Persistent Volumes (PVs) in Kubernetes

Persistent Volumes (PVs) are a key component in Kubernetes for managing storage resources. They provide a way to abstract and manage storage independently of pods, ensuring data persistence even when pods are deleted or rescheduled.

Persistent Volume (PV)

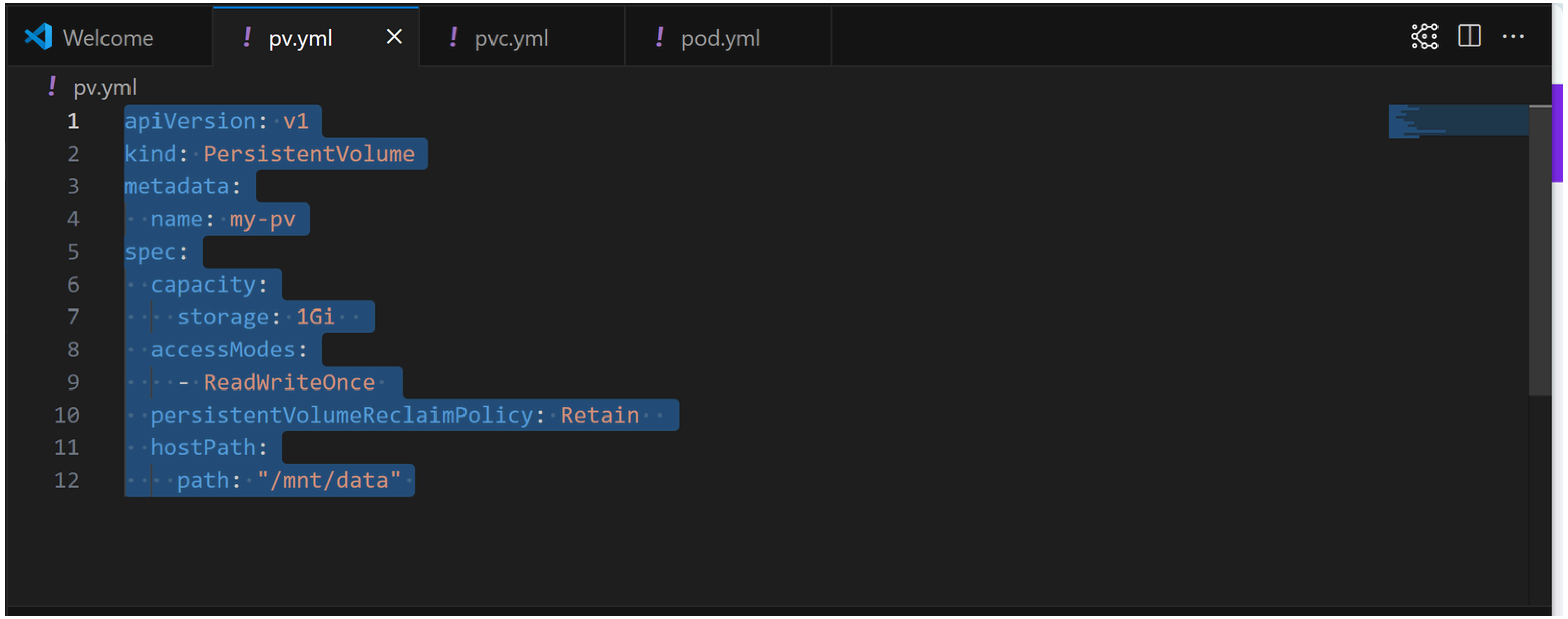
A cluster-wide storage resource provisioned by an administrator.
Represents a piece of storage (e.g., NFS, AWS EBS, GCP Persistent Disk, local storage).
Has a lifecycle independent of pods.

Persistent Volume Claim (PVC)

A request for storage by a user or application.
Binds to a PV that matches the requested size and access mode.
Acts as a "ticket" for pods to access the storage.

