Disclaimer: **These slides are copyrighted and strictly for personal use only**

- This document is reserved for people enrolled into the [Ultimate Big Data Masters Program (Cloud Focused) by Sumit Sir]

- **Please do not share this document,** it is intended for personal use and exam preparation only, thank you.

- If you've obtained these slides for free on a website that is not the course's website, please reach out to legal@trendytech.in . Thanks!

- All the Best and Happy Learning!

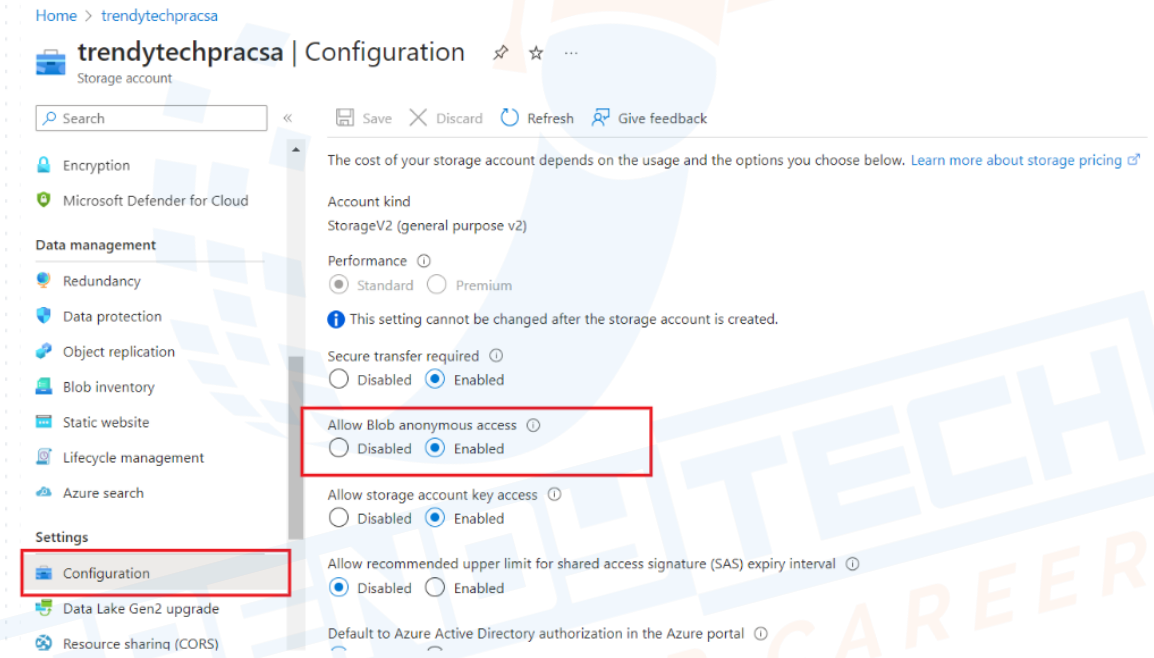
Data Ingestion using HTTP Connector

By **Parameterizing the Services,** data can be ingested with minimum components.

Example Use-case 1 : Ingesting Retail-db datasets from Source and loading to Target

STEPS

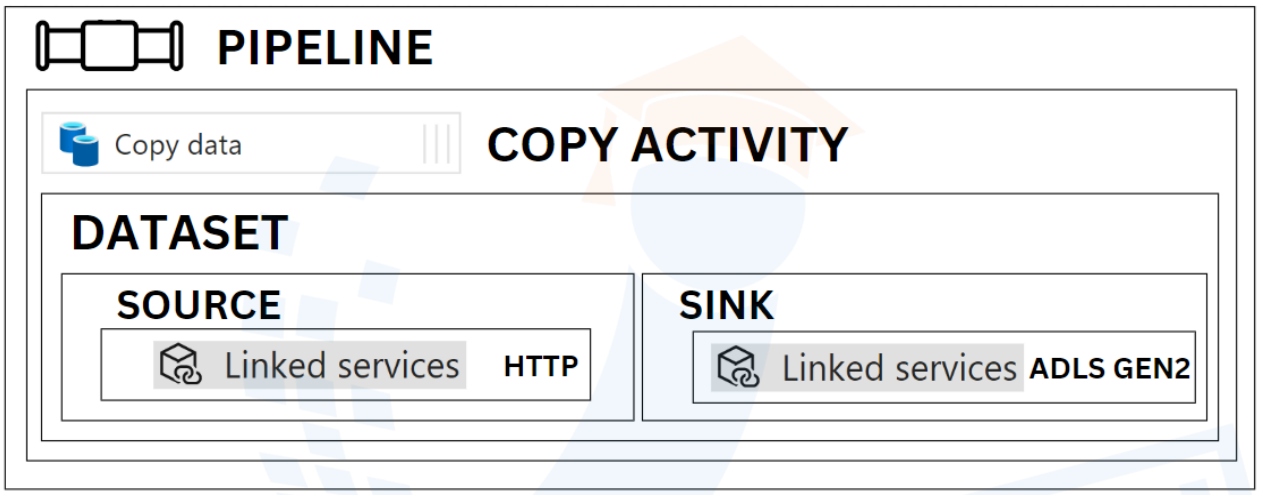
1. Create a Resource Group and pin it to the dashboard
2. Create a Storage Account resource- Blob storage and upload the datasets (products.csv, order.csv and order_item.csv from the retail-db.zip provided in the downloadable section)



