Azure Data Factory: Data Integration in the Cloud

by Sahil Gupta, Engineering Consultant

Contents

Data Analytics Today

Scenarios

Pricing

Using ADF Pipelines

A Closer Look at Pipelines

Conclusion

GlobalLogic®
A Hitachi Group Company

Data Analytics Today

Effective data analytics provides enormous business value for many organizations.

Effective data analytics provides enormous business value for many organizations. As ever-greater amounts of diverse data become available, analytics can provide even more value. But to benefit from this change, your organization must embrace the new approaches to data analytics that cloud computing makes possible.

Microsoft Azure provides a broad set of cloud technologies for data analysis, designed to help you derive more value from your data. These services include the following:

- Azure SQL Data Warehouse, providing scalable relational data warehousing in the cloud.
- Azure Blob Storage, commonly called just Blobs, provides low-cost cloud storage of binary data.
- Azure Data Lake Store, implementing the Hadoop Distributed File System (HDFS) as a cloud service.
- Azure Data Lake Analytics offers U-SQL, a tool for distributed data analysis in Azure Data Lake Store.
- Azure Analysis Services, a cloud offering based on SQL Server Analysis Services.
- Azure HDInsight, with support for Hadoop technologies, such as Hive and Pig, along with Spark.
- Azure Databricks, a Spark-based analytics platform.
- Azure Machine Learning is a set of data science tools for finding patterns in existing data, then generating models that can recognize those patterns in new data.

You can combine these services as needed to analyze both relational and unstructured data.

But there's one essential aspect of data analytics that none address: **data integration**.

This might require extracting data from where it originates (such as in one or more operational databases), then loading it into where it needs to be for analysis (such as in a data warehouse).

You might also need to transform the data in some ways during this process. And while all of these tasks can be done manually, it usually makes more sense to automate them.

Azure Data Factory (ADF) is designed to help you address challenges like these. This cloud-based data integration service is aimed at two distinct worlds: big data and traditional data warehousing.

The big data community, which relies on technologies for handling large amounts of diverse data.

For this audience, ADF offers a way to create and run ADF pipelines in the cloud. A pipeline can access both on-premises and cloud data services. It typically works with technologies such as Azure SQL Data Warehouse, Azure Blobs, Azure Data Lake, Azure HD Insight, Azure Databricks, and Azure Machine Learning.

The traditional relational data warehousing community, which relies on technologies such as SQL Server. These practitioners use SQL Server Integration Services (SSIS) to create SSIS packages. A package is analogous to an ADF pipeline; each defines a process to extract, load, transform, or otherwise work with data.

ADF allows this audience to run SSIS packages on Azure and access both on-premises and cloud data services.

The critical point is this: ADF is a single cloud service for data integration across all of your data sources, whether they're on Azure, on-premises, or on another public cloud such as Amazon Web Services (AWS).

It provides a single set of tools and a common management experience for all of your data integration. What follows takes a closer look at ADF, starting with ADF pipelines.

GlobalLogic®
A Hitachi Group Company

Using ADF Pipelines

An effective data integration service must provide several components:

A way to perform specific actions. You might need to copy data from one datastore to another, for example, or to run a Spark job to process data. To allow this, ADF provides activities, each focused on carrying out a specific task.

A mechanism to specify the overall logic of your data integration process. This is what an ADF pipeline does, calling activities to carry out each step in the process.

A tool for authoring and monitoring the execution of pipelines and the activities they depend on.

Figure 1 illustrates how these aspects of ADF fit together.

As the figure shows, you can create and monitor a pipeline using the pipeline authoring and monitoring tool. This browser-based graphical environment lets you create new pipelines without being a developer. People who prefer to use code can do this.

However, ADF also provides SDKs that allow the creation of pipelines in several languages. Each pipeline runs in the Azure data center you choose, calling on one or more activities to carry out its work

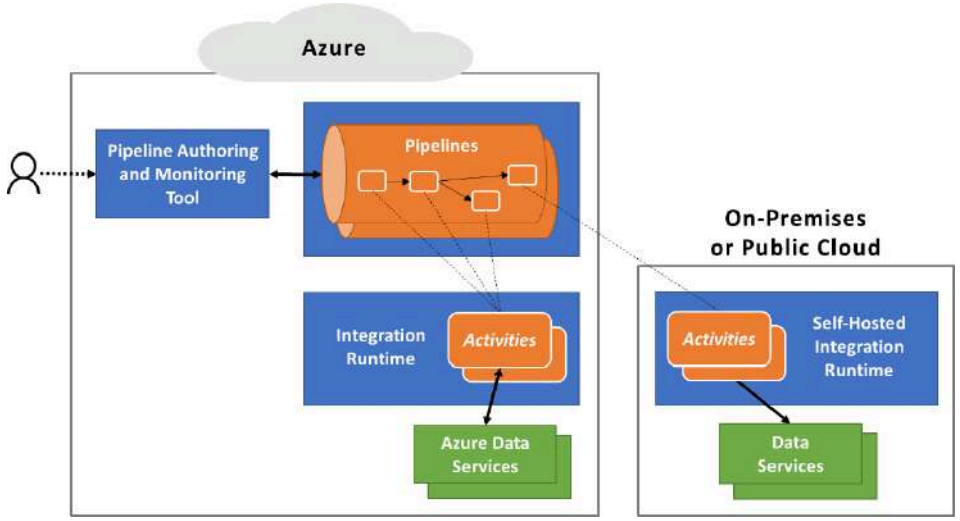


Figure 1: An ADF pipeline controls the execution of activities, each of which runs on an integration runtime.

