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!

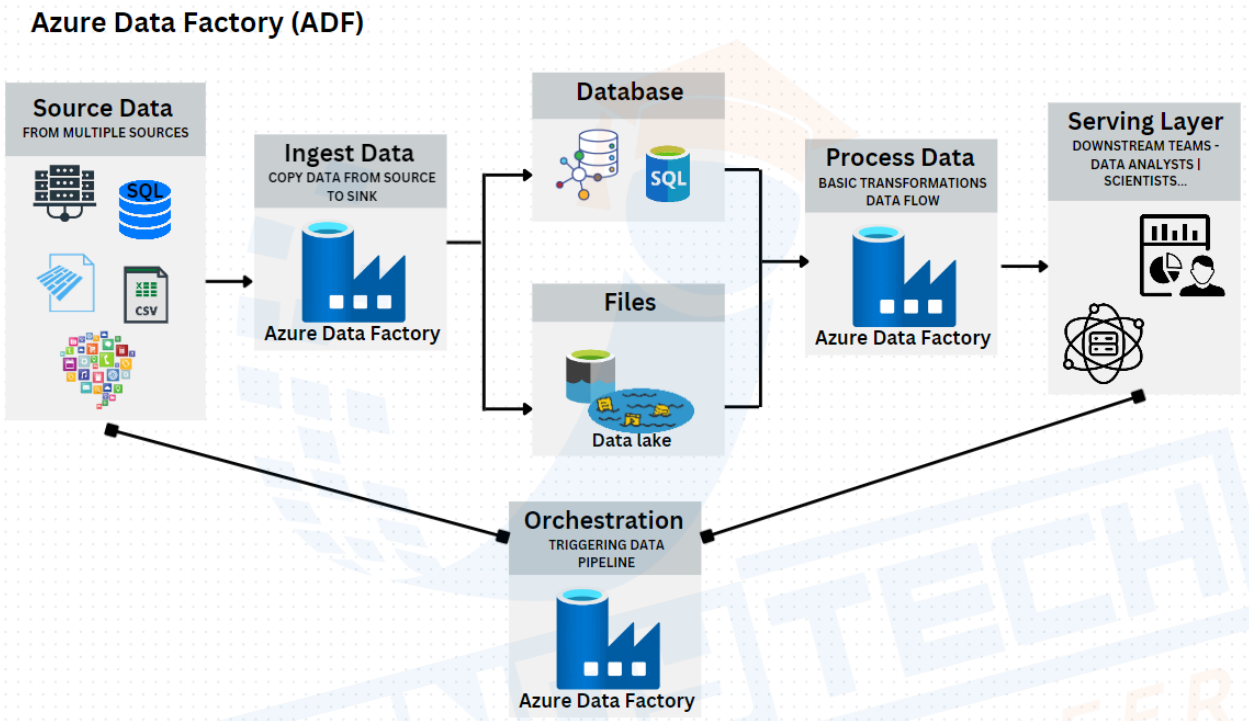
- All the Best and Happy Learning!

Azure Data Factory

Requirements :

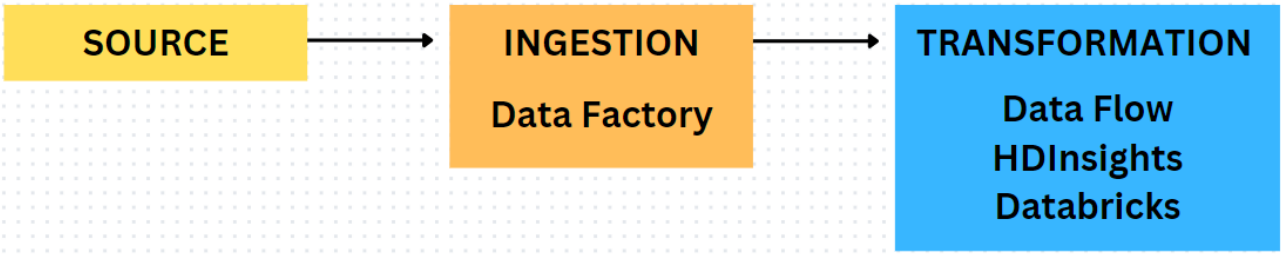
1. Create a platform for the Data Science team to perform advanced analytics and extract insights from data.
2. Create a platform for the Data Analytics to generate visual dashboards and also execute AD hoc queries.