Please ensure to enable the Blob anonymous access under configurations after the storage account is deployed.

3. Change the access level of container to - anonymous read access to containers and blobs
4. You can use the respective URLs associated with the datasets as a link to download the datasets from the storage account.
5. Create a Data Factory
6. Create a Linked Service pointing to HTTP with base URL (Source).
7. Create a Linked Service pointing to ADLS Gen2 (Sink).
8. Create a Source Dataset to fetch data from HTTP and mention the relative URL and select the source http linked service.

9. Create a Target Dataset - Choose the format and the file path(target) where the data needs to be copied.
10. Test the connections and publish.
11. Create and Launch Azure Data Factory Studio.
12. Create a pipeline and a copy activity in the Data Factory and Debug the pipeline.
13. Once the pipeline runs successfully, the data will be copied from source http URL to the target path in the ADLS Gen2.



Note : There is a drawback in this process that, say you have to copy 6 different files from different HTTP links, then, all of the above steps have to be repeated.

For every Dataset - a linked service, dataset and a pipeline needs to be created leading to too many components. Say you are required to ingest 100 datasets, then this would lead to the creation of a huge number of components.

Pipeline Parametrization

helps in avoiding unnecessary creation of multiple components.

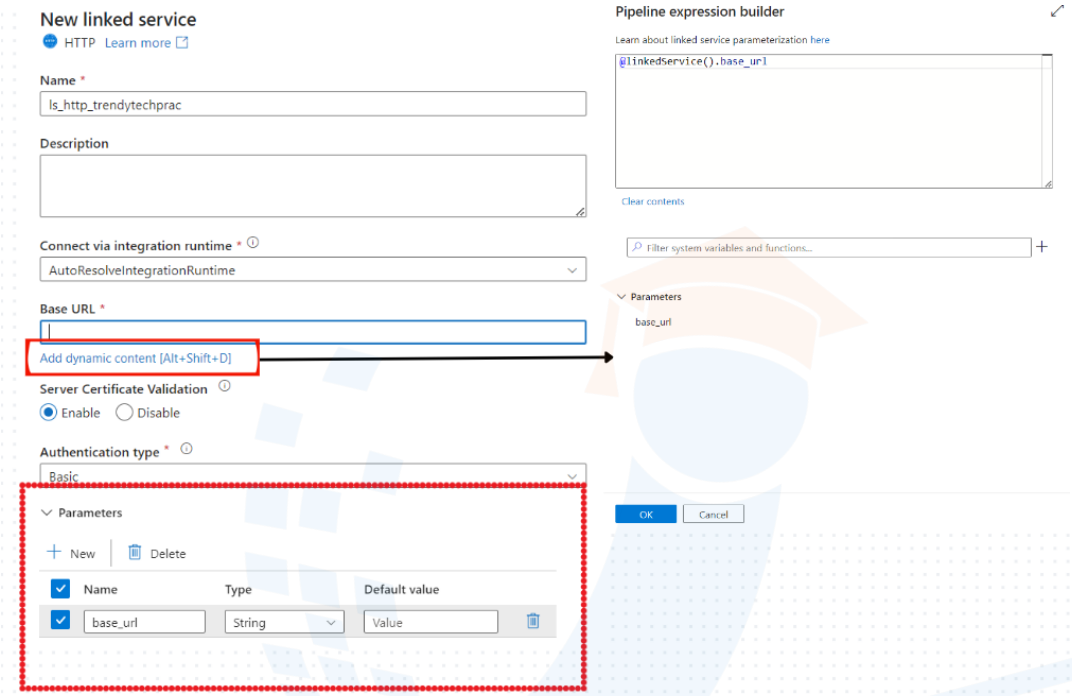
The Base URL provided while creating the Source Linked Service and the Sink target-path filename can be parametrized in-place of hard-coding.

