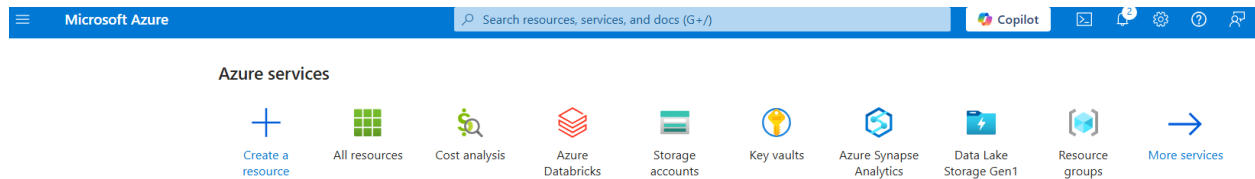


Data pipeline SQL on-premise to Data Lake

Azure Data Factory Pipeline Configuration Details

The file provides a detailed walkthrough of setting up an Azure Data Factory (ADF) pipeline to extract data from an on-premises SQL Server database, transform it, and load it into an Azure Data Lake. Below, I'll summarize the exact configurations, queries, and parameters used in the ADF pipeline for the **source** and **sink** components, as described in the file.



Step1- Setting Up the Azure Environment

1- Create a Resource Group

1-1- Log in to Azure Portal: Go to portal.azure.com and sign in.

1-2- Create a Resource Group:

- In the Azure portal, search for Resource Group in the search bar.
- Click Create.
- Fill in the details:
 - Subscription: Select your subscription.
 - Resource Group Name: xxxx-RG.
 - Region: Choose the closest region.
- Click Review + Create and then Create.

A screenshot of the "Create a resource group" form in the Azure portal. The form has three tabs: "Basics", "Tags", and "Review + create". The "Basics" tab is selected. Below the tabs, there's a description of a "Resource group". The form contains three input fields: "Subscription" with a dropdown menu showing "Azure for Students", "Resource group name" with a text input field containing "Projects_RG", and "Region" with a dropdown menu showing "(Europe)".

2- Create a Storage Account (Azure Data Lake)

2-1- Search for Storage Account:

- In the Azure portal, search for **Storage Account**.
- Click **Create**.

Fill in the Details:

- **Subscription:** Select your subscription.
- **Resource Group:** Select the resource group you created (e.g., Xxxx-RG).
- **Storage Account Name:** e.g., Xxxx-SG.
- **Region:** Choose the same region as your resource group.
- **Enable Hierarchical Namespace:** Check this box to enable Data Lake capabilities.
- Click **Review + Create** and then **Create**.

Microsoft Azure Search resources, services, and docs (G+/J)

Home > Resource groups > Projects_RG > Marketplace > Storage account >

Create a storage account ...

manage your storage account together with other resources.

Subscription * Azure for Students

Resource group * Projects_RG
[Create new](#)

Instance details

Storage account name * ① projectboxsg

Region * ① (Europe) Italy North
[Deploy to an Azure Extended Zone](#)

Primary service ① Select a primary service

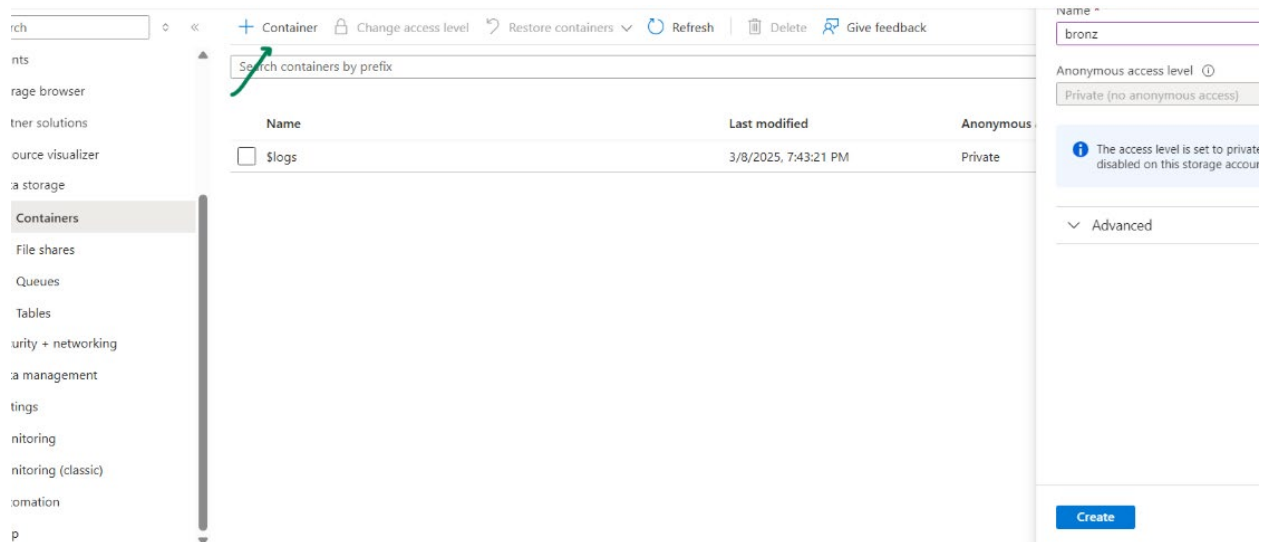
Performance * ① ☒ **Standard:** Recommended for most scenarios (general-purpose v2 account)
☐ **Premium:** Recommended for scenarios that require low latency.

