AKS Demo Lab – With results

You will need the following.

Azure Subscription with credits

Azure CLI - <https://aka.ms/installazurecliwindows>

Docker - <https://docs.docker.com/v17.09/docker-for-windows/install/>

Organization and Project on Dev.Azure.com

Open Elevated command prompt to run the CLI commands. These commands will create the following infrastructure.

1. Resource Group
2. Key Vault
3. Azure Application and Service Principal for access using Keyvault API
4. Azure Container Registry
5. AKS Cluster
6. KubeCtl Extensions for Managed Identity
7. Managed Pod Identity for access using Flex Volume
8. KubeCtl Extensions for Flex Volume
9. Some demo secrets in Key Vault
10. Permissions for those secrets

Note that there are three identities being used. Service principal of the AKS cluster itself. That SP is created by Azure when you create the cluster. Another service Principal that you can use in your application to authenticate and get access token. Third is a managed pod identity that is associated with a pod and you can use it to access secrets with flex volume.

Advantage of this setup

You can access keyvault from within your application using Keyvault API or you can access keyvault from your pod definition in case you are using standard images with your own data store for configured values.

Advantage of using Keyvault is that the CI/CD pipeline is simplified to include on the keyvault information in the variables for build and release. All application specific configuration is completely isolated in keyvault. So that there is no danger of information leaking when you put code and images into public repository/registry. Also the environment Dev/Test/Prod can be changed just by changing the Keyvault URL

**Commands to Run**

**az login**

Note, we have launched a browser for you to login. For old experience with device code, use "az login --use-device-code"

You have logged in. Now let us find all the subscriptions to which you have access...

[

{

"cloudName": "AzureCloud",

"id": "bb507692-5667-43b8-a316-cdddc4b71d23",

"isDefault": true,

"name": "Visual Studio Premium with MSDN",

"state": "Enabled",

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158",

"user": {

"name": "himanshu\_zinzu@hotmail.com",

"type": "user"

}

}

]

Regarding how to set azure subscription when multiple accounts.

“az account set –subscription –s<subscription Id>”

All subscription ids can be found from portal or they are displayed when az login step is done as shown above

**az group create -l eastus -n aksdemo-rg**

{

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg",

"location": "eastus",

"managedBy": null,

"name": "aksdemo-rg",

"properties": {

"provisioningState": "Succeeded"

},

"tags": null,

"type": null

}

**az keyvault create --resource-group aksdemo-rg --name aksdemo-kv --location eastus**

{

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.KeyVault/vaults/aksdemo-kv",

"location": "eastus",

"name": "aksdemo-kv",

"properties": {

"accessPolicies": [

{

"applicationId": null,

"objectId": "5fb378cb-7956-4806-bafa-9f8462362790",

"permissions": {

"certificates": [

"get",

"list",

"delete",

"create",

"import",

"update",

"managecontacts",

"getissuers",

"listissuers",

"setissuers",

"deleteissuers",

"manageissuers",

"recover"

],

"keys": [

"get",

"create",

"delete",

"list",

"update",

"import",

"backup",

"restore",

"recover"

],

"secrets": [

"get",

"list",

"set",

"delete",

"backup",

"restore",

"recover"

],

"storage": [

"get",

"list",

"delete",

"set",

"update",

"regeneratekey",

"setsas",

"listsas",

"getsas",

"deletesas"

]

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158"

}

],

"createMode": null,

"enablePurgeProtection": null,

"enableSoftDelete": null,

"enabledForDeployment": false,

"enabledForDiskEncryption": null,

"enabledForTemplateDeployment": null,

"networkAcls": null,

"provisioningState": "Succeeded",

"sku": {

"name": "standard"

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158",

"vaultUri": "https://aksdemo-kv.vault.azure.net/"

},

"resourceGroup": "aksdemo-rg",

"tags": {},

"type": "Microsoft.KeyVault/vaults"

}

**az keyvault secret set --name SampleSecret --value SecretValue --description "Sample Secret Value" --vault-name aksdemo-kv**

