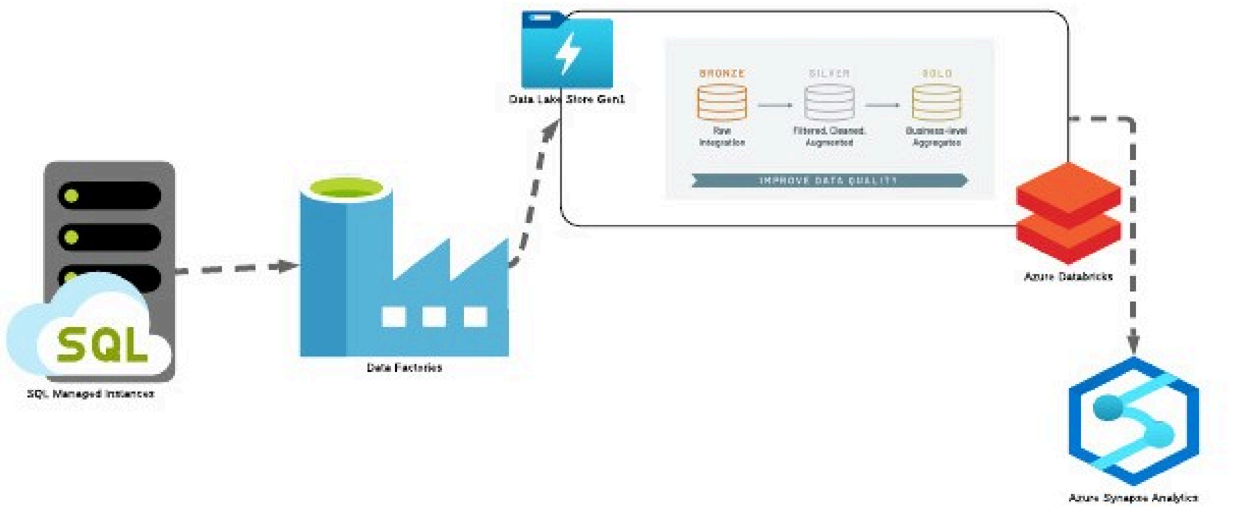
End to End Data Migration



Hey guys this is the project path we will be following

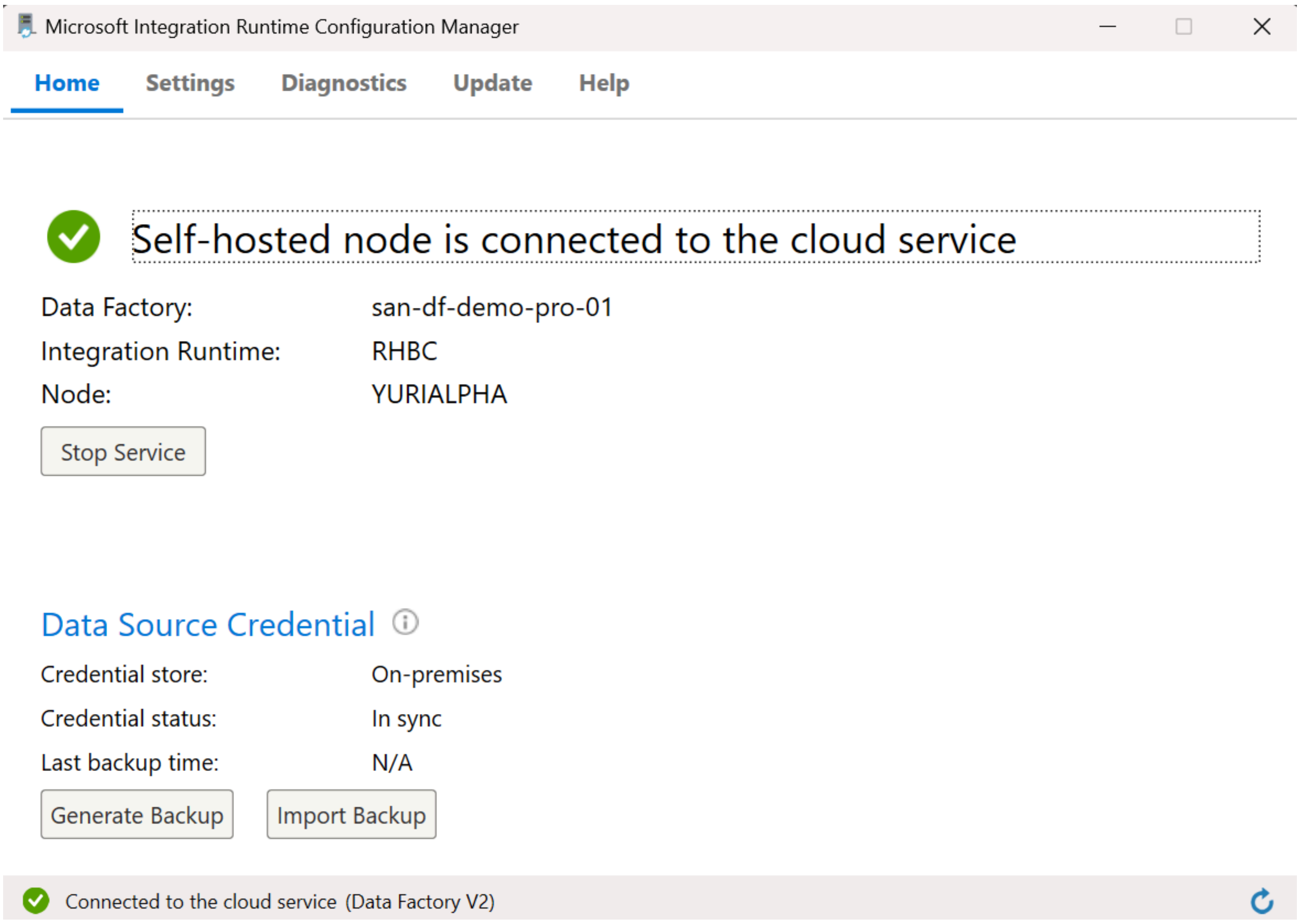
First Create these:

1. Resource Group
   a. Azure DataBricks
   b. Azure Synapse
   c. Azure Key vault
   d. Azure Data Factory
   e. Azure Storage account
2. Prep up the SQL SSMS:
   a. Backup the database given
   b. Creating the role and giving access
   c. Storing all the information sanath has stored 'password', and username 'sanath'

Now for the first part we are creating the **Integration run time** inside the Azure Data Factory

We can go to Manage in ADF and then Click on Integration Run Time and follow through clicking next and download the express download and finish with your integration.

