**Integrate data with Azure Data Factory or Azure Synapse Pipeline**

# Understand Azure Data Factory

The need to trigger the batch movement of data, or to set up a regular schedule is a requirement for most analytics solutions. Azure Data Factory (ADF) is the service that can be used to fulfill such a requirement. ADF provides a cloud-based data integration service that orchestrates the movement and transformation of data between various data stores and compute resources.

Azure Data Factory 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. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores. You can build complex ETL processes that transform data visually with data flows or by using compute services such as Azure HDInsight Hadoop, Azure Databricks, and Azure Synapse Analytics.

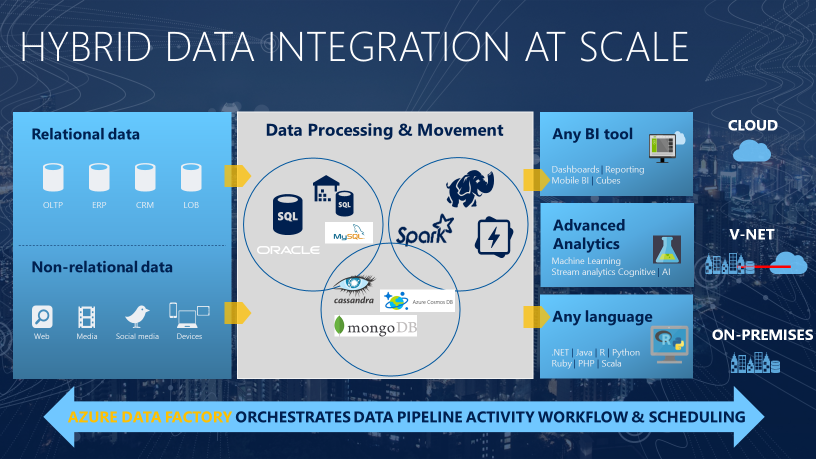
Much of the functionality of Azure Data Factory appears in Azure Synapse Analytics as a feature referred to as Pipelines, which enables you to integrate data pipelines between SQL Pools, Spark Pools and SQL Serverless, providing a one stop shop for all your analytical needs.

## What is meant by orchestration

To use an analogy, think about a symphony orchestra. The central member of the orchestra is the conductor. The conductor does not play the instruments, they simply lead the symphony members through the entire piece of music that they perform. The musicians use their own skills to produce particular sounds at various stages of the symphony, so they may only learn certain parts of the music. The conductor orchestrates the entire piece of music, and therefore is aware of the entire score that is being performed. They will also use specific arm movements that provide instructions to the musicians how a piece of music should be played.

ADF can use a similar approach, whilst it has native functionality to ingest and transform data, sometimes it will instruct another service to perform the actual work required on its behalf, such as a Databricks to execute a transformation query. So, in this case, it would be Databricks that performs the work, not ADF. ADF merely orchestrates the execution of the query, and then provides the pipelines to move the data onto the next step or destination.

It also provides rich visualizations to display the lineage and dependencies between your data pipelines, and monitor all your data pipelines from a single unified view to easily pinpoint issues and setup monitoring alerts.



# Describe data integration patterns

Microsoft Azure provides a variety of data platform services that enables you to perform different types of analytics. Whether it is a descriptive analytics solution in a data warehouse, through to predictive analytics within HDInsight, Azure Databricks or Machine Learning Services. There is a need for a service to deal with the important aspect of data integration.

Data integration firstly involves the collection of data from one or more sources. Optionally, it typically then includes a process where the data may be cleansed and transformed, or perhaps augmented with additional data and prepared. Finally, the amalgamated data is stored in a data platform service that handles the type of analytics that you want to perform. This process can be automated by Azure Data Factory in a pattern known as Extract, Transform and Load (ETL).

### **Extract**

During the extraction process, data engineers define the data and its source:

* **Define the data source**: Identify source details such as the resource group, subscription, and identity information such as a key or secret.
* **Define the data**: Identify the data to be extracted. Define data by using a database query, a set of files, or an Azure Blob storage name for blob storage.

### **Transform**

* **Define the data transformation**: Data transformation operations can include splitting, combining, deriving, adding, removing, or pivoting columns. Map fields between the data source and the data destination. You might also need to aggregate or merge data.

### **Load**

* **Define the destination**: During a load, many Azure destinations can accept data formatted as a JavaScript Object Notation (JSON), file, or blob. You might need to write code to interact with application APIs.

Azure Data Factory offers built-in support for Azure Functions. You'll also find support for many programming languages, including Node.js, .NET, Python, and Java. Although Extensible Markup Language (XML) was common in the past, most systems have migrated to JSON because of its flexibility as a semistructured data type.

