Lab 03: Rebuilding the Extract and Load with ADF

Technologies showcased: ADF GUI, S3 and Azure Storage, Pipeline Parameters, ADF Branching

Author: Joshuha Owen

Date: 12/7/2017

Table of Contents

[Summary 3](#_Toc502319897)

[Pre-requisites 4](#_Toc502319898)

[Scenario 4](#_Toc502319899)

[Part 1 – Connect ADF to a code repository to begin using the ADF GUI (Optional) 4](#_Toc502319900)

[Part 2 – Setting up the Connections in the ADF GUI 7](#_Toc502319901)

[Part 3 – Setting up the Copy Activity in the ADF GUI 14](#_Toc502319902)

[Part 4 – Setting up branching in the ADF GUI 19](#_Toc502319903)

## Summary

This tutorial walks through creating a pipeline copy activity to copy a file from a S3 storage location to an Azure blob storage container so we can prepare the file to be processed later for transformation.

In this lab we will:

* Show the graphical user interface for creating a pipeline
* Copy CSV file via a Copy Activity
* Creating branching success and failure paths to send an email
* Use parameters to make the pipeline easy to change and more reusable
* Call an Azure Logic app to send an email via a Web Activity

In this lab we will be using the following S3 information to copy CSV files.

[show screenshot of s3 bucket]

[S3 user access key id]

[S3 user access secret]

## Pre-requisites

* Azure Subscription with rights to use/deploy Azure services, and X of Azure credit
* Azure Blob storage container
* Azure Data Factory
* Visual Studio Team Services Git project (optional)
* Azure Logic App for sending emails

## Scenario

|  |  |  |
| --- | --- | --- |
| Part 1 – Connect ADF to a code repository to begin using the ADF GUI (Optional) | | |
| **Scenario** | | |
| One option to be able to sync our code is to connect ADF to a code repository. This section walks through the steps to connect ADF to a Visual Studio Team Services Git project so we can save our code for later re-use. Note that this is not required but a recommended best practice. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| We will be using the Azure Data Factory we created in Lab 01 called adflab-adf. We also will use a VSTS Git (or Github) account to version code in the GUI pipeline tool. This was also setup in Lab01 but we will show how to connect it. | 1. Navigate to the Azure portal within your web browser and navigate to <https://portal.azure.com> 2. Open the Azure Data Factory blade [adflab-adf], pinned from a previous lab but if not navigate to it using the All Resources menu item. | *TO-DO: Replace with GUI Pipeline editor tile exists* |
| !!! GUI tile button doesn’t exist yet!!! | 1. In the Overview blade you should see the following Quick Links: |  |
| !!! GUI tile button doesn’t exist yet!!! | 1. Click the Pipeline Editor button and you should see graphical user interface Overview page. |  |
| We can connect to a code repository before we can start creating pipelines using the GUI in order to version our code. In Lab 01 we setup a VSTS Git project and will link it here. Github can also be used. | 1. Click the Set up repository button to begin connecting this Azure data factory to a code repository. |  |
| We will be filling the settings to correspond with the VSTS Git we created in Lab 01. | 1. The Repository Settings pane will appear on the right. Fill this out to match the Visual Studio Team Service Git project created previously in Lab 01. 2. Click the Finished button when you have verified your settings. |  |

|  |  |  |
| --- | --- | --- |
| Part 2 – Setting up the Connections in the ADF GUI | | |
| **Scenario** | | |
| We now want to use the GUI to create a pipeline to copy the CSV files from S3 to Azure blob storage to be ready for transformation. Our first step is setting up the connections and linked services need for the source and destination. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| We will be using the previously code repository connected to ADF in Part 1. | 1. Click the Create Pipeline button under Let’s Get Started. |  |
|  | 1. You should now see Pipeline GUI editor. |  |
|  | 1. Fill out the pipeline name. We will be using S3 to Blob Copy as our pipeline name. 2. Click the Save Icon on top of the screen. |  |
| We now need to setup our Connections i.e. Linked Services to for both the source (S3) and destination (Azure blob storage) | 1. In the Left Menu click the Connections menu item. 2. Click the +New button under Linked Services. |  |
| Azure S3. | 1. In the right pane you should now see the list of possible Linked Services. 2. Click on the Amazon S3 tile. 3. Click Continue. |  |
| !!! TBD when S3 account is setup!!! | 1. In the right pane you should see the properties to configure the Amazon S3 account link. 2. We will name this linked service AmazonS3-Source and using the Default runtime. Use the following for the Access Key ID [key id] and Secret Access Key [key] 3. Click the Test Connection to verify settings are entered correctly. 4. Click Save. |  |
| Now we need to add destination, Azure storage. | 1. Click the +New button under Linked services. 2. In the right pane you should now see the list of possible Linked Services. 3. Click on the Azure Storage tile. 4. Click Continue. |  |
|  | 1. In the right pane you should see the properties to configure the Azure Storage account link. 2. We will name this linked service AzStorage-Staging and using the Default runtime. For the Account selection method use From Azure subscription and then choose your subscription used for these labs and the storage account created for the Staging area [adflabstaging] 3. Click the Test Connection to verify settings are entered correctly. 4. Click Save. |  |

