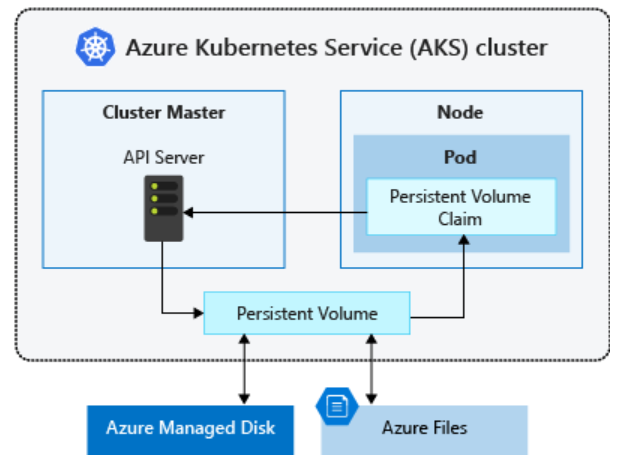
AKS - Storage

13 July 2023

18:39

**AKS Storage - Core Concepts**

* **Volumes**
* **Persistent Volumes**
* **Persistent Volume Claims**
* **Storage Classes**



**Volumes**

A volume represents a **way to store, retrieve, and persist data** across pods and through the application lifecycle. Traditional volumes are created as Kubernetes resources backed by Azure Storage. You can manually create data volumes to be assigned to pods directly or have Kubernetes automatically create them. Data volumes can use: **Azure Disk**, **Azure Files**, **Azure NetApp Files**, or **Azure Blobs**.

**Azure Disk**

* Use Azure Disk to create a Kubernetes Data Disk resource. Disks types include:
  + Ultra Disks
  + Premium SSDs
  + Standard SSDs
  + Standard HDDs
* Because Azure Disk is mounted as ReadWriteOnce, they're only available to a single node.
* For storage volumes accessible by pods on multiple nodes simultaneously, use Azure Files.

**Azure Files**

* Use Azure Files to mount a SMB version 3.1.1 share or NFS version 4.1 share backed by an Azure storage account to pods.
* Azure Files let you share data across multiple nodes and pods and can use:
  + Azure Premium storage backed by high-performance SSDs
  + Azure Standard storage backed by regular HDDs

**Azure NetApp Files**

* Ultra Storage
* Premium Storage
* Standard Storage

**Azure Blob Storage**

* Use Azure Blob Storage to create a blob storage container and mount it using the NFS v3.0 protocol or BlobFuse.
  + Block Blobs

**Volume Types**

Kubernetes volumes represent more than just a traditional disk for storing and retrieving information. Kubernetes volumes can also be used as a way to inject data into a pod for use by the containers. Common volume types in Kubernetes include:

**emptyDir**

Commonly used as **temporary space for a pod**. All containers within a pod can access the data on the volume. Data written to this volume type persists only for the lifespan of the pod. Once you delete the pod, the volume is deleted. This volume typically uses the underlying local node disk storage, though it can also exist only in the node's memory.

**Secret**

You can use secret volumes to **inject sensitive data into pod**s, such as passwords.

* Create a Secret using the Kubernetes API.
* Define your pod or deployment and request a specific Secret.
* Secrets are only provided to nodes with a scheduled pod that requires them.
* The Secret is stored in tmpfs, not written to disk.
* When you delete the last pod on a node requiring a Secret, the Secret is deleted from the node's tmpfs.
* Secrets are stored within a given namespace and are only accessed by pods within the same namespace.

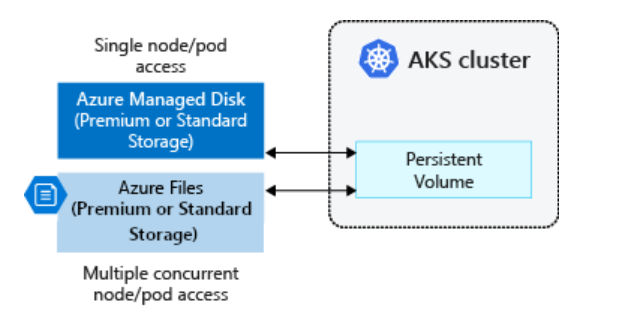
**configMap**

You can use configMap to **inject key-value pair properties into pods**, such as application configuration information.

* Create a ConfigMap using the Kubernetes API.
* Request the ConfigMap when you define a pod or deployment.
* ConfigMaps are stored within a given namespace and are only accessed by pods within the same namespace.

