## Exercise 8.4: Using a ResourceQuota to Limit PVC Count and Usage

The flexibility of cloud-based storage often requires limiting consumption among users. We will use the `ResourceQuota` object to both limit the total consumption as well as the number of persistent volume claims.

1. Begin by deleting the deployment we had created to use NFS, the pv and the pvc.

```
student@cp:~$ kubectl delete deploy nginx-nfs
deployment.apps "nginx-nfs" deleted
```

```
student@cp:~$ kubectl delete pvc pvc-one
persistentvolumeclaim "pvc-one" deleted
```

```
student@cp:~$ kubectl delete pv pvvol-1
persistentvolume "pvvol-1" deleted
```

2. Create a yaml file for the `ResourceQuota` object. Set the storage limit to ten claims with a total usage of 500Mi.

```
student@cp:~$ vim storage-quota.yaml
```

storage-quota.yaml

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storagequota
spec:
  hard:
    persistentvolumeclaims: "10"
    requests.storage: "500Mi"
```

3. Create a new namespace called `small`. View the namespace information prior to the new quota. Either the long name with double dashes `--namespace` or the nickname `ns` work for the resource.

```
student@cp:~$ kubectl create namespace small
namespace/small created
```

```
student@cp:~$ kubectl describe ns small
Name:           small
Labels:         <none>
Annotations:    <none>
Status:         Active

No resource quota.

No resource limits.
```

4. Create a new pv and pvc in the small namespace.

```
student@cp:~$ kubectl -n small create -f PVol.yaml
persistentvolume/pvvol-1 created
```

```
student@cp:~$ kubectl -n small create -f pvc.yaml
persistentvolumeclaim/pvc-one created
```

5. Create the new resource quota, placing this object into the `small` namespace.

```
student@cp:~$ kubectl -n small create -f storage-quota.yaml
resourcequota/storagequota created
```

6. Verify the `small` namespace has quotas. Compare the output to the same command above.

```
student@cp:~$ kubectl describe ns small
Name:         small
Labels:       <none>
Annotations:  <none>
Status:       Active

Resource Quotas
 Name:                   storagequota
 Resource                Used   Hard
 --------                ---    ---
 persistentvolumeclaims  1      10
 requests.storage        200Mi  500Mi

No resource limits.
```

7. Remove the namespace line from the `nfs-pod.yaml` file. Should be around line 11 or so. This will allow us to pass other namespaces on the command line.

```
student@cp:~$ vim nfs-pod.yaml
```

8. Create the container again.

```
student@cp:~$ kubectl -n small create -f nfs-pod.yaml
deployment.apps/nginx-nfs created
```

9. Determine if the deployment has a running pod.

```
student@cp:~$ kubectl -n small get deploy
NAME        READY  UP-TO-DATE  AVAILABLE   AGE
nginx-nfs   1/1    1           1           43s
```

```
student@cp:~$ kubectl -n small describe deploy nginx-nfs
<output_omitted>
```

10. Look to see if the pods are ready.

```
student@cp:~$ kubectl -n small get pod
NAME                         READY     STATUS    RESTARTS   AGE
nginx-nfs-2854978848-g3khf   1/1       Running   0          37s
```

11. Ensure the Pod is running and is using the NFS mounted volume. If you pass the namespace first `Tab` will auto-complete the pod name.

```
student@cp:~$ kubectl -n small describe pod \
          nginx-nfs-2854978848-g3khf
```

```
Name:           nginx-nfs-2854978848-g3khf
Namespace:      small
<output_omitted>

    Mounts:
      /opt from nfs-vol (rw)
<output_omitted>
```

12. View the quota usage of the namespace

```
student@cp:~$ kubectl describe ns small
```

```
<output_omitted>

Resource Quotas
 Name:                   storagequota
 Resource                Used   Hard
 --------                ---    ---
 persistentvolumeclaims  1      10
 requests.storage        200Mi  500Mi

No resource limits.
```

13. Create a 300M file inside of the `/opt/sfw` directory on the host and view the quota usage again. Note that with NFS the size of the share is not counted against the deployment.

```
student@cp:~$ sudo dd if=/dev/zero of=/opt/sfw/bigfile bs=1M count=300
```

```
300+0 records in
300+0 records out
314572800 bytes (315 MB, 300 MiB) copied, 0.196794 s, 1.6 GB/s
```

```
student@cp:~$ kubectl describe ns small
```

```
<output_omitted>
Resource Quotas
 Name:                        storagequota
 Resource                Used        Hard
 --------                ---         ---
 persistentvolumeclaims       1         10
 requests.storage        200Mi        500Mi
<output_omitted>
```

```
student@cp:~$ du -h /opt/
```

```
301M        /opt/sfw
41M        /opt/cni/bin
41M        /opt/cni
341M        /opt/
```

14. Now let us illustrate what happens when a deployment requests more than the quota. Begin by shutting down the existing deployment.

