

**Lab Manual- Create and use a volume with Azure Disks in Azure Kubernetes Service (AKS)**

**Prepared for**:

**Date:** 18th Dec 2023

**Prepared by:**

Document Name: Lab Manual **Document Number** AZLabn916

**Contributor:**

Contents

[1. Objective 3](#_Toc157162215)

[1. Built-in storage classes 3](#_Toc157162216)

[2. Persistent Volume Access Mode: 3](#_Toc157162217)

[2. Create 1 Node Azure Kubernetes Cluster 5](#_Toc157162218)

[3. Create a Custom Storage Class 6](#_Toc157162219)

[4. Create a Persistent Volume Claim 7](#_Toc157162220)

[5. Use the persistent volume 8](#_Toc157162222)

[6. Storage data in the persistent volume 11](#_Toc157162223)

# Objective

A persistent volume represents a piece of storage provisioned for use with Kubernetes pods. You can use a persistent volume with one or many pods, and you can provision it dynamically or statically. This article shows you how to dynamically create persistent volumes with Azure Disks in an Azure Kubernetes Service (AKS) cluster.

### Built-in storage classes

Storage classes define how a unit of storage is dynamically created with a persistent volume. For more information on Kubernetes storage classes, see [Kubernetes storage classes](https://kubernetes.io/docs/concepts/storage/storage-classes/).

Each AKS cluster includes four precreated storage classes, two of them configured to work with Azure Disks:

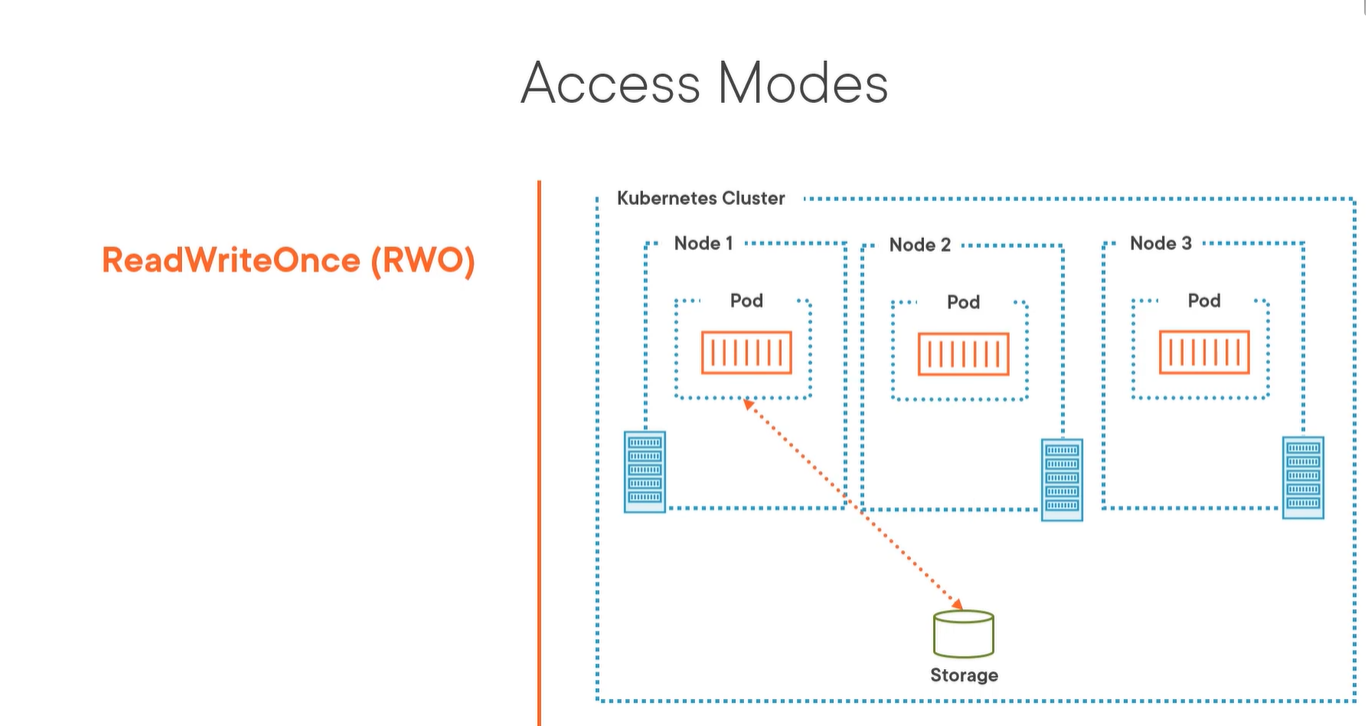
1. The default storage class provisions a standard SSD Azure Disk.
   * Standard SSDs backs Standard storage and delivers cost-effective storage while still delivering reliable performance.
2. The managed-csi-premium storage class provisions a premium Azure Disk.
   * SSD-based high-performance, low-latency disks back Premium disks. They're ideal for VMs running production workloads. When you use the Azure Disk CSI driver on AKS, you can also use the managed-csi storage class, which is backed by Standard SSD locally redundant storage (LRS).

### Persistent Volume Access Mode:

In Kubernetes, the access modes for Persistent Volumes (PV) determine how the volume can be accessed by pods. Here are the differences between the three access modes you mentioned in AKS:

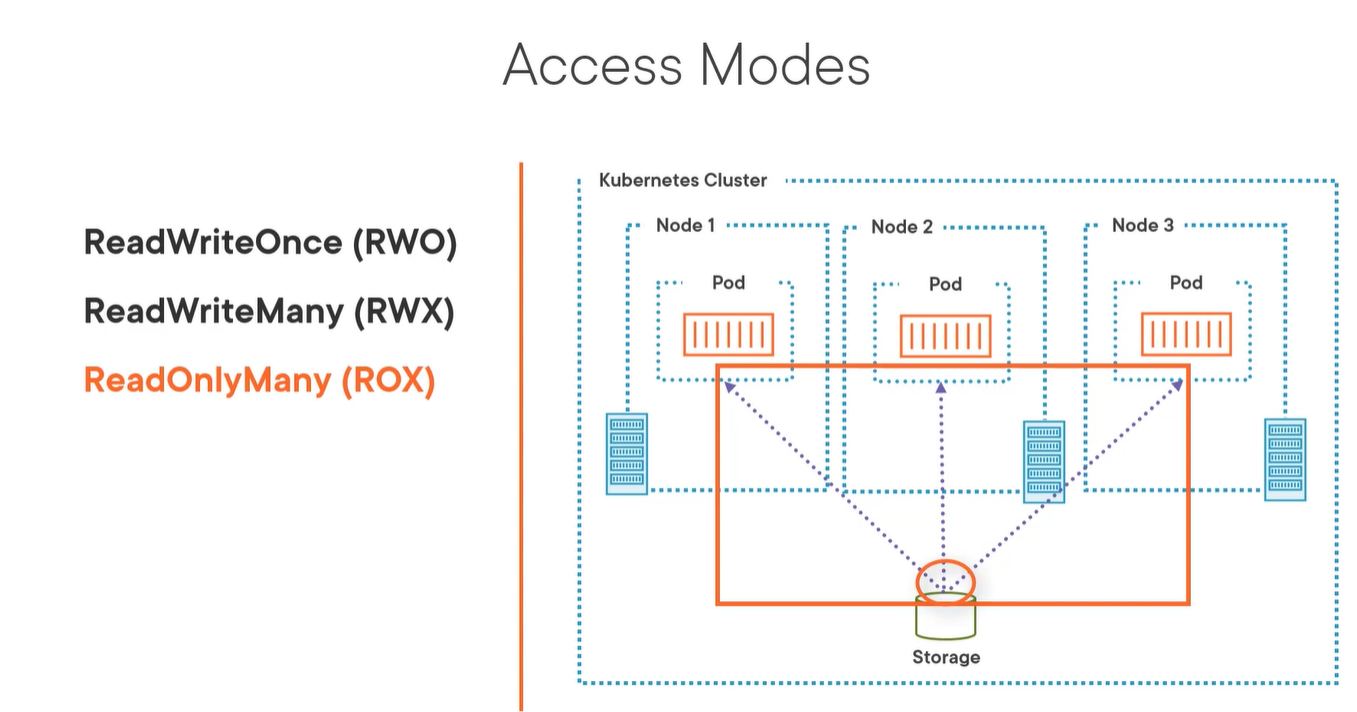
**ReadWriteOnce**: This access mode allows the volume to be mounted as read-write by a single node at a time. This is the most common access mode used in AKS and is suitable for most workloads where only one pod at a time needs to read and write data to the volume.

