Lesson 06 Demo 06

Configuring Pod Using NFS-Based PV and PVC

Objective: To configure a pod using NFS-based PersistentVolume (PV) and PersistentVolumeClaim (PVC) for more efficient storage management

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster should already be set up (refer to the steps provided in Lesson 02, Demo 01 for guidance)

Steps to be followed:

1. Configure the NFS kernel server
2. Set the permissions
3. Configure the NFS common on client machines
4. Create PersistentVolume
5. Create PersistentVolumeClaim
6. Create the deployment for MySQL

Step 1: Configure the NFS kernel server

1.1 Create a directory on the **worker-node-1** using the following command:
sudo mkdir /mydbdata

```
labsuser@worker-node-1:~$ sudo mkdir /mydbdata
labsuser@worker-node-1:~$
```

1.2 Install the NFS kernel server on the machine:
sudo apt install nfs-kernel-server

```
labsuser@worker-node-1:~$ sudo mkdir /mydbdata
labsuser@worker-node-1:~$ sudo apt install nfs-kernel-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  keyutils nfs-common rpcbind
Suggested packages:
  watchdog
The following NEW packages will be installed:
  keyutils nfs-common nfs-kernel-server rpcbind
0 upgraded, 4 newly installed, 0 to remove and 27 not upgraded.
1 not fully installed or removed.
Need to get 478 kB of archives.
After this operation, 1755 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 rpcbind amd64
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 keyutils amd64
```

Step 2: Set the permissions

2.1 On the **worker-node-1**, open the exports file in the /etc directory using the following command:
sudo nano /etc/exports

```
labsuser@worker-node-1:~$ sudo nano /etc/exports
```

2.2 Inside the file, append the following code:
/mydbdata *(rw,sync,no_root_squash)

```
 GNU nano 6.2                                                    /etc/exports *
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/mydbdata       *(rw,sync,no_root_squash)
```

2.3 Use the **cat** command to view the file:

```
labsuser@worker-node-1:~$ sudo nano /etc/exports
labsuser@worker-node-1:~$ sudo cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/mydbdata       *(rw,sync,no_root_squash)
labsuser@worker-node-1:~$
```

2.4 Export all shared directories defined in the **/etc/exports** file using the following command:
sudo exportfs -rv

```
labsuser@worker-node-1:~$ sudo exportfs -rv
exportfs: /etc/exports [1]: Neither 'subtree_check' or 'no_subtree_check' specified for export "*:/mydbdata".
  Assuming default behaviour ('no_subtree_check').
  NOTE: this default has changed since nfs-utils version 1.0.x

exporting *:/mydbdata
labsuser@worker-node-1:~$
```

2.5 Make the folder publicly accessible by changing its owner user and group using the following command:
sudo chown nobody:nogroup /mydbdata/

```
  Assuming default behaviour ('no_subtree_check').
  NOTE: this default has changed since nfs-utils version 1.0.x

exporting *:/mydbdata
labsuser@worker-node-1:~$ sudo chown nobody:nogroup /mydbdata/
labsuser@worker-node-1:~$
```

2.6 Assign full permissions to read, write, and execute files in this directory using the following command:
sudo chmod 777 /mydbdata/

```
labsuser@worker-node-1:~$ sudo chown nobody:nogroup /mydbdata/
labsuser@worker-node-1:~$ sudo chmod 777 /mydbdata/
labsuser@worker-node-1:~$
```

2.7 Restart the NFS kernel server to apply the changes using the following command:
sudo systemctl restart nfs-kernel-server

```
labsuser@worker-node-1:~$ sudo chown nobody:nogroup /mydbdata/
labsuser@worker-node-1:~$ sudo chmod 777 /mydbdata/
labsuser@worker-node-1:~$ sudo systemctl restart nfs-kernel-server
labsuser@worker-node-1:~$
```

2.8 Retrieve the internal IP of the node where the NFS server is installed using the following command:
ip a

```
labsuser@worker-node-1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens5: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:3f:4e:9e:56:a3 brd ff:ff:ff:ff:ff:ff
    altname enp0s5
    inet 172.31.16.178/20 metric 100 brd 172.31.31.255 scope global dynamic ens5
       valid_lft 3454sec preferred_lft 3454sec
    inet6 fe80::3f:4eff:fe9e:56a3/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:69:52:95:2f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: tunl0@NONE: <NOARP,UP,LOWER_UP> mtu 8981 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
    inet 192.168.47.128/32 scope global tunl0
       valid_lft forever preferred_lft forever
7: cali81635d883c8@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8981 qdisc noqueue state UP group default qlen 1000
    link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netns cni-ef8a4135-ff31-e159-7fea-a8485a001ad9
    inet6 fe80::ecee:eeff:feee:eeee/64 scope link
       valid_lft forever preferred_lft forever
labsuser@worker-node-1:~$
```

After running this command, look for the relevant IP address in the output. This IP will be used to associate the PV with the NFS server.

