Placement Empowerment Program

Cloud Computing and DevOps Centre

Automate Docker Container Management: Create a script that starts, stops, or removes specific Docker containers based on user input.

Name: Hema S

Department: ECE

Introduction

In modern DevOps and cloud environments, managing Docker containers efficiently is crucial. Manually starting, stopping, or removing containers can be time-consuming, especially when dealing with multiple instances. This Proof of Concept (PoC) aims to automate Docker container management using a Windows Batch script. The script allows users to start, stop, or remove specific containers with a simple command, making container administration more efficient and error-free.

Overview

This PoC involves creating a **batch script (docker_manager.bat)** that interacts with Docker CLI commands to manage container lifecycle operations. The script uses conditional statements (IF conditions) to determine user input and execute corresponding Docker commands. It provides an easy-to-use command-line interface for container operations.

Key functionalities include:

- ✅ **Starting** a Docker container.
- ✅ **Stopping** a running container.
- ✅ **Removing** an existing container.

Objectives

The primary goals of this PoC are:

1. **Automate** Docker container lifecycle management using a Windows Batch script.

2. **Simplify** repetitive container management tasks (start, stop, remove).

3. **Improve Efficiency** by reducing manual effort in handling containers.

4. **Enhance Usability** by providing a simple command-line interface.

5. **Ensure Flexibility** so users can extend or modify the script for their needs.

Importance

1. **Reduces Manual Work**: No need to manually type Docker commands every time.

2. **Speeds Up Operations**: One command automates container management.

3. **Prepares for Advanced Automation**: This PoC is a foundation for integrating container management into larger DevOps workflows.

4. **Boosts Productivity**: DevOps teams can focus on development rather than container administration.

5. **Scalability**: The script can be enhanced to support multiple containers or advanced Docker functionalities like logs, volume management, and networking.

Step-by-Step Overview

Step 1:

We need to create a Bash script that will manage Docker containers.

In Git Bash run:

touch docker_manager.sh

This will create an empty file named docker_manager.sh.

```
MINGW32:/c/Users/HP

HP@DESKTOP-S8GOFLP MINGW32 ~
$ touch docker_manager.sh
```

Step 2:

Next, open the file in a text editor.

nano docker_manager.sh

```
HP@DESKTOP-S8GOFLP MINGW32 ~
$ nano docker_manager.sh
```

Step 3:

Add the Script Code into the docker_manager.sh file and then

press **Ctrl + O**, then **Enter**, and **Ctrl + X** to save.

```
MINGW32:/c/Users/HP
  GNU nano 7.2                                docker_manager.sh
#!/bin/bash

# Function to display usage
usage() {
    echo "Usage: $0 {start|stop|remove} <container_name>"
    exit 1
}

# Check if correct number of arguments is provided
if [ $# -ne 2 ]; then
    usage
fi

# Assign input parameters to variables
action=$1
container_name=$2

# Perform action based on user input
case "$action" in
    start)
        echo "Starting container: $container_name"
        docker start "$container_name" || echo "Failed to start container."
        ;;
    stop)
        echo "Stopping container: $container_name"
        docker stop "$container_name" || echo "Failed to stop container."
        ;;
    remove)
        echo "Removing container: $container_name"
        docker rm "$container_name" || echo "Failed to remove container."
        ;;
    *)
        usage
        ;;
esac

^G Help      ^O Write Out   ^W Where Is    ^K Cut       ^T Execute   ^C Location   M-U Undo     M-A Set Mark   M-] To Bracket
^X Exit      ^R Read File   ^\ Replace     ^U Paste     ^J Justify   ^/ Go To Line M-E Redo     M-6 Copy       ^Q Where Was
```

Step 4:

Before we can run the script, we need to make it executable.

chmod +x docker_manager.sh

```
sppra@DESKTOP-S8GOFLP MINGW32 ~ (master)
$ chmod +x docker_manager.sh
```

Step 5:

Create a Test Container and view if it is running by:

docker run -d --name my_test_container nginx

docker ps -a

```
HP@DESKTOP-S8GOFLP MINGW32 ~
$ docker run -d --name my_test_container nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
103f50cb3e9f: Pulling fs layer
7cf63256a31a: Pulling fs layer
943ea0f0c2e4: Pulling fs layer
513c3649bb14: Pulling fs layer
bf9acace214a: Pulling fs layer
d014f92d532d: Pulling fs layer
9dd21ad5a4a6: Pulling fs layer
943ea0f0c2e4: Download complete
103f50cb3e9f: Download complete
513c3649bb14: Download complete
9dd21ad5a4a6: Download complete
d014f92d532d: Download complete
7cf63256a31a: Download complete
bf9acace214a: Download complete
Digest: sha256:9d6b58feebd2dbd3c56ab5853333d627cc6e281011cfd6050fa4bcf2072c9496
Status: Downloaded newer image for nginx:latest
93d2006e406a6377bc175a3aafba659a13581b8677a4199b562da86ced4f62f7
```

```
HP@DESKTOP-S8GOFLP MINGW32 ~
$ docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
93d2006e406a   nginx     "/docker-entrypoint.…"   22 seconds ago   Up 19 seconds   80/tcp    my_test_container
```

