Lesson-End Project

Deploying WordPress and MySQL Using PersistentVolume

Project agenda: To deploy WordPress and MySQL on Kubernetes with PersistentVolume using NFS and hostPath for web-based access

Description: This project involves the deployment of WordPress and MySQL on a Kubernetes cluster, leveraging PersistentVolume with NFS and hostPath configurations to host a web-based WordPress application with data persistence and accessibility.

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Expected deliverables: A fully deployed Kubernetes cluster and MySQL and WordPress applications, with PersistentVolume, that allow web-based access to the WordPress site

Steps to be followed:

1. Configure the NFS kernel server
2. Set the permissions
3. Configure NFS common on client machines
4. Create a MySQL manifest file and deploy it using NFS-based PersistentVolume
5. Create a WordPress manifest file and deploy it using hostPath-based PersistentVolume

Step 1: Configure the NFS kernel server

1.1 Switch to superuser, create a directory **/data** with the **-p** option to create parent directories on **worker-node-1**, and then list the contents of **/data** using the following commands:
sudo su
mkdir -p /data
ls -alrt /data/

```
labsuser@worker-node-1:~$ sudo su
root@worker-node-1:/home/labsuser# mkdir -p /data
root@worker-node-1:/home/labsuser# ls -alrt /data/
total 8
drwxr-xr-x 21 root root 4096 Nov  6 13:31 ..
drwxr-xr-x  2 root root 4096 Nov  6 13:31 .
root@worker-node-1:/home/labsuser# exit
exit
labsuser@worker-node-1:~$
```

Note: Type **exit** and press the **enter** key

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

```
labsuser@worker-node-1:~$ sudo su
root@worker-node-1:/home/labsuser# mkdir -p /data
root@worker-node-1:/home/labsuser# ls -alrt /data/
total 8
drwxr-xr-x 21 root root 4096 Nov  6 13:31 ..
drwxr-xr-x  2 root root 4096 Nov  6 13:31 .
root@worker-node-1:/home/labsuser# exit
exit
labsuser@worker-node-1:~$ sudo apt install nfs-kernel-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nfs-kernel-server is already the newest version (1:2.6.1-1ubuntu1.2).
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
labsuser@worker-node-1:~$
```

Step 2: Set the permissions

2.1 On **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, add the following code:
/data *(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)
#
/data   *(rw,sync,no_root_squash)
```

2.3 Use the following **cat** command to view the file:
sudo cat /etc/exports

```
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)
#
/data   *(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

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

exporting *:/data
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 /data/

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

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

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

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

exporting *:/data
labsuser@worker-node-1:~$ sudo chown nobody:nogroup /data/
labsuser@worker-node-1:~$ sudo chmod 777 /data/
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 /data/
labsuser@worker-node-1:~$ sudo chmod 777 /data/
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:~$ sudo chown nobody:nogroup /data/
labsuser@worker-node-1:~$ sudo chmod 777 /data/
labsuser@worker-node-1:~$ sudo systemctl restart nfs-kernel-server
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:d4:2a:86:ec:33 brd ff:ff:ff:ff:ff:ff
    altname enp0s5
    inet 172.31.26.27/20 metric 100 brd 172.31.31.255 scope global dynamic ens5
       valid_lft 2152sec preferred_lft 2152sec
    inet6 fe80::d4:2aff:fe86:ec33/64 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:ee:06:4e:22 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
labsuser@worker-node-1:~$
```

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

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

Step 3: Configure NFS common on client machines

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

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

```
       valid_lft forever preferred_lft forever
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).
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
labsuser@worker-node-1:~$
```

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

```
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:~$
```

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-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 Mon 2023-11-06 17:01:22 UTC; 12s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 11911 ExecStart=/etc/init.d/nfs-common start (code=exited, status=0/SUCCESS)
        CPU: 125ms

Nov 06 17:01:21 worker-node-1.example.com systemd[1]: Starting LSB: NFS support files common to client and server...
Nov 06 17:01:21 worker-node-1.example.com nfs-common[11911]:  * Starting NFS common utilities
Nov 06 17:01:22 worker-node-1.example.com nfs-common[11911]:    ...done.
Nov 06 17:01:22 worker-node-1.example.com systemd[1]: Started LSB: NFS support files common to client and server.
labsuser@worker-node-1:~$
```