Redundancy * ① Locally-redundant storage (LRS)

[Previous](#) [Next](#) [Review + create](#)

Create Containers:

- Once the storage account is created, go to the storage account.
- Under **Containers**, create the container for raw data.



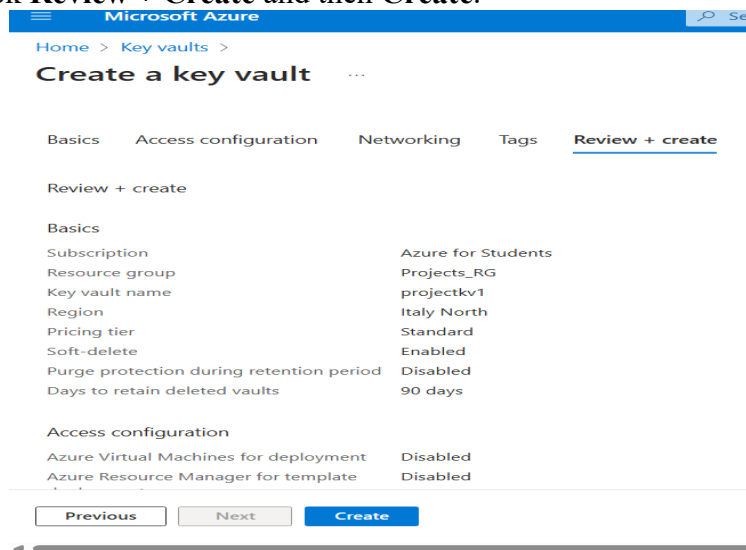
3- Create a Key Vault

1. Search for Key Vault:

- In the Azure portal, search for **Key Vault**.
- Click **Create**.

2. Fill in the Details:

- **Subscription:** Select your subscription.
- **Resource Group:** Select the resource group you created (Xxxx-RG).
- **Key Vault Name:** Xxxx-KeyVault.
- **Region:** Choose the same region as your resource group.
- **Pricing Tier:** Standard.
- Click **Review + Create** and then **Create**.



4- Create Azure Data Factory

1. Search for Data Factory:

- In the Azure portal, search for **Data Factory**.
- Click **Create**.

2. Fill in the Details:

- **Subscription:** Select your subscription.
- **Resource Group:** Select the resource group you created (e.g., Xxxx-RG).
- **Name:** e.g., Xxxx-DF.
- **Region:** Choose the same region as your resource group.
- **Version:** Leave as default.
- Click **Review + Create** and then **Create**.

The screenshot shows the 'Create Data Factory' page in the Azure portal. The page has a blue header with the Microsoft Azure logo and a search bar. Below the header, there's a breadcrumb trail: Home > Create Data Factory. The main content area is titled 'Create Data Factory' and has a tabbed interface with 'Basics' selected. Below the tabs, there's a link to 'One-click to create data factory with sample pipeline and datasets. Try it'. The 'Project details' section is visible, with instructions to 'Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.' The form fields are as follows: Subscription (Azure for Students), Resource group (Projects_RG), Name (ADFprojectsxp), Region (Italy North), and Version (V2).

Step 2: Setting Up SQL Server and Database

1- Install SQL Server and SSMS

Download SQL Server Express:

- Go to the [SQL Server download page](#) and download SQL Server Express.
- Install SQL Server Express on your local machine.

Download SQL Server Management Studio (SSMS):

- Download and install SSMS from the [official Microsoft page](#).

Load the AdventureWorks Database:

- Download the AdventureWorks sample database from the [Microsoft website](#).