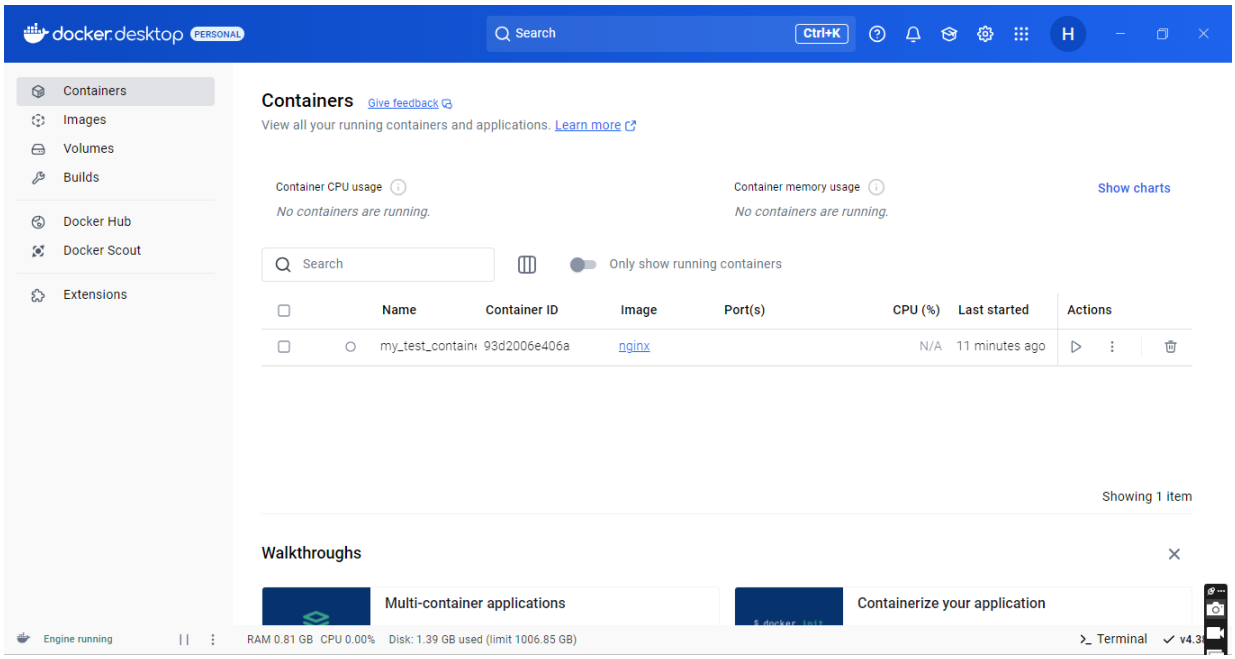
Step 6:

Test the Script with the Correct Command Format

To stop the container:

./docker_manager.sh stop my_test_container

```
HP@DESKTOP-S8GOFLP MINGW32 ~
$ ./docker_manager.sh stop my_test_container
Stopping container: my_test_container
my_test_container
```