PVC can only be mounted as read-write by a single pod at a time. EG Single Database POD ( No replica)



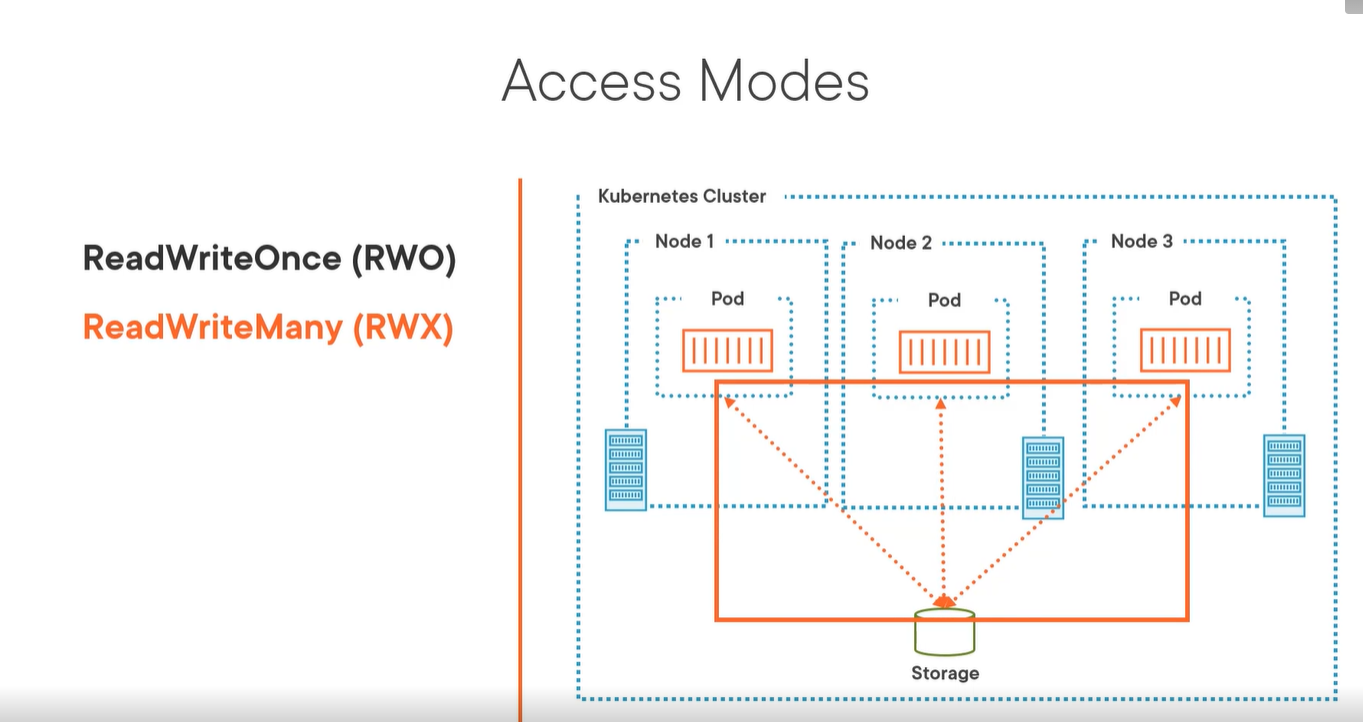
**ReadOnlyMany**: This access mode allows the volume to be mounted as read-only by multiple nodes simultaneously. This access mode is useful in scenarios where multiple nodes need to read the same data, but only one node should be able to write to the volume.

