# AZURE DATA FACTORY

1. Introduction to Azure Data Factory
   1. Overview of Azure Data Factory
   2. ADF architecture and components
   3. ETL (Extract, Transform, Load) and ELT (Extract, Load, Transform) processes
2. Pipelines
   1. Definition and purpose of pipelines
   2. Creating and managing pipelines
   3. Triggering and scheduling pipelines
   4. Debugging and monitoring pipelines
3. Activities in Azure Data Factory
   1. Data movement activities: Copy activity
   2. Data transformation activities: Mapping Data Flows, Data Bricks, Hive, Stored Procedure
   3. Control flow activities: ForEach, If Condition, Wait, Switch, Execute Pipeline, and Web activity
   4. Chaining activities in a pipeline
4. Linked Services
   1. Creating and configuring linked services
   2. Understanding data sources and destinations (Azure SQL Database, Blob Storage, Data Lake, On-premises databases, REST APIs, etc.)
5. Datasets
   1. Dataset concept in ADF
   2. Mapping datasets to data stores
   3. Dataset properties and schema definition
6. Triggers
   1. Schedule triggers
   2. Event-based triggers (e.g., when files are created or modified in Blob Storage)
   3. Tumbling window triggers
7. Integration Runtime (IR)
   1. Types of Integration Runtimes: Azure IR, Self-hosted IR, and Azure-SSIS IR
   2. When to use self-hosted vs. Azure IR
   3. Setting up and managing IR
8. Data Flows (Mapping Data Flows
9. & Wrangling Data Flows)
   1. Creating and configuring mapping data flows
   2. Transformations: join, aggregate, filter, pivot, sort, etc.
   3. Debugging and optimizing data flows
10. Monitoring and Debugging
    1. Monitoring pipeline runs
    2. Understanding the activity logs
    3. Handling errors and retries in pipelines.
11. Parameterized Pipelines
    1. Using parameters in pipelines, datasets, and linked services
    2. Expressions and variables
    3. Passing parameters between pipelines and activities.
12. Security
    1. Managing access via Azure Active Directory (AAD) and role-based access control (RBAC)
    2. Setting up Azure Key Vault for secure storage of secrets (e.g., database credentials)
    3. Data encryption and compliance
13. Integration with Other Azure Services
    1. Integration with Azure Data Lake Storage, Azure SQL Database, and Blob Storage
    2. Using Azure Logic Apps for automation
    3. Integration with Azure Databricks, HDInsight, and Azure Synapse Analytics
14. Git Integration and CI/CD
    1. Git integration for version control in Azure Data Factory
    2. Setting up CI/CD pipelines using Azure DevOps or GitHub for ADF
15. Data Migration
    1. Best practices for migrating data using Azure Data Factory
    2. Moving data between on-premises and cloud environments
16. Costs and Optimization
    1. ADF pricing model (data movement, activity runs, etc.)
    2. Cost optimization techniques in Azure Data Factory
17. Azure Synapse Analytics Integration
    1. Using Azure Data Factory with Synapse Pipelines
    2. Integrating with Synapse for big data processing

---------------------------------------------------------------------------------------------------------------------

**why do we need azure data factory?**

1. **It** is a fully managed, cloud-based data integration service by Microsoft that enables you to orchestrate and automate data movement and transformation workflows at scale.
2. It is primarily used for **extract, transform, and load (ETL)** and **extract, load, and transform (ELT)** processes.
3. It Supports **90+ connector** for data sources.

**1. What are the key components of Azure Data Factory?**

* Pipelines
* Datasets
* Linked Services
* Triggers
* Integration Runtimes

Azure Data Factory (ADF) is a cloud-based data integration service that enables the creation of data-driven workflows for orchestrating and automating data movement and transformation. It allows you to create, schedule, and monitor data pipelines that ingest, prepare, and transform data from different sources.

Here are the **key components** of Azure Data Factory explained in detail, along with examples:

Pipeline, Activities, Datasets, Linked Services, Integration Runtime(IR), Triggers, Dataflow, Monitoring, Parameters and Variables

**1. Pipelines**

A **Pipeline** is a logical grouping of activities that perform a specific data integration task. Each pipeline can include one or more activities, and it allows for data movement, data transformation, and control flow operations.

**Example:** Imagine you have a pipeline to copy data from an on-premises SQL Server database to Azure Blob Storage and then process the data using Azure Databricks. In this pipeline:

* The **Copy Activity** moves data from SQL Server to Azure Blob Storage.
* The **Databricks Activity** processes the data using Databricks Notebooks.

Pipeline: SalesDataPipeline

Activities:

- Copy data from on-prem SQL to Azure Blob (Copy Activity)

- Run transformation in Azure Databricks (Databricks Activity)

**2. Activities**

Activities represent the work to be performed in a pipeline. There are three main types of activities:

* **Data Movement Activities**: Copy data from one place to another (e.g., Copy Activity).
* **Data Transformation Activities**: Perform operations such as data cleansing or transformation using services like Data Flows, Azure Databricks, or SQL Stored Procedures.
* **Control Activities**: Manage the flow of data by using control logic such as ForEach, If Condition, or Wait.

**Example:** A pipeline that copies files from an FTP server to Azure Blob Storage and then processes the files using an Azure Data Flow.

Activity 1: Copy Activity (FTP to Blob Storage)

Activity 2: Data Flow Activity (Data Transformation in Mapping Data Flow)

**3. Datasets**

A **Dataset** represents the data structure (such as tables, files, or blobs) that you intend to work with. It defines the schema and location of the data and serves as a reference to data within a linked service.

**Example:** Consider a dataset that refers to a SQL Server table containing customer information.

Dataset:

Name: CustomerDataset

Type: SQL Server Table

Linked Service: OnPremisesSQLLinkedService

Table Name: dbo.Customers

This dataset might then be used in the Copy Activity to move data from the Customers table to Azure Blob Storage.

**4. Linked Services:**

A **Linked Service** defines the connection to a data store or a compute service (such as SQL Server, Azure Blob Storage, or an FTP server). It contains the connection details like the data store URL, authentication type, and credentials.

**Example:** A linked service connecting to an Azure SQL Database might include the server name, database name, and authentication details (either SQL authentication or Azure Active Directory).

Linked Service:

Name: AzureSQLLinkedService

Type: Azure SQL Database

Connection String: Server=tcp:mydbserver.database.windows.net;Database=mydb

Authentication: Azure Active Directory

This linked service would be referenced by a dataset to interact with the Azure SQL Database.

**5. Integration Runtime (IR):**

The **Integration Runtime (IR)** is the compute infrastructure used by Azure Data Factory to perform data movement and transformation. There are three types of Integration Runtimes:

* **Azure Integration Runtime**: Used for cloud-based data movement and transformations.
* **Self-Hosted Integration Runtime**: Used for data movement between on-premises data stores and cloud, or between two on-premises data stores.
* **Azure-SSIS Integration Runtime**: Used to run SQL Server Integration Services (SSIS) packages in Azure.

**Example:** If you need to copy data from an on-premises SQL Server to Azure Blob Storage, you would use a **Self-Hosted Integration Runtime** installed on a local machine or virtual machine within your on-premises environment.

Integration Runtime:

Name: SelfHostedIR

Type: Self-Hosted

Machine: OnPremServer

**6. Triggers**

Triggers in Azure Data Factory determine when a pipeline should be executed. ADF provides several types of triggers:

* **Schedule Trigger**: Executes pipelines at specific intervals (daily, weekly, etc.).
* **Tumbling Window Trigger**: Executes pipelines in fixed-size windows without overlap. A trigger that operates on a periodic interval, while also retaining its state.
* **Event-Based Trigger**: Executes pipelines based on events such as the creation or deletion of files in Azure Blob Storage.

**Example:** A **Schedule Trigger** that runs a pipeline every day at 8:00 AM to copy sales data from an FTP server to Azure Blob Storage.

Trigger:

Name: DailySalesTrigger

Type: Schedule

Frequency: Daily

Time: 8:00 AM

**7. Data Flows**

**Multiple inputs/outputs:-** Join, Conditional Split, exists, union, Lookup

**Schema Modifier:-**Derived Column, select, Aggregate, Surrogate key, pivot, unpivot, window, rank, external call, cast

**Formatter:-**Flatten, Parse, stringify

**Row Modifier:-**filter, sort, alter row, assert

**Flowlet:-**Flowlet

**Destination:-** Sink

**surrogate key transformation** is used to add an incrementing key value to each row of data. This is useful when designing dimension tables in a star schema analytical data model. In a star schema, each member in your dimension tables requires a unique key that is a non-business key.

**External Call** refers to invoking an external system or service outside the ADF environment like HTTP or API calls to interact with your data flow. This is commonly done through **Web activities** or **Stored Procedure** activities, which can integrate with other systems such as APIs, databases, or even custom services. There are ways to make **external calls** in **Azure Data Factory**:-

* **Using web activity transformation in adf pipeline.**
* **Using external call transformation in dataflow**
* **Using Custom activities:-** Package your logic into a custom script (Python, .NET, etc.) and store it in Azure Blob Storage. ADF executes the custom activity, and you can have it interact with external services through the logic you implement in your script.

**Mapping Data Flows** allow you to visually design data transformations in Azure Data Factory without writing code. You can use Data Flows to perform operations like filtering, joining, aggregating, and sorting data.

**Example:** A data flow that cleanses customer data by:

* Filtering out customers without an email address.
* Aggregating customer transactions by customer ID.
* Sorting customers by their total purchase amount.

