Module 5: Managing State with Deployments

DEMO-3

DEMO Steps:

Using Persistent Volume and Persistent Volume Claims

1. Create a new YAML file to create a Persistent Volume

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: test-vp
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
  - ReadWriteMany
  hostPath:
    path: "/home/ubuntu/data"
```

2. Deploy the Persistent Volume

```
ubuntu@kmaster:~$ kubectl create -f pv.yaml
persistentvolume/test-vp created
ubuntu@kmaster:~$ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
test-vp   1Gi        RWX            Retain           Available           manual                  4s
```

3. Create another yaml file for your Persistent Volume Claim

```
kind: PersistentVolumeClaim
apiversion: v1
metadata:
  name: test-vpc
  labels:
    type: local
spec:
  storageClassName: manual
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
```

4. Deploy the persistentVolumeClaim. It will automatically bind itself to the persistent volume

```
ubuntu@kmaster:~$ kubectl create -f pvc.yaml
persistentvolumeclaim/test-vpc created
ubuntu@kmaster:~$ kubectl get pvc
NAME       STATUS    VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
test-vpc   Bound     test-vp   1Gi        RWX            manual         5s
```

5. Now create a new deployment yaml file to mount the persistent volume

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: httpd
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpd
  template:
    metadata:
      labels:
        app: httpd
    spec:
      volumes:
      - name: test
        persistentVolumeClaim:
          claimName: test-vpc
      containers:
      - name: httpd
        image: httpd
        ports:
        - containerPort: 80
```

6. Create the deployment and curl the IP address of the pod created

```
ubuntu@kmaster:~$ kubectl create -f deploy.yaml
deployment.extensions/httpd created
ubuntu@kmaster:~$ kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
httpd-6d897df555-nnnrs   1/1     Running   0          5s
```

```
admin@ip-172-20-35-51:~$ curl 100.96.2.3
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
 <head>
  <title>Index of /</title>
 </head>
 <body>
<h1>Index of /</h1>
<ul></ul>
</body></html>
```

7. Now change the index.html file inside /usr/local/apache2/htdocs by accessing the container

   Syntax: kubectl exec -it <containerID> bash

```
ubuntu@kmaster:~$ kubectl exec -it httpd-6d897df555-nnnrs bash
root@httpd-6d897df555-nnnrs:/usr/local/apache2# cd htdocs/
root@httpd-6d897df555-nnnrs:/usr/local/apache2/htdocs# echo "Happy Learning" > index.html
root@httpd-6d897df555-nnnrs:/usr/local/apache2/htdocs# cat index.html
Happy Learning
```

8. If we curl the container from outside we can see that it writes the new message

```
admin@ip-172-20-35-51:~$ curl 100.96.2.3
Happy Learning
```

9. Now to verify, delete the current pod and let the deployment generate a new pod. Then curl the IP address of the new pod

```
ubuntu@kmaster:~$ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
httpd-6d897df555-nnnrs     1/1     Running   0          52m
ubuntu@kmaster:~$ kubectl delete pod httpd-6d897df555-nnnrs
pod "httpd-6d897df555-nnnrs" deleted
admin@ip-172-20-35-51:~$ kubectl get pods -o wide
NAME                       READY   STATUS    RESTARTS   AGE   IP           NODE
httpd-6d897df555-xvxvh     1/1     Running   0          21s   100.96.2.4   ip-172-20-57-161.us-east-2.compute.internal
admin@ip-172-20-35-51:~$ curl 100.96.2.4
Happy Learning
```