Welcome to ADF : Zero To Hero

Alias: C hawk

- ETL Exp: 9+ years
- Azure Exp: 7+ years
- Microsoft Certified Azure Data Engineer
- Worked for clients in US, Europe , Japan & India

Microsoft CERTIFIED
AZURE DATA ENGINEER
ASSOCIATE

What is Cloud Computing ?

Cloud Computing is the on-demand delivery of IT resources via the internet with pay-as-you-go pricing.

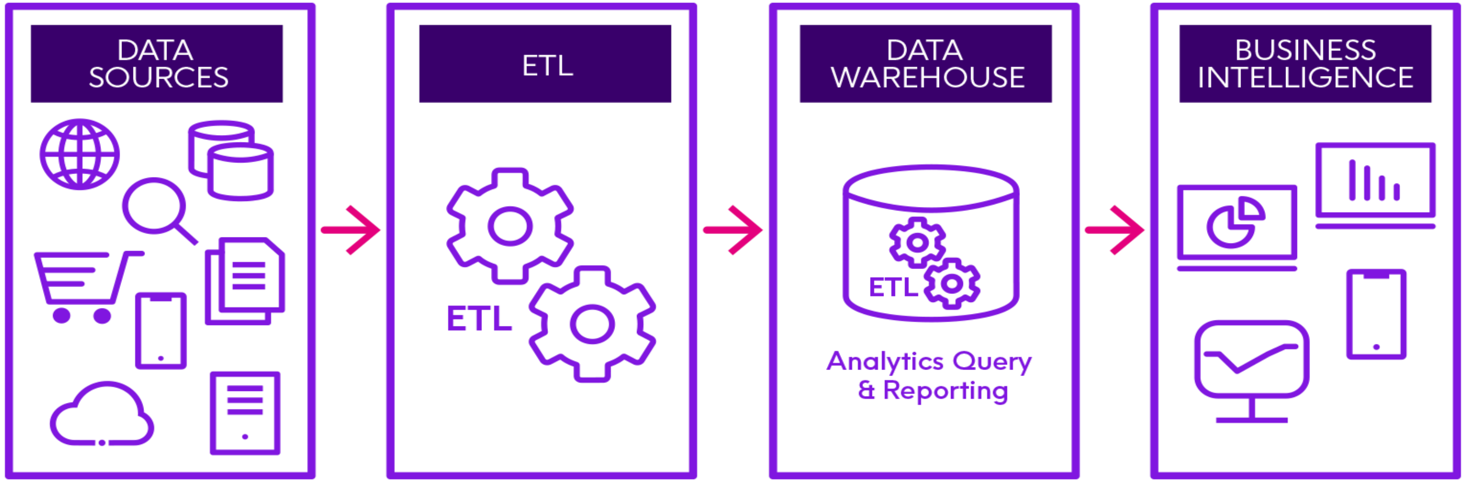
Advantages:

- ✓ Cost Saving
- ✓ Scalability
- ✓ Flexibility
- ✓ Loss prevention & Disaster Recovery
- ✓ Security
- ✓ High Availability
- ✓ Quick & Easy deployment across the Globe

What is Azure Data Factory(ADF) ?

- Azure Data Factory is Azure's cloud ETL service for scale-out serverless data integration and data transformation.
- It offers a code-free UI for intuitive authoring and single-pane-of-glass monitoring and management.
- It is the cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale.
- You can also lift and shift existing SSIS packages to Azure and run them with full compatibility in ADF.
- Behind the scene it uses Apache Spark technology to do the processing.

Extract-Transform-Load (ETL)



Module 2 : Environment Setup

In this module, we will create below resources -

- Azure Account
- Azure Blob Storage
- Azure Datalake Gen 2
- Azure SQL DB
- Azure Data Factory
- Azure SQL Server VM

Create Azure Account

Azure Portal Walkthrough

Create Azure Storage Account

Create Azure DataLake Gen 2

Create Azure SQL DB

Create Azure Data Factory

Create Azure SQL Server VM

Module 3: Building Blocks of ADF

- Pipelines
- Activities
- Dataset
- Linked Services
- Triggers
- Interconnection of above components

Pipelines

- A pipeline is a **logical grouping of activities** that performs a unit of work. Together, the activities in a pipeline perform a task.

  Example:
  A pipeline can contain group of activities that ingest data from data lake storage to Azure Blob storage and then store the ingestion details inside a SQL DB.

Activities

- An activity represent an **elementary or atomic processing step** in a pipeline.

  Example:
  Use a copy activity to copy data from one data store to another data store.

Linked Service

- Linked Services are much like connection strings, which define the connection information that's needed for Data Factory to connect to external resources.