Likewise the Relative URL provided during the Dataset creation can be parametrized rather than hard-coding.

This will allow for providing the URLs dynamically at a later point in time.

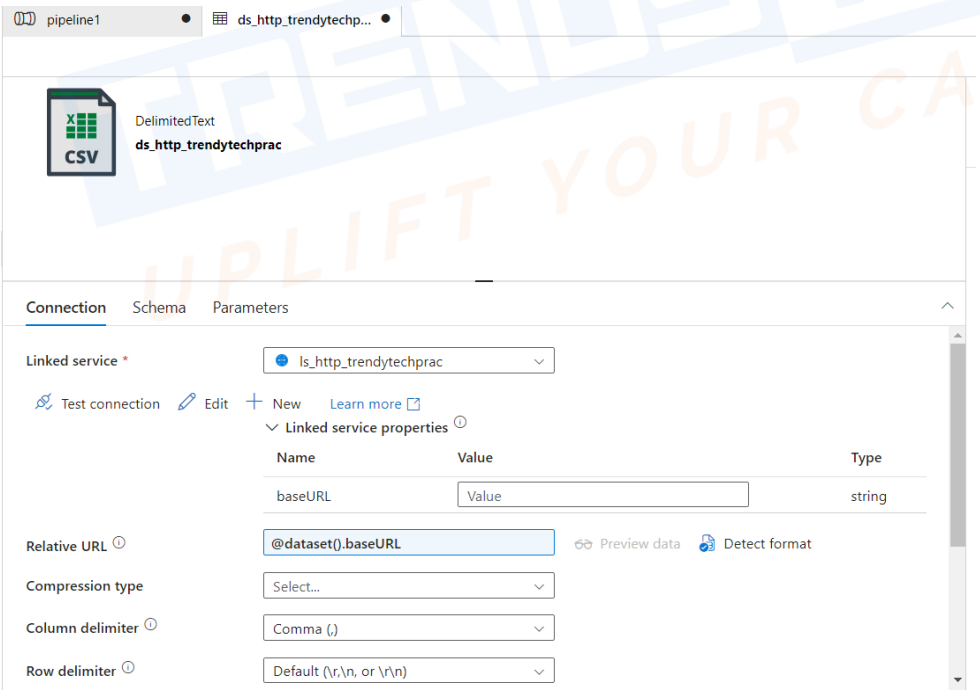
Parameterizing the Source Linked Service

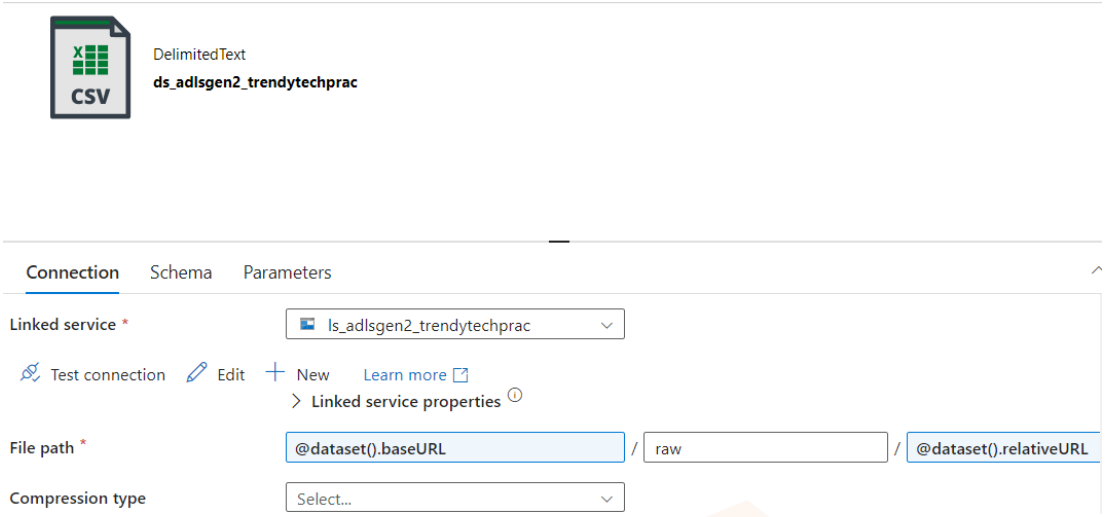
- Parametrize the source linked service by adding a new parameter in the linked service by providing a name to the parameter (Ex: baseUrl)
- In the Base url field, choose the "**Add Dynamic Content**" option and select the newly created parameter baseUrl.



