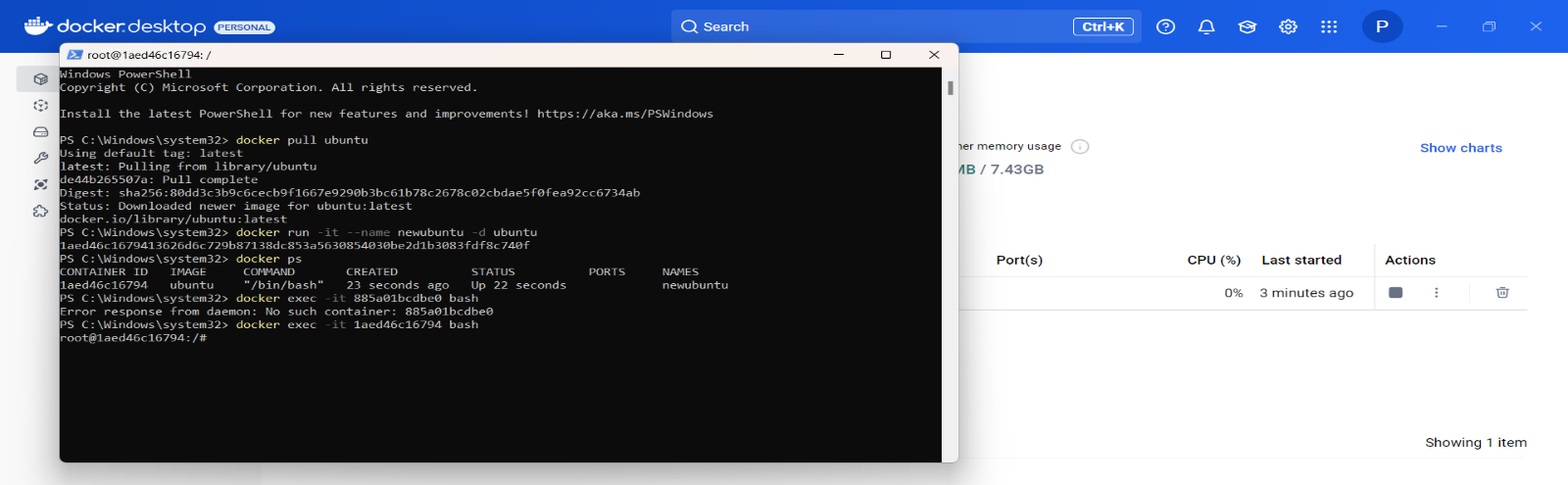
**Docker Basic Commands and Image Creation**

**Docker Basic Commands:-**

**1. Pull the Ubuntu image**

docker pull ubuntu

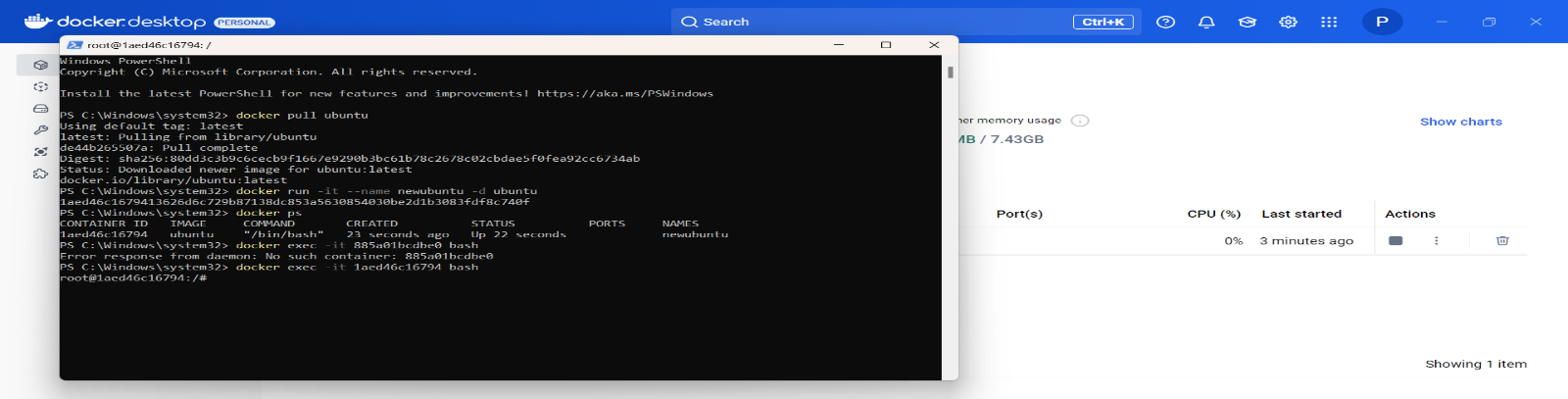


**•What it does**: Downloads the official Ubuntu base image from Docker Hub to your local

system. This image is like a minimal operating system ready to run inside a Docker container.

**2. Run a container from the Ubuntu image**

**docker run -it --name newubuntu -d ubuntu**



**•What it does**: Creates and starts a new container from the Ubuntu image.

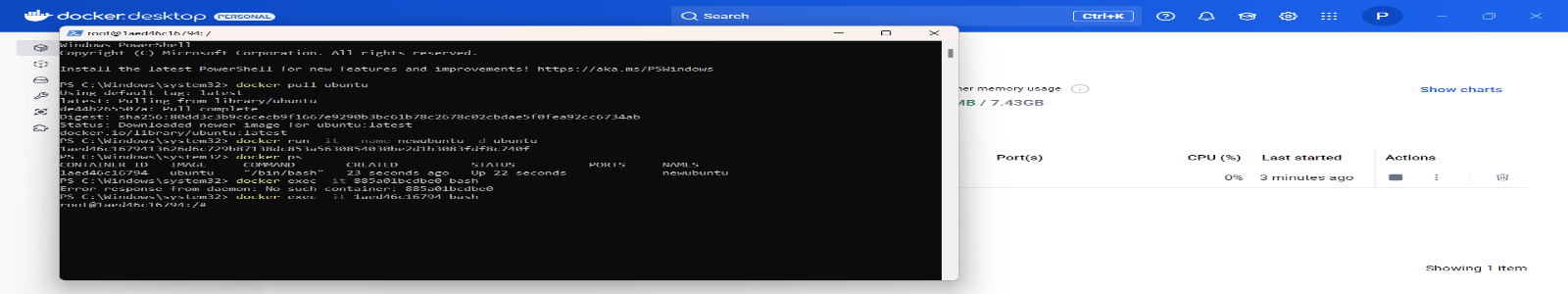
•-it: Allows you to interact with the container (interactive terminal mode).

•--name newubuntu: Names the container "newubuntu" for easy identification.

•-d: Runs the container in the background (detached mode).

**3. List all running containers**

**docker ps**

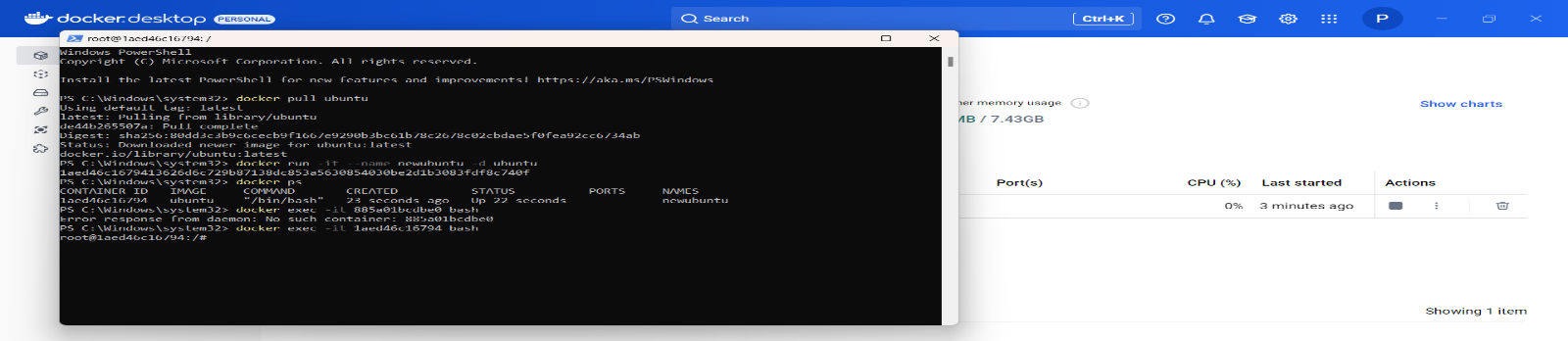


•**What it does**: Displays a list of all currently running containers, showing details like the

container ID, name, image used, and uptime.

**4. Access the running container**

**docker exec -it 885a01bcdbe0 bash**



**•What it does:** Opens a shell (terminal) inside the running container.

•exec: Executes a command in a running container.