Example:

An Azure Datalake Gen2 linked service specifies a connection string to connect to the Datalake Gen2 storage account.

Dataset

- Datasets represents data structures within the data stores, which simply point to or reference the data you want to use in your activity.

  Example:
  A dataset can point to a particular file inside a container of a Datalake Gen2 storage.

Supported File Formats

Azure Data Factory supports the following file formats –

- Avro format
- Binary format
- Delimited text format
- Excel format
- JSON format
- ORC format
- Parquet format
- XML format

https://docs.microsoft.com/en-us/azure/data-factory/supported-file-formats-and-compression-codecs

JSON

```
data = {“Key1":“Value1",
           “Key2":“Value2"};


myObj = {"name":"John", "age":30, "car":null};
```

Avro / ORC / Parquet

- All three of the formats are optimized for storage on Hadoop, and provide some degree of compression.
- Files stored in ORC, Parquet, and Avro formats can be split across multiple disks, which lends themselves to scalability and parallel processing.
- All three formats carry the data schema in the files themselves, which is to say they're self-described.

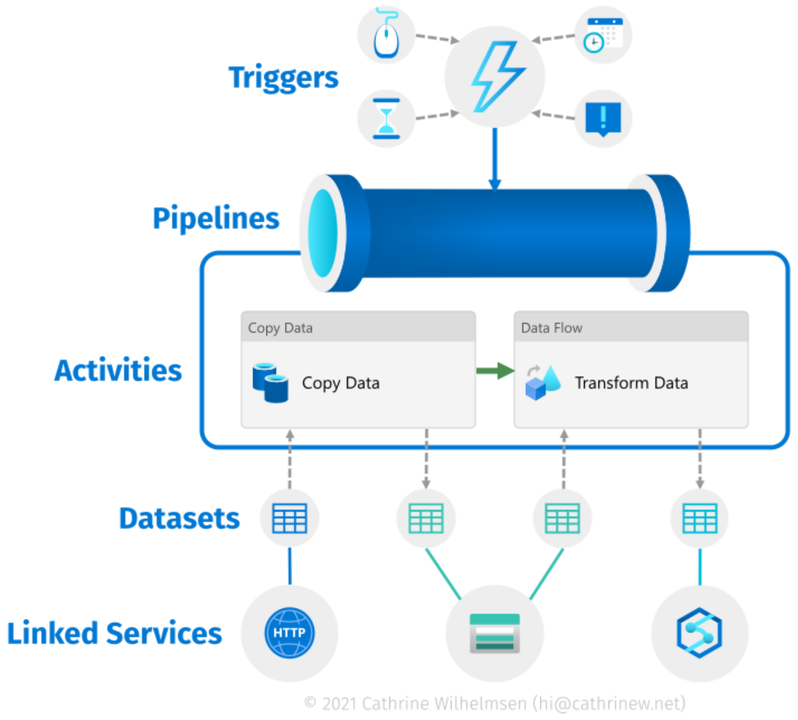
- The biggest difference between ORC, Avro, and Parquet is how the store the data. Parquet and ORC both store data in columns, while Avro stores data in a row-based format. By their very nature, column-oriented data stores are optimized for read-heavy analytical workloads, while row-based databases are best for write-heavy transactional workloads.

Triggers

- Triggers Determines when a pipeline execution needs to be kicked off.

- There are different kind of triggers –
  - Schedule Trigger
  - Tumbling Window Trigger
  - Storage Event Based Trigger
  - Custom Event Base Trigger

Interconnection of components



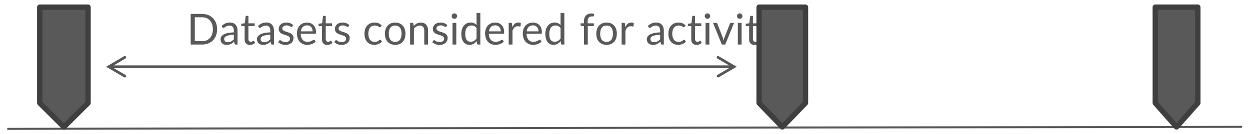
Copy Data from Blob to ADLS

Source
Sink/Destination

Azure Blob Storage Gen2 $\longrightarrow$ Copy

Azure DataLake

Copy Data from Blob to ADLS



Start time End time Activity(Copy data)

Real World Scenario #3

Dynamically add filename while copying data from source to sink

Copy Data from SQL DB to ADLS

Source
Sink/Destination

⟶ Copy

Azure SQL DB
Azure
DataLake Gen2

Setting Up Self Hosted IR

Copy Data On Prem DB to ADLS

On Prem DB
Sink/Destination

$\longrightarrow$ Copy

Azure SQL VM Azure
DataLake Gen2

Ingest data from API to ADLS

Public API to consume **sample JSON data without any authentication** –

http://dummy.restapiexample.com/api/v1/employees

Parameters

Parameters are external values to be passed into pipelines, datasets, linked services, and data flows.