|  |  |  |
| --- | --- | --- |
| Part 3 – Setting up the Copy Activity in the ADF GUI | | |
| **Scenario** | | |
| We now want to use the GUI to create a Copy Activity in the pipeline to move the files from the S3 source to our Azure storage destination. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| We will be using the previously created pipeline in part 2 called S3 to Blob copy. | 1. Click the S3 to Blob Copy Pipeline in the left menu to return to the pipeline GUI. |  |
|  | 1. In the Pipeline GUI, drag the Copy activity to the empty pane above General. 2. Rename the activity to S3 to Azure blob. 3. Click Save. |  |
|  | 1. Click the Source Tab in the Copy Activity GUI. 2. Click the +New button next to Source Dataset. |  |
|  | 1. You should now see the list of source dataset connectors. 2. Choose the Amazon S3 dataset and click Finish. |  |
| We will be using the Linked Service we created earlier. The Master file contains the tailnumber and aircraft model number we will need to link it to the BTS data and the FAA aircraft reference data. The S3 location also contains files with schema information on the master and aircraft reference data. | 1. You should now the connection property information. 2. Name this dataset S3 – Master FAA File 3. Fill out the following information: Linked Service -> AmazonS3-Source, File Path -> Click the Browse button and drill down to the adflab folder and the FAAmaster.txt file, Click the “Sources files contains column names in the first row” checkbox. 4. Click Preview Data to preview the first several data rows. |  |
| Since this is a CSV file all columns will be considered string which is expected. | 1. Click the Schema tab to define the schema for the file. 2. Click the Import Schema to automatically use the file to determine the Schema. 3. Click the Save Icon at the top of the data flow to save the data flow. |  |
|  | 1. Click back on the S3 to Blob Copy pipeline. 2. Click the S3 to Azure Blob copy activity. 3. Click the Sink Tab in the Copy Activity GUI. 4. Click the +New button next to Source Dataset. |  |
|  | 1. You should now see the list of sink dataset connectors. 2. Choose the Azure Blob storage dataset and click Finish. |  |
| We will be using the Azure blob storage linked service we created in the previous section. | 1. Name the data Azure Blob – Master FAA File 2. Fill out the following information: Linked Service -> AzStorage-Staging, File Path -> Click the Browse button and drill down to the input container, append FAAMaster.txt in the input box, Verify the “Sources files contains column names in the first row” checkbox is checked. |  |
|  | 1. Click the Schema tab to define the schema for the file. 2. Click the Import Schema to automatically use the file to determine the Schema. 3. Click the Save Icon at the top of the data flow to save the data flow. |  |
| We now need to map the columns between each file. Since our source and destination file are copies of each other the schema should auto-map. | 1. Click the S3 to Blob Copy pipeline. 2. Click the S3 to Azure Blob copy activity. 3. Click the Mapping tab in the copy activity GUI. 4. Click the Import Schemas button. 5. Click the Save icon. |  |
| Now we can test our copy activity from within the ADF GUI editor. | 1. Click the Run icon at the top menu to test and run our copy activity. |  |

|  |  |  |
| --- | --- | --- |
| Part 4 – Setting up branching in the ADF GUI | | |
| **Scenario** | | |
| We now want to use the GUI to create success and failure branching to send emails. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
| First, we want to setup a parameter to use for the email address to easily change the email per run. | 1. Click the S3 to Block Copy Pipeline in the left menu to return to the pipeline GUI. 2. Click the Parameters tab. 3. Click the +New button and create parameter with the name: recipient, type: string, and value of your email address. |  |
| We will be using the previously created pipeline in part 2 called S3 to Blob copy. We will be creating one branch for success of the copy and another for failure along with web activities to call an Azure Logic App to send emails with the appropriate information. | 1. Click the S3 to Block Copy Pipeline in the left menu to return to the pipeline GUI. 2. In the Pipeline GUI, drag the Web activity to the right of the S3 to Azure Blob copy activity. 3. Name the web activity Send Success Email. 4. Click the Save Icon. |  |
| We now want to link the success branch of the copy activity with the Send Success Email activity | 1. Drag from the green square of the S3 to Azure Blob activity to the Send Success Email activity. 2. Click the Success Email web activity. |  |
| We now are going to fill out the information needed to send the email. Note we use built-in parameter notation to reference values from the pipleline as well our our custom parameter. The body field names match the names setup in the Azure Logic email app to fill out the template.  The header type needs to be set to JSON here for the Azure email logic app. | 1. Click the Settings tab. 2. Fill out the following information: URL: Paste your Azure Logic App URL created in Lab Module 1. Method: Post Body: {"dataFactoryName":"@{pipeline().DataFactory}","successtype":"Succeeded","message":"@{activity('S3 to Azure Blob').output.dataWritten}","pipelineName":"@{pipeline().Pipeline}","recipient":"@{pipeline().parameters.recipient}"} 3. Click the + Add Header button and for the Header: Content-Type Expression: Application/json 4. Click the Save icon. |  |
| Now we will create the failure email branch. This mirrors the previous steps with slight differences in naming and the email body parameters. | 1. Click the S3 to Block Copy Pipeline in the left menu to return to the pipeline GUI. 2. In the Pipeline GUI, drag the Web activity to the right of the S3 to Azure Blob copy activity. