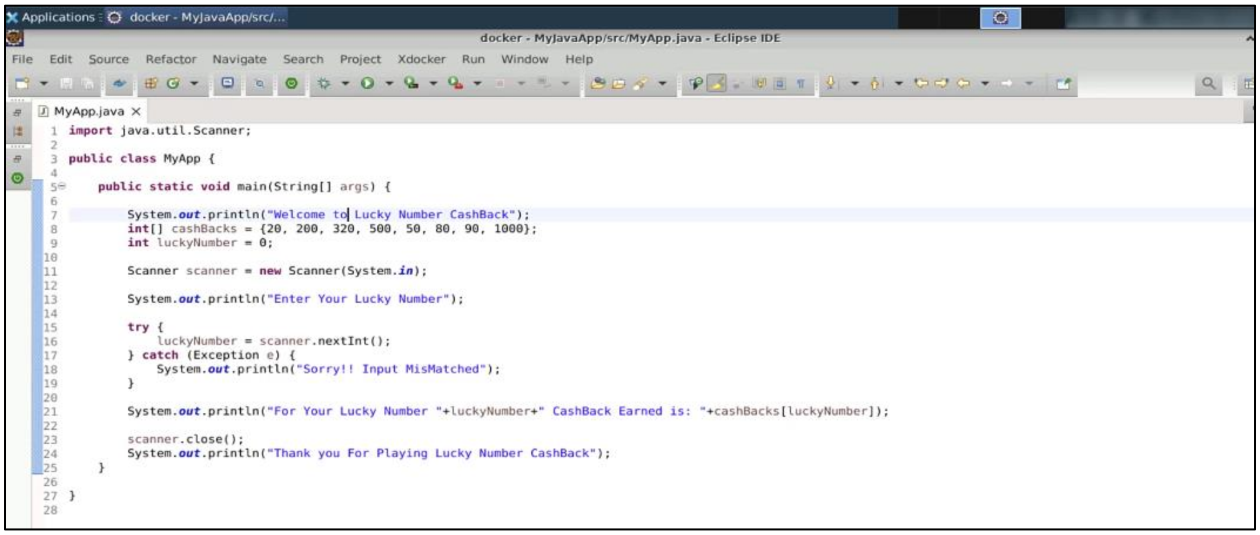
```
import java.util.Scanner;

public class MyApp {

    public static void main(String[] args) {


        int[] cashBacks = {120, 200, 320, 500, 50, 80, 90, 1000};
        int luckyNumber = 0;

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter Your Lucky Number");
        luckyNumber = scanner.nextInt();

    }

}
```

2.5 Surround the code with a try-catch block to handle any potential exceptions



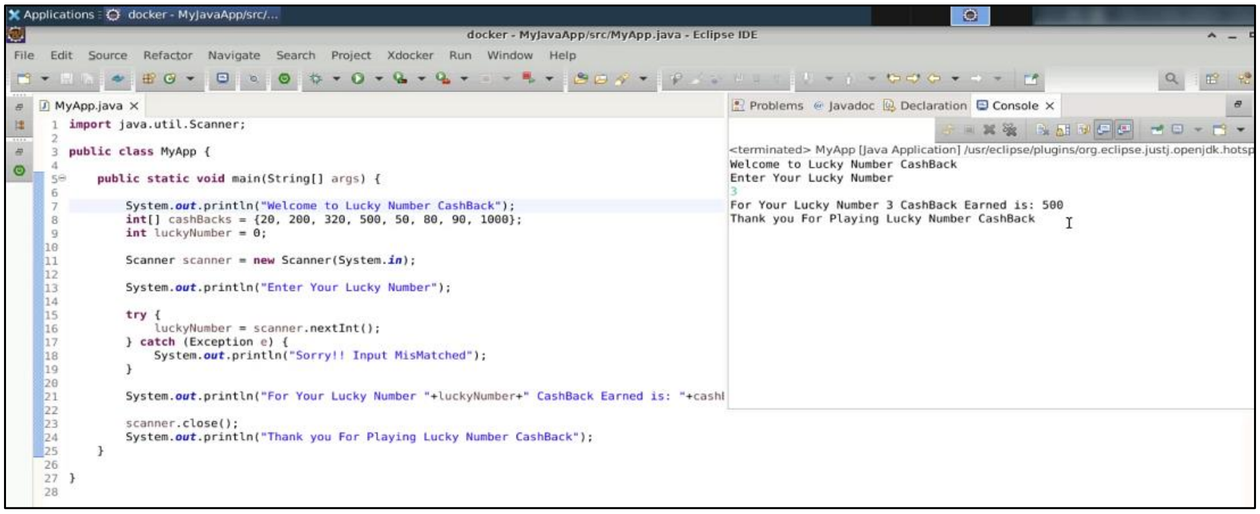
```
import java.util.Scanner;

public class MyApp {

    public static void main(String[] a


        int[] cashBacks = {120, 200, 3
        int luckyNumber = 0;

        Scanner scanner = new Scanner(

        System.out.println("Enter You

        luckyNumber = scanner.nextInt

    }
}
```

