In Kubernetes, **by default**, when a **Pod dies**, its container filesystem is **wiped out** ❌.  
You lose:

* Logs
* Data
* Config files
* Any app-generated files inside the container

**Volume** gives you **persistent**, **shareable**, and **safe** storage **outside the container lifecycle**.  
Even if the Pod dies or restarts, your data is **safe** in a volume.

**Real-World Use Cases**

| **Use Case** | **Example** |
| --- | --- |
| **Database Storage** | MySQL, PostgreSQL, MongoDB store data in volumes so it survives |
| **File Uploads** | Web apps (Django, Node.js) saving user-uploaded files |
| **Shared Data Between Containers** | Two containers in one Pod sharing processing files |
| **Logging** | Fluentd/Splunk sidecar collecting app logs written into shared volume |
| **Configuration Management** | Mount configuration files into Pods (example: nginx.conf) |
| **Main Usage of Volumes**   | **Usage** | **Description** | | --- | --- | | **Persistence** | Store application data that should survive pod restarts/crashes | | **Sharing Data** | Share files between multiple containers inside the same Pod | | **Configuration Injection** | Inject config files (using ConfigMap volumes or Secret volumes) | | **Backup & Recovery** | Backup important data stored in Volumes easily | | **Migration & Upgrade** | Migrate app state/data separately without container disruption | | **Logs Storage** | Store container logs in volumes for centralized log collection | |  |

**Different Types of Volumes in Kubernetes**

1. **EmptyDir**
2. **Hostpath**
3. **PV**
4. **PVC**
5. **StorageClass(Dynamic)**
6. **ConfigMaps/Secrets**