{

"attributes": {

"created": "2019-08-13T00:28:54+00:00",

"enabled": true,

"expires": null,

"notBefore": null,

"recoveryLevel": "Purgeable",

"updated": "2019-08-13T00:28:54+00:00"

},

"contentType": "Sample Secret Value",

"id": "https://aksdemo-kv.vault.azure.net/secrets/SampleSecret/5c07740661ba4fd3b015c498651a58cf",

"kid": null,

"managed": null,

"tags": {

"file-encoding": "utf-8"

},

"value": "SecretValue"

}

**az ad app create --display-name aks-demo-kv-reader --identifier-uris https://aks-demo-kv-reader.neudesic.com --query objectId**

"9dc6a8cf-f66d-41c9-9f6b-08799c9a63f2"

**az ad sp create --id 9dc6a8cf-f66d-41c9-9f6b-08799c9a63f2 --query appId**

"4f41878b-2cc2-4cb6-9d5c-a45497b5ea2b"

Application Id 4f41878b-2cc2-4cb6-9d5c-a45497b5ea2b

Create these manually in

Open the [Azure Portal](https://portal.azure.com/) in your Browser and Navigate to the AD App Registration you created a minute ago. From the Azure DASHBOARD go to AZURE ACTIVE DIRECTORY and open the APP REGISTRATIONS blade. Here, select the Registration named: aks-demo-kv-reader from the list of all registrations. Click on SETTINGS (purple square), KEYS (green square), provide a new description and set an expiration date for the Key (yellow square). Copy the key name and value

AksDemoClientSecret

Keyvalue =.6XLw/\*NqGOJg4ygF75K1b0ASlJ-E:N - 1bdj=\*0IZ?ATF+JtqrOfUXChiVDTLx59

**az keyvault set-policy --name aksdemo-kv --spn 4f41878b-2cc2-4cb6-9d5c-a45497b5ea2b --secret-permissions get list**

{

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.KeyVault/vaults/aksdemo-kv",

"location": "eastus",

"name": "aksdemo-kv",

"properties": {

"accessPolicies": [

{

"applicationId": null,

"objectId": "5fb378cb-7956-4806-bafa-9f8462362790",

"permissions": {

"certificates": [

"get",

"list",

"delete",

"create",

"import",

"update",

"managecontacts",

"getissuers",

"listissuers",

"setissuers",

"deleteissuers",

"manageissuers",

"recover"

],

"keys": [

"get",

"create",

"delete",

"list",

"update",

"import",

"backup",

"restore",

"recover"

],

"secrets": [

"get",

"list",

"set",

"delete",

"backup",

"restore",

"recover"

],

"storage": [

"get",

"list",

"delete",

"set",

"update",

"regeneratekey",

"setsas",

"listsas",

"getsas",

"deletesas"

]

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158"

},

{

"applicationId": null,

"objectId": "c5dc922d-5255-407e-a933-eea3e80a1f76",

"permissions": {

"certificates": null,

"keys": null,

"secrets": [

"get",

"list"

],

"storage": null

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158"

}

],

"createMode": null,

"enablePurgeProtection": null,

"enableSoftDelete": null,

"enabledForDeployment": false,

"enabledForDiskEncryption": null,

"enabledForTemplateDeployment": null,

"networkAcls": null,

"provisioningState": "Succeeded",

"sku": {

"name": "standard"

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158",

"vaultUri": "https://aksdemo-kv.vault.azure.net/"

},

"resourceGroup": "aksdemo-rg",

"tags": {},

"type": "Microsoft.KeyVault/vaults"

}

Grant Reader role to aks-demo-kv-reader in Keyvault

**az role assignment create --role Reader --assignee** 4f41878b-2cc2-4cb6-9d5c-a45497b5ea2b **--scope** /subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.KeyVault/vaults/aksdemo-kv

kubectl create secret generic kvcreds --from-literal clientid=4f41878b-2cc2-4cb6-9d5c-a45497b5ea2b --from-literal clientsecret=6XLw/\*NqGOJg4ygF75K1b0ASlJ-E:N --type="azure/kv"

