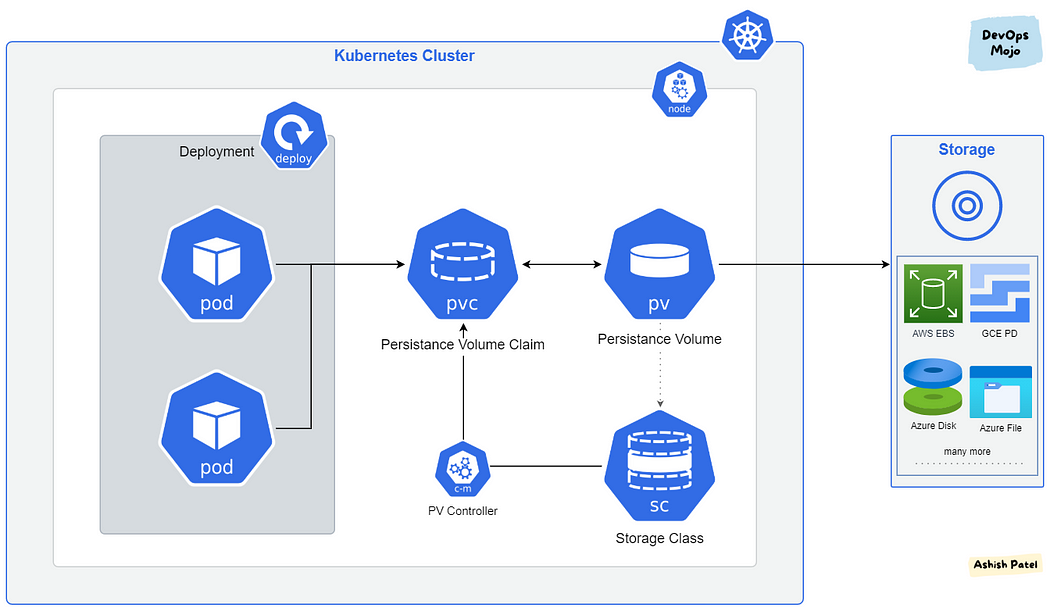
Persistent Volume (PV) is an abstraction for the physical storage device that you have attached to the cluster. Pods can use this storage space using Persistent Volume Claims (PVC). The easiest way to create the PV-PVC pair for Pod is to use a StorageClass (SC) object, and then use the StorageClass to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



Kubernetes has several 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 is 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 inside Namespace, it is available to the whole cluster. i.e., PV is accessible to all namespaces.

PV is a 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

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.

Before creating a Persistent Volume, you must create the PD.

gcloud beta compute disks create --size=200GB my-data-disk-demo --region us-central1 --replica-zones us-central1-a,us-central1-b

persistentvolume.yaml:

====================

---

apiVersion: v1

kind: PersistentVolume

metadata:

name: test-volume

spec:

accessModes:

- ReadWriteOnce

capacity:

storage: 20Gi

gcePersistentDisk:

fsType: ext4

pdName: my-data-disk-demo

storageClassName: gcp-disk

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 reclaim Policy field of the class, which can be either Delete or Retain.

If no reclaim Policy 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.

storage\_class.yaml:

-------------------

---

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: standard

provisioner: kubernetes.io/gce-pd

reclaimPolicy: Delete

volumeBindingMode: Immediate

parameters:

type: pd-standard

Persistent Volume Claims (PVC):

===============================

PVC is binding between a Pod and PV. Pod requests the Volume through the PVC.

PVC is the request to provision persistent storage with a specific type and configuration.

PVC is like a Pod. Pods consume node resources and PVC consumes PV resources.

Kubernetes looks for a PV that meets the criteria defined in the PVC, and if there is one, it matches the 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 the 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).

persistentvolumeclaim.yaml

=====================

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: test-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 200Gi

storageClassName: gcp-disk

PVC Usage in Deployment:

========================

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

---

apiVersion: v1

kind: Pod

metadata:

labels:

name: webserver

name: nginx-webserver

spec:

containers:

-

image: nginx

name: webserver

ports:

-

containerPort: 80

name: http

volumeMounts:

-

mountPath: /usr/local/nginx/html

name: app-data

volumes:

-

name: app-data

persistentVolumeClaim:

claimName: test-pvc