Technical Document: Data Ingestion Using Azure Data Factory (Part 1)

Introduction

In this technical document, we will walk through the process of setting up Azure Data Factory (ADF) for data ingestion from an on-premises SQL Server database to Azure Data Lake Storage. This is Part 1 of the data ingestion process, focusing on establishing the necessary connections and copying a single table as a test.

Prerequisites

Before we begin, ensure you have the following prerequisites in place:

An Azure subscription.

Azure Data Factory created in your Azure environment.

An on-premises SQL Server database.

Part 1: Establishing the Connection

Step 1: Access Azure Data Factory

Navigate to your Azure Resource Group.

Click on the Azure Data Factory resource to access its page.

Step 2: Launch Azure Data Factory Studio

Inside the Azure Data Factory page, locate and click the "Launch Studio" button. This will take you to the Azure Data Factory workspace, where you will develop data ingestion pipelines.

Step 3: Install Self-Hosted Integration Runtime

To connect to an on-premises SQL Server database, you need to install the "Self-Hosted Integration Runtime" on the machine where the database resides. In this example, we're using a personal laptop.

In the Azure Data Factory workspace, click on the "Manage" tab on the left.

Under "Integration Runtimes," you will see an existing integration runtime called "Auto Resolve Integration Runtime," primarily used for connecting to cloud-based resources. However, to connect to on-premises data sources, we need to create a new integration runtime called "Self-Hosted Integration Runtime."

Step 4: Create Self-Hosted Integration Runtime

Click on the "New" button and select "Self-Hosted" Integration Runtime.

Give it a name, e.g., "Self-Hosted IR," and provide a description (e.g., "Used to connect to SQL Server").

Choose the "Express Setup" option for simplicity, or "Manual Setup" if installed on a different machine. The Express Setup installs and configures the runtime with minimal user input.

Click the link provided for Express Setup to download and run the installation application.

The installation will automatically handle key-based authentication and establish a connection between Azure Data Factory and you're on-premises database. Once installation is complete, close the installation window.

Back in Azure Data Factory, you will see that the Self-Hosted Integration Runtime is now running, indicating a successful connection to your local machine.

Step 5: Verify Integration Runtime

To verify the integration runtime, search for "Integration Runtime" in your system, and you should see "Microsoft Integration Runtime." Click on it.

In the opened window, you should see the Self-Hosted node connected to the cloud service, displaying the integration runtime name you created (e.g., "Self-Hosted IR") and the Azure Data Factory name (e.g., "ADF-demo01").

Part 2: Configuring Data Ingestion

Step 1: Create a Data Ingestion Pipeline

In Azure Data Factory, go to the "Author" tab and click on "Pipelines."

Create a new pipeline by clicking the "+" icon and naming it (e.g., "Copy Pipeline").

Step 2: Add a Copy Data Activity

Drag and drop the "Copy Data" activity onto the pipeline canvas. Rename it (e.g., "Copy Address Table").

Step 3: Configure Source Data Set

Click on the source settings of the "Copy Address Table" activity.

Create a new source data set to represent the "Address" table from your on-premises SQL Server database.

Select "SQL Server" as the data store type.

Provide a name for the data set (e.g., "Address").

Configure the link service connection to your on-premises SQL Server. This includes specifying the server's name, database name, authentication type, username, and password.

Note: If you want to secure your password, you can use Azure Key Vault to store and retrieve it securely.

Test the connection to ensure it's successful.

Step 4: Configure Sink Data Set

Click on the sink settings of the "Copy Address Table" activity.

Create a new sink data set to represent your Azure Data Lake Storage.

Select the desired file format (e.g., Parquet).

Configure the link service connection to your Azure Data Lake Storage, specifying the storage account name, integration runtime (Auto Resolve for cloud-based resources), and authentication type (e.g., Access Key).

Specify the file path within your Data Lake where you want to copy the data (e.g., "bronze" container).

Step 5: Run the Pipeline

Save your change and use the "Debug" option to run the pipeline.

Monitor the pipeline's progress to ensure that it successfully copies the "Address" table from your on-premises SQL Server database to Azure Data Lake Storage.

Verify that the data is copied to the designated location in Azure Data Lake Storage.

Configuring a pipeline to copy multiple tables from SQL Server:

Creating a new pipeline.

Defining a pipeline name ("Copy All Tables").

Using a Lookup Activity to retrieve table names:

Creating a Lookup Activity in the pipeline.

Configuring the source dataset for SQL Server.

Using a SQL query to retrieve schema and table names.

Previewing the output data.

Utilizing ForEach Activity to iterate through tables:

Adding a ForEach Activity to the pipeline.

Linking the Lookup Activity as the input for ForEach.

Configuring ForEach to iterate through retrieved tables.

Configuring the folder structure for storing data in Data Lake:

Creating parameters for schema name and table name.

Using parameters to dynamically generate the folder structure.

Configuring source and sink datasets in the copy data activity.

Running and monitoring the pipeline:

Triggering the pipeline execution.

Monitoring pipeline progress.

Verifying the data ingestion in the Azure Data Lake Storage Gen2.

Conclusion

In this technical document, we successfully established a connection between Azure Data Factory and an on-premises SQL Server database using a Self-Hosted Integration Runtime. We also configured a data ingestion pipeline to copy a single table from the on-premises database to Azure Data Lake Storage for testing purposes. In Part 2, we will expand on this setup to copy all tables from the SQL Server database.

Note: This document provides a step-by-step guide for educational purposes. Actual configurations may vary based on your specific Azure environment and requirements. Always follow best practices for security and data management.