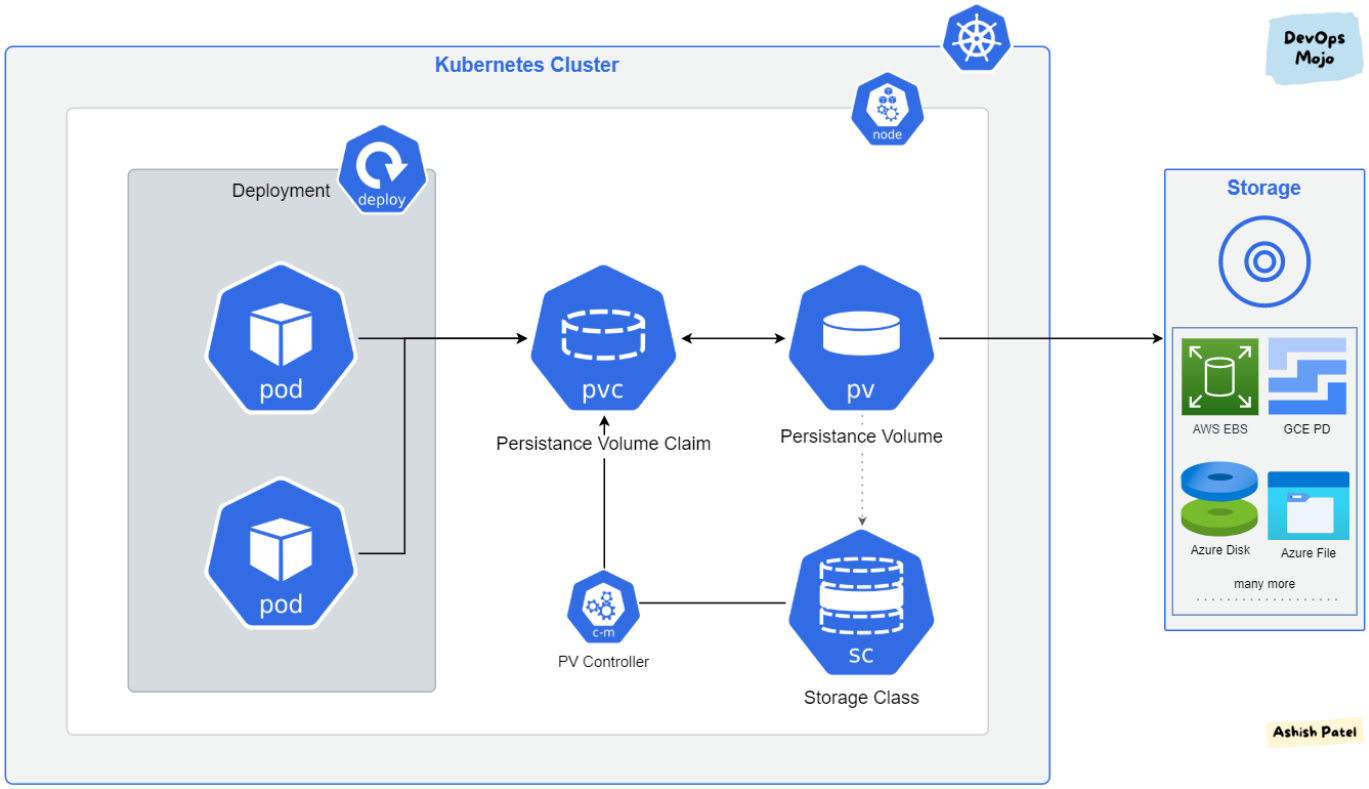
Storage Overview — PV, PVC and Storage Class

Kubernetes Storage Options — Persistent Volumes (PV), Persistent Volume Claims (PVC), Storage Classes (SC).



Kubernetes Storage — PV, PVC and Storage Class

TL;DR

Kubernetes has a number of storage types, and these can be mixed and matched within a pod. The important objects for running stateful containers are:

- Persistent Volume — low level representation of a storage volume.
- Persistent Volume Claim — binding between a Pod and Persistent Volume.
- Storage Class — allows for dynamic provisioning of Persistent Volumes.

