HOW TO SAVE SET OF APPLICATION IN AKS

MsSql operator
Azure files

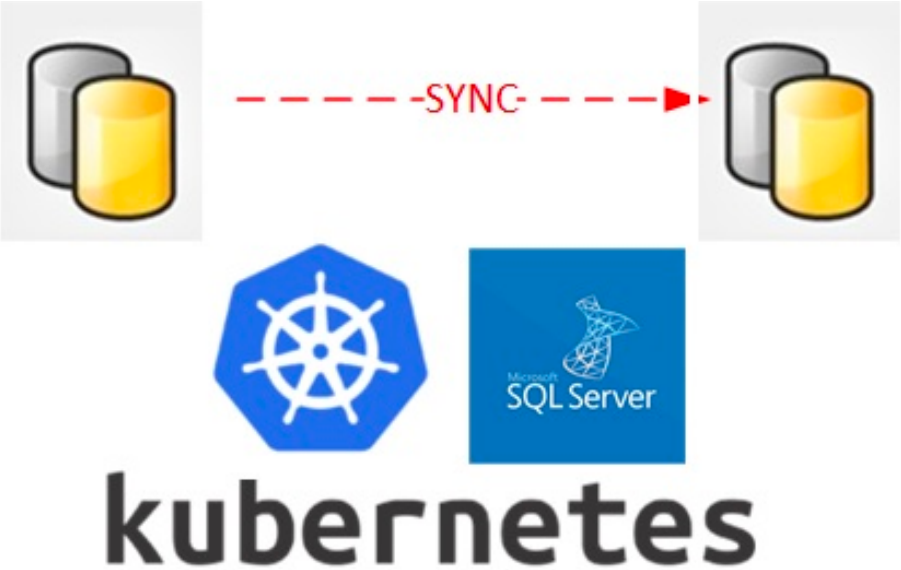
CREATED BY LEONID CHETVERIKOV

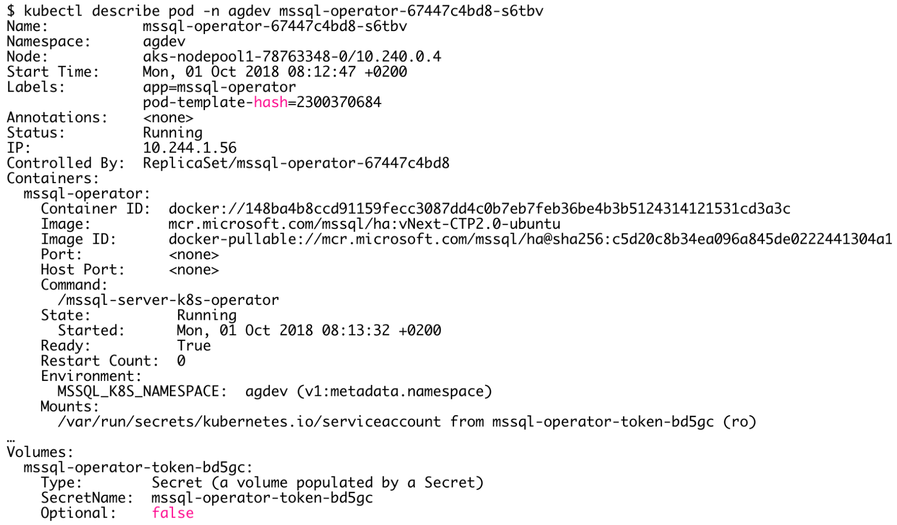
NEXT SQL SERVER

Enhanced performance, security, and availability:

- Industry-leading performance – The Intelligent Database
- Advanced security – Confidential Computing
- Mission-critical availability – High uptime
- Developer experience
- Platform of choice

