**What is Azure Data Factory?**

Data Factory is a fully managed, cloud-based, data-integration ETL service that automates the movement and transformation of data. Like a factory that runs equipment to transform raw materials into finished goods, Azure Data Factory orchestrates existing services that collect raw data and transform it into ready-to-use information.

By using Azure Data Factory, you can create data-driven workflows to move data between on-premises and cloud data stores. And you can process and transform data with Data Flows. ADF also supports external compute engines for hand-coded transformations by using compute services such as Azure HDInsight, Azure Databricks, and the SQL Server Integration Services (SSIS) integration runtime.

With Data Factory, you can execute your data processing either on an Azure-based cloud service or in your own self-hosted compute environment, such as SSIS, SQL Server, or Oracle. After you create a pipeline that performs the action you need, you can schedule it to run periodically (hourly, daily, or weekly, for example), time window scheduling, or trigger the pipeline from an event occurrence. For more information, see [Introduction to Azure Data Factory](https://docs.microsoft.com/en-us/azure/data-factory/introduction).

**Control flows and scale**

To support the diverse integration flows and patterns in the modern data warehouse, Data Factory enables flexible data pipeline modeling. This entails full control flow programming paradigms, which include conditional execution, branching in data pipelines, and the ability to explicitly pass parameters within and across these flows. Control flow also encompasses transforming data through activity dispatch to external execution engines and data flow capabilities, including data movement at scale, via the Copy activity.

Data Factory provides freedom to model any flow style that's required for data integration and that can be dispatched on demand or repeatedly on a schedule. A few common flows that this model enables are:

* Control flows:
  + Activities can be chained together in a sequence within a pipeline.
  + Activities can be branched within a pipeline.
  + Parameters:
    - Parameters can be defined at the pipeline level and arguments can be passed while you invoke the pipeline on demand or from a trigger.
    - Activities can consume the arguments that are passed to the pipeline.
  + Custom state passing:
    - Activity outputs, including state, can be consumed by a subsequent activity in the pipeline.
  + Looping containers:
    - The foreach activity will iterate over a specified collection of activities in a loop.
* Trigger-based flows:
  + Pipelines can be triggered on demand or by wall-clock time.
* Delta flows:
  + Parameters can be used to define your high-water mark for delta copy while moving dimension or reference tables from a relational store, either on-premises or in the cloud, to load the data into the lake.

For more information, see [Tutorial: Control flows](https://docs.microsoft.com/en-us/azure/data-factory/tutorial-control-flow).

**Data transformed at scale with code-free pipelines**

The new browser-based tooling experience provides code-free pipeline authoring and deployment with a modern, interactive web-based experience.

For visual data developers and data engineers, the Data Factory web UI is the code-free design environment that you will use to build pipelines. It's fully integrated with Visual Studio Online Git and provides integration for CI/CD and iterative development with debugging options.

**Rich cross-platform SDKs for advanced users**

Data Factory V2 provides a rich set of SDKs that can be used to author, manage, and monitor pipelines by using your favorite IDE, including:

* Python SDK
* PowerShell CLI
* C# SDK

Users can also use the documented REST APIs to interface with Data Factory V2.

**Iterative development and debugging by using visual tools**

Azure Data Factory visual tools enable iterative development and debugging. You can create your pipelines and do test runs by using the **Debug** capability in the pipeline canvas without writing a single line of code. You can view the results of your test runs in the **Output** window of your pipeline canvas. After your test run succeeds, you can add more activities to your pipeline and continue debugging in an iterative manner. You can also cancel your test runs after they are in progress.

You are not required to publish your changes to the data factory service before selecting **Debug**. This is helpful in scenarios where you want to make sure that the new additions or changes will work as expected before you update your data factory workflows in development, test, or production environments.

**Ability to deploy SSIS packages to Azure**

If you want to move your SSIS workloads, you can create a Data Factory and provision an Azure-SSIS integration runtime. An Azure-SSIS integration runtime is a fully managed cluster of Azure VMs (nodes) that are dedicated to run your SSIS packages in the cloud. For step-by-step instructions, see the [Deploy SSIS packages to Azure](https://docs.microsoft.com/en-us/azure/data-factory/tutorial-create-azure-ssis-runtime-portal) tutorial.

**SDKs**

If you are an advanced user and looking for a programmatic interface, Data Factory provides a rich set of SDKs that you can use to author, manage, or monitor pipelines by using your favorite IDE. Language support includes .NET, PowerShell, Python, and REST.

