**Transfer and transform data with Azure Synapse Analytics pipelines**

With the wide range of data stores available in Azure, there's the need to manage and orchestrate the movement data between them. In fact, you'll usually want to automate extract, transform, and load (ETL) workloads as a regular process in a wider enterprise analytical solution. Pipelines are a mechanism for defining and orchestrating data movement activities. In this module, you'll be introduced to Azure Synapse Analytics pipelines, their component parts, and how to implement and run a pipeline in Azure Synapse Studio.

**Note**

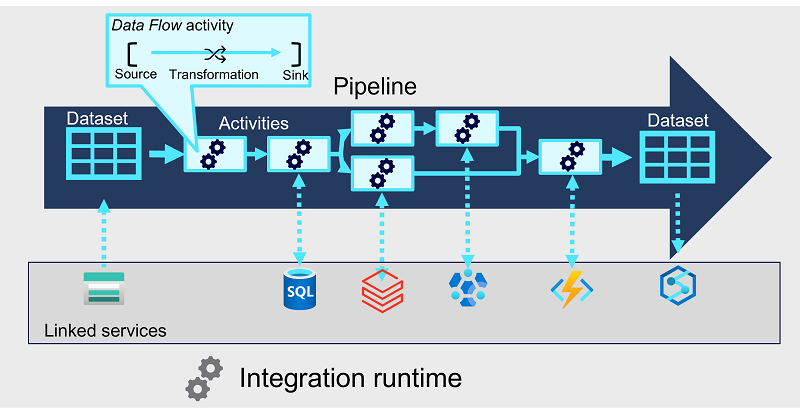
Azure Synapse Analytics pipelines are built on the same technology as Azure Data Factory, and offer a similar authoring experience. The authoring processes described in this module are also applicable to Azure Data Factory. For a detailed discussion of the differences between Azure Synapse Analytics pipelines and Azure Data Factory, see [**Data integration in Azure Synapse Analytics versus Azure Data Factory**](https://learn.microsoft.com/en-us/azure/synapse-analytics/data-integration/concepts-data-factory-differences).

# Understand pipelines in Azure Synapse Analytics

Pipelines in Azure Synapse Analytics encapsulate a sequence of activities that perform data movement and processing tasks. You can use a pipeline to define data transfer and transformation activities, and orchestrate these activities through control flow activities that manage branching, looping, and other typical processing logic. The graphical design tools in Azure Synapse Studio enable you to build complex pipelines with minimal or no coding required.

## Core pipeline concepts

Before building pipelines in Azure Synapse Analytics, you should understand a few core concepts.



### Activities

Activities are the executable tasks in a pipeline. You can define a flow of activities by connecting them in a sequence. The outcome of a particular activity (success, failure, or completion) can be used to direct the flow to the next activity in the sequence.

Activities can encapsulate data transfer operations, including simple data copy operations that extract data from a source and load it to a target (or sink), as well as more complex data flows that apply transformations to the data as part of an extract, transfer, and load (ETL) operation. Additionally, there are activities that encapsulate processing tasks on specific systems, such as running a Spark notebook or calling an Azure function. Finally, there are control flow activities that you can use to implement loops, conditional branching, or manage variable and parameter values.

### Integration runtime

The pipeline requires compute resources and an execution context in which to run. The pipeline's integration runtime provides this context, and is used to initiate and coordinate the activities in the pipeline.

### Linked services

While many of the activities are run directly in the integration runtime for the pipeline, some activities depend on external services. For example, a pipeline might include an activity to run a notebook in Azure Databricks or to call a stored procedure in Azure SQL Database. To enable secure connections to the external services used by your pipelines, you must define linked services for them.

**Note**

Linked services are defined at the Azure Synapse Analytics workspace level, and can be shared across multiple pipelines.

### Datasets

Most pipelines process data, and the specific data that is consumed and produced by activities in a pipeline is defined using datasets. A dataset defines the schema for each data object that will be used in the pipeline, and has an associated linked service to connect to its source. Activities can have datasets as inputs or outputs.

