Azure Data Factory

Data Integration in the Cloud

Contents

Data analytics today

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 needs to embrace the new approaches to data analytics that cloud computing makes possible.

Microsoft Azure provides a broad set of cloud technologies for data analysis, all 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 *Blob*s, which 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, offering U-SQL, a tool for distributed analysis of data 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, 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 an important aspect of data analytics that none of them 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:

- 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, and 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 one defines a process to extract, load, transform, or otherwise work with data. For this audience, ADF provides the ability to run SSIS packages on Azure, letting them access both on-premises and cloud data services.

The key 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.

ADF v2

This paper describes the second release of ADF, sometimes called ADF v2. It's a big change from the original ADF, with additions that include these:

- A new approach to creating and running ADF pipelines, providing a significantly broader set of options that address a wider range of scenarios.
- Graphical tools for authoring and monitoring pipelines, making it easier for non-developers to use this technology.
- Connectors to many more data sources, including other cloud platforms and SaaS applications, that make it simpler to move data.
- Support for running SSIS packages in the cloud.

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 data store to another, for example, or to run a Spark job to process data. To allow this, ADF provides a set of activities, each focused on carrying out a specific kind of 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.

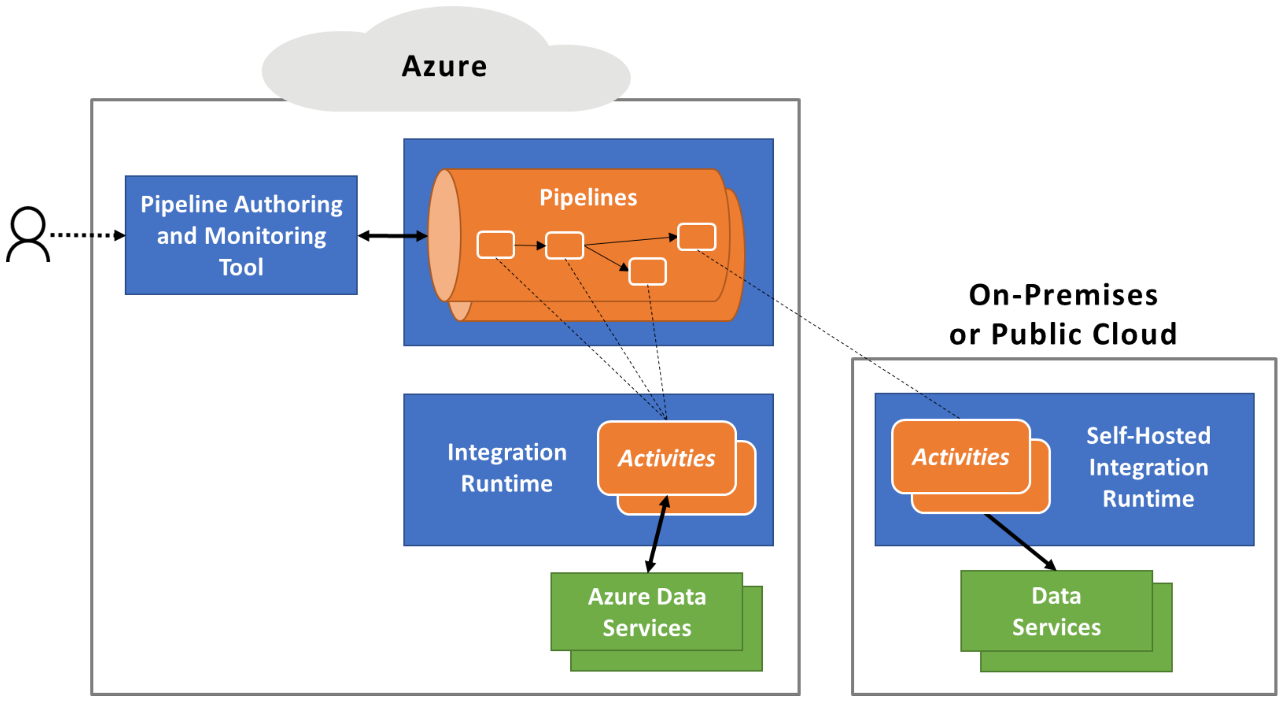

Azure
Pipeline Authoring and Monitoring Tool
Pipelines
Integration Runtime
Activities
Azure Data Services
On-Premises or Public Cloud
Activities
Self-Hosted Integration Runtime
Data Services

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

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 creating pipelines in several different languages.

Each pipeline runs in the Azure datacenter you choose, calling on one or more activities to carry out its work. If an activity runs in Azure (either in the same datacenter as the pipeline or in some other Azure datacenter), it relies on the *Integration Runtime (IR)*. An activity can also run on-premises or in some other 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 often won’t 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 this isn’t always possible. 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 that 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 still giving that activity 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 inside a partner organization. All but the activities on Azure could run on instances of the Self-Hosted IR.

