

Lesson 07 Demo 07

Configuring Multi-Container Pods with RWX Access Using PV and PVC

Objective: To configure multi-container pods with ReadWriteMany (RWX) access in Kubernetes using PersistentVolume (PV) and PersistentVolumeClaim (PVC) for shared storage and data operations

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. Create PersistentVolume
2. Create PersistentVolumeClaim
3. Deploy a pod in a new namespace
4. Demonstrate shared storage and data operations
5. Continue data operations

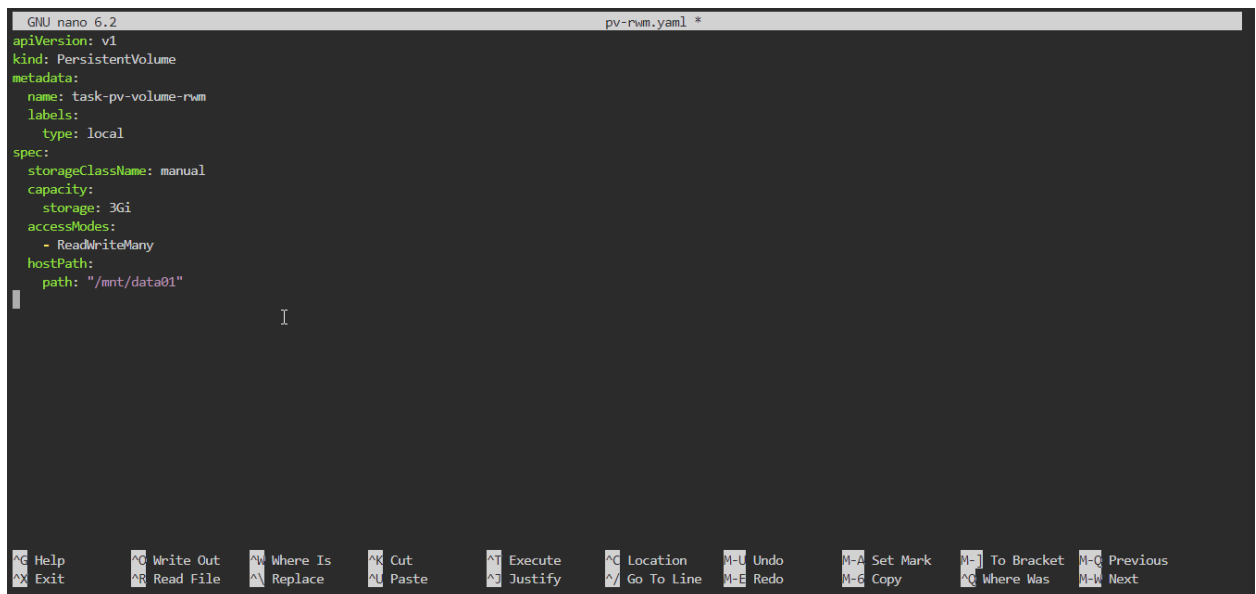
Step 1: Create PersistentVolume

- 1.1 Create the YAML file using the command below:
nano pv-rwm.yaml

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

1.2 Add the following code to the **pv-rwm.yaml** file to create the pod:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume-rwm
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data01"
```



```
GNU nano 6.2                                pv-rwm.yaml *
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume-rwm
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data01"
I

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

1.3 Use the **cat** command to validate the content of the **pv-rwm.yaml** file

```
labsuser@master:~$ nano pv-rwm.yaml
labsuser@master:~$ cat pv-rwm.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume-rwm
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data01"
labsuser@master:~$
```

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

```
labsuser@master:~$ cat pv-rwm.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume-rwm
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 3Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data01"
labsuser@master:~$ kubectl apply -f pv-rwm.yaml
persistentvolume/task-pv-volume-rwm created
labsuser@master:~$
```