**Create Private Container Registry**

**az acr create -n aksdemos -g aksdemo-rg --sku Standard**

{

"adminUserEnabled": false,

"creationDate": "2019-08-13T00:52:45.331733+00:00",

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.ContainerRegistry/registries/aksdemos",

"location": "eastus",

"loginServer": "aksdemos.azurecr.io",

"name": "aksdemos",

"networkRuleSet": null,

"provisioningState": "Succeeded",

"resourceGroup": "aksdemo-rg",

"sku": {

"name": "Standard",

"tier": "Standard"

},

"status": null,

"storageAccount": null,

"tags": {},

"type": "Microsoft.ContainerRegistry/registries"

}

Enable Admin user on Portal Copy Password1 NfmvG4tkd89bNll3wddClcjrO/vITrET Password2 1tpEvsQ+ulvDzCmprBBmmbA319fKVu3i

**Create the AKS cluster**

**az aks create --resource-group aksdemo-rg --name AKSDemoCluster --node-count 1 --enable-addons monitoring --generate-ssh-keys --enable-rbac**

[K{- Finished ..ion done[############################################] 100.0000%

"aadProfile": null,

"addonProfiles": {

"omsagent": {

"config": {

"logAnalyticsWorkspaceResourceID": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/defaultresourcegroup-eus/providers/microsoft.operationalinsights/workspaces/defaultworkspace-bb507692-5667-43b8-a316-cdddc4b71d23-eus"

},

"enabled": true

}

},

"agentPoolProfiles": [

{

"availabilityZones": null,

"count": 1,

"enableAutoScaling": null,

"maxCount": null,

"maxPods": 110,

"minCount": null,

"name": "nodepool1",

"orchestratorVersion": "1.12.8",

"osDiskSizeGb": 100,

"osType": "Linux",

"provisioningState": "Succeeded",

"type": "AvailabilitySet",

"vmSize": "Standard\_DS2\_v2",

"vnetSubnetId": null

}

],

"apiServerAuthorizedIpRanges": null,

"dnsPrefix": "AKSDemoClu-aksdemo-rg-bb5076",

"enablePodSecurityPolicy": null,

"enableRbac": true,

"fqdn": "aksdemoclu-aksdemo-rg-bb5076-8474265d.hcp.eastus.azmk8s.io",

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/aksdemo-rg/providers/Microsoft.ContainerService/managedClusters/AKSDemoCluster",

"kubernetesVersion": "1.12.8",

"linuxProfile": {

"adminUsername": "azureuser",

"ssh": {

"publicKeys": [

{

"keyData": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC/zgr3xn1Xl8IkUPuN7wiClDqRA3YlZDuccyNNwFhAzgBXBBZgC5vu74G+xJmFKPCgxwsNIgmtoyUOHSMABCQORF7XpAfMwulHk+wJJk+pFe/HLPhRJviXla0xFJ/h/e0JkR4slqurWPdNr/JzkrR1WRHKks6IMpZg6Z2t1VWr1Mm0ENqt7gktJBX4q2lBQ58sEJg9BKVG/K5dStN8WHVpClIUceGRkiM+ayFLMYr2pt34uXZDvb2gVx4YwrkUBAfXoMXw4267nVLqHoYikVXMICDV+5GaqohMBKj7kx9mgp1BSB0XXBsjD1rdxcKPqjVN3TQXxscrhBNhdsElT2FD"

}

]

}

},

"location": "eastus",

"name": "AKSDemoCluster",

"networkProfile": {

"dnsServiceIp": "10.0.0.10",

"dockerBridgeCidr": "172.17.0.1/16",

"networkPlugin": "kubenet",

"networkPolicy": null,

"podCidr": "10.244.0.0/16",

"serviceCidr": "10.0.0.0/16"

},

"nodeResourceGroup": "MC\_aksdemo-rg\_AKSDemoCluster\_eastus",

"provisioningState": "Succeeded",

"resourceGroup": "aksdemo-rg",

"servicePrincipalProfile": {

"clientId": "df01b1cd-dcf3-48d4-a445-910035ef82ed",

"secret": null

},

"tags": null,

"type": "Microsoft.ContainerService/ManagedClusters"