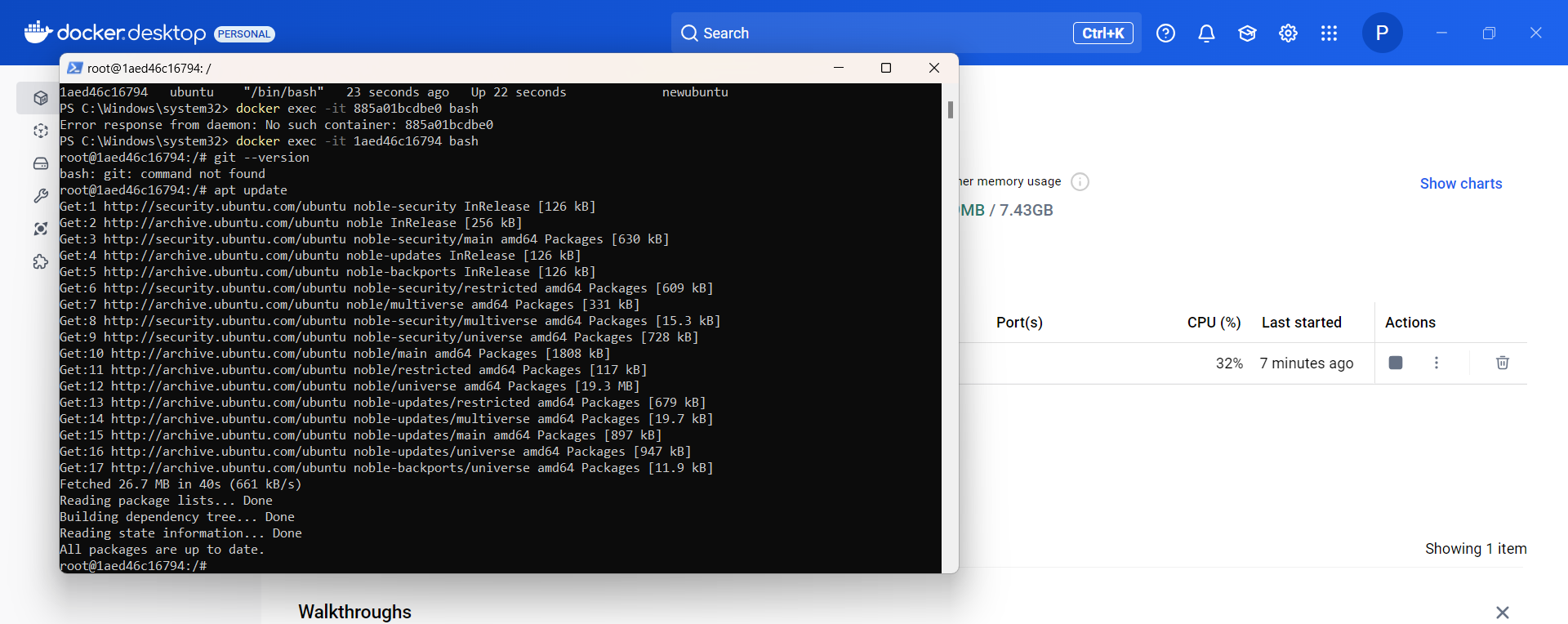
•-it: Allows interactive access.

•885a01bcdbe0: The unique container ID of the running container.

•bash: Opens the bash shell inside the container.

**5. Check if Git is installed**

**git --version**

****

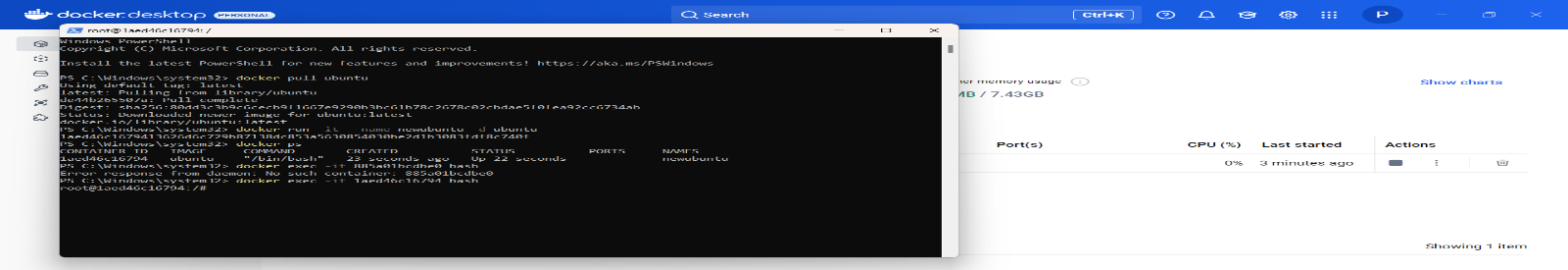
**•What it does:** Checks the version of Git installed in the container.

**•Why it failed:** The error bash: git: command not found means Git is not installed in the

container.

**6. Update the package list**

**apt update**

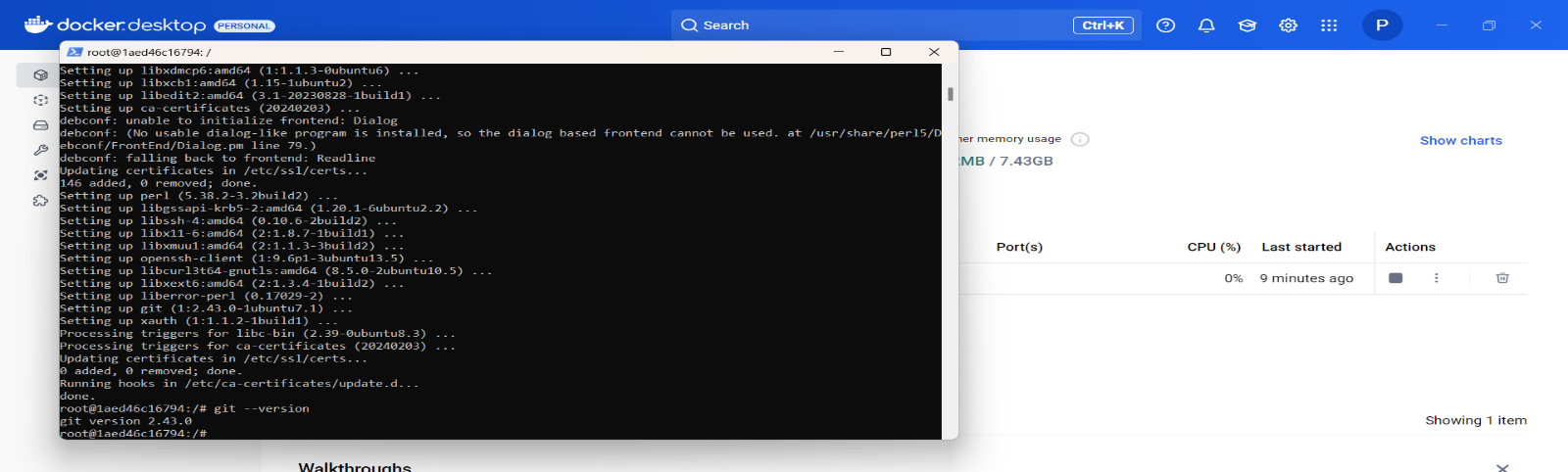


**•What it does:** Updates the list of available software packages in the container. It prepares the

system for installing new software by fetching the latest versions from online repositories.

**7. Install Git**

**apt install git -y**

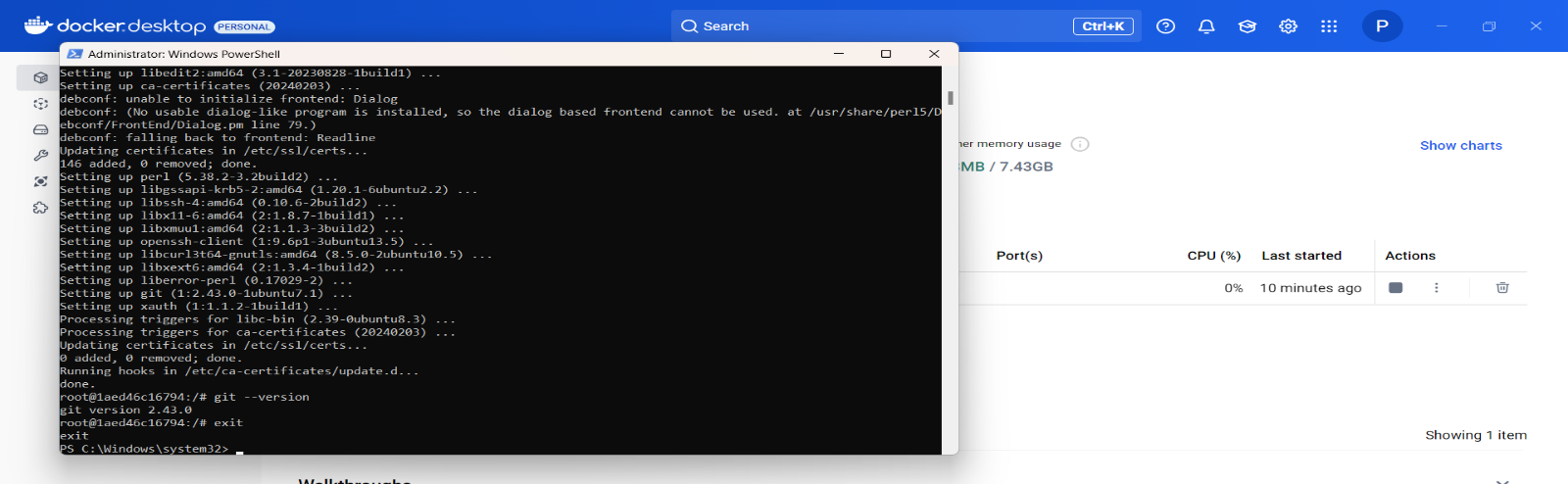
****

**•What it does:** Installs Git inside the container.

•-y: Automatically confirms the installation (avoids asking for "yes/no").