Data Flow:

Name: CleanseCustomerDataFlow

Transformations:

- Filter: Customers with missing email

- Aggregate: Total purchases by customer ID

- Sort: By total purchases

**There are 4 states of flow of tasks in ADF:-**

 **Success**: Activity runs as expected.

 **Failure**: Activity encounters an error, and the pipeline may take remedial action.

 **Completion**: Activity finishes, regardless of the outcome (success or failure).

 **Skip**: Activity is intentionally skipped, often based on conditions.

**Different ways to avoid failure in pipelines:-**

**1.Degree of copy parallelism:-** It determines how many parallel copies or threads ,ADF will use to transfer data, both within the source and the destination. There are two main ways to control the degree of parallelism:

1. Partitioning (Source-side parallelism):Range, Round-Robin, Hash
2. Degree of Parallelism in the Copy Activity Settings(0-32)

**2. Data consistency verification:** copy activity will do additional data consistency verification between source and destination store after data movement. The verification includes file size check and checksum verification for binary files, and row count verification for tabular data.

3. Timeout

1. Retry(maxm. No. of retry)
2. Fault tolerance can implement these condition to avoid failure in pipelines

* Skip incompatible rows
* Skip missing files
* Skip forbidden files
* Skip files with invalid names

**8. Monitoring**

Azure Data Factory provides a **Monitoring** tool to track the execution of pipelines and activities. You can view detailed logs of pipeline runs, activity runs, and integration runtimes to troubleshoot and optimize performance.

**Example:** You run a pipeline to copy large datasets from a SQL database to Azure Data Lake Storage. The **Monitor** tab provides a view of:

* Successful activity runs.
* Failed activity runs, along with error details.
* Execution times and performance metrics.

Monitoring:

Pipeline: DataLoadPipeline

Status: Completed (Successful)

Duration: 00:05:30

**9. Parameters and Variables**

* **Parameters**: Allow you to pass dynamic values into a pipeline or activity at runtime. For example, you can pass different filenames, dates, or other configuration values. Global parameters can be used for multiple pipeline.
* **Variables**: Store values during pipeline execution and can be updated dynamically using activities like the **Set Variable** activity.

**Example:** You have a pipeline that copies files based on the current date. You can create a parameter FileDate and pass it to the Copy Activity to dynamically pick up the correct file.

Parameters:

Name: FileDate

Value: 2024-09-12

Variables:

Name: ProcessedRecords

Type: Integer

**Summary of Key Components with Example Use Case**

In a common scenario:

1. A pipeline (SalesDataPipeline) is scheduled to run every day at 8:00 AM.
2. A **Copy Activity** copies sales data from an on-premises SQL Server to Azure Blob Storage, using a **Self-Hosted Integration Runtime**.
3. The data is then cleaned and transformed using a **Mapping Data Flow**.
4. A **Dataset** represents the source SQL Server table (SalesData), and a **Linked Service** provides the connection details for both SQL Server and Blob Storage.
5. Monitoring tools are used to track the pipeline's success or failure.

A **Resource Group** in **Azure Data Factory** is a logical container that holds related resources for an Azure solution. It helps to manage and organize all the resources involved in your data integration workflows, such as pipelines, linked services, datasets, and other associated Azure resources.

#### ****Resources in an Azure Data Factory Resource Group****

When using Azure Data Factory, you often work with various resources, such as:

* **Data Factory instance**: The main resource for orchestrating data workflows.
* **Linked Services**: Connection strings to data sources or destinations (e.g., Azure Blob Storage, Azure SQL Database).
* **Datasets**: Metadata definitions that represent data in the source or sink.
* **Pipelines**: Collections of activities for moving and transforming data.
* **Integration Runtime**: The compute infrastructure used for data movement and transformation.
* **Other Azure Resources**: Databases, storage accounts, and Key Vaults used within the data factory pipelines.

Different storage types supported in **Azure Data Factory (ADF)**:

| **Storage Type** | **Description** | **Typical Use Case** | **Example Data Formats** |
| --- | --- | --- | --- |
| **Azure Blob Storage** | Scalable object storage for unstructured data. | Store raw data files or backup data. | CSV, JSON, Parquet, Avro, Binary |
| **Azure Data Lake Storage (Gen1 & Gen2)** | Hierarchical storage for large-scale analytics workloads. | Big data processing and storage for analytics. | CSV, JSON, Parquet, Avro, ORC |
| **Azure SQL Database** | Fully managed relational database in the cloud. | Store structured data, such as cleaned data for reporting. | Tables (Structured Data) |
| **Azure SQL Managed Instance** | Managed SQL Server instance with more control. | Enterprise-scale relational databases. | Tables (Structured Data) |
| **Azure Synapse Analytics** | Enterprise-level data warehouse for big data and analytics. | Analytical workloads and large data storage. | Structured data (SQL) |
| **Azure Cosmos DB** | Globally distributed, multi-model NoSQL database. | Store semi-structured NoSQL data for web and mobile apps. | JSON (NoSQL) |
| **Azure Table Storage** | NoSQL key-value store for structured, non-relational data. | Storing key-value pairs and logs. | Key-Value pairs |
| **Azure File Storage** | Fully managed file shares in the cloud via SMB protocol. | File-based workloads, such as backups or file storage. | Binary, File data |
| **Azure Queue Storage** | Service for storing large numbers of messages. | Queue messages for event-driven processing. | Queue Messages |
| **Azure Data Explorer** | Fully managed service for real-time data ingestion and querying. | Analyzing large volumes of log and telemetry data in real time. | Logs, Telemetry |
| **On-Premises Storage** | On-premises storage accessed via Self-hosted Integration Runtime. | Hybrid data integration with on-premises systems. | Files, Tables (Structured/Unstructured Data) |
| **Amazon S3 (via connector)** | Scalable object storage from Amazon Web Services. | Cross-cloud data movement and integration. | CSV, JSON, Parquet, Avro, Binary |
| **Google Cloud Storage (via connector)** | Scalable object storage from Google Cloud Platform. | Cross-cloud data movement and integration. | CSV, JSON, Parquet, Avro, Binary |
| **SFTP (via connector)** | Secure File Transfer Protocol for transferring files. | File transfers with secure encryption. | Binary, Text Files |

**Variables in Azure Data Factory:-**

ADF variables can be used to store:

* **String**: Stores text values.
* **Boolean**: Stores true or false.
* **Array**: Stores lists of items.

**We can set or update a variable value using Set Variable and Append Variable activities.**

* **Set Variable**: Sets a variable to a new value.
  + Add a **Set Variable** activity to your pipeline.
  + In the **Settings** tab, choose the variable name from the dropdown and enter the value you want to set.
* **Append Variable**: Adds a value to an array variable.
  + Add an **Append Variable** activity to your pipeline.
  + In the **Settings** tab, select the array variable and specify the value to append.

**Key Variable Functions in ADF**

| **Function** | **Description** |
| --- | --- |
| **Set Variable** | Assigns a value to a variable. |
| **Append Variable** | Adds a value to an array variable. |
| **@variables('name')** | Retrieves the value of a variable. |
| **@equals, @not, @and** | Used to evaluate variables in conditions. |

**System variables in Azure Data Factory:-**

**System Variables in ADF**

| **Category** | **Variables** | **Use Case** |
| --- | --- | --- |
| **Pipeline** | pipeline().PipelineName, pipeline().RunId, Pipeline().TriggerName, Pipeline().TriggerTime, Pipeline().DataFactory | Metadata about pipeline execution. |
| **Trigger** | trigger().Name, trigger().Time, trigger().Type, trigger().ScheduledTime | Information about the trigger that started the pipeline. |
| **Activity** | activity('activityName').Output, activity('activityName').Error | Details about specific activity execution. |
| **Run** | run().Start, run().End, run().Status | Timestamp of pipeline run. |

Here’s a comparison of **variables** and **parameters** in Azure Data Factory (ADF) in a tabular format:

| **Aspect** | **Parameters** | **Variables** |
| --- | --- | --- |
| **Definition** | Values defined at the start of a pipeline run, used to pass external inputs. | Temporary storage within a pipeline for dynamic values. |
| **Scope** | Available at **Pipeline**, **Dataset**, and **Data Flow** levels. | Available at **Pipeline** level only. |
| **Mutability** | **Immutable** (set once when pipeline starts and cannot be changed). | **Mutable** (can be changed during pipeline execution). |
| **Usage** | Pass static configurations or external inputs. | Store intermediate values or states within the pipeline. |
| **Typical Use Cases** | File path, date range, configuration settings. | Loop counters, flags, temporary values. |
| **Set/Update Method** | Defined in pipeline trigger or passed as input to datasets/data flows. | Updated using **Set Variable** or **Append Variable** activities. |
| **Dynamic Usage** | Cannot be changed after being set, used for constant values. | Can be updated dynamically, used in conditional logic. |
| **Example** | "@pipeline().parameters.filePath" | "@variables('counter')" |
| **Limitations** | Cannot store changing values during pipeline execution. | Cannot be accessed outside of their pipeline. |

**2. Explain the difference between ETL and ELT in the context of Azure Data Factory.**

* ETL: Data is extracted, transformed, and then loaded.
* ELT: Data is extracted, loaded into the destination, and transformed within the target system.

**3. What is a Pipeline in Azure Data Factory, and how does it function?**

* A pipeline is a logical grouping of activities that performs a unit of work. It allows for orchestration of activities such as data movement, transformation, and control flows.

**4. What are Linked Services in Azure Data Factory, and why are they important?**

* Linked Services define the connection information to external resources like databases, cloud storage, etc. They enable communication between ADF and the connected resources.

**5. What is an Integration Runtime (IR) in ADF, and what types are available?**

* **Azure Integration Runtime**: Used for cloud-based data movement and transformation.
* **Self-hosted Integration Runtime**: Used for on-premises data sources.
* **Azure-SSIS Integration Runtime**: For running SSIS packages.

**6. How do you set up and use a Self-Hosted Integration Runtime in ADF?**

* Install a self-hosted IR on an on-premises machine or VM, register it with ADF, and configure it to interact with on-premises resources.
* **Explain the different types of Triggers in Azure Data Factory.**

Triggers are used to invoke pipelines and schedule the execution of data workflows based on different criteria.

