# 📦 Docker Volumes & Bind Mounts - Day 5 Notes

Hands-on exploration of how data persistence works in Docker using volumes and bind mounts, and experimenting with installing dependencies inside containers.

## ✅ Objective:

Understand how Docker containers handle **data persistence**, and how to use **volumes** and **bind mounts** to store and retrieve data across container restarts. Also experimented with installing and persisting a tool (vim) inside a container.

## 🧪 Experiment 1: Installing ping in Ubuntu container

docker run --interactive --tty --rm ubuntu:22.04

Inside container:

bash

CopyEdit

ping google.com -c 1

# bash: ping: command not found

apt update

apt install iputils-ping --yes

ping google.com -c 1 # ✅ Now it works!

But when you **exit and re-run** a new container:

bash

CopyEdit

docker run -it --rm ubuntu:22.04

ping google.com -c 1 # ❌ Fails again. Installation was lost.

**Why?** Docker containers are ephemeral — any changes made inside are lost when the container is removed.

## 🧪 Solution: Reuse the same container

docker run -it --name my-ubuntu-container ubuntu:22.04

Inside:

bash

CopyEdit

apt update && apt install iputils-ping --yes

ping google.com -c 1

exit

To reuse:

docker start my-ubuntu-container

docker attach my-ubuntu-container

ping google.com -c 1 # ✅ Works

## ✅ Better Solution: Build custom image with pre-installed ping

docker build -t my-ubuntu-image -<<EOF

FROM ubuntu:22.04

RUN apt update && apt install iputils-ping --yes

EOF

Run container:

docker run -it --rm my-ubuntu-image

ping google.com -c 1 # ✅ Works!

## 📁 Experiment 2: Where does Docker store files?

### Try creating a file inside a container:

docker run -it --rm ubuntu:22.04

Inside:

mkdir my-data

echo "Hello from the container!" > my-data/hello.txt

cat my-data/hello.txt

exit

Run a new container:

docker run -it --rm ubuntu:22.04

cat my-data/hello.txt # ❌ File not found

**Why?** Data is not persistent across new containers.

## 📦 Volume Mounts

### Step 1: Create a volume

bash

CopyEdit

docker volume create my-volume

### Step 2: Run container with volume mounted

docker run -it --rm --mount source=my-volume,destination=/data ubuntu:22.04

Inside:

echo "Hello from volume!" > /data/hello.txt

cat /data/hello.txt

exit

### Step 3: Run a new container and access same data

docker run -it --rm --mount source=my-volume,destination=/data ubuntu:22.04

cat /data/hello.txt # ✅ File persists!

**Under the hood:** Docker stores volumes on Linux at /var/lib/docker/volumes/my-volume/\_data.

## 🔗 Bind Mounts

### Mount a host directory into container

Create a folder on your **host system**:

mkdir my-bind-data

Then run:

docker run -it --rm --mount type=bind,source="$(pwd)/my-bind-data",target=/data ubuntu:22.04

Inside container:

echo "Hello from bind mount!" > /data/hello.txt

exit

Check on **host**:

cat my-bind-data/hello.txt # ✅ File written from container!

Bind mount = full control + visibility from host.

## 🛠️ Installing Vim in a container and persisting it

You tried this:

docker run -it --name vim-container ubuntu:22.04

apt update && apt install vim --yes

vim

exit

If you rerun it with --rm, Vim is gone. But if you restart the container:

docker start vim-container

docker attach vim-container

vim # ✅ Works!

## 🔧 Git inside container + Volume

You used a **volume-mount** to persist data like a .txt file, then initialized a Git repo:

Inside container (with mounted volume):

cd /mount

git init

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

echo "Test file" > aakrit.txt

git add .

git commit -m "newfileadded"

Git commit now persisted in volume!

## 🧱 Vim inside image (optional way without Dockerfile)

Start container, install Vim:

docker run -it --name vim-persist ubuntu:22.04

apt update && apt install vim --yes

exit

Then **commit** the container as an image:

docker commit vim-persist ubuntu-with-vim

docker run -it --rm ubuntu-with-vim

vim # ✅ Pre-installed!

## 🧠 Key Docker Syntax Cheatsheet📦 Docker Volumes & Bind Mounts - Day 5 Notes

Hands-on exploration of how data persistence works in Docker using volumes and bind mounts, and experimenting with installing dependencies inside containers.

✅ Objective:

Understand how Docker containers handle data persistence, and how to use volumes and bind mounts to store and retrieve data across container restarts. Also experimented with installing and persisting a tool (vim) inside a container.