2.6 Provide an error message for incorrect input and display the cashback earned based on the lucky number



```java
import java.util.Scanner;

public class MyApp {

    public static void main(String[] args) {

        System.out.println("Welcome to Lucky Number CashBack");
        int[] cashBacks = {20, 200, 320, 500, 50, 80, 90, 1000};
        int luckyNumber = 0;

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter Your Lucky Number");

        try {
            luckyNumber = scanner.nextInt();
        } catch (Exception e) {
            System.out.println("Sorry!! Input MisMatched");
        }

        System.out.println("For Your Lucky Number "+luckyNumber+" CashBack Earned is: "+cashBacks[luckyNumber]);

        scanner.close();
        System.out.println("Thank you For Playing Lucky Number CashBack");
    }

}
```

2.7 Run the code and verify the output

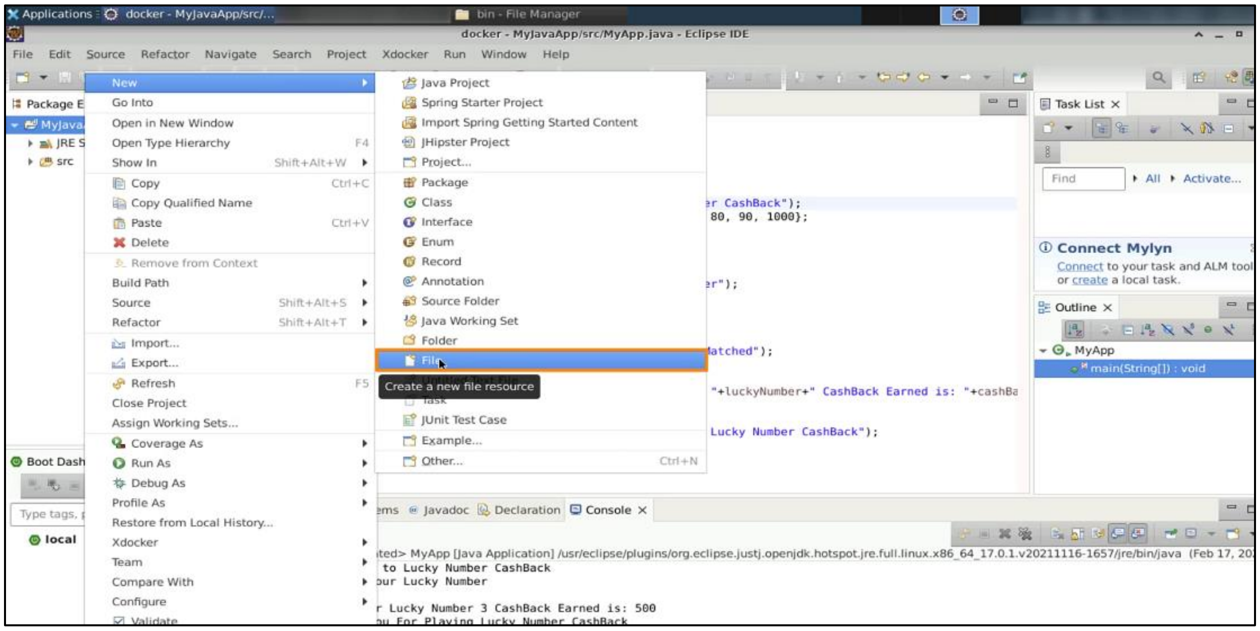


```
<terminated> MyApp [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
Welcome to Lucky Number CashBack
Enter Your Lucky Number
3
For Your Lucky Number 3 CashBack Earned is: 500
Thank you For Playing Lucky Number CashBack
```

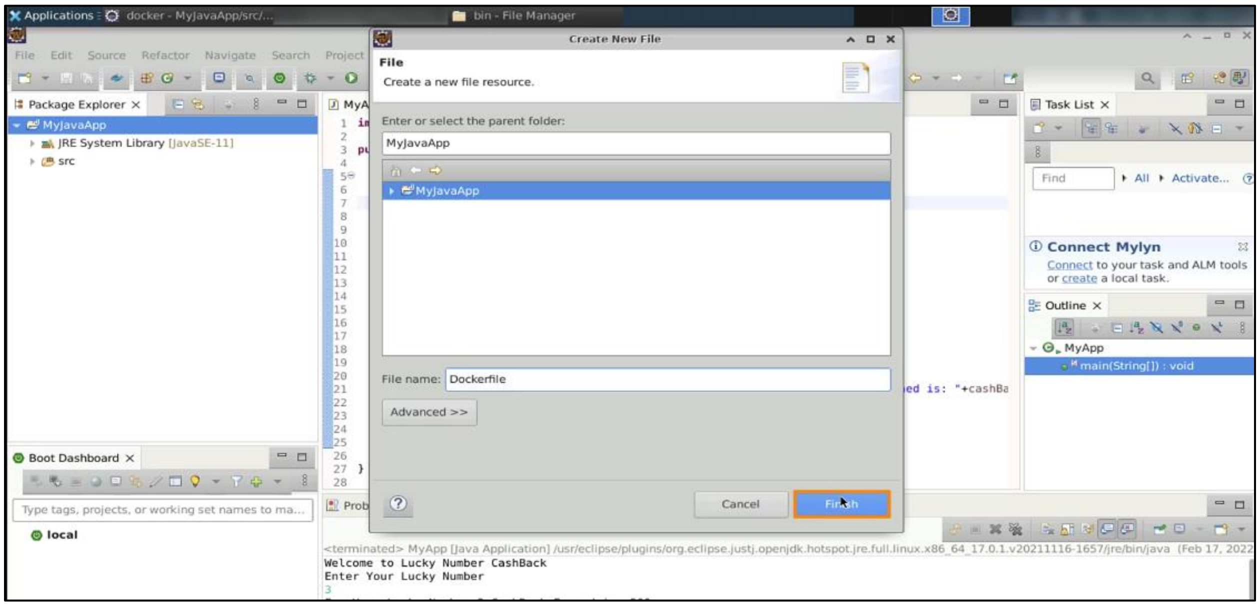
For example, if the lucky number entered is **3**, the output should display **For Your Lucky Number CashBack Earned is: 500**.