1.5 List all PVs in the Kubernetes cluster using the following command:

kubectl get pv

```
labsuser@master:~$ kubectl apply -f pv-rwm.yaml
persistentvolume/task-pv-volume-rwm created
labsuser@master:~$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
task-pv-volume-rwm	3Gi	RWX	Retain	Available		manual		76s
test	10Gi	RWX	Retain	Bound	default/mypvc1			75m

```
labsuser@master:~$
```

Step 2: Create PersistentVolumeClaim

2.1 Create the YAML file using the following command:

nano pvc.yaml

```
labsuser@master:~$ kubectl apply -f pv-rwm.yaml
persistentvolume/task-pv-volume-rwm created
labsuser@master:~$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
task-pv-volume-rwm	3Gi	RWX	Retain	Available		manual		76s
test	10Gi	RWX	Retain	Bound	default/mypvc1			75m

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

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

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: task-pv-claim

spec:

storageClassName: manual

accessModes:

- ReadWriteMany

resources:

requests:

storage: 3Gi

```
GNU nano 6.2 pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim-rwm
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 3Gi
```

2.3 Use the **cat** command to validate the content of the **pvc.yaml** file

```
labsuser@master:~$ nano pvc.yaml
labsuser@master:~$ cat pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim-rwm
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 3Gi
labsuser@master:~$
```

2.4 Apply the Kubernetes resource configuration defined in the **pvc.yaml** file to create the PVC using the following command:

kubectl apply -f pvc.yaml

```
requests:
  storage: 3Gi
labsuser@master:~$ kubectl apply -f pvc.yaml
persistentvolumeclaim/task-pv-claim-rwm created
labsuser@master:~$
```

2.5 List all PVs in the Kubernetes cluster using the following command:

kubectl get pv

```
labsuser@master:~$ kubectl apply -f pvc.yaml
persistentvolumeclaim/task-pv-claim-rwm created
labsuser@master:~$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
task-pv-volume-rwm	3Gi	RWX	Retain	Bound	default/task-pv-claim-rwm	manual		21m
test	10Gi	RWX	Retain	Bound	default/mypvc1			95m

```
labsuser@master:~$
```

2.6 List all PVCs in the Kubernetes cluster using the following command:

kubectl get pvc

```
labsuser@master:~$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
task-pv-volume-rwm	3Gi	RWX	Retain	Bound	default/task-pv-claim-rwm	manual		21m
test	10Gi	RWX	Retain	Bound	default/mypvc1			95m

```
labsuser@master:~$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
mypvc1	Bound	test	10Gi	RWX		3d
task-pv-claim-rwm	Bound	task-pv-volume-rwm	3Gi	RWX	manual	12m

```
labsuser@master:~$
```

Step 3: Deploy a pod in a new namespace

3.1 Create the YAML file using the following command:

nano pod-pvc.yaml

```
labsuser@master:~$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
mypvc1	Bound	test	10Gi	RWX		3d
task-pv-claim-rwm	Bound	task-pv-volume-rwm	3Gi	RWX	manual	12m

```
labsuser@master:~$ nano pod-pvc.yaml
```

3.2 Add the following code to the **pod-pvc.yaml** file:

apiVersion: v1

kind: Pod

metadata:

name: sharevol-rwm

spec:

containers:

- name: container1

image: centos:7

command:

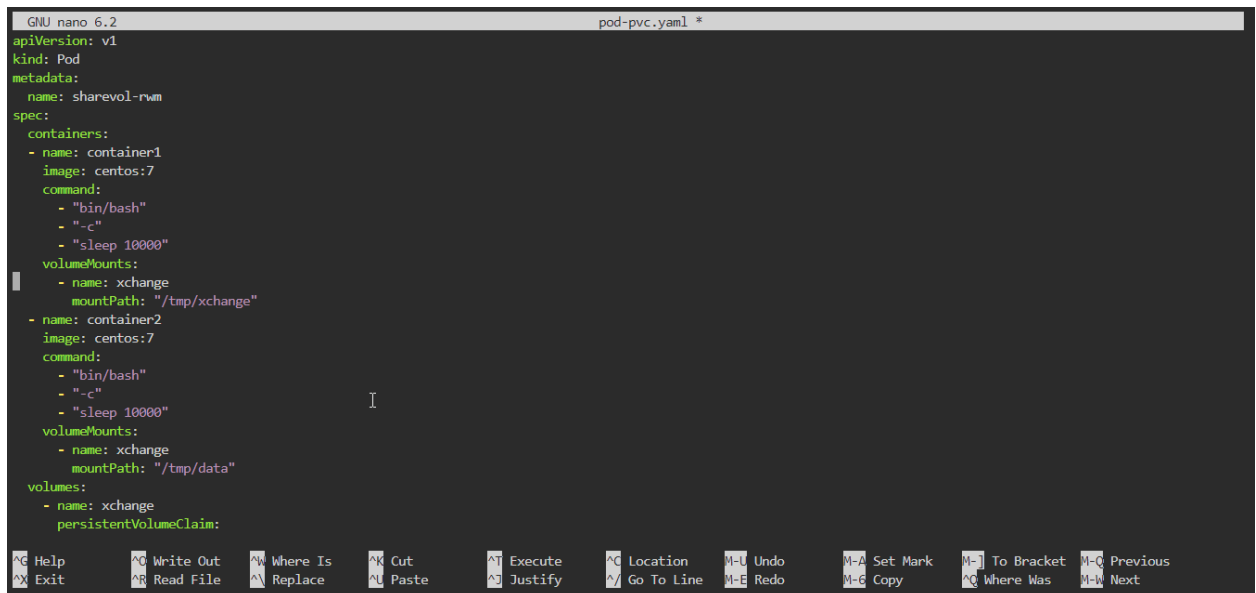
- "bin/bash"

- "-c"

```

- "sleep 10000"
volumeMounts:
  - name: xchange
    mountPath: "/tmp/xchange"
- name: container2
  image: centos:7
  command:
    - "bin/bash"
    - "-c"
    - "sleep 10000"
  volumeMounts:
    - name: xchange
      mountPath: "/tmp/data"
volumes:
  - name: xchange
    persistentVolumeClaim:
      claimName: task-pv-claim-rwm

```



```

GNU nano 6.2 pod-pvc.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: sharevol-rwm
spec:
  containers:
    - name: container1
      image: centos:7
      command:
        - "bin/bash"
        - "-c"
        - "sleep 10000"
      volumeMounts:
        - name: xchange
          mountPath: "/tmp/xchange"
    - name: container2
      image: centos:7
      command:
        - "bin/bash"
        - "-c"
        - "sleep 10000"
      volumeMounts:
        - name: xchange
          mountPath: "/tmp/data"
  volumes:
    - name: xchange
      persistentVolumeClaim:

```

3.3 Use the **cat** command to validate the content of the **pod-pvc.yaml** file

```
labsuser@master:~$ nano pod-pvc.yaml
labsuser@master:~$ cat pod-pvc.yaml
apiVersion: v1
kind: Pod
metadata:
  name: sharevol-rwm
spec:
  containers:
  - name: container1
    image: centos:7
    command:
      - "bin/bash"
      - "-c"
      - "sleep 10000"
    volumeMounts:
      - name: xchange
        mountPath: "/tmp/xchange"
  - name: container2
    image: centos:7
    command:
      - "bin/bash"
      - "-c"
      - "sleep 10000"
    volumeMounts:
      - name: xchange
        mountPath: "/tmp/data"
  volumes:
  - name: xchange
    persistentVolumeClaim:
      claimName: task-pv-claim-rwm
labsuser@master:~$
```