If an activity runs in Azure (either in the same data center as the pipeline or another Azure data center), it relies on the Integration Runtime (IR).

An activity can also run on-premises or in another public cloud, such as AWS. In this case, the activity relies on the Self-Hosted Integration Runtime. This is essentially the same code as the Azure IR, but you must install it wherever you need it to run. But why bother with the Self-Hosted IR?

Why can’t all activities run on Azure?

The most common answer is that activities on Azure may not be able to directly access on-premises data sources, such as those that sit behind firewalls.

It’s often possible to configure the connection between Azure and on-premises data sources so that there is a direct connection (if you do, you don’t need to use the Self-Hosted IR), but not always. For example, setting up a direct connection from Azure to an on-premises data source might require working with your network administrator to configure your firewall in a specific way, something admins aren’t always happy to do.

The Self-Hosted IR exists for situations like this. It provides a way for an ADF pipeline to use an activity that runs outside Azure while giving it a direct connection back to the cloud.

A single pipeline can use many different Self-Hosted IRs, along with the Azure IR, depending on where its activities need to execute. It’s entirely possible, for example, that a single pipeline uses activities running on Azure, on AWS, inside your organization, and in a partner organization. All but the activities on Azure could run on instances of the Self-Hosted IR.

GlobalLogic®
A Hitachi Group Company

Scenarios

To get a sense of how you can use ADF pipelines, it's helpful to look at real scenarios. This section describes two:

1. Building a modern data warehouse on Azure, and
2. Providing the data analysis back end for a Software as a Service (SaaS) application.

Building a Modern Data Warehouse

Data warehouses let an organization store large amounts of historical data, then analyze it to understand its customers, revenue, or other things. Most data warehouses today are on-premises, using technology such as SQL Server.

Going forward, however, data warehouses are moving into the cloud. There are some excellent reasons for this, including low-cost data storage (which means you can store more data) and massive amounts of processing power (which lets you do more analysis on that data).

In any case, creating a modern data warehouse in the cloud requires a way to automate data integration throughout your environment. ADF pipelines are designed to do precisely this.

Figure 2 shows an example of data movement and processing that can be automated using ADF pipelines.

In this scenario, data is first extracted from an on-premises Oracle database and Salesforce.com (step 1).

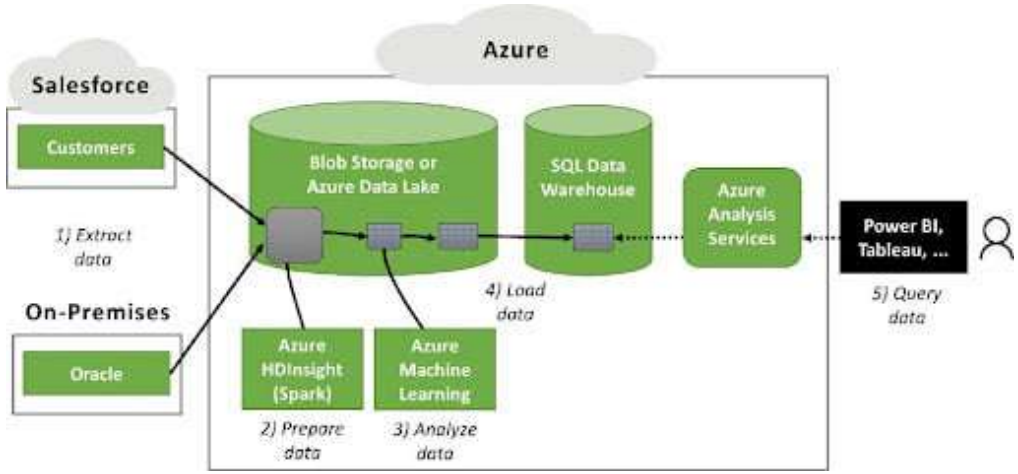


Figure 2: A modern data warehouse loads diverse data into a data lake, does some processing on that data, then loads a relevant subset into a relational data warehouse for analysis.

This data isn't moved directly into the data warehouse, however. Instead, it's copied into a data lake, a much less expensive form of storage implemented using either Blob Storage or Azure Data Lake.

Unlike a relational data warehouse, a data lake typically stores data in its original form. If this data is relational, the data lake can store traditional tables. But if it's not relational (you might be working with a stream of tweets, for example, or clickstream data from a web application), the data lake stores your data in whatever form it's in.