* **Start the job**: Test the ETL job in a development or test environment. Then migrate the job to a production environment to load the production system.
* **Monitor the job**: ETL operations can involve many complex processes. Set up a proactive and reactive monitoring system to provide information when things go wrong. Set up logging according to the technology that will use it.

## **ETL tools**

As a data engineer, there are several available tools for ETL. Azure Data Factory provides nearly 100 enterprise connectors and robust resources for both code-free and code-based users to accomplish their data movement and transformation needs.

## **Evolution from ETL**

Azure has opened the way for technologies that can handle unstructured data at an unlimited scale. This change has shifted the paradigm for loading and transforming data from ETL to extract, load, and transform (ELT).

The benefit of ELT is that you can store data in its original format, be it JSON, XML, PDF, or images. In ELT, you define the data's structure during the transformation phase, so you can use the source data in multiple downstream systems.

In an ELT process, data is extracted and loaded in its native format. This change reduces the time required to load the data into a destination system. The change also limits resource contention on the data sources.

The steps for the ELT process are the same as for the ETL process. They just follow a different order.

Another process like ELT is called extract, load, transform, and load (ELTL). The difference with ELTL is that it has a final load into a destination system.

There are two common types of data integration patterns that can be supported by Azure Data Factory.

### **Modern Data Warehouse workloads:**

A Modern Data Warehouse is a centralized data store that provides descriptive analytics and decision support services across the whole enterprise using structured, unstructured, or streaming data sources. Data flows into the warehouse from multiple transactional systems, relational databases, and other data sources on a periodic basis. The stored data is used for historical and trend analysis reporting. The data warehouse acts as a central repository for many subject areas and contains the "single source of truth."

Azure Data factory is typically used to automate the process of extracting, transforming, and loading the data through a batch process against structured and unstructured data sources.

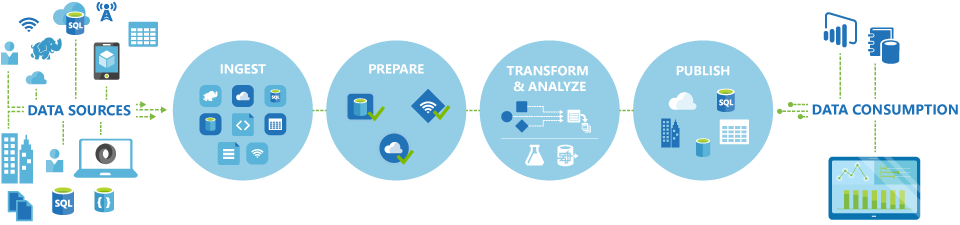
### **Advanced Analytical Workloads**

You can perform advanced analytics in the form of predictive or preemptive analytics using a range of Azure data platform services. Azure Data Factory provides the integration from source systems into a Data Lake store, and can initiate compute resources such as Azure Databricks, or HDInsight to use the data to perform the advanced analytical work

# Explain the data factory process

## Data-driven workflows

The pipelines (data-driven workflows) in Azure Data Factory typically perform the following four steps:



## Connect and collect

The first step in building an orchestration system is to define and connect all the required sources of data together, such as databases, file shares, and FTP web services. The next step is to ingest the data as needed to a centralized location for subsequent processing.

## Transform and enrich

Compute services such as Databricks and Machine Learning can be used to prepare or produce transformed data on a maintainable and controlled schedule to feed production environments with cleansed and transformed data. In some instances, you may even augment the source data with additional data to aid analysis, or consolidate it through a normalization process to be used in a Machine Learning experiment as an example.

## Publish

After the raw data has been refined into a business-ready consumable form from the transform and enrich phase, you can load the data into Azure Data Warehouse, Azure SQL Database, Azure Cosmos DB, or whichever analytics engine your business users can point to from their business intelligence tools

## Monitor

Azure Data Factory has built-in support for pipeline monitoring via Azure Monitor, API, PowerShell, Azure Monitor logs, and health panels on the Azure portal, to monitor the scheduled activities and pipelines for success and failure rates.

