**Azure Key Vault FlexVolume for Kubernetes  —  A journey to integrate Azure Key Vault and AKS**

**Introduction :** This module tell us about how to integrate with Azure Key vault and Kubernetes AKS step by step .

## **Introducing Azure Key Vault FlexVolume for Kubernetes**

Azure Key Vault FlexVolume for Kubernetes is a driver that allows you to consume typed data from Azure Key Vault (like secrets, keys or certificates) and attach that data directly to Pods. You can find the project itself [directly on GitHub](https://github.com/Azure/kubernetes-keyvault-flexvol).

# Step 1: Deploy MIC and NMI in kube cluster

1. az group create --name ITF-DEV-DEMO--location eastus // create one resource group
2. Create one Kubernetes cluster name “democluster” from UI/ CLI
3. Create one keyvault instance with name “**consumnersecret123” from UI and CLI**
4. Create one Secret name “URL” in **consumnersecret123**
5. Now login to Azure in CLI – az login
6. Configure the Azure cluster config - az aks get-credentials --resource-group ITF-DEV-DEMO --name democluster
7. Now deploy the component - kubectl create -f https://raw.githubusercontent.com/Azure/aad-pod-identity/master/deploy/infra/deployment.yaml

# Step 2: Create POD identity

# az identity create -g ITF-DEV-DEMO -n demo\_pod\_identity -o json

# Step 3. Assign the Reader role

# Grab the principalId and assign the Reader role for the targeting Resource Group (the resource group where you’ve placed your Azure Key Vault instance) to that identity

1. az role assignment create --role Reader --assignee 3226e21d-b930-4f1d-9863-3e57fd0f9a6c --scope /subscriptions/ subscriptionsid/resourceGroups/ITF-DEV-DEMO

# Step 4 . Assign the Managed Identity Operator

# Grab the <aks\_service\_principal\_id\_here> using -az aks list --resource-group ITF-DEV-DEMO. The second value in the command is <azure\_identitys\_id> from previous create pod identity command

az role assignment create --role "Managed Identity Operator" --assignee 0faf6d59-4bfd-4b82-8823-e2dd527c26bb --scope /subscriptions/efa3402c-d906-4951-8645-08fcc393a151/resourcegroups/ITF-DEV-DEMO/providers/Microsoft.ManagedIdentity/userAssignedIdentities/demo\_pod\_identity

# Step 5 . Deploying an Azure Identity to AKS

# Name : aad\_identity.yaml

* Obviously, name is used later on to identify the instance of AzureIdentity
* spec.type is set to 0 which represents a *Managed Service Identity* (MSI). spec.type could also be set to 1, which tells the *AAD Pod Identity* infrastructure that you want to use a *Service Principal* instead of an *Azure Identity*.
* ResourceId is the id value, taken from your *Azure Identity*.
* ClientId is the clientId value, taken from your *Azure Identity*.

# 

# Step 5 . Adding an AAD Identity binding to AKS

In this yaml file, the Selector property is the most important one. It is used to identify which Pods should run in the context of the linked AzureIdentity. The binding needs also to be deployed to AKS using kubectl create



## Step 6 . Installing Azure Key Vault FlexVolume

kubectl create -f https://raw.githubusercontent.com/Azure/kubernetes-keyvault-flexvol/master/deployment/kv-flexvol-installer.yaml

## **Step 7 . GET Secret permission**

$PrincipalID = az identity show

--name demo\_pod\_identity

--resource-group demo-resource-group

--output json

--query principalId

$ClientID = az identity show

--name demo\_pod\_identity

--resource-group demo-resource-group

--output json

--query clientId

$keyVaultID = az keyvault show

--name demokv

--resource-group demo-resource-group

--output json

--query id

az role assignment create

--role Reader

--assignee $PrincipalID

--scope $keyVaultID

# Grant GET Secret access to the Azure Identity

az keyvault set-policy

--name demokv

--secret-permissions get

--spn $ClientID

Example

az role assignment create --role Reader --assignee 051e47cc-95a0-4139-af13-94e66764145c --scope /subscriptions/efa3402c-d906-4951-8645-08fcc393a151/resourceGroups/ITF-DEV-DEMO/providers/Microsoft.KeyVault/vaults/consumnersecret123

az keyvault set-policy --name consumnersecret123 --secret-permissions get --spn c14fc101-a5a7-437a-bccc-4d3b9588a190

## **STEP-8 : Create a demo Docker Image**

As described during the introduction, the Azure KeyVault FlexVolume will take secrets from Azure Key Vault and make them available inside of containers. The value from every secret is written to an individual file in the specified volumeMount. In order to verify the entire setup of AAD Pod Identity and Azure Key Vault FlexVolume for Kubernetes, we will now create a simple nginx Docker Image and use it later to examine the container and read the secret values at runtime.

Content

FROM alpine

LABEL maintainer="Thorsten Hans <thorsten.hans@gmail.com>"

CMD ["cat", "/kv/URL"]

File

## **STEP 9- build Docker image and push in docker hub**

docker build . -t victordey2007/aks-keyvault-sample:latest

docker login

docker push victordey2007/aks-keyvault-sample:latest

## **STEP 10- Creating the Pod definition**

Create one POD using the attached aks-keyvault-sample.yaml file but replace the following values.

The Azure Key Vault itself is specified from the combination of four properties:

* keyvaultname: Key Vault’s custom domain name ( its given)
* resourcegroup: Key Vault’s Azure-Resource-Group name( its given)
* subscriptionid: Key Vault’s Azure-Subscription Identifier
* tenantid: Key Vault’s Azure-Tenant Identifie



## **STEP11: Deploy in AKs and monitor the logs**

1. kubectl create -f aks-keyvault-spring.yaml
2. kubectl logs aks-kv-spring-pod// in log you should see the value of URL
3. kubectl get pods
4. kubectl describe pods aks-kv-spring-pod