Volume — specification of a pod

- allows for sharing files or data external to the pod

○ Define volume w/in the pod

• Define volume mount which references the volume w/in a container

○ volumes can have many types

empty Dir — temporary storage used to share data between containers while running

gitRepo — volume loads a repository when created

- no private repo support, use sidecar container

hostfile - allows access to node filesystem
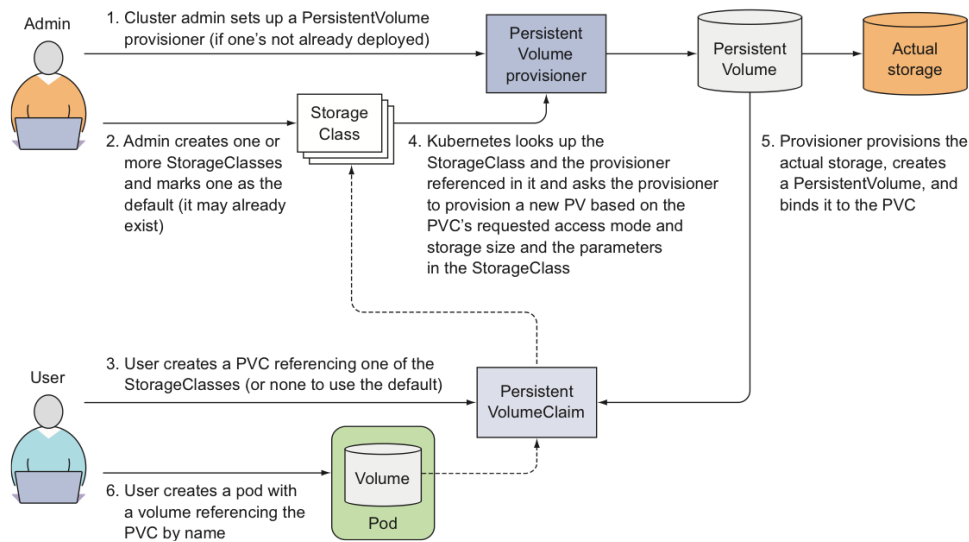- used for Daemon processes (mostly)

Persistent Storage

- store data external to the cluster
- lots of options — NAS, Google, AWS, ...

* A number of ways to use persistent storage but ideally

- abstract it from the pods view
- allow it to work dynamically

Diagram: Dynamic provisioning of PersistentVolumes

Admin
1. Cluster admin sets up a PersistentVolume provisioner (if one's not already deployed)
2. Admin creates one or more StorageClasses and marks one as the default (it may already exist)

Persistent Volume provisioner

Storage Class

4. Kubernetes looks up the StorageClass and the provisioner referenced in it and asks the provisioner to provision a new PV based on the PVC's requested access mode and storage size and the parameters in the StorageClass

Persistent Volume

Actual storage

5. Provisioner provisions the actual storage, creates a PersistentVolume, and binds it to the PVC

User
3. User creates a PVC referencing one of the StorageClasses (or none to use the default)

Persistent VolumeClaim

6. User creates a pod with a volume referencing the PVC by name

Volume / Pod

Actual Storage — dependent on the system

Persistent Volume — storage partitioned from the pool of actual storage
— cluster level resource

Persistent Volume Claim — binds a pod to a persistent volume
— reference claim in your pod, K8s assigns PV from PVC
— reference is consistent

unless deleted or removed

Storage Class – dynamically creates a PV
for a PVC

    – needs a provisioner to dynamically
create PVs

Steps

① create a storage class

② create a PVC

③ link pod to PVC