Lesson 02 Demo 02

Running Java App Image as a Container

Objective: To run a Java app image as a container, understand the container lifecycle, and perform various actions on the container

Tool required: Eclipse IDE and Docker

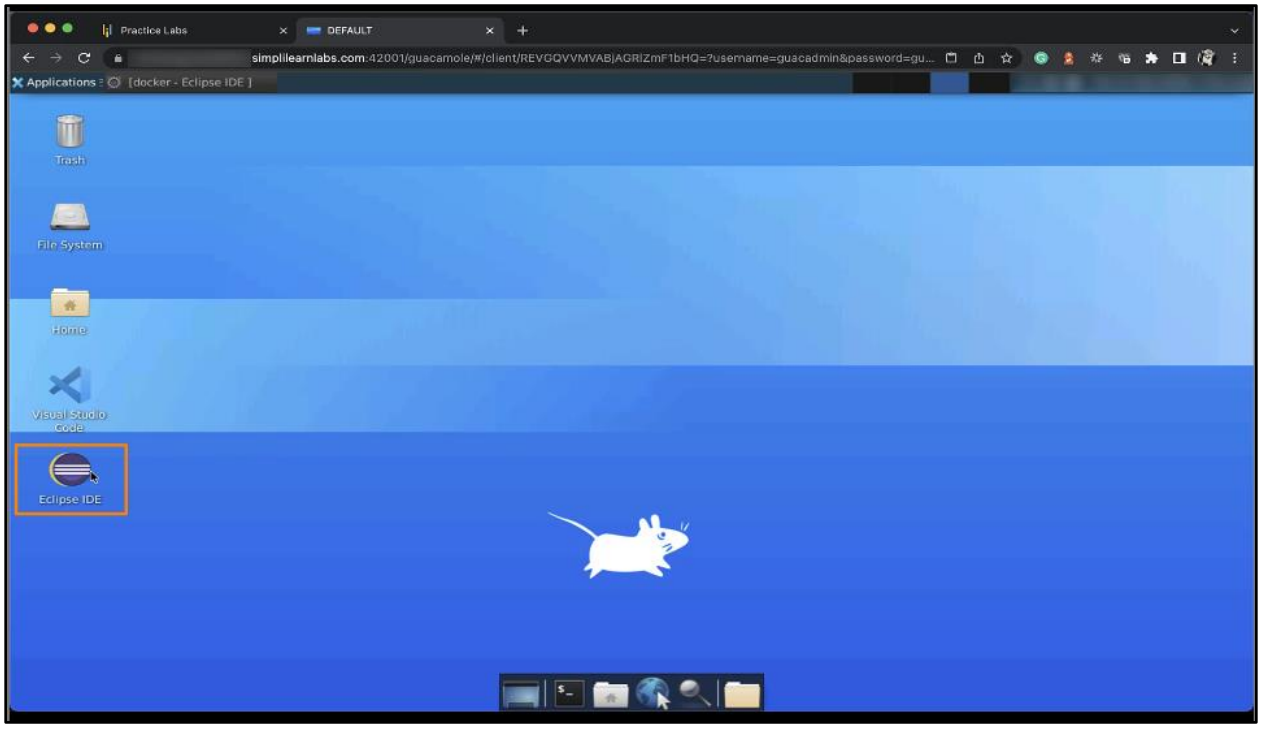
Prerequisites: Lesson 02 Demo 01

Steps to be followed:

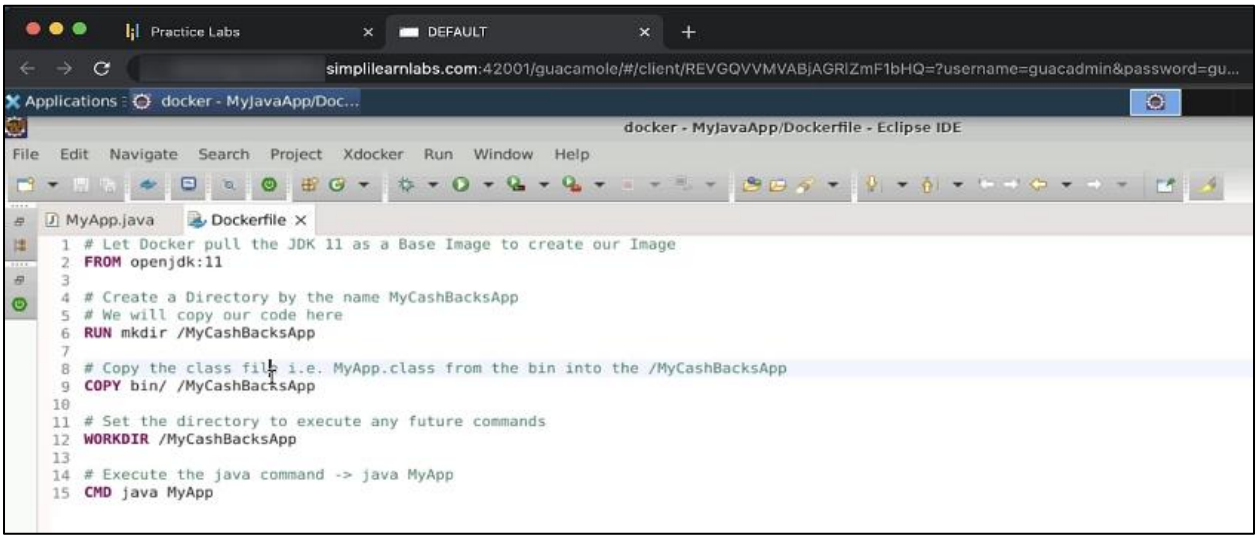
1. Executing the **docker images** command
2. Executing the **docker run** command
3. Restarting the container