Step 4: Create a MySQL manifest file and deploy it using NFS-based PersistentVolume

4.1 On the master node, retrieve information about the nodes in the Kubernetes cluster using the following command:
kubectl get node

```
labsuser@master:~$ kubectl get node
NAME                      STATUS   ROLES           AGE   VERSION
master.example.com        Ready    control-plane   13m   v1.28.2
worker-node-1.example.com Ready    <none>          11m   v1.28.2
worker-node-2.example.com Ready    <none>          11m   v1.28.2
labsuser@master:~$
```

4.2 Create a YAML file using the following command:
nano nfs-mysql.yaml

```
labsuser@master:~$ kubectl get node
NAME                      STATUS   ROLES           AGE   VERSION
master.example.com        Ready    control-plane   14m   v1.28.2
worker-node-1.example.com Ready    <none>          12m   v1.28.2
worker-node-2.example.com Ready    <none>          12m   v1.28.2
labsuser@master:~$ nano nfs-mysql.yaml
```

4.3 Add the following code to the **nfs-mysql.yaml** file, replacing **server: IP** with the internal IP of the NFS server from step 2.8 as shown in the screenshot below:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-nfs
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  mountOptions:
    - hard
    - nfsvers=4.1
```

```
  nfs:

    path: /data
    server: 172.31.26.27
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-nfs
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 500Mi
---
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql-wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: mysql-wordpress
    product: mysql
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  labels:
    app: mysql-wordpress
spec:
  selector:
    matchLabels:
      app: mysql-wordpress
      product: mysql
  strategy:
    type: Recreate
  template:
    metadata:
```

```
    labels:
      app: mysql-wordpress
      product: mysql
  spec:
    containers:
    - image: mysql
      name: mysql-container
      env:
      - name: MYSQL_DATABASE
        value: wordpress
      - name: MYSQL_ROOT_PASSWORD
        value: rootroot
      ports:
      - containerPort: 3306
        name: mysql
      volumeMounts:
      - name: mysql-storage
        mountPath: /var/lib/mysql
    volumes:
    - name: mysql-storage
      persistentVolumeClaim:
        claimName: mysql-nfs
```

```
  GNU nano 6.2                                   nfs-mysql.yaml *
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-nfs
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /data
    server: 172.31.26.27
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-nfs
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 500Mi

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

4.4 Use the following **cat** command to view the content of **nfs-mysql.yaml** file:
cat nfs-mysql.yaml

```
labsuser@master:~$ cat nfs-mysql.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-nfs
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Recycle
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /data
    server: 172.31.26.27
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: mysql-nfs
spec:
```

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

```
labsuser@master:~$ kubectl apply -f nfs-mysql.yaml
persistentvolume/mysql-nfs created
persistentvolumeclaim/mysql-nfs created
service/mysql created
deployment.apps/mysql created
labsuser@master:~$
```

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

```
deployment.apps/mysql created
labsuser@master:~$ kubectl get pv
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM               STORAGECLASS   REASON   AGE
mysql-nfs   1Gi        RWX            Recycle          Bound    default/mysql-nfs                           57s
labsuser@master:~$
```

4.7 List all the PVCs in the cluster using the following command:
kubectl get pvc

```
labsuser@master:~$ kubectl get pv
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                 STORAGECLASS   REASON   AGE
mysql-nfs   1Gi        RWX            Recycle          Bound    default/mysql-nfs                            57s
labsuser@master:~$ kubectl get pvc
NAME        STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-nfs   Bound    mysql-nfs   1Gi        RWX                           110s
labsuser@master:~$
```

4.8 List deployments in the Kubernetes cluster and display additional wide output details using the following command:
kubectl get deployments -o wide

```
labsuser@master:~$ kubectl get deployments -o wide
NAME      READY   UP-TO-DATE   AVAILABLE   AGE   CONTAINERS        IMAGES      SELECTOR
mysql     1/1     1            1           62m   mysql-container   mysql       app=mysql-wordpress,product=mysql
```

4.9 Retrieve a list of services in the Kubernetes cluster and display additional wide output details using the following command:
kubectl get services -o wide

```
labsuser@master:~$ kubectl get services -o wide
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE   SELECTOR
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP   155m  <none>
mysql        ClusterIP   10.111.153.124   <none>        3306/TCP  64m   app=mysql-wordpress,product=mysql
```

