Chapter 9

Volumes and Data

9.1 Labs

Exercise 9.1: Create a ConfigMap

Overview

Container files are ephemeral, which can be problematic for some applications. Should a container be restarted the files will be lost. In addition, we need a method to share files between containers inside a Pod.

A `Volume` is a directory accessible to containers in a Pod. Cloud providers offer volumes which persist further than the life of the Pod, such that AWS or GCE volumes could be pre-populated and offered to Pods, or transferred from one Pod to another. **Ceph** is also another popular solution for dynamic, persistent volumes.

Unlike current **Docker** volumes a Kubernetes volume has the lifetime of the Pod, not the containers within. You can also use different types of volumes in the same Pod simultaneously, but Volumes cannot mount in a nested fashion. Each must have their own mount point. Volumes are declared with `spec.volumes` and mount points with `spec.containers.volumeMounts` parameters. Each particular volume type, 24 currently, may have other restrictions. https://kubernetes.io/docs/concepts/storage/volumes/#types-of-volumes

We will also work with a `ConfigMap`, which is basically a set of key-value pairs. This data can be made available so that a Pod can read the data as environment variables or configuration data. A `ConfigMap` is similar to a `Secret`, except they are not base64 byte encoded arrays. They are stored as strings and can be read in serialized form.

Create a ConfigMap

There are three different ways a `ConfigMap` can ingest data, from a literal value, from a file or from a directory of files.

1. We will create a `ConfigMap` containing primary colors. We will create a series of files to ingest into the `ConfigMap`. First, we create a directory `primary` and populate it with four files. Then we create a file in our home directory with our favorite color.

```
student@lfs458-node-1a0a:~$ mkdir primary

student@lfs458-node-1a0a:~$ echo c > primary/cyan

student@lfs458-node-1a0a:~$ echo m > primary/magenta

student@lfs458-node-1a0a:~$ echo y > primary/yellow

student@lfs458-node-1a0a:~$ echo k > primary/black

student@lfs458-node-1a0a:~$ echo "known as key" >> primary/black

student@lfs458-node-1a0a:~$ echo blue > favorite
```

2. Now we will create the `ConfigMap` and populate it with the files we created as well as a literal value from the command line.

```
student@lfs458-node-1a0a:~$ kubectl create configmap colors \
     --from-literal=text=black  \
     --from-file=./favorite  \
     --from-file=./primary/
configmap "colors" created
```

3. View how the data is organized inside the cluster.

```
student@lfs458-node-1a0a:~$ kubectl get configmap colors
NAME      DATA      AGE
colors    6         30s

student@lfs458-node-1a0a:~$ kubectl get configmap colors -o yaml
apiVersion: v1
data:
  black: |
    k
    known as key
  cyan: |
    c
  favorite: |
    blue
  magenta: |
    m
  text: black
  yellow: |
    y
kind: ConfigMap
<output_omitted>
```

4. Now we can create a Pod to use the `ConfigMap`. In this case a particular parameter is being defined as an environment variable.

```
student@lfs458-node-1a0a:~$ vim simpleshell.yaml


apiVersion: v1
kind: Pod
metadata:
  name: shell-demo
spec:
  containers:
  - name: nginx
    image: nginx
    env:
```



```
    - name: ilike
      valueFrom:
        configMapKeyRef:
          name: colors
          key: favorite
```

5. Create the Pod and view the environmental variable. After you view the parameter, exit out and delete the pod.

```
student@lfs458-node-1a0a:~$ kubectl create -f simpleshell.yaml
pod "shell-demo" created

student@lfs458-node-1a0a:~$ kubectl exec -it shell-demo \
        -- /bin/bash -c 'echo $ilike'
blue

student@lfs458-node-1a0a:~$ kubectl delete pod shell-demo
pod "shell-demo" deleted
```

