Azure Data Lake Initiative: Data Lake Framework Setup

Generic Template

December 10, 2019

Table of Contents

[1 Purpose 5](#_Toc53742306)

[2 Databricks Framework Workspace Set-up 5](#_Toc53742307)

[2.1 Provision Databricks for Azure Active Directory (SCIM) 5](#_Toc53742308)

[3 Azure Key Vault - Secrets 5](#_Toc53742309)

[4 Azure Key Vault – Access Policy 7](#_Toc53742310)

[4.1 Secrets 7](#_Toc53742311)

[4.1.1 Databricks Secret Scopes 7](#_Toc53742312)

[4.2 Step5 – Set-up Azure Databricks Framework Notebooks 8](#_Toc53742313)

[4.2.1 Import Framework Notebooks 8](#_Toc53742314)

[4.2.1.1 Download and the latest framework notebooks 8](#_Toc53742315)

[4.2.1.2 Generate databricks user access token 8](#_Toc53742316)

[4.2.1.3 Download, edit, and execute the ImportDatabricksFolder PowerShell script 9](#_Toc53742317)

[4.2.1.4 Verify the import of the framework notebooks 9](#_Toc53742318)

[5 Storage Account – Storage Blob Contributor Role 10](#_Toc53742319)

[6 Create Databricks Framework Cluster 10](#_Toc53742320)

[6.1 Run Cluster Init Script Setup and add init scripts to start-up 10](#_Toc53742321)

[6.2 Cluster Libraries 11](#_Toc53742322)

[7 Create Databricks Container Cluster 12](#_Toc53742323)

[7.1 Run Cluster Init Script Setup and add init scripts to start-up 12](#_Toc53742324)

[7.2 Cluster Libraries 13](#_Toc53742325)

[7.2.1 com.microsoft.azure:azure-sqldb-spark:1.0.2 (Maven) 13](#_Toc53742326)

[8 Grant Permissions to Databricks Cluster 13](#_Toc53742327)

[9 Create Azure Data Factory (V2) Copy Pipeline and Linked Services 14](#_Toc53742328)

[9.1 AzureDatabricksData001 15](#_Toc53742329)

[9.2 AzureDatabricks Framework 16](#_Toc53742330)

[9.3 AzureDataLakeStorageGen2 18](#_Toc53742331)

[9.4 AzureKeyVault 19](#_Toc53742332)

[9.5 AzureSQLDatabase 20](#_Toc53742333)

[10 Azure Automation 21](#_Toc53742334)

[11 Sanctioned Zone 22](#_Toc53742335)

[11.1 ADLS Gen2 Temp Folder 22](#_Toc53742336)

[11.2 SQL Data Warehouse 22](#_Toc53742337)

[11.2.1 Create Master Key 22](#_Toc53742338)

[11.2.2 Create Managed Identity 23](#_Toc53742339)

[11.2.3 Register Azure SQL Server in Active Directory (Service Endpoint) 23](#_Toc53742340)

[11.2.4 Add Service Endpoint to ADLS Gen2 Storage Account 23](#_Toc53742341)

[11.3 Azure Analysis Services 24](#_Toc53742342)

[11.3.1 Add Service Principal Credentials to Azure Automation 24](#_Toc53742343)

[11.3.2 Add Service Principal as Admin on Analysis Services 25](#_Toc53742344)

[Appendix A – Databricks Framework workspace Secrets 26](#_Toc53742345)

# Purpose

The purpose of this document is to define the steps necessary to set up the Data Lake Design Framework.

The audience for this document includes both THE CLIENT infrastructure and Cloud development teams as well as the Data Lake project technical staff.

# Containers

Container is an analogous term used for Client, Firm Dataset, and Benchmark. In this document, we will use Client0000001 as the container example.

# Container Set-up

## Step1 - Active Directory Groups

AD Groups will be created for each Container for each Azure environment as follows and will be used to assign ACL permissions to services within Azure and the Azure Data Lake (Gen1 and Gen2) (Client container example below):

Azure-DADevAnalyticsClient000000001Admin

Includes IT / Pipeline developers – access to Landing, Raw, and Query Zone

Azure-DADevAnalyticsClient000000001Publish

Business Publishers for the client – can update pipeline notebooks for Summary Zone

Note: for membership to this group an individual would need to be certified (method TBD)

Azure-DADevAnalyticsClient000000001Contribute

Business Developers for the client – can work in Sandbox

Azure-DADevAnalyticsClient000000001Read

Read access to the client data

Create the Container AD Groups by running the Powershell script, the key vault should have been setup at this point via the da-dev-wcus-analytics-kv.deploy ARM template with tenant permissions.

**Link:**

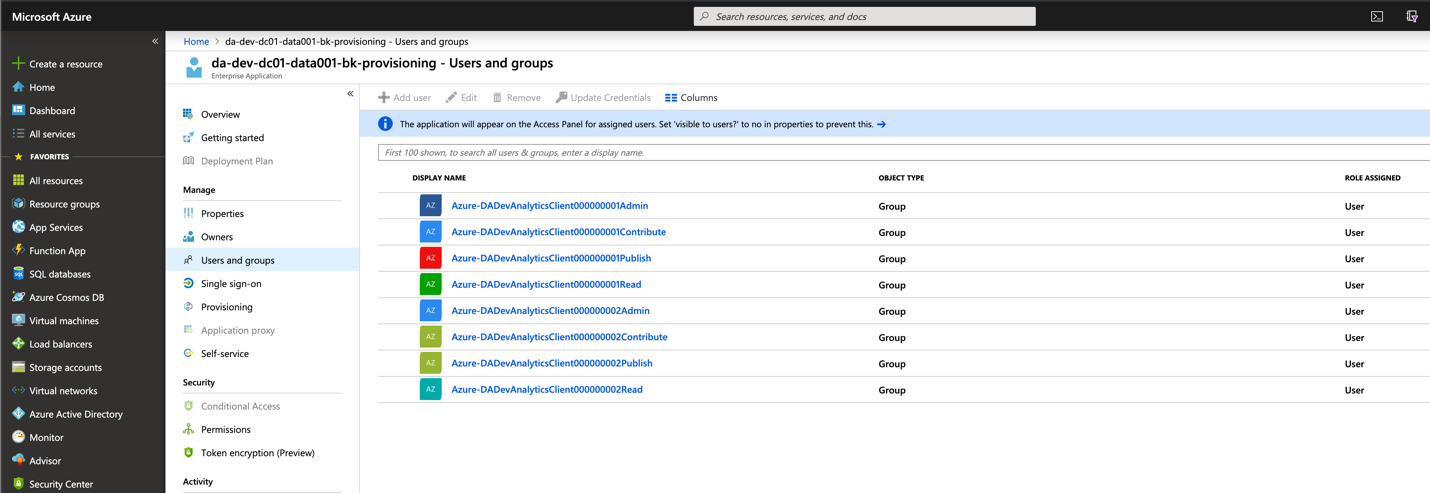
\Solution\Powershell\CreateServicePrincipalsAndADGroupsContainer.ps1

### Step 1b – Provision Databricks for Azure Active Directory (SCIM)

Follow the steps outlined in the link below to configure Databricks workspace to recognize AD Groups and users.

<https://docs.azuredatabricks.net/administration-guide/admin-settings/scim/aad.html>

Note: As container Azure AD groups are added, the enterprise application provisioning needs to be updated to include the new container groups (Otherwise the databricks workspace will not recognize them).



## Step2 - Azure Active Directory Application Registrations (Service Principals)

Register Azure Active Directory Applications that correspond to the AD security groups for each Container and each Environment (Client container example below):

DA-Dev-Client000000001-Admin

DA-Dev-Client000000001-Publish

DA-Dev-Client000000001-Contribute

DA-Dev-Client000000001-Read

Note: Save the Application Keys for later use.

Create the Application Registrations (Service Principals) by running the following Powershell script:

**Link:**

\Solution\Powershell\CreateAzureServicePrincipalsAndCredsDatabricks.ps1

Note: Save the output text file for later use in setting up the Key Vault Secrets.

## Step3 – Azure Data Lake

### Azure Data Lake Gen2 (Storage Account)

A single storage account will be used for this initiative and will contain the following structure and assigned permissions (ACLs).

### Storage Account – Storage Blob Contributor Role

It is necessary to add the Admin, Publish, and Contribute Registered Applications to the “Storage Blob Contributor Role” to the Azure Storage Service (IAM role). Members of this role should include the admin registered application for the framework.

DA-Dev-Client000000001-Admin

DA-Dev-Client000000001-Publish

DA-Dev-Client000000001-Contribute

A screen shot of a computer

Description automatically generated

### Permissions Matrix (Registered Applications and AD Groups):



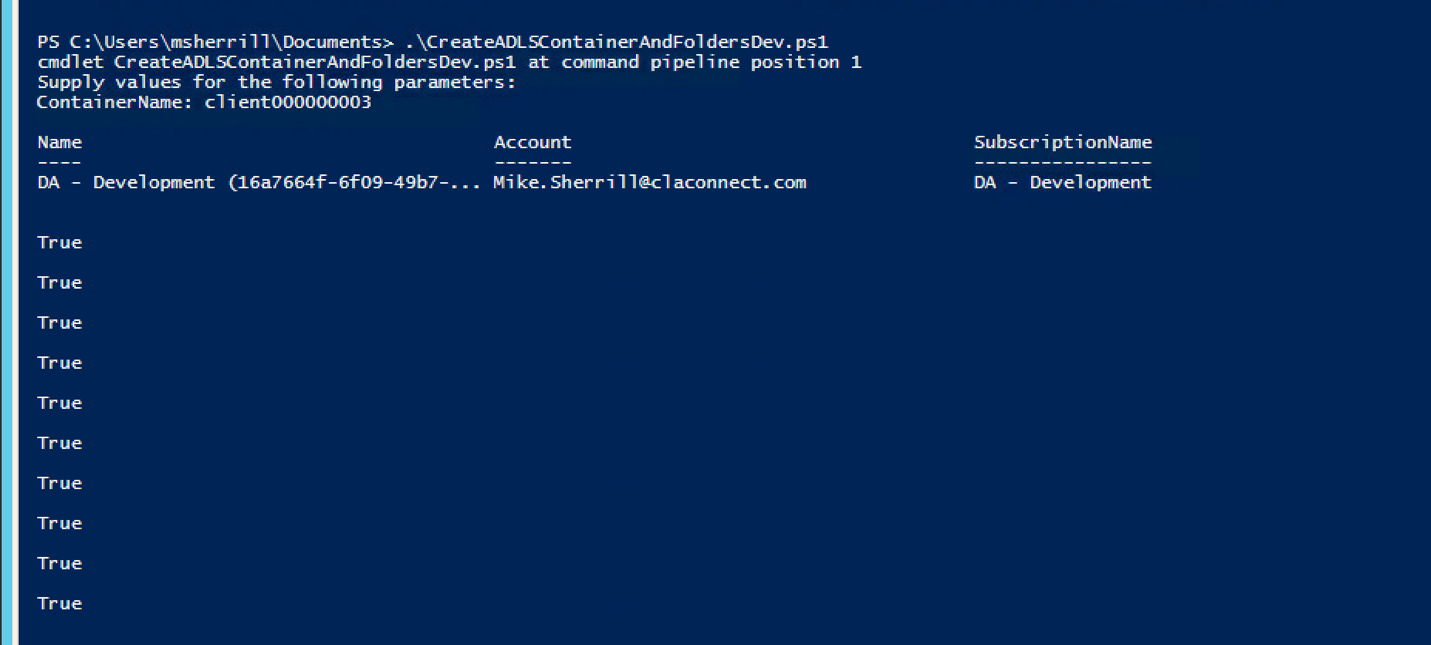
### Create Data Lake Container and Folders