It should show like this:



Microsoft Integration Runtime Configuration Manager

Home | Settings | Diagnostics | Update | Help

Self-hosted node is connected to the cloud service

Data Factory:	san-df-demo-pro-01
Integration Runtime:	RHBC
Node:	YURIALPHA

Stop Service

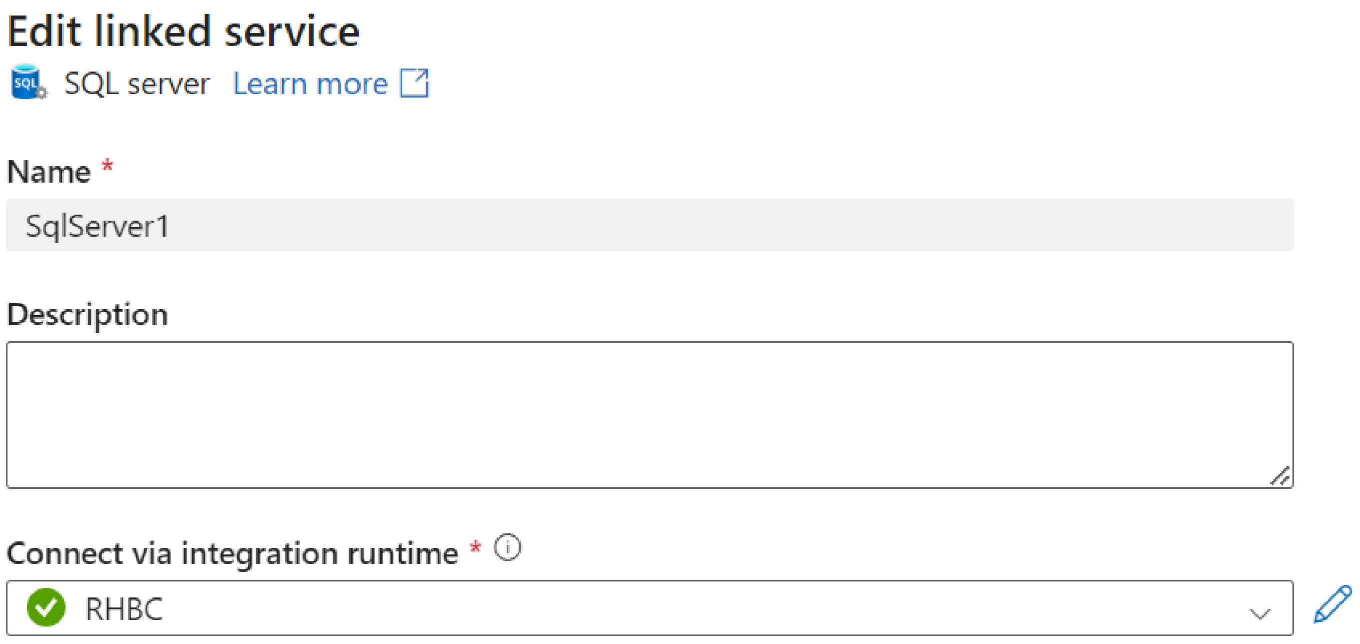
Data Source Credential

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

Generate Backup | Import Backup

Connected to the cloud service (Data Factory V2)

Now Moving to next part we will create a pipeline and you can name anything
Now in the activities tab search **Lookup** and drag that to dashboard.
You can name anything you want and you want to create dataset click on pencil icon in the settings tab next to dataset and name that and create a linked service which should look like this