🧪 Experiment 1: Installing ping in Ubuntu container

docker run --interactive --tty --rm ubuntu:22.04

Inside container:

ping google.com -c 1

# bash: ping: command not found

apt update

apt install iputils-ping --yes

ping google.com -c 1 # ✅ Now it works!

But when you exit and re-run a new container:

docker run -it --rm ubuntu:22.04

ping google.com -c 1 # ❌ Fails again. Installation was lost.

Why? Docker containers are ephemeral — any changes made inside are lost when the container is removed.

🧪 Solution: Reuse the same container

docker run -it --name my-ubuntu-container ubuntu:22.04

Inside:

apt update && apt install iputils-ping --yes

ping google.com -c 1

exit

To reuse:

docker start my-ubuntu-container

docker attach my-ubuntu-container

ping google.com -c 1 # ✅ Works

✅ Better Solution: Build custom image with pre-installed ping

docker build -t my-ubuntu-image -<<EOF

FROM ubuntu:22.04

RUN apt update && apt install iputils-ping --yes

EOF

Run container:

docker run -it --rm my-ubuntu-image

ping google.com -c 1 # ✅ Works!

📁 Experiment 2: Where does Docker store files?

Try creating a file inside a container:

docker run -it --rm ubuntu:22.04

Inside:

mkdir my-data

echo "Hello from the container!" > my-data/hello.txt

cat my-data/hello.txt

exit

Run a new container:

docker run -it --rm ubuntu:22.04

cat my-data/hello.txt # ❌ File not found

Why? Data is not persistent across new containers.

📦 Volume Mounts

Step 1: Create a volume

docker volume create my-volume

Step 2: Run container with volume mounted

docker run -it --rm --mount source=my-volume,destination=/data ubuntu:22.04

Inside:

echo "Hello from volume!" > /data/hello.txt

cat /data/hello.txt

exit

Step 3: Run a new container and access same data

bash

docker run -it --rm --mount source=my-volume,destination=/data ubuntu:22.04

cat /data/hello.txt # ✅ File persists!

Under the hood: Docker stores volumes on Linux at /var/lib/docker/volumes/my-volume/\_data.

🔗 Bind Mounts

Mount a host directory into container

Create a folder on your host system:

mkdir my-bind-data

Then run:

docker run -it --rm --mount type=bind,source="$(pwd)/my-bind-data",target=/data ubuntu:22.04

Inside container:

echo "Hello from bind mount!" > /data/hello.txt

exit

Check on host:

cat my-bind-data/hello.txt # ✅ File written from container!

Bind mount = full control + visibility from host.

🛠️ Installing Vim in a container and persisting it

You tried this:

docker run -it --name vim-container ubuntu:22.04

apt update && apt install vim --yes

vim

exit

If you rerun it with --rm, Vim is gone. But if you restart the container:

docker start vim-container

docker attach vim-container

vim # ✅ Works!

🔧 Git inside container + Volume

You used a volume-mount to persist data like a .txt file, then initialized a Git repo:

Inside container (with mounted volume):

cd /mount

git init

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

echo "Test file" > aakrit.txt

git add .

git commit -m "newfileadded"

Git commit now persisted in volume!

🧱 Vim inside image (optional way without Dockerfile)

Start container, install Vim:

docker run -it --name vim-persist ubuntu:22.04

apt update && apt install vim --yes

exit

Then commit the container as an image:

docker commit vim-persist ubuntu-with-vim

docker run -it --rm ubuntu-with-vim

vim # ✅ Pre-installed!

🧠 Key Docker Syntax Cheatsheet

Command Purpose

docker volume create my-vol Creates a named volume

docker run -v my-vol:/data ... Mount volume to container

docker run --mount source=vol,target=/data ... Alternative syntax

docker run -v $(pwd)/dir:/data ... Bind mount host dir

docker build -t img . Build custom image

docker commit container\_id new\_image Save container state as image

🧘‍♂️ Summary of What I Learned

How Docker containers are ephemeral by default.

Volume mount = persistent data managed by Docker.

Bind mount = persistent data shared with host.

Installed Vim and committed it into an image.

Used Git inside a volume to track changes.

| Command | Purpose |
| --- | --- |
| docker volume create my-vol | Creates a named volume |
| docker run -v my-vol:/data ... | Mount volume to container |
| docker run --mount source=vol,target=/data ... | Alternative syntax |
| docker run -v $(pwd)/dir:/data ... | Bind mount host dir |
| docker build -t img . | Build custom image |
| docker commit container\_id new\_image | Save container state as image |

## 🧘‍♂️ Summary of What I Learned

* How Docker containers are ephemeral by default.
* Volume mount = persistent data managed by Docker.
* Bind mount = persistent data shared with host.
* Installed Vim and committed it into an image.
* Used Git inside a volume to track changes.