Scenarios

To get a sense of how you can use ADF pipelines, it's useful to look at real scenarios. This section describes two of these: building a modern data warehouse on Azure, and 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 that data to better 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 very good 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). But creating a modern data warehouse in the cloud requires a way to automate data integration throughout your environment. ADF pipelines are designed to do exactly this. Figure 2 shows an example of data movement and processing that can be automated using ADF pipelines.

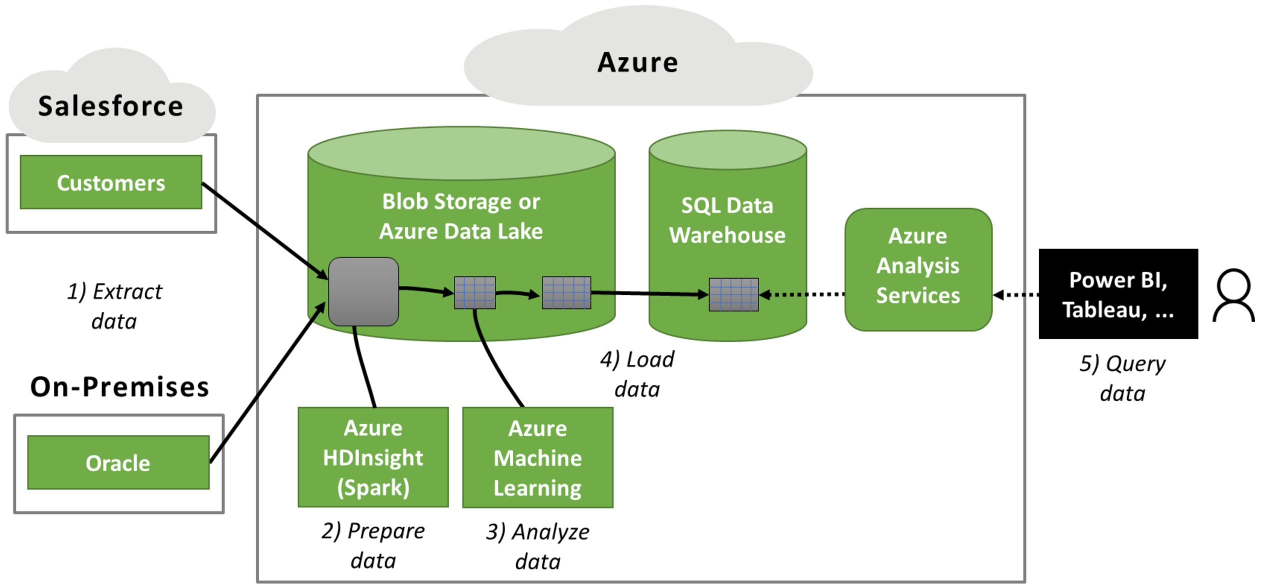


Figure 2: A modern data warehouse loads diverse data into a data lake, does some processing on that data, then loads a relevant subset into a relational data warehouse for analysis.

In this scenario, data is first extracted from an on-premises Oracle database and Salesforce.com (step 1). 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 instead 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 that 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 the big data era, using a data lake along with your cloud data warehouse gives you more options at a lower cost.

Now 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 in some way, such as by deleting duplicates, and 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 it won't take too long. On Azure, you might do this by running 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 useful to first do some more work on the prepared data. 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, i.e., to switch to a competitor. In the scenario shown in Figure 2, the organization uses Azure Machine Learning to do this (step 3). If each row in the table produced in step 2 represents a customer, for example, this step could add another column to the table containing the estimated probability that each customer will churn. The key 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 done on it, 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 an option called 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 exactly 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 that's driving every step in this scenario.

Providing data analysis for a SaaS application

Most enterprises today use data analysis to guide their internal decisions. Increasingly, however, data analysis is also important 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 on a regular schedule, then making the results available to the SaaS application. Even simpler scenarios, such as providing detailed customization for each of an app's users, 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.

