Introduction to Azure Data Factory

Azure Data Factory

- In the world of big data, raw, unorganized data is often stored in relational, non-relational, and other storage systems
- The raw data doesn't have the proper context or meaning to provide meaningful insights to analysts, data scientists, or business decision makers
- Big data requires a service that can orchestrate and operationalize processes to refine these enormous stores of raw data into actionable business insights
- **Azure Data Factory is a managed cloud service that's built for these complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects**

Usage scenarios

- Gaming company
- Petabytes of game logs
- Analyze Logs to gain insights into customer preferences,
  - Demographics
  - Usage behavior
- Wants to identify up-sell and cross-sell opportunities,
  - develop compelling new features,
  - drive business growth
  - provide a better experience to its customers
- Need Ref from on-premises data center
  - customer information
  - Game information
  - Marketing campaign
- To extract insights
  - Joined data by using a Spark cluster in the cloud (Azure HDInsight)
  - Publish the transformed data into a cloud data warehouse such as Azure Synapse
- Want to automate this workflow
- Monitor and manage it on a daily schedule.
- Want to execute it when files land in a blob store container
- Azure Data Factory is the platform that solves such

Pipelines and activities

Pipelines and activities

- **A pipeline** is a logical grouping of activities that together perform a task.
  - A Data Factory can have one or more pipelines
- **For example,** a pipeline could contain a set of activities that
  - Ingest and clean log data,
  - then kick off a mapping data flow to analyze the log data
- The pipeline allows you to manage the activities as a set instead of each one individually
- You deploy and schedule the pipeline instead of the activities independently

Pipelines and activities

- **The activities** in a pipeline define actions to perform on your data
- **For example,** you may use a
  - Copy activity to copy data from SQL Server to an Azure Blob Storage.
  - Then, use a data flow activity or a Databricks Notebook activity to process and transform data from the blob storage to an Azure Synapse Analytics pool
  - Then top of which business intelligence reporting solutions are built

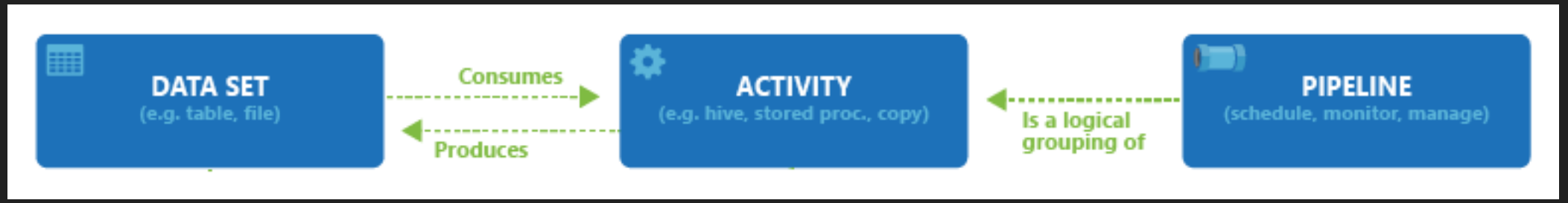
Pipelines and activities

- **Azure Data Factory have three groupings of activities:**

1. Data movement activities
2. Data transformation activities
3. Control flow activities

- An activity can take zero or more input datasets and produce one or more output datasets

