Lab: Working with emptyDir

Introduction:

An **emptyDir** volume is first created when a Pod is assigned to a node, and exists as long as that Pod is running on that node. As the name says, the emptyDir volume is initially empty.

All containers in the Pod can read and write the same files in the emptyDir volume, though that volume can be mounted at the same or different paths in each container. When a Pod is removed from a node for any reason, the data in the emptyDir is deleted permanently.

Objective:

- **Basic emptyDir Example**
- **Pod with 3 Containers Sharing emptyDir**
- **Cleanup**

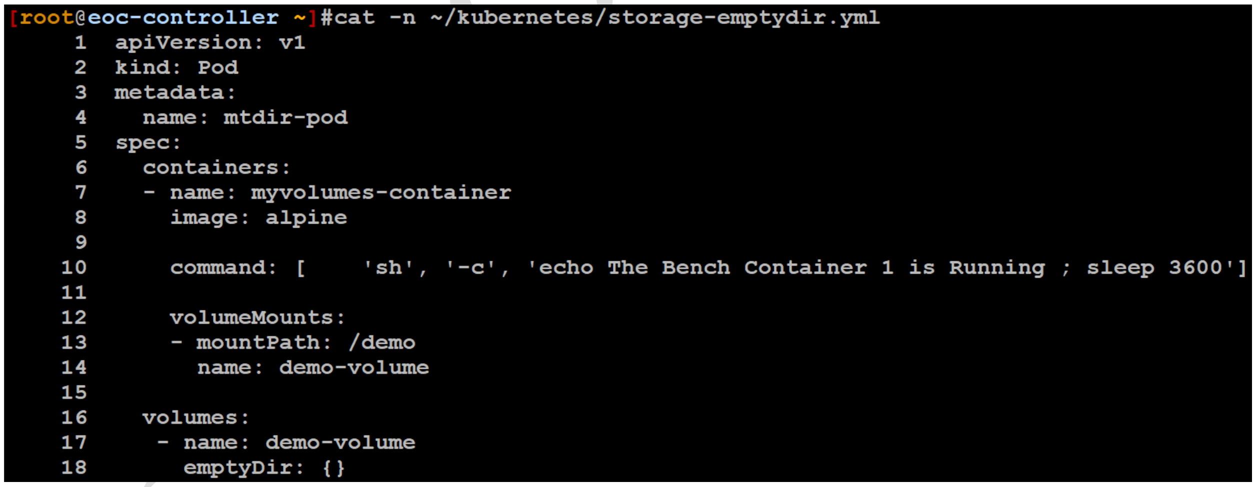
Ensure that you have logged-in as **root** user on **eoc-controller** node.

1 Basic emptyDir example

1.1 Let's **view** the yaml manifest file by executing the below command.

```
# cat -n ~/kubernetes/storage-emptydir.yml
```

Output:

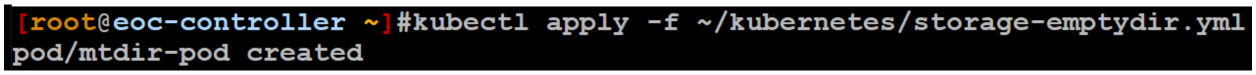


```
[root@eoc-controller ~]#cat -n ~/kubernetes/storage-emptydir.yml
apiVersion: v1
kind: Pod
metadata:
  name: mtdir-pod
spec:
  containers:
  - name: myvolumes-container
    image: alpine

    command: [    'sh', '-c', 'echo The Bench Container 1 is Running ; sleep 3600']

    volumeMounts:
    - mountPath: /demo
      name: demo-volume

  volumes:
   - name: demo-volume
     emptyDir: {}
```

1.2 Let's **create** the **Volume-emptyDir** with help of "storage-emptydir.yml" by executing the below command.

```
# kubectl apply -f ~/kubernetes/storage-emptydir.yml
```

Output:

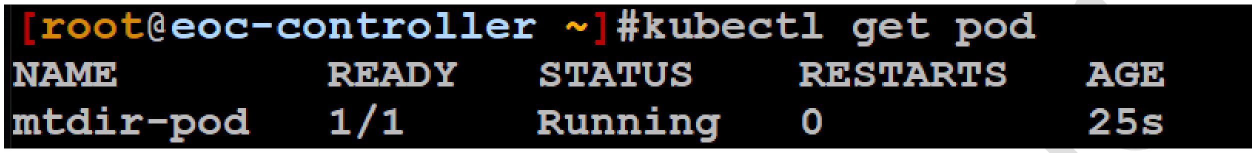


```
[root@eoc-controller ~]#kubectl apply -f ~/kubernetes/storage-emptydir.yml
pod/mtdir-pod created
```

