**Lesson 07 Demo 06**

**Configuring Pod Using NFS based PersistentVolume (PV) and PersistentVolumeClaim (PVC)**

**Objective:** To configure a pod using NFS based PV and 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 the PersistentVolume
5. Create the PersistentVolumeClaim
6. Create the deployment for MySQL

**Step 1: Configure the NFS kernel server**

1. Create a directory on the **worker-node-1** using the following command:

**sudo mkdir /mydbdata**

**A black background with white text

Description automatically generated**

1. Install the NFS kernel server on the machine:

**sudo apt install nfs-kernel-server**

**A screenshot of a computer

Description automatically generated**

**Step 2: Set the permissions**

1. To grant permission to access the host server machine, open the exports file in the /etc directory:

**sudo nano /etc/exports**

****

1. Inside the file, append the following code:

**/mydbdata \*(rw,sync,no\_root\_squash)**

**A screen shot of a computer program

Description automatically generated**

1. Use the **cat** command to view the file:

**A screenshot of a computer program

Description automatically generated**

1. To export all shared directories, you have defined in the **/etc/exports file**, use:

**sudo exportfs -rv**

**A screen shot of a computer program

Description automatically generated**

1. Make the folder publicly accessible by changing its owner user and group:

**sudo chown nobody:nogroup /mydbdata/**

**A screenshot of a computer program

Description automatically generated**

1. Assign full permissions to ensure everyone can read, write, and execute files in this directory:

**sudo chmod 777 /mydbdata/**

**A screenshot of a computer

Description automatically generated**

1. Restart the NFS kernel server to apply the changes:  
     
   **sudo systemctl restart nfs-kernel-server**  
     
   A screenshot of a computer program

   Description automatically generated
2. Retrieve the internal IP of the node where NFS Server is installed, which will be used to link the PV

**ip a**

**A screenshot of a computer program

Description automatically generated**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. We will be using IP in step 4.

**Step 3: Configure the NFS common on client machines**

|  |
| --- |
| **Note:** Perform these steps on each worker node intended for sharing. |

1. Run the following command to install the NFS common package:

**sudo apt install nfs-common**

**A screenshot of a computer program

Description automatically generated**

1. Execute the following commands to refresh the NFS common service and verify its current status:  
     
   sudo rm /lib/systemd/system/nfs-common.service  
   sudo systemctl daemon-reload  
     
   A screenshot of a computer

   Description automatically generated

**sudo systemctl restart nfs-common**

**sudo systemctl status nfs-common**

A screen shot of a computer program

Description automatically generated  
  
A computer screen shot of a program

Description automatically generated

**Step 4: Create the PersistentVolume**

1. On the master node, draft the following YAML for the PV and save it as **pv.yaml:**  
     
   **nano** **pv.yaml**  
     
   A black and white screen with white text

   Description automatically generated
2. :

**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"  
  
A screenshot of a computer

Description automatically generated**

|  |
| --- |
| **Note:** Replace **YOUR\_NFS\_SERVER\_IP\_HERE** with the internal IP of the NFS server. |

1. Create and check the PV using:  
     
   **kubectl apply -f pv.yaml  
     
   A screen shot of a computer

   Description automatically generated**

**kubectl get pv**

**A screenshot of a computer screen

Description automatically generated**

**Step 5: Create the PersistentVolumeClaim**

1. **nano pvc.yaml**A screen shot of a computer

   Description automatically generated
2. On the master node, draft the YAML for the PVC and save it as **pvc.yaml**:

**apiVersion: v1**

**kind: PersistentVolumeClaim**

**metadata:**

**name: mypvc1**

**labels:**

**app: wordpress**

**spec:**

**accessModes:**

**- ReadWriteMany**

**resources:**

**requests:**

**storage: 6Gi**

**A computer screen with a black background

Description automatically generated**

1. Create and verify the PVC with:  
   **kubectl apply -f pvc.yaml**  
     
   A black screen with white text

   Description automatically generated
2. **kubectl get pv  
   kubectl get pvc**  
   A black background with white text

   Description automatically generated

**Step 6: Create the deployment for MySQL**

1. **nano mysql.yaml**
2. To bind the PVC to the MySQL Pod, draft the following YAML and save it as **mysql.yaml**:

**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**

**A computer screen with a black background

Description automatically generated**

**A computer screen shot of a black screen

Description automatically generated**

1. Deploy MySQL and verify its pods using:  
     
     
   **kubectl apply -f mysql.yaml**  
     
   A black screen with white text

   Description automatically generated
2. Check the status of deployment,

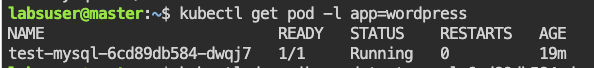
**kubectl get deploy test-mysql**  
  
**Error: NotReady**  
  
**A screen shot of a computer

Description automatically generated**  
  
A black background with white text

Description automatically generated

1. Check the status of pod,

kubectl get pod -l app=wordpress  
A screen shot of a computer program

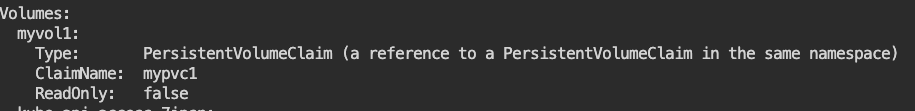
Description automatically generated  
  


1. Verify the Pod is using NFS Volume ‘mypvc1’,

**kubectl describe pod test-mysql-6cd89db584-dwqj7**

**A screenshot of a computer

Description automatically generated**



By following these steps, you have successfully set up a MySQL pod connected to PersistentVolume (uses NFS) and PersistentVolumeClaim. This ensures that data remains intact even if the pod terminates.