Step 3: Creating a Dockerfile

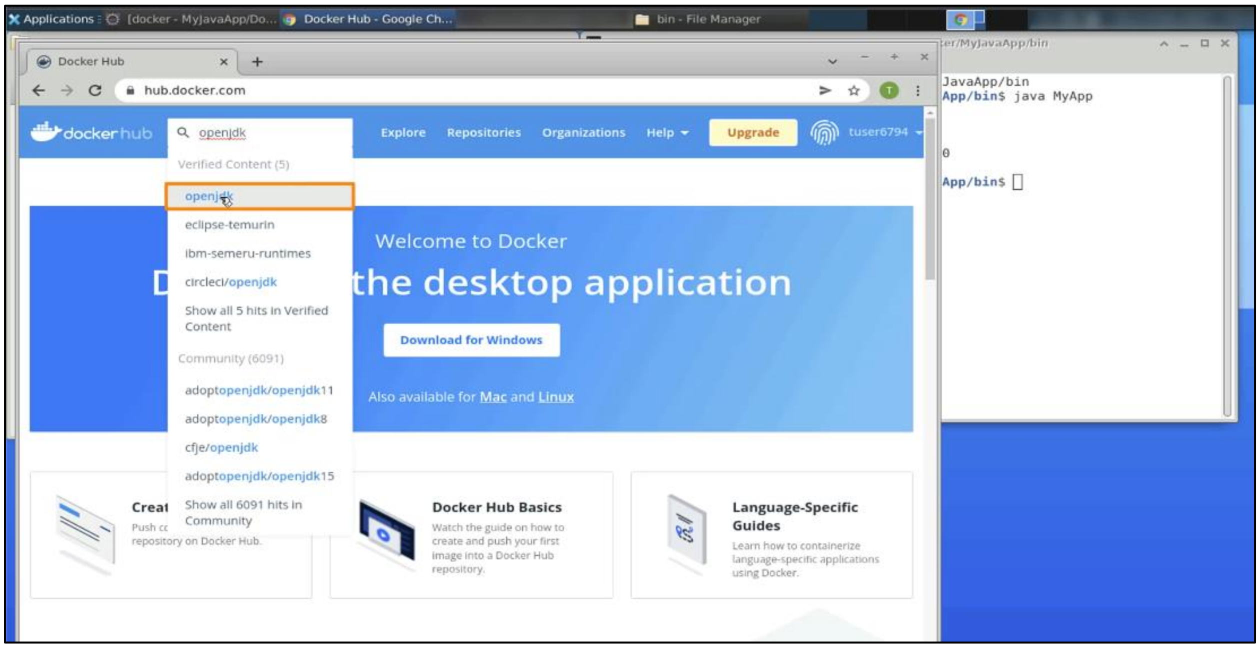
3.1 In the **Eclipse IDE**, create a new file by right-clicking on the project, selecting **New**, and clicking on **File**



