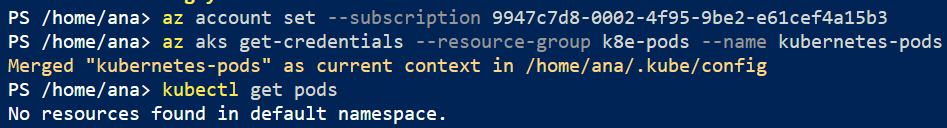
Kubernetes Storage

# Practice 1: Direct provisioning of Azure File storage

1. Login to Azure and connect to your AKS cluster.
2. Check if any pods run under the default namespace if so delete everything under the default namespace.



1. In this practice we will directly provision Azure Files to a pod running inside AKS.
2. First create the Azure Files share. Run the following commands:

# Change these four parameters as needed for your own environment

AKS\_PERS\_STORAGE\_ACCOUNT\_NAME=mystorageaccount$RANDOM

AKS\_PERS\_RESOURCE\_GROUP=myAKSShare

AKS\_PERS\_LOCATION=eastus

AKS\_PERS\_SHARE\_NAME=aksshare

Text

Description automatically generated

# Create a resource group

az group create --name $AKS\_PERS\_RESOURCE\_GROUP --location $AKS\_PERS\_LOCATION

Graphical user interface, text, application

Description automatically generated

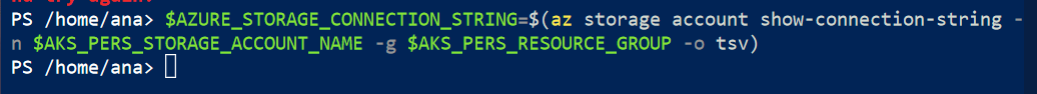
# Create a storage account

az storage account create -n $AKS\_PERS\_STORAGE\_ACCOUNT\_NAME -g $AKS\_PERS\_RESOURCE\_GROUP -l $AKS\_PERS\_LOCATION --sku Standard\_LRS

Text

Description automatically generated

# Export the connection string as an environment variable, this is used when creating the Azure file share

export AZURE\_STORAGE\_CONNECTION\_STRING=$(az storage account show-connection-string -n $AKS\_PERS\_STORAGE\_ACCOUNT\_NAME -g $AKS\_PERS\_RESOURCE\_GROUP -o tsv)

# Create the file share

az storage share create -n $AKS\_PERS\_SHARE\_NAME --connection-string

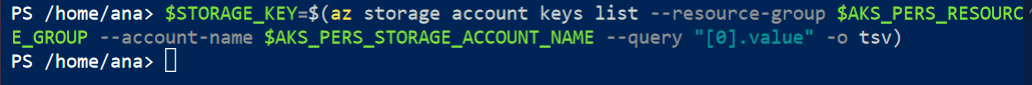
$AZURE\_STORAGE\_CONNECTION\_STRING

Text

Description automatically generated with medium confidence

# Get storage account 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)



# Echo storage account name and key

echo Storage account name: $AKS\_PERS\_STORAGE\_ACCOUNT\_NAME

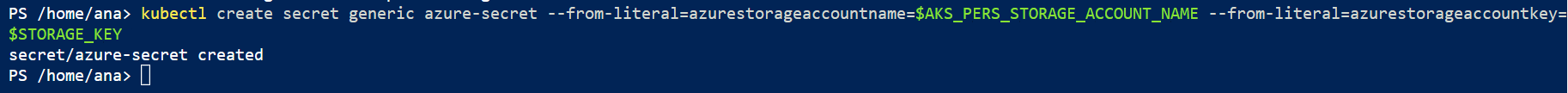
echo Storage account key: $STORAGE\_KEY

Graphical user interface, text

Description automatically generated

1. Make a note of the storage account name and key shown at the end of the script output. These values are needed when you create the Kubernetes volume in one of the following steps.
2. Now we will need to create a Kubernetes secret that will be used to mount the Az File Share to the pod. You need to hide this information from the pod’s definition and K8S secret is the best way to do it.
3. Run the following (single) command to create the secret:

kubectl create secret generic azure-secret --from-literal=azurestorageaccountname=$AKS\_PERS\_STORAGE\_ACCOUNT\_NAME --from-literal=azurestorageaccountkey=$STORAGE\_KEY



1. Check if secret was created. Run **kubectl get secret -A**.

Text

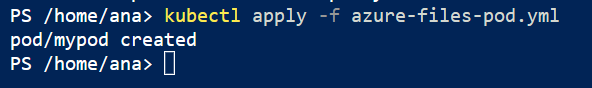
Description automatically generated

1. Now we can create the pod and mount the Azure File. Create a new file named azure-files-pod.yaml with the following contents:

Text

Description automatically generated

1. Run **kubectl apply -f azure-files-pod.yaml**.



1. You now have a running pod with an Azure Files share mounted at /mnt/azure.
2. You can use **kubectl describe pod mypod** to verify the share is mounted successfully. Search for the Volumes section of the output.

