2018

Azure Data Factory: SSIS in the Cloud

THE CASE FOR LIFT AND SHIFT

JOSHUHA OWEN

PRAGMATIC WORKS

Technical Reviewers:
Sandy Winarko, Microsoft & Mark Kromer, Microsoft

Contents

The Modern Data Warehouse

Today's landscape for data warehousing is a rapidly evolving space and most of that evolution is happening in the cloud. The cloud has long offered ways to scale analytic workloads for data warehousing with services like Azure SQL Data Warehouse. However, this only solves one part of the data warehousing workload and as the complexity and velocity of data ingestion grow, there are tools needed to be able to scale with the ingestion and orchestration of data. Azure offers the Azure Data Factory service to be able to handle this task. This service is the data orchestration tool of choice that can handle both the constantly shifting cloud data sources and terabytes of flat files both structured and unstructured. It can also spin up other Azure resources such as HDInsight for your Spark jobs or Data Lake Analytics for your U-SQL jobs.

Traditional data warehouses have long been a mainstay, but with the increasing amount of data sources and processing demands, ETL tools such as SQL Server Integration Services (SSIS) need the scalability that the cloud can offer. While you may have heard of Azure Data Factory and thought the only way to use this for data orchestration was to rewrite all your SSIS packages, it now has the ability to run your SSIS packages in managed Azure-SSIS Integration Runtimes (IRs), so you can scale the processing to your growing needs. In this eBook we will go over why you would want to migrate your existing SSIS workloads to Azure Data Factory and address common considerations and concerns. We'll then walk you through the technical details of creating an Azure-SSIS IR and then show you how to upload, execute, and monitor your packages through Azure Data Factory using the tools you are probably are familiar with like SQL Server Management Studio (SSMS).

The Case for a Lift and Shift to Azure Data Factory

The term for migrating your existing workloads to Azure with minimal effort has been typically called a "lift and shift." Whenever I hear this phrase I like to imagine the large Chinook style transport helicopters using a hoist, picking up my server room and dropping it off somewhere else never to be seen again. While the visual is an amusing one, it's not too far from the truth. In the case of migrating your SSIS packages to Azure Data Factory, the effort is not complicated, and as you'll find out later, you get to use all the same tools you are used to using when executing and working with your SSIS packages.

So now that we know what a "lift and shift" is, what are some of the scenarios that make this a good idea?

Perhaps the vast majority of your data sources and destinations are already in the cloud, so by moving your packages to the cloud, you can make the data transfer much faster. Azure probably lives a lot closer to your existing cloud data sources and has better bandwidth to cloud vendors than your on-premises SSIS Server.

Maybe you are already utilizing a mostly cloud-first approach to data warehousing using Azure SQL Data Warehouse, Azure Analysis Services, and Power BI, but your ETL, which contains your business logic, is still local. This could be due to the development

time to refactor that logic to use other technologies like Spark, or even native ADF activities. Being able to migrate the packages with no changes in this scenario could be an easy win.

You could be due for a SQL Server upgrade including SSIS and looking at moving to the cloud rather than performing an in-place upgrade. Perhaps you just don't want to manage OS patches, SQL Server patches, and worry about hardware failures. While the Azure-SSIS IR does run on Virtual Machines (VMs) in the backend, you don't have to manage the day-to-day operations of these VMs and can scale them up and down or add more nodes as needed.

Whatever the reason, the ability for Azure Data Factory to utilize the Azure-SSIS IR to run your existing packages in the cloud, with scalable processing nodes, is a great way to start that cloud-first approach your organization may be considering.

Another potential scenario is you could lift and shift to your own VMs running in Azure, but then you do still have to manage the infrastructure, hardware specs, patches, SQL upgrades, etc.; and while SQL Server 2017 running on those virtual machines would allow the ability to use the master/worker node (scale-out) feature, it is something you need to manage yourself and not quite as easy as the scaling offered in Azure Data Factory.

The bottom line is that the Azure SSIS IR extrapolates away almost all of that infrastructure management while still providing the ability to scale when you need to.

Common Considerations and Concerns

Before we address the processing of actually preparing and migrating your SSIS packages to Azure Data Factory, let's discuss some of the common considerations and concerns that customers typically have when performing an SSIS migration. These areas typically cover topics such as security, tools, pricing, networking infrastructure, and custom SSIS components.

Security

Azure Data Factory itself does not store any data except for the credentials it needs to access cloud data stores, and these credentials are encrypted using certificates. Data movement in Azure Data Factory has been certified for several compliances, including HIPPA, HITECH, ISO 27001/27018, and more. For more details on the certifications of Azure Data Factory visit https://docs.microsoft.com/en-us/azure/data-factory/data-movement-security-considerations.

For data in transit into Azure services, Azure uses a secure channel vita HTTPS and TLS over TCP to prevent man-in-the middle attacks. With IPSec VPN or the Azure ExpressRoute option, you can further secure the data communication while in transit to Azure.

For more information about ports used to communicate with Azure services, please see the link above.

If your SSIS destinations are Azure services such as Azure SQL Database or Azure SQL Data Warehouse, these services support the use of Transparent Data Encryption to protect your data at rest. Azure Data Lake Store and Azure Blob storage also support encryption protocols to protect the data at rest.

A good resource to review all of Azure's security options, compliances, and privacy features is the Azure Trust Center. The following link will take you to the Trust Center landing page: https://www.microsoft.com/en-us/trustcenter.

Microsoft makes security of Azure and data within Azure a top priority and continues to invest over $1 billion a year on cyber security.

Tools

When working with any new product, you may be worried about what new tools and skill sets that you may need to learn. With the Azure-SSIS IR you'll get to use the tools you are familiar with to manage, report, and execute your package. In our setup chapters later in this eBook you'll find that we use SSMS to upload and manage our packages. Since the Azure-SSIS IR uses an Azure SQL Database/Managed Instance to host Integration Service Catalog (SSISDB), you'll find that it functions the exact same way from SSMS that you are used to. You can still use the built-in reports to view package executions and messages, and you still have the same power to use environments, variables, and configurations to manage your packages. The only new change is setting up Azure Data Factory and the Azure-SSIS IR, but these are simple web-UI driven tasks and only need to be done once.

Pricing

Since pricing on Azure services can sometimes change (usually for the better) here is a link to the pricing page for Azure Data Factory: https://azure.microsoft.com/en-us/pricing/details/data-factory/.

Note that for normal Azure Data Factory operations you are usually charged for the data movement, the number of activities you run, and for whether a pipeline is active or not.

However, since the Azure-SSIS IR is managing and setting up VMs in the background, you are billed on a per hour cost for those VMs just like any other VM in the Azure environment. You don't have to worry about a SQL Server license as that cost gets added on a per hour basis. If your organization has a Software Assurance agreement with Azure Hybrid Benefit (AHB), you can apply your SQL Server license to waive the per hour cost of the SQL Server license and only pay the virtual machine management cost.

Also note that the costs discussed above are only for the Azure Data Factory components. Any Azure storage/database, such as Azure SQL Database/Managed

Instance to host SSISDB, or other services you access, such as HDInsight, are billed for those services with the models appropriate to the service.

Networking

Depending on your data access needs, networking setup and infrastructure are areas you should make sure to address. If your current data sources already exist in Azure (such as Azure SQL databases) or are WAN addressable in SSIS such as with the OData source, then Azure Data Factory will work right away as long as you have enabled services like Azure SQL Database to communicate with other Azure services.

If you need your Azure-SSIS IR to continue to pull from on-premises sources from Azure, then you need to set up an Azure VNET connected to your on-premises network via VPN gateway or ExpressRoute. Here are links for VNet and ExpressRoute info:

https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

https://docs.microsoft.com/en-us/azure/expressroute/expressroute-introduction

If you do have on-premises data but do not want to setup a VNet to enable pulling from Azure, another potential option is to set up a push operation of your on-premises data to an Azure location. The most suitable end-points for most traditional data warehouse loads would be Azure SQL Database; or for flat files in either Azure Blob Storage or Azure Data Lake Store. Then your SSIS packages would be addressing cloud locations, but this does require some effort to change your package to potentially use a new data source, as well as writing the process to push the data up to Azure.

