Docker Hands-On Lab 3 — Building a Custom Container with Apache

Trainer: Sandeep Kumar Sharma

Duration: 45–60 minutes

Goal: In this lab, you will learn how to create a custom Docker container from a base image, install Apache HTTP server inside it, name the container as *Sandeep Application Container*, and expose it to the internet using port mapping.

---

1) Objective

By the end of this exercise, you will: - Create a container from a base image (Ubuntu) - Install Apache manually inside the container - Configure and test Apache web server - Use port mapping to access it from a browser

---

2) Step-by-Step Lab

Step 1: Pull the Base Image

We will use the official Ubuntu image from Docker Hub.

```
docker pull ubuntu:latest
```

Verify the image:

```
docker images
```

Step 2: Run a New Container in Interactive Mode

We'll create a new container from Ubuntu and name it **sandeep-app-container**.

```
docker run -it --name sandeep-app-container ubuntu:latest /bin/bash
```

You are now inside the container shell.

Step 3: Update the Container and Install Apache

Inside the container, run the following commands:

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

Start the Apache service:

```
service apache2 start
```

Check the Apache status:

```
service apache2 status
```

You should see it running inside the container.

Step 4: Verify Apache Inside the Container

Run this command to confirm Apache is listening on port 80:

```
netstat -tulpn | grep apache2
```

You should see something like:

```
tcp6       0      0 :::80                   :::*
LISTEN      /usr/sbin/apache2
```

Step 5: Exit the Container

```
exit
```

This will take you back to the host terminal.

Step 6: Commit the Container as a New Image

Now let's save our configured container as a reusable image.

```
docker commit sandeep-app-container sandeep/apache:v1
```

Check the new image:

```
docker images
```

You should see your custom image listed.

Step 7: Run a Container with Port Mapping

Now we'll run a new container from our custom image and map port **8080** on the host to port **80** inside the container.

```
docker run -d --name sandeep-webapp -p 8080:80 sandeep/apache:v1
```

Verify it's running:

```
docker ps
```

Expected output:

```
CONTAINER ID   IMAGE                COMMAND       PORTS                   NAMES
abcd1234       sandeep/apache:v1    "/bin/bash"   0.0.0.0:8080->80/tcp    sandeep-
webapp
```

Step 8: Access Apache via Browser

Now open your browser and go to:

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

If using a cloud VM, use your public IP:

```
http://<your-vm-ip>:8080
```

You should see the **Apache2 Default Page**.

Step 9: Verify Container Networking and Logs

```
docker port sandeep-webapp
docker logs sandeep-webapp
```

Both commands should confirm that Apache is active and accessible.

Step 10: Cleanup

```
docker stop sandeep-webapp
docker rm sandeep-webapp
```

(Optional) Remove the image if you want to free up space:

```
docker rmi sandeep/apache:v1
```

---

3) Practice Tasks

1. Create another container from **Debian** and install Apache.
2. Change the port mapping to **9090:80** and test.
3. Modify Apache's index page inside the container (edit `/var/www/html/index.html`) and refresh the browser.

---

4) Summary

- You learned to create a custom container from a base image.
- Installed Apache web server inside the container manually.
- Saved the container as a reusable image.
- Mapped ports to make the web server accessible from the internet.

✅ **Checkpoint:** You have successfully built, configured, and published your first web server container named **Sandeep Application Container**!