2. Restore the Database:

- Open SSMS and connect to your SQL Server instance.
- Right-click on **Databases** and select **Restore Database**.
- Choose **Device** and browse to the .bak file you downloaded.

- Restore the database.

2- Create a SQL User for Data Factory

1. Create a SQL Login:

- Open SSMS and connect to your SQL Server instance.
- Run the following SQL query to create a login and user:
- The azure key vault needs a username and password to connect the SQL


```
CREATE LOGIN username WITH password = 'XXXX';
CREATE USER username FOR LOGIN username;
```

2. Store SQL Credentials in Key Vault:

1. Grant Data Factory Access to Key Vault:

- Go to **Access control** in Key Vault.
- Add a new role assignment:
 - Choose Key vault administrator and user, group and service principals.
 - Select the member.

The screenshot shows the 'Members' tab in the Azure Key Vault Access Control interface. At the top, there are tabs for 'Role', 'Members', 'Conditions', and 'Review + assign'. The 'Members' tab is selected. Below the tabs, the 'Selected role' is 'Key Vault Administrator'. Under 'Assign access to', the 'User, group, or service principal' radio button is selected. The 'Members' section shows a list with a '+ Select members' link. Below this, there is a table with columns 'Name' and 'Description'. The 'Name' column contains 'Data Factory (Guest)' and the 'Description' column contains 'Optional'. At the bottom, there are three buttons: 'Review + assign' (highlighted in blue), 'Previous', and 'Next'.

- Save the policy.

2. Add Secrets to Key Vault:

- Go to your Key Vault in the Azure portal.
- Under **Secrets**, click **Generate/Import**.
- Add two secrets:
 - **Username:** xxxx.
 - **Password:** xxxx.

Home > projectkv1 | Secrets >

Home > projectkv1 | Secrets >

Create a secret

Upload options: Manual

Name * ⓘ: PASSWORD

Secret value * ⓘ:

Content type (optional):

Set activation date ⓘ: ☐

Set expiration date ⓘ: ☐

Enabled: Yes

Tags: 0 tags

Upload options: Manual

Name * ⓘ: usernmae

Secret value * ⓘ:

Content type (optional):

Set activation date ⓘ: ☐

Set expiration date ⓘ: ☐

Enabled: Yes

Tags: 0 tags

Source Dataset Configuration:

- **Dataset Name:** SQL Server on-prem
- **Table Name:** Initially, dbo.DimCustomer (later modified to dynamically fetch all tables).
- **Query:**
 - For a single table: SELECT * FROM dbo.DimCustomer
 - For all tables: The lookup activity dynamically generates the query using the schema and table names.

Lookup Activity (to fetch table names):

- **Query:**
 SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
 WHERE TABLE_TYPE = 'BASE TABLE'
 ORDER BY TABLE_SCHEMA, TABLE_NAME;
 This query retrieves all table names under the dbo schema.
- **Output:** The lookup activity returns a JSON object containing the schema and table names, which is used in the ForEach loop.

2. Sink Configuration (Azure Data Lake Gen 2)

Linked Service Configuration:

- **Linked Service Name:** Azure Data Lake Storage Gen 2
- **Integration Runtime:** Auto-resolve (since the Data Lake is in Azure).
- **Authentication Type:** Default (uses the Azure subscription credentials).
- **Storage Account Name:** The name of the Azure Data Lake Storage account.

Sink Dataset Configuration:

- **Dataset Name:** Parquet sink
- **File Format:** Parquet (column-based format optimized for querying).

- **File Path:**
 - **Container:** name
 - **Directory:** dbo/{table_name}
 - **File Name:** {table_name}.parquet
- **Parameters:**
 - schemaName: Dynamically populated from the ForEach loop (e.g., dbo).
 - tableName: Dynamically populated from the ForEach loop (e.g., Address, Customer, etc.).

Dynamic File Path Configuration:

- The file path and file name are dynamically generated using the following expressions:
 - **Directory Path:**
@concat(dataset().schemaName, '/', dataset().tableName)
 - **File Name:**
@concat(dataset().tableName, '.parquet')
-

3. Pipeline Configuration



Pipeline Name: Copy All Tables

Activities:

1. **Lookup Activity:**
 - Fetches the list of tables from the SQL Server database.
 - Outputs a JSON object with schema and table names.
 2. **ForEach Activity:**
 - Iterates over the list of tables returned by the Lookup activity.
 - **Items:** @activity('Lookup1').output.value
 - **Activities Inside ForEach:**
 - **Copy Data Activity:**
 - **Source:** new dataset>> sql server>> name: sqlserveron_prem
 - **Linked Service Configuration:**
- **Linked Service Name:** sqlserver_ls
 - **Integration Runtime:** Self-hosted integration runtime (since the SQL Server is on-premises)