3.4 Apply the Kubernetes pod using the configuration defined in the **pod-pvc.yaml** file using the following command: **kubectl apply -f pod-pvc.yaml**

```
volumes:
  - name: xchange
    persistentVolumeClaim:
      claimName: task-pv-claim-rwm
labsuser@master:~$ kubectl apply -f pod-pvc.yaml
pod/sharevol-rwm configured
labsuser@master:~$
```


3.5 List all PVCs in the Kubernetes cluster using the following command:

kubectl get pod sharevol-rwm

```
labsuser@master:~$ kubectl apply -f pod-pvc.yaml
pod/sharevol-rwm configured
labsuser@master:~$ kubectl get pod sharevol-rwm
```

NAME	READY	STATUS	RESTARTS	AGE
sharevol-rwm	2/2	Running	0	3m8s

```
labsuser@master:~$
```

3.6 Retrieve detailed information about the pod using the following command:

kubectl describe pod sharevol-rwm

```
labsuser@master:~$ kubectl get pod sharevol-rwm
```

NAME	READY	STATUS	RESTARTS	AGE
sharevol-rwm	2/2	Running	0	3m8s

```
labsuser@master:~$ kubectl describe pod sharevol-rwm
```

```
Name: sharevol-rwm
Namespace: default
Priority: 0
Service Account: default
Node: worker-node-1.example.com/172.31.16.178
Start Time: Mon, 06 Nov 2023 11:53:44 +0000
Labels: <none>
Annotations: cni.projectcalico.org/containerID: c37db946af76337b52d2af9a9465c2f1b07deee6b0c974d842ddeac2297db0d3
              cni.projectcalico.org/podIP: 192.168.47.132/32
              cni.projectcalico.org/podIPs: 192.168.47.132/32
Status: Running
IP: 192.168.47.132
IPs:
  IP: 192.168.47.132
```

```
Volumes:
  xchange:
    Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName: task-pv-claim-rwm
    ReadOnly: false
  kube-api-access-hzqqt:
    Type: Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI: true
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
```

Type	Reason	Age	From	Message
Normal	Scheduled	5m16s	default-scheduler	Successfully assigned default/sharevol-rwm to worker-node-1.example.com
Normal	Pulling	5m16s	kubelet	Pulling image "centos:7"
Normal	Pulled	5m10s	kubelet	Successfully pulled image "centos:7" in 5.792s (5.792s including waiting)
Normal	Created	5m10s	kubelet	Created container container1
Normal	Started	5m10s	kubelet	Started container container1
Normal	Pulled	5m10s	kubelet	Container image "centos:7" already present on machine
Normal	Created	5m10s	kubelet	Created container container2
Normal	Started	5m9s	kubelet	Started container container2

```
labsuser@master:~$
```

Step 4: Demonstrate shared storage and data operations

- 4.1 Access the **sharevol-rwm** pod with **container1** and open a bash shell inside it using the following command:

```
kubectl exec -it sharevol-rwm -c container1 -- bash
```

```
Events:
  Type      Reason      Age   From              Message
  ----      -
  Normal    Scheduled   10m   default-scheduler Successfully assigned default/sharevol-rwm to worker-node-1.example.com
  Normal    Pulling     10m   kubelet           Pulling image "centos:7"
  Normal    Pulled      10m   kubelet           Successfully pulled image "centos:7" in 5.792s (5.792s including waiting)
  Normal    Created     10m   kubelet           Created container container1
  Normal    Started     10m   kubelet           Started container container1
  Normal    Pulled      10m   kubelet           Container image "centos:7" already present on machine
  Normal    Created     10m   kubelet           Created container container2
  Normal    Started     10m   kubelet           Started container container2
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container1 -- bash
[root@sharevol-rwm /]#
```

- 4.2 Navigate to the **/tmp/xchange** directory and create 10 text files named **container1-file1.txt** to **container1-file10.txt** using following commands:

```
cd /tmp/xchange
touch container1-file{1..10}.txt
```