Custom/Third Party SSIS components

The Azure SSIS runtime currently supports all the standard out-of-the-box components that ship with SQL Server 2017, as well as the Azure SSIS Feature Pack. This means for most typical workloads, your packages should function the same after uploading them to your Azure SSISDB. If you are using custom/third-party components, you can also install them on Azure-SSIS IR, see https://docs.microsoft.com/en-us/azure/data-factory/how-to-configure-azure-ssis-ir-custom-setup.

The Process of a Lift and Shift

Now we are ready to discuss the process of migrating your SSIS packages to Azure Data Factory. You'll find a walkthrough of the steps in the next three chapters.

At a high level we first need to use Azure Data Factory to provision our Azure-SSIS IR. We'll be showing you how to do this via the web portal, but this could be done using the various client SDKs using languages like C# or PowerShell. As part of creating the Azure-SSIS IR, we'll be creating the SSISDB in an Azure SQL Database to host our Integration Services Catalog. We'll then show you how to upload your packages via the

ISPAC deployment method, and then execute them using the same tools you are already familiar with: SSMS. Finally, we'll show how you can monitor your SSIS package executions as well as use Azure Data Factory to monitor and scale the Azure-SSIS IR.

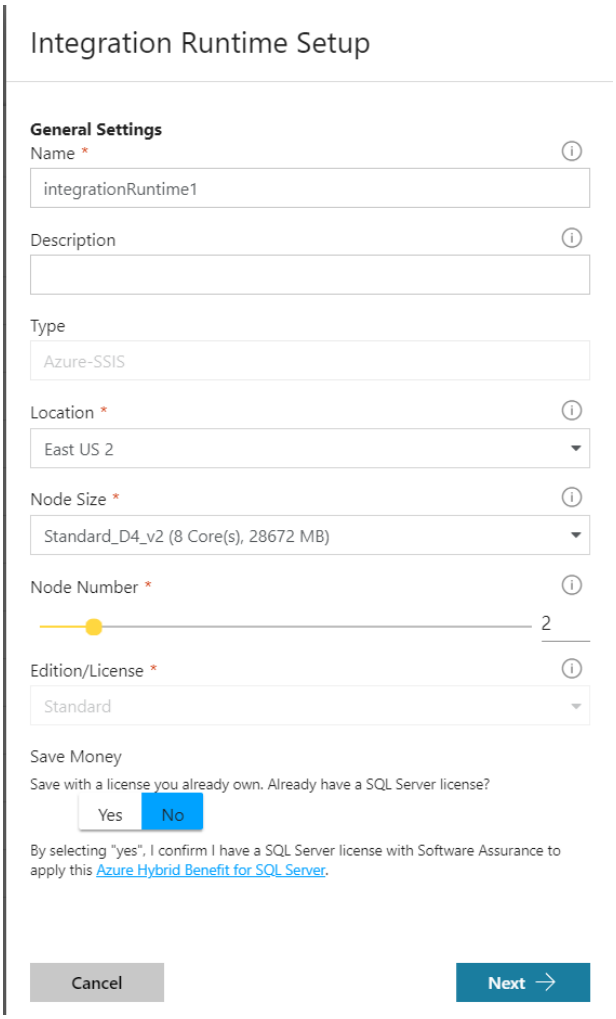
By the end of these tutorials you should have a good feel of how easy the process can be. The only truly new process will be creating the Azure-SSIS IR. The rest of the steps of managing and executing your SSIS packages should closely resemble how you are currently working with your on-premises SQL Server.

Creating an Azure-SSIS IR

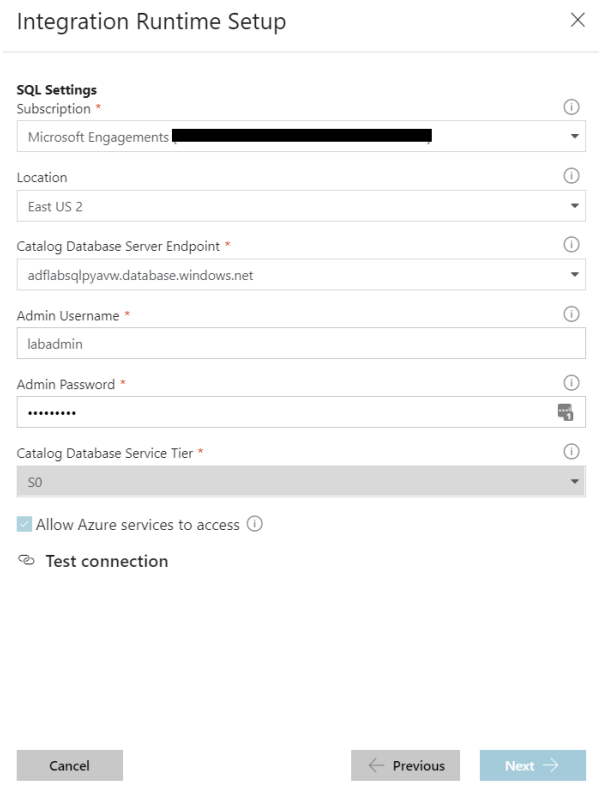
Our first step in migrating our SSIS workload to Azure is to create the Azure-SSIS IR. We create this runtime using the Azure Data Factory interface to allow our packages to have scalable compute resources to execute our SSIS packages.

To start, we are going to navigate to the Azure portal and to your Azure Data Factory instance. We'll click on **Author and Monitor** to get to the main Azure Data Factory landing page.

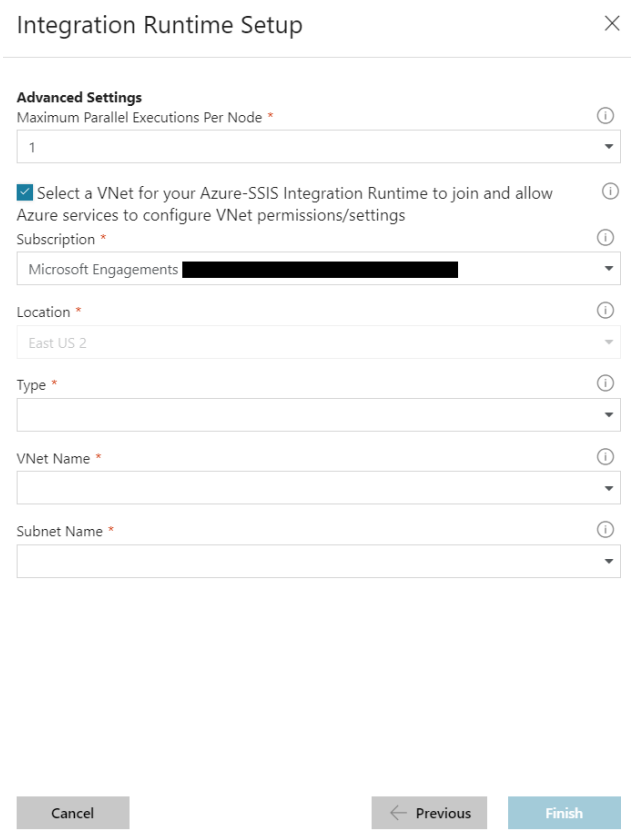
From here we should see a **Configure SSIS IR** icon, which we will click to get to the configuration panel for setting up our runtime.



