**Name: Parmita Patre**

**Class: Final Year CSE Div: B**

**Roll: B-58**

**Assignment 7**

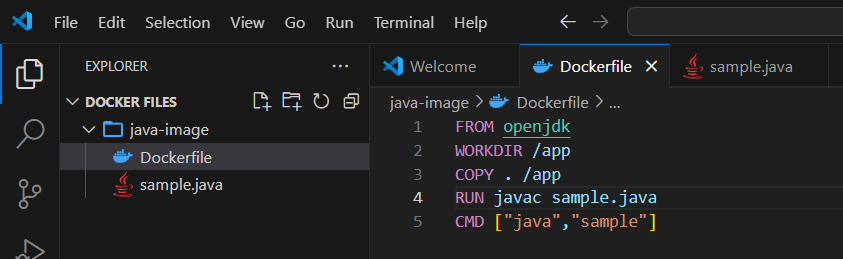
**Problem Statement:** Build a Docker image and push it to Docker Hub. Pull it on another system.Create and run a new container from an image.

**Building a docker image:**

Dockerfile:

A Dockerfile is a script that contains a set of instructions to build a Docker image for an application. It typically includes:

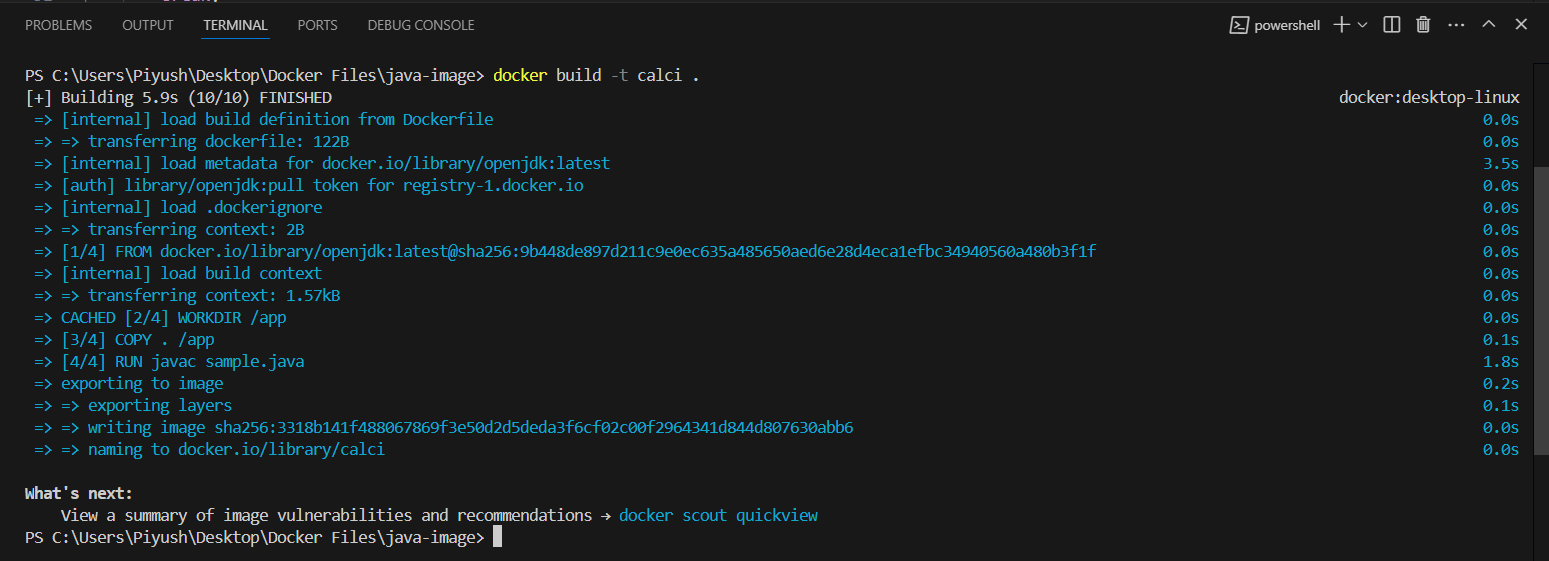
* FROM openJDK: Uses an OpenJDK image as the base, providing a Java runtime environment.
* WORKDIR /app: Sets /app as the working directory within the container where all commands will execute.
* COPY . /app: Copies the project files from your local directory to the /app directory inside the container.
* RUN javac sample.java: Compiles the sample.java file to produce a sample.class bytecode file.
* CMD ["java", "sample"]: Runs the compiled sample program when the container starts, using java to execute the sample.class.



