注意：该文档仅为新版操作步骤，具体详情请听视频

[第一章 Rook安装 2](#_Toc58956151)

[1.1 下载Rook安装文件 2](#_Toc58956152)

[1.2 配置更改 2](#_Toc58956153)

[1.3 部署rook 3](#_Toc58956154)

[第二章 创建ceph集群 3](#_Toc58956155)

[2.1 配置更改 3](#_Toc58956156)

[2.2 创建Ceph集群 4](#_Toc58956157)

[2.3 安装ceph snapshot控制器 5](#_Toc58956158)

[第三章 安装ceph客户端工具 5](#_Toc58956159)

[第四章 Ceph dashboard 6](#_Toc58956160)

[4.1 暴露服务 6](#_Toc58956161)

[4.2 登录 7](#_Toc58956162)

[第五章 ceph块存储的使用 7](#_Toc58956163)

[5.1 创建StorageClass和ceph的存储池 7](#_Toc58956164)

[5.2 挂载测试 9](#_Toc58956165)

[第六章 共享文件系统的使用 10](#_Toc58956166)

[6.1 创建共享类型的文件系统 10](#_Toc58956167)

[6.2 创建共享类型文件系统的StorageClass 11](#_Toc58956168)

[6.3 挂载测试 11](#_Toc58956169)

[第七章 PVC扩容 12](#_Toc58956170)

[7.1 扩容文件共享型PVC 12](#_Toc58956171)

[7.2 扩容块存储 14](#_Toc58956172)

[第八章 PVC快照 14](#_Toc58956173)

[8.1 块存储快照 14](#_Toc58956174)

[8.1.1 创建snapshotClass 14](#_Toc58956175)

[8.1.2 创建快照 15](#_Toc58956176)

[8.1.3 指定快照创建PVC 16](#_Toc58956177)

[8.1.4 数据校验 16](#_Toc58956178)

[8.2 文件共享类型快照 17](#_Toc58956179)

[第九章 PVC克隆 18](#_Toc58956180)

[第十章 测试数据清理 18](#_Toc58956181)

# Rook安装

## 下载Rook安装文件

下载指定版本Rook，也可以参考视频中的步骤下载最新版

git clone --single-branch --branch v1.5.3 https://github.com/rook/rook.git

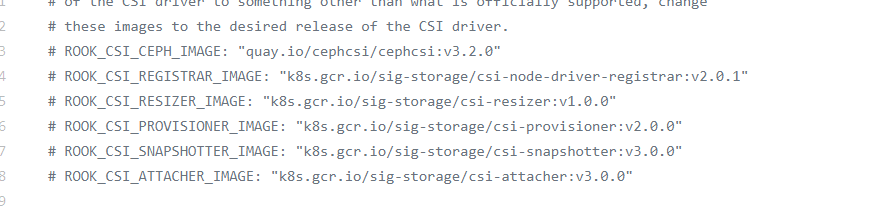
## 配置更改

cd rook/cluster/examples/kubernetes/ceph

修改Rook CSI镜像地址，原本的地址可能是gcr的镜像，但是gcr的镜像无法被国内访问，所以需要同步gcr的镜像到阿里云镜像仓库，文档版本已经为大家完成同步，可以直接修改如下：

vim operator.yaml

将



改为:

ROOK\_CSI\_REGISTRAR\_IMAGE: "registry.cn-beijing.aliyuncs.com/dotbalo/csi-node-driver-registrar:v2.0.1"

ROOK\_CSI\_RESIZER\_IMAGE: "registry.cn-beijing.aliyuncs.com/dotbalo/csi-resizer:v1.0.0"

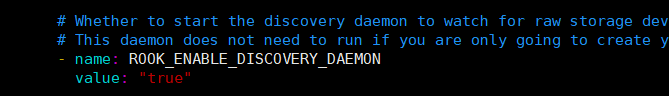
ROOK\_CSI\_PROVISIONER\_IMAGE: "registry.cn-beijing.aliyuncs.com/dotbalo/csi-provisioner:v2.0.0"

ROOK\_CSI\_SNAPSHOTTER\_IMAGE: "registry.cn-beijing.aliyuncs.com/dotbalo/csi-snapshotter:v3.0.0"

ROOK\_CSI\_ATTACHER\_IMAGE: "registry.cn-beijing.aliyuncs.com/dotbalo/csi-attacher:v3.0.0"