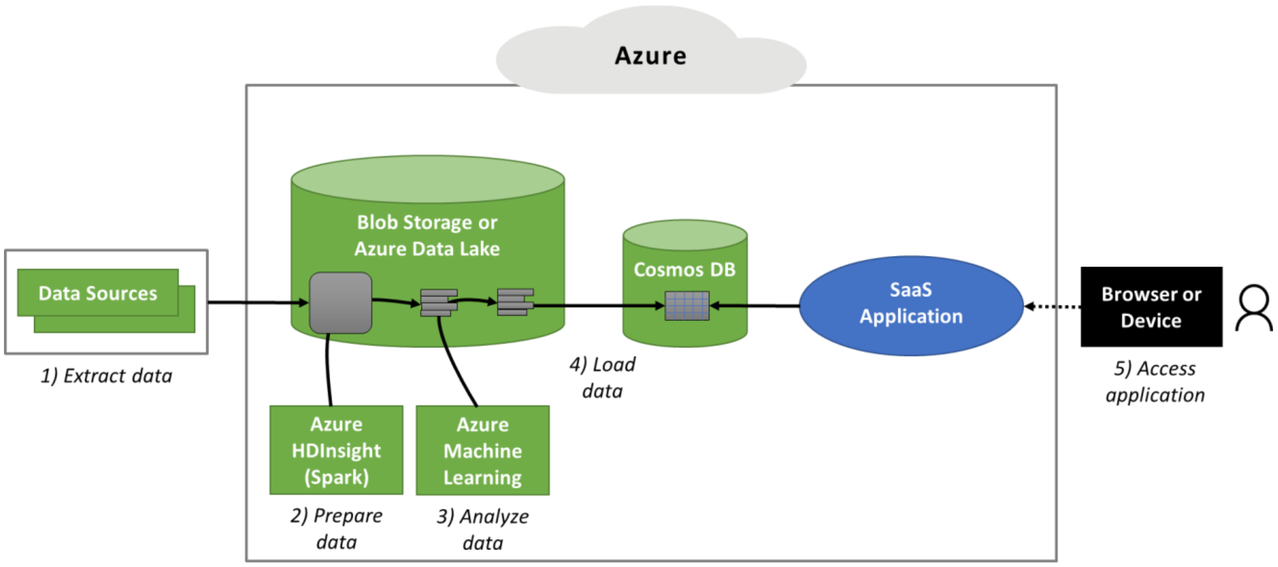


Figure 3: A SaaS application can require extensive back-end data processing.

This scenario looks much like the previous example. It begins with data extracted from various sources into a data lake (step 1). 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 is using, 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 common BI tools (although an ISV might provide that as well for its own 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.

A number of applications already use ADF for scenarios like these, including Adobe Marketing Cloud and Lumdex, a healthcare data intelligence company. As big data becomes more and more important, expect to see others follow suit.

A closer look at pipelines

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

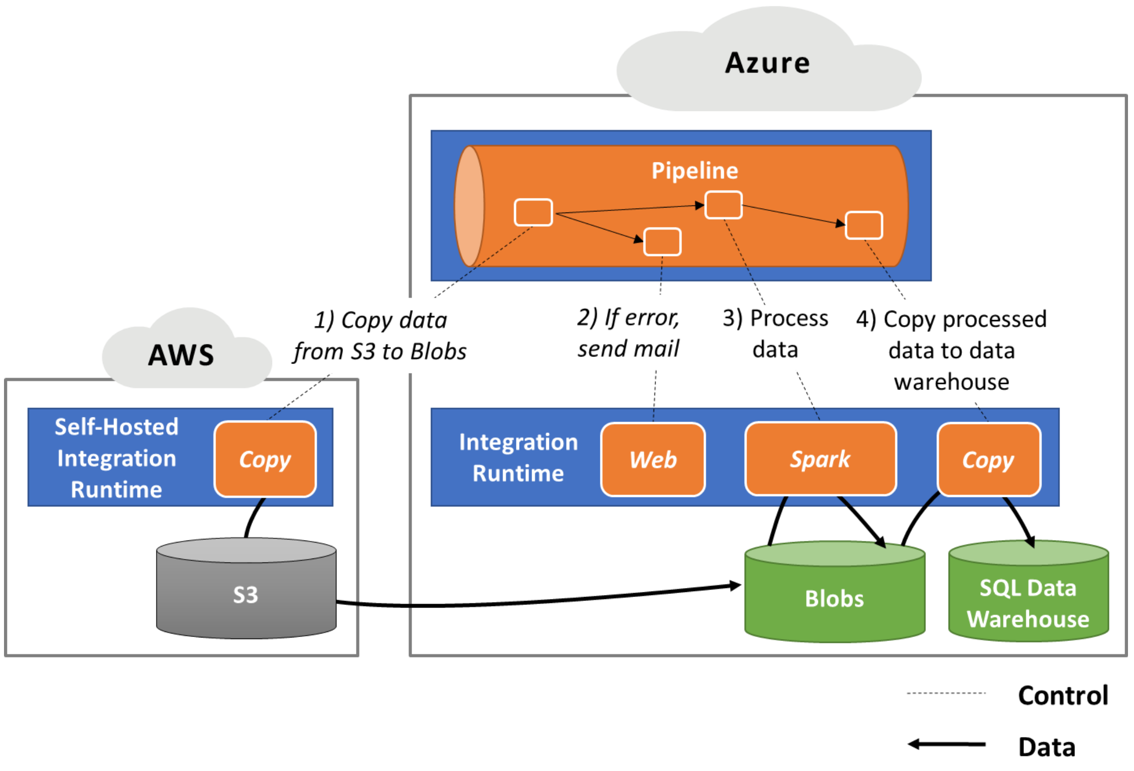


Figure 4: A pipeline executes one or more activities, each of which carries out a step in a data integration workflow.

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 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 datacenter. The activities a pipeline uses might run either on the Azure IR, which is also in an Azure datacenter, 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.

Using activities

Pipelines are the boss of the operation, but activities do the real work. Which activities a pipeline invokes depend 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 is running on an instance of the Self-Hosted IR that has been 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.