3.2 Name the file **Dockerfile** and click on **Finish**

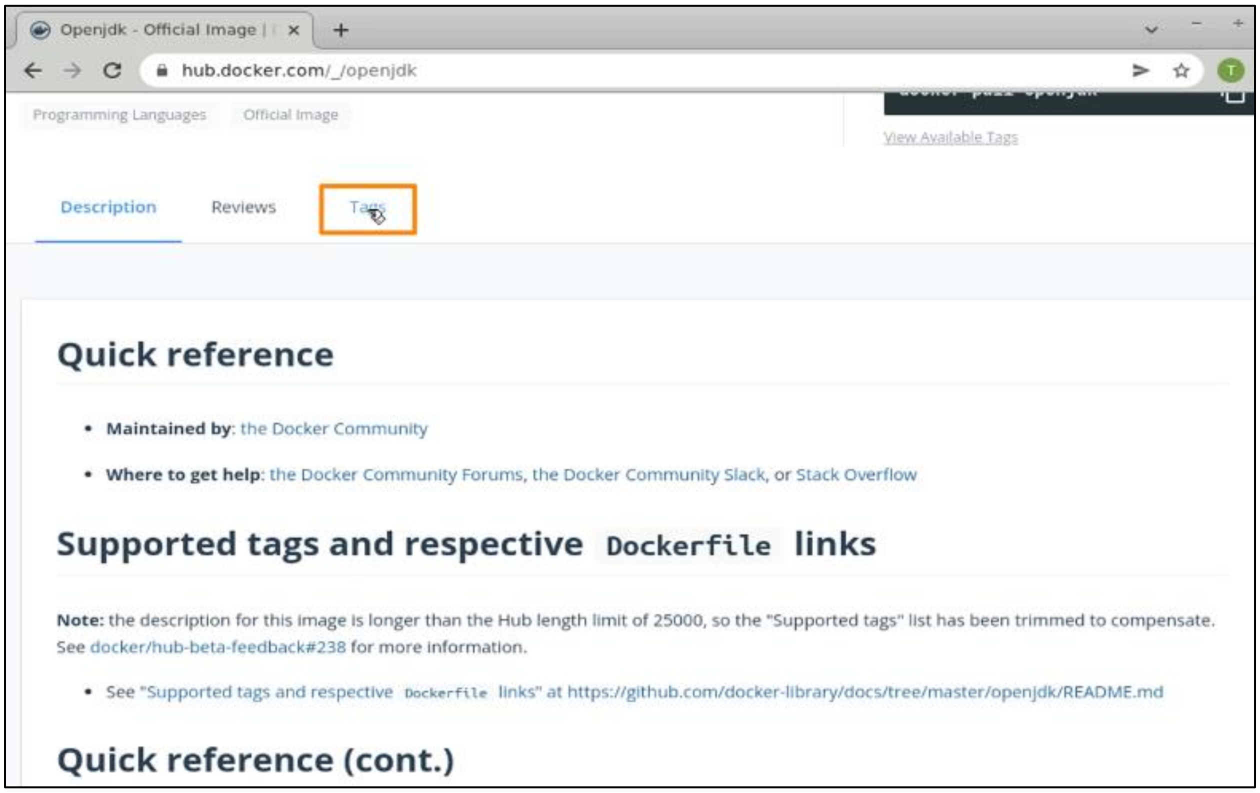


3.3 Open a web browser, navigate to **hub.docker.com,** and search for **openjdk**



```
er/MyJavaApp/bin
JavaApp/bin
App/bin$ java MyApp

0
App/bin$
```

3.4 Check for the available tags, specifically for JDK 11



Quick reference

- **Maintained by**: the Docker Community
- **Where to get help**: the Docker Community Forums, the Docker Community Slack, or Stack Overflow

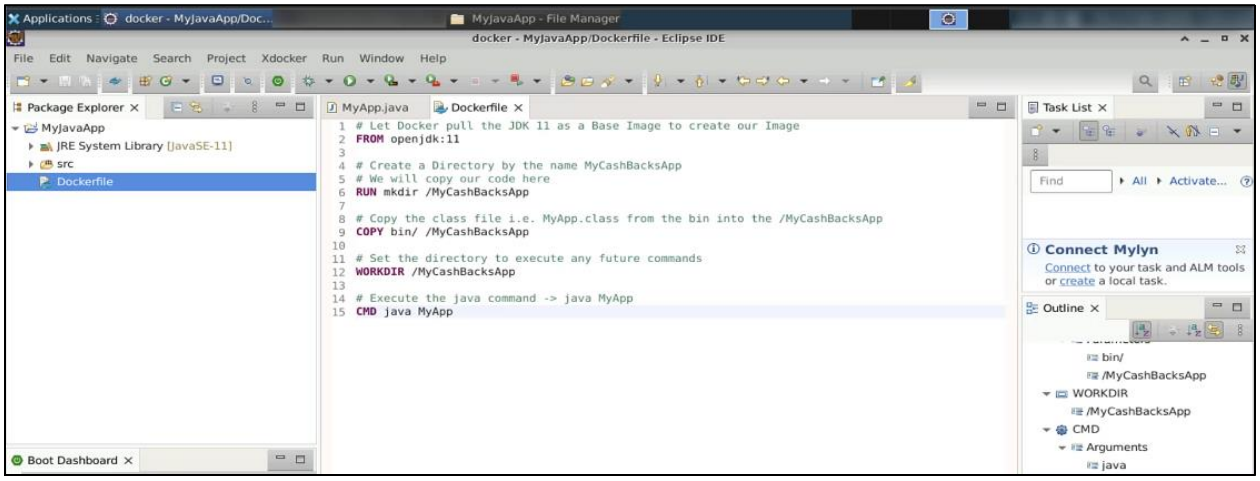
Supported tags and respective `Dockerfile` links

Note: the description for this image is longer than the Hub length limit of 25000, so the "Supported tags" list has been trimmed to compensate. See docker/hub-beta-feedback#238 for more information.

- See "Supported tags and respective `Dockerfile` links" at https://github.com/docker-library/docs/tree/master/openjdk/README.md

