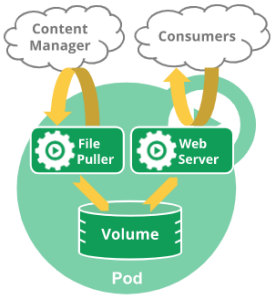
26Oct2022
Day**15**
Kubernetes Storage II-access to remote storage



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

Implement Storage in K8s Cluster

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

PV, PVC parameters explanation

-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.

Mohammad Ali Naghval
ali@prodevans.com

Attach Remote Storage to K8s Deployment

Remote Storages:

1-File-Based Storage Protocol

1-1-NFS

1-2-CIFS

2-Block-Based Storage Protocol

2-1-iSCSI/Tragetcli

1-File-Based Storage Protocol

1-1-NFS Network File Sharing/System

Implement NFS on RHEL

Install NFS on RHEL Linux machine out of K8s Cluster but accessible by K8s Cluster.

login to NFS-Server Host

u: root

p: redhat

-Basic Configuration

-Hostname

```
# hostnamectl set-hostname generic.example.com
# hostnamectl
```

-Network

```
# nmcli connection show
enp0s3  5abdbad1-f4d4-48be-bc18-2c552b4aea9a  ethernet  enp0s3
virbr0  5abdbad1-f4d4-48be-bc18-2c552b4aea9a  ethernet  virbr0
```

NOTE: don't delete virbr0

```
# nmcli connection delete "enp0s3"
# nmcli connection reload
# ifconfig
# nmcli connection add con-name "enp0s3" type ethernet autoconnect yes ifname enp0s3 ipv4.addresses "192.168.29.110/24" ipv4.dns "192.168.29.1"
ipv4.gateway "192.168.29.1" +ipv4.dns "8.8.8.8" +ipv4.dns "8.8.4.4" ipv4.method manual
# nmcli connection reload
# reboot
# nmcli connection show
# ifconfig
# ping 8.8.8.8
```

-Storage

```
# lsblk
sdb           8:16   0    5G  0 disk
# pvcreate /dev/sdb
# vgcreate vg1 /dev/sdb
# lvcreate -n lv1 -l 100%FREE vg1
# pvs
# vgs
# lvs
# mkfs.xfs /dev/mapper/vg1-lv1
# mkdir /mnt/disk1
# ls -ld /mnt/disk1/
drwxr-xr-x 2 root root 6 Oct 26 19:52 /mnt/disk1/
# chmod 757 /mnt/disk1/
# ls -ld /mnt/disk1/
drwxr-xrwx 2 root root 6 Oct 26 19:52 /mnt/disk1/
# blkid
/dev/mapper/vg1-lv1: UUID="c766c56d-f5a5-45e9-8ffa-443d931eaa05" BLOCK_SIZE="512" TYPE="xfs"
# echo "/dev/mapper/vg1-lv1 /mnt/disk1 xfs defaults 0 0" >>/etc/fstab
# mount -a
# df -hT
/dev/mapper/vg1-lv1   xfs       5.0G   68M  5.0G   2% /mnt/disk1
```

-Config NFS-server

install NFS-Server packages

Local Repository

```
# lsblk
sr0          11:0    1 10.7G  0 rom
# mkdir /media/cdrom
# echo "/dev/sr0 /media/cdrom iso9660 defaults 0 0" >>/etc/fstab
# mount -a
# df -hT
/dev/mapper/vg1-lv1  xfs          5.0G   68M  5.0G   2%     /mnt/disk1
/dev/sr0             iso9660      11G    11G     0     100% /media/cdrom
# vim  /etc/yum.repos.d/redhat.repo
[App]
name=AppStream
baseurl=file:///media/cdrom/AppStream/
gpgcheck=0
enabled=1
[Base]
name=BaseOS
baseurl=file:///media/cdrom/BaseOS/
gpgcheck=0
enabled=1
:wq!
# yum repolist
```

EPEL repository

https://docs.fedoraproject.org/en-US/epel/

```
# dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y
# yum repolist
```

RedHat subscription Manager

```
# subscription-manager register
username: naghval                                   ->RHN account
password: ****
# subscription-manager attach --auto
# subscription-manager release --set=8.6            ->its option
# yum repolist
```