Go ahead and give your SSIS IR a name and choose a location preferably in the same region where you expect your Azure SQL DB and other sources to exist. As mentioned earlier, the Azure-SSIS IR is running on VMs that Azure configures and manages for you. For the node size property, we now get to choose the resources the VM has available for each node. Choose an appropriate VM based on the amount of CPU cores and RAM you expect to use. Do note that you have the flexibility to scale up and down as needed. Next, choose the number of nodes you want to start with. Finally, if you have a software assurance license that includes the AHB option you can choose Yes on the SQL Server license to save cost per hour with the VMs. See the pricing information in the Common Considerations chapter for more information.



Click **Next** to configure SQL settings for Azure to create and manage SSISDB. This is required and needs to be either an Azure SQL DB or Azure SQL Managed Instance. This is where the SSIS catalog will exist as well as the SSISDB. You will be able to run all the reports and queries that come built-in to SSIS to see package executions, messages, etc. While the default option for the Catalog Database Service Tier is Basic, for anything other than exploration I would recommend S0 or above to provide adequate performance when querying your SSISDB. You can click **Test Connection** to verify connectivity to your Azure SQL database.



Click **Next** to get to the final setup screen. Here you can configure how many executions are spread per node. As a rule, I would not configure this to be more than the number of cores you assigned to your VM. If you have an Azure VNet configured to be able to access on-premises resources, you can also configure that now.

Finally, click **Finish** to deploy your Azure-SSIS IR. Do note that this does provision your VMs behind the scenes and can take a little time. It's typical for this process to take 15-30 minutes before the Azure-SSIS IR is available. Once this is complete we are ready to move and start uploading our packages!

Managing your SSIS Packages in Azure

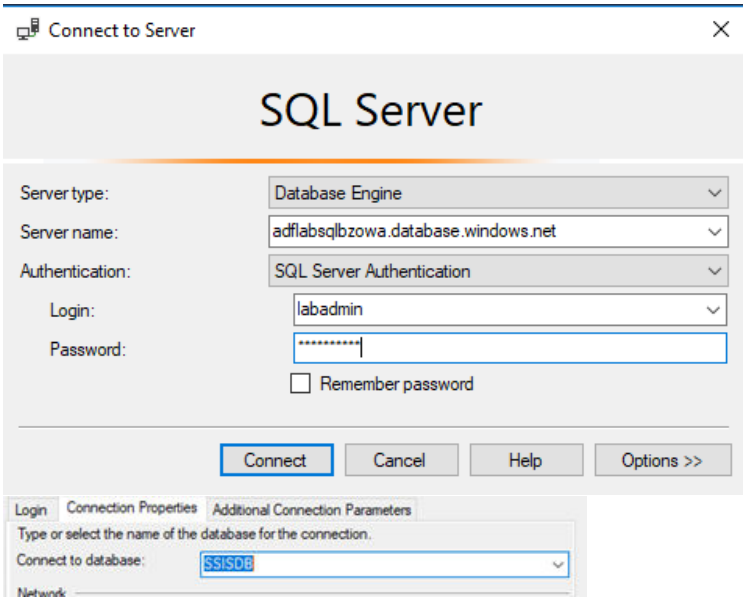
In the previous chapter we setup the Azure-SSIS IR. This is a prerequisite before we can upload our SSIS packages to our Azure database. As part of the creation of the Azure-SSIS IR, we specified an Azure SQL Database or Azure SQL Managed instance. This server location and database name will be needed for us to upload and manage our SSIS packages.

The great news is that we can use the tools we are familiar with to upload and manage our SSIS packages. For this chapter, we will be walking through how to use SSMS to upload your SSIS packages. The first thing to note is that we will be deploying using the ISPAC file. Currently SSIS in Azure Data Factory only supports using the ISPAC file to deploy rather than uploading individual DTSX files. When working with SSIS packages

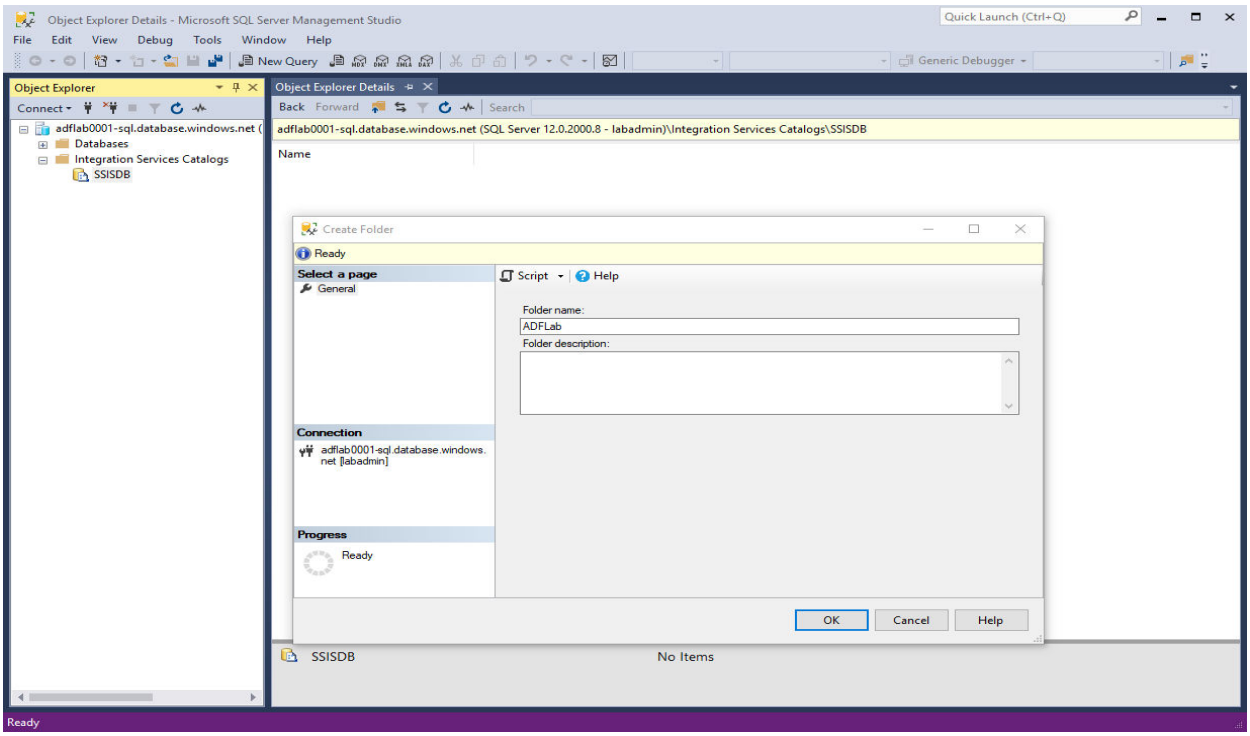
in Visual Studio Data Tools in Project Mode, this ISPAC file is automatically generated when you build your solution, and the default location for the ISPAC file will be in your solution's BIN folder.

Before we start, if you are using the Azure SSIS Feature Pack, please make sure this is installed on the computer that is deploying the ISPAC file.

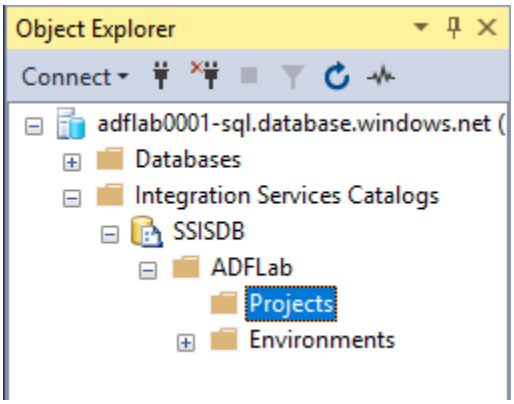
In SSMS we are going to click on the **File Menu** and choose **Connect Object Explorer**.