Persistent Volumes (PV)

- PV is an abstraction for the physical storage device that attached to the cluster. PV is used to manage durable storage which needs actual physical storage.
- PV is independent of the lifecycle of the Pods. It means that data represented by a PV continue to exist as the cluster changes and as Pods are deleted and recreated.
- PV is not Namespaced, it is available to whole cluster. i.e. PV is accessible to all namespaces.
- PV is resource in the cluster that can be provisioned dynamically using Storage Classes, or they can be explicitly created by a cluster administrator.
- Unlike Volumes, the PVs lifecycle is managed by Kubernetes.

Types of Persistent Volumes

- AWS Elastic Block Store (EBS)
- Azure Disk
- Azure File
- GCE Persistent Disk
- Network File System (NFS) storage
- vSphere VMDK volume
- Many more…

Access Modes

- ReadWriteOnce(RWO) — volume can be mounted as read-write by a single node.
- ReadOnlyMany(ROX) — volume can be mounted read-only by many nodes.
- ReadWriteMany(RWX) — volume can be mounted as read-write by many nodes.
- ReadWriteOncePod(RWOP) — volume can be mounted as read-write by a single Pod.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: myapp-mongo-pv
spec:
  capacity:
    storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  storageClassName: myapp-sc
  # persistentVolumeReclaimPolicy: Retain  # StorageClass has a reclaim
policy default so it'll be "inherited" by the PV
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /tmp
    server: 172.17.0.2
```

Persistent Volume Claims (PVC)

- PVC is binding between a Pod and PV. Pod request the Volume through the PVC.
- PVC is the request to provision persistent storage with a specific type and configuration.
- PVC is similar to a Pod. Pods consume node resources and PVC consume PV resources.
- Kubernetes looks for a PV that meets the criteria defined in the PVC, and if there is one, it matches claim to PV.
- PVCs describe the storage capacity and characteristics a pod requires, and the cluster attempts to match the request and provision the desired persistent volume.
- PVC must be in same namespace as the Pod. For each Pod, a PVC makes a storage consumption request within a namespace.
- Claims can request specific size and access modes (e.g., they can be mounted ReadWriteOnce, ReadOnlyMany or ReadWriteMany).

> *PVC to PV binding is a one-to-one mapping. The process uses a ClaimRef, which creates a bi-directional binding between PV and PVC.*

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myapp-mongo-pvc
  namespace: myapp-ns
spec:
  storageClassName: myapp-sc
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

Storage Classes (SC)

- StorageClass allows dynamic provisioning of Persistent Volumes, when PVC claims it.
- StorageClass abstracts underlying storage provider.
- StorageClass is used in conjunction with PVC that allow Pods to dynamically request a new storage.
- StorageClass use provisioners that are specific to the storage platform or cloud provider to give Kubernetes access to the physical storage.
- Each storage backend has own provisioner. Storage Backend is defined in the StorageClass component via `provisioner` attribute.

Reclaim Policy

- PV that are dynamically created by a StorageClass will have the reclaim policy specified in the `reclaimPolicy` field of the class, which can be either `Delete` or `Retain`.
- If no `reclaimPolicy` is specified when a StorageClass object is created, it will default to `Delete`.
- PV that are created manually and managed via a StorageClass will have whatever reclaim policy they were assigned at creation.
- The reclaim policy applies to the persistent volumes not to the storage class itself. PVs and PVCs that are created using that StorageClass will inherit the reclaim policy set in StorageClass.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: myapp-sc
  namespace: myapp-ns
provisioner: kubernetes.io/aws-ebs
reclaimPolicy: Retain
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

PVC Usage in Deployment or StatefulSet

Deploying a stateful Pod on Kubernetes will always involve a PV. They represent the backend storage entity that a Pod might consume.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: myapp-mongodb-sts
  namespace: myapp-ns
  labels:
    app: myapp-mongodb
spec:
  serviceName: mongodb
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: myapp-mongodb
        image: mongo
        imagePullPolicy: IfNotPresent
        ports:
        - name: mongodb
          containerPort: 27017
        volumeMounts:
        - name: myapp-mongo-volume
          mountPath: /var/lib/mongodb
        # Mongo Secret code is ommited due to brevity.
      volumes:
      - name: myapp-mongo-volume
        persistentVolumeClaim:
          claimName: myapp-mongo-pvc
```

Summary

Persistent Volume (PV) is an abstraction for the physical storage device that you have attached to the cluster. Pods can use this storage space using Persistent Volume Claims (PVC). The easiest way to create the PV-PVC pair for Pod is to use a StorageClass (SC) object, and then using the StorageClass to create PV-PVC pair dynamically whenever you need to use it.