Prerequisites and Setup

ADLS Gen2 Storage Account : https://learn.microsoft.com/en-us/azure/storage/blobs/create-data-lake-storage-account

Azure Data Factory : https://learn.microsoft.com/en-us/azure/data-factory/quickstart-create-data-factory

Azure Key Vault : https://learn.microsoft.com/en-us/azure/key-vault/general/quick-create-portal

Creating a secret : https://learn.microsoft.com/en-us/azure/key-vault/secrets/quick-create-portal

Dataset Used

For this project, we used the AI Bank Dataset available on Kaggle, which simulates core banking data used for analyzing customer behavior, loans, and transactions. The dataset consists of five key CSV files representing different aspects of a banking system.

Source: https://www.kaggle.com/datasets/varunkumari/ai-bank-dataset
Files Used:

- `accounts.csv` – Contains information about customer accounts, including account numbers and associated customer IDs.
- `customers.csv` – Holds demographic and profile information about the bank's customers.
- `loan_payments.csv` – Records the repayment schedules and amounts for issued loans.
- `loans.csv` – Contains data about the loans issued to customers, including loan amounts, terms, and types.
- `transactions.csv` – Provides detailed information about individual transactions, including amounts, types, and timestamps.

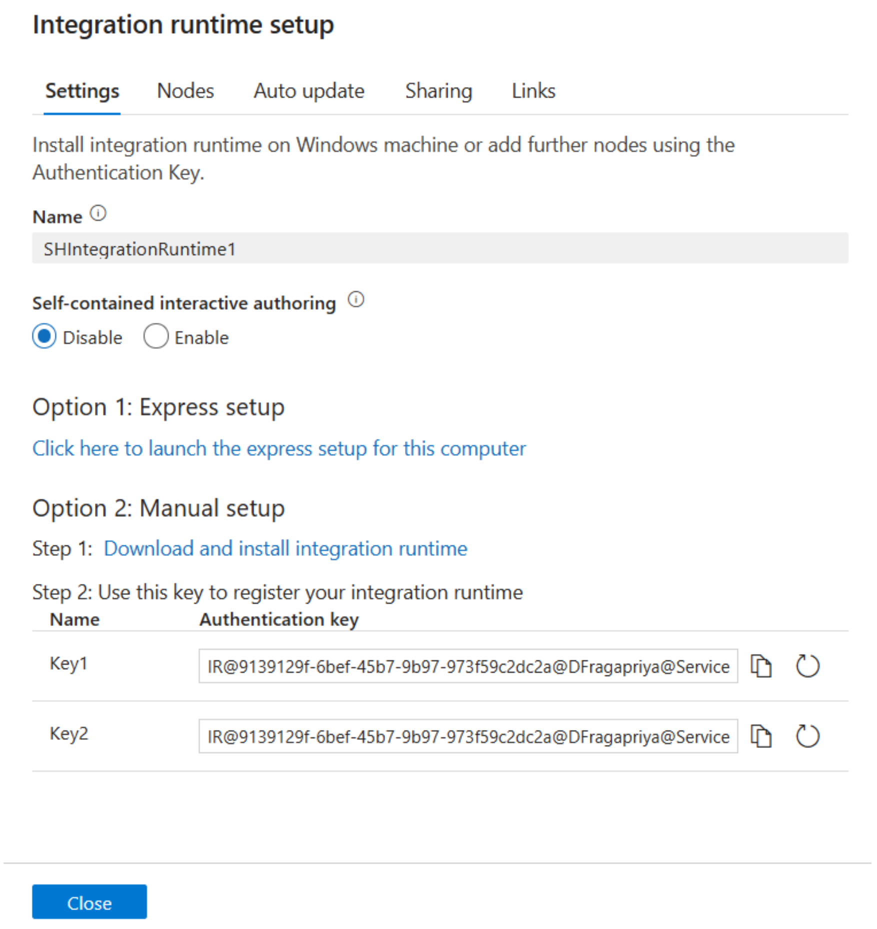
These datasets were used throughout the project to perform data ingestion, cleaning, transformation, and reporting using Azure Data Factory, Azure Data Lake Gen2, Azure SQL Database, and Power BI.

Creating a Self-Hosted Integration Runtime (SHIR)

A Self-Hosted Integration Runtime (SHIR) is required to securely transfer data from the on-premise SQL Server to Azure Data Lake Storage (ADLS) and Azure SQL Database. It acts as a bridge between the on-premise environment and Azure.

