Docker Hands-On Lab (HOL) — Creating and Managing Containers with Apache

Author: Dr. Sandeep Kumar Sharma

Lab Description

This Hands-On Lab (HOL) demonstrates how to work with Docker containers at a very practical level using **Ubuntu containers**. You will:

- Clean up existing containers
- Create a web container
- Install and configure Apache manually inside the container
- Access the application from the browser using port mapping
- Attach back to a running container
- Create an additional container (DB-style container)

This lab is designed to help you understand **container lifecycle, port mapping, service management, and container interaction** in a simple and clear way.

Topics Covered

- Removing all running containers
- Running interactive containers
- Port mapping (`-p` option)
- Installing Apache inside a container
- Managing services inside containers
- Container filesystem navigation
- Attaching to running containers
- Running multiple containers

Learning Objectives

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

- Create and manage Docker containers interactively
- Install software inside a running container
- Expose container services to the host system
- Reattach to running containers
- Run multiple containers simultaneously

Learning Outcomes

After completing this lab, you will:

- Understand how containers behave like lightweight VMs
- Be confident in installing and running services inside containers
- Clearly understand container vs host interaction

Section 1 — Clean Up Existing Containers

Before starting, remove all existing containers to avoid conflicts.

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

Verify:

```
docker ps -a
```

No containers should be running.

Section 2 — Create a Web Container

Run an Ubuntu container with port mapping:

```
docker run -it --name webcontainer -p 8080:80 ubuntu
```

Explanation:

- `-it` → interactive terminal
- `--name webcontainer` → container name
- `-p 8080:80` → map host port 8080 to container port 80
- `ubuntu` → base image

You are now inside the container.

Section 3 — Install Apache Inside the Container

Run the following commands **inside the container**:

```
apt update -y
apt install apache2 -y
```

Check Apache status:

```
service apache2 status
```

Start Apache service:

```
service apache2 start
```

Verify again:

```
service apache2 status
```

Section 4 — Explore Apache Web Directory

Navigate through the filesystem:

```
cd /var
ls
cd www
ls
cd html
ls
```

Create a custom web page:

```
echo "<h1>This is my first application</h1>" > index.html
```

Check command history:

```
history
```

Section 5 — Access the Web Application

Open a browser on your host machine and visit:

```
http://localhost:8080
```

You should see:

```
This is my first application
```

This confirms:

- Apache is running inside the container
- Port mapping is working correctly

Section 6 — Verify Running Containers

Exit the container:

```
exit
```

Check running containers:

```
docker ps
```

Section 7 — Attach Back to the Running Container

Reattach to the running web container:

```
docker attach webcontainer
```

You are now back inside the same container.

Section 8 — Create Another Container (DB Container)

Open a new terminal and run:

```
docker run -it --name dbcontainer1 -p 8081:80 ubuntu
```

This creates a second container.

Check history:

```
history
```

At this stage:

- `webcontainer` is running Apache on port 8080
- `dbcontainer1` is a fresh Ubuntu container

Section 9 — Verify Multiple Containers

From host:

```
docker ps
```

You should see:

- webcontainer → port 8080
- dbcontainer1 → port 8081

Section 10 — Cleanup (Optional)

Stop and remove containers:

```
docker stop webcontainer dbcontainer1
docker rm webcontainer dbcontainer1
```

Summary

In this Hands-On Lab, you:

- Created an interactive Ubuntu container
- Installed and configured Apache manually
- Exposed the application using port mapping
- Attached back to a running container
- Ran multiple containers simultaneously