Step 1: Executing the docker images command

1.1 Open **Eclipse IDE**

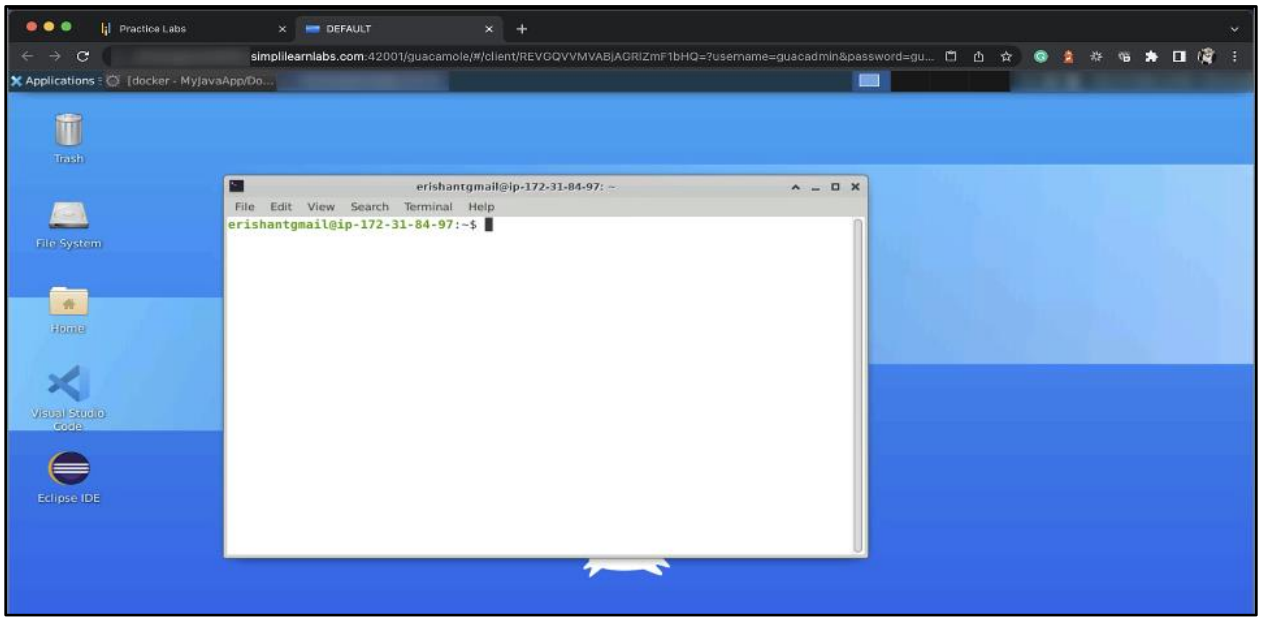


1.2 Open **Dockerfile** created in the previous demo

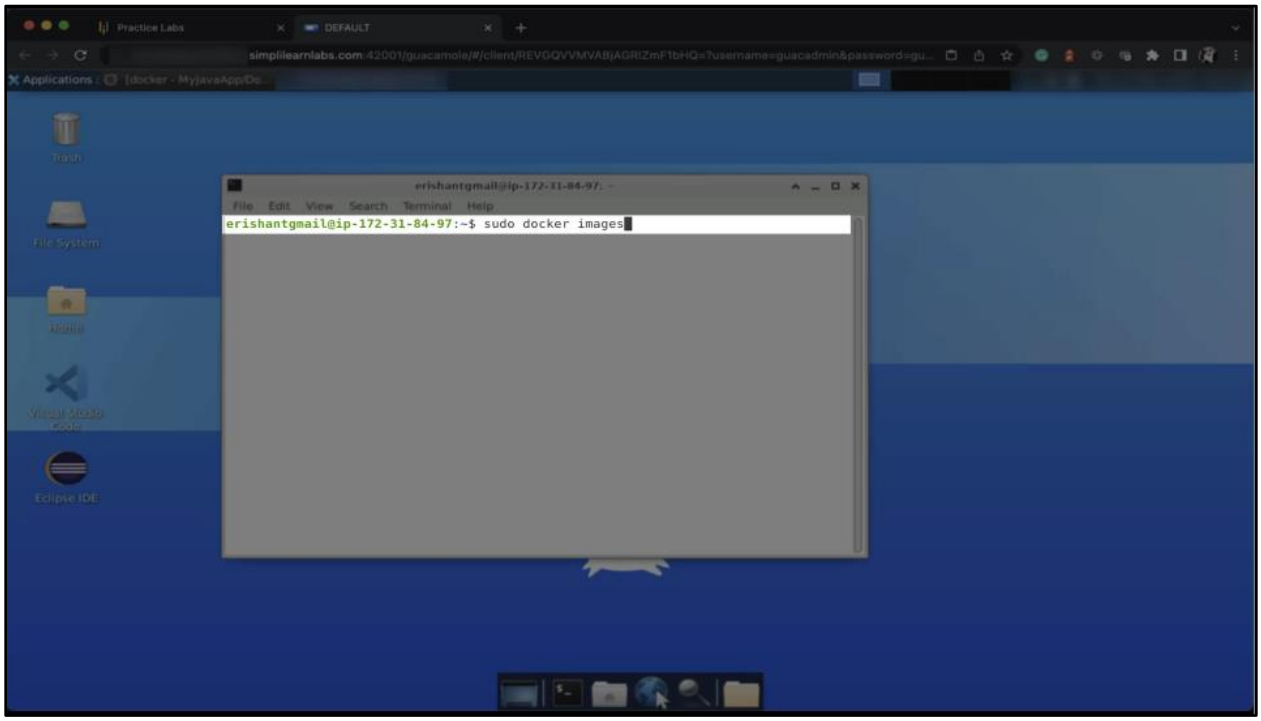


Please note that two Docker images were already created in the previous demo: one for the Java application as a cashback (1.0) and another with an infinite loop running (**2.0**).

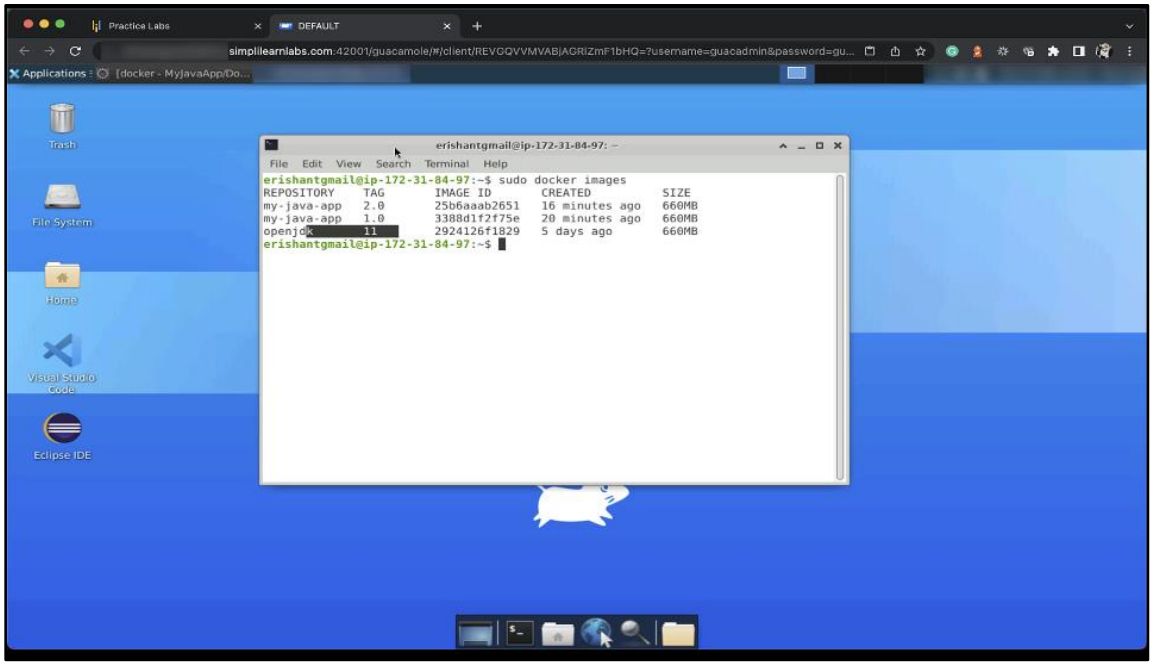
1.3 Open the terminal window



1.4 Run the **sudo docker images** command to see the available Docker images

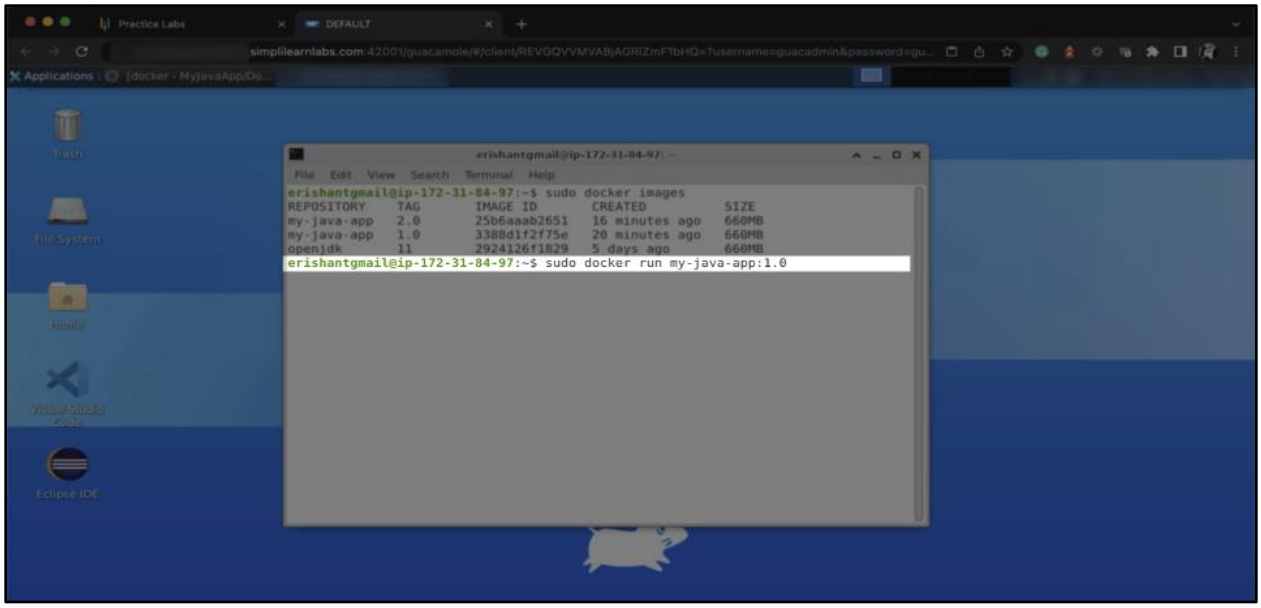


The output should display two images tagged as **1.0** and **2.0**, representing the Java application programs.