PVC can be mounted as read-only by multiple pods simultaneously . For e.g web application in AKS that needs to serve static assets ( Image , File etc) from a shared volume that can be accessed by multiple pods. You can define a PVC with the ReadOnlyMany access mode to allow multiple pods to read the data from the volume, but only one pod should be allowed to write to the volume



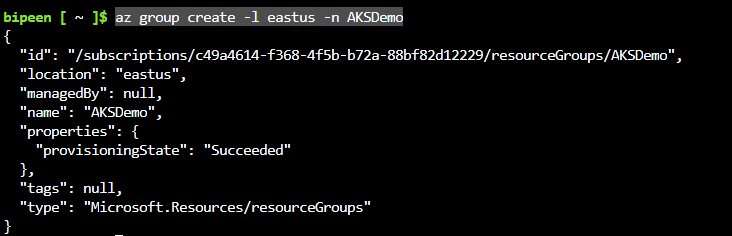
**ReadWriteMany**: This access mode allows the volume to be mounted as read-write by multiple nodes simultaneously. This access mode is suitable for distributed file systems or other scenarios where multiple pods need to read and write data to the same volume simultaneously.

Let's say you have a distributed application in AKS that needs to share data between multiple pods. You can define a PVC with the ReadWriteMany access mode to allow multiple pods to read and write data to the same volume simultaneously



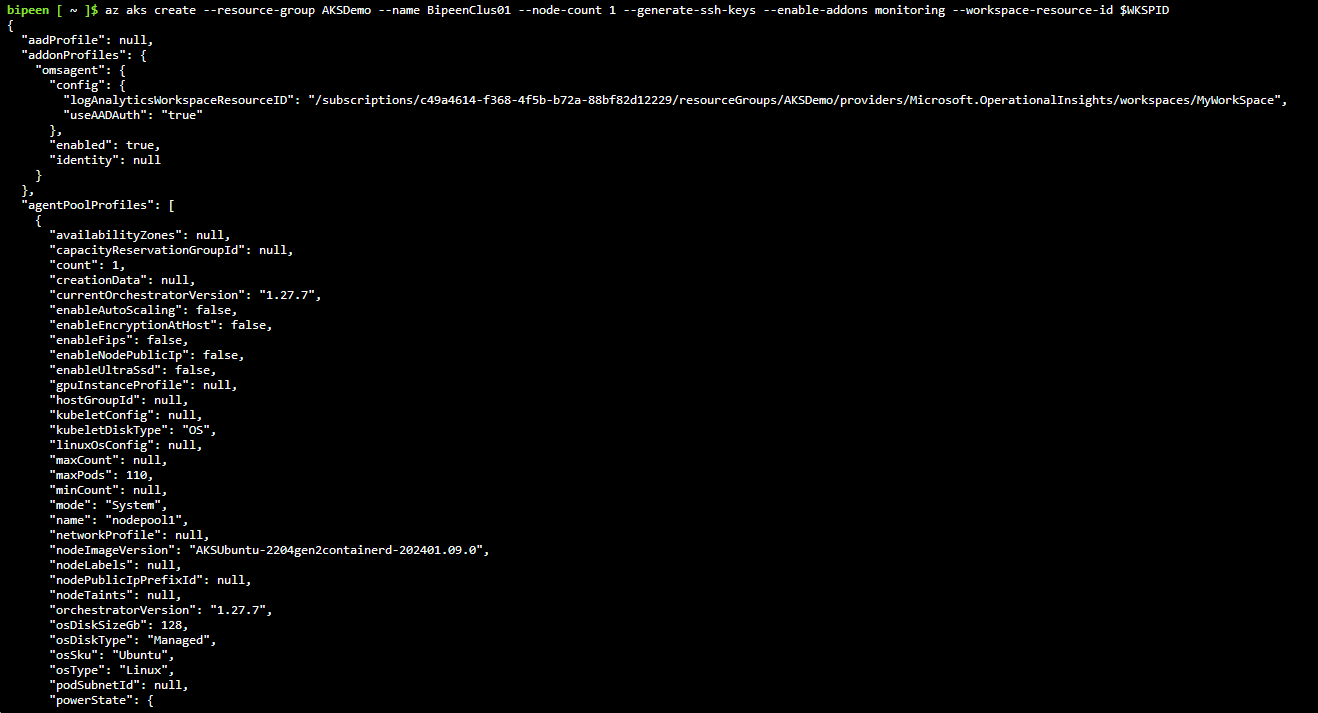
# Create 1 Node Azure Kubernetes Cluster

az group create -l eastus -n AKSDemo



To create an AKS cluster, use the [az aks create](https://learn.microsoft.com/en-us/cli/azure/aks" \l "az-aks-create) command. The following example creates a cluster named BipeenClus01 with one node and generate SSH-key. It will automatically use 3 Nodes when you don’t specify node count