1.3 Let’s **list** the status of pods by executing the below command.

```
# kubectl get pod
```

Output:

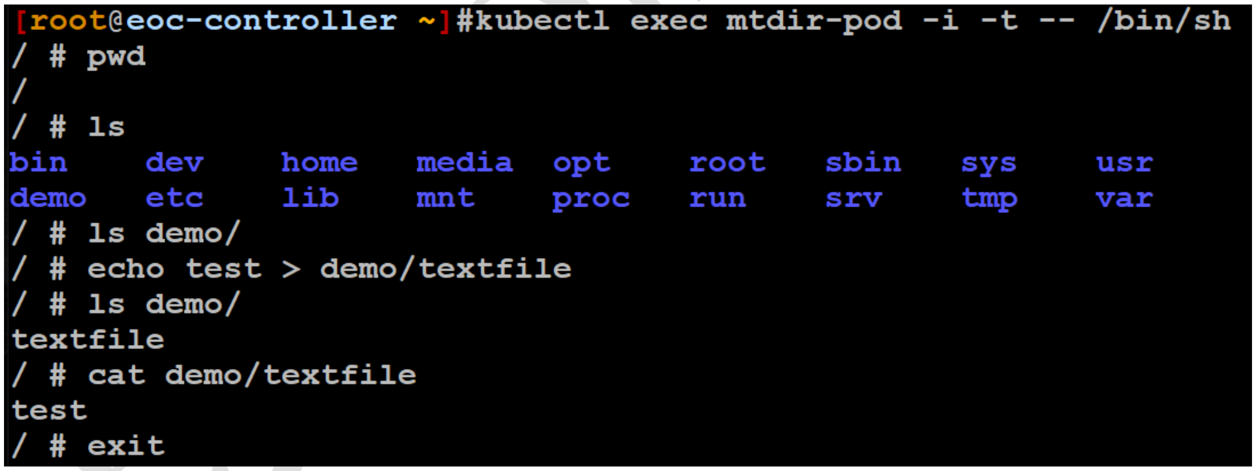


```
[root@eoc-controller ~]#kubectl get pod
NAME          READY    STATUS     RESTARTS    AGE
mtdir-pod     1/1      Running    0           25s
```

1.4 Let’s dig **inside** the pod analyses the volume behavior.

```
# kubectl exec mtdir-pod -i -t -- /bin/sh
# pwd
# ls
# ls demo/
# echo test > demo/textfile
# ls demo/
# cat demo/textfile
# exit
```

Output:

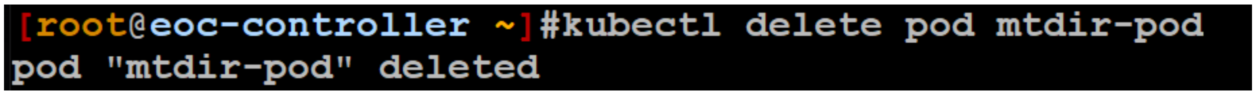


```
[root@eoc-controller ~]#kubectl exec mtdir-pod -i -t -- /bin/sh
/ # pwd
/
/ # ls
bin    dev    home   media  opt    root   sbin   sys    usr
demo   etc    lib    mnt    proc   run    srv    tmp    var
/ # ls demo/
/ # echo test > demo/textfile
/ # ls demo/
textfile
/ # cat demo/textfile
test
/ # exit
```

1.5 Let’s **delete** pod **mtdir-pod** by executing the below command.

```
# kubectl delete pod mtdir-pod
```

Output:



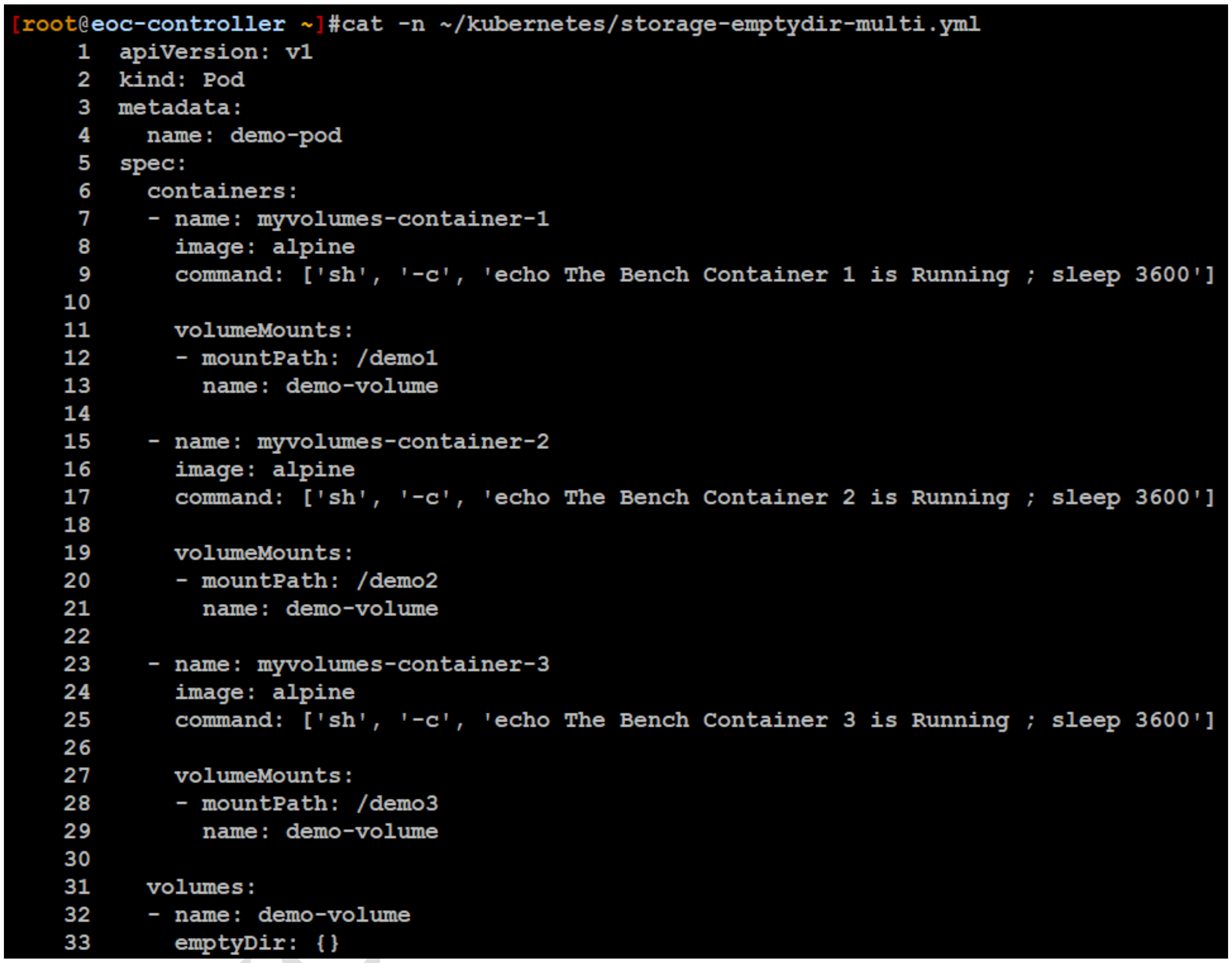
```
[root@eoc-controller ~]#kubectl delete pod mtdir-pod
pod "mtdir-pod" deleted
```

2 Pod with 3 containers sharing emptyDir

2.1 Let's **view** the yaml manifest file by executing below command.

```
# cat -n ~/kubernetes/storage-emptydir-multi.yml
```

Output:

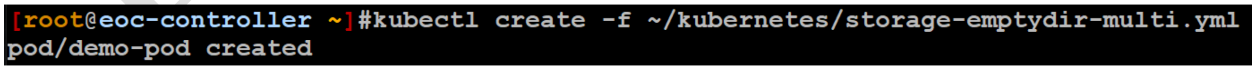


```
[root@eoc-controller ~]#cat -n ~/kubernetes/storage-emptydir-multi.yml
     1  apiVersion: v1
     2  kind: Pod
     3  metadata:
     4    name: demo-pod
     5  spec:
     6    containers:
     7    - name: myvolumes-container-1
     8      image: alpine
     9      command: ['sh', '-c', 'echo The Bench Container 1 is Running ; sleep 3600']
    10
    11      volumeMounts:
    12      - mountPath: /demo1
    13        name: demo-volume
    14
    15    - name: myvolumes-container-2
    16      image: alpine
    17      command: ['sh', '-c', 'echo The Bench Container 2 is Running ; sleep 3600']
    18
    19      volumeMounts:
    20      - mountPath: /demo2
    21        name: demo-volume
    22
    23    - name: myvolumes-container-3
    24      image: alpine
    25      command: ['sh', '-c', 'echo The Bench Container 3 is Running ; sleep 3600']
    26
    27      volumeMounts:
    28      - mountPath: /demo3
    29        name: demo-volume
    30
    31    volumes:
    32    - name: demo-volume
    33      emptyDir: {}
```