The screenshot displays the Azure Data Factory (ADF) portal interface. On the left, the 'Factory Resources' pane shows a tree view with 'Pipelines' expanded, listing 'copy data'. The main canvas shows the 'copy data' pipeline with a 'ForEach' activity. A 'Copy data' activity is being configured within the 'ForEach' loop, with its source set to 'new dataset' and 'sql server'. The 'Integration runtime setup' pane on the right shows the 'Settings' tab, where the 'Name' is 'integrationRuntime'. A green box highlights the 'Option 1: Express setup' link, which says 'Click here to launch the express setup for this computer'. Below this, 'Option 2: Manual setup' is visible, including steps for downloading and installing the integration runtime, and a table for authentication keys (Key1 and Key2) with their respective values.


-
- **Server Name:** The name of the SQL Server instance
- **Database Name:** AdventureWorks DW 2019
- **Authentication Type:** SQL Authentication
- **Username:** xxxx (created in SQL Server)
- **Password:** Stored in Azure Key Vault (retrieved dynamically using a linked service to Key Vault).

New linked service
 SQL server [Learn more](#) 

Name *


Description

Connect via integration runtime * ⓘ
☒ integrationRuntime  


Version
☒ Recommended ☐ Legacy
 Import from connection string

Server name *

Database name *

Authentication type
 

User name *

[Password](#) 

AKV linked service:

New linked service

 Azure Key Vault

Name *

AzureKeyVault1

Description

Azure key vault selection method [ⓘ]

☒ From Azure subscription ☐ Enter manually

Azure subscription [ⓘ]

Azure for Students (40ac502f-f518-495f-9cc2-6ffa85309a86)

Azure key vault name *

projectkv1


[Edit key vault](#)

Authentication method

System-assigned managed identity

Managed identity name: **ADFprojectsxp**

Managed identity object ID: **fa5fcd8c-347b-45b0-8192-b76517cec089**


Grant Data Factory service managed identity access to your Azure Key Vault. [Learn more](#) 


Test connection

☒ To linked service ☐ To secret

Create

Cancel

 Connection successful

 Test connection

Secret name:

Go back to key vault, in the control access(IAM) add new role as a key vault secret user, select a member for managed identity

Add role assignment ...

Role **Members** ^{*} Conditions [Review + assign](#)

Selected role Key Vault Secrets User

Assign access to ☐ User, group, or service principal
☒ Managed identity

Members [+ Select members](#)

Name	Object ID	Type
No members selected		


Description

Optional

[Review + assign](#)

[Previous](#)

[Next](#)

 Some results might be hidden due to your ABAC

Subscription *

Azure for Students

Managed identity

Data factory (V2) (1)

Select [ⓘ]

Search by name

Selected members:



ADFprojectsxp
/subscriptions/40ac502f-f518-495f-9cc2-6i

Select


Close


New linked service


 SQL server [Learn more](#) 


Password

Azure Key Vault


AKV linked service * 


AzureKeyVault1 

Secret name * 


PASSWORD 

☐ Edit


Secret version 


Latest version 


☐ Edit

Always encrypted 

☐

Encrypt 


Optional 

Trust server certificate 

☐

Host name in certificate


Additional connection properties


 New

Annotations

Create

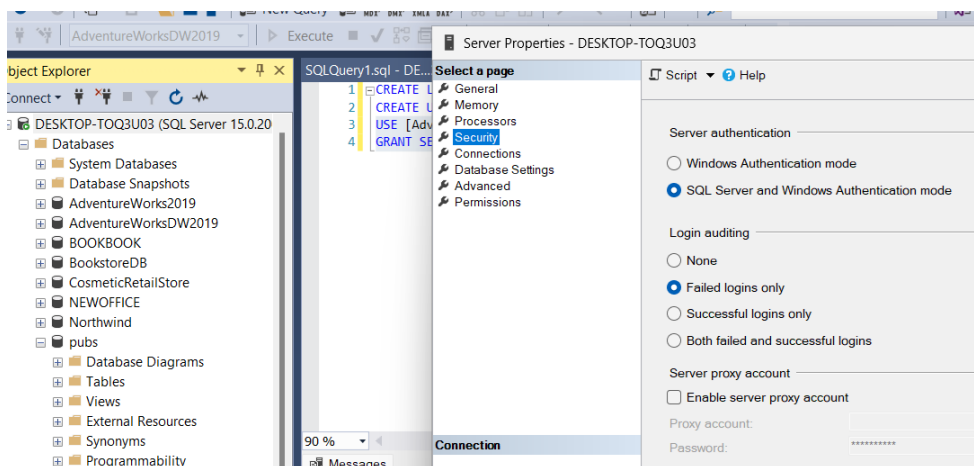
Cancel

 Connection success

 Test connect

Grant Permissions:

Go to sql on premise and properties of sql server name and select security:




Grant the user access to the AdventureWorksDW2019 database:

```
USE [AdventureWorksDW2019];
```

```
GRANT SELECT ON SCHEMA:::SCHEMANAME TO USERNAME
```

Set properties

Name
sqlserveron_prem

Linked service *
SqlServer_Is 




Connect via integration runtime * ⓘ
 integrationRuntime 

Table name
Select... 
☐ Enter manually

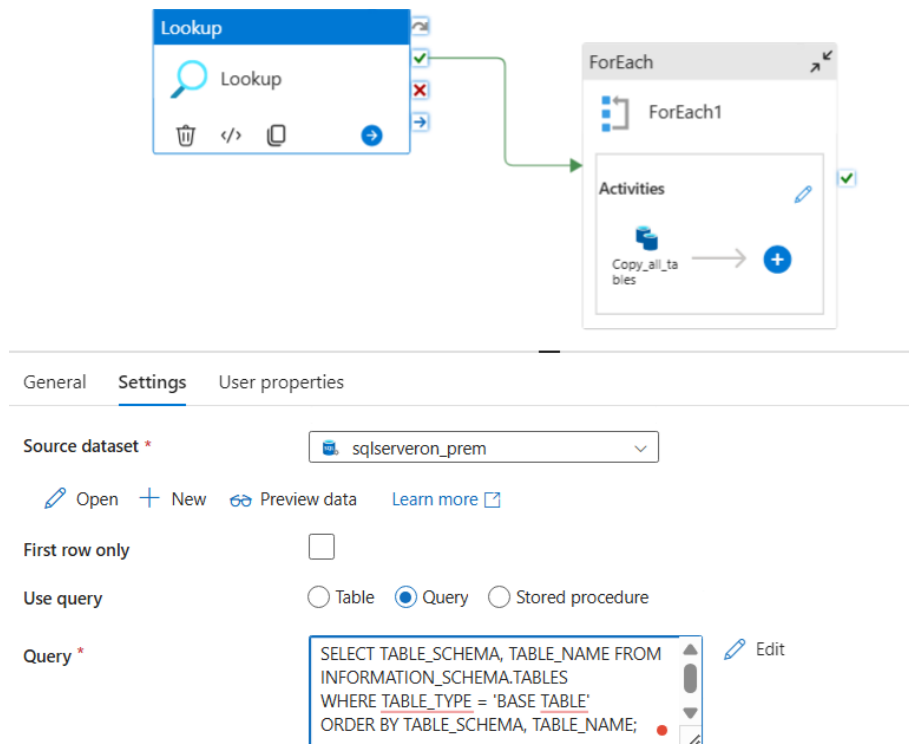
Import schema
☐ From connection/store ☒ None

> Advanced

Go back to lookup and select query

Query:

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM INFORMATION_SCHEMA.TABLES  
WHERE TABLE_TYPE = 'BASE TABLE'  
ORDER BY TABLE_SCHEMA, TABLE_NAME;
```



This query dynamically selects data from each table in the dbo schema.

[For make validate the look up activity you can first do look up and then connect to for each loop.]

Dataset: SQLServer on-prem

Query:

```
@{concat('select * from ',item().TABLE_SCHEMA,'.',item().TABLE_NAME)}
```

The screenshot displays the Azure Data Factory (ADF) interface. On the left, a pipeline is shown with a 'Lookup' activity connected to a 'ForEach' loop. The 'ForEach' loop contains a 'Copy' activity. The 'Copy' activity's 'Source' field is set to a dynamic SQL query: `@concat('select * from ', item().TABLE_SCHEMA, '.', item().TABLE_NAME)}`. The 'Sink' field is currently empty. The 'Dynamic content' pane on the right shows the 'ForEach' iterator and its 'Current item' variable, `ForEach1`.

Add dynamic content below using any combination of expressions, functions and system variables.