Text

Description automatically generated

1. Now exec to the pod and try to access the mounted file share. Run the following command **kubectl exec -it mypod – bash.**

**kubectl exec -it mypod -- sh**

1. Go to /mnt/azure and create a blank file test.txt file.

A picture containing text

Description automatically generated

1. Go to the portal and locate your Azure storage provisioned for this practice.
2. Under the Files section, check the contents of the Azure file share and check if test.txt file exists.

Diagnose and solve problems 
Access Control (IAM) 
Browse 
Name 
i •am-a •new•file.txt 
test. 
File 
0B 

1. Delete the mypod. What happens to the Azure File share?

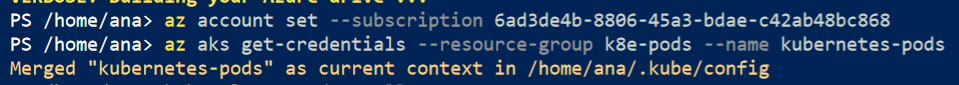
Text

Description automatically generated

* Azure File share and everything we created still exists even after the deletion of mypod.

# Practice 2: Provisioning Azure File storage using PVs and PVCs

1. Login to Azure and connect to your AKS cluster.



1. Check if any pods run under the default namespace if so, delete everything under the default namespace.

Text

Description automatically generated

1. Now we will provision Azure files storage to a pod using PV and PVC.
2. Create a azurefile-mount-options-pv.yaml file with a PersistentVolume like this:

Text

Description automatically generated

1. Note the access mode. Can you use other modes with Azure files?

* There are other access modes: ReadWriteOnce, ReadOnlyMany, ReadWriteMany and ReadWriteOncePod. But Azure file supports only: ReadWriteOnce, ReadOnlyMany, ReadWriteMany.

1. Now create an azurefile-mount-options-pvc.yaml file with a PersistentVolumeClaim that uses the PersistentVolume like this:

Text

Description automatically generated

1. Execute **kubectl apply -f azurefile-mount-options-pv.yaml** and **kubectl apply -f azurefile-mount-optionspvc.yaml**.

Text

Description automatically generated

1. Verify your PersistentVolumeClaim is created and bound to the PersistentVolume. Run **kubectl get pvc azurefile**.

Text

Description automatically generated

1. Now we can embed the PVC info inside our pod definition. Create the following file azure-files-pod.yaml with following content:

Text

Description automatically generated

1. Run **kubectl apply -f azure-files-pod.yaml**.

A blue screen with white text

Description automatically generated with medium confidence

1. You now have a running pod with an Azure Files share mounted at /mnt/azure.
2. You can use **kubectl describe pod mypod** to verify the share is mounted successfully. Search for the Volumes section of the output.

Graphical user interface, text, application

Description automatically generated

1. Now exec to the pod and try to access the mounted file share. Run the following command **kubectl exec -it mypod – bash.**

* We use **kubectl exec -it mypod – sh**

1. Go to /mnt/azure and create a blank file test.txt file.

Graphical user interface, text, application

Description automatically generated

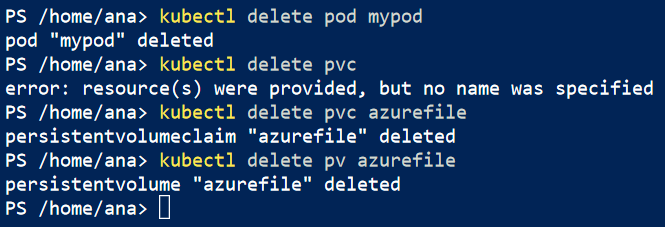
1. Go to the portal and locate your Azure storage provisioned for this practice.
2. Under the Files section, check the contents of the Azure file share and check if test.txt file exists.

Graphical user interface, application

Description automatically generated

1. Delete the mypod the pv and pvc you have created so far. What happens to the Azure File share?

* Azure File share and everything we created still exists even after the deletion of mypod, pv and pvc.



# Practice 3: Provisioning Azure file storage using Storage Classes

1. Login to Azure and connect to your AKS cluster.
2. Check if any pods run under the default namespace if so, delete everything under the default namespace.

