QUES 3 : PersistentVolume (PV) and PersistentVolumeClaim (PVC)

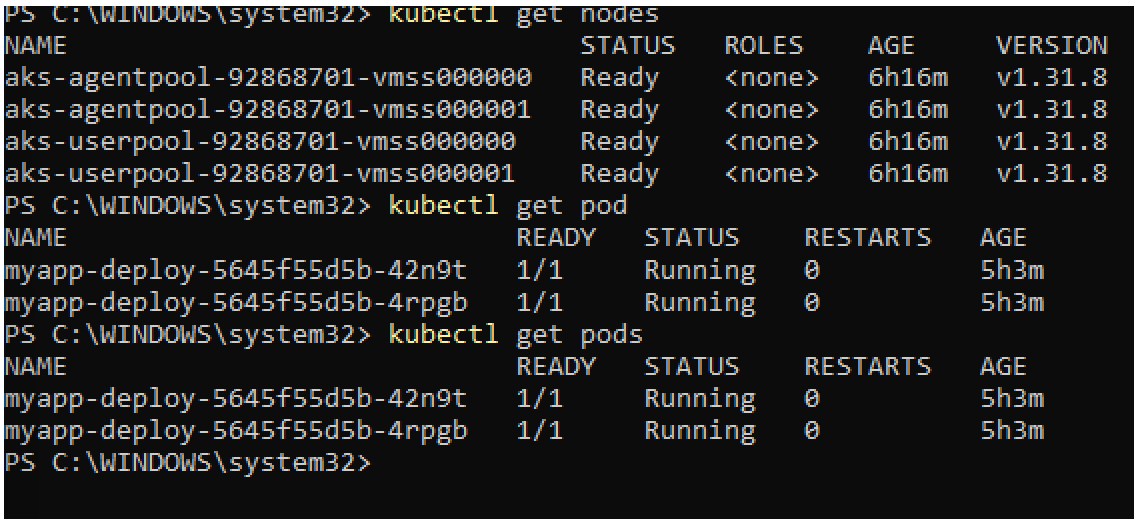
SOLN:

In Kubernetes, a **PersistentVolume (PV)** is a piece of storage in the cluster that has been provisioned by an admin or dynamically provisioned using StorageClasses. A **PersistentVolumeClaim (PVC)** is a request for that storage by a user. This mechanism allows containers to store data persistently, even across pod restarts or rescheduling.

In simple words, **PersistentVolume (PV)** is like a hard disk that's kept aside for use in your Kubernetes system. And **PersistentVolumeClaim (PVC)** is like a request your app makes saying, "Hey, I need some space to store my data."

Normally, when a pod stops or restarts, it loses all its data. But when you use PV and PVC, your app gets permanent storage — like saving your files on a pen drive instead of RAM, so they don't get lost.

In the previous tasks (Q1 & Q2), I successfully deployed a sample application (`myapp-deploy`) and exposed it via a Kubernetes service. Now, in this task (Q3), I am implementing **persistent storage** using **PersistentVolume (PV)** and **PersistentVolumeClaim (PVC)** to ensure the application can store and retain data even if the pod restarts or gets rescheduled.



```
PS C:\WINDOWS\system32> kubectl get nodes
NAME                                STATUS   ROLES    AGE     VERSION
aks-agentpool-92868701-vmss000000   Ready    <none>   6h16m   v1.31.8
aks-agentpool-92868701-vmss000001   Ready    <none>   6h16m   v1.31.8
aks-userpool-92868701-vmss000000    Ready    <none>   6h16m   v1.31.8
aks-userpool-92868701-vmss000001    Ready    <none>   6h16m   v1.31.8
PS C:\WINDOWS\system32> kubectl get pod
NAME                            READY   STATUS    RESTARTS   AGE
myapp-deploy-5645f55d5b-42n9t   1/1     Running   0          5h3m
myapp-deploy-5645f55d5b-4rpgb   1/1     Running   0          5h3m
PS C:\WINDOWS\system32> kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
myapp-deploy-5645f55d5b-42n9t   1/1     Running   0          5h3m
myapp-deploy-5645f55d5b-4rpgb   1/1     Running   0          5h3m
PS C:\WINDOWS\system32>
```