Steps to Create SHIR via Azure UI

1. Go to Azure Data Factory
   - Navigate to the Azure Portal and open your Azure Data Factory instance.
2. Create a New Integration Runtime
   - In Manage → Integration Runtimes, click + New.
   - Select "Self-Hosted" and click Continue.
3. Download and Install SHIR
   - Click Download to get the SHIR installer on the on-premise machine.
   - Run the installer and follow the setup instructions.
4. Register the SHIR
   - After installation, open SHIR and enter the authentication key provided in Azure.
   - This registers the SHIR to your Azure Data Factory instance.
5. Test the Connection
   - In Azure, check the status of SHIR (should be Running).
   - Connect the on-premise SQL Server using Linked Services in Data Factory.



Integration runtime setup

Install integration runtime on Windows machine or add further nodes using the Authentication Key.

Name: SHIntegrationRuntime1

Self-contained interactive authoring: [✓ Disable] Enable

Option 1: Express setup

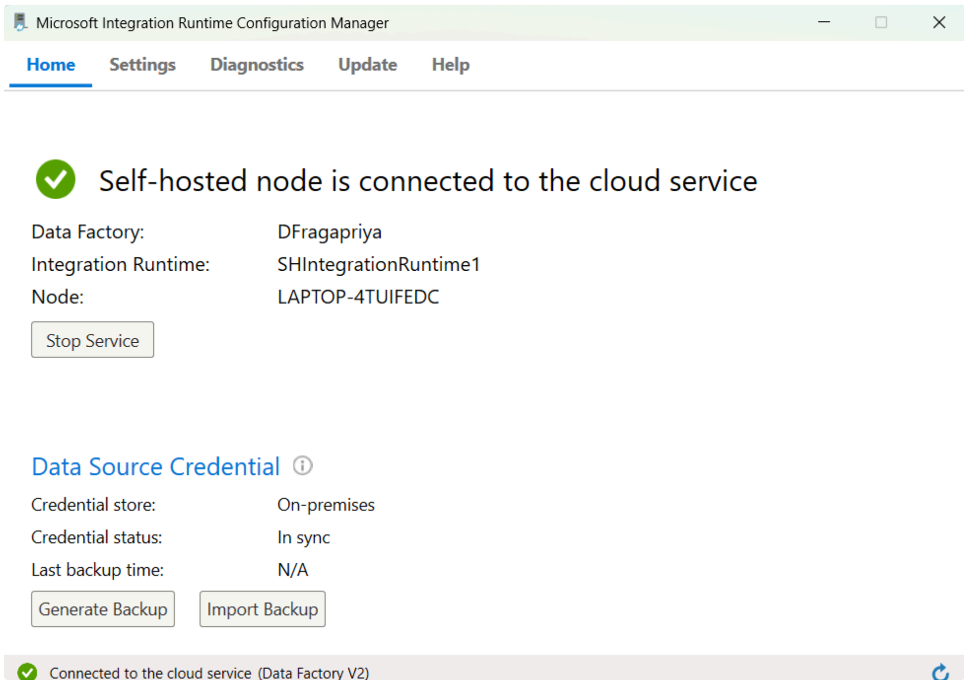
Click here to launch the express setup for this computer

Option 2: Manual setup

Step 1: Download and install integration runtime

Step 2: Use this key to register your integration runtime

Name	Authentication key
Key1	IR@9139129f-6bef-45b7-9b97-973f59c2dc2a@DFragapriya@Service
Key2	IR@9139129f-6bef-45b7-9b97-973f59c2dc2a@DFragapriya@Service



Microsoft Integration Runtime Configuration Manager

Self-hosted node is connected to the cloud service

Data Factory:	DFragapriya
Integration Runtime:	SHIntegrationRuntime1
Node:	LAPTOP-4TUIFEDC

Data Source Credential

Credential store:	On-premises
Credential status:	In sync
Last backup time:	N/A

Connected to the cloud service (Data Factory V2)

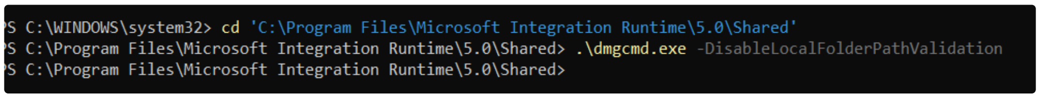
Disabling Local Folder Path Validation in Integration Runtime

As part of the data pipeline setup, we utilized an on-premises File Server as one of the data sources. This involved configuring a File System linked service in Azure Synapse Analytics that accessed a local folder on the machine where the Self-Hosted Integration Runtime (IR) was installed.