Pipelines and activities



Credit: Azure Cloud

Relationship between pipeline, activity, and dataset

Pipelines and activities

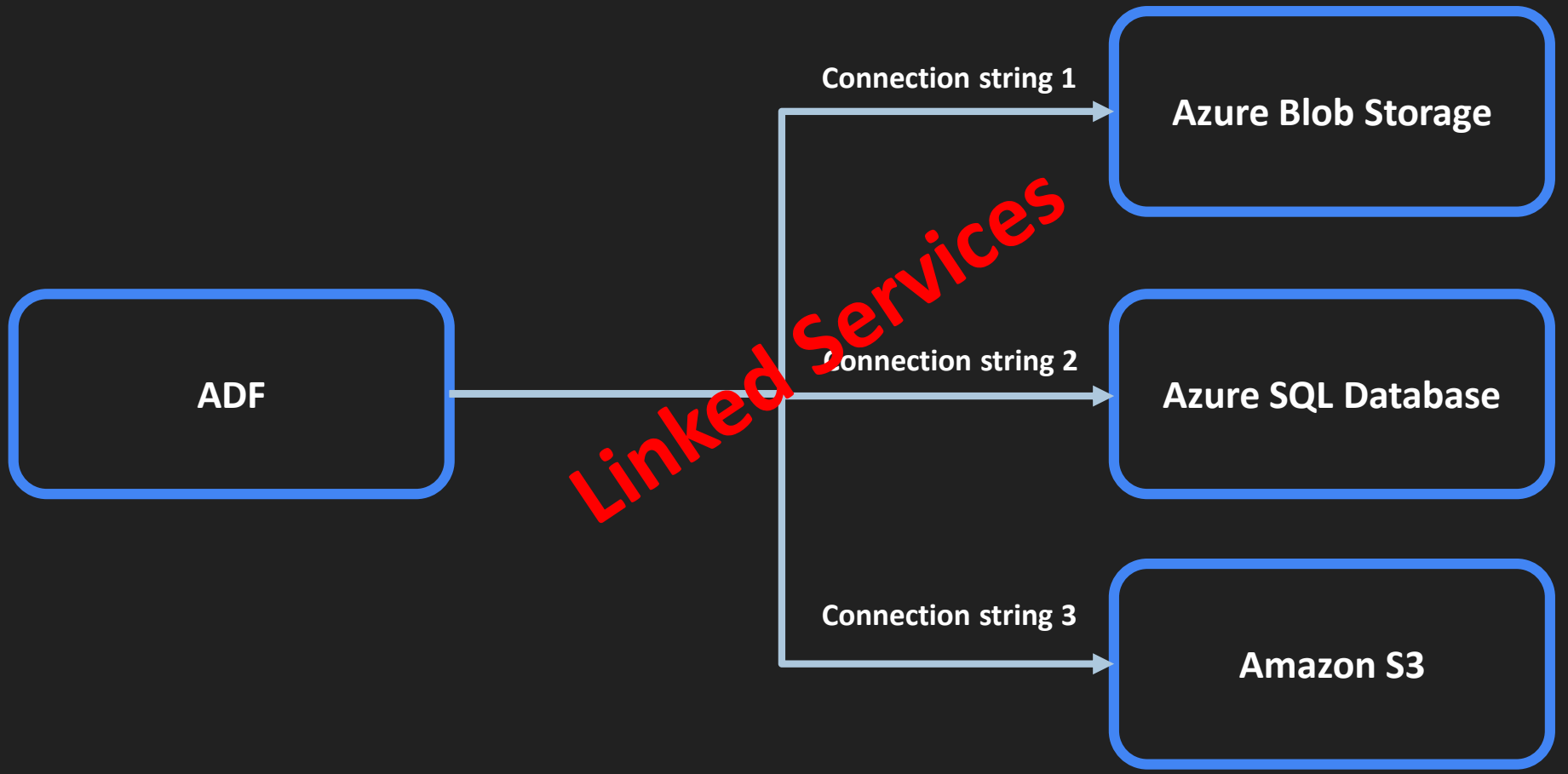
- Azure documentation link: https://learn.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities?tabs=data-factory

Linked services & Datasets

Linked services

- **Linked services** are much like connection strings,
  - which define the connection information needed for the service to connect to external resources.
  - These resources can be on-premises or in the cloud, and they can include data stores, compute resources, and other Azure services
- **For example,** an Azure Storage linked service links a storage account to the service