ALL THE DEPLOYMENTS ARE WORKING FINE

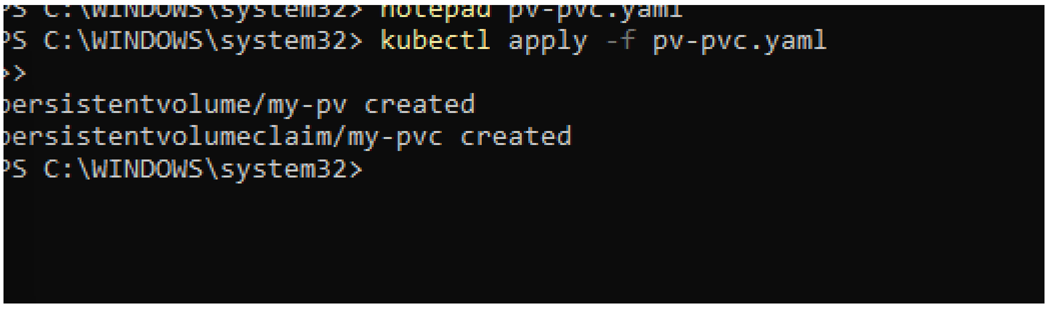
LETS CREATE A YAML FILE TO CONFIGURE PV & PVC FOR THE PROJECT:

pv-pvc.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  storageClassName: manual
  hostPath:
    path: "/mnt/data"  # For demo/testing only
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: manual
  resources:
    requests:
      storage: 1Gi
```

lets apply it :

```
kubectl apply -f pv-pvc.yaml
```



```
PS C:\WINDOWS\system32> notepad pv-pvc.yaml
PS C:\WINDOWS\system32> kubectl apply -f pv-pvc.yaml
>>
persistentvolume/my-pv created
persistentvolumeclaim/my-pvc created
PS C:\WINDOWS\system32>
```

PVC AND PV HAS BEEN SUCCESSFULLY CREATED

LETS TRY TO MOUNT THESE PVC AND PV TO OUR PREVIOUS BUILT DEPLOYMENT

STEP : MODIFY THE DEPLOYMENT.YAML FILE

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deploy
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
        image: prateek2004/my-frontend
        ports:
        - containerPort: 80
        volumeMounts:
        - name: app-storage
          mountPath: /app/data   # ✅ You can change this to wherever your app writes
files
      volumes:
      - name: app-storage
        persistentVolumeClaim:
          claimName: my-pvc
```

NOTE:

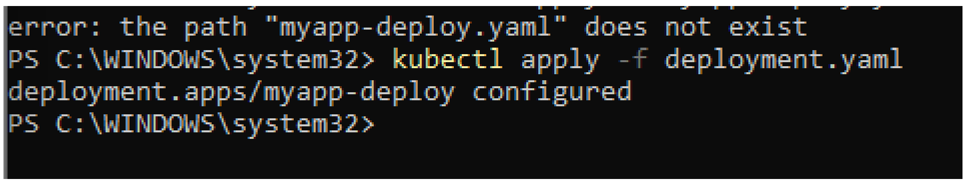
`volumeMounts` tells your container to use storage inside the container at `/app/data`.

- `volumes` connects the PVC (`my-pvc`) to the pod.
- You can change `/app/data` to wherever your frontend app writes data (like user uploads, logs, etc.).

Save and close.

3. Then reapply it:

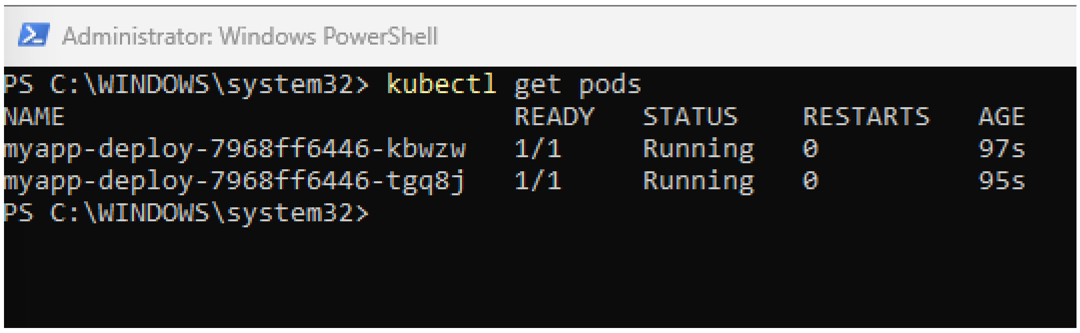
```
kubectl apply -f deployment.yaml
```



```
error: the path "myapp-deploy.yaml" does not exist
PS C:\WINDOWS\system32> kubectl apply -f deployment.yaml
deployment.apps/myapp-deploy configured
PS C:\WINDOWS\system32>
```