6. All variables from a file can be included as environment variables as well. Comment out the previous `env:` stanza and add a slightly different `envFrom` to the file. Having new and old code at the same time can be helpful to see and understand the differences. Recreate the Pod, check all variables and delete the pod again. They can be found spread throughout the environment variable output.

```
student@lfs458-node-1a0a:~$ vim simpleshell.yaml
<output_omitted>
    image: nginx
#    env:
#    - name: ilike
#      valueFrom:
#        configMapKeyRef:
#          name: colors
#          key: favorite
    envFrom:
    - configMapRef:
        name: colors

student@lfs458-node-1a0a:~$ kubectl create -f simpleshell.yaml

student@lfs458-node-1a0a:~$ kubectl exec -it shell-demo \
           -- /bin/bash -c 'env'
HOSTNAME=shell-demo
NJS_VERSION=1.13.6.0.1.14-1~stretch
NGINX_VERSION=1.13.6-1~stretch
black=k
know as key

favorite=blue
<output_omitted>

student@lfs458-node-1a0a:~$ kubectl delete pod shell-demo
pod "shell-demo" deleted
```

7. A `ConfigMap` can also be created from a YAML file. Create one with a few parameters to describe a car.

```
student@lfs458-node-1a0a:~$ vim car-map.yaml


apiVersion: v1
kind: ConfigMap
metadata:
  name: fast-car
  namespace: default
```



```
data:
  car.make: Ford
  car.model: Mustang
  car.trim: Shelby
```

8. Create the ConfigMap and verify the settings.

```
student@lfs458-node-1a0a:~$ kubectl create -f car-map.yaml
configmap "fast-car" created

student@lfs458-node-1a0a:~$ kubectl get configmap fast-car -o yaml
apiVersion: v1
data:
  car.make: Ford
  car.model: Mustang
  car.trim: Shelby
kind: ConfigMap
<output_omitted>
```

9. We will now make the ConfigMap available to a Pod as a mounted volume. You can again comment out the previous environmental settings and add the following new stanza. The containers: and volumes: entries are indented the same number of spaces.

```
student@lfs458-node-1a0a:~$ vim simpleshell.yaml
<output_omitted>
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
      - name: car-vol
        mountPath: /etc/cars
  volumes:
    - name: car-vol
      configMap:
        name: fast-car
```

10. Create the Pod again. Verify the volume exists and the contents of a file within. Due to the lack of a carriage return in the file your next prompt may be on the same line as the output, Shelby.

```
student@lfs458-node-1a0a:~$ kubectl create -f simpleshell.yaml
pod "shell-demo" created

student@lfs458-node-1a0a:~$  kubectl exec -it shell-demo -- \
      /bin/bash -c 'df -h'
Filesystem      Size  Used Avail Use% Mounted on
none             20G  1.8G   18G  10% /
tmpfs           3.9G     0  3.9G   0% /dev
tmpfs           3.9G     0  3.9G   0% /sys/fs/cgroup
/dev/xvda1       20G  1.8G   18G  10% /etc/cars
<output_omitted>

student@lfs458-node-1a0a:~$ kubectl exec -it shell-demo -- \
    /bin/bash -c 'cat /etc/cars/car.trim'
Shelby
```

11. Delete the Pod and ConfigMaps we were using.

```
student@lfs458-node-1a0a:~$ kubectl delete pods shell-demo
pod "shell-demo" deleted

student@lfs458-node-1a0a:~$ kubectl delete configmap fast-car colors
```



```
configmap "fast-car" deleted
configmap "colors" deleted
```

Exercise 9.2: Creating a Persistent NFS Volume (PV)

We will first deploy an NFS server. Once tested we will create a persistent NFS volume for containers to claim.

1. Install the software on your master node.

```
student@lfs458-node-1a0a:~$ sudo apt-get update && sudo \
            apt-get install -y nfs-kernel-server
<output_omitted>
```

2. Make and populate a directory to be shared. Also give it similar permissions to `/tmp/`

```
student@lfs458-node-1a0a:~$ sudo mkdir /opt/sfw

student@lfs458-node-1a0a:~$ sudo chmod 1777 /opt/sfw/

student@lfs458-node-1a0a:~$ sudo bash -c \
              'echo software > /opt/sfw/hello.txt'
```