### Schedule Trigger: Trigger pipelines at scheduled intervals.

* **Purpose**: Execute pipelines at predefined time intervals.
* **It can associate with multiple pipeline**.
* **How it Works**: You specify a start date, an end date, and a recurrence frequency (e.g., daily, weekly, hourly).
* **Use Case**: Automating daily or weekly data processing jobs, such as a nightly batch job that loads data from a database into a data warehouse.
* You can set the time zone for the trigger.
* Supports recurrence frequencies such as minutes, hours, days, weeks, and months.
* Can be paused and resumed at any time.
* Support all kind of Data Source and **there can be overlap**.

**Tumbling Window Trigger**: It runs a pipeline every hour to process the last hour’s log data. Each hour is a separate window of data that is processed.Trigger at fixed intervals without overlap.

* **Purpose**: Execute pipelines at fixed intervals without overlap, creating time-based windows for data processing.
* **How it Works**: You define a time window (e.g., every hour, day, or week) and pipelines are triggered at the start of each window. Once the window closes, the next one opens.
* **Use Case**: Processing time-sliced data, such as hourly IoT data ingestion, where each batch of data is processed separately for a specific time window.
* **Non-overlapping Windows**: Each window represents a specific period, and pipelines are triggered only once per window.
* **Window Size**: You can set the size of the window (e.g., 1 hour, 1 day).
* **Dependency Handling**: A pipeline can be set to depend on the successful completion of the previous window’s execution.
* Supports retry policies in case of failures.
* Provides advanced scheduling options for missed or late windows (e.g., catching up on missed windows if the trigger was paused).
* Processing real-time IoT data every 10 minutes to aggregate sensor data.
* Hourly processing of streaming data in time-based batches for analysis.
* Support all kind of Data Source

**We can update Following Retry Policy to avoid trigger failure in tumbling window:-**

1. Retry Policy: Count(number of retry if failed)
2. Retry Policy: interval in second(waiting time)
3. Max Concurrency

**Event-Based Trigger**: Trigger pipelines based on file creation or deletion in storage.

* **Purpose**: Trigger pipelines based on events in Azure storage, such as the arrival of a new file or the deletion of an existing file.
* **How it Works**: The trigger listens for events on Azure Blob Storage or Data Lake Storage (Gen2), and when a specified event occurs (e.g., file creation, modification, or deletion), the pipeline is triggered.
* **Use Case**: Event-driven data processing, such as triggering a pipeline when a new CSV file is uploaded to Azure Blob Storage for real-time data processing.
* **Blob Events**: Triggers based on events like **Blob Created or Blob Deleted.**
* **File or Folder Specific**: You can set the trigger to monitor a specific folder or file pattern (e.g., all .csv files in a folder).
* **Real-Time Data Processing**: Useful for real-time or near-real-time ingestion and processing of data as files arrive.
* Can also work with hierarchical namespaces in Data Lake Storage Gen2.
* Only supports Azure Blob Storage or Azure Data Lake.

**How Event-Based Triggers Work:**

* **Event Grid Integration**: Event-based triggers in ADF are powered by **Azure Event Grid**, which listens for events (file creation or deletion) in Azure Blob Storage or Data Lake Storage and pushes these events to ADF.
* **File or Folder Specific**: You can configure the trigger to respond to events happening within a specific folder or match certain file patterns (e.g., trigger the pipeline only for .csv files).
* **Retry Policies**: You can set retry policies to handle transient failures when the trigger attempts to start the pipeline.

**8. What is a Mapping Data Flow, and how is it different from the Copy Activity?**

* Mapping Data Flow: Performs data transformation operations such as filter, join, aggregate, and derived columns.
* Copy Activity: Primarily used to copy data from one source to another without heavy transformation.

### ****1. Mapping Data Flow****

A **Mapping Data Flow** in Azure Data Factory allows you to visually design and execute data transformation logic without writing code. It is a low-code/no-code environment for performing complex data transformations, similar to ETL (Extract, Transform, Load) operations. **Mapping Data Flows are designed for data preparation, cleaning, aggregation, joining, and filtering before loading it into a destination.**

#### Key Features of Mapping Data Flow:

* **Visual Design**: You create data transformation logic using a visual interface with drag-and-drop components.
* **Transformation Operations**: Mapping Data Flows support a wide range of transformations, including:
  + **Joins**: Merge two or more datasets based on matching keys.
  + **Aggregations**: Perform sum, average, min, max, etc.
  + **Sorting**: Order data based on specified columns.
  + **Derived Columns**: Create new columns or modify existing ones.
  + **Filtering**: Remove rows that do not meet specified conditions.
  + **Expression Builder**: Supports complex expressions and functions for custom transformations.
* **Data Preview**: Allows you to see the results of your transformations in real-time before executing the pipeline.
* **Partitioning & Optimization**: Mapping Data Flow can handle large datasets, and you can configure partitioning, resource optimization, and scaling.
* **Execution in Spark Cluster**: Behind the scenes, Mapping Data Flows are executed on a Spark cluster managed by Azure Data Factory, allowing for parallelism and distributed processing of large data volumes.
* **Data Sources and Destinations**: Supports a variety of sources and destinations like Azure Blob Storage, SQL databases, Data Lakes, etc.

#### Example Use Case:

You have a dataset containing raw sales data that includes irrelevant columns and incorrect data types. You want to clean the data by removing unnecessary columns, converting data types, aggregating sales by region, and finally loading the cleaned data into a data warehouse.

Mapping Data Flow:

* Remove unnecessary columns (Select transformation)
* Convert data types (Derived Columns)
* Aggregate sales by region (Aggregate transformation)
* Load into Azure SQL Data Warehouse (Sink)

### ****2. Copy Activity****

The **Copy Activity** is designed for moving data from one location to another without performing any complex transformations. It is essentially a data transfer activity, and it is the go-to solution for simple ETL (Extract, Load) tasks.

#### Key Features of Copy Activity:

* **Data Movement**: Moves data between various data stores (e.g., SQL Server, Azure Blob Storage, Data Lakes, etc.).
* **Limited Transformations**: It provides only basic transformations, such as column mapping, data type conversion, and filtering specific rows, but it is not intended for complex transformation logic.
* **Supports Various Data Formats**: Copy Activity supports multiple data formats, such as CSV, JSON, Parquet, Avro, etc.
* **Performance**: Copy Activity is optimized for high-speed data transfers and can move data in parallel, especially for large datasets.
* **Source and Sink**: The activity uses a **Source** (where the data comes from) and a **Sink** (the destination for the data).
* **Fault Tolerance and Retry Logic**: Built-in mechanisms for retries and fault tolerance during data transfer.

#### Example Use Case:

You need to copy raw sales data from an on-premises SQL Server to Azure Blob Storage for backup purposes. No transformation is required; you just need the data to be moved as-is.

Copy Activity:

* Source: On-premises SQL Server
* Sink: Azure Blob Storage (CSV format)

**9. How do you pass parameters between pipelines in Azure Data Factory?**

* You can define parameters in the pipeline and pass them between activities or linked pipelines using expressions like @pipeline().parameters.parameterName.

**10. Control Flow activities in Azure Data Factory?**

* **If Condition**: Conditional branching.
* **ForEach**: Looping over a collection of items.
* **Wait**: Pausing pipeline execution.
* **Execute Pipeline**: Running another pipeline within the current pipeline.

**What is Activity in Azure Data Factory?**

ADF pipelines are a group of one or more activities. Activities in Azure Data Factory can be broadly categorized as:

1. **Data Movement Activities**
2. **Data Transformation Activities**
3. **Control Activities**

**1. DATA MOVEMENT ACTIVITIES :-**

**Copy Activity:** It simply copies the data from Source location to destination location. Azure supports multiple data store locations such as Azure Storage, Azure Blob Storage, Azure SQL Database, Azure Data Lake Storage, on-premises SQL Server, NoSQL, Files, etc.

**2. DATA TRANSFORMATION ACTIVITIES:-** These activities are used to process and transform data within pipelines. They include transformations such as mapping, filtering, aggregating, and joining data. Examples include Data Flow, Join, Filter, and Aggregate.

1. **Data Flow:**In data flow, First, you need to design data transformation workflow to transform or move data. Then you can call Data Flow activity inside the ADF pipeline. It runs on Scaled out Apache Spark Clusters. There are two types of DataFlows: Mapping and Wrangling DataFlows
2. **MAPPING DATA FLOW:** It provides a platform to graphically design data transformation logic. You don’t need to write code. Once your data flow is complete, you can use it as an Activity in ADF pipelines.
3. **WRANGLING DATA FLOW:**It provides a platform to use power query in Azure Data Factory which is available on Ms excel. You can use power query M functions also on the cloud.
4. **Hive Activity:** This is a HD insight activity that executes Hive queries on windows/linux based HDInsight cluster. It is used to process and analyze structured data.
5. **Pig activity:** This is a HD insight activity that executes Pig queries on windows/linux based HDInsight cluster. It is used to analyze large datasets.
6. **MapReduce:**This is a HD insight activity that executes MapReduce programs on windows/linux based HDInsight cluster. It is used for processing and generating large datasets with a parallel distributed algorithm on a cluster.
7. **Hadoop Streaming:** This is a HD Insight activity that executes Hadoop streaming program on windows/linux based HDInsight cluster. It is used to write mappers and reducers with any executable script in any language like Python, C++ etc.
8. **Spark:**This is a HD Insight activity that executes Spark program on windows/linux based HDInsight cluster. It is used for large scale data processing.
9. **Stored Procedure:**In Data Factory pipeline, you can use execute Stored procedure activity to invoke a SQL Server Stored procedure. You can use the following data stores: Azure SQL Database, Azure Synapse Analytics, SQL Server Database, etc.
10. **U-SQL:** It executes U-SQL script on Azure Data Lake Analytics cluster. It is a big data query language that provides benefits of SQL.
11. **Custom Activity:** In custom activity, you can create your own data processing logic that is not provided by Azure. You can configure .Net activity or R activity that will run on Azure Batch service or an Azure HDInsight cluster.
12. **Databricks Notebook:**It runs your databricks notebook on Azure databricks workspace. It runs on Apache spark and run Spark jobs as part of their data workflows.
13. **Databricks Python Activity:** This activity will run your python files on Azure Databricks cluster.
14. **Azure Functions:** It is Azure Compute service that allows us to write code logic and use it based on events without installing any infrastructure. It stores your code into Storage and keep the logs in application Insights.Key points of Azure Functions are :
15. It is a Serverless service.
16. It has Multiple languages available : C#, Java, Javascript, Python and PowerShell
17. It is a Pay as you go Model.