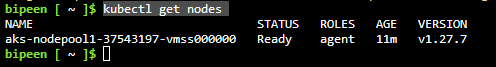
az aks create --resource-group AKSDemo --name BipeenClus01 --node-count 1 --generate-ssh-keys



az aks get-credentials --resource-group AKSDemo --name BipeenClus01 --overwrite-existing



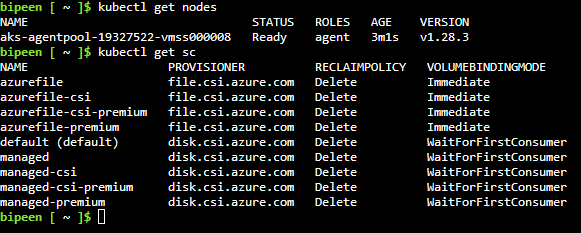
kubectl get nodes



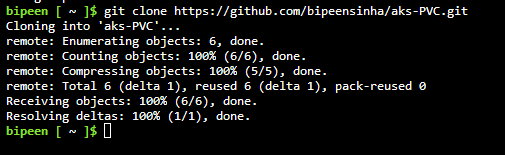
# Create a Custom Storage Class

* Review **default storage** class ( for Azure File and Azure Disk) and notice **Reclaim Policy** is set to delete not retain.

kubectl get sc

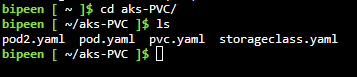


git clone https://github.com/bipeensinha/aks-PVC.git



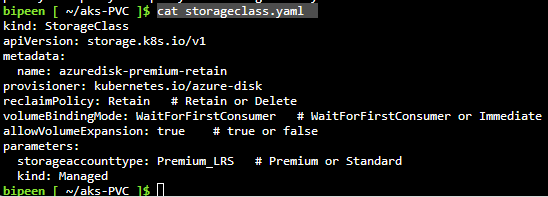
cd aks-PVC/

ls



We have one file to create custom storage class is **azure-premimum-retain** , The reclaim policy is Retain ( Instead of delete)

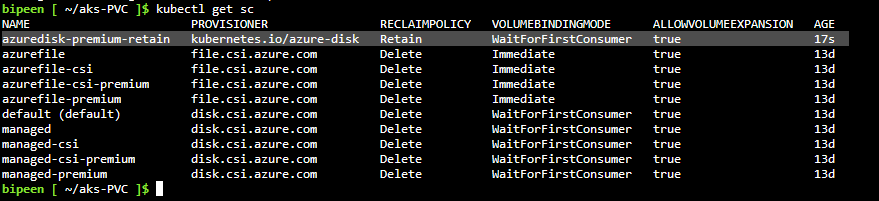
cat storageclass.yaml



kubectl apply -f storageclass.yaml



kubectl get sc



# Create a Persistent Volume Claim

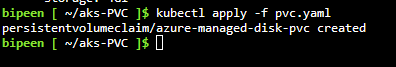
In this File claim requests a disk named azure-managed-disk-pvc  that's 4 GB in size with **ReadWriteOnce** access. The azuredisk-premimum-retain storage class is specified which we have created in previous stage

cat pvc.yaml

# 

kubectl apply -f pvc.yaml

.