```
@concat('select * from ', item().TABLE_SCHEMA, '.', item().TABLE_NAME)}
```

Clear contents

ForEach iterator Activity outputs Parameters System variables ...

Search



ForEach1
Current item

Sink:

Configure the Sink:

- **Dataset:** Create a new dataset for Azure Data Lake.
- **File Path:** Set the path to the container.


New linked service

 Azure Data Lake Storage Gen2 [Learn more](#) 

Name *

AzureDataLakeStorage_Is

Description

Connect via integration runtime * 


 AutoResolveIntegrationRuntime 


Authentication type

Account key 

Account selection method 

☒ From Azure subscription ☐ Enter manually

Azure subscription 

Azure for Students (40ac502f-f518-495f-9cc2-6ffa85309a86) 

Storage account name *

projectboxsg  


Test connection 

☒ To linked service ☐ To file path


Annotations


 New

> Parameters

> Advanced 

Create Cancel

 Connection successful

 Test connection

Set properties

Name

Parquet1

Linked service *

AzureDataLakeStorage1

File path

bronz

/ Directory

/ File name

Import schema

☒ From connection/store ☐ From sample file ☐ None

> Advanced

Parameters:

Sink>>open>>parameters

schemaname: @item().TABLE_SCHEMA

tablename: @item().TABLE_NAME

☒ Validate ☒ Cancel options ☒ Add trigger

The screenshot shows the Azure Data Factory designer interface. A 'Lookup' activity is connected to a 'ForEach' loop. The 'ForEach' loop contains a 'Copy' activity. Below the canvas, the 'Sink' tab is selected, showing the 'Sink dataset' as 'Parquet1'. The 'Dataset properties' section is expanded, showing a table with the following data:

Name	Value	Type
schemaname	@item().TABLE_SCHEMA	string
tablename	@item().TABLE_NAME	string

File Path: Dynamically generated as described above.

4. Key Parameters and Dynamic Content

Dynamic Content in Source Query:

- The source query dynamically selects data from each table using the schema and table names from the ForEach loop:

```
SELECT * FROM @item().schema_name.@item().table_name
```

Dynamic Content in Sink File Path:

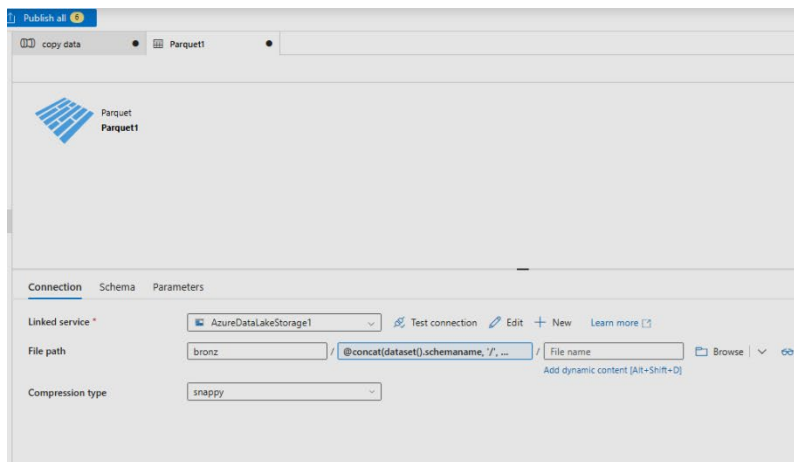
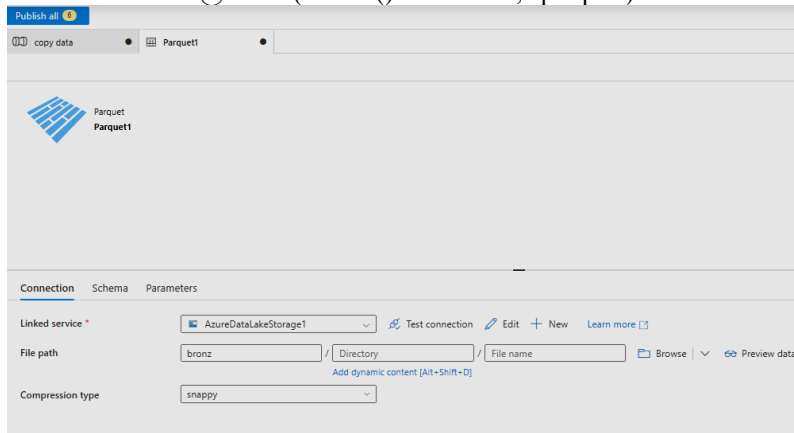
- The file path and file name in the sink are dynamically generated using the following expressions:

- **Directory Path:**

`@concat(dataset().schemaName, '/', dataset().tableName)`

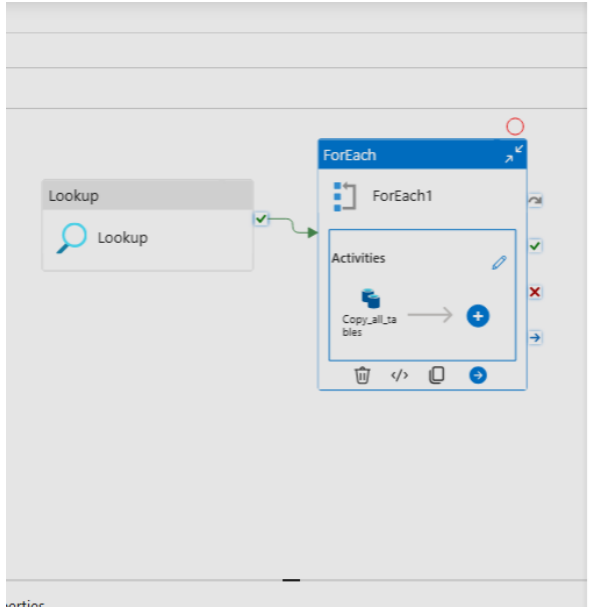
- **File Name:**

`@concat(dataset().tableName, '.parquet')`



For Each Activity:

- Iterates over the list of tables returned by the Lookup activity.
- **Items:** `@activity('Lookup1').output.value`



Pipeline expression builder

Add dynamic content below using any combination of **expressions**, **functions** and **system variables**.

