1.1 PV 和 PVC 实验手册

1.1.1 PV 和 PVC

步骤 1　在开始本次实验前，需要已经配置好 nfs 服务，本次实验使用的 nfs 服务搭建在 master

```
[root@k8s-master storagefile]# showmount -e localhost
Export list for localhost:
/nfs/ *
```

步骤 2　在 nfs 文件夹下创建 pv1 目录

```
[root@k8s-master nfs]# mkdir pv1
[root@k8s-master nfs]# ls
pv1
```

步骤 3　编写 PV 的 yaml 文件

```
[root@k8s-master storagefile]# vim pv1.yaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mypv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  nfs:
    path: /nfs/pv1
    server: 127.0.0.1   #此处为 nfs 服务器 IP（即 master 节点 IP）
```

步骤 4　创建 PV

```
[root@k8s-master storagefile]# kubectl apply -f pv1.yaml
persistentvolume/mypv created
```

查看 PV

```
[root@k8s-master storagefile]# kubectl get pv
NAME    CAPACITY    ACCESS MODES    RECLAIM POLICY    STATUS       CLAIM        AGE
```

```
mypv    1Gi          RWO              Recycle          Available               101s
```

步骤 5 创建 PVC 的 yaml 文件，配置时指定 PV 的名称。

```
[root@k8s-master storagefile]# vim pvc1.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeName: mypv
  resources:
    requests:
      storage: 1Gi
```

步骤 6 创建 PVC

```
[root@k8s-master storagefile]# kubectl apply -f pvc1.yaml
```

```
persistentvolumeclaim/mypvc created
```

步骤 7 查看 PV 和 PVC 的状态，可以看到，PV 的状态从 Available 变为 Bound，而 PVC 的状态也是 Bound。

```
[root@k8s-master storagefile]# kubectl get pv
```

```
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
mypv   1Gi         RWO            Recycle           Bound    default/mypvc
```

```
 [root@k8s-master storagefile]# kubectl get pvc
```

```
NAME    STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mypvc   Bound    mypv     1Gi           RWO                          112s
```

步骤 8 创建 Pod，使用该 PVC

```
[root@k8s-master storagefile]# vim testpod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: pvctest
  name: pvcpod
spec:
  containers:
  - name: busybox
```

```
    args:
    - /bin/sh
    - -c
    - sleep 30000;
    image: busybox
    volumeMounts:
    - mountPath: /pvcdir
      name: pvc-volume
  volumes:
    - name: pvc-volume
      persistentVolumeClaim:
        claimName: mypvc
```

步骤 9 创建 Pod，并进入 pod 写入文件。

```
[root@k8s-master storagefile]# kubectl apply -f testpod.yaml
pod/pvcpod created
 [root@k8s-master storagefile]# kubectl exec -it pvcpod /bin/sh
/ # cd /pvcdir
/pvcdir # ls
/pvcdir # touch hello
/pvcdir # exit
```

步骤 10 在 nfs 目录查看是否存在 hello 文件

```
[root@k8s-master storagefile]# cd /nfs/pv1
[root@k8s-master pv1]# ls
hello
```

步骤 11 删除 pod 和 PVC，并等待系统自动回收完成。

```
[root@k8s-master pv1]# kubectl delete pod pvcpod
pod "pvcpod" deleted
```

```
 [root@k8s-master storagefile]# kubectl delete pvc mypvc && kubectl get pod -w
persistentvolumeclaim "mypvc" deleted
NAME                 READY   STATUS              RESTARTS   AGE
recycler-for-mypv    0/1     ContainerCreating   0          0s
recycler-for-mypv    0/1     Completed           0          0s
recycler-for-mypv    0/1     Terminating         0          0s
recycler-for-mypv    0/1     Terminating         0          0s
```

步骤 12 再次查看 PV 状态，再次变为 available 的可用状态

```
[root@k8s-master storagefile]# kubectl get pv
```

```
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mypv   1Gi        RWO            Recycle          Available                                   2m31s
```

步骤 13 查看 nfs 中是否存在 hello 文件，可以看到已经被清空

```
[root@k8s-master storagefile]# cd /nfs
[root@k8s-master nfs]# ls
pv1
[root@k8s-master nfs]# cd pv1
[root@k8s-master pv1]# ls
[root@k8s-master pv1]#
```

步骤 14 使用 PVC2.yaml 文件创建 PVC

```
[root@k8s-master storagefile]# vim pvc2.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc2
spec:
  accessModes:
    - ReadWriteOnce
  volumeName: mypv
  resources:
    requests:
      storage: 1Gi
```

```
[root@k8s-master storagefile]# kubectl apply -f pvc2.yaml
persistentvolumeclaim/mypvc2 created
```

步骤 15 可以看到 PV 的状态再次变为 bound 状态，并且绑定给了 mypvc2

```
[root@k8s-master storagefile]# kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM            AGE
mypv   1Gi        RWO            Recycle          Bound    default/mypvc2   12m
```

```
[root@k8s-master storagefile]# kubectl get pvc
NAME     STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mypvc2   Bound    mypv     1Gi        RWO                           37s
```

1.1.2 使用 StorageClass 方式关联 PV 和 PVC

步骤 1 在 nfs 目录中创建 PV2 目录

```
[root@k8s-master storagefile]# mkdir /nfs/pv2
```

步骤 2　创建 PV 的 yaml 文件

```
[root@k8s-master storagefile]# vim pv-sc.yaml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-sc
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  storageClassName: nfs
  nfs:
    path: /nfs/pv2
     server: 192.168.137.11
```

步骤 3　创建 pvc 的 yaml 文件

```
[root@k8s-master storagefile]# vim pvc-sc.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-sc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: nfs
  resources:
    requests:
      storage: 1Gi
```

步骤 4　创建 pv 和 pvc

```
[root@k8s-master storagefile]# kubectl apply -f pv-sc.yaml
```

```
persistentvolume/pv-sc created
```

```
[root@k8s-master storagefile]# kubectl apply -f pvc-sc.yaml
```

```
persistentvolumeclaim/pvc-sc created
```

步骤 5　查看 PV 和 PVC 的绑定

```
[root@k8s-master storagefile]# kubectl get pv
```

```
NAME    CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS    AGE
```

```
pv-sc    1Gi         RWO            Recycle          Bound     default/pvc-sc    nfs              33s

[root@k8s-master storagefile]# kubectl get pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc-sc    Bound    pv-sc    1Gi        RWO              nfs            28s
```