Lab: Deploying Static NFS Provisioning

Introduction:

Dynamic NFS Provisioning allows storage volumes to be created on-demand. The dynamic provisioning feature eliminates the need for cluster administrators to code-provision storage. Instead, it automatically provisions storage when it is requested by users.

Objectives:

- **Install and Configure NFS Server**
- **Create Static-NFS-PV**
- **Create PVC**
- **Create Pod to Consume PVC**
- **Cleanup**

Execute below commands **on all of the Servers** (**controller**, **node1** and **node2**) directly.

1. Install and Configure NFS Server

1.1 Let's **Install** the **nfs-utils** package by executing the below commands.

```
# dnf install -y nfs-utils
```

Note: Ignore this step if Package already installed.

1.2 Let's **enable** and **start** the NFS server by executing the below command.

```
# systemctl enable --now nfs-server
```

```
# systemctl status nfs-server --no-pager
```

Output:

```
[root@eoc-controller ~]#systemctl status nfs-server --no-pager
● nfs-server.service - NFS server and services
   Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor preset: disabled)
   Active: active (exited) since Tue 2023-09-12 07:38:08 EDT; 1s ago
  Process: 76790 ExecStart=/bin/sh -c if systemctl -q is-active gssproxy; then systemctl reload gss
proxy ; fi (code=exited, status=0/SUCCESS)
  Process: 76776 ExecStart=/usr/sbin/rpc.nfsd (code=exited, status=0/SUCCESS)
  Process: 76774 ExecStartPre=/usr/sbin/exportfs -r (code=exited, status=0/SUCCESS)
 Main PID: 76790 (code=exited, status=0/SUCCESS)

Sep 12 07:38:08 eoc-controller systemd[1]: Starting NFS server and services...
Sep 12 07:38:08 eoc-controller systemd[1]: Started NFS server and services.
```

Ensure that you have logged-in as **root** user on **eoc-controller** node.

1.3 In this particular example we'll allocate a local filesystem from which PersistenceVolume Claims can be made. Let's create **"/srv/nfs/kubedata",** by executing the below command.

```
# mkdir -p /srv/nfs/kubedata
```

1.4 Let's create the **/etc/exports** file by executing below command.

```
# cat > /etc/exports <<EOF
/srv/nfs/kubedata    *(rw,sync,no_root_squash,insecure)
EOF
```

Note:

rw: This option enables the NFS server to use both read and write requests on a NFS volume

sync: This option enables the NFS server to reply to requests only after the changes have been committed to stable storage

root squash: will allow the root user on the client to both access and create files on the NFS server as root.

insecure: This option accepts any or all ports.

1.5 Let's verify the **/etc/exports** file by executing below command.

```
# cat /etc/exports
```

Output:

```
[root@eoc-controller ~]#cat /etc/exports
/srv/nfs/kubedata    *(rw,sync,no_root_squash,insecure)
```

1.6 Let's **verify** the maintain table of exported NFS file systems by executing the below command.

```
# exportfs -avr
```

Output:

```
[root@eoc-controller ~]#exportfs -avr
exporting *:/srv/nfs/kubedata
```

1.7 If you want to check more details about our export file system you can run "**exportfs -v**".

```
# exportfs -v
```

Output:

```
[root@eoc-controller ~]#exportfs -v
/srv/nfs/kubedata
                <world>(sync,wdelay,hide,no_subtree_check,sec=sys,rw,insecure,no_root_squash,no_all_squash)
```

2. Create Static-NFS-PV

2.1 Let's **view** the yaml manifest file by executing below command.

```
# cat -n ~/kubernetes/storage-static-pv.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/storage-static-pv.yml
     1  apiVersion: v1
     2  kind: PersistentVolume
     3  metadata:
     4    name: pv-static-nfs
     5  spec:
     6    capacity:
     7      storage: 5Gi
     8    accessModes:
     9      - ReadWriteOnce
    10    persistentVolumeReclaimPolicy: Recycle
    11    nfs:
    12      path: /srv/nfs/kubedata
    13      server: 192.168.100.11
```

Note: Replace the above IP Address [192.168.100.11] in the manifest with your master IP

