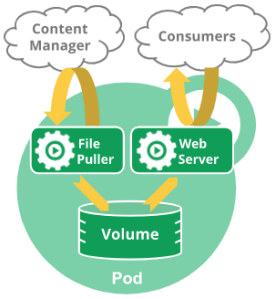
21,25Oct2022
Day**13, 14**
Kubernetes Storage



Pods

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.
Pods natively provide two kinds of shared resources for their constituent containers: **Networking** and **Storage**.

after create K8s cluster, by default one **network plug-in** be available and has been installed, name is **CNCF-CNI PLUGIN** when create POD, Ip will set on it not container/s

Networking

what is Container Network Interface-CNI

https://github.com/containernetworking/cni
CNI-Container Network Interface, a **CNCF** project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins.

Storage

Container Storage Interface-CSI
container storage is ephemeral.
needs map container's storage to outside of pod.
ex
$ kubectl run pod1 --image=nginx
$ kubectl get pods
$ kubectl exec -it pod1 -- ls /
$ kubectl exec -it pod1 -- cat /usr/share/nginx/html/index.html
$ kubectl exec -it pod1 -- /bin/bash
root@pod1:/# curl localhost
Welcome to Nginx!
root@pod1:/# exit
$ kubectl delete pod pod1 --force --grace-period=0
Now, Pod has been deleted, data gone. to solve this issue, need practice on Storage in K8s

Storage

https://kubernetes.io/docs/concepts/storage/
in Kubernetes Pod will get connection to external storage through **PV** and **PVC**

Persistent Volume-PV
https://kubernetes.io/docs/concepts/storage/persistent-volumes/
-The PersistentVolume subsystem provides an API for users and administrators that abstracts details of how storage is provided from how it is consumed.
-A PersistentVolume-PV is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes.
-It is a resource in the cluster just like a node is a cluster resource.
-its cluster-based

Persistent Volume Claim-PVC
https://kubernetes.io/docs/concepts/storage/persistent-volumes/
A PersistentVolumeClaim-PVC is a request for storage by a user.
It is similar to a Pod; Pods consume node resources and PVCs consume PV resources.
Pods can request specific levels of resources (CPU and Memory) and Claims can request specific size and access modes from PV
-namespace/project-based resource

Mohammad Ali Naghval
ali@prodevans.com

Implement Storage in K8s Cluster

step1-create PV
step2-create PVC
step3-maps PVC to PV
step4-maps container/s to PV through PVC

step1-create PV
step2-create PVC
step3-maps PVC to PV

-Reclaiming

when a user is done with their volume, they can delete the PVC objects from the API that allows reclamation of the resource.
The reclaim policy for a PersistentVolume tells the cluster what to do with the volume after it has been released of its claim.
Currently, volumes can either be Retained, Recycled, or Deleted.

1-Retain

The Retain reclaim policy allows for manual reclamation of the resource.
When the PersistentVolumeClaim-PVC is deleted, the PersistentVolume-PV still exists and the volume is considered "released".

2-Delete

deletion removes both the PersistentVolume-PV object from Kubernetes, as well as the associated storage asset in the external infrastructure, such as an AWS EBS, GCE PD, Azure Disk.

3-Recycle

If supported by the underlying volume plugin, the Recycle reclaim policy performs a basic scrub (rm -rf /thevolume/*) on the volume and makes it available again for a new claim.

-Access Modes

A PersistentVolume can be mounted on a host in any way supported by the resource provider.

1-ReadWriteOnce-RWO

the volume can be mounted as read-write by a single node.
ReadWriteOnce access mode still can allow multiple pods to access the volume when the pods are running on the same node.

2-ReadWriteMany-RWX

the volume can be mounted as read-write by many nodes.

3-ReadOnlyMany-ROX

the volume can be mounted as read-only by many nodes.