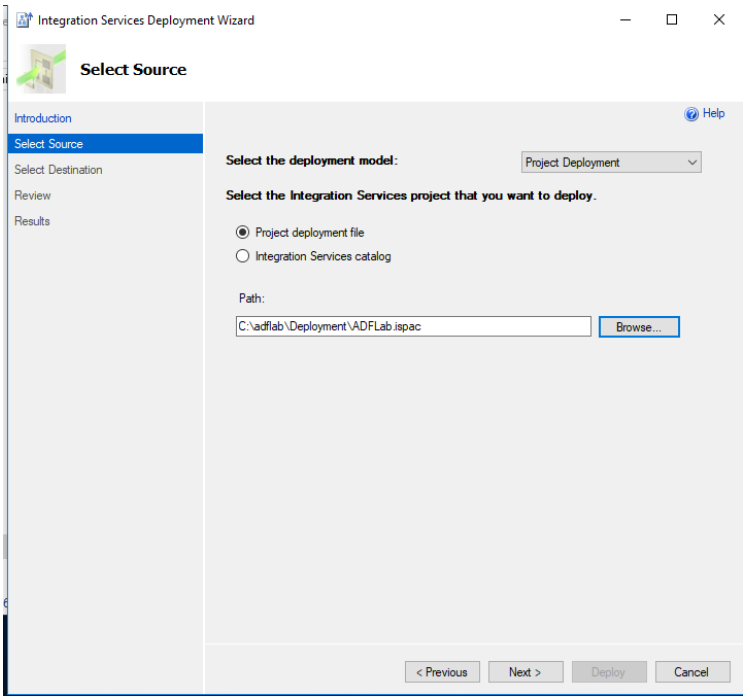
Once you do that, then put in the Azure SQL server endpoint that hosts your SSISDB. Under the **Connection Properties** tab, and then under C**onnect to Database,** go ahead and fill in SSISDB. If you don't do this the Integration Services Catalog won't appear in SSMS.



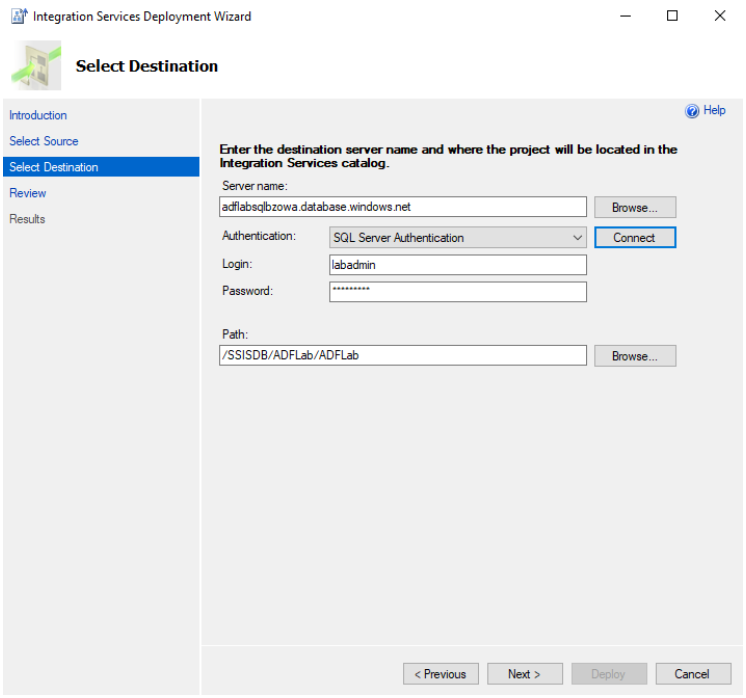
You should now see the Integration Services Catalog in your Azure SQL Database. Right-click the **SSISDB** in the Catalog and create a folder to house your deployed SSIS project.



Navigate to the **Projects** subfolder and right click and select **Deploy Projects**.



Click **Next** if you get the Introduction page in the wizard to get to the **Source** step. Here browse to your ISPAC file and then click **Next**.



Enter your Azure SQL Database server endpoint and credentials and verify the path is in the folder you want the SSIS packages to be deployed to. Click **Deploy** to review your results and click **Deploy** again to upload your SSIS packages to your project folder.

Once your ISPAC file is deployed your SSIS packages should now be available to browse via the Integration Services Catalog on the SSISDB we specified.

From here you have all the same tools available, including the ability to specify environment variables, override connections strings, and access to the built-in reports for monitoring your SSIS packages. Go ahead and make any configuration changes you want now before we show how we can execute and monitor our SSIS packages.

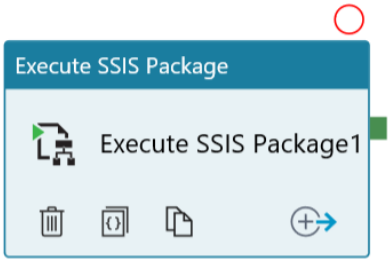
Executing and Monitoring your SSIS Packages

In the previous chapter we showed you how to upload your SSIS packages to the SSISDB associated with the Azure-SSIS IR. Now we execute those packages to utilize this Azure compute resource.

Since this Integration Service Catalogs works the same as your on-premises SQL Server we can execute the packages manually from the Catalog with SQL Server Management Studio. It may be a good idea to test out if everything is communicating correctly with your sources and destination before scheduling your packages.

Azure Data Factory "Execute SSIS Package" Activity

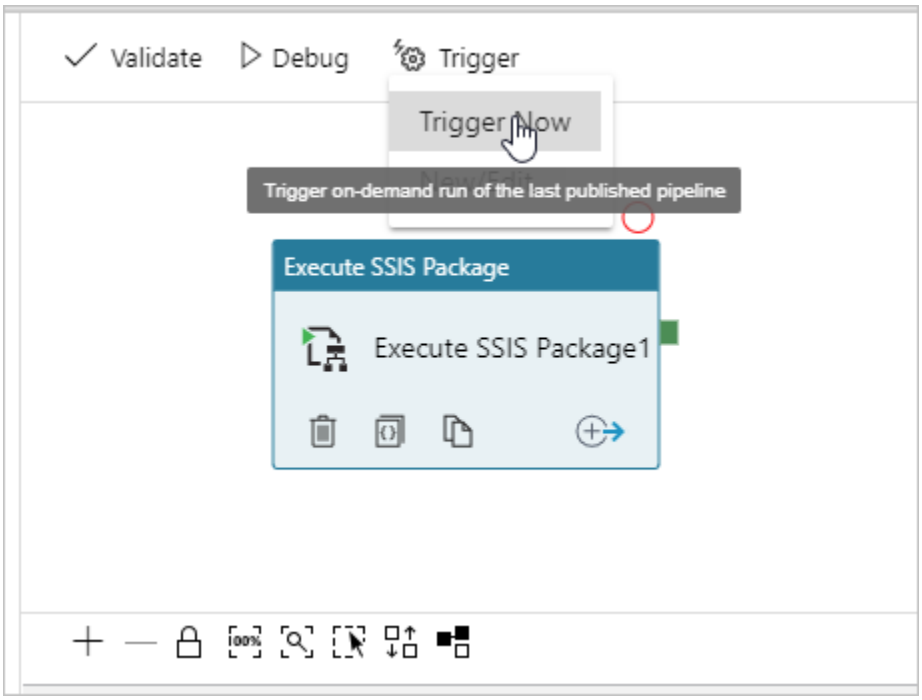
The easiest way to schedule, monitor and manage SSIS package execution in the Cloud is natively within an Azure Data Factory pipeline: https://docs.microsoft.com/en-us/azure/data-factory/how-to-invoke-ssis-package-ssis-activity.



You can start by creating a new ADF Pipeline and add the Execute SSIS Package activity. In the General properties at the bottom of the window, give a name to your new SSIS activity. Tab over to Settings and pick the Azure-SSIS IR that you created above. Pick the logging level for package executions and then provide the package path to the SSIS package to execute. Lastly, set any catalog execution environments that you wish to utilize in the package execution. On the Parameters tab you will see settings that can be parameterized in the activity, which follows normal ADF parameterization facilities: https://docs.microsoft.com/en-us/azure/data-factory/control-flow-expression-language-functions.

