Azure Data Factory + Azure Synapse Analytics : Hands on - End to End project

Course mainly focuses on:

- Project oriented hands-on practical learning
- Real time use cases of ADF and other Azure data engineering services
- Transforming data using Azure Synapse Analytics
- Building an end to end ETL project using Azure Data Stack

Pre-requisites:

- Azure Account with a Subscription
- Fundamental knowledge on Azure Data Factory
- Fundamental knowledge on Azure Synapse Analytics

  would be beneficial, not mandatory

Services used in this Project

Azure Data Factory

Azure Data Lake Storage Gen2

Azure Synapse Analytics

Power BI

Azure SQL

Services used:

- Azure DataLake Storage Gen2
- Azure Data Factory
- Azure Synapse Analytics
- Azure SQL Database
- Power BI

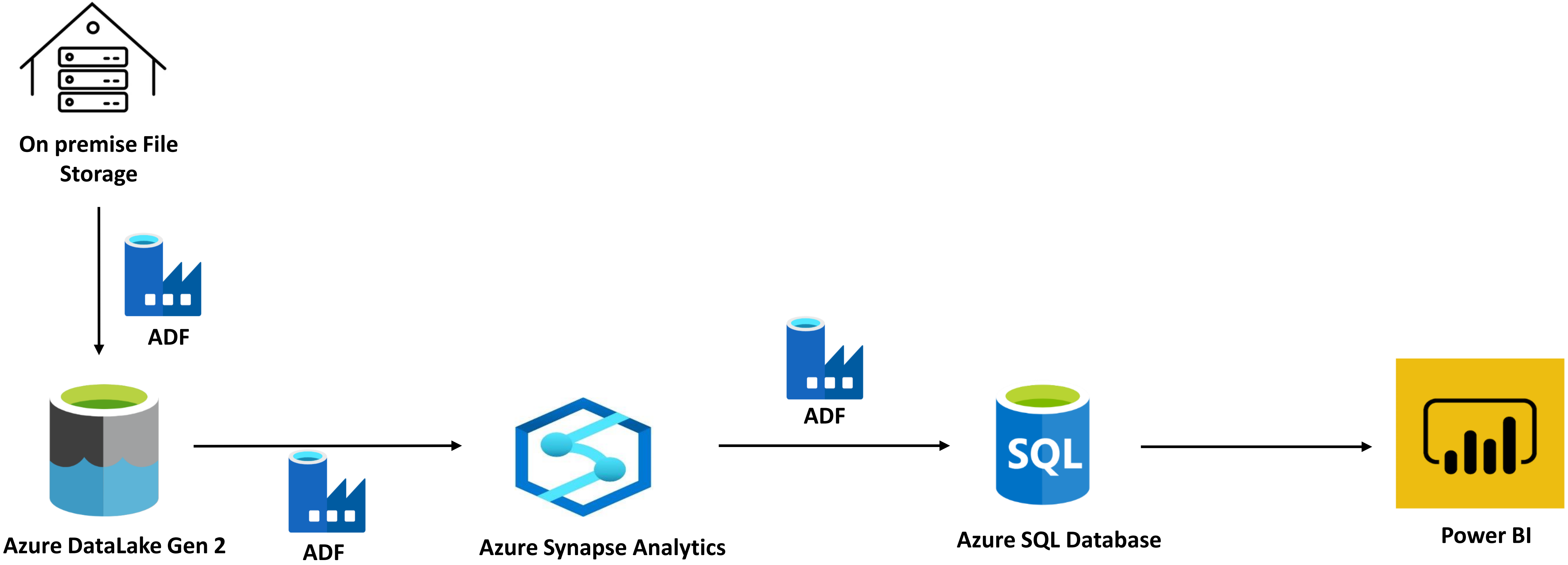
Project Overview:

- Dummy data of an OTT platform (Netflix, Prime, Disney+, etc.) from Kaggle
- OTT platform hosts movies/TV shows in the platform of all languages.

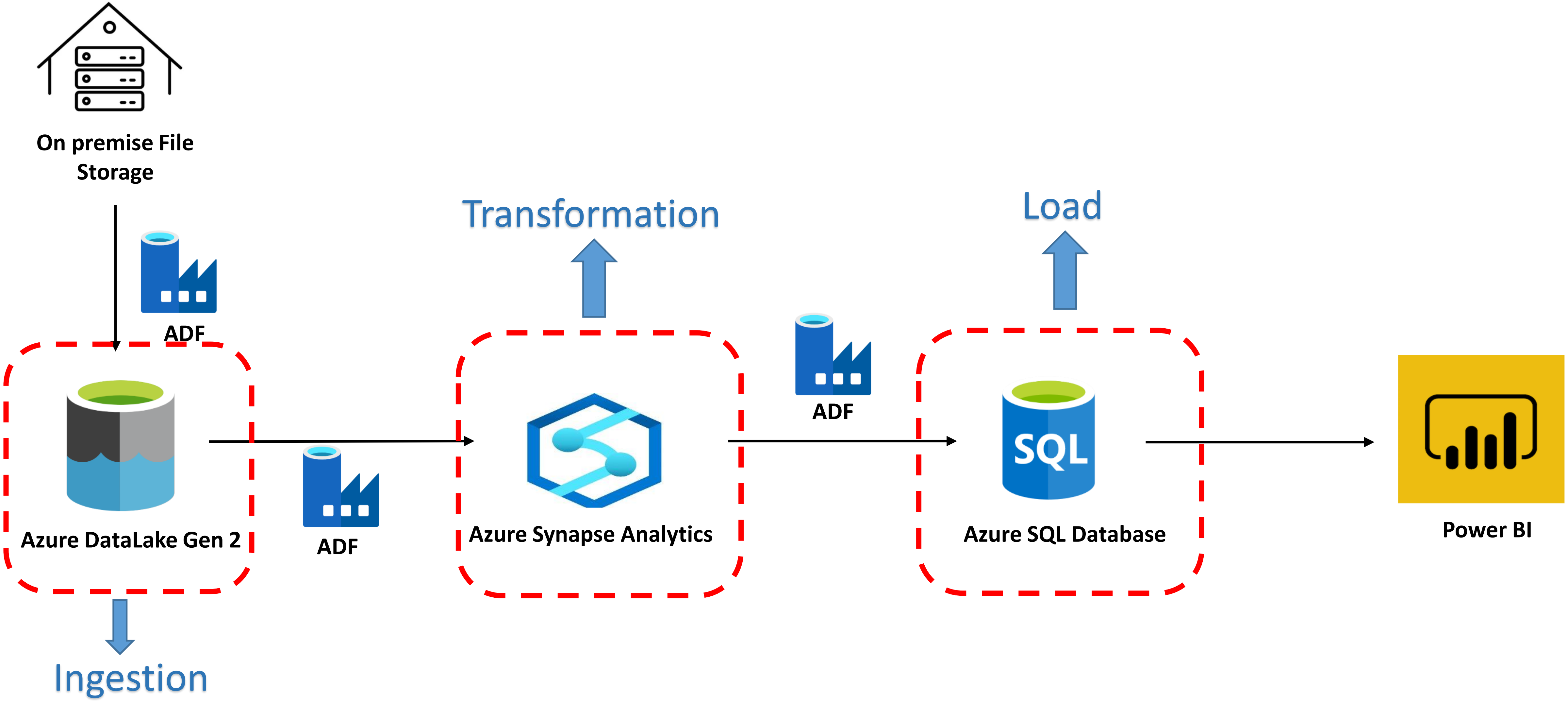
Data source: On-premise file storage.

- Ingest the On-premise data to Azure Cloud using ADF
- Perform the ETL (Extract , Transform, Load) operation on the data
- Report the data using Power BI
- Analyse the trends of OTT platform for decision making

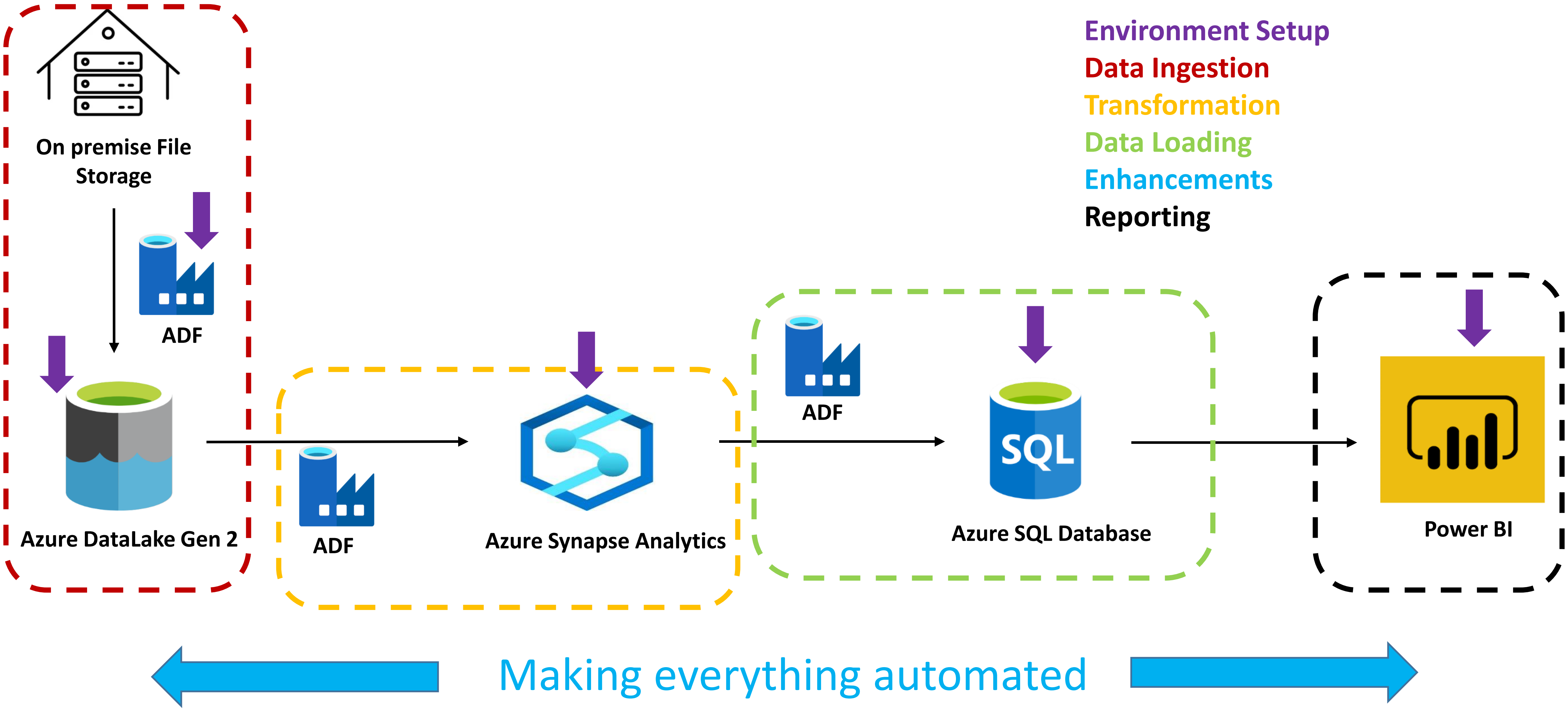
Project Architecture



Project Architecture



Course Structure



Understanding Dataset:

Dummy raw dataset have below columns:

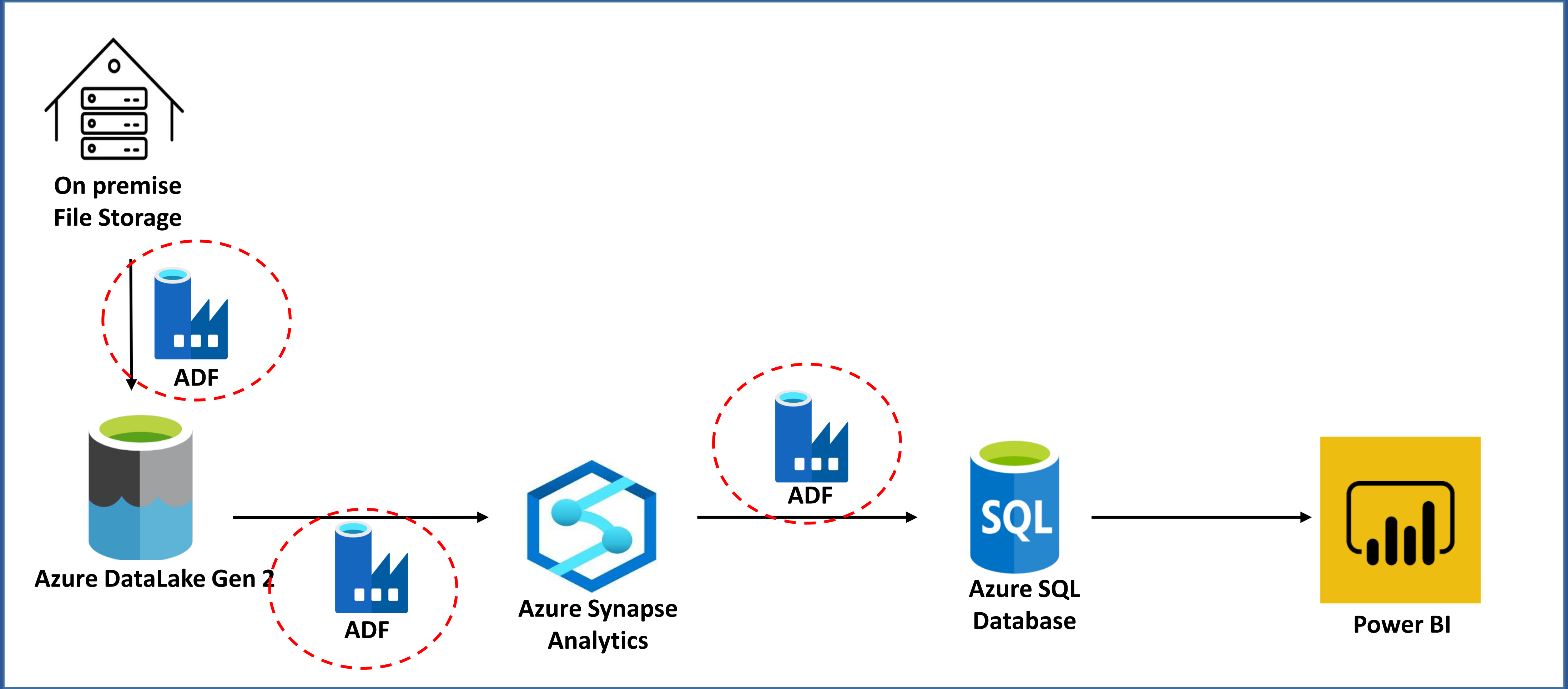
• Title	- Title of the film
• Genre	- Genre of the film
• Release date	- Movie/TV Show release date
• Runtime	- Runtime in minutes
• IMDB scores	- Scores given by IMDB
• Language	- Languages currently available
• Views	- Number of views
• Added Date	- Date in which its added to OTT