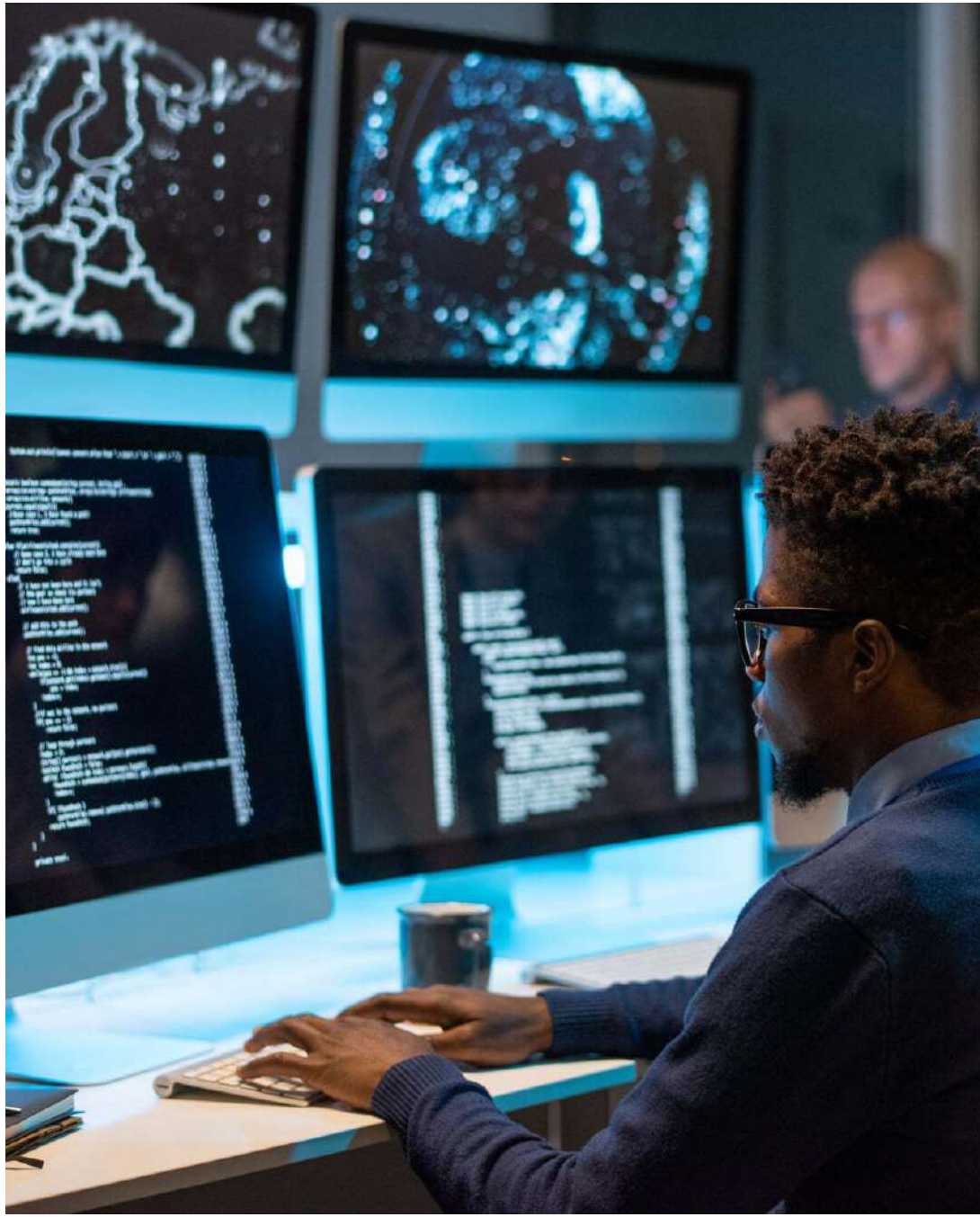
Why do this?

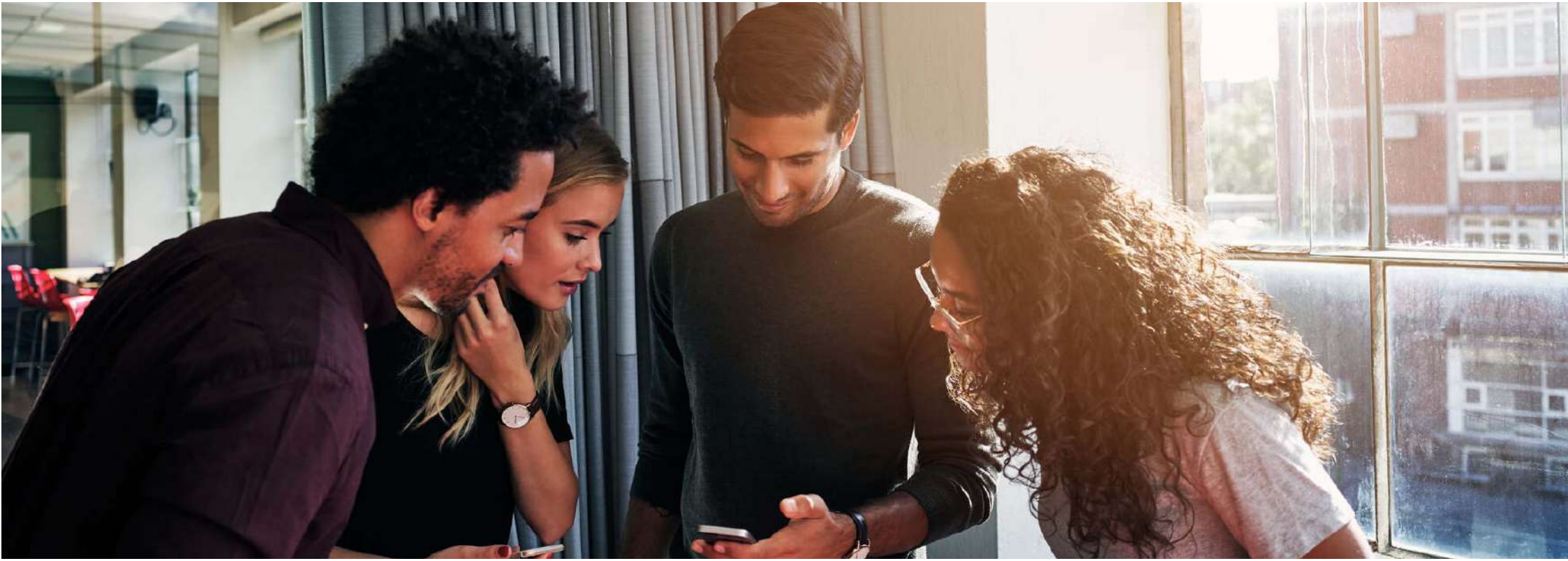
Rather than using a data lake, why not transform the data as needed and dump it directly into a data warehouse?