Edit linked service

SQL server Learn more

Name *

SqlServer1

Description

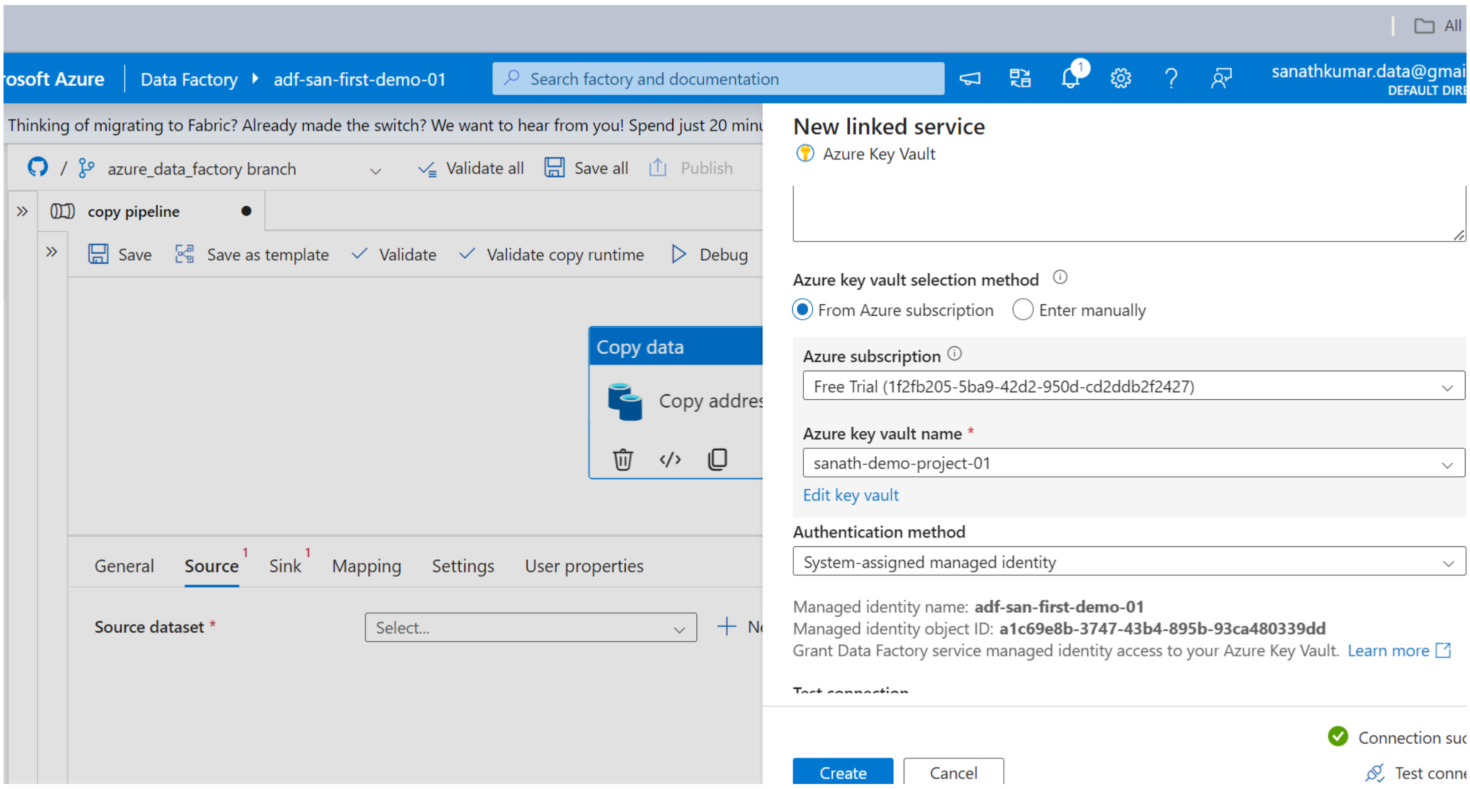
Connect via integration runtime *

RHBC

then select your created linked integration runtime.

Now i want you to go the key credentials you created and register two new secrets that would be username and password with the a role as you have created inside the SSMS and in the same page go to access and management(IAM) and click on add then select add role assignment and then give **secret admin** role and save for your user.

Now come back to Data Factory and now continue with your creation of linked service and for credentials use azure key vault and continue creating what it ask for as it asks for another new linked service connecting ADF to Azure Key Vault so now you have access to your credentials now drop down and select and test connection.



New linked service
Azure Key Vault

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

Azure subscription: Free Trial (1f2fb205-5ba9-42d2-950d-cd2ddb2f2427)

Azure key vault name *: sanath-demo-project-01

Edit key vault

Authentication method: System-assigned managed identity

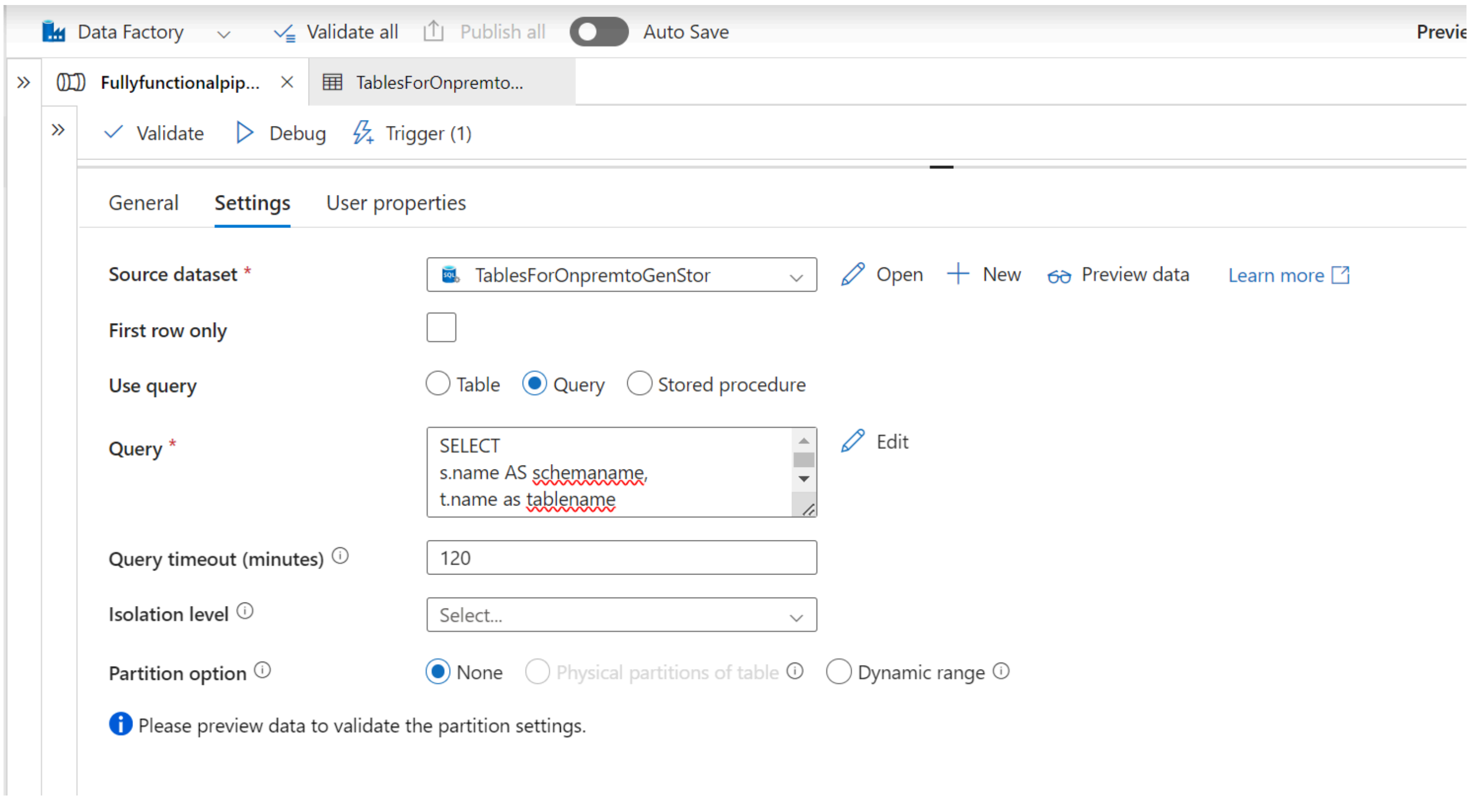
Managed identity name: **adf-san-first-demo-01**
Managed identity object ID: **a1c69e8b-3747-43b4-895b-93ca480339dd**
Grant Data Factory service managed identity access to your Azure Key Vault. Learn more

Connection suc

Create Cancel Test conne

You should see like this.