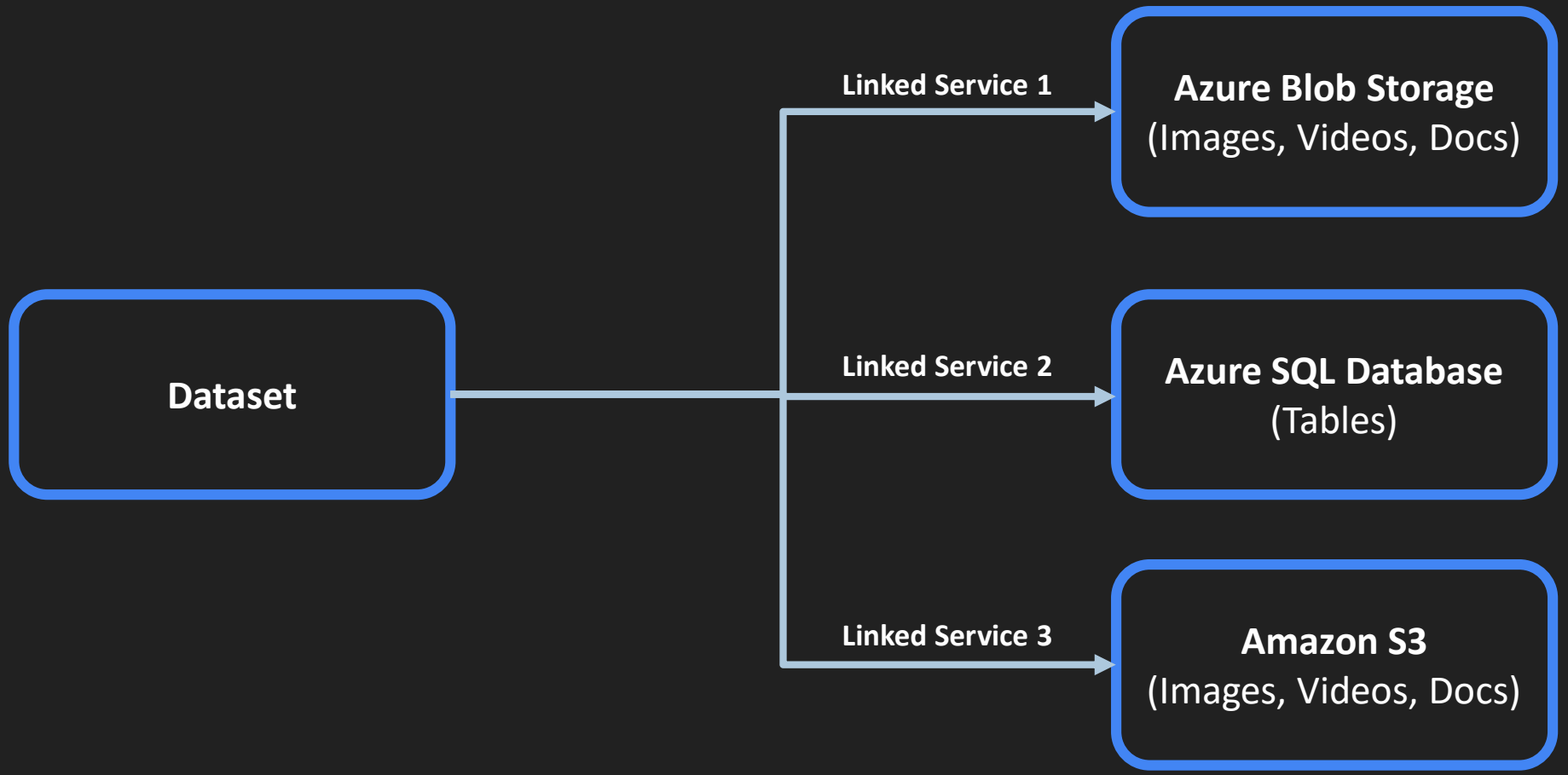
Linked Services



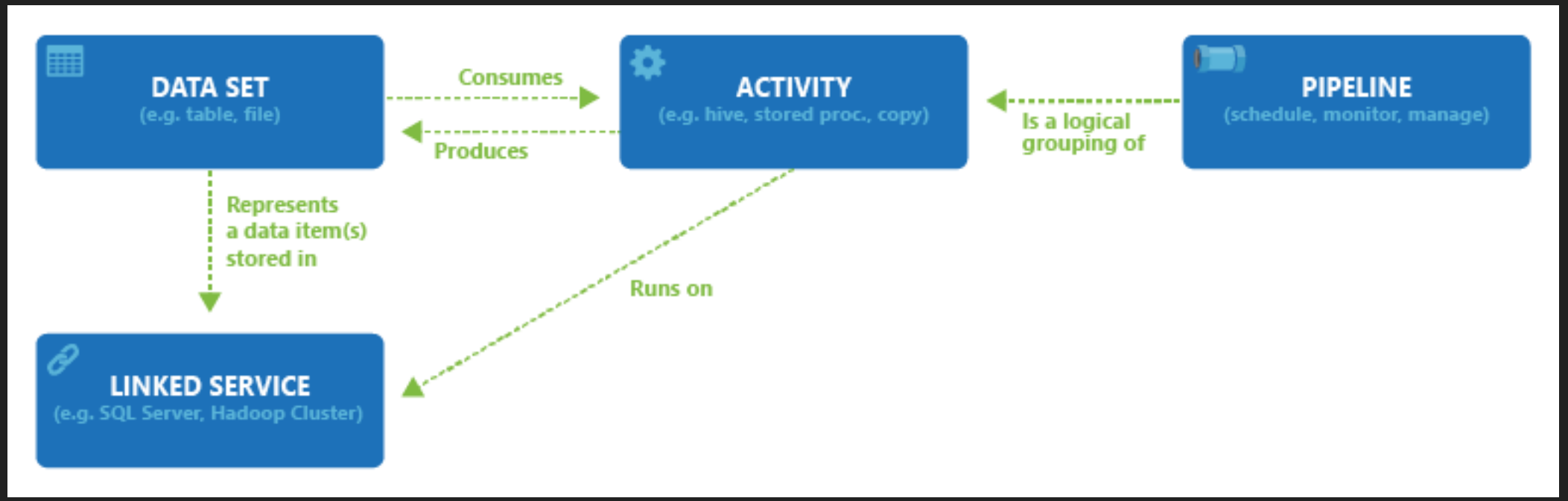
Datasets

- **A dataset** is a named view of data that simply points or references the data you want to use in your activities as inputs and outputs
- Datasets identify data within different data stores, such as tables, files, folders, and documents
- **For example,** an Azure Blob dataset specifies the blob container and folder in Blob Storage from which the activity should read the data
- Before you create a dataset, you must create a linked service to link your data store to the service

Datasets



Linked services & Datasets



Credit: Azure Cloud

Triggers

Triggers

- A trigger in Azure Data Factory is a mechanism that determines
  - when to start or
  - Invoke an end-to-end pipeline execution
- Triggers can be scheduled to run at
  - specific times
  - intervals, or
  - They can be event-based.

Types of Triggers

- **Schedule trigger:**
  - Runs a pipeline at a specified time or interval
- **Tumbling window trigger:**
  - Runs a pipeline on a regular schedule,
  - But only processes data that has arrived within a specific time window
- **Storage event trigger:**
  - Runs a pipeline when a specific event occurs in Azure Storage,
  - Such as when a new file is uploaded or when a file is deleted
- **Custom event trigger:**
  - Runs a pipeline when a specific event occurs in an external system.
  - Custom events can be raised by other Azure services, such as Azure Event Grid, or by third-party app

Schedule vs Tumbling Window Trigger

Schedule vs Tumbling Window Trigger

Characteristic	Schedule trigger	Tumbling window trigger
Type	Time-based	Time-based
Execution	Fire-and-forget	Tracks run status
State	No state	Retains state
Dependency	Cannot depend on other triggers	Can depend on other tumbling window triggers

Use Cases

- **Schedule triggers** are typically used to run pipelines on a regular basis, such as once a day or once a week
- **Tumbling window triggers** are typically used to run pipelines on a periodic interval, while also retaining state
