**Docker Volume**

**🔍 1. What Is a Volume?**

* A volume is a persistent storage mechanism managed by Docker.
* It allows containers to store data outside the container’s lifecycle, meaning:
  + Deleting a container does NOT delete the volume or the data.
  + Volumes are decoupled from containers, so we can recreate containers without losing data.
* Volumes are decoupled from containers, so we can recreate containers without losing data.

**📦 2. Volume Types in Docker**

| **Type** | **Description** |
| --- | --- |
| Named Volume | Managed by Docker in a default path (/var/lib/docker/volumes/) |
| Bind Mount | Directly mounts a specific path from the host filesystem |
| Anonymous Volume | Created automatically by Docker with no name (not easily reusable) |

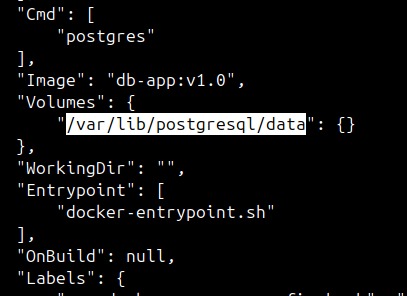
**📘 3. Examples of Each Volume Type**

✅ **Example 1: Named Volume (Recommended for Databases)**

A screenshot of a computer program

AI-generated content may be incorrect.

➡ Docker will auto-create a volume named project\_pgdata (project = folder name) unless we define it with an **external name**.

Note: /var/lib/postgresql/data 🡪 PostgreSQL stores all its database files.  


**Q. How Docker handles it?**

* When we run: docker-compose up -d
* Docker does two things automatically:
  + **Creates** the named volume pgdata (if it doesn’t exist already).
  + **Mounts** it to the path inside the container where PostgreSQL stores data.

✅ **Example 2: External Named Volume (Pre-created by we)**

* Create the volume:
  + docker volume create my\_custom\_volume
* Use it in docker-compose.yml:  
  A screenshot of a computer program

  AI-generated content may be incorrect.  
    
  ➡ Useful for **sharing volume** between projects.  
    
  ➡ The key part is **external: true**, which tells Docker **not to auto-create the volume**, but instead use one we created manually.
* Note: If we delete the external named volume, and again try to run the container, will get an error – ‘external volume <name of volume> not found’.
  + Fix: create the volume manually  
    docker volume create <name of volume>

✅ **Example 3: Bind Mount (Host Path)**

* This gives us full control of **where the data is stored** on our host system. A screenshot of a computer

  AI-generated content may be incorrect.  
    
  ➡ Data is saved to the hostdata folder (relative to our docker-compose.yml file) on our host machine.  
    
  ➡ The folder will be created on the host if it doesn't exist.

➡ **bind mounts** are **not listed** in the output of docker volume ls.  
  
➡docker volume ls only shows **named and external volumes** that are created and managed by Docker (using docker volume create, or -v volume\_name:/container/path).

➡A **bind mount** is just a reference to a **host directory** or file, directly mapped into the container.

✅ **Example 4: Anonymous Volume**

A screen shot of a computer

AI-generated content may be incorrect.

➡ Docker will auto-create a volume **with a random name**, but we can’t easily reference it later.

**💬 4. Use Cases**

| **Use Case** | **Best Volume Type** |
| --- | --- |
| Production Database | Named volume |
| Development with quick edits | Bind mount (host folder) |
| Data backup/restore | External named volume |
| Temporary data | Anonymous volume |
|  |  |
| **⚖️ 5. Pros & Cons**   | **Feature** | | **Named Volume** | **Bind Mount** | | --- | --- | --- | --- | | Docker-managed | ✅ Yes | | ❌ No | | Works across OS | ✅ Yes | | ⚠️ Host path must exist/accessible | | Portability | ✅ Easy to reuse in Compose | | ❌ Host path may break portability |   **🧪 6. Commands**   * docker volume ls * docker volume inspect <volume\_name> * docker volume rm my\_custom\_volume |  |

| **Method** | **Host Location** | **Docker Control** |
| --- | --- | --- |
| Named volume | /var/lib/docker/volumes/.../\_data | Yes |
| External volume | Same as above, but manually created by us | Yes (manual) |
| Bind mount | Anywhere on host (./my/custom/path) | No |