Lesson 07 Demo 01

Sharing Data Between Containers in the Same Pod

Objective: To demonstrate data-sharing between containers in the same pod through hostPath volumes for mounting pod files onto the file system of the host node

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 and launch the pod with the shared volume
2. Interact with the shared volume from both containers
3. Test data persistence and sharing capability

Step 1: Configure and launch the pod with the shared volume

1.1 Open the YAML configuration file with the command:
nano emptydir.yaml

1.2 Enter the following code into the **emptydir.yaml** file to define the pod with two containers sharing a volume:

```
apiVersion: v1
kind: Pod
metadata:
  name: container-share-volume
spec:
  containers:
  - name: container1
    image: centos:7
    command:
      - "bin/bash"
      - "-c"
      - "sleep 10000"
    volumeMounts:
      - name: container-volume
        mountPath: "/tmp/xchange"
  - name: container2
    image: centos:7
    command:
      - "bin/bash"
      - "-c"
      - "sleep 10000"
    volumeMounts:
      - name: container-volume
        mountPath: "/tmp/data"
  volumes:
  - name: container-volume
    emptyDir: {}
```

```
 GNU nano 6.2                                                emptydir.yaml *
apiVersion: v1
kind: Pod
metadata:
  name: container-share-volume
spec:
  containers:
  - name: container1
    image: centos:7
    command:
      - "bin/bash"
      - "-c"
      - "sleep 10000"
    volumeMounts:
      - name: container-volume
        mountPath: "/tmp/xchange"

^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
```

```
 GNU nano 6.2                                                emptydir.yaml *
      - name: container-volume
        mountPath: "/tmp/xchange"
  - name: container2
    image: centos:7
    command:
      - "bin/bash"
      - "-c"
      - "sleep 10000"
    volumeMounts:
      - name: container-volume
        mountPath: "/tmp/data"
  volumes:
  - name: container-volume
    emptyDir: {}

^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
```

1.3 Launch the web service by applying the configuration with the command:
kubectl apply -f emptydir.yaml

```
labsuser@master:~$ kubectl apply -f emptydir.yaml
pod/container-share-volume created
labsuser@master:~$
```

1.4 Confirm that the pod is running and the volume is correctly shared:
kubectl describe pod container-share-volume

```
labsuser@master:~$ kubectl apply -f emptydir.yaml
pod/container-share-volume created
labsuser@master:~$ kubectl describe pod container-share-volume
Name:             container-share-volume
Namespace:        default
Priority:         0
Service Account:  default
Node:             worker-node-2.example.com/172.31.29.159
Start Time:       Thu, 02 Nov 2023 18:47:12 +0000
Labels:           <none>
Annotations:      cni.projectcalico.org/containerID: d763440455a308894d8378d1e952e5baf1ed41815dbd6f4b7d18ed00177c2b53
                  cni.projectcalico.org/podIP: 192.168.232.193/32
                  cni.projectcalico.org/podIPs: 192.168.232.193/32
Status:           Running
IP:               192.168.232.193
```

```
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  88s   default-scheduler  Successfully assigned default/container-share-volume to worker-node-2.example.com
  Normal  Pulling    87s   kubelet            Pulling image "centos:7"
  Normal  Pulled     81s   kubelet            Successfully pulled image "centos:7" in 5.852s (5.852s including waiting)
  Normal  Created    81s   kubelet            Created container container1
  Normal  Started    81s   kubelet            Started container container1
  Normal  Pulled     81s   kubelet            Container image "centos:7" already present on machine
  Normal  Created    81s   kubelet            Created container container2
  Normal  Started    81s   kubelet            Started container container2
labsuser@master:~$
```

Step 2: Interact with the shared volume from both containers

2.1 Open a shell session in **container1**:

kubectl exec -it container-share-volume -c container1 -- bash

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  34m   default-scheduler  Successfully assigned default/container-share-volume to worker-node-2.example.com
  Normal  Pulling    34m   kubelet            Pulling image "centos:7"
  Normal  Pulled     34m   kubelet            Successfully pulled image "centos:7" in 5.852s (5.852s including waiting)
  Normal  Created    34m   kubelet            Created container container1
  Normal  Started    34m   kubelet            Started container container1
  Normal  Pulled     34m   kubelet            Container image "centos:7" already present on machine
  Normal  Created    34m   kubelet            Created container container2
  Normal  Started    34m   kubelet            Started container container2
labsuser@master:~$ kubectl exec -it container-share-volume -c container1 -- bash
[root@container-share-volume /]#
```

2.2 Within the shell session, navigate to the shared volume and create files:

cd /tmp/xchange

touch container1-file{1..10}.txt

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

2.3 List the files to confirm their creation:

ls

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

Note: Exit the shell session with the **exit** command

2.4 Open a shell session in **container2**:

kubectl exec -it container-share-volume -c container2 -- bash