4.10 Obtain a list of pods in the Kubernetes cluster and display additional wide output details using the following command:
kubectl get pods -o wide

```
labsuser@master:~$ kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP               NODE                   NOMINATED NODE   READINESS GATES
mysql-5c9cd7fd4d-hs2c8   1/1     Running   0          64m   192.168.232.195  worker-node-2.example.com  <none>       <none>
```

Note: Save the pod name for the next step

4.11 View the last 10 lines of logs for a specific pod using the following command:

kubectl logs <pod_name> | tail -n 10

```
labsuser@master:~$ kubectl logs mysql-5c9cd7fd4d-hs2c8 | tail -n 10
2023-11-06T18:21:27.324673Z 0 [System] [MY-015015] [Server] MySQL Server - start.
2023-11-06T18:21:27.669292Z 0 [Warning] [MY-011068] [Server] The syntax '--skip-host-cache' is deprecated and will be removed in a future release. Pl
ease use SET GLOBAL host_cache_size=0 instead.
2023-11-06T18:21:27.682398Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.2.0) starting as process 1
2023-11-06T18:21:27.712267Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2023-11-06T18:21:28.992024Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2023-11-06T18:21:31.916012Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.
2023-11-06T18:21:31.916054Z 0 [System] [MY-013602] [Server] Channel mysql_main configured to support TLS. Encrypted connections are now supported for
 this channel.
2023-11-06T18:21:31.920017Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessi
ble to all OS users. Consider choosing a different directory.
2023-11-06T18:21:32.481789Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/m
ysqlx.sock
2023-11-06T18:21:32.482061Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.2.0'  socket: '/var/run/mysqld/mysql
d.sock'  port: 3306  MySQL Community Server - GPL.
labsuser@master:~$
```

Note: Replace **<pod_name>** with the name of the pod from step **4.10** as shown in the screenshot above

Step 5: Create a WordPress manifest file and deploy it using hostPath-based PersistentVolume

5.1 Create a YAML file using the following command:

nano wordpress-volume.yaml

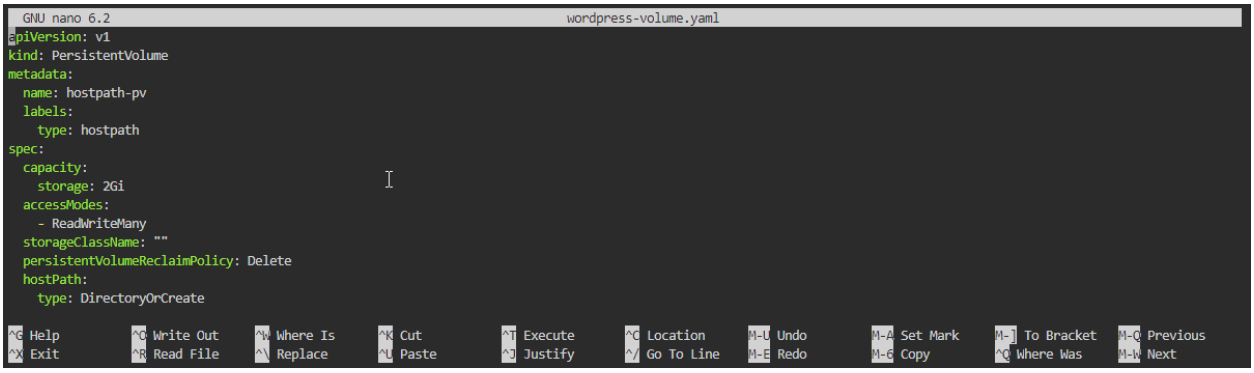
```
2023-11-06T18:21:31.920017Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessi
ble to all OS users. Consider choosing a different directory.
2023-11-06T18:21:32.481789Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '::' port: 33060, socket: /var/run/mysqld/m
ysqlx.sock
2023-11-06T18:21:32.482061Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.2.0'  socket: '/var/run/mysqld/mysql
d.sock'  port: 3306  MySQL Community Server - GPL.
labsuser@master:~$ nano wordpress-volume.yaml
```

5.2 Add the following code to the **wordpress-volume.yaml** file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: hostpath-pv
  labels:
    type: hostpath
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
```