8. Verify Git installation

**git --version**

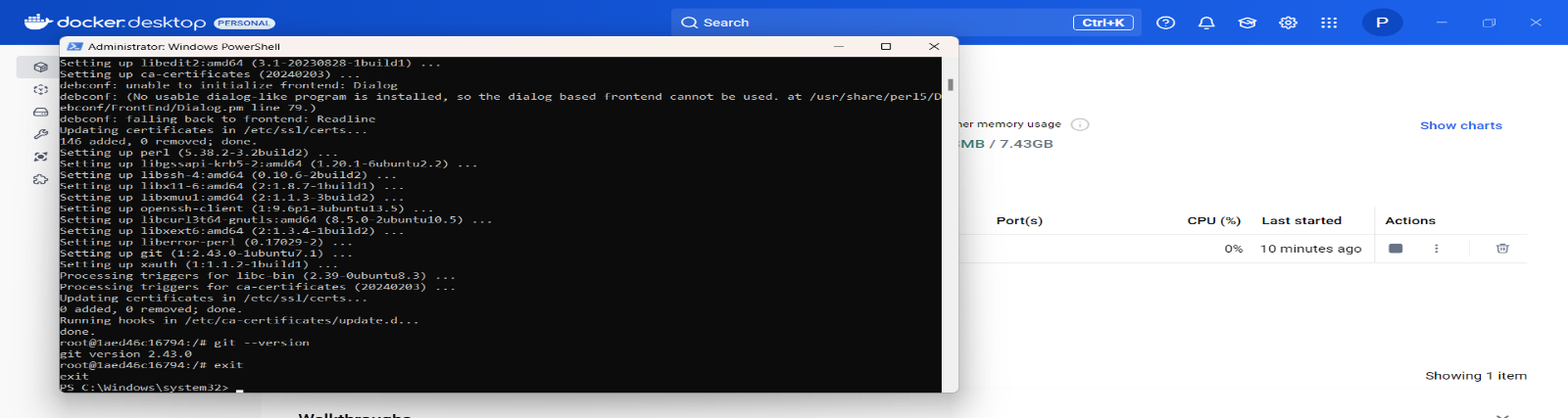
****

•What it does: Checks if Git is installed correctly and displays its version (e.g., git version

2.43.0).

**9. Exit the container**

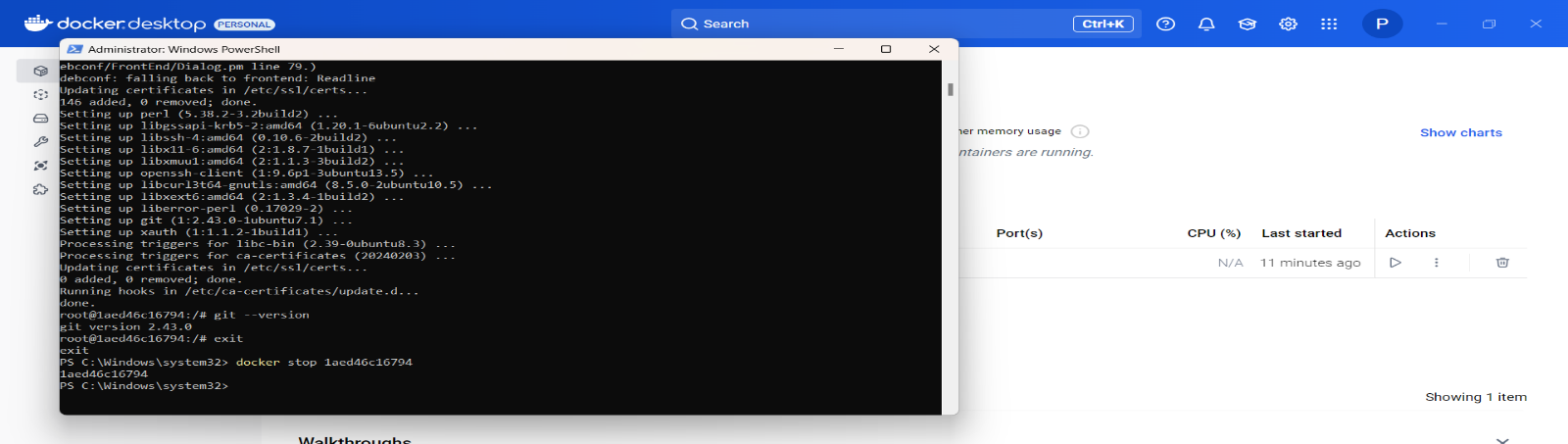
**exit**

****

**•What it does:** Closes the shell session inside the container and returns to your host system.

**10. Stop the running container**

**docker stop 885a01bcdbe0**

****

**•What it does:** Stops the running container. It doesn’t delete the container, but it halts its

operation.

**11. Save the container as an image**

**docker commit 885a01bcdbe0 penkulamaheshkumar/newubuntu2024**

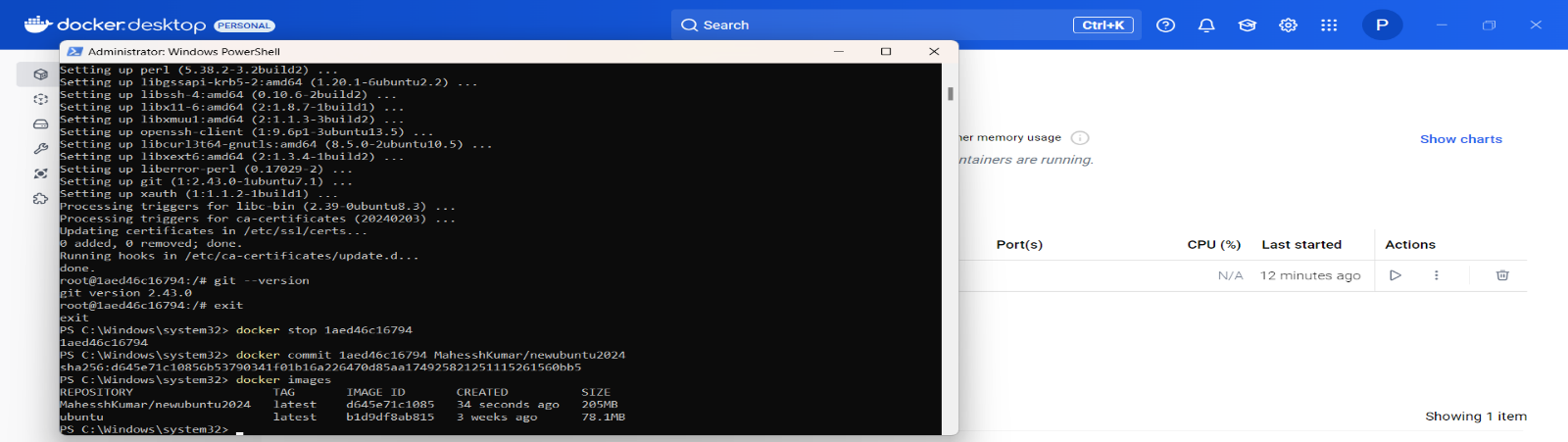
**•What it does:** Creates a new image from the stopped container with all the changes (like the

Git installation).

