Hands-On Lab (HOL): Docker Bind Mounts

Author: Dr. Sandeep Kumar Sharma

🎯Learning Objectives

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

- Understand what Docker Bind Mounts are
- Identify real-world use cases for Bind Mounts
- Create and use Bind Mounts with Docker containers
- Observe real-time file synchronization between host and container
- Compare Bind Mount behavior with Docker Volumes

Learning Outcomes

After completing this lab, learners will:

- Confidently use Bind Mounts for development scenarios
- Clearly understand how host directories are shared with containers
- Explain why Bind Mounts are tightly coupled with host OS paths
- Decide when to use Bind Mounts vs Volumes

Concept Overview (Very Simple Explanation)

What is a Docker Bind Mount?

A **Bind Mount** is a way to **directly link a host machine directory** to a directory inside a Docker container.

👉Docker does **NOT manage** the data location. 👉You decide **which host folder** is mounted. 👉Any change on the host is **immediately visible** inside the container (and vice versa).

Think of it like:

> "Docker container is borrowing a folder from my laptop/server."

Bind Mount vs Volume (Quick Reminder)

Feature	Bind Mount	Volume
Managed by Docker	👈No	👈🏽Yes
Host path required	👈🏽Yes	👈No
Best for	Development	Production
OS dependency	High	Low
Easy migration	👈No	👈🏽Yes

🧨Hands-On Lab Steps

👉🏿Prerequisite: Docker installed and running. Root or sudo access required.

🏐Step 1: Create a Directory on Host Machine

```
mkdir -p /opt/bindmount
cd /opt/bindmount
```

Create files and folders:

```
touch hostfile1 hostfile2
mkdir dir1 dir2
ls
```

Add content to a file:

```
cat > hostfile1
Hello Sandeep, This is bind mount host file
CTRL+D
```

🏐Step 2: Run Container with Bind Mount

```
docker run -it --name bindcontainer --mount type=bind,source="/opt/
bindmount",target="/testdata" ubuntu
```

🏐Step 3: Verify Data Inside Container

Inside container:

```
cd /testdata
ls
cat hostfile1
```

👉Files created on host are visible inside the container.

🏐Step 4: Create Files Inside Container

Inside container:

```
touch containerfile1
mkdir containerdir1
ls
```

Exit container:

```
exit
```

🏐Step 5: Verify Changes on Host

On host machine:

```
cd /opt/bindmount
ls
```

👉Files created inside container are now visible on host.

🏐Step 6: Delete Container and Check Data

```
docker rm -f bindcontainer
```

Check host directory:

```
ls /opt/bindmount-data
```

👉Data is still present because it belongs to the **host**, not Docker.

Key Observations (Trainer Notes)

- Bind Mounts directly expose host filesystem
- Docker does not isolate or protect the data
- If host path is deleted → data is lost
- Same container image behaves differently on different hosts

---

🎷Real-Time Use Case Example

- Live code editing
- Local development with source code
- Debugging applications
- Log file access

👉**Not recommended for production workloads**

🎤Lab Complete!

You have successfully completed the Docker Bind Mounts Hands-On Lab and learned:

- Host ↔ Container directory sharing
- Real-time file synchronization
- Practical development use cases
