Version - **2025.01**

Last update : 2025/01/17 15:30

DOE306 - Volume Management in K8s

Contents

Overview

Volumes

The file system of a container in a pod is ephemeral, i.e. it exists only during the life cycle of the container. If the container is deleted or re-created, the

file system is lost.

Volumes enable data to be stored outside the container's file system, while still allowing the container to access it.

Persistent Volumes

A **Persistent Volume** is an abstract resource that can be *consumed* by pods. To access the Persistent Volume, the pod needs a **Persistent Volume Claim** to mount the Persistent Volume in the pod.

Volume types

Volumes and Persistent Volumes have a **Volume Type** (*Type of Volume*). The Volume Type determines the **Storage Method** of the data. Storage methods include:

- NFS,
- AWS,
- Azure,
- GCP,
- ConfigMaps,
- Secrets,

> **Important**: For more information on Storage Methods, see the K8s documentation **at this page**.

LAB #1 - Using K8s Volumes

1.1 - Volumes and volumeMounts

Volumes are configured in the **pod** specification, not the container. The two most important Volume Types are **hostPath** and **emptyDir**:

- **hostPath**,
  - Data is stored locally in a static directory on the K8s node,
- **emptyDir**,
  - Data is stored locally in a dynamic directory,
  - The directory exists **only** while the pod exists on the node,
  - K8s deletes the directory and data when the pod is deleted or moved,
  - This Volume Type is mainly used to share data between two containers in a pod.

A **volumeMount** is configured in the **container** specification, not the pod.

Start by creating the **volume.yaml** file:

To do: Copy the content from **here** **and paste it into your file.**

```
root@kubemaster:~# vi volume.yaml
root@kubemaster:~# cat volume.yaml
apiVersion: v1
kind: Pod
metadata:
  name: volumepod
spec:
  restartPolicy: Never
  containers:
  - name: busybox
    image: busybox
    command: ['sh', '-c', 'echo Success! > /output/success.txt']
    volumeMounts:
```

```
    - name: myvolume
      mountPath: /output
  volumes:
  - name: myvolume
    hostPath:
      path: /var/data
```

> **Important**: This pod will write the string **Success!** to the file **/output/success.txt** inside the container and then stop because the value of **restartPolicy** is **Never**. The **myvolume** volume will be mounted on **/output** in the container, thanks to the volumeMount configuration, and on **/var/data/** in the node hosting the pod.

Create the **volumepod** pod:

```
root@kubemaster:~# kubectl create -f volume.yaml
pod/volumepod created
```

Identify the node on which the pod runs:

```
root@kubemaster:~# kubectl get pod volumepod -o wide
NAME        READY   STATUS      RESTARTS   AGE     IP               NODE                        NOMINATED NODE
READINESS GATES
volumepod   0/1     Completed   0          3m10s   192.168.150.41   kubenode2.ittraining.loc   <none>
<none>
```

Connect to the identified node:

```
root@kubemaster:~# ssh -l trainee kubenode2
trainee@kubenode2's password: trainee
Linux kubenode2.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Sep 4 13:24:39 2022 from 192.168.56.2
```

Check for the presence and content of the file **/var/data/success.txt**:

```
trainee@kubenode2:~$ cat /var/data/success.txt
Success!
```

1.2 - Sharing volumes between containers

Return to the **kubemaster** VM:

```
trainee@kubenode2:~$ exit
logout
Connection to kubenode2 closed.
root@kubemaster:~#
```

Now create the **shared.yaml** file:

To do: Copy the content from **here** **and paste it into your file.**

```
root@kubemaster:~# vi shared.yaml
root@kubemaster:~# cat shared.yaml
apiVersion: v1
```

```
kind: Pod
metadata:
  name: sharedvolume
spec:
  containers:
  - name: busybox1
    image: busybox
    command: ['sh', '-c', 'while true; do echo Success! > /output/output.txt; sleep 5; done']
    volumeMounts:
      - name: myvolume
        mountPath: /output
  - name: busybox2
    image: busybox
    command: ['sh', '-c', 'while true; do cat /input/output.txt; sleep 5; done']
    volumeMounts:
    - name: myvolume
      mountPath: /input
  volumes:
  - name: myvolume
    emptyDir: {}
```