Environment Setup

Environment Setup

Creating Azure Data Factory

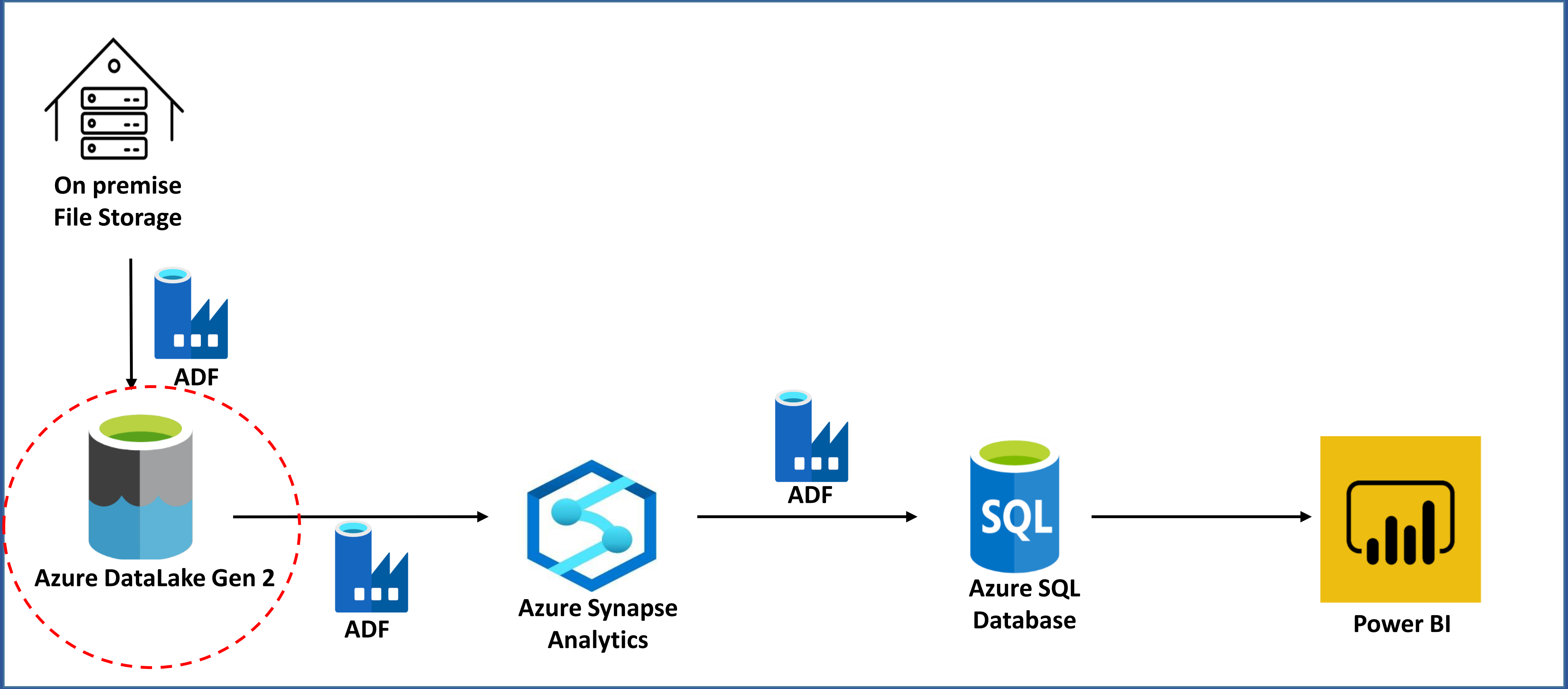
Name: ott-datafactory-011



Environment Setup

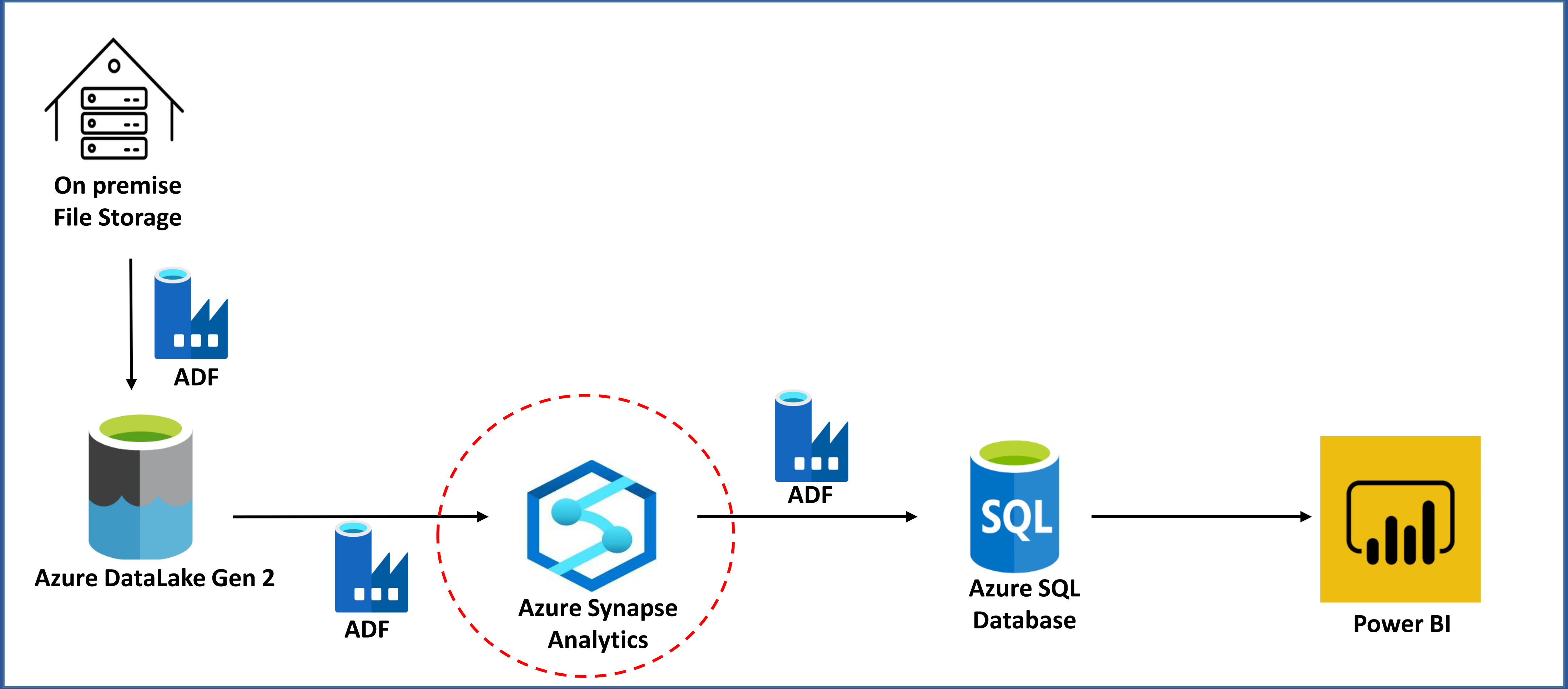
Creating Azure Data lake Storage Gen2

Name: ottadls011



Environment Setup

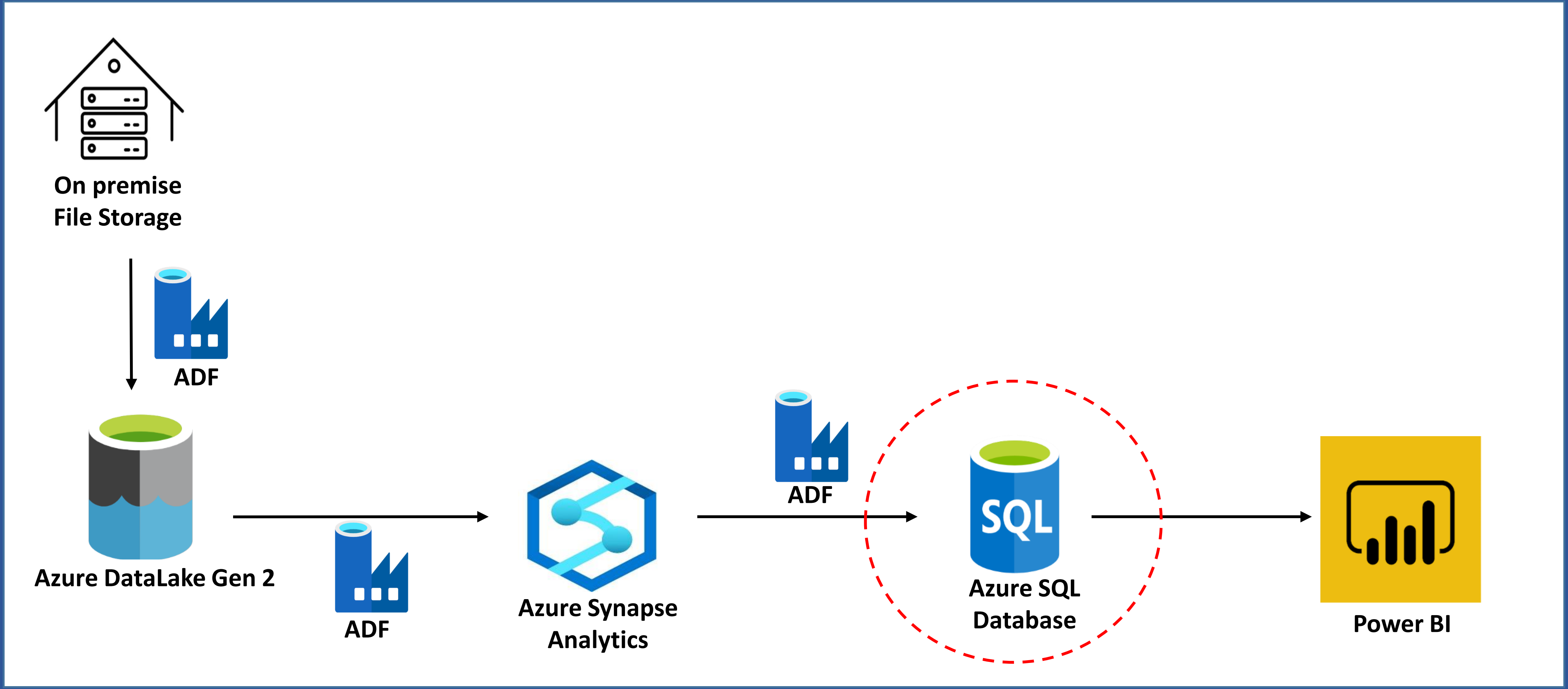
Creating Azure Synapse Analytics Name: ott-synapse-011