What is new on sql server 2019?





```
$ kubectl describe pod -n agdev mssql-operator-67447c4bd8-s6tbv
Name:           mssql-operator-67447c4bd8-s6tbv
Namespace:      agdev
Node:           aks-nodepool1-78763348-0/10.240.0.4
Start Time:     Mon, 01 Oct 2018 08:12:47 +0200
Labels:         app=mssql-operator
                pod-template-hash=2300370684
Annotations:    <none>
Status:         Running
IP:             10.244.1.56
Controlled By:  ReplicaSet/mssql-operator-67447c4bd8
Containers:
  mssql-operator:
    Container ID:  docker://148ba4b8ccd91159fecc3087dd4c0b7eb7feb36be4b3b5124314121531cd3a3c
    Image:         mcr.microsoft.com/mssql/ha:vNext-CTP2.0-ubuntu
    Image ID:      docker-pullable://mcr.microsoft.com/mssql/ha@sha256:c5d20c8b34ea096a845de0222441304a1
    Port:          <none>
    Host Port:     <none>
    Command:
      /mssql-server-k8s-operator
    State:          Running
      Started:      Mon, 01 Oct 2018 08:13:32 +0200
    Ready:          True
    Restart Count:  0
    Environment:
      MSSQL_K8S_NAMESPACE:  agdev (v1:metadata.namespace)
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from mssql-operator-token-bd5gc (ro)
…
Volumes:
  mssql-operator-token-bd5gc:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  mssql-operator-token-bd5gc
    Optional:    false
```

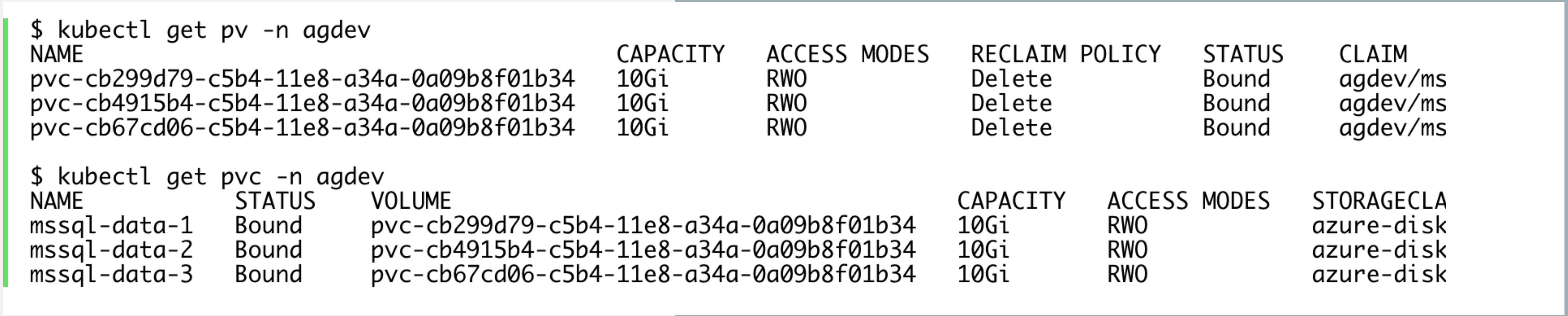
ALWAYS ON AVAILABILITY GROUPS ON AKS

MSSQL-OPERATOR

The SQL Server CTP image – mcr.microsoft.com/mssql/ha – comes from the new Microsoft Container Registry (MCR). The current tag is vNext-CTP2.0-ubuntu at the moment of this write-up