Note: Save the IP address to use in the next steps

Step 3: Configure the NFS common on client machines

> **Note:** Perform the below steps on each worker node intended for sharing

3.1 Run the following command to install the NFS common package:
sudo apt install nfs-common

```
labsuser@worker-node-2:~$ sudo apt install nfs-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  keyutils rpcbind
Suggested packages:
  watchdog
The following NEW packages will be installed:
  keyutils nfs-common rpcbind
0 upgraded, 3 newly installed, 0 to remove and 10 not upgraded.
1 not fully installed or removed.
Need to get 338 kB of archives.
After this operation, 1229 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 rpcbind amd64 1.2.6-2build1 [46.6 kB]
Get:2 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 keyutils amd64 1.6.1-2ubuntu3 [50.4 kB]
Get:3 http://us-west-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 nfs-common amd64 1:2.6.1-1ubuntu1.2 [241 kB]
Fetched 338 kB in 0s (9213 kB/s)
```

3.2 Execute the following commands to refresh the NFS common service and verify its status:
sudo rm /lib/systemd/system/nfs-common.service
sudo systemctl daemon-reload

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
labsuser@worker-node-2:~$ sudo rm /lib/systemd/system/nfs-common.service
labsuser@worker-node-2:~$ sudo systemctl daemon-reload
labsuser@worker-node-2:~$
```

3.3 Restart the NFS client service and check its status using the following commands:

sudo systemctl restart nfs-common

sudo systemctl status nfs-common

```
labsuser@worker-node-2:~$ sudo systemctl daemon-reload
labsuser@worker-node-2:~$ sudo systemctl restart nfs-common
labsuser@worker-node-2:~$ sudo systemctl status nfs-common
● nfs-common.service - LSB: NFS support files common to client and server
     Loaded: loaded (/etc/init.d/nfs-common; generated)
     Active: active (running) since Fri 2023-11-03 11:24:40 UTC; 15s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 54967 ExecStart=/etc/init.d/nfs-common start (code=exited, status=0/SUCCESS)
      Tasks: 2 (limit: 9379)
     Memory: 6.6M
        CPU: 171ms
     CGroup: /system.slice/nfs-common.service
             ├─54980 /sbin/rpc.statd
             └─55000 /usr/sbin/rpc.idmapd

Nov 03 11:24:39 worker-node-2.example.com systemd[1]: Starting LSB: NFS support files common to client and server...
Nov 03 11:24:39 worker-node-2.example.com nfs-common[54967]:  * Starting NFS common utilities
Nov 03 11:24:39 worker-node-2.example.com rpc.statd[54980]: Version 2.6.1 starting
Nov 03 11:24:39 worker-node-2.example.com sm-notify[54981]: Version 2.6.1 starting
Nov 03 11:24:39 worker-node-2.example.com sm-notify[54981]: Already notifying clients; Exiting!
Nov 03 11:24:39 worker-node-2.example.com rpc.statd[54980]: Failed to read /var/lib/nfs/state: No such file or directory
Nov 03 11:24:39 worker-node-2.example.com rpc.statd[54980]: Initializing NSM state
Nov 03 11:24:40 worker-node-2.example.com rpc.idmapd[55000]: Setting log level to 0
Nov 03 11:24:40 worker-node-2.example.com nfs-common[54967]:    ...done.
Nov 03 11:24:40 worker-node-2.example.com systemd[1]: Started LSB: NFS support files common to client and server.
labsuser@worker-node-2:~$
```

```
labsuser@worker-node-1:~$ sudo apt install nfs-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nfs-common is already the newest version (1:2.6.1-1ubuntu1.2).
nfs-common set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
labsuser@worker-node-1:~$ sudo rm /lib/systemd/system/nfs-common.service
labsuser@worker-node-1:~$ sudo systemctl daemon-reload
labsuser@worker-node-1:~$ sudo systemctl restart nfs-common
labsuser@worker-node-1:~$ sudo systemctl status nfs-common
● nfs-common.service - LSB: NFS support files common to client and server
     Loaded: loaded (/etc/init.d/nfs-common; generated)
     Active: active (exited) since Fri 2023-11-03 11:27:40 UTC; 7s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 56399 ExecStart=/etc/init.d/nfs-common start (code=exited, status=0/SUCCESS)
        CPU: 97ms