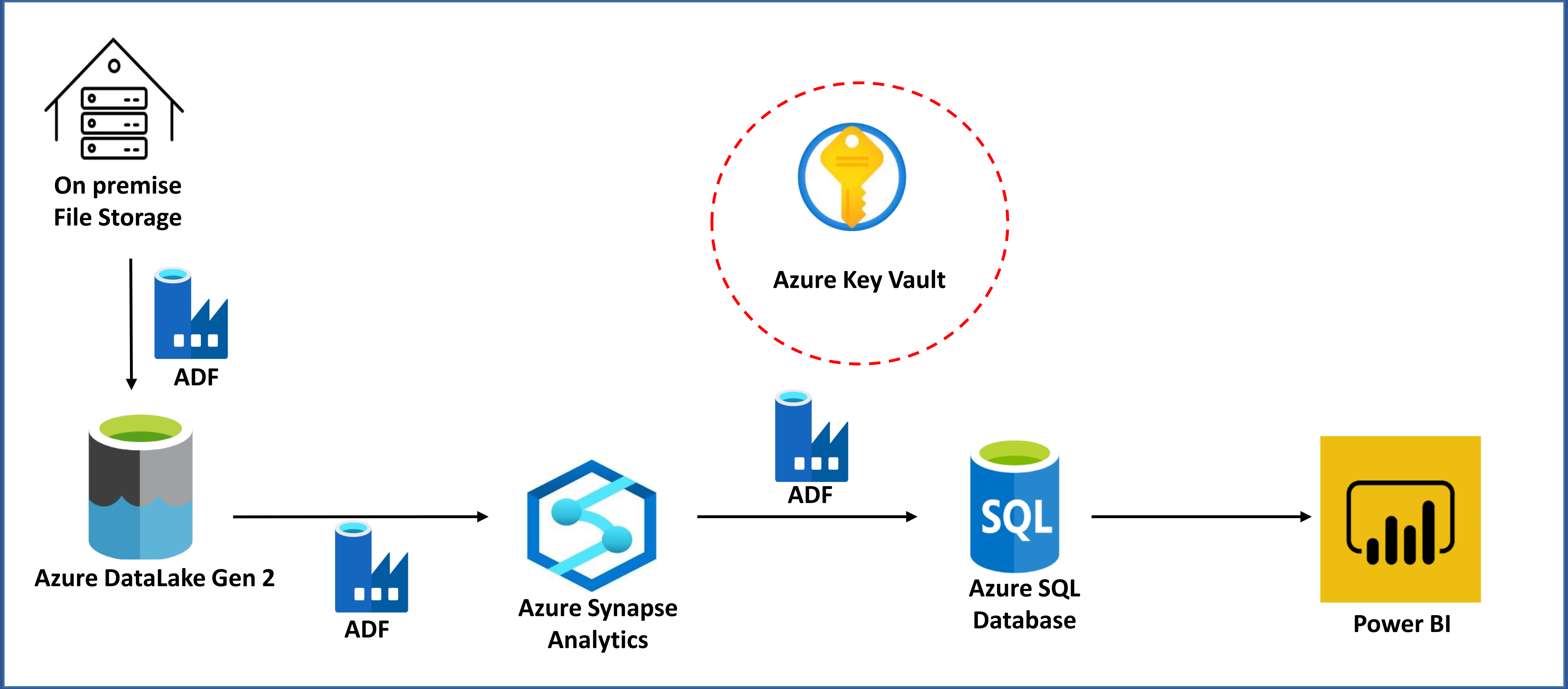
Environment Setup

Creating Azure SQL Database

Name:ott-db-022

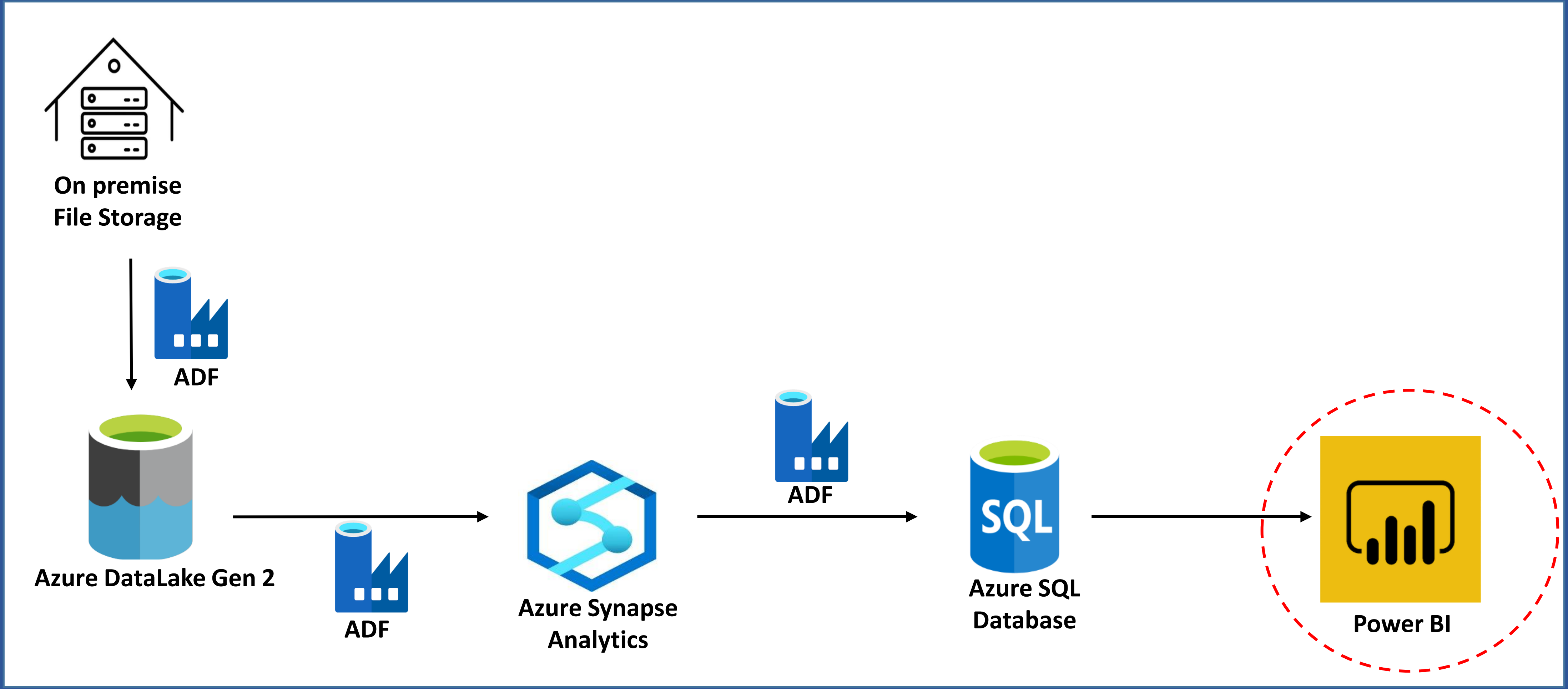


Creating an Azure Key vault for storing secrets



Environment Setup

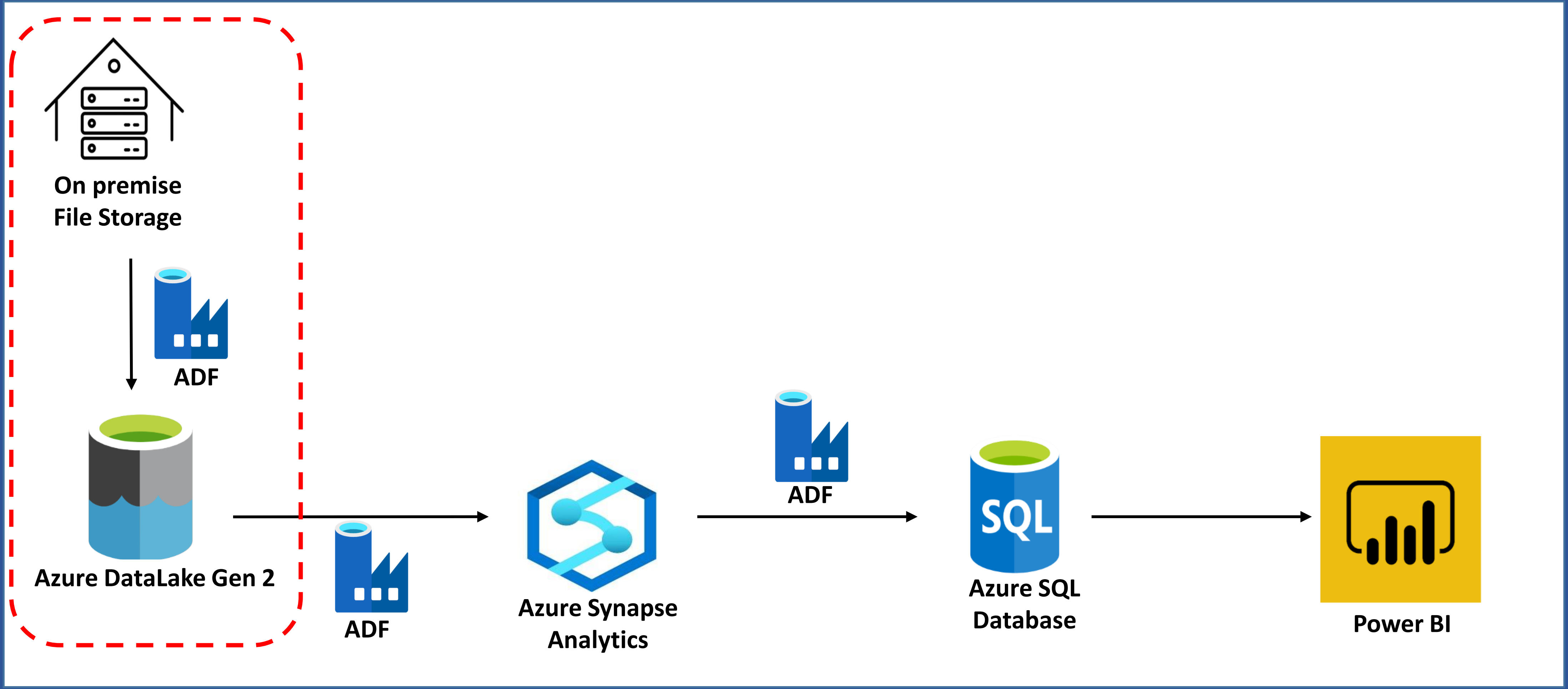
Installing Power BI



Data Ingestion

Data Ingestion

Ingesting data from On-premise environment using ADF



Integration Runtimes

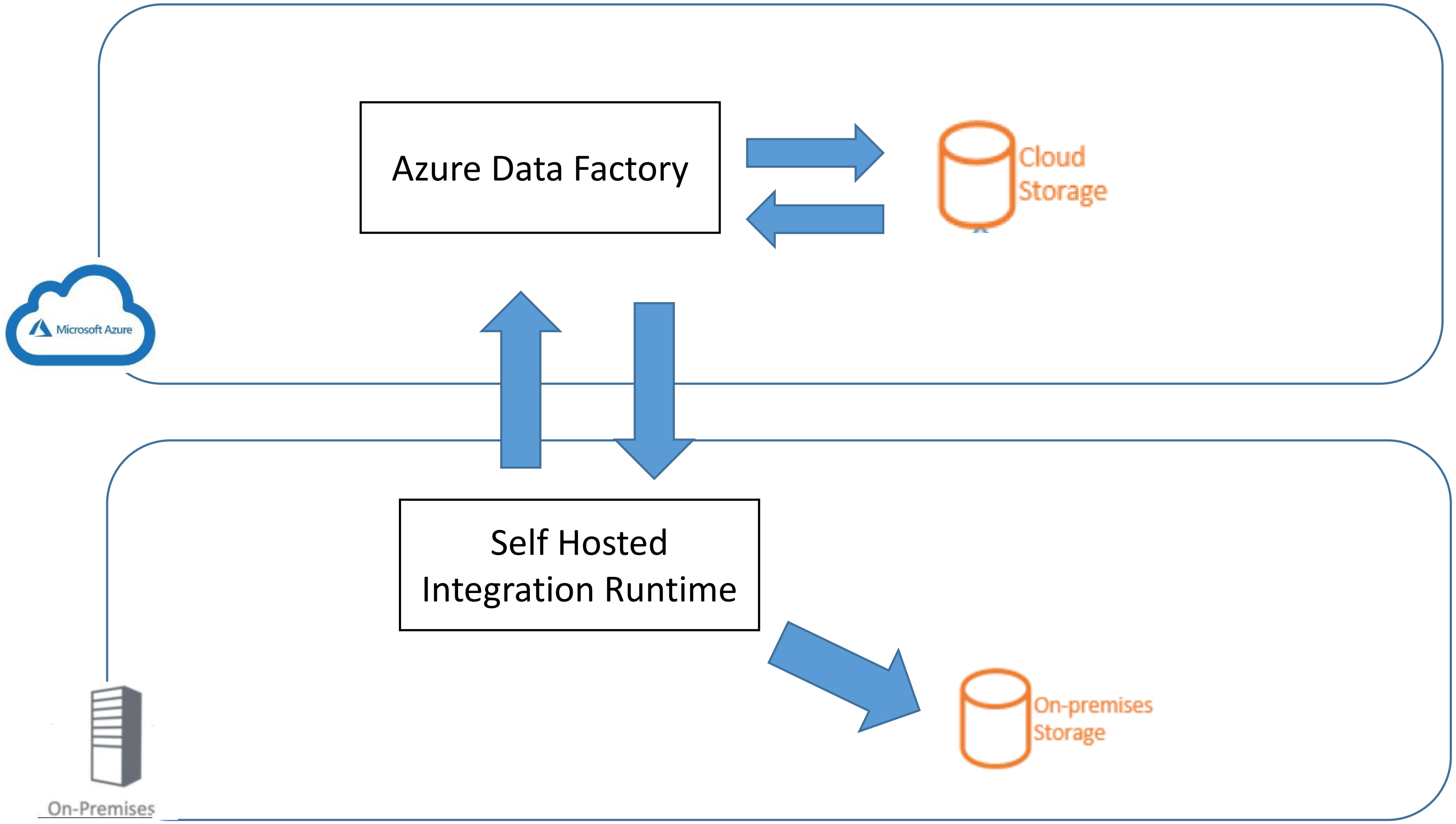
Compute Infrastructure used by Azure Data Factory

Integration Runtimes
Azure Integration Runtime
Self-hosted Integration Runtime
Azure-SSIS Integration Runtime

Self Hosted Integration Runtime

- Our data is located in an On-premise environment, which is outside of the Azure Cloud.
- Azure Data Factory has a feature called “Self Hosted Integration Runtime” to access on-premise environment data sources.
- Self Hosted Integration Runtime is a tool which acts like a bridge and provides a computation infrastructure to integrate between on-premise and cloud.