如果是其他版本，需要自行同步，同步方法可以在网上找到相关文章。

还是operator文件，新版本rook默认关闭了自动发现容器的部署，可以找到ROOK\_ENABLE\_DISCOVERY\_DAEMON改成true即可：



## 部署rook

1.5.3版本的部署步骤如下：

cd cluster/examples/kubernetes/ceph

kubectl create -f crds.yaml -f common.yaml -f operator.yaml

等待operator容器和discover容器启动

[root@k8s-master01 ceph]# kubectl -n rook-ceph get pod

NAME READY STATUS RESTARTS AGE

rook-ceph-operator-7c7d8846f4-fsv9f 1/1 Running 0 25h

rook-discover-qw2ln 1/1 Running 0 28h

rook-discover-wf8t7 1/1 Running 0 28h

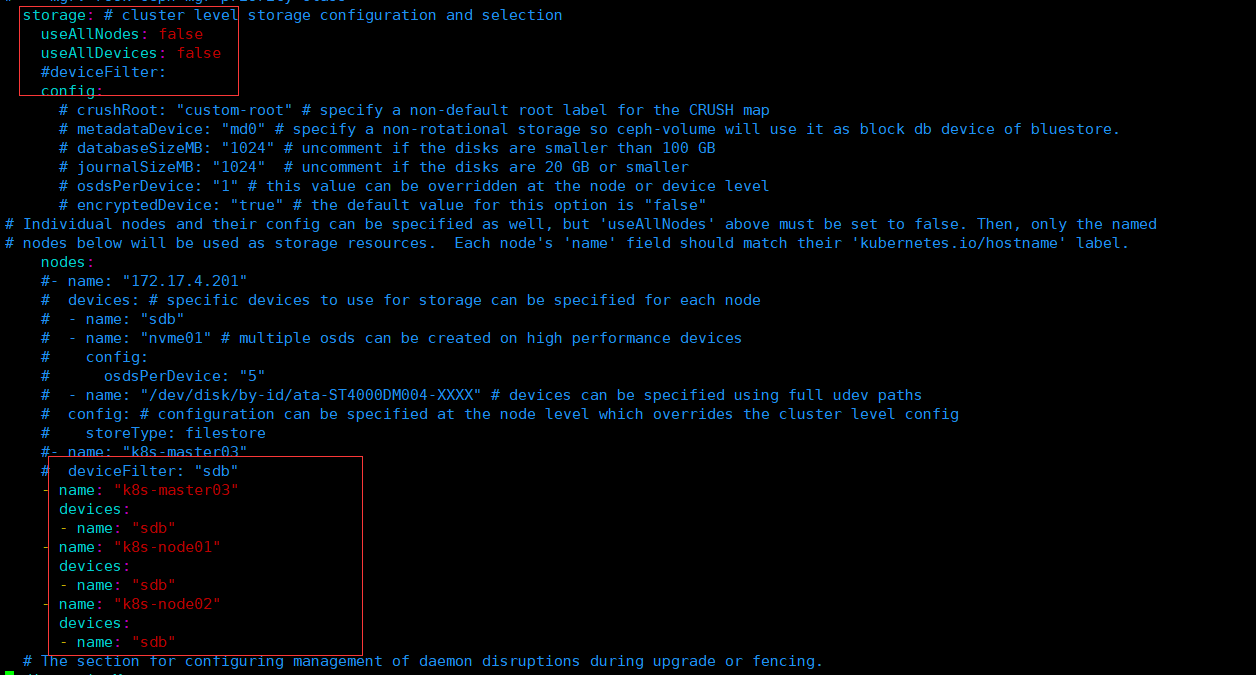
rook-discover-z6dhq 1/1 Running 0 28h

全部变成1/1 Running 才可以创建Ceph集群

# 创建ceph集群

## 配置更改

主要更改的是osd节点所在的位置：



注意：新版必须采用裸盘，即未格式化的磁盘。其中k8s-master03 k8s-node01 node02有新加的一个磁盘，可以通过lsblk -f查看新添加的磁盘名称。建议最少三个节点，否则后面的试验可能会出现问题

## 创建Ceph集群

kubectl create -f cluster.yaml

创建完成后，可以查看pod的状态：

[root@k8s-master01 ceph]# kubectl -n rook-ceph get pod

NAME READY STATUS RESTARTS AGE

csi-cephfsplugin-6nwbp 3/3 Running 0 25h

csi-cephfsplugin-b7h6k 3/3 Running 0 25h

csi-cephfsplugin-provisioner-785798bc8f-78tvr 6/6 Running 0 25h

csi-cephfsplugin-provisioner-785798bc8f-krsdx 6/6 Running 0 25h

csi-rbdplugin-2mmmj 3/3 Running 0 25h

csi-rbdplugin-85sbg 3/3 Running 0 25h

csi-rbdplugin-provisioner-75cdf8cd6d-ghl8f 6/6 Running 0 25h

csi-rbdplugin-provisioner-75cdf8cd6d-wf6h8 6/6 Running 0 25h

rook-ceph-crashcollector-k8s-master03-64c56d8d8b-9vqrk 1/1 Running 0 28h

rook-ceph-crashcollector-k8s-node01-7fc9b79798-6r2rn 1/1 Running 0 28h

rook-ceph-crashcollector-k8s-node02-6954497cb9-pqll7 1/1 Running 0 28h

rook-ceph-mgr-a-dd4bf8445-scsrt 1/1 Running 0 25h

rook-ceph-mon-a-856779ddfd-8v7s2 1/1 Running 0 28h

rook-ceph-mon-b-6c94bddf8c-wb69x 1/1 Running 0 28h

rook-ceph-mon-c-5659bcb5c9-pjn5f 1/1 Running 0 28h

rook-ceph-operator-7c7d8846f4-fsv9f 1/1 Running 0 25h

rook-ceph-osd-0-7c6cdb8546-kkcts 1/1 Running 0 28h

rook-ceph-osd-1-f8b598d47-qjnwl 1/1 Running 0 28h

rook-ceph-osd-2-55846dbcd9-jvm62 1/1 Running 0 28h

rook-ceph-osd-prepare-k8s-master03-h8lt2 0/1 Completed 0 5h1m

rook-ceph-osd-prepare-k8s-node01-jqz7x 0/1 Completed 0 5h1m

rook-ceph-osd-prepare-k8s-node02-hm8lc 0/1 Completed 0 5h1m

rook-discover-qw2ln 1/1 Running 0 29h

rook-discover-wf8t7 1/1 Running 0 29h

rook-discover-z6dhq 1/1 Running 0 29h

需要注意的是，osd-x的容器必须是存在的，且是正常的。如果上述Pod均正常，则认为集群安装成功。

## 安装ceph snapshot控制器

k8s 1.19版本以上需要单独安装snapshot控制器，才能完成pvc的快照功能，所以在此提前安装下，如果是1.19以下版本，不需要单独安装，直接参考视频即可。

snapshot控制器的部署在集群安装时的k8s-ha-install项目中，需要切换到1.20.x分支：

cd /root/k8s-ha-install/

git checkout manual-installation-v1.20.x

创建snapshot controller：

kubectl create -f snapshotter/ -n kube-system

[root@k8s-master01 k8s-ha-install]# kubectl get po -n kube-system -l app=snapshot-controller

NAME READY STATUS RESTARTS AGE

snapshot-controller-0 1/1 Running 0 27h

具体文档：<https://rook.io/docs/rook/v1.5/ceph-csi-snapshot.html>

# 安装ceph客户端工具

[root@k8s-master01 ceph]# pwd

/root/rook/cluster/examples/kubernetes/ceph

[root@k8s-master01 ceph]# kubectl create -f toolbox.yaml -n rook-ceph

deployment.apps/rook-ceph-tools created

待容器Running后，即可执行相关命令

[root@k8s-master01 ceph]# kubectl get po -n rook-ceph -l app=rook-ceph-tools

NAME READY STATUS RESTARTS AGE

rook-ceph-tools-6f7467bb4d-qzsvg 1/1 Running 0 <invalid>

[root@k8s-master01 ceph]# **kubectl -n rook-ceph exec -it deploy/rook-ceph-tools -- bash**