```
  storageClassName: ""

  persistentVolumeReclaimPolicy: Delete
  hostPath:
    type: DirectoryOrCreate
    path: "/opt/ "
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: wordpress-hostpath
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 500Mi
```



5.3 Use the following **cat** command to view the content of the **wordpress-volume.yaml** file:
cat wordpress-volume.yaml

```
labsuser@master:~$ nano wordpress-volume.yaml
labsuser@master:~$ cat wordpress-volume.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: hostpath-pv
  labels:
    type: hostpath
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  persistentVolumeReclaimPolicy: Delete
  hostPath:
```

5.4 Apply the configuration defined in **wordpress-volume.yaml** using the following command:
kubectl apply -f wordpress-volume.yaml

```
metadata:
  name: wordpress-hostpath
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: ""
  resources:
    requests:
      storage: 500Mi
labsuser@master:~$ kubectl apply -f wordpress-volume.yaml
persistentvolume/hostpath-pv created
persistentvolumeclaim/wordpress-hostpath created
labsuser@master:~$
```

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

```
labsuser@master:~$ kubectl get pv
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                          STORAGECLASS   REASON   AGE
hostpath-pv   2Gi        RWX            Delete           Bound    default/wordpress-hostpath                              87s
mysql-nfs     1Gi        RWX            Recycle          Bound    default/mysql-nfs                                       103m
labsuser@master:~$ kubectl get pvc
NAME                 STATUS   VOLUME        CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-nfs            Bound    mysql-nfs     1Gi        RWX                           104m
wordpress-hostpath   Bound    hostpath-pv   2Gi        RWX                           101s
labsuser@master:~$
```

5.6 Create a YAML file using the following command:

nano wordpress-deployment.yaml

```
labsuser@master:~$ kubectl get pv
NAME          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                          STORAGECLASS   REASON   AGE
hostpath-pv   2Gi        RWX            Delete           Bound    default/wordpress-hostpath                              87s
mysql-nfs     1Gi        RWX            Recycle          Bound    default/mysql-nfs                                       103m
labsuser@master:~$ kubectl get pvc
NAME                  STATUS   VOLUME        CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mysql-nfs             Bound    mysql-nfs     1Gi        RWX                           104m
wordpress-hostpath    Bound    hostpath-pv   2Gi        RWX                           101s
labsuser@master:~$ nano wordpress-deployment.yaml
```

5.7 Add the following code to the **wordpress-deployment.yaml** file:

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: mysql-wordpress
spec:
  ports:
    - port: 80
  selector:
    app: mysql-wordpress
    tier: frontend
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: mysql-wordpress
spec:
  selector:
    matchLabels:
      app: mysql-wordpress
      tier: frontend
  strategy:
    type: Recreate
  template:
```

```
metadata:
  labels:
    app: mysql-wordpress
    tier: frontend
spec:
  containers:
  - image: wordpress
    name: wordpress
    env:
    - name: WORDPRESS_DB_HOST
      value: mysql
    - name: WORDPRESS_DB_USER
      value: root
    - name: WORDPRESS_DB_PASSWORD
      value: rootroot
    ports:
    - containerPort: 80
      name: wordpress
    volumeMounts:
    - name: wordpress-storage
      mountPath: /var/www/html
  volumes:
  - name: wordpress-storage
    persistentVolumeClaim:
```

claimName: wordpress-hostpath

```
  GNU nano 6.2                                   wordpress-deployment.yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress
  labels:
    app: mysql-wordpress
spec:
  ports:
    - port: 80
  selector:
    app: mysql-wordpress
    tier: frontend
  type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: mysql-wordpress
spec:
  selector:
    matchLabels:
      app: mysql-wordpress
      tier: frontend
  strategy:
    type: Recreate

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

5.8 Apply the configuration defined in **wordpress-deployment.yaml** using the following command:
kubectl apply -f wordpress-deployment.yaml

```
          claimName: wordpress-hostpath
labsuser@master:~$ kubectl apply -f wordpress-deployment.yaml
service/wordpress created
deployment.apps/wordpress created
labsuser@master:~$
```

5.9 List deployments in the Kubernetes cluster and display additional wide output details using the following command:
kubectl get deployments -o wide

