Volume：NFS、CEPH、GFS

PersistentVolume：NFS、CEPG/GFS

K8s：Volume 🡪 CEPH

K8s：不只是管理员在使用。开发人员volume

PV、PVC

PV：由k8s配置的存储，PV同样是集群的一类资源，yaml。

PVC：对PV的申请，

PersistentVolumeClaim

PV文档：<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>

Nfs类型的PV：

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv0003

spec:

capacity:

storage: 5Gi

volumeMode: Filesystem

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Recycle

storageClassName: slow

mountOptions:

- hard

- nfsvers=4.1

nfs:

path: /tmp

server: 172.17.0.2

persistentVolumeReclaimPolicy：

Recycle: 回收，rm -rf

Deployment -> PVC 🡪 PV, Recycle。

Retain：保留。

Delete：PVC –-> PV,删除PVC后PV也会被删掉，这一类的PV，需要支持删除的功能，动态存储默认方式。

Capacity：PV的容量。

volumeMode：挂载的类型，Filesystem，block

accessModes：这个的PV访问模式：

ReadWriteOnce：RWO，可以被单节点以读写的模式挂载。

ReadWriteMany：RWX，可以被多节点以读写的形式挂载。

ReadOnlyMany：ROX，可以被多个节点以只读的形式挂载。

storageClassName：PV的类，可以说是一个类名，PVC和PV的这个名字一样，才能被绑定。

Pv的状态：

Available：空闲的PV，没有被任何PVC绑定。

Bound：已经被PVC绑定

Released：PVC被删除，但是资源未被重新使用

Failed：自动回收失败。

首先创建了一个NFS的PV：

[root@k8s-master01 ~]# cat nfs-pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: pv0001

spec:

capacity:

storage: 5Gi

volumeMode: Filesystem

accessModes:

- ReadWriteMany

persistentVolumeReclaimPolicy: Recycle

storageClassName: nfs-slow

mountOptions:

- hard

- nfsvers=4.1

nfs:

path: /data/k8s-data/testDir

server: 192.168.1.22

创建一个PVC，绑定到指定类型的PV：

[root@k8s-master01 ~]# cat test-pvc.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: myclaim

spec:

accessModes:

- ReadWriteMany

volumeMode: Filesystem

resources:

requests:

storage: 2Gi

storageClassName: nfs-slow

更改deployment使用PVC类型的volume

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: myclaim

spec:

accessModes:

- ReadWriteMany

volumeMode: Filesystem

resources:

requests:

storage: 2Gi

storageClassName: nfs-slow

[root@k8s-master01 ~]# kubectl get deploy demo-nginx -oyaml

apiVersion: apps/v1

kind: Deployment

metadata:

annotations:

deployment.kubernetes.io/revision: "15"

creationTimestamp: "2020-03-16T14:36:14Z"

generation: 16

labels:

app: demo-nginx

name: demo-nginx

namespace: default

resourceVersion: "2644673"

selfLink: /apis/apps/v1/namespaces/default/deployments/demo-nginx

uid: ca0c135f-31a3-464c-b1f2-10f88e9f1c90

spec:

progressDeadlineSeconds: 600

replicas: 2

revisionHistoryLimit: 10

selector:

matchLabels:

app: demo-nginx

strategy:

rollingUpdate:

maxSurge: 25%

maxUnavailable: 1

type: RollingUpdate

template:

metadata:

creationTimestamp: null

labels:

app: demo-nginx

spec:

containers:

- command:

- sh

- -c

- sleep 36000000000

image: nginx

imagePullPolicy: IfNotPresent

name: nginx2

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /mnt

name: cache-volume

- mountPath: /tmp/nfs

name: nfs-test

- command:

- sh

- -c

- sleep 36000000000

image: nginx

imagePullPolicy: IfNotPresent

name: nginx

ports:

- containerPort: 80

name: web

protocol: TCP

resources:

limits:

cpu: 100m

memory: 270Mi

requests:

cpu: 100m

memory: 70Mi

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /etc/nginx/nginx.conf

name: config-volume

subPath: etc/nginx/nginx.conf

- mountPath: /mnt/

name: config-volume-non-subpath

- mountPath: /tmp/1

name: test-hostpath

- mountPath: /tmp/2

name: cache-volume

- mountPath: /tmp/pvc

name: pvc-test

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

shareProcessNamespace: true

terminationGracePeriodSeconds: 30

volumes:

- hostPath:

path: /etc/hosts

type: File

name: test-hostpath

- configMap:

defaultMode: 420

items:

- key: nginx.conf

path: etc/nginx/nginx.conf

name: nginx-conf

name: config-volume

- configMap:

defaultMode: 420

name: nginx-conf

name: config-volume-non-subpath

- emptyDir:

medium: Memory

name: cache-volume

- name: nfs-test

nfs:

path: /data/k8s-data/testDir

server: 192.168.1.22

- name: pvc-test

persistentVolumeClaim:

claimName: myclaim #PVC的名称

很多情况下：

创建PVC之后，一直绑定不上PV（Pending）：

1. PVC的空间申请大小大于PV的大小
2. PVC的StorageClassName没有和PV的一致
3. PVC的accessModes和PV的不一致

创建挂载了PVC的Pod之后，一直处于Pending状态：

1. PVC没有被创建成功，或者被创建
2. PVC和Pod不在同一个Namespace

删除PVC后🡪k8s会创建一个用于回收的Pod🡪根据PV的回收策略进行pv的回收🡪回收完以后PV的状态就会变成可被绑定的状态也就是空闲状态🡪其他的Pending状态的PVC如果匹配到了这个PV，他就能和这个PV进行绑定。

CronJob：

在k8s里面运行周期性的计划任务，crontab。

* \* \* \* \* \* 分时日月周
* 你的计划任务可能需要调用应用的接口。
* 你的计划任务可能需要依赖某些环境。
* php xxx,直接用php项目的镜像进行执行计划任务。
* php-wordpress:v1.0.1
* CronJob被调用的时间，是用的controller-manager的时间。

创建一个CronJob：

[root@k8s-master01 ~]# kubectl run hello --schedule="\*/2 \* \* \* \*" --restart=OnFailure --image=nginx --image-pull-policy=IfNotPresent -- /bin/sh -c "date"

kubectl run --generator=cronjob/v1beta1 is DEPRECATED and will be removed in a future version. Use kubectl run --generator=run-pod/v1 or kubectl create instead.

cronjob.batch/hello created

[root@k8s-master01 ~]# kubectl get cj

NAME SCHEDULE SUSPEND ACTIVE LAST SCHEDULE AGE

hello \*/2 \* \* \* \* False 0 <none> 7s

[root@k8s-master01 ~]# kubectl get cj hello -oyaml

apiVersion: batch/v1beta1

kind: CronJob

metadata:

labels:

run: hello

name: hello

namespace: default

spec:

concurrencyPolicy: Allow #并发调度策略：Allow运行同时运行过个任务。

# Forbid：不运行并发执行。

# Replace：替换之前的任务

failedJobsHistoryLimit: 1 # 保留失败的任务数。

jobTemplate:

metadata:

creationTimestamp: null

spec:

template:

metadata:

creationTimestamp: null

labels:

run: hello

spec:

containers:

- args:

- /bin/sh

- -c

- date

image: nginx

imagePullPolicy: IfNotPresent

name: hello

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

dnsPolicy: ClusterFirst

restartPolicy: OnFailure

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

schedule: '\*/2 \* \* \* \*' #调度的策略 分时日月周

successfulJobsHistoryLimit: 3 # 成功的Job保留的次数

suspend: false # 挂起，true：cronjob不会被执行。

status: {}