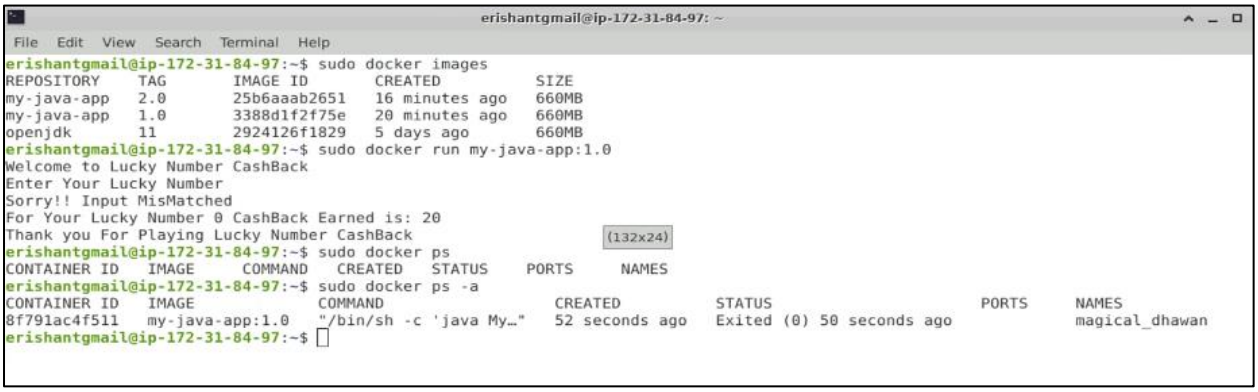


Step 2: Executing the docker run command

2.1 Run the **sudo docker run my-java-app:1.0** command to run the Java app image as a container



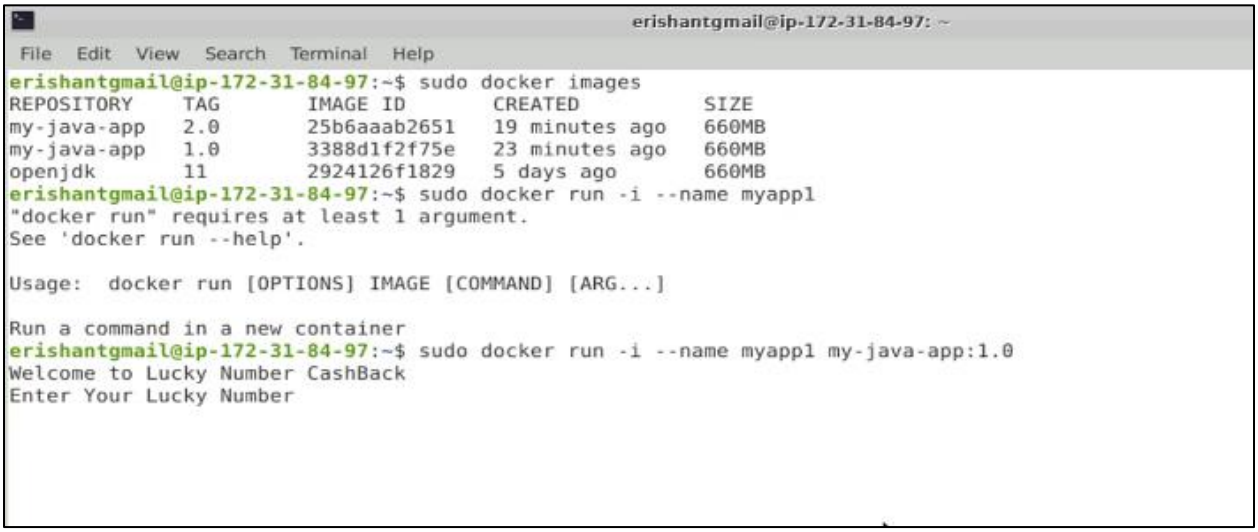
2.2 Run the **sudo docker ps -a** command to see the list of processes



You should find a process named **magical_dhawan**.

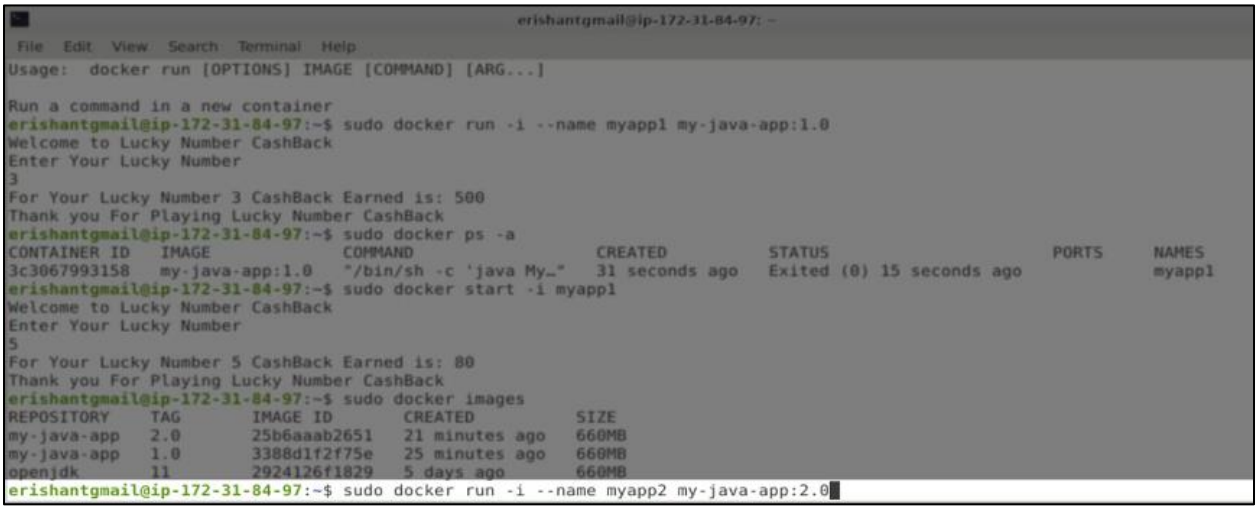
2.3 Remove the container using the **sudo docker rm magical_dhawan** command

2.4 Run the **sudo docker run -i --name myapp1 my-java-app:1.0** command to create a container named **myapp1** from the image **my-java-app:1.0**

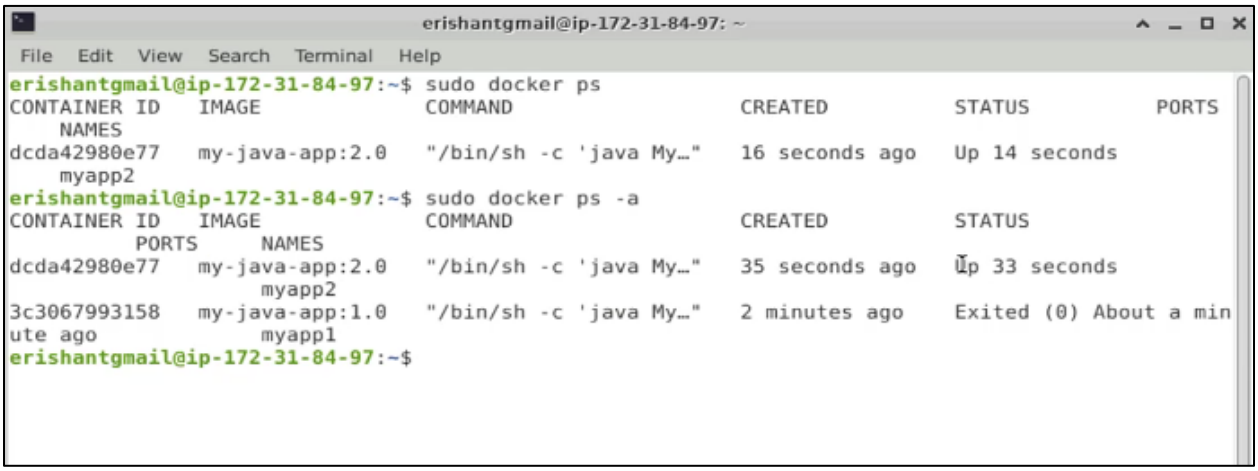


2.5 Enter the lucky number of **3**. The program should print **Thank you For Playing Lucky Number CashBack**, indicating that your application is encapsulated within the Docker container named **myapp1**.