```
labsuser@master:~$ kubectl exec -it container-share-volume -c container1 -- bash
[root@container-share-volume /]# cd /tmp/xchange
[root@container-share-volume xchange]# touch container1-file{1..10}.txt
[root@container-share-volume 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@container-share-volume xchange]# exit
exit
labsuser@master:~$ kubectl exec -it container-share-volume -c container2 -- bash
[root@container-share-volume /]#
```

2.5 Verify that **container2** can see the files created by **container1**:

cd /tmp/data

ls

```
labsuser@master:~$ kubectl exec -it container-share-volume -c container2 -- bash
[root@container-share-volume /]# cd /tmp/data
[root@container-share-volume 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@container-share-volume data]#
```

Note: Remain in the shell session without exiting

Step 3: Test data persistence and sharing capability

3.1 Create additional files in container2:
touch container2-file{1..10}.txt

```
labsuser@master:~$ kubectl exec -it container-share-volume -c container2 -- bash
[root@container-share-volume /]# cd /tmp/data
[root@container-share-volume 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@container-share-volume data]# touch container2-file{1..10}.txt
[root@container-share-volume data]#
```

3.2 List the files to confirm their creation:
ls

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

3.3 Write to a file from **container2**:
echo “testing from container2” >> container1-file.txt

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

3.4 Read the content of the file to verify the write operation:
cat container1-file.txt

```
[root@container-share-volume data]# ls
container1-file1.txt   container1-file4.txt  container1-file8.txt   container2-file2.txt  container2-file6.txt
container1-file10.txt  container1-file5.txt  container1-file9.txt   container2-file3.txt  container2-file7.txt
container1-file2.txt   container1-file6.txt  container2-file1.txt   container2-file4.txt  container2-file8.txt
container1-file3.txt   container1-file7.txt  container2-file10.txt  container2-file5.txt  container2-file9.txt
[root@container-share-volume data]# echo “testing from container2” >> container1-file.txt
[root@container-share-volume data]# cat container1-file.txt
“testing from container2”
[root@container-share-volume data]# exit
exit
labsuser@master:~$
```

Note: Exit the shell session with the **exit** command

3.5 Return to **container1**:
kubectl exec -it container-share-volume -c container1 – bash

```
[root@container-share-volume data]# cat container1-file.txt
“testing from container2”
[root@container-share-volume data]# exit
exit
labsuser@master:~$ kubectl exec -it container-share-volume -c container1 -- bash
[root@container-share-volume /]#
```

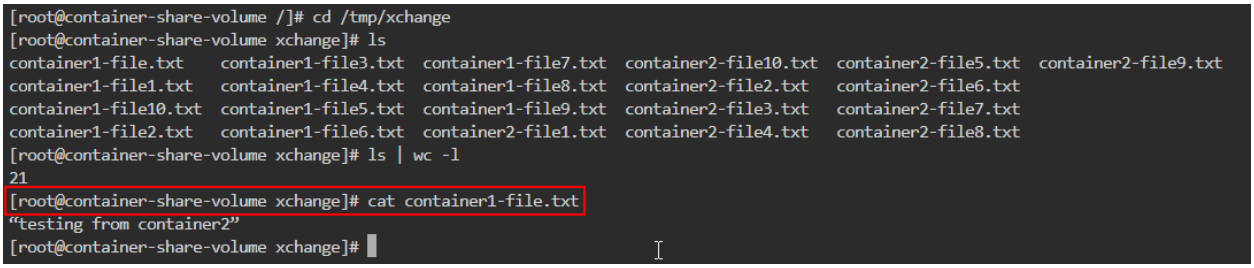
3.6 Validate that **container1** can see the changes:
cd /tmp/xchange
ls

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

3.7 Count the number of files:
ls | wc -l

```
[root@container-share-volume /]# cd /tmp/xchange
[root@container-share-volume xchange]# ls
container1-file.txt    container1-file3.txt  container1-file7.txt  container2-file10.txt  container2-file5.txt  container2-file9.txt
container1-file1.txt   container1-file4.txt  container1-file8.txt  container2-file2.txt   container2-file6.txt
container1-file10.txt  container1-file5.txt  container1-file9.txt  container2-file3.txt   container2-file7.txt
container1-file2.txt   container1-file6.txt  container2-file1.txt  container2-file4.txt   container2-file8.txt
[root@container-share-volume xchange]# ls | wc -l
21
[root@container-share-volume xchange]#
```

3.8 Read the contents of the file written by **container2**:
cat container1-file.txt



By following these steps, you have successfully demonstrated data-sharing between containers in the same pod using hostPath volumes in Kubernetes.