Important: This file will create two pods. The first, **busybox1**, will write the string **Success!** to the container's **/output/output.txt** file every 5 seconds. The **/output** directory is known as **myvolume**. This same volume will be available to the second pod container, **busybox2**, where it will be mounted at **/input**. The busybox2 container will print the contents of **/input/output.txt** on standard output every 5 seconds.

Create the two pods:

```
root@kubemaster:~# kubectl create -f shared.yaml
pod/sharedvolume created
```

Check that both pods are running:

```
root@kubemaster:~# kubectl get pods sharedvolume
NAME           READY   STATUS    RESTARTS   AGE
sharedvolume   2/2     Running   0          5m55s
```

Now look at the logs for the second container:

```
root@kubemaster:~# kubectl logs sharedvolume -c busybox2
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
Success!
```

Important: Note that busybox2 has printed the contents of the file **/input/output.txt** to its standard output.

LAB #2 - Persistent Volumes

2.1 - Storage Classes

- **StorageClassName**,
  - A StorageClassName is used to specify the **StorageClass**.
- **StorageClass**,
  - A StorageClass is used to specify the type of storage service used, e.g. local disk, cloud etc,
  - If the **allowVolumeExpansion** value is true and the storage service type allows it, a **PersistentVolumeClaim** can be hot-resized.

Create the **localdisk.yaml** file to define the **StorageClass** called **localdisk**:

To do: Copy the content from **here** **and paste it into your file.**

```
root@kubemaster:~# vi localdisk.yaml
root@kubemaster:~# cat localdisk.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: localdisk
provisioner: kubernetes.io/no-provisioner
allowVolumeExpansion: true
```

Important: Note that the allowVolumeExpansion value is true.

Create the StorageClass **localdisk** :

```
root@kubemaster:~# kubectl create -f localdisk.yaml
```

2.2 - Persistent Volumes

Create the **mypv.yaml** file to define the **PersistentVolume** called **mypv**:

To do: Copy the content from **here** **and paste it into your file.**

```
root@kubemaster:~# vi mypv.yaml
root@kubemaster:~# cat mypv.yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: mypv
spec:
  storageClassName: localdisk
  persistentVolumeReclaimPolicy: Recycle
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /var/output
```

Important: Note the value of accessModes.

There are four types of **accessModes**:

- **ReadWriteOnce** or **RWO**,
  - the volume can only be mounted by a single node,
- **ReadOnlyMany** or **ROX**,
  - the volume can be mounted read-only by several nodes,
- **ReadWriteMany** or **RWX**,
  - the volume can be mounted as read-write by several nodes,
- **ReadWriteOncePod** or **RWOP**,
  - the volume can only be mounted by a single pod.

> **Important**: Note that the persistentVolumeReclaimPolicy value is **Recycle**.

> **Important**: AccessMode availability depends on the type of storage service. The **ReadWriteOnce** mode is always available. For more information on accessModes, see **this page**.

There are three types of PersistentVolumeReclaimPolicy:

- **Retain**,
  - Data is not deleted when a PersistentVolumeClaim is deleted,
- **Delete**,
  - Automatically deletes the **storage resource** when a PersistentVolumeClaim is deleted. Note that **Delete** only works with public cloud services such as AWS, GCP etc,
- **Recycle**,
  - Automatically deletes data. Note that **Recycle** allows immediate reuse of storage resources freed up when a PersistentVolumeClaim is deleted.

Create the PersistentVolume **mypv**:

```
root@kubemaster:~# kubectl create -f mypv.yaml
```

```
persistentvolume/mypv created
```

Check the PersistentVolume status:

```
root@kubemaster:~# kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mypv   1Gi        RWO            Recycle          Available           localdisk               5m28s
```

Important: Note that the STATUS value is **Available**.