By default, the Microsoft Integration Runtime enforces validation on local folder paths for security and consistency. During configuration, the following error was encountered:

> *"Path validation failed for host path in File System linked service."*

To resolve this issue and allow the use of local folder paths (e.g., `C:\Users\...\dataset`), we disabled the folder path validation using the `dmgcmd.exe` utility provided with the IR installation.



```
S C:\WINDOWS\system32> cd 'C:\Program Files\Microsoft Integration Runtime\5.0\Shared'
S C:\Program Files\Microsoft Integration Runtime\5.0\Shared> .\dmgcmd.exe -DisableLocalFolderPathValidation
S C:\Program Files\Microsoft Integration Runtime\5.0\Shared>
```

Implementation Steps:

1. Open Windows PowerShell with administrator privileges on the machine where the self-hosted IR is installed.
2. Change the directory to the shared tools folder of the Integration Runtime:

```
cd "C:\Program Files\Microsoft Integration Runtime\5.0\Shared"
```

3. Execute the following command to disable local folder path validation:

```
.\dmgcmd.exe -DisableLocalFolderPathValidation
```

This command allowed the Integration Runtime to accept the specified local path in the File Server linked service configuration.

By applying this fix, we successfully integrated on-premises data into our Azure Synapse pipelines and enabled seamless data ingestion from the local file system to Azure Data Lake Storage Gen2.

Assigning Folder-Level Permissions for On-Premise Folder Access

As part of the project, we needed to access a local on-premise folder from Azure Data Factory (ADF) using the Self-hosted Integration Runtime (SHIR). This required proper folder-level permissions to be set up on the local machine to allow successful authentication and data access.

Folder Path: `C:\Users\ragap\Desktop\Data Engineering\Bootcamp\Project_1\dataset`

Steps to Assign Folder-Level Permissions

1. **Locate the Folder**:
   - Navigate to:
     `C:\Users\ragap\Desktop\Data Engineering\Bootcamp\Project_1\dataset`
2. **Right-Click and Choose ‘Properties’**:
   - Go to the **Security** tab.
3. **Click ‘Edit’ to Change Permissions**.
4. **Add the User**:
   - Click **Add**, then enter the Windows username that is being used in your File Server linked service.
     (In this case: `ragapriya.saravanan14@gmail.com`)
5. **Grant Required Permissions**:
   - Check the following permissions:
     - **Read & execute**
     - **List folder contents**
     - **Read**
6. **Click Apply and OK** to save the changes.

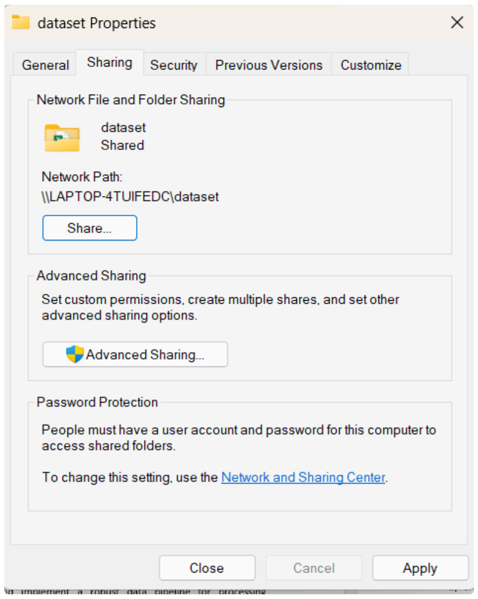
> ✅ Note: These permissions ensure that the Integration Runtime can read files from the specified local folder and move them to Azure or process them as required by your pipeline.

Why This Was Important

Without the proper permissions:

- The Self-hosted Integration Runtime would fail to access the folder.
- You might encounter errors like **"Access Denied"** or **"The system cannot find the path specified"** when running the pipeline.

This permission setup was critical for securely and successfully integrating on-premise data sources into our Azure-based data pipeline.


dataset Properties

Network File and Folder Sharing

dataset
Shared

Network Path:
\\LAPTOP-4TUIFEDC\dataset

Advanced Sharing

Set custom permissions, create multiple shares, and set other advanced sharing options.

Password Protection

People must have a user account and password for this computer to access shared folders.

To change this setting, use the Network and Sharing Center.