The answer stems from the fact that organizations are storing ever-larger amounts of increasingly diverse data. Some of that data might be worth processing and copying into a data warehouse, but much of it might not.

Because data lake storage is so much less expensive than data warehouse storage, you can afford to dump large amounts of data into your lake, then decide later which of it is worth processing and copying to your more expensive data warehouse.

In this era of big data, using a data lake and your cloud data warehouse together gives you more options at a lower cost.





Suppose you'd like to prepare some of the data just copied into the data lake to get it ready to load into a relational data warehouse for analysis.

Doing this might require cleaning that data somehow, such as by deleting duplicates. It might also require transforming it, such as by shaping it into tables. If there's a lot of data to process, you want this work to be done in parallel so that it won't take too long.

On Azure, you might run your prepare and transform application on an HDInsight Spark cluster (step 2). In some situations, an organization might copy the resulting data directly into Azure SQL Data Warehouse.

But it can also be helpful to do some more work on the prepared data first. For example, suppose the data contains calls made by customers of a mobile phone company. Using machine learning, the company can use this call data to estimate how likely each customer is to churn (switch to a competitor).

In the scenario shown in Figure 2, the organization uses Azure Machine Learning to do this (step 3).

Suppose each row in the table produced in step 2 represents a customer, for example. In that case, this step could add another column to the table containing the estimated probability that each customer will churn.

The critical thing to realize is that, along with traditional analysis techniques, you're also free to use data science tools on the contents of your Azure data lake.

Now that the data has been prepared and had some initial analysis, it's finally time to load it into SQL Data Warehouse (step 4).

(While this technology focuses on relational data, it can also access non-relational data using PolyBase.)

Most likely, the warehoused data will be accessed by Azure Analysis Services, which allows scalable interactive queries from users via Power BI, Tableau, and other tools (step 5).

This complete process has several steps. If it needed to be done just once, you might choose to do each step manually. In most cases, though, the process will run over and over, regularly feeding new data into the warehouse.

This implies that the entire process should be automated, which is precisely what ADF allows.

You can create one or more ADF pipelines to orchestrate the process, with an ADF activity for each step. Even though ADF isn't shown in Figure 2, it is nonetheless the cloud service driving every step in this

Providing Data Analysis for a SaaS Application

Most enterprises today use data analysis to guide their internal decisions. Increasingly, however, data analysis is also crucial to independent software vendors (ISVs) building SaaS applications.

For example, suppose an application provides connections between you and other users, including recommendations for new people to connect with. Doing this requires processing a significant amount of data regularly, then making the results available to the SaaS application.

Even simpler scenarios, such as providing detailed customization for each app user, can require significant back-end data processing.

This processing looks much like what's required to create and maintain an enterprise data warehouse, and ADF pipelines can be used to automate the work.

Figure 3 shows an example of how this might look.

This scenario looks much like the previous example. It begins with data extracted from various sources into a data lake (step 1).