Install NFS-Server

disable SELinux and firewalld

```
# sestatus
# getenforce
Enforcing
# vim /etc/selinux/config
SELINUX=disabled
:wq!
# setenforce 0
# systemctl disable firewalld.service && systemctl stop firewalld.service && systemctl mask firewalld.service
# reboot
# sestatus
SELinux status:              disabled
# systemctl status firewalld.service
# yum list nfs-utils rpcbind
# yum install  nfs-utils rpcbind -y
# systemctl enable nfs-server.service
# systemctl start nfs-server.service
# systemctl status nfs-server.service
# df -hT
/dev/mapper/vg1-lv1  xfs      5.0G   68M  5.0G   2% /mnt/disk1
# vim /etc/exports
<export storage>  <to whom should be accessible>(<permissions>)
/mnt/disk1 *.example.com(rw,sync)                                   ->its DNS example/sample
/mnt/disk1 192.168.29.0/24(rw,sync)
:wq!
# systemctl restart nfs-server.service
```

verify

local(on NFS-Server only)

```
# exportfs -rva
exporting 192.168.29.0/24:/mnt/disk1
```

global(client-side)

```
# showmount -e 192.168.29.110
Export list for 192.168.29.110:
/mnt/disk1 192.168.29.0/24
```

Mohammad Ali Naghval
ali@prodevans.com

Install NFS-Client

config K8s Cluster Nodes as NFS-Client
it depends on your K8s Cluster nodes OS

RHEL

```
# yum install nfs-utils -y
# showmount -e 192.168.29.110
```

UBUNTU

```
# apt install nfs-common -y
# ping 192.168.29.110 -c 2
# showmount -e 192.168.29.110
Export list for 192.168.29.110:
/mnt/disk1 192.168.29.0/24
```

NOTE: install 'nfs-common' on whole K8s Cluster Nodes.

Continue on MasterNode

```
# kubectl api-resources | grep -i "pv"
# vim pv1.yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv1
spec:
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  nfs:
    path: /mnt/disk1
    server: 192.168.29.110
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 2Gi
:wq!
# kubectl apply -f pv1.yaml --dry-run=client
# kubectl apply -f pv1.yaml
# kubectl get pv
pv1   3Gi      RWX          Recycle        Bound    default/pvc1                  3s
# kubectl get pvc
pvc1   Bound    pv1     3Gi      RWX                   7s
# kubectl create deployment dpl1 --image nginx --replicas=3 -o yaml --dry-run=client
# kubectl create deployment dpl1 --image nginx --replicas=3 -o yaml --dry-run=server
# kubectl create deployment dpl1 --image nginx --replicas=3 -o yaml --dry-run=client >dpl1.yaml
# vim dpl1.yaml
# vim dpl1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dpl1
  name: dpl1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginxapp
  template:
    metadata:
      labels:
        app: nginxapp
    spec:
      containers:
      - image: nginx
        name: nginxcnt
        volumeMounts:
          - name: nginxstr
            mountPath: "/usr/share/nginx/html"
      volumes:
      - name: nginxstr
        persistentVolumeClaim:
          claimName: pvc1
:wq!
```

```
# kubectl apply -f dpl1.yaml --dry-run=client
# kubectl apply -f dpl1.yaml
# kubectl get deployments.apps
dpl1   3/3    3          3         2m13s
# kubectl get po
dpl1-85f74f8c75-h2f2z   1/1     Running   0       2m53s   172.16.221.1   node2.example.com
dpl1-85f74f8c75-mtccs   1/1     Running   0       2m53s   172.16.206.1   node4.example.com
dpl1-85f74f8c75-nsczg   1/1     Running   0       2m53s   172.16.11.65   node1.example.com
# curl 172.16.221.1
<html>
<head><title>403 Forbidden</title></head>
```

ClusterIP

```
# kubectl expose deployment dpl1 --name dpl1int --port 80 --protocol TCP --type ClusterIP
```

NodePort

```
# kubectl expose deployment dpl1 --name dpl1ext --port 80 --protocol TCP --type NodePort
# kubectl get svc
dpl1ext     NodePort    10.105.142.163   <none>      80:32165/TCP   17s
dpl1int     ClusterIP   10.106.59.137    <none>      80/TCP         52s
```

verify

```
# curl 10.106.59.137
<html>
<head><title>403 Forbidden</title></head>
# hostname -i
192.168.29.104
open web browser
http://192.168.29.104:32165
403 Forbidden
```

now, back to NFS-Server and put customer NGINX server content in to DircetoryIndex file

```
root@master1:~# ssh root@192.168.29.110
password: redhat
[root@generic ~]# echo "Hellooooooooooooooo!" >/mnt/disk1/index.html
# cat /mnt/disk1/index.html
Hellooooooooooooooo!
[root@generic ~]# logout
root@master1:~# # curl 10.106.59.137
Hellooooooooooooooo!
open web browser
http://192.168.29.104:32165
Hellooooooooooooooo!
```