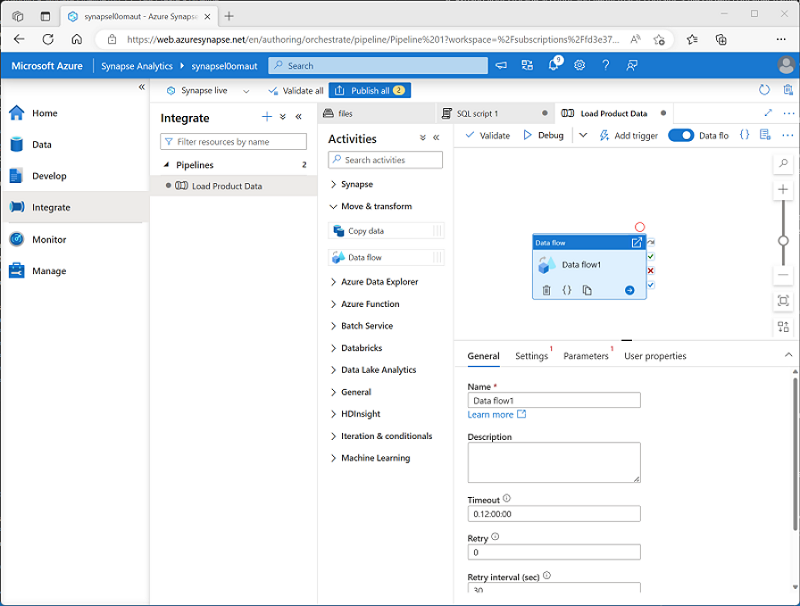
**Note**

Similarly to linked services, datasets are defined at the Azure Synapse Analytics workspace level, and can be shared across multiple pipelines.

# Create a pipeline in Azure Synapse Studio

You can create a pipeline in Azure Synapse Studio by using shortcuts on the **Home** page, but the primary place where pipelines are created and managed is the **Integrate** page.

When you create a pipeline in Azure Synapse Studio, you can use the graphical design interface.

.

The pipeline designer includes a set of activities, organized into categories, which you can drag onto a visual design canvas. You can select each activity on the canvas and use the properties pane beneath the canvas to configure the settings for that activity.

To define the logical sequence of activities, you can connect them by using the **Succeeded**, **Failed**, and **Completed** dependency conditions, which are shown as small icons on the right-hand edge of each activity.

## Defining a pipeline with JSON

While the graphical development environment is the preferred way to create a pipeline, you can also create or edit the underlying JSON definition of a pipeline. The following code example shows the JSON definition of a pipeline that includes a **Copy Data** activity:

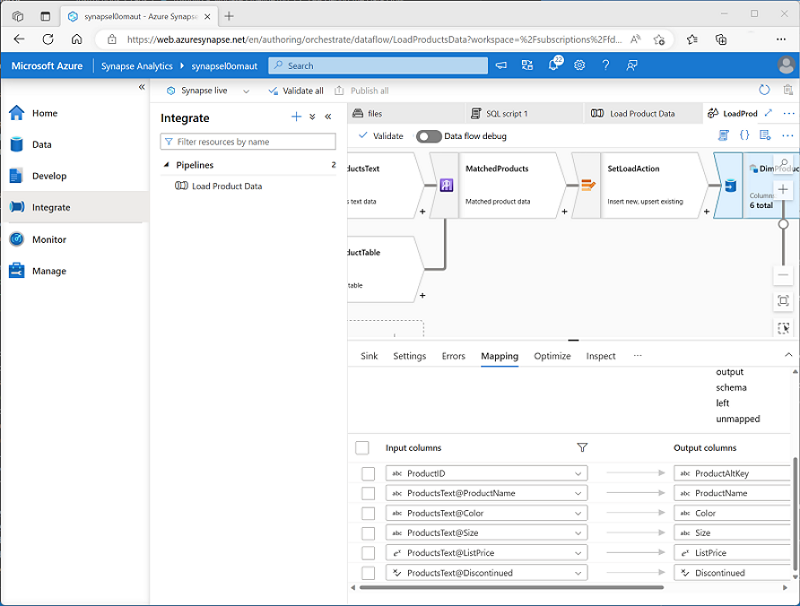


# Define data flows

A **Data Flow** is a commonly used activity type to define data flow and transformation. Data flows consist of:

* **Sources** - The input data to be transferred.
* **Transformations** – Various operations that you can apply to data as it streams through the data flow.
* **Sinks** – Targets into which the data will be loaded.

When you add a **Data Flow** activity to a pipeline, you can open it in a separate graphical design interface in which to create and configure the required data flow elements.



An important part of creating a data flow is to define mappings for the columns as the data flows through the various stages, ensuring column names and data types are defined appropriately. While developing a data flow, you can enable the **Data flow debug** option to pass a subset of data through the flow, which can be useful to test that your columns are mapped correctly.