**3. Control Flow Activities:-** These are used to manage the flow of execution within pipelines. They include activities for branching, looping, conditional execution, and error handling. Examples include If Condition, For Each, Execute Pipeline, and Wait.

1. **Append Variable Activity:**It assigns a value to the array variable.
2. **Execute Pipeline Activity:** It allows you to call Azure Data Factory pipelines.
3. **Filter Activity:** It allows you to apply different filters on your input dataset.
4. **For Each Activity:** It provides the functionality of a for each loop that executes for multiple iterations.
5. **Get Metadata Activity:**It is used to get metadata of files/folders. You need to provide the type of metadata you require: childItems, columnCount, contentMDS, exists, itemName, itemType, lastModified, size, structure, created etc.
6. **If condition Activity:** It provides the same functionality as If statement, it executes the set of expressions based on if the condition evaluates to true or false.
7. **Lookup Activity:** It reads and returns the content of multiple data sources such as files or tables or databases. It could also return the result set of a query or stored procedures.
8. **Set Variable Activity:** It is used to set the value to a variable of type String, Array, etc.
9. **Switch Activity:**It is a Switch statement that executes the set of activities based on matching cases.
10. **Until Activity:** It is same as do until loop. It executes a set of activities until the condition is set to true.
11. **Validation Activity:** It is used to validate the input dataset.
12. **Wait Activity:**It just waits for the given interval of time before moving ahead to the next activity. You can specify the number of seconds.
13. **Web Activity:** Web activities enable interaction with external web services and APIs as part of data workflows. They can be used to make HTTP requests, call REST APIs, or interact with web endpoints for data exchange. We can use it for different use cases such as ADF pipeline execution.
14. **Webhook Activity:** It is used to to call the endpoint URLs to start/stop the execution of the pipelines. You can call external URLs also.

**Activity dependency**:- It defines how subsequent activities depend on previous activities, determining the condition of whether to continue executing the next task. An activity can depend on one or multiple previous activities with different dependency conditions.

**The different dependency conditions are**: Succeeded, Failed, Skipped, Completed.

To know more about Pipeline and activities use below link:

[Pipelines and activities in Azure Data Factory - Azure Data Factory | Microsoft Docs](https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities#control-flow-activities)

### What is Orchestration in Azure Data Factory (ADF)?

**It** refers to the process of designing, coordinating, scheduling and managing the execution of complex workflows or **pipelines** that involve data m ovement, transformation, and processing tasks across various data sources and compute services. In ADF, orchestration ensures that each step in the data pipeline is executed in the correct sequence and under the specified conditions, allowing for efficient, automated data workflows.

**11. What is the difference between Azure Data Factory and Azure Synapse Analytics?**

* ADF is mainly used for orchestrating ETL/ELT processes, while Synapse Analytics is a complete platform for big data processing, including integrated pipelines.

**Azure Data Factory (ADF)** and **Azure Synapse Analytics** are both Microsoft Azure services used for data integration, transformation, and analytics, but they are designed for different purposes and offer distinct features. Here’s a detailed comparison highlighting their differences:

**Debug Mode**

**Debug Mode** is a development/testing feature in ADF that allows data engineers to test their pipelines and data transformations interactively, without needing to create and activate a formal trigger. It is used to ensure that the pipeline logic, data transformations, and activities work as expected before running them in production.

* **Purpose**: Debugging lets you test and troubleshoot your pipeline and its activities, making it ideal for development and QA.
* **Execution**: You manually run the pipeline (or a part of it) from the ADF interface. You can set breakpoints or run only specific activities or portions of the pipeline.
* **Session**: When you enter Debug mode, ADF creates a temporary **debug session** for the pipeline. This session allows you to interactively monitor and view the execution results, logs, and data previews for each activity.
* **Costs**: Using Debug incurs charges, as it involves running activities and using compute resources temporarily, but it is generally used only during the development stage.
* **Immediate Results**: Debug mode provides immediate feedback for each pipeline activity, which is helpful for troubleshooting and making real-time adjustments to the pipeline logic.
* **No Scheduling**: Debug mode is run manually by the user and does not support scheduling or recurrence.

### ****Azure Data Factory (ADF)****

**Azure Data Factory** is a cloud-based **data integration** service that allows you to create, schedule, and orchestrate data workflows for moving and transforming data. It is primarily focused on **ETL (Extract, Transform, Load)** and **ELT (Extract, Load, Transform)** processes across various data sources.

#### Key Features of ADF:

1. **Data Integration**: ADF allows you to connect to over 90+ native connectors, enabling you to integrate data from on-premises and cloud sources, such as SQL Server, Oracle, Azure Blob Storage, and Amazon S3.
2. **Data Movement (ETL/ELT)**: ADF specializes in moving and transforming data. It can perform data movement between different systems and apply basic or complex transformations via activities like **Copy Activity**, **Mapping Data Flow**, or **Data Wrangling**.
3. **Orchestration and Scheduling**: You can create pipelines that orchestrate data movement and transformations, schedule them to run at specific intervals, and create complex workflows using control flow activities (e.g., If, ForEach, Until, Wait).
4. **Triggers**: ADF supports **Schedule Triggers**, **Event-Based Triggers**, and **Tumbling Window Triggers** to automate pipeline execution based on various events or schedules.
5. **Transformations**:
   * **Copy Activity** for simple data transfers.
   * **Mapping Data Flow** for complex transformations.
   * **Wrangling Data Flow** for data preparation (powered by Power Query).
6. **Data Sources**: ADF supports a wide variety of data sources, including **on-premises databases**, **cloud storage**, and **data lakes**.
7. **Integration with Azure Services**: ADF integrates well with other Azure services such as Azure Data Lake, Azure SQL Database, Azure Synapse Analytics, and Power BI.
8. **Code-Free**: ADF offers a low-code/no-code experience for designing ETL pipelines through a visual interface.

#### Example Use Case:

* Automating the process of extracting data from on-premises SQL Server, transforming it using Mapping Data Flow, and loading it into Azure SQL Data Warehouse for analytics.

### ****Azure Synapse Analytics****

**Azure Synapse Analytics** (formerly known as **Azure SQL Data Warehouse**) is an **end-to-end analytics platform** that combines **big data analytics**, **data integration**, and **data warehousing** in one unified platform. It extends the capabilities of Azure Data Factory and adds the ability to perform advanced analytics and big data processing.

#### Key Features of Synapse Analytics:

1. **Data Integration and Orchestration**: Synapse Analytics includes all the capabilities of Azure Data Factory for **data integration** and **ETL/ELT**. You can use **Synapse Pipelines**, which are identical to ADF pipelines, for data movement and transformation.
2. **Data Warehousing**: Synapse offers a fully managed **distributed data warehouse** solution, enabling you to run large-scale **SQL-based** analytics over relational and non-relational data.
3. **Big Data Analytics with Apache Spark**: In addition to SQL-based analytics, Synapse also supports **Apache Spark** pools, enabling you to perform **big data processing** on distributed datasets using **Spark SQL**, **PySpark**, **Scala**, and **Notebooks**.
4. **Synapse Studio**: Provides a unified workspace where you can work with SQL, Spark, pipelines, and data exploration in one place. It allows users to create and manage data workflows, run SQL queries on the Synapse SQL pool, and interact with big data using Spark.
5. **Integrated Analytics**: Synapse integrates **SQL and Spark** in one platform, allowing you to run **on-demand queries** against massive datasets stored in data lakes and integrate the results with data warehousing workloads.
6. **Serverless SQL Pool**: Synapse provides **serverless SQL pools** that allow you to query data directly from Azure Data Lake Storage or other file-based storage (e.g., CSV, Parquet, or JSON) without needing to provision a dedicated SQL pool.
7. **Data Lake Exploration**: Synapse allows you to explore and query your **Azure Data Lake** with both SQL and Spark, making it easier to integrate **data lakes** with your data analytics workflows.
8. **Analytics Performance**: With Synapse, you can scale compute and storage independently, providing flexibility and optimized performance for **petabyte-scale analytics**.

#### Example Use Case:

* A retail company wants to perform analytics on millions of sales records stored in a data lake. They use Synapse's **serverless SQL pool** to query the data and then run **Spark jobs** to analyze customer behavior patterns, all in one platform.

### ****Key Differences Between Azure Data Factory and Azure Synapse Analytics****

| **Feature** | **Azure Data Factory (ADF)** | **Azure Synapse Analytics** |
| --- | --- | --- |
| **Primary Focus** | Data integration, ETL/ELT, data movement | End-to-end analytics platform combining big data, data warehousing, and integration |
| **Data Movement & Transformation** | Yes, via Copy Activity, Mapping Data Flow, and Wrangling Data Flow | Yes, via Synapse Pipelines (same as ADF Pipelines) |
| **Data Warehousing** | No data warehouse capabilities | Includes a distributed SQL-based data warehouse (formerly SQL DW) |
| **Big Data Processing** | No built-in big data analytics (integration only) | Includes Spark clusters for big data processing and analytics |
| **Orchestration & Scheduling** | Yes, supports triggers and schedules | Yes, identical pipeline and orchestration features as ADF |
| **Unified Analytics Platform** | No, focused on ETL/ELT | Yes, combines ETL, big data, and SQL data warehousing in one platform |
| **Spark Integration** | No Spark integration | Supports Apache Spark for big data processing and advanced analytics |
| **SQL Pools** | No SQL pool | Dedicated and serverless SQL pools for querying structured and unstructured data |
| **Code-Free Experience** | Yes, through a visual interface | Yes, with Synapse Studio, supporting SQL, Spark, and pipelines in one place |
| **Data Exploration** | Limited to data orchestration and movement | Full data exploration capabilities with SQL and Spark across data lakes and data warehouses |
| **Typical Use Case** | ETL pipelines for moving and transforming data | End-to-end analytics involving ETL, big data, and data warehousing analytics |

### What are the steps involved in creating a ADF:-

### Create a container in azure blob storage as source.

### Create a table in azure sql as destination.

### Now create a linked service for azure blob and azure sql in manage tab of ADF.

### Create a dataset for both source and destination.

### Create a copy data activity in pipeline fill the source and sink.

### Now validate and publish all.

### Create a new trigger.

### When to Use Azure Data Factory vs. Azure Synapse Analytics

* **Use Azure Data Factory (ADF)** if:
  + You need to create **ETL/ELT pipelines** for **data integration**.
  + Your primary goal is **moving and transforming data** across different systems and environments.
  + You don’t need big data analytics or a data warehouse, but just need to **orchestrate data workflows**.
* **Use Azure Synapse Analytics** if:
  + You need a comprehensive **end-to-end analytics platform** for both **big data processing** and **data warehousing**.
  + You want to combine SQL-based analytics with **big data analytics** using **Spark**.
  + You require both **on-demand querying** (serverless) and dedicated **data warehouse capabilities** for scalable analytics.