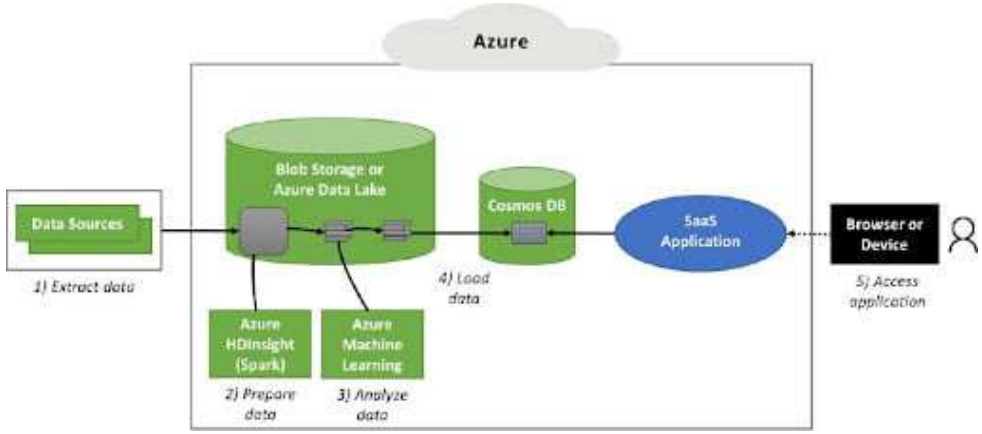


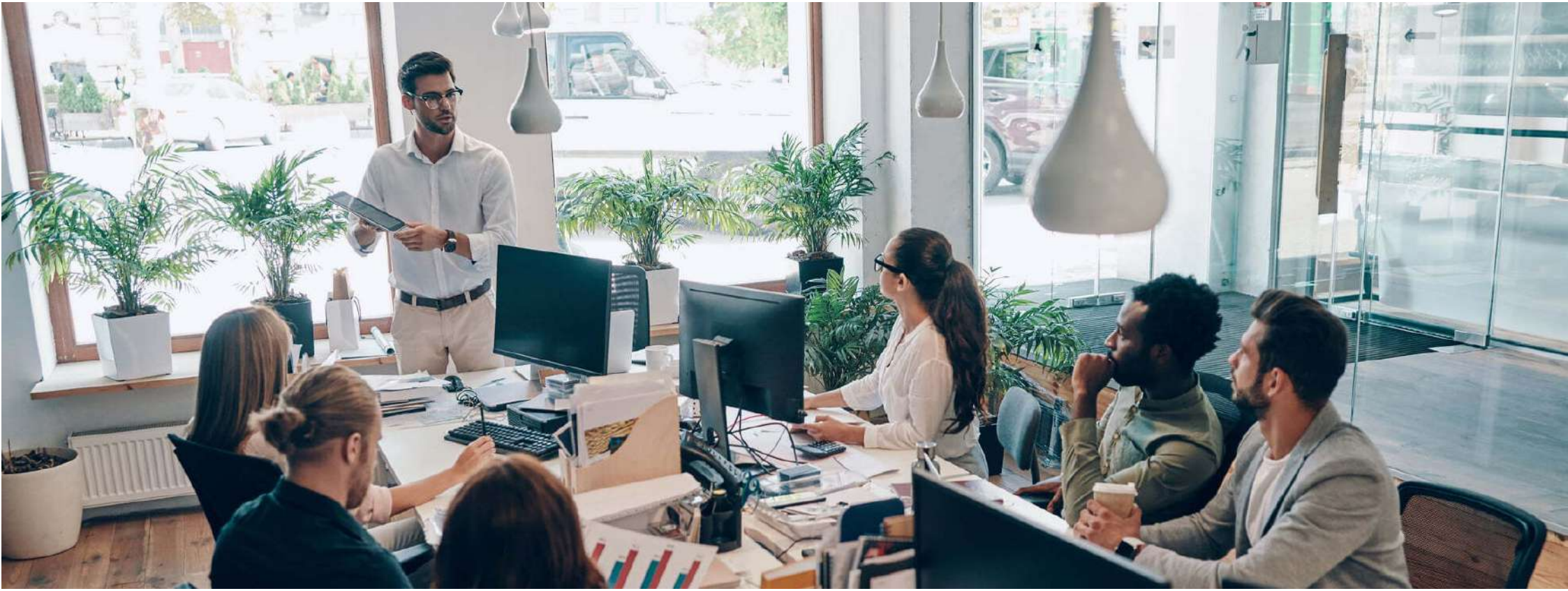
Figure 3: A SaaS application can require extensive back-end data processing.

This data is then prepared, such as with a Spark application (step 2), and perhaps processed using data science technologies such as Azure Machine Learning (step 3).

The resulting data isn't typically loaded into a relational data warehouse, however.

Instead, this data is a fundamental part of the service the application provides to its users.

Accordingly, it's copied into the operational database this application uses, which in this example is Azure Cosmos DB (step 4).



Unlike the scenario shown in Figure 2, the primary goal here isn't to allow interactive queries on the data through standard BI tools (although an ISV might also provide that for its internal use).

Instead, it's to give the SaaS application the data it needs to support its users, who access this app through a browser or device (step 5). And as in the previous scenario, an ADF pipeline can be used to automate this entire process.

Several applications already use ADF for scenarios like these, including Adobe Marketing Cloud and Lumdex, a healthcare data intelligence company.

As big data becomes increasingly important, expect to see others follow suit.

GlobalLogic®
A Hitachi Group Company

A Closer Look at Pipelines

Understanding the basics of ADF pipelines isn't hard. Figure 4 shows the components of a simple example.

One way to start a pipeline running is to execute it on demand. You can do this through PowerShell, by calling a RESTful API, through .NET, or by using Python.

A pipeline can also start executing because of some trigger. For example, ADF provides a scheduler trigger that starts a pipeline running at a specific time. However it starts, a pipeline always runs in some Azure data center.

The activities a pipeline uses might run either on the Azure IR, which is also in an Azure data center or on the Self-Hosted IR, which runs either on-premises or on another cloud platform. The pipeline shown in Figure 4 uses both options.

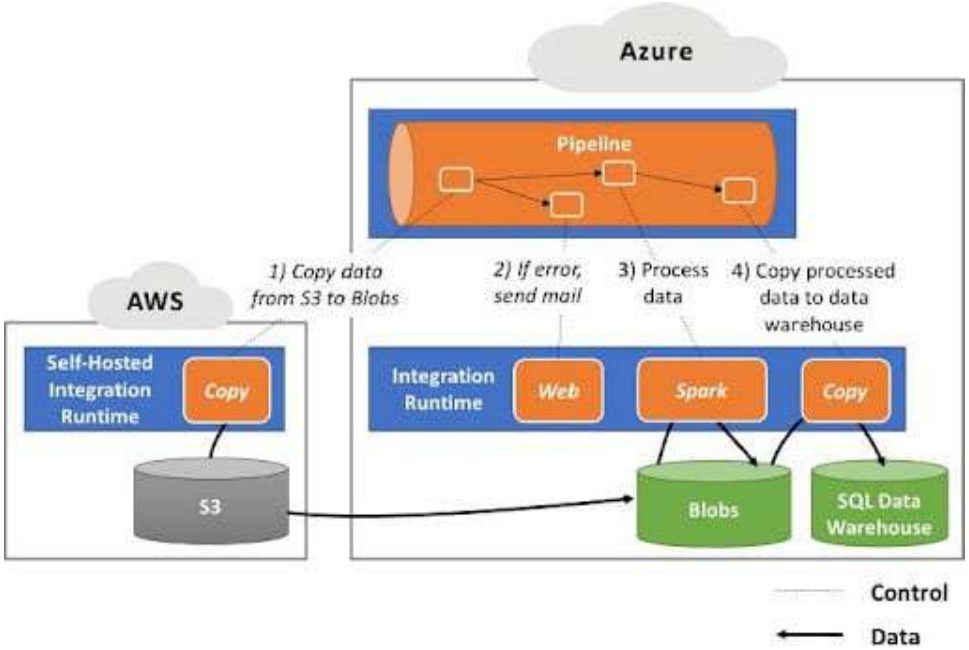


Figure 4: A pipeline executes one or more activities, each carrying out a step in a data integration workflow.