**Tip**

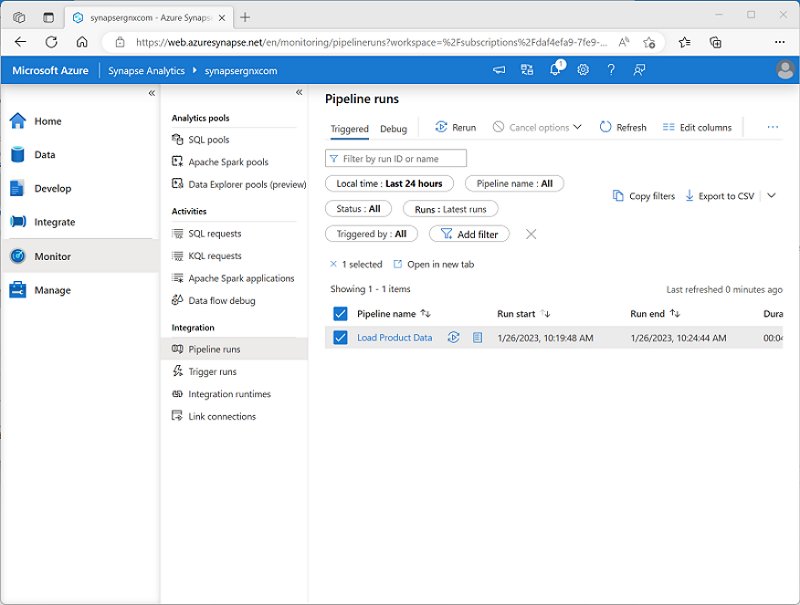
To learn more about implementing a **Data Flow** activity, see [**Data Flow activity in Azure Data Factory and Azure Synapse Analytics**](https://learn.microsoft.com/en-us/azure/data-factory/control-flow-execute-data-flow-activity) in the Azure documentation.

# Run a pipeline

When you’re ready, you can publish a pipeline and use a trigger to run it. Triggers can be defined to run the pipeline:

* Immediately
* At explicitly scheduled intervals
* In response to an event, such as new data files being added to a folder in a data lake.

You can monitor each individual run of a pipeline in the **Monitor** page in Azure Synapse Studio.



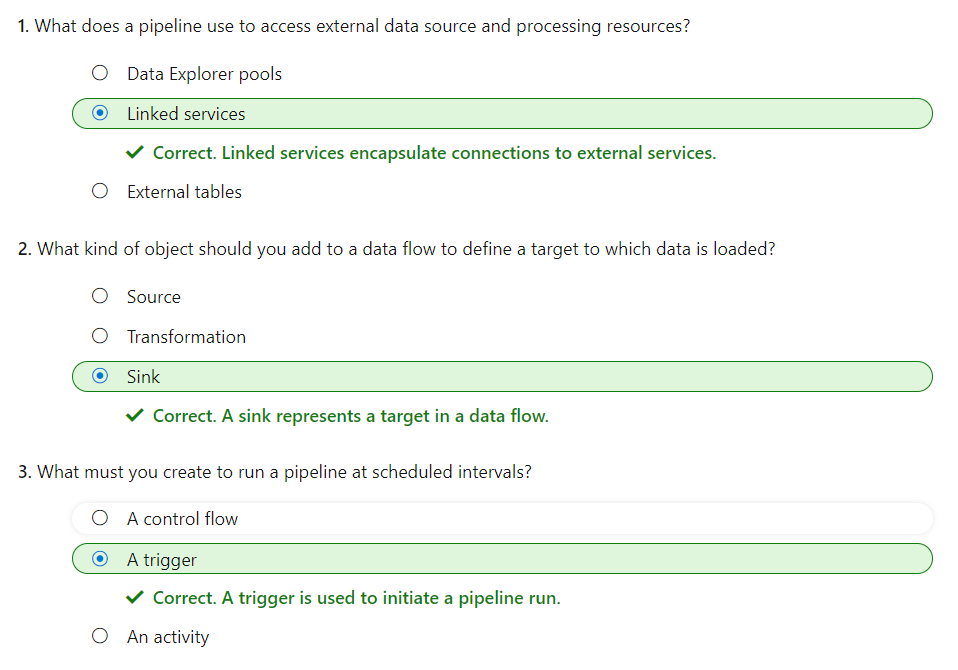
The ability to monitor past and ongoing pipeline runs is useful for troubleshooting purposes. Additionally, when combined with the ability to integrate Azure Synapse Analytics and Microsoft Purview, you can use pipeline run history to track data lineage data flows.