**12. How does Azure Data Factory handle failure in pipeline execution?**

* ADF has built-in retry policies and options for error handling, such as "Continue on Failure," "Retry Activity," or "Stop Pipeline."

**Integration runtime** is the compute infrastructure used by Azure Data Factory (ADF) to provide various data integration capabilities across different network environments. There are three types of integration runtimes offered by Data Factory:

* Azure integration runtime
* Self-hosted integration runtime
* Azure-SQL Server Integration Services (SSIS) integration runtime

**Node Size and Node Number in setting SSIS integration Runtime:-**

**Node Size:-** Size of virtual machine running the SSIS Package. i.e d8\_v3 (8 core, 32768 MB).

**Node Number:-** Number of virtual machine. Cost depends on node size and node number.

## Azure integration runtime:- The compute resource for an Azure integration runtime is fully managed elastically in Azure and is used to perform data movement and transformation activities within the Azure cloud environment. It is optimized for transferring data between Azure services and can scale dynamically based on workload demands. The following table provides descriptions for properties returned by the ****Get-AzDataFactoryV2IntegrationRuntime**** command.

The following table provides descriptions of properties returned by the cmdlet for an Azure integration runtime:- **Name, State/status, Location, DataFactoryName, ResourceGroupName, Description**

**Possible statuses of an Azure integration runtime:-**

1. **Online:-** The Azure integration runtime is online and ready to be used.
2. **Offline :-** The Azure integration runtime is offline due to an internal error.

## Self-hosted integration runtime:- This runtime is installed on your on-premises network or virtual machines (VMs) and enables Azure Data Factory to interact with on-premises data sources and destinations. It provides secure connectivity to on-premises systems without exposing them to the internet and supports data movement between on-premises and cloud environments. It provides descriptions for properties returned by the Get-AzDataFactoryV2IntegrationRuntime cmdlet.

The following table provides descriptions of monitoring Properties for **each node** are ***Name, Status, Version, Available Memory, CPU utilization, Networking(in/out), Concurrent Job(Running/Limit), Role***

**Two types of Concurrent Jobs:-**

1. **Running**:- Number of jobs or tasks running on each node. This value is a near real-time snapshot.
2. **Limit**:- Limit signifies the maximum concurrent jobs for each node. This value is defined based on the machine size. You can increase the limit to scale up concurrent job execution in advanced scenarios, when activities are timing out even when CPU, memory, or network is under-utilized. This capability is also available with a single-node self-hosted integration runtime.

**There are two types of roles in a multi-node self-hosted integration runtime –**

1. **Worker:-** All nodes are workers, which means they can all be used to execute jobs.
2. **Dispatcher:-** There is only one dispatcher node, which is used to pull tasks/jobs from cloud services and dispatch them to different worker nodes. The dispatcher node is also a worker node.

**Status (per node):-**

1. **Online**:- Node is connected to the Data Factory service
2. **Offline**:- Node is offline.
3. **Upgrading**:- The node is being auto-updated.
4. **Limited**:- Due to a connectivity issue. May be due to HTTP port 8060 issue, service bus connectivity issue, or a credential sync issue.
5. **Inactive**:- Node is in a configuration different from the configuration of other majority nodes.

**Azure-SSIS integration Runtime**:- This runtime is used specifically for executing SQL Server Integration Services (SSIS) packages within Azure Data Factory. It allows you to lift and shift existing SSIS workloads to the cloud and provides native support for running SSIS packages in Azure with scalability and flexibility. Azure-SSIS IR doesn't run any other ADF activities. Once provisioned, you can monitor its overall/node-specific properties and statuses via Azure PowerShell, Azure portal, and Azure Monitor.

The following table provides descriptions of properties returned by the above cmdlet for an Azure-SSIS IR are ***CreateTime, Nodes, OtherErrors, LastOperation, State, Location, NodeSize, NodeCount, MaxParallelExecutionsPerNode, CatalogServerEndpoint, CatalogAdminUserName, CatalogAdminPassword, CatalogPricingTier, VNetId, Subnet, ID, Type, ResourceGroupName, DataFactoryName, Name, Description***

* **Status (per Azure-SSIS IR node):-** Starting, Available, Recycling, Unavailable
* **Status (overall Azure-SSIS IR):-** Initial, Starting, Started, Stopping, Stopped

**13. What is the use of the Lookup Activity in Azure Data Factory?**

* Lookup Activity retrieves data from a specified dataset and returns it for further use in the pipeline.

**14. Explain the use of Variables in Azure Data Factory pipelines.**

* Variables store intermediate values during pipeline execution. They can be set using the Set Variable activity or through expressions.

**15. What is the difference between Dataset and Linked Service in ADF?**

* **Linked Service**: Connects to a data source (e.g., SQL Server, Blob Storage).
* **Dataset**: Represents the structure of the data within the linked service.

**16. How can you optimize the performance of a Copy Activity in Azure Data Factory?**

* Enable parallelism by configuring the "Degree of Copy Parallelism."
* Use staged copy when copying large datasets between incompatible stores.
* Tune the integration runtime and batch sizes.

**17. How do you use the Copy Activity to load data into Azure Data Lake or Blob Storage?**

* Define a source dataset (e.g., SQL database), a destination dataset (e.g., Azure Data Lake), and configure the copy activity to map data from source to destination.

**Data Flow Debug**

The Data Flow Debug option is available inside of a data flow activity and allows you to pass a subset of data through the flow, which can be useful to test whether columns are mapped correctly.

**18. Explain how to monitor pipeline executions in Azure Data Factory.**

* Use the "Monitor" tab in ADF to track pipeline execution, view activity run history, and analyze logs for success and failure.

Monitoring pipeline executions in **Azure Data Factory (ADF)** is essential for tracking data movement, identifying failures, and optimizing performance. Azure Data Factory provides built-in monitoring tools that allow you to view the status of your pipelines, diagnose issues, and take corrective actions. Here's how you can monitor pipeline executions in ADF:

### 1. ****Azure Data Factory Monitoring Interface****

Azure Data Factory provides a **Monitoring and Management** dashboard in the Azure portal where you can track the status of pipeline runs, activities, triggers, and integration runtimes. Here's how you can use it:

#### Accessing the Monitoring Dashboard:

1. **Open Azure Portal**.
2. Navigate to your **Azure Data Factory** instance.
3. Select **Monitor** from the left-hand menu.

In this dashboard, you can monitor **pipeline runs**, **activity runs**, **trigger runs**, and the status of the **integration runtime**.

### 2. ****Pipeline Runs****

The **Pipeline Runs** section allows you to monitor the execution of your pipelines. You can:

* View the **status** of each pipeline run (Success, Failed, In Progress, Canceled).
* Track the **start time**, **end time**, and **duration** of pipeline executions.
* Monitor detailed metrics for **throughput** and **data movement** performance.

#### Key Features:

* **Filter and Search**: You can filter pipeline runs based on the pipeline name, status, start/end time, and execution duration.
* **Retry**: For failed pipeline runs, you can retry the execution from the monitoring interface.
* **Drill Down**: Click on any pipeline run to view the **activity runs** (the individual tasks within the pipeline) and their status.

#### Example:

You can monitor a pipeline run that copies data from Blob Storage to Azure SQL Database. If a pipeline fails, you'll see detailed error messages in the activity logs, such as "Timeout error" or "Invalid credentials," which can help in troubleshooting.

### 3. ****Activity Runs****

Once you drill down into a pipeline run, you can view the status of each individual activity (e.g., Copy, Data Flow, ForEach, etc.) in the **Activity Runs** section. For each activity, you can monitor:

* **Execution status**: Success, Failed, Queued, In Progress, etc.
* **Start and end time**.
* **Input/Output data**: You can view the data passed between activities, such as the size of the data or number of rows copied.
* **Error details**: If an activity fails, detailed error messages are provided for troubleshooting.

#### Key Features:

* **Detailed Logs**: For each activity, you can inspect logs that provide detailed input/output, retry count, and specific error details.
* **Data Movement Details**: For copy activities, you can view data movement details such as throughput, file size, and number of rows processed.

### 4. ****Trigger Runs****

In the **Trigger Runs** section, you can monitor:

* **Scheduled triggers**: View the status of time-based triggers (e.g., scheduled pipelines).
* **Event-based triggers**: Monitor event triggers, such as file-based triggers (e.g., a pipeline that runs when a file is uploaded to Blob Storage).
* **Manual triggers**: Track the pipelines that were manually triggered by users.

#### Key Features:

* **Trigger History**: Provides a history of when each trigger was activated and whether the pipeline ran successfully or failed.

### 5. ****Integration Runtime Monitoring****

The **Integration Runtime (IR)** is the compute infrastructure used by ADF to move and transform data. Monitoring IR is essential for tracking the health and performance of data movement processes, especially if you use **self-hosted IR** for on-premises or hybrid data sources.

