Azure Storage

- Save money with lifecycle policies
- List table content with PowerShell via OAuth 2.0 authentication
- Write table content with PowerShell via OAuth 2.0 authentication

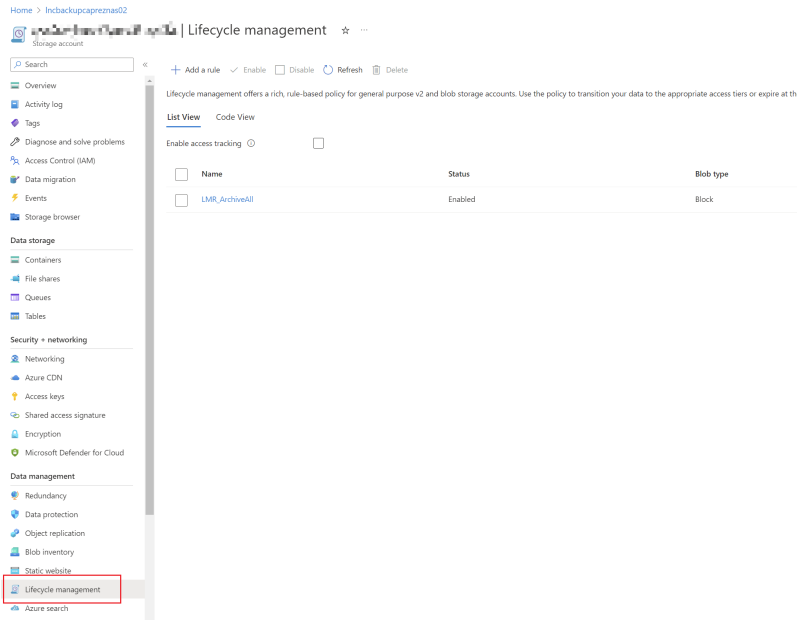
Save money with lifecycle policies

General information

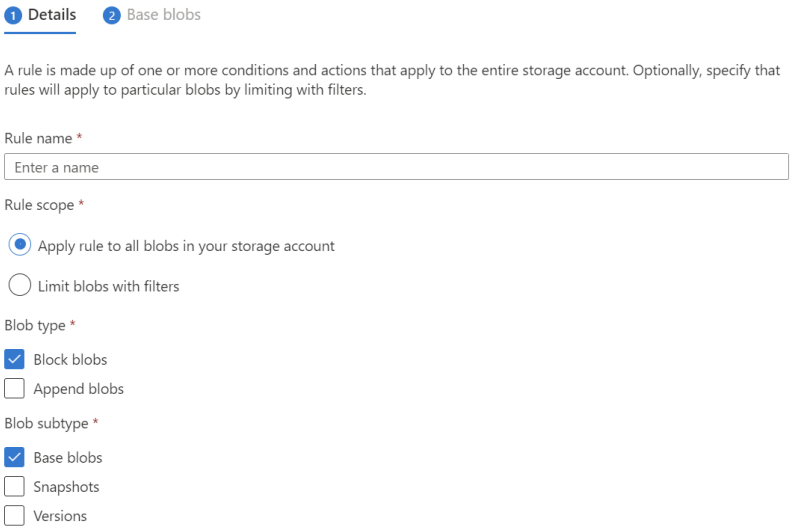
Data sets have unique lifecycles. Early in the lifecycle, people access some data often. But the need for access often drops drastically as the data ages. Some data remains idle in the cloud and is rarely accessed once stored. Some data sets expire days or months after creation, while other data sets are actively read and modified throughout their lifetimes. Azure Storage lifecycle management offers a rule-based policy that you can use to transition blob data to the appropriate access tiers or to expire data at the end of the data lifecycle.

