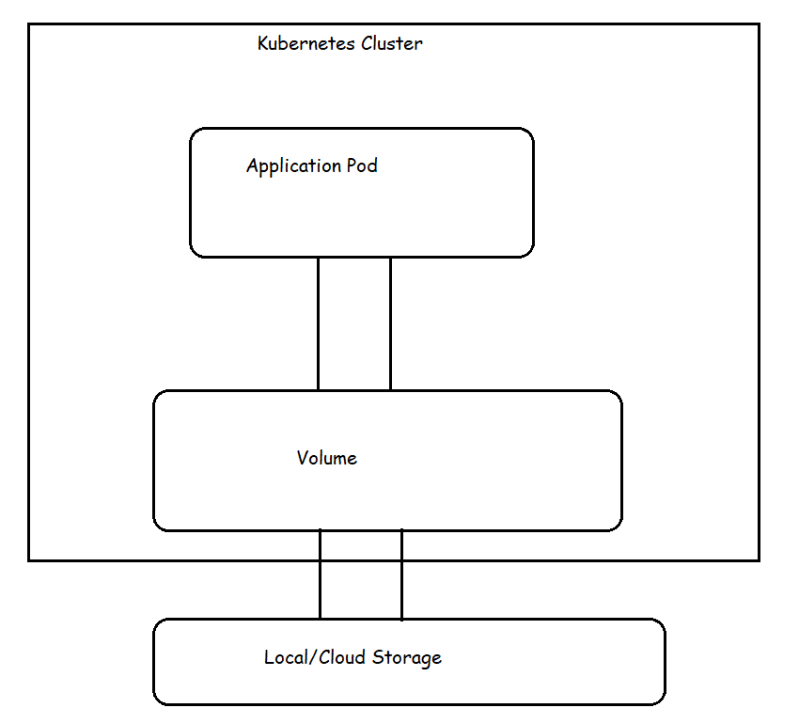
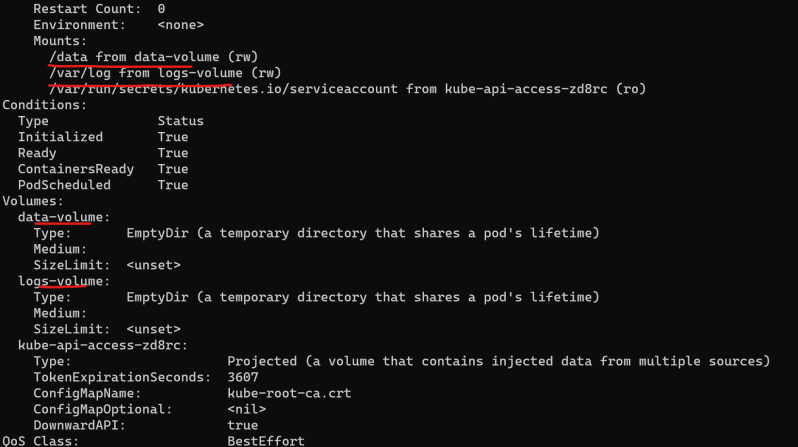
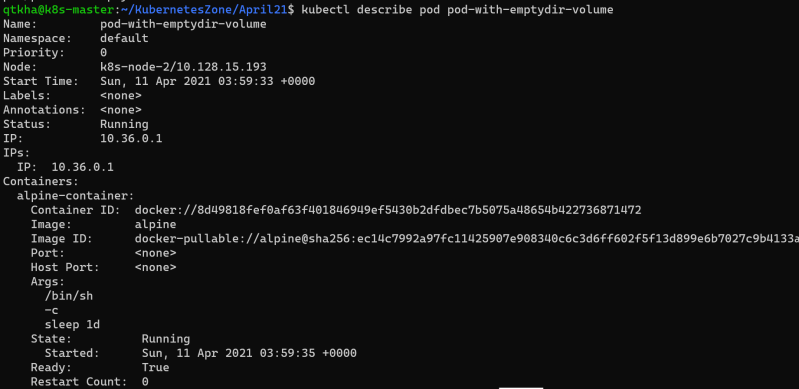
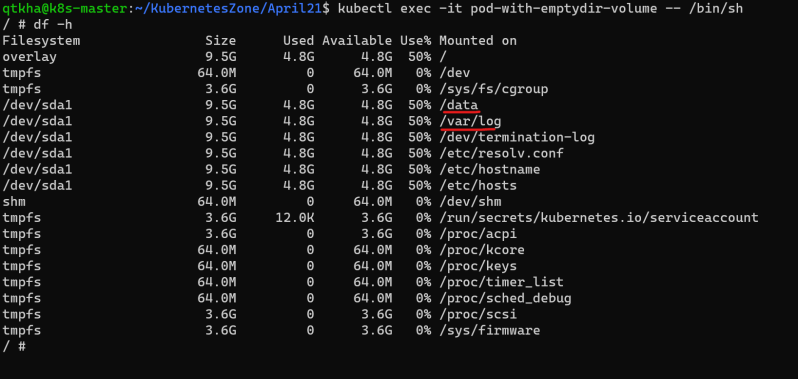
**Kubernetes Volumes**

* 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
* Volume as a storage abstraction for pod applications (In Docker Area)
* Lifetime of k8s volume is same as pod that uses it.
* 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**:
  + **hostPath**: this is used to mount a file or directory from the host node’s file system to a pod
  + **There are other cloud-based** volumes(https://kubernetes.io/docs/concepts/storage/volumes/#volume-types)