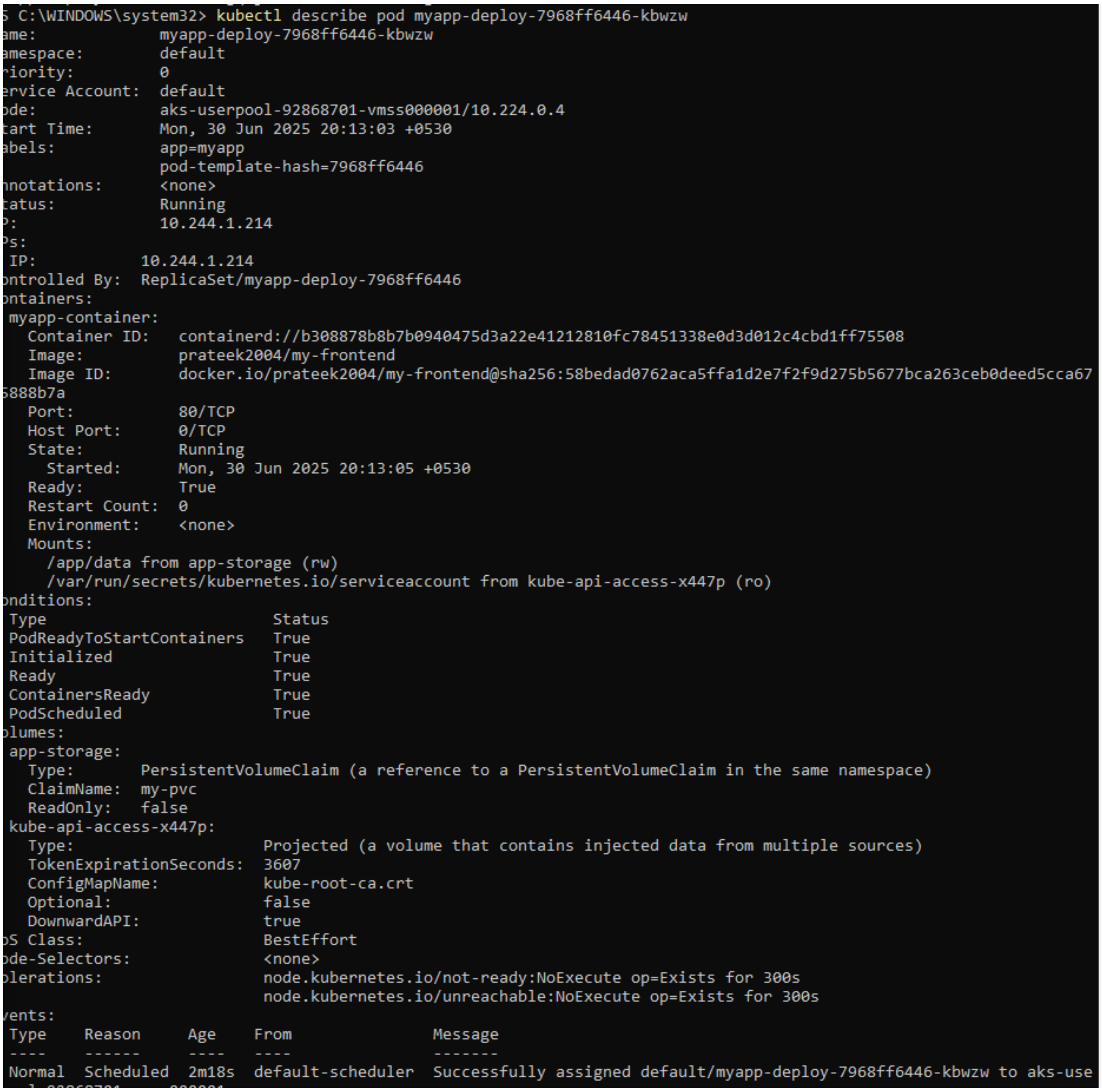
Verify It Worked:

kubectl get pods



```
PS C:\WINDOWS\system32> kubectl get pods
NAME                            READY   STATUS    RESTARTS   AGE
myapp-deploy-7968ff6446-kbwzw   1/1     Running   0          97s
myapp-deploy-7968ff6446-tgq8j   1/1     Running   0          95s
PS C:\WINDOWS\system32>
```