Now click on create and then you would be redirected to this page.



Source dataset *: TablesForOnpremtoGenStor

First row only: ☐

Use query: Table [✓ Query] Stored procedure

Query *:
```
SELECT
s.name AS schemaname,
t.name as tablename
```

Query timeout (minutes): 120

Isolation level: Select...

Partition option: [✓ None] Physical partitions of table Dynamic range

Please preview data to validate the partition settings.

now click on query and then write this script.

```
SELECT
s.name AS schemaname,
t.name as tablename
from sys.tables as t
inner join sys.schemas s
```

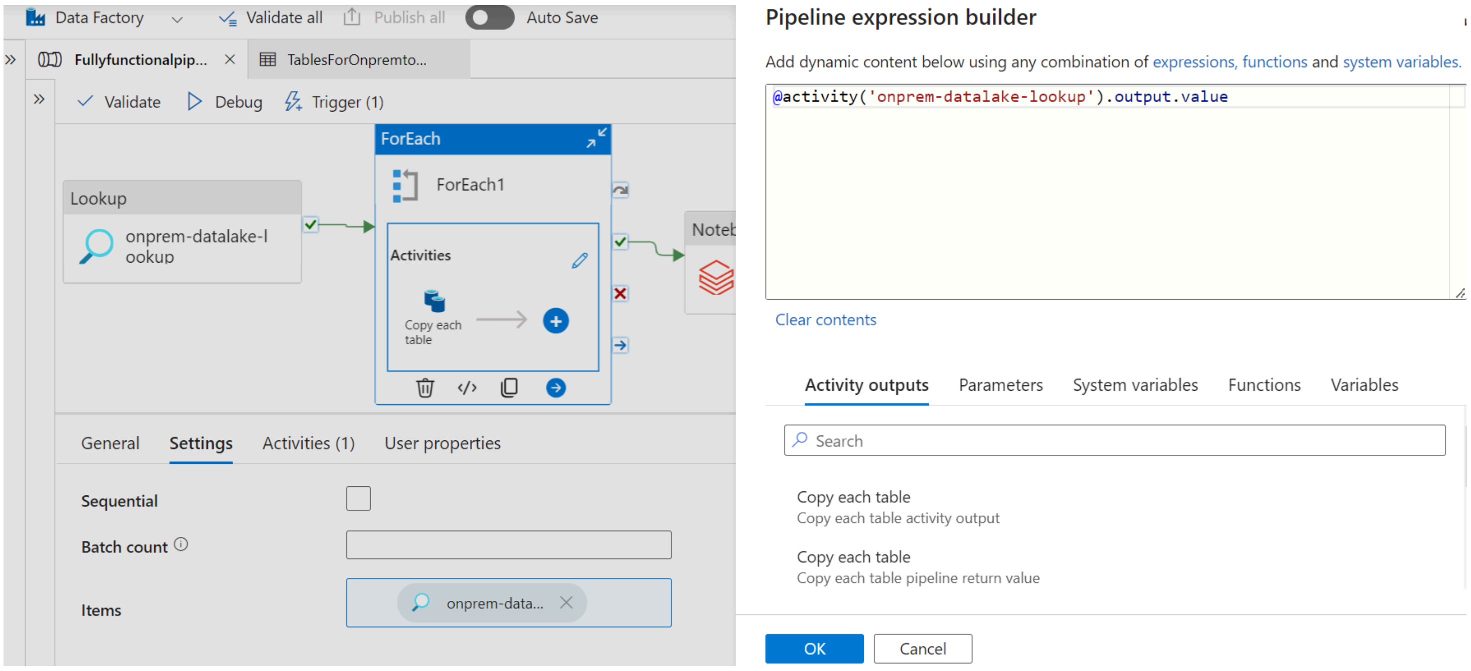
```
on t.schema_id = s.schema_id
where s.name = 'Sales'
```

What this does is get all the **Sales Tables** into **Azure From On-Prem**.

Data Ingestion

Now we have all the table name and schema name into the lookup now what we will do is create a for-each activity to grab all the table names and schema name and create a bronze layer container and ingest all those data in the format of 'bronze/ Sales/ Table_name/ Table_name.parquet'.

We will have bronze container created inside the storage account.
Now drag the for-each activity and click on settings and items and then you will click dynamic functions and enter this code to get all the items from lookup to here.



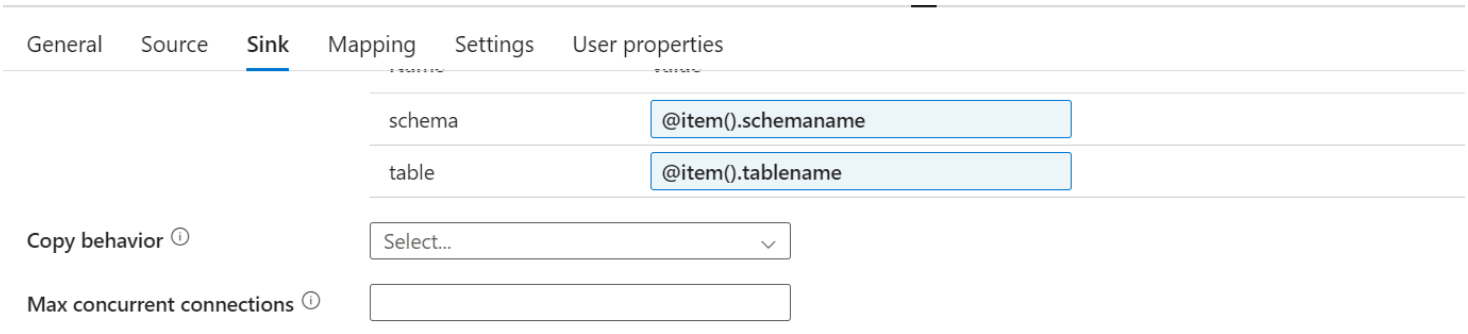
```
@activity('onprem-datalake-lookup').output.value
```

Now go to activities and create Copy activity and then in source click on data source and search for sql server and then click on ok and don't give any files or folder, just create it.
Now come to the Copy Table and now in the source click on query and enter this query using dynamic functions.

```
SELECT *

FROM @{concat(item().schemaname, '.', item().tablename)}
```

This will give the guide to run this on your lookup to get these items which will be in your json format.