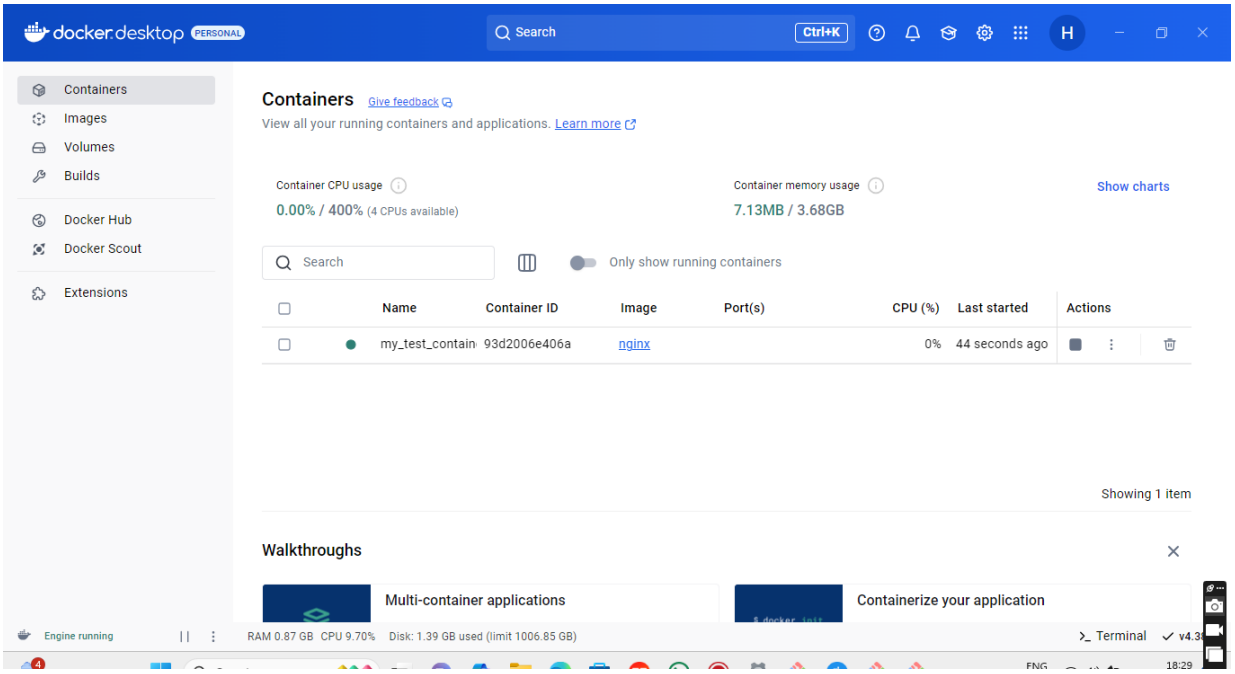


Step 7:

To start the container:

./docker_manager.sh start my_test_container

```
HP@DESKTOP-S8GOFLP MINGW32 ~
$ ./docker_manager.sh start my_test_container
Starting container: my_test_container
my_test_container
```



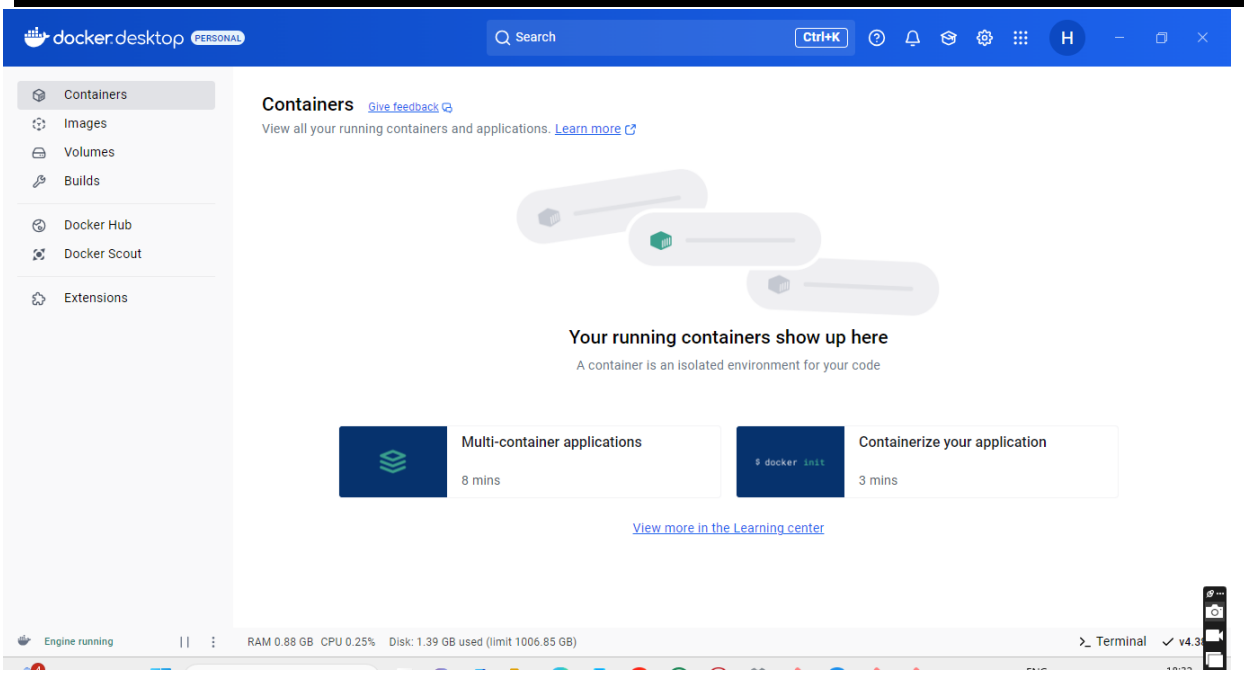
Step 8:

To remove the container (after stopping it):

./docker_manager.sh remove my_test_container

```
HP@DESKTOP-S8GOFLP MINGW32 ~
$ ./docker_manager.sh stop my_test_container
Stopping container: my_test_container
my_test_container

HP@DESKTOP-S8GOFLP MINGW32 ~
$ ./docker_manager.sh remove my_test_container
Removing container: my_test_container
my_test_container
```



Before wrapping up, ensure:

1. Your script correctly starts, stops, and removes the container.

2. Running docker ps -a reflects the expected status after each operation.

Outcomes

By completing this PoC, you will:

1. **Automate Docker Container Management** – Develop a script to start, stop, and remove containers with a single command, reducing manual effort.

2. **Enhance Shell Scripting Skills** – Gain hands-on experience in writing and executing Bash scripts for automating Docker workflows.

3. **Improve Docker Command Proficiency** – Master essential commands like docker start, docker stop, and docker rm for efficient container lifecycle management.

4. **Simplify Deployment Processes** – Learn how scripting can streamline container operations, making it easier to manage applications in a real-world environment.

5. **Understand the Importance of Infrastructure as Code (IaC)** – Explore how automation using scripts enhances efficiency, reduces errors, and supports scalable containerized environments.