- This makes them ideal for scenarios such as:

1. Processing streaming data in real time
2. Processing batch data in batches of a fixed size
3. Processing data from multiple sources in a coordinated manner

Examples

- **Schedule trigger:** Running a pipeline to copy data from a database to a data lake once a day
- **Tumbling window trigger:** Running a pipeline to process streaming data from a Kafka topic in 1-hour batches
- **Tumbling window trigger with dependency:** Running a pipeline to process data from a database in 1-hour batches, with the pipeline run depending on a successful run of a previous pipeline that processes data from a different database.

Custom Event Trigger

Custom Event Trigger

- **A custom event trigger** in Azure Data Factory (ADF) is a type of trigger that allows you to start a pipeline when a custom event is published to an Event Grid topic
- **This can be useful for a variety of scenarios, such as:**

1. Starting a pipeline when a new file is uploaded to a storage account
2. Starting a pipeline when a new row is inserted into a database table
3. Starting a pipeline when a message is received in a queue or service bus
4. Starting a pipeline when a custom event is published from another Azure service, such as Azure Logic Apps or Azure Functions

Custom Event Trigger

- **To create a custom event trigger in ADF, you will need to:**

1. Create an Event Grid topic
2. Create a pipeline in ADF
3. Add a custom event trigger to the pipeline
4. Configure the trigger to listen for the custom events that you want to start the pipeline
5. Publish the pipeline

- Once the pipeline is published, it will start whenever a custom event is published to the Event Grid topic that the trigger is listening for

Event Grid topic

- **An Event Grid topic** is a central place where you can publish and consume events. It acts as a router and distributor of events to event handlers
- You can publish events to a topic from any source, and you can subscribe to events from any topic by creating an event subscription

