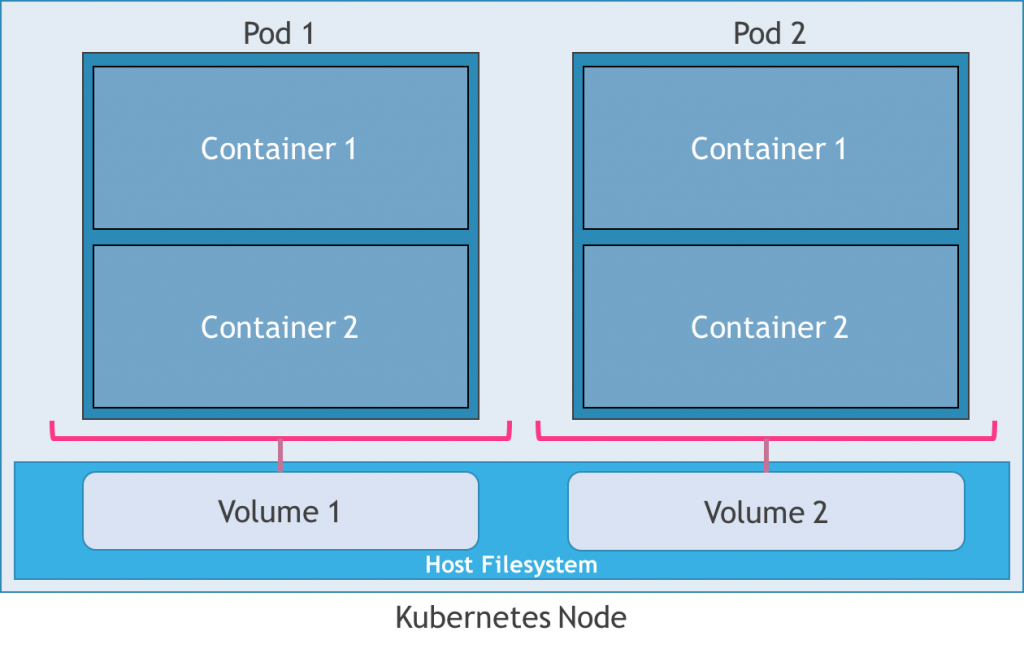
****

**Kubernetes volume**

Kubernetes volume is essentially a directory accessible to all containers running in a pod.

In contrast to the container-local filesystem, the data in volumes is preserved across container restarts. The medium backing a volume and its contents are determined by the volume type:

* node-local types such as emptyDir or hostPath
* file-sharing types such as nfs
* cloud provider-specific types like awsElasticBlockStore, azureDisk, or gcePersistentDisk
* distributed file system types, for example glusterfs or cephfs
* special-purpose types like secret, gitRepo



kubernetes Volume------Temp (Current)

Kubernetes Persistant Volume (10GB)-----hostpath, cloud,storage systems

Kubernetes persistant Volume Claim (5 Gb)--- PersistentVolumeClaim (PVC) is a request for storage by a user

Persistent Volumes

Kubernetes persistent volumes remain available outside of the pod lifecycle – this means that the volume will remain even after the pod is deleted. It is available to claim by another pod if required, and the data is retained.

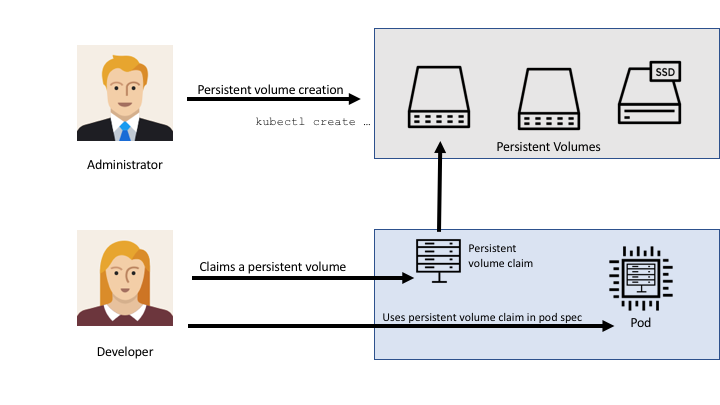
So how does the usage of Kubernetes volumes differ from Kubernetes persistent volumes? The answer is quite simple : **Kubernetes persistent volumes are used in situations where the data needs to be retained regardless of the pod lifecycle. Kubernetes volumes are used for storing temporary data.**