}

**az aks install-cli**

Downloading client to "C:\Users\himanshu.zinzuwadia\.azure-kubectl\kubectl.exe" from "https://storage.googleapis.com/kubernetes-release/release/v1.15.2/bin/windows/amd64/kubectl.exe"

Please add "C:\Users\himanshu.zinzuwadia\.azure-kubectl" to your search PATH so the `kubectl.exe` can be found. 2 options:

1. Run "set PATH=%PATH%;C:\Users\himanshu.zinzuwadia\.azure-kubectl" or "$env:path += 'C:\Users\himanshu.zinzuwadia\.azure-kubectl'" for PowerShell. This is good for the current command session.

2. Update system PATH environment variable by following "Control Panel->System->Advanced->Environment Variables", and re-open the command window. You only need to do it once

**Alternatively if you don’t want to set path, you can download and kubectl from any directory you need to.**

**Download curl to working directory**[**https://curl.haxx.se/windows/dl-7.65.3\_1/curl-7.65.3\_1-win64-mingw.zip**](https://curl.haxx.se/windows/dl-7.65.3_1/curl-7.65.3_1-win64-mingw.zip)

**From working directory run the following.** This will download kubectl.exe.

**curl -LO** [**https://storage.googleapis.com/kubernetes-release/release/v1.15.0/bin/windows/amd64/kubectl.exe**](https://storage.googleapis.com/kubernetes-release/release/v1.15.0/bin/windows/amd64/kubectl.exe)

**Install aks credentials**

**az aks get-credentials --resource-group aksdemo-rg --name AKSDemoCluster**

Merged "AKSDemoCluster" as current context in C:\Users\himanshu.zinzuwadia\.kube\config

Enter Registry Credentials in KubeCtl

**kubectl create secret docker-registry regcred --docker-server=** aksdemos.azurecr.io **--docker-username=aksdemos --docker-password=** NfmvG4tkd89bNll3wddClcjrO/vITrET **--docker-email=Himanshu.zinzuwadia@neudesic.com**

**kubectl get nodes**

NAME STATUS ROLES AGE VERSION

aks-nodepool1-15636592-0 Ready agent 9m v1.12.8

**Install Managed Identity**

**kubectl create -f https://raw.githubusercontent.com/Azure/aad-pod-identity/master/deploy/infra/deployment-rbac.yaml**

serviceaccount "aad-pod-id-nmi-service-account" created

customresourcedefinition.apiextensions.k8s.io "azureassignedidentities.aadpodidentity.k8s.io" created

customresourcedefinition.apiextensions.k8s.io "azureidentitybindings.aadpodidentity.k8s.io" created

customresourcedefinition.apiextensions.k8s.io "azureidentities.aadpodidentity.k8s.io" created

customresourcedefinition.apiextensions.k8s.io "azurepodidentityexceptions.aadpodidentity.k8s.io" created

clusterrole.rbac.authorization.k8s.io "aad-pod-id-nmi-role" created

clusterrolebinding.rbac.authorization.k8s.io "aad-pod-id-nmi-binding" created

daemonset.extensions "nmi" created

serviceaccount "aad-pod-id-mic-service-account" created

clusterrole.rbac.authorization.k8s.io "aad-pod-id-mic-role" created

clusterrolebinding.rbac.authorization.k8s.io "aad-pod-id-mic-binding" created

deployment.extensions "mic" created

Create an Azure Identity to use a managed identity

**az identity create -g MC\_aksdemo-rg\_AKSDemoCluster\_eastus -n aksdemo\_pod\_identity -o json - Note you need to create this in the Azure created resource group which will be prefixed with MC\_**