Parameterizing the Sink Linked Service

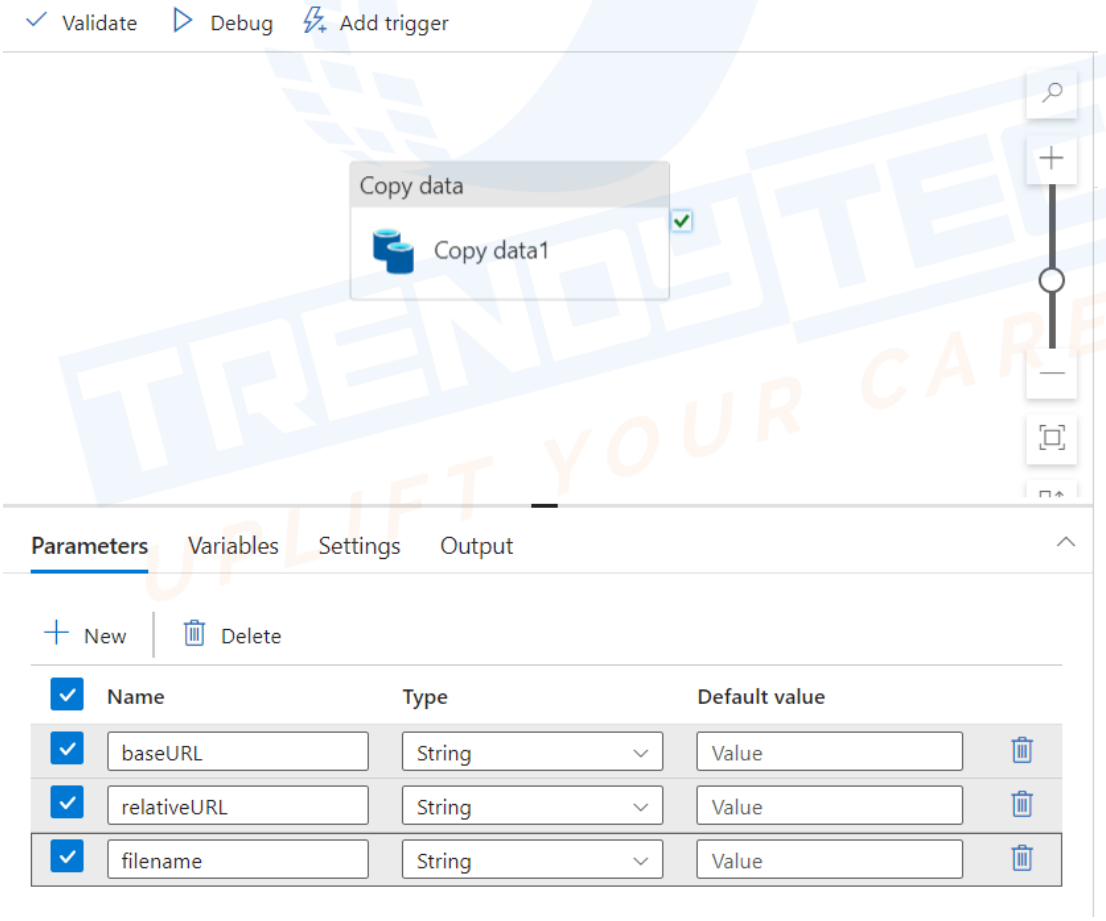
- We have to parametrize the Dataset where the relative url is present.



