­­­­AD SSO Deployment for AKSHCI

# Overview

Without Active Directory Authentication, users have to rely on certificate based kubeconfig when connecting to the api-server via kubectl (command line tool). This kubeconfig contains secrets such as private keys and certificates that need to be carefully distributed which represents a significant security risk.

In lieu of using certificate based kubeconfig, an alternate and secure way is to use Active Directory (AD) logged in credentials (SSO) to connect to the api-server. AD integration with AKS-HCI lets a user on a windows domain joined machine connect to the api-server (via kubectl) using their SSO credentials. This removes the need to manage and distribute certificate based kubeconfigs that contain raw credential information.

AD integration uses AD kubeconfigs, these are distinct from certificate based kubeconfigs and don’t contain any secrets.

The certificate based kubeconfig can be used for backup purposes such as troubleshooting if there are any issues with connecting with Active Directory credentials.

Another security benefit with AD integration is that the users and groups are stored as [SIDs](https://docs.microsoft.com/en-us/troubleshoot/windows-server/identity/security-identifiers-in-windows), unlike group names, [SIDs](https://docs.microsoft.com/en-us/troubleshoot/windows-server/identity/security-identifiers-in-windows) are immutable and unique and pose no naming conflict.

***Note****: We are currently rolling out AD SSO support for managing target clusters only.*

# Concepts

There are several steps involved in setting up Active Directory as the identity provider and enabling SSO via kubectl.

* Before beginning, create the AD account for the api-server and then create [keytab](https://web.mit.edu/kerberos/krb5-devel/doc/basic/keytab_def.html) file associated with the account, refer to below section for creating AD account and generating the keytab file
* Next, use the [keytab](https://web.mit.edu/kerberos/krb5-devel/doc/basic/keytab_def.html) file to install AD Auth on the K8 cluster, a default RBAC configuration is automatically created as part of this step
* Use helper tool below to convert group names to SIDs and vice-versa while creating or editing RBAC configurations

Before starting, ensure the latest PowerShell modules have been downloaded to the windows server machine. Refer **this link** for guidance on downloading the PowerShell modules. Follow **this link** to set up mgmt. cluster.

# **Part A: Use keytab file to install AD Auth**

Step 1: Install target cluster with “-enableADAuth” option

*New-AksHciCluster -clusterName* ***mynewcluster1*** *-kubernetesVersion v1.18.8 -controlPlaneNodeCount 1 -linuxNodeCount 1 -windowsNodeCount 0 -controlPlaneVmSize Standard\_A2\_v2 -loadBalancerVmSize Standard\_A2\_v2 -linuxNodeVmSize Standard\_K8S3\_v1 -windowsNodeVmSize Standard\_K8S3\_v1 -****enableADAuth***

For details on creating the target cluster, please refer to this link

Few things to note before proceeding to the next steps

1. The keytab file is available before continuing
2. The keytab file is generated for specific SPN (service principal name), this SPN is corresponding to the api-server AD account for the target cluster, ensure the same SPN is used in the next command
3. One api-server AD account is created per target cluster
4. **Note, the keytab should be named as current.keytab file**

Step 2: Install AD Authentication

*Install-AksHciAdAuth -clusterName mynewcluster1 -keytab .\current.keytab -SPN k8s/apiserver@BAKSDOM.NTTEST.MICROSOFT.COM -adminUser BAKSDOM\bugbash*

Few things to note before proceeding to the next steps

1. The keytab file is named exactly as **current.keytab**
2. Replace the SPN corresponding to your environment
3. -adminUser creates a corresponding role binding for the specified AD group with cluster admin privileges, replace “bugbash” with AD group or user corresponding to your environment
4. Pass the admin username or group name in SID format if the **cluster host is not domain joined, refer example below**

*Install-AksHciAdAuth -clusterName mynewcluster1 -keytab .\current.keytab -SPN k8s/apiserver@BAKSDOM.NTTEST.MICROSOFT.COM -adminuserSID (or admingroupSID)*

Refer to this section below on how to get the SID

Step 3: Test to make sure AD webhook is running and keytab is stored as K8 secret

1. Generate kubeconfig (note this is the cert based kubeconfig), we will use this to connect to the cluster as local host, use command *Get-AksHciCredential mynewCluster1*
2. Run kubectl on the server you are connected using the certificate based kubeconfig generated in the prior step and check for the AD webhook deployment, it is of the form “ad-auth-webhook-XXXX..”,   
   *kubectl.exe get pods -n=kube-system*
3. Similarly, run kubectl to check keytab is deployed as a secret, keytab should be listed as a K8 secret,  
   *kubectl*.exe get secrets -n=kube-system