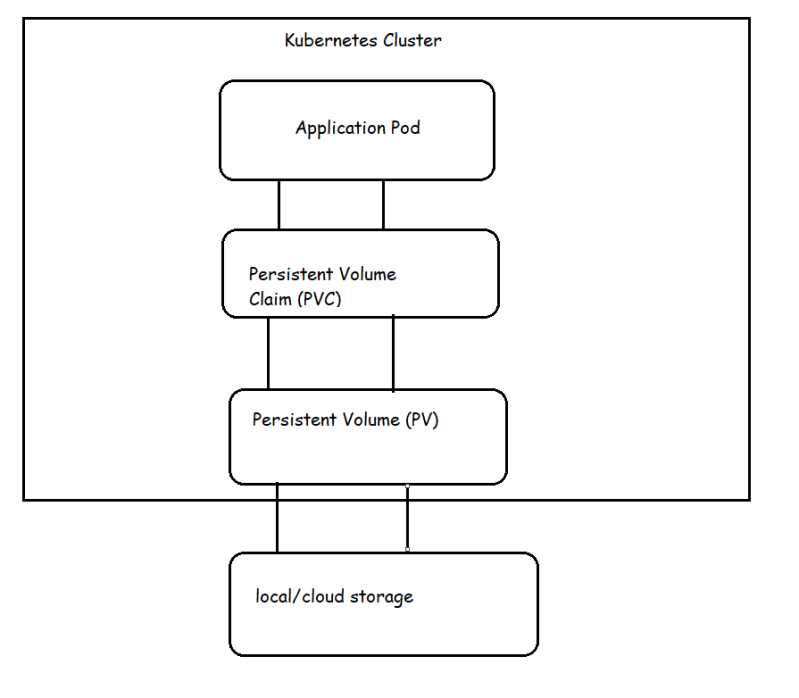
Let’s try a sample pod with volume



Let’s create the pod 

Let’s log in into this pod 

**Persistent Volumes**

* Kubernetes supports persistent storage in the form of Persistent Volume (PV).
* PV is a Kubernetes object that represents block of storage in the cluster.
* This can be either be provisioned beforehand by the admins or we can dynamically provision
* A PV is a cluster resource. Lifecycle of PV doesn’t depend on lifecycle of pod.
* In order to use a PV, a PersistentVolumeClaim (PVC) needs to be created.
* A PVC is a request to the storage by the user or pod.
* PVC can request specific size of storage and specific access mode
* PVC are scoped by namespace
* How PV and PVC work together to provide the storage to application pod 
* k8s supports different types of PersistentVolumes (Link: https://kubernetes.io/docs/concepts/storage/persistent-volumes/#types-of-persistent-volumes)
* Each PV belongs to certain **storage class**.
* Persistent Volume Reclaim policy
  + Retain
  + Recycle
  + Delete
* PV Status
  + Available
  + Bound
  + Released
  + Failed

**Access Modes**

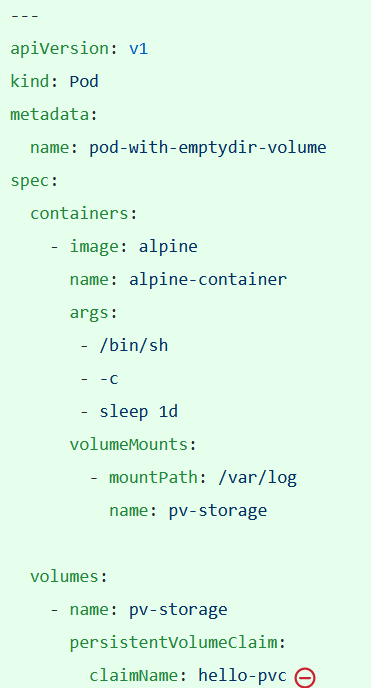
A Persistent Volume can be mounted on a host in any way supported by the resource provider. As shown in the table below, providers will have different capabilities and each PV's access modes are set to the specific modes supported by that particular volume. For example, NFS can support multiple read/write clients, but a specific NFS PV might be exported on the server as read-only. Each PV gets its own set of access modes describing that specific PV's capabilities.

The access modes are:

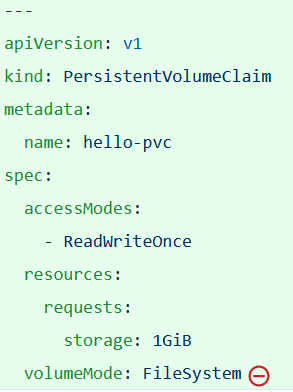
* ReadWriteOnce -- the volume can be mounted as read-write by a single node
* ReadOnlyMany -- the volume can be mounted read-only by many nodes
* ReadWriteMany -- the volume can be mounted as read-write by many nodes

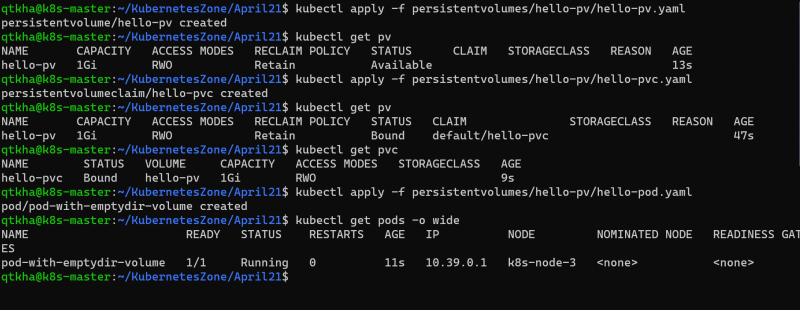
In the CLI, the access modes are abbreviated to:

* RWO - ReadWriteOnce
* ROX - ReadOnlyMany
* RWX - ReadWriteMany









Storage classes:

A StorageClass provides a way for administrators to describe the "classes" of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators. Kubernetes itself is unopinionated about what classes represent. This concept is sometimes called "profiles" in other storage systems