2.6 Similarly, create an image for **my-java-app 2.0** by running the **sudo docker run -i --name myapp2 my-java-app:2.0** command

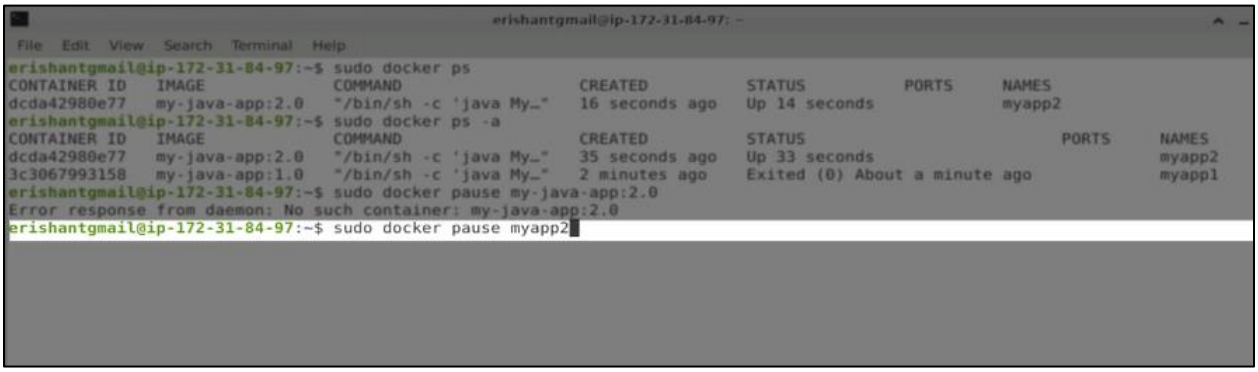


2.7 Use the **sudo docker ps -a** command to check the status of the Docker containers



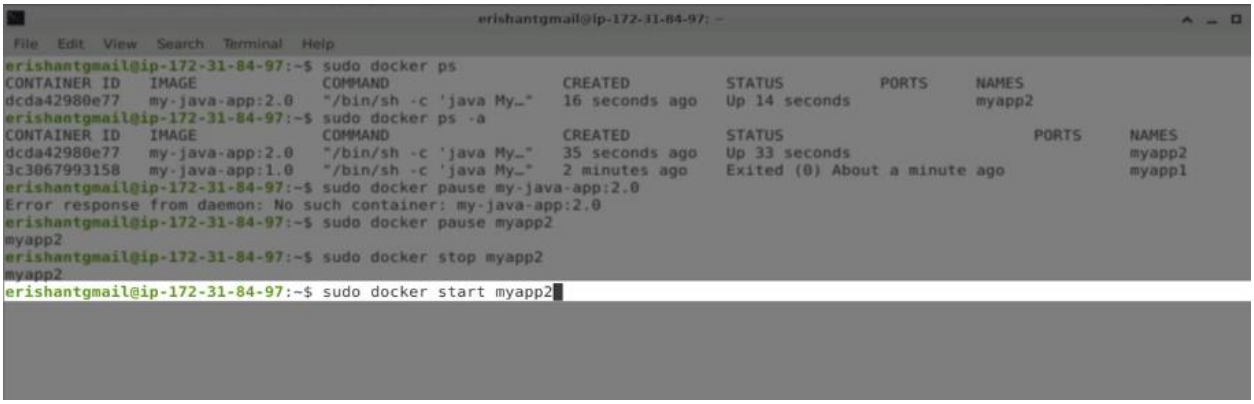
You should see two containers: one in an exited state and one running.

2.8 Stop the Docker container **myapp2** by running the **sudo docker stop myapp2** command

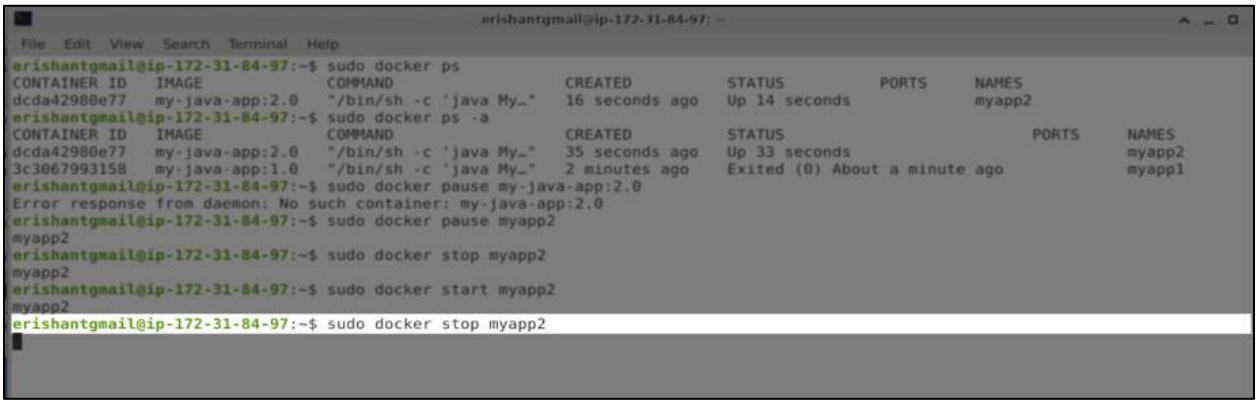


Step 3: Restarting the container

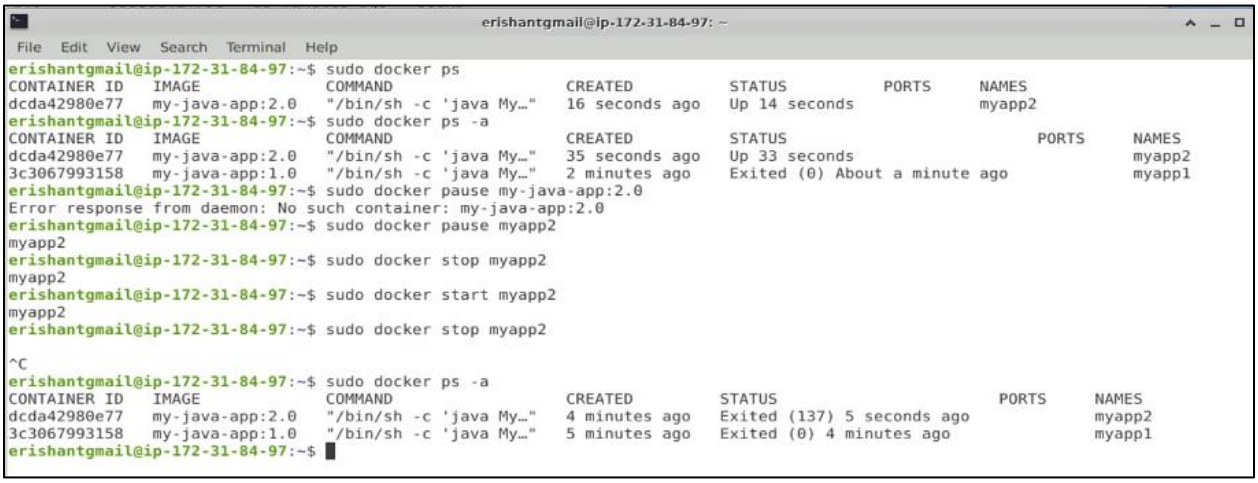
3.1 Restart the container myapp2 using the **sudo docker start myapp2** command