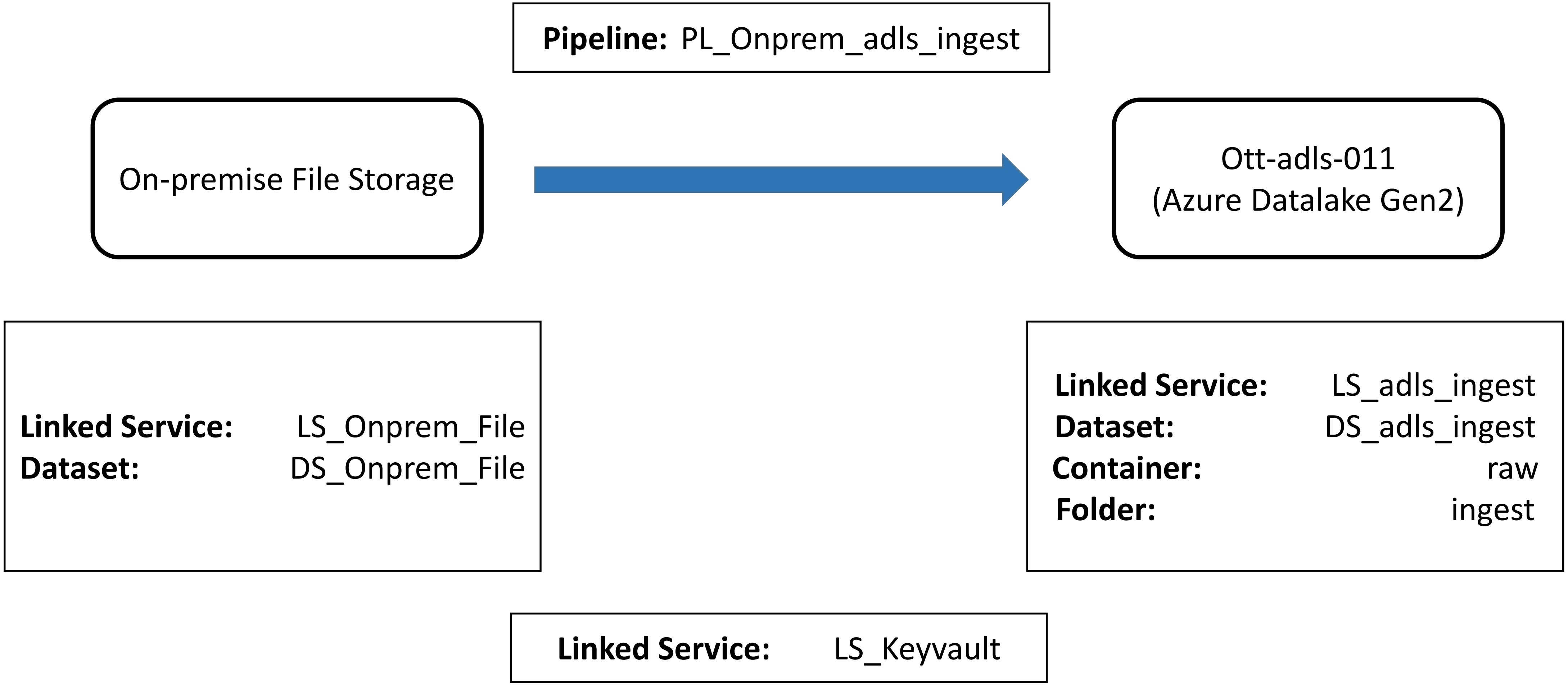
Self Hosted Integration Runtime



Data Ingestion

- Create a self hosted Integrated runtime in ADF
- Download and configure self hosted IR in your On-premise environment

Data Ingestion:



Pipeline: PL_Onprem_adls_ingest

On-premise File Storage

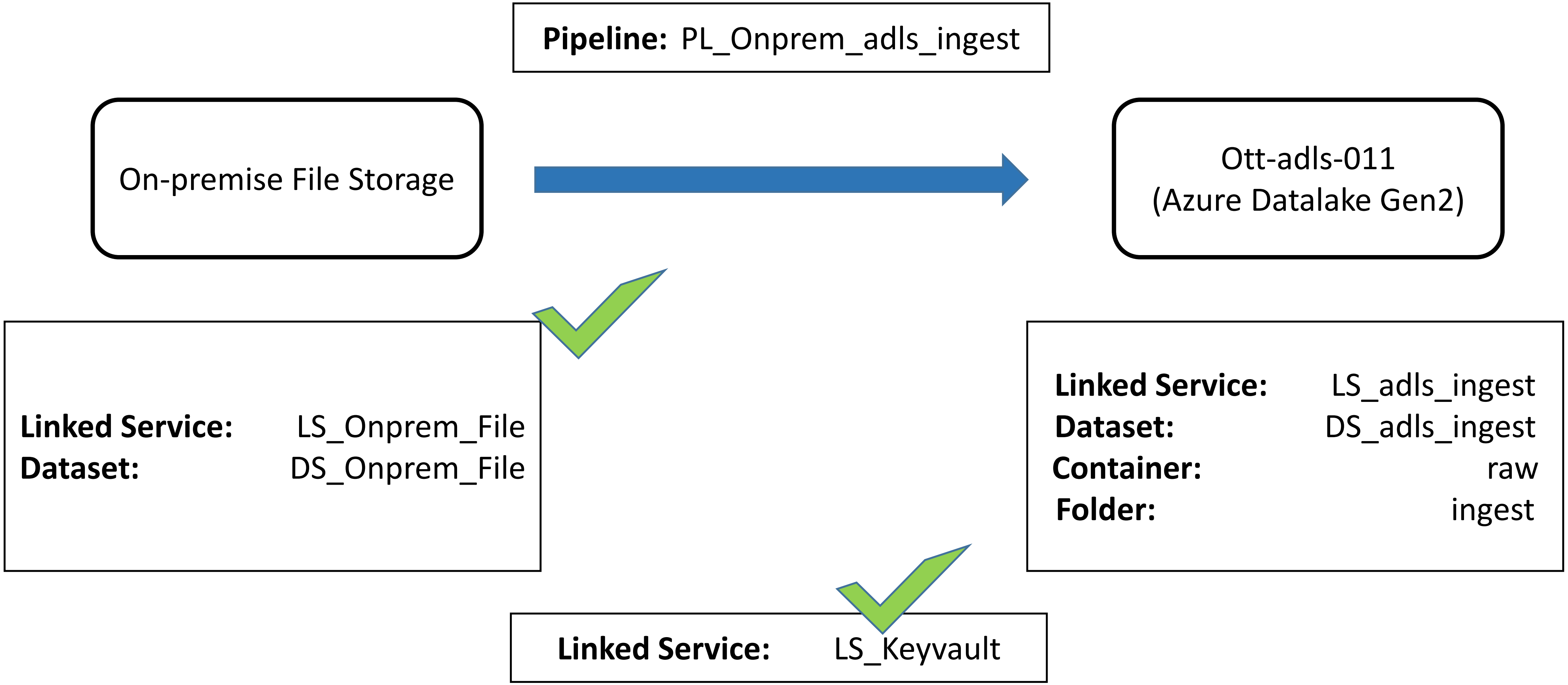
Ott-adls-011
(Azure Datalake Gen2)

Linked Service:	LS_Onprem_File
Dataset:	DS_Onprem_File

Linked Service:	LS_adls_ingest
Dataset:	DS_adls_ingest
Container:	raw
Folder:	ingest

Linked Service: LS_Keyvault

Data Ingestion:



Pipeline: PL_Onprem_adls_ingest

On-premise File Storage

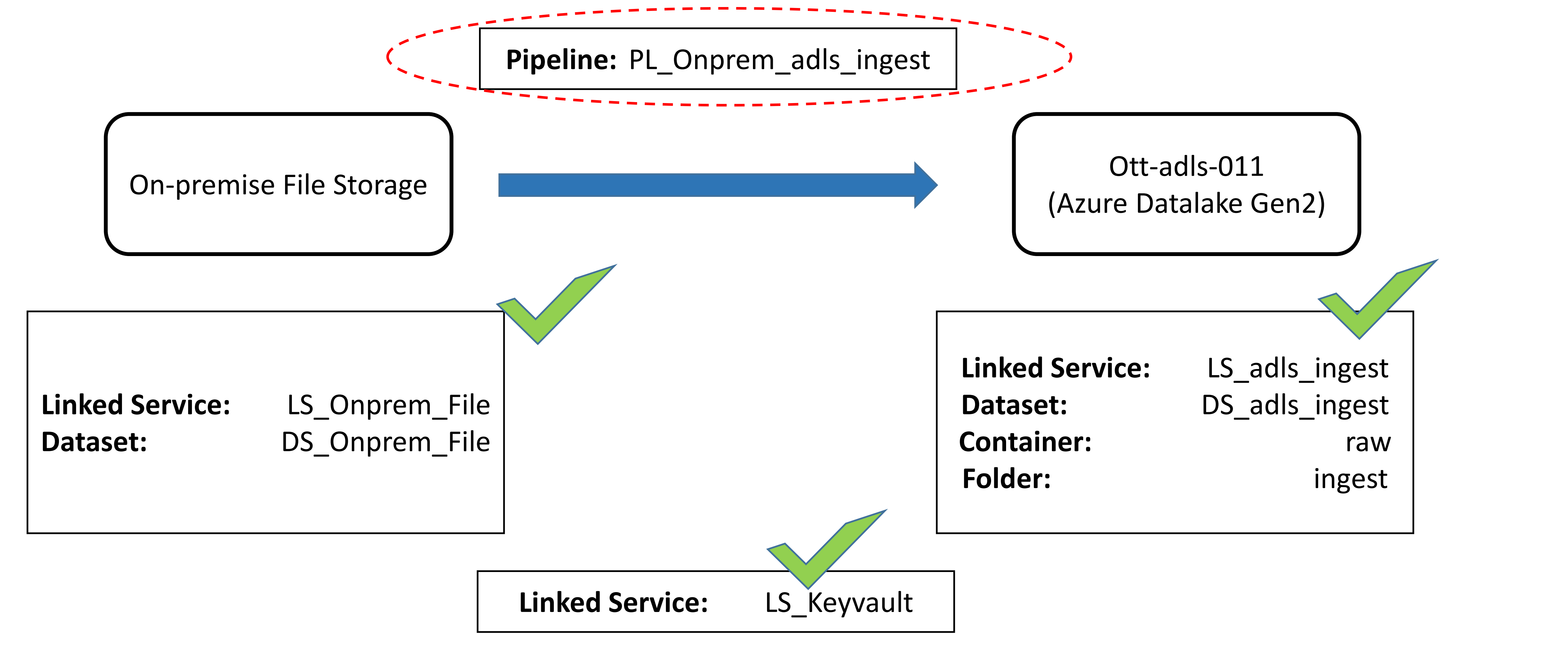
Ott-adls-011
(Azure Datalake Gen2)

Linked Service: LS_Onprem_File
Dataset: DS_Onprem_File

Linked Service: LS_adls_ingest
Dataset: DS_adls_ingest
Container: raw
Folder: ingest

Linked Service: LS_Keyvault

Data Ingestion:



Pipeline: PL_Onprem_adls_ingest

On-premise File Storage

Ott-adls-011
(Azure Datalake Gen2)

Linked Service:	LS_Onprem_File
Dataset:	DS_Onprem_File

Linked Service:	LS_adls_ingest
Dataset:	DS_adls_ingest
Container:	raw
Folder:	ingest

Linked Service:	LS_Keyvault

Incremental Loading:

- Activity of loading only new records from a source into Treasure Data.
- Incremental Load is way faster than the Full Load and also consumes relatively fewer resources too.
- It requires lesser time

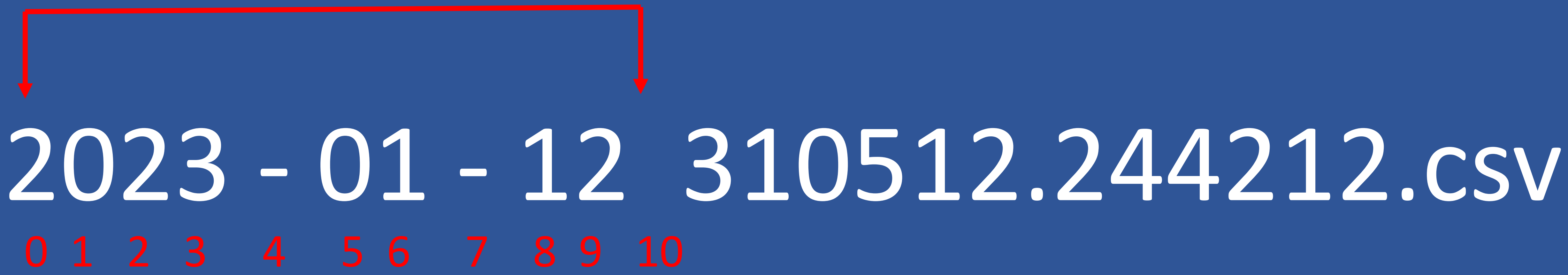
Incrementally loading files using last modified date

- Files will be added to source on daily basis
- Using last modified date of file we can do incremental data load
- File that is modified today only will be picked by Azure Data Factory

Incrementally loading files using date in File Name

- We can also perform incremental load using format of File Name
- By extracting the date from File Name incremental load can be performed

Extracting date from File Name



2023 - 01 - 12 310512.244212.csv

0 1 2 3 4 5 6 7 8 9 10

2023 - 01 - 12 = Today’s Date

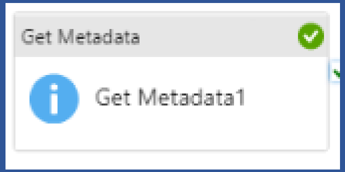
To perform the date extraction and copy:

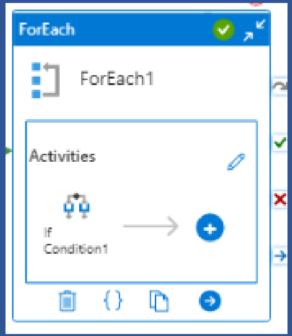
Get Metadata

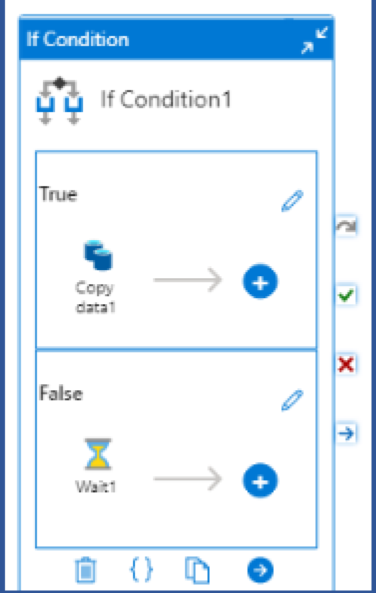
For Each

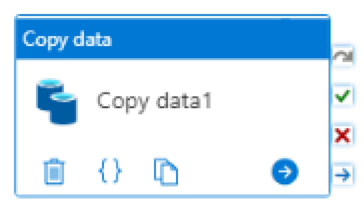
If condition

Copy Activity

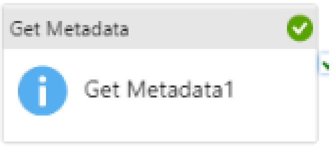


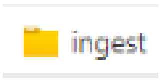


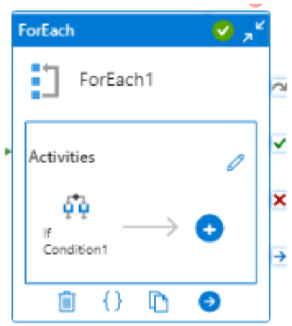


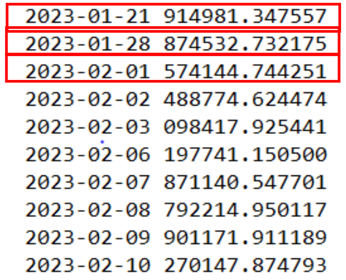


Extracting date and comparing with current date

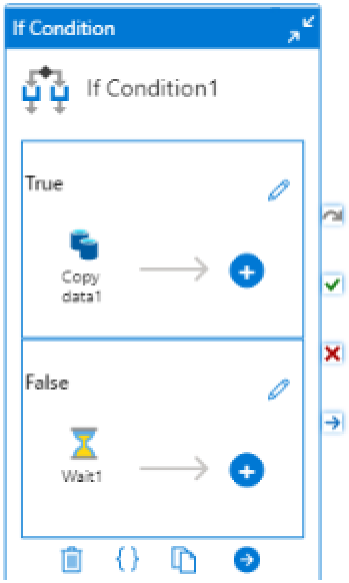




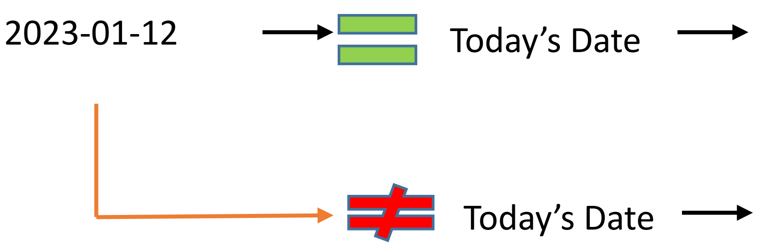




```
2023-01-21 914981.347557
2023-01-28 874532.732175
2023-02-01 574144.744251
2023-02-02 488774.624474
2023-02-03 098417.925441
2023-02-06 197741.150500
2023-02-07 871140.547701
2023-02-08 792214.950117
2023-02-09 901171.911189
2023-02-10 270147.874793
```