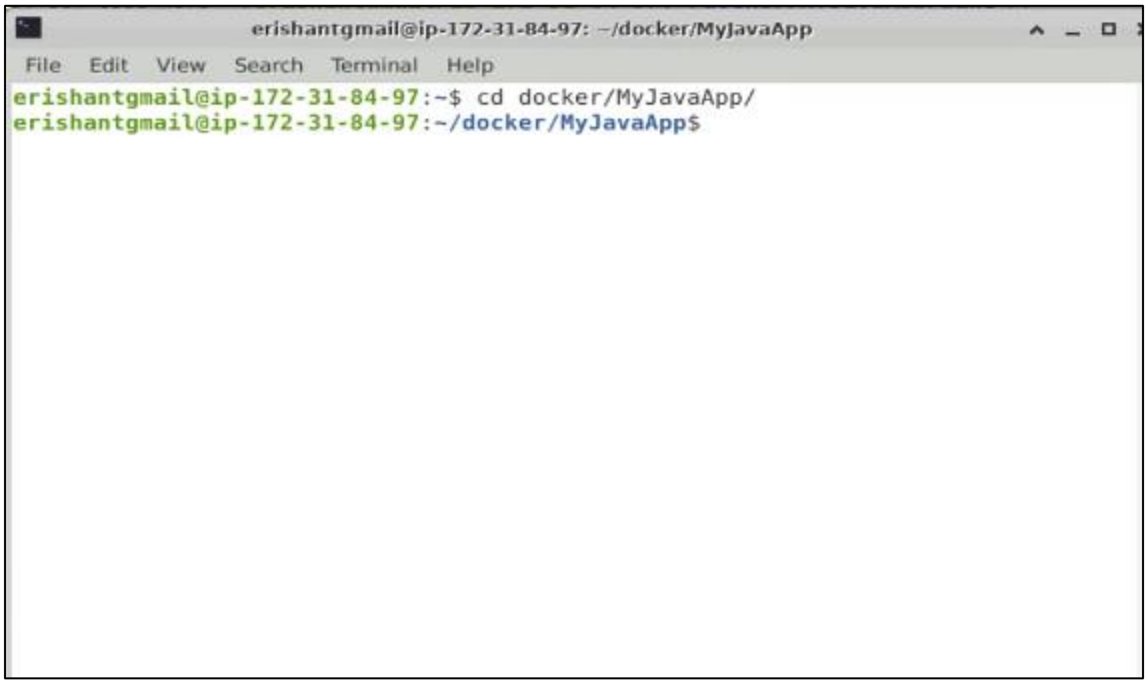
Quick reference (cont.)

3.5 Go back to the **Dockerfile** and write the code to pull JDK 11 as the base image for creating the Java app image. Create a directory named **MyCashBacksApp** and copy the code from the **bin** directory into the **MyCashBacksApp** directory



```
# Let Docker pull the JDK 11 as a Base Image to create our Image
FROM openjdk:11

# Create a Directory by the name MyCashBacksApp
# We will copy our code here
RUN mkdir /MyCashBacksApp

# Copy the class file i.e. MyApp.class from the bin into the /MyCashBacksApp
COPY bin/ /MyCashBacksApp

# Set the directory to execute any future commands
WORKDIR /MyCashBacksApp

# Execute the java command -> java MyApp
CMD java MyApp
```

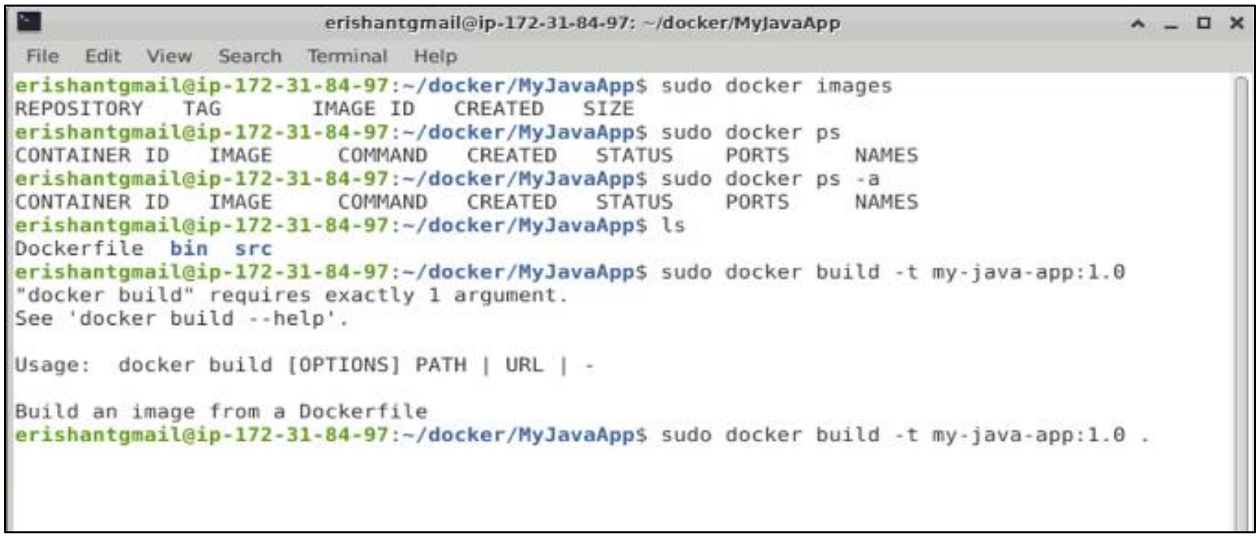
3.6 Open a new terminal window and navigate to the Docker directory and the project's root folder



```
erishantgmail@ip-172-31-84-97:~$ cd docker/MyJavaApp/
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$
```

