Docker Data Storage: Quick Reference Guide

# 1. Why Data Management in Containers Matters

- Containers are ephemeral: Data inside is lost when they stop.  
- Use volumes or bind mounts for persistent storage.  
  
Analogy:  
- Ephemeral: Hotel room – leave, lose your stuff.  
- Persistent: Apartment – leave, your stuff stays.

# 2. Docker Storage Methods

|  |  |  |  |
| --- | --- | --- | --- |
| Storage Type | Persistent | Location | Best For |
| Container Filesystem | No | Inside container | Temporary data |
| Volumes | Yes | Managed by Docker (/var/lib/...) | Application data |
| Bind Mounts | Yes | Specific host directory | Sharing files between host & container |

# 3. Container Filesystem (Ephemeral)

docker run -it ubuntu bash  
# Inside container  
echo "Hello Docker!" > myfile.txt  
exit  
# Restart and check  
docker start -ai <container\_id>  
ls # myfile.txt is gone

# 4. Docker Volumes (Preferred Method)

## 4.1 Create and Use Volume

docker volume create mydata  
docker run -it -v mydata:/data ubuntu bash  
# Inside container  
echo "Persistent Data!" > /data/persistent.txt  
exit  
# Verify in new container  
docker run -it -v mydata:/data ubuntu bash  
cat /data/persistent.txt

## 4.2 Inspect and Manage Volumes

docker volume ls  
docker volume inspect mydata  
docker volume rm mydata # Deletes volume and data

# 5. Bind Mounts (Host ↔ Container Sync)

## 5.1 Use Bind Mount

mkdir ~/myhostdata  
echo "Hello from the host!" > ~/myhostdata/hostfile.txt  
  
docker run -it -v ~/myhostdata:/data ubuntu bash  
# Inside container  
cat /data/hostfile.txt  
  
echo "Hello from the container!" > /data/containerfile.txt  
exit  
  
# On host  
cat ~/myhostdata/containerfile.txt

## 5.2 Volumes vs. Bind Mounts

|  |  |  |
| --- | --- | --- |
| Feature | Docker Volumes | Bind Mounts |
| Managed by Docker | Yes | No |
| OS Compatibility | High | May vary (e.g., Windows) |
| Security | High | Direct host access |
| Performance | Better | Can be slower |
| Use Case | App data | Dev with live sync |

# 6. Sharing Data Between Containers

docker volume create shared-data  
  
docker run -it -v shared-data:/data ubuntu bash  
# Container 1  
echo "Hello from Container 1!" > /data/shared.txt  
exit  
  
docker run -it -v shared-data:/data ubuntu bash  
# Container 2  
cat /data/shared.txt

# 🔧 Summary

- Use Volumes: Persistent data, best for production.  
- Use Bind Mounts: Local dev, real-time sync.  
- Inspect Volumes: `docker volume inspect <volume\_name>`  
- Manage Data: Plan your storage for stability and performance.