Create lifecycle policy

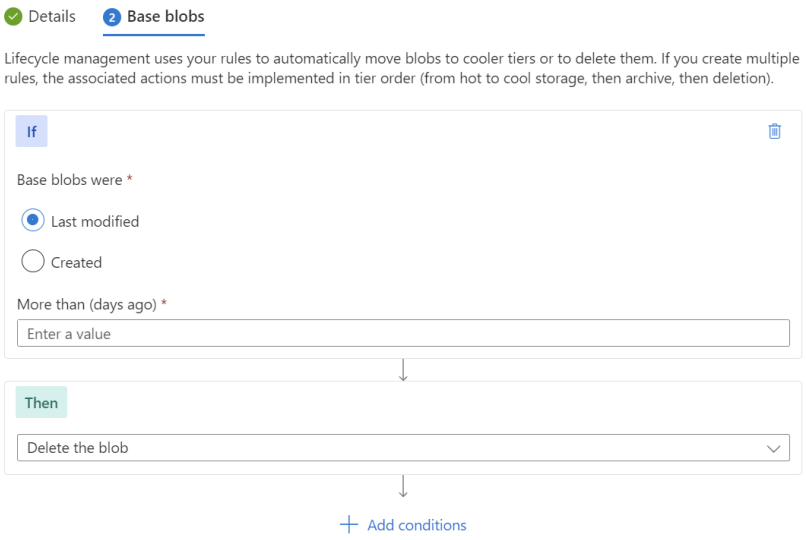
1. Open storage account



2. Create rule and give name to the policy and set the scope and blob type / subtype.



3. The last step is to create conditions and apply actions on the blob.



List table content with PowerShell via OAuth 2.0 authentication

> Requirements: Permissions to create an App Registration and PowerShell Modules "AzTable" & "Az.Storage".

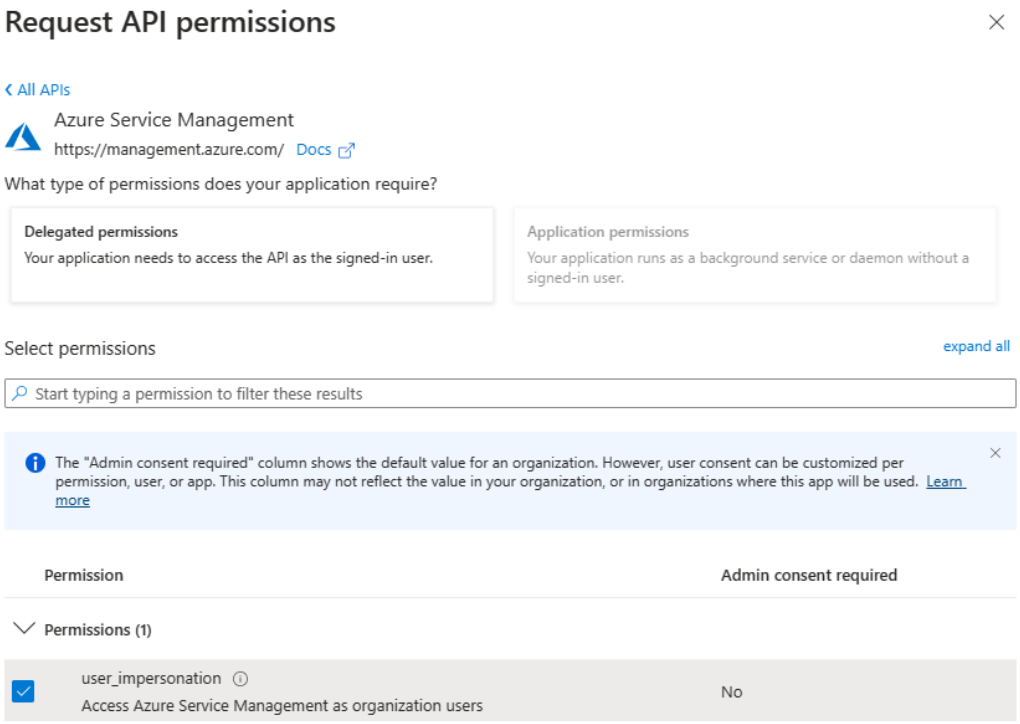
This tutorial describes how to use content from an Azure Storage Table in a PowerShell script. The authentication against the Azure Storage API is unattended and credentials are handled with an App Registration.

Create App Registration

Create a new App Registration and get the three variables as described in this guide: Get app details and gr... | LNC DOCS (lucanoahcaprez.ch)