**•penkulamaheshkumar/newubuntu2024:** Names the new image with a custom name and tag.

**12. List all local images**

**docker images**

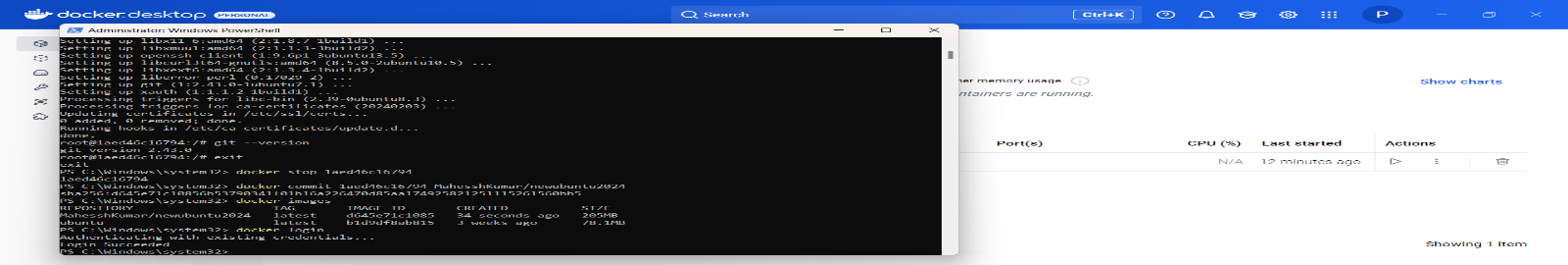
****

**•What it does:** Shows all the Docker images stored on your system, including the newly created

image penkulamaheshkumar/newubuntu2024.

**13. Log in to Docker Hub**

**docker login**

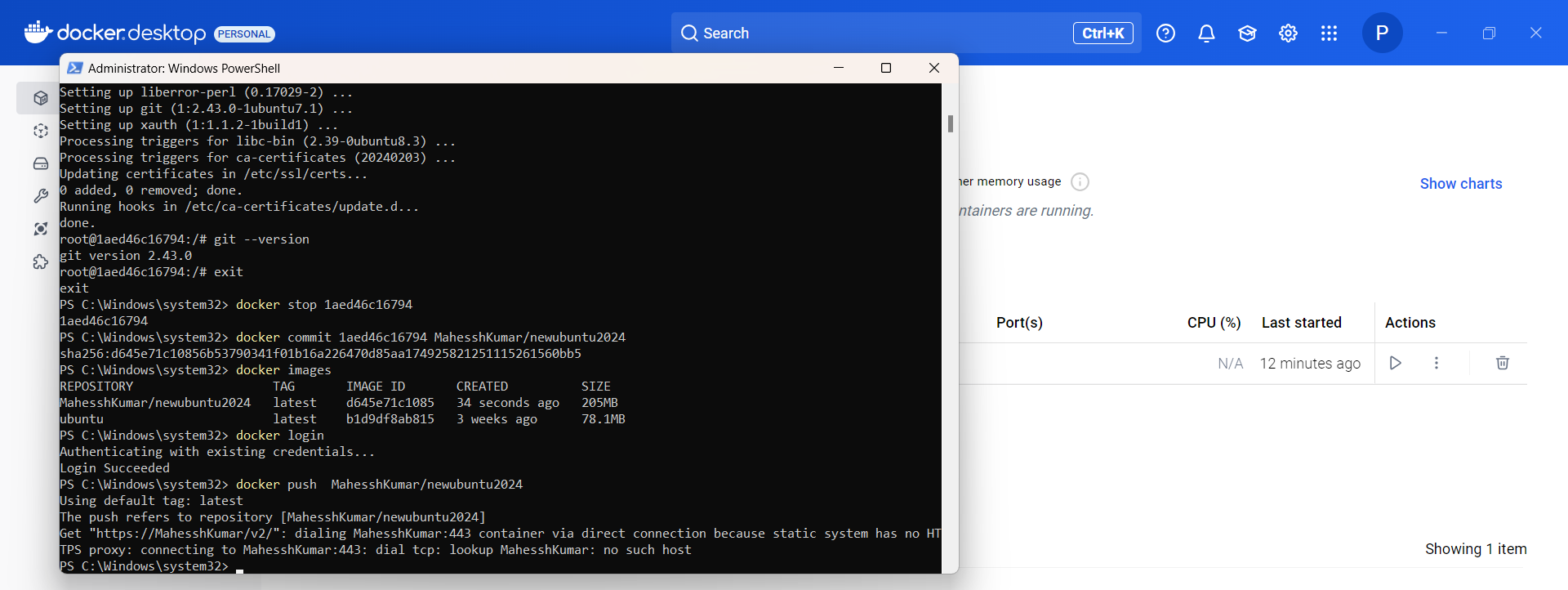
****

**•What it does:** Logs you into your Docker Hub account so you can upload (push) your image.

•It will prompt for your Docker Hub username and password.

**14. Push the image to Docker Hub**

**docker push penkulamaheshkumar/newubuntu2024**

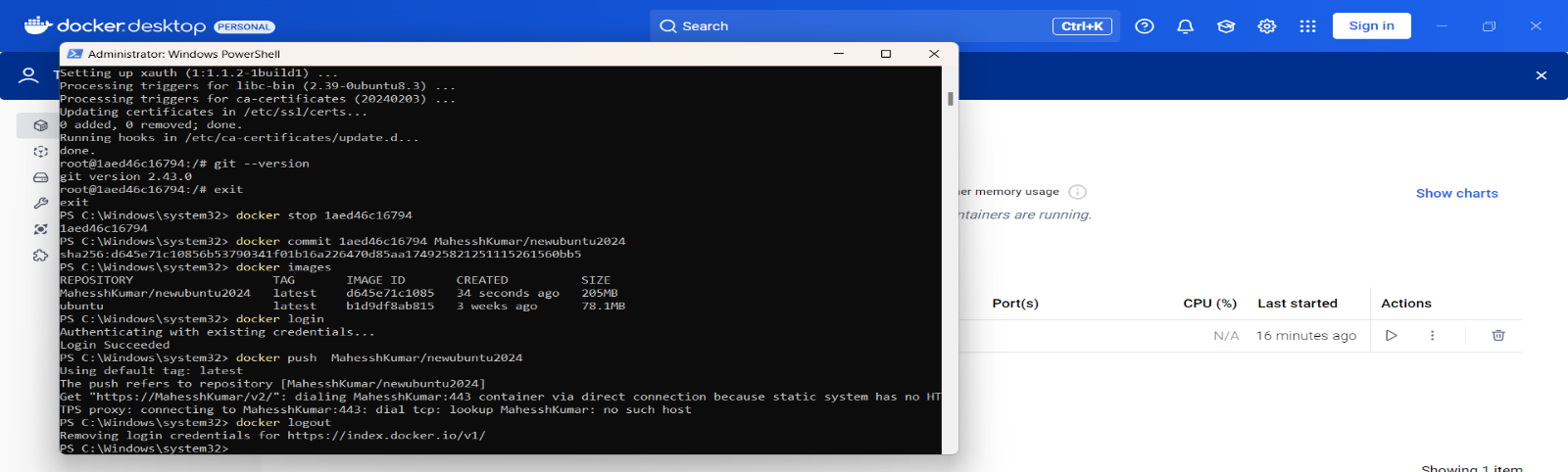
****

**•What it does:** Uploads the newly created image to your Docker Hub account so it can be

accessed from anywhere.