Text

Description automatically generated

1. Now we will provision file storage using the definition of storage classes. Create a file named azure-file-sc.yaml and copy in the following example manifest:

Text

Description automatically generated

1. Create the storage class with **kubectl apply -f azure-file-sc.yaml** .

Text

Description automatically generated

1. Now we will create the PVC that will consume the storage class defined previously. Create a file named azurefile-pvc.yaml and copy in the following YAML:

Text

Description automatically generated

1. Create the persistent volume claim with the **kubectl apply -f azure-file-pvc.yaml**.

Text

Description automatically generated

1. Once completed, the file share will be created. A Kubernetes secret is also created that includes connection information and credentials. You can use the **kubectl get pvc my-azurefile** command to view the status of the PVC.

Text

Description automatically generated

1. Now we will create the pod that consumes the PVC. Create a file named azure-pvc-files.yaml, and copy in the following YAML. Make sure that the claimName matches the PVC created in the last step:

Text

Description automatically generated

1. Create the pod with **kubectl apply -f azure-pvc-files.yaml**.

Text

Description automatically generated

1. Do a describe on the pod and check the volumes mounted.

Graphical user interface, text

Description automatically generated

1. Delete everything created under this practice including the storage class.

# Practice 4: Direct provisioning of Azure Disk storage

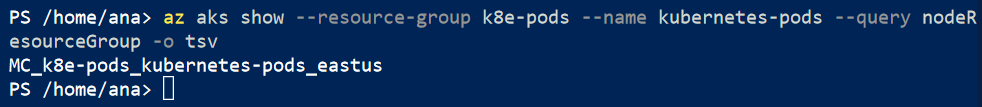
1. Login to Azure and connect to your AKS cluster.
2. Check if any pods run under the default namespace if so delete everything under the default namespace.

Text

Description automatically generated

1. In this practice we will directly provision Azure Disk to a pod running inside AKS.
2. First create the disk in the node resource group. First, get the node resource group name with **az aks show --resource-group myResourceGroup --name myAKSCluster --query nodeResourceGroup -o tsv**.

* az aks show --resource-group k8e-pods --name kubernetes-pods --query nodeResourceGroup -o tsv



1. Now create a disk using:

az disk create \

--resource-group MC\_myResourceGroup\_myAKSCluster\_eastus \

--name myAKSDisk \

--size-gb 20 \

--query id --output tsv

Text

Description automatically generated

1. Make a note of the disk resource ID shown at the end of the script output. This value is needed when you create the Kubernetes volume in one of the following steps.

* The disk resource ID: /subscriptions/6ad3de4b-8806-45a3-bdae-c42ab48bc868/resourceGroups/MC\_k8e-pods\_kubernetes-pods\_eastus/providers/Microsoft.Compute/disks/myAKSDisk

1. Now we can create the pod and mount the Azure Disk. Create a new file named azure-disk-pod.yaml with the following contents:

Text

Description automatically generated

1. Run **kubectl apply -f azure-disk-pod.yaml**.

Text

Description automatically generated

1. You now have a running pod with an Azure Disk mounted at /mnt/azure.
2. You can use **kubectl describe pod mypod** to verify the share is mounted successfully. Search for the Volumes section of the output.

Graphical user interface, text, application

Description automatically generated

1. Now exec to the pod and try to access the mounted volume. Run the following command **kubectl exec -it mypod – bash**.

* kubectl exec -it mypod -- sh

1. Go to /mnt/azure and try create a blank file test.txt file.

Text

Description automatically generated

1. Delete everything created by this practice.

Graphical user interface, text

Description automatically generated

# Practice 5: Provisioning Azure Disk storage using Storage Classes

1. Login to Azure and connect to your AKS cluster.
2. Check if any pods run under the default namespace if so delete everything under the default namespace.

Text

Description automatically generated

1. Now we will provision Azure disk and attach it to a running pod but this time using dynamic provisioning with storage classes. List the available storage classes, run **kubectl get sc**.

Graphical user interface, text, application

Description automatically generated

1. Examine the output. Each AKS cluster includes four pre-created storage classes, two of them configured to work with Azure disks, default and managed-premium. We will use the managed-premium in our PVC definition since it uses premium type of disks.
2. Now we will create the PVC that will consume the storage class defined previously. Create a file named azure-premium.yaml and copy in the following YAML:

Text

Description automatically generated

1. Create the persistent volume claim with **the kubectl apply -f azure-premium.yaml**.

Text

Description automatically generated

1. Check the status of your PVC.

Text

Description automatically generated

1. Now we will create the pod that consumes the PVC. Create a file named azure-pvc-disk.yaml, and copy in the following YAML. Make sure that the claimName matches the PVC created in the last step:

Text

Description automatically generated

1. Create the pod with **kubectl apply -f azure-pvc-disk.yaml**.

A blue screen with white text

Description automatically generated with medium confidence

1. Do a describe on the pod and check the volumes mounted.

Graphical user interface, text

Description automatically generated

1. Delete everything created under this practice including the storage class.

Graphical user interface, text

Description automatically generated