```
student@cp:~$ kubectl -n small get deploy
```

```
NAME        READY   UP-TO-DATE  AVAILABLE   AGE
nginx-nfs   1       1           1           11m
```

```
student@cp:~$ kubectl -n small delete deploy nginx-nfs
```

```
deployment.apps "nginx-nfs" deleted
```

15. Once the Pod has shut down view the resource usage of the namespace again. Note the storage did not get cleaned up when the pod was shut down.

```
student@cp:~$ kubectl describe ns small
```

```
<output_omitted>
Resource Quotas
 Name:                   storagequota
 Resource                Used    Hard
 --------                ---     ---
 persistentvolumeclaims  1       10
 requests.storage        200Mi   500Mi
```

16. Remove the pvc then view the pv it was using. Note the `RECLAIM POLICY` and `STATUS`.

```
student@cp:~$ kubectl -n small get pvc
```

```
NAME      STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
pvc-one   Bound     pvvol-1   1Gi        RWX                          19m
```

```
student@cp:~$ kubectl -n small delete pvc pvc-one
```

```
persistentvolumeclaim "pvc-one" deleted
```

```
student@cp:~$ kubectl -n small get pv
```

```
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS     CLAIM
STORAGECLASS   REASON    AGE
pvvol-1   1Gi        RWX           Retain          Released   small/pvc-one   44m
```

17. Dynamically provisioned storage uses the `ReclaimPolicy` of the `StorageClass` which could be `Delete`, `Retain`, or some types allow `Recycle`. Manually created persistent volumes default to `Retain` unless set otherwise at creation. The default storage policy is to retain the storage to allow recovery of any data. To change this begin by viewing the yaml output.

```
student@cp:~$ kubectl get pv/pvvol-1 -o yaml
```

```
....
    path: /opt/sfw
    server: k8scp
  persistentVolumeReclaimPolicy: Retain
status:
  phase: Released
```

18. Currently we will need to delete and re-create the object. Future development on a deleter plugin is planned. We will re-create the volume and allow it to use the `Retain` policy, then change it once running.

```
student@cp:~$ kubectl delete pv/pvvol-1
```

```
persistentvolume "pvvol-1" deleted
```

```
student@cp:~$ grep Retain PVol.yaml
```

```
  persistentVolumeReclaimPolicy: Retain
```

```
student@cp:~$ kubectl create -f PVol.yaml
```

```
persistentvolume "pvvol-1" created
```

19. We will use `kubectl patch` to change the retention policy to `Delete`. The yaml output from before can be helpful in getting the correct syntax.

```
student@cp:~$  kubectl patch pv pvvol-1 -p \
'{"spec":{"persistentVolumeReclaimPolicy":"Delete"}}'
```

```
persistentvolume/pvvol-1 patched
```

```
student@cp:~$ kubectl get pv/pvvol-1
```

```
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS      CLAIM
 STORAGECLASS   REASON    AGE
pvvol-1   1Gi        RWX           Delete          Available   2m
```

20. View the current quota settings.

```
student@cp:~$ kubectl describe ns small
```

```
....
 requests.storage        0        500Mi
```

21. Create the pvc again. Even with no pods running, note the resource usage.

```
student@cp:~$ kubectl -n small create -f pvc.yaml
```

```
persistentvolumeclaim/pvc-one created
```

```
student@cp:~$ kubectl describe ns small
```

```
....
 requests.storage        200Mi        500Mi
```

22. Remove the existing quota from the namespace.

```
student@cp:~$ kubectl -n small get resourcequota
```

```
NAME           CREATED AT
storagequota   2019-11-25T04:10:02Z
```

```
student@cp:~$ kubectl -n small delete resourcequota storagequota
```

```
resourcequota "storagequota" deleted
```

23. Edit the storagequota.yaml file and lower the capacity to 100Mi.

```
student@cp:~$ vim storage-quota.yaml
```

```
....
    requests.storage: "100Mi"
```

24. Create and verify the new storage quota. Note the hard limit has already been exceeded.

```
student@cp:~$ kubectl -n small create -f storage-quota.yaml
resourcequota/storagequota created
```

```
student@cp:~$ kubectl describe ns small
....
 persistentvolumeclaims        1         10
 requests.storage        200Mi        100Mi

No resource limits.
```

25. Create the deployment again. View the deployment. Note there are no errors seen.

```
student@cp:~$ kubectl -n small create -f nfs-pod.yaml
deployment.apps/nginx-nfs created
```

```
student@cp:~$ kubectl -n small describe deploy/nginx-nfs
Name:                   nginx-nfs
Namespace:              small
<output_omitted>
```

26. Examine the pods to see if they are actually running.