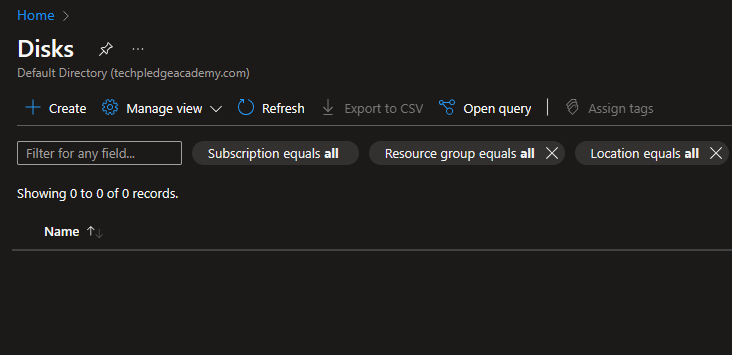


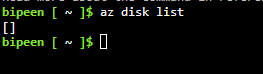
* Check persistant volume , Notice the **pending** state because no pod has claim it so far and until one pod claim there will be no disk. ( It is defined in storage class kubectl get sc show wait for consumer)

kubectl get pvc



Event on portal or clie show no disk



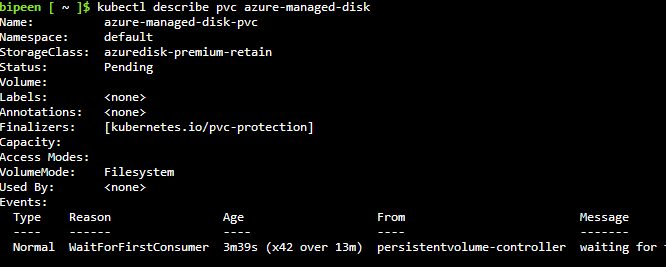
’

# Use the persistent volume

After you create the persistent volume claim, you must verify it has a status of Pending. The Pending status indicates it's ready to be used by a pod.

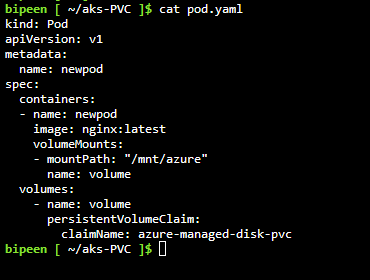
* Verify the status of the PVC using the kubectl describe pvc command.

kubectl describe pvc azure-managed-disk



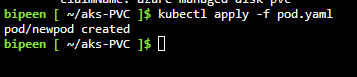
* Now we have Basic Pod definition **NGINX** pod that uses the persistent volume claim named *azure-managed-disk* to mount the Azure Disk at the path **/mnt/azure**. For Windows Server containers, specify a *mountPath* using the Windows path convention, such as *'D:'*

cat pod.yaml

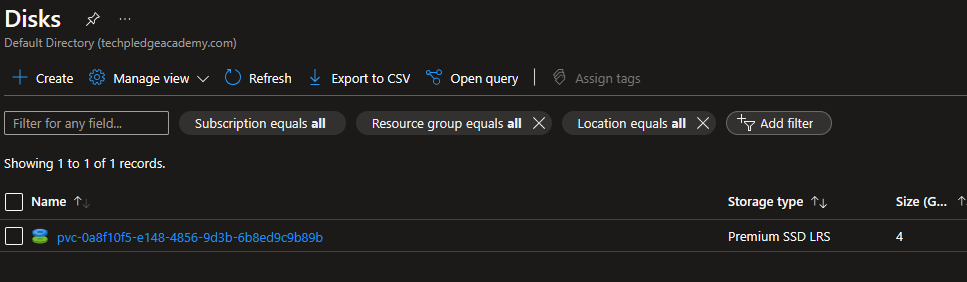


* Create the pod using the [kubectl apply](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands" \l "apply) command.