```
labsuser@master:~$ kubectl apply -f wordpress-deployment.yaml
service/wordpress created
deployment.apps/wordpress created
labsuser@master:~$ kubectl get deployments -o wide
NAME        READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS        IMAGES      SELECTOR
mysql       1/1     1            1           113m   mysql-container   mysql       app=mysql-wordpress,product=mysql
wordpress   1/1     1            1           88s    wordpress         wordpress   app=mysql-wordpress,tier=frontend
labsuser@master:~$
```

5.10 Retrieve a list of services in the Kubernetes cluster and display additional wide output details using the following command:
kubectl get services -o wide

```
labsuser@master:~$ kubectl get deployments -o wide
NAME        READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS        IMAGES      SELECTOR
mysql       1/1     1            1           113m   mysql-container   mysql       app=mysql-wordpress,product=mysql
wordpress   1/1     1            1           88s    wordpress         wordpress   app=mysql-wordpress,tier=frontend
labsuser@master:~$ kubectl get services -o wide
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE     SELECTOR
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        3h27m   <none>
mysql        ClusterIP   10.111.153.124   <none>        3306/TCP       116m    app=mysql-wordpress,product=mysql
wordpress    NodePort    10.99.109.187    <none>        80:31319/TCP   4m11s   app=mysql-wordpress,tier=frontend
labsuser@master:~$
```

Note: Save the port number for the next steps

5.11 Obtain a list of pods in the Kubernetes cluster and display additional wide output details using the following command:
kubectl get pods -o wide