ADF is a fully managed serverless service



3 main features on ADF

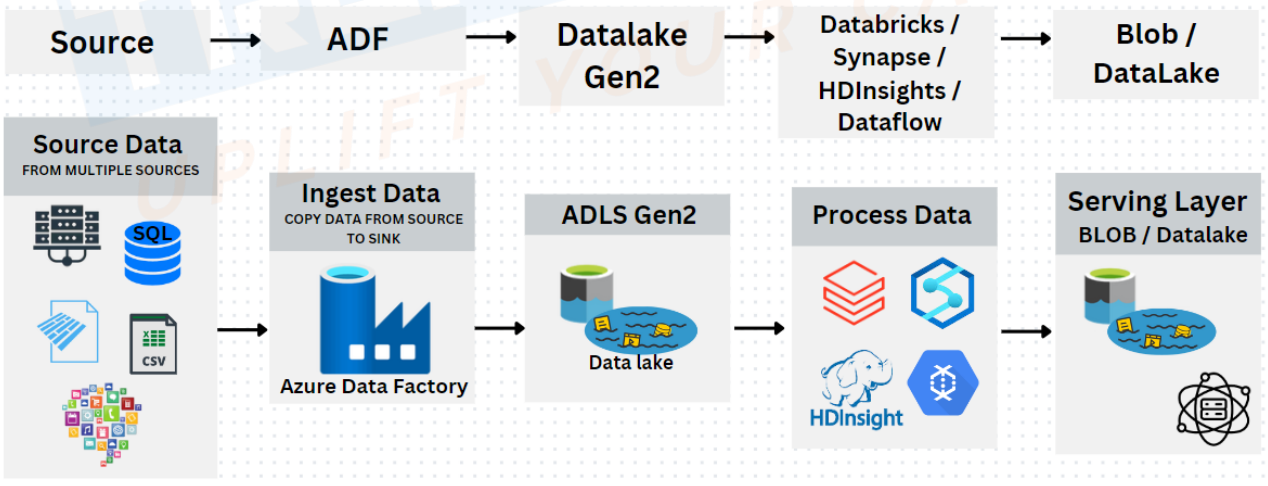
- **Transfer the Data from Source to Sink** blob -> adls gen2 | RDBMS-> adls gen2 | adls gen2 -> RDBMS)
- **Basic Transformations** Data Flow in ADF performs only basic transformations
- **Orchestration** strategies to trigger the data pipeline



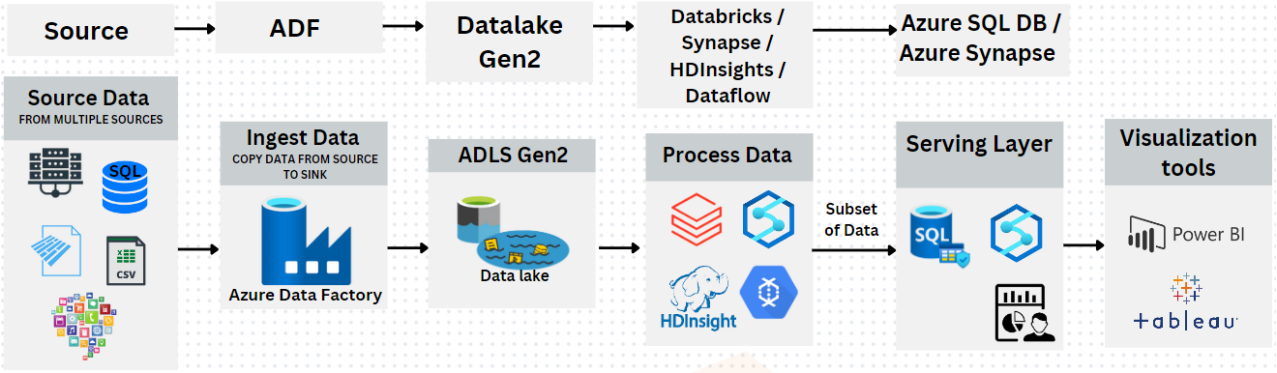
Data Processing Services

- Dataflow is for basic transformation without having to write any code.
- HDInsights is a Managed hadoop cluster on Azure - Hortonworks cluster
- Databricks is specifically for spark and is an optimized spark cluster to execute spark code at a much faster rate.