2.2 Let's **create** multi pods by executing the below command.

```
# kubectl create -f ~/kubernetes/storage-emptydir-multi.yml
```

Output:

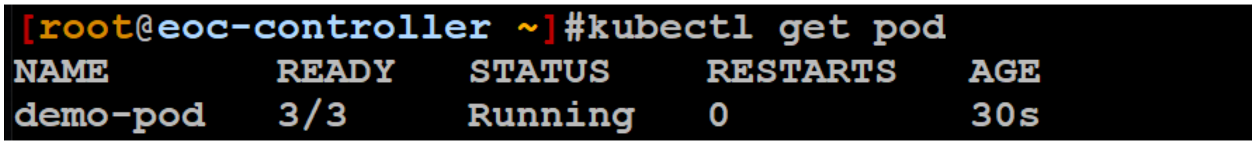


```
[root@eoc-controller ~]#kubectl create -f ~/kubernetes/storage-emptydir-multi.yml
pod/demo-pod created
```

2.3 Let's **list** the pods by executing the below command

```
# kubectl get pod
```

Output:

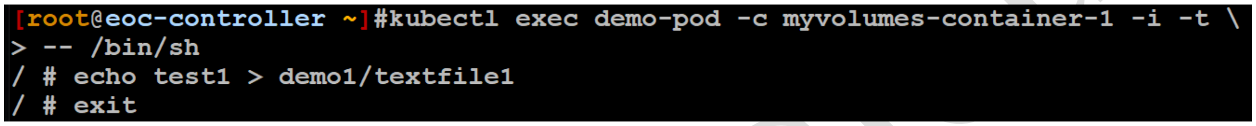


```
[root@eoc-controller ~]#kubectl get pod
NAME         READY    STATUS     RESTARTS    AGE
demo-pod     3/3      Running    0           30s
```

2.4 Let's **dig inside** the pod to **verify** the volume status in container-1.

```
# kubectl exec demo-pod -c myvolumes-container-1 -i -t \
-- /bin/sh
# echo test1 > demo1/textfile1
# exit
```

Output:



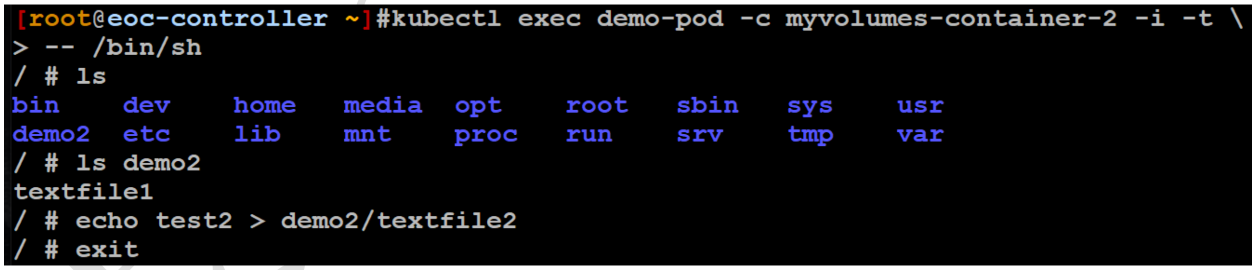
```
[root@eoc-controller ~]#kubectl exec demo-pod -c myvolumes-container-1 -i -t \
> -- /bin/sh
/ # echo test1 > demo1/textfile1
/ # exit
```

Note: Mount point demo1 exists and we could create a file there.