[root@rook-ceph-tools-6f7467bb4d-qzsvg /]# **ceph status**

cluster:

id: 356e0bac-b3e3-4abc-891b-ee6b8a0bc56f

health: HEALTH\_OK

services:

mon: 3 daemons, quorum a,b,c (age 29h)

mgr: a(active, since 5h)

osd: 3 osds: 3 up (since 29h), 3 in (since 29h)

data:

pools: 1 pools, 1 pgs

objects: 0 objects, 0 B

usage: 3.0 GiB used, 57 GiB / 60 GiB avail

pgs: 1 active+clean

[root@rook-ceph-tools-6f7467bb4d-qzsvg /]# **ceph osd status**

ID HOST USED AVAIL WR OPS WR DATA RD OPS RD DATA STATE

0 k8s-node01 1029M 18.9G 0 0 0 0 exists,up

1 k8s-master03 1029M 18.9G 0 0 0 0 exists,up

2 k8s-node02 1029M 18.9G 0 0 0 0 exists,up

[root@rook-ceph-tools-6f7467bb4d-qzsvg /]# **ceph df**

--- RAW STORAGE ---

CLASS SIZE AVAIL USED RAW USED %RAW USED

hdd 60 GiB 57 GiB 17 MiB 3.0 GiB 5.03

TOTAL 60 GiB 57 GiB 17 MiB 3.0 GiB 5.03

--- POOLS ---

POOL ID STORED OBJECTS USED %USED MAX AVAIL

device\_health\_metrics 1 0 B 0 0 B 0 18 GiB

# Ceph dashboard

## 暴露服务

默认情况下，ceph dashboard是打开的，可以通过以下命令查看ceph dashboard的service：

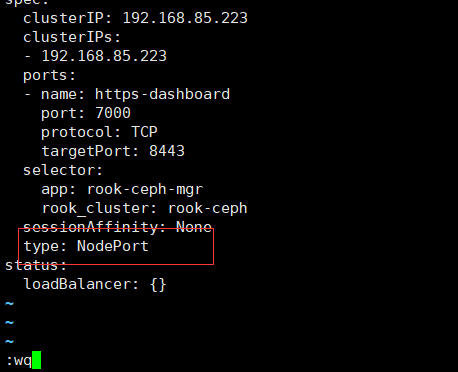
kubectl -n rook-ceph get service rook-ceph-mgr-dashboard