Building the docker image:

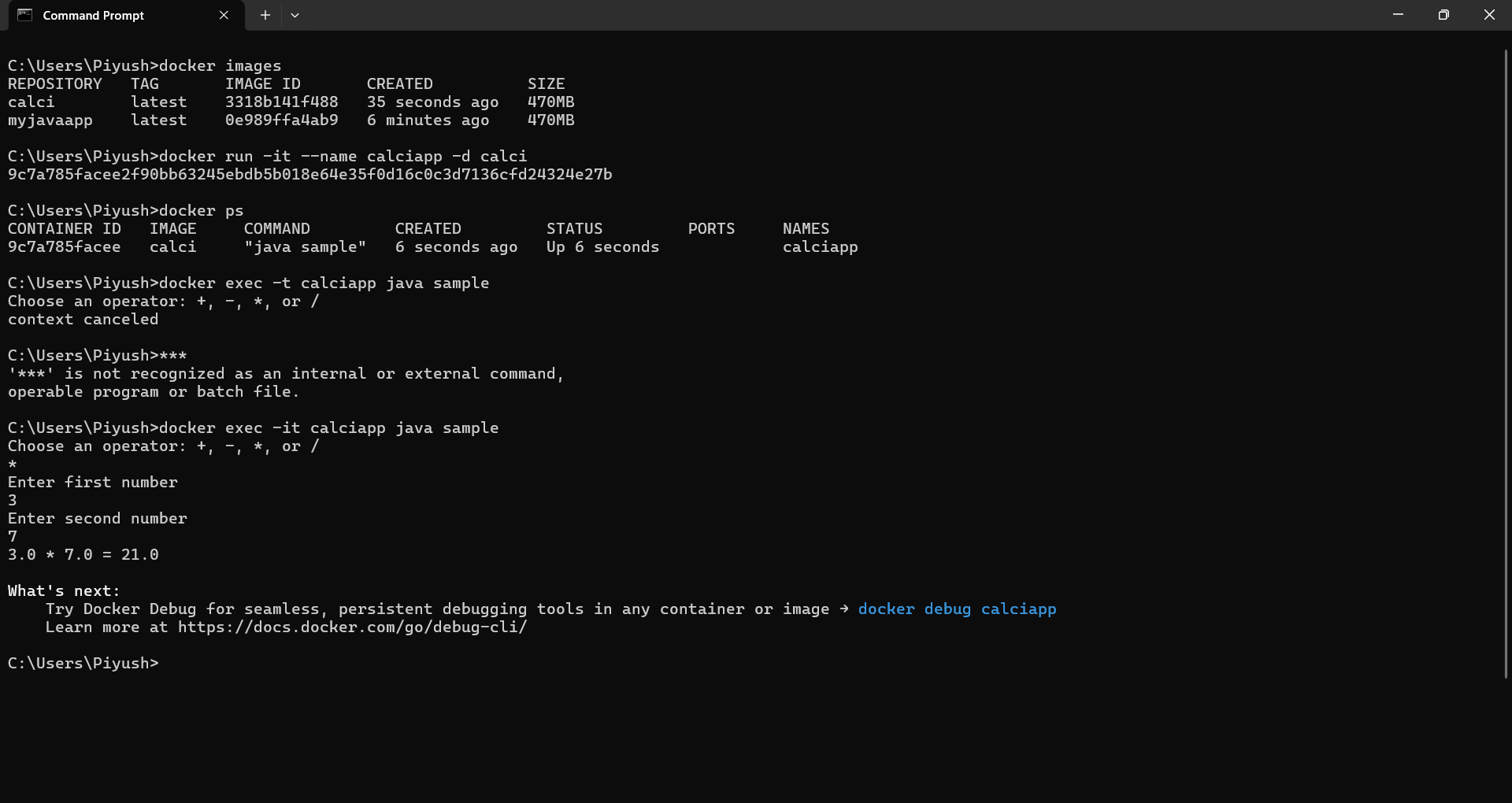
The command docker build -t <tag-name> . is used to build a Docker image from a Dockerfile located in the current directory. Here's a breakdown:

* docker build: Initiates the build process to create a Docker image.
* -t <tag-name>: Tags the image with a specified name, making it easier to reference. For example, -t calci would name the image "calci."
* .(dot) : Specifies the build context, which is the current directory, where Docker will look for the Dockerfile and any other files required to build the image.



Creating a container and running the docker image:

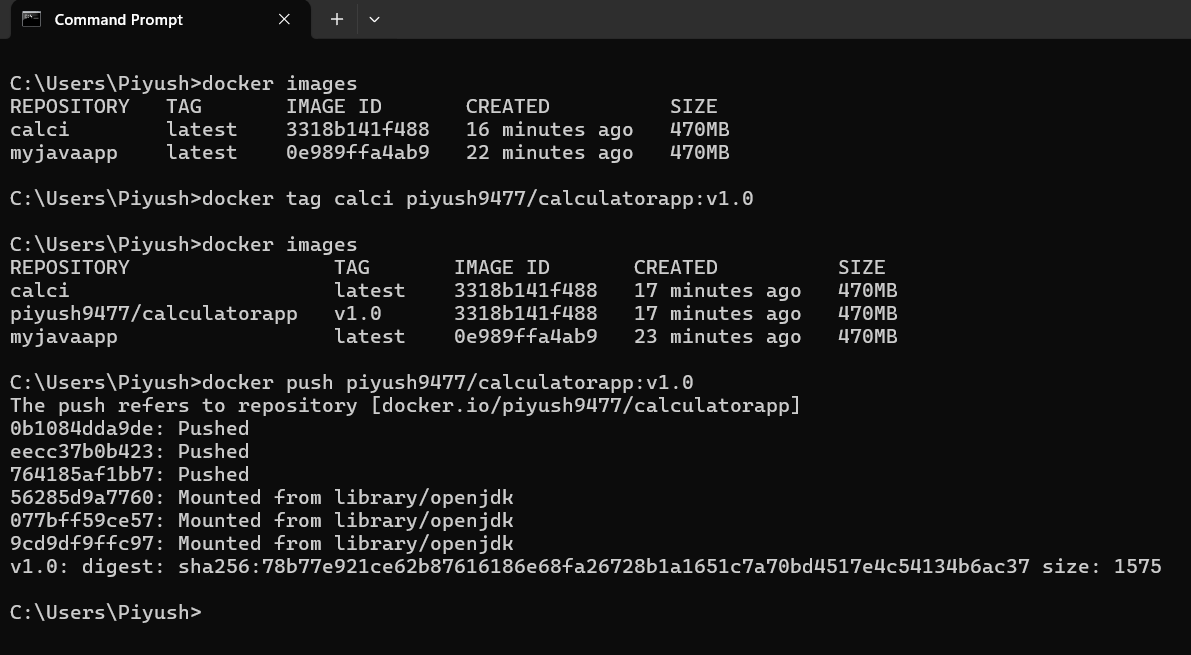
* docker images: Lists all Docker images available on your system, showing details like repository name, tag, image ID, creation date, and size.
* docker run -it --name <name> -d <tag-name>: Runs a Docker container from the specified image (tagged <tag-name>) in detached mode (-d), giving it a specific name (--name <name>), and enabling interactive mode (-it), which allows user input.
* docker ps: Displays all currently running containers with details like container ID, names, image used, status, ports, and command.
* docker exec -it <name> java sample: Executes the command java sample inside the running container named <name>, with -it enabling interactive mode to see output and provide input if needed. This is typically used to run Java programs or other commands inside a live container.