```
labsuser@master:~$ kubectl get deployments -o wide
NAME        READY   UP-TO-DATE   AVAILABLE   AGE    CONTAINERS        IMAGES      SELECTOR
mysql       1/1     1            1           113m   mysql-container   mysql       app=mysql-wordpress,product=mysql
wordpress   1/1     1            1           88s    wordpress         wordpress   app=mysql-wordpress,tier=frontend
labsuser@master:~$ kubectl get services -o wide
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE     SELECTOR
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        3h27m   <none>
mysql        ClusterIP   10.111.153.124   <none>        3306/TCP       116m    app=mysql-wordpress,product=mysql
wordpress    NodePort    10.99.109.187    <none>        80:31319/TCP   4m11s   app=mysql-wordpress,tier=frontend
labsuser@master:~$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP                NODE                        NOMINATED NODE   READINESS GATES
mysql-5c9cd7fd4d-hs2c8       1/1     Running   0          117m    192.168.232.195   worker-node-2.example.com   <none>           <none>
wordpress-6bd8cc8884-68msn   1/1     Running   0          5m20s   192.168.47.134    worker-node-1.example.com   <none>           <none>
labsuser@master:~$
```

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

5.12 Stream and monitor **WordPress** pod logs by executing the following command:
kubectl logs <pod-name> -f

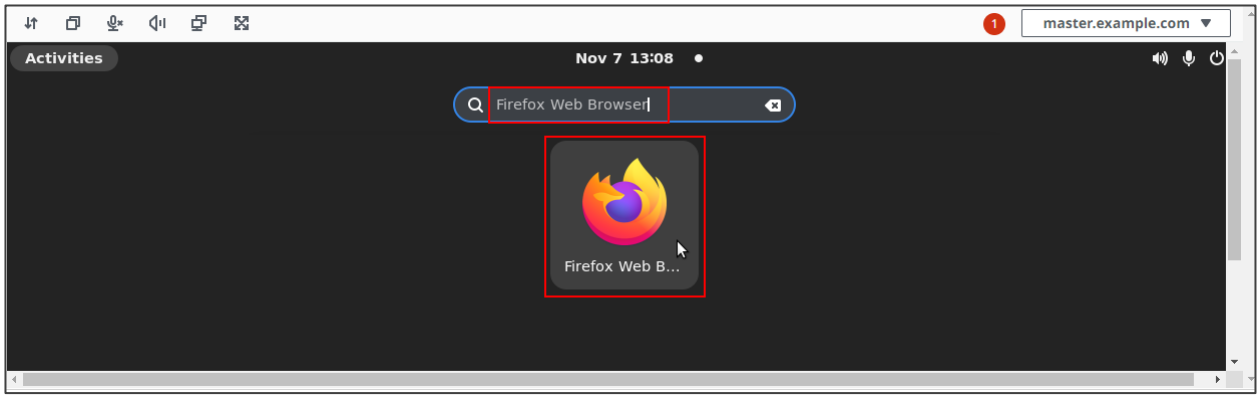
```
labsuser@master:~$ kubectl get pods -o wide
NAME                         READY   STATUS    RESTARTS   AGE     IP                NODE                        NOMINATED NODE   READINESS GATES
mysql-5c9cd7fd4d-hs2c8       1/1     Running   0          117m    192.168.232.195   worker-node-2.example.com   <none>           <none>
wordpress-6bd8cc8884-68msn   1/1     Running   0          5m20s   192.168.47.134    worker-node-1.example.com   <none>           <none>
labsuser@master:~$ kubectl logs wordpress-6bd8cc8884-68msn -f
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 192.168.47.134. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 192.168.47.134. Set the 'ServerName' directive globally to suppress this message
[Mon Nov 06 20:12:54.145377 2023] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.56 (Debian) PHP/8.0.30 configured -- resuming normal operations
[Mon Nov 06 20:12:54.145628 2023] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
```

Note: Replace **<pod-name>** with the name of your pod from step 5.11

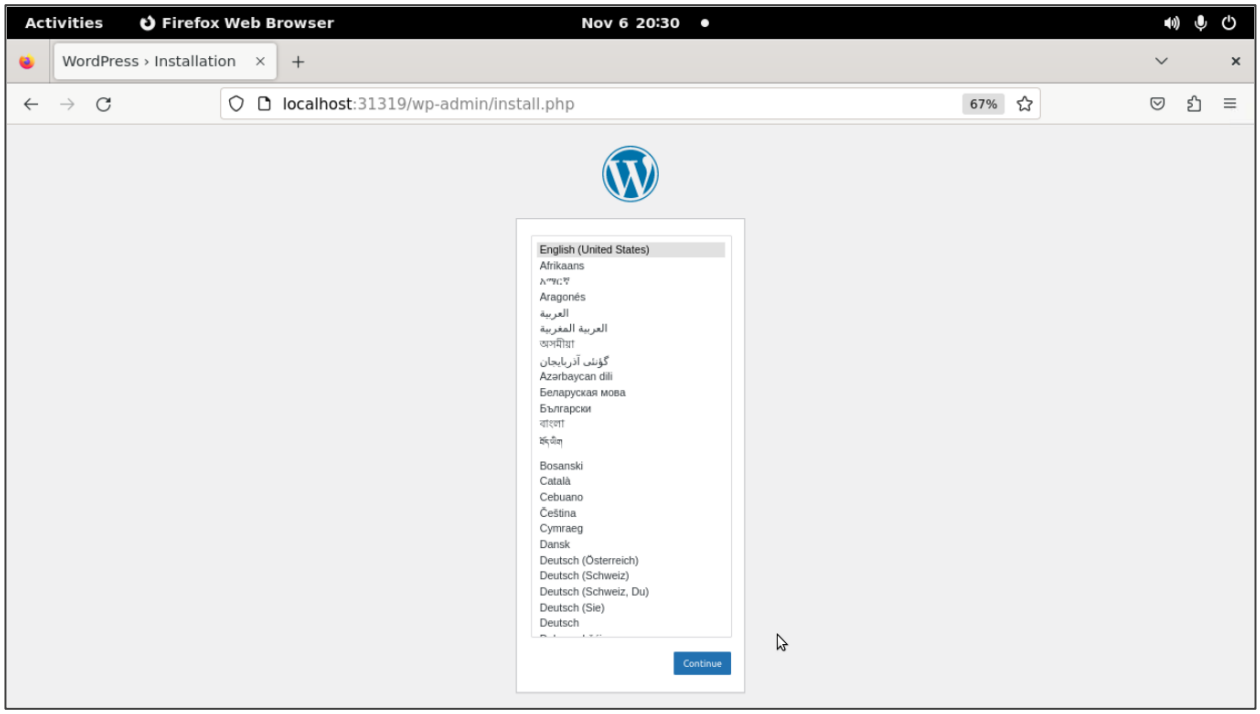
5.13 Click on the **master** button and then on the **desktop** option

5.14 Click on the **Activities** button

5.15 Search for and click on **Firefox Web Browser**



5.16 In the Firefox browser, add the following URL to access the WordPress application: **http://localhost:<node-port>**



Note: Replace **<node-port>** with the port number saved earlier from step 5.10

By following these steps, you have successfully configured a Kubernetes cluster to deploy WordPress and MySQL with PersistentVolume, ensuring data persistence and enabling web access to your WordPress application.