Using PersistentVolumes in Kubernetes

Your company, BeeBox, is developing some applications for Kubernetes. They are anticipating some increasingly complex storage needs in the future, so they want to make sure that they can leverage the full potential of Kubernetes storage with PersistentVolumes.

Your task is to build an application that uses a PersistentVolume for storage. Ensure that the application's volume is able to be expanded in case its storage needs might increase later.

So we need to create a PV that allows volume expansion

Create a PersistentVolume That Allows Claim Expansion

1. So sine we need to create a PV, we need to first create a storage class

vi localdisk.yml

Figure 1-1

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: localdisk
provisioner: kubernetes.io/no-provisioner
allowVolumeExpansion: true
~
~
```

Above is what is going to ensure that our PV supports volume expansion. Which is why we set the volume expansion to true.

2. Create the storage class (using kubectl)

So since that is created, we can now create the PV itself.

3. Create a YAML file for the PV

vi host-pv.yml

Figure 1-2

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
   name: host-pv
spec:
   storageClassName: localdisk
   persistentVolumeReclaimPolicy: Recycle
   capacity:
      storage: 1Gi
   accessModes:
      - ReadWriteOnce
   hostPath:
      path: /var/output
```

4. Create the PV (Using kubectl)

Now that the PV is created, lets take a look at the status of the PV

kubectl get pv

Figure 1-3

```
cloud_user@k8s-control:~$ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
host-pv   1Gi        RWO            Recycle          Available           localdisk               11s
```

We can see the capacity, status, and age of the PV. But it is not bound to anything. That is because we still need to create a PVC (PersistentVolumeClaim) that is capable of binding this volume.

Create a PersistentVolumeClaim

5. Since we created the pv.yml we will create pvc.yml

Figure 1-4

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
   name: host-pvc
spec:
   storageClassName: localdisk
   accessModes:
      - ReadWriteOnce
   resources:
      requests:
         storage: 100Mi
~
```

*The settings will correspond with the PV we created earlier. The resource request needs to be equal to or less than the available resources in the PV. Which the PV had 1GB so we set this one up to 100Mi. So the PV should be able to bind automatically to that PV

6. Create the pvc we just did in yml

So now that the volume and claim are both created, lets do the kubectl get pv again.

Figure 1-5

```
cloud_user@k8s-control:~$ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
host-pv   1Gi        RWO            Recycle          Bound    default/host-pvc   localdisk               14m
```

We now have a status from available > Bound. That is because it is bound to our PVC. Just to make sure type in kubectl get pvc just to make sure both pv and pvc are aligned.

All we need to do is create a pod that uses that PVC for storage.

Create a Pod That Uses a PersistentVolume for Storage

7. Create a file called pv-pod.yml

Figure 1-6

```
apiVersion: v1
kind: Pod
metadata:
   name: pv-pod
spec:
   containers:
      - name: busybox
        image: busybox
        command: ['sh', '-c', 'while true; do echo Success! > /output/success.txt; sleep 5; done']
~
```

Above is just going to write the “Success!” string to the output file we included.

So we need to mount our PV to the /output location so that our container here will be writing data to that PV

8. In the yml file add the necessary things to mount our PV to the /output location

Figure 1-7

```
apiVersion: v1
kind: Pod
metadata:
   name: pv-pod
spec:
   containers:
   - name: busybox
     image: busybox
     command: ['sh', '-c', 'while true; do echo Success! > /output/success.txt; sleep 5; done']
     volumeMounts:
     - name: pv-storage
       mountPath: /output
   volumes:
   - name: pv-storage
     persistentVolumeClaim:
      claimName: host-pvc
~
```

This should be the final things to include. You can see the different form the previous step to this one. So we added the volume and the volumeMount to the pod spec here.

9. Verify the pv-pod we created is running
10. Go to the worker node and see if you can see the output text we created by using the cat command

cat /var/output/success.txt

Figure 1-8

```
cloud_user@k8s-worker1:~$ cat /var/output/success.txt
Success!
cloud_user@k8s-worker1:~$
```

We have built at PV and built a simple pod that uses PV for storage and