Run the powershell script associated with the Azure environment to create a Blob Container, the standard folder structure. The adl should have been setup via the ARM template at: dadevwus2analyticsdl.deploy

**Link:**

.\ CreateADLSContainerAndFoldersDev.ps1

Entering the ContainerName for the new container



The CreateADLSContainerAndFoldersDev.ps1 calls an additional powershell script:

1. CreateADLSGen2ContainerAndFolders.ps1, passing the StorageAccountName, StorageAccountKey, and ContainerName as parameters.
2. Note that the key vault secrets should have already been configured for this step(\Powershell\ConfigureAzureKeyVault.ps1)

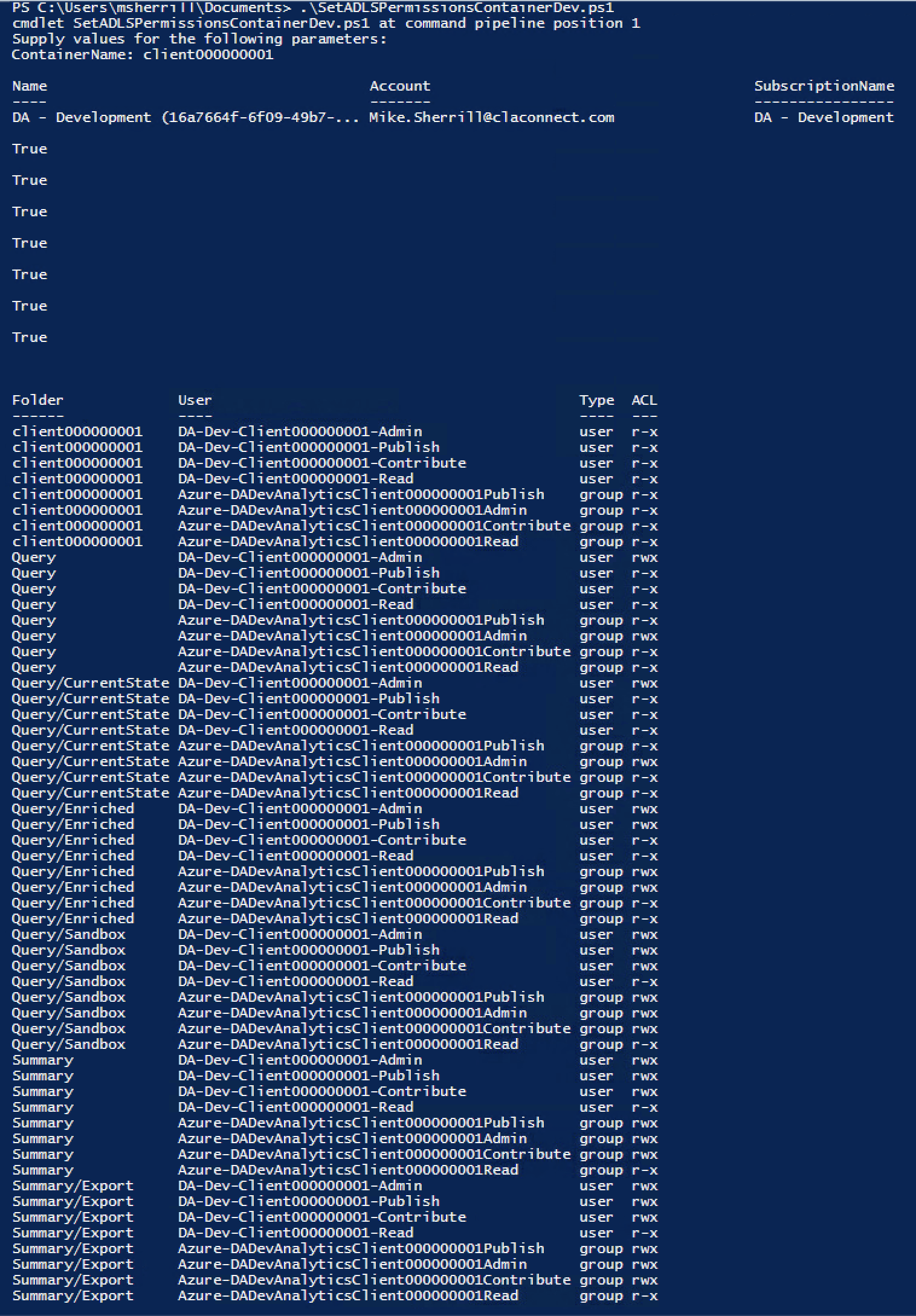
Note: Be sure to add the appropriate values to CreateADLSContainerAndFoldersDev.ps1.

### Grant Service Principal (Applications) and Azure AD Groups permissions to Azure Data Lake Gen2 (Storage Account) Container and folders.

Run the powershell script associated with the Azure environment to grant permissions to the new ADLS Container. The Service Principals and AD Groups must already exist (see 3.1 Step1 and 3.2 Step2 above)

.\SetADLSPermissionsContainerDev.ps1

Entering the ContainerName for the new container



The SetADLSPermissionsContainerDev.ps1 calls two additional powershell scripts:

1. CreateADLSGen2ContainerAndFolders.ps1, passing the StorageAccountName, StorageAccountKey, and ContainerName as parameters
2. GetADLSPermissions.ps1, passing the StorageAccountName, StorageAccountKey, and ContainerName as parameters

Note: Be sure to add the appropriate values to CreateADLSContainerAndFoldersDev.ps1.

# Databricks Framework Workspace Set-up

## Provision Databricks for Azure Active Directory (SCIM)

Follow the steps outlined in the link below to configure Databricks workspace to recognize AD Groups and users.

<https://docs.azuredatabricks.net/administration-guide/admin-settings/scim/aad.html>

Note: As container Azure AD groups are added, the enterprise application provisioning needs to be updated to include the new container groups (Otherwise the databricks workspace will not recognize them).

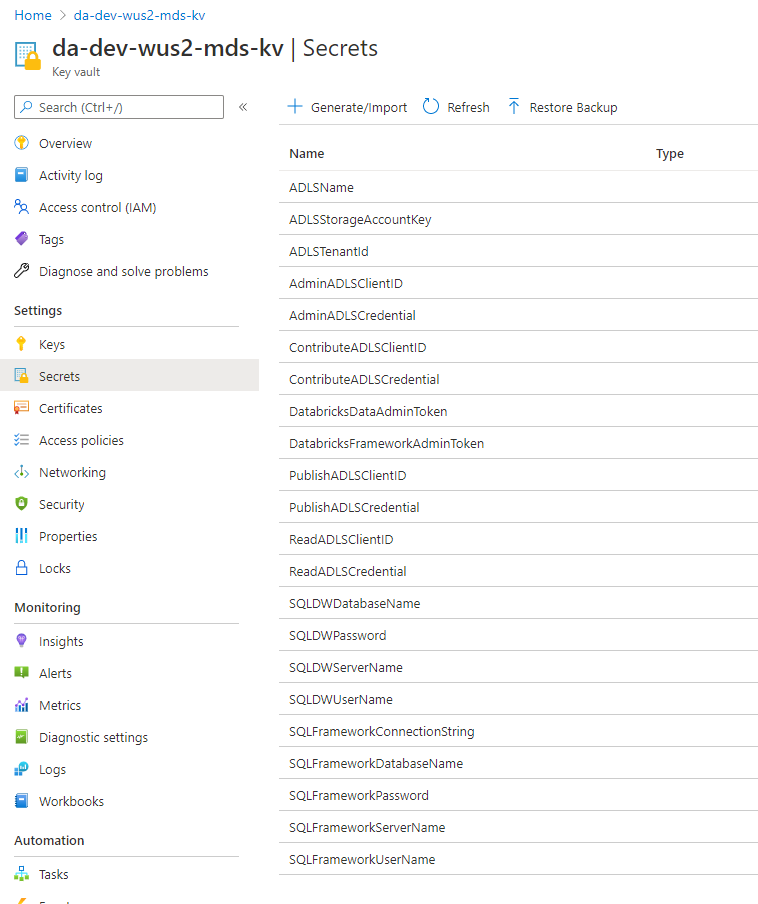
# Azure Key Vault - Secrets

The following key vault secrets are needed for the solution. Some of the Key Vault Secrets are create as part of the service set-up, and the others will need to be created by running the Powershell script “AzureKeyVaultEnv.ps1” where “Env” is replaced by the Environment, Dev, Tst, Prd. The script has to be modified as needed per environment.

