# ­A close up of a logo Description automatically generated

**Lesson 7 Demo 2**

**Understanding PersistentVolume (PV) And PersistentVolumeClaim (PVC)**

**Objective:** To configure a Pod with a PersistentVolume and a PersistentVolumeClaim for storage management

**Tools required:** kubeadm, kubectl, kubelet, and etcd

**Prerequisites:** A Kubernetes cluster must be set up (follow steps of Lesson 2 Demo 1)

Steps to be followed:

1. Configuring the NFS kernel server
2. Setting permissions
3. Configuring the NFS common on client machines
4. Creating PersistentVolume
5. Creating a PersistentVolumeClaim
6. Creating Deployment for MySQL

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

1. Create a directory on any machine that you wish to share with the **client** system.

**sudo mkdir /mydbdata**

**A picture containing graphical user interface

Description automatically generated**

1. Run the following command to install the **NFS kernel server** on the machine:

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

Text

Description automatically generated

Text

Description automatically generated

**Step 2: Creating files within the Pod**

1. Open the **exports** file in the **/etc** directory for permission to access the host server machine.

**sudo vi /etc/exports**

**A screenshot of a computer

Description automatically generated with medium confidence**

Add the following code to the file:

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

**Text

Description automatically generated**

|  |
| --- |
| Note: Exit the file and save the changes. |

1. Use the **exportfs** command to export all shared folders you registered in **/etc/exports** file after making the appropriate changes.

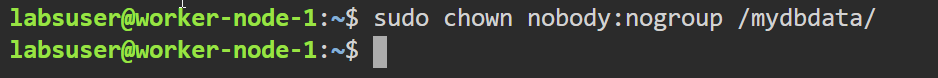
**sudo exportfs -rv**

**Text

Description automatically generated**

1. Change the **owner user** and **group** to **nobody** and **nogroup**.

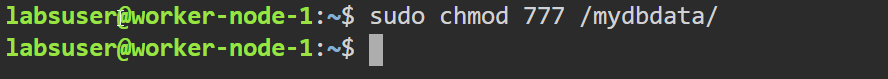
**sudo chown nobody:nogroup /mydbdata/**

****

This option makes the folder publicly accessible.

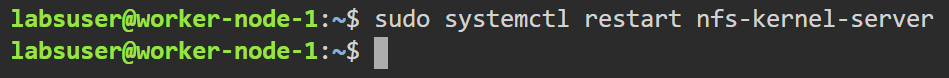
1. Set permissions to **777** to allow everyone to read, write, and execute files in this directory.

**sudo chmod 777 /mydbdata/**

****

1. Next, restart the **NFS kernel server** to apply the configuration changes.

**sudo systemctl restart nfs-kernel-server**

****

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

|  |
| --- |
| Note: Follow the steps below on all the worker nodes you wish to use as sharing clients. |

1. Install the **NFS common** package on client machines to enable **NFS**.

**sudo apt install nfs-common**

**Text

Description automatically generated**

1. Copy the internal IP of the **nfs-server** to attach **PV** to it.

**ip a**

**Text

Description automatically generated**

**Step 4: Creating PersistentVolume**

1. To create **PV** on the master node, write the following code in the **pv.yaml** file:

**apiVersion: v1**

**kind: PersistentVolume**

**metadata:**

**name: test**

**labels:**

**app: wordpress**

**spec:**

**capacity:**

**storage: 10Gi**

**accessModes:**

**- ReadWriteMany**

**nfs:**

**server: 172.31.52.235**

**# Exported path of your NFS server dont forget to change the above line**

**path: "/mydbdata"**

**Text

Description automatically generated**

|  |
| --- |
| Note: In the **server** attribute of the **PV** file, add the **nfs-server** internal IP. |

1. Using the following commands, create and verify **PV**:

**kubectl create -f pv.yaml**

**kubectl get pv**

**Text

Description automatically generated**

**Step 5: Creating PersistentVolumeClaim**

1. To create and attach **PVC** to **PV**, add the following code in the **pvc.yaml** file on the **master** node:

**apiVersion: v1**

**kind: PersistentVolumeClaim**

**metadata:**

**name: mypvc1**

**labels:**

**app: wordpress**

**spec:**

**accessModes:**

**- ReadWriteMany**

**resources:**

**requests:**

**storage: 6Gi**

**Text

Description automatically generated**

1. Run the following commands to create and verify **PVC**:

**kubectl create -f pvc.yaml**

**kubectl get pvc**

**Text

Description automatically generated**

**Step 6: Creating Deployment for MySQL**

1. To attach **PVC** to **MySQL** Pod, add the following code in the **mysql.yaml** file on the **master** node:

**apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2**

**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**

Text

Description automatically generated

Text

Description automatically generated

1. Run the following commands to create and verify a **MySQL** Deployment:

**kubectl create -f mysql.yaml**

**kubectl get pods**

**Text

Description automatically generated**

**Text

Description automatically generated**

The **MySQL** Pod attached with **PV** and **PVC** persistent volumes has been successfully created to manage storage, as seen in the above screenshot, and the data will no longer be lost even if the Pods are terminated.