#### Key Monitoring Metrics:

* **Availability**: Monitor whether the IR is online and functioning.
* **Resource Utilization**: Track CPU and memory utilization for self-hosted IR, ensuring that it's not overloaded.
* **Queue and Execution Details**: For each pipeline execution, you can view the queue time (how long the job waited before being executed) and execution time.

### 6. ****Alerts and Metrics in Azure Monitor****

Azure Data Factory integrates with **Azure Monitor** and **Log Analytics** to provide deeper insights into pipeline executions. Using Azure Monitor, you can:

* **Create Alerts**: Set up alerts based on specific conditions, such as pipeline failures, execution times, or data thresholds.
  + Example: You can configure an alert to notify you via email if any pipeline fails or takes more than 1 hour to complete.
* **Track Metrics**: You can view a variety of built-in metrics, including:
  + **Pipeline Run Failures**: Number of failed pipelines over time.
  + **Activity Run Failures**: Number of failed activities within pipelines.
  + **Data Read/Write Volume**: Track the volume of data being processed by copy activities.

#### Configuring Alerts:

1. Go to **Azure Monitor** in the Azure portal.
2. Select **Metrics** and choose the **Data Factory** resource.
3. Create a new alert by specifying the condition (e.g., pipeline failure) and the notification channel (email, SMS, webhook).

### 7. ****Logs in Log Analytics****

* You can configure **Log Analytics** for more advanced querying and log aggregation. This allows you to analyze historical pipeline runs, trends, and performance over time.
* **Querying Logs**: Using **Kusto Query Language (KQL)**, you can query logs and visualize data on custom dashboards. For example, you can query all failed pipelines in the last 24 hours and break them down by error type.

#### Example Query:

ADFActivityRun

| where Status == "Failed"

| summarize count() by PipelineName, ActivityType, ErrorType

This query returns a count of failed activity runs grouped by pipeline name, activity type, and error type.

### 8. ****Self-Hosted Integration Runtime Logging****

If you are using a **self-hosted integration runtime**, logs related to the IR’s performance and connectivity are written to local logs on the machine where the IR is hosted. You can access these logs for troubleshooting purposes.

### 9. ****Debugging and Error Handling****

During development, you can run pipelines in **debug mode** to see the execution flow in real-time and troubleshoot issues before final deployment. Debugging provides:

* **Real-time monitoring** of pipeline activities.
* Immediate feedback on data processing and transformations.

For production pipelines, use **error handling** activities like **Try-Catch**, **Retry**, and **Timeout** policies to ensure that errors are managed efficiently.

### 10. ****Integration with Power BI (for Visualization)****

For advanced reporting, you can export pipeline execution data to **Power BI** and create custom visualizations. This allows for historical analysis and performance monitoring across multiple pipelines.

### Summary:

Monitoring in Azure Data Factory is crucial for ensuring that your data pipelines run efficiently, reliably, and securely. By using ADF’s built-in monitoring features, Azure Monitor, and Log Analytics, you can:

* Track pipeline, activity, and trigger executions in real time.
* Set up alerts for failures and performance issues.
* Drill down into detailed logs and metrics for troubleshooting.
* Ensure proactive management of data workflows, preventing downtime or data loss.

**19. What are Tumbling Window Triggers, and what are they used for?**

* Tumbling Window Triggers execute pipelines at fixed intervals without overlapping and are used for processing data in batches over specific time windows.

**20. How do you handle dynamic partitioning in Copy Activity?**

* Use dynamic partitioning by configuring the "Max Concurrent Connections" and specifying partition keys in the source settings.

**21. What is a Sink in Azure Data Factory, and how is it used?**

* **Sink** refers to the **destination or target location** such as Azure Blob, SQL Database, or Data Lake where the transformed or moved data is written after it has been processed by a pipeline. Essentially, the Sink is the endpoint where the data is delivered or stored.

**Key Concepts of a Sink in ADF:**

1. **Data Destination**: A **Sink** is the final destination where data is loaded after being extracted from a **Source** and optionally transformed. This could be a database, a file in cloud storage, or any other supported data store.
2. **Sink Dataset**: The Sink is associated with a **dataset**, which defines the schema, file type, structure, and other properties of the target data store. The dataset specifies the **connection** to the Sink and how data is written to it.
3. **Supported Sink Types**: Azure Data Factory supports a wide range of Sink types, including:
   * **Azure Blob Storage** (for writing files in formats like CSV, JSON, Parquet, etc.)
   * **Azure Data Lake Storage**
   * **Azure SQL Database**
   * **SQL Server**
   * **Azure Synapse Analytics**
   * **Azure Cosmos DB**
   * **Amazon S3**
   * **Oracle, MySQL, PostgreSQL** databases
   * **File Systems** (on-premises or cloud)
4. **Sink Properties**: Each Sink has specific configuration settings:
   * **File format**: For file-based sinks (e.g., Azure Blob Storage), you need to specify the output file format (CSV, Parquet, Avro, etc.).
   * **Schema mapping**: Defines how the incoming data from the source or transformation is mapped to the schema of the destination.
   * **Write behavior**: Controls how data is written to the sink. Options include:
     + **Append**: Adds data to the existing data in the target (e.g., appending rows to a table).
     + **Overwrite**: Replaces existing data with the new data.
     + **Upsert**: Updates existing records or inserts new ones.
     + **Bulk Insert**: For fast data loading into databases.
5. **Data Transformation**: Data is often transformed before being written to a Sink. This could involve filtering, aggregating, joining, or any other transformation applied through **Mapping Data Flows** or custom transformations (e.g., Databricks).
6. **Sink in Copy Activity**: In the **Copy Activity** (which is the primary activity for moving data from source to destination in ADF), the Sink is configured to specify where the data will be written. You can set up transformations like column mapping or type conversion during the copy process.

**22. How does Azure Data Factory handle schema changes?**

* ADF supports schema drift and allows for schema flexibility in Mapping Data Flows by auto-mapping incoming fields or using dynamic expressions to handle schema variations.

**23. Explain the use of the ForEach activity in Azure Data Factory.**

* ForEach activity iterates over an array of items, such as file paths or records, and performs a set of activities for each item.

**24. How do you ensure security and encryption in Azure Data Factory?**

* Use Azure Key Vault to securely store and manage sensitive information like connection strings and API keys. Use SSL/TLS for encrypted data transfer.

Ensuring security and encryption in **Azure Data Factory (ADF)** is crucial for safeguarding data during movement, processing, and storage. ADF provides a range of features and best practices to enhance security at various levels, including authentication, encryption, network security, and monitoring. Here’s a breakdown of how you can ensure security in Azure Data Factory:

**1. Data Encryption**

* **Encryption at Rest**:
  + Data stored in Azure services (e.g., Azure Blob Storage, Data Lake, SQL Database) is encrypted at rest using **Azure Storage Service Encryption (SSE)**. Azure automatically manages encryption keys through **Azure-managed keys** or **customer-managed keys (CMK)** stored in **Azure Key Vault**.
  + You can enforce encryption on datasets and linked services to ensure sensitive data is encrypted when stored in destinations.
* **Encryption in Transit**:
  + Data is encrypted in transit using **Transport Layer Security (TLS)** to prevent unauthorized access during data movement. By default, ADF uses HTTPS to ensure secure transmission.
  + You can configure your source and sink linked services to ensure that connections use HTTPS or other secure protocols.

**2. Authentication and Authorization**

* **Azure Active Directory (Azure AD) Integration**:
  + **Azure AD** is used for authenticating access to Azure Data Factory resources and managing access control.
  + For **user access**, you can assign **role-based access control (RBAC)** through Azure AD to control who can manage, read, or contribute to ADF resources.
* **Managed Identity**:
  + ADF can use a **managed identity** to securely access Azure services like Blob Storage, SQL Database, Key Vault, etc., without requiring secrets or credentials in your pipelines.
  + You can assign **User-Assigned Managed Identities (UAMI)** or **System-Assigned Managed Identities (SAMI)** to pipelines, ensuring secure access to resources.
* **Service Principal Authentication**:
  + ADF can be configured to use **Azure AD service principals** for authenticating external services (e.g., when accessing SQL Database or Data Lake).
  + This eliminates the need to store sensitive credentials (like passwords or access keys) in the linked services.

**3. Access Control and Role-Based Access Control (RBAC)**

* **Role-Based Access Control (RBAC)** in Azure Data Factory allows you to assign specific roles with appropriate permissions to users and applications. These roles include:
  + **Reader**: View but cannot modify ADF resources.
  + **Contributor**: Create and modify ADF resources.
  + **Owner**: Full control over ADF, including granting access to others.
* **Granular Access Control**: You can apply RBAC at different scopes, such as the resource group, data factory, or even specific pipelines. This allows for fine-grained control over who can access what.

**4. Key Management with Azure Key Vault**

* **Azure Key Vault** is used to securely store ***secrets, keys, and certificates***. Azure Data Factory can integrate with Key Vault to manage sensitive information like connection strings, access keys, and API keys.
  + You can reference **secrets** stored in Azure Key Vault in linked services, avoiding the need to store sensitive information in plain text in your pipeline definitions.
  + You can configure pipelines to use **Customer-Managed Keys (CMK)** for encrypting data, giving you greater control over encryption processes.

**5. Network Security**

* **Private Endpoints (Private Link)**:
  + You can configure **Azure Data Factory to connect using Private Endpoints**, ensuring that data movement happens over a private, secure network, rather than over the public internet.
  + Private endpoints allow you to securely connect ADF to Azure services (e.g., Azure Blob Storage, SQL Database) over an **Azure Virtual Network (VNet)**.
* **Integration Runtime Security**:
  + **Self-hosted Integration Runtime (IR)** provides secure connectivity to on-premises or virtual network-based resources. Data is transferred via encrypted communication channels between ADF and your resources.
  + You can install IR on machines inside your **virtual network (VNet)** or **on-premises** to securely access local resources without exposing them to the public internet.
* **Firewall Rules**:
  + Enable firewall rules for **Azure Data Factory** to restrict access to only specific IP ranges, providing an extra layer of protection.
  + You can also configure firewall rules on resources such as Azure SQL Database or Storage Accounts to allow access only from ADF or specific networks.