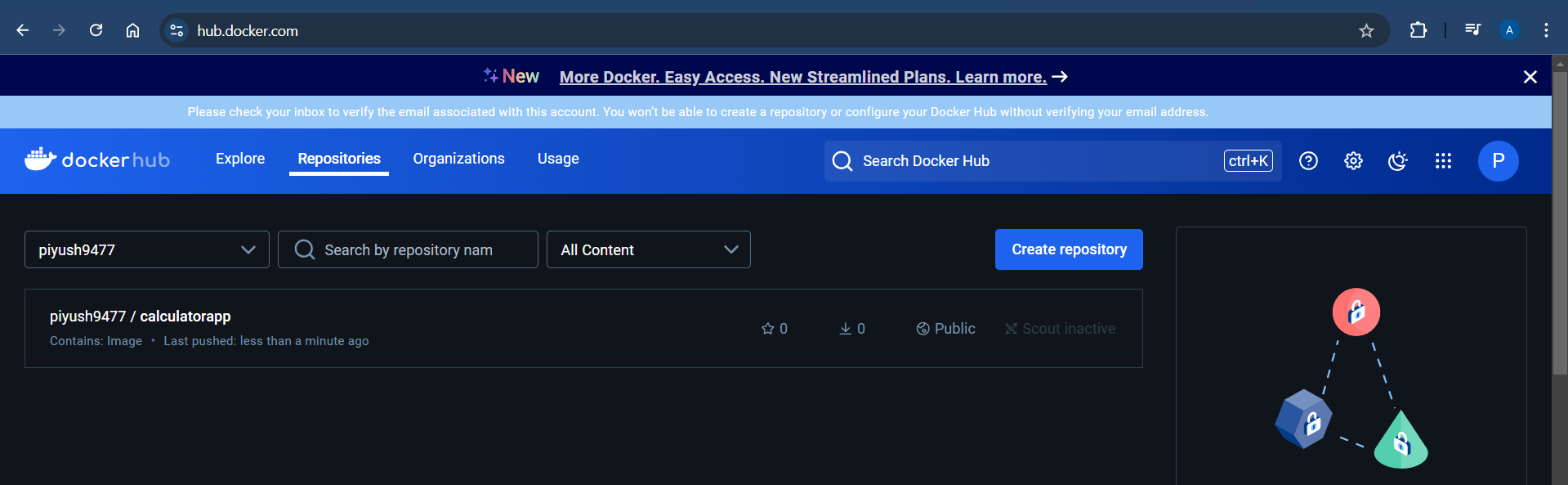
**Pushing docker image to Docker Hub:**

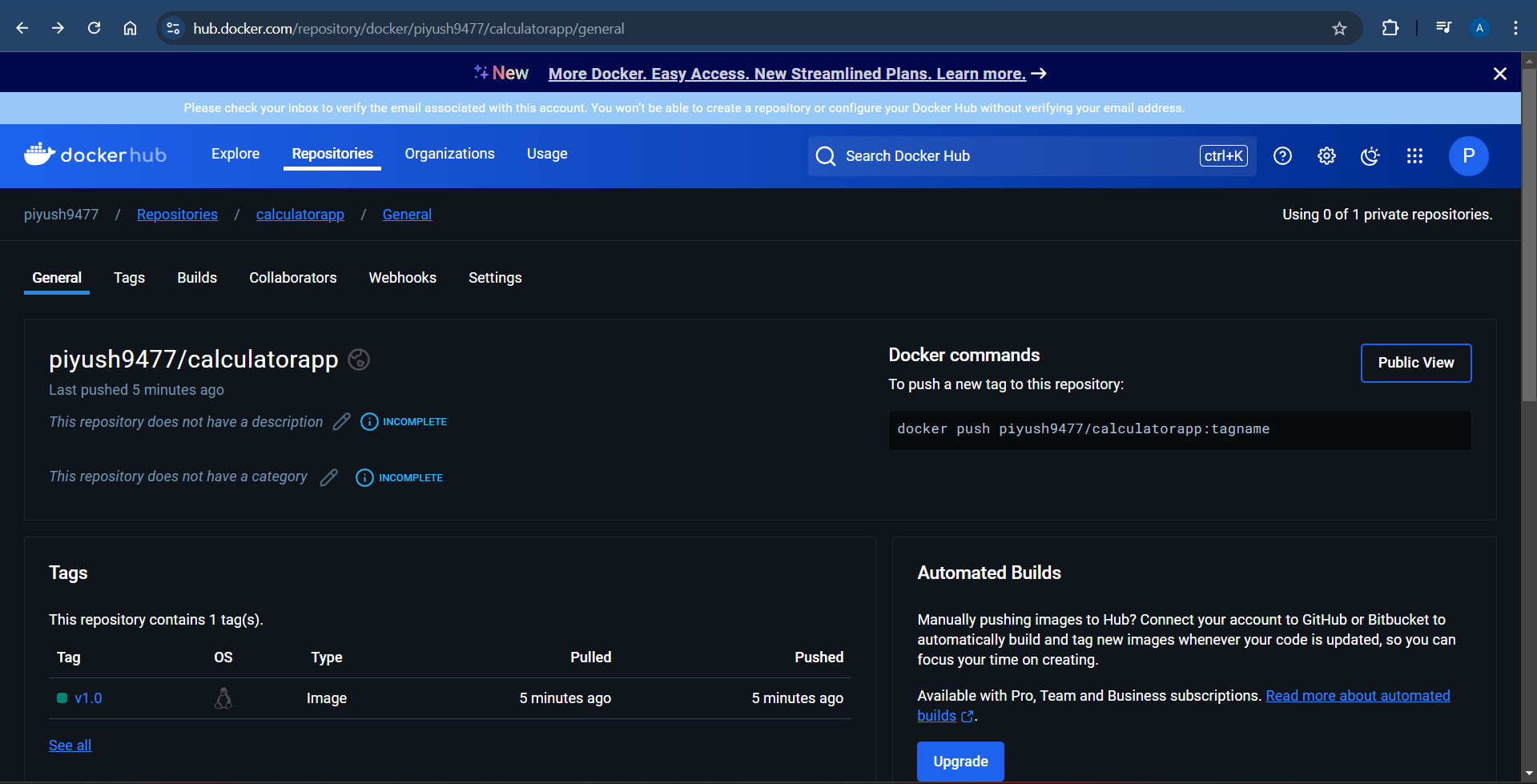
Logging in and pushing image using “docker push”:

* docker tag <source-image> <repository:tag>: Creates a new tag for an existing Docker image. This is useful for renaming images or preparing them for upload to a registry. For example, docker tag myapp:latest myrepo/myapp:v1.0 tags myapp for the repository myrepo with version v1.0.
* docker push <repository:tag>: Uploads the specified image (identified by <repository:tag>) to a Docker registry, such as Docker Hub. This allows the image to be shared or used on other systems. For example, docker push myrepo/myapp:v1.0 pushes the image to the repository myrepo.



