KUBERNETES – VOLUMES

Github repo..

https://github.com/vsaini44/KubernetesRepo.git

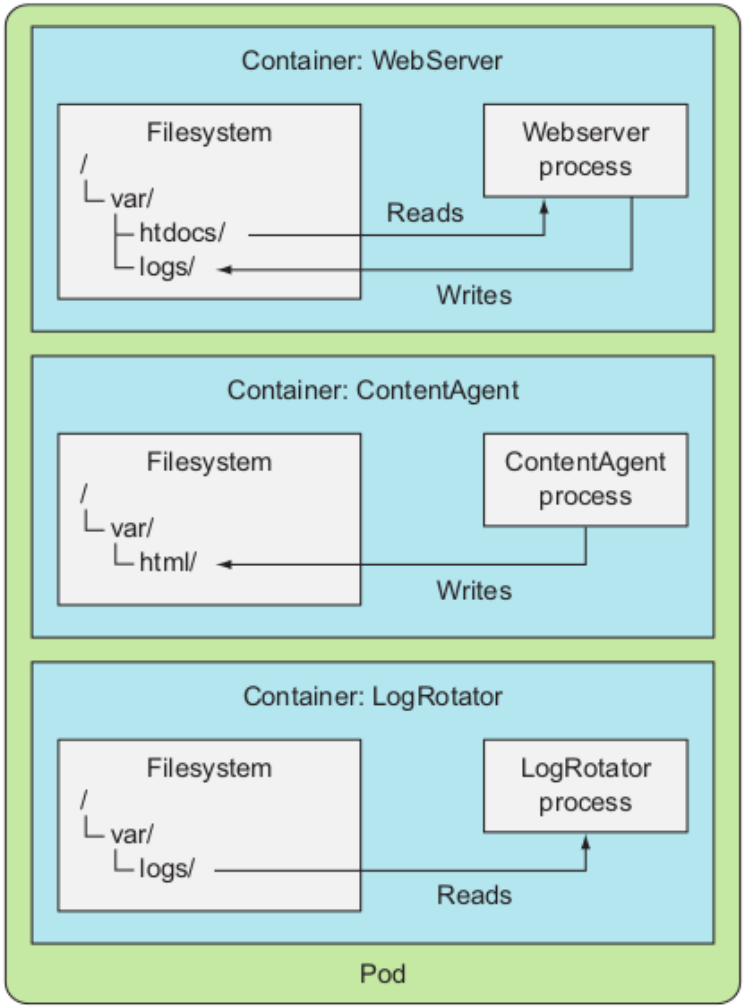
Storage in Containers

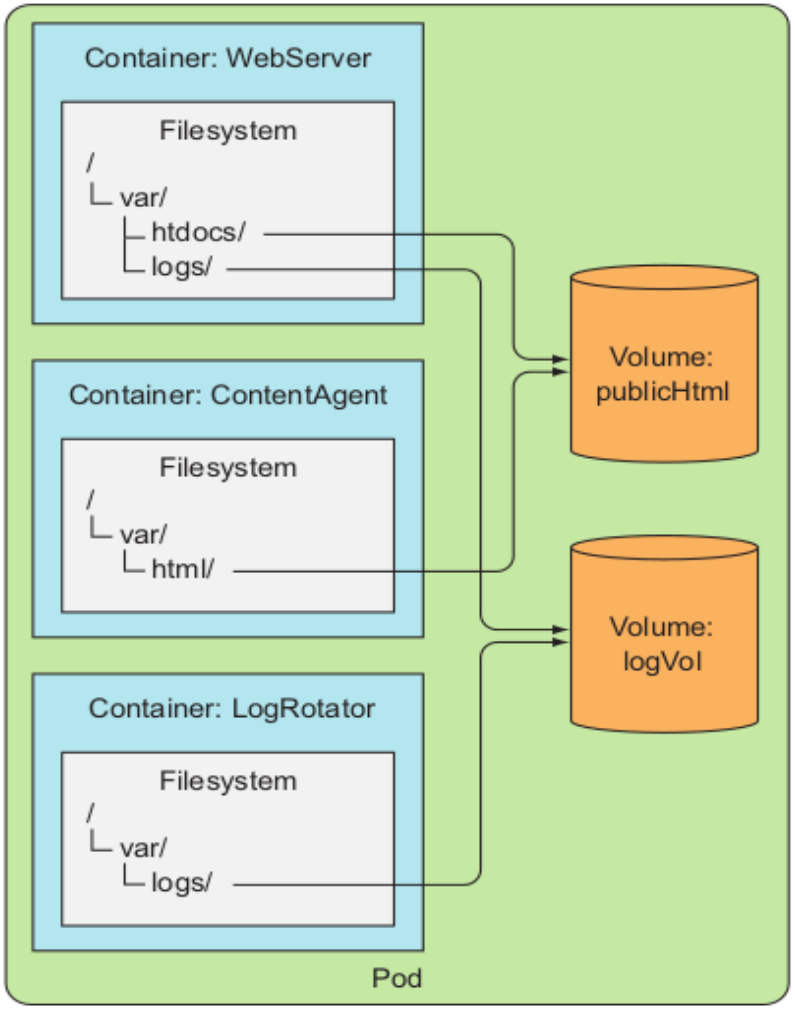
Containers Storage is ephemeral.