**Tip**

To learn more about integration between Azure Synapse Analytics and Microsoft Purview, consider completing the [**Integrate Microsoft Purview and Azure Synapse Analytics**](https://learn.microsoft.com/en-us/training/modules/integrate-microsoft-purview-azure-synapse-analytics) module.

# Exercise - Build a data pipeline in Azure Synapse Analytics

<https://microsoftlearning.github.io/dp-203-azure-data-engineer/Instructions/Labs/10-Synpase-pipeline.html>



# Use Spark Notebooks in an Azure Synapse Pipeline

With Azure Synapse Analytics pipelines, you can orchestrate data transfer and transformation activities and build data integration solutions across multiple systems. When you're working with analytical data in a data lake, Apache Spark provides a scalable, distributed processing platform that you can use to process huge volumes of data efficiently.

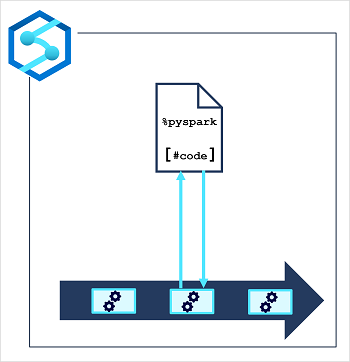
The **Synapse Notebook** activity enables you to run data processing code in Spark notebooks as a task in a pipeline; making it possible to automate big data processing and integrate it into extract, transform, and load (ETL) workloads.

# Understand Synapse Notebooks and Pipelines

Completed100 XP

* 5 minutes

Azure Synapse Pipelines enable you to create, run, and manage data integration and data flow activities. While many of these activities are built-into the Azure Synapse Pipeline platform and run natively in the integration runtime for your pipeline, you can also use external processing resources to perform specific tasks. One such external resource is an Apache Spark pool in your Azure Synapse Analytics workspace on which you can run code in a notebook.



It's common in big data analytics solutions for data engineers to use Spark notebooks for initial data exploration and interactive experimentation when designing data transformation processes. When the transformation logic has been completed, you can perform some final code optimization and refactoring for maintainability, and then include the notebook in a pipeline. The pipeline can then be run on a schedule or in response to an event (such as new data files being loaded into the data lake).

The notebook is run on a Spark pool, which you can configure with the appropriate compute resources and Spark runtime for your specific workload. The pipeline itself is run in an integration runtime that orchestrates the activities in the pipeline, coordinating the external services needed to run them.

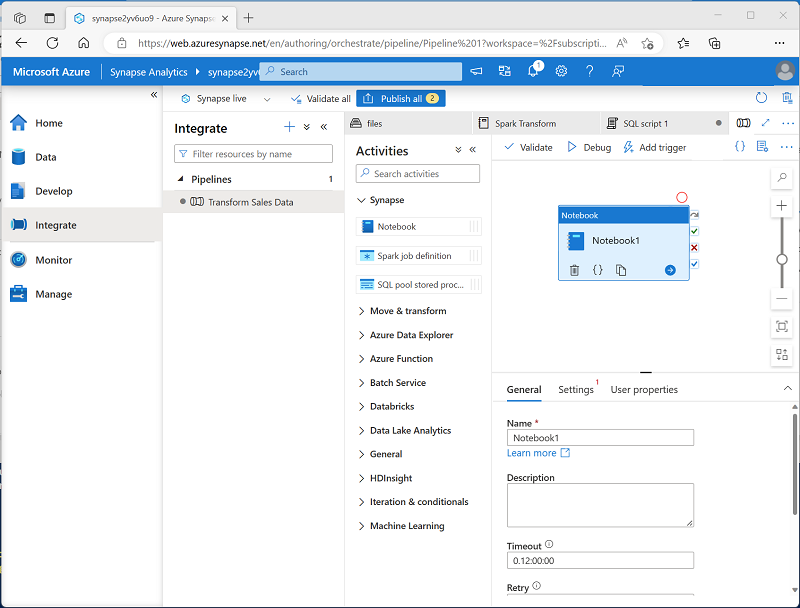
**Tip**

There are several best practices that can help make working with Spark notebooks more efficient and effective. Some of these include:

* Keep your code organized: Use clear and descriptive variable and function names, and organize your code into small, reusable chunks.
* Cache intermediate results: Spark allows you to cache intermediate results, which can significantly speed up the performance of your notebook.
* Avoid unnecessary computations: Be mindful of the computations you are performing and try to avoid unnecessary steps. For example, if you only need a subset of your data, filter it out before running any further computations.
* Avoid using collect() unless necessary: When working with large datasets, it is often better to perform operations on the entire dataset rather than bringing the data into the driver node using the collect() method.
* Use Spark UI for monitoring and debugging: Spark's web-based user interface (UI) provides detailed information about the performance of your Spark jobs, including task execution times, input and output data sizes, and more.
* Keep your dependencies version-consistent and updated: when working with Spark, it is important to keep dependencies version-consistent across your cluster and to use the latest version of Spark and other dependencies if possible.

