Hands-On Lab (HOL): Docker Volumes

Author: Dr. Sandeep Kumar Sharma

🎯Learning Objectives

By the end of this Hands-On Lab, learners will be able to:

- Understand what Docker Volumes are and why they are used
- Differentiate between Docker Volumes and Bind Mounts
- Create and manage Docker Volumes
- Mount a Docker Volume to one or more containers
- Verify data persistence even after container deletion

Learning Outcomes

After completing this lab, learners will:

- Confidently use Docker Volumes for persistent storage
- Understand how Docker internally manages volume data
- Be able to share data across multiple containers using a single volume
- Clearly explain why volumes are preferred over bind mounts in production

---

Concept Overview (Simple Explanation)

What is a Docker Volume?

A **Docker Volume** is a storage mechanism **fully managed by Docker**. It is **independent of the host OS directory structure** and lives inside Docker's internal storage area.

👉When you create a volume, Docker automatically creates a directory under:

```
/var/lib/docker/volumes/
```

Docker takes complete responsibility for managing this directory.

Why Docker Volumes?

- Data **persists even if the container is stopped or deleted**

- Easy to **backup and migrate**
- Safer and cleaner than using host directories directly
- Recommended for **production environments**

Docker Storage Options

Docker provides **two ways** to store container data on the host:

1. **Volumes (Recommended)**

2. Managed by Docker

3. Stored under `/var/lib/docker/volumes/`

4. Portable and secure

5. **Bind Mounts**

6. Direct mapping of host directory

7. Depends on host OS and directory structure
8. Mostly used for local development

🧨Hands-On Lab Steps

> 👉🏿Prerequisite: Docker must be installed and running. You should have root or sudo access.

🏐Step 1: List Existing Docker Volumes

```
docker volume ls
```

👉This shows all volumes currently managed by Docker.

🏐Step 2: Create a Docker Volume

```
docker volume create myvolume
```

👉Docker creates a new volume named **myvolume**.

🏐Step 3: Verify Volume Location on Host

```
cd /var/lib/docker/volumes/
ls
```

You will see:

```
myvolume
```

🏐Step 4: Explore Volume Data Directory

```
cd myvolume/
ls
```

Output:

```
_data
```

Now move inside:

```
cd _data/
ls
```

🏐Step 5: Create Files and Directories in the Volume

```
touch file1 file2 file3
mkdir test1 test2 test3
ls
```

Add content to a file:

```
cat > file2
Hello Docker Volume
CTRL+D
```

🏐Step 6: Run First Container with Volume Mounted

```
docker run -it --name datacontainer -v myvolume:/applicationdata ubuntu
```

Inside container:

```
cd /applicationdata
ls
```

👉You will see the same files created earlier.

Exit container:

```
exit
```

🏐Step 7: Run Second Container Using the Same Volume

```
docker run -it --name container2 -v myvolume:/rohandata ubuntu
```

Inside container:

```
cd /rohandata
ls
```

👉Same data is visible → **Volume is shared!**

Exit container:

```
exit
```

🏐Step 8: Verify Data Persistence After Container Deletion

Remove all containers:

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

Now check volume data again on host:

```
cd /var/lib/docker/volumes/myvolume/_data
ls
```

👉**Data is still present** 👂

Key Takeaways (Trainer Notes)

- Containers are **temporary**, volumes are **permanent**
- Volumes live **outside container lifecycle**
- Multiple containers can **share the same volume**
- Even if containers are deleted, **volume data remains safe**

🎤Lab Complete!

You have successfully completed the Docker Volumes Hands-On Lab and understood:

- Volume creation
- Volume mounting
- Data persistence
- Multi-container data sharing