**Persistent Volumes**

A PersistentVolume (PV) is a **storage resource** created and managed by the Kubernetes API that can **exist beyond the lifetime of an individual pod**.

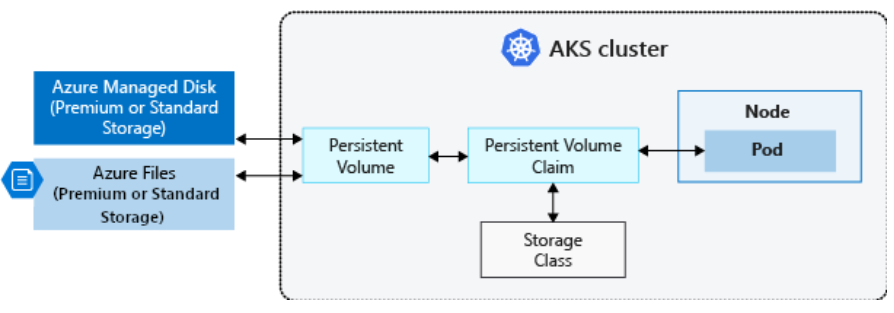


**Note:**

* You can use Azure Disk or Azure Files to provide the PersistentVolume. The choice of Disks or Files is often determined by the **need for concurrent access to the data** or the **performance tier**.
* A cluster administrator can **statically create a PersistentVolume**, or the **volume is created dynamically** by the Kubernetes API server.
* If a pod is scheduled and requests currently unavailable storage, **Kubernetes can create the underlying Azure Disk or File storage** and attach it to the pod.
* **Dynamic provisioning uses a StorageClass** to identify what type of Azure storage needs to be created.
* Persistent volumes can't be shared by Windows and Linux pods due to differences in file system support between the two operating systems.

**Persistent Volume Claims**

A PersistentVolumeClaim (PVC) **requests storage of a particular StorageClass, access mode, and size**. The Kubernetes API server can **dynamically provision the underlying Azure storage resource** if no existing resource can fulfil the claim based on the defined StorageClass.

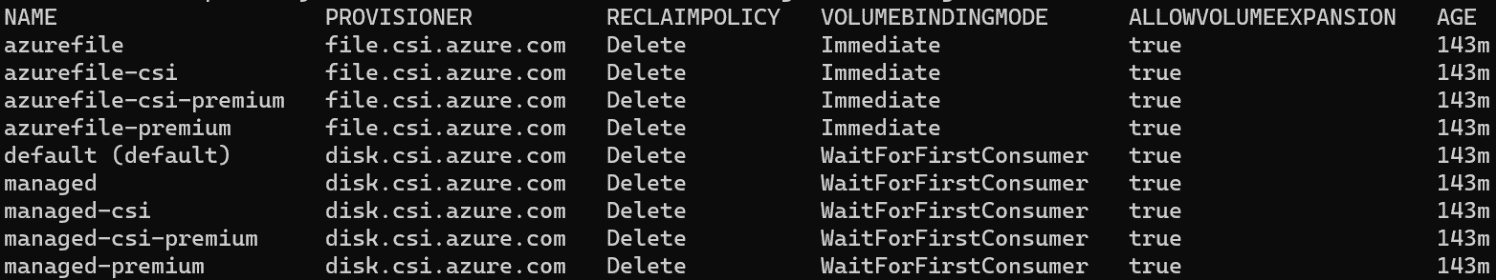


**Note:**

* The pod definition includes the volume mount once the volume has been connected to the pod.
* Once an available storage resource has been assigned to the pod requesting storage, PersistentVolume is bound to a PersistentVolumeClaim.
  + Persistent volumes are **1:1** mapped to claims.
* When you create a pod definition, you also specify:
  + The **persistentVolumeClaim** to **request the desired storage**.
  + The **volumeMount** for your applications to **read and write data**.

**Storage Classes**

A StorageClass provides a way for administrators to describe the "classes" of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators.



**Note:**

* Unless you specify a StorageClass for a persistent volume, the default StorageClass is used.
* Ensure volumes use the appropriate storage you need when requesting persistent volumes.
* The StorageClass also defines the reclaimPolicy.
  + When you delete the persistent volume, the reclaimPolicy controls the behavior of the underlying Azure storage resource.
  + The underlying storage resource can either be deleted or kept for use with a future pod.