**MC\_aksdemo-rg\_AKSDemoCluster\_eastus**

{

"clientId": "fd288811-6a4e-4f1f-931c-e9b90896c8f1",

"clientSecretUrl": "https://control-eastus.identity.azure.net/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/aksdemo-rg/providers/Microsoft.ManagedIdentity/userAssignedIdentities/aksdemo\_pod\_identity/credentials?tid=5413df59-d861-48ae-b40f-d1c9de5f9158&oid=b924fe03-4797-4618-b475-248fdabe74ad&aid=fd288811-6a4e-4f1f-931c-e9b90896c8f1",

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/mc\_aksdemo-rg\_AKSDemoCluster\_eastus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/aksdemo\_pod\_identity",

"location": "eastus",

"name": "aksdemo\_pod\_identity",

"principalId": "b924fe03-4797-4618-b475-248fdabe74ad",

"resourceGroup": "aksdemo-rg",

"tags": {},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158",

"type": "Microsoft.ManagedIdentity/userAssignedIdentities"

}

Note down principalId returned above also recollect subscription id noted in the first step

Aksdemo\_pod\_identity needs read access to resource group containing keyvault.

**az role assignment create --role Reader --assignee** b924fe03-4797-4618-b475-248fdabe74ad **--scope /subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/ MC\_aksdemo-rg\_AKSDemoCluster\_eastus**

{

"canDelegate": null,

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/aksdemo-rg/providers/Microsoft.Authorization/roleAssignments/99a46b78-c660-4516-bd14-4863d8b33064",

"name": "99a46b78-c660-4516-bd14-4863d8b33064",

"principalId": "b924fe03-4797-4618-b475-248fdabe74ad",

"resourceGroup": "aksdemo-rg",

"roleDefinitionId": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/providers/Microsoft.Authorization/roleDefinitions/acdd72a7-3385-48ef-bd42-f606fba81ae7",

"scope": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/aksdemo-rg",

"type": "Microsoft.Authorization/roleAssignments"

}

Service Principal Id was noted in the step az ad sp create

Azure identity id is from the return value of az identity create

AKS Service Principal Id **df01b1cd-dcf3-48d4-a445-910035ef82ed from output of az aks above goes under –assignee. the highlighted id of aksdemo\_pod\_identity above goes in –scope**

The aks cluster service principal needs Managed Identity Operator role assignment to obtain token on behalf of the aksdemo\_pod\_identity

**az role assignment create --role "Managed Identity Operator" --assignee df01b1cd-dcf3-48d4-a445-910035ef82ed --scope** "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/ **MC\_aksdemo-rg\_AKSDemoCluster\_eastus** /providers/Microsoft.ManagedIdentity/userAssignedIdentities/aksdemo\_pod\_identity **"**

{

"canDelegate": null,

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/ MC\_aksdemo-rg\_AKSDemoCluster\_eastus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/aksdemo\_pod\_identity/providers/Microsoft.Authorization/roleAssignments/11e53f56-ee1e-4112-9c16-eb58f46994c8",

"name": "11e53f56-ee1e-4112-9c16-eb58f46994c8",

"principalId": "804d903f-011a-49ed-b1a7-3404db51abe7",

"resourceGroup": " **MC\_aksdemo-rg\_AKSDemoCluster\_eastus** ",

"roleDefinitionId": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/providers/Microsoft.Authorization/roleDefinitions/f1a07417-d97a-45cb-824c-7a7467783830",

"scope": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/MC\_aksdemo-rg\_AKSDemoCluster\_eastus/providers/Microsoft.ManagedIdentity/userAssignedIdentities/aksdemo\_pod\_identity",

"type": "Microsoft.Authorization/roleAssignments"

}

**Now create aad\_identity.yaml file containing the Managed Operator Id above and ClientId is the id of aksdemo\_pod\_identity created before.**

**Contents of the file**

apiVersion: "aadpodidentity.k8s.io/v1"

kind: AzureIdentity

metadata:

name: demo-aks-pod-id

spec:

type: 0

ResourceID: /subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourcegroups/aksdemo-rg/providers/Microsoft.ManagedIdentity/userAssignedIdentities/aksdemo\_pod\_identity