By parameterizing resources, you can reuse them with different values each time.

Variables

Variables are internal to pipelines.

Difference between Paramaters and Variables - We cannot reset paramater values inside pipeline whereas we can reset variable values multiple times.

System Variables

Pipeline scope
Schedule trigger scope
Tumbling window trigger scope
Storage event trigger scope
Custom event trigger scope

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

Filter Activity

You can use a Filter activity in a pipeline to apply a filter expression to an input **array.**

To use individual array elements use **item()**

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-filter-activity

ForEach Activity

The ForEach Activity defines a repeating control flow in an Azure Data Factory. This activity is used to iterate over a collection and executes specified activities in a loop. The loop implementation of this activity is similar to Foreach looping structure in programming languages.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-for-each-activity

Lookup Activity

Lookup activity can **read / retrieve a dataset from any of the data sources** supported by data factory. You can use it to dynamically determine which objects to operate on in a subsequent activity, instead of hard coding the object name. Some object examples are files and tables.

Lookup activity reads and returns the content of a configuration file or table. It also returns the result of executing a query or stored procedure. The output can be a singleton value or an array of attributes, which can be consumed in a subsequent copy, transformation, or control flow activities like ForEach activity.

Limitation: **Fetched / Looked up data should be less than 5000 rows or 4 MB in size.**

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-lookup-activity

Get Metadata Activity

You can use the Get Metadata activity to retrieve the metadata of any data in Azure Data Factory. You can use the output from the Get Metadata activity in conditional expressions to perform validation, or consume the metadata in subsequent activities.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-get-metadata-activity

If Condition Activity

It executes a set of activities when the condition evaluates to true and another set of activities when the condition evaluates to false.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-if-condition-activity

Wait activity

When you use a Wait activity in a pipeline, the pipeline waits for the specified period of time before continuing with execution of subsequent activities.

Switch activity

The Switch activity provides the same functionality that a switch statement provides in programming languages.

It evaluates a set of activities corresponding to a case that matches the condition evaluation.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-switch-activity

Validation Activity

You can use a Validation in a pipeline to ensure the pipeline only continues execution once it has validated the attached dataset reference exists, that it meets the specified criteria, or timeout has been reached.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-validation-activity

Stored Procedure Activity

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system (RDBMS) as a group, so it can be reused and shared by multiple programs.

Using Stored Procedure Activity you can execute a stored procedure inside your DB via ADF

https://docs.microsoft.com/en-us/azure/data-factory/transform-data-using-stored-procedure

Delete Activity

You can use the Delete Activity in Azure Data Factory to delete files or folders from on-premises storage stores or cloud storage stores. Use this activity to clean up or archive files when they are no longer needed.

Fail Activity

You might occasionally want to throw an error in a pipeline intentionally. A Lookup activity might return no matching data, or a Custom activity might finish with an internal error. Whatever the reason might be, now you can use a Fail activity in a pipeline and customize both its error message and error code.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-fail-activity

Execute Pipeline

The Execute Pipeline activity allows a Data Factory pipeline to invoke another pipeline.

DataFlow activity

Use the Data Flow activity to transform and move data via mapping data flows.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-execute-data-flow-activity

Module 8: Mapping DataFlows

Mapping data flows are **visually designed data transformations** in Azure Data Factory. Data flows allow data engineers to develop data transformation logic without writing code. The resulting data flows are executed as activities within Azure Data Factory pipelines that **use scaled-out Apache Spark clusters**. Data flow activities can be operationalized using existing Azure Data Factory scheduling, control, flow, and monitoring capabilities.

https://docs.microsoft.com/en-us/azure/data-factory/concepts-data-flow-overview

Source transformation

A source transformation configures your data source for the data flow. When you design data flows, your first step is always configuring a source transformation. To add a source, select the **Add Source** box in the data flow canvas.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-source

Sink transformation

After you finish transforming your data, write it into a destination store by using the sink transformation. Every data flow requires at least one sink transformation, but you can write to as many sinks as necessary to complete your transformation flow. To write to additional sinks, create new streams via new branches and conditional splits.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-sink

Select transformation

Use the select transformation to **rename, drop, or reorder columns**. This transformation **doesn't alter row data**, but chooses which columns are propagated downstream.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-select

Filter transformation

The Filter transforms allows row filtering based upon a condition. The output stream includes all rows that matching the filtering condition. The filter transformation is similar to a WHERE clause in SQL.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-filter

Conditional split transformation

The conditional split transformation routes data rows to different streams based on matching conditions.