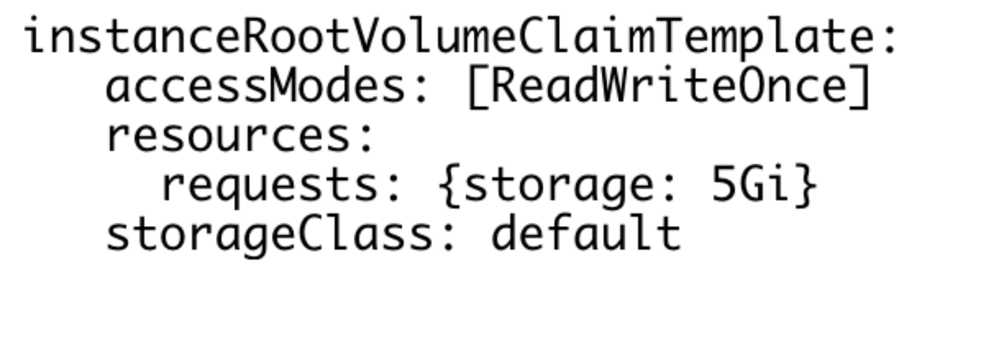
Volume secret is mounted to pass sensitive data that concerns a K8s service account used by the pod.

MSSQL STATEFULSET



```
$ kubectl get pv -n agdev
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS    CLAIM
pvc-cb299d79-c5b4-11e8-a34a-0a09b8f01b34   10Gi       RWO            Delete           Bound     agdev/ms
pvc-cb4915b4-c5b4-11e8-a34a-0a09b8f01b34   10Gi       RWO            Delete           Bound     agdev/ms
pvc-cb67cd06-c5b4-11e8-a34a-0a09b8f01b34   10Gi       RWO            Delete           Bound     agdev/ms

$ kubectl get pvc -n agdev
