Hands-On Lab (HOL)

Lab Name

HOL-02: Docker Container Deletion and Inspect Commands on Ubuntu 22.04

Author : Sandeep Kumar Sharma

Learning Objectives

By the end of this lab, participants will be able to:

Understand how Docker containers are deleted safely and forcefully. Differentiate between deleting stopped containers and running containers. Remove single containers as well as multiple containers at once. Understand what `docker inspect` is and how to analyze container and image metadata. Read and interpret important information such as container ID, IP address, mounts, network details, and image configuration.

Lab Outcome

After completing this lab, you will be confident in cleaning up Docker environments by removing unused containers and inspecting Docker resources for troubleshooting and validation. You will clearly understand when to use normal delete versus force delete and how Docker internally stores container and image metadata.

Pre-requisites

Docker must be installed and running on Ubuntu 22.04. Basic understanding of Docker images and containers (HOL-01). Root or sudo access.

Lab Explanation and Hands-On Steps

Section 1: Listing Existing Containers

Before deleting any container, it is very important to check the current state of containers on the system.

```
docker ps -a
```

This command lists all containers, including running, stopped, and exited containers. Only stopped containers can be removed using the normal `docker rm` command.

Section 2: Deleting Stopped Containers

```
docker rm c3
```

This command deletes a container named `c3`, but it works **only if the container is in a stopped or exited state**. Docker protects running containers from accidental deletion.

Verify the result:

```
docker ps -a
```

Attempt to delete another stopped container:

```
docker rm c2
```

If the container is stopped, it will be deleted successfully. If it is running, Docker will throw an error.

Section 3: Force Deleting Running Containers

```
docker rm -f c2
```

The `-f` (force) option stops the running container first and then deletes it. Internally, Docker sends a SIGKILL signal to the container process before removing it.

Check container list again:

```
docker ps -a
```

Section 4: Creating Containers for Practice

```
docker run -it --name c3 ubuntu
```

This command creates and runs a new container named `c3` using the Ubuntu image.

Exit the container:

```
exit
```

Create one more container:

```
docker run -it --name c4 ubuntu
```

Exit again and verify:

```
docker ps -a
```

Section 5: Deleting All Containers at Once

```
docker rm -f $(docker ps -aq)
```

This is a very powerful cleanup command:

- `docker ps -aq` returns the IDs of **all containers** (running and stopped).
- `docker rm -f` forcefully deletes all containers in one shot.

This command is commonly used in lab environments and CI/CD pipelines but should be used carefully in production systems.

Verify cleanup:

```
docker ps -a
```

Section 6: Viewing Command History

```
history
```

This displays all commands executed in the current shell session. Useful for revising steps or auditing lab activity.

Section 7: Inspecting Docker Containers

```
docker inspect container_name
```

Example:

```
docker inspect c3
```

This command displays detailed JSON-formatted metadata about the container, including:

- Container ID and name
- Image used to create the container
- Network settings and IP address
- Mount points and volumes
- Environment variables
- Container state (running, exited, paused)

`docker inspect` is widely used for debugging networking issues and verifying runtime configurations.

Section 8: Inspecting Docker Images

```
docker inspect image_name
```

Example:

```
docker inspect ubuntu
```

This command shows image-level information such as:

- Image layers
- Entrypoint and CMD instructions
- Environment variables
- Architecture and OS type
- Image creation time

This is helpful for understanding how an image was built and what defaults it carries.

Summary

In this lab, you learned how to delete Docker containers safely and forcefully, clean up all containers in one command, and inspect both containers and images. These skills are essential for Docker troubleshooting, environment cleanup, and real-world DevOps workflows. With this lab, you now have full control over Docker container lifecycle management.