Now come to main copy table and now click on sink to push the data from on-prem to DataLakeGen2. Here create parameters schema and table, before that new dataset and i want datalakegen2 and then format is parquet and do the usual procedure creating linked service and everything now come to sink page and now create new 2 parameters and give values as follows.



schema	@item().schemaname
table	@item().tablename
Copy behavior	Select...
Max concurrent connections	

and again go back to your dataset on these and then we will be storing in the format as said above and code would be

```
@{concat(dataset().table, '/',  dataset().schema)}
```

For directory and for file name

```
{concat(dataset().table, '.parquet')}
```

Now publish all and trigger once you would be ingested data once.

Data Transformation

Here we will be using DataBricks for transformation, before that create two new containers silver and gold to store our transformed files.

Now open DataBricks from you resource group and then open workspace and there we will start below...

First create a cluster to run our jobs, basics would be enough and now go to workspaces click on shared procedure and now click three notebooks

1. Mount Storage
2. Bronze to Silver transform
3. Silver to Gold Transform

Now i will be given code for each of the notebooks in an order

```
configs = {

  "fs.azure.account.auth.type": "CustomAccessToken",

  "fs.azure.account.custom.token.provider.class":
spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")

}
```

```
# Optionally, you can add <directory-name> to the source URI of your mount point.

dbutils.fs.mount(

  source = "abfss://bronze@snaazsynapsedemo2024.dfs.core.windows.net/",

  mount_point = "/mnt/bronze",

  extra_configs = configs)
#------------------------------------------------------------

configs = {

  "fs.azure.account.auth.type": "CustomAccessToken",

  "fs.azure.account.custom.token.provider.class":
spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")

}



# Optionally, you can add <directory-name> to the source URI of your mount point.

dbutils.fs.mount(

  source = "abfss://silver@snaazsynapsedemo2024.dfs.core.windows.net/",

  mount_point = "/mnt/silver",

  extra_configs = configs)

#----------------------------------------------------------------
configs = {

  "fs.azure.account.auth.type": "CustomAccessToken",

  "fs.azure.account.custom.token.provider.class":
spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")

}



# Optionally, you can add <directory-name> to the source URI of your mount point.

dbutils.fs.mount(

  source = "abfss://gold@snaazsynapsedemo2024.dfs.core.windows.net/",

  mount_point = "/mnt/gold",

  extra_configs = configs)
```

Now run these cells and u will have resource connected to your DataLakeGenV2

Now for Bronze to Silver

```
# Import necessary functions
from pyspark.sql.functions import from_utc_timestamp, date_format
from pyspark.sql.types import TimestampType

# Initialize an empty list to store table names
tables = []

# List files in the bronze layer's "Sales" directory and extract table names
for i in dbutils.fs.ls("/mnt/bronze/Sales/"):
    tables.append(i.name.split('/')[0])

# Iterate over each table name
for i in tables:
    # Construct the path to the parquet file
    path = '/mnt/bronze/Sales/' + i + '/' + i + '.parquet'

    # Load the parquet file into a DataFrame
    df = spark.read.format('parquet').load(path)

    # Get the list of columns in the DataFrame
    columns = df.columns

    # Iterate over each column to check and format date columns
    for col in columns:
        if "Date" in col or "date" in col:
            # Convert date columns to 'yyyy-MM-dd' format
            df = df.withColumn(col,
date_format(from_utc_timestamp(df[col].cast(TimestampType()), "UTC"), "yyyy-MM-dd"))

    # Define the output path for the silver layer
    output_path = '/mnt/silver/Sales/' + i + '/'

    # Write the DataFrame to the silver layer in Delta format
    df.write.format('delta').mode('overwrite').save(output_path)

    # Display the DataFrame
    display(df)
```