**Link:**

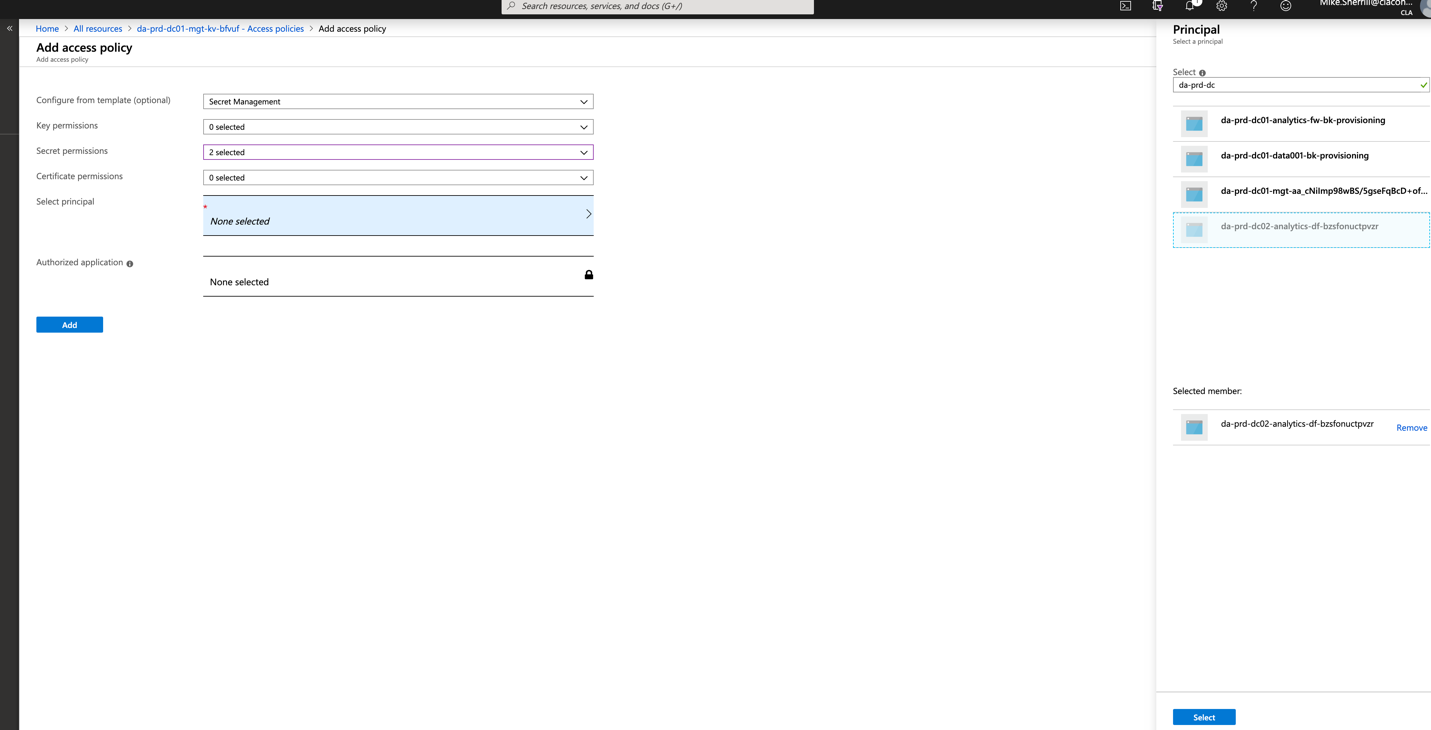
\Powershell\ConfigureAzureKeyVault.ps1

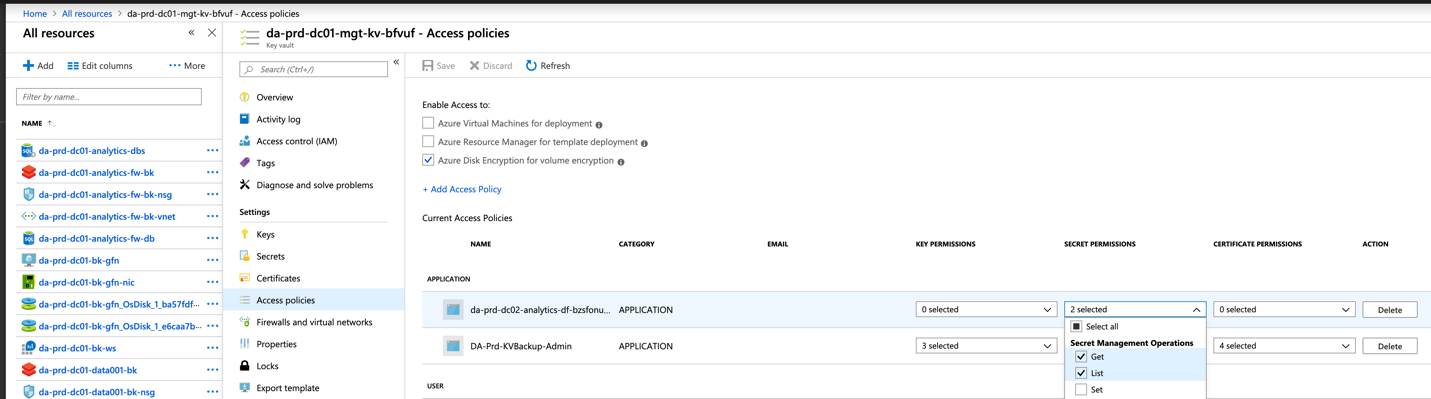
The image below shows the Key Vault secrets needed for each Environment (Dev, UAT, Prod)



# Azure Key Vault – Access Policy

An Access Policy is required for the Data Factory Service Principal, so that the Data Factory can retrieve secrets from the Key Vault. It should be provisioned with “Secret Management” “Get”, and “List” permissions.





## Secrets

### Databricks Secret Scopes

Create Secrets, Secret Scopes, and Secret Scope using PowerShell. Azure Key Vault Secrets must be created first.

Run the following Powershell

**Link:**

..source\repos\ContainerFramework\notebooks\Secrets\DevSetDatabricksScopes.ps1

**Link:**

\Powershell\CreateDatabricksSecretsCLI\_option2

Secret Scopes will belong to a profile that is representative of the Azure Subscription and Region where the Databricks Workspace resides.

The Framework secrets are listed below.

SQLFrameworkDatabaseName

SQLFrameworkUserName

SQLFrameworkPassword

SQLFrameworkServerName

SQLAnalyticsDatabaseName

SQLDWDatabaseName

SQLDWServerName

SQLDWUserName

SQLDWPassword

ADLSTenantId

ADLSClientId

ADLSCredential

ADLSGen2StorageAccountName

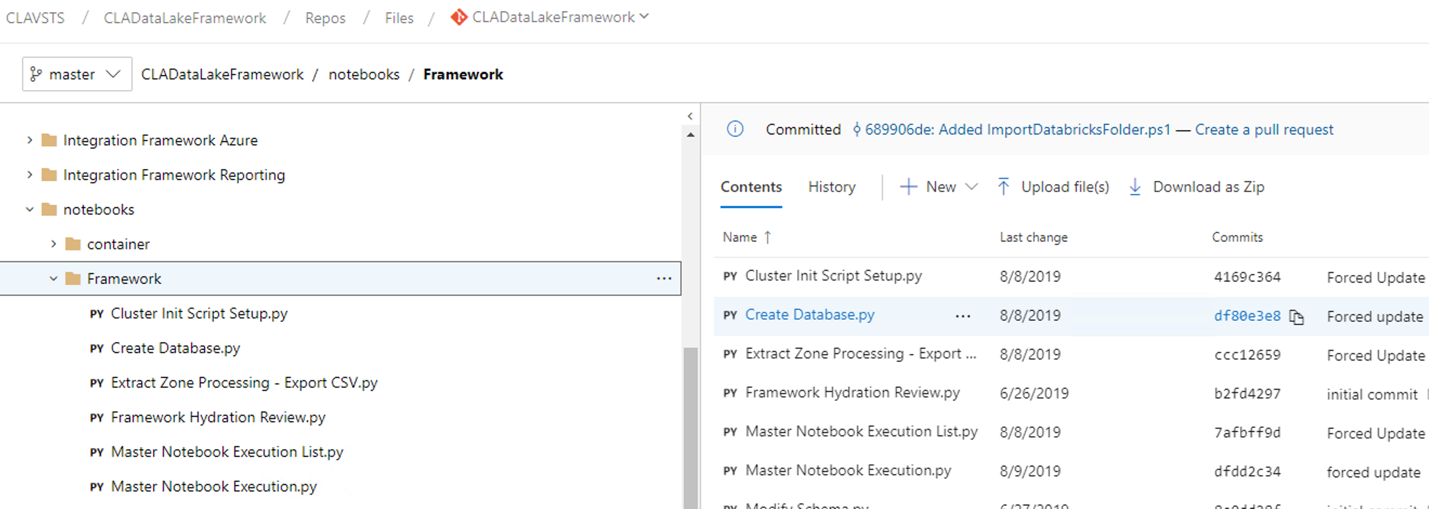
ADLSGen2StorageAccountKey

## Step5 – Set-up Azure Databricks Framework Notebooks

### Import Framework Notebooks

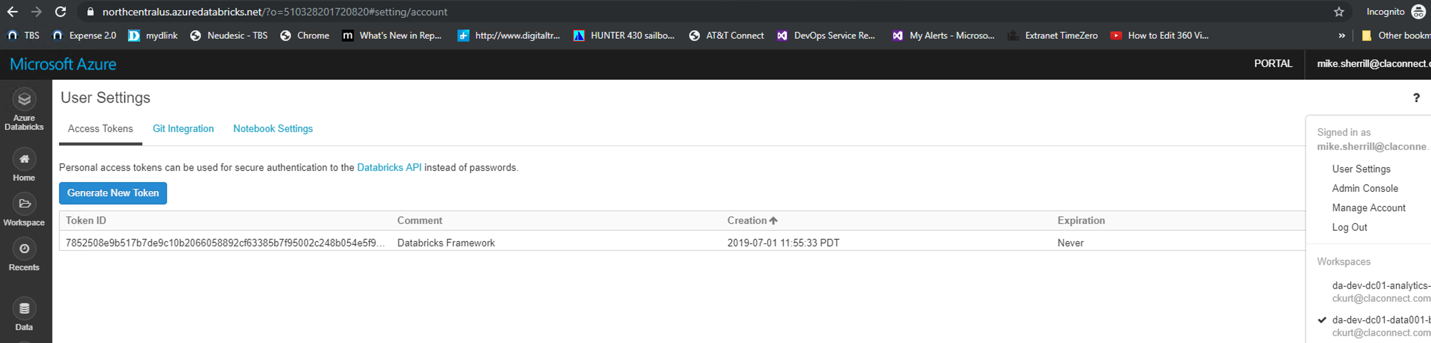
#### Download and the latest framework notebooks

Download and the latest framework notebooks from DevOps. Download as a Zip and extract locally.