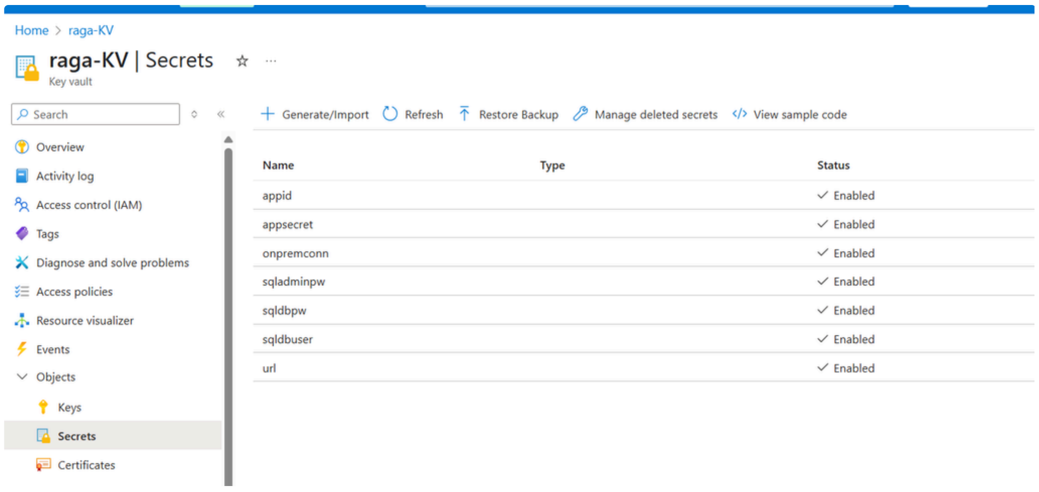
Storing Secrets in Azure Key Vault and Securing Access for ADF

To enhance security and follow best practices, sensitive credentials like database passwords and on-premise file system credentials were stored securely in Azure Key Vault. This approach avoids hardcoding credentials directly within linked services and ensures centralized, managed secret storage.

Secrets Used in the Project

In this project, the following secrets were stored in the Key Vault named `raga-KV`:

1. `sqldbpw` – Password for the SQL Server user (`sqladmin`) used in the Azure SQL Database linked service (`ls_sql_db`).
2. `onpremconn` – Password for accessing the local file system used in the on-premises File Server linked service (`FileServer1`).



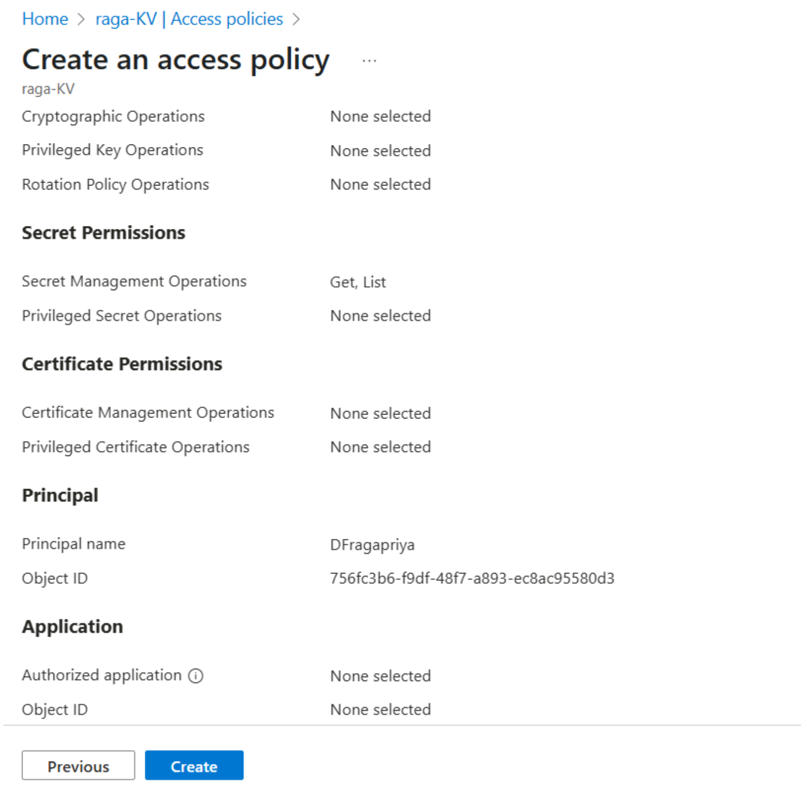
Name	Type	Status
appid		✓ Enabled
appsecret		✓ Enabled
onpremconn		✓ Enabled
sqladminpw		✓ Enabled
sqldbpw		✓ Enabled
sqldbuser		✓ Enabled
url		✓ Enabled

Configuring Key Vault Access for Azure Data Factory

To allow Azure Data Factory (ADF) to retrieve these secrets securely during runtime, we performed the following steps:

1. **Create Azure Key Vault (`raga-KV`)**
   - Secrets (`sqldbpw` and `onpremconn`) were added manually.
2. **Grant ADF Access Using Access Policies**

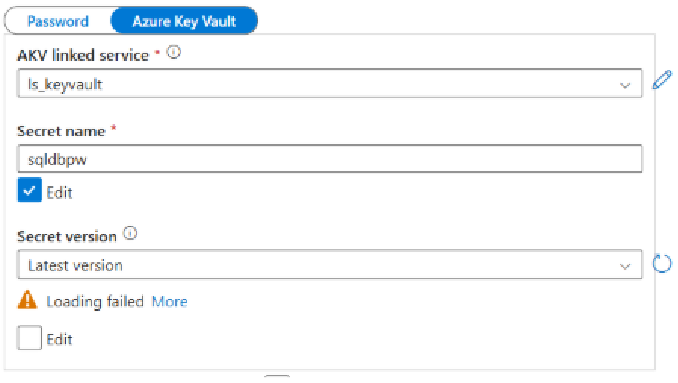
   In the Key Vault, under Access policies, a new policy was added:
   - **Principal**: The Managed Identity of the Azure Data Factory instance.
   - **Permissions**:
     - Secret Permissions: `Get` and `List` (only these are required for ADF to read secrets).



