Exercise 9.5: Using StorageClass to Dynamically provision a volume

`StorageClasses` in Kubernetes simplify and automate the process of provisioning and managing storage resources, provide users with the flexibility to choose appropriate storage types for their workloads, and help administrators enforce policies and manage storage infrastructure more effectively. `StorageClasses` enables dynamic provisioning of storage resources. Without `StorageClasses`, administrators have to manually create PersistentVolumes (PVs) for each PersistentVolumeClaim (PVC) made by users. With `StorageClasses`, this process is automated. When a user creates a PVC and specifies a `StorageClasses`, the system automatically creates a corresponding PV that meets the requirements.

1. Begin by listing to see if we have any storage class available on our cluster.

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

```
No resources found
```

2. We dont have any StorageClass created. Before we can create the sc, we need to deploy the provisioner. Kubernetes doesn't include an internal NFS provisioner. We need to use an external provisioner to create a StorageClass for NFS. Let us deploy a nfs provisioner.

```
student@cp:~$ helm repo add nfs-subdir-external-provisioner \
              https://kubernetes-sigs.github.io/nfs-subdir-external-provisioner/
```

```
"nfs-subdir-external-provisioner" has been added to your repositories
```

```
student@cp:~$ helm install nfs-subdir-external-provisioner \
            nfs-subdir-external-provisioner/nfs-subdir-external-provisioner \
            --set nfs.server=cp \
            --set nfs.path=/opt/sfw/
```

```
NAME: nfs-subdir-external-provisioner
LAST DEPLOYED: Mon Jan  8 12:11:39 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

3. The installation also created a StorageClass for us.

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

```
NAME         PROVISIONER                                     RECLAIMPOLICY   VOLUMEBINDINGMODE
↪   ALLOWVOLUMEEXPANSION   AGE
nfs-client   cluster.local/nfs-subdir-external-provisioner   Delete          Immediate
↪   true                   12m
```

4. List to see if there are any PV and PVC available. Clean up in previous lab should have removed all of them.

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

```
No resources found
```

5. Create a YAML file for the new pvc.

```
student@cp:˜$ cp /home/student/LFS258/SOLUTIONS/s_09/pvc-sc.yaml .
```

```
student@cp:˜$ vim pvc-sc.yaml
```

pvc-sc.yaml

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

```

6. Create and verify when the new pvc is created, a dynamic volume is provisioned.

```
student@cp:˜$ kubectl create -f pvc-sc.yaml
```

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

```
student@cp:˜$ kubectl get pv,pvc
```

```
NAME                                                        CAPACITY   ACCESS MODES   RECLAIM
↪   POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
persistentvolume/pvc-71149612-33f1-4b18-916d-c67f79aca797   200Mi      RWX            Delete
↪   Bound    default/pvc-one   nfs-client              28s

NAME                            STATUS   VOLUME                                     CAPACITY
↪   ACCESS MODES   STORAGECLASS   AGE
persistentvolumeclaim/pvc-one   Bound    pvc-71149612-33f1-4b18-916d-c67f79aca797   200Mi      RWX
↪   nfs-client     28s
```

7. Create a new pod to use the pvc.

```
student@cp:˜$ cp /home/student/LFS258/SOLUTIONS/s_09/pod-sc.yaml .
```

```
student@cp:˜$ vim pod-sc.yaml
```

pod-sc.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
```

```yaml
  - image: nginx
    name: web-container
    volumeMounts:
    - name: nfs-volume
      mountPath: /usr/share/nginx/html
  volumes:
  - name: nfs-volume
    persistentVolumeClaim:
      claimName: pvc-one
```

8. Create the pod using the file.

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

```
pod/web-server created
```

9. Create a new file and copy it inside the pod.

```
student@cp:~$ echo "Welcome to the demo of storage class" > index.html
student@cp:~$ kubectl cp index.html web-server:/usr/share/nginx/html
```

10. The file was copied on to the default location of the nginx server. Instead of the ephemeral read-write layer of the container, the file is saved on the NFS server as we have made use of the PV.

```
student@cp:~$ ls -l /opt/sfw/default-pvc-one-pvc-<Hit the Tab key>
```

```
-rw-rw-r-- 1 student student 37 Jan  8 13:08 index.html
```

11. Cleanup by deleting the pod,volume claim.

```
student@cp:~$ kubectl delete pod/web-server pvc/pvc-one
```

```
pod "web-server" deleted
persistentvolumeclaim "pvc-one" deleted
```