Access Modes

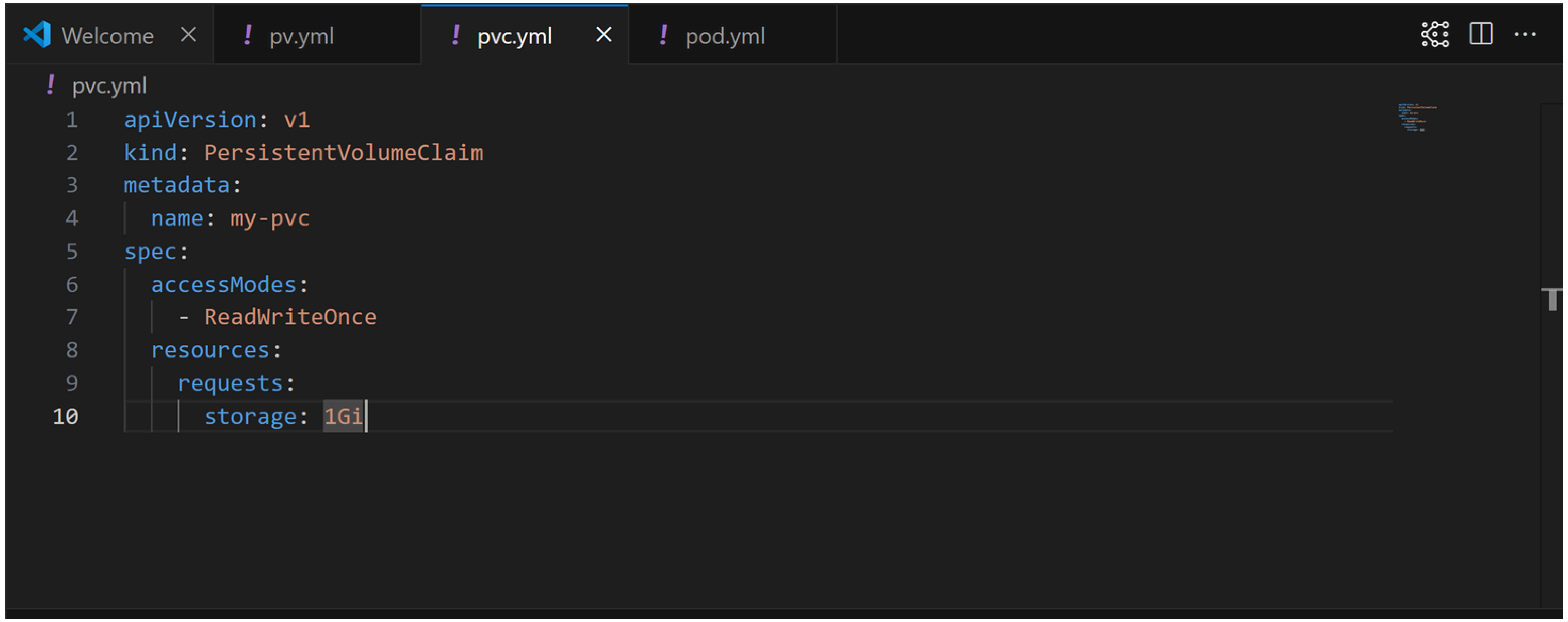
ReadWriteOnce (RWO): Read-write by a single node.
ReadOnlyMany (ROX): Read-only by multiple nodes.
ReadWriteMany (RWX): Read-write by multiple nodes



```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 1Gi  
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain  
  hostPath:
    path: "/mnt/data"
```