Using Activities

Pipelines are the operation's boss, but activities do the actual work. Which activities a pipeline invokes depends on what the pipeline needs to do. For example, the pipeline in Figure 4 carries out several steps, using an activity for each one. Those steps are:

1. **Copy data from AWS Simple Storage Service (S3) to Azure Blobs.** This uses ADF's Copy activity, which runs on an instance of the Self-Hosted IR installed on AWS.

2. **If this copy fails, the pipeline invokes ADF's Web activity** to send an email informing somebody of this. The Web activity can call an arbitrary REST endpoint, so in this case, it invokes an email service to send the failure message.

3. **If the copy succeeds, the pipeline invokes ADF's Spark activity**. This activity runs a job on an HDInsight Spark cluster. In this example, that job does some processing on the newly copied data, then writes the result back to Blobs.

4. **Once the processing is complete, the pipeline invokes another Copy activity**, this time to move the processed data from Blobs into SQL Data Warehouse.

The example in Figure 4 gives you an idea of what activities can do, but it's pretty simple. Activities can do much more.

For example, the Copy activity is a general-purpose tool to move data efficiently from one place to another. It provides built-in support for dozens of data sources and sinks—it's data movement as a service.

Among the options it supports are virtually all Azure data technologies, AWS S3 and Redshift, SAP HANA, Oracle, DB2, Mongo DB, and many more.

These can be scaled as needed, speeding up data transfers by letting them run in parallel, with speeds up to one gigabit per second.

ADF also supports a much more comprehensive range of activities than in Figure 4. Along with the Spark activity, for example, it provides activities for other approaches to data transformation, including Hive, Pig, U-SQL, and stored procedures.

ADF also provides a range of control activities, including If Condition for branching, Until for looping, and For Each for iterating over a collection.

These activities can also scale out, letting you run loops and more in parallel for better performance.

Authoring Pipelines

Pipelines are described using JavaScript Object Notation (JSON), and anyone using ADF is free to author a pipeline by writing JSON directly. But many people who work with data integration aren't developers; they prefer graphical tools. For this audience, ADF provides a web-based tool for authoring and monitoring pipelines. There's no need to use Visual Studio. Figure 5 shows an example of authoring a simple pipeline.

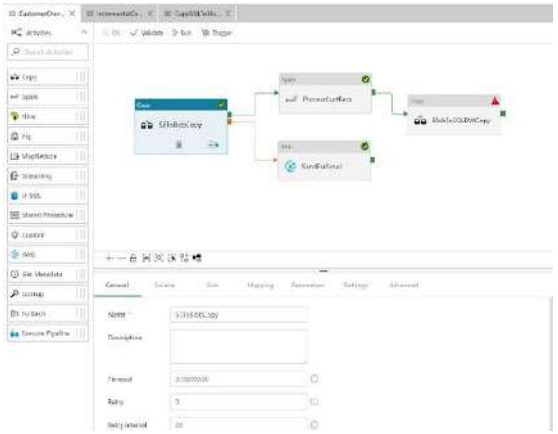


Figure 5: The ADF authoring and monitoring tool lets you create pipelines graphically by dragging and dropping activities onto a design surface.

This example shows the same simple pipeline illustrated earlier in Figure 4. Each of the pipeline's activities — the two Copies, Spark, and Web — is represented by a rectangle, with arrows defining the connections between them. Some other available activities are shown on the left, ready to be dragged and dropped into a pipeline as needed.

The first Copy activity is highlighted, bringing up space at the bottom to give it a name (used in monitoring the pipeline's execution), a description, and a way to set parameters for this activity.

Note: It's possible to pass parameters into a pipeline, such as the name of the AWS S3 bucket to copy from and to pass the state from one activity to another within a pipeline.

Every pipeline also exposes its REST interface, which an ADF trigger uses to start a pipeline.

This tool generates JSON, which can be examined directly as it's stored in a git repository.

Still, this isn't necessary to create a pipeline. This graphical tool lets an ADF user create fully functional pipelines with no knowledge of how those pipelines are described under the covers.

Monitoring Pipelines

In a perfect world, all pipelines would complete successfully, and there would be no need to monitor their execution.

In the real world, however, pipelines can fail. One reason is that a single pipeline might interact with multiple cloud services, each of which has its failure modes.

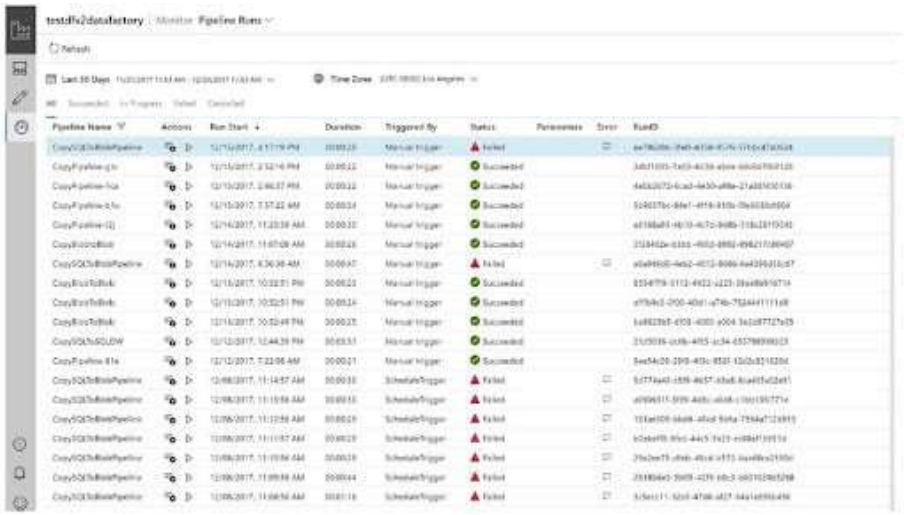


Figure 6: The ADF authoring and monitoring tool lets you monitor pipeline execution, showing when each pipeline started, how long it ran, its current status, and more.

