What Kubernetes resources help you handle persistent storage?

🔷 Persistent Volume (PV)

- **Definition**: A piece of storage in the cluster provisioned by an administrator or dynamically provisioned using StorageClasses.
- **Lifecycle**: Independent of pods; exists beyond the pod's lifetime.
- **Types**: Supports NFS, iSCSI, AWS EBS, GCE PD, Azure Disk, CephFS, GlusterFS, HostPath (for dev), etc.
- **Attributes**:
  - `capacity`: Size of the storage.
  - `accessModes`: Defines how the volume can be mounted (e.g., `ReadWriteOnce`, `ReadOnlyMany`, `ReadWriteMany`).
  - `reclaimPolicy`: What happens after PVC is deleted (`Retain`, `Recycle`, `Delete`).
  - `storageClassName`: Defines the StorageClass for dynamic provisioning.
- **Binding**: PVs get bound to PVCs via matching specs (size, access modes, etc.).
- **Static vs Dynamic**:
  - **Static**: Manually created by admin.
  - **Dynamic**: Created on-the-fly using StorageClasses when PVCs are created.

🔷 Persistent Volume Claim (PVC)

- **Definition**: A request for storage by a user.
- **Usage**: Pods use PVCs to access storage without knowing the details of the underlying PV.
- **Attributes**:
  - `resources.requests.storage`: The requested size.
  - `accessModes`: Required access type.

    - `storageClassName`: Requests a specific class of storage.
- **Binding**: Kubernetes matches the PVC to an available PV that satisfies the request.
- **Lifecycle**: PVCs are bound to PVs and can be deleted, releasing the volume based on `reclaimPolicy`.

🔷 Volume Binding & Reclaim Policy

- **Binding**: Automatic or manual binding between PVC and PV.
- **Reclaim Policies**:
    - `Retain`: Keeps data for manual recovery.
    - `Recycle`: Wipes the data (deprecated).
    - `Delete`: Deletes the associated storage asset.

🔷 Access Modes

- `ReadWriteOnce (RWO)`: Mounted as read-write by a single node.
- `ReadOnlyMany (ROX)`: Mounted as read-only by many nodes.
- `ReadWriteMany (RWX)`: Mounted as read-write by many nodes.

🔷 StorageClass (for Dynamic Provisioning)

- Defines provisioner, parameters, reclaim policy, mount options.
- PVCs with `storageClassName` automatically get dynamic PVs.

Example using Kind

1. Create a Kind cluster

If you haven't already, install Kind:

```
bash
--------
curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.22.0/kind-linux-amd64
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind
```

Create a cluster:

```
bash
--------
kind create cluster --name pv-demo
```

---

2. Create a local directory for storage

We'll simulate persistent storage using a local folder:

```
bash
--------
mkdir -p /mnt/kind-pv
```

---

3. Create a PersistentVolume (PV) YAML file

`pv.yaml`

```
yaml
--------
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/kind-pv"
  persistentVolumeReclaimPolicy: Retain
```

Apply it:

```
bash
--------
kubectl apply -f pv.yaml
```

`persistentVolumeReclaimPolicy: Retain` means that when a **PersistentVolumeClaim (PVC)** `is deleted, the` **PersistentVolume (PV)** `is` **not** `automatically deleted or reused — instead, the data and the volume remain intact.`

Reclaim Policy	What Happens When PVC is Deleted?
`Retain`	**Keeps** the underlying storage. Manual cleanup needed.
`Delete`	**Deletes** the storage backend (like EBS, GCE disk, etc.)
`Recycle` *(deprecated)*	Clears data and makes PV available again.

---

4. Create a PersistentVolumeClaim (PVC) YAML file

`pvc.yaml`

```
yaml
--------
```

apiVersion: v1

kind: PersistentVolumeClaim
metadata:
  name: local-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
  volumeName: local-pv   # force bind to your PV
  storageClassName: ""   *# Empty string must be explicitly set otherwise default StorageClass will be set*

Apply it:

```
bash
--------
kubectl apply -f pvc.yaml
```

Check if it's bound:

```
bash
--------
kubectl get pvc
```

Output should show `STATUS: Bound`.

📦 5. Use the PVC in a Pod

`pod.yaml`

```
yaml
--------
apiVersion: v1
kind: Pod
metadata:
  name: test-pv-pod
spec:
  containers:
    - name: test-container
      image: busybox
      command: ["sleep", "3600"]
      volumeMounts:
        - mountPath: "/data"
          name: test-storage
  volumes:
    - name: test-storage
      persistentVolumeClaim:
        claimName: local-pvc
```

Apply it:

```
bash
--------
kubectl apply -f pod.yaml
```

✅ 6. Test it!

Enter the pod:

```
bash
--------
kubectl exec -it test-pv-pod -- sh
```

Inside the pod:

```
sh
--------
echo "Hello PV!" > /data/hello.txt
cat /data/hello.txt
```

Exit the pod and check the file exists on host:

```
bash
--------
cat /mnt/kind-pv/hello.txt
```

You should see: `Hello PV!`

🧹 Cleanup

```
bash
--------
kind delete cluster --name pv-demo
rm -rf /mnt/kind-pv
```

---

1. What is a Persistent Volume (PV) in Kubernetes?

A. A temporary storage attached to a pod
B. A storage resource provisioned manually or dynamically in a cluster
C. A configuration file for volumes
D. A container runtime feature

✅ **Correct Answer**: B

---

2. What does a Persistent Volume Claim (PVC) do?

A. Deletes a volume from the cluster
B. Requests storage resources like size and access mode
C. Monitors pod performance
D. Installs a storage plugin

✅ **Correct Answer**: B

---

3. Which of the following is NOT a valid `accessMode` for PV/PVC?

A. ReadWriteOnce (RWO)
B. ReadOnlyMany (ROX)
C. ReadWriteMany (RWX)
D. ReadWriteAll (RWA)

✅ **Correct Answer**: D

---

4. What happens when a PVC is deleted and the PV has `reclaimPolicy: Delete`?

A. The PV is retained
B. The storage asset is deleted
C. The data is backed up
D. Nothing happens

✅ **Correct Answer**: B

---

5. Which component is responsible for dynamic provisioning of Persistent Volumes?

A. ConfigMap
B. Pod Controller
C. StorageClass
D. VolumeMount

✅ **Correct Answer**: C

---

6. Can a Persistent Volume exist without being bound to a PVC?

A. No
B. Yes
C. Only in local clusters
D. Only in stateless pods

✅ **Correct Answer**: B

---

7. What is required in a PVC for it to be successfully bound to a PV?

A. Same pod name
B. Matching access modes and size requirements
C. Same container runtime
D. Mount path match

✅ **Correct Answer**: B