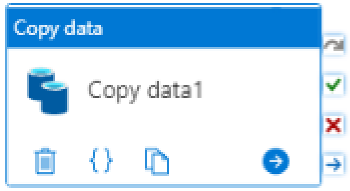


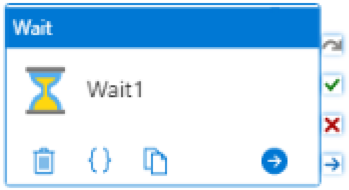
Extracts date from file name:
2023-01-12



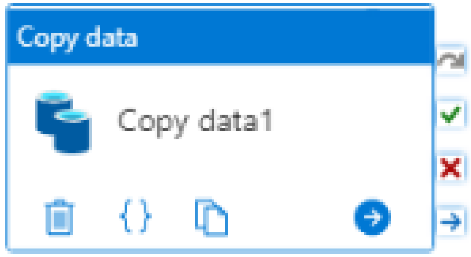
2023-01-12 → = Today's Date →

≠ Today's Date →





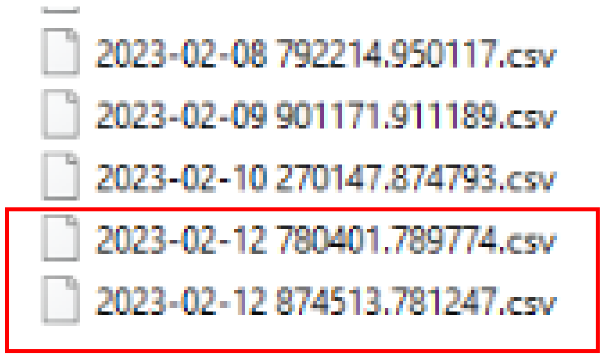
Copying files to ADLS



Today's Date: 2023-02-12

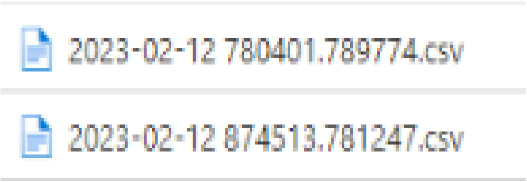
Source:

On premise Files

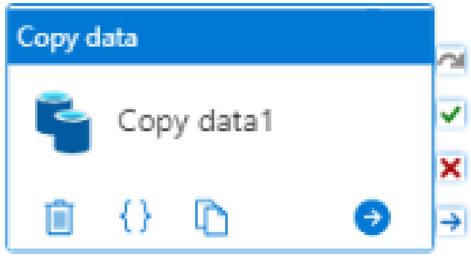

2023-02-08 792214.950117.csv
2023-02-09 901171.911189.csv
2023-02-10 270147.874793.csv
2023-02-12 780401.789774.csv
2023-02-12 874513.781247.csv

Sink: (Destination)

Azure Data Lake


2023-02-12 780401.789774.csv
2023-02-12 874513.781247.csv

Copying files to ADLS



Source:

On premise Files

Sink: (Destination)

Azure Data Lake

We are taking File names as parameters in copy activity to hold current file

Dataset:
DS_Onprem_File_Para

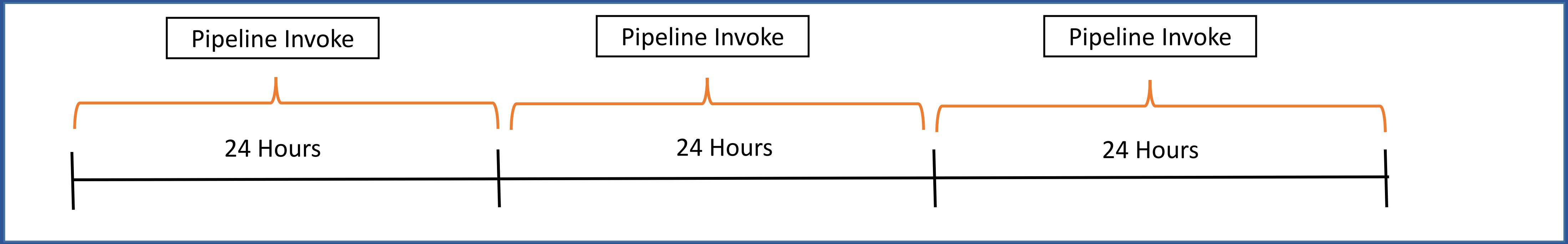
Dataset:
DS_ADLS_ingest_Para

Automating pipeline execution:

- Data is getting added on daily basis
- Pipeline need to be executed for every 24 hours
- Adding a trigger will automate the execution of pipeline
- Triggers help to invoke pipeline on a given time interval
- Automate pipeline execution with ease
- Monitor the success and failure of pipelines

Tumbling Window Trigger

- Executes data pipelines at a pre-determined periodic time interval
- Can work on historical data to copy or migrate data
- We can set a Tumbling window trigger for every 24 hours to invoke pipeline

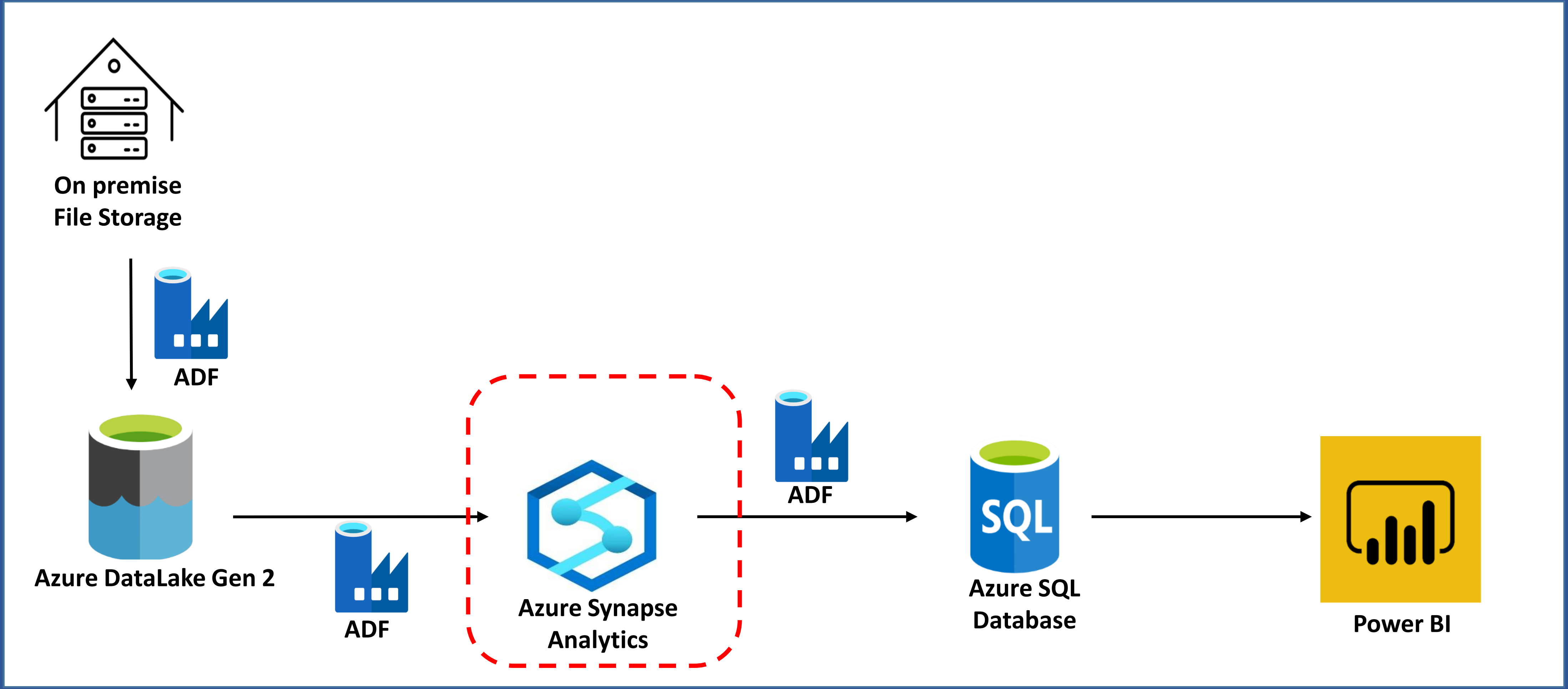


Data Transformation

(Azure Synapse Analytics)

Transformation

Transforming data using Azure Synapse Analytics



Azure Synapse Analytics

Enhanced SQL Data warehouse with Big data analytics capabilities

Components of Azure Synapse Analytics:

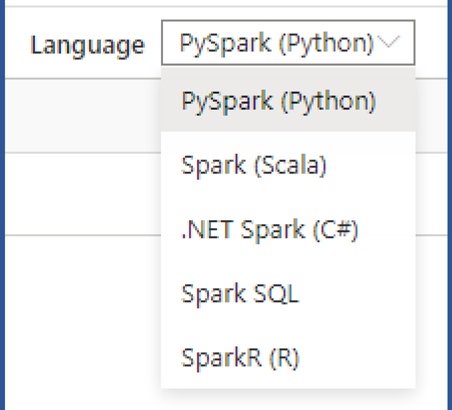
- Synapse SQL (data warehousing)
  - Dedicated SQL Pool
  - Server less SQL Pool
- Spark (Big data)
  - Apache Spark Pool
- Synapse Pipelines (For ELT, ETL)
- Deep integration (Power BI)

Synapse Notebook:

It is a web interface where we can write and execute our transformation code

Supports following languages:

- Spark (Scala)
- PySpark (Python)
- Spark SQL
- .NET Spark (C#)
- SparkR (R)

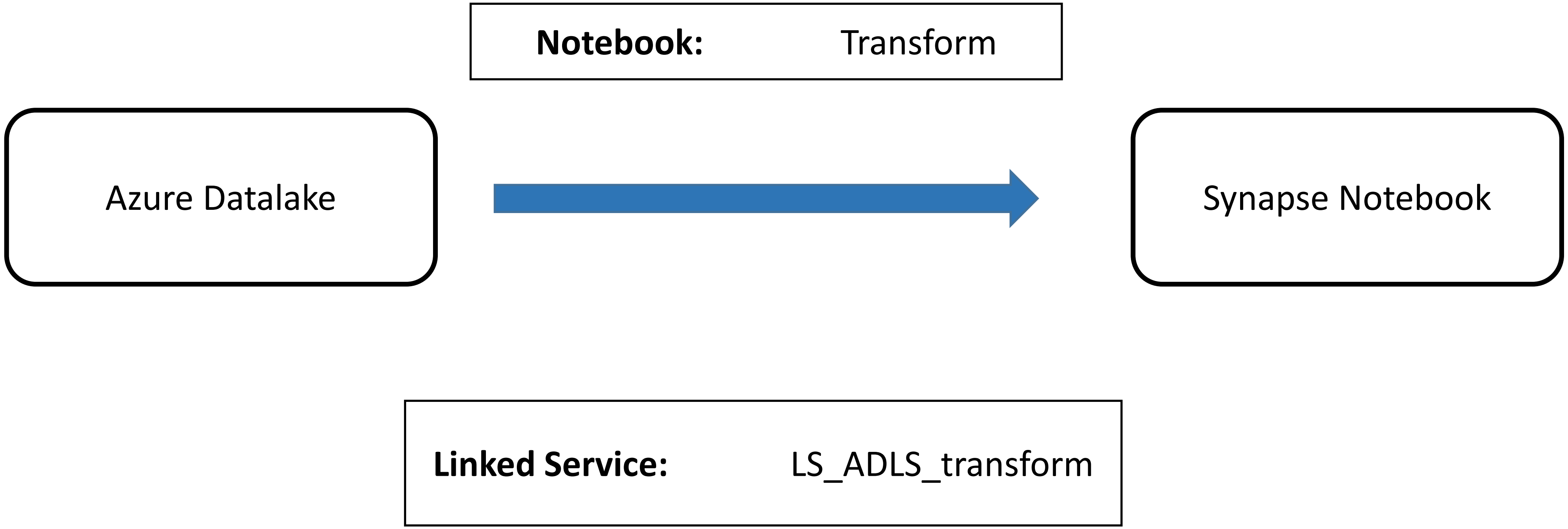


Language PySpark (Python)

Steps in Transformation of data

- Create a Synapse Notebook
- Read the data that is present in Azure Datalake
- Apply transformation logic
- Write the data to Azure Datalake