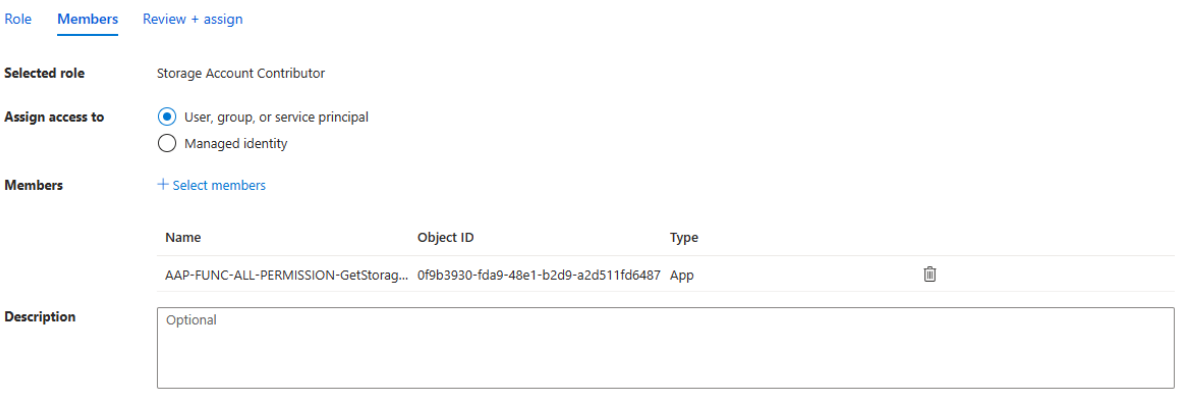
Add API permission

The only required permission for this App Registration is "user_impersonation". This permission can be found under the Azure Service Management API.



Grant permissions to Azure Storage Account

After you got all variables (Tenant ID, Client ID & Client Secret) you can add the permissions for Azure RBAC to the created App Registration. You need to go to the corresponding storage account within the azure portal. There you have to add the "Storage Account Contributor" role under "Access Control IAM":



Unfortunately, the "Storage Account Contributor" role is mandatory and restricting to a Storage Table Reader for example is not possible, otherwise the data cannot be read.

PowerShell Code

The following code can be used to read the data from the storage table specified. Here it is important that the variables are filled in correctly and that the PowerShell modules are accessible.

```
$TenantID = "<tenantid>"
$ClientId = "<cliendid>"
$ClientSecret = "<clientsecret>"

$SubscriptionId = "<subscriptionid>"
$resourceGroupName = "<resourceGroupName>"
$storageAccName = "<storageaccountname>"
$tableName = "<tablename>"

Import-Module -Name Az.Storage
Import-Module -Name AzTable

$Password = ConvertTo-SecureString -AsPlainText $ClientSecret -Force
$Credential = New-Object System.Management.Automation.PSCredential ($ClientId, $Password)
$ctx = Connect-AzAccount -ServicePrincipal -Credential $Credential -Tenant $TenantId -Subscription
$SubscriptionId
$ctx=(Get-AzStorageAccount -ResourceGroupName $resourceGroupName -Name $storageAccName).Context

$cloudTable = (Get-AzStorageTable –Name $tableName –Context $ctx.context).CloudTable
$TableContent = Get-AzTableRow -table $cloudTable
```

Write table content with PowerShell via OAuth 2.0 authentication

This tutorial describes how to add new content to an Azure Storage Table with a PowerShell script. The authentication against the Azure Storage API is unattended and credentials are handled with an App Registration.

Preparations

To gain access to the storage table you need to initialize an App Registration and add permission & a client secret. Then you have to add the Azure RBAC role to the storage account. Everything is documented here in detail: List table content wit... | LNC DOCS (lucanoahcaprez.ch)

PowerShell Code

The following code can be used to write data to the storage table specified. Here it is important that the variables are filled in correctly and that the PowerShell modules are accessible.

```
$TenantID = "<tenantid>"
$ClientId = "<cliendid>"
$ClientSecret = "<clientsecret>"

$SubscriptionId = "<subscriptionid>"
$resourceGroupName = "<resourceGroupName>"
$storageAccName = "<storageaccountname>"
$tableName = "<tablename>"



Import-Module -Name Az.Storage
Import-Module -Name AzTable
```

```
$Password = ConvertTo-SecureString -AsPlainText $ClientSecret -Force
$Credential = New-Object System.Management.Automation.PSCredential ($ClientId, $Password)
$ctx = Connect-AzAccount -ServicePrincipal -Credential $Credential -Tenant $TenantId -Subscription
$SubscriptionId
$ctx = (Get-AzStorageAccount -ResourceGroupName $resourceGroupName -Name $storageAccName).Context

$cloudTable = (Get-AzStorageTable –Name $tableName –Context $ctx.context).CloudTable

Add-AzTableRow -partitionKey "dn" -Rowkey "$DomainName" -table $cloudTable -property @{ "<property1>" =
"value1"; "<property2>" = "value2"; "<property3>" = "value3"; } | Out-Null
```