3.2 Stop the container using the **sudo docker stop myapp2** command

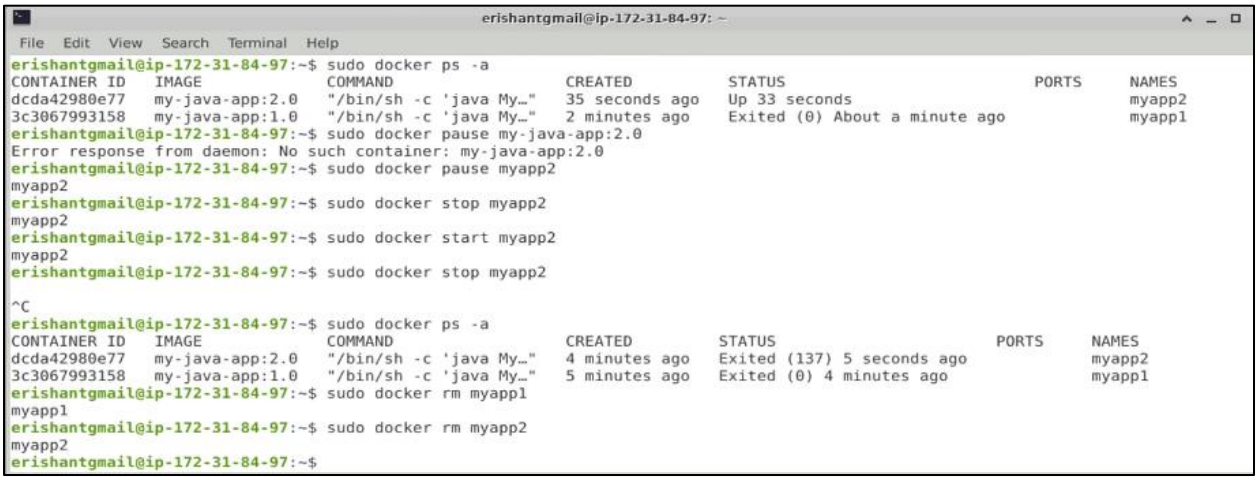


3.3 List all containers using the **sudo docker ps -a** command

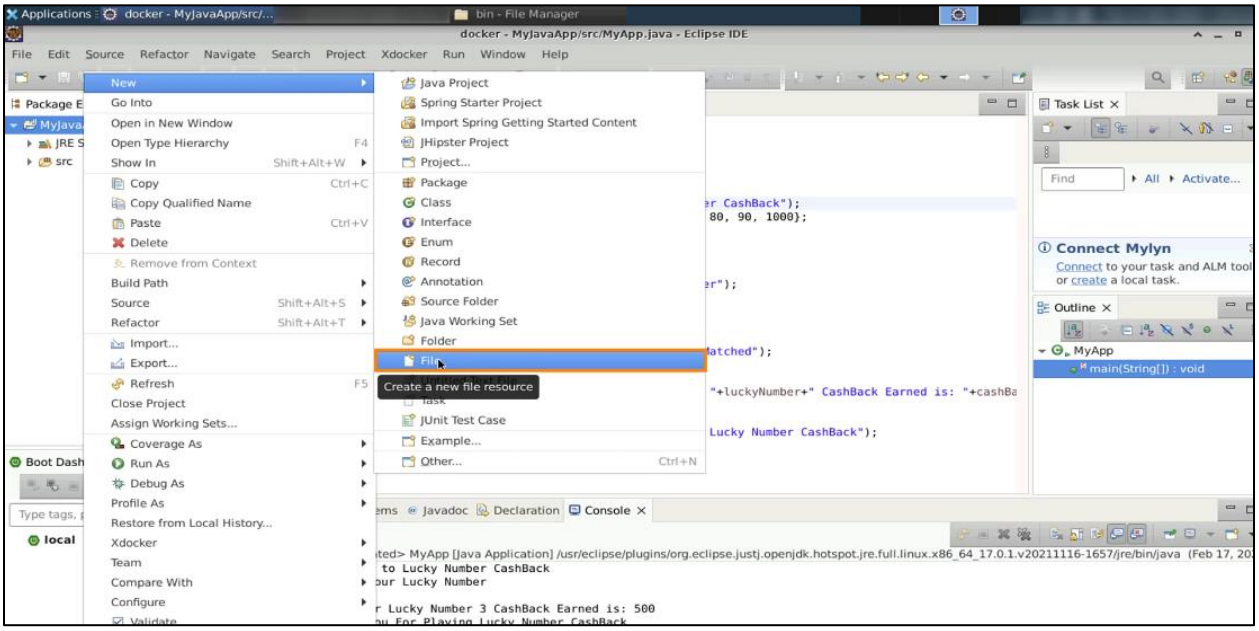


You will see that both containers are now in the exited state.

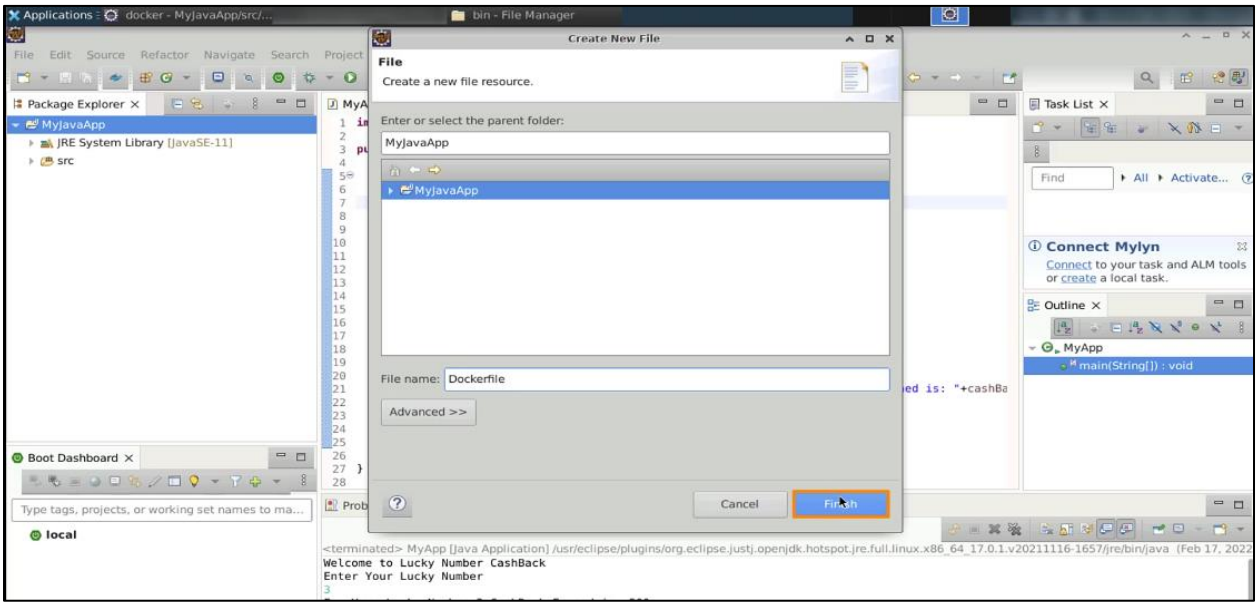
3.4 Remove the containers myapp1 and myapp2 using the **sudo docker rm myapp1** and **sudo docker rm myapp2** commands, respectively