Reading ADLS data from Synapse Notebook



Notebook: Transform

Azure Datalake

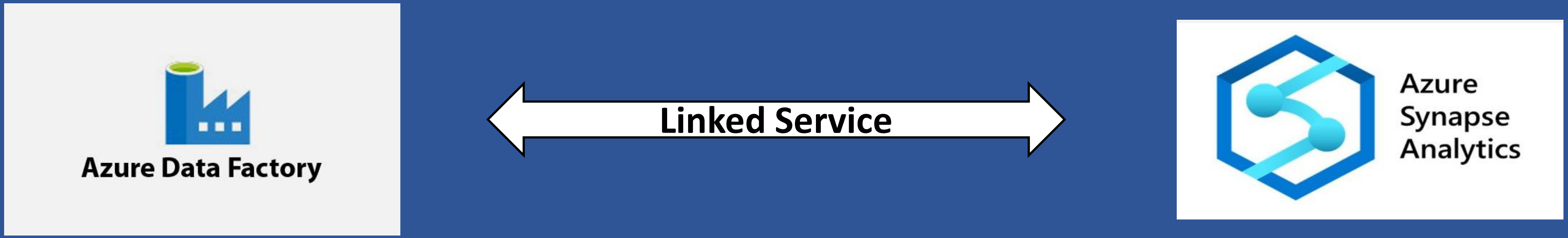
Synapse Notebook

Linked Service: LS_ADLS_transform

Transforming Data

Access to ADLS using Managed Identity

Storage Blob Data Contributor



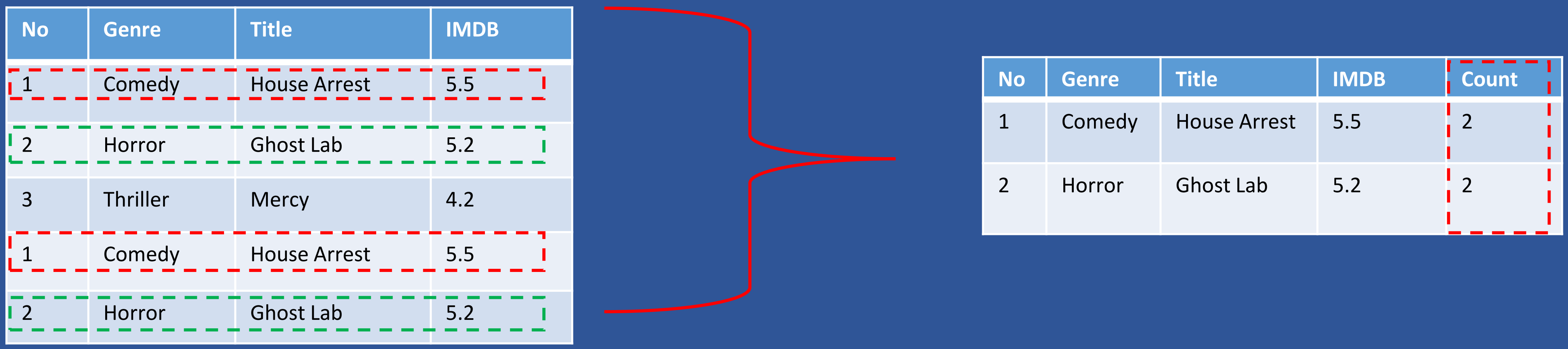
Azure Data Factory

Linked Service

Azure Synapse Analytics

Transforming Data (PySpark)

Identify and delete duplicate rows



No	Genre	Title	IMDB
1	Comedy	House Arrest	5.5
2	Horror	Ghost Lab	5.2
3	Thriller	Mercy	4.2
1	Comedy	House Arrest	5.5
2	Horror	Ghost Lab	5.2

No	Genre	Title	IMDB	Count
1	Comedy	House Arrest	5.5	2
2	Horror	Ghost Lab	5.2	2

Transforming Data (PySpark)

Identify and Replace NULL values

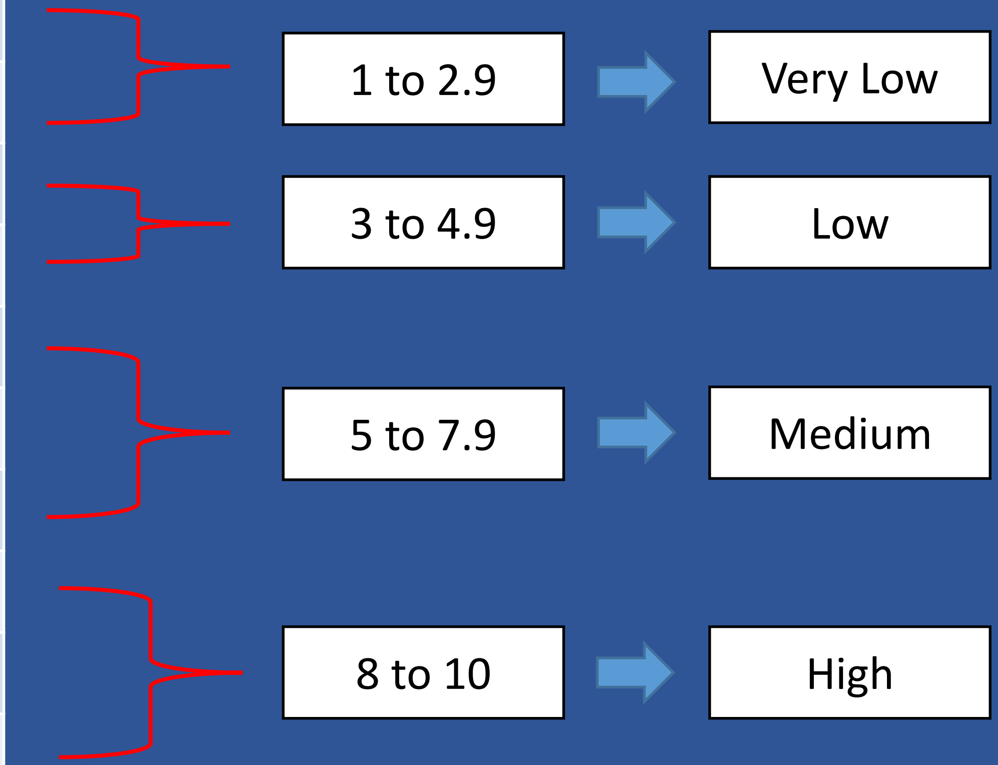
No	Genre	Title	IMDB
1	Comedy	House Arrest	5.5
2		Ghost Lab	5.2
3	Thriller	Mercy	4.2
4	Drama	Burning Sands	6.1
5		The Class Family	5.8

No	Genre	Title	IMDB
1	Comedy	House Arrest	5.5
2	**Unknown**	Ghost Lab	5.2
3	Thriller	Mercy	4.2
4	Drama	Burning Sands	6.1
5	**Unknown**	The Class Family	5.8

Transforming Data (PySpark)

Creating New column based on IMDB rating

IMDB Rating
1
2
3
4
5
6
7
8
9
10



1 to 2.9

Very Low

3 to 4.9

Low

5 to 7.9

Medium

8 to 10

High

IMBD Rating	Rating Category
1	Very Low
2.9	Very Low
3	Low
4.9	Low
5	Medium
6	Medium
7.9	Medium
8	High
9	High
10	High

Transforming Data (PySpark)

Creating New column taking runtime in “Mins” column to “Hrs”

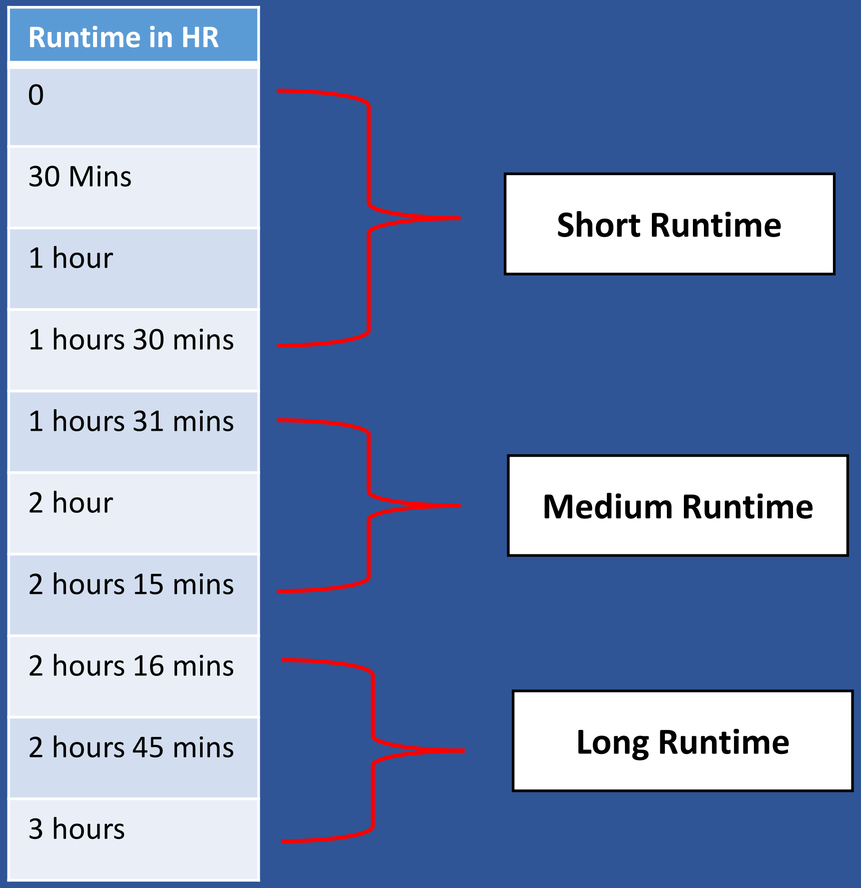
No	Genre	Title	Runtime in Mins
1	Comedy	House Arrest	104
2	Horror	Ghost Lab	117
3	Thriller	Mercy	87
4	Drama	Burning Sands	102
5	Comedy	The Class Family	98

/60

No	Genre	Title	Runtime in Hours
1	Comedy	House Arrest	1.73
2	Horror	Ghost Lab	1.95
3	Thriller	Mercy	1.45
4	Drama	Burning Sands	1.7
5	Comedy	The Class Family	1.62

Transforming Data (PySpark)

Creating New column based on Runtime in Hours

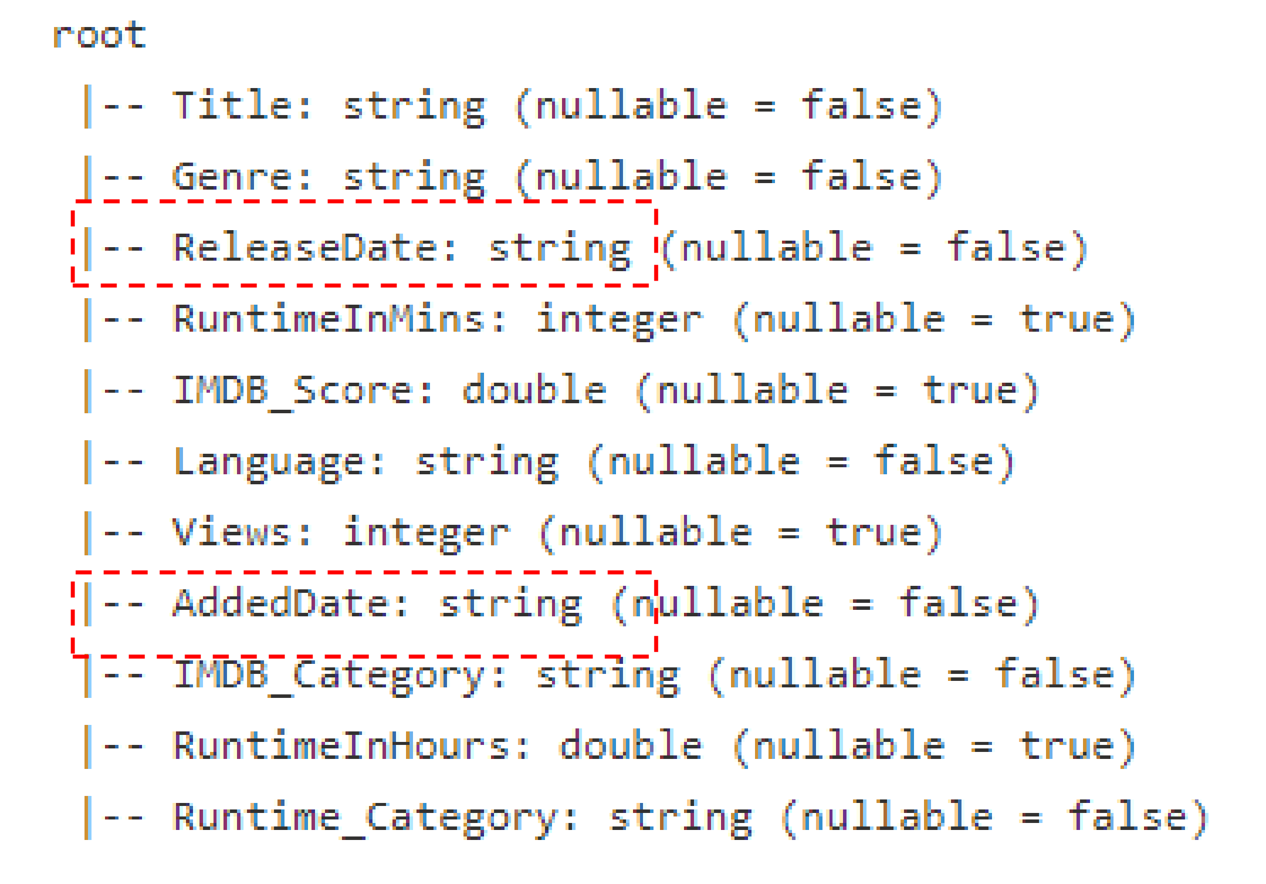


Runtime in HR
0
30 Mins
1 hour
1 hours 30 mins
1 hours 31 mins
2 hour
2 hours 15 mins
2 hours 16 mins
2 hours 45 mins
3 hours

Runtime in HR	Runtime Category
20 Mins	Short Runtime
30 Mins	Short Runtime
1 hour	Short Runtime
1 hours 30 mins	Short Runtime
1 hours 31 mins	Medium Runtime
2 hour	Medium Runtime
2 hours 15 mins	Medium Runtime
2 hour 30 mins	Long Runtime
2 hours 45 mins	Long Runtime
3 hours	Long Runtime