可以两种方式访问：

1. 将该service改为NodePort
2. 通过ingress代理

本文档演示NodePort，ingress可以参考课程的ingress章节。

kubectl -n rook-ceph edit service rook-ceph-mgr-dashboard

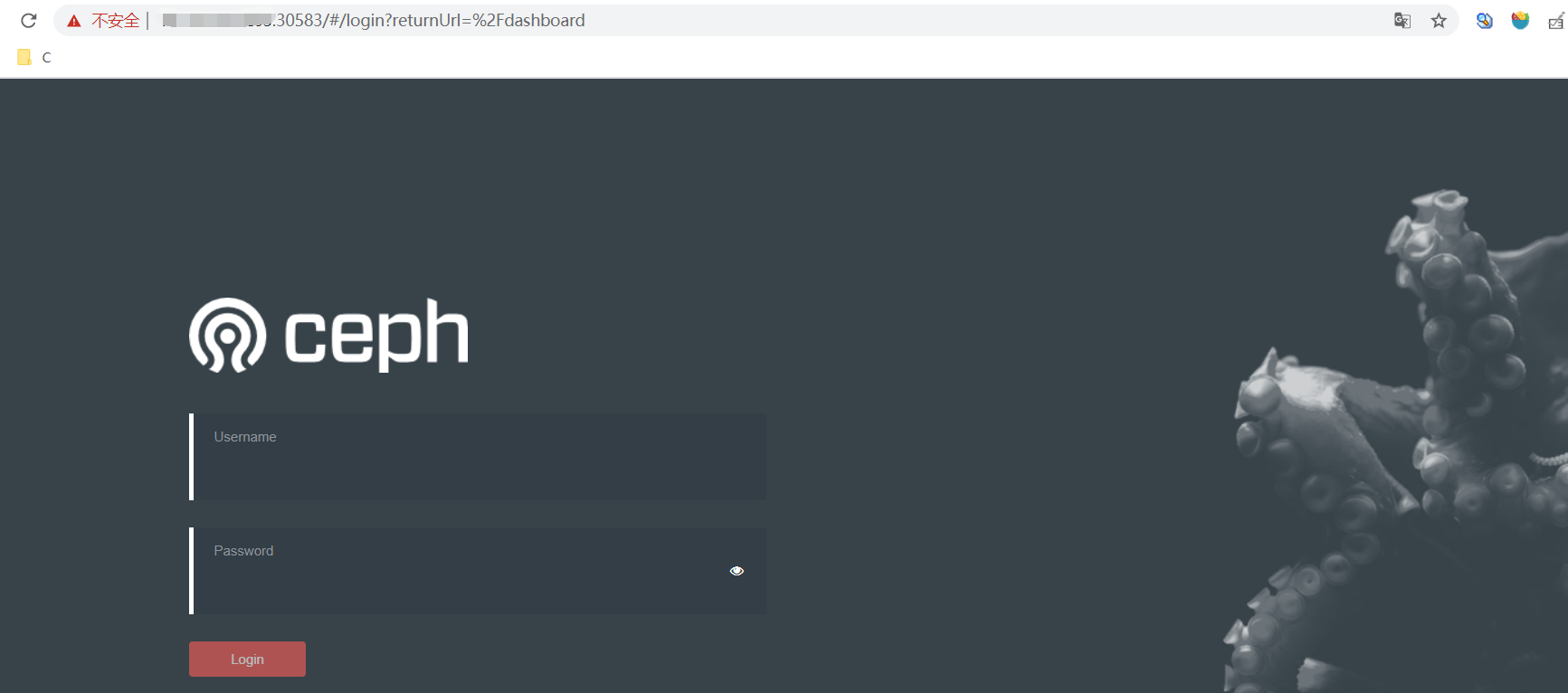


保存退出后，会创建一个端口，然后通过任意k8s节点的IP+该端口即可访问该dashboard：

[root@k8s-master01 ceph]# kubectl get svc -n rook-ceph rook-ceph-mgr-dashboard

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

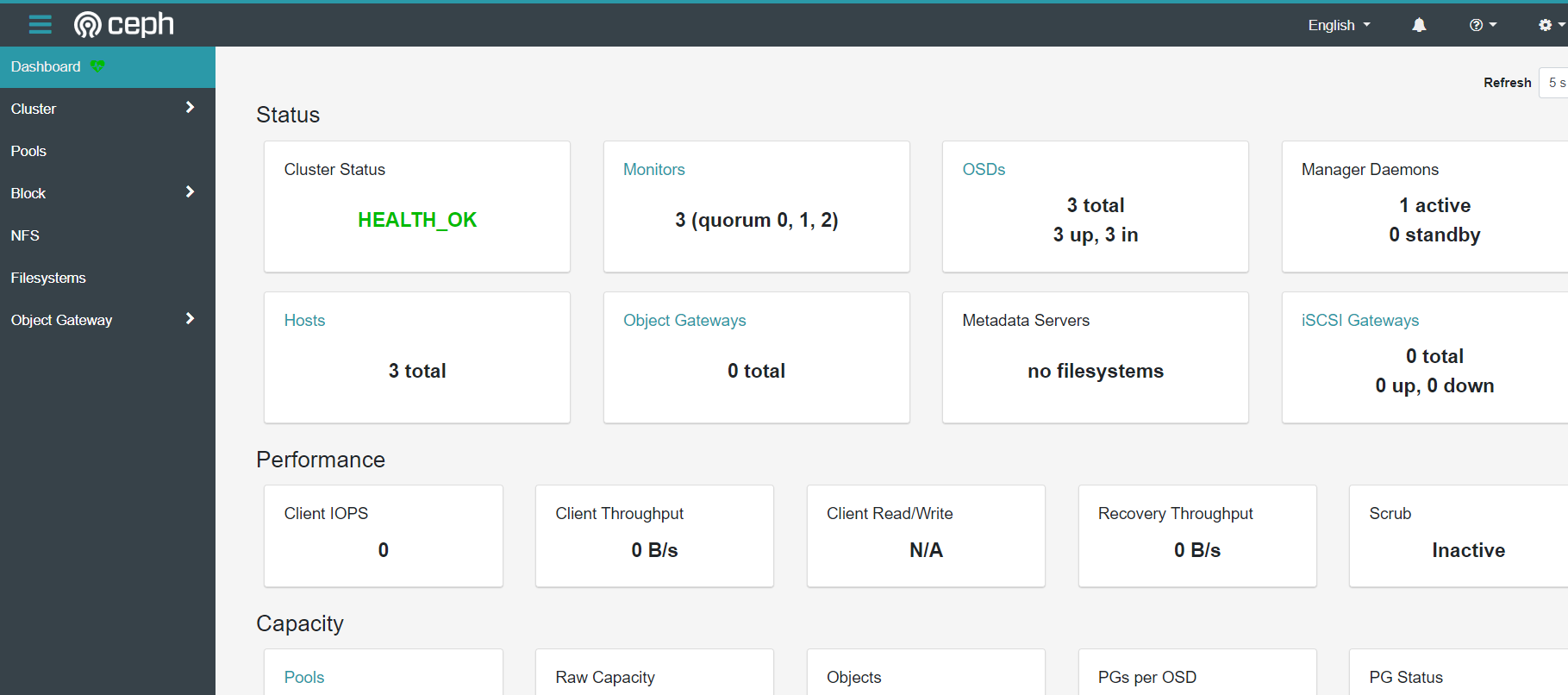
rook-ceph-mgr-dashboard NodePort 192.168.85.223 <none> 7000:30583/TCP 29h



## 登录

账号为admin，查看密码：

kubectl -n rook-ceph get secret rook-ceph-dashboard-password -o jsonpath="{['data']['password']}" | base64 --decode && echo



# ceph块存储的使用

块存储一般用于一个Pod挂载一块存储使用，相当于一个服务器新挂了一个盘，只给一个应用使用。

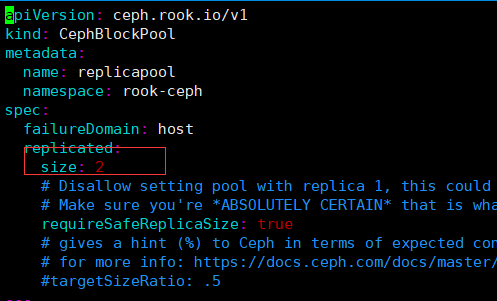
## 创建StorageClass和ceph的存储池

[root@k8s-master01 ceph]# pwd

/root/rook/cluster/examples/kubernetes/ceph

[root@k8s-master01 ceph]# vim csi/rbd/storageclass.yaml

因为我是试验环境，所以将副本数设置成了2（不能设置为1），生产环境最少为3，且要小于等于osd的数量，其他配置可以参考视频：



创建StorageClass和存储池：

[root@k8s-master01 ceph]# kubectl create -f csi/rbd/storageclass.yaml -n rook-ceph

cephblockpool.ceph.rook.io/replicapool created

storageclass.storage.k8s.io/rook-ceph-block created

查看创建的cephblockpool和storageClass（StorageClass没有namespace隔离性）：

[root@k8s-master01 ceph]# kubectl get cephblockpool -n rook-ceph

NAME AGE

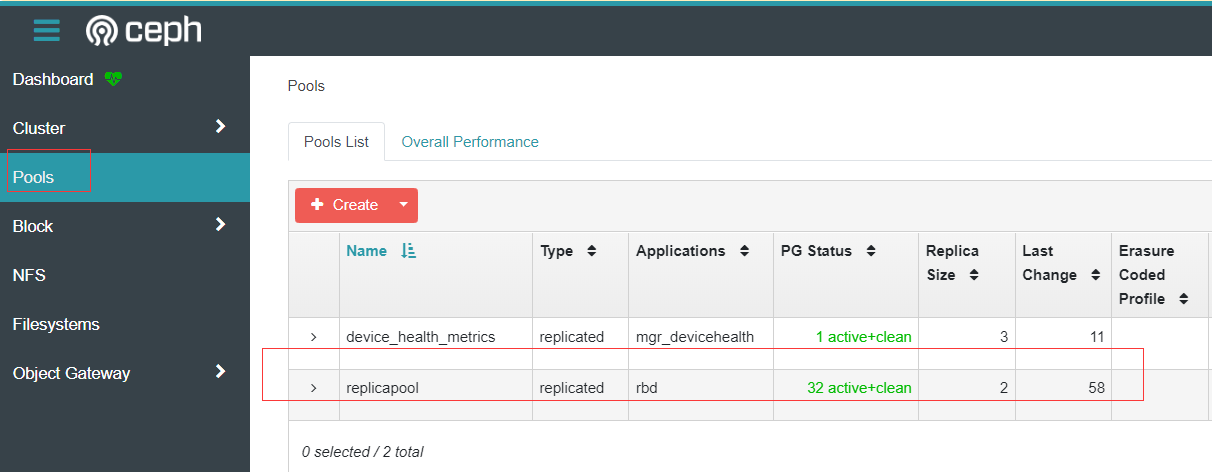
replicapool 39s

[root@k8s-master01 ceph]# kubectl get sc

NAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE

rook-ceph-block rook-ceph.rbd.csi.ceph.com Delete Immediate true 107s

此时可以在ceph dashboard查看到改Pool，如果没有显示说明没有创建成功



## 挂载测试

创建一个MySQL服务

[root@k8s-master01 kubernetes]# pwd

/root/rook/cluster/examples/kubernetes

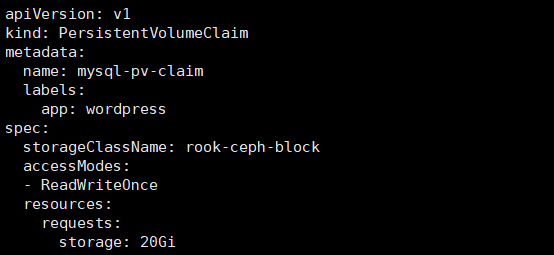
[root@k8s-master01 kubernetes]# kubectl create -f mysql.yaml

