Always Encrypted: Set Up and Configuration

Last updated by | Soma Jagadeesh | Jan 11, 2021 at 12:18 AM PST

Contents

Issue

Customer is asking for help on setting AlwaysEncrypted with PowerShell and Azure Key Vault

Mitigation

CREATE TABLE ALWAYSENCRYPTEDTABLE2 (ID INT IDENTITY PRIMARY KEY, CHARACTERS VARCHAR(MAX), OTHER_COL VARCHAR(MAX));

DECLARE @COUNT INT; SET @COUNT = 1;

WHILE @COUNT <= 10 BEGIN INSERT INTO ALWAYSENCRYPTEDTABLE2(CHARACTERS, OTHER_COL) VALUES (REPLICATE('ASDF', 2), REPLICATE('ASDF', 2)); SET @COUNT = @COUNT + 1; END

SELECT CHARACTERS, OTHER_COL FROM ALWAYSENCRYPTEDTABLE2;

-SECTION 1

This section covers:

1. Connect to Azure
2. Set Azure context

----START SECTION---- --Connect to Azure and set the subscription.

Connect-AzAccount $SubscriptionId = " < SubId > "$resourceGroup = "<Resource Group>" $azureLocation = " < Region > "$akvName = "<Azure Key Vault Name>" $akvKeyName = " < AKVKeyName > "$azureCtx = Set-AzConteXt -SubscriptionId $SubscriptionId # Sets the context for the below cmdlets to the specified subscription.

--Create a new resource group and key vault.

New-AzResourceGroup -Name $resourceGroup -Location $azureLocation # Creates a new resource group - skip, if you desire group already exists. New-AzKeyVault -VaultName $akvName -ResourceGroupName $resourceGroup -Location $azureLocation # Creates a new key vault - skip if your vault already exists. Set-AzKeyVaultAccessPolicy -VaultName $akvName -ResourceGroupName $resourceGroup -PermissionsToKeys get, create, delete, list, update, import, backup, restore, wrapKey,unwrapKey, sign, verify -UserPrincipalName $azureCtx.Account$akvKey = Add-AzureKeyVaultKey -VaultName $akvName -Name $akvKeyName -Destination "Software"

--Get the key from the vault. $akvKey = Get-AzKeyVaultKey -VaultName $akvName -Name $akvKeyName ----END SECTION---

--SECTION 2

This section covers:

1. Connecting to your Managed Instance through the public endpoint with port 3342 (for my case).
2. Create Column Master Key (CMK)
3. Create Column Encryption Key (CEK)
4. Set Encryption to desired columns.

----START SECTION--- --Import the SqlServer module. Import-Module "SqlServer"

--Connect to your managed database (Azure SQL Managed database). $serverName = ” < SQLManagedInstanceName > ”$databaseName = "<DB Name>"

--Change the authentication method in the connection string, if needed. I am using AAD - Integrated for my example. $connStr = "Server = " + $serverName + "; Database = " + $databaseName + ”; Authentication = ActiveDirectoryIntegrated; ColumnEncryptionSetting = Disabled”$ database = Get-SqlDatabase -ConnectionString $connStr

--Create a SqlColumnMasterKeySettings object for your column master key. $cmkSettings = New-SqlAzureKeyVaultColumnMasterKeySettings -KeyURL $akvKey.ID

--Create column master key metadata in the database. $cmkName = "CMK1" New-SqlColumnMasterKey -Name $cmkName -InputObject $database -ColumnMasterKeySettings $cmkSettings

--Authenticate to Azure Add-SqlAzureAuthenticationContext -Interactive

-- Generate a column encryption key, encrypt it with the column master key and create column encryption key metadata in the database. $cekName = "CEK1" New-SqlColumnEncryptionKey -Name $cekName -InputObject $database -ColumnMasterKey $cmkName

--Change encryption schema $encryptionChanges = @()

-- Add changes to tables for the columns you wish to encrypt. Replace -ColumnName for the [schema].[table].[column] you want to encrypt and also replace -EncryptionType to the type you wish to use. $encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName Schema.Table.Column.Goes.Here -EncryptionType TypeYouWishToUse -EncryptionKey $cekName$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName Schema.Table.Column.Goes.Here -EncryptionType TypeYouWishToUse -EncryptionKey $cekName

Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $database -- END SECTION --

-Query the database with the existing connection string from above. Run separately from the above. If you wish to NOT see the plain text data, change the setting to Disabled. $query = "SELECT [COLUMNS] FROM

" Invoke-Sqlcmd -ConnectionString $connStr -Query $query

RCA (optional)

More Information (optional)

Detailed information/background that may be useful but isn't strictly required for troubleshooting. Often this is the "verbose" details that one usually doesn't need

Public Doc Reference (optional)

If you encounter permission issues when connecting to AKV, make sure you have configured AKV using the following steps:

- Enable client application access
- Create a key vault to store your keys

Classification

Root cause tree: Security/User Request/How-to/advisory

How good have you found this content?