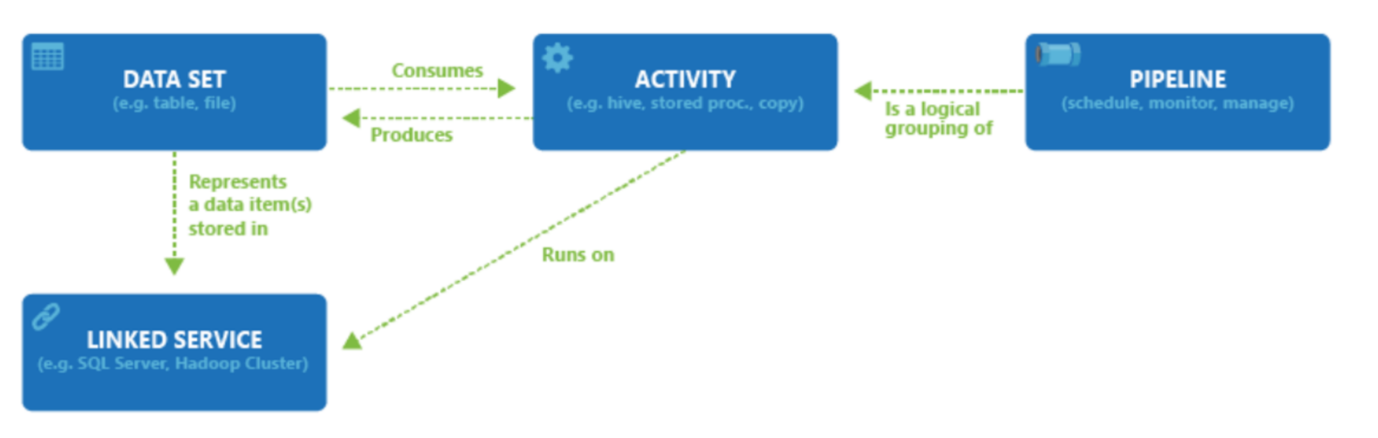
**Understand Azure Data Factory components**

An Azure subscription might have one or more Azure Data Factory instances. Azure Data Factory is composed of four core components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.

Data Factory supports a wide variety of data sources that you can connect to through the creation of an object known as a **Linked Service**, which enables you to ingest the data from a data source in readiness to prepare the data for transformation and/or analysis. In addition, Linked Services can fire up compute services on demand. For example, you may have a requirement to start an on-demand HDInsight cluster for the purpose of just processing data through a Hive query. So Linked Services enables you to define data sources, or compute resource that is required to ingest and prepare data.

With the linked service defined, Azure Data Factory is made aware of the datasets that it should use through the creation of a **Datasets** object. Datasets represent data structures within the data store that is being referenced by the Linked Service object. Datasets can also be used by an ADF object known as an Activity.

**Activities** typically contain the transformation logic or the analysis commands of the Azure Data Factory’s work. Activities includes the Copy Activity that can be used to ingest data from a variety of data sources. It can also include the Mapping Data Flow to perform code-free data transformations. It can also include the execution of a stored procedure, Hive Query, or Pig script to transform the data. You can push data into a Machine Learning model to perform analysis. It is not uncommon for multiple activities to take place that may include transforming data using a SQL stored procedure and then perform analytics with Databricks. In this case, multiple activities can be logically grouped together with an object referred to as a **Pipeline**, and these can be *scheduled* to execute, or a *trigger* can be defined that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.



*Control flow* is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, and For-each iterators.

*Parameters* are key-value pairs of read-only configuration.  Parameters are defined in the pipeline. The arguments for the defined parameters are passed during execution from the run context that was created by a trigger or a pipeline that was executed manually. Activities within the pipeline consume the parameter values.

Azure Data Factory has an *integration runtime* that enables it to bridge between the activity and linked Services objects. It is referenced by the linked service, and provides the compute environment where the activity either runs on or gets dispatched from. This way, the activity can be performed in the region closest possible. There are three types of Integration Runtime, including Azure, Self-hosted, and Azure-SSIS.

Once all the work is complete, you can then use Data Factory to publish the final dataset to another linked service that can then be consumed by technologies such as Power BI or Machine Learning.

# Azure Data Factory security

To create Data Factory instances, the user account that you use to sign in to Azure must be a member of the contributor or owner role, or an administrator of the Azure subscription.

To create and manage Data Factory objects including datasets, linked services, pipelines, triggers, and integration runtimes, the following requirements must be met:

* To create and manage child resources in the Azure portal, you must belong to the Data Factory Contributor role at the resource group level or above.
* To create and manage resources with PowerShell or the SDK, the contributor role at the resource level or above is sufficient.

## Data Factory Contributor role

When you are added as a member of this role, you have the following permissions:

* Create, edit, and delete data factories and child resources including datasets, linked services, pipelines, triggers, and integration runtimes.
* Deploy Resource Manager templates. Resource Manager deployment is the deployment method used by Data Factory in the Azure portal.
* Manage App Insights alerts for a data factory.
* At the resource group level or above, lets users deploy Resource Manager template.
* Create support tickets.

If the Data Factory Contributor role does not meet your requirement, you can create your own custom role.