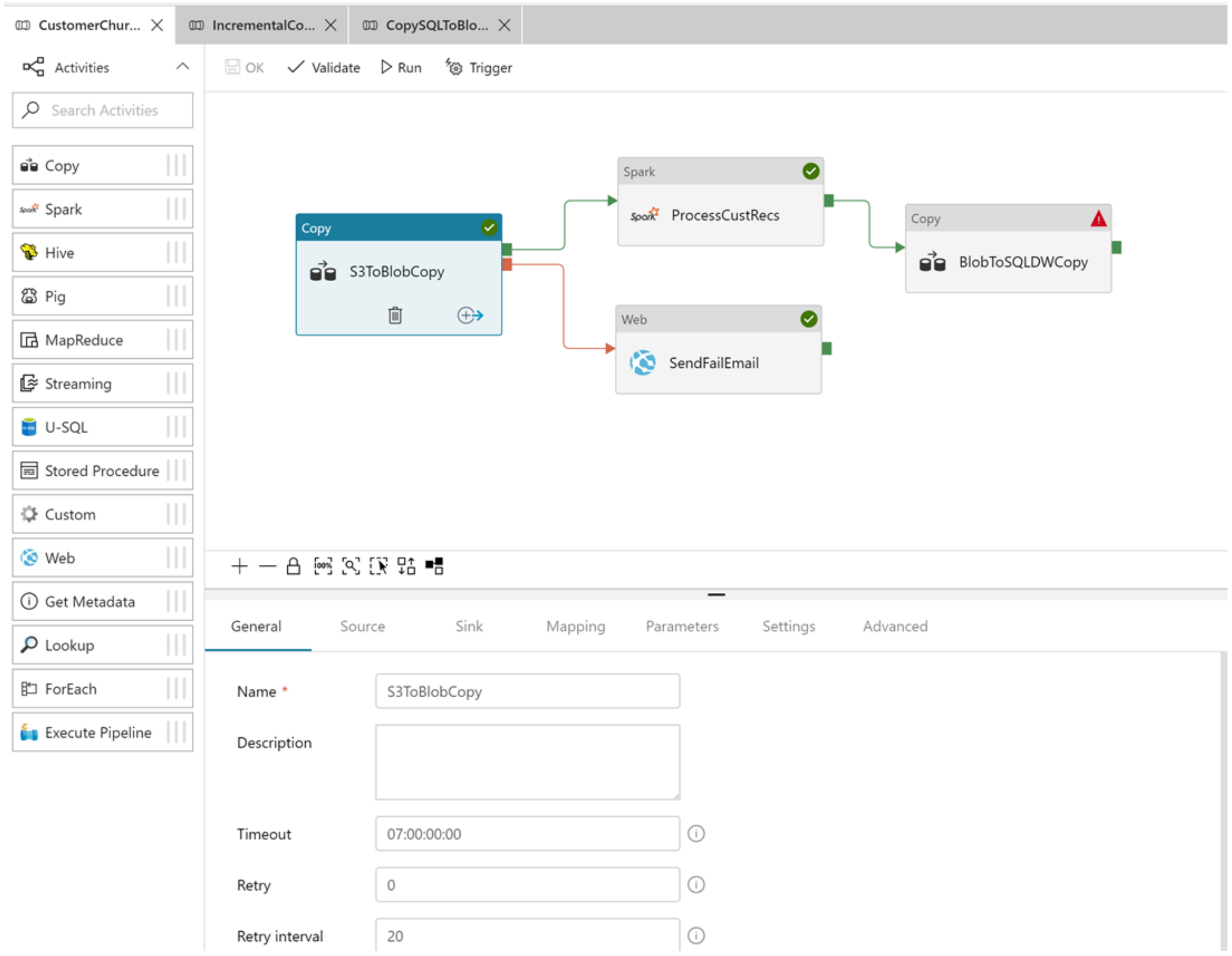
4) Once the processing is complete, the pipeline invokes another Copy activity, this time to move the processed data from Blobs into SQL Data Warehouse.

The example in Figure 4 gives you an idea of what activities can do, but it’s quite simple. In fact, activities can do much more. For example, the Copy activity is actually 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. All of 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 wider range of activities than those shown 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 ForEach 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 anybody 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.