Transforming Data (PySpark)

Changing String Datatype to Date Datatype



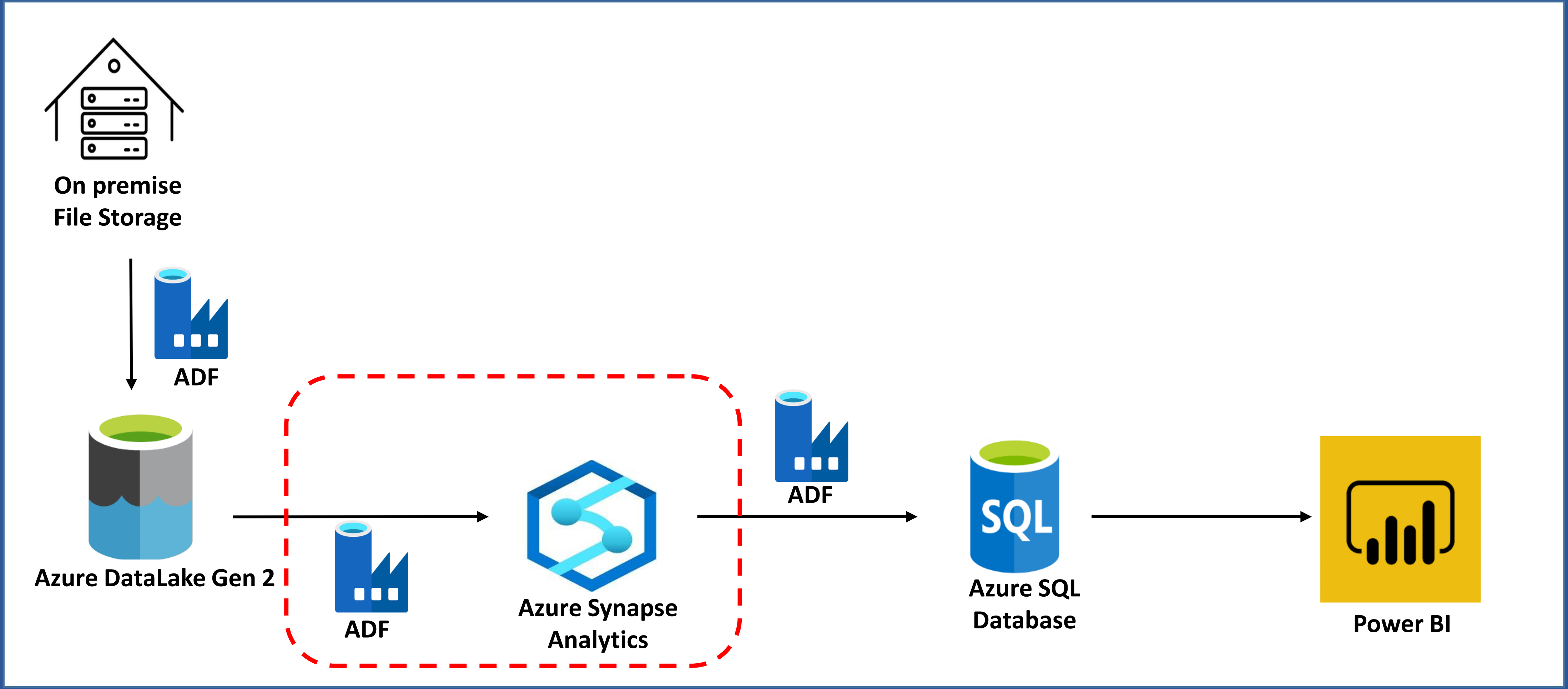
```
root
 |-- Title: string (nullable = false)
 |-- Genre: string (nullable = false)
 |-- ReleaseDate: string (nullable = false)
 |-- RuntimeInMins: integer (nullable = true)
 |-- IMDB_Score: double (nullable = true)
 |-- Language: string (nullable = false)
 |-- Views: integer (nullable = true)
 |-- AddedDate: string (nullable = false)
 |-- IMDB_Category: string (nullable = false)
 |-- RuntimeInHours: double (nullable = true)
 |-- Runtime_Category: string (nullable = false)
```

Transforming Data (PySpark)

- Final phase of transformation
- Writing transformed data to Refined Container
- We are using Datalake as Refined layer
- Storing data in parquet format
- This helps to copy the data into Azure Database using ADF

Transformation

Calling synapse notebook from Azure Data Factory



Calling Synapse notebook from Azure Data Factory

- Using Notebook Activity in Azure Data Factory (ADF)
- Both Managed Identity of ADF and User should have “Synapse Administrator” access in Synapse Analytics.

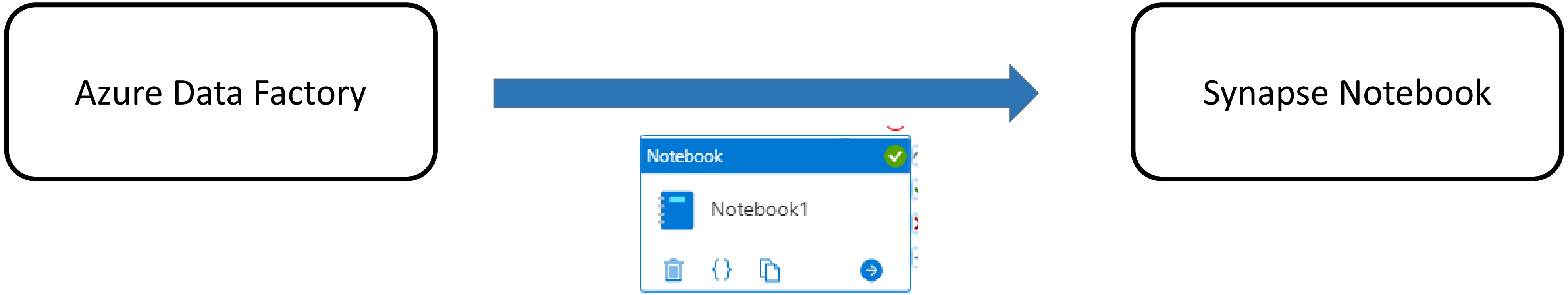
Azure Data Factory

Synapse Administrator

Azure Synapse Analytics

Calling Synapse notebook from Azure Data Factory

Pipeline: PL_Synapse_Transform



Azure Data Factory

Synapse Notebook

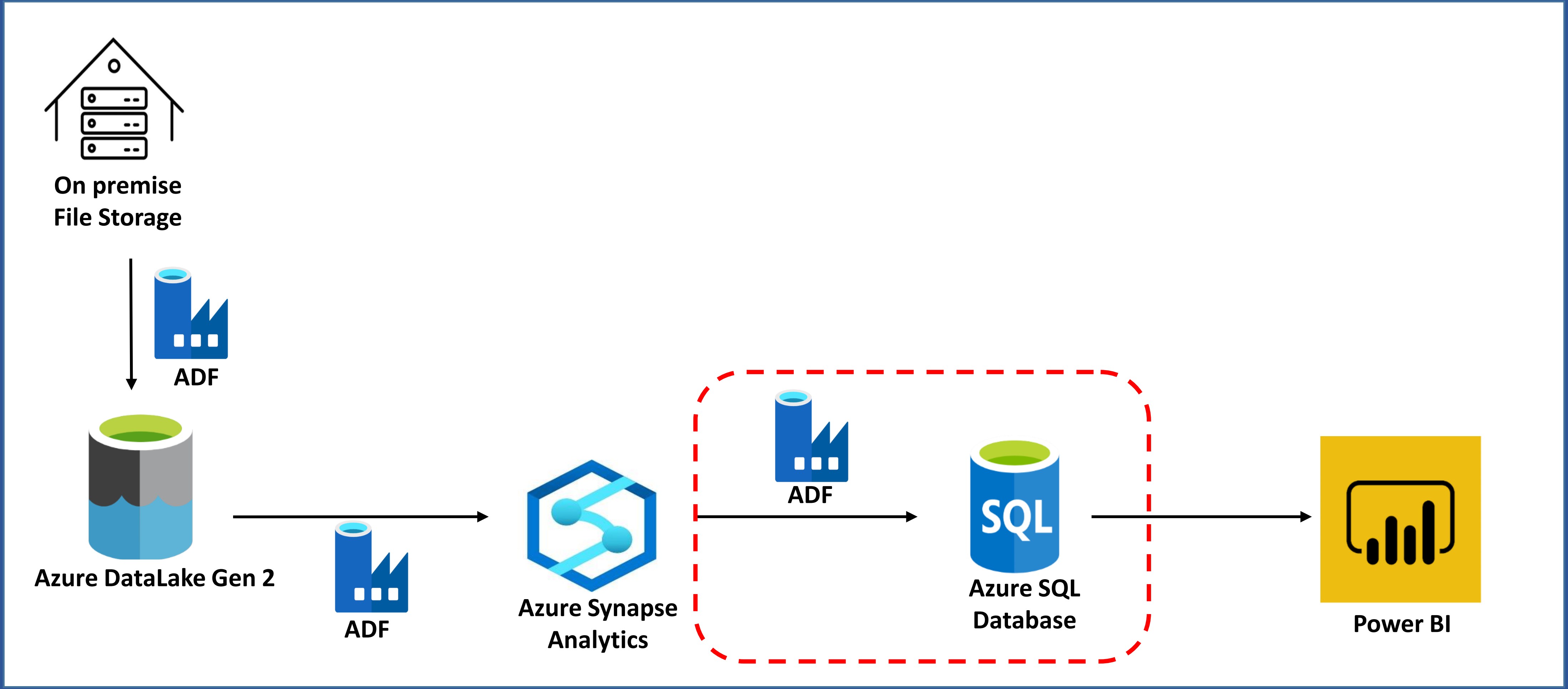
Linked Service: LS_Notebook_Transform

Data Loading

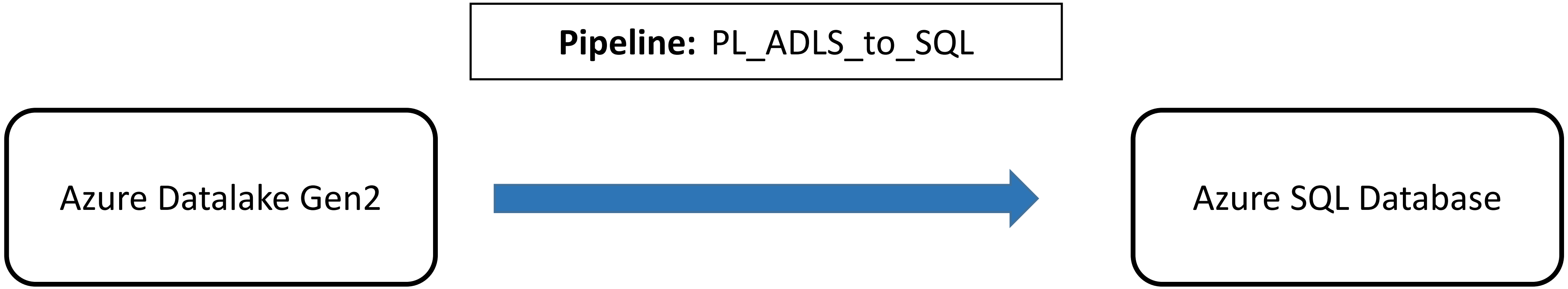
(Azure SQL Database)

Loading

Loading data into Azure SQL Database using ADF



Loading data into Azure SQL Database



Pipeline: PL_ADLS_to_SQL

Azure Datalake Gen2

Azure SQL Database

Linked Service:	LS_ADLS_Refined
Dataset:	DS_ADLS_Refined
Container:	Refined
Folder:	data

Linked Service:	LS_SQL_Load
Dataset:	DS_SQL_Load

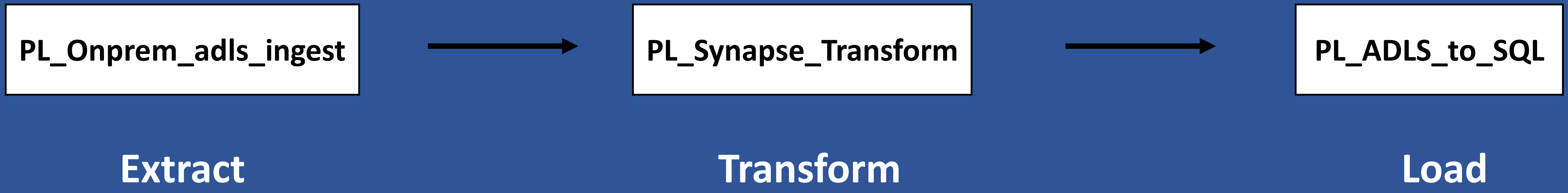
Data Loading

(Conclusion)

Orchestration

(Intro)

Orchestrating all the pipelines

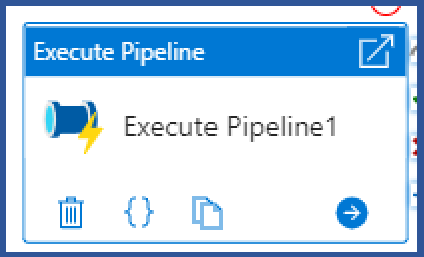


PL_Onprem_adls_ingest → PL_Synapse_Transform → PL_ADLS_to_SQL

Extract

Transform

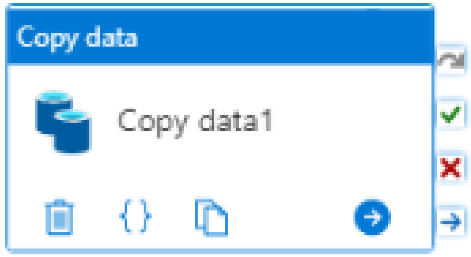
Load



Enhancements for Copy to ADLS from On-premise

- Copy today’s file to a specific folder in ADLS while ingesting from on-premise
- Folder name should be Today’s date (Same date as file)
- All files of today’s date will be stored in a folder that have today’s date.