3.7 Create a Docker image using the docker build command with the tag parameter. Run the following command:

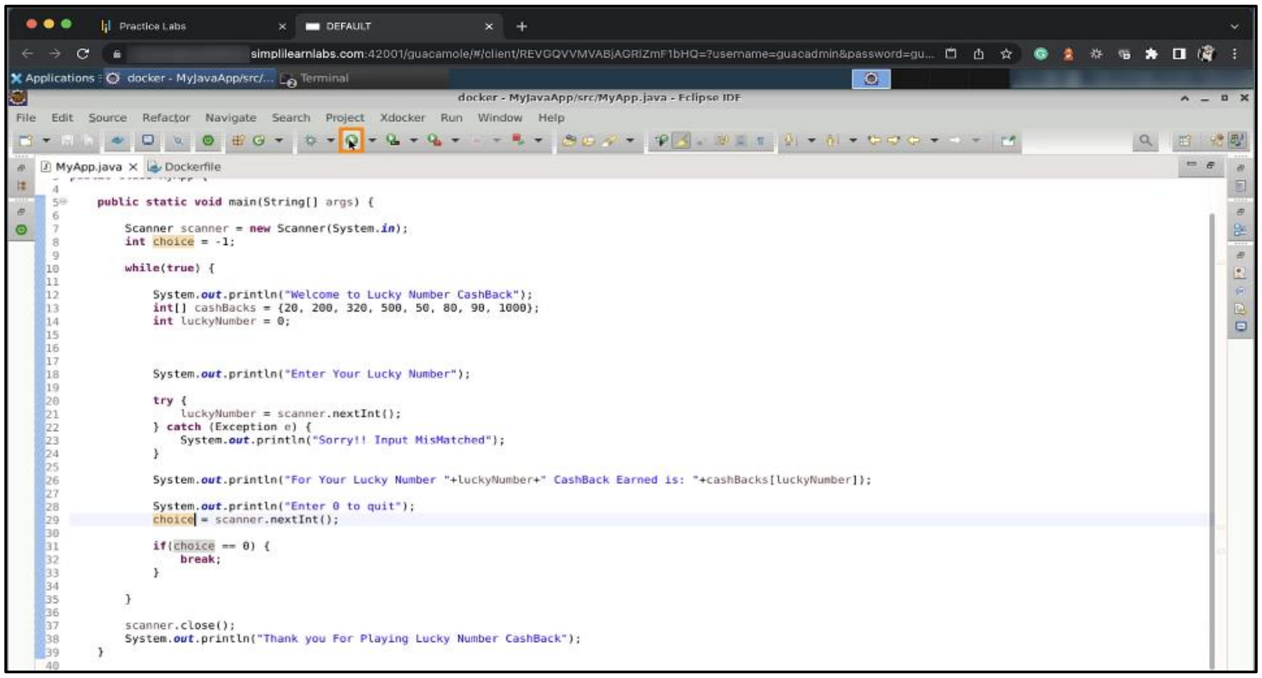
sudo docker build -t my-java-app:1.0 .



```
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ ls
Dockerfile  bin  src
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker build -t my-java-app:1.0
"docker build" requires exactly 1 argument.
See 'docker build --help'.

Usage:  docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker build -t my-java-app:1.0 .
```

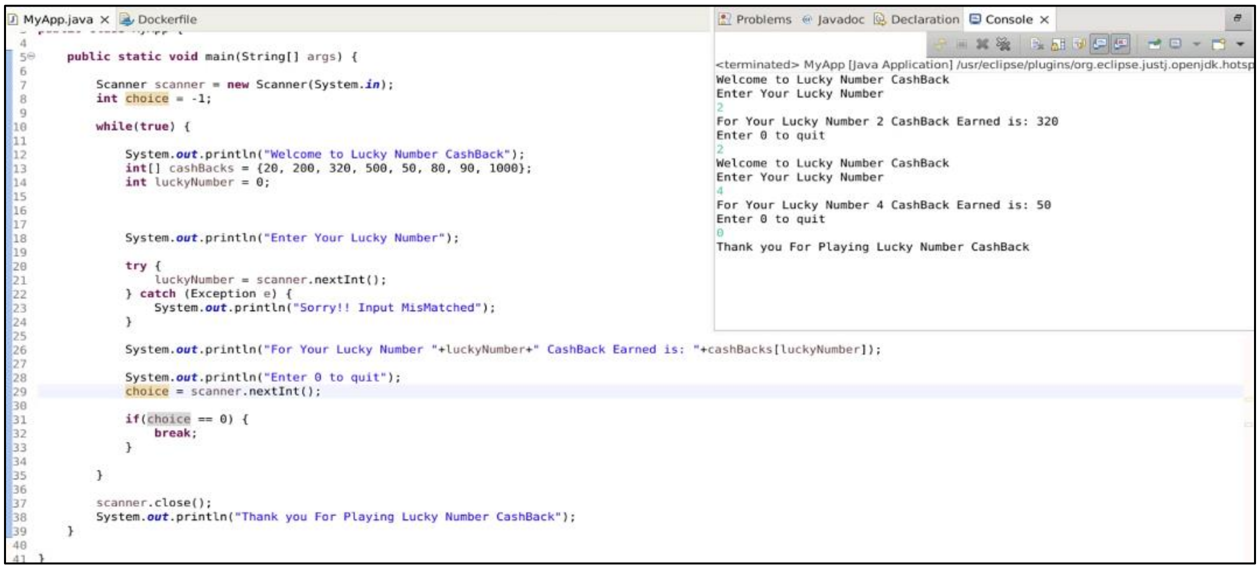
Note: The dot at the end specifies the path to the Dockerfile.