service/wordpress-mysql created

persistentvolumeclaim/mysql-pv-claim created

deployment.apps/wordpress-mysql created

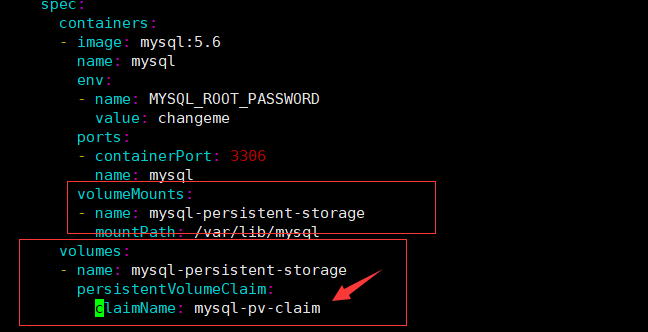
该文件有一段pvc的配置



pvc会连接刚才创建的storageClass，然后动态创建pv，然后连接到ceph创建对应的存储

之后创建pvc只需要指定storageClassName为刚才创建的StorageClass名称即可连接到rook的ceph。如果是statefulset，只需要将volumeTemplateClaim里面的Claim名称改为StorageClass名称即可动态创建Pod，具体请听视频。

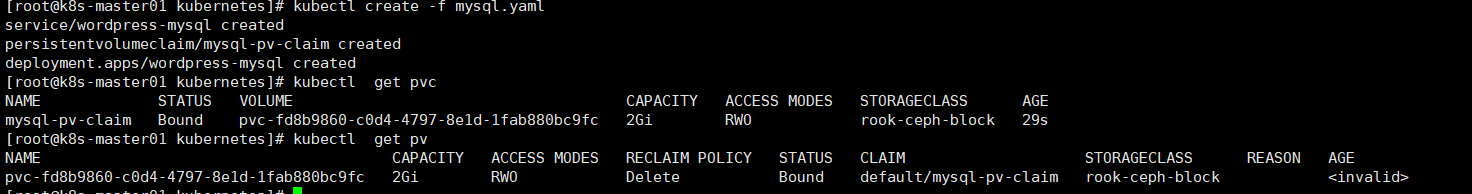
其中MySQL deployment的volumes配置挂载了该pvc：



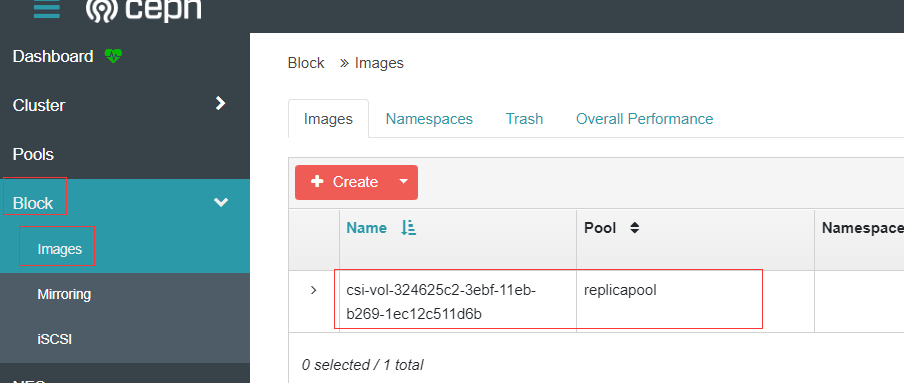
claimName为pvc的名称

因为MySQL的数据不能多个MySQL实例连接同一个存储，所以一般只能用块存储。相当于新加了一块盘给MySQL使用。

创建完成后可以查看创建的pvc和pv



此时在ceph dashboard上面也可以查看到对应的image



# 共享文件系统的使用

共享文件系统一般用于多个Pod共享一个存储

## 创建共享类型的文件系统

[root@k8s-master01 ceph]# pwd

/root/rook/cluster/examples/kubernetes/ceph

[root@k8s-master01 ceph]# kubectl create -f filesystem.yaml

cephfilesystem.ceph.rook.io/myfs created

[root@k8s-master01 ceph]# vim filesystem.yaml

创建完成后会启动mds容器，需要等待启动后才可进行创建pv

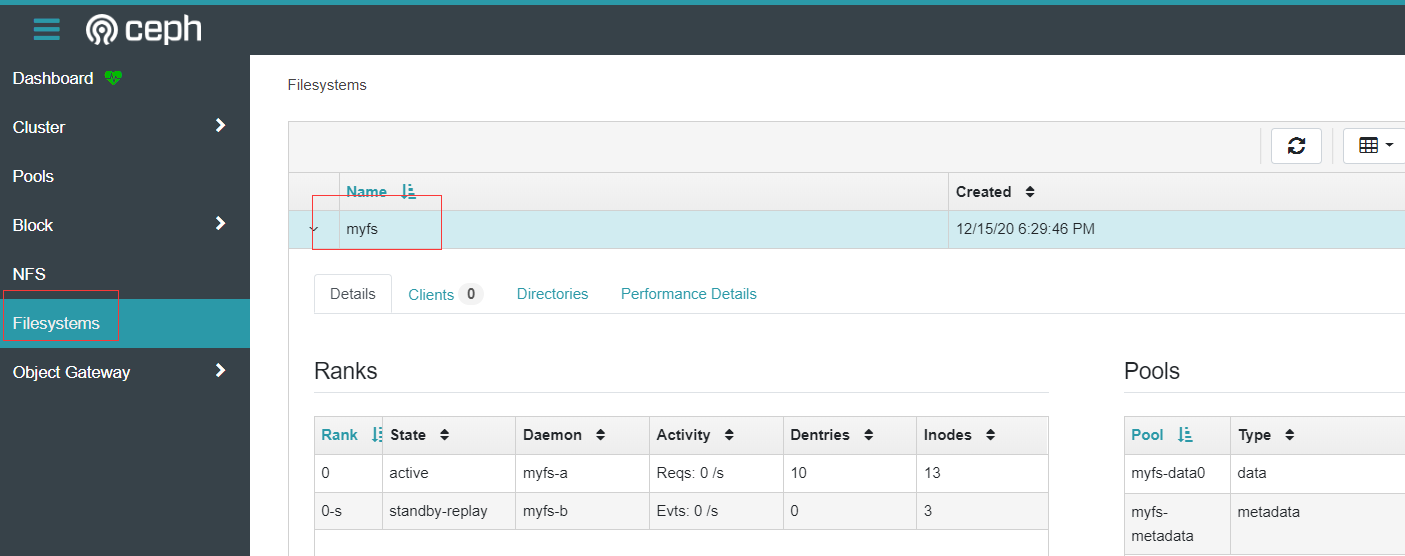
[root@k8s-master01 ceph]# kubectl -n rook-ceph get pod -l app=rook-ceph-mds