ClientID: fd288811-6a4e-4f1f-931c-e9b90896c8f1

**kubectl create -f aad\_identity.yaml**

azureidentity.aadpodidentity.k8s.io "demo-aks-pod-id" created

Azure Identity Binding file aad\_identity\_binding.yaml as below.

apiVersion: "aadpodidentity.k8s.io/v1"

kind: AzureIdentityBinding

metadata:

name: demo-aad-identity-binding

spec:

AzureIdentity: aksdemo\_pod\_identity

Selector: demo\_app

Now create aad\_identity\_binding

**kubectl create -f aad\_identity\_binding.yaml**

azureidentitybinding.aadpodidentity.k8s.io "demo-aad-identity-binding" created

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

namespace "kv" created

daemonset.extensions "keyvault-flexvolume" created

**Create some more secrets to test with Flexvolume**

**az keyvault secret set --vault-name aksdemo-kv --name sample1 --value "1st sensitive value"**

{

"attributes": {

"created": "2019-08-13T01:45:01+00:00",

"enabled": true,

"expires": null,

"notBefore": null,

"recoveryLevel": "Purgeable",

"updated": "2019-08-13T01:45:01+00:00"

},

"contentType": null,

"id": "https://aksdemo-kv.vault.azure.net/secrets/sample1/8a4e58fe6f52416992dbf939c3ca1b1f",

"kid": null,

"managed": null,

"tags": {

"file-encoding": "utf-8"

},

"value": "1st sensitive value"

}

**az keyvault secret set --vault-name aksdemo-kv --name sample2 --value "2nd sensitive value"**

{

"attributes": {

"created": "2019-08-13T01:46:05+00:00",

"enabled": true,

"expires": null,

"notBefore": null,

"recoveryLevel": "Purgeable",

"updated": "2019-08-13T01:46:05+00:00"

},

"contentType": null,

"id": "https://aksdemo-kv.vault.azure.net/secrets/sample2/017c3c33857a422f8d011a848ddfdcdd",

"kid": null,

"managed": null,

"tags": {

"file-encoding": "utf-8"

},

"value": "2nd sensitive value"

}

**Grant Access to those secrets for the Managed Pod Identity**

**Change to PS**

**PS C:\windows\system32> $PrincipalID = az identity show --name aksdemo\_pod\_identity --resource-group aksdemo-rg --output json --query principalId**

**PS C:\windows\system32> $ClientID = az identity show --name aksdemo\_pod\_identity --resource-group aksdemo-rg --output json --query clientId**

**PS C:\windows\system32> $keyVaultID = az keyvault show --name aksdemo-kv --resource-group aksdemo-rg --output json --query id**

>>

**PS C:\windows\system32> az role assignment create --role Reader --assignee $PrincipalID --scope $keyVaultID**

{

"canDelegate": null,

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.KeyVault/vaults/aksdemo-kv/providers/Microsoft.Authorization/roleAssignments/9b2a3770-76f6-4888-9824-9764e6b9804e",

"name": "9b2a3770-76f6-4888-9824-9764e6b9804e",

"principalId": "b924fe03-4797-4618-b475-248fdabe74ad",

"resourceGroup": "aksdemo-rg",

"roleDefinitionId": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/providers/Microsoft.Authorization/roleDefinitions/acdd72a7-3385-48ef-bd42-f606fba81ae7",

"scope": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.KeyVault/vaults/aksdemo-kv",

"type": "Microsoft.Authorization/roleAssignments"

}

**PS C:\windows\system32> az keyvault set-policy --name aksdemo-kv --secret-permissions get --spn $ClientID**

{

"id": "/subscriptions/bb507692-5667-43b8-a316-cdddc4b71d23/resourceGroups/aksdemo-rg/providers/Microsoft.KeyVault/vaults/aksdemo-kv",

"location": "eastus",

"name": "aksdemo-kv",

"properties": {

"accessPolicies": [

{

"applicationId": null,

"objectId": "5fb378cb-7956-4806-bafa-9f8462362790",

"permissions": {

"certificates": [

"get",

"list",

"delete",

"create",

"import",

"update",

"managecontacts",

"getissuers",

"listissuers",

"setissuers",

"deleteissuers",

"manageissuers",

"recover"