Cryptographic Operations	None selected
Privileged Key Operations	None selected
Rotation Policy Operations	None selected
Secret Permissions	
Secret Management Operations	Get, List
Privileged Secret Operations	None selected
Certificate Permissions	
Certificate Management Operations	None selected
Privileged Certificate Operations	None selected
Principal	
Principal name	DFragapriya
Object ID	756fc3b6-f9df-48f7-a893-ec8ac95580d3
Application	
Authorized application	None selected
Object ID	None selected

3. **Referencing Secrets in Linked Services**

   In the linked service JSON definitions, instead of plain text passwords, secrets were referenced using the `AzureKeyVaultSecret` type. Example from `ls_sql_db` :



Password | Azure Key Vault

AKV linked service *
ls_keyvault

Secret name *
sqldbpw
✓ Edit

Secret version
Latest version
Loading failed More
Edit

- This means:
- `ls_keyvault` is the linked service pointing to Azure Key Vault.
    - `sqldbpw` is the name of the secret inside the vault.

Benefits in the Project

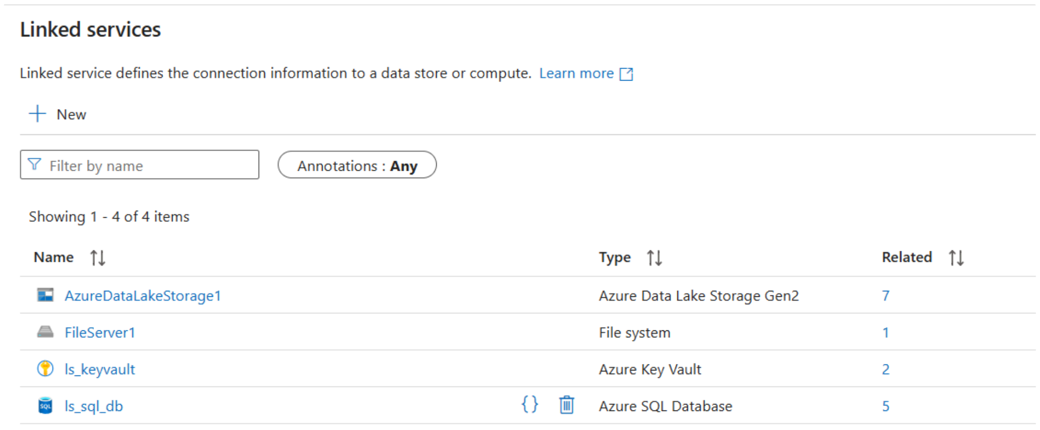
- Improved Security: Passwords are never exposed in plain text.
- Centralized Management: Secrets can be updated in one place without modifying linked services.
- Compliance & Auditing: Key Vault provides logging and monitoring through Azure Monitor and Activity Logs.

This integration between Azure Key Vault and Azure Data Factory ensures secure and scalable credential management in the project.

Linked Services

A Linked Service in Azure Data Factory (ADF) or Synapse Pipelines is similar to a connection string—it defines the connection information that the service uses to connect to external data sources or destinations. This can include databases, storage accounts, key vaults, file systems, REST APIs, etc.

In your project, several linked services were configured to handle different stages of your data pipeline—from raw data ingestion to cleaned data storage and reporting. Here's how each one was used:



Linked services

Linked service defines the connection information to a data store or compute. Learn more

Showing 1 - 4 of 4 items

Name	Type	Related
AzureDataLakeStorage1	Azure Data Lake Storage Gen2	7
FileServer1	File system	1
ls_keyvault	Azure Key Vault	2
ls_sql_db	Azure SQL Database	5