NAME           STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLA
mssql-data-1   Bound     pvc-cb299d79-c5b4-11e8-a34a-0a09b8f01b34   10Gi       RWO            azure-disk
mssql-data-2   Bound     pvc-cb4915b4-c5b4-11e8-a34a-0a09b8f01b34   10Gi       RWO            azure-disk
mssql-data-3   Bound     pvc-cb67cd06-c5b4-11e8-a34a-0a09b8f01b34   10Gi       RWO            azure-disk
```

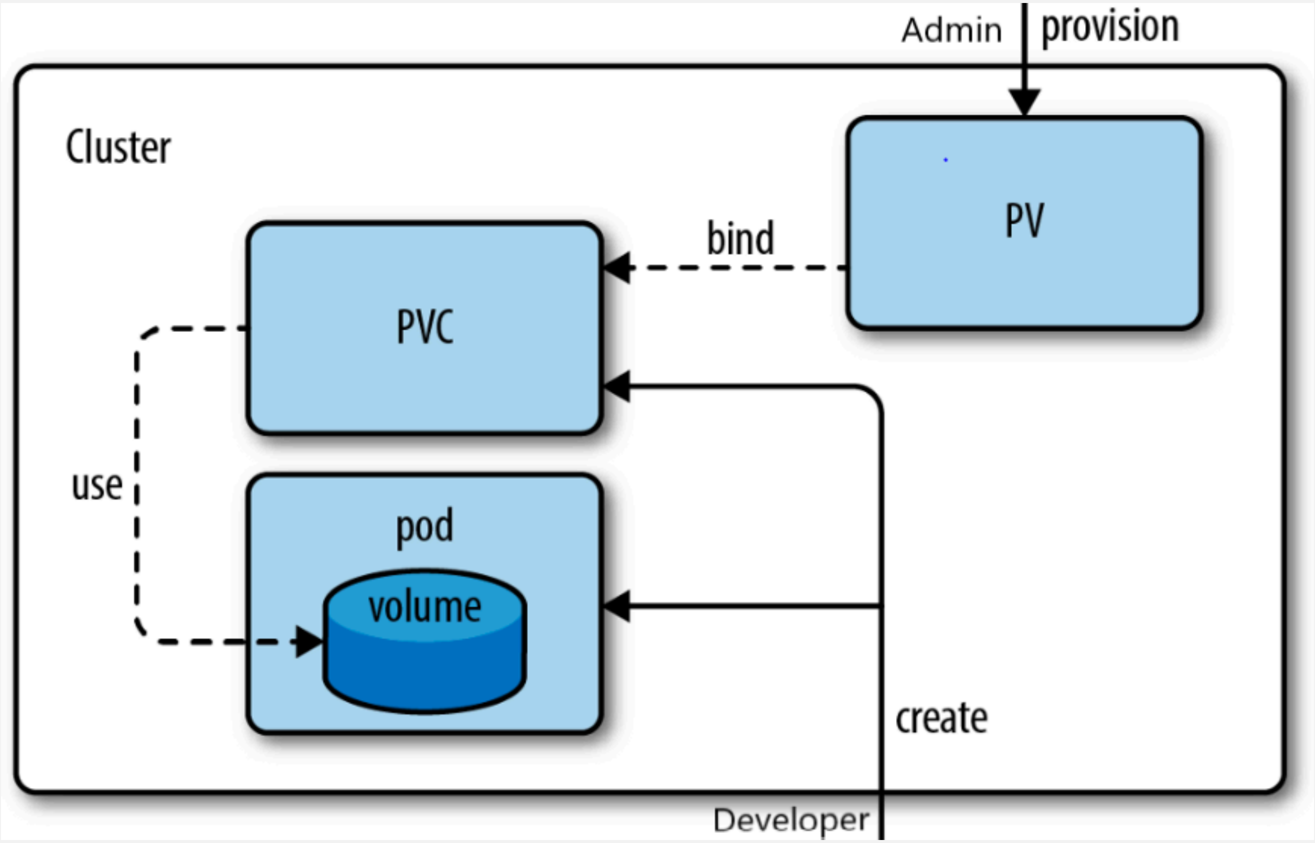


```
instanceRootVolumeClaimTemplate:
   accessModes: [ReadWriteOnce]
   resources:
     requests: {storage: 5Gi}
   storageClass: default
```