3. Edit the NFS server file to share out the newly created directory. In this case we will share the directory with all. You can always **snoop** to see the inbound request in a later step and update the file to be more narrow.

```
student@lfs458-node-1a0a:~$ sudo vim /etc/exports
/opt/sfw/ *(rw,sync,no_root_squash,subtree_check)
```

4. Cause `/etc/exports` to be re-read:

```
student@lfs458-node-1a0a:~$ sudo exportfs -ra
```

5. Test by mounting the resource from your **second** node.

```
student@lfs458-node-2b2b:~$ sudo apt-get -y install nfs-common
<output_omitted>

student@lfs458-node-2b2b:~$ showmount -e lfs458-node-1a0a
Export list for lfs458-node-1a0a:
/opt/sfw  *

student@lfs458-node-2b2b:~$ sudo mount 10.128.0.3:/opt/sfw /mnt

student@lfs458-node-2b2b:~$ ls -l /mnt
total 4
-rw-r--r-- 1 root root 9 Sep 28 17:55 hello.txt
```

6. Return to the master node and create a YAML file for the object with `kind`, `PersistentVolume`. Use the hostname of the master server and the directory you created in the previous step. Only syntax is checked, an incorrect name or directory will not generate an error, but a Pod using the resource will not start. Note that the `accessModes` do not currently affect actual access and are typically used as labels instead.

```
student@lfs458-node-1a0a:~$ vim PVol.yaml


apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvvol-1
spec:
  capacity:
```



```
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  nfs:
    path: /opt/sfw
    server: lfs458-node-1a0a   #<-- Edit to match master node
    readOnly: false
```

7. Create the persistent volume, then verify its creation.

```
student@lfs458-node-1a0a:~$ kubectl create -f PVol.yaml
persistentvolume "pvvol-1" created

student@lfs458-node-1a0a:~$ kubectl get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS
   CLAIM     STORAGECLASS   REASON    AGE
pvvol-1   1Gi        RWX           Retain      Available   4s
```

Exercise 9.3: Creating a Persistent Volume Claim (PVC)

Before Pods can take advantage of the new PV we need to create a **Persistent Volume Claim** (**PVC**).

1. Begin by determining if any currently exist.

```
student@lfs458-node-1a0a:~$ kubectl get pvc
No resources found.
```

2. Create a YAML file for the new pvc.

```
student@lfs458-node-1a0a:~$ vim pvc.yaml


apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-one
spec:
  accessModes:
  - ReadWriteMany
  resources:
     requests:
       storage: 200Mi
```

3. Create and verify the new pvc is bound. Note that the size is 1Gi, even though 200Mi was suggested. Only a volume of at least that size could be used.

```
student@lfs458-node-1a0a:~$ kubectl create -f pvc.yaml
persistentvolumeclaim "pvc-one" created

student@lfs458-node-1a0a:~$ kubectl get pvc
NAME      STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
pvc-one   Bound     pvvol-1   1Gi        RWX                          4s
```

4. Look at the status of the pv again, to determine if it is in use. It should show a status of Bound.

```
student@lfs458-node-1a0a:~$ kubectl get pv
NAME      CAPACITY   ACCESSMODES   RECLAIMPOLICY   STATUS    CLAIM             STORAGECLASS   REASON    AGE
pvvol-1   1Gi        RWX           Retain          Bound     default/pvc-one                            5m
```


5. Create a new deployment to use the pvc. We will copy and edit an existing deployment yaml file. We will change the deployment name then add a volumeMounts section under containers and volumes section to the general spec. The name used must match in both places, whatever name you use. The claimName must match an existing pvc. As shown in the following example.

```
student@lfs458-node-1a0a:~$ cp first.yaml nfs-pod.yaml

student@lfs458-node-1a0a:~$ vim nfs-pod.yaml


apiVersion: apps/v1beta1
kind: Deployment
metadata:
  annotations:
    deployment.kubernetes.io/revision: "1"
  generation: 1
  labels:
    run: nginx
  name: nginx-nfs
  namespace: default
  resourceVersion: "1411"
spec:
  replicas: 1
  selector:
    matchLabels:
      run: nginx
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        run: nginx
    spec:
      containers:
      - image: nginx
        imagePullPolicy: Always
        name: nginx
        volumeMounts:
        - name: nfs-vol
          mountPath: /opt
        ports:
        - containerPort: 80
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termination-log
        terminationMessagePolicy: File
      volumes:                          #<<-- These four lines
      - name: nfs-vol
        persistentVolumeClaim:
          claimName: pvc-one
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```

6. Create the pod using the newly edited file.