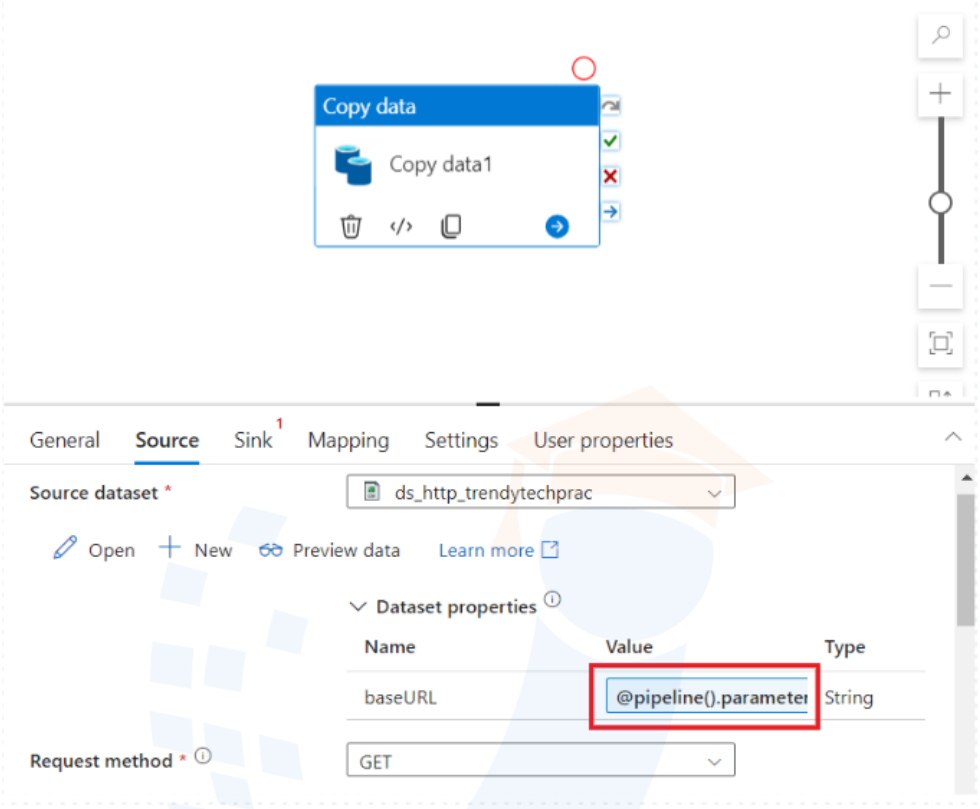


Parameterizing the Pipeline

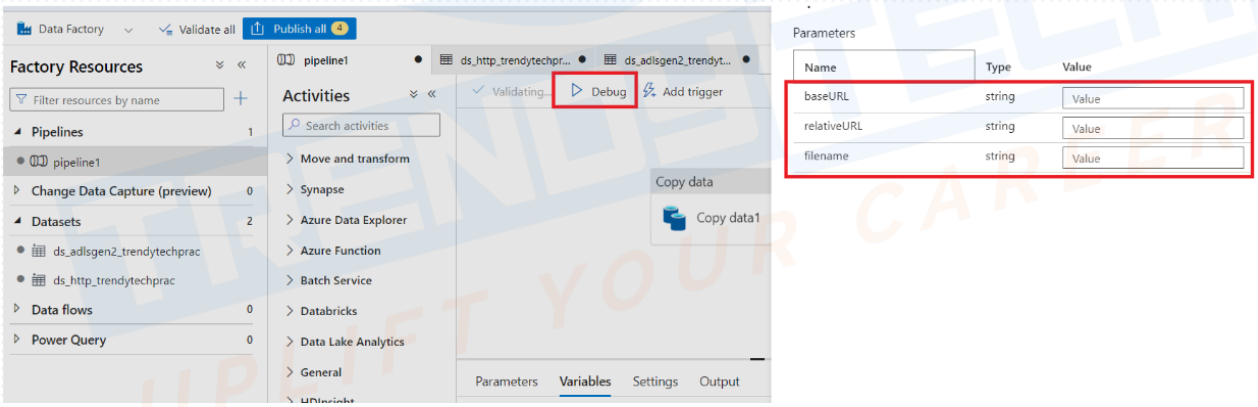
- While executing the pipeline, the parameters provided will be used by the underneath functionalities - Copy activity, Dataset and Linked Service.



- With this, passing the parameters is pushed to the pipeline level from the copy activity level.