```
@activity('Lookup').output.value
```

[Clear contents](#)

Activity outputs
Parameters
System variables
Functions
Variables

Copy_all_tables
Copy_all_tables activity output

Lookup
Lookup activity output

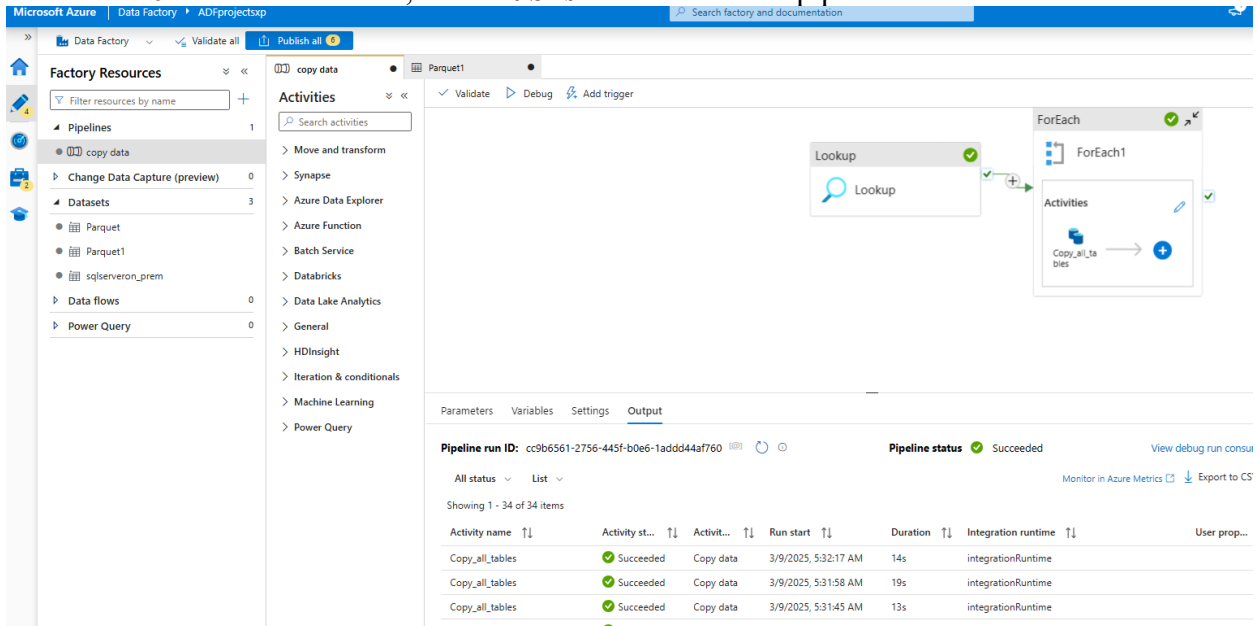
Lookup count
Count of the rows

Lookup value array

Step 3: Setting Up Azure Data Factory Pipeline

2. Debug the Pipeline:

- Click **Debug** to test the pipeline.
- Once successful, click **Publish All** to save the pipeline.

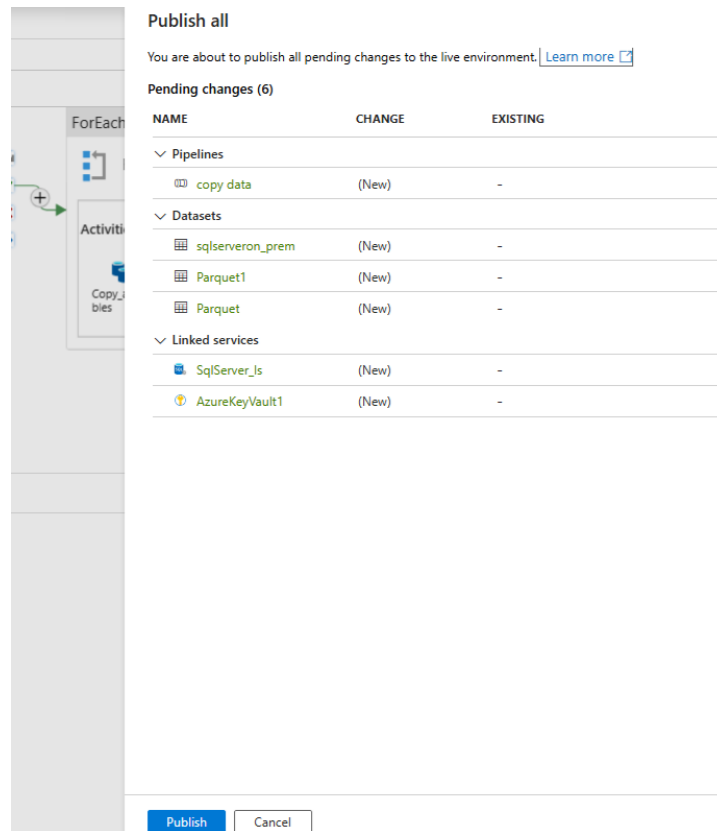


The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' pane lists various resources including Pipelines, Datasets, and Data flows. The main area displays the 'copy data' pipeline, which includes a 'Lookup' activity and a 'ForEach' loop containing a 'Copy' activity. The 'Output' tab is selected, showing the results of a pipeline run.

Pipeline run ID: cc9b6561-2756-445f-b0e6-1addd44af760 **Pipeline status:** Succeeded

Showing 1 - 34 of 34 items

Activity name	Activity status	Activity name	Run start	Duration	Integration runtime	User properties
Copy_all_tables	Succeeded	Copy data	3/9/2025, 5:32:17 AM	14s	integrationRuntime	
Copy_all_tables	Succeeded	Copy data	3/9/2025, 5:31:58 AM	19s	integrationRuntime	
Copy_all_tables	Succeeded	Copy data	3/9/2025, 5:31:45 AM	13s	integrationRuntime	



Step 4: Automating the Pipeline

4.1 Create a Trigger

1. **Add a Trigger:**
 - In the Data Factory UI, go to **Author > Triggers**.
 - Click **New** and choose **Schedule**.
 - Set the trigger to run daily.