# Use a Synapse notebook activity in a pipeline

To run a Spark notebook in a pipeline, you must add a notebook activity and configure it appropriately. You'll find the **Notebook** activity in the **Synapse** section of the activities pane in the Azure Synapse Analytics pipeline designer.



**Tip**

You can also add a notebook to a pipeline from within the notebook editor.

To configure the notebook activity, edit the settings in the properties pane beneath the pipeline designer canvas. Notebook activity specific settings include:

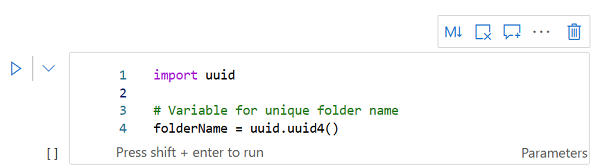
* **Notebook**: The notebook you want to run. You can select an existing notebook in your Azure Synapse Analytics workspace, or create a new one.
* **Spark pool**: The Apache Spark pool on which the notebook should be run.
* **Executor size**: The node size for the worker nodes in the pool, which determines the number of processor cores and the amount of memory allocated to worker nodes.
* **Dynamically allocate executors**: Configures Spark dynamic allocation, enabling the pool to automatically scale up and down to support the workload.
* **Min executors**: The minimum number of executors to be allocated.
* **Max executors**: The maximum number of executors to be allocated.
* **Driver size**: The node size for the driver node.

# Use parameters in a notebook

Parameters enable you to dynamically pass values for variables in the notebook each time it's run. This approach provides flexibility, enabling you to adjust the logic encapsulated in the notebook for each run.

## Create a parameters cell in the notebook

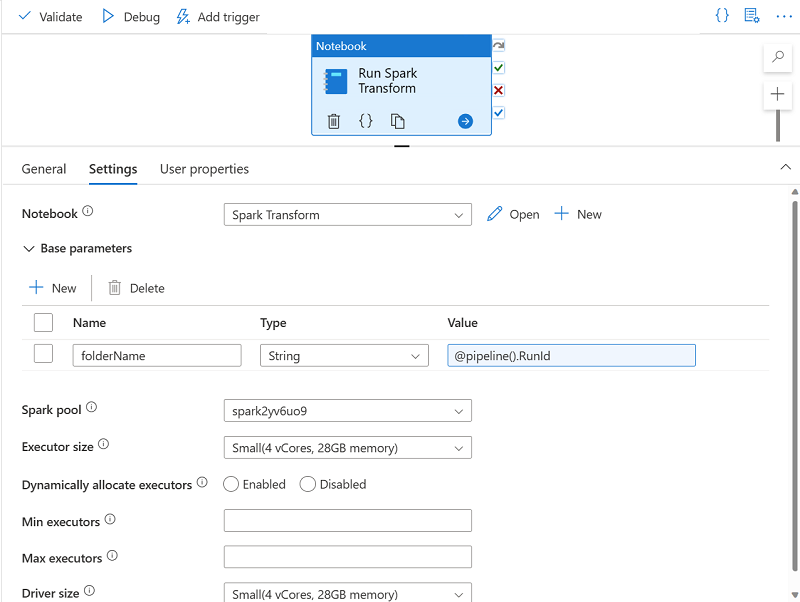
To define the parameters for a notebook, you declare and initialize variables in a cell, which you then configure as a **Parameters** cell by using the toggle option in the notebook editor interface.



Initializing a variable ensures that it has a default value, which will be used if the parameter isn't set in the notebook activity.

## Set base parameters for the notebook activity

After defining a parameters cell in the notebook, you can set values to be used when the notebook is run by a notebook activity in a pipeline. To set parameter values, expand and edit the **Base parameters** section of the settings for the activity.



You can assign explicit parameter values, or use an expression to assign a dynamic value. For example, the expression @pipeline().RunId returns the unique identifier for the current run of the pipeline.

# Exercise - Use an Apache Spark notebook in a pipeline

<https://microsoftlearning.github.io/dp-203-azure-data-engineer/Instructions/Labs/11-Spark-nobook-in-Synapse-Pipeline.html>

