

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@cp:~$ kubectl get pvc
```

```
No resources found in default namespace.
```

2. Create a YAML file for the new pvc.

```
student@cp:~$ cp /home/student/LFS258/SOLUTIONS/s_09/pvc.yaml .
```

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

YAML

pvc.yaml

```
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4   name: pvc-one
5 spec:
6   accessModes:
7     - ReadWriteMany
8   resources:
9     requests:
10      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@cp:~$ kubectl create -f pvc.yaml
```

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

```
student@cp:~$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
pvc-one	Bound	pvvol-1	1Gi	RWX		<unset>	7s

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

```
student@cp:~$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS
↪ VOLUMEATTRIBUTESCLASS		REASON	AGE			
pvvol-1	1Gi	RWX	Retain	Bound	default/pvc-one	
↪ <unset>			3m45s			

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 a 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. The volumes line is the same indent as containers and dnsPolicy.

```
student@cp:~$ cp /home/student/LFS258/SOLUTIONS/s_09/nfs-pod.yaml .
```

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

YAML

nfs-pod.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    annotations:
5      deployment.kubernetes.io/revision: "1"
6    generation: 1
7    labels:
8      run: nginx
9    name: nginx-nfs          #<-- Edit name
10   namespace: default
11  spec:
12   replicas: 1
13   selector:
14     matchLabels:
15       run: nginx
16   strategy:
17     rollingUpdate:
18       maxSurge: 1
19       maxUnavailable: 1
20     type: RollingUpdate
21   template:
22     metadata:
23       creationTimestamp: null
24     labels:
25       run: nginx
26     spec:
27       containers:
28       - image: nginx
29         imagePullPolicy: Always
30         name: nginx
31         volumeMounts:      #<-- Add these three lines
32         - name: nfs-vol
33           mountPath: /opt
34         ports:
35         - containerPort: 80
36           protocol: TCP
37         resources: {}
38         terminationMessagePath: /dev/termination-log
39         terminationMessagePolicy: File
40       volumes:              #<-- Add these four lines
41       - name: nfs-vol
42         persistentVolumeClaim:
43           claimName: pvc-one
44       dnsPolicy: ClusterFirst
45       restartPolicy: Always
46       schedulerName: default-scheduler
47       securityContext: {}
48       terminationGracePeriodSeconds: 30
```

6. Create the pod using the newly edited file.

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

```
deployment.apps/nginx-nfs created
```

7. Look at the details of the pod. You may see the daemonset pods running as well.

```
student@cp:~$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-nfs-1054709768-s8g28	1/1	Running	0	3m

```
student@cp:~$ kubectl describe pod nginx-nfs-1054709768-s8g28
```

```
Name:          nginx-nfs-1054709768-s8g28
Namespace:     default
Priority:       0
Node:          worker/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>
```