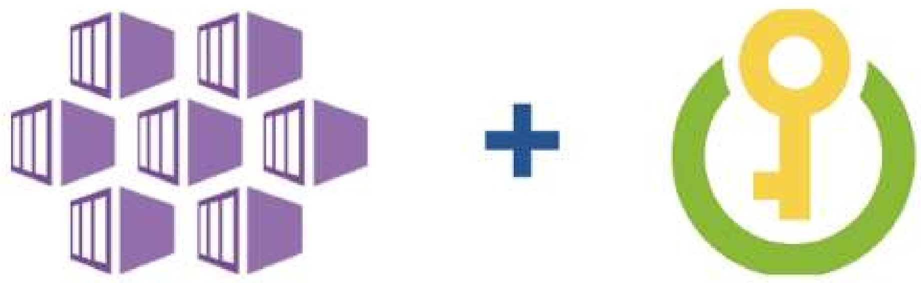
Integrate Azure Key Vault with AKS — Introduction (Part 1/3)

Leon Jalfon
Apr 27, 2020 · 2 min read

In this 3-parts tutorial we will explain how to integrate AKS with Azure Key Vault using "FlexVolumes" and "Azure Key Vault to Kubernetes". However, before we get down to work let's talk a little about each approach.



FlexVolumes

With FlexVolumes Key Vault secrets, keys, and certificates become a volume accessible to pods. Once the volume is mounted, its data is available directly in the container filesystem for your application.

Pros: Easy to use and configure

Cons: Anyone inside the container can see the secrets and the secrets are available only from the file system (not as environment variable)

Azure Key Vault to Kubernetes (akv2k8s)

Azure Key Vault to Kubernetes (akv2k8s) use two main components (Azure Key Vault Controller and Azure Key Vault Env Injector) to inject a secret, key or certificate as environment variable accessible only for the main process of the container.

Pros: Only the main process of the container can access the secrets and it's available as environment variable

Cons: More complicated infrastructure that requires a kubernetes CRD and a Mutating Admission Webhook

Prerequisites

For this tutorial we will need:

- AKS Cluster (v0.0.14 or later)
- Key Vault (including a secret)
- Service Principal with "get" Access to Key Vault

If you don't have the prerequisites above you can configure them following the instructions below

Environment Configuration

- Install and configure Azure CLI (I am using version 2.1.0)

```
https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?
view=azure-cli-latest
```

- Configure your local environment (env variables to use in the next steps)

```
SERVICE_PRINCIPAL_NAME=aks-keyvault-tutorial
RESOURCE_GROUP_NAME=aks-keyvault-tutorial
AKS_CLUSTER_NAME=aks-keyvault-tutorial
KEY_VAULT_NAME=akskeyvaulttutorial   # Must be globally unique
KEY_VAULT_SECRET_NAME=mySecret
KEY_VAULT_SECRET_VALUE=myValue
AZURE_LOCATION=westeurope
```

- Create a Service Principal (store the service principal details retrieved in the output in a secure place, we will need them later)

```
az ad sp create-for-rbac --name ${SERVICE_PRINCIPAL_NAME}
```

- Create a Resource Group

```
az group create --name ${RESOURCE_GROUP_NAME} --location
${AZURE_LOCATION}
```

- Create an AKS cluster

```
az aks create --resource-group ${RESOURCE_GROUP_NAME} --name
${AKS_CLUSTER_NAME} --node-count 1
```

- Create a Key Vault

```
az keyvault create -n ${KEY_VAULT_NAME} -g ${RESOURCE_GROUP_NAME}
```

- Create a Key Vault Secret

```
az keyvault secret set --vault-name ${KEY_VAULT_NAME} --name
${KEY_VAULT_SECRET_NAME} --value ${KEY_VAULT_SECRET_VALUE}
```

- Authorize Access to Secrets for your service principal:

```
az keyvault set-policy --n ${KEY_VAULT_NAME} --spn
${SERVICE_PRINCIPAL_NAME} --secret-permissions get
```

- Connect to the cluster (I assume you have kubectl already installed)

```
az aks get-credentials --resource-group ${RESOURCE_GROUP_NAME} --
name ${AKS_CLUSTER_NAME}
```

- Test connection

```
kubectl get nodes
```

And that's it, we are ready to start!

Integrate Azure Key Vault with AKS — Using "FlexVolume" (Part 2/3)

Integrate Azure Key Vault with AKS — Using "akv2k8s" (Part 3/3)

Kubernetes Azure Keyvault Devsecops DevOps