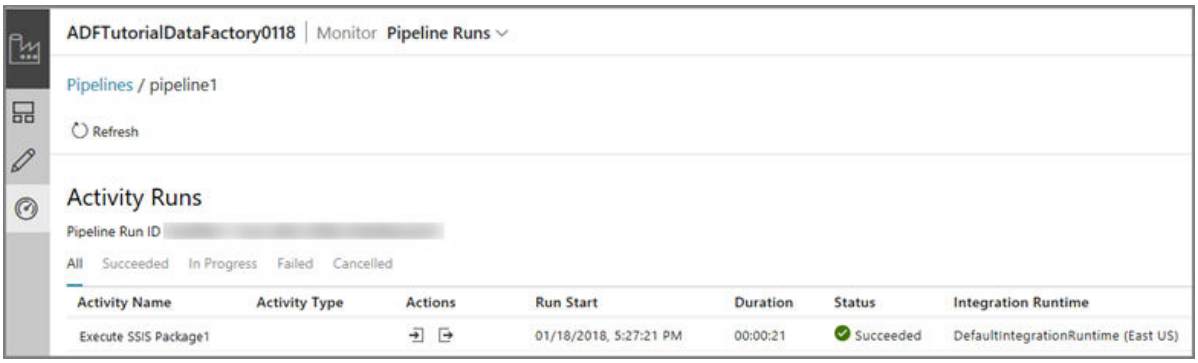
You can now continue to build out your data pipeline in ADF by connecting SSIS with other activities, or if you wish to test just the SSIS package execution, you can use the Debug button feature in ADF to test the SSIS execution in-place while designing the pipeline. Once you are satisfied with the pipeline state, you can trigger a live execution

of the pipeline with the SSIS package execution using the Trigger | Trigger Now option (see screenshot below) or set the recurring schedule for your SSIS package similar to what you may have done previously using SQL Server Agent jobs.

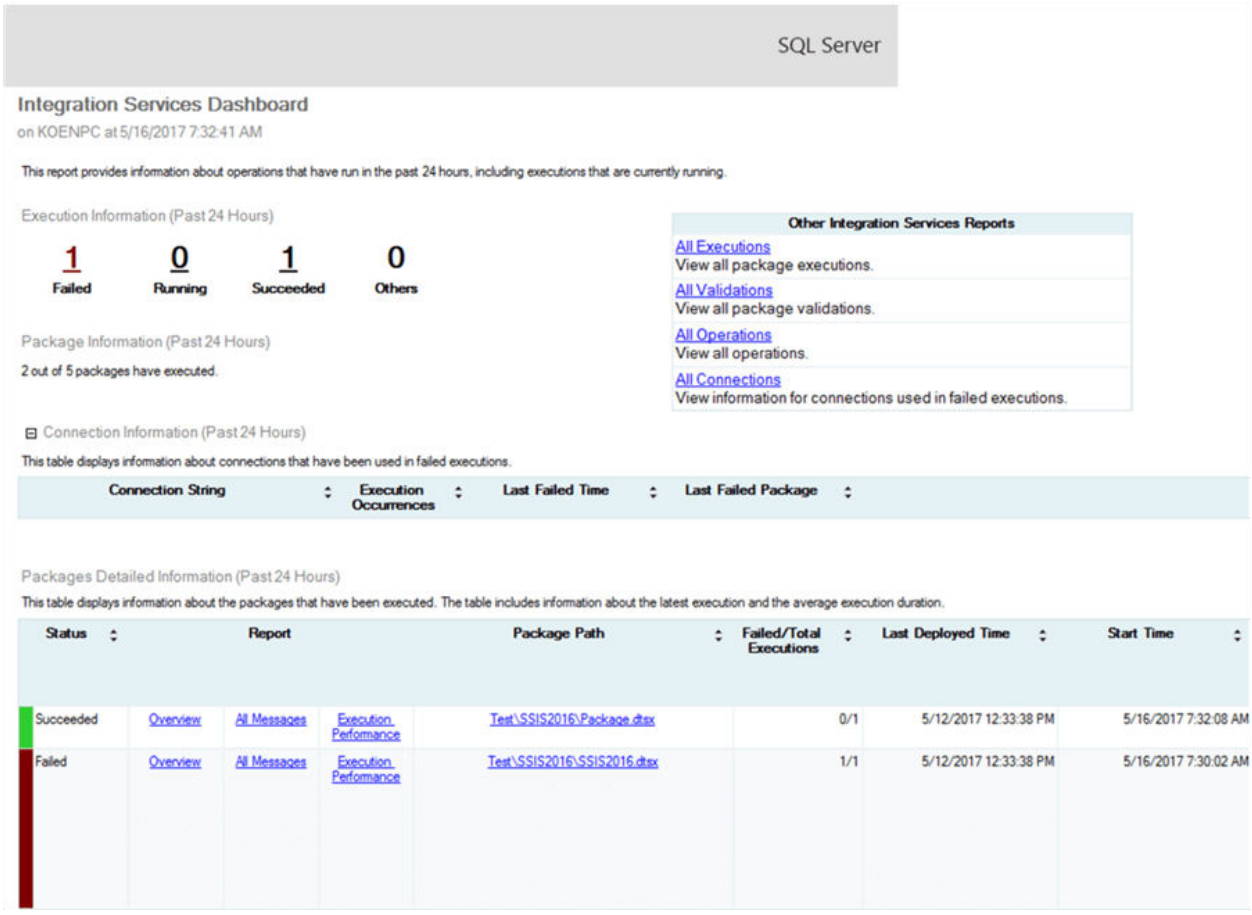


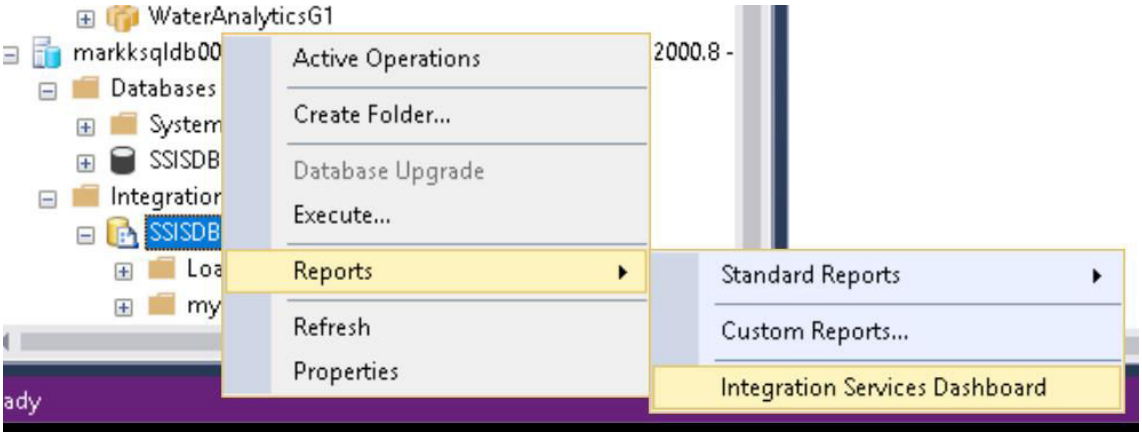
Details on using the ADF pipeline wall-clock scheduler are here: https://docs.microsoft.com/en-us/azure/data-factory/how-to-create-schedule-trigger.

