**🗃️ What is a Persistent Volume (PV)?**

A **Persistent Volume** is a **pre-provisioned piece of storage** in your Kubernetes cluster. It can come from various sources—like a physical disk on a node, a network storage system (like NFS, iSCSI), or a cloud provider’s block storage (like AWS EBS, GCP Persistent Disk, Azure Disk). The key point is:

**A PV lives independently of any pod, which means data stored on it survives even if the pod is deleted or crashes.**

**✅ Key Features of PVs:**

* **Cluster-wide resource**: Defined by admins, not tied to a specific pod.
* **Pre-provisioned or dynamically provisioned**.
* **Supports multiple types**: hostPath, nfs, gcePersistentDisk, awsElasticBlockStore, etc.
* Can have access modes like:
  + ReadWriteOnce – One node can read/write.
  + ReadOnlyMany – Multiple nodes can read.
  + ReadWriteMany – Multiple nodes can read/write.

**📝 What is a Persistent Volume Claim (PVC)?**

A **Persistent Volume Claim** is like a **request** from a developer or an application for storage. When an app needs disk space (say 1Gi), it creates a PVC. Kubernetes then looks for a suitable PV that satisfies the request and **binds** the PVC to that PV.

Think of PV as a storage room and PVC as someone requesting a room with certain size and features. Kubernetes matches them and gives access.

**✅ PVC Can Specify:**

* Storage size (e.g., 5Gi)
* Access mode (e.g., ReadWriteOnce)
* Storage class (for dynamic provisioning)

**🔄 Relationship Between PV and PVC**

* **PV is the supply**, **PVC is the demand**.
* When PVC is created, Kubernetes **automatically finds and binds** a matching PV.
* Once bound, the **relationship is exclusive**: that PV is now dedicated to the claiming PVC until released.

**⚙️ How It Works (Step-by-Step):**

1. **Admin or dynamic provisioning system** sets up PVs in the cluster.
2. **User creates a PVC** asking for storage.
3. Kubernetes finds a suitable PV (based on size, access mode, storage class).
4. **Binding happens automatically**, and the PV is marked as "Bound".
5. PVC is used in a pod via volumeMounts and volumes sections.
6. Pod reads/writes data to this persistent storage.

**📦 Why Use PVs & PVCs?**

* **Data persistence**: Data survives beyond the pod lifecycle.
* **Decoupling**: Storage configuration is managed independently from applications.
* **Dynamic provisioning**: Automatically creates volumes if none are available.
* **Multi-environment support**: Works in cloud and on-prem setups.

**🧩 Example YAML Flow:**

**1. PVC (User creates):**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: myclaim

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

**2. Pod using the PVC:**

apiVersion: v1

kind: Pod

metadata:

name: mypod

spec:

containers:

- name: app

image: busybox

command: [ "sleep", "3600" ]

volumeMounts:

- mountPath: "/data"

name: mypd

volumes:

- name: mypd

persistentVolumeClaim:

claimName: myclaim

**🧠 Summary**

| **Concept** | **Description** |
| --- | --- |
| **PV** | Actual storage unit in the cluster (supply) |
| **PVC** | User's request for storage (demand) |
| **Binding** | Automatic process matching PVC to PV |
| **Persistence** | Data survives pod restarts/deletions |
| **Use Case** | Databases, logs, uploaded files, stateful apps |