Hands-On Lab (HOL)

Dockerfile Concept – Build a Custom Web Server Image Using Apache

Author: Dr. Sandeep Kumar Sharma

🎯Learning Objectives

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

- Understand what a Dockerfile is and why it is used
- Understand key Dockerfile instructions (FROM, RUN, EXPOSE, CMD)
- Create a Dockerfile using the `vi` editor
- Build a Docker image using `docker build`
- Run a container from a Dockerfile-based image
- Access a web application running inside a Docker container

Learning Outcomes

After completing this HOL, you will:

- Clearly understand Dockerfile-based image creation
- Know the difference between `docker commit` and Dockerfile approach
- Be able to automate image creation in a standard and reusable way
- Run and expose applications using Docker containers

Conceptual Understanding

What is a Dockerfile?

A **Dockerfile** is a text file that contains a set of instructions used by Docker to **automatically build an image**.

👉Think of a Dockerfile as a **recipe**:

- Docker Image = Final Dish 😌
- Dockerfile = Recipe 💜

Each instruction in a Dockerfile creates a **new image layer**, making the build process efficient and reusable.

Why Dockerfile is Preferred Over `docker commit`?

Dockerfile	docker commit
Version controlled	Manual & non-repeatable
Industry standard	Learning & debugging only
CI/CD friendly	Not CI/CD friendly
Declarative & documented	Imperative & ad-hoc

👉**In real production projects, Dockerfile is always preferred.**

🧨Lab Prerequisites

- Ubuntu 22.04
- Docker installed and running
- Basic Linux command knowledge
- Internet connectivity

Hands-On Lab Steps

Step 1: Create a Working Directory

```
mkdir dockerfile-lab
cd dockerfile-lab
```

Explanation:

- Creates a dedicated directory for Dockerfile and build context

---

Step 2: Create Dockerfile Using `vi` Editor

```
vi Dockerfile
```

Press `i` to enter **insert mode** and add the following content:

```
FROM ubuntu:18.04
MAINTAINER sandeep
```

```
RUN apt-get update -y
RUN apt-get install git -y
RUN apt-get install apache2 -y

RUN echo "<h3> THIS IS MY WENSERVER </h3>" > /var/www/html/index.html

RUN apt-get install apache2-utils -y

EXPOSE 80

CMD ["apache2ctl","-D","FOREGROUND"]
```

Save and exit `vi`:

```
ESC :wq
```

🧩 Dockerfile Instruction Explanation

- **FROM ubuntu:18.04** → Base operating system
- **MAINTAINER** → Image author information
- **RUN** → Executes commands during image build
- **EXPOSE 80** → Documents container listening port
- **CMD** → Starts Apache in foreground mode

Step 3: Build Docker Image

```
docker build -t webimage .
```

Explanation:

- `-t webimage` → Assigns name to the image
- `.` → Current directory as build context

Step 4: Verify Docker Image

```
docker images
```

Explanation:

- Confirms `webimage` is successfully created

Step 5: Run Container from the Image

```
docker run -d --name webcontainer -p 8080:80 webimage
```

Explanation:

- `-d` → Detached mode
- `--name webcontainer` → Container name
- `-p 8080:80` → Maps host port 8080 to container port 80

Step 6: Verify Running Container

```
docker ps
```

Explanation:

- Ensures Apache container is running

Step 7: Access Web Server

Open browser and hit:

```
http://<server-ip>:8080
```

You should see:

```
THIS IS MY WENSERVER
```

Summary

- Dockerfile enables automated and repeatable image builds
- Each instruction creates a layer
- Apache runs in foreground to keep container alive

- Port mapping exposes container services externally

🦻 **Congratulations! You have successfully built and deployed a web server using Dockerfile.**