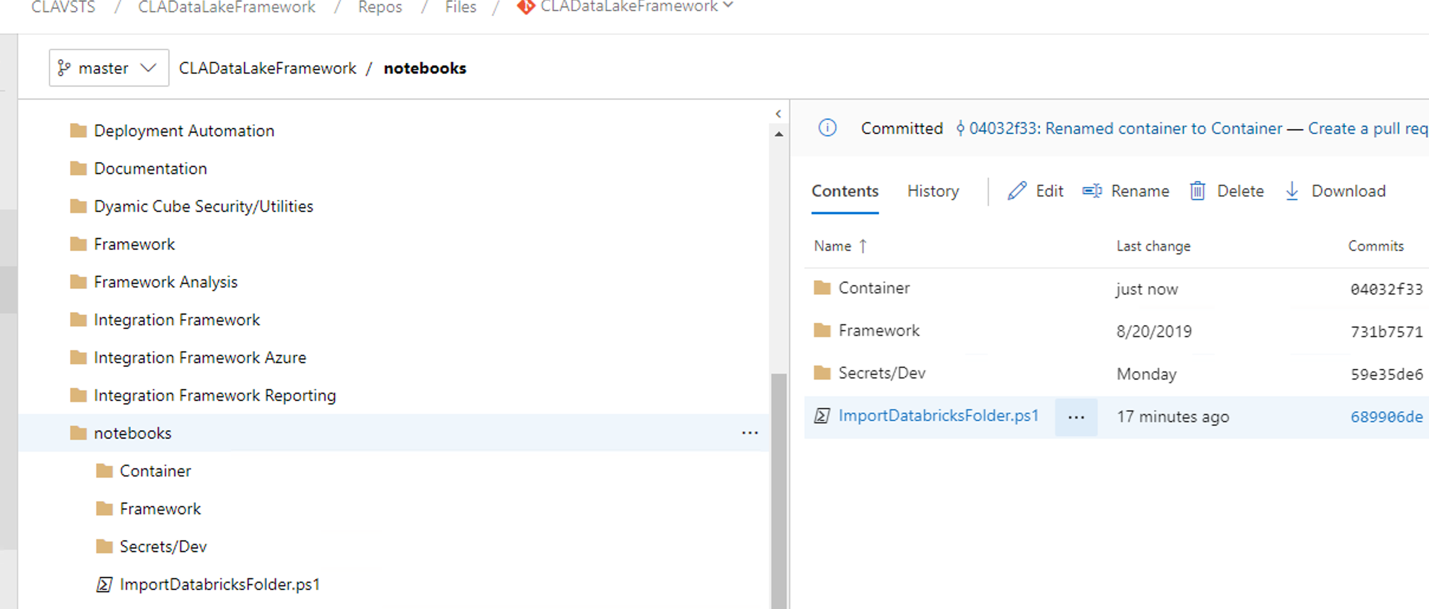
#### Generate databricks user access token

From within the Framework databricks workspace, generate a user access token. Copy the token for the following step.



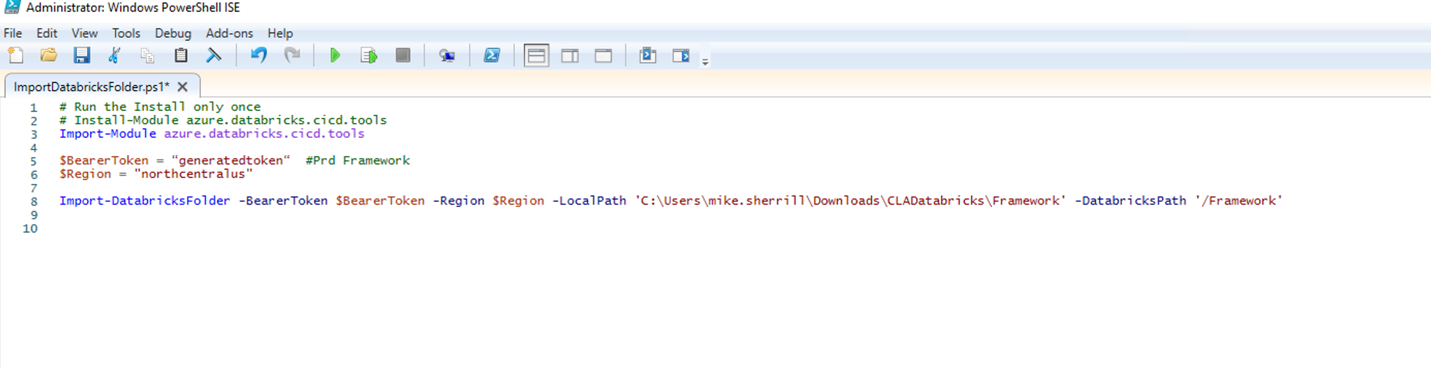
#### Download, edit, and execute the ImportDatabricksFolder PowerShell script

Download the ImportDatabricksFolder.ps1 PowerShell script from DevOps



Edit the path to the location of the downloaded notebooks and set the $BearerToken (access token from 2.3.1.2 above) and execute the PowerShell script.

(Note: the script does not produce any output during execution)

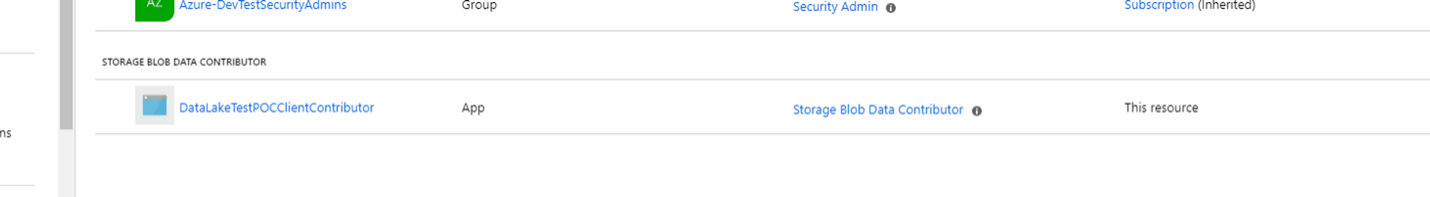


#### Verify the import of the framework notebooks

Verify the existence of the framework notebooks in the Framework Databricks Workspace.

# Storage Account – Storage Blob Contributor Role

It is necessary to add the Administrator and Publisher Registered Applications to the “Storage Blob Contributor Role” to the Azure Storage Service (IAM role). Members of this role should include the admin registered application for the framework. This is not needed for each container application.



# Create Databricks Framework Cluster

## Run Cluster Init Script Setup and add init scripts to start-up

Run the “Cluster Init Script Setup” notebook to create the initialization objects and add them to the cluster.

A screenshot of a cell phone

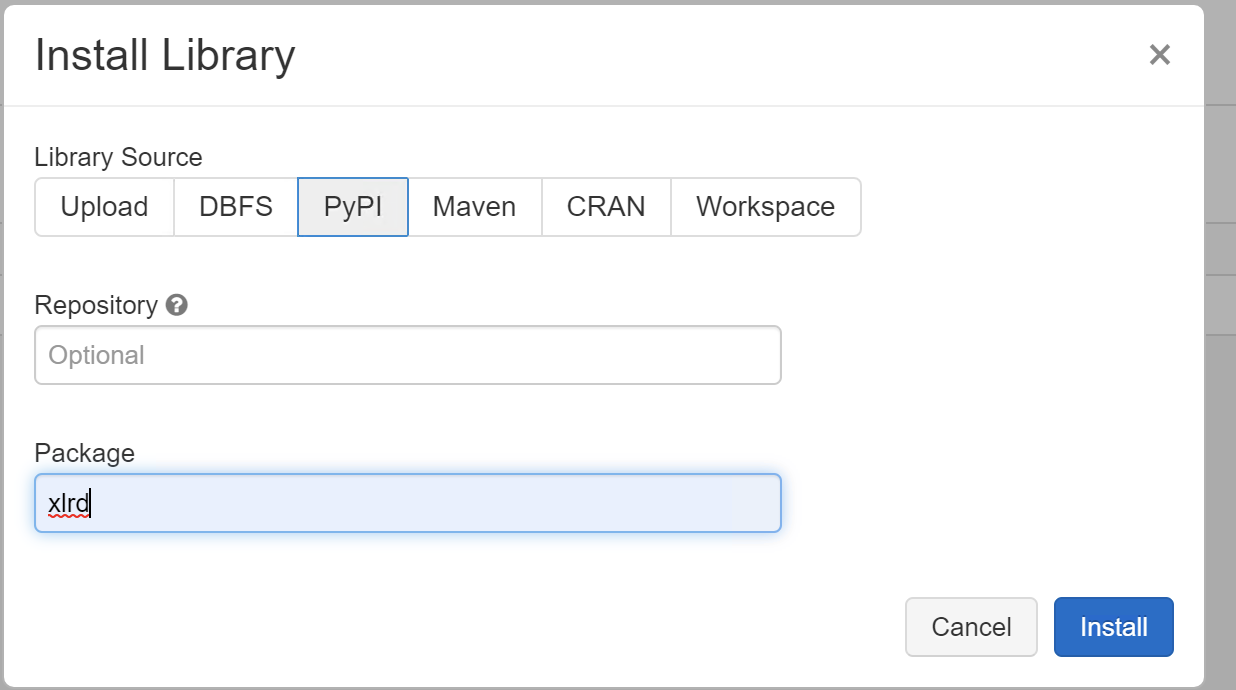
Description automatically generated

A screenshot of a cell phone

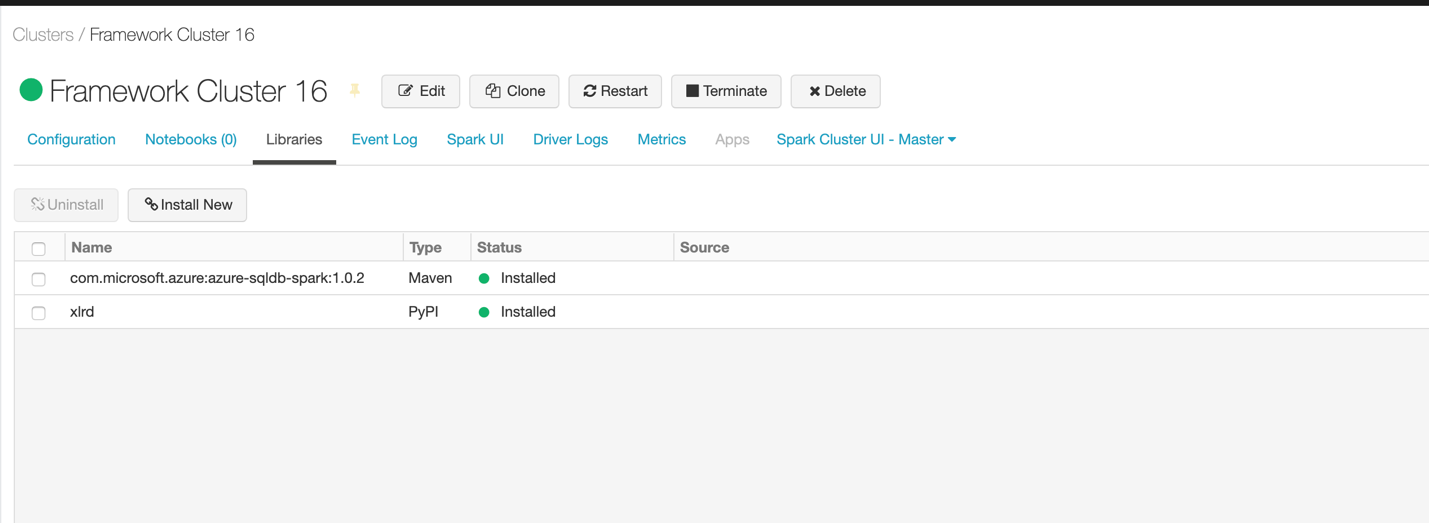
Description automatically generated

## Cluster Libraries

When creating the Databricks Framework Cluster, you must install the xlrd library from PyPI. It is needed for the xlsx converstion to json process.