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container1 -- bash
[root@sharevol-rwm /]# cd /tmp/xchange
[root@sharevol-rwm xchange]# touch container1-file{1..10}.txt
[root@sharevol-rwm xchange]#
```

- 4.3 List the contents of the current directory using the **ls** command

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container1 -- bash
[root@sharevol-rwm /]# cd /tmp/xchange
[root@sharevol-rwm xchange]# touch container1-file{1..10}.txt
[root@sharevol-rwm xchange]# ls
container1-file1.txt  container1-file2.txt  container1-file4.txt  container1-file6.txt  container1-file8.txt
container1-file10.txt container1-file3.txt  container1-file5.txt  container1-file7.txt  container1-file9.txt
[root@sharevol-rwm xchange]# exit
exit
labsuser@master:~$
```

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

4.4 Access the **sharevol-rwm** pod with **container2** and open a bash shell inside it using the following command:

kubectl exec -it sharevol-rwm -c container2 -- bash

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container1 -- bash
[root@sharevol-rwm /]# cd /tmp/xchange
[root@sharevol-rwm xchange]# touch container1-file{1..10}.txt
[root@sharevol-rwm xchange]# ls
container1-file1.txt  container1-file2.txt  container1-file4.txt  container1-file6.txt  container1-file8.txt
container1-file10.txt container1-file3.txt  container1-file5.txt  container1-file7.txt  container1-file9.txt
[root@sharevol-rwm xchange]# exit
exit
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container2 -- bash
[root@sharevol-rwm /]#
```

4.5 Change the directory to **/tmp/data** and list the contents of that directory using the following commands:

cd /tmp/data

ls

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container2 -- bash
[root@sharevol-rwm /]# cd /tmp/data
[root@sharevol-rwm data]# ls
container1-file1.txt  container1-file2.txt  container1-file4.txt  container1-file6.txt  container1-file8.txt
container1-file10.txt container1-file3.txt  container1-file5.txt  container1-file7.txt  container1-file9.txt
[root@sharevol-rwm data]#
```

Note: Stay in the same container without exiting

Step 5: Continue data operations

5.1 Create 10 text files named **container2-file1.txt** to **container2-file10.txt** and then list the contents of the directory using the following commands:

touch container2-file{1..10}.txt

ls

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container2 -- bash
[root@sharevol-rwm /]# cd /tmp/data
[root@sharevol-rwm data]# ls
container1-file1.txt  container1-file2.txt  container1-file4.txt  container1-file6.txt  container1-file8.txt
container1-file10.txt container1-file3.txt  container1-file5.txt  container1-file7.txt  container1-file9.txt
[root@sharevol-rwm data]# touch container2-file{1..10}.txt
[root@sharevol-rwm data]# ls
container1-file1.txt  container1-file3.txt  container1-file6.txt  container1-file9.txt  container2-file2.txt  container2-file5.txt  container2-file8.txt
container1-file10.txt container1-file4.txt  container1-file7.txt  container2-file1.txt  container2-file3.txt  container2-file6.txt  container2-file9.txt
container1-file2.txt  container1-file5.txt  container1-file8.txt  container2-file10.txt container2-file4.txt  container2-file7.txt
[root@sharevol-rwm data]#
```

- 5.2 Append the text **testing from container2** to the **container1-file.txt** file and then display the contents of **container1-file.txt** using the following commands:

```
echo "testing from container2" >> container1-file.txt
cat container1-file.txt
```

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container2 -- bash
[root@sharevol-rwm /]# cd /tmp/data
[root@sharevol-rwm data]# ls
container1-file1.txt  container1-file2.txt  container1-file4.txt  container1-file6.txt  container1-file8.txt
container1-file10.txt  container1-file3.txt  container1-file5.txt  container1-file7.txt  container1-file9.txt
[root@sharevol-rwm data]# touch container2-file{1..10}.txt
[root@sharevol-rwm data]# ls
container1-file1.txt  container1-file3.txt  container1-file6.txt  container1-file9.txt  container2-file2.txt  container2-file5.txt  container2-file8.txt
container1-file10.txt  container1-file4.txt  container1-file7.txt  container2-file1.txt  container2-file3.txt  container2-file6.txt  container2-file9.txt
container1-file2.txt  container1-file5.txt  container1-file8.txt  container2-file10.txt  container2-file4.txt  container2-file7.txt
[root@sharevol-rwm data]# echo "testing from container2" >> container1-file.txt
[root@sharevol-rwm data]# cat container1-file.txt
"testing from container2"
[root@sharevol-rwm data]# exit
exit
labsuser@master:~$
```

