**Azure Data Factory Basic Concepts**

There are good resources for step-by-step learning of ADF, but few clear explanations of its basic underlying concepts. This article fills that gap.



Photo by [Carlos Muza](https://unsplash.com/@kmuza?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/data-factory?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText" \t "_blank)

Azure Data Factory (ADF) is a data pipeline orchestrator and ETL tool that is part of the Microsoft Azure cloud ecosystem. ADF can pull data from the outside world (FTP, Amazon S3, Oracle, and [many more](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview)), transform it, filter it, enhance it, and move it along to another destination. In my work for a health-data project we are using ADF to drive our data flow from raw ingestion to polished analysis that is ready to display.

There are many good resources for learning ADF, including an [introduction](https://docs.microsoft.com/en-us/azure/data-factory/introduction) and a [quickstart](https://docs.microsoft.com/en-us/azure/data-factory/quickstart-create-data-factory-portal" \t "_blank). When I was starting out with ADF however, I did not find a clear explanation for the basic underlying concepts it is built upon. This article is an attempt to fill that gap.

Getting ADF to do real work for you involves the following layers of technology, listed from the highest level of abstraction that you interact with down to the software closest to the data.

* Pipeline, the graphical user interface where you place widgets and draw data paths
* Activity, a graphical widget that does something to your data
* Source and Sink, the parts of an activity that specify where data is coming from and going to
* Data Set, an explicitly defined set of data that ADF can operate on
* Linked Service, the connection information that allows ADF to access a specific outside data resource
* Integration Runtime, a glue/gateway layer that lets ADF talk to software outside of itself

Understanding the purpose of each layer and how it contributes to an overall ADF solution is key to using the tool well. I find it easiest to understand ADF by considering the layers in reverse order, starting at the bottom near the data.

**Integration Runtime**

An integration runtime provides the gateway between ADF and the actual data or compute resources you need. If you are using ADF to marshal native Azure resources, such as an Azure Data Lake or Databricks, then ADF knows how to talk to those resources. Just use the built-in integration runtime and don’t think about it — no set up or configuration required.

But suppose you want ADF to operate on data that is stored on an Oracle Database server under your desk, or computers and data within your company’s private network. In these cases you must set up the gateway with a self-hosted integration runtime.

This screenshot shows the built-in integration runtime. It comes out of the box and and is always there when you access native Azure resources.

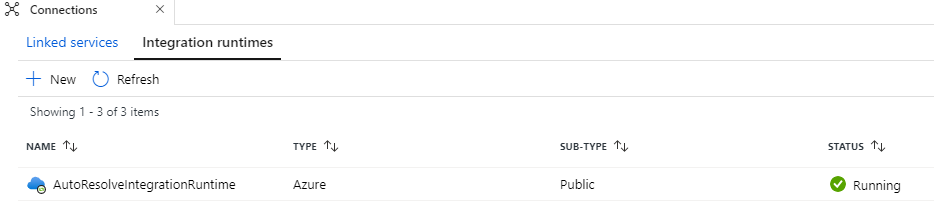


Image by Author

**Linked Service**

A linked service tells ADF how to see the particular data or computers you want to operate on. To access a specific Azure storage account, you create a linked service for it and include access credentials. To read/write another storage account, you create another linked service. To allow ADF to operate on an Azure SQL database, your linked service will state the Azure subscription, server name, database name, and credentials.

This screenshot shows a standard linked service for the most common Azure storage (Azure Data Lake Gen 2 container/blob).

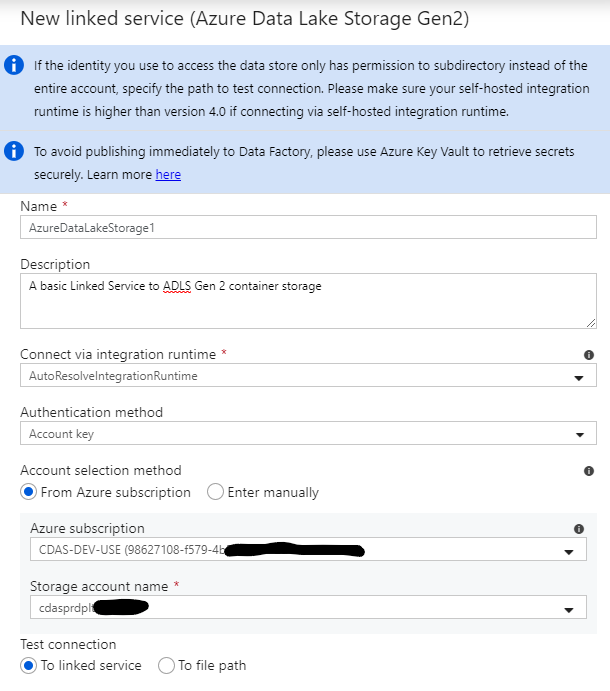


Image by Author

**Data Set**

A data set makes a linked service more specific; it describes the folder you are using within a storage container, or the table within a database, etc.

The data set in this screenshot points to one directory in one container in one Azure storage account. (The container and directory names are set in the Parameters tab.) Note how the data set references a linked service. Note also that this data set specifies that the data is zipped, which allows ADF to automatically unzip the data as you read it.