In Addition the com.microsoft.azure:azure-sqldb-spark:1.0.2 Maven library is required for writing to the Azure SQL Server Database.



# Create Databricks Container Cluster

## Run Cluster Init Script Setup and add init scripts to start-up

Run the “Cluster Init Script Setup” notebook to create the initialization objects and add them to the cluster.

A screenshot of a cell phone

Description automatically generated

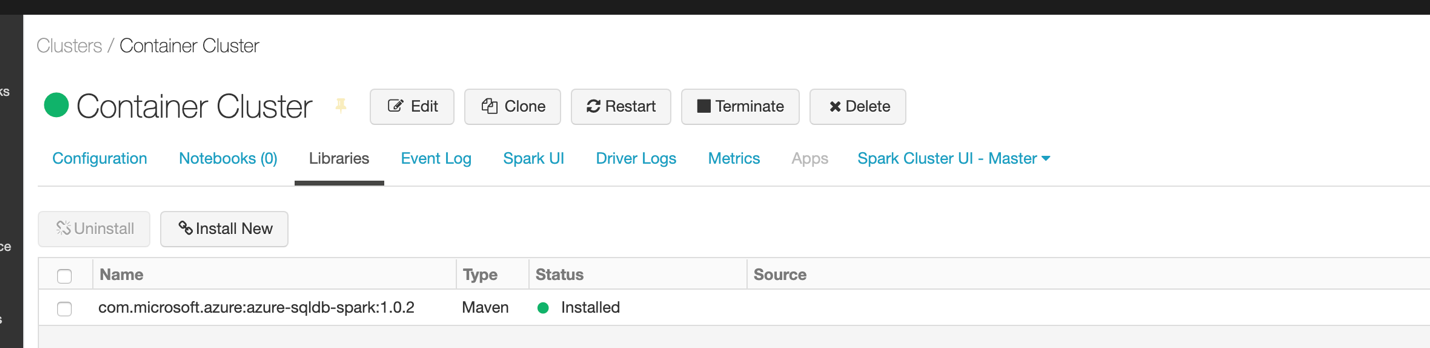
A screenshot of a cell phone

Description automatically generated

## Cluster Libraries

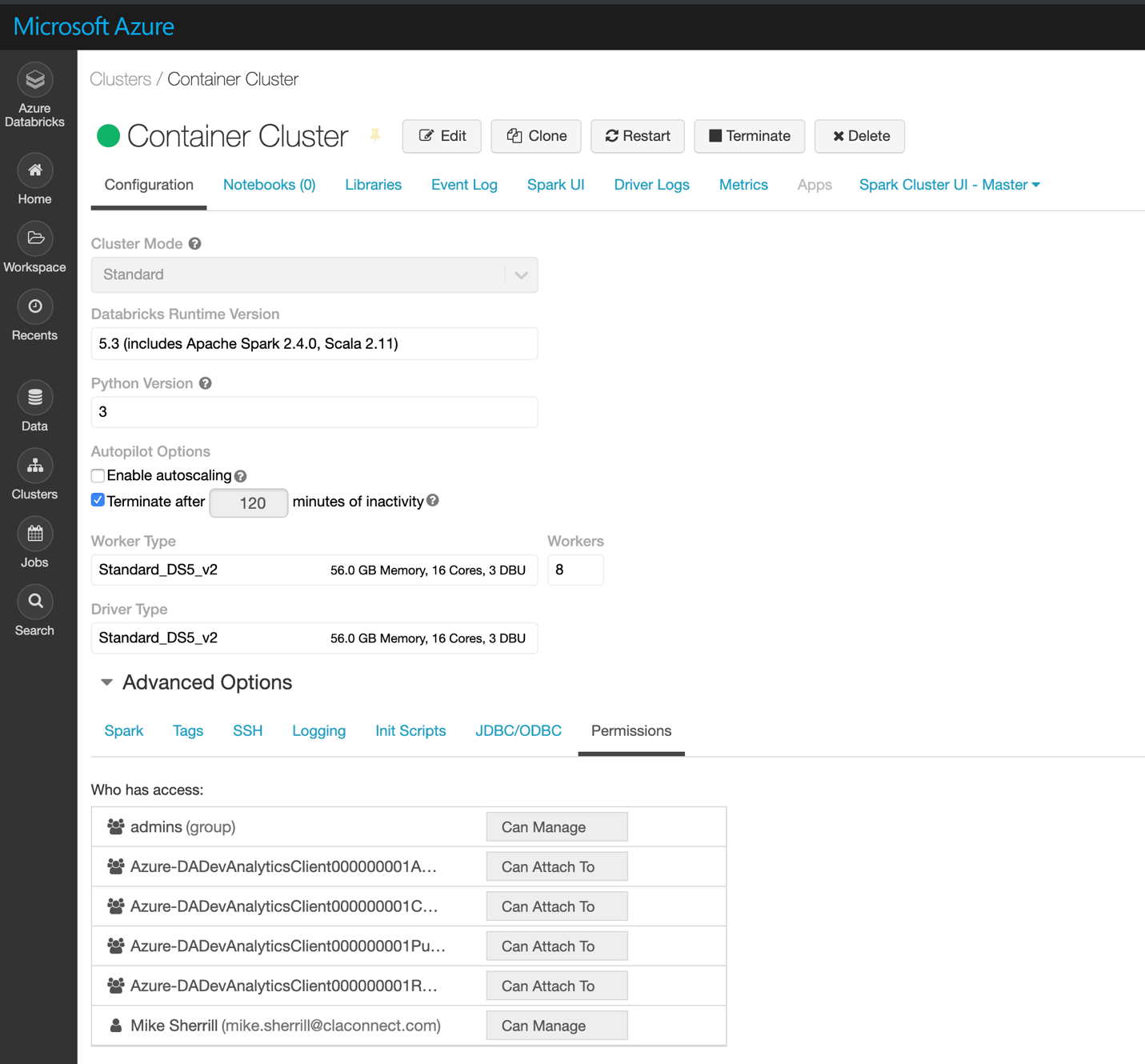
### com.microsoft.azure:azure-sqldb-spark:1.0.2 (Maven)

The com.microsoft.azure:azure-sqldb-spark:1.0.2 maven library must be installed on each of the Container databricks clusters to support writing to the Sanctioned Layer SQL Server.



# Grant Permissions to Databricks Cluster

Permissions are set at the cluster level (Advaced Options). Grant “Can Attach to” permissions to each of the Azure AD Groups (Admin, Contribute, Publish, and Read).



# Create Azure Data Factory (V2) Copy Pipeline and Linked Services

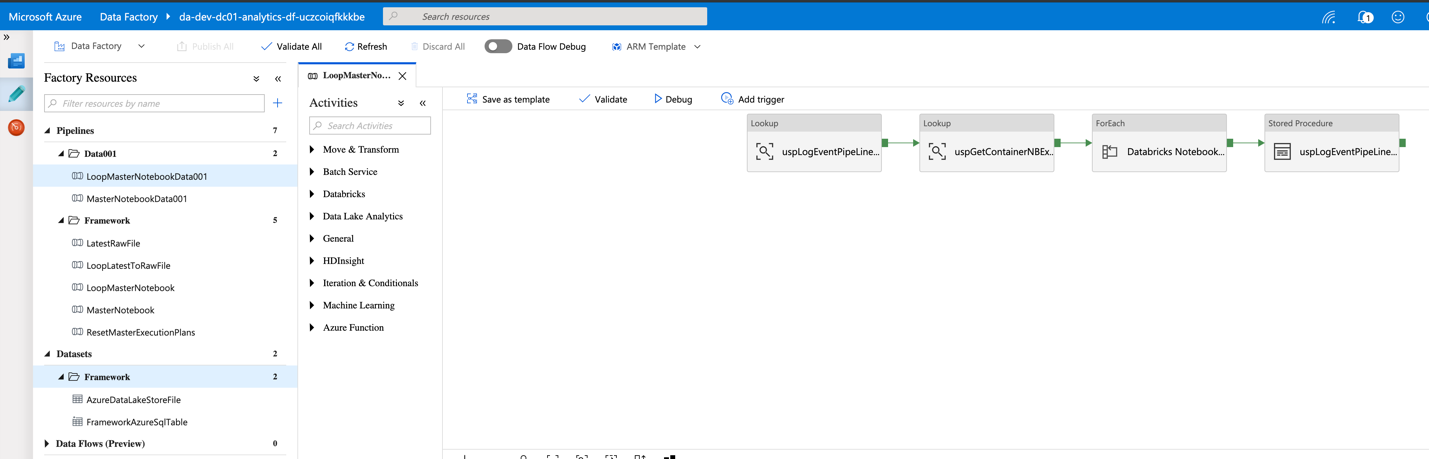
The pipelines and linked services below are created from the .ARM template. After it is deployed, each linked service should be tested for successful conection.

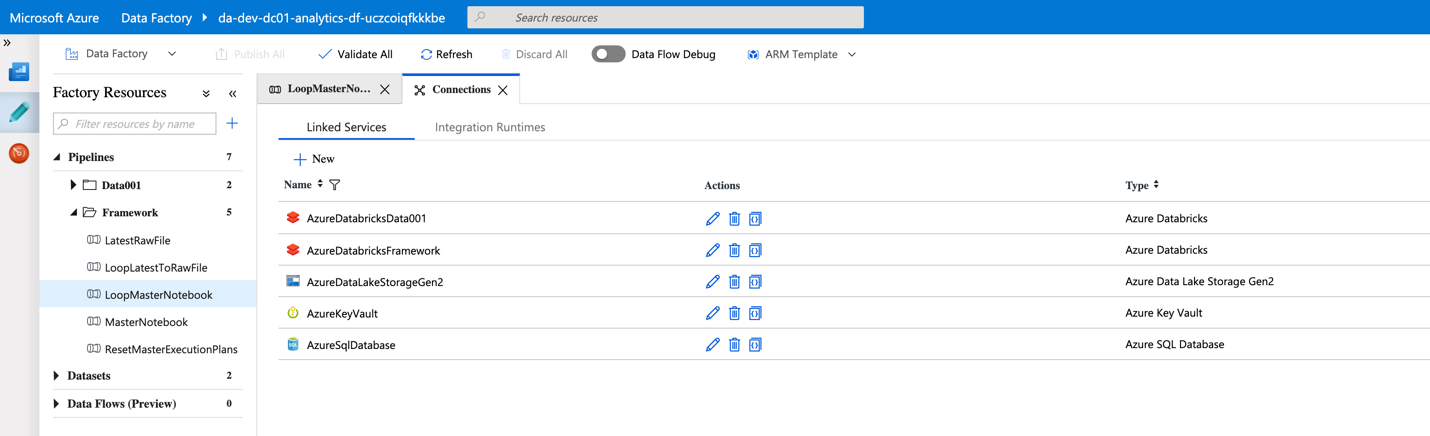
Deploy the ARM template from the following location using the Powershell script.

source\repos\ContainerFramework\Data Factory\ARMTemplate\da-dev-analytics-df.deploy.ps1

A set of initial Pipelines are provided that populate the Raw Zone from a file (.json) or SQL sources.