**Monitoring**

You can monitor your Data Factories via PowerShell, SDK, or the Visual Monitoring Tools in the browser user interface. You can monitor and manage on-demand, trigger-based, and clock-driven custom flows in an efficient and effective manner. Cancel existing tasks, see failures at a glance, drill down to get detailed error messages, and debug the issues, all from a single pane of glass without context switching or navigating back and forth between screens.

**New features for SSIS in Data Factory**

Since the initial public preview release in 2017, Data Factory has added the following features for SSIS:

* Support for three more configurations/variants of Azure SQL Database to host the SSIS database (SSISDB) of projects/packages:
* SQL Database with virtual network service endpoints
* SQL Managed Instance
* Elastic pool
* Support for an Azure Resource Manager virtual network on top of a classic virtual network to be deprecated in the future, which lets you inject/join your Azure-SSIS integration runtime to a virtual network configured for SQL Database with virtual network service endpoints/MI/on-premises data access. For more information, see also [Join an Azure-SSIS integration runtime to a virtual network](https://docs.microsoft.com/en-us/azure/data-factory/join-azure-ssis-integration-runtime-virtual-network).
* Support for Azure Active Directory (Azure AD) authentication and SQL authentication to connect to the SSISDB, allowing Azure AD authentication with your Data Factory managed identity for Azure resources
* Support for bringing your existing SQL Server license to earn substantial cost savings from the Azure Hybrid Benefit option
* Support for Enterprise Edition of the Azure-SSIS integration runtime that lets you use advanced/premium features, a custom setup interface to install additional components/extensions, and a partner ecosystem. For more information, see also [Enterprise Edition, Custom Setup, and 3rd Party Extensibility for SSIS in ADF](https://blogs.msdn.microsoft.com/ssis/2018/04/27/enterprise-edition-custom-setup-and-3rd-party-extensibility-for-ssis-in-adf/).
* Deeper integration of SSIS in Data Factory that lets you invoke/trigger first-class Execute SSIS Package activities in Data Factory pipelines and schedule them via SSMS. For more information, see also [Modernize and extend your ETL/ELT workflows with SSIS activities in ADF pipelines](https://blogs.msdn.microsoft.com/ssis/2018/05/23/modernize-and-extend-your-etlelt-workflows-with-ssis-activities-in-adf-pipelines/).

**What is the integration runtime?**

The integration runtime is the compute infrastructure that Azure Data Factory uses to provide the following data integration capabilities across various network environments:

* **Data movement**: For data movement, the integration runtime moves the data between the source and destination data stores, while providing support for built-in connectors, format conversion, column mapping, and performant and scalable data transfer.
* **Dispatch activities**: For transformation, the integration runtime provides capability to natively execute SSIS packages.
* **Execute SSIS packages**: The integration runtime natively executes SSIS packages in a managed Azure compute environment. The integration runtime also supports dispatching and monitoring transformation activities running on a variety of compute services, such as Azure HDInsight, Azure Machine Learning, SQL Database, and SQL Server.

You can deploy one or many instances of the integration runtime as required to move and transform data. The integration runtime can run on an Azure public network or on a private network (on-premises, Azure Virtual Network, or Amazon Web Services virtual private cloud [VPC]).

For more information, see [Integration runtime in Azure Data Factory](https://docs.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime).

**What is the limit on the number of integration runtimes?**

There is no hard limit on the number of integration runtime instances you can have in a data factory. There is, however, a limit on the number of VM cores that the integration runtime can use per subscription for SSIS package execution. For more information, see [Data Factory limits](https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/azure-subscription-service-limits#data-factory-limits).

**What are the top-level concepts of Azure Data Factory?**

An Azure subscription can have one or more Azure Data Factory instances (or data factories). Azure Data Factory contains four key components that work together as a platform on which you can compose data-driven workflows with steps to move and transform data.

**Pipelines**

A data factory can have one or more pipelines. A pipeline is a logical grouping of activities to perform a unit of work. Together, the activities in a pipeline perform a task. For example, a pipeline can contain a group of activities that ingest data from an Azure blob and then run a Hive query on an HDInsight cluster to partition the data. The benefit is that you can use a pipeline to manage the activities as a set instead of having to manage each activity individually. You can chain together the activities in a pipeline to operate them sequentially, or you can operate them independently, in parallel.

**Data flows**

Data flows are objects that you build visually in Data Factory which transform data at scale on backend Spark services. You do not need to understand programming or Spark internals. Just design your data transformation intent using graphs (Mapping) or spreadsheets (Wrangling).

**Activities**

Activities represent a processing step in a pipeline. For example, you can use a Copy activity to copy data from one data store to another data store. Similarly, you can use a Hive activity, which runs a Hive query on an Azure HDInsight cluster to transform or analyze your data. Data Factory supports three types of activities: data movement activities, data transformation activities, and control activities.