The deployment includes 3 StatefulSets that manage pods with 2 containers

WHY PERSISTENT STORAGE NEEDS TO BE PROVISIONED IN K8S?

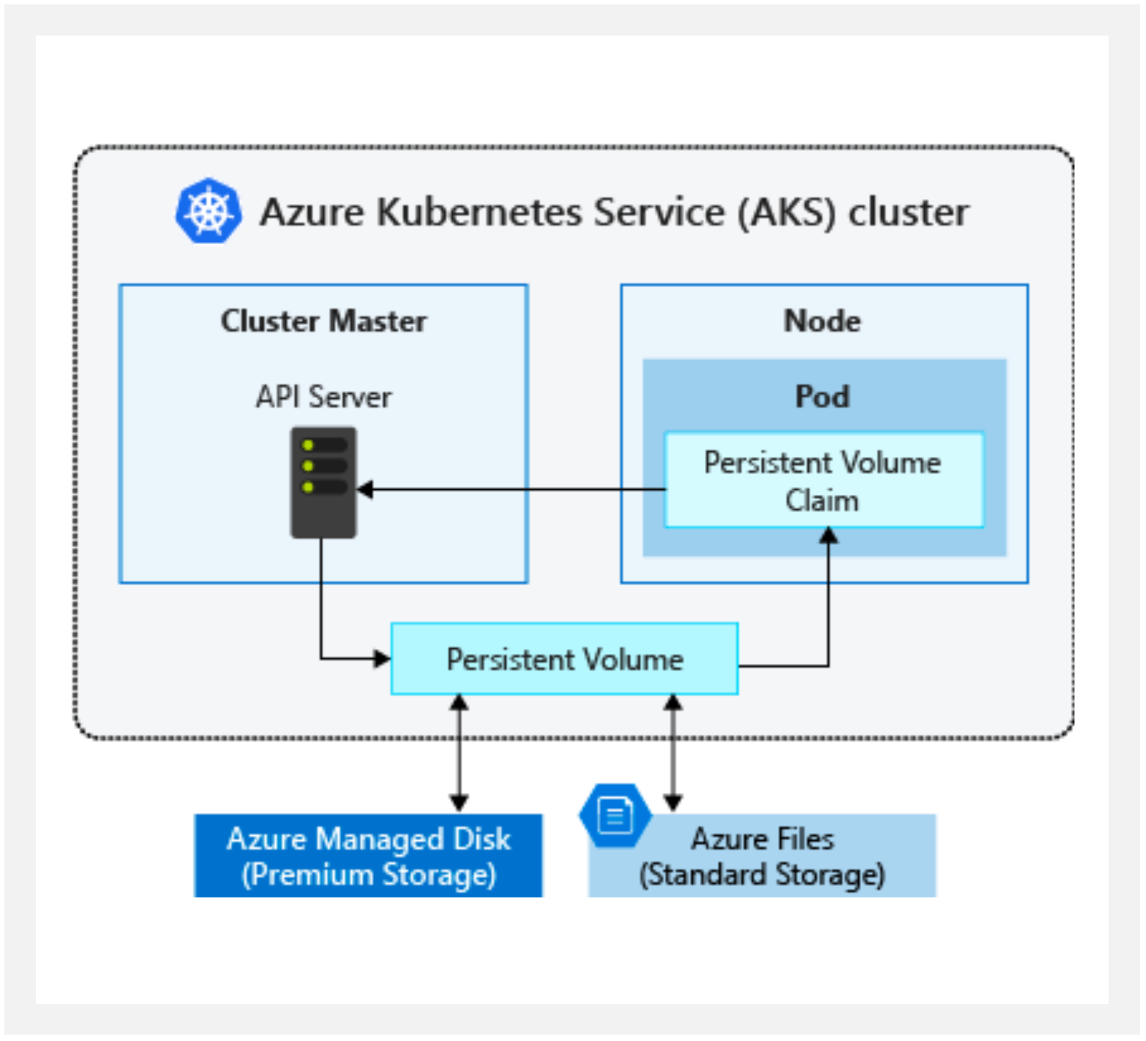
- Not every workload is stateless, any meaningful application will eventually deal with data and data needs to be persisted. Pods often expect their storage to remain if a pod is rescheduled on a different host during a maintenance event, especially in StatefulSets.