Event Grid topic

- Event Grid topics are used to decouple applications and services, and to enable event-driven architectures
- **They can be used to implement a variety of scenarios, such as:**

1. Starting a pipeline in Azure Data Factory when a new file is uploaded to a storage account
2. Sending a notification to a user when a new email arrives in their inbox
3. Triggering a workflow in Azure Logic Apps when a new row is inserted into a database table

- Event Grid topics are highly scalable and reliable, and they can be used to distribute events to any number of event handler

Integration Runtime

Integration Runtime

- **The Integration Runtime (IR)**
  - Compute infrastructure used by Azure Data Factory pipelines
  - To do ELT, ETL & data integration
- **Provide below capabilities**
  - Data Flow
  - Data movement
  - Activity dispatch
  - SSIS package execution

Integration Runtime

- In Data Factory pipelines,
  - an **activity defines** the action to be performed.
  - **A linked service defines** a target data store
- **An integration runtime provides the bridge between activities and linked services**
- Integration runtime is referenced by the linked service or activity
- Provides the compute environment where the activity is run directly
- This allows the activity to be performed in the closest possible region to the target data store

Integration Runtime Types

- Data Factory offers three types of Integration Runtime (IR)
- You should choose the type that best serves your data integration capabilities and network environment requirements

1. Azure Integration Runtime
2. Self-hosted Integration Runtime
3. Azure-SSIS Integration Runtime

[Hands-on] Azure Integration Runtime

Why Azure IR

- Why Need to create Azure IR type in ADF if it is already created by default?
  - To use a different compute type
  - To use a different region
  - To use a different concurrency level
  - To improve performance

Pipeline parameters and variables

Pipeline Parameters

- **Pipeline Parameters:**
  - Defined at the pipeline level
  - Cannot be modified during a pipeline run
  - Can be used to control the behavior of a pipeline and its activities,
  - Such as by passing in the connection details for a dataset
  - Path of a file to be processed

Pipeline Variables

- **Pipeline variables**
  - are values that can be set and modified during a pipeline run
- Unlike pipeline parameters, which are defined at the pipeline level & cannot be changed during a pipeline run
- pipeline variables can be set and modified within a pipeline using a Set Variable activity
- Pipeline variables can be used to store and manipulate data during a pipeline run,
  - Such as by storing the results of a computation
  - Current state of a process

System Variables

System Variables

- Built-in variables in ADF
- Can be used within every Data Factory pipeline
- Can be used to capture commonly used pipeline-related information & pass it dynamically anywhere within the pipeline

Usage of System Variables

- **Specifying dynamic file paths and folder names:**
  - This allows you to generate unique file names and paths for each pipeline run
- **Setting conditional expressions :**
  - To check the status of a previous activity before running the next activity
- **Passing data between activities:**
  - To pass data between activities in a pipeline.
  - This allows you to reuse data from one activity in another activity

System Variables

- Azure documentation: https://learn.microsoft.com/en-us/azure/data-factory/control-flow-system-variables

Connectors

Connectors

- **Connectors**
  - Components that allow you to connect to & interact with external data sources
- ADF provides a wide range of built-in connectors,
  - Connectors for on-premises and cloud data sources, SaaS applications, and other Azure services

Usage of Connectors

- You can use connectors to perform a variety of tasks, such as:
- **Ingesting data:**
  - From a variety of sources, such as on-premises databases, cloud storage, and SaaS applications
- **Loading data:**
  - To load data into a variety of destinations, such as Azure Data Lake Storage, Azure Synapse Analytics, and Azure SQL Database

Connectors

- Azure documentation: https://learn.microsoft.com/en-us/azure/data-factory/connector-overview

Control Flow Activities

- Set variable Activity
- Append Variable Activity
- Get Metadata Activity
- Execute Pipeline Activity
- Fail Activity
- Wait Activity
- ForEach Activity
- If Condition Activity
- Switch Activity
- Web & Webhook Activity
- Validation Activity
- Lookup Activity
- Filter Activity
- Until Activity
- Pipeline return variable

DataFlow Transformation

- Filter Transformation
- Aggregate Transformation
- Join Transformation
- Fuzzy join
- Conditional split
- Exists transformation
- Union transformation
- Lookup transformation
- Sort transformation
- Creating new Branch
- Select
- Pivot & unpivot
- Surrogate key transformation
- Window transformation
- Flatten transformation
- Assert transformation
- Cast transformation
- Parse transformation
- Rank transformation
- Stringify transformation