```
student@cp:~$ kubectl -n small get po
NAME                         READY     STATUS    RESTARTS   AGE
nginx-nfs-2854978848-vb6bh   1/1       Running   0          58s
```

27. As we were able to deploy more pods even with apparent hard quota set, let us test to see if the reclaim of storage takes place. Remove the deployment and the persistent volume claim.

```
student@cp:~$ kubectl -n small delete deploy nginx-nfs
deployment.apps "nginx-nfs" deleted
```

```
student@cp:~$ kubectl -n small delete pvc/pvc-one
persistentvolumeclaim "pvc-one" deleted
```

28. View if the persistent volume exists. You will see it attempted a removal, but failed. If you look closer you will find the error has to do with the lack of a `deleter volume plugin` for NFS. Other storage protocols have a plugin.

```
student@cp:~$ kubectl -n small get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS     CLAIM
     STORAGECLASS   REASON    AGE
pvvol-1   1Gi        RWX           Delete          Failed     small/pvc-one   20m
```

29. Ensure the deployment, pvc and pv are all removed.

```
student@cp:~$ kubectl delete pv/pvvol-1
persistentvolume "pvvol-1" deleted
```

30. Edit the persistent volume YAML file and change the `persistentVolumeReclaimPolicy`: to Recycle.

```
student@cp:~$ vim PVol.yaml
```

PVol.yaml

```
....
  persistentVolumeReclaimPolicy: Recycle
....
```

31. Add a `LimitRange` to the namespace and attempt to create the persistent volume and persistent volume claim again. We can use the `LimitRange` we used earlier.

```
student@cp:~$ kubectl -n small create -f low-resource-range.yaml
limitrange/low-resource-range created
```

32. View the settings for the namespace. Both quotas and resource limits should be seen.

```
student@cp:~$ kubectl describe ns small
<output_omitted>
Resource Limits
 Type       Resource  Min  Max  Default Request  Default Limit  ...
 ----       --------  ---  ---  ---------------  -------------  -...
 Container  cpu       -    -    500m             1              -
 Container  memory    -    -    100Mi            500Mi          -
```

33. Create the persistent volume again. View the resource. Note the `Reclaim Policy` is `Recycle`.

```
student@cp:~$ kubectl -n small create -f PVol.yaml
persistentvolume/pvvol-1 created
```

```
student@cp:~$ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      ...
pvvol-1   1Gi        RWX            Recycle          Available   ...
```

34. Attempt to create the persistent volume claim again. The quota only takes effect if there is also a resource limit in effect.

```
student@cp:~$ kubectl -n small create -f pvc.yaml
Error from server (Forbidden): error when creating "pvc.yaml":
 persistentvolumeclaims "pvc-one" is forbidden: exceeded quota:
storagequota, requested: requests.storage=200Mi, used:
requests.storage=0, limited: requests.storage=100Mi
```

35. Edit the `resourcequota` to increase the `requests.storage` to 500mi.

```
student@cp:~$ kubectl -n small edit resourcequota
```

```
....
spec:
  hard:
    persistentvolumeclaims: "10"
    requests.storage: 500Mi
status:
  hard:
    persistentvolumeclaims: "10"
....
```

36. Create the pvc again. It should work this time. Then create the deployment again.

```
student@cp:~$ kubectl -n small create -f pvc.yaml
persistentvolumeclaim/pvc-one created

student@cp:~$ kubectl -n small create -f nfs-pod.yaml
deployment.apps/nginx-nfs created
```

37. View the namespace settings.

```
student@cp:~$ kubectl describe ns small
<output_omitted>
```

38. Delete the deployment. View the status of the pv and pvc.

```
student@cp:~$ kubectl -n small delete deploy nginx-nfs
deployment.apps "nginx-nfs" deleted

student@cp:~$ kubectl -n small get pvc
NAME      STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc-one   Bound     pvvol-1   1Gi        RWX                           7m

student@cp:~$ kubectl -n small get pv
NAME     CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM          ...
pvvol-1  1Gi       RWX           Recycle         Bound   small/pvc-one  ...
```

39. Delete the pvc and check the status of the pv. It should show as Available.

```
student@cp:~$ kubectl -n small delete pvc pvc-one
persistentvolumeclaim "pvc-one" deleted

student@cp:~$ kubectl -n small get pv
NAME    CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORA...
pvvol-1 1Gi      RWX          Recycle        Available         ...
```

40. Remove the pv and any other resources created during this lab.

```
student@cp:~$ kubectl delete pv pvvol-1
persistentvolume "pvvol-1" deleted
```