Kubernetes persistent volumes are administrator provisioned volumes. These are created with a particular filesystem, size, and identifying characteristics such as volume IDs and names.

A Kubernetes persistent volume has the following attributes

* It is provisioned either dynamically or by an administrator
* Created with a particular filesystem
* Has a particular size
* Has identifying characteristics such as volume IDs and a name

In order for pods to start using these volumes, they need to be claimed (via a persistent volume claim) and the claim referenced in the spec for a pod. A Persistent Volume Claim describes the amount and characteristics of the storage required by the pod, finds any matching persistent volumes and claims these. Storage Classes describe default volume information (filesystem,size,block size etc). The below image describes these processes:



Kubernetes Volumes vs Persistent Volumes

There are currently two types of storage abstracts available with Kubernetes: Volumes and Persistent Volumes.

Kubernetes Volumes

A Kubernetes volume exists only while the containing pod exists. Once the pod is deleted, the associated volume is also deleted. As a result, Kubernetes volumes are useful for storing temporary data that does not need to exist outside of the pod’s lifecycle.

Persistent Volumes

Kubernetes persistent volumes remain available outside of the pod lifecycle – this means that the volume will remain even after the pod is deleted. It is available to claim by another pod if required, and the data is retained.

So how does the usage of Kubernetes volumes differ from Kubernetes persistent volumes? The answer is quite simple : **Kubernetes persistent volumes are used in situations where the data needs to be retained regardless of the pod lifecycle. Kubernetes volumes are used for storing temporary data.**

### Persistent Volumes: A use case

The most common use case for Persistent volumes in Kubernetes is for databases. Obviously a database needs to have access to its data at all times, and by leveraging PVs, we can start using databases like MySQL, Cassandra, CockroachDB and even MS SQL for our applications

A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plugins like Volumes, but have a lifecycle independent of any individual pod that uses the PV. This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

A PersistentVolumeClaim (PVC) is a request for storage by a user. It is similar to a pod. Pods consume node resources and PVCs consume PV resources. Pods can request specific levels of resources (CPU and Memory). Claims can request specific size and access modes (e.g., can be mounted once read/write or many times read-only).

**Types of Persistent Volumes**

PersistentVolume types are implemented as plugins. Kubernetes currently supports the following plugins:

* GCEPersistentDisk
* AWSElasticBlockStore
* AzureFile
* AzureDisk
* FC (Fibre Channel)
* FlexVolume
* Flocker
* NFS
* iSCSI
* RBD (Ceph Block Device)
* CephFS
* Cinder (OpenStack block storage)
* Glusterfs
* VsphereVolume
* Quobyte Volumes
* HostPath (Single node testing only – local storage is not supported in any way and WILL NOT WORK in a multi-node cluster)
* Portworx Volumes
* ScaleIO Volumes
* StorageOS

the important thing here are the access modes:

* **ReadWriteOnce** – Mount a volume as read-write by a single node
* **ReadOnlyMany** – Mount the volume as read-only by many nodes
* **ReadWriteMany** – Mount the volume as read-write by many nodes

Access mode will define how a pod will consume this volume. In most cases, you will set ReadWriteOnce so that only one node can do read-write. Please note that this means more pods on single node can still use the same volume. In some cases for stateless apps you want to have read-only volumes and for that, you need to use ReadOnlyMany.

The rare case is ReadWriteMany because only a few storage providers have the support for it. Think of ReadWriteMany as NFS.