**6. Secure Access to External Data Sources**

* **Shared Access Signatures (SAS)**:
  + For accessing resources like Blob Storage or Data Lake, use **SAS tokens** with restricted permissions (e.g., read-only access) and time-limited expiry to control and secure access to data.
* **OAuth2 Authentication**:
  + Use **OAuth2** for securely accessing certain data stores or APIs, ensuring token-based access control, which is more secure than static credentials.

**7. Monitoring and Logging**

* **Azure Monitor and Log Analytics**:
  + Azure Data Factory integrates with **Azure Monitor** and **Log Analytics** to provide monitoring, diagnostics, and logging capabilities. You can use these tools to track activities, detect suspicious behaviors, and monitor security breaches.
  + **Activity logs** can capture details about pipeline execution, including successes, failures, and access attempts.
* **Alerts and Notifications**:
  + Configure **alerts** based on pipeline failures, data access errors, or unauthorized attempts. You can set up notifications through **email, SMS**, or other channels to promptly act on security incidents.

**8. Data Masking and Anonymization**

* **Data Masking**: If you're working with sensitive data, consider implementing **data masking** at the source database to obscure sensitive information.
* **Anonymization**: Use **data anonymization techniques** during data processing to protect personal or sensitive data when it’s being moved or transformed.

**9. Compliance Certifications**

* **Azure Data Factory** is certified for many compliance standards, including **ISO 27001**, **SOC 1, 2, and 3**, **HIPAA**, **GDPR**, and **FedRAMP**. These certifications ensure that Azure services, including ADF, meet stringent security and privacy requirements.

**Best Practices for Ensuring Security in Azure Data Factory:**

1. **Use Managed Identities** for secure and seamless resource access.
2. **Leverage Azure Key Vault** for managing secrets and sensitive information.
3. **Encrypt data in transit** using HTTPS and at rest with service-managed or customer-managed keys.
4. **Implement Role-Based Access Control (RBAC)** to restrict access based on roles and follow the principle of least privilege.
5. **Use Private Endpoints** to prevent data movement over the public internet.
6. **Monitor and audit** your pipelines using Azure Monitor and Log Analytics to track security incidents.
7. **Rotate keys and credentials** stored in linked services periodically for better security hygiene.

**25. How do you integrate Git with Azure Data Factory?**

* ADF allows integration with GitHub or Azure DevOps repositories to version control pipelines, linked services, and datasets, facilitating CI/CD.

**26. What is the difference between Append and Merge in Mapping Data Flows?**

* **Append**: Combines data from two or more datasets into one.
* **Merge**: Merges rows from multiple datasets based on matching conditions.

**27. What is the use of the Data Flow Debug feature in Azure Data Factory?**

* Data Flow Debug allows you to interactively test, troubleshoot, and debug your Mapping Data Flows by previewing data transformations at different steps.

**28. What are some best practices for managing large data loads in Azure Data Factory?**

* Use partitioning and parallelism to process data in chunks.
* Optimize integration runtime scaling.
* Leverage incremental data loads instead of full loads.

**29. How do you create an event-based trigger in Azure Data Factory?**

* Use the Event Trigger to monitor Blob Storage or Data Lake for file creation or deletion events and trigger pipelines based on these events.

**30. How do you handle file formats such as JSON, Parquet, or Avro in ADF?**

* Azure Data Factory natively supports reading and writing data in different file formats like JSON, Parquet, and Avro. You can define source and sink datasets with these file formats and perform transformations accordingly.

1. **List five types of data sources supported by Azure Data Factory.**
   1. Relational databases (e.g., Azure SQL Database, SQL Server)
   2. Cloud storage services (e.g., Azure Blob Storage, Azure Data Lake Storage
   3. On-premises data sources (e.g., SQL Server on-premises, file servers)
   4. SaaS applications (e.g., Salesforce, Dynamics 365)
   5. NoSQL databases (e.g., Azure Cosmos DB, MongoDB)
2. **Can Azure Data Factory process multiple pipelines?**

* Yes, **Azure Data Factory (ADF)** can process multiple pipelines simultaneously.
* Each pipeline can use its own set of resources, such as **Integration Runtimes (IRs)**
* Different pipelines can be triggered using different **schedules** or **event-based triggers.**

While ADF does allow for multiple pipeline executions, there are some **throttling limits** to be aware of. For example:

* **Activity concurrency**: There are limits on how many activities can be running concurrently in a single pipeline.
* **Integration Runtime (IR) resources**: The number of pipelines you can run in parallel depends on the scaling limits of the integration runtime.
* **Data movement**: There are limits on data movement operations per unit of time, depending on the service tier used.

1. **What is the distinction between Azure Data Lake and Azure Data Warehouse?**

|  |  |
| --- | --- |
| **Azure Data Lake** | **Azure Data Warehouse** |
| 1. Store raw, unprocessed data of any type or format 2. Supports structured, semi-structured, and unstructured data 3. Supports various formats like JSON, CSV, Parquet, Avro, etc. 4. Batch processing, real-time analytics, machine learning, etc. 5. Offers limitless scalability for storing petabytes of data 6. Suitable for data science, exploratory analytics, and big data processing 7. Pay-as-you-go pricing based on storage usage and data egress fees | **1.** Analyze structured, relational data using SQL-based tools 2. Designed for structured, tabular data with defined schemas 3. Requires data to be structured and loaded into tables 4. SQL-based analytics, reporting, and business intelligence 5. Provides scalable compute and storage resources 6. Optimized for high-concurrency analytical queries and reporting 7. Consumption-based pricing based on compute and storage usage |

1. **What is the purpose of Linked services?**

The primary purpose of linked services is to enable Azure Data Factory to:  
  
Ingest Data  
Transform Data  
Load Data  
Orchestrate Workflows

1. **What are ARM Templates in Azure Data Factory?**

ARM (Azure Resource Manager) templates in Azure Data Factory are declarative JSON files that define the infrastructure and configuration of Azure Data Factory resources within an Azure environment. These templates follow the ARM template syntax and structure, allowing you to define and deploy Azure Data Factory resources in a consistent and repeatable manner using infrastructure as code (IaC) principles.

1. **What is Blob Storage in Azure?**

**Azure Blob Storage** is a scalable, object storage solution offered by Microsoft Azure for storing large amounts of unstructured data, such as text or binary data.

**Multiple Tiers of Storage**: Blob Storage offers three storage tiers to optimize cost based on how frequently data is accessed:

* + **Hot**: For data that is accessed frequently.
  + **Cool**: For data that is infrequently accessed but stored for at least 30 days.
  + **Archive**: For data that is rarely accessed and stored for a longer period (cheapest tier).

**Types of Blobs:**

1. **Block Blobs**: Used to store files, documents, images, and other large objects. These blobs are ideal for streaming and storing large amounts of unstructured data.
2. **Append Blobs**: Optimized for logging scenarios, where data is appended over time, such as log files.
3. **Page Blobs**: Used for storing random-access files, such as virtual hard drive (VHD) files used by Azure virtual machines.
4. **Difference between Azure HDInsight and Azure Data Lake Analytics?**

|  |  |
| --- | --- |
| **Azure HDInsight** | Azure Data Lake Analytics |
| 1. It is a Platform as a Service. 2. Processing data in it requires configuring the cluster with predefined nodes. Further, by using languages like pig or hive, we can process the data. 3. Users can easily configure HDInsight Clusters at their convenience. Users can also use Spark, and Kafka, without restrictions. | 1. It is a Software as a Service. 2. It is all about passing the queries written for data processing. Data Lake Analytics further creates compute nodes to process the data set. 3. It does not give that much flexibility in terms of configuration and customization. But, Azure manages it automatically for its users. |

1. **How can we schedule a pipeline?**

The trigger follows a world clock calendar schedule that can schedule pipelines periodically or in calendar-based recurrent patterns. We can schedule a pipeline in two ways:

1. Schedule Trigger

2. Window Trigger

1. **An Azure Data Factory Pipeline can be executed using three methods?**

Methods to execute Azure Data Factory Pipeline:  
  
1. Debug Mode  
2. Manual execution using trigger now  
3. Adding schedule, tumbling window/event trigger

1. **Which Data Factory version is used for creating data flows?**

Data Factory V2 version is used to create data flows

1. **What are the two levels of security in ADLS Gen2?**

**1. Role-Based Access Control** – It includes built-in azure rules such as reader, contributor, owner or customer roles. It is specified for two reasons. The first is, who can manage the service itself, and the second is, to permit the reasons is to permit the users built-in data explorer tools.   
**2. Access Control List** – An ACL is a set of rules that can be applied to a resource, such as a   
directory, file, or database. The rules can be used to specify who can perform actions on the   
resource, and when those actions can take place.

1. **What are the two types of compute environments that are supported by Data Factory?**

On-demand compute environment(ODCE) – On-Demand Compute Environment (ODCE) provides a set of tools that enable you to build, manage, and operate a fully managed cloud service that provides compute resources as a service.

Bring your own environment(BYOE)– It is a service in Azure that allows you to run your own cloud-based data warehouse wherein you can manage the compute environment with ADF.

1. **What are the cross-platform SDKs in Azure Data Factory?**

* Python SDK
* PowerShell SDK
* .NET SDK
* Java SDK
* C# SDK

1. **Differentiate between the Mapping data flow and Wrangling data flow transformation activities in Adf?**

The Mapping data flow activity transforms one or more source datasets into a single destination dataset while the Wrangling data flow activity transforms one or more source datasets into multiple destination datasets.