```
student@lfs458-node-1a0a:~$ kubectl create -f nfs-pod.yaml
```


7. Look at the details of the pod.

```
student@lfs458-node-1a0a:~$ kubectl get pods
NAME                          READY     STATUS    RESTARTS   AGE
nginx-nfs-1054709768-s8g28    1/1       Running   0          3m


student@lfs458-node-1a0a:~$ kubectl describe pod nginx-nfs-1054709768-s8g28
Name:               nginx-nfs-1054709768-s8g28
Namespace:        default
Node:               lfs458-node-2b2b/10.128.0.5

<output_omitted>

    Mounts:
      /opt from nfs-vol (rw)

<output_omitted>

Volumes:
  nfs-vol:
    Type:        PersistentVolumeClaim (a reference to a PersistentV...
    ClaimName:        pvc-one
    ReadOnly:        false
<output_omitted>
```

8. View the status of the `PVC`. It should show as bound.

```
student@lfs458-node-1a0a:~$ kubectl get pvc
NAME     STATUS VOLUME  CAPACITY ACCESS MODES  STORAGECLASS  AGE
pvc-one  Bound  pvvol-1 1Gi      RWX                          2m
```

Exercise 9.4: 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@lfs458-node-1a0a:~$ kubectl delete deploy nginx-nfs
deployment "nginx-nfs" deleted

student@lfs458-node-1a0a:~$ kubectl delete pvc pvc-one
persistentvolumeclaim "pvc-one" deleted

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ vim 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@lfs458-node-1a0a:~$ kubectl create --namespace small
namespace "small" created

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl create -f PVol.yaml -n small
persistentvolume "pvvol-1" created

student@lfs458-node-1a0a:~$ kubectl create -f pvc.yaml -n small
persistentvolumeclaim "pvc-one" created
```

5. Create the new resource quota, placing this object into the `low-usage-limit` namespace.

```
student@lfs458-node-1a0a:~$ kubectl create -f storage-quota.yaml \
    -n small
resourcequota "storagequota" created
```

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

```
student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ vim nfs-pod.yaml
```

8. Create the container again.

```
student@lfs458-node-1a0a:~$ kubectl create -f nfs-pod.yaml \
        -n small
deployment "nginx-nfs" created
```

9. Determine if the deployment has a running pod.


```
student@lfs458-node-1a0a:~$ kubectl get deploy --namespace=small
NAME        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-nfs   1         1         1            0           43s

student@lfs458-node-1a0a:~$ kubectl describe deploy nginx-nfs \
        -n small
<output_omitted>
```

10. Look to see if the pods are ready.

```
student@lfs458-node-1a0a:~$ kubectl get po --namespace=small
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.

```
student@lfs458-node-1a0a:~$ kubectl describe po \
          nginx-nfs-2854978848-g3khf -n small
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@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl describe ns small
<output_omitted>
Resource Quotas
 Name:                     storagequota
 Resource                  Used        Hard
 --------                  ---         ---
 persistentvolumeclaims       1        10
 requests.storage          200Mi       500Mi
<output_omitted>

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl get deploy -n small
NAME        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-nfs   1         1         1            1           11m

student@lfs458-node-1a0a:~$ kubectl delete deploy nginx-nfs -n small
deployment "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@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl get pvc -n small
NAME      STATUS    VOLUME    CAPACITY   ACCESSMODES   STORAGECLASS   AGE
pvc-one   Bound     pvvol-1   1Gi        RWX                          19m

student@lfs458-node-1a0a:~$ kubectl delete pvc pvc-one -n small
persistentvolumeclaim "pvc-one" deleted

student@lfs458-node-1a0a:~$ kubectl get pv -n small
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@lfs458-node-1a0a:~$ kubectl get pv/pvvol-1 -o yaml
....
   path: /opt/sfw
    server: lfs458-node-1a0a
  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@lfs458-node-1a0a:~$ kubectl delete pv/pvvol-1
persistentvolume "pvvol-1" deleted

student@lfs458-node-1a0a:~$ grep Retain PVol.yaml
  persistentVolumeReclaimPolicy: Retain