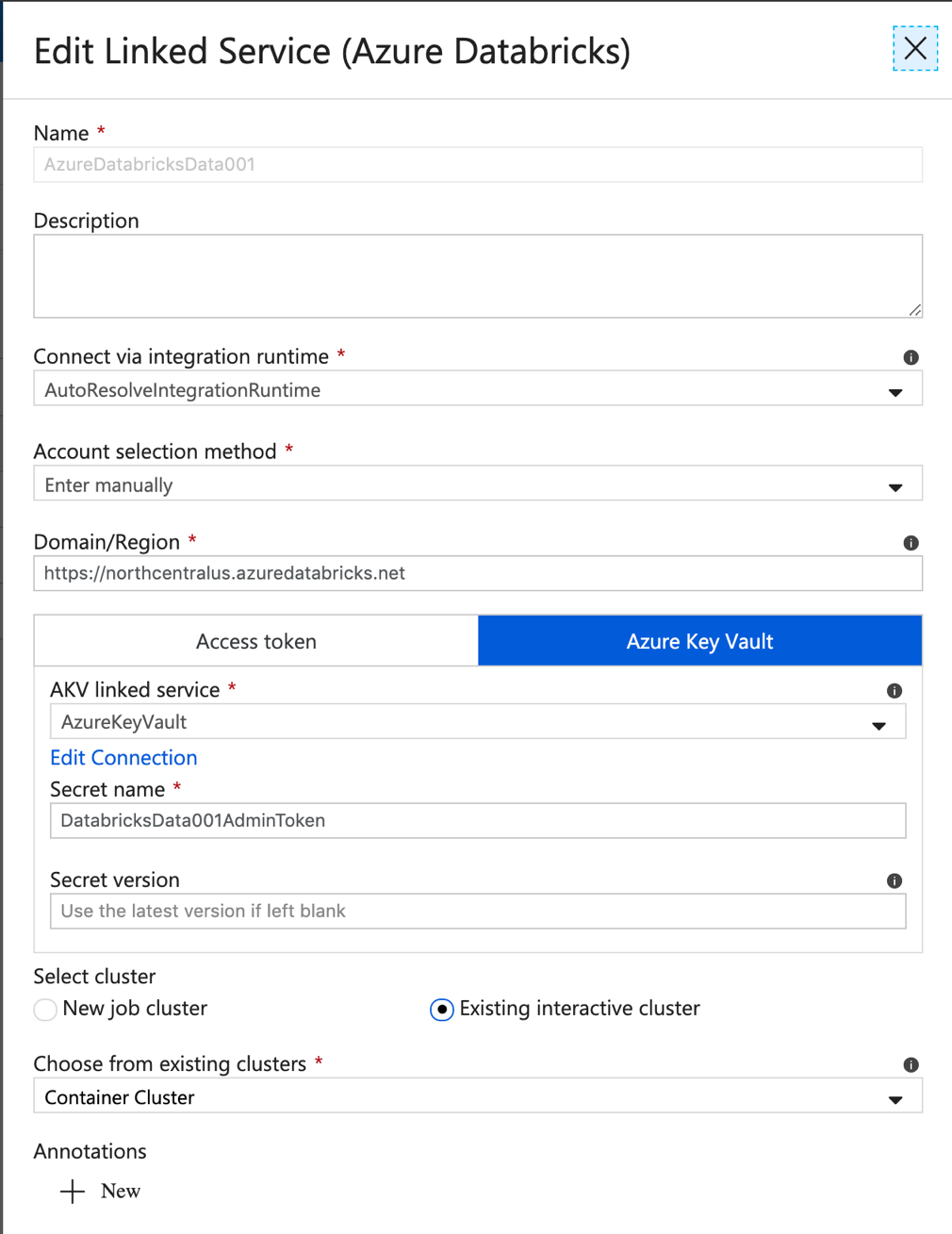
Pipelines to run databricks notebooks using a LoopMaster and Master are deployed for the framework and each databricks workspace (instance). They can be used to execute the Convert, Ingest, Enrich, and Publish notebook processes.





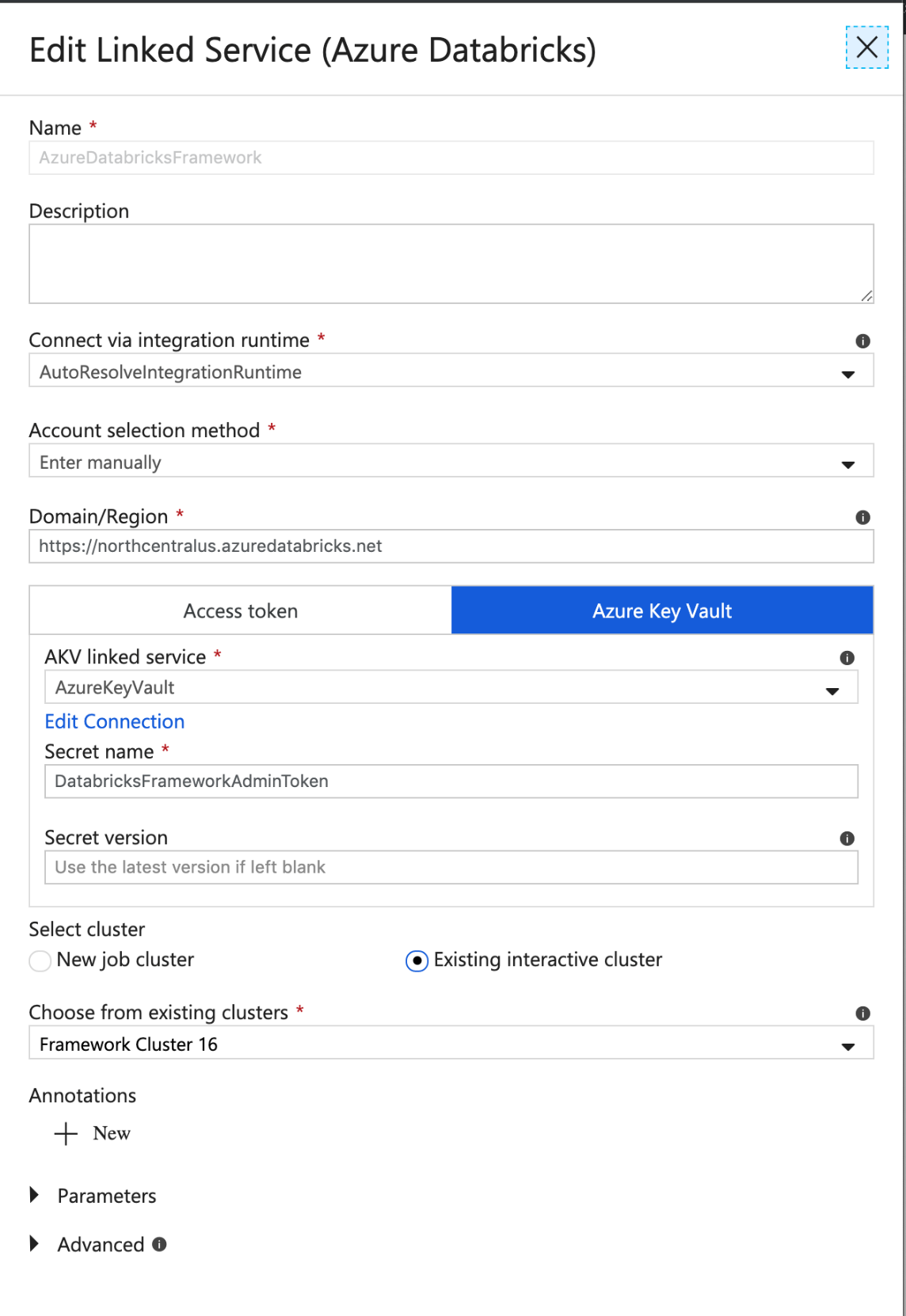
## AzureDatabricksData001

A Linked Service for each databricks workspace, and will be used by all containers within that workspace. Before testing this Linked Service, select an existing cluster in the databricks workspace.