Nov 03 11:27:40 worker-node-1.example.com systemd[1]: Starting LSB: NFS support files common to client and server...
Nov 03 11:27:40 worker-node-1.example.com nfs-common[56399]:  * Starting NFS common utilities
Nov 03 11:27:40 worker-node-1.example.com nfs-common[56399]:    ...done.
Nov 03 11:27:40 worker-node-1.example.com systemd[1]: Started LSB: NFS support files common to client and server.
labsuser@worker-node-1:~$
```

Note: These steps are to be performed on both worker nodes as shown in the screenshots above.

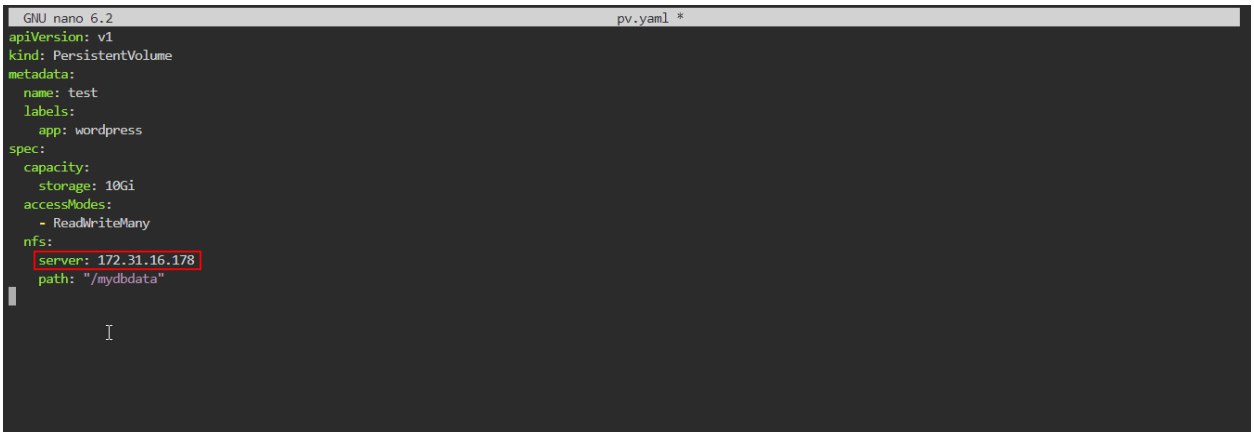
Step 4: Create PersistentVolume

4.1 On the **master** node, create the YAML file using the following command:
nano pv.yaml

```
labsuser@master:~$ nano pv.yaml
```

4.2 Add the following code to the **pv.yaml** file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: test
  labels:
    app: wordpress
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: YOUR_NFS_SERVER_IP_HERE
    path: "/mydbdata"
```



> **Note:** Replace **YOUR_NFS_SERVER_IP_HERE** with the internal IP of the NFS server from step **2.8** as shown in the screenshot above

4.3 Apply the configuration defined in **pv.yaml** using the following command:
kubectl apply -f pv.yaml

```
labsuser@master:~$ nano pv.yaml
labsuser@master:~$ kubectl apply -f pv.yaml
persistentvolume/test created
labsuser@master:~$
```

4.4 List all the **PVs** in the cluster using the following command:
kubectl get pv

```
persistentvolume/test created
labsuser@master:~$ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
test   10Gi       RWX            Retain           Available                                   87s
labsuser@master:~$
```

Step 5: Create PersistentVolumeClaim

5.1 Create the YAML file using the following command:
nano pvc.yaml

```
labsuser@master:~$ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM           STORAGECLASS   REASON   AGE
test   10Gi       RWX            Retain           Bound    default/mypvc1                          2m1s
labsuser@master:~$ nano pvc.yaml
```

5.2 Add the following code to the **pvc.yaml** file:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc1
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 6Gi
```

```
 GNU nano 6.2                                                mysql.yaml *
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mypvc1
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 6Gi

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set Mark   M-] To Bracket  M-Q Previous
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy       ^Q Where Was    M-W Next
```

5.3 Apply the configuration defined in **pvc.yaml** using the following command:
kubectl apply -f pvc.yaml

```
labsuser@master:~$ nano pvc.yaml
labsuser@master:~$ kubectl apply -f pvc.yaml
persistentvolumeclaim/mypvc1 created
labsuser@master:~$
```

5.4 List all the **PVs** and **PVCs** in the cluster using the following commands:
kubectl get pv
kubectl get pvc

```
labsuser@master:~$ kubectl apply -f pvc.yaml
persistentvolumeclaim/mypvc1 created
labsuser@master:~$ kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM            STORAGECLASS   REASON   AGE
test   10Gi       RWX            Retain           Bound    default/mypvc1                           7m42s
labsuser@master:~$ kubectl get pvc
NAME     STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mypvc1   Bound    test     10Gi       RWX                           3m2s
labsuser@master:~$
```

Step 6: Create the deployment for MySQL

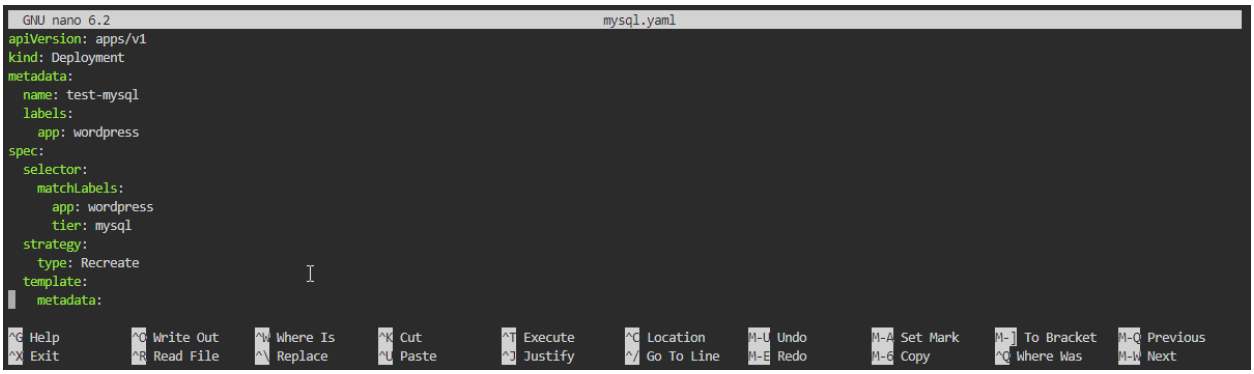
6.1 Create the YAML file using the following command:
nano mysql.yaml

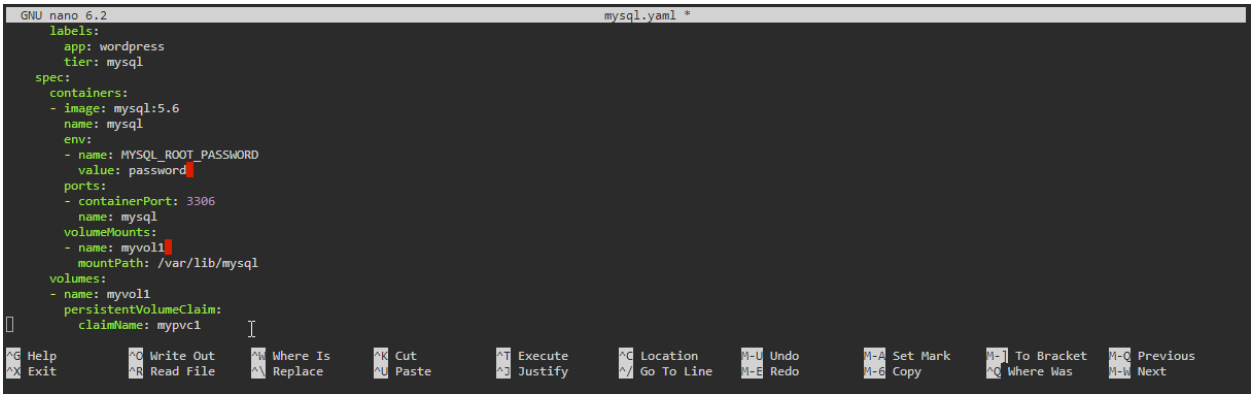
```
labsuser@master:~$ kubectl get pvc
NAME     STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mypvc1   Bound    test     10Gi       RWX                           2d22h
labsuser@master:~$ nano mysql.yaml
```

6.2 Add the following code to the **mysql.yaml** file:

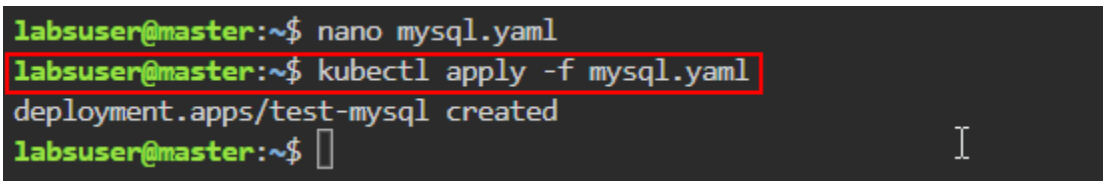
```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: test-mysql
 labels:
  app: wordpress
spec:
 selector:
  matchLabels:
   app: wordpress
   tier: mysql
 strategy:
  type: Recreate
 template:
  metadata:
   labels:
    app: wordpress
    tier: mysql
  spec:
   containers:
   - image: mysql:5.6
    name: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
     value: password
    ports:
    - containerPort: 3306
     name: mysql
    volumeMounts:
    - name: myvol1
     mountPath: /var/lib/mysql
   volumes:
   - name: myvol1
```

persistentVolumeClaim:
claimName: mypvc1





6.3 Apply the configuration defined in **mysql.yaml** using the following command:
kubectl apply -f mysql.yaml



6.4 Check the status of deployment using the following command:
kubectl get deploy test-mysql

