Lift and Shift of SSIS Package to Azure SSIS Runtime

Technologies showcased: Azure SSIS Runtime

Author: Michael L French

Date: 12/15/2017

Table of Contents

[Summary 3](#_Toc501530489)

[Pre-requisites 4](#_Toc501530490)

[Scenario 4](#_Toc501530491)

[Part 1 – Create Azure SSIS Runtime 4](#_Toc501530492)

[Part 2 – Upload SSIS Package to Catalog 9](#_Toc501530493)

[Part 3 – Use SSDT to run manually 18](#_Toc501530494)

[Part 4 – Create Pipeline to call Stored Procedure to do a Trigger Based Scheduled Run 19](#_Toc501530495)

## Summary

This tutorial walks through the process of creating an Azure Data Factory Integration Runtime to schedule the execute a SSIS Package. The package will simulate a typical Data Warehouse Extract, Transform, and Load cycle.



In this lab we will:

* Create Azure SSIS Integration Runtime using Graphical User Interface
* Upload SSIS Package to Integration Services Catalog through SSDT
* Use SSDT to run manually package and monitor run using standard catalog report
* Create a pipeline to call a Stored Procedure to do a trigger based scheduled run of the package

## Pre-requisites

* Azure Subscription with rights to use/deploy Azure services, and X of Azure Credit
* Azure Data Factory
* ADFLab.ipsac
* SQL Server Management Studio
* Azure Databases from Module 1:
  + AirlinePerformance-OLTP
  + AirlinePerformance-ODS
  + AirlinePerformance-Staging
  + AirlinePerformance-DW
  + Azure Blob Storage Container

## Scenario

|  |  |  |
| --- | --- | --- |
| Part 1 – Create Azure SSIS Runtime | | |
| **Scenario** | | |
| This section walks through the steps to use the Azure Data Factory GUI to create new | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| We can connect to the Azure Data Factory as set up in Module 1 called adflab-adf. | 1. Go to Azure Data Factory Portal and choose Configure SSIS Integration Runtime |  |
| When you choose Configure SSIS Integration Runtime the following dialog starts. | 1. Name Service ADFLab-SSIS and click Next |  |
| Then we enter the database connection information. | 1. Configure Azure SQL Settings. |  |
| Adjusting Advanced Settings is not required for this lab. | 1. Configure Advanced Settings. For this lab choose default settings. |  |
| The SSISDB is created by creating the service | 1. Confirm that the Status has changed to from Starting to Running. |  |

|  |  |  |
| --- | --- | --- |
| Part 2 – Upload SSIS Package to Catalog | | |
| **Scenario** | | |
| This section describes the steps to connect to Integration Services Catalog to upload the provided package. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| From MS SQL Server Management Studio, connect to the Integration Services Catalog. | 1. Connect to Azure SQL End Point with Management Studio. Endure that SSISDB is chosen in the Options/Connection Properties. |  |
| Build the structure necessary to deploy a package to the Catalog. | 1. Right Click on the SSISDB in the Integration Services Catalogs to Create folder for ADFLab. |  |
|  | 1. Start the Deploy Package Wizard by right clicking on the Projects in the folder we just created. |  |
| Select the source for the SSIS Project Deployment. | 1. Choose Project Deployment and locate the ADFLab.ipsac file |  |
| Select the Destination Integration Services Catalog | 1. Choose the server that holds the SSISDB as created in Part 1. Remember to use SQL Server Authentication. 2. Press the Connect Button and verify the Path. |  |
|  | 1. Review Validate Results. |  |
| Review configurations and Deploy the SSIS Project | 1. Deploy Project. |  |
| Verify that the Project was deployed successfully. | 1. Verify Deployment Results. |  |
| We will now configure the connection strings for the services that we created in Module 1. The pattern for the name will be as follows <prefix><service><character suffix>. For example: prefixsqlnjezw or prefixstoragenjezw | 1. From the Azure Portal, retrieve the names of the SQL Server and Blob Storage Account |  |
| It is important to configure each connection string for the Entry-point packages and project. | 1. Right click on the ADFLab Project that we just deployed and choose Configure the connection strings for each connection strings. |  |

|  |  |  |
| --- | --- | --- |
| Part 3 – Use SSDT to run manually | | |
| **Scenario** | | |
| Execute the Package manually from Management Studio | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| We will now verify that the package has been deployed properly and executes successfully. | 1. Right Click on the Package and choose Execute |  |
|  | 1. Execute Package |  |
| We can monitor the execution results from MS SQL Server Management Studio. | 1. Monitor Package Execution from Integration Services Dashboard |  |

|  |  |  |
| --- | --- | --- |
| Part 4 – Create Pipeline to call Stored Procedure to do a Trigger Based Scheduled Run | | |
| **Scenario** | | |
| In this section we show how to create an Azure Data Factory Pipeline to schedule the execution of the uplifted SSIS Package | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| Using the GUI, create a pipeline to execute a package from the Integration Services Catalog. | 1. From the Data Integration App, click the “+” to create a new Pipeline |  |
|  | 1. Name the Pipeline Module2 and Save |  |
| Begin to build a simple pipeline with one activity. | 1. Drag a Stored Procedure Activity to the Design Surface and name it Execute SQL |  |
| The next set of steps we will walk through the detailed steps to configure the Stored Procedure Activity. | 1. Add New Linked Service named SSISDB Azure Endpoint. Choose type as Azure SQL Database |  |
| A single parameter containing the SQL required to create an execution instance from the Integration Services Catalog. | 1. Configure the stored procedure sp\_executesql by adding a new parameter named stmt of type string with the following value:  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 |  |
| Verify that the Stored Procedure Activity has been configured properly. | 1. Validate Pipeline |  |
| Once the pipeline has passed validation, then test the actual execution to ensure that the Module2.dstx package has executed.  We can monitor the execution results from MS SQL Server Management Studio as in Part 3 using the Integration Services Dashboard. | 1. Execute a Test Run |  |
| Now that we have successfully configured and tested the pipeline, we can begin to create an automated execution schedule.                                       When you click next the following reminder appears. | 1. Add a New Trigger named Module2 Trigger. Use the Recurrence of Hourly.           Note: Please set the start date to some point soon so that we can observe the execution. |  |
| From the Data Integration App, use a two-step process to Sync and Publish our changes. | 1. Sync your changes |  |
| Ensure that no conflicts exist in our code. | 1. Verify the changes and click Sync                                Note: Uncheck Publish changes after sync. |  |
| Once we click publish then the Trigger is live, and the pipeline will begin to execute on the frequency configured. | 1. Publish your Changes |  |
|  | 1. Verify that Triger is Scheduled |  |
| See that our pipeline has executed using the Portal. | 1. Monitor Pipeline Runs from Data Integration App |  |
| After we completed all steps in this module, remember to deactivate our Trigger to stop further executions | 1. Deactivate the Trigger. 2. Sync your changes. 3. Publish your changes. |  |

**IMPORTANT: AVOID INCURRING EXTRA CHARGES BY PAUSING YOUR SUBSCRIPTION RESOURCES**