The transformation evaluates expressions, and based on the results, directs the data row to the specified stream.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-conditional-split

Union transformation

Union will combine multiple data streams into one, with the SQL Union of those streams as the new output from the Union transformation. All of the schema from each input stream will be combined inside of your data flow, without needing to have a join key.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-union

Exists transformation

The exists transformation is a row filtering transformation that checks whether your data exists in another source or stream.

The output stream includes all rows in the left stream that either exist or don't exist in the right stream.

The exists transformation is similar to SQL WHERE EXISTS and SQL WHERE NOT EXISTS.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-exists

Sort transformation

The sort transformation allows you to sort the incoming rows on the current data stream. You can choose individual columns and sort them in ascending or descending order.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-sort

Rank transformation

Use the rank transformation to generate an ordered ranking based upon sort conditions specified by the user.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-rank

Aggregate transformation

The Aggregate transformation defines aggregations of columns in your data streams. Using the Expression Builder, you can define different types of aggregations such as SUM, MIN, MAX, and COUNT grouped by existing or computed columns.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-aggregate

Join transformation

Use the join transformation to combine data from two sources or streams in a mapping data flow. The output stream will include all columns from both sources matched based on a join condition.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-join

Pivot Transformation

Use the pivot transformation to create multiple columns from the unique row values of a single column. Pivot is an aggregation transformation where you select group by columns and generate pivot columns using aggregate functions.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-pivot

Lookup transformations

Use the lookup transformation to reference data from another source in a data flow stream. The lookup transformation appends columns from matched data to your source data.

A lookup transformation is similar to a left outer join. All rows from the primary stream will exist in the output stream with additional columns from the lookup stream.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-lookup

Derived Column

Use the derived column transformation to generate new columns in your data flow or to modify existing fields.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-derived-column

https://docs.microsoft.com/en-us/azure/data-factory/concepts-data-flow-expression-builder

New Branch

Add a new branch to do multiple sets of operations and transformations against the same data stream. Adding a new branch is useful when you want to use the same source to for multiple sinks or for self-joining data together.

Surrogate key transformation

Use the surrogate key transformation 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.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-surrogate-key

Flatten transformation

Use the flatten transformation to take array values inside hierarchical structures such as JSON and unroll them into individual rows. This process is known as denormalization.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-flatten

Alter Row transformation

Use the Alter Row transformation to **set insert, delete, update, and upsert policies on rows.** You can add one-to-many conditions as expressions. These conditions should be specified **in order of priority**, as each row will be marked with the policy corresponding to the first-matching expression. Each of those conditions can result in a row (or rows) being inserted, updated, deleted, or upserted.

Alter Row transformations will only operate on database, REST, or CosmosDB sinks in your data flow. The actions that you assign to rows (insert, update, delete, upsert) won't occur during debug sessions. Run an Execute Data Flow activity in a pipeline to enact the alter row policies on your database tables.

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-alter-row

Assert Transformation

The assert transformation enables you to build **custom rules** inside your mapping data flows **for data quality and data validation**. You can build rules that will determine whether values meet an expected value domain. The assert transformation also allows you to set custom error messages when data validation rules are not met.

- Expect true – for data type Validations
- Expect unique – for uniqueness check in a column
- Expect exists – for referencing column value to another dataset
- Use isError() and hasError('assert id') to get bad rows

https://docs.microsoft.com/en-us/azure/data-factory/data-flow-assert

Expressions & Functions

We use expressions to make activities and pipelines dynamic in nature.

Expressions will be converted to proper values during runtime.

Must Read:
https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions

Module 10: Triggers

Triggers determine when a pipeline execution needs to be kicked off. Using triggers you can control when to execute your pipeline.

Pipelines and triggers have many to many relationship (except for Tumbling window trigger)

https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipeline-execution-triggers

Types of Triggers

Below are currently available trigger types in ADF –

- Schedule Trigger – Invokes a pipeline on pre-specified clock timings.
- Tumbling Window Trigger – Operates on periodic interval, while also retaining the state.
- Storage Event Based Trigger- runs a pipeline against events happening in a Storage account, such as the arrival of a file, or the deletion of a file in Azure Blob Storage account.
- Custom Event Based Trigger - processes and handles custom articles in Event Grid.

Schedule Trigger

A schedule trigger runs pipelines on a wall-clock schedule. This trigger supports periodic and advanced calendar options.

For example, the trigger supports intervals like "weekly" or "Monday at 5:00 PM and Thursday at 9:00 PM.“

Pipelines and Scheduled triggers have a many-to-many relationship. Multiple triggers can kick off a single pipeline. A single trigger can kick off multiple pipelines.