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$  kubectl patch pv pvvol-1 -p \
'{"spec":{"persistentVolumeReclaimPolicy":"Delete"}}'
persistentvolume "pvvol-1" patched

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl describe ns small
.
 requests.storage        0        500Mi
```

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

```
student@lfs458-node-1a0a:~$ kubectl create -f pvc.yaml -n small
persistentvolumeclaim "pvc-one" created

student@lfs458-node-1a0a:~$ kubectl describe ns small
.
 requests.storage        200Mi        500Mi
```

22. Remove the existing quota from the namespace.

```
student@lfs458-node-1a0a:~$ kubectl get resourcequota -n small
NAME           AGE
storagequota   12m

student@lfs458-node-1a0a:~$ kubectl delete resourcequota \
    storagequota -n small
resourcequota "storagequota" deleted
```

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

```
student@lfs458-node-1a0a:~$ vim storage-quota.yaml

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

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

```
student@lfs458-node-1a0a:~$ kubectl create -f storage-quota.yaml -n small
resourcequota "storagequota" created

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl create -f nfs-pod.yaml \
      --namespace=small
deployment "nginx-nfs" created

student@lfs458-node-1a0a:~$ kubectl describe deploy/nginx-nfs -n small
Name:                       nginx-nfs
Namespace:                  small
<output_omitted>
```

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

```
student@lfs458-node-1a0a:~$ kubectl get po -n small
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@lfs458-node-1a0a:~$ kubectl delete deploy nginx-nfs -n small
deployment "nginx-nfs" deleted

student@lfs458-node-1a0a:~$ kubectl delete pvc/pvc-one -n small
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@lfs458-node-1a0a:~$ kubectl get pv -n small
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@lfs458-node-1a0a:~$ kubectl delete pv/pvvol-1
persistentvolume "pvvol-1" deleted
```

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

```
student@lfs458-node-1a0a:~$ vim 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@lfs458-node-1a0a:~$ kubectl create -f low-resource-range.yaml\
       -n small
limitrange "low-resource-range" created
```

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

```
student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl create -f PVol.yaml -n small
persistentvolume "pvvol-1" created

student@lfs458-node-1a0a:~$ 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@lfs458-node-1a0a:~$ kubectl create -f pvc.yaml -n small
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@lfs458-node-1a0a:~$ kubectl edit resourcequota -n small
....
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@lfs458-node-1a0a:~$ kubectl create -f pvc.yaml -n small
persistentvolumeclaim "pvc-one" created

student@lfs458-node-1a0a:~$ kubectl create -f nfs-pod.yaml -n small
deployment "nginx-nfs" created
```

37. View the namespace settings.

```
student@lfs458-node-1a0a:~$ kubectl describe ns small
<output_omitted>
```

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

```
student@lfs458-node-1a0a:~$ kubectl delete deploy nginx-nfs -n small
deployment "nginx-nfs" deleted

student@lfs458-node-1a0a:~$ kubectl get pvc -n small
NAME      STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
pvc-one   Bound     pvvol-1   1Gi        RWX                           7m

student@lfs458-node-1a0a:~$ kubectl get pv -n small
NAME    CAPACITY ACCESS MODES RECLAIM POLICY STATUS  CLAIM  STORA...
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@lfs458-node-1a0a:~$ kubectl delete pvc pvc-one -n small
persistentvolumeclaim "pvc-one" deleted

student@lfs458-node-1a0a:~$ kubectl get pv -n small
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@lfs458-node-1a0a:~$ kubectl delete pv pvvol-1
persistentvolume "pvvol-1" deleted
```