Name *	S3ToBlobCopy
Description	
Timeout	07:00:00:00
Retry	0
Retry interval	20

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

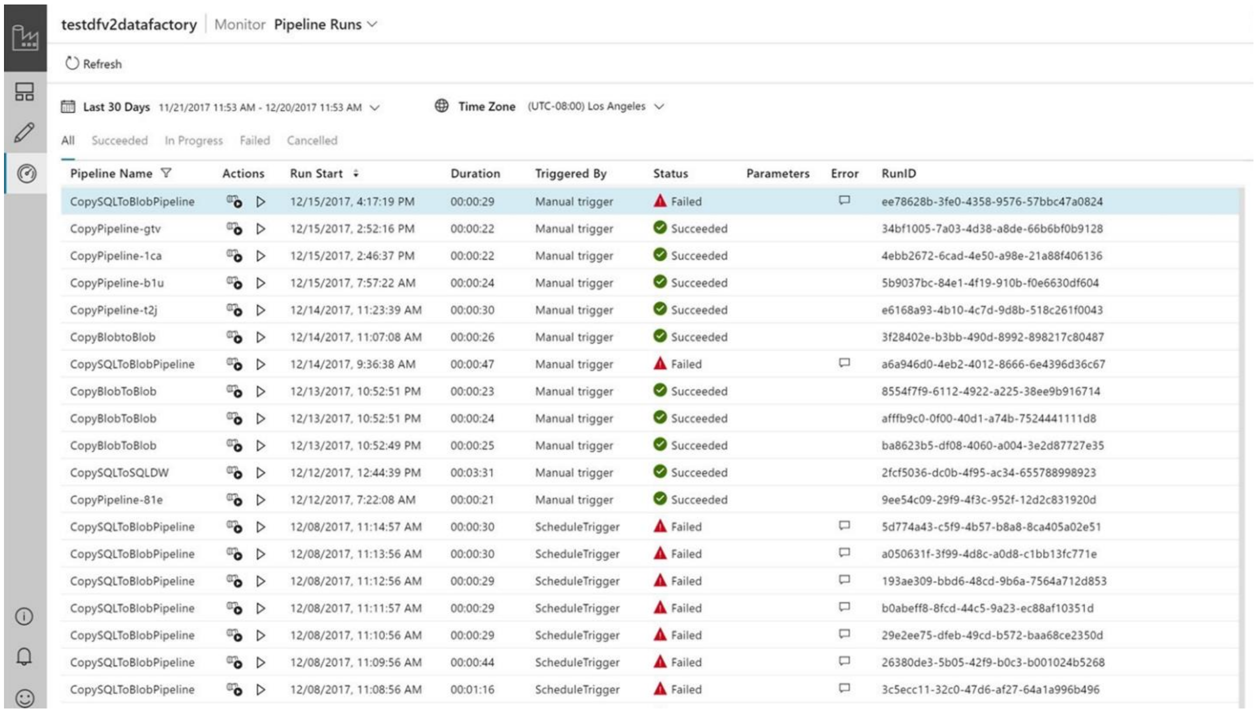
This example shows the same simple pipeline illustrated earlier in Figure 4. Each of the pipeline's activities—the two Copys, Spark, and Web—is represented by a rectangle, with arrows defining the connections between them. Some of the other available activities are shown on the left, ready to be dragged 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. It's possible to pass parameters into a pipeline, such as the name of the AWS S3 bucket to copy from, and to pass state from one activity to another within a pipeline. Every pipeline also exposes its own REST interface, which is what an ADF trigger uses to start a pipeline.

This tool generates JSON, which can be examined directly—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 world we live in, however, pipelines can fail. One reason for this is that a single pipeline might interact with multiple cloud services, each of which has its own failure modes. 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.


testdfv2datafactory
Monitor
Pipeline Runs
Refresh
Last 30 Days 11/21/2017 11:53 AM - 12/20/2017 11:53 AM
Time Zone (UTC-08:00) Los Angeles
All
Succeeded
In Progress
Failed
Cancelled

Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Error	RunID
CopySQLToBlobPipeline		12/15/2017, 4:17:19 PM	00:00:29	Manual trigger	Failed			ee78628b-3fe0-4358-9576-57bbc47a0824
CopyPipeline-gtv		12/15/2017, 2:52:16 PM	00:00:22	Manual trigger	Succeeded			34bf1005-7a03-4d38-a8de-66b6bf0b9128
CopyPipeline-1ca		12/15/2017, 2:46:37 PM	00:00:22	Manual trigger	Succeeded			4ebb2672-6cad-4e50-a98e-21a88f406136
CopyPipeline-b1u		12/15/2017, 7:57:22 AM	00:00:24	Manual trigger	Succeeded			5b9037bc-84e1-4f19-910b-f0e6630df604
CopyPipeline-t2j		12/14/2017, 11:23:39 AM	00:00:30	Manual trigger	Succeeded			e6168a93-4b10-4c7d-9d8b-518c261f0043
CopyBlobtoBlob		12/14/2017, 11:07:08 AM	00:00:26	Manual trigger	Succeeded			3f28402e-b3bb-490d-8992-898217c80487
CopySQLToBlobPipeline		12/14/2017, 9:36:38 AM	00:00:47	Manual trigger	Failed			a6a946d0-4eb2-4012-8666-6e4396d36c67
CopyBlobToBlob		12/13/2017, 10:52:51 PM	00:00:23	Manual trigger	Succeeded			8554f7f9-6112-4922-a225-38ee9b916714
CopyBlobToBlob		12/13/2017, 10:52:51 PM	00:00:24	Manual trigger	Succeeded			afffb9c0-0f00-40d1-a74b-7524441111d8
CopyBlobToBlob		12/13/2017, 10:52:49 PM	00:00:25	Manual trigger	Succeeded			ba8623b5-df08-4060-a004-3e2d87727e35
CopySQLToSQLDW		12/12/2017, 12:44:39 PM	00:03:31	Manual trigger	Succeeded			2fcf5036-dc0b-4f95-ac34-655788998923
CopyPipeline-81e		12/12/2017, 7:22:08 AM	00:00:21	Manual trigger	Succeeded			9ee54c09-29f9-4f3c-952f-12d2c831920d
CopySQLToBlobPipeline		12/08/2017, 11:14:57 AM	00:00:30	ScheduleTrigger	Failed			5d774a43-c5f9-4b57-b8a8-8ca405a02e51
CopySQLToBlobPipeline		12/08/2017, 11:13:56 AM	00:00:30	ScheduleTrigger	Failed			a050631f-3f99-4d8c-a0d8-c1bb13fc771e
CopySQLToBlobPipeline		12/08/2017, 11:12:56 AM	00:00:29	ScheduleTrigger	Failed			193ae309-bbd6-48cd-9b6a-7564a712d853
CopySQLToBlobPipeline		12/08/2017, 11:11:57 AM	00:00:29	ScheduleTrigger	Failed			b0abeff8-8fcd-44c5-9a23-ec88af10351d
CopySQLToBlobPipeline		12/08/2017, 11:10:56 AM	00:00:29	ScheduleTrigger	Failed			29e2ee75-dfeb-49cd-b572-baa68ce2350d
CopySQLToBlobPipeline		12/08/2017, 11:09:56 AM	00:00:44	ScheduleTrigger	Failed			26380de3-5b05-42f9-b0c3-b001024b5268
CopySQLToBlobPipeline		12/08/2017, 11:08:56 AM	00:01:16	ScheduleTrigger	Failed			3c5ecc11-32c0-47d6-af27-64a1a996b496

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.

As this example shows, the tool lets you monitor the execution of individual pipelines. You can see when each one started, for example, along with 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.

Pricing

Pricing for ADF pipelines depends primarily on two factors:

- How many activities 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 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 are different for data movement with the Azure IR vs. the Self-Hosted IR (although in this case, using the Self-Hosted IR is cheaper). You will also incur the standard charges for moving data out of an Azure datacenter.

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.

Running SSIS packages on ADF

Even though more and more data analysis uses cloud-native services, plenty of this work is still done in on-premises datacenters. With SQL Server, the most commonly used data integration option is SSIS, which lets organizations create SSIS packages.

If you want to move an existing data analytics system to Azure, one approach is to rearchitect and rebuild this system using ADF pipelines and services such as Azure SQL Data Warehouse. But suppose you're not ready to do this just yet. What if you instead want to move your existing data warehouse to the cloud? Maybe you need the extra scale that the cloud provides, or perhaps your organization has decided to move everything to the cloud, including data analytics. In situations like these, you need a way to lift and shift what you have, including SSIS packages, to Azure.

To make this possible, ADF provides the SSIS Integration Runtime. This technology is distinct from the other integration runtimes that ADF provides; its sole purpose is to run SSIS packages in the cloud. Figure 7 shows how this looks.

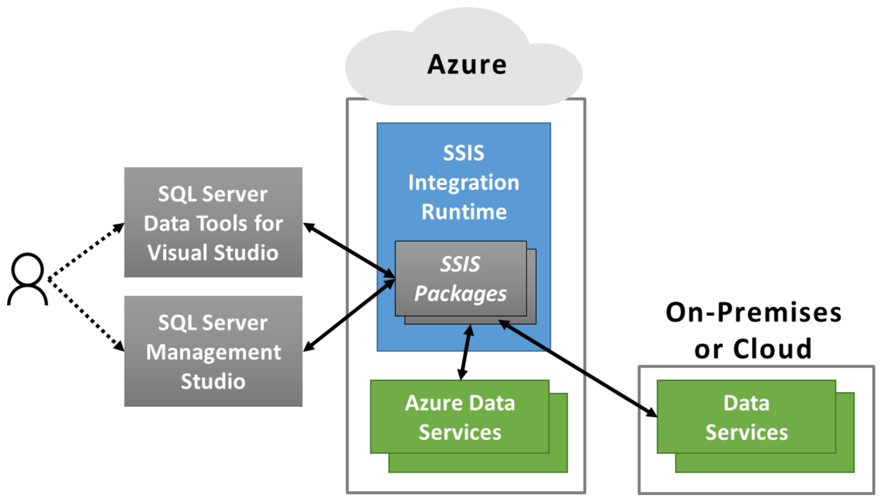


Figure 7: ADF's SSIS Integration Runtime is a managed service for running SSIS packages in the cloud.

To use this aspect of ADF, someone in your organization uses PowerShell or ADF visual tools to create an instance of the SSIS IR, then executes one or more SSIS packages on it. These packages can access Azure data services, data services in other cloud platforms, and on-premises data services. ADF doesn't provide tools for creating, deploying, or managing SSIS packages, however—there's no need to do this. Instead, you create SSIS packages using SQL Server Data Tools for Visual Studio, just as you do today, then use SQL Server Management Studio to deploy and manage those packages. In other words, working with SSIS packages on the SSIS IR looks just the same as working with SSIS packages on-premises.

The SSIS IR runs SSIS packages in ordinary Azure virtual machines. When you create an instance of this IR, you specify how many VMs you want to run based on the performance you need for your packages. ADF then creates and monitors this number of VMs, as Figure 8 shows.

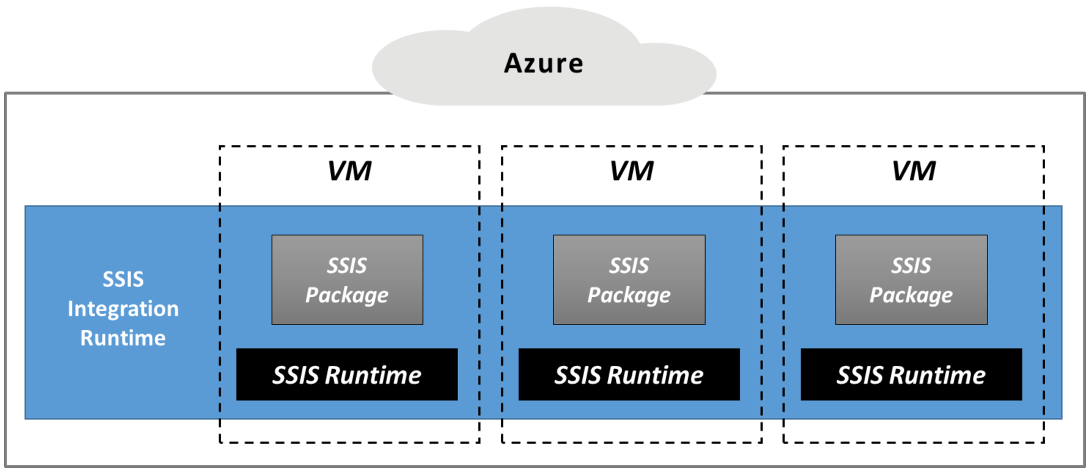


Figure 8: The SSIS Integration Runtime runs SSIS packages in a managed and scalable set of VMs.

Why bother to do this? Why not just create your own VMs, then run SSIS in them? You're free to do this, of course—ADF isn't the only way to run SSIS packages on Azure. But rather than creating, configuring, and managing your own VMs, why not let this ADF service do it for you? The SSIS IR can also increase and decrease the number of VMs you use based on demand, providing automatic scalability. The result is a managed and scalable service that runs existing SSIS packages unchanged on Azure.

Scenario: Moving an SSIS package to the cloud

Many organizations today use SSIS to automate the extract/transform/load (ETL) process for a data warehouse. Figure 9 shows a simple example.

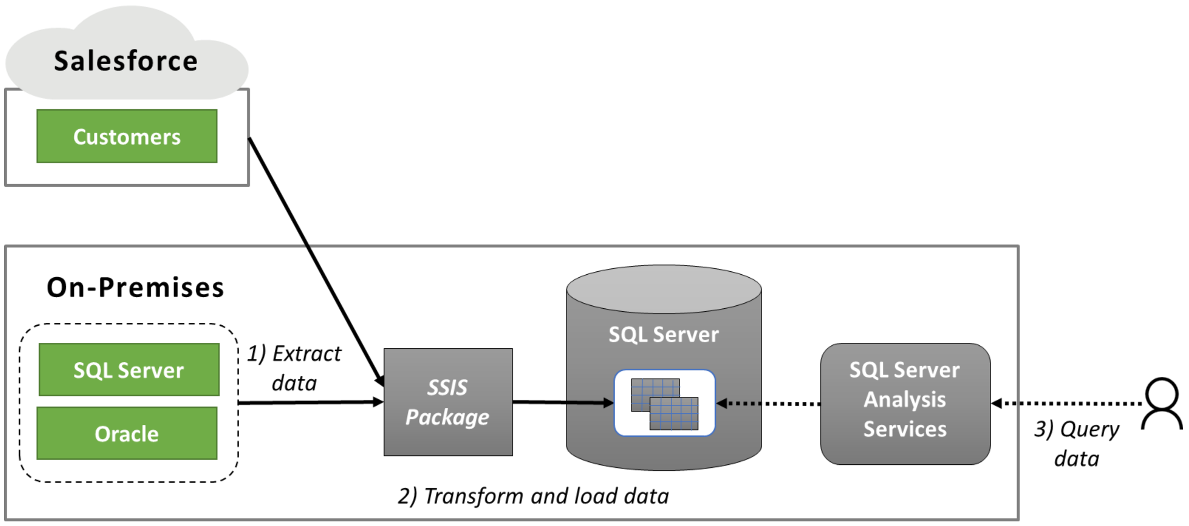


Figure 9: An SSIS package running on-premises today can access both on-premises and cloud data.