2. **Link the Trigger to the Pipeline:**
 - Select the CopyData pipeline.
 - Save and publish the trigger.

Step 5: Monitoring and Troubleshooting

1. **Monitor the Pipeline:**
 - Go to **Monitor** in the Data Factory UI.
 - Check the status of pipeline runs and troubleshoot any errors.
2. **Check Data in Azure Data Lake:**
 - Go to your storage account and check the container to ensure data is being copied.

Conclusion

This tutorial walks you through setting up a data pipeline using SQL Server, Azure Data Factory, and other Azure services. You've learned how to:

- Set up a Resource Group, Storage Account, and Key Vault in Azure.
- Create and configure a Data Factory pipeline to copy data from an on-prem SQL Server to Azure Data Lake.
- Automate the pipeline using triggers.

Key Points to Note

- **Lookup Activity:** Used to fetch the list of tables from the SQL Server database.
- **For Each Activity:** Iterates over the list of tables and performs the copy operation for each table.
- **Dynamic Content:** Used extensively to dynamically generate queries, file paths, and file names based on the schema and table names.
- **Parameters:** schemaName and tableName are used to dynamically configure the sink dataset.
- **Output:** All tables are copied from the SQL Server database to the Azure Data Lake in Parquet format

This configuration ensures that the pipeline is fully automated and can handle any number of tables in the dbo schema. The use of dynamic content and parameters makes the pipeline flexible and scalable.