**Kubernetes Persistent Storage (PV & PVC)**

Persistent storage in Kubernetes ensures that data remains available even if a pod is deleted, restarted, or moved. Kubernetes provides **PersistentVolumes (PV)** and **PersistentVolumeClaims (PVC)** to manage storage independently of pods.

**Types of Persistent Storage in Kubernetes**

1. **Persistent Volume (PV) and Persistent Volume Claim (PVC)** – For dynamically or statically provisioned storage.
2. **HostPath** – Uses a directory on the worker node.
3. **EmptyDir** – A temporary volume tied to a pod’s lifecycle.
4. **ConfigMap/Secret-backed Volumes** – Injects configuration or secrets.
5. **NFS (Network File System)** – Uses a remote NFS server.
6. **Cloud Storage** – Such as AWS EBS, Azure Disk, or Google Persistent Disk.

**1. Persistent Volume (PV) and Persistent Volume Claim (PVC)**

A **PersistentVolume (PV)** is a storage resource provisioned by an administrator, while a **PersistentVolumeClaim (PVC)** is a request for storage by a pod.

**Example: Using PV & PVC**

**Step 1: Create a PersistentVolume (PV)**

apiVersion: v1

kind: PersistentVolume

metadata:

name: my-pv

spec:

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Retain

storageClassName: manual

hostPath:

path: "/mnt/data"

**Explanation:**

* Defines a 1Gi **PersistentVolume**.
* Uses a hostPath (for demonstration; in production, use cloud-based storage).
* ReadWriteOnce: Only one node can write.
* Retain: The data will persist after PVC release.

Apply the PV:

kubectl apply -f pv.yaml

**Step 2: Create a PersistentVolumeClaim (PVC)**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

storageClassName: manual

**Explanation:**

* Requests 1Gi of storage from a **PersistentVolume**.
* Must match PV’s storageClassName.

Apply the PVC:

kubectl apply -f pvc.yaml

**Step 3: Create a Pod that Uses the PVC**

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx

image: nginx

volumeMounts:

- mountPath: "/usr/share/nginx/html"

name: storage-volume

volumes:

- name: storage-volume

persistentVolumeClaim:

claimName: my-pvc

Apply the pod:

kubectl apply -f pod.yaml

**Step 4: Verify the Setup**

Check PV and PVC status:

kubectl get pv

kubectl get pvc

Check pod logs:

kubectl logs nginx-pod

**2. HostPath Storage (For Single-Node Development)**

The **hostPath** volume mounts a directory from the worker node into a pod.

**Example: Using hostPath**

apiVersion: v1

kind: Pod

metadata:

name: hostpath-pod

spec:

containers:

- name: busybox

image: busybox

command: ["sleep", "3600"]

volumeMounts:

- mountPath: "/data"

name: host-storage

volumes:

- name: host-storage

hostPath:

path: "/mnt/hostpath"

Apply the pod:

kubectl apply -f hostpath.yaml

**Explanation:**

* Mounts the /mnt/hostpath directory from the node to the pod’s /data directory.

Check if the volume is available:

kubectl exec -it hostpath-pod -- /bin/sh

Create a file in the Data folder

Delete the pod

Create the pod again

Exec into the pod

Check if the file exist in the Data folder

**3. EmptyDir Storage (Temporary Ephemeral Storage)**

The **emptyDir** volume provides temporary storage for a pod, which is deleted when the pod is removed.

**Example: Using emptyDir**

apiVersion: v1

kind: Pod

metadata:

name: emptydir-pod

spec:

containers:

- name: busybox

image: busybox

command: ["sleep", "3600"]

volumeMounts:

- mountPath: "/cache"

name: temp-storage

volumes:

- name: temp-storage

emptyDir: {}

Apply the pod:

kubectl apply -f emptydir.yaml

**Explanation:**

* The /cache directory is backed by emptyDir, meaning its contents exist only as long as the pod runs.

Test storage:

kubectl exec -it emptydir-pod -- sh

touch /cache/test.txt

ls /cache

Delete the pod:

kubectl delete pod emptydir-pod

The /cache directory is lost when the pod is removed.

**4. ConfigMap-backed Storage**

Kubernetes ConfigMaps allow injecting configuration files into a pod as a volume.

**Example: Using ConfigMap as a Volume**

Create a ConfigMap:

kubectl create configmap my-config --from-literal=config.txt="Hello, Kubernetes!"

Define a pod:

apiVersion: v1

kind: Pod

metadata:

name: configmap-pod

spec:

containers:

- name: busybox

image: busybox

command: ["sleep", "3600"]

volumeMounts:

- mountPath: "/config"

name: config-volume

volumes:

- name: config-volume

configMap:

name: my-config

Apply the pod:

kubectl apply -f configmap-pod.yaml

Check the file:

kubectl exec -it configmap-pod -- cat /config/config.txt

**5. NFS-backed Persistent Storage**

A **Network File System (NFS)** volume allows multiple pods to share data.

**Example: Using NFS**

**Step 1: Create a PersistentVolume**

apiVersion: v1

kind: PersistentVolume

metadata:

name: nfs-pv

spec:

capacity:

storage: 2Gi

accessModes:

- ReadWriteMany

persistentVolumeReclaimPolicy: Retain

nfs:

server: 192.168.1.100 # Change to your NFS server IP

path: "/exported/path"

Apply the PV:

kubectl apply -f nfs-pv.yaml

**Step 2: Create a PVC**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: nfs-pvc

spec:

accessModes:

- ReadWriteMany

resources:

requests:

storage: 2Gi

Apply the PVC:

kubectl apply -f nfs-pvc.yaml

**Step 3: Use the NFS Volume in a Pod**

apiVersion: v1

kind: Pod

metadata:

name: nfs-pod

spec:

containers:

- name: busybox

image: busybox

command: ["sleep", "3600"]

volumeMounts:

- mountPath: "/nfs"

name: nfs-storage

volumes:

- name: nfs-storage

persistentVolumeClaim:

claimName: nfs-pvc

Apply the pod:

kubectl apply -f nfs-pod.yaml

Check if storage works:

kubectl exec -it nfs-pod -- touch /nfs/testfile.txt

kubectl exec -it nfs-pod -- ls /nfs

**Summary of Storage Types**

| **Storage Type** | **Persistent?** | **Multi-node access?** | **Use Case** |
| --- | --- | --- | --- |
| PersistentVolume (PV) & PVC | ✅ | ✅ (RWX mode) | Stateful applications |
| HostPath | ✅ | ❌ | Development/testing |
| EmptyDir | ❌ | ❌ | Temporary storage |
| ConfigMap | ❌ | ✅ | Configuration management |
| NFS | ✅ | ✅ | Shared storage |