In certain scenarios you want the new container to continue where the last one finished.

Kubernetes provides this by defining storage volumes. They aren’t top-level resources like pods, but are instead defined as a part of a pod and share the same lifecycle as the pod. This means a volume is created when the pod is started and is destroyed when the pod is deleted

Volumes ?


Container: WebServer
Filesystem
/
var/
htdocs/
logs/
Webserver process
Reads
Writes
Container: ContentAgent
Filesystem
/
var/
html/
ContentAgent process
Writes
Container: LogRotator
Filesystem
/
var/
logs/
LogRotator process
Reads
Pod


Container: WebServer
Filesystem
/
var/
htdocs/
logs/
Container: ContentAgent
Filesystem
/
var/
html/
Container: LogRotator
Filesystem
/
var/
logs/
Volume: publicHtml
Volume: logVol
Pod

Volume Types in kubernetes

Kubernetes supports varieties of Storage

> emptydir

> hostPath

> git Repo

> nfs

> Google persistent disk , Aws EBS, azuredisk

> configmap, secret

> persistentVolumeClaim

Emptydir

The volume starts out as an empty directory. The app running inside the pod can then write any files it needs to it.

Because the volume's lifetime is tied to that of the pod, the volume's contents are lost when the pod is deleted.

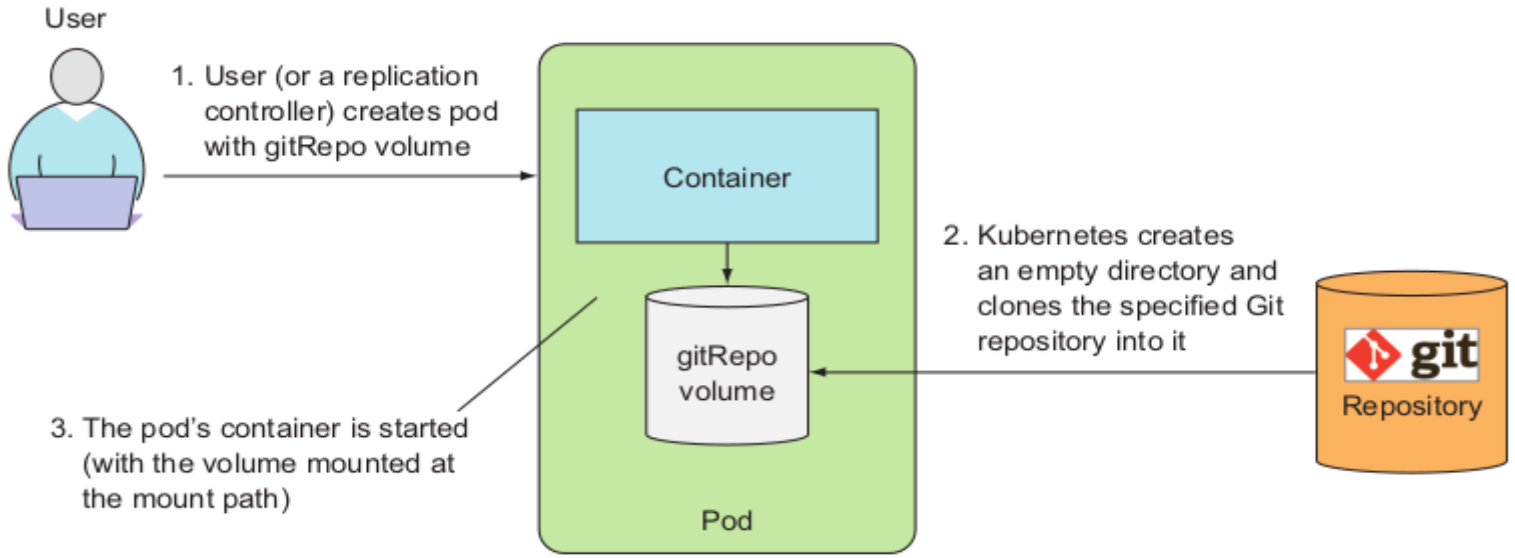
An emptyDir volume is especially useful for sharing files between containers running in the same pod.