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

- 5.3 Access the **sharevol-rwm** pod with **container1** and open a bash shell inside it using the following command:

```
kubectl exec -it sharevol-rwm -c container1 -- bash
```

```
exit
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container1 -- bash
[root@sharevol-rwm /]#
```

- 5.4 Change the directory to **/tmp/xchange** and list the contents of that directory using the following commands:

```
cd /tmp/xchange
ls
```

```
labsuser@master:~$ kubectl exec -it sharevol-rwm -c container1 -- bash
[root@sharevol-rwm /]# cd /tmp/xchange
[root@sharevol-rwm xchange]# ls
container1-file.txt  container1-file2.txt  container1-file5.txt  container1-file8.txt  container2-file10.txt  container2-file4.txt  container2-file7.txt
container1-file1.txt  container1-file3.txt  container1-file6.txt  container1-file9.txt  container2-file2.txt  container2-file5.txt  container2-file8.txt
container1-file10.txt  container1-file4.txt  container1-file7.txt  container2-file1.txt  container2-file3.txt  container2-file6.txt  container2-file9.txt
[root@sharevol-rwm xchange]#
```

- 5.5 Count the number of files in the current directory using the following command:

```
ls | wc -l
```

```
container1-file10.txt  container1-file4.txt  container1-file7.txt  container2-file1.txt  container2-file3.txt  container2-file6.txt  container2-file9.txt
[root@sharevol-rwm xchange]# ls | wc -l
21
[root@sharevol-rwm xchange]#
```

5.6 Display the contents of **container1-file.txt** using the following command:
cat container1-file.txt

```
[root@sharevol-rwm xchange]# ls
container1-file.txt  container1-file2.txt  container1-file5.txt  container1-file8.txt  container2-file10.txt  container2-file4.txt  container2-file7.txt
container1-file1.txt  container1-file3.txt  container1-file6.txt  container1-file9.txt  container2-file2.txt  container2-file5.txt  container2-file8.txt
container1-file10.txt  container1-file4.txt  container1-file7.txt  container2-file1.txt  container2-file3.txt  container2-file6.txt  container2-file9.txt
[root@sharevol-rwm xchange]# ls | wc -l
21
[root@sharevol-rwm xchange]# cat container1-file.txt
"testing from container2"
[root@sharevol-rwm xchange]#
```

5.7 Append the text **testing from container1** to the **container2-file.txt** file using the following command:
echo "testing from container1" >> container2-file.txt

```
[root@sharevol-rwm xchange]# ls | wc -l
21
[root@sharevol-rwm xchange]# cat container1-file.txt
"testing from container2"
[root@sharevol-rwm xchange]# echo "testing from container1" >> container2-file.txt
[root@sharevol-rwm xchange]#
```

5.8 Display the contents of **container2-file.txt** using the following command:
cat container2-file.txt

```
[root@sharevol-rwm xchange]# ls | wc -l
21
[root@sharevol-rwm xchange]# cat container1-file.txt
"testing from container2"
[root@sharevol-rwm xchange]# echo "testing from container1" >> container2-file.txt
[root@sharevol-rwm xchange]# cat container2-file.txt
"testing from container1"
[root@sharevol-rwm xchange]# exit
exit
labsuser@master:~$
```

By following these steps, you have successfully set up a multi-container pod with RWX access in Kubernetes, demonstrating shared storage and data operations between containers.