VOLUMES

About Volumes:-

To Store the data persistently, there are many types in volumes, however we are going to see four types (emptydir, hostpath, GCP persistent volumes (static and dynamic).

emptyDir volume:-

volumes will be created inside the pod, incase pod crashed then data will be lost. so, it's not persistent and not safe.

hostPath:-

in host machine we will be creating directory and that path will be attached to pod, so data will be persistent however the hostpath is machine depend, incase pod got crashed in cluster, then it can be created on any of machine in the cluster, so data which is stored on anyone hostpath, will not visible to the pod, because pod will get created on any node after the crash. there is a concept called node affinity, where we can point the pod to recreate on same node, but which is not advisable.

Persistent volume:-

its elastic storage, while creating pod we can attach the elastic storage from cloud provider, incase pod crash also data will be safer since data will be storage on cloud provider.

Actual Readme:-

On-disk files in a Container are ephemeral, which presents some problems for non-trivial applications when running in Containers. First, when a Container crashes, kubelet will restart it, but the files will be lost - the Container starts with a clean state. Second, when running Containers together in a Pod it is often necessary to share files between those Containers. The Kubernetes Volume abstraction solves both problems. To use a volume, a Pod specifies what volumes to provide for the Pod (the .spec.volumes field) and where to mount those into Containers (the .spec.containers.volumeMounts field).

EmptyDir

An emptyDir volume is first created when a Pod is assigned to a Node, and exists as long as that Pod is running on that node. As the name says, it is initially empty. Containers in the Pod can all read and write the same files in the emptyDir volume, though that volume can be mounted at the same or different paths in each Container. When a Pod is removed from a node for any reason, the data in the emptyDir is deleted forever.

Note: A Container crashing does NOT remove a Pod from a node, so the data in an emptyDir volume is safe across Container crashes.

Some uses for an emptyDir are:

- scratch space, such as for a disk-based merge sort
- checkpointing a long computation for recovery from crashes
- holding files that a content-manager Container fetches while a webserver Container serves the data

By default, emptyDir volumes are stored on whatever medium is backing the node - that might be disk or SSD or network storage, depending on your environment. However, you can set the emptyDir.medium field to "Memory" to tell Kubernetes to mount a tmpfs (RAM-backed filesystem) for you instead. While tmpfs is very fast, be aware that unlike disks, tmpfs is cleared on node reboot and any files you write will count against your Container's memory limit.

hostPath

A hostPath volume mounts a file or directory from the host node's filesystem into your Pod. This is not something that most Pods will need, but it offers a powerful escape hatch for some applications.

For example, some uses for a hostPath are:

running a Container that needs access to Docker internals; use a hostPath of /var/lib/docker

running cAdvisor in a Container; use a hostPath of /sys

allowing a Pod to specify whether a given hostPath should exist prior to the Pod running, whether it should be created, and what it should exist as

In addition to the required path property, user can optionally specify a type for a hostPath volume.

The supported values for field type are:

Value	Behavior
	Empty string (default) is for backward compatibility, which means that no checks will be performed before mounting the hostPath volume.
DirectoryOrCreate	If nothing exists at the given path, an empty directory will be created there as needed with permission set to 0755, having the same group and ownership with Kubelet.
Directory	A directory must exist at the given path
FileOrCreate	If nothing exists at the given path, an empty file will be created there as needed with permission set to 0644, having the same group and ownership with Kubelet.
File	A file must exist at the given path
Socket	A UNIX socket must exist at the given path
CharDevice	A character device must exist at the given path
BlockDevice	A block device must exist at the given path

Watch out when using this type of volume, because:

Pods with identical configuration (such as created from a podTemplate) may behave differently on different nodes due to different files on the nodes

when Kubernetes adds resource-aware scheduling, as is planned, it will not be able to account for resources used by a hostPath

the files or directories created on the underlying hosts are only writable by root. You either need to run your process as root in a privileged Container or modify the file permissions on the host to be able to write to a hostPath volume

GCPersistentDisk

A gcePersistentDisk volume mounts a Google Compute Engine (GCE) Persistent Disk into your Pod. Unlike emptyDir, which is erased when a Pod is removed, the contents of a PD are preserved, and the volume is merely unmounted. This means that a PD can be pre-populated with data, and that data can be "handed off" between Pods.

Caution: You must create a PD using gcloud or the GCE API or UI before you can use it.

There are some restrictions when using a gcePersistentDisk:

the nodes on which Pods are running must be GCE VMs

those VMs need to be in the same GCE project and zone as the PD

A feature of PD is that they can be mounted as read-only by multiple consumers simultaneously. This means that you can pre-populate a PD with your dataset and then serve it in parallel from as many Pods as you need. Unfortunately, PDs can only be mounted by a single consumer in read-write mode - no simultaneous writers allowed.

Using a PD on a Pod controlled by a ReplicationController will fail unless the PD is read-only, or the replica count is 0 or 1

##Create an emptyDir volume##

cat pod-emptydir.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
spec:
  containers:
  - image:  nginx
    name: test-container
    volumeMounts:
    - mountPath: /cache
      name: cache-volume
  volumes:
  - name: cache-volume
    emptyDir: {}
```

kubectl apply -f pod-emptydir.yml

kubectl get pods

```
[root@ansikube manifest]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
test-pod   1/1     Running   0          2m7s
[root@ansikube manifest]#
```

kubectl exec -it test-pod /bin/bash

```
[root@ansikube manifest]# kubectl exec -it test-pod /bin/bash
root@test-pod:/# df -hTP /cache/
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sda1      ext4   95G  2.8G   92G   3% /cache
root@test-pod:/#
```

##Create the hostpath volume##

#cat hostpath-pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd2
spec:
  containers:
  - image: nginx:1.16
    name: test-container
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume1
  volumes:
  - name: test-volume1
    hostPath:
      #directory location on host
      path: /tmp
      #this feild is optional
      type: Directory
```

kubectl apply -f hostpath-pod.yml

#kubectl get pods

```
[root@ansikube manifest]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
test-pd2   1/1     Running   0          21s
test-pod   1/1     Running   0          6m53s
[root@ansikube manifest]#
```

@@Login to pod and create files under /test-pd.

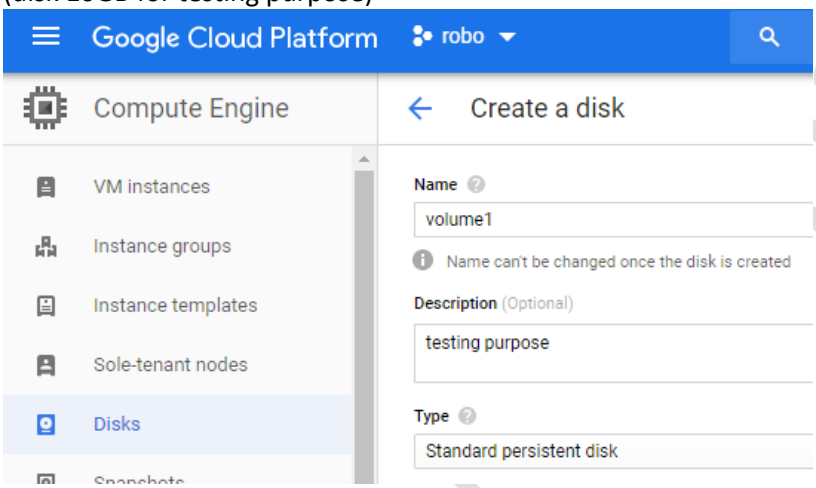
#kubectl exec -it test-pd2 /bin/bash

```
[root@ansikube manifest]# kubectl exec -it test-pd2 /bin/bash
root@test-pd2:/# df -hTP /test-pd
Filesystem     Type   Size  Used Avail Use% Mounted on
tmpfs          tmpfs  1.9G     0  1.9G   0% /test-pd
root@test-pd2:/# cd /test-pd
root@test-pd2:/test-pd# touch tst
root@test-pd2:/test-pd# ls
systemd-private-f48acd99e54045de82c814f582003f99-systemd-resolved.service-cHcuju   tst
systemd-private-f48acd99e54045de82c814f582003f99-systemd-timesyncd.service-KZHrm0
```

Note:- Go to worker node, under /tmp tst file will be visible.

##GcePersistentDisk Static volume##

First step:- go to google cloud --> select Compute Engine --> Disks --> CREATE DISK -->then give the info (disk 10GB for testing purpose)



cat gcppd-pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd3
spec:
  containers:
  - image: nginx
    name: test-container3
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    # This GCE PD must already exist.
    gcePersistentDisk:
      pdName: mydisk1
      fsType: ext4
```

kubectl apply -f gcppd-pod.yml

kubectl get pods

```
[root@ansikube manifest]# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
test-pd3   1/1     Running   0          78s
```

@Login to pod and check the FS details.

kubectl exec -it test-pd3 /bin/bash

```
[root@ansikube manifest]# kubectl exec -it test-pd3 /bin/bash
root@test-pd3:/# df -hTP /test-pd
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sdb       ext4  9.8G   37M  9.8G   1% /test-pd
root@test-pd3:/#
```

:-Create one test file under /test-pd.