@trigger().scheduledTime
@trigger().startTime

https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipeline-execution-triggers#schedule-trigger-with-json

Tumbling Window Trigger

Tumbling windows are a series of **fixed-sized, non-overlapping, and contiguous** time intervals.

Tumbling window triggers are a type of trigger that fires at a periodic time interval from a specified start time, **while retaining state.**

A tumbling window trigger has a one-to-one relationship with a pipeline and can only reference a singular pipeline.

@trigger().outputs.windowStartTime
@trigger().outputs.windowEndTime

https://docs.microsoft.com/en-us/azure/data-factory/how-to-create-tumbling-window-trigger?tabs=data-factory%2Cazure-powershell

https://docs.microsoft.com/en-us/azure/data-factory/tumbling-window-trigger-dependency

Storage Event based Trigger

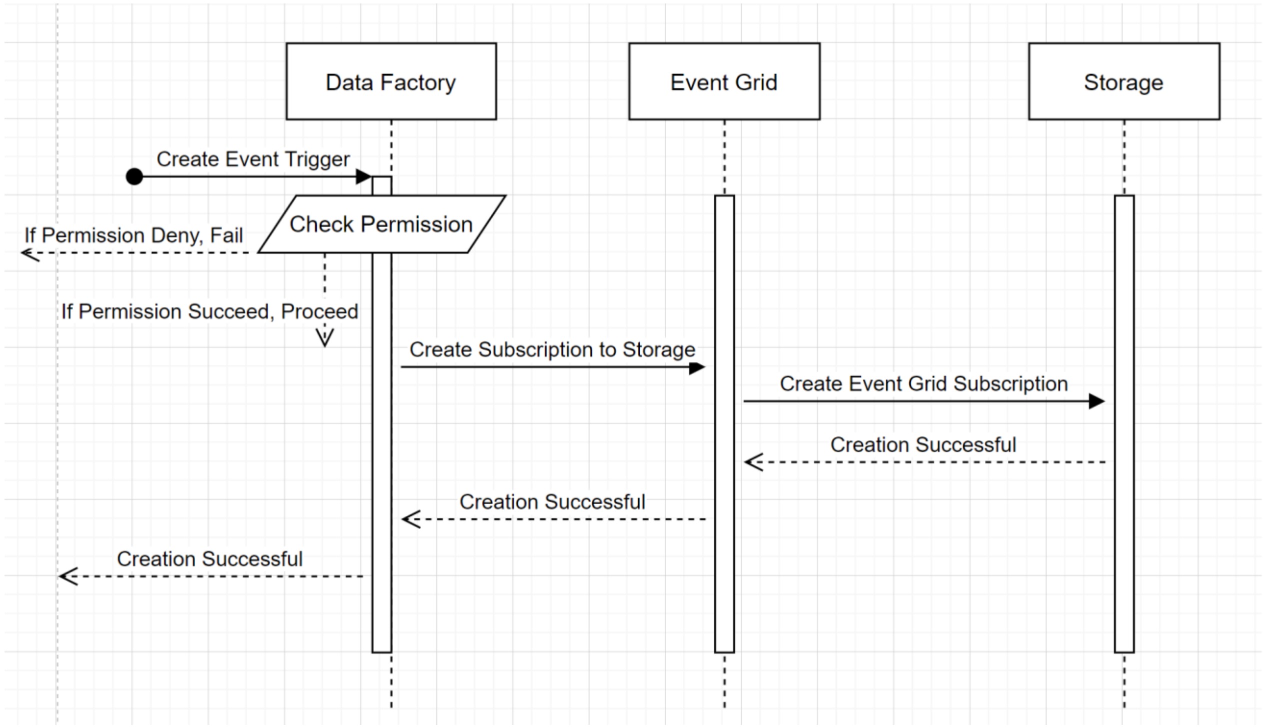
Data integration scenarios often require customers to trigger pipelines based on events happening in storage account, such as the arrival or deletion of a file in Azure Blob Storage account.

Invokes a pipeline against events happening in a Storage account, such as the arrival of a file, or the deletion of a file in Azure Blob Storage account.