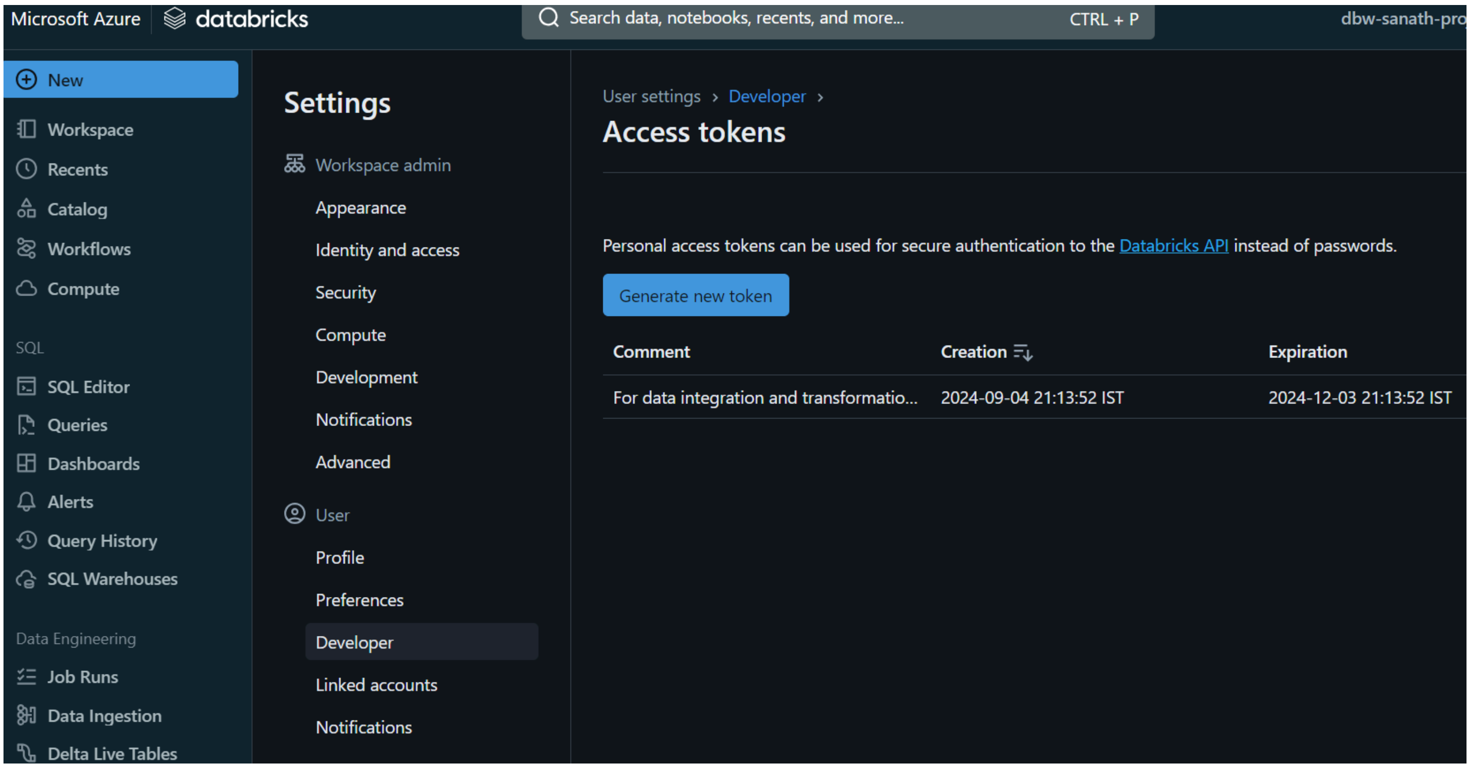
Silver to Gold

```
table_name = []
for i in dbutils.fs.ls('mnt/silver/Sales/'):
    table_name.append(i.name.split('/')[0])

for name in table_name:
    path = '/mnt/silver/Sales/' + name
    print(path)
    df = spark.read.format('delta').load(path)
    column_names = df.columns
    for old_col_name in column_names:
        new_col_name = "".join(["_" + char if char.isupper() and not old_col_name[i -
1].isupper() else char
                                for i, char in enumerate(old_col_name)]).lstrip("_")
        df = df.withColumnRenamed(old_col_name, new_col_name)
    output_path = '/mnt/gold/Sales/' + name + '/'
    df.write.format('delta').mode("overwrite").save(output_path)
display(df)
```

and also copy the access token from setting-profile-user-access token-create-copy and then store that inside the key vault.

Now come to ADF and click on Your pipeline and create a activity of notebook of 2 and then name them bronze to silver and silver to gold and connect success to each after.

Now create and connect your workspace and then in your access token connect your azure vault and give that you have created at that time of access token.

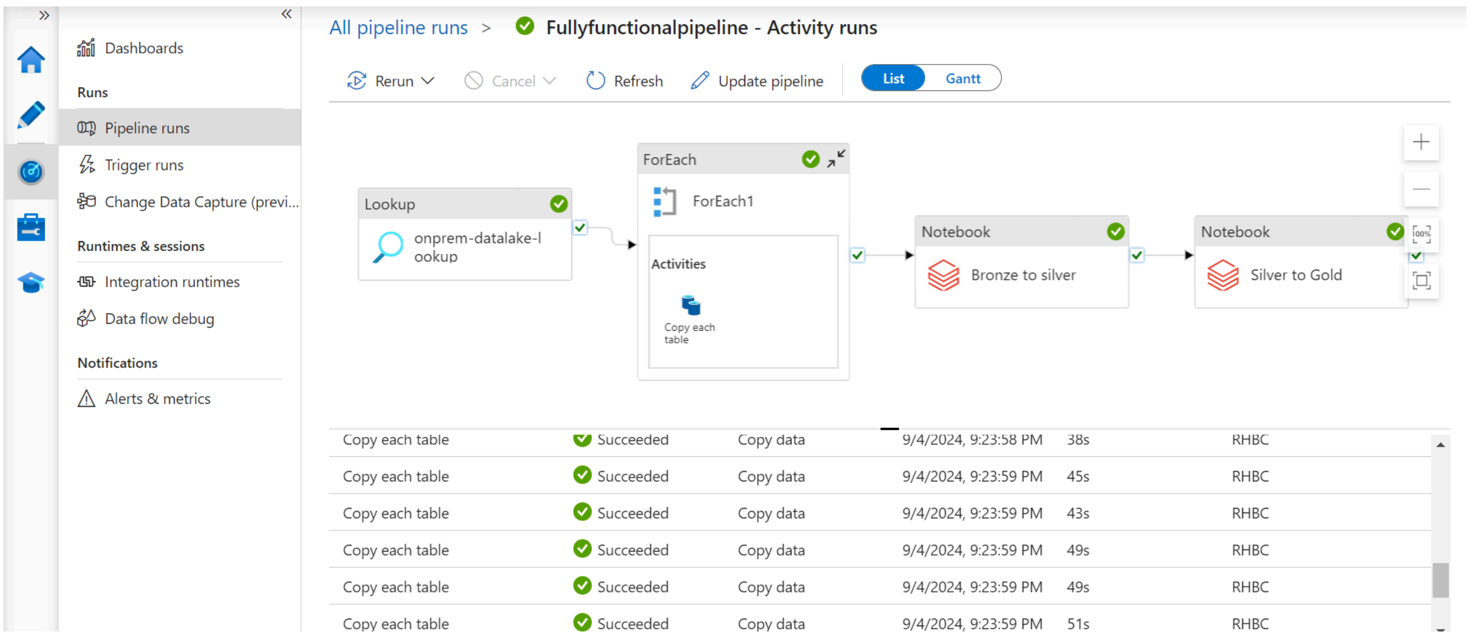


Access tokens

Personal access tokens can be used for secure authentication to the Databricks API instead of passwords.