Few things to note before proceeding to the next steps

1. Once the AD webhook and keytab are successfully deployed, generate the AD kubeconfig
2. The AD kubeconfig should be copied over to the client machine and this is used to authenticate to the api-server, unlike certificate based kubeconfig, AD kubeconfig is not a secret and is safe to copy as plain text

Step 4: Generate the AD kubeconfig

*Get-AksHciCredential -clusterName mynewcluster1 -outputLocation .\AdKubeconfig -adAuth*

Step 5: Copy the following files from the server to your client machine, create a folder c:\adsso and copy the files

1. AdKubeconfig file created in the previous step
2. Kubectl.exe under $env:ProgramFiles\AksHci
3. Kubectl-adssso.exe under $env:ProgramFiles\AksHci

# **Part B: Creating and Updating AD group role binding**

As mentioned in step 2, a default role binding is created for the mentioned group with cluster admin privileges.

While creating or editing additional AD group RBAC entries, the subject name should be pre-fixed by “microsoft:activedirectory:CONTOSO\<group name>”

Note: Before deploying the yaml the <group name> **should always be converted to SID using the command**

*Kubectl-adsso.exe nametosid <rbac.yml>*

Similarly, in order to update an existing rbac, SID can be converted to user friendly group name before making changes

*Kubectl-adsso.exe sidtoname <rbac.yml>*

# **Part C: Troubleshooting and Maintenance**

## Changing the AD account password associated with api-server account

When the password is changed for the api-server account, uninstall the AD Auth add-on and reinstall it using the updated current and previous keytabs. **The files must be named current.keytab and previous.keytab respectively**. Note, the existing role bindings are not affected by this.

*Uninstall-AksHciAdAuth -clusterName <mynewcluster>*

*Install-AksHciAdAuth -clusterName <mynewcluster> -keytab <.\current.keytab> -previousKeytab <.\previous.keytab> -SPN <service/principal@CONTOSO.COM> -adminUser <CONTOSO\Bob>*

**The keytabs should be named exactly as current.keytab and previous.keytab**

## Uninstall and reinstall AD SSO

Use the following command to uninstall and re-install AD SSO, note, this doesn’t require the api-server to be restarted

*Uninstall-AksHciAdAuth -clusterName <mynewcluster>*

*Install-AksHciAdAuth -clusterName <mynewcluster> -keytab <.\current.keytab> -SPN <service/principal@CONTOSO.COM> -adminUser <CONTOSO\Bob>*

# **Additional: Creating api-server AD Account, SPN and keytab**

There are two steps involved in this process. First, create a new AD account/user for the api-server with service principal option. Next, create keytab file for the AD account.

Step 1: Create a new AD account / user for the api-server with service principal option

*New-ADUser -Name "<apiserver>"*

*-ServicePrincipalNames "<K8s/apiserver>"*

*-AccountPassword (ConvertTo-SecureString "<password>" -AsPlainText -Force)*

*-KerberosEncryptionType <AES128,AES256>*

*-Enabled 1*

Example:

*New-ADUser -Name "apiserver\_acct" -ServicePrincipalNames "k8s/apiserver" -AccountPassword (ConvertTo-SecureString "p@$$w0rd" -AsPlainText -Force) -KerberosEncryptionType AES256 -Enabled 1*

Step 1: Create a new AD account / user for the api-server with service principal option

*ktpass /out <current.keytab> /princ <service/principal@CONTOSO.COM> /mapuser <Domain\account> /crypto all /pass <password> /ptype <KRB5\_NT\_PRINCIPAL>*

Example:

*ktpass /out current.keytab /princ k8s/apiserver@BAKSDOM.NTTEST.MICROSOFT.COM /mapuser baksdom\apiserver\_acct /crypto all /pass p@$$w0rd /ptype KRB5\_NT\_PRINCIPAL*

# **Reference: How to get SID**

## Getting SID associated with your account

On the command line on your HOME directory type *whoami/user*

## Getting SID associated with another account

On the command line on your HOME directory

* Type wmic
* Wmic prompt will show up
* Then type useraccount where name=”<USERNAME>” get sid