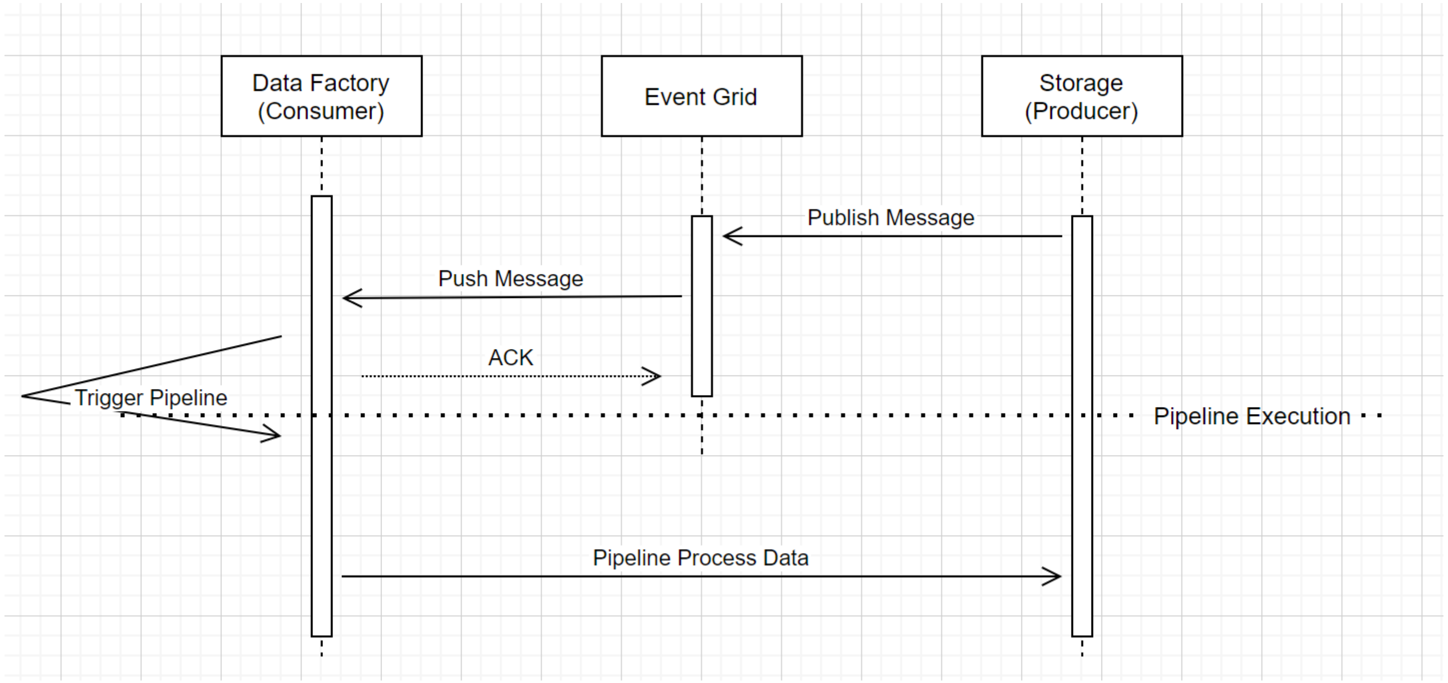
Data Factory pipelines natively integrate with Azure Event Grid, which lets you trigger pipelines on such events.

https://docs.microsoft.com/en-us/azure/data-factory/how-to-create-event-trigger?tabs=data-factory

Storage Event Trigger Creation



Storage Event Trigger work flow



Storage Event Trigger System Variables

Variable Name	Description
@triggerBody().fileName	Name of the file whose creation or deletion caused the trigger to fire.
@triggerBody().folderPath	Path to the folder that contains the file specified by @triggerBody().fileName. The first segment of the folder path is the name of the Azure Blob Storage container.
@trigger().startTime	Time at which the trigger fired to invoke the pipeline run.