For example, it's possible to pause a SQL Data Warehouse instance, something that might cause an ADF pipeline using this instance to fail.

But whatever the reason, the reality is the same: We need an effective tool for monitoring pipelines. ADF provides this as part of the authoring and monitoring tool. Figure 6 shows an example.

As this example shows, the tool lets you monitor the execution of individual pipelines. You can see when each one started, for example, how it was started, whether it succeeded or failed, and more. A primary goal of this tool is to help you find and fix failures. To help do this, the tool lets you look further into the execution of each pipeline.

For example, clicking on the Actions column for a specific pipeline brings up an activity-by-activity view of that pipeline's status, including any errors that have occurred, what ADF Integration Runtime it's using, and other information.

If an activity failed because someone paused the SQL Data Warehouse instance it depended on, for example, you'll be able to see this directly.

The tool also pushes all of its monitoring data to Azure Monitor, the common clearinghouse for monitoring data on Azure.

GlobalLogic®
A Hitachi Group Company

Pricing

Pricing for ADF pipelines depends primarily on two factors: the number of activities being run and the volume of data being moved.

How many activities do your pipelines run?

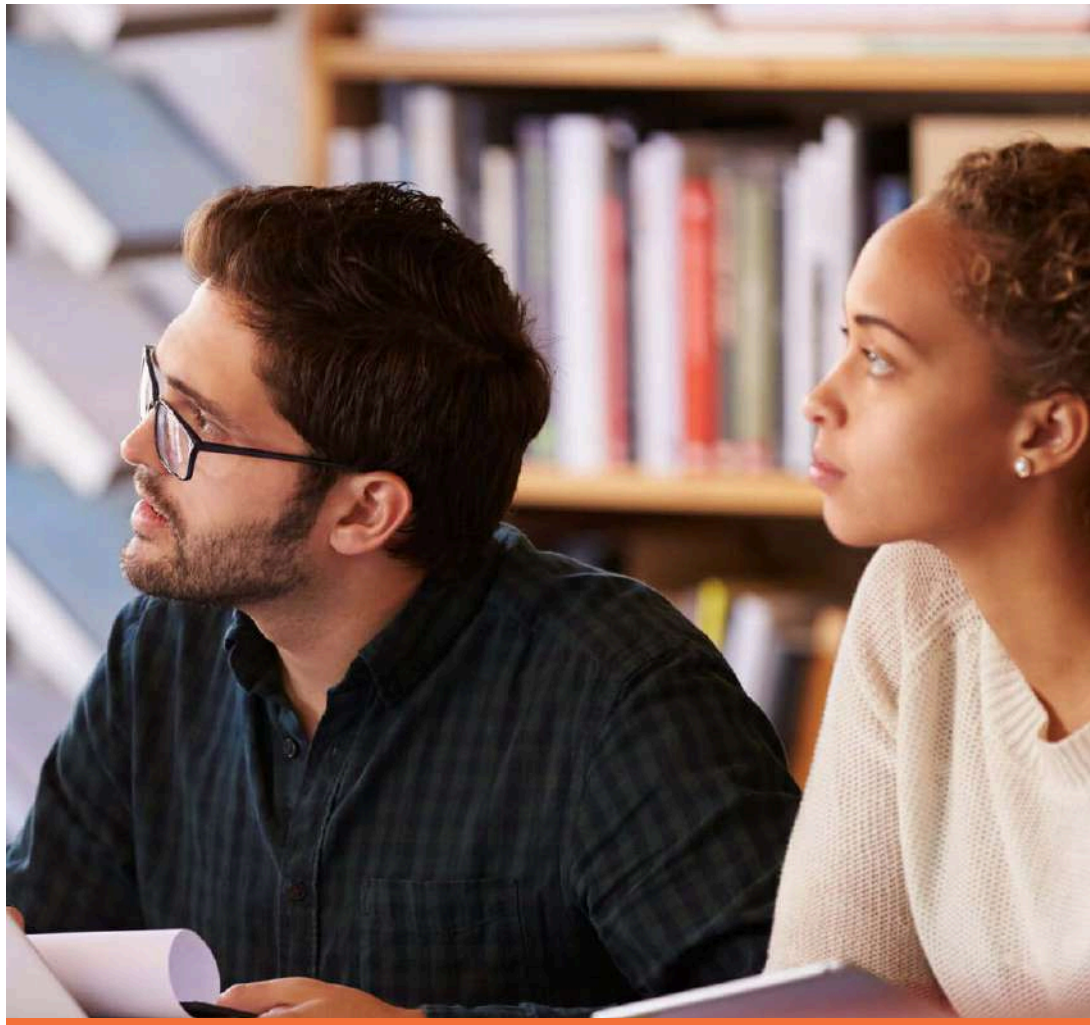
Activities that run on the Azure IR are a bit cheaper than those run on the Self-Hosted IR.

How much data do you move?