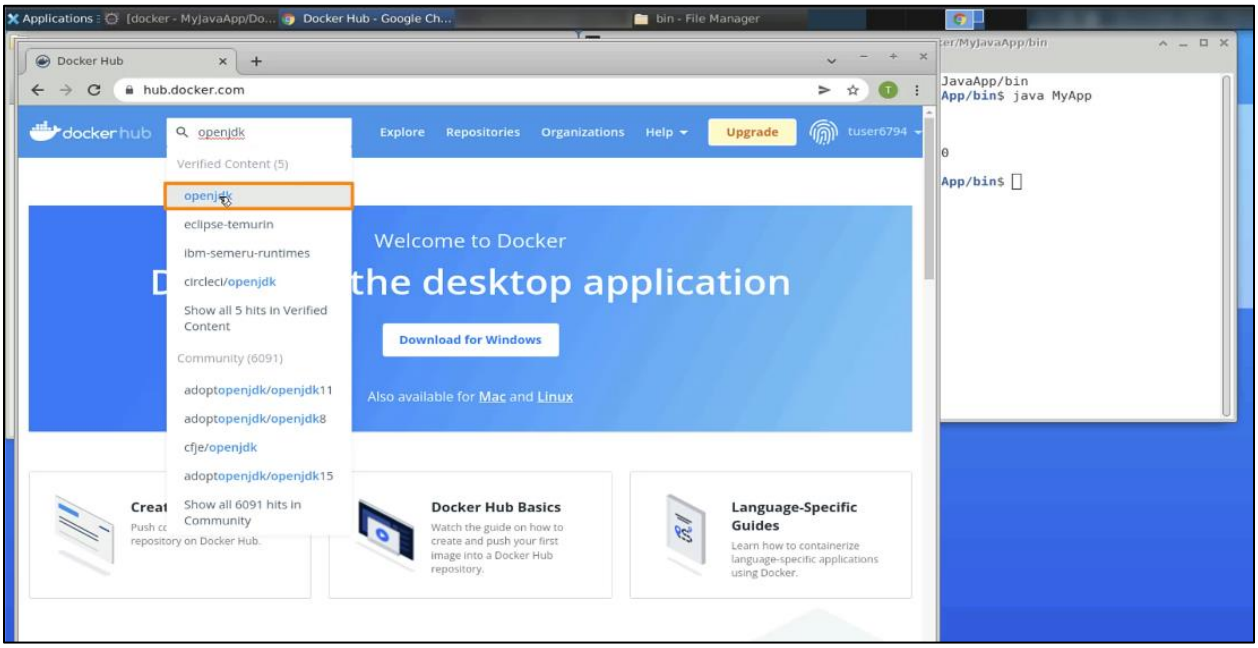
After removing the containers, no containers will be available, but Docker images will still exist. The container lifecycle is straightforward: it is created, moves to the running state, and can be paused, stopped, or removed.



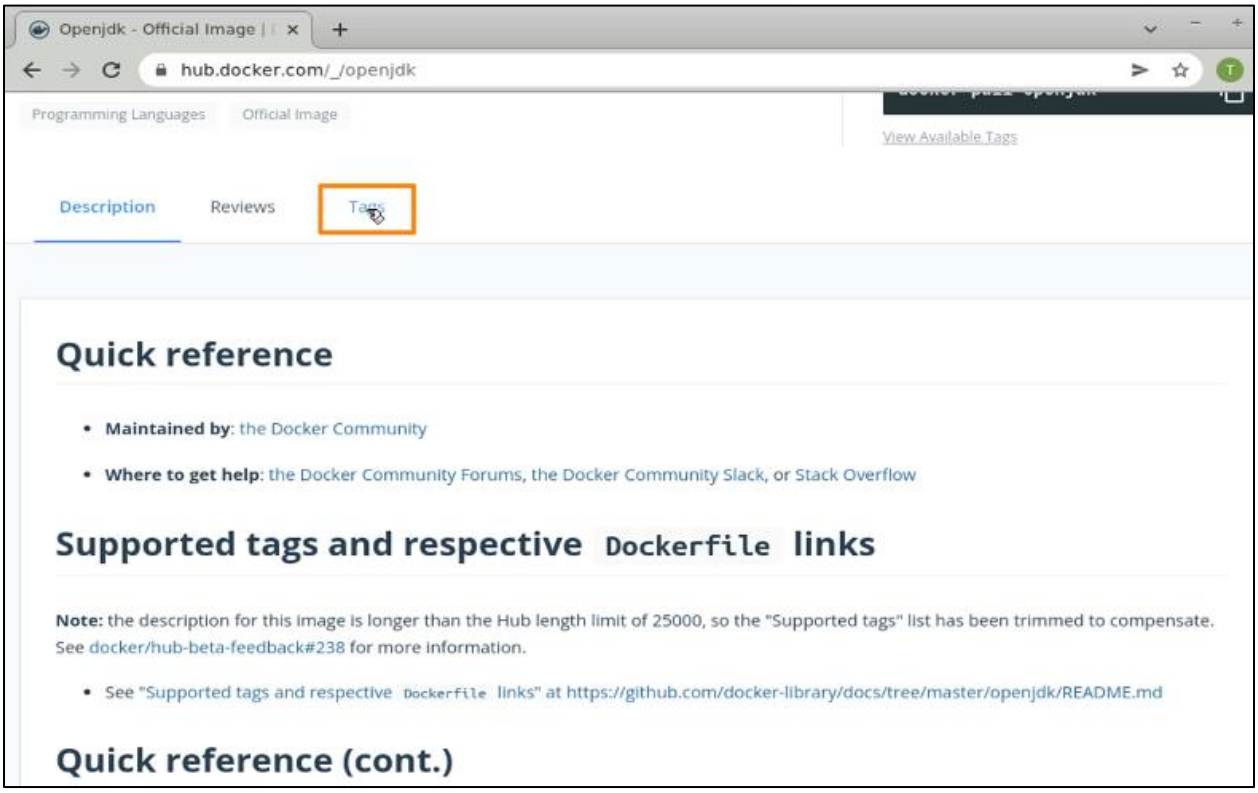
3.5 Name the file as **Dockerfile** and click **Finish**



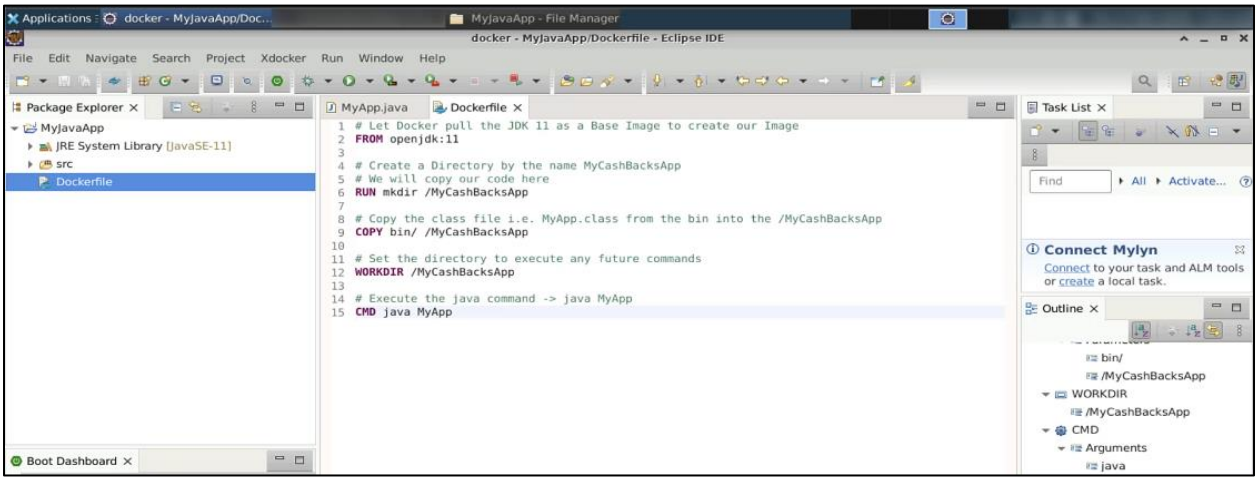
3.6 Open a web browser, navigate to **hub.docker.com,** and search for **openjdk**