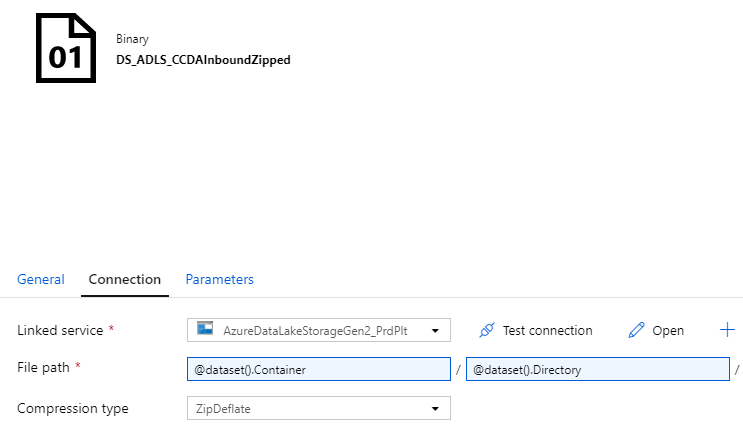


Image by Author

**Source and Sink**

A source and a sink are, as their names imply, places data comes from and goes to. Sources and sinks are built on data sets. ADF is mostly concerned with moving data from one place to another, often with some kind of transformation along the way, so it needs to know where to move the data.

It is important to understand that there is mushy distinction between data sets and sources/sinks. A data set defines a particular collection of data, but a source or sink can redefine the collection. For example, suppose DataSet1 is defined as the folder /Vehicles/GM/Trucks/. When a source uses DataSet1, it can take that collection as-is (the default), or narrow the set to /Vehicles/GM/Trucks/Silverado/ or expand it to /Vehicles/.

There is artful design involved in the trade-offs between data set scope and source/sink scope. My practice is to define data sets somewhat broadly (thereby reducing the number of them), and then allow sources and sinks to narrow down what each needs in a particular situation.

This source uses the zipped data set shown above, narrows it, and makes sure to select only files actually named \*.zip (otherwise the unzipping will fail).

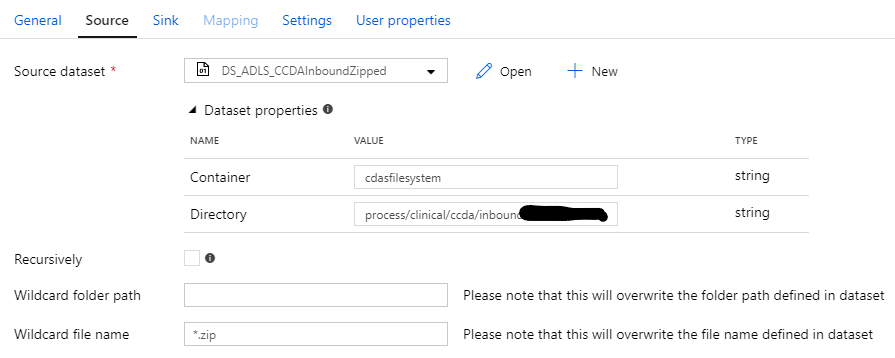


Image by Author

**Activity**

Activities are the GUI widgets within Data Factory that do specific kinds of data movement or transformation. There is a CopyData activity to move data, a ForEach activity to loop over a file list, a Filter activity that chooses a subset of files, etc. Most activities have a source and a sink.

**Pipeline**

An ADF pipeline is the top-level concept that you work with most directly. Pipelines are composed of activities and data flow arrows. You program ADF by creating pipelines. You get work done by running pipelines, either manually or via automatic triggers. You look at the results of your work by monitoring pipeline execution.

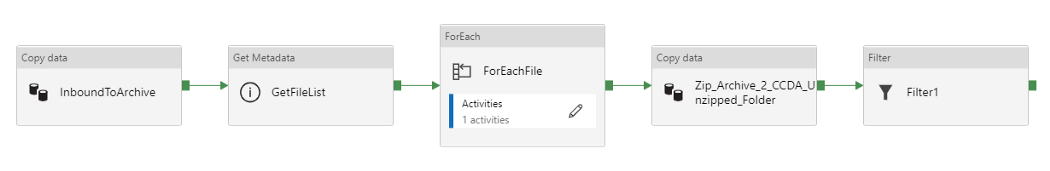


Image by Author

This pipeline takes inbound data from an initial Data Lake folder, moves it to cold archive storage, gets a list of the files, loops over each file, copies those files to an unzipped working folder, then applies an additional filter by file type.

**Conclusion**

This article described the basic concepts and operations of Azure Data Factory. ADF also has the ability to do more complex transformations, by running PySpark code in Databricks, invoking a custom virtual machine, making webhook callouts, etc. After the movements and transformations are done, the data can be delivered to a wide range of destinations, including an SQL database, a plain file system, and outbound FTP.

(https://towardsdatascience.com/azure-data-factory-basic-concepts-e10448e54023#:~:text=Azure%20Data%20Factory%20(ADF)%20is,it%20along%20to%20another%20destination.)