**Datasets**

Datasets represent data structures within the data stores, which simply point to or reference the data you want to use in your activities as inputs or outputs.

**Linked services**

Linked services are much like connection strings, which define the connection information needed for Data Factory to connect to external resources. Think of it this way: A linked service defines the connection to the data source, and a dataset represents the structure of the data. For example, an Azure Storage linked service specifies the connection string to connect to the Azure Storage account. And an Azure blob dataset specifies the blob container and the folder that contains the data.

Linked services have two purposes in Data Factory:

* To represent a *data store* that includes, but is not limited to, a SQL Server instance, an Oracle database instance, a file share, or an Azure Blob storage account. For a list of supported data stores, see [Copy Activity in Azure Data Factory](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview).
* To represent a *compute resource* that can host the execution of an activity. For example, the HDInsight Hive activity runs on an HDInsight Hadoop cluster. For a list of transformation activities and supported compute environments, see [Transform data in Azure Data Factory](https://docs.microsoft.com/en-us/azure/data-factory/transform-data).

**Triggers**

Triggers represent units of processing that determine when a pipeline execution is kicked off. There are different types of triggers for different types of events.

**Pipeline runs**

A pipeline run is an instance of a pipeline execution. You usually instantiate a pipeline run by passing arguments to the parameters that are defined in the pipeline. You can pass the arguments manually or within the trigger definition.

**Parameters**

Parameters are key-value pairs in a read-only configuration. You define parameters in a pipeline, and you pass the arguments for the defined parameters during execution from a run context. The run context is created by a trigger or from a pipeline that you execute manually. Activities within the pipeline consume the parameter values.

A dataset is a strongly typed parameter and an entity that you can reuse or reference. An activity can reference datasets, and it can consume the properties that are defined in the dataset definition.

A linked service is also a strongly typed parameter that contains connection information to either a data store or a compute environment. It's also an entity that you can reuse or reference.

**Control flows**

Control flows orchestrate pipeline activities that include chaining activities in a sequence, branching, parameters that you define at the pipeline level, and arguments that you pass as you invoke the pipeline on demand or from a trigger. Control flows also include custom state passing and looping containers (that is, foreach iterators).

For more information about Data Factory concepts, see the following articles:

* [Dataset and linked services](https://docs.microsoft.com/en-us/azure/data-factory/concepts-datasets-linked-services)
* [Pipelines and activities](https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipelines-activities)
* [Integration runtime](https://docs.microsoft.com/en-us/azure/data-factory/concepts-integration-runtime)

**What is the pricing model for Data Factory?**

For Azure Data Factory pricing details, see [Data Factory pricing details](https://azure.microsoft.com/pricing/details/data-factory/).

**How can I stay up-to-date with information about Data Factory?**

For the most up-to-date information about Azure Data Factory, go to the following sites:

* [Blog](https://azure.microsoft.com/blog/tag/azure-data-factory/)
* [Documentation home page](https://docs.microsoft.com/en-us/azure/data-factory)
* [Product home page](https://azure.microsoft.com/services/data-factory/)

**Technical deep dive**

**How can I schedule a pipeline?**

You can use the scheduler trigger or time window trigger to schedule a pipeline. The trigger uses a wall-clock calendar schedule, which can schedule pipelines periodically or in calendar-based recurrent patterns (for example, on Mondays at 6:00 PM and Thursdays at 9:00 PM). For more information, see [Pipeline execution and triggers](https://docs.microsoft.com/en-us/azure/data-factory/concepts-pipeline-execution-triggers).

**Can I pass parameters to a pipeline run?**

Yes, parameters are a first-class, top-level concept in Data Factory. You can define parameters at the pipeline level and pass arguments as you execute the pipeline run on demand or by using a trigger.

**Can I define default values for the pipeline parameters?**

Yes. You can define default values for the parameters in the pipelines.

**Can an activity in a pipeline consume arguments that are passed to a pipeline run?**

Yes. Each activity within the pipeline can consume the parameter value that's passed to the pipeline and run with the @parameter construct.

**Can an activity output property be consumed in another activity?**

Yes. An activity output can be consumed in a subsequent activity with the @activity construct.

**How do I gracefully handle null values in an activity output?**

You can use the @coalesce construct in the expressions to handle null values gracefully.

**Mapping data flows**

**I need help troubleshooting my data flow logic. What info do I need to provide to get help?**

When Microsoft provides help or troubleshooting with data flows, please provide the Data Flow Script. This is the code-behind script from your data flow graph. From the ADF UI, open your data flow, then click the "Script" button at the top-right corner. Copy and paste this script or save it in a text file.