1. `ls_keyvault` – Azure Key Vault Linked Service

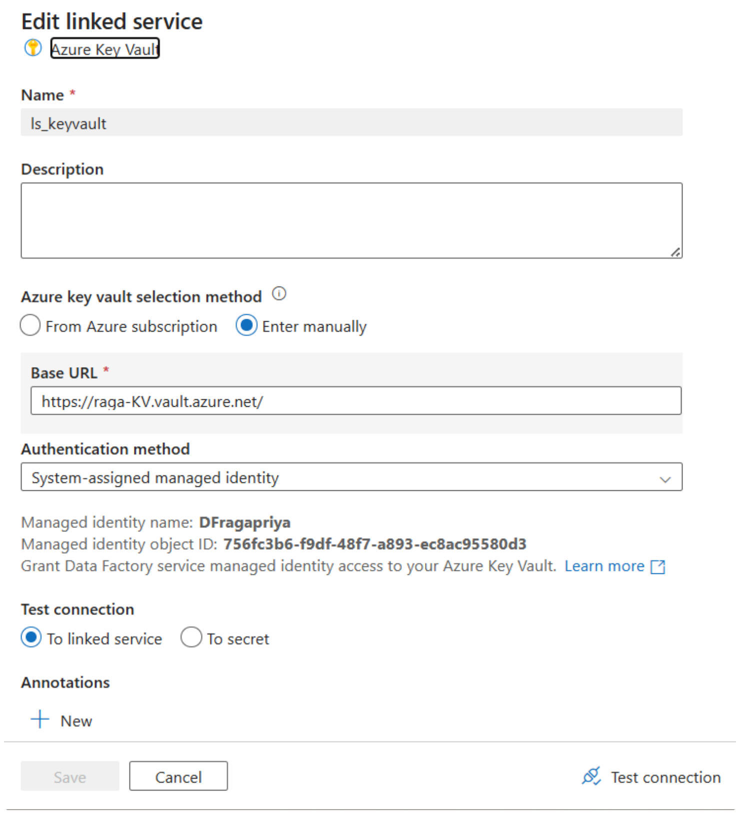
Purpose:
This linked service connects to Azure Key Vault for securely storing and retrieving secrets like passwords, keys, and connection strings.

- **Used For**: Securely storing sensitive credentials like passwords for SQL DB and on-prem connections.

- **How It’s Used**: Other linked services (`ls_sql_db`, `FileServer1`) reference `ls_keyvault` to fetch credentials at runtime, ensuring no hard-coded secrets are exposed in the pipeline.

Key Configurations:

- **Base URL**: `https://raga-KV.vault.azure.net/`
- Used by other services like `ls_sql_db` and `FileServer1` to retrieve secrets like `sqldbpw` and `onpremconn`.



Edit linked service
Azure Key Vault

Name *
ls_keyvault

Description

Azure key vault selection method
[ ] From Azure subscription [✓ Enter manually]

Base URL *
https://raga-KV.vault.azure.net/

Authentication method
System-assigned managed identity

Managed identity name: **DFragapriya**
Managed identity object ID: **756fc3b6-f9df-48f7-a893-ec8ac95580d3**
Grant Data Factory service managed identity access to your Azure Key Vault. Learn more

Test connection
[✓ To linked service] [ ] To secret

Annotations
+ New

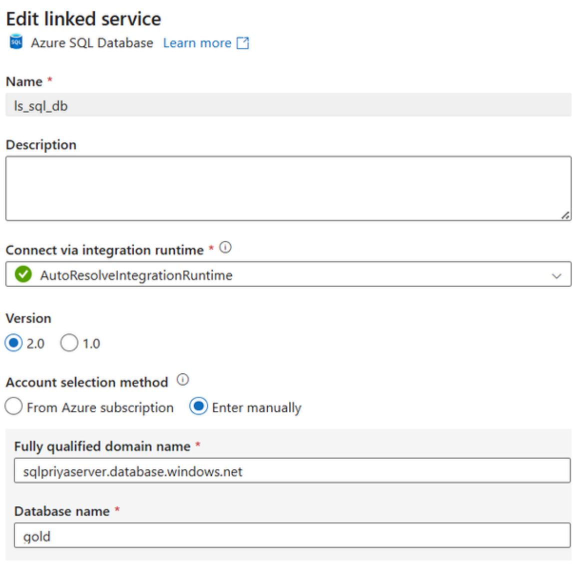
2. `ls_sql_db` – Azure SQL Database Linked Service

Purpose:
This linked service connects your Synapse or Data Factory pipeline to an Azure SQL Database where your final (gold layer) data is stored. It's used to read from or write to this database during pipeline execution or report creation in Power BI.

- **Used For**: Writing the final, curated data into Azure SQL Database so it can be used for reporting and analytics.
- **How It’s Used**: At the final stage of your pipeline, cleaned and transformed data is pushed into Azure SQL DB tables using this linked service. It’s also used by Power BI to fetch data for report creation.

