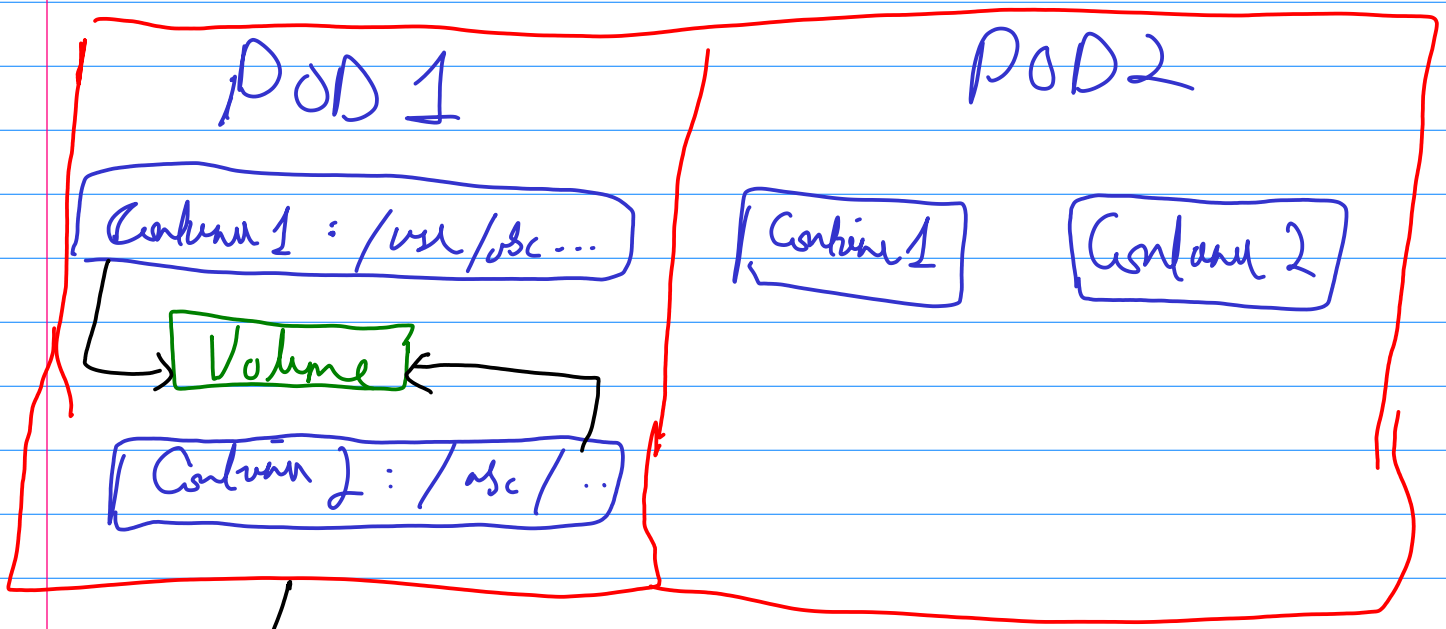
Volumes :-

- Container contains its own directories & files
- If for any reason K8's restart any containers, all files will be lost that we might created at run time.
- Volumes in K8's can be thought of shared directory for the containers in a pod at pod level.
- Volumes are not standalone K8's object & cannot be created / deleted on their own.
- Kubernetes volumes are the component of a pod and are thus defined in the pod's specification

- Volume is available to all containers in the pod, but it must be mounted in each container that needs to access it.



→ If we delete contain 1 or contain 2 then there will be no effect because the files are save at volumes.

Types of Volumes:

- Empty Dir
- configMap, secret
- persistentVolumeClaim

```
kind: Pod
apiVersion: v1
metadata:
  name: pod-with-vol
spec:
  volumes:
  - name: sharedfolder
    emptyDir: {}
  containers:
  - name: container1
    image: usama9876/firstdocker-app
    ports:
    - containerPort: 8080
    volumeMounts:
    - name: sharedfolder
      mountPath: /alpha
  - name: container2
    image: usama9876/myphpapp
    ports:
    - containerPort: 80
    volumeMounts:
    - name: sharedfolder
      mountPath: /beta
```

kubectl exec pod-with-vol -c container1 -it sh

Persistent Volume :-

- Volumes saves us from data lost incase of container restart.
- Volume hold data at a pod level
- What if for any reason kubernetes terminates the pod. e.g rescheduling the pod. ??
- In case of pod termination, data in the volumes will be lost.
- To solve this issue kubernetes provide us option of persistent volume.
- Persistent volume add a volume at a cluster level instead pod level.

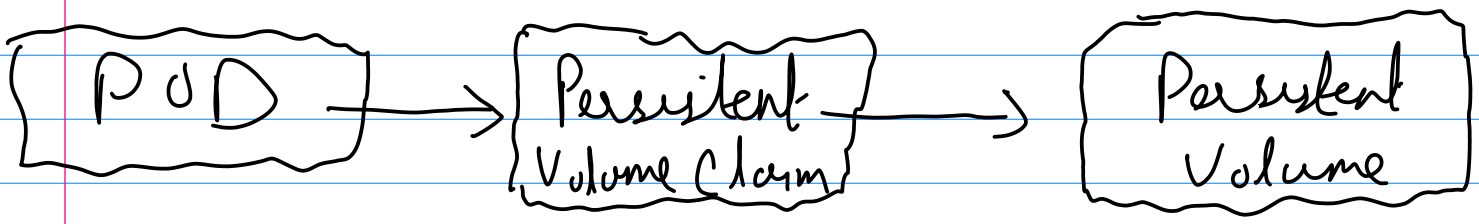
- We create a "PV" resource in which we offer cluster level volume that can be used by any pod.

- Any pod can use the "PV" by using another resource persistent volume claims.

- Kubernetes "PV" remains available out-side of the pod life cycle.

- This means volume will remain even after the pod is deleted.

- This volume will be available to claim by another pod if required, and the data is retained.



Persistent Volume Claim :-

- A kind of formal request from user for claiming a persistent volume.
- A "PVC" describes the amount and characteristics of the storage required by the pod.
- Based on requirement from user PVC finds any matching PV & claims it.

- Depending on the configuration options used for persistent volume resource, then PV resource can later be used/claim by other pods.

Persistent Volume Specs :-

- Access Modes :-

(1) • **ReadWriteOnce**
  - only a single node can mount the volume for reading & writing

(2) • **ReadOnlyMany**
  - Multiple nodes can mount the volume for reading

(3) • ReadWriteMany:

- Multiple nodes can mount the volume for reading & writing

RWO, ROX, RWX pertain to the number of worker nodes that can use the volume at the same time, not to the number of Pods.

Reclaim Policy :-

- Reclaim Policy controls the action the cluster will take when a pod releases its ownership of the storage.
- The lifetime of a PV is determined by its reclaim policy.
- Persistent Volume Reclaim Policy tag can be used in YAML configuration file at the time of creating PV.

- Reclaim Policy can be set to
  - Delete
  - Recycle
  - Retain (Default)

- If PV reclaim policy is delete
  - PV will be deleted when the PVC is deleted but the data will persist

- If PV reclaim policy is recycle
  - Volume's contents will be deleted
  - Persistent volume will be available to be claimed again

- If PV reclaim policy is Retain
  - Kubernetes will retain the volume & its contents after its released from its claim.

  - To make PV available again for claims can be done by delete & re-create the PV resource manually.

  - Underlying storage can either delete or left to be reused by the next pod.