Module 11: Pipeline Runs Monitoring

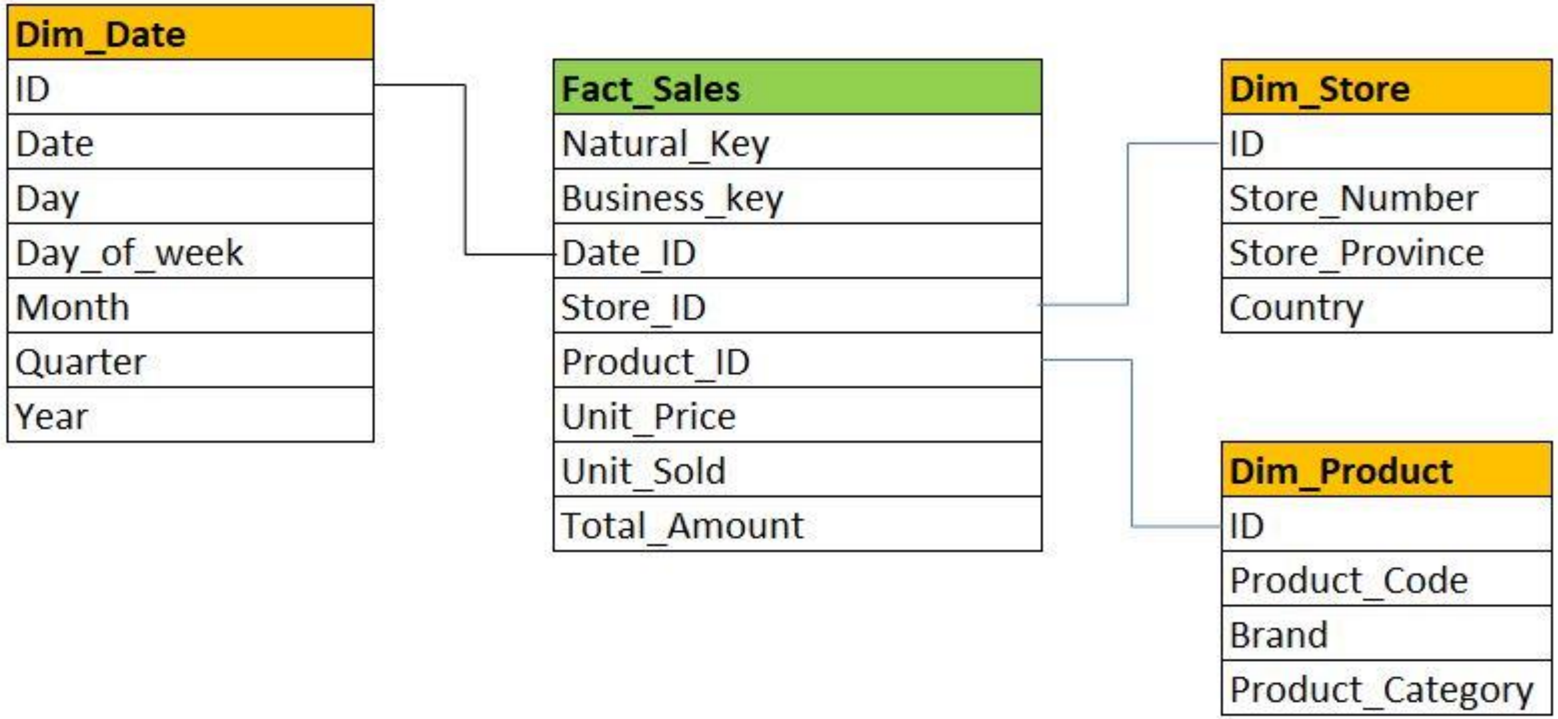
Facts & Dimensions

Fact in data warehousing describes quantitative **transactional data** like measurements, metrics, or the values ready for analysis. These include order numbers, ticket numbers, transaction numbers, transaction currency, transaction amount etc. Generally Fact tables consist of Keys(PK & FK) and measures (M).

Dimensions are companions to facts and are attributes of facts like the date of a sale. They describe different objects and are denormalized because of one-to-many relationships.

Facts form foreign key relationships with different dimension tables.

Fact & Dimension tables



Dim_Date
ID
Date
Day
Day_of_week
Month
Quarter
Year

Fact_Sales
Natural_Key
Business_key
Date_ID
Store_ID
Product_ID
Unit_Price
Unit_Sold
Total_Amount

Dim_Store
ID
Store_Number
Store_Province
Country

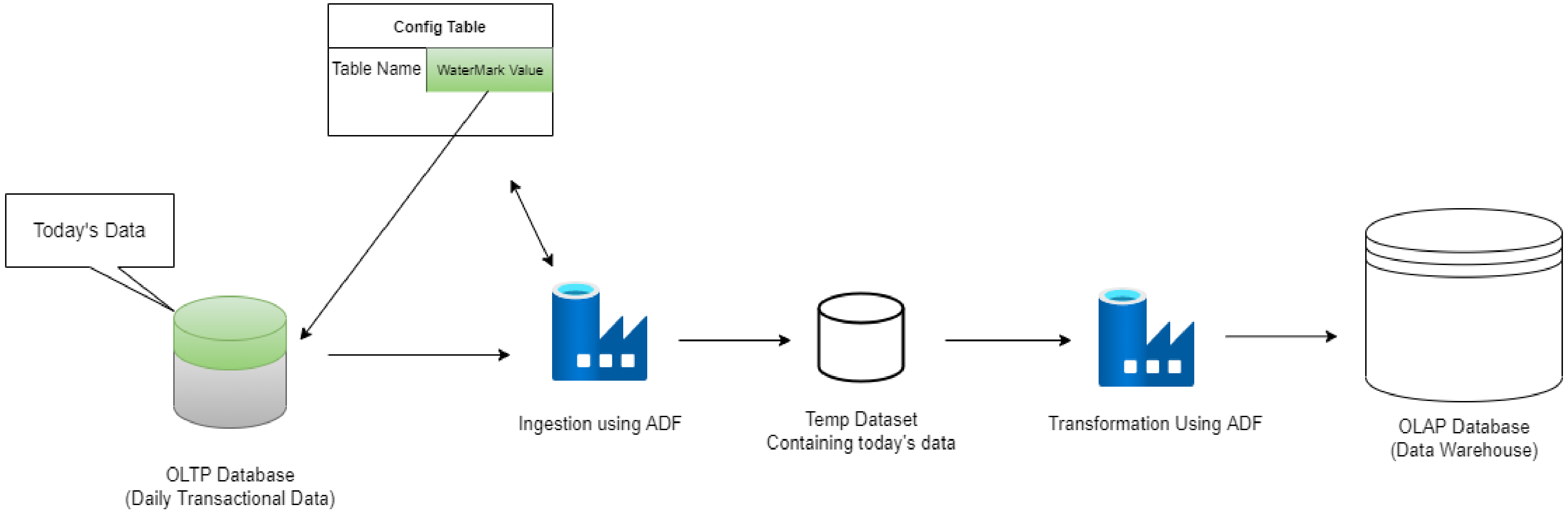
Dim_Product
ID
Product_Code
Brand
Product_Category