**15. Log out of Docker Hub**

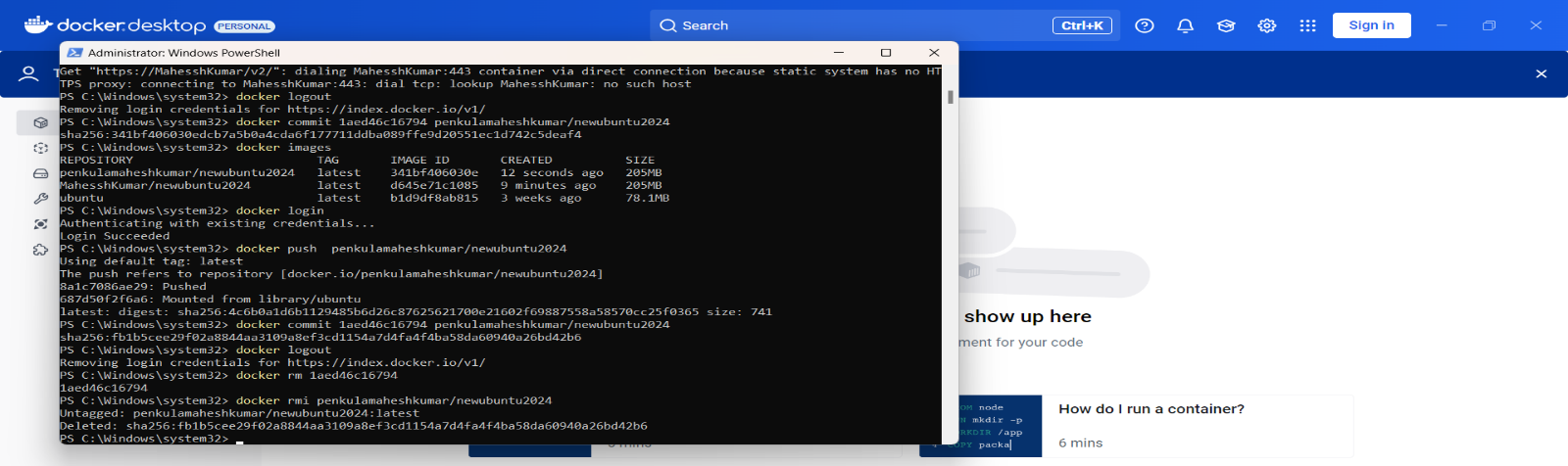
**docker logout**

****

**•What it does:** Logs you out of Docker Hub for security.

**16. Remove the container**

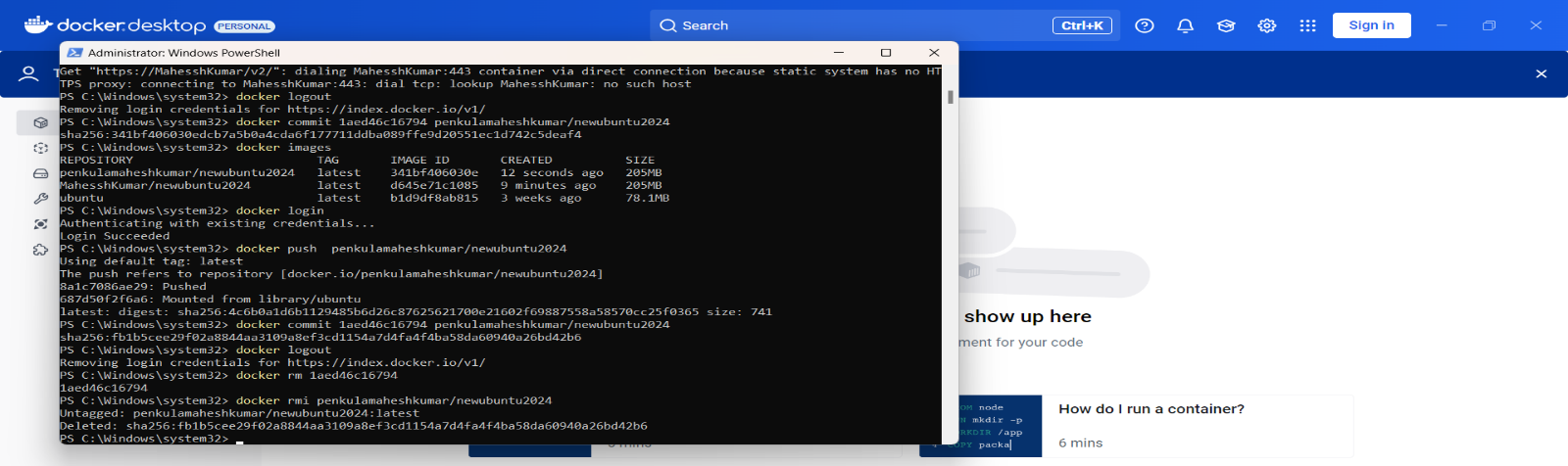
**docker rm 885a01bcdbe0**

****

**•What it does:** Deletes the stopped container permanently from your system.

**17. Remove the local image**

**docker rmi penkulamaheshkumar/newubuntu2024**

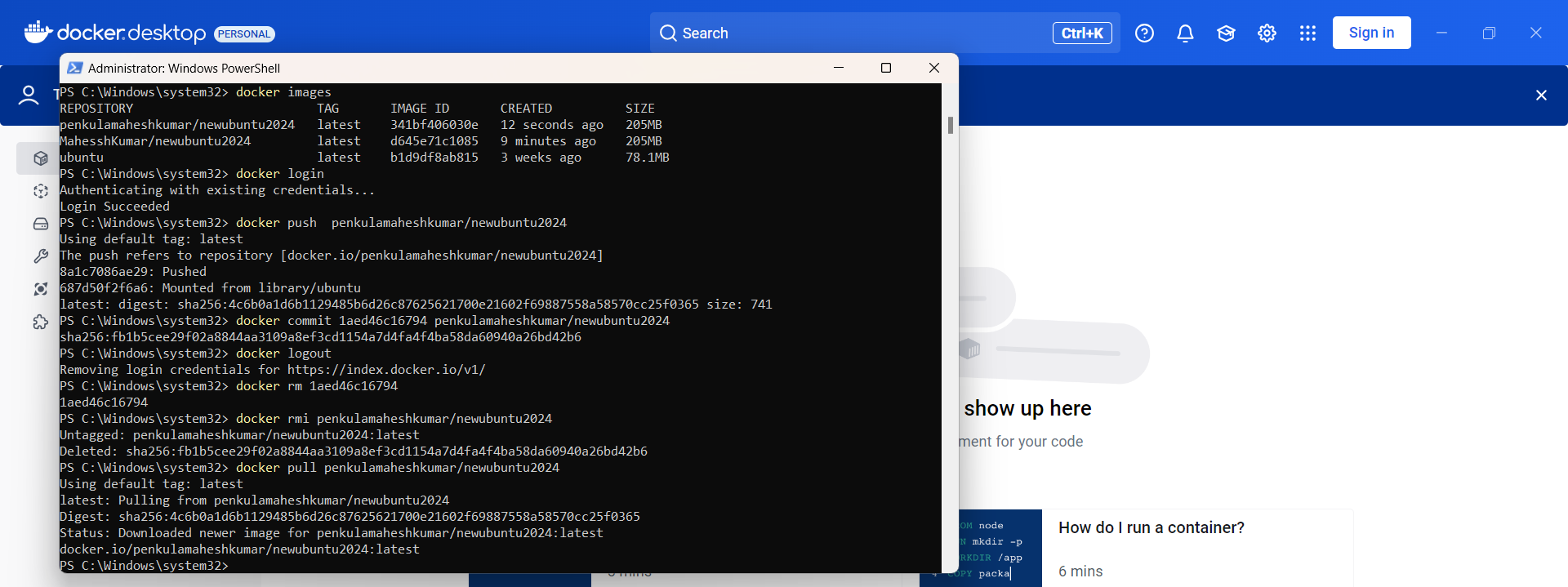
****

**•What it does:** Deletes the custom image from your local system, freeing up space. (The image

is still available on Docker Hub.)

**18. Pull the image from Docker Hub**

**docker pull penkulamaheshkumar/newubuntu2024**

****

**•What it does:** Downloads the custom image penkulamaheshkumar/newubuntu2024 from Docker

Hub to your local system.

**Image Creation**

**Step 1: Create the JavaScript File**

Create a file named calculator.js in your project directory.

Add the following code to define simple calculator functions:

**// calculator.js**

**function add(a, b) {**

**return a + b;**

**}**

**function subtract(a, b) {**

**return a - b;**

**}**

**function multiply(a, b) {**

**return a \* b;**

**}**

**function divide(a, b) {**

**if (b === 0) {**

**return "Cannot divide by zero!";**

**}**

**return a / b;**

**}**

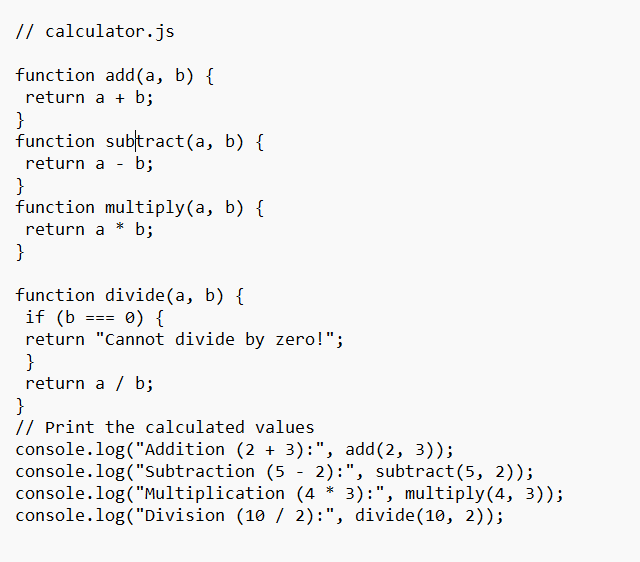
**// Print the calculated values**

**console.log("Addition (2 + 3):", add(2, 3));**

**console.log("Subtraction (5 - 2):", subtract(5, 2));**

**console.log("Multiplication (4 \* 3):", multiply(4, 3));**

**console.log("Division (10 / 2):", divide(10, 2));**



**Purpose:** This script performs basic arithmetic operations and logs the results to the console.

**Output:** When run, this program prints the results of the calculations.

**Step 2: Create a Dockerfile**

The Dockerfile contains instructions for building the Docker image.

