KV Secrets Store CSI Driver for AKS cluster Using Workload Identity

Last updated by | Leslie Cardoza | Aug 9, 2023 at 9:39 AM GMT+2

Contents

Overview

Azure Key Vault Provider for Secrets Store CSI Driver allows for the integration of an Azure key vault as a secret store with an Azure Kubernetes Service (AKS) cluster by mounting it as a CSI volume.

Secrets Store CSI Driver have the following methods to access an Azure key vault:

- An Azure Active Directory pod identity (preview)
- An Azure Active Directory workload identity

Pre-requisites

To Use Secrets store CSI driver, we need to enable two add-ons (Workload Identity and azureKeyVaultSecretsProvider) for the AKS cluster.

OIDC which is already enabled for FSCP 3.0 AKS clusters. OIDC is must for using workload identity.

For more details on OIDC refer

Steps to use Secret store CSI driver

1.Enable add-on

For **New** clusters

We can enable them either at clutser creation by adding csisecretstore add-on to add-on profiles and workload identity in the agentpool profile.

example

```
addonProfiles: {
   ...
     azureKeyVaultSecretsProvider: {
       enabled: true
     }
   }
agentPoolProfiles: [{
...
   securityProfile: {
   workloadIdentity: {
       enabled: true
     }
]}
```

For **Existing** clusters you can use below commands to enable workload identity and secrets store provider

```
#Enabling workload identity
az aks update -g myResourceGroup -n myAKSCluster --enable-oidc-issuer --enable-workload-identity

#Enabling secrets store provider
az aks enable-addons --addons azure-keyvault-secrets-provider --name myAKSCluster --resource-group myResour
```

After we enable the add-on, if you check pods of `kube-system` namespace it should have below pods running

```
$ k get po -n kube-system
aks-secrets-store-csi-driver-swdwr                        3/3     Running   0          24h
aks-secrets-store-csi-driver-vrnp8                        3/3     Running   0          24h
aks-secrets-store-provider-azure-ghjf5                    1/1     Running   0          24h
aks-secrets-store-provider-azure-ltkxq                    1/1     Running   0          24h
azure-wi-webhook-controller-manager-6b8546c7b6-kg2xb     1/1     Running   0          24h
azure-wi-webhook-controller-manager-6b8546c7b6-sqr45     1/1     Running   0          24h
```

2.Configuring workload identity

Once the add-on is enabled check the below documentation to create *service account* and _federated credentials_for Workload identity.

refer: FSCP workload identity

▶ You can refer below example to create service account and federated credentials yaml as example.(Click to expand!)

3.Creating Secreproviderclass object

Once we have the service account and federated credentials created, Create `SecretProviderClass` which is useful to identify on which keyvault and which key/secret we would like to access.

▶ You can refer below example for SecretProviderClass yaml .(Click to expand!)

4.Access KV secret from a test pod.

Now we have to use the service account in our test pod to authenticate with the keyvault mentioned in secret provider class.

▶ You can refer below example for Creating test pod and accessing secrets from keyvault from pod using workload identity .(Click to expand!)
From inside the pod you can check whether you are able to access the secret. example:

```
LAUNCHER+C93289@NLVHPRAABW45151 MINGW64 ~
$ k exec -it -n csikv csikv-tester -- sh
/ # cd /mnt/secrets-store/
/mnt/secrets-store # cat -n csikv
     1  testing-successful #(This is my secret value)
/mnt/secrets-store #
```

Note: Application teams should take care of encryption/ decryption of the secrets while storing in the pod or keyvault

Troubleshooting

- If you are seeing `403 errors`, while using `secretProviderClass` in your application pod, check whether your AKS cluster UAMI has access to keyvault, yo can assign access policy to AKS UAMI using below command.

  example:

  ```
  #this only assigns get permissions to keys, based on your requirement add the permissions
  az keyvault set-policy -n mcpk-d-kv --secret-permissions get --spn a79c2f17-d943-4e39-8510-99972
  ```

- If you are seeing any `connection timeouts` for your application pods while conecting to azure, check whether you have created egress and ingress network policies in your application namespace.

Refer: https://kubernetes.io/docs/concepts/services-networking/network-policies/

References

https://learn.microsoft.com/en-us/azure/aks/workload-identity-overview

https://confluence.int.abnamro.com/display/GRIDAD/FSCP+2+to+3+AKS+Workload+Identity

https://dev.azure.com/cbsp-abnamro/GRD0001007/_wiki/wikis/AKS Documentation/90723/Workload-Identity

https://learn.microsoft.com/en-us/azure/aks/csi-secrets-store-driver

https://learn.microsoft.com/en-us/azure/aks/csi-secrets-store-identity-access