Slowly changing dimension

https://en.wikipedia.org/wiki/Slowly_changing_dimension

RWS #4 - Incremental Loading

Incremental loading is the activity of **loading only new or updated records** from a source into sink. Incremental loads are useful because they run efficiently when compared to full loads, and particularly for large data sets.



Real World Scenario #1

Copy files from a folder, which was modified within last one day, to a sink folder.

https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions

Real World Scenario #2

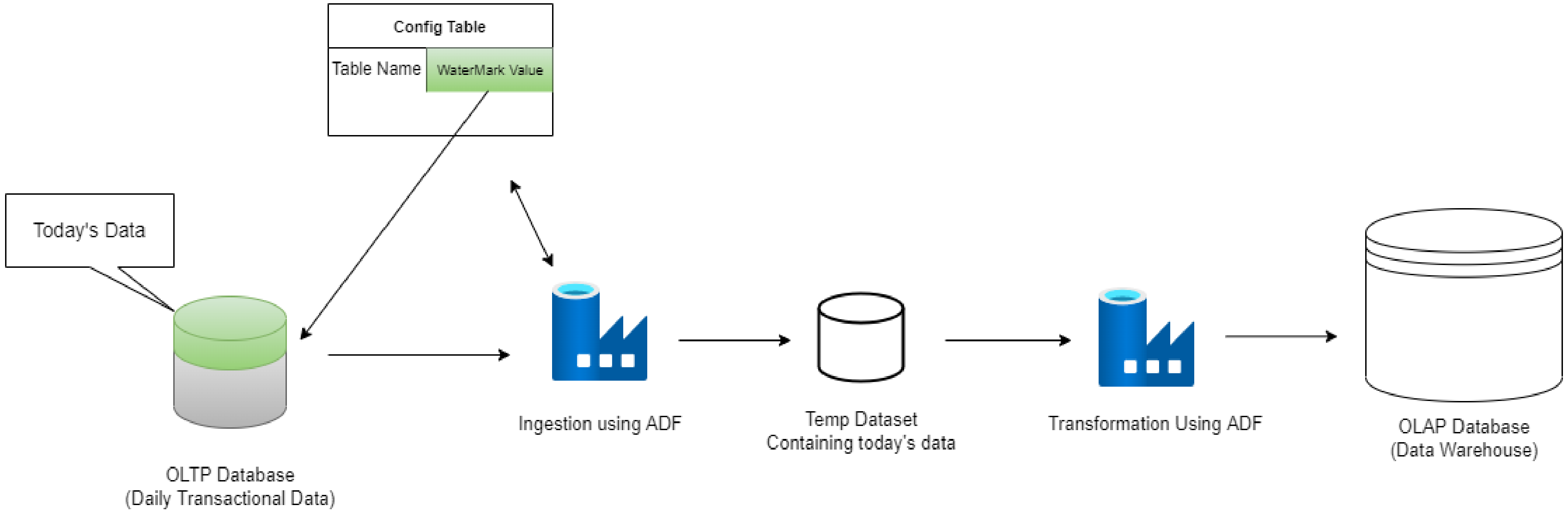
Bulk copy all the tables present in Database to ADLS.

Real World Scenario #3

Dynamically add filename while copying data from source to sink

RWS #4 - Incremental Loading

Incremental loading is the activity of **loading only new or updated records** from a source into sink. Incremental loads are useful because they run efficiently when compared to full loads, and particularly for large data sets.



Real World Scenario #5

Insert pipeline run details in a SQL DB for audit purpose.

Congratulations on Course Completion !

Thank You