Comment	Creation	Expiration
For data integration and transformatio...	2024-09-04 21:13:52 IST	2024-12-03 21:13:52 IST

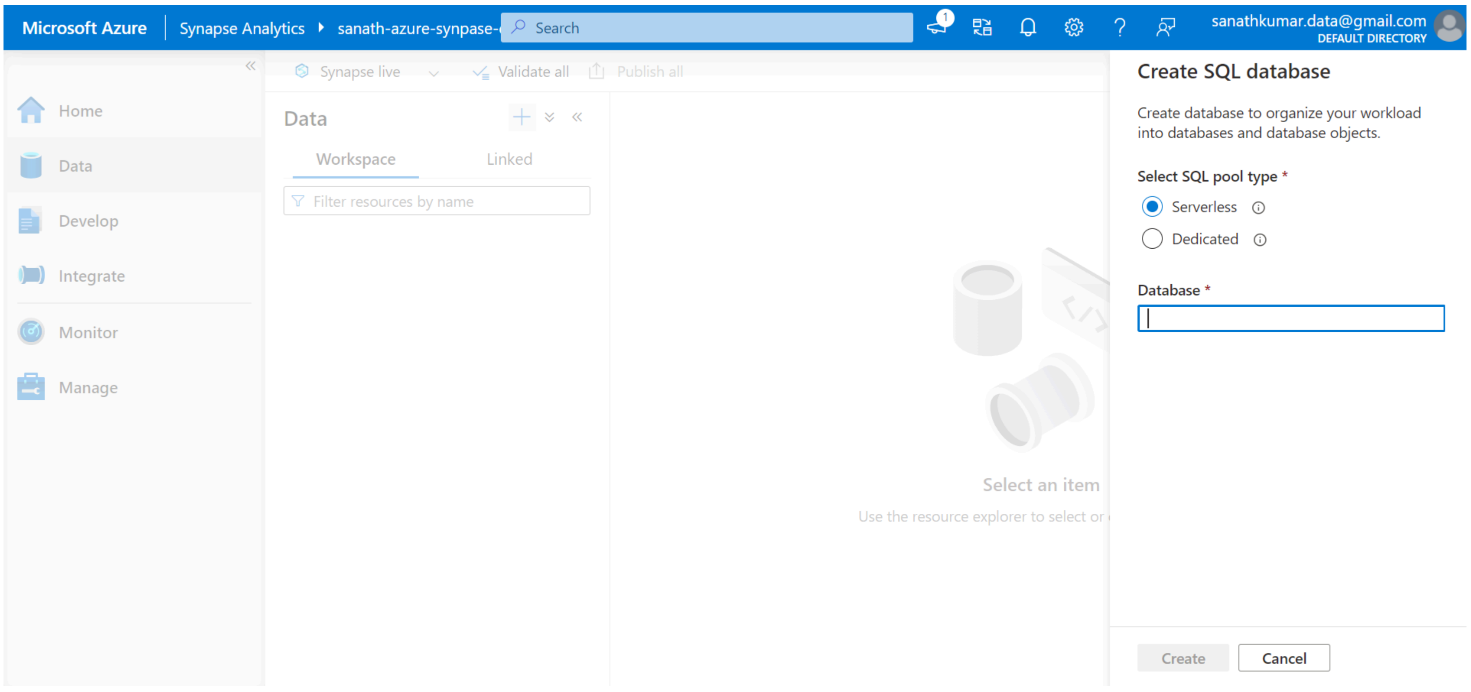
After doing for both publish all and run manual trigger once. Your transformation is ready. Now we have finished our work in ADF as of now.



Copy each table	✓ Succeeded	Copy data	9/4/2024, 9:23:58 PM	38s	RHBC
Copy each table	✓ Succeeded	Copy data	9/4/2024, 9:23:59 PM	45s	RHBC
Copy each table	✓ Succeeded	Copy data	9/4/2024, 9:23:59 PM	43s	RHBC
Copy each table	✓ Succeeded	Copy data	9/4/2024, 9:23:59 PM	49s	RHBC
Copy each table	✓ Succeeded	Copy data	9/4/2024, 9:23:59 PM	49s	RHBC
Copy each table	✓ Succeeded	Copy data	9/4/2024, 9:23:59 PM	51s	RHBC

Data Loading

Now Open our azure synapse and now create database in serverless database named gold_db as we will be using this to create our pipeline to send our views to external BI tools.



Create SQL database

Create database to organize your workload into databases and database objects.

Select SQL pool type *

[✓ Serverless]

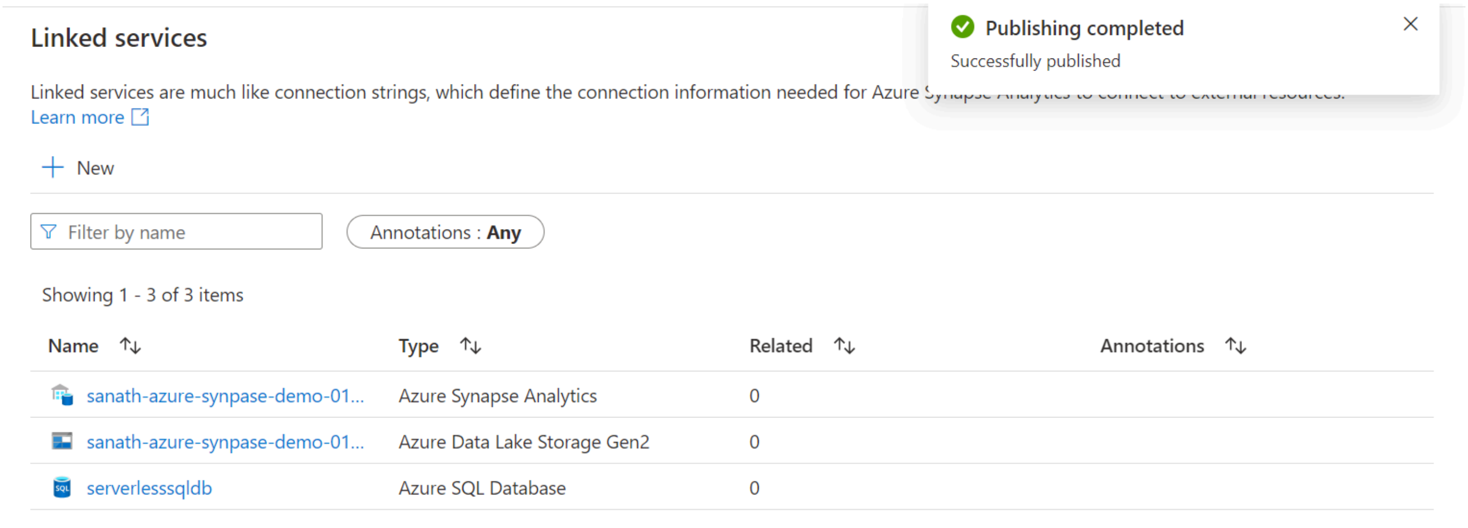
Dedicated

Database *

Create

Cancel

Now we will be creating our linked service for our database



Linked services

Publishing completed
Successfully published

Linked services are much like connection strings, which define the connection information needed for Azure Synapse Analytics to connect to external resources.
Learn more