METHOD — 1 : MOUNT VOLUME VIA SECRET KEY

- TERRAFORM SCRIPT

```
resource "azurerm_resource_group" "test" {
                         name = "azuretest”
                         location = "West Europe" }
resource "azurerm_storage_account" "test" {
                         name = "azureteststorage" resource_group_
                         name = "${azurerm_resource_group.test.name}”
                          location = "${azurerm_resource_group.test.location}”
                          account_tier = "Standard"
                         account_replication_type = "LRS" }
resource "azurerm_storage_share" "testshare" {
                          name = "sharename”
                         storage_account_name = "${azurerm_storage_account.test.name}"
                         quota = 50 }
```

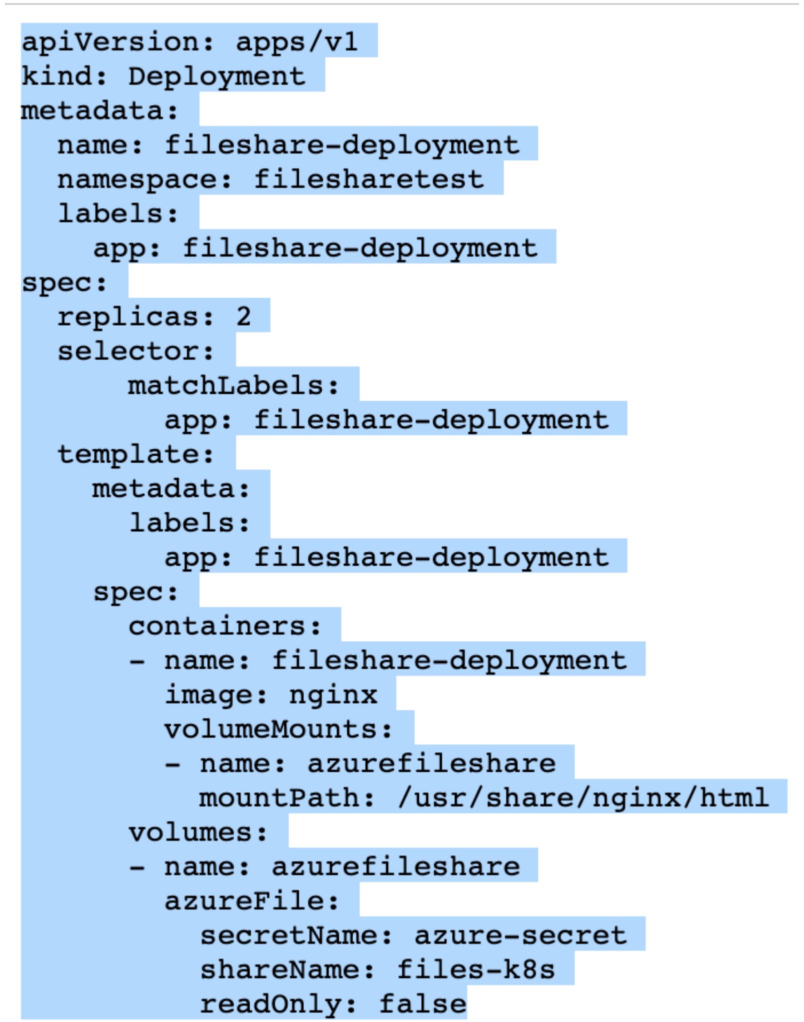

Azure Kubernetes Service (AKS) cluster
Cluster Master
API Server
Node
Pod
Persistent Volume Claim
Persistent Volume
Azure Managed Disk (Premium Storage)
Azure Files (Standard Storage)

METHOD — 1 : MOUNT VOLUME VIA SECRET KEY

```
STORAGE_KEY=$(az storage account keys list --resource-group $AKS_PERS_RESOURCE_GROUP --account-name $AKS_PERS_STORAGE_ACCOUNT_NAME --query "[0].value" -o tsv)
```

