Exercise 5.2: Configure the Deployment: Attaching Storage

There are several types of storage which can be accessed with Kubernetes, with flexibility of storage being essential to scalability. In this exercise we will configure an NFS server. With the NFS server we will create a new **persistent volume (pv)** and a **persistent volume claim (pvc)** to use it.

1. Search for `pv` and `pvc` YAML example files on http://kubernetes.io/docs and http://kubernetes.io/blog.

2. Use the `CreateNFS.sh` script from the tarball to set up NFS on your master node. This script will configure the server, export `/opt/sfw` and create a file `/opt/sfw/hello.txt`. Use the **find** command to locate the file if you don't remember where you extracted the tar file. This example narrows the search to your `$HOME` directory. Change for your environment. directory. You may find the same file in more than one sub-directory of the tarfile.

   ```
   student@master:~$ find $HOME -name CreateNFS.sh
   ```
   ```
   1  /home/student/LFD259/SOLUTIONS/s_05/CreateNFS.sh
   ```

   ```
   student@master:~$ cp /home/student/LFD259/SOLUTIONS/s_05/CreateNFS.sh $HOME
   ```

   ```
   student@master:~$ bash $HOME/CreateNFS.sh
   ```
   ```
   1  Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu xenial InRelease
   2  Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
   3
   4  <output_omitted>
   5
   6  Should be ready. Test here and second node
   7
   8  Export list for localhost:
   9  /opt/sfw *
   ```

3. Test by mounting the resource from your **second node**. Begin by installing the client software.

   ```
   student@worker:~$ sudo apt-get -y install nfs-common nfs-kernel-server
   ```
   ```
   1  <output_omitted>
   ```

4. Test you can see the exported directory using **showmount** from you second node.

   ```
   student@worker:~$ showmount -e master #<-- Edit to be first node's name or IP
   ```
   ```
   1  Export list for master:
   2  /opt/sfw *
   ```

5. Mount the directory. Be aware that unless you edit `/etc/fstab` this is not a persistent mount. Change out the node name for that of your master node.

   ```
   student@worker:~$ sudo mount master:/opt/sfw /mnt
   ```

6. Verify the `hello.txt` file created by the script can be viewed.

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

7. Return to the master node and create a YAML file for an object with kind **PersistentVolume**. The included example file needs an edit to the `server:` parameter. 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 incorrect resource will not start. Note that the `accessModes` do not currently affect actual access and are typically used as labels instead.

```
student@master:~$ find $HOME -name PVol.yaml
```

```
1  /home/student/LFD259/SOLUTIONS/s_05/PVol.yaml
```

```
student@master:~$ cp /home/student/LFD259/SOLUTIONS/s_05/PVol.yaml $HOME
```

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

PVol.yaml

```
1   apiVersion: v1
2   kind: PersistentVolume
3   metadata:
4     name: pvvol-1
5   spec:
6     capacity:
7       storage: 1Gi
8     accessModes:
9       - ReadWriteMany
10    persistentVolumeReclaimPolicy: Retain
11    nfs:
12      path: /opt/sfw
13      server: master                       #<-- Edit to match master node name or IP
14      readOnly: false
```

8. Create and verify you have a new 1Gi volume named **pvvol-1**. Note the status shows as `Available`. Remember we made two persistent volumes for the image registry earlier.

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

```
1  persistentvolume/pvvol-1 created
```

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

	NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
1	NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
2	pvvol-1	1Gi	RWX	Retain	Available				4s
3	registryvm	200Mi	RWO	Retain	Bound	default/nginx-claim0			4d
4	task-pv-volume	200Mi	RWO	Retain	Bound	default/registry-claim0			4d

9. Now that we have a new volume we will use a **persistent volume claim (pvc)** to use it in a Pod. We should have two existing claims from our local registry.

```
student@master:~/$ kubectl get pvc
```

	NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
1	NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
2	nginx-claim0	Bound	registryvm	200Mi	RWO		4d
3	registry-claim0	Bound	task-pv-volume	200Mi	RWO		4d

10. Create or copy a yaml file with the kind **PersistentVolumeClaim**.

```
student@master:~$ 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
```

11. Create and verify the new pvc status is `bound`. Note the size is `1Gi`, even though `200Mi` was suggested. Only a volume of at least that size could be used, the first volume with found with at least that much space was chosen.

    `student@master:~$ kubectl create -f pvc.yaml`

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

    `student@master:~$ kubectl get pvc`

    ```
    1   NAME            STATUS     VOLUME         CAPACITY   ACCESS MODES   STORAGECLASS   AGE
    2   nginx-claim0    Bound      registryvm     200Mi      RWO                           4d
    3   pvc-one         Bound      pvvol-1        1Gi        RWX                           4s
    4   registry-claim0 Bound      task-pv-volume 200Mi      RWO                           4d
    ```

12. Now look at the status of the physical volume. It should also show as bound.

    `student@master:~$ kubectl get pv`

    ```
    1   NAME            CAPACITY ACCESS MODES RECLAIM POLICY STATUS
    2    CLAIM          STORAGECLASS    REASON     AGE
    3   pvvol-1         1Gi      RWX          Retain         Bound
    4    default/pvc-one                14m
    5   registryvm      200Mi    RWO          Retain         Bound
    6    default/nginx-claim0           4d
    7   task-pv-volume 200Mi     RWO          Retain         Bound
    8    default/registry-claim0        4d
    ```

13. Edit the `simpleapp.yaml` file to include two new sections. One section for the container while will use the volume mount point, you should have an existing entry for `car-vol`. The other section adds a volume to the deployment in general, which you can put after the configMap volume section.

    `student@master:~$ vim $HOME/app1/simpleapp.yaml`

YAML **simpleapp.yaml**

```
1   ....
2          volumeMounts:
3          - name: car-vol
4            mountPath: /etc/cars
5          - name: nfs-vol              #<-- Add this and following line
6            mountPath: /opt
7   ....
8        volumes:
9         - name: car-vol
10          configMap:
11            defaultMode: 420
```

```
12            name: fast-car
13      - name: nfs-vol                        #<-- Add this and following two lines
14        persistentVolumeClaim:
15          claimName: pvc-one
16  status:
17  ....
```

14. Delete and re-create the deployment.

    `student@master:~$ kubectl delete deployment try1 ; kubectl create -f $HOME/app1/simpleapp.yaml`

```
1  deployment.apps "try1" deleted
2  deployment.apps/try1 created
```

15. View the details any of the pods in the deployment, you should see `nfs-vol` mounted under `/opt`. The use to command
    line completion with the **tab** key can be helpful for using a pod name.

    `student@master:~$ kubectl describe pod try1-594fbb5fc7-5k7sj`

```
1  <output_omitted>
2      Mounts:
3        /etc/cars from car-vol (rw)
4        /opt from nfs-vol (rw)
5        /var/run/secrets/kubernetes.io/serviceaccount from default-token-j7cqd (ro)
6  <output_omitted>
```

V_2021-01-26   LINUX FOUNDATION | Training & Certification