**Docker Hub screenshots after pushing:**

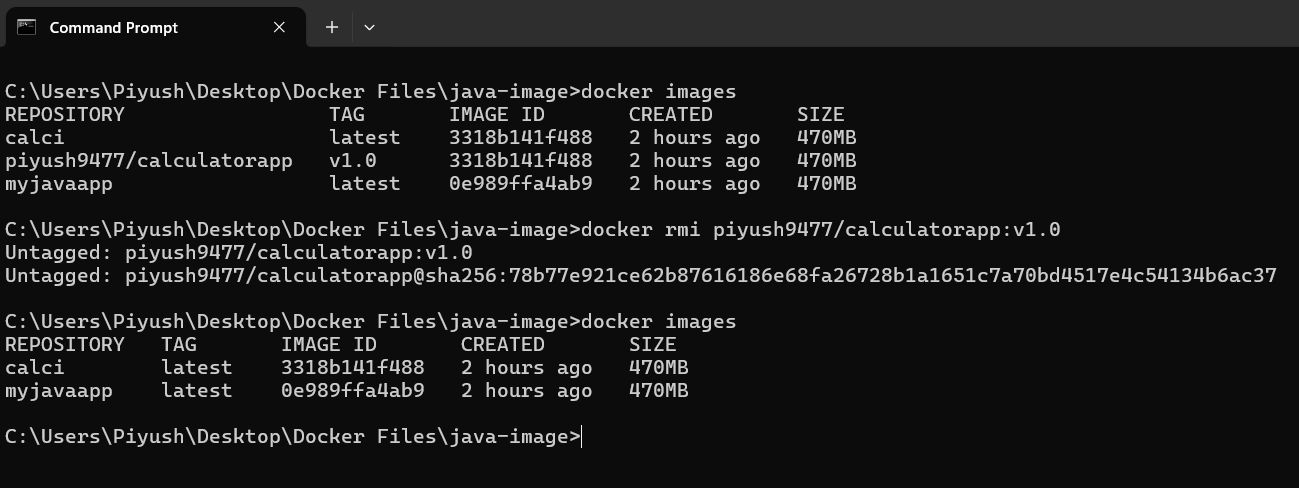




**Pulling the docker image from Docker Hub:**

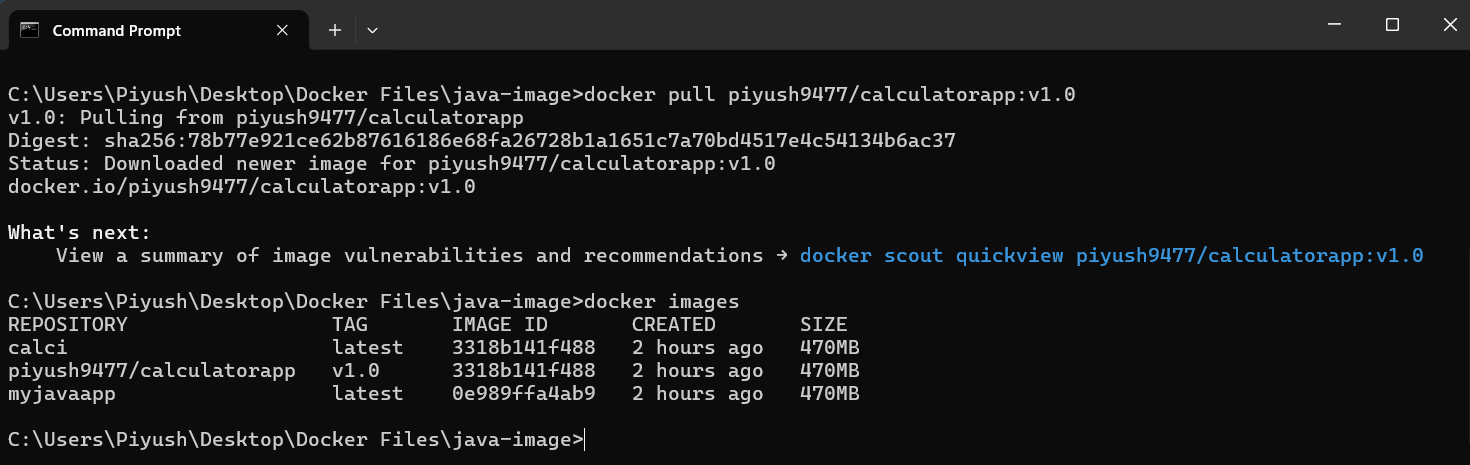
Deleting existing image from local machine:

* docker rmi <image>: This command is used to remove one or more Docker images from your local system. You can specify the image by its name, tag, or image ID. For example, docker rmi myapp:latest will remove the image tagged as latest from the myapp repository.