],

"keys": [

"get",

"create",

"delete",

"list",

"update",

"import",

"backup",

"restore",

"recover"

],

"secrets": [

"get",

"list",

"set",

"delete",

"backup",

"restore",

"recover"

],

"storage": [

"get",

"list",

"delete",

"set",

"update",

"regeneratekey",

"setsas",

"listsas",

"getsas",

"deletesas"

]

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158"

},

{

"applicationId": null,

"objectId": "c5dc922d-5255-407e-a933-eea3e80a1f76",

"permissions": {

"certificates": null,

"keys": null,

"secrets": [

"get",

"list"

],

"storage": null

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158"

},

{

"applicationId": null,

"objectId": "c1d4d64e-a502-4e2e-b822-db37e36fc18c",

"permissions": {

"certificates": null,

"keys": null,

"secrets": [

"get"

],

"storage": null

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158"

}

],

"createMode": null,

"enablePurgeProtection": null,

"enableSoftDelete": null,

"enabledForDeployment": false,

"enabledForDiskEncryption": null,

"enabledForTemplateDeployment": null,

"networkAcls": null,

"provisioningState": "Succeeded",

"sku": {

"name": "standard"

},

"tenantId": "5413df59-d861-48ae-b40f-d1c9de5f9158",

"vaultUri": "https://aksdemo-kv.vault.azure.net/"

},

"resourceGroup": "aksdemo-rg",

"tags": {},

"type": "Microsoft.KeyVault/vaults"

}

Create a dockerfile like this

FROM alpine:3.7

LABEL maintainer="Himanshu Zinzuwadian<Himanshu.zinzuwadia@neudesic.com"

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

**This will display the value of secret – SecretValue in the container logs.**

**To keep container running use this command in the last line. instead** CMD tail -f /dev/null

**docker login**

**docker build . -t askdemos.azurecr.io/aks-keyvault-sample:latest**

Sending build context to Docker daemon 665.6kB

Step 1/3 : FROM alpine:3.7

3.7: Pulling from library/alpine

5d20c808ce19: Pull complete

Digest: sha256:8421d9a84432575381bfabd248f1eb56f3aa21d9d7cd2511583c68c9b7511d10

Status: Downloaded newer image for alpine:3.7

---> 6d1ef012b567

Step 2/3 : LABEL maintainer="Himanshu Zinzuwadian<Himanshu.zinzuwadia@neudesic.com"

---> Running in 54876a064be9

Removing intermediate container 54876a064be9

---> 719b020183de

Step 3/3 : CMD ["cat", "/kv/mysecret"]

---> Running in 73a1694300f5

Removing intermediate container 73a1694300f5

---> ce5b42fe524f

Successfully built ce5b42fe524f

Successfully tagged askdemos.azurecr.io/aks-keyvault-sample:latest

SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and directories added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions for sensitive files and directories.

**Docker login aksdemos.azurecr.io**

Username and password from portal acr repository

**docker push aksdemos.azurecr.io/aks-keyvault-sample:latest**

The push refers to repository [aksdemos.azurecr.io/aks-keyvault-sample]

3fc64803ca2d: Pushed

latest: digest: sha256:402ad80d4c655459ab97da1dfa875de9ab6a5674e124a1004975900939c84f12 size: 528

**Though we created managed identity in the steps earlier, there is something broken in the NMI extension and the pod keeps timing out. So we are going to use service principal creds to demo flex volume**

create a aks-kv-sample-pod.yaml file like this

apiVersion: v1

kind: Pod

metadata:

name: aks-kv-sample-pod

labels:

app: aks-kv-demo

spec:

containers:

- name: aks-kv-container

image: aksdemos.azurecr.io/aks-keyvault-sample:latest // you also can use nginx for testing

imagePullPolicy: Always

restartPolicy: Never

volumeMounts:

- name: kvvolume

mountPath: /kv

readOnly: true

imagePullSecrets:

- name: regcred

volumes:

- name: kvvolume

flexVolume:

driver: "azure/kv"

secretRef:

name: kvcreds

options:

usepodidentity: "false"

keyvaultname: "aksdemo-kv"

keyvaultobjectnames: "SampleSecret"

keyvaultobjecttypes: "secret"

resourcegroup: "aksdemo-rg"

subscriptionid: "bb507692-5667-43b8-a316-cdddc4b71d23"

tenantid: "5413df59-d861-48ae-b40f-d1c9de5f9158"

**If you want to try using managed identity then change the above yaml file usepodidentity:”true” and add label directly under app.** aadpodidbinding: "demo\_app". In that case the secretRef: section is not required.