Data Pipeline for Machine Learning Team



Data Pipeline for Adhoc Reporting needs for Analytics Team



Use-case 1

Ingesting the data from **Source(RDBMS Table)**, transferring and loading the data to **Sink(ADLS GEN2)**

Pre-Steps

- Create a Dashboard and a Resource Group for the project to organize the resources related to the project at one place.

Creation of Source System - Azure SQL DB

- Create an Azure SQL Database and pin the resource to a dashboard for better organisation of resources.
- Once the SQL DB resource is deployed, set the server firewall rule under overview tab by adding your respective IP-address
- Login to the SQL terminal in your azure account (You could also access the SQL server from your local system by installing and remotely connecting to Azure Data Studio or SQL server management studio)
- Create Table (Courses Table)
- Insert the Data into the table

Creation of Sink System - ADLS Gen2

- Create a Storage Account (Enable Hierarchical namespace to make it as data lake storage and not just the blob storage)
- Create a Container inside the Storage Account.
- Create a Directory under the Container.

Creation of Data Integration Service - Azure Data Factory (To ingest data from source and load it to sink)

- Create Azure Data Factory resource.
- Open the Azure Data Factory Studio after the resource is deployed.
- Firstly, connect to the source and sink using the **Linked Service**.
- Connect to the Source using the Linked Service Azure SQL Database. (Authentication to connect to SQL server and enable this service to be accessible by other services under firewall settings)
- Connect to the Sink using the Linked Service ADLS Gen2.
- Create and Publish Datasets on-top of Source and Sink indicating the format type of data to be stored underneath.
- Create a Pipeline and a Copy activity within the pipeline. Select the Source as Azure SQL DB and the Sink as ADLS Gen2 as created in the previous steps.
- Debug to run the pipeline to execute the activity and validate it.
- Monitor the pipeline under the monitor tab.

Note : Mapping Data Flow is a feature for creating transformation flow graphically and the spark code for the same is generated internally.

Use-cases for which Azure Data Factory is not meant for :

- ADF doesn't provide storage capability.
- Suitable for only simple transformations not for complex transformations.
- ADF doesn't provide data streaming capabilities.
- ADF is not a migration tool.

Use-case 2 : Ingesting Orders & Customers data

Dataset orders.csv can be downloaded from - https://files.cdn.thinkific.com/file_uploads/349536/attachments/c28/5fb/25b/orders.csv

Dataset customers.csv is provided in the Week18 downloadable section in the portal