Showing 1 - 3 of 3 items

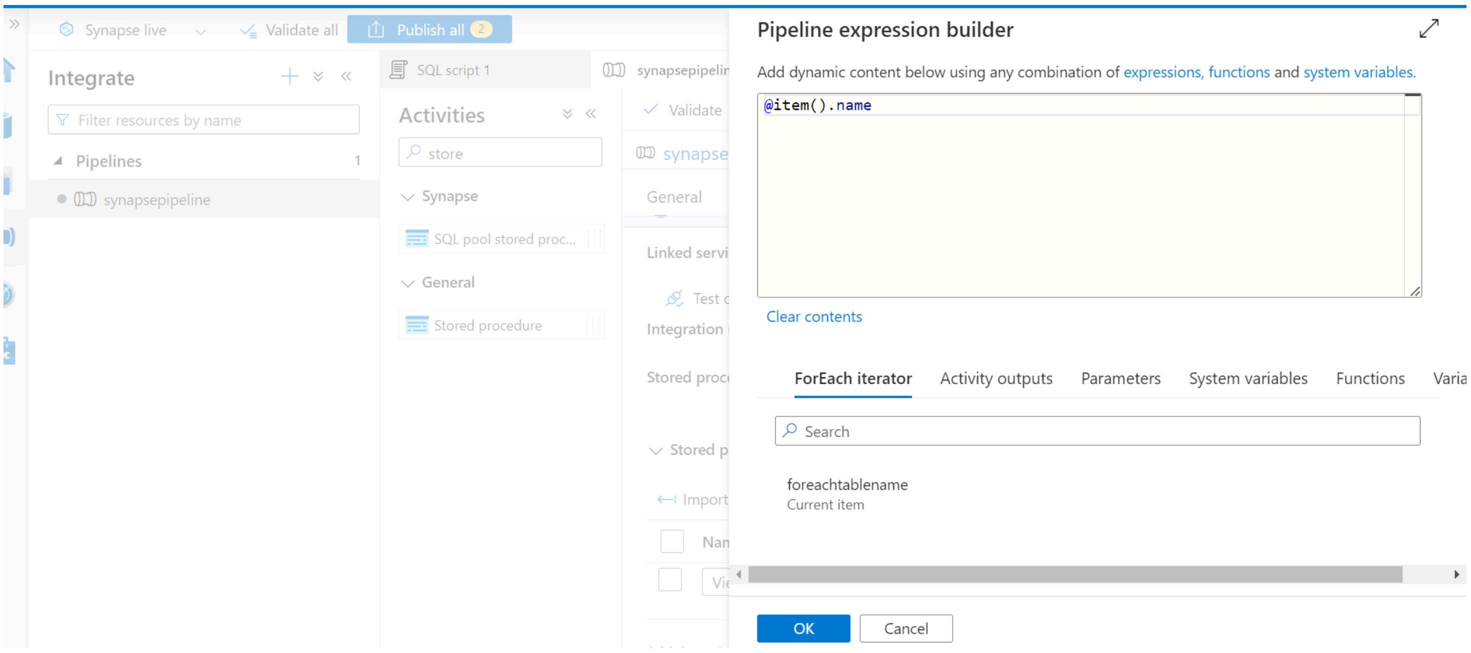
Name	Type	Related	Annotations
sanath-azure-synpase-demo-01...	Azure Synapse Analytics	0	
sanath-azure-synpase-demo-01...	Azure Data Lake Storage Gen2	0	
serverlesssqldb	Azure SQL Database	0	

Now we will be creating a sql script to run to create view in Azure Synapse

```
USE gold_db
GO

CREATE OR ALTER PROC CreateSQLServerlessView_gold @ViewName nvarchar(100)
AS
BEGIN
    DECLARE @statement VARCHAR(MAX)

    SET @statement = N'CREATE OR ALTER VIEW ' + @ViewName + ' AS
    SELECT *
    FROM
    OPENROWSET(
        BULK ''https://mrkdatalakegen2.dfs.core.windows.net/gold/Sales/' + @ViewName
+ '/'',
        FORMAT = ''DELTA''
    ) as [result]'

    EXEC (@statement)
END
GO
```

This will be our SQL Script

Next is to create our pipeline now go to integrate and create activity and search for Stored Procedure and use that linked service we created and SQL scripted view name well be showing for dataset. drag for-each to get name and we will be passing the value parameters as this



Pipeline expression builder

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

```
@item().name
```

Clear contents

ForEach iterator | Activity outputs | Parameters | System variables | Functions | Varia

Search

foreachtablename
Current item

OK | Cancel

Now this pipeline will automatically trigger when the changes in layers happens so no need to automate this and now we can connect this with our end point key to any external source and grab transformed views and do there analysis.

Now open data factory to create a new set of triggers and create a trigger at 12Am to run this job, after 12 AM each day you can just refresh your BI tool your data will be updated.

Estimated Costs in INR

1. **Azure Data Factory (ADF)**
   - **Pipeline Activity Runs**: $2.25 * 83 = ₹186.75
2. **Azure DataBricks**
   - **Compute Costs**: $24.00 * 83 = ₹1,992.00
3. **Azure Storage (Blob Storage)**
   - **Storage Costs**: $0.10 * 83 = ₹8.30
4. **Azure Synapse Analytics**
   - **Serverless SQL Pools**: $0.10 * 83 = ₹8.30

Total Estimated Cost in INR: ₹186.75 + ₹1,992.00 + ₹8.30 + ₹8.30 = ₹2,195.35