4-ReadWriteOncePod-RWOP

the volume can be mounted as read-write by a single Pod.
Use ReadWriteOncePod access mode if you want to ensure that only one pod across whole cluster can read that PVC or write to it.

```
$ kubectl api-resources | grep -i "pv"
persistentvolumes                 pv          v1                             false        PersistentVolume
# vim pv1.yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv1
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /mnt/disk1
:wq!
# kubectl apply -f pv1.yaml --dry-run=client
# kubectl apply -f pv1.yaml
# kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv1    1Gi        RWX            Retain           Available                                   2m47s
# vim pvc1.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
spec:
  resources:
    requests:
      storage: 2Gi
  accessModes:
    - ReadWriteOnce
:wq!
# kubectl create -f pvc1.yaml
# kubectl get pvc
NAME   STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc1   Pending                                                    4s
```

NOTE: **PV**, Cluster-based resource
PVC, namespace-based resource

NOTE
to bind PVC to PV first permission will check on PV's then size.
if Permission passes, size should be acceptable, example PVC with higher size than PV wont bind. **check above example**

Now,

```
# kubectl edit pv pv1
    22    storage: 3Gi
:wq!
PV size has been changed/updated
# kubectl get pvc
NAME  STATUS  VOLUME  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc1  Bound   pv1     3Gi       RWX                         58s
# kubectl get pv
NAME  CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM        STORAGECLASS  REASON  AGE
pv1   3Gi       RWX           Retain          Bound   default/pvc1                       80s
```

ex of change in size and permission

```
# vim pv1.yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv1
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  hostPath:
    path: /mnt/disk1
:wq!
# vim pvc1.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
spec:
  resources:
    requests:
      storage: 2Gi
  accessModes:
    - ReadWriteOnce
:wq!
# kubectl apply -f pv1.yaml
# kubectl apply -f pvc1.yaml
# kubectl get pv
NAME  CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM   STORAGECLASS  REASON  AGE
pv1   3Gi       RWX           Retain          Released                                  20m
# kubectl get pvc
NAME  STATUS   VOLUME  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc1  Pending                                                8m21s
```

step4-maps container/s to PV through PVC

```
# kubectl run pod1 --image=nginx -o yaml --dry-run=client >pod1.yaml
# vim pod1.yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: pod1
  name: pod1
spec:
  containers:
  - image: nginx
    name: nginxcnt
    volumeMounts:
      - name: nginxstr
        mountPath: "/usr/share/nginx/html"
  volumes:
    - name: nginxstr
      persistentVolumeClaim:
        claimName: pvc1
:wq!
```

NOTE: container through volumeMount's name(**nginxstr**) will connect to PVC volumes name. both name should be same.

```
# kubectl apply -f pod1.yaml --dry-run=client
# kubectl get pv
# kubectl get pvc
# kubectl get po
```

```
# kubectl expose pod pod1 --name=podint --port=80 --protocol=TCP --type=ClusterIP
# kubectl get svc
# kubectl get po -o wide
NAME   READY   STATUS    RESTARTS   AGE     IP              NODE
pod1   1/1     Running   0          7m13s   172.18.184.3   k8sworker1.example.com
# curl 10.98.40.149
<html>
<head><title>403 Forbidden</title></head>
<body>
<center><h1>403 Forbidden</h1></center>
# ssh root@k8sworker1.example.com ls /mnt
root@k8sworker1.example.com's password:
disk1
# ssh root@k8sworker1.example.com
root@k8sworker1.example.com's password: ****
Last login: Tue Oct 25 19:44:38 2022 from 10.0.0.12
[root@k8sworker1 ~]# echo "Hello!" >/mnt/disk1/index.html
[root@k8sworker1 ~]# exit
# curl 10.98.40.149
Hello!
```

Now, delete POD

```
# kubectl delete pod pod1 --force
# curl 10.98.40.149
curl: (7) Failed to connect to 10.98.40.149 port 80: Connection refused
# ssh root@k8sworker1.example.com ls /mnt/disk1
root@k8sworker1.example.com's password: ****
index.html
[root@k8smaster1 ~]# ssh root@k8sworker1.example.com cat /mnt/disk1/index.html
root@k8sworker1.example.com's password: ****
Hello!
```

Now, delete the PVC

```
# kubectl get pv pv1 -o yaml | grep -i "persistentvolumereclaimpolicy"
  persistentVolumeReclaimPolicy: Retain
# kubectl delete pvc pvc1
# ssh root@k8sworker1.example.com ls /mnt/disk1
root@k8sworker1.example.com's password: ****
index.html
[root@k8smaster1 ~]# ssh root@k8sworker1.example.com cat /mnt/disk1/index.html
root@k8sworker1.example.com's password: ****
Hello!
```