EmptyDir example

```
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
    - name: cont1
      image: nginx
      volumeMounts:
        - name: vol1
          mountPath: /testdir
  volumes:
    - name: vol1
      emptyDir: {}
```

Gitrepo

A gitRepo volume is basically an emptyDir volume that gets populated by cloning a Git repository and checking out a specific revision when the pod is starting up (but before its containers are created)

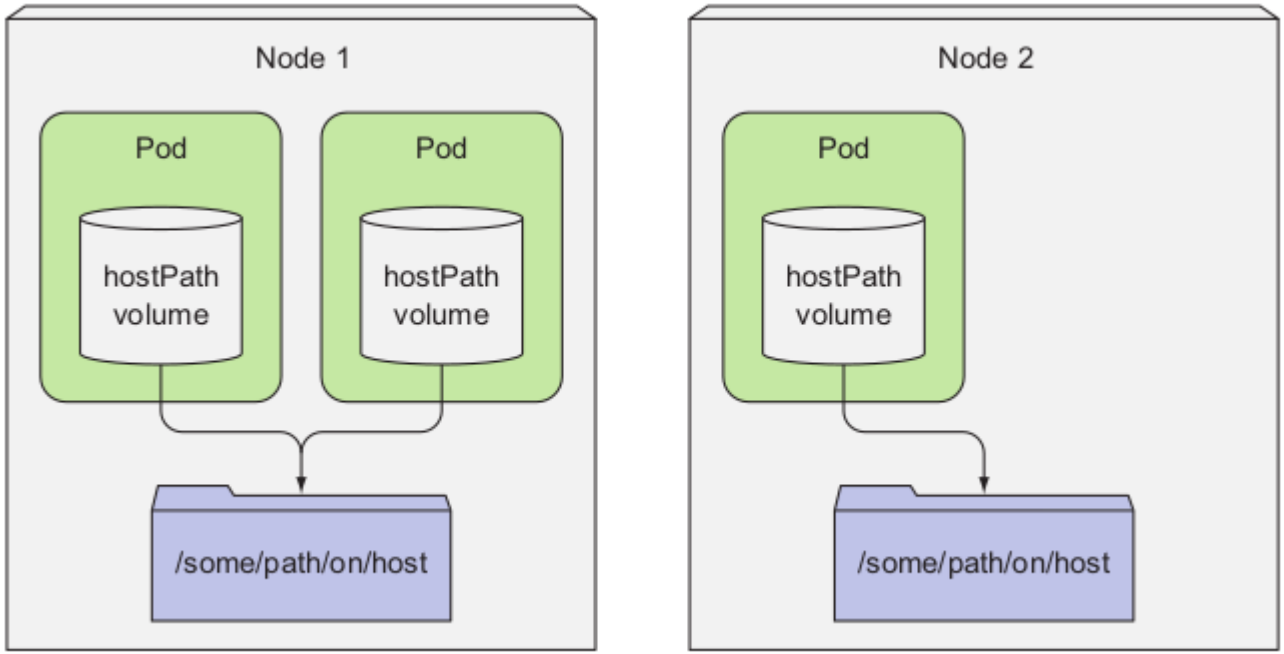


Gitrepo

```
apiVersion: v1
kind: Pod
metadata:
  - name: pod1
spec:
  containers:
    - image: nginx
      name: web-server
      volumeMounts:
        - name: html
          mountPath: /usr/share/nginx/html
  volumes:
    - name: html
      gitRepo:
          repository: https://github.com/vsaini44/webserver
          revision: master
          directory: .
```

hostPath

A hostPath volume points to a specific file or directory on the node's filesystem. Pods running on the same node and using the same path in their host-Path volume see the same files.



PersistentStorage with nfs

If your cluster is running on your own set of servers, you have a vast array of other supported options for mounting external storage inside your volume.