kubectl apply -f pod.yaml



* Now after few minutes you can see this disk in portal



* Now Check POD claim status it was **pending** , now it must be bound

kubectl get pvc

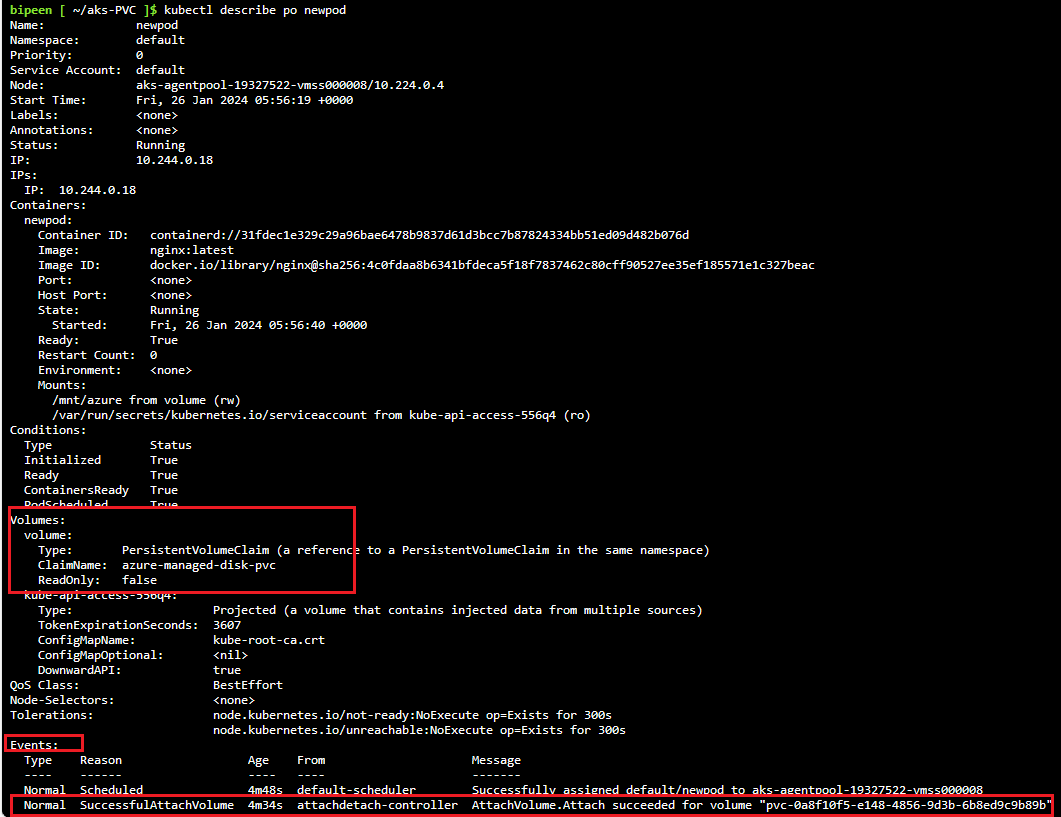


* You now have a running pod with your Azure Disk mounted in the /mnt/azure directory. Check the pod configuration using the [kubectl describe](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands" \l "describe) command.

kubectl get po



kubectl describe po newpod



# Storage data in the persistent volume

* Lets login to container

kubectl exec --stdin --tty newpod -- /bin/bash



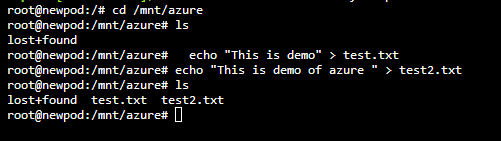
* Go inside mount directory and create two file

cd /mnt/azure

ls

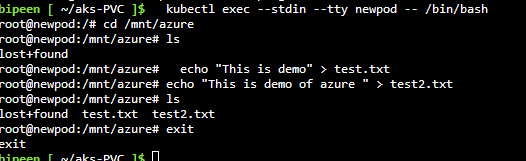
echo "This is demo" > test.txt

echo "This is demo of azure " > test2.txt



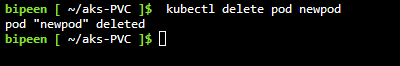
* Exit from the pod

exit



* Now delete the pod

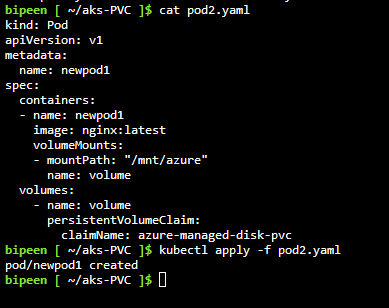
kubectl delete pod newpod



Now Create New POD

cat pod2.yaml

kubectl apply -f pod2.yaml



kubectl exec --stdin --tty newpod1 -- /bin/bash

cd /mnt/azure

ls