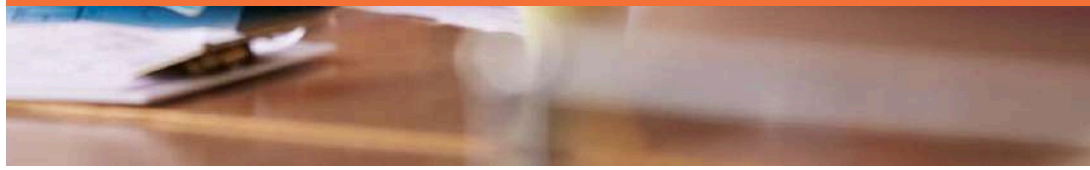
You pay by the hour for the compute resources used for data movement, e.g., the data moved by a Copy activity.

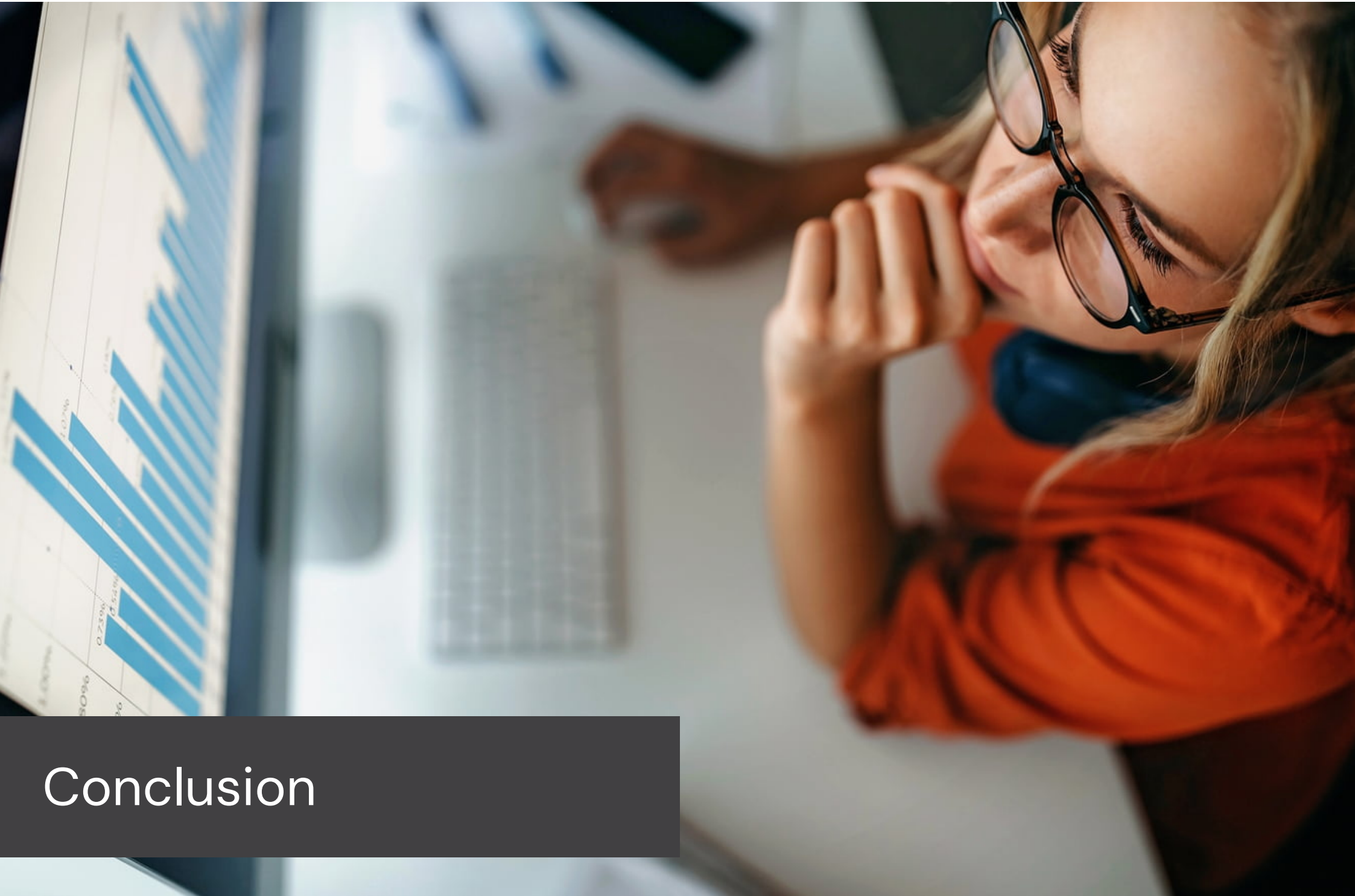
As with activities, the prices for data movement with the Azure IR vs. the Self-Hosted IR differ (although, in this case, using the SelfHosted IR is cheaper). You will also incur the standard charges for moving data from an Azure data center.

It's also worth noting that you'll be charged separately for any other Azure resources your pipeline uses, such as blob storage or a Spark cluster.



For current details on ADF pipeline pricing, see here.





Conclusion

Data integration is a critical function in many on-premises data centers. As our industry moves to the cloud, it will remain a fundamental part of working with data.

Azure Data Factory addresses two main data integration concerns that organizations have today:

1. A way to automate data workflows in Azure, on-premises, and across other clouds using ADF pipelines. This includes the ability to run data transformation activities both on Azure or elsewhere, along with a single view for scheduling, monitoring, and managing your pipelines.

2. A managed service for running SSIS packages on Azure.

If you're an Azure user facing these challenges, ADF is almost certainly in your future.

The time to start understanding this new technology is now.

About the Author

Sahil Gupta is a results-driven Engineering Consultant with more than 12 years of experience in software design and development, primarily using Oracle Database PL/SQL Technologies.