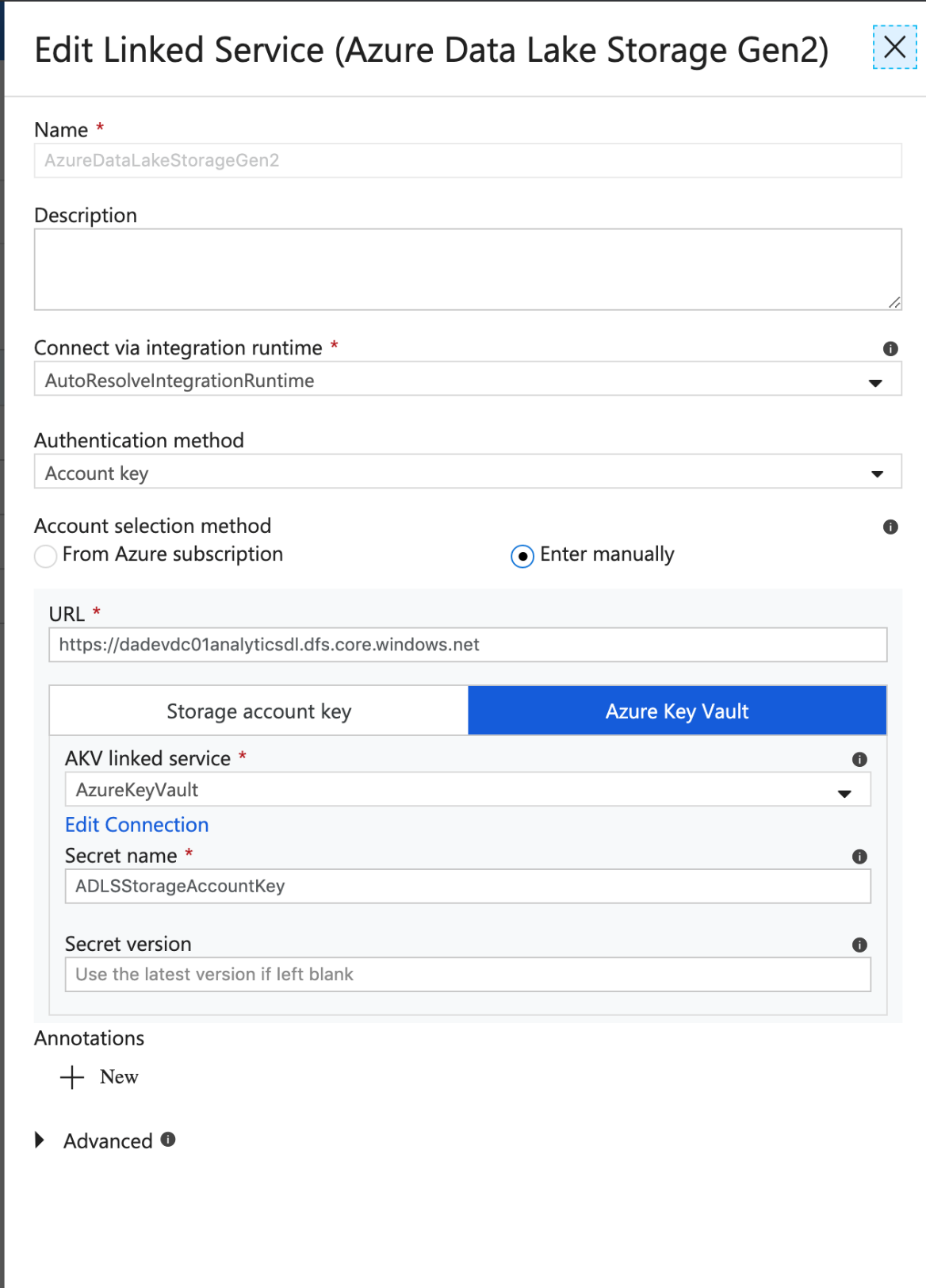


## AzureDatabricks Framework

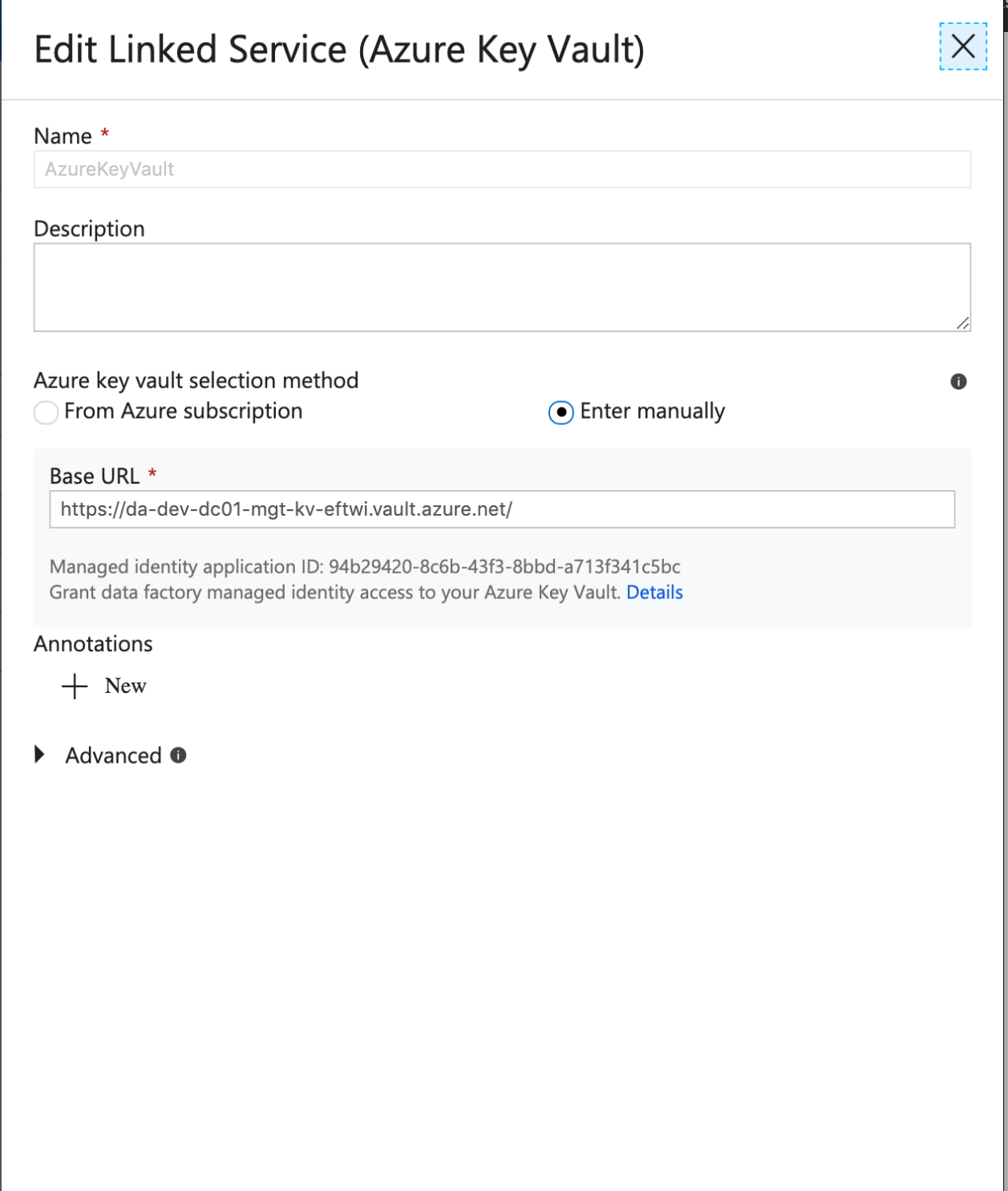
Before testing this Linked Service, select an existing cluster in the databricks workspace.



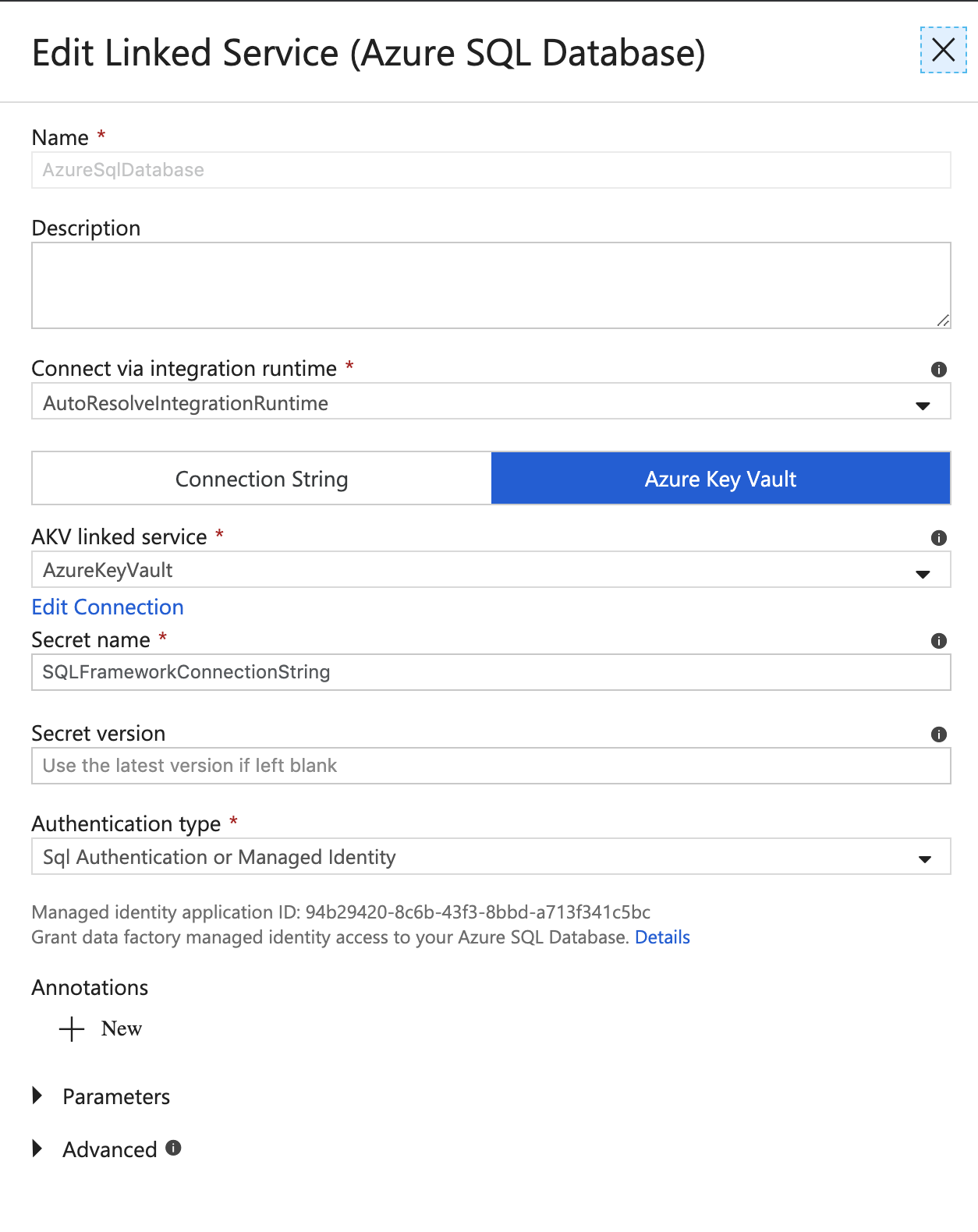
## AzureDataLakeStorageGen2



## AzureKeyVault



## AzureSQLDatabase

This linked service points to the framework sql server.