NAME READY STATUS RESTARTS AGE

rook-ceph-mds-myfs-a-5979d75f84-m9nv9 1/1 Running 0 2m38s

rook-ceph-mds-myfs-b-55c794c7b8-pr2vs 1/1 Running 0 2m37s

也可以在ceph dashboard上面查看状态



## 创建共享类型文件系统的StorageClass

[root@k8s-master01 cephfs]# pwd

/root/rook/cluster/examples/kubernetes/ceph/csi/cephfs

[root@k8s-master01 cephfs]# kubectl create -f storageclass.yaml

storageclass.storage.k8s.io/rook-cephfs created

之后将pvc的storageClassName设置成rook-cephfs即可创建共享文件类型的存储，类似于NFS，可以给多个Pod共享数据。

## 挂载测试

[root@k8s-master01 cephfs]# pwd

/root/rook/cluster/examples/kubernetes/ceph/csi/cephfs

[root@k8s-master01 cephfs]# ls

kube-registry.yaml pod.yaml pvc-clone.yaml pvc-restore.yaml pvc.yaml snapshotclass.yaml snapshot.yaml storageclass.yaml

[root@k8s-master01 cephfs]# kubectl create -f kube-registry.yaml

persistentvolumeclaim/cephfs-pvc created

deployment.apps/kube-registry created

查看创建的pvc和pod

[root@k8s-master01 cephfs]# kubectl get po -n kube-system -l k8s-app=kube-registry

NAME READY STATUS RESTARTS AGE

kube-registry-66d4c7bf47-46bpq 1/1 Running 0 34s

kube-registry-66d4c7bf47-jp88x 1/1 Running 0 34s

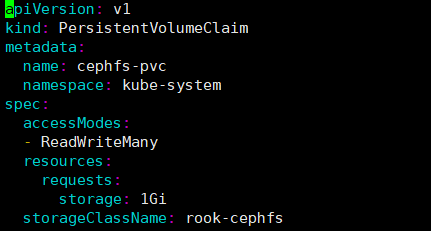
kube-registry-66d4c7bf47-xdtpx 1/1 Running 0 34s

[root@k8s-master01 cephfs]# kubectl get pvc -n kube-system

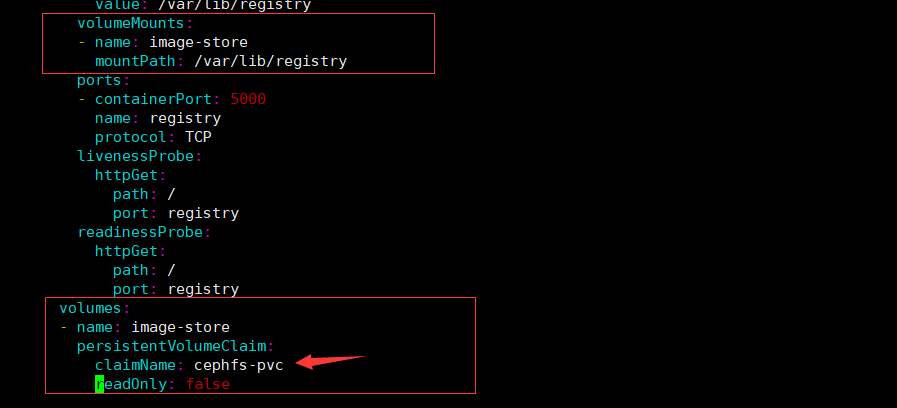
NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

cephfs-pvc Bound pvc-0149518a-7346-4d16-9030-97b2f9b8e9d2 1Gi RWX rook-cephfs 100s

该文件的pvc配置，用于连接ceph创建存储，如下：



然后在deployment的volumes配置添加了该pvc，并将其挂载到了/var/lib/registry



注意claimName为pvc的名称。

此时一共创建了三个Pod，这三个Pod共用了一个存储，挂载到了/var/lib/registry，该目录三个容器共享数据。

# PVC扩容

文件共享类型的PVC扩容需要k8s 1.15+

块存储类型的PVC扩容需要k8s 1.16+

PVC扩容需要开启ExpandCSIVolumes，新版本的k8s已经默认打开了这个功能，可以查看自己的k8s版本是否已经默认打开了该功能：

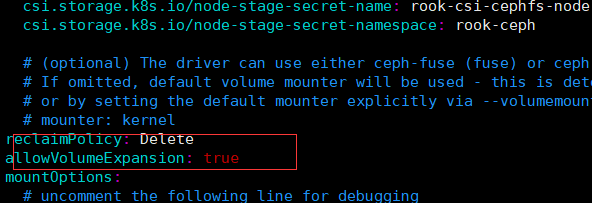
[root@k8s-master01 kubernetes]# kube-apiserver -h |grep ExpandCSIVolumes

ExpandCSIVolumes=true|false (BETA - default=true)

如果default为true就不需要打开此功能，如果default为false，需要根据视频开启该功能。

## 扩容文件共享型PVC

找到刚才创建的文件共享型StorageClass，将allowVolumeExpansion设置为true（新版rook默认为true，如果不为true更改后执行kubectl replace即可）：



找到第六章创建的pvc：

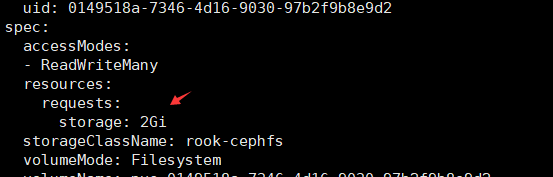
[root@k8s-master01 cephfs]# kubectl get pvc -n kube-system

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

cephfs-pvc Bound pvc-0149518a-7346-4d16-9030-97b2f9b8e9d2 1Gi RWX rook-cephfs 44m

[root@k8s-master01 cephfs]# kubectl edit pvc cephfs-pvc -n kube-system

将大小修改为2Gi，之前是1Gi



保存退出：

[root@k8s-master01 cephfs]# kubectl edit pvc cephfs-pvc -n kube-system

persistentvolumeclaim/cephfs-pvc edited

查看PV和PVC的大小：

[root@k8s-master01 cephfs]# kubectl get pvc -n kube-system

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

cephfs-pvc Bound pvc-0149518a-7346-4d16-9030-97b2f9b8e9d2 2Gi RWX rook-cephfs 46m