## Grant AKS access to ACR

When you create an AKS cluster, Azure also creates a service principal to support cluster operability with other Azure resources. You can use this auto-generated service principal for authentication with an ACR registry. To do so, you need to create an Azure AD [role assignment](https://docs.microsoft.com/en-us/azure/role-based-access-control/overview#role-assignments) that grants the cluster's service principal access to the container registry.

Use the following script on Cloud Shell to grant the AKS-generated service principal pull access to an Azure container registry. Modify the AKS\_\* and ACR\_\* variables for your environment before running the script.

#!/bin/bash

AKS\_RESOURCE\_GROUP=myAKSResourceGroup

AKS\_CLUSTER\_NAME=myAKSCluster

ACR\_RESOURCE\_GROUP=myACRResourceGroup

ACR\_NAME=myACRRegistry

# Get the id of the service principal configured for AKS

CLIENT\_ID=$(az aks show --resource-group $AKS\_RESOURCE\_GROUP --name $AKS\_CLUSTER\_NAME --query "servicePrincipalProfile.clientId" --output tsv)

# Get the ACR registry resource id

ACR\_ID=$(az acr show --name $ACR\_NAME --resource-group $ACR\_RESOURCE\_GROUP --query "id" --output tsv)

# Create role assignment

az role assignment create --assignee $CLIENT\_ID --role acrpull --scope $ACR\_ID

**Or you can do this from Portal on ACR role assignments add role assignment ACRPull and add user AKSDemo (or whatever is your cluster user assigned when you created cluster)**

**You can create your first deployment now.**

**kubectl create -f aks-kv-sample-pod.yaml**

**Try it. kubectl exec -it** aks-kv-sample-pod **cat /kv/SampleSecret**

**Note the following is required for Dev Ops exercise that is based on aksworkshop.io site.**

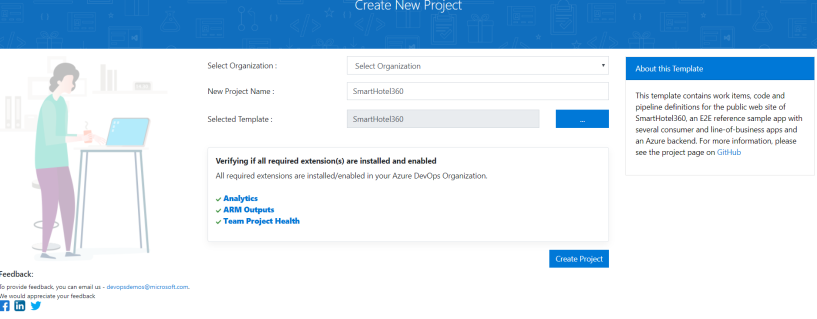
**Install and configure Helm**

**Install Mongo DB using Helm**

**Dev Ops**

In order to get started with AzureDevops follow these simple steps: –

* Browse to [devops.azure.com](http://devops.azure.com/)
* Sign Up
* Create a new Organization
* Browse to <https://azuredevopsdemogenerator.azurewebsites.net/>
* Sign In
* Fill out the form like so



Follow the directions below to setup a Sample DevOps project in your own registry and cluster.

First setup MongoDB on your cluster. You need this because the sample uses it.

<https://aksworkshop.io/#db>

Then setup this Devops project

[https://aksworkshop.io/#cicd](https://aksworkshop.io/" \l "cicd)

Make sure to update the files in manifest directory and instructions with your private registry