Key Configurations:

- **Server**: `sqlpriyaserver.database.windows.net`
- **Database**: `gold`
- **Authentication**: SQL authentication using a **username (**`sqladmin`**)** and a **password** retrieved securely from **Azure Key Vault (**`sqldbpw`**)**.
- **Security**: Encryption enforced (`encrypt: mandatory`) and server certificate not trusted blindly (`trustServerCertificate: false`).


Edit linked service
Azure SQL Database Learn more

Name *
ls_sql_db

Description

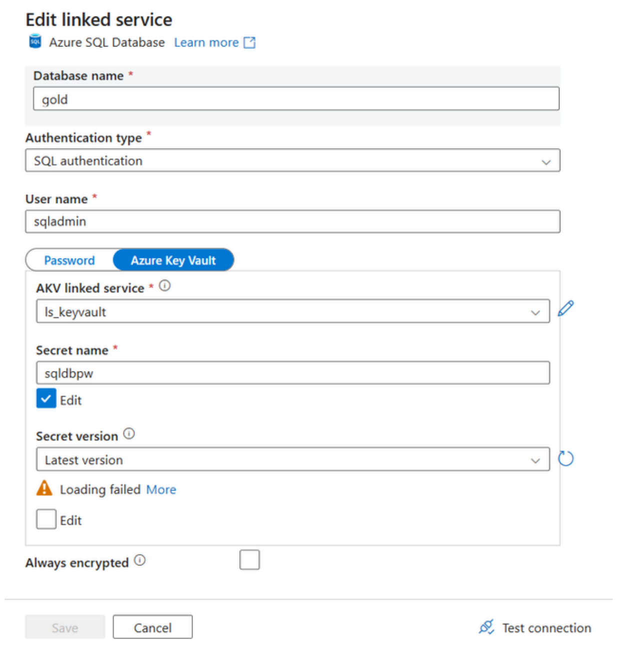
Connect via integration runtime *
AutoResolveIntegrationRuntime

Version
[✓ 2.0] 1.0

Account selection method
From Azure subscription [✓ Enter manually]

Fully qualified domain name *
sqlpriyaserver.database.windows.net

Database name *
gold


Edit linked service
Azure SQL Database Learn more

Database name *
gold

Authentication type *
SQL authentication

User name *
sqladmin

Password [✓ Azure Key Vault]

AKV linked service *
ls_keyvault

Secret name *
sqldbpw
[✓ Edit]

Secret version
Latest version
Loading failed More
Edit

Always encrypted

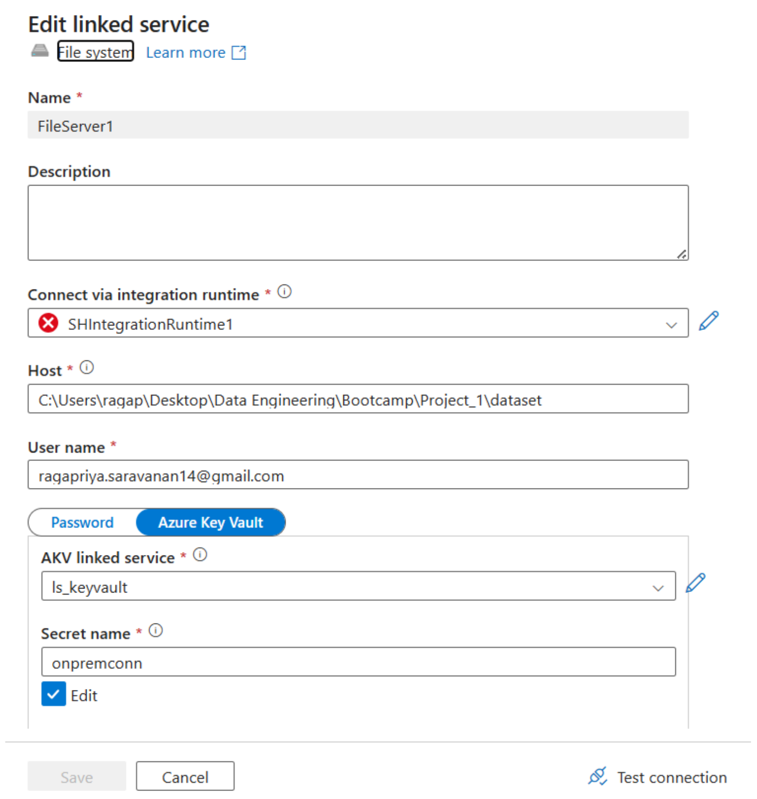
3. `FileServer1` – On-premises File Server Linked Service

Purpose:
This linked service allows access to on-premises files (like CSV or Excel datasets) stored on a local machine, typically used for initial raw data ingestion.