[root@k8s-master01 cephfs]# kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

pvc-0149518a-7346-4d16-9030-97b2f9b8e9d2 2Gi RWX Delete Bound kube-system/cephfs-pvc rook-cephfs 47m

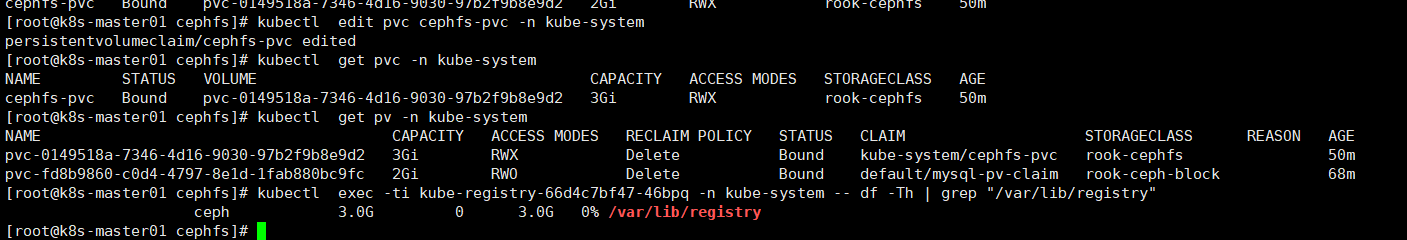
pvc-fd8b9860-c0d4-4797-8e1d-1fab880bc9fc 2Gi RWO Delete Bound default/mysql-pv-claim rook-ceph-block 64m

查看容器内是否已经完成扩容：

[root@k8s-master01 cephfs]# kubectl exec -ti kube-registry-66d4c7bf47-46bpq -n kube-system -- df -Th | grep "/var/lib/registry"

ceph 2.0G 0 2.0G 0% /var/lib/registry

同样的步骤可以扩容到3G：



## 扩容块存储

扩容步骤类似，找到第五章创建PVC，直接edit即可

[root@k8s-master01 cephfs]# kubectl edit pvc mysql-pv-claim

persistentvolumeclaim/mysql-pv-claim edited

[root@k8s-master01 cephfs]# kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

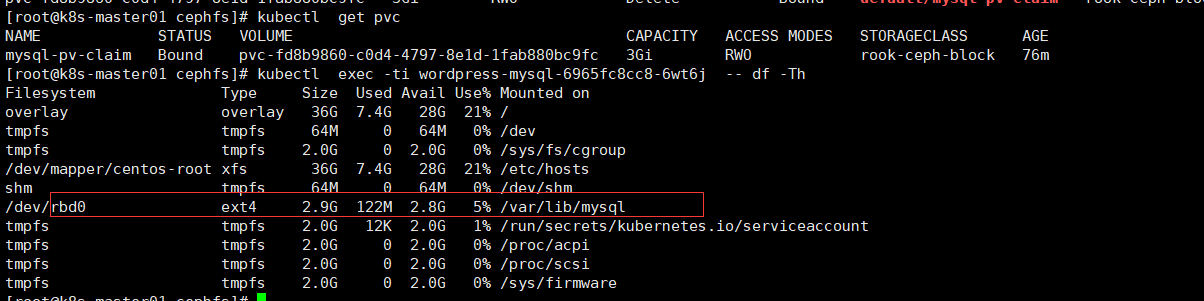
mysql-pv-claim Bound pvc-fd8b9860-c0d4-4797-8e1d-1fab880bc9fc 2Gi RWO rook-ceph-block 70m

可以看到此时pvc并没有扩容，但是pv已经扩容：

[root@k8s-master01 cephfs]# kubectl get pv | grep default/mysql-pv-claim

pvc-fd8b9860-c0d4-4797-8e1d-1fab880bc9fc 3Gi RWO Delete Bound default/mysql-pv-claim rook-ceph-block 75m

也可以看到ceph dashboard的image也完成了扩容，但是pvc和pod里面的状态会有延迟，大概等待5-10分钟后，即可完成扩容：



# PVC快照

注意：PVC快照功能需要k8s 1.17+

## 块存储快照

具体原理和注意事项参考视频

### 创建snapshotClass

[root@k8s-master01 rbd]# pwd

/root/rook/cluster/examples/kubernetes/ceph/csi/rbd

[root@k8s-master01 rbd]# kubectl create -f snapshotclass.yaml

volumesnapshotclass.snapshot.storage.k8s.io/csi-rbdplugin-snapclass created

### 创建快照

首先在之前创建的MySQL容器里创建一个文件夹，并创建一个文件

[root@k8s-master01 rbd]# **kubectl get po**

NAME READY STATUS RESTARTS AGE

nginx-6799fc88d8-m5gj5 1/1 Running 1 3d4h

wordpress-mysql-6965fc8cc8-6wt6j 1/1 Running 0 86m

[root@k8s-master01 rbd]# **kubectl exec -ti wordpress-mysql-6965fc8cc8-6wt6j -- bash**

root@wordpress-mysql-6965fc8cc8-6wt6j:/# **ls**

bin boot dev docker-entrypoint-initdb.d entrypoint.sh etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var

root@wordpress-mysql-6965fc8cc8-6wt6j:/# **cd /var/lib/mysql**

root@wordpress-mysql-6965fc8cc8-6wt6j:/var/lib/mysql# **ls**

auto.cnf ib\_logfile0 ib\_logfile1 ibdata1 lost+found mysql performance\_schema

root@wordpress-mysql-6965fc8cc8-6wt6j:/var/lib/mysql# **mkdir test\_snapshot**

root@wordpress-mysql-6965fc8cc8-6wt6j:/var/lib/mysql# **ls**

auto.cnf ib\_logfile0 ib\_logfile1 ibdata1 lost+found mysql performance\_schema test\_snapshot

root@wordpress-mysql-6965fc8cc8-6wt6j:/var/lib/mysql# **echo "test for snapshot" > test\_snapshot/1.txt**

root@wordpress-mysql-6965fc8cc8-6wt6j:/var/lib/mysql# **cat test\_snapshot/1.txt**

test for snapshot

修改snapshot.yaml文件的source pvc为创建的MySQL pvc：



创建快照及查看状态：

[root@k8s-master01 rbd]# kubectl create -f snapshot.yaml

volumesnapshot.snapshot.storage.k8s.io/rbd-pvc-snapshot created

[root@k8s-master01 rbd]# kubectl get volumesnapshotclass

NAME DRIVER DELETIONPOLICY AGE

csi-rbdplugin-snapclass rook-ceph.rbd.csi.ceph.com Delete 8m37s

[root@k8s-master01 rbd]# kubectl get volumesnapshot

NAME READYTOUSE SOURCEPVC SOURCESNAPSHOTCONTENT RESTORESIZE SNAPSHOTCLASS SNAPSHOTCONTENT CREATIONTIME AGE

rbd-pvc-snapshot true mysql-pv-claim 3Gi csi-rbdplugin-snapclass snapcontent-715c2841-d1ba-493a-9eb9-52178df3c2e6 <invalid> <invalid>

### 指定快照创建PVC

如果想要创建一个具有某个数据的PVC，可以从某个快照恢复：

[root@k8s-master01 rbd]# cat pvc-restore.yaml

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: rbd-pvc-restore

spec:

storageClassName: rook-ceph-block

dataSource:

name: rbd-pvc-snapshot

kind: VolumeSnapshot

apiGroup: snapshot.storage.k8s.io

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

[root@k8s-master01 rbd]# kubectl create -f pvc-restore.yaml

persistentvolumeclaim/rbd-pvc-restore created

注意：dataSource为快照名称，storageClassName为新建pvc的storageClass，storage的大小不能低于原pvc的大小，具体请查看视频。

[root@k8s-master01 rbd]# kubectl create -f pvc-restore.yaml

persistentvolumeclaim/rbd-pvc-restore created

[root@k8s-master01 rbd]# kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

mysql-pv-claim Bound pvc-fd8b9860-c0d4-4797-8e1d-1fab880bc9fc 3Gi RWO rook-ceph-block 95m

rbd-pvc-restore Bound pvc-d86a7535-2331-4fef-ae93-c570c8d3f9e7 3Gi RWO rook-ceph-block 2s

### 数据校验

创建一个容器，挂载该PVC，查看是否含有之前的文件：

[root@k8s-master01 rbd]# cat restore-check-snapshot-rbd.yaml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: check-snapshot-restore

spec:

selector:

matchLabels:

app: check

strategy:

type: Recreate

template:

metadata:

labels:

app: check

spec:

containers:

- image: alpine:3.8

name: check

command:

- sh

- -c

- sleep 36000

volumeMounts:

- name: check-mysql-persistent-storage

mountPath: /mnt

volumes:

- name: check-mysql-persistent-storage

persistentVolumeClaim:

claimName: rbd-pvc-restore

[root@k8s-master01 rbd]# kubectl create -f restore-check-snapshot-rbd.yaml

deployment.apps/check-snapshot-restore created

查看数据是否存在

[root@k8s-master01 rbd]# kubectl get po

NAME READY STATUS RESTARTS AGE

check-snapshot-restore-64b85c5f88-zvr62 1/1 Running 0 97s

nginx-6799fc88d8-m5gj5 1/1 Running 1 3d5h

wordpress-mysql-6965fc8cc8-6wt6j 1/1 Running 0 104m

You have new mail in /var/spool/mail/root

[root@k8s-master01 rbd]# kubectl exec -ti check-snapshot-restore-64b85c5f88-zvr62 -- ls

bin etc lib mnt root sbin sys usr

dev home media proc run srv tmp var

[root@k8s-master01 rbd]# kubectl exec -ti check-snapshot-restore-64b85c5f88-zvr62 -- ls /mnt

auto.cnf ibdata1 performance\_schema

ib\_logfile0 lost+found test\_snapshot

ib\_logfile1 mysql

[root@k8s-master01 rbd]# kubectl exec -ti check-snapshot-restore-64b85c5f88-zvr62 -- ls /mnt/test\_snapshot

1.txt

[root@k8s-master01 rbd]# kubectl exec -ti check-snapshot-restore-64b85c5f88-zvr62 -- cat /mnt/test\_snapshot/1.txt

test for snapshot

测试无误后清理数据（snapshotclass可以不删除，后期创建rbd快照直接用该snapshotclass即可）：

[root@k8s-master01 rbd]# kubectl delete -f restore-check-snapshot-rbd.yaml -f pvc-restore.yaml -f snapshot.yaml

deployment.apps "check-snapshot-restore" deleted

persistentvolumeclaim "rbd-pvc-restore" deleted

volumesnapshot.snapshot.storage.k8s.io "rbd-pvc-snapshot" deleted

## 文件共享类型快照

操作步骤和块存储类型无区别，可以参考：

<https://rook.io/docs/rook/v1.5/ceph-csi-snapshot.html#cephfs-snapshots>

# PVC克隆

[root@k8s-master01 rbd]# pwd

/root/rook/cluster/examples/kubernetes/ceph/csi/rbd

[root@k8s-master01 rbd]# cat pvc-clone.yaml

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: rbd-pvc-clone

spec:

storageClassName: rook-ceph-block

dataSource:

name: mysql-pv-claim

kind: PersistentVolumeClaim

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 3Gi

[root@k8s-master01 rbd]# kubectl create -f pvc-clone.yaml

persistentvolumeclaim/rbd-pvc-clone created

[root@k8s-master01 rbd]# kubectl get pvc

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

mysql-pv-claim Bound pvc-fd8b9860-c0d4-4797-8e1d-1fab880bc9fc 3Gi RWO rook-ceph-block 110m

rbd-pvc-clone Bound pvc-6dda14c9-9d98-41e6-9d92-9d4ea382c614 3Gi RWO rook-ceph-block 4s

需要注意的是pvc-clone.yaml的dataSource的name是被克隆的pvc名称，在此是mysql-pv-claim，storageClassName为新建pvc的storageClass名称，storage不能小于之前pvc的大小。

# 测试数据清理

[root@k8s-master01 ceph]# kubectl delete deploy kube-registry -n kube-system

deployment.apps "kube-registry" deleted

[root@k8s-master01 ceph]# kubectl get deploy

NAME READY UP-TO-DATE AVAILABLE AGE

nginx 1/1 1 1 3d10h

wordpress-mysql 1/1 1 1 114m

[root@k8s-master01 ceph]# kubectl delete deploy wordpress-mysql -n default

deployment.apps "wordpress-mysql" deleted

删除PVC之前需要把挂载了该PVC的资源先删掉，删除PVC之后PV也会被删除

[root@k8s-master01 ceph]# kubectl delete pvc mysql-pv-claim rbd-pvc-clone -n default

[root@k8s-master01 ceph]# kubectl delete pvc cephfs-pvc -n kube-system

persistentvolumeclaim "cephfs-pvc" deleted

[root@k8s-master01 ceph]# kubectl get pv

No resources found

此清理只是清理测试用的数据，没有清理之前创建的各类Class和ceph的存储池（因为之后还要在生产用），如果需要清理整个集群，请参考：<https://rook.io/docs/rook/v1.5/ceph-teardown.html>