3.8 Modify the Java program by adding a while loop to run the code continuously until the user enters **0** to quit the program



```java
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        int choice = -1;

        while(true) {

            System.out.println("Welcome to Lucky Number CashBack");
            int[] cashBacks = {20, 200, 320, 500, 50, 80, 90, 1000};
            int luckyNumber = 0;



            System.out.println("Enter Your Lucky Number");

            try {
                luckyNumber = scanner.nextInt();
            } catch (Exception e) {
                System.out.println("Sorry!! Input MisMatched");
            }

            System.out.println("For Your Lucky Number "+luckyNumber+" CashBack Earned is: "+cashBacks[luckyNumber]);

            System.out.println("Enter 0 to quit");
            choice = scanner.nextInt();

            if(choice == 0) {
                break;
            }

        }

        scanner.close();
        System.out.println("Thank you For Playing Lucky Number CashBack");
    }
```

3.9 Run the modified Java program and test its functionality by entering various lucky numbers. The program should keep asking for a new number until the user enters **0**, upon which it will display the message **Thank you For Playing Lucky Number CashBack**.

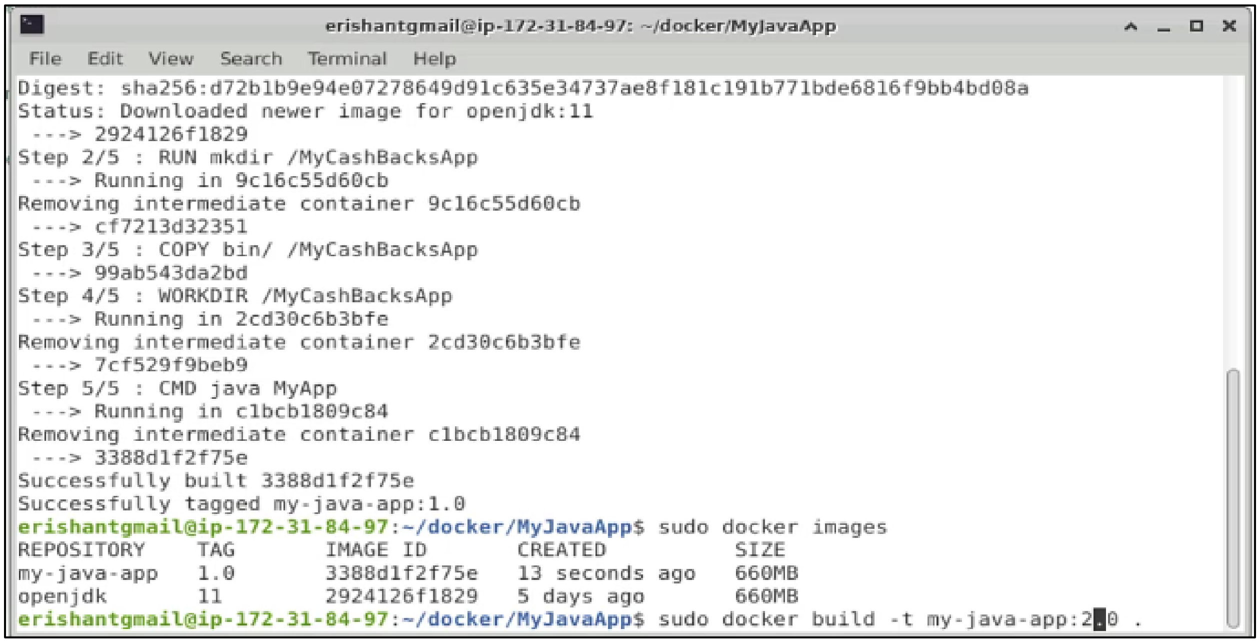


```java
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        int choice = -1;

        while(true) {

            System.out.println("Welcome to Lucky Number CashBack");
            int[] cashBacks = {20, 200, 320, 500, 50, 80, 90, 1000};
            int luckyNumber = 0;



            System.out.println("Enter Your Lucky Number");

            try {
                luckyNumber = scanner.nextInt();
            } catch (Exception e) {
                System.out.println("Sorry!! Input MisMatched");
            }

            System.out.println("For Your Lucky Number "+luckyNumber+" CashBack Earned is: "+cashBacks[luckyNumber]);

            System.out.println("Enter 0 to quit");
            choice = scanner.nextInt();

            if(choice == 0) {
                break;
            }

        }

        scanner.close();
        System.out.println("Thank you For Playing Lucky Number CashBack");
    }

}
```

```
<terminated> MyApp [Java Application] /usr/eclipse/plugins/org.eclipse.justj.openjdk.hotsp
Welcome to Lucky Number CashBack
Enter Your Lucky Number
2
For Your Lucky Number 2 CashBack Earned is: 320
Enter 0 to quit
2
Welcome to Lucky Number CashBack
Enter Your Lucky Number
4
For Your Lucky Number 4 CashBack Earned is: 50
Enter 0 to quit
0
Thank you For Playing Lucky Number CashBack
```