Created pv.yml file

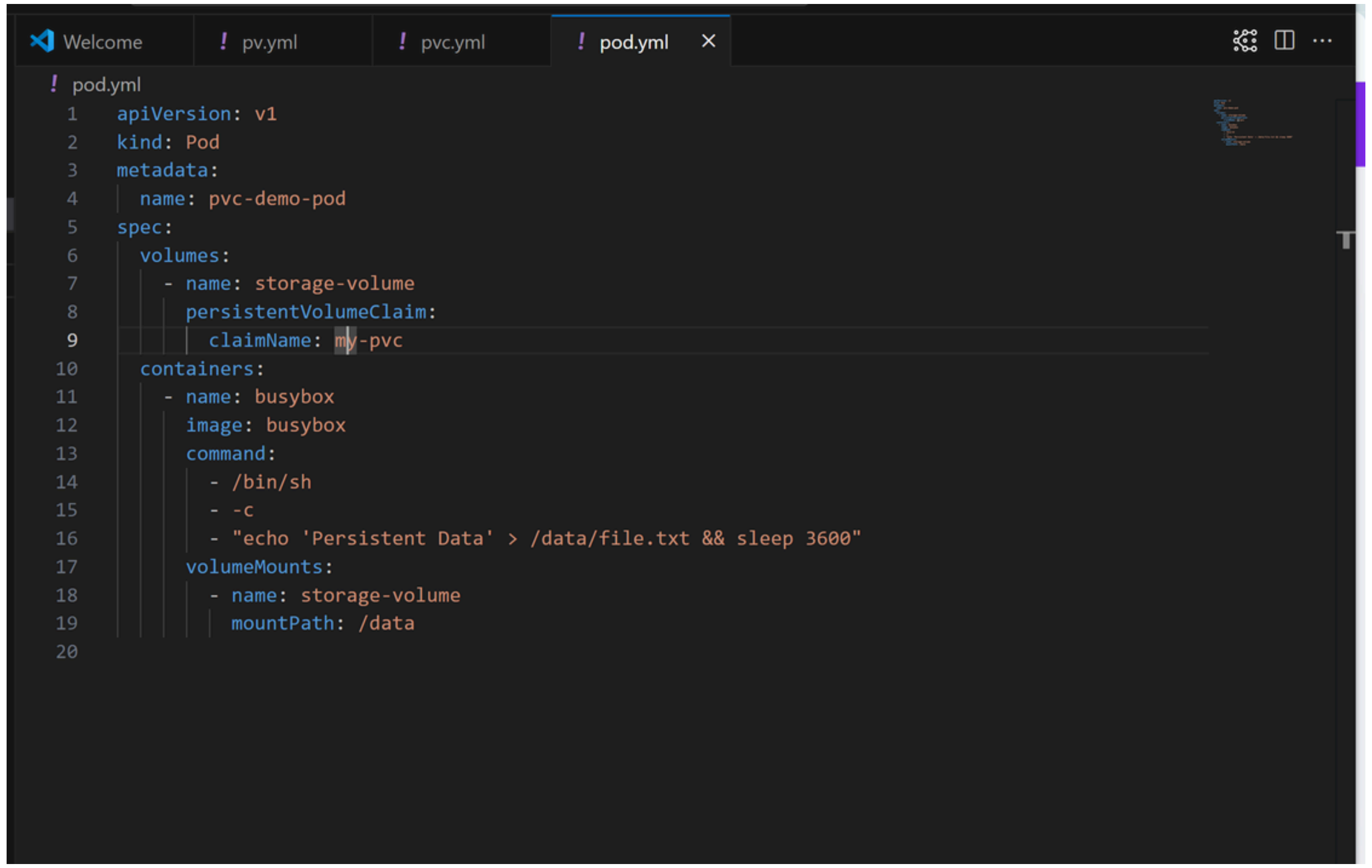
- First create pv.ymll file
- Then set a name
- Then set a spec
- storage for 1gb
- set access mode RWO
- Then set Retain policy
- Then set host path
- Normaly pv file created by admin



```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Created pvc.yml file

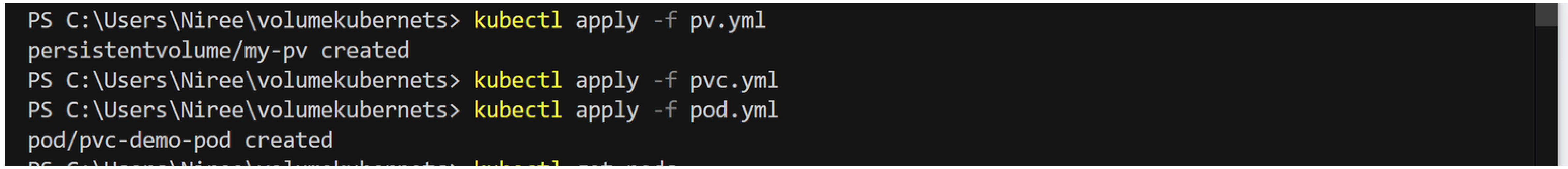
- Second Create PVC.YML FILE
- Then set a kind of persistentvolumeclaim
- Then specify the name wich created in pv.yml
- Then spec set rwo mode
- Then storage set it for 1gb



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pvc-demo-pod
spec:
  volumes:
    - name: storage-volume
      persistentVolumeClaim:
        claimName: my-pvc
  containers:
    - name: busybox
      image: busybox
      command:
        - /bin/sh
        - -c
        - "echo 'Persistent Data' > /data/file.txt && sleep 3600"
      volumeMounts:
        - name: storage-volume
          mountPath: /data
```