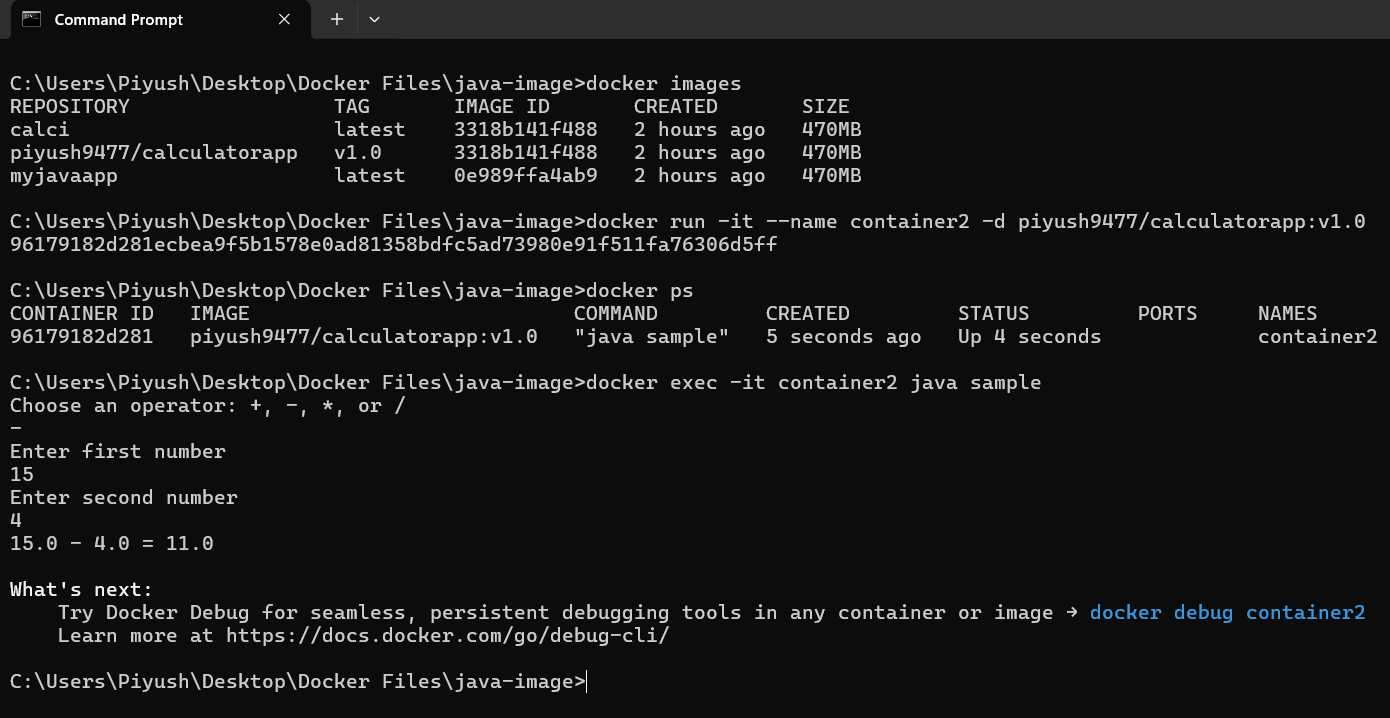


Pulling the image from Docker Hub using “docker pull”:

* docker pull <repository:tag>: This command is used to download a Docker image from a Docker registry (such as Docker Hub) to your local system. You can specify the image by its repository name and tag.



**Creating and running a new container from an image:**



**Conclusion:**

In this assignment, we successfully built a Docker image and pushed it to Docker Hub, demonstrating effective image management. We first deleted the image form local system and then pulled that image from docker hub, showcasing Docker's portability. Finally, we created and ran a new container from the image, illustrating Docker's ease of application deployment. This process highlights the efficiency and flexibility of using Docker in modern software development.