For example, to mount a simple NFS share, you only need to specify the NFS server and the path exported by the server, as shown in the following listing.

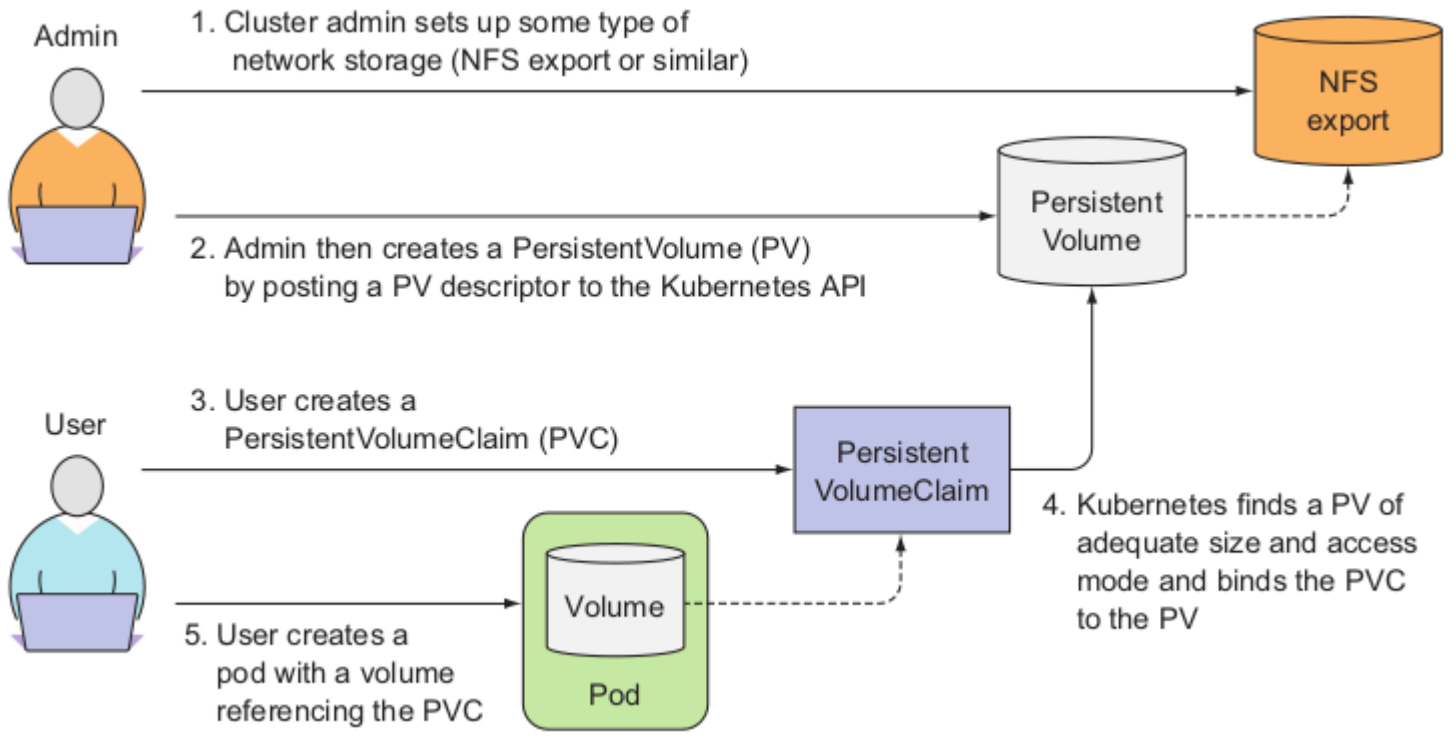
```
volumes:
  - name: mongodb-data
    nfs:
      server: 1.2.3.4
      path: /some/path
```

PV and PVC

To enable apps to request storage in a Kubernetes cluster without having to deal with infrastructure specifics, two new resources were introduced. They are

- **Persistent-Volumes**
- **PersistentVolumeClaims.**

PV and PVC


Admin
1. Cluster admin sets up some type of network storage (NFS export or similar)
NFS export
Persistent Volume
2. Admin then creates a PersistentVolume (PV) by posting a PV descriptor to the Kubernetes API
User
3. User creates a PersistentVolumeClaim (PVC)
Persistent VolumeClaim
4. Kubernetes finds a PV of adequate size and access mode and binds the PVC to the PV
Volume
Pod
5. User creates a pod with a volume referencing the PVC