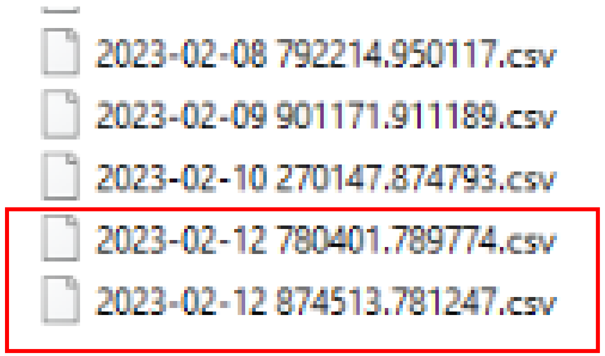
Enhancements - Copying files to ADLS



Today's Date: 2023-02-12

Source:

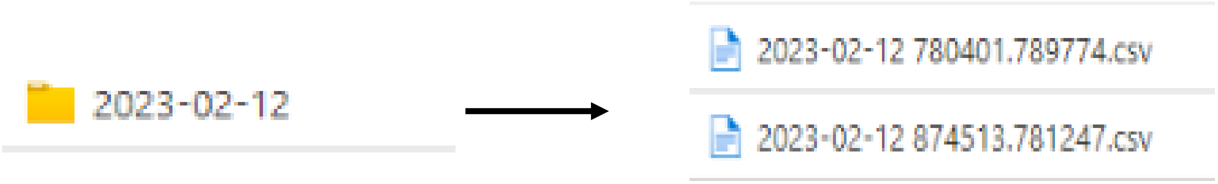
On premise Files



2023-02-08 792214.950117.csv
2023-02-09 901171.911189.csv
2023-02-10 270147.874793.csv
2023-02-12 780401.789774.csv
2023-02-12 874513.781247.csv

Sink: (Destination)

Azure Data Lake



2023-02-12

2023-02-12 780401.789774.csv
2023-02-12 874513.781247.csv

Enhancements to Synapse notebook

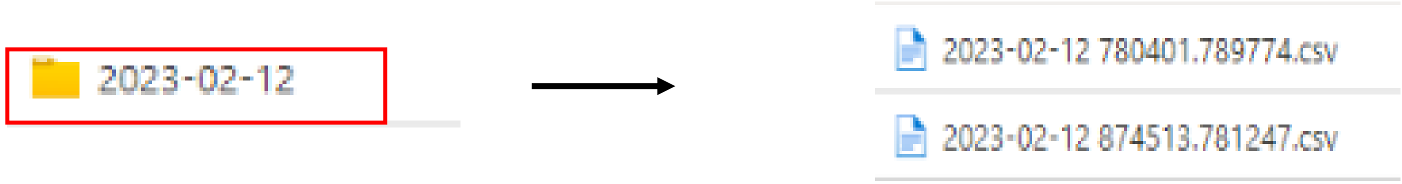
- Transformation will be done only to that particular file as folder will have only today’s file
- Compute resources can be saved in Spark
- Execution time will be reduced as it will computes only today’s rows

Enhancements - Taking Source Path in Synapse Notebook

Today's Date: 2023-02-12

Source path:

raw/ingest/{Today's Date}/



2023-02-12

2023-02-12 780401.789774.csv

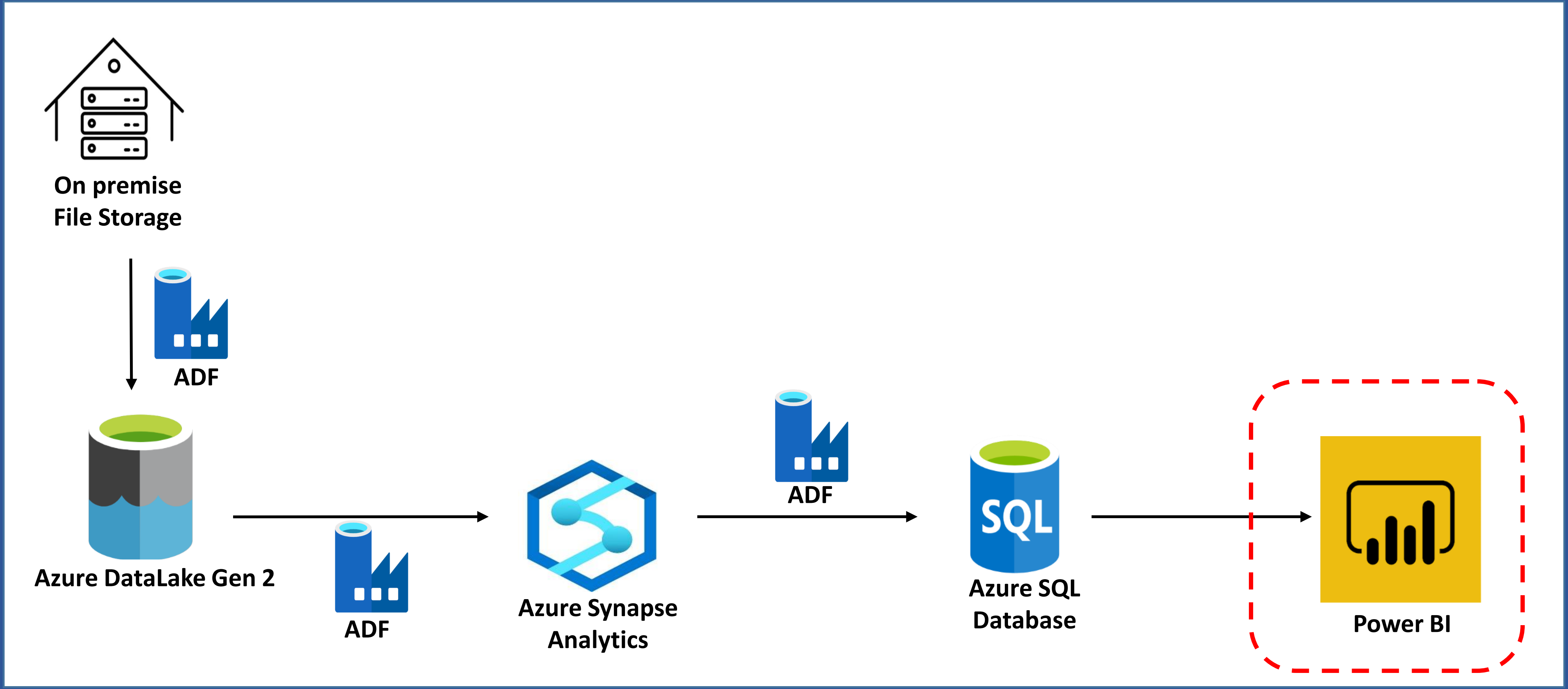
2023-02-12 874513.781247.csv

Reporting

(Power BI)

Reporting

Reporting data to Power BI

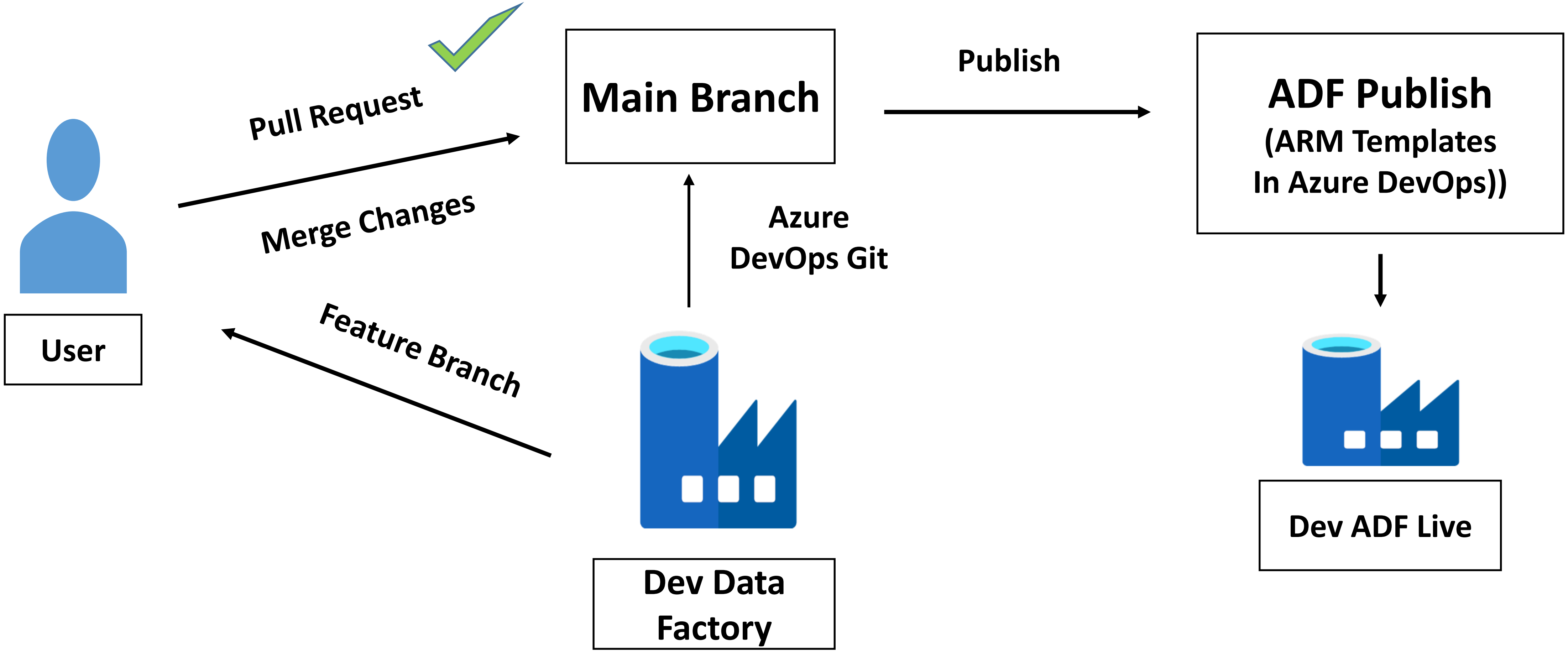


Get Data

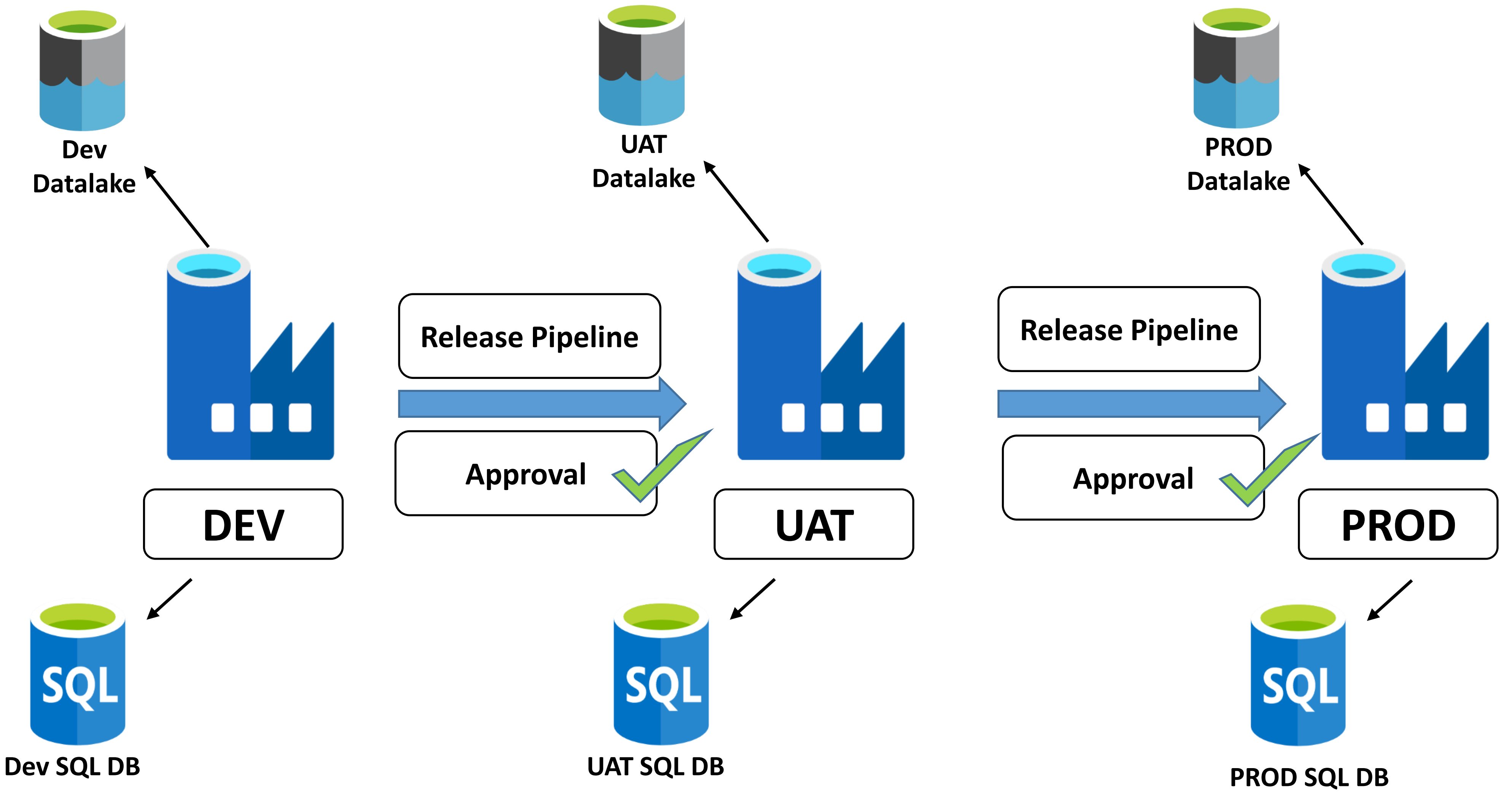
- Open Power BI Desktop
- Get data from Azure SQL Database from Power BI
- Authenticate
- Load data to Model

Bonus Section
Azure Data Factory – CICD Setup

Continuous Integration



Continuous Deployment



Dev Datalake

UAT Datalake

PROD Datalake

Release Pipeline

Approval

Release Pipeline

Approval

DEV

UAT

PROD

Dev SQL DB

UAT SQL DB

PROD SQL DB

Conclusion