```
labsuser@master:~$ nano mysql.yaml
labsuser@master:~$ kubectl apply -f mysql.yaml
deployment.apps/test-mysql created
labsuser@master:~$ kubectl get deploy test-mysql
NAME         READY   UP-TO-DATE   AVAILABLE   AGE
test-mysql   1/1     1            1           44s
labsuser@master:~$
```

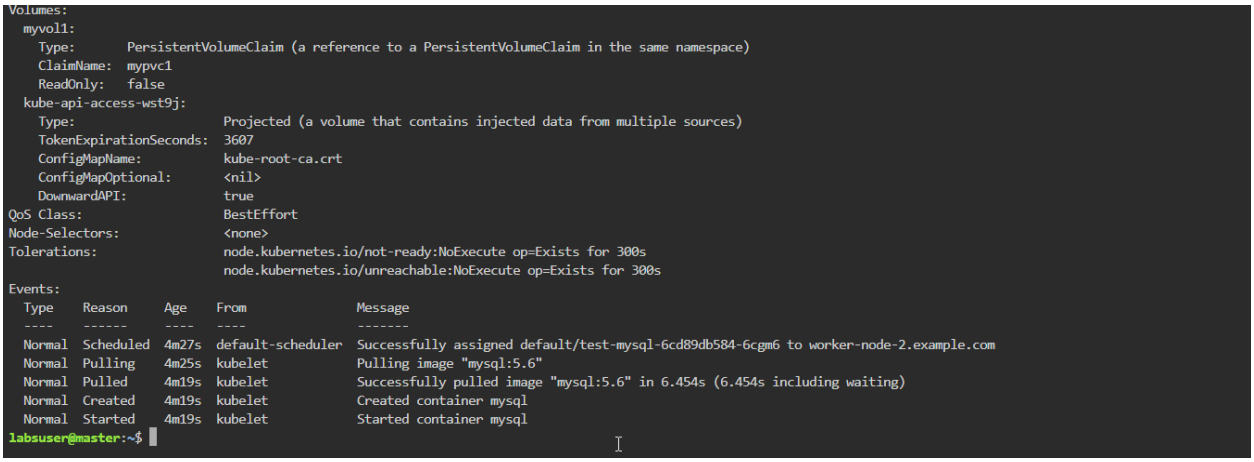
6.5 Check the status of the pod using the following command:
kubectl get pod -l app=wordpress

```
labsuser@master:~$ kubectl get pod -l app=wordpress
NAME                          READY   STATUS    RESTARTS   AGE
test-mysql-6cd89db584-6cgm6   1/1     Running   0          113s
labsuser@master:~$
```

Note: Save the **name** of the pod for the next step

6.6 View detailed information about the pod using the following command:
kubectl describe pod <pod-name>

```
labsuser@master:~$ kubectl describe pod test-mysql-6cd89db584-6cgm6
Name:             test-mysql-6cd89db584-6cgm6
Namespace:        default
Priority:         0
Service Account:  default
```



Note: Replace the **<pod-name>** with the **name** of your pod, as shown in the screenshots above

By following these steps, you have successfully configured a Kubernetes pod using NFS-based PV and PVC for efficient storage management.