1. Create a file named Dockerfile (no file extension).

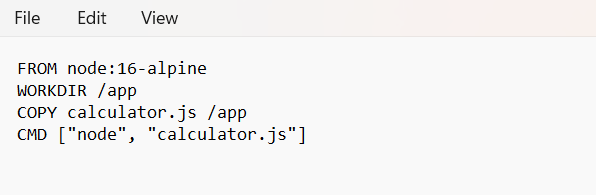
2. Add the following content:

**FROM node:16-alpine**

**WORKDIR /app**

**COPY calculator.js /app**

**CMD ["node", "calculator.js"]**

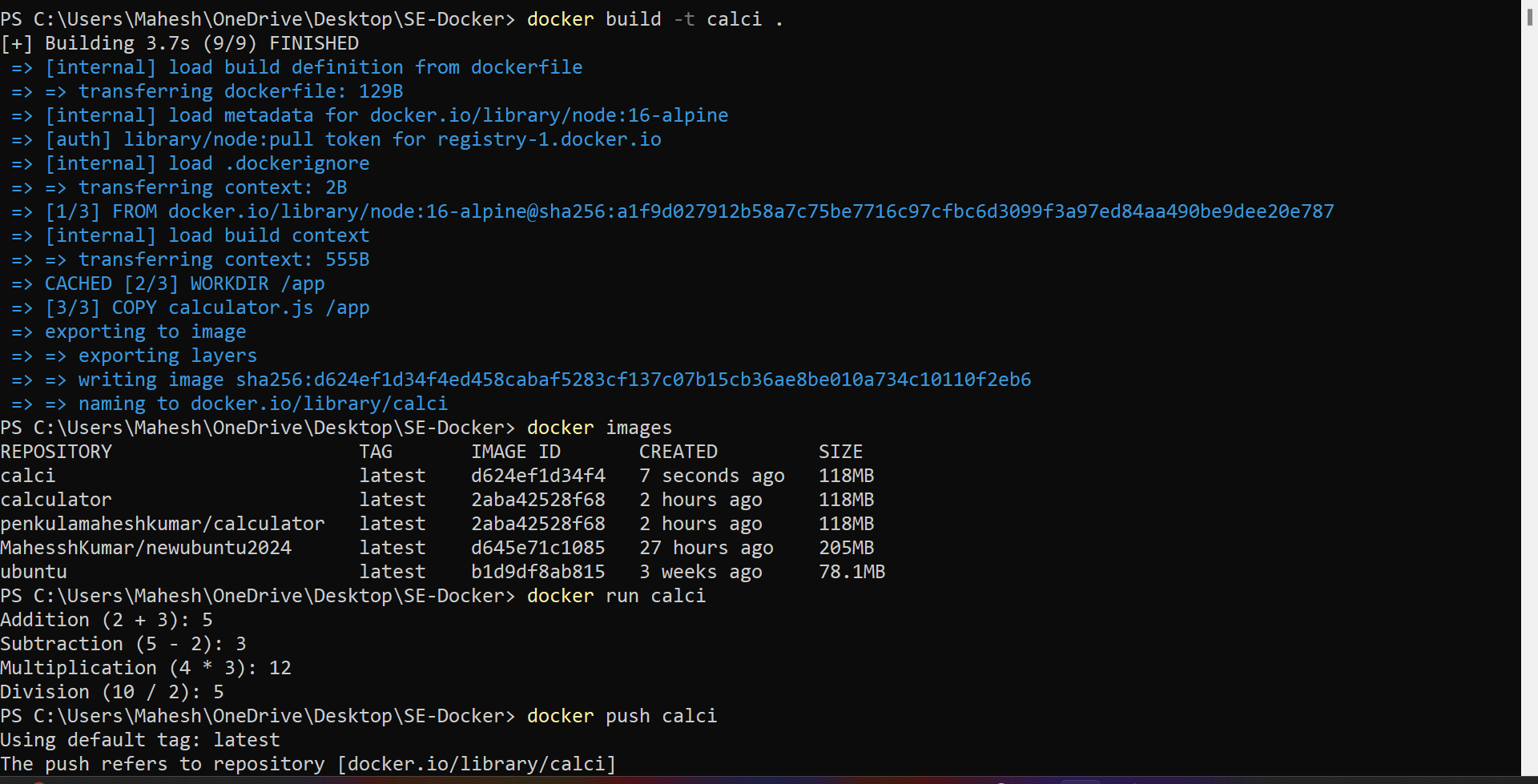
****

**Step 3: Build the Docker Image**

1. Open a terminal in the directory containing your Dockerfile and calculator.js.

2. Run the following command:

**docker build -t simple-calculator** .



• docker build: This command builds an image from the Dockerfile.

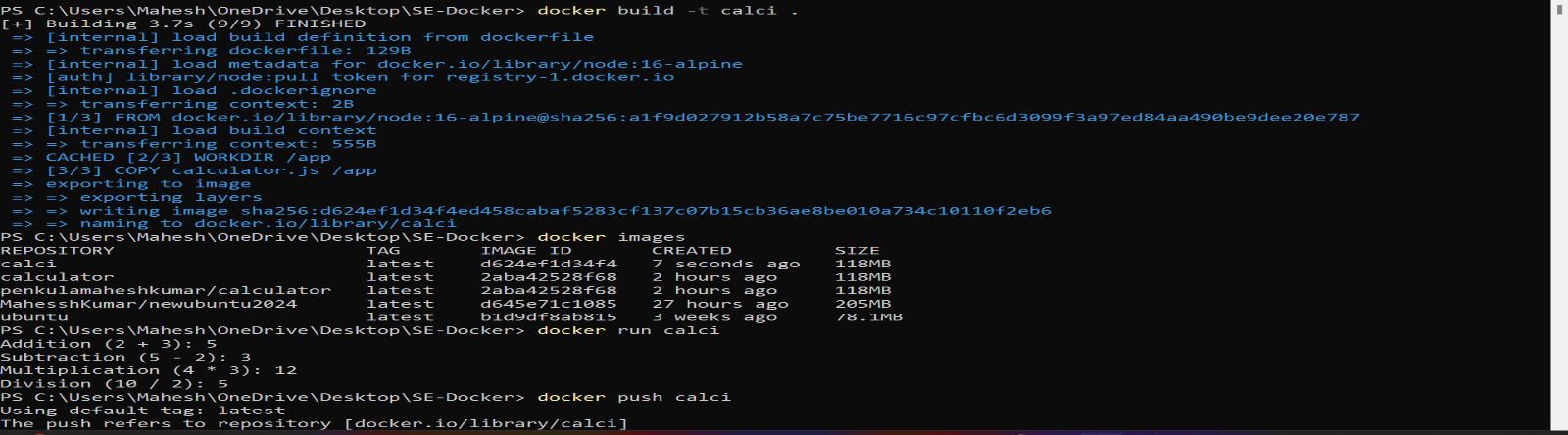
• -t simple-calculator: Tags the image with the name simple-calculator.

• .: Refers to the current directory where the Dockerfile is located.

**Step 4: Run the Docker Container**

Run the container using the image you just created:

**docker run simple-calculator**



• docker run: Starts a new container from the image.

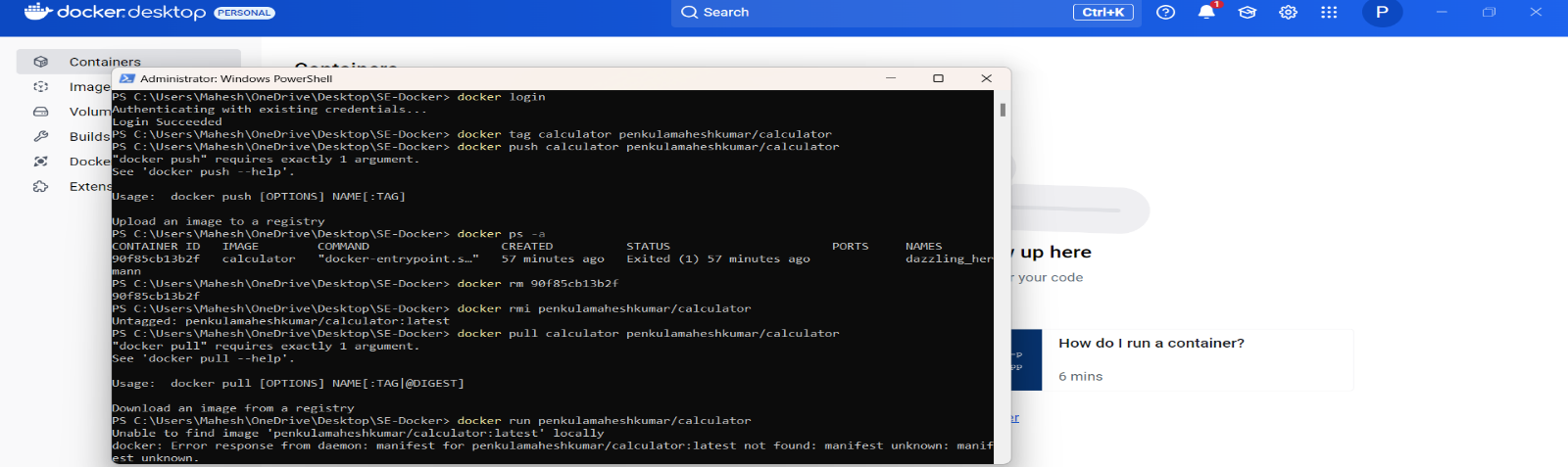
• simple-calculator: The name of the image to use.