**How do I access data by using the other 90 dataset types in Data Factory?**

The mapping data flow feature currently allows Azure SQL Database, Azure SQL Data Warehouse, delimited text files from Azure Blob storage or Azure Data Lake Storage Gen2, and Parquet files from Blob storage or Data Lake Storage Gen2 natively for source and sink.

Use the Copy activity to stage data from any of the other connectors, and then execute a Data Flow activity to transform data after it's been staged. For example, your pipeline will first copy into Blob storage, and then a Data Flow activity will use a dataset in source to transform that data.

**Is the self-hosted integration runtime available for data flows?**

Self-hosted IR is an ADF pipeline construct that you can use with the Copy Activity to acquire or move data to and from on-prem or VM-based data sources and sinks. Stage the data first with a Copy, then Data Flow for transformation, and then a subsequent copy if you need to move that transformed data back to the on-prem store.

**Does the data flow compute engine serve multiple tenants?**

Clusters are never shared. We guarantee isolation for each job run in production runs. In case of debug scenario one person gets one cluster, and all debugs will go to that cluster which are initiated by that user.

**Wrangling data flows**

**What are the supported regions for wrangling data flow?**

Wrangling data flow is currently supported in data factories created in following regions:

* Australia East
* Canada Central
* Central India
* East US
* East US 2
* Japan East
* North Europe
* Southeast Asia
* South Central US
* UK South
* West Central US
* West Europe
* West US
* West US 2

**What are the limitations and constraints with wrangling data flow?**

Dataset names can only contain alpha-numeric characters. The following data stores are supported:

* DelimitedText dataset in Azure Blob Storage using account key authentication
* DelimitedText dataset in Azure Data Lake Storage gen2 using account key or service principal authentication
* DelimitedText dataset in Azure Data Lake Storage gen1 using service principal authentication
* Azure SQL Database and Data Warehouse using sql authentication. See supported SQL types below. There is no PolyBase or staging support for data warehouse.

At this time, linked service Key Vault integration is not supported in wrangling data flows.

**What is the difference between mapping and wrangling data flows?**

Mapping data flows provide a way to transform data at scale without any coding required. You can design a data transformation job in the data flow canvas by constructing a series of transformations. Start with any number of source transformations followed by data transformation steps. Complete your data flow with a sink to land your results in a destination. Mapping data flow is great at mapping and transforming data with both known and unknown schemas in the sinks and sources.

Wrangling data flows allow you to do agile data preparation and exploration using the Power Query Online mashup editor at scale via spark execution. With the rise of data lakes sometimes you just need to explore a data set or create a dataset in the lake. You aren't mapping to a known target. Wrangling data flows are used for less formal and model-based analytics scenarios.

**What is the difference between Power Platform Dataflows and wrangling data flows?**

Power Platform Dataflows allow users to import and transform data from a wide range of data sources into the Common Data Service and Azure Data Lake to build PowerApps applications, Power BI reports or Flow automations. Power Platform Dataflows use the established Power Query data preparation experiences, similar to Power BI and Excel. Power Platform Dataflows also enable easy reuse within an organization and automatically handle orchestration (e.g. automatically refreshing dataflows that depend on another dataflow when the former one is refreshed).

Azure Data Factory (ADF) is a managed data integration service that allows data engineers and citizen data integrator to create complex hybrid extract-transform-load (ETL) and extract-load-transform (ELT) workflows. Wrangling data flow in ADF empowers users with a code-free, serverless environment that simplifies data preparation in the cloud and scales to any data size with no infrastructure management required. It uses the Power Query data preparation technology (also used in Power Platform dataflows, Excel, Power BI) to prepare and shape the data. Built to handle all the complexities and scale challenges of big data integration, wrangling data flows allow users to quickly prepare data at scale via spark execution. Users can build resilient data pipelines in an accessible visual environment with our browser-based interface and let ADF handle the complexities of Spark execution. Build schedules for your pipelines and monitor your data flow executions from the ADF monitoring portal. Easily manage data availability SLAs with ADF's rich availability monitoring and alerts and leverage built-in continuous integration and deployment capabilities to save and manage your flows in a managed environment. Establish alerts and view execution plans to validate that your logic is performing as planned as you tune your data flows.

**Supported SQL Types**

Wrangling data flow supports the following data types in SQL. You will get a validation error for using a data type that isn't supported.

* short
* double
* real
* float
* char
* nchar
* varchar
* nvarchar
* integer
* int
* bit
* boolean
* smallint
* tinyint
* bigint
* long
* text
* date
* datetime
* datetime2
* smalldatetime
* timestamp
* uniqueidentifier
* xml

Other data types will be supported in the future.