```
root@test-pd3:/test-pd# ls -rlt
total 16
drwx------ 2 root root 16384 Nov  6 08:37 lost+found
-rw-r--r-- 1 root root     0 Nov  6 08:40 test
-rw-r--r-- 1 root root     0 Nov  6 08:40 test1
root@test-pd3:/test-pd#
```

@@Testing (delete the pod and recreate, to check whether data is available on pod)@@

kubectl delete -f gcppd-pod.yml

kubectl apply -f gcppd-pod.yml

kubectl get pods

kubectl exec -it test-pd3 /bin/bash

```
[root@ansikube manifest]# kubectl exec -it test-pd3 /bin/bash
root@test-pd3:/# df -hTP /test-pd
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sdb       ext4  9.8G   37M  9.8G   1% /test-pd
root@test-pd3:/# cd /test-pd
root@test-pd3:/test-pd# ls -lrt
total 16
drwx------ 2 root root 16384 Nov  6 08:37 lost+found
-rw-r--r-- 1 root root     0 Nov  6 08:40 test
-rw-r--r-- 1 root root     0 Nov  6 08:40 test1
root@test-pd3:/test-pd#
```

kubectl get pods -o wide

```
[root@ansikube manifest]# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE     IP          NODE
test-pd3   1/1     Running   0          3m28s   10.32.0.13  gke-robo-default-pool-682616c4-ml55
test-pod   1/1     Running   0          35m     10.32.0.10  gke-robo-default-pool-682616c4-ml55
[root@ansikube manifest]#
```

Note:- if you see the wide output of pods above, you will find that first time pod was created on one node and after delete and recreate now it's in different node, however data is visible even pod created on different node.

##Physical Volumes Claims (being used in production) dynamic provisioning##

Step1:- Create a storage class with magnetic persistent disk option (pd-ssd).

kubectl get sc

```
[root@ansikube manifest]# kubectl get sc
NAME                 PROVISIONER            AGE
standard (default)   kubernetes.io/gce-pd   8h
```

cat storageclass1.yml

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gold
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-ssd
```

kubectl apply -f storageclass1.yml
kubectl get sc

```
[root@ansikube manifest]# kubectl get sc
NAME                 PROVISIONER            AGE
gold                 kubernetes.io/gce-pd   14s
standard (default)   kubernetes.io/gce-pd   8h
[root@ansikube manifest]#
```

:- Create a storage class with Standard persistent disk option (pd-standard).

cat storageclass2.yml

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: slow
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  replication-type: none
```

kubectl apply -f storageclass2.yml
kubectl get sc

```
[root@ansikube manifest]# kubectl get sc
NAME                 PROVISIONER            AGE
gold                 kubernetes.io/gce-pd   3m19s
slow                 kubernetes.io/gce-pd   18s
standard (default)   kubernetes.io/gce-pd   9h
```

Step2:- Create a PersistentVolumeClaim.

kubectl get pvc
cat pvc.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: gold
```

kubectl apply -f pvc.yml
kubectl get pvc -o wide

```
[root@ansikube manifest]# kubectl get pvc -o wide
NAME    STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mypvc   Bound    pvc-7da273d0-0074-11ea-81c5-42010a80011e   10Gi       RWO            gold           15s
[root@ansikube manifest]#
```

:- To verify about PhysicalVolumesClaims on cloud level. got to google cloud --> slect Compute Engine --> Disks --> refresh and check.

☐	gke-robo-9db0ce0f-dyna-pvc-7da273d0-0074-11ea-81c5-42010a80011e	✔	SSD persistent disk	10 GB	us-central1-a	None	⋮

Step3:- Create pod with PersistentVolumeClaim

cat pvc-pod.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: mypvc
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```

kubectl apply -f pvc-pod.yml

kubectl get pods

```
[root@ansikube manifest]# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
task-pv-pod   1/1     Running   0          30s
test-pd3      1/1     Running   0          30m
test-pod      1/1     Running   0          62m
[root@ansikube manifest]#
```

@Login to pod and check.

kubectl exec -it task-pv-pod /bin/bash

```
[root@ansikube manifest]# kubectl exec -it task-pv-pod /bin/bash
root@task-pv-pod:/# df -hTP /usr/share/nginx/html
Filesystem     Type  Size  Used Avail Use% Mounted on
/dev/sdc       ext4  9.8G   37M  9.8G   1% /usr/share/nginx/html
root@task-pv-pod:/#
```

Volume Mode:-

Prior to Kubernetes 1.9, all volume plugins created a filesystem on the persistent volume. Now, you can set the value of volumeMode to block to use a raw block device, or filesystem to use a filesystem. filesystem is the default if the value is omitted. This is an optional API parameter.