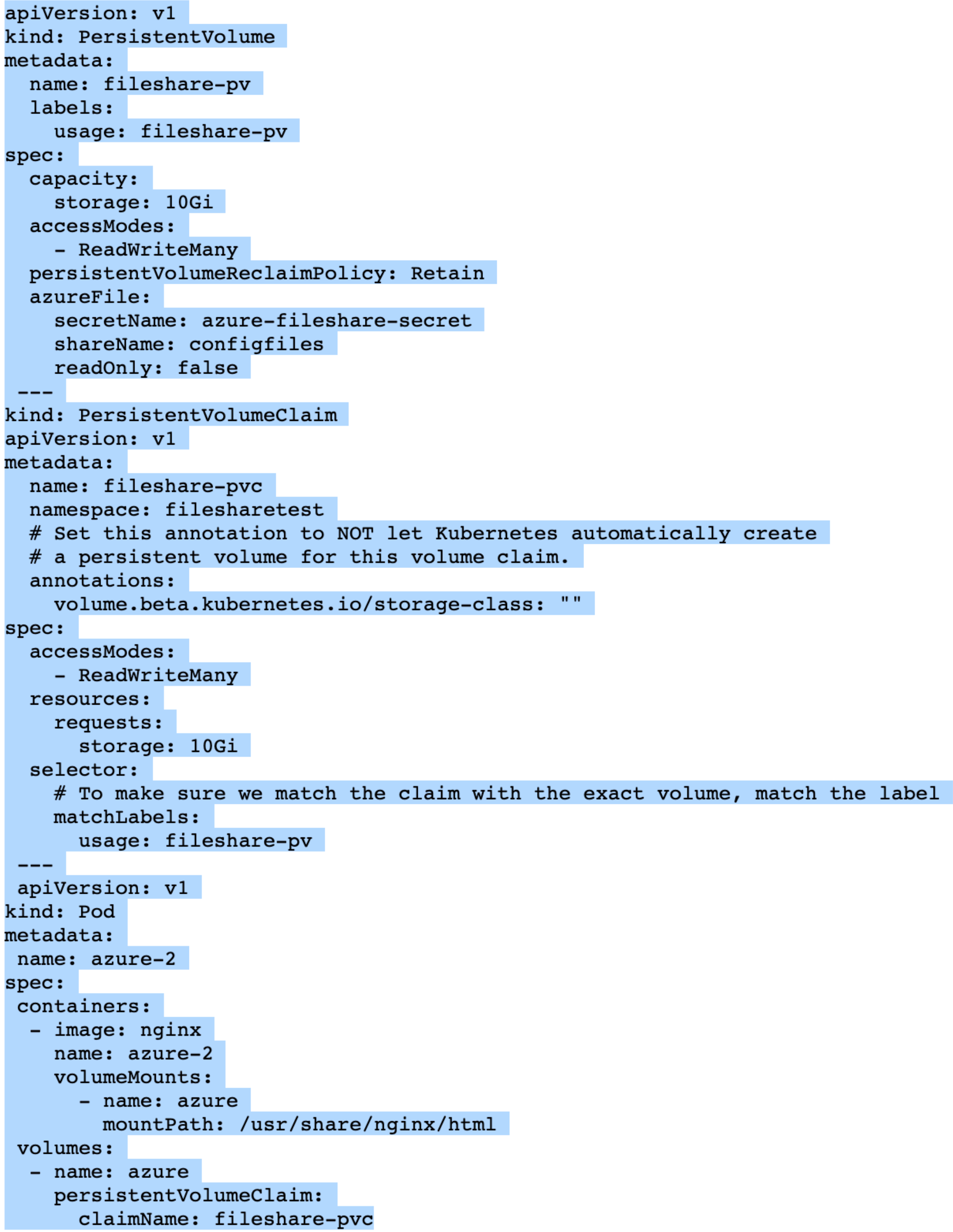
```
kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=<...> --from-literal=azurestorageaccountkey=<...>
```

METHOD — 1 : MOUNT VOLUME VIA SECRET KEY



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fileshare-deployment
  namespace: filesharetest
  labels:
    app: fileshare-deployment
spec:
  replicas: 2
  selector:
      matchLabels:
        app: fileshare-deployment
  template:
    metadata:
      labels:
        app: fileshare-deployment
    spec:
      containers:
      - name: fileshare-deployment
        image: nginx
        volumeMounts:
        - name: azurefileshare
          mountPath: /usr/share/nginx/html
      volumes:
      - name: azurefileshare
        azureFile:
          secretName: azure-secret
          shareName: files-k8s
          readOnly: false
```

METHOD — 2: MOUNT VOLUME VIA PV AND PVC



```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: fileshare-pv
  labels:
    usage: fileshare-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  azureFile:
    secretName: azure-fileshare-secret
    shareName: configfiles
    readOnly: false
 ---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: fileshare-pvc
  namespace: filesharetest
  # Set this annotation to NOT let Kubernetes automatically create
  # a persistent volume for this volume claim.
  annotations:
    volume.beta.kubernetes.io/storage-class: ""
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  selector:
    # To make sure we match the claim with the exact volume, match the label
    matchLabels:
      usage: fileshare-pv
 ---
 apiVersion: v1
kind: Pod
metadata:
 name: azure-2
spec:
 containers:
  - image: nginx
    name: azure-2
    volumeMounts:
      - name: azure
        mountPath: /usr/share/nginx/html
 volumes:
  - name: azure
    persistentVolumeClaim:
      claimName: fileshare-pvc
```

REFERENCES

- **Persistent Volumes**
- Edit This Page This document describes the current state of PersistentVolumes in Kubernetes. Familiarity with volumes…
- kubernetes.io

- https://b1thunt3r.se/2019/07/sql-aks-ag