Once you've operationalized your pipeline, you should monitor the health of your pipeline executions in the ADF monitoring view https://docs.microsoft.com/en-us/azure/data-factory/monitor-visually:

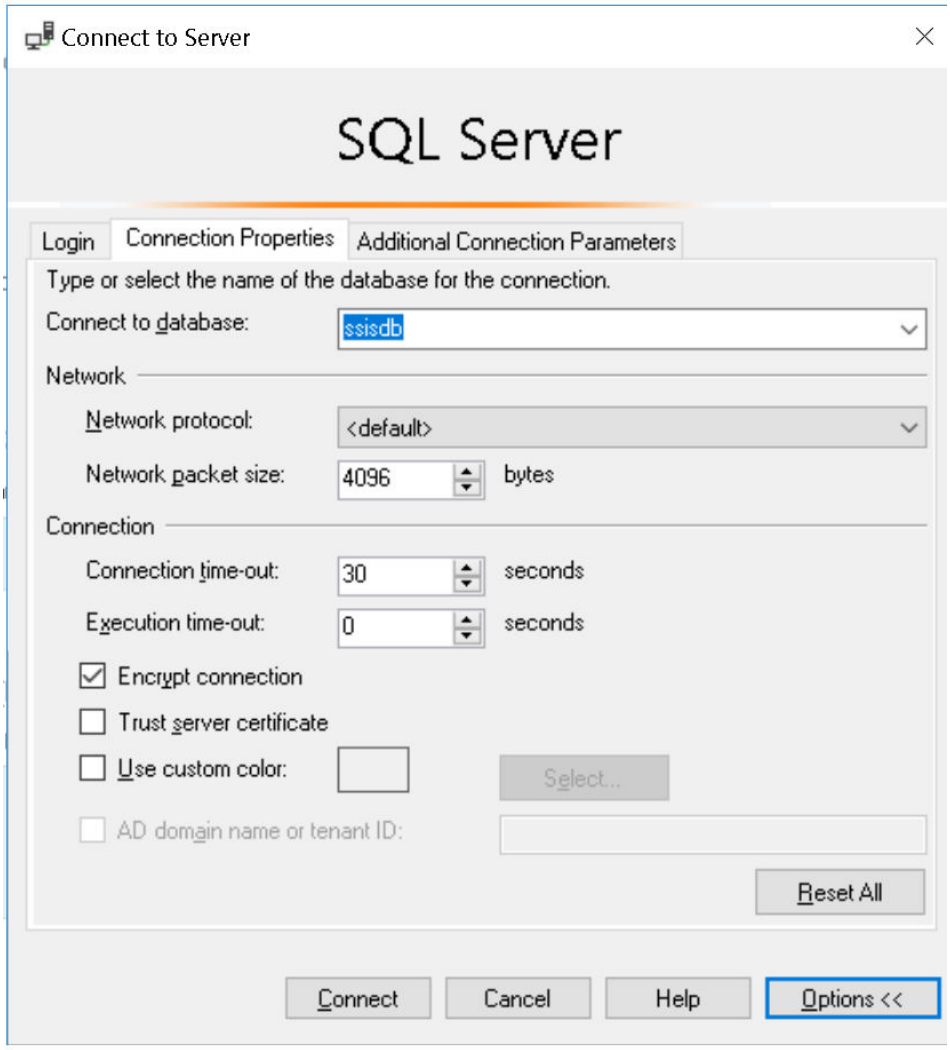


Additionally, the SSIS executions will log to the SSISDB which can be viewed from the SSMS dashboard:





NOTE: Remember to set your “Connect to Database” under Connection Properties to “SSISDB” in SQL Server Management Studio (SSMS) in order to see the “Integration Services Dashboard” link under “Integration Services Catalogs”.



Azure Data Factory Stored Procedure Method

We can also execute our packages with the built-in SSISDB stored procedures. While this option has been available since SSIS 2005 it may be new to you. To do this we call the catalog.create_execution, catalog.set_execution_parameter_value, and the catalog.start_execution stored procedures. An example for a DTSX package called Module2.dtsx would be:

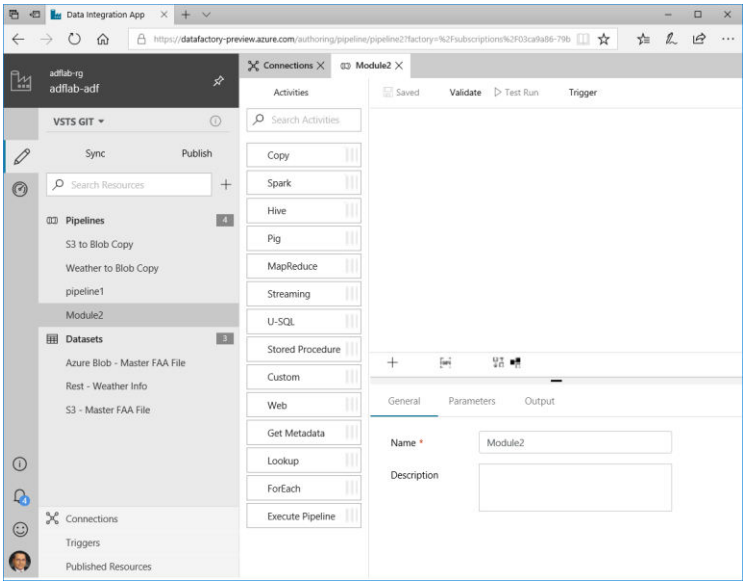
```
DECLARE @return_value INT, @exe_id BIGINT, @err_msg NVARCHAR(150)

EXEC @return_value=[SSISDB].[catalog].[create_execution] @folder_name=N'ADFLab',
@project_name=N'ADFLab', @package_name=N'Module2.dtsx', @use32bitruntime=0,
@runinscaleout=1, @useanyworker=1, @execution_id=@exe_id OUTPUT

EXEC [SSISDB].[catalog].[set_execution_parameter_value] @exe_id, @object_type=50,
@parameter_name=N'SYNCHRONIZED', @parameter_value=1

EXEC [SSISDB].[catalog].[start_execution] @execution_id=@exe_id, @retry_count=0
```

```
IF(SELECT [status] FROM [SSISDB].[catalog].[executions] WHERE execution_id=@exe_id)<>7
BEGIN SET @err_msg=N'Your package execution did not succeed for execution ID: ' +
CAST(@exe_id AS NVARCHAR(20)) RAISERROR(@err_msg,15,1) END
```

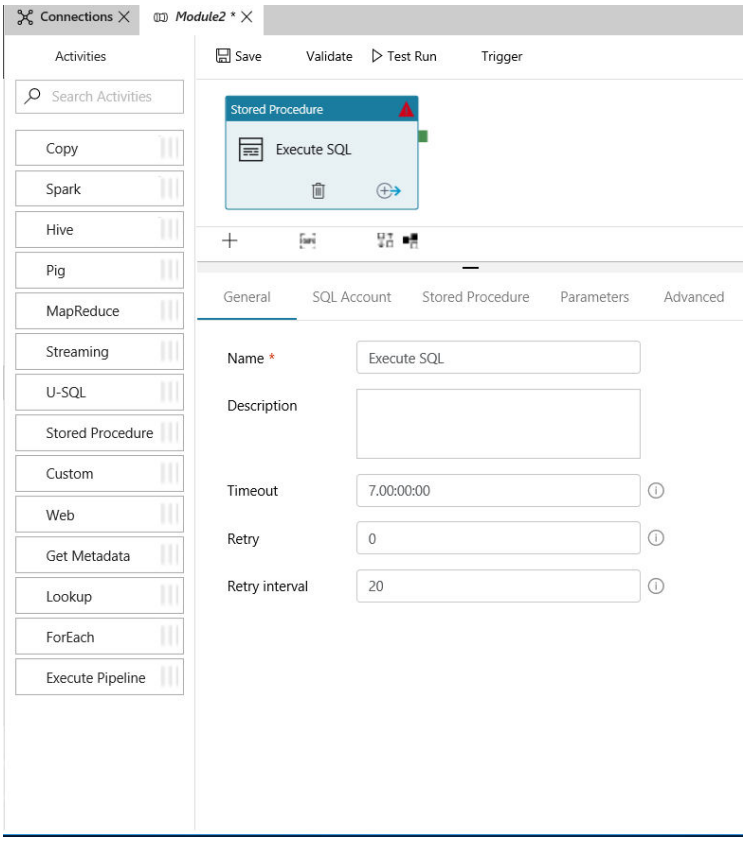
The stored procedure method is interesting, as this allows us to use other tools that can call stored procedures in Azure to also call our SSIS packages. One great example of this is, of course, Azure Data Factory. Azure Data Factory has an activity that can execute stored procedures on SQL Servers. And since Azure Data Factory pipelines can have scheduled triggers, this could be one method you could use to schedule and monitor your SSIS packages. This method works great if you plan to have native Azure Data Factory pipelines being scheduled along with your Azure SSIS workloads.