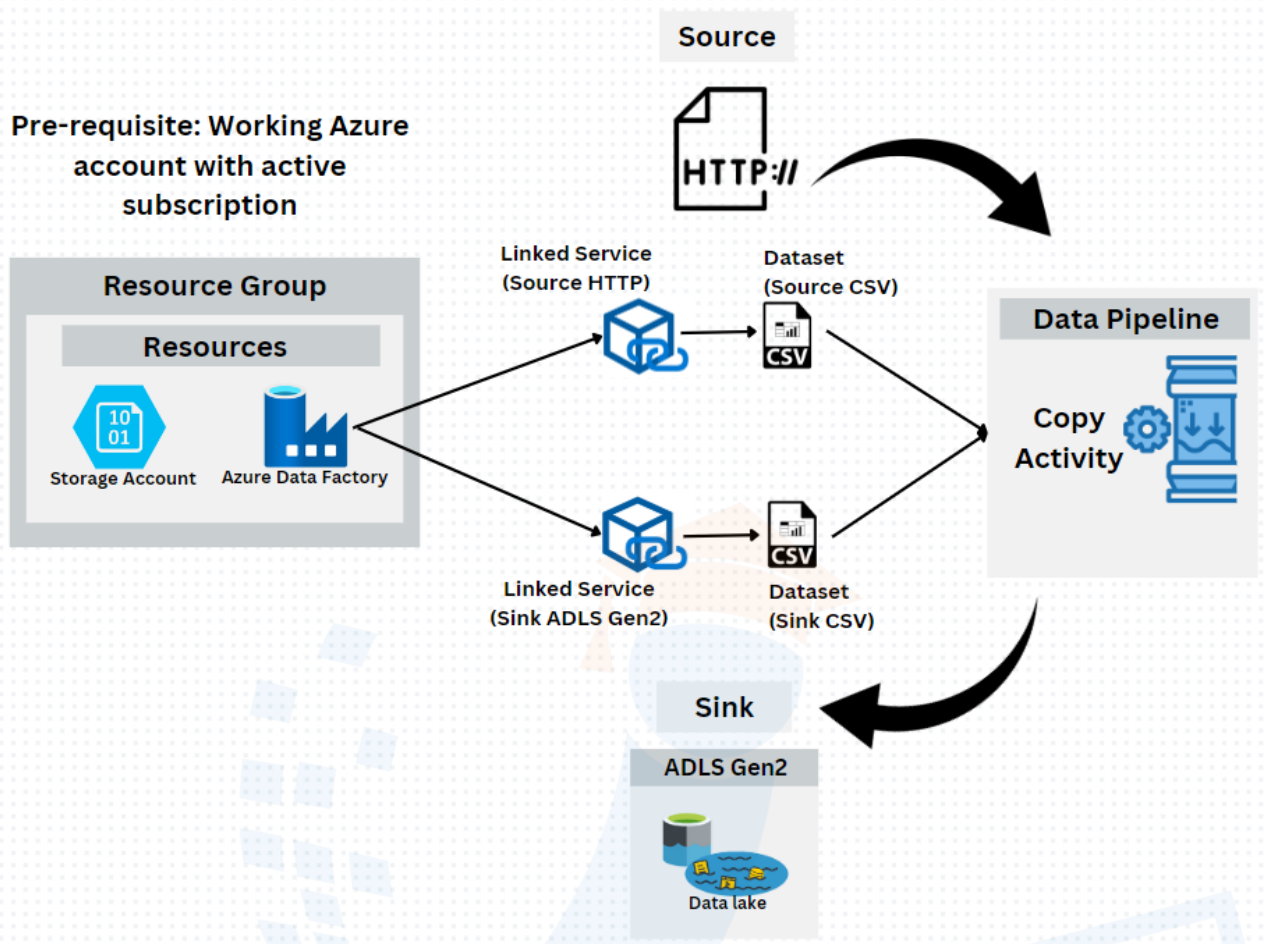
Requirement

- **Ingest orders.csv file from external URL to ADLS Gen2**
  a. **Require working Azure account**
  b. **Active Subscription**
  c. **Create Resource Group for efficient organization of project Resources.**
  d. **Create a Resource - Storage Account (Datalake)**
  e. **Create a Resource - Azure Data Factory**

  [Create the following in the Azure Data Factory service]

  f. **Create a Linked Service for Source (choose HTTP connector)**
  g. **Create a Linked Service for Sink (choose Datalake Gen 2 connector)**
  h. **Create Dataset for Source (choose CSV format and also provide the relative URL as it is for the HTTP Linked Service)**
  i. **Create Dataset for Sink**
  j. **Create a Data Pipeline**
  k. **Create a Copy activity within the Pipeline**

- **Perform some basic transformations**
  a. **Remove order_date column**
  b. **Rename order_customer_id to customer_id**
  c. **Calculate the count of each order status**



Dataflow is used to perform basic transformations on the ingested data

- Create a new Dataflow service and add the required transformations to be performed on the data as activities in the workflow.
- **Dataflow** should have a source -> we perform transformations on the data ingested from the source and then the transformed data will be loaded to sink.
- Source types (**Dataset / Inline**) : When the Dataset would be used multiple times for other transformations as well then we need to choose Dataset as source type. If the Data is used only once in the current transformation then we need to choose Inline as source type.
- Enabling **Schema Drift** option in the source settings option of the dataflow support schema evolution i.e., it supports the changes to data columns.
- Data Preview is possible if a spark cluster is turned on. For development purposes, **Debug mode** can be used to preview the data after small transformation changes are made.