- **Used For**: Ingesting data from on-premises folders where your original datasets were stored.
- **How It's Used**: Connects ADF/Synapse to your local machine using a self-hosted integration runtime. The dataset is pulled into the cloud from this local directory and sent to Azure Data Lake for further processing.

Key Configurations:

- **Host path**: Local folder on the machine – `C:\Users\ragap\Desktop\Data Engineering\Bootcamp\Project_1\dataset`
- **Authentication**: Windows login (user ID and password retrieved from Azure Key Vault `onpremconn`)
- **Integration Runtime**: `SHIntegrationRuntime1` – a Self-hosted Integration Runtime that bridges cloud services with on-premises data sources.



Edit linked service

File system Learn more

Name *

FileServer1

Description

Connect via integration runtime *

SHIntegrationRuntime1

Host *

C:\Users\ragap\Desktop\Data Engineering\Bootcamp\Project_1\dataset

User name *

ragapriya.saravanan14@gmail.com

Password | Azure Key Vault

AKV linked service *

ls_keyvault

Secret name *

onpremconn

✓ Edit

Save Cancel Test connection

4. `AzureDataLakeStorage1` – Azure Data Lake Gen2 Linked Service

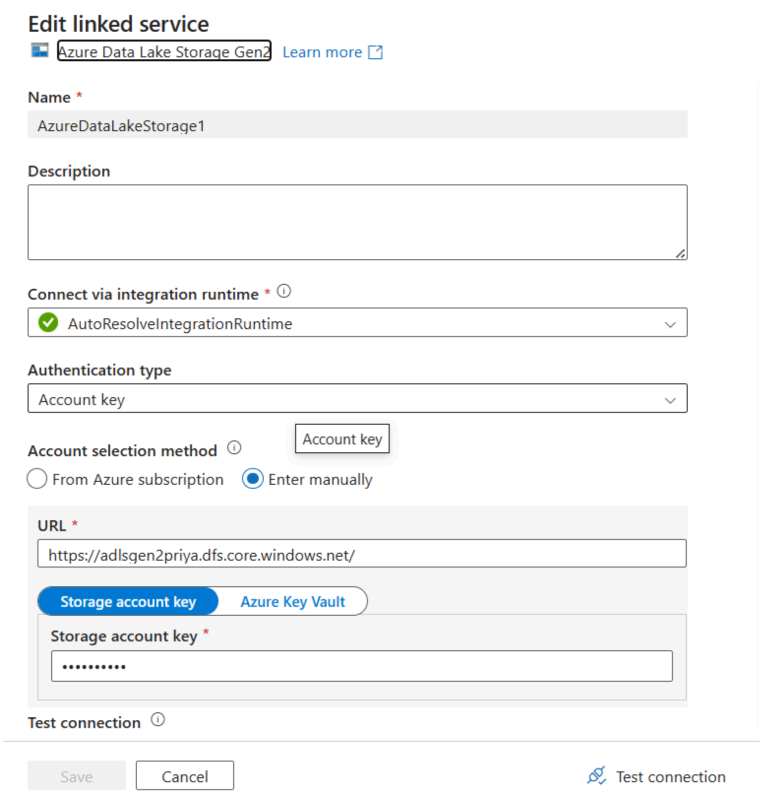
Purpose:

This linked service connects to Azure Data Lake Storage Gen2, used for storing raw (bronze), cleaned (silver), and curated (gold) data during your pipeline execution and transformations.

- **Used For**: Storing raw, cleaned, and transformed data in Azure Data Lake Storage Gen2 (bronze, silver, gold layers).
- **How It's Used**: After the data is ingested from the on-premises file server, it is staged in the raw layer of Data Lake. The transformed outputs are saved here as well, based on the pipeline's logic.

Key Configurations:

- **URL**: `https://adlsgen2priya.dfs.core.windows.net/` – Data Lake account endpoint
- **Credential**: Encrypted credential for secure access to the storage account


Edit linked service

Azure Data Lake Storage Gen2 Learn more

Name *

AzureDataLakeStorage1

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method

( ) From Azure subscription [✓ Enter manually]

URL *

https://adlsgen2priya.dfs.core.windows.net/

[✓ Storage account key] Azure Key Vault

Storage account key *

••••••••••

Test connection

Summary Table

Linked Service	Type	Purpose	Authentication
`ls_sql_db`	Azure SQL Database	Connect to SQL DB to read/write gold data	Azure Key Vault + SQL Auth
`ls_keyvault`	Azure Key Vault	Secure storage of secrets (passwords, keys)	N/A
`FileServer1`	File Server (On-Prem)	Access on-premise datasets for ingestion	Azure Key Vault + Self-hosted IR
`AzureDataLakeStorage1`	Azure Data Lake Gen2	Store data in bronze, silver, and gold layers	Encrypted Credential