Let's quickly walk through setting up a SQL Activity in Azure Data Factory and setting up a trigger.

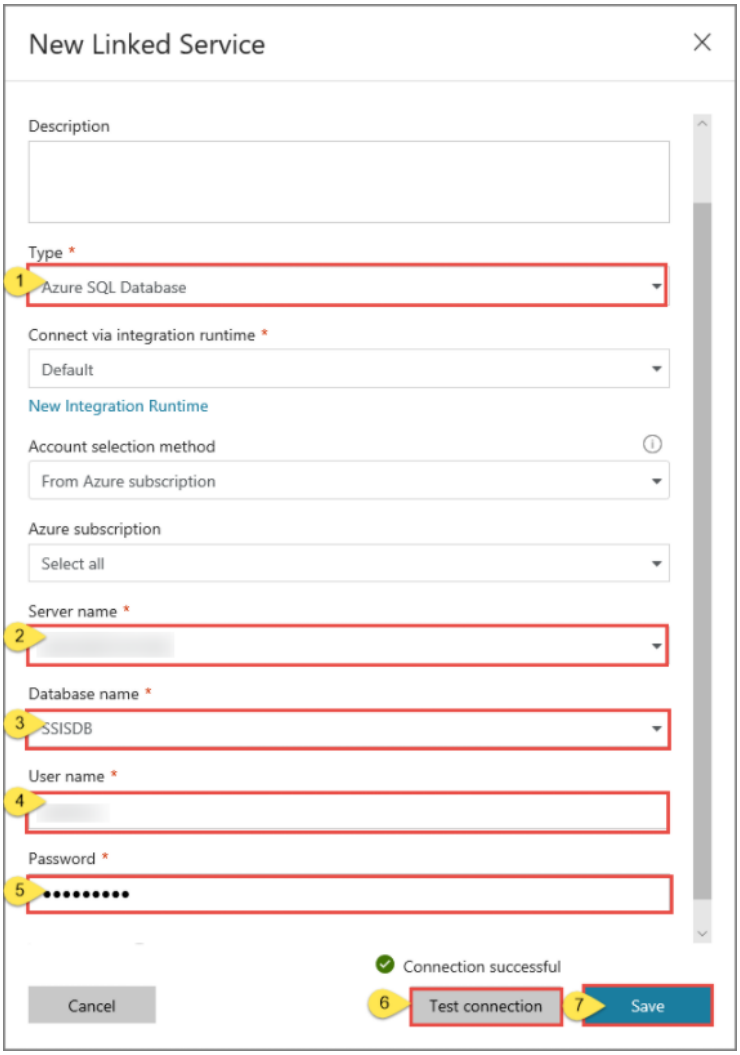
Navigate to your Azure Data Factory on the Azure Portal. Click on **Author** and **Monitor** to get to the Azure Data Factory landing page. Click the p**encil icon** to get to the **Pipeline Editor GUI**.



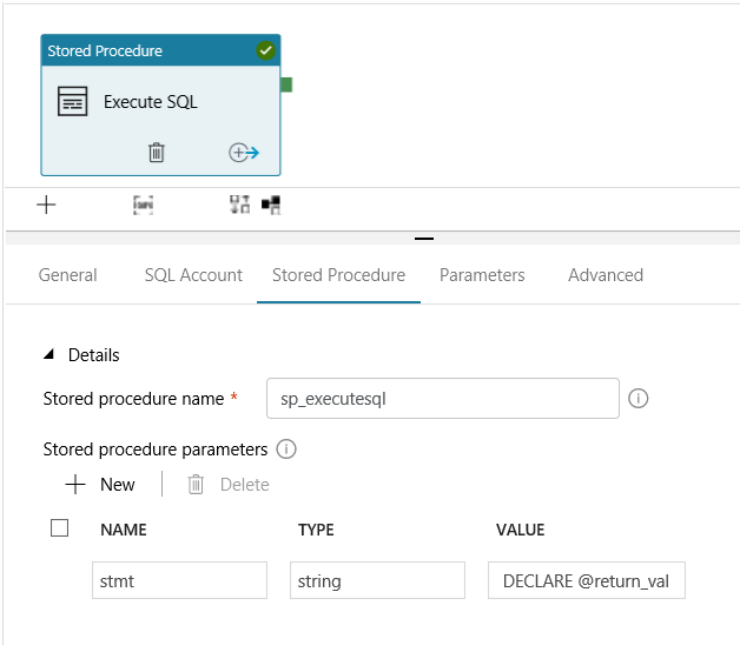
Go ahead and name your Pipeline.



Drag the Stored Procedure activity (under General) to the design panel and name it **Execute SQL**.



Under the **SQL Account** tab go ahead and click the **+New** button to configure a new SQL Server connection. Fill out the **Linked Service** to connect to the Azure SQL Database where your SSISDB was deployed to – Select the **Default** Azure IR for the connection. Click **Save** and navigate back to your pipeline.

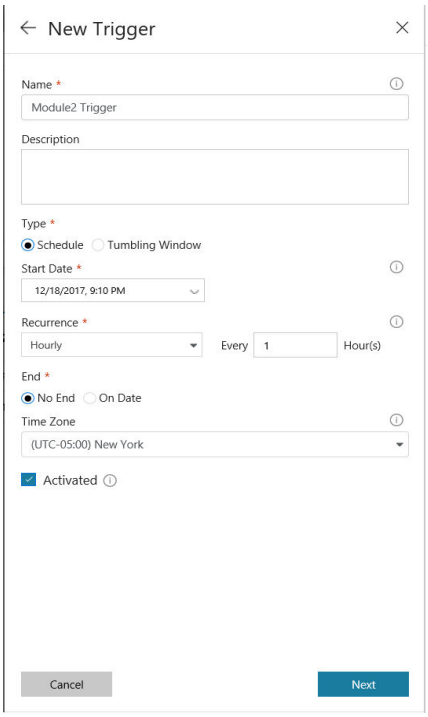


Under the **Stored Procedure** tab of our Execute SQL activity we will be using the sp_executesql stored procedure to call multiple lines of SQL. Go ahead and add a new stored procedure parameter called stmt, type of string, and for value paste in the following:

DECLARE @return_value INT, @exe_id BIGINT, @err_msg NVARCHAR(150) EXEC @return_value=[SSISDB].[catalog].[create_execution] @folder_name=N'ADFLab', @project_name=N'ADFLab', @package_name=N'Module2.dtsx', @use32bitruntime=0, @runinscaleout=1, @useanyworker=1, @execution_id=@exe_id OUTPUT EXEC [SSISDB].[catalog].[set_execution_parameter_value] @exe_id, @object_type=50, @parameter_name=N'SYNCHRONIZED', @parameter_value=1 EXEC [SSISDB].[catalog].[start_execution] @execution_id=@exe_id, @retry_count=0 IF(SELECT [status] FROM [SSISDB].[catalog].[executions] WHERE execution_id=@exe_id)<>7 BEGIN SET @err_msg=N'Your package execution did not succeed for execution ID: ' + CAST(@exe_id AS NVARCHAR(20)) RAISERROR(@err_msg,15,1) END

You can now save the pipeline.