Expected Output: The console will display the results of the calculations.

**Step 6: Push the Image to Docker Hub**

Ensure you have a Docker Hub account. Log in using the following command:

**docker login**

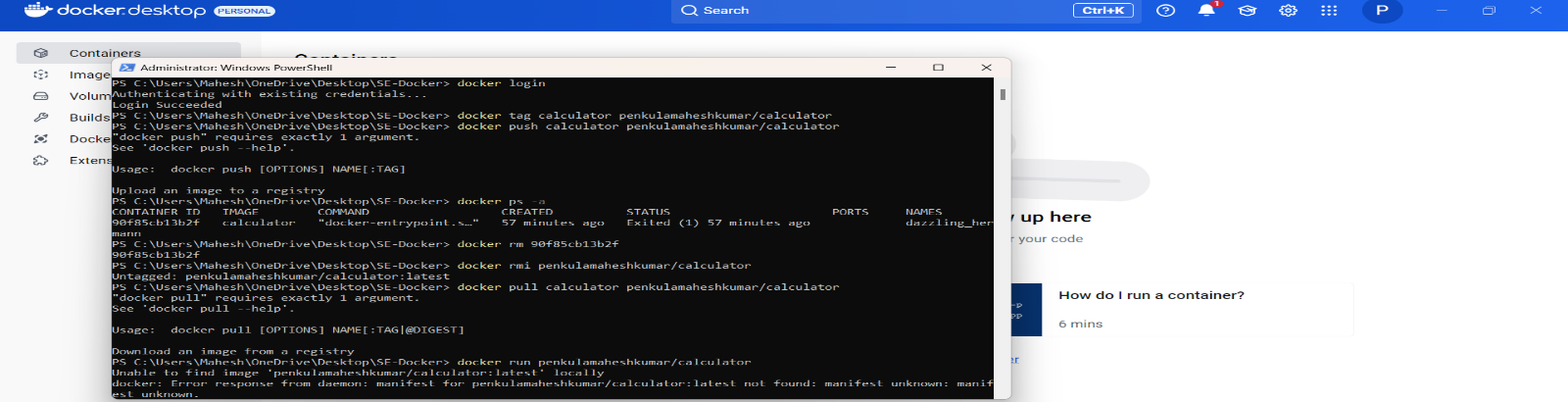
****

Enter your Docker Hub username and password if prompted

Tag the Image for Docker Hub:

Docker images need to be tagged with your Docker Hub username before they can be uploaded:

**docker tag simple-calculator your-dockerhub-username/simple-calculator**

****

Replace your-dockerhub-username with your actual Docker Hub username.

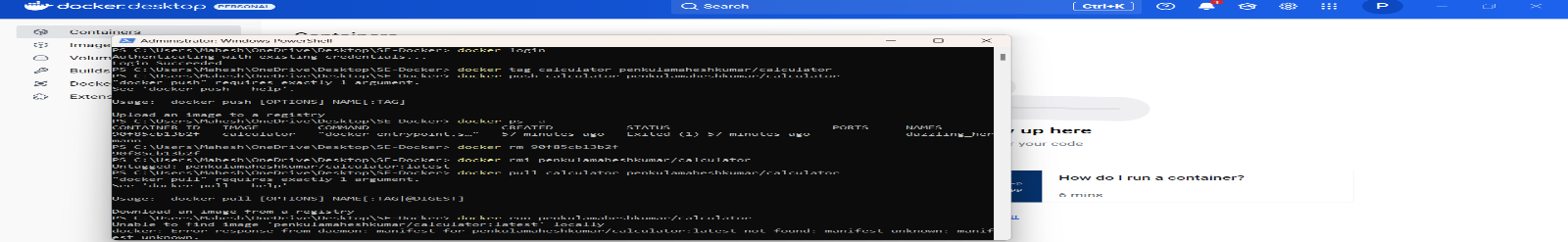
Push the Image to Docker Hub:

Push the tagged image to Docker Hub:

docker push your-dockerhub-username/simple-calculator

Once complete, your image will be available in your Docker Hub repository.

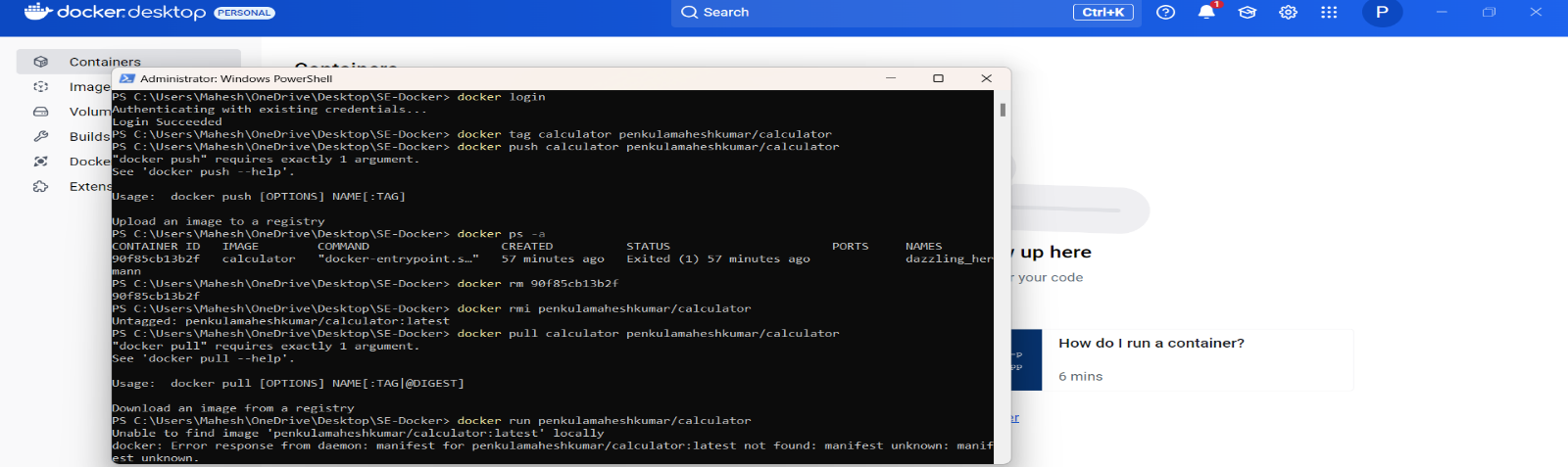
**Step 7: Pull the Image from Docker Hub**

****

To test the reusability, first, remove the existing image and container:

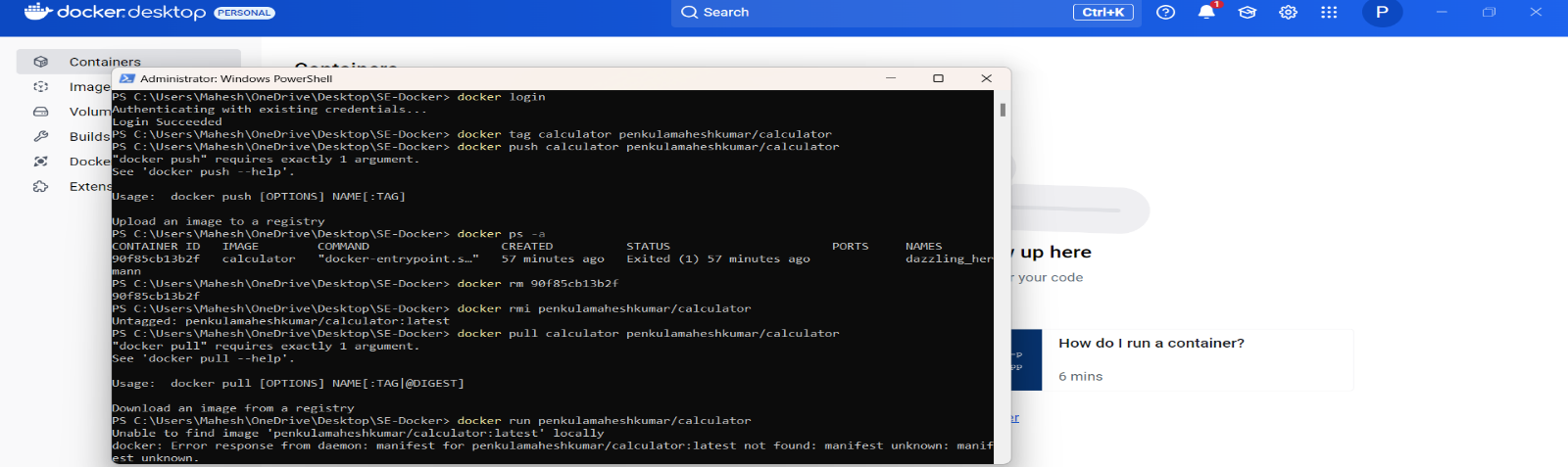
# List all containers (even stopped ones)