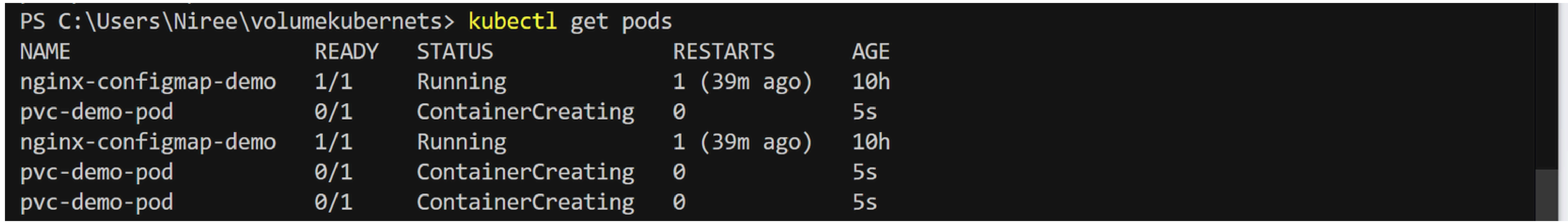
Finally Create POD.YML file

- Finally Create Pod.yml
- Then Set a Pod name
- Then spec create for volume
- Then Set from we need to claim set pv file name
- Then create container as busybox
- This means the container will write the "Persistent Data" to the file /data/file.txt, which is actually stored in the PersistentVolume linked to the my-pvc PersistentVolumeClaim.
- then mount on data path



```
PS C:\Users\Niree\volumekubernets> kubectl apply -f pv.yml
persistentvolume/my-pv created
PS C:\Users\Niree\volumekubernets> kubectl apply -f pvc.yml
PS C:\Users\Niree\volumekubernets> kubectl apply -f pod.yml
pod/pvc-demo-pod created
```

Then Apply thoes three files and check pod is running or not



```
PS C:\Users\Niree\volumekubernets> kubectl get pods
NAME                    READY   STATUS              RESTARTS      AGE
nginx-configmap-demo    1/1     Running             1 (39m ago)   10h
pvc-demo-pod            0/1     ContainerCreating   0             5s
nginx-configmap-demo    1/1     Running             1 (39m ago)   10h
pvc-demo-pod            0/1     ContainerCreating   0             5s
pvc-demo-pod            0/1     ContainerCreating   0             5s
```

- After That With Help this command **kubectl exec -it pvc-demo-pod -- cat /data/file.txt**
- Check persistent data file created or not
- Then it created means delete that pod and recreate that time also that file will be avalabile