1. **Can you push code and have CI/CD in ADF?**

Data Factory fully supports CI/CD for data pipelines using Azure DevOps and GitHub.   
This allows the ETL process to be developed and deployed in stages before releasing the   
finished product. Once the raw data is refined into a ready-to-manipulate, consumable format,  
 load the data into Azure Data Warehouse or Azure SQL Azure Data Lake, Azure Cosmos DB, or   
other analytics engine that your organization can reference from business intelligence tools.

1. **What is the purpose of Lookup activity in the Azure Data Factory?**

The Lookup activity in Azure Data Factory (ADF) is used to retrieve data from a data source. This activity allows you to perform a SQL query or other data retrieval operations, such as reading from a file or database, and then use the retrieved data within the data pipeline for further processing or decision-making.  
  
Here are some common purposes and uses of the Lookup activity:  
  
**1. Retrieve Single or Multiple Rows:** The Lookup activity can be configured to return either a single row or multiple rows from the specified data source, depending on the query and the needs of your pipeline.  
  
**2. Conditional Logic:** You can use the data retrieved by the Lookup activity to make decisions in your pipeline. For instance, based on the data retrieved, you might decide to execute different activities or branches within the pipeline.  
  
**3. Dynamic Parameterization:** The results from the Lookup activity can be used to dynamically set parameters or variables in subsequent activities within the pipeline.  
  
**4. Data Validation**: Before processing data further, you can use the Lookup activity to validate certain conditions by retrieving specific values from a data source and ensuring they meet expected criteria.  
 **5. Data Integration:** In more complex scenarios, you can use the Lookup activity to integrate or join data from different sources by retrieving specific data points that can be combined with other datasets.

# Change Data Capture (CDC) in Azure Data Factory

Microsoft recently announced a new feature in Azure Data Factory (ADF) and Azure Synapse Pipelines designer mode as a factory resource. This feature is called **Change Data Capture (CDC)**, which refers to the process of identifying and capturing changes made to data in source data sources and then delivering those changes in real time to a downstream process or system [1].

## ****Use Cases?****

Till now, this feature has been one of the most wanted ones in ADF for both data migration and modernisation projects in the Azure cloud. With the growing demand for data and AI projects, this feature will be utilised more frequently.

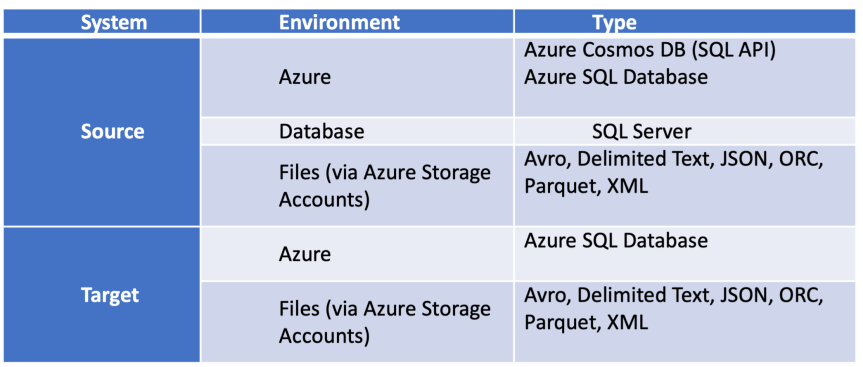
## ****Benefits?****

CDC is an optimised approach in cloud data architecture as it is an efficient way to move data across components and networks. This is due to extracting, loading, and transforming in near real-time while keeping the systems in sync. The benefits are:

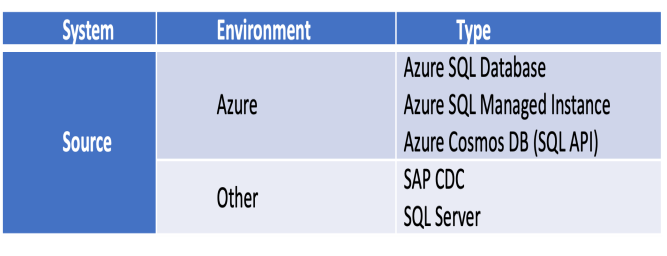
* No bulk load by enabling incremental load
* Less networking bandwidth required
* Less cloud cost
* Less time and more performance
* More source and target synchronisation

## ****CDC and/or Incremental Load Approaches:****

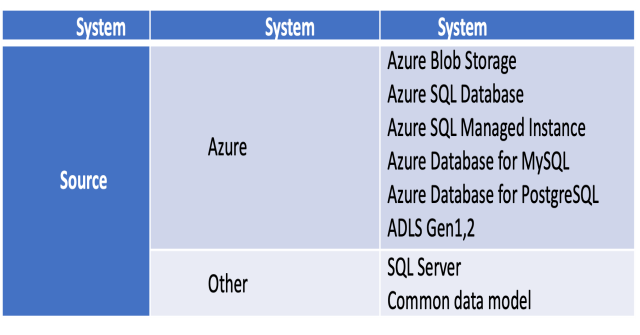
1. **CDC Resource (Preview):** The easiest and quickest method that allows for full CDC that continuously runs in near real-time through a guided configuration experience. Note that this approach requires a timestamp or ID column. The key point for this approach is while ADF pipelines are in batch mode, this CDC resource is running continuously, where you set a latency that wakes up the ADF and looks for changed data [3].



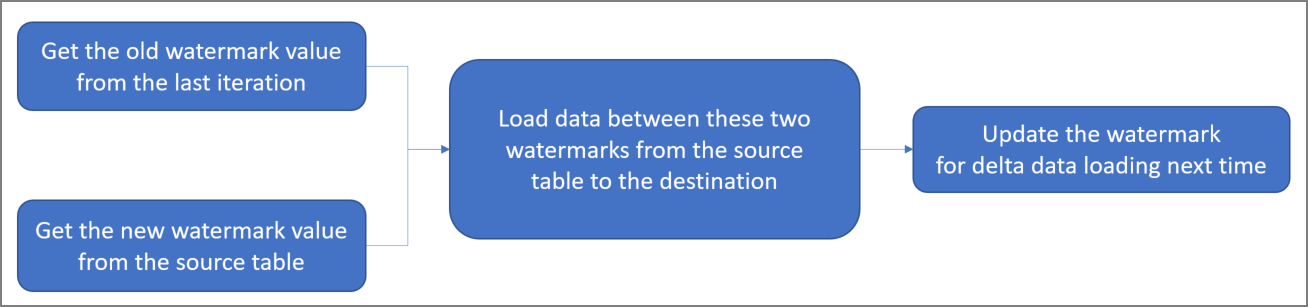
2. **CDC in Mapping Data Flow**: This approach uses data flow in ADF pipelines. It chains the source and target systems and synchronises them while giving you the luxury of transformation [4]. This approach does NOT require a timestamp or ID column ID as the changes can automatically be detected and extracted by ADF.



1. **Auto Incremental Extraction in Mapping Data Flow**: This approach uses data flow in ADF pipelines. This requires an incremental column [5].



4. **Customised Delta Extraction:** This approach uses a custom-build pipeline approach that uses a lookup with a stored procedure that finds the watermark value stored in the control table [3].



**How to connect AKV from ADB(Azure Databricks) and read the secrets of AKV?**

To connect Azure Key Vault (AKV) with Azure Data Factory (ADF) and read secrets securely, follow these steps:

**1. Set Up Azure Key Vault and Store Secrets**

* In the Azure portal, navigate to your **Azure Key Vault** resource.
* Add any secrets, certificate, Keys you need, such as connection strings, API keys, or passwords (e.g., db-password).

**2. Assign Access Permissions for ADF in Azure Key Vault**

* In the **Access Policies** of your Azure Key Vault, assign a policy that grants **"Get"** permissions on secrets to the **Azure Data Factory Managed Identity**.
* To do this:
  + Go to **Key Vault** -> **Access Policies** -> **Add Access Policy**.
  + Under **Secret Permissions**, select **Get** (and other necessary permissions if required).
  + Under **Principal**, search for and select your **Azure Data Factory** managed identity (this is the managed identity that represents ADF).
  + Click **Add** and **Save**.

**3. Add Key Vault to Linked Services in Azure Data Factory**

* In the **Azure Data Factory** portal, go to **Manage** -> **Linked Services**.
* Click **+ New** to create a new linked service.
* Search for **Azure Key Vault** and select it.
* Configure the linked service:
  + **Name**: Give it a name (e.g., AKV-LinkedService).
  + **Authentication method**: Choose **System-assigned managed identity**.
  + **Subscription**: Select your Azure subscription.
  + **Vault URI**: Provide the Key Vault URI (this is available in the Key Vault overview page).
* Click **Create** to finalize the linked service.

**4. Use the Key Vault Secret in Pipelines or Activities**

* Now you can use secrets stored in Azure Key Vault within activities (e.g., Linked Service configurations) in your ADF pipelines. Here’s how:
  + In activities like **Copy Data**, **Lookup**, etc., when you configure a **Linked Service** for sources, destinations, or connection settings, select the **"Enter manually"** option for sensitive fields.
  + Use the following syntax to reference a Key Vault secret:

@Microsoft.KeyVault(SecretUri=https://<your-key-vault-name>.vault.azure.net/secrets/<your-secret-name>)

* + Replace <your-key-vault-name> with the name of your Key Vault, and <your-secret-name> with the name of the secret.

**5. Example of Using a Key Vault Secret in a Dataset or Linked Service**

* For instance, to use a database password stored in Key Vault, configure the **password** field of the Linked Service to read from Key Vault:

"password": {

"type": "AzureKeyVaultSecret",

"secretName": "db-password"

}

* This will securely pull the db-password secret from Key Vault and use it in the ADF Linked Service.

**Additional Tips:**

* For using secrets in databrick or any other azure services, you have to update the principle in **access policy**.
* **Test Connection**: Always test the connection after setting up Key Vault secrets in your Linked Services to ensure they are configured correctly.
* **Rotation**: If secrets are updated or rotated in Key Vault, ADF will retrieve the updated secrets without further configuration.

Using Azure Key Vault with ADF ensures secure access to sensitive information and prevents the need to hard-code secrets in pipeline configurations.