**docker ps -a**

****

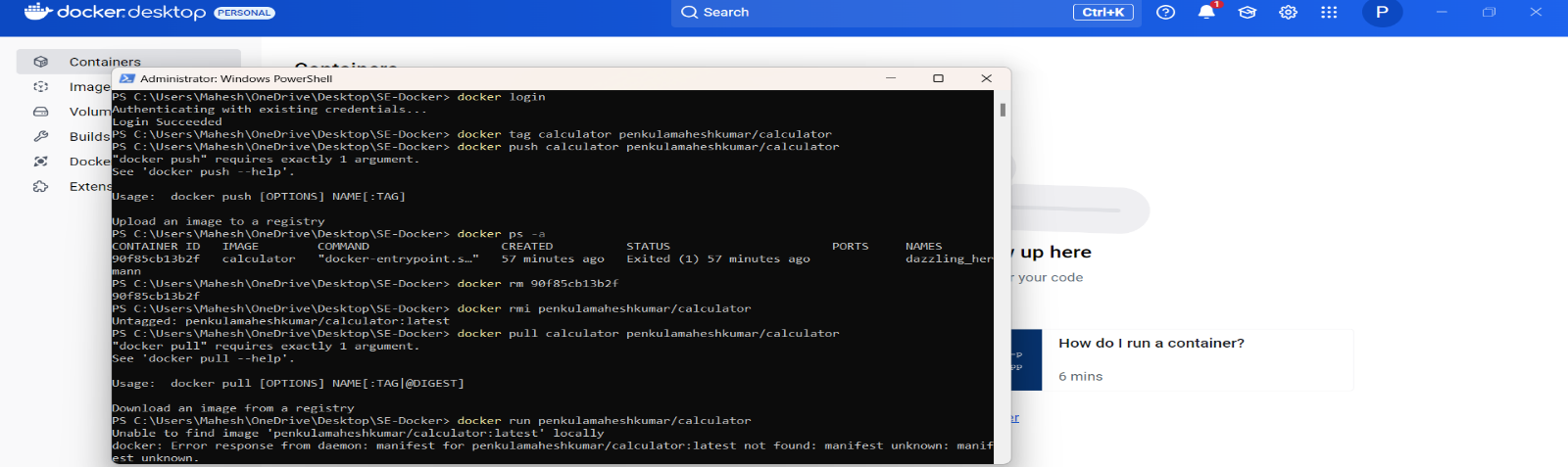
# Remove the container by ID

**docker rm <container-id>**

****

# Remove the local image

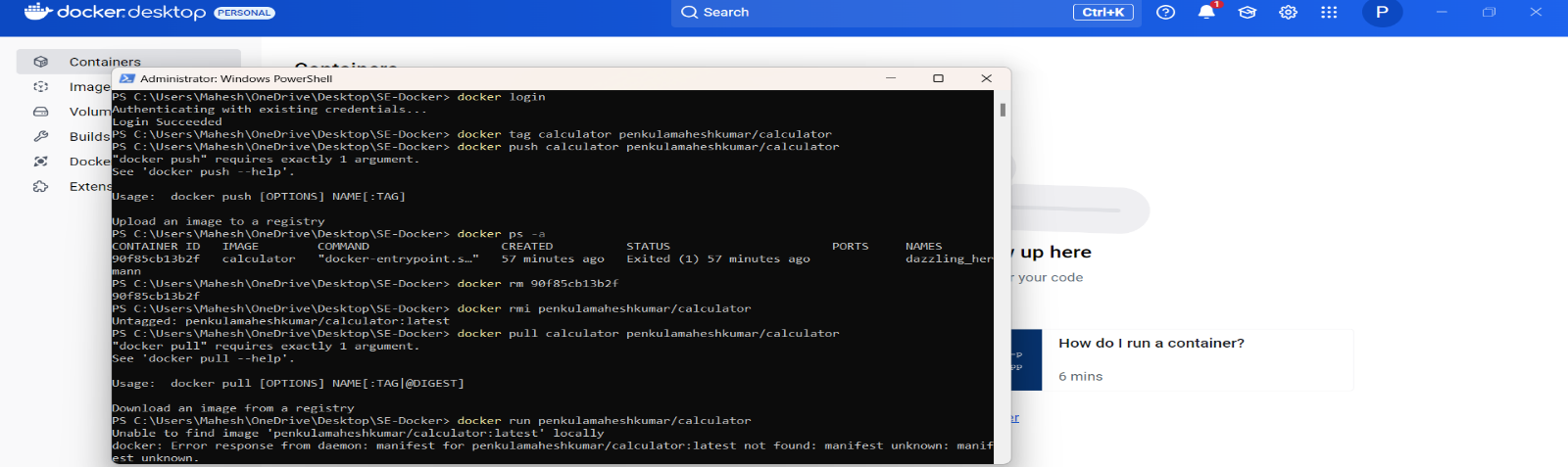
**docker rmi your-dockerhub-username/simple-calculator**

****

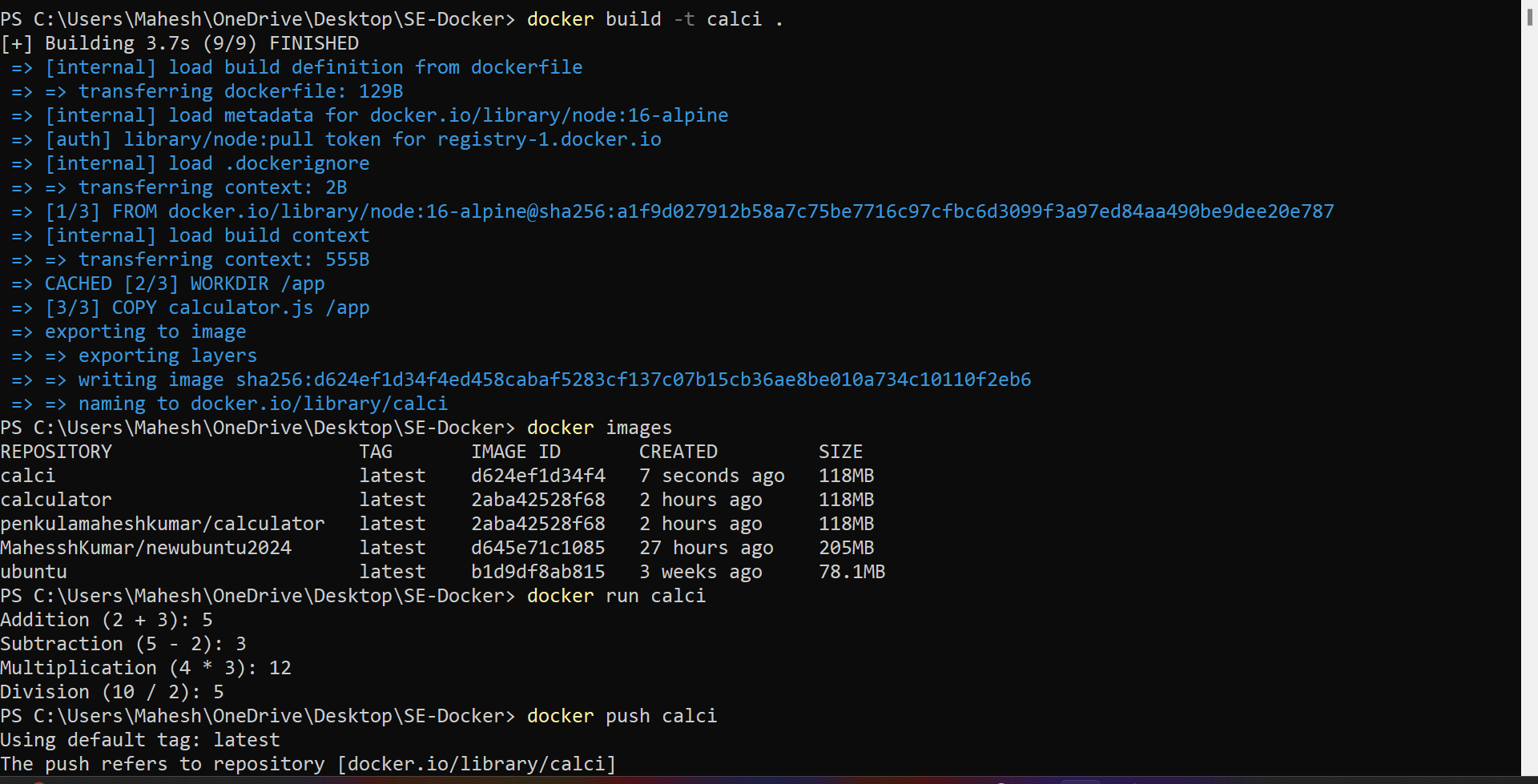
On another machine or after deleting the local image and container, pull the image from Docker

Hub:

**docker pull your-dockerhub-username/simple-calculator**

****

**Step 8: Run the Pulled Image**

****

Run the container from the pulled image:

docker run your-dockerhub-username/simple-calculator