kubectl describe pod <your-pod-name>



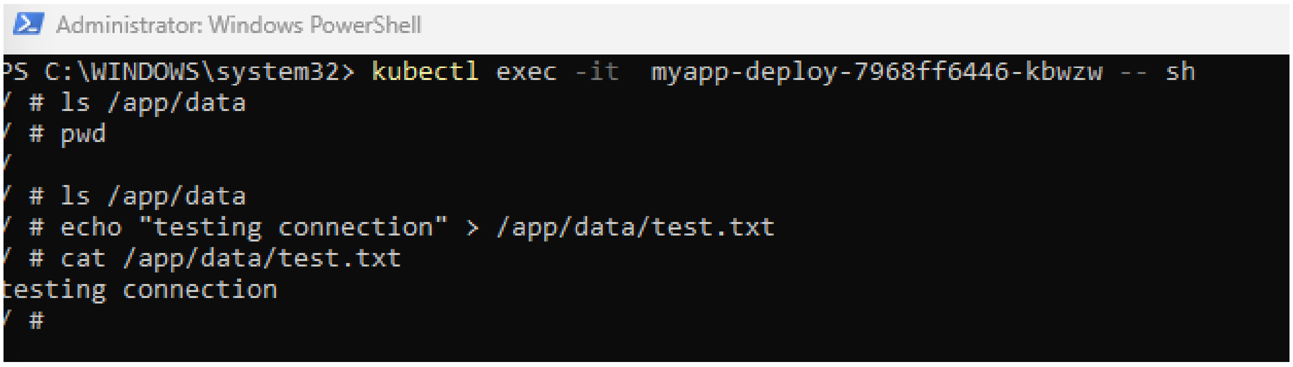
```
5 C:\WINDOWS\system32> kubectl describe pod myapp-deploy-7968ff6446-kbwzw
ame:             myapp-deploy-7968ff6446-kbwzw
amespace:        default
riority:         0
ervice Account:  default
ode:             aks-userpool-92868701-vmss000001/10.224.0.4
tart Time:       Mon, 30 Jun 2025 20:13:03 +0530
abels:           app=myapp
                 pod-template-hash=7968ff6446
nnotations:      <none>
tatus:           Running
P:               10.244.1.214
Ps:
 IP:           10.244.1.214
ontrolled By:  ReplicaSet/myapp-deploy-7968ff6446
ontainers:
 myapp-container:
   Container ID:   containerd://b308878b8b7b0940475d3a22e41212810fc78451338e0d3d012c4cbd1ff75508
   Image:          prateek2004/my-frontend
   Image ID:       docker.io/prateek2004/my-frontend@sha256:58bedad0762aca5ffa1d2e7f2f9d275b5677bca263ceb0deed5cca67
5888b7a
   Port:           80/TCP
   Host Port:      0/TCP
   State:          Running
     Started:      Mon, 30 Jun 2025 20:13:05 +0530
   Ready:          True
   Restart Count:  0
   Environment:    <none>
   Mounts:
     /app/data from app-storage (rw)
     /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-x447p (ro)
onditions:
 Type                        Status
 PodReadyToStartContainers   True
 Initialized                 True
 Ready                       True
 ContainersReady             True
 PodScheduled                True
olumes:
 app-storage:
   Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
   ClaimName:  my-pvc
   ReadOnly:   false
 kube-api-access-x447p:
   Type:                    Projected (a volume that contains injected data from multiple sources)
   TokenExpirationSeconds:  3607
   ConfigMapName:           kube-root-ca.crt
   Optional:                false
   DownwardAPI:             true
oS Class:                   BestEffort
ode-Selectors:              <none>
olerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                            node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
vents:
 Type    Reason     Age    From               Message
 ----    ------     ----   ----               -------
 Normal  Scheduled  2m18s  default-scheduler  Successfully assigned default/myapp-deploy-7968ff6446-kbwzw to aks-use
```

. we can see a successfull mount

kubectl exec -it <your-pod-name> -- sh

ls /app/data
echo "Testing PVC connection" > /app/data/test.txt
cat /app/data/test.txt

HENCE ITS SUCCESSFULLY VERIFIED AND TESTED:


Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> kubectl exec -it  myapp-deploy-7968ff6446-kbwzw -- sh
/ # ls /app/data
/ # pwd
/
/ # ls /app/data
/ # echo "testing connection" > /app/data/test.txt
/ # cat /app/data/test.txt
testing connection
/ #
```

In real-world applications, PersistentVolumes and PersistentVolumeClaims are used wherever long-term storage is required — such as databases (MySQL, MongoDB), file upload systems, media platforms, and content management systems like WordPress. Without persistent storage, all application data would be lost every time a pod is recreated or updated.