2.5 Let's enter the **container 2** and **create** a file at its mount point as shown below.

```
#  kubectl exec demo-pod -c myvolumes-container-2 -i -t \
-- /bin/sh
# ls
# ls demo2
# echo test2 > demo2/textfile2
# exit
```

Output:



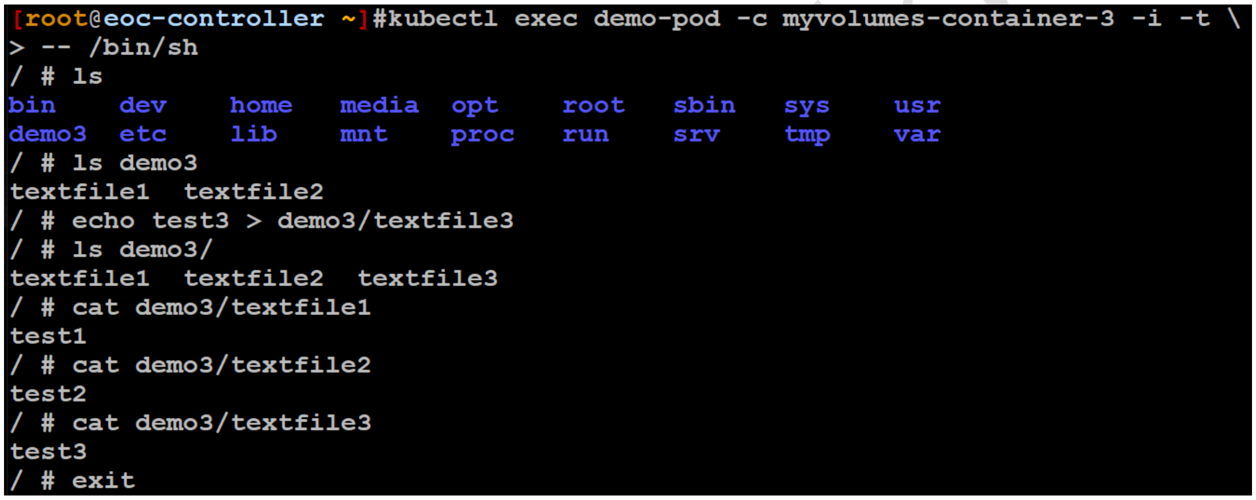
```
[root@eoc-controller ~]#kubectl exec demo-pod -c myvolumes-container-2 -i -t \
> -- /bin/sh
/ # ls
bin     dev     home    media   opt     root    sbin    sys     usr
demo2   etc     lib     mnt     proc    run     srv     tmp     var
/ # ls demo2
textfile1
/ # echo test2 > demo2/textfile2
/ # exit
```

Note: Note ls demo2/ shows the file container 1 created

2.6 Let's enter the **container 3** and **create** a file at its mount point as shown below.

```
#  kubectl exec demo-pod -c myvolumes-container-3 -i -t \
-- /bin/sh
# ls
# ls demo3
# echo test3 > demo3/textfile3
# ls demo3/
# cat demo3/textfile1
# cat demo3/textfile2
# cat demo3/textfile3
# exit
```

Output:



```
[root@eoc-controller ~]#kubectl exec demo-pod -c myvolumes-container-3 -i -t \
> -- /bin/sh
/ # ls
bin    dev    home   media  opt    root   sbin   sys    usr
demo3  etc    lib    mnt    proc   run    srv    tmp    var
/ # ls demo3
textfile1  textfile2
/ # echo test3 > demo3/textfile3
/ # ls demo3/
textfile1  textfile2  textfile3
/ # cat demo3/textfile1
test1
/ # cat demo3/textfile2
test2
/ # cat demo3/textfile3
test3
/ # exit
```

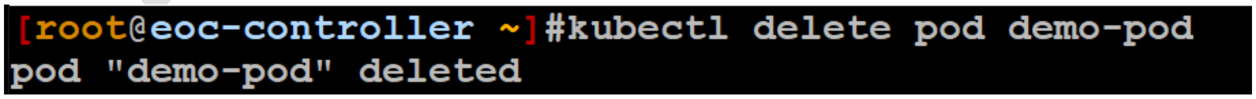
Note: All containers in a Pod have read/write access to the same emptyDir - if they requested a mount point for it. Containers can access the emptyDir using the same or different mount points.

3 Cleanup

3.1 Let's **delete** the pods by executing below command.

```
# kubectl delete pod demo-pod
```

Output:



```
[root@eoc-controller ~]#kubectl delete pod demo-pod
pod "demo-pod" deleted
```