2.3 - Persistent Volume Claims

- A PersistentVolumeClaim represents a user's request for a storage resource,
  - The PersistentVolumeClaim specifies a **StorageClassName**, an **AccessMode** and a **size**,
- When created, the PersistentVolumeClaim searches for a **PersistentVolume** capable of satisfying the request,
  - If the result of the search is positive, the PersistentVolumeClaim is automatically linked to the PersistentVolume,
  - Otherwise, the PersistentVolumeClaim **remains on hold** until a PersistentVolume capable of satisfying the request is created,
  - To resize a PersistentVolumeClaim without interrupting service, change the value of **spec.resources.requests.storage** in the yaml file.

Create the **mypvc.yaml** file to define the **PersistentVolumeClaim** called **my-pvc**:

To do: Copy the content from **here** **and paste it into your file.**

```
root@kubemaster:~# vi mypvc.yaml
root@kubemaster:~# cat mypvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  name: my-pvc
spec:
  storageClassName: localdisk
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

Important: Note that the value of storageClassName is **localdisk**.

Create the PersistentVolumeClaim **my-pvc**:

```
root@kubemaster:~# kubectl create -f mypvc.yaml
persistentvolumeclaim/my-pvc created
```

Check the PersistentVolume status:

```
root@kubemaster:~# kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM            STORAGECLASS   REASON   AGE
mypv   1Gi        RWO            Recycle          Bound    default/my-pvc   localdisk               9m33s
```

Important: Note that the STATUS value is **Bound**. Also note that a PersistentVolume can only be associated with one PersistentVolumeClaim at a time.

Check the PersistentVolumeClaim status:

```
root@kubemaster:~# kubectl get pvc
NAME     STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
my-pvc   Bound    mypv     1Gi        RWO            localdisk      72s
```

Important: Note that the STATUS value is **Bound**.

2.4 - Using a PersistentVolumeClaim in a pod

Create the **mypvcpod.yaml** file to define the **pod** called **pv-pod**:

To do: Copy the content from **here** **and paste it into your file.**

```
root@kubemaster:~# vi mypvcpod.yaml
root@kubemaster:~# cat mypvcpod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: pv-pod
spec:
  restartPolicy: Never
  containers:
  - name: busybox
    image: busybox
    command: ['sh', '-c', 'echo Success! > /output/success.txt']
    volumeMounts:
    - name: pv-storage
      mountPath: /output
```

```
  volumes:
  - name: pv-storage
    persistentVolumeClaim:
      claimName: my-pvc
```

Create the **pv-pod** pod:

```
root@kubemaster:~# kubectl create -f mypvcpod.yaml
pod/pv-pod created
```

> **Important**: Note that the pod uses the persistentVolumeClaim **my-pvc** which is mounted on /output in the busybox container.

2.5 - Resizing a PersistentVolumeClaim

Modify the **storage:** value of the PersistentVolumeClaim:

```
root@kubemaster:~# kubectl edit pvc my-pvc --record
...
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 200Mi
  storageClassName: localdisk
  volumeMode: Filesystem
  volumeName: mypv
...
```

Save the modification:

```
root@kubemaster:~# kubectl edit pvc my-pvc --record
Flag --record has been deprecated, --record will be removed in the future
persistentvolumeclaim/my-pvc edited
```

Important: Note the edit confirmation message.

Delete the **pv-pod** pod and the **my-pvc** PersistentVolumeClaim:

```
root@kubemaster:~# kubectl delete pod pv-pod
pod “pv-pod” deleted

root@kubemaster:~# kubectl delete pvc my-pvc
persistentvolumeclaim “my-pvc” deleted
```

Check the PersistentVolume status:

```
root@kubemaster:~# kubectl get pv
NAME   CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mypv   1Gi        RWO            Recycle          Available           localdisk                23m
```

Important: Note that the STATUS value is Available again.