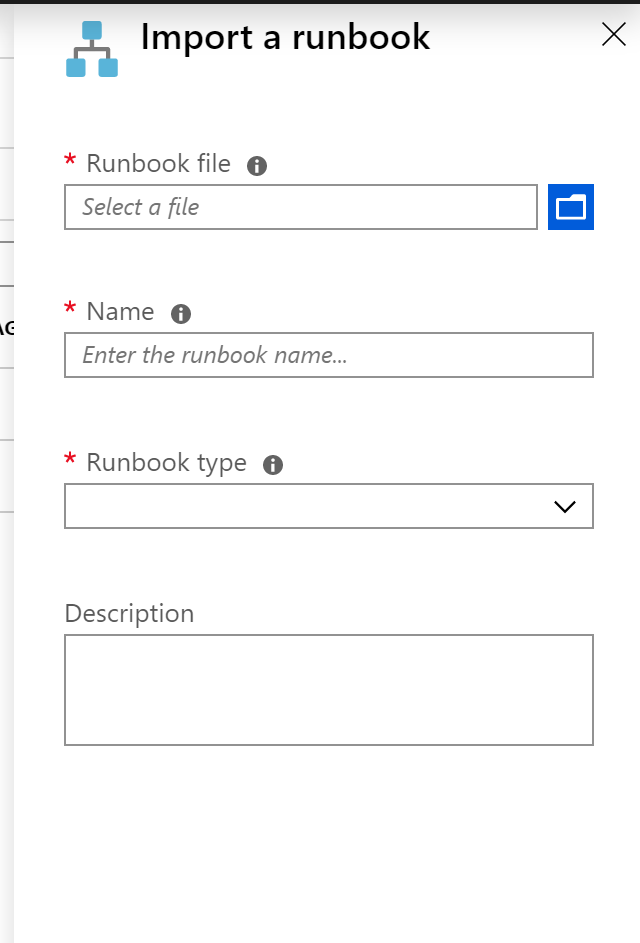
# Azure Automation

The following modules are required to be installed into Azure Automation:

* AZ.DataFactory – used to execute the data factory pipelines
* AZ.AnalysisServices – used to process the Analysis Services databases
* Sqlserver – used to process Analysis Services databases
* AZ.Sql – used to resume Azure SQL Datawarehouse

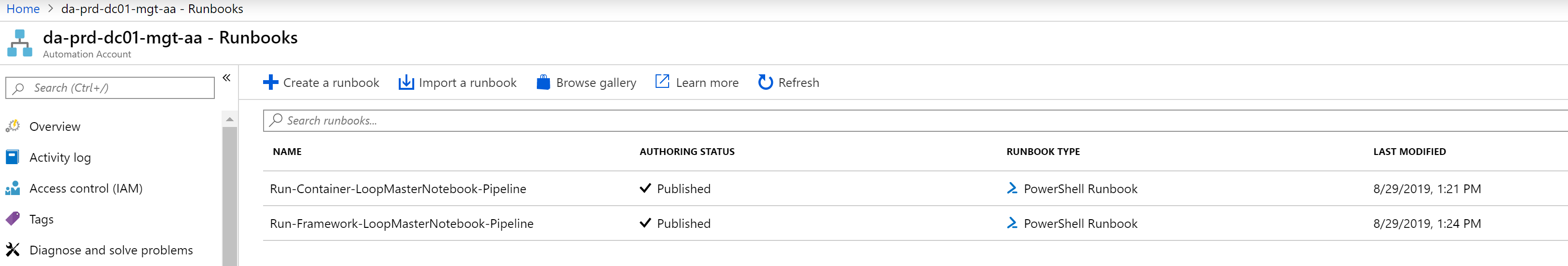
There is an intial set of Azure Automation Runbooks in DevOps that will need to be imported into the Azure Automation Service. Import the .ps1 file for each from the local DevOps folder location. (…\AzureAutomation\da-dev\BI and Analytics\”env folder (dev, tst, or prd)”)

* Start-DataWarehouse.ps1
* Pause-DataWarehouseWF.ps1
* Start-AnalysisServices.ps1
* Pause-AnalysisServicesWF.ps1



After import, publish each Runbook.

When complete you should have at least the two Runbooks shown below



# Sanctioned Zone

## ADLS Gen2 Temp Folder

A temporary folder in ADLS Gen2 is required for the Databricks Azure SQL Data Warehouse connector to function. It is used as a temporary file storage, and should be cleaned up as part of the Databricks notebook processing.

Create a temp blob container as required for the Databricks notebooks. No additional ACLs/Permissions are required.

A screenshot of a cell phone

Description automatically generated

## SQL Data Warehouse

### Create Master Key

In order for the Databricks SQL Data Warehouse connector (Polybase) to work follow the steps below from SSMS:

CREATE MASTER KEY ENCRYPTION BY PASSWORD = '!2398795234nl0zBe==';

### Create Managed Identity

From SSMS, create a Managed Identity, which will be used by the Databricks SQL Datawarehouse connector (Polybase) as follows:

--Create database scoped credential with IDENTITY = 'Managed Service Identity':

CREATE DATABASE SCOPED CREDENTIAL msi\_cred WITH IDENTITY = 'Managed Service Identity';

### Register Azure SQL Server in Active Directory (Service Endpoint)

In **PowerShell**, register your Azure SQL Server hosting your Azure SQL Data Warehouse instance with Azure Active Directory (AAD):

#Connect-AzAccount

Select-AzSubscription -SubscriptionId "7622f100-6233-41cd-a234"

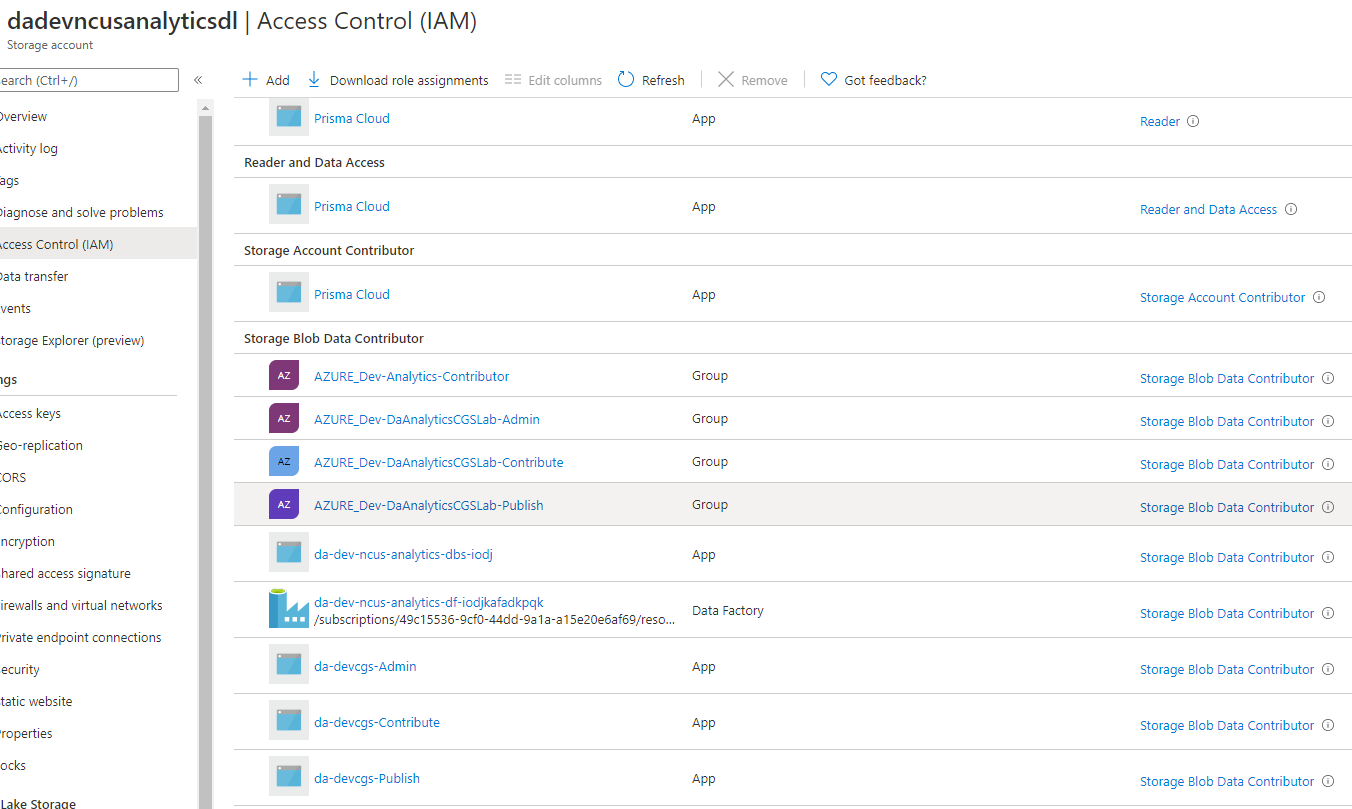
Set-AzSqlServer -ResourceGroupName "da-dev-analytics-rg" -ServerName "da-prd-dc01-analytics-dbs -AssignIdentity

### Add Service Endpoint to ADLS Gen2 Storage Account

Add the Service Endpoint (created above) to the “Storage Blob Data Contributor” IAM Role on the ADLS Gen2 Storage Account.

A screenshot of a social media post

Description automatically generated



## Azure Analysis Services

In order for Azure Analysis Services databases to be processed through Azure Data Facoty the following steps are required: (ref: <https://github.com/furmangg/automating-azure-analysis-services>)

### Add Service Principal as Admin on Analysis Services

The same service principal used above has to be added as Admin to Azure Analysis Services. At the time of this document this must be accomplished via PowerShell. It cannot be done using the Azure Portal.

An example from the Dev environment is shown below.

(Note the $SPAdmin = “app:ApplicationID@TenantID”)

Connect-AzAccount

Set-AzContext -Subscriptionid "b3930f7b-e52b-4bc7-8e82-8f08a334ead3"

$SPAdmin = "app:ca4715af-98c3-4e10-adf8-b7b7faebfa45@ c4d4f36e-6c27-42a7-a8b8-f465f0e75aae"

$resourceGroupName = "da-dev-dc01-analytics-rg"

$analysisServerName = "dadevdc01analyticsaxuczc"

Set-AzAnalysisServicesServer -Name $analysisServerName -ResourceGroupName $resourceGroupName -Administrator $SPAdmin

To get the ADF MSI ID, go to your Azure Data Factory blade and the Properties tab. Copy the Managed Identity Application ID value and the Managed Identity Tenant value. [](https://github.com/furmangg/automating-azure-analysis-services/blob/master/images/AdfMsiID.png)

Combine these two IDs together as follows and add this "user" as an Analysis Services administrator in the Azure portal (Analysis Services Admins)

app:<ApplicationID>@<TenantID>

Example: app:ca4715af-98c3-4e10-adf8-b7b7faebfa45@ c4d4f36e-6c27-42a7-a8b8-f465f0e75aae

Once the Powershell script is executed successfully, the ADF AppID should appear in the list of Analysis Services Admins (as shown below).