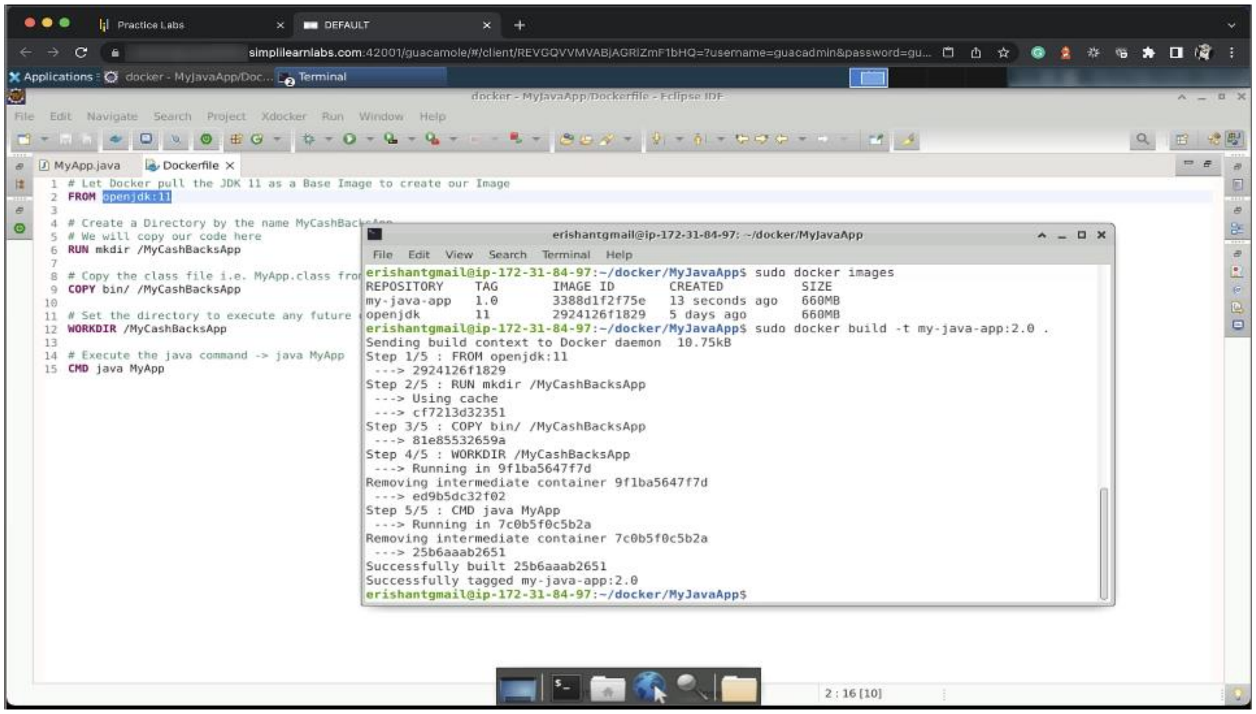
Step 4: Rebuilding the image

4.1 Go back to the terminal window and rebuild the Docker image using the following command:
sudo docker build -t my-java-app:2.0 .



```
Digest: sha256:d72b1b9e94e07278649d91c635e34737ae8f181c191b771bde6816f9bb4bd08a
Status: Downloaded newer image for openjdk:11
 ---> 2924126f1829
Step 2/5 : RUN mkdir /MyCashBacksApp
 ---> Running in 9c16c55d60cb
Removing intermediate container 9c16c55d60cb
 ---> cf7213d32351
Step 3/5 : COPY bin/ /MyCashBacksApp
 ---> 99ab543da2bd
Step 4/5 : WORKDIR /MyCashBacksApp
 ---> Running in 2cd30c6b3bfe
Removing intermediate container 2cd30c6b3bfe
 ---> 7cf529f9beb9
Step 5/5 : CMD java MyApp
 ---> Running in c1bcb1809c84
Removing intermediate container c1bcb1809c84
 ---> 3388d1f2f75e
Successfully built 3388d1f2f75e
Successfully tagged my-java-app:1.0
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
my-java-app   1.0       3388d1f2f75e   13 seconds ago   660MB
openjdk       11        2924126f1829   5 days ago       660MB
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker build -t my-java-app:2.0 .
```

Note: This creates a new image with tag **2.0**.



```
# Let Docker pull the JDK 11 as a Base Image to create our Image
FROM openjdk:11

# Create a Directory by the name MyCashBack
# We will copy our code here
RUN mkdir /MyCashBacksApp

# Copy the class file i.e. MyApp.class fro
COPY bin/ /MyCashBacksApp

# Set the directory to execute any future
WORKDIR /MyCashBacksApp

# Execute the java command -> java MyApp
CMD java MyApp
```

```
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
my-java-app   1.0       3388d1f2f75e   13 seconds ago   660MB
openjdk       11        2924126f1829   5 days ago       660MB
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$ sudo docker build -t my-java-app:2.0 .
Sending build context to Docker daemon  10.75kB
Step 1/5 : FROM openjdk:11
 ---> 2924126f1829
Step 2/5 : RUN mkdir /MyCashBacksApp
 ---> Using cache
 ---> cf7213d32351
Step 3/5 : COPY bin/ /MyCashBacksApp
 ---> 81e85532659a
Step 4/5 : WORKDIR /MyCashBacksApp
 ---> Running in 9f1ba5647f7d
Removing intermediate container 9f1ba5647f7d
 ---> ed9b5dc32f02
Step 5/5 : CMD java MyApp
 ---> Running in 7c0b5f0c5b2a
Removing intermediate container 7c0b5f0c5b2a
 ---> 25b6aaab2651
Successfully built 25b6aaab2651
Successfully tagged my-java-app:2.0
erishantgmail@ip-172-31-84-97:~/docker/MyJavaApp$
```

The output should show **Successfully built** and **Successfully tagged**.

By following these steps, you can create multiple versions of your Java app image. The **-t** parameter is used to add a tag to the image for identification.