2.2 Let's **create** the **pv** using **~/kubernetes/storage-static-pv.yml** manifest file

```
# kubectl apply -f ~/kubernetes/storage-static-pv.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/storage-static-pv.yml
persistentvolume/pv-static-nfs created
```

2.3 Let's **list** the pv created by executing below command.

```
# kubectl get pv
```

Output:

```
[root@eoc-controller ~]#kubectl get pv
NAME            CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
pv-static-nfs   5Gi        RWO            Recycle          Available                                   47s
```

3 Create PVC

3.1 Let's **view** the yaml manifest file by executing the below command.

```
# cat -n ~/kubernetes/storage-static-pvc.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/storage-static-pvc.yml
     1  apiVersion: v1
     2  kind: PersistentVolumeClaim
     3  metadata:
     4    name: nfs-static-claim1
     5  spec:
     6    accessModes:
     7      - ReadWriteOnce
     8    resources:
     9      requests:
    10        storage: 1Gi
```

3.2 Let's **create** the **pvc** using **~/kubernetes/storage-static-pvc.yml** manifest file.

```
# kubectl apply -f ~/kubernetes/storage-static-pvc.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/storage-static-pvc.yml
persistentvolumeclaim/nfs-static-claim1 created
```

3.3 Let's **list** the pvc created by executing below command.

```
# kubectl get pvc
```

Output:

```
[root@eoc-controller ~]#kubectl get pvc
NAME              STATUS   VOLUME          CAPACITY   ACCESS MODES   STORAGECLASS   AGE
nfs-static-claim1   Bound    pv-static-nfs   5Gi        RWO                           35s
```

4 Create Pod to consume PVC

4.1 Let's **view** the yaml manifest file by executing below command.

```
# cat -n ~/kubernetes/storage-static-testpod.yml
```

Output:

```
[root@eoc-controller ~]#cat -n ~/kubernetes/storage-static-testpod.yml
     1  apiVersion: v1
     2  kind: Pod
     3  metadata:
     4    name: mypod-1
     5  spec:
     6    containers:
     7      - name: myfrontend
     8        image: nginx
     9
    10        volumeMounts:
    11        - mountPath: /usr/share/nginx/html
    12          name: mypd
    13
    14    volumes:
    15      - name: mypd
    16        persistentVolumeClaim:
    17          claimName: nfs-static-claim1
```

4.2 Ensure that **nfs-utils** package is installed on both the nodes (eoc-node1 and eoc-node2)

```
# dnf install -y nfs-utils
```

4.3 Let's **create** the pod by executing the below command

```
# kubectl apply -f ~/kubernetes/storage-static-testpod.yml
```

Output:

```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/storage-static-testpod.yml
pod/mypod-1 created
```

4.4 Enter into the container to view the mount point and create a file for testing

```
# echo "Test nfs-static Storage" > ~/index.html
# kubectl cp ~/index.html \
mypod-1:/usr/share/nginx/html/index.html
# kubectl get pods -o wide
```

Output:

```
[root@eoc-controller ~]#kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP          NODE        NOMINATED NODE   READINESS GATES
mypod-1   1/1     Running   0          58s   10.32.0.2   eoc-node1   <none>           <none>
```

```
# curl PODIP
```

Output:

```
[root@eoc-controller ~]#curl 10.32.0.2
Test nfs-static Storage
```

5 CleanUp

5.1 Let's **delete** the Pod by executing the below command.

```
# kubectl delete pod mypod-1
```

Output:

```
[root@eoc-controller ~]#kubectl delete pod mypod-1
pod "mypod-1" deleted
```

5.2 Let's **delete** the **pv**, **pvc** volume by executing the below command.

```
# kubectl delete pv pv-static-nfs
```

Output:

```
[root@eoc-controller ~]#kubectl delete pv pv-static-nfs
persistentvolume "pv-static-nfs" deleted
```

```
# kubectl delete pvc nfs-static-claim1
```

Output:

```
[root@eoc-controller ~]#kubectl delete pvc nfs-static-claim1
persistentvolumeclaim "nfs-static-claim1" deleted
```