- As per our requirement we need to remove the order_date column and rename order_customer_id to customer_id. For both of these activities, we need **SELECT** transformation and there will be graphical ways to make the necessary changes.
- For the next requirement of calculating the count of each order status, we would need, **AGGREGATE** transformation.
- Data preview will display the results of transformation.
- If the results in the preview are as per the requirement, then load it to the sink in the desired format.
- If all of the transformations in the dataflow are as desired, then publish the dataflow
- Create a Data pipeline and add the Data flow with the transformations to the Data pipeline.

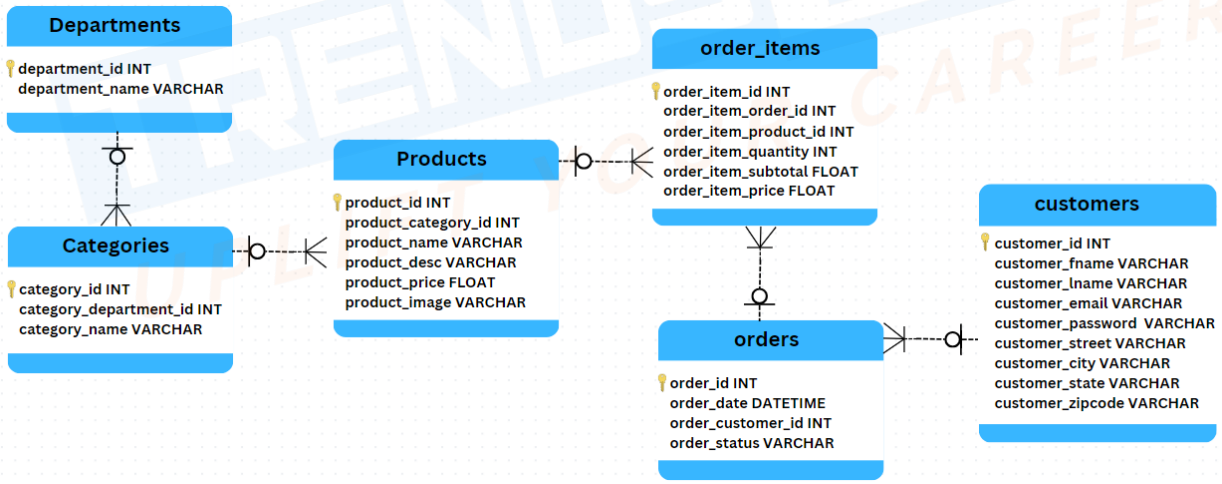
Note :

ADF is majorly preferred for **Ingestion** and **Orchestration** and not for performing transformations in the industry.

-Complex spark code needs to be developed in order to meet the business requirement which calls for flexibility in developing code and thereby more control over data.

Use-case 3 : Retail Data

Datasets



Requirement :

Products.csv will be updated by third party vendors regularly. Currently in the blob storage and we need to bring the products dataset to Datalake.

Solution :

- Create Resource Group
- Create **Storage Account** Resource with hierarchical namespace enabled under the resource group
- Upload the products dataset to a container in blob storage
- Create **Data Factory** Resource
- Create **two Linked Services** in ADF, one pointing to the Source(Blob Storage) and the other pointing to the Sink(ADLS Gen2)
- Create **two Datasets,** one for the products.csv source file in blob storage and the target dataset in csv format in ADLS Gen2. Publish the datasets.
- Create **Pipeline** and choose the source and since to execute the required copy activity.

To enhance the above process to make it production ready

1. Pick the file as soon as it is available in the blob storage without manual intervention.

   -Bring in the **Validation activity** present under the general category. Set the values for timeout and sleep parameters to automatically pick any new files that are added to the source.

2. Perform basic validation / sanity check before ingesting to Datalake.

   -Use the **Get Metadata activity** to perform basic validations like column count, file size…

   -Use **If Condition** to validate dynamic content from the dataset with a norm value.

3. Notify when the pipeline execution fails.

   -Use **Fail activity** to indicate in case the above If condition data validation process fails and use **Alert Rule** to notify the pipeline failure.

4. Create a **Trigger** and associate it to the pipeline to schedule the pipeline execution.