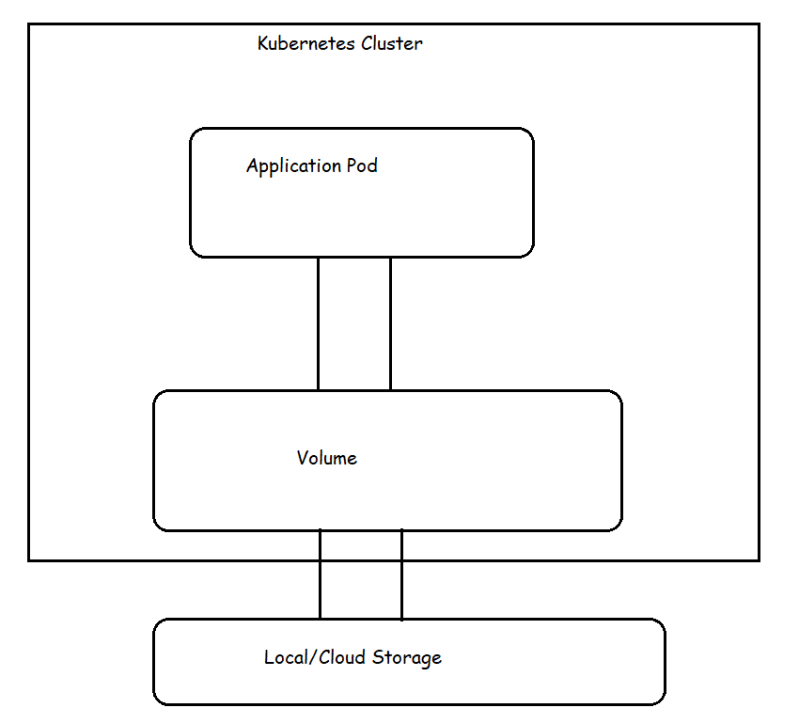
**Persistent Volume & Persistent Volume claim**

* If the containers that store the data crashes and is restarted the data will be lost. The new container will start with empty disk space allocated



* Even if the containers in the pod restart the same volume will be used by new container as well. Data will not be lost across container restarts, but when the pod is terminated or restart the volume ceases to exist (volume will be deleted)
* If you want to solve this problem of volumes getting deleted when pods get terminated then we need to use persistent volumes
* Types of volumes
* **emptyDir** − It is a type of volume that is created when a Pod is first assigned to a Node. It remains active as long as the Pod is running on that node. The volume is initially empty and the containers in the pod can read and write the files in the emptyDir volume. Once the Pod is removed from the node, the data in the emptyDir is erased.
* **hostPath** − This type of volume mounts a file or directory from the host node’s filesystem into your pod.
* **gcePersistentDisk** − This type of volume mounts a Google Compute Engine (GCE) Persistent Disk into your Pod. The data in a gcePersistentDisk remains intact when the Pod is removed from the node.
* **awsElasticBlockStore** − This type of volume mounts an Amazon Web Services (AWS) Elastic Block Store into your Pod. Just like gcePersistentDisk, the data in an awsElasticBlockStore remains intact when the Pod is removed from the node.
* **nfs** − An nfs volume allows an existing NFS (Network File System) to be mounted into your pod. The data in an nfs volume is not erased when the Pod is removed from the node. The volume is only unmounted.
* **azureDiskVolume** − An **AzureDiskVolume** is used to mount a Microsoft Azure Data Disk into a Pod.

EmptyDir volume

emptyDir is a temporary directory and when the pod is destroyed, it will destroy the shared volume and all its contents.

Let’s create a pod with two containers that use an **emptyDir**volume to exchange data:

we will be using **redis**image , name of the volume mount will be **redis-storage** and mount location will be **/data/redis and**type of volume that we are using is**emptyDir.**

PV:

1. Create a Persistent Volume named pv, with access mode ReadWriteMany, storage classname shared, 512MB of storage capacity and the host path /data/config.

kubectl apply -f pv.yaml

kubectl get pv

* Our persistent volume status is available meaning it is available and it has not been mounted yet. This status will change when we mount the persistentVolume to a persistentVolumeClaim.

**PersistentVolumeClaim**

In a real ecosystem, a system admin will create the PersistentVolume then a developer will create a PersistentVolumeClaim which will be referenced in a pod. A PersistentVolumeClaim is created by specifying the minimum size and the access mode they require from the persistentVolume.

1. Create a Persistent Volume Claim that requests the Persistent Volume we had created above. The claim should request 256MB. Ensure that the Persistent Volume Claim has the same storageClassName as the persistentVolume you had previously created.

kubectl apply -f pvc.yaml

kubectl get pvc

Our status has now changed from **available** to **bound**.

Create a new pod named myapp with image nginx that will be used to Mount the Persistent Volume Claim with the path /var/app/config.

Mounting a Claim

kubectl apply -f app.yaml