In this scenario, an SSIS package extracts data from a SaaS application and from two on-premises databases (step 1). The package next performs whatever transformations are required on that data, then loads it into a SQL Server data warehouse (step 2). Users can now query the warehoused data using SQL Server Analysis Services (step 3).

It's possible to move this environment, including the two on-premises databases, into Azure. In many situations, though, the group that owns the data warehousing function is different from the one that owns the on-premises databases. Even if the data warehousing group wants to move everything to the cloud, they might not have the ability to force the database group to comply. Figure 10 shows a realistic example of how this scenario might look, at least initially, when it's moved to the cloud.

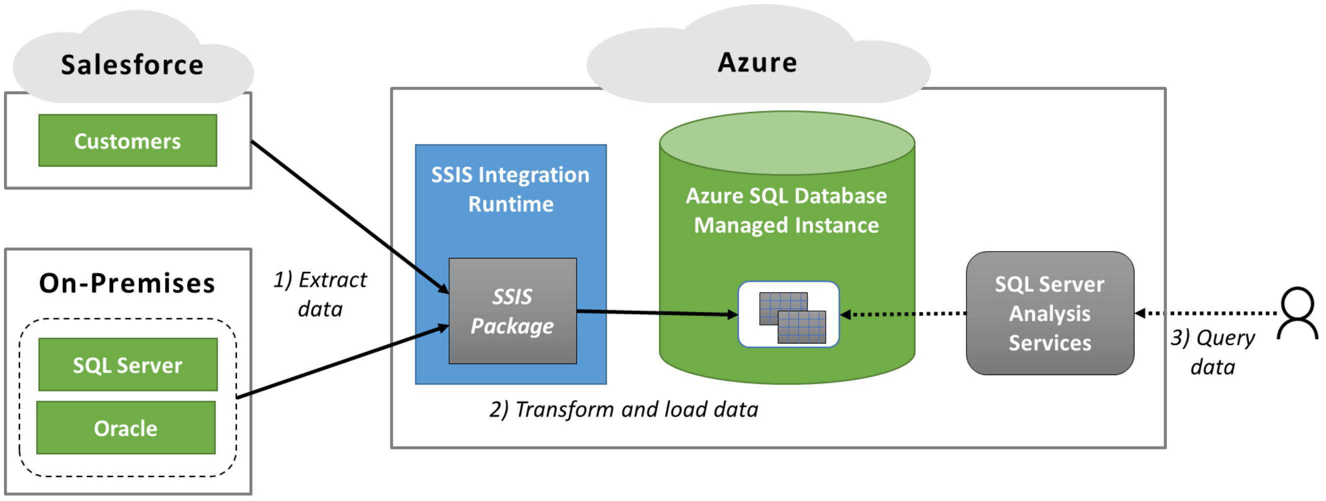


Figure 10: An SSIS package running on Azure can still access both on-premises and cloud data, and it's likely to use SQL Database Managed Instance for a data warehouse.

The general flow of the ETL process remains the same: extract data from the same data sources (step 1), transform and load it into a SQL Server data warehouse (step 2), then let users access it through SQL Server Analysis Services (step 3). There are some important differences, however. For example, the two databases remain on-premises, but the package can still access them. This communication doesn't rely on anything like the Self-Hosted IR used with ADF pipelines, however. Instead, administrators must configure their Azure VNETs and on-premises networks to make this access possible.

Also, the SQL Server data warehouse uses a specific aspect of the Azure SQL Database service: the Managed Instance option. SQL Database itself is a scalable relational service based on SQL Server. This service is designed for operational data, however, and so it doesn't support the data warehouse aspects of SQL Server. (It also doesn't support SSIS.) But SQL Database Managed Instance offers the ability to run an Azure-managed instance of full SQL Server, including its data warehousing capabilities. This makes Managed Instance a good fit for lifting and shifting SQL Server workloads up to 30 TBs into the cloud.

If your organization runs one or more smaller data warehouses or data marts, moving them to Azure can be straightforward. Doing this has the immediate benefit of reducing the work required to support them. Moving larger data warehouses can be more complex, but it's often a good way to start the process of re-architecting your analysis environment for the cloud. Whatever you choose to do, ADF can support your data integration needs.

Pricing

Since the SSIS IR relies on VMs, pricing is based on the number and size of the VMs you use. Each VM you request is charged per hour, so your overall cost will depend on how you use them. While you can leave your VMs running constantly, you might choose to save money by starting and stopping them based on what your SSIS packages need. For example, you might spin up your SSIS VMs at 6 pm each night, run them until the ETL processes they run are complete, then shut them down. It's also worth noting that the hourly price for these VMs includes the use of SSIS; you don't need to buy a SQL Server license.

Conclusion

Data integration is a critical function in many on-premises datacenters. 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:

- 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.

- 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.