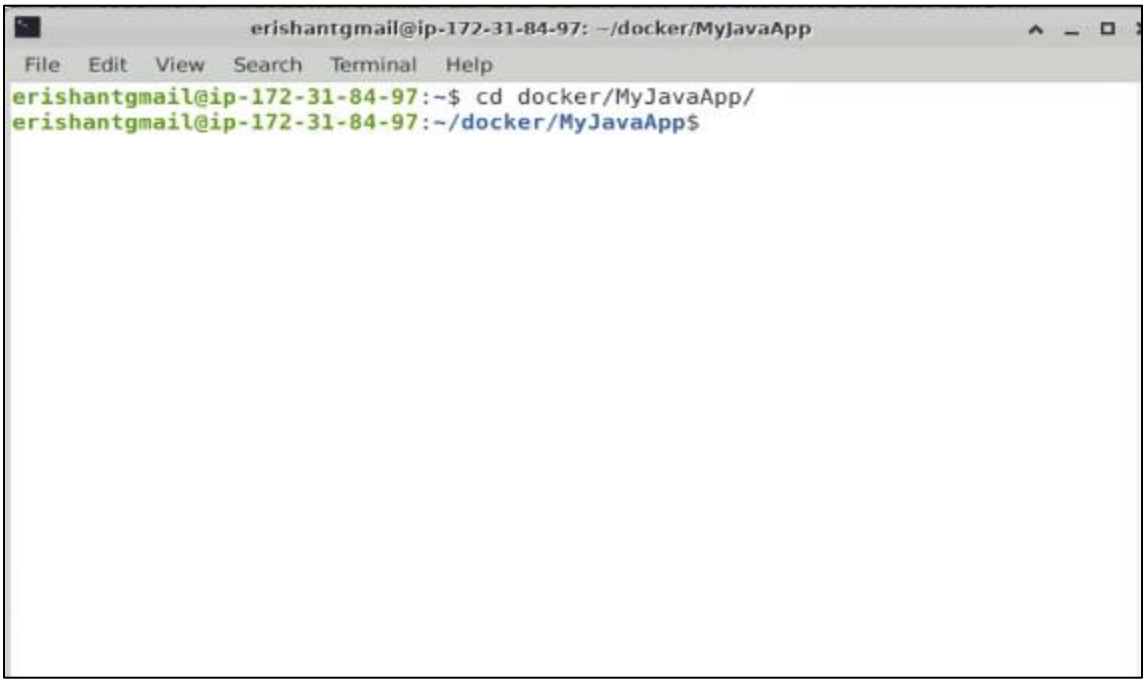
3.7 Check for the available tags, specifically for JDK 11



3.8 Go back to **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

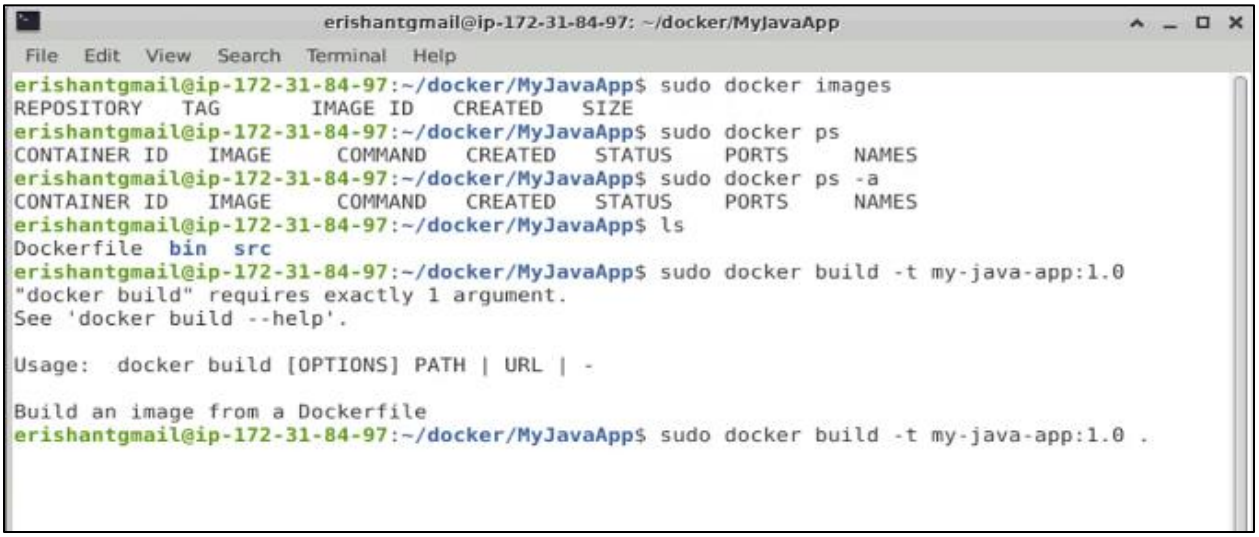


3.9 Open a new terminal window and navigate to the Docker directory and the project's root folder



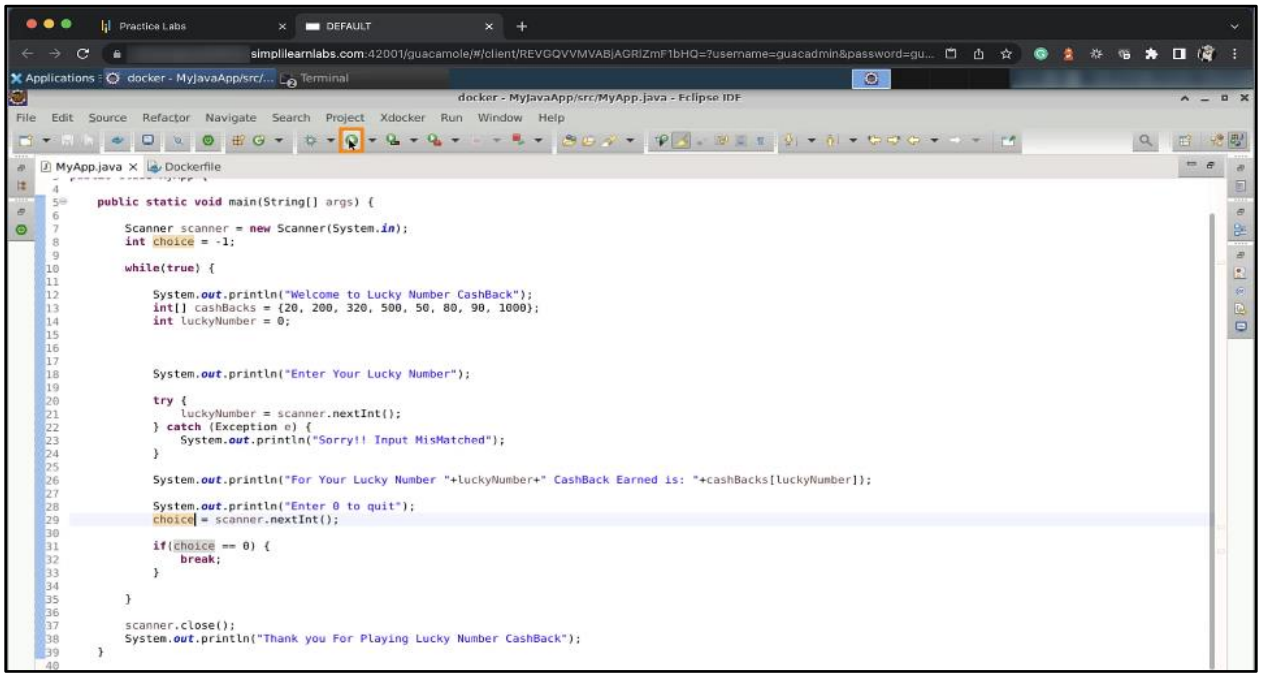
3.10 Use the **docker build** command with the tag parameter to create a Docker image. Run the following command:

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

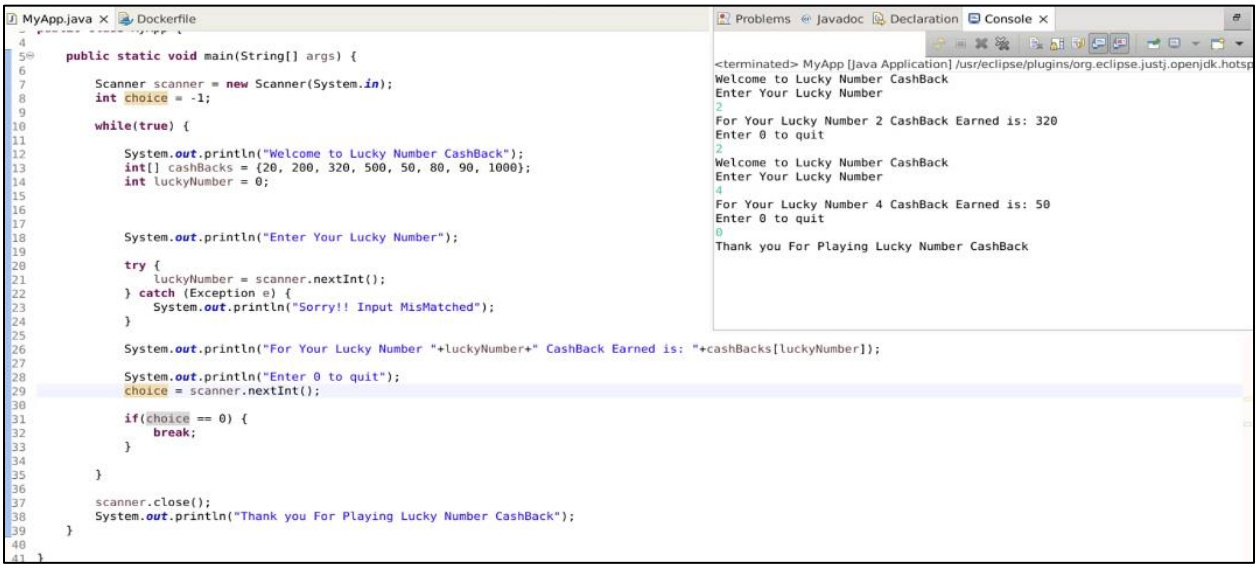


Note: The dot at the end specifies the path to the Dockerfile.

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



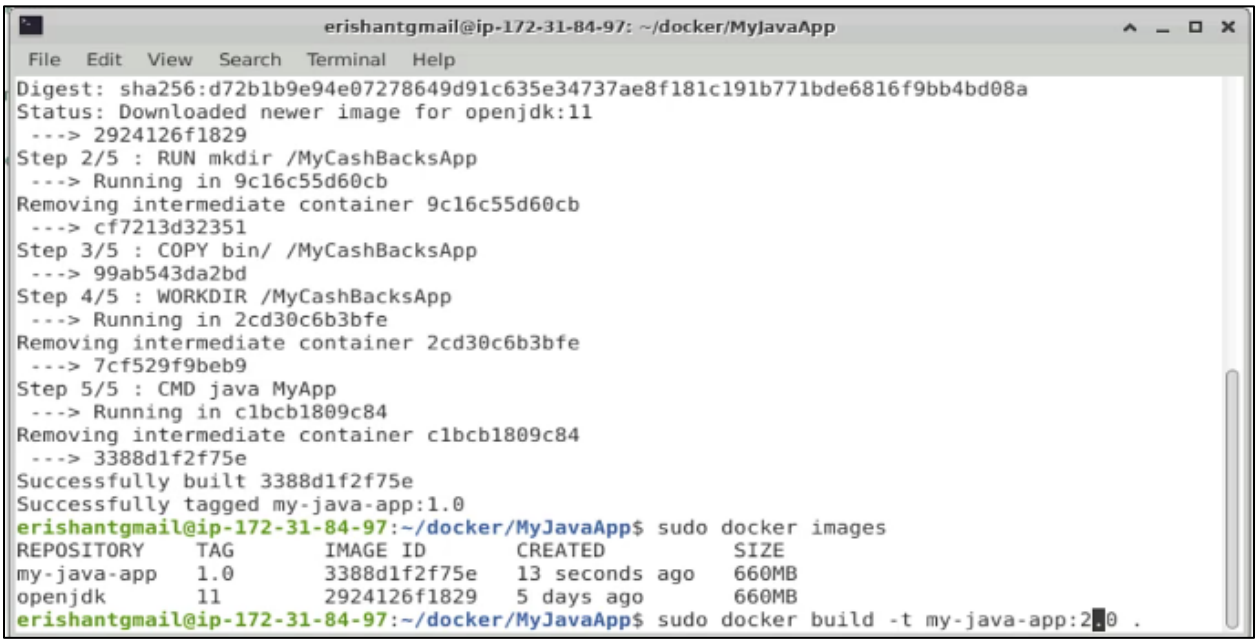
3.12 Run the modified Java program and test its functionality by entering various lucky numbers. The program should keep asking for a new number until the user enters **0**, upon which it will display the message **Thank you For Playing Lucky Number CashBack**.



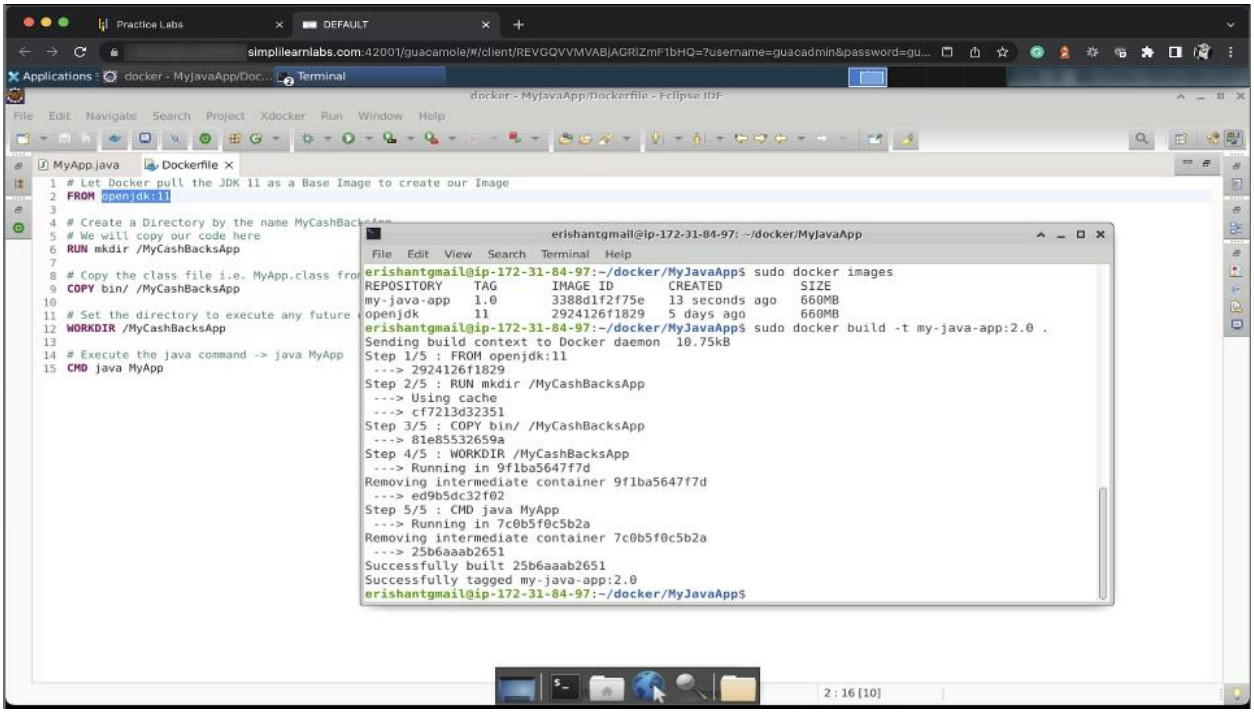
Step 4: Rebuilding the image

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 .



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



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.