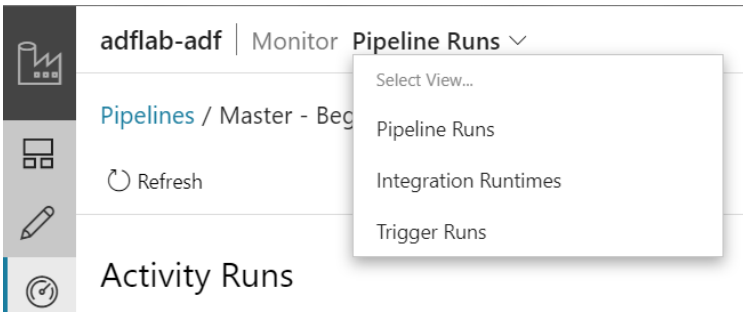
Now that we have a wrapper pipeline and activity to execute our SSIS package we can use the Azure Data Factory built-in triggers to schedule. At the top of your pipeline click the **Trigger** menu icon and click **New/Edit**.



Go ahead and configure the schedule to a time/frequency that makes sense for your SSIS package.

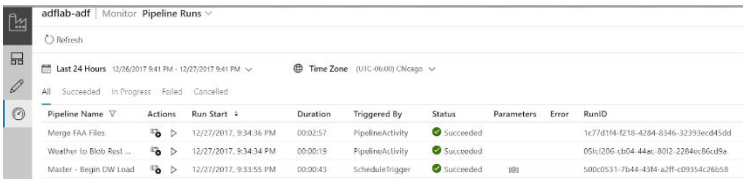
Azure Data Factory Pipeline Monitoring

If you use the Azure Data Factory pipeline wrapper and trigger method to execute your packages you can also use Azure Data Factory to monitor the status of those executions.



From the Azure Data Factory landing page click on the **Monitor** icon (gauge) to get to the **Monitoring Dashboard**. From the **Monitor** menu dropdown at the top select **Trigger Runs**.

You should now be at the Trigger Runs activity log which is essentially a job overview status. You can see the status of your triggers and the actual trigger time when it was kicked off.

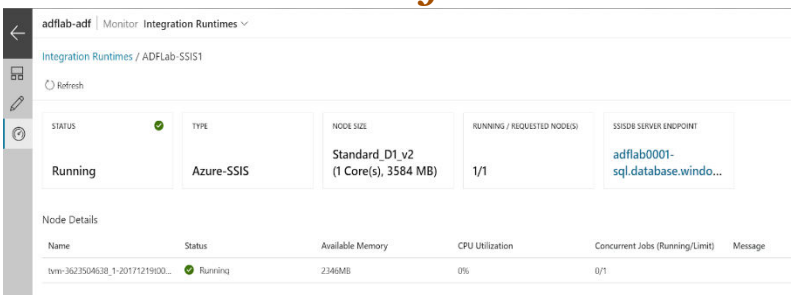


At the top of the **Monitor** menu icon select **Pipeline Runs** to see the actual execution log of the pipelines themselves. From here you can drill down into the individual activities of a pipeline run and see the statuses as well. In our case we had a single stored procedure activity to run our SSIS package, but if you chained those together to run multiple SSIS packages, you could drill-down into the activities to see the status of each activity execution.

Azure SQL Managed Instance

If you host your SSIDB in Azure SQL Managed Instance, you should also have the capabilities to use Managed Instance Agent to schedule your SSIS packages to execute. This is probably the easiest method to transition to as it most likely resembles how you are executing your packages on your on-premises SQL Server.

Azure-SSIS IR Monitoring

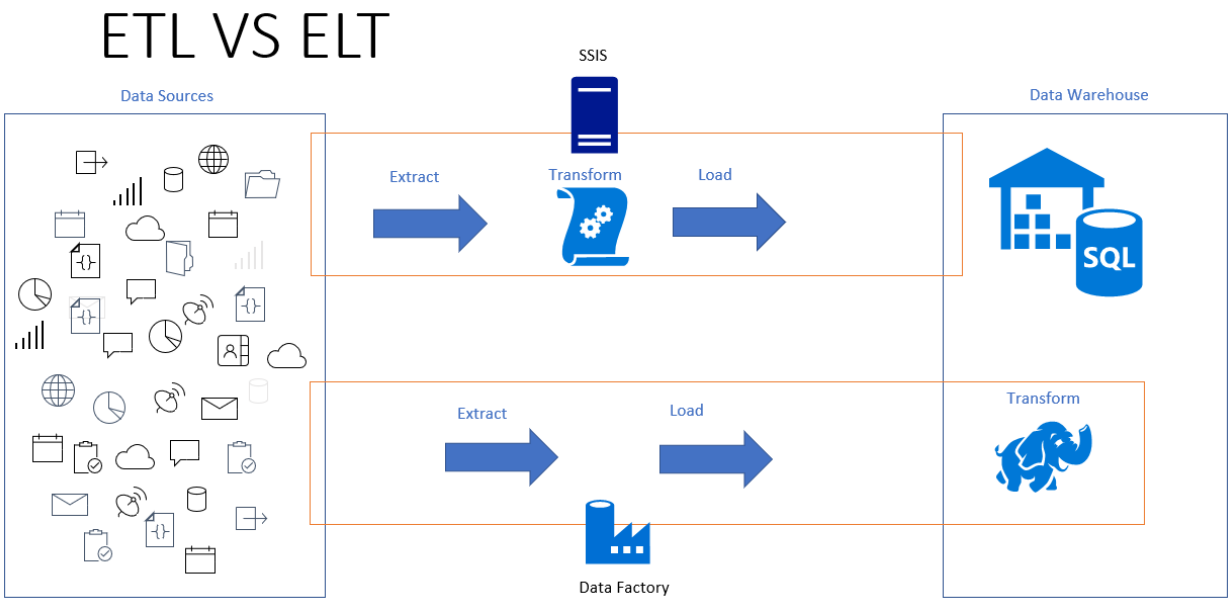


Finally, the above methods have shown how to execute and monitor your SSIS package executions, but Azure Data Factory also provides the ability to monitor the Azure-SSIS IR itself. This lets you see what kind of resources you assigned to your SSIS IR, as well as what the current utilization of your nodes is. To access this, navigate back to your Azure Data Factory landing page and click the **Monitoring** icon again. Choose the **Runtimes** tab and then click on the Azure-SSIS IR you created. You should see information about your Azure-SSIS IR including Node Size, Available Memory, and CPU Utilization. This is a good place to see the health of your compute resources as well as if you need to potentially scale up your nodes.

Next Steps

Once you have lifted and shifted your current packages and workflow up to the cloud, begin to think about how you can begin to combine/replace pieces of that logic with native ADF activities. If you are already pulling from cloud sources such as Salesforce and your destination is an Azure resource such as Azure SQL Data Warehouse, you'll find that rewriting those pulls can be quick and easy wins since ADF has already written the efficient, scalable cloud-to-cloud linked services for you. Scaling that throughput up

with DTUs can approach speeds of 1 gigabit per second, which is probably much faster than you could perform the same copy out of SSIS.



When migrating to native Azure Data Factory pipelines you may want to start thinking of an Extract – Load – Transform (ELT) pattern as opposed to what you are mostly likely doing with SSIS now, which is an Extract – Transform – Load (ETL). With SSIS operating in memory on an individual server (or multiple nodes with 2017), the transformation is all done in one place. To fully utilize the cloud resources that Azure can provide (such as Spark with HDInsight, U-SQL with Data Lake Analytics, or PolyBase with Azure SQL Data Warehouse), you want to shift your transformation closer to where the data will live. This allows those scalable compute resources to be able to work with the data in massively parallel fashion where it rests rather than in the SSIS style data buffer at a time.