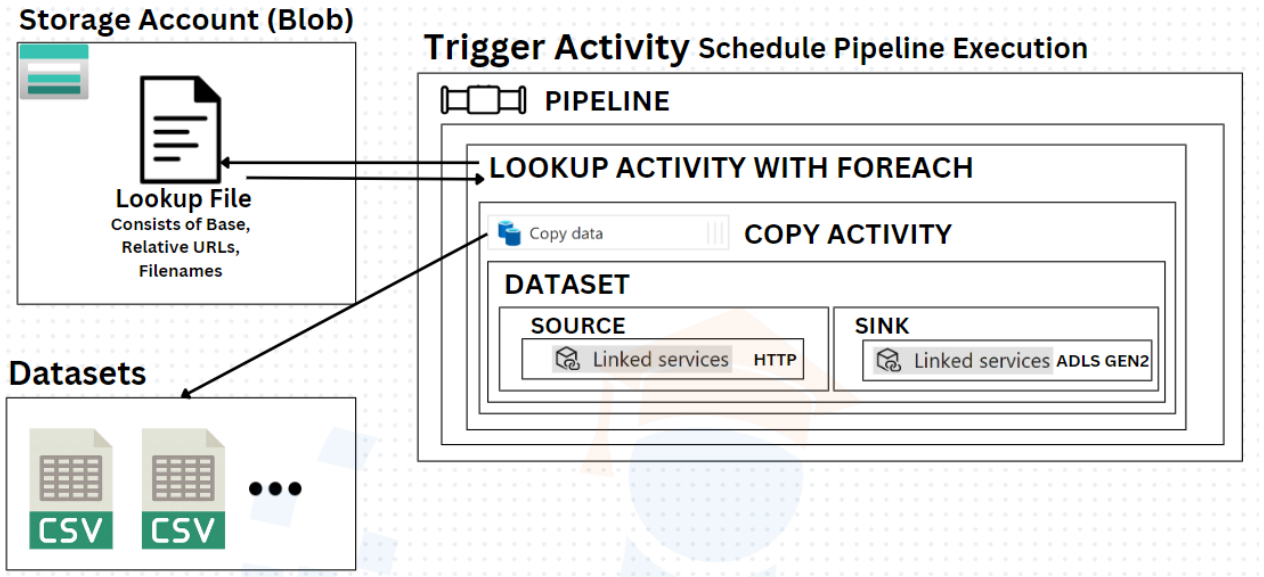
On clicking on **Debug** Pipeline, it prompts for the URL values



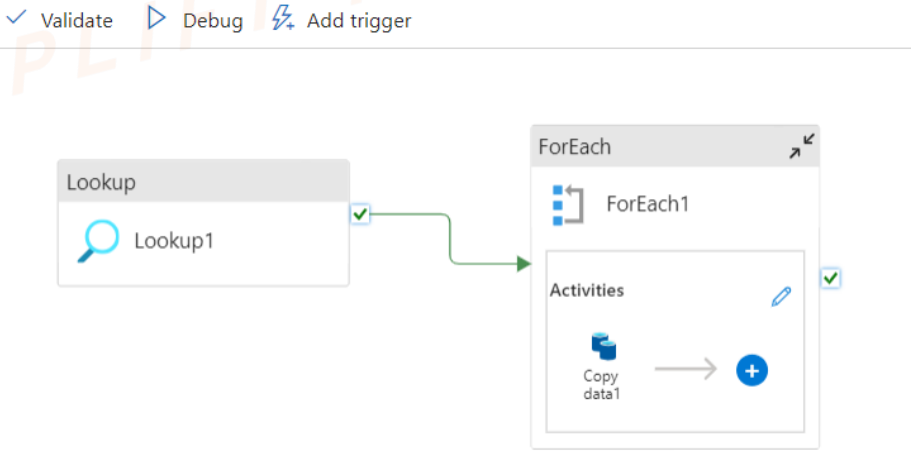
Scalable Solution : Here you can provide different URL values to download different datasets without creating a large number of components.

Lookup Activity

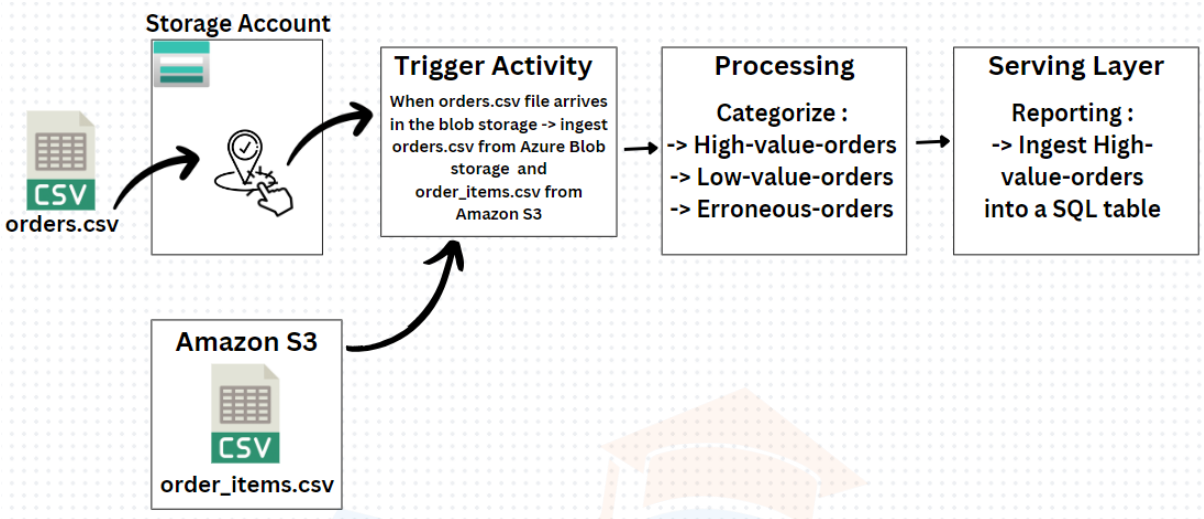
Automating the download of required dataset without having to manually pass the URL values.



- The lookup configuration files consist of the required URLs to access the datasets.
- A Trigger activity is used to schedule the pipeline execution.
- LOOKUP activity is used to access the lookup file present in the blob storage to read the URLs (Base, Relative & Filenames) for each dataset.
- A FOREACH activity is used to individually read the URLs of each dataset.
- The COPY activity inside the FOREACH activity is used to ingest the data from source and load it to the sink.



Example Use-case 2 : Ingesting the data from Amazon S3 to Azure ADLS Gen2



Ingestion

- Create a Resource Group in Azure
- Create a normal Blob Storage Account
- Create ADLS Gen2 Storage Account (with hierarchical namespace enabled)
- Create an Amazon Web Services account with S3 storage. Create a bucket within the S3 Storage and add the order_items.csv file into the bucket.
- Create an Azure Data Factory within a Resource Group.
- Launch Azure Studio and create a Linked Service to connect to Amazon S3 (Access Key ID and Secret Access Key can be obtained from the Amazon S3 -Manage IAM)
- Instead of directly entering the Key ID and Secret Access in the Linked Service, use Key vault for a more secure way of accessing the secret keys. Create 2 secrets for Access Key ID and the Secret Access Key)
- Create a Linked Service to connect to the Key vault and grant the access permission to the data factory service principal under the access policies.
- Create a Dataset for the Source(Amazon S3)
- Create a Linked Service and Dataset for Sink(ADLS Gen2)
- Create a Pipeline with Copy activity to ingest the orders.csv and order_items.csv

Processing

- For the given Use-case, data processing is carried out using **Data Flow**. Processing involves segregating the data into 3 categories : **high_value_orders(> 500) | low_value_orders(< = 500) | erroneous_orders(no_order_amount).**
- Create 3 folders for the above categories in the output folder of the storage container.
- Create a new Data Flow, add Source for order_item dataset and import the schema(*.csv). Preview the data to check if it is in the desired form. If required, the data types can be changed (Like - order_item_quantity can be changed from string to integer, order_item_subtotal and order_item_product_price can be changed to float type)
- Turn-on the **Debug mode** to check if the processing is as per the requirement.
- Use aggregate transformation activity to calculate the subtotal of respective order_items.
- Create a new **Data Flow**, add Source for orders dataset and import the schema(*.csv). Preview the data to check if it is in the desired form.
- Use the transformation **Join** to perform a full outer join for orders and order_item datasets on the condition **order_item_id == order_id**
- Use the transformation **Select** to remove duplicate columns like order_item_order_id and rename the columns if required.
- Use the transformation **Conditional Split** to segregate the data into high_value | low_value | erroneous
- Write the outputs high_value | low_value | erroneous to the respective sinks in the output folder of ADLS GEN2.

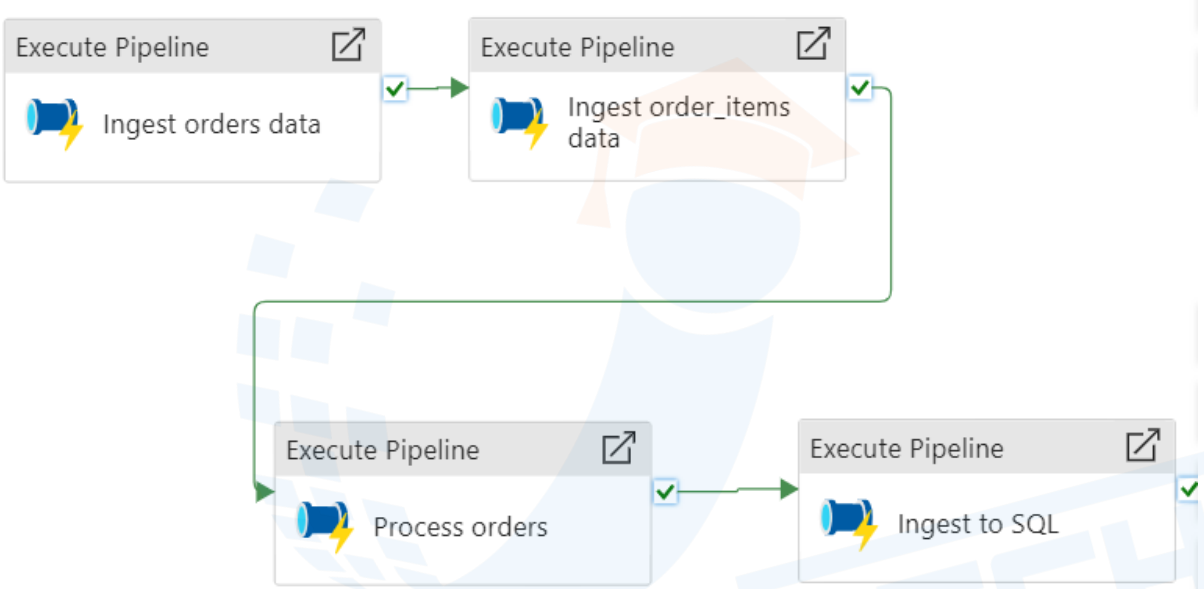
Populating the SQL Database for use by Reporting team

- Create and deploy an Azure SQL database.
- Create a Linked Service pointing to SQL DB.
- Create a Linked Service pointing to ADLS GEN2 where the high_value_order output is stored.
- Create a Dataset pointing to the high_value_order data.
- Create a Dataset pointing to the table in SQL DB.
- Create a table definition for premium orders(high_value_orders)
- Create a Pipeline to push the data from ADLS GEN2 (Source) to Azure SQL DB (Sink).

Organizing the Pipelines and Datasets

Organizing the datasets and pipelines into folders

- Create folders for datasets and add the datasets to their respective folders (orders dataset, order_items dataset, sql)
- Create folders for pipelines and add the pipelines associated to specific activities to their respective folders (ingest, process, sql ingestion)
- Create an **Execute Pipeline Activity** to chain all the pipelines and to get executed in a specific order.



- Create a **Trigger** to ingest and process data with **Storage Event Type**(the trigger will get initiated whenever a new data file arrives at the blob storage as mentioned in the storage account and container name fields while creating the trigger)
- Attach the created Trigger to the Execute Pipeline to trigger the pipeline execution without any manual intervention whenever a new file gets added to the blob storage.

Different types of Triggers

1. Scheduled Trigger - Scheduling activities for a future point in time.
2. Tumbling Window - It handles slices of data and these triggers can be executed for past intervals as well.
3. Storage Event - Related to Storage. The trigger gets fired when a file is created or deleted.
4. Custom Event - Any custom event on which a trigger has to be fired.

Key Points :

- Just like chaining of pipelines, chaining of triggers is possible but only for tumbling window triggers.
- One trigger can invoke multiple pipelines.
- Many triggers can be attached to one pipeline.
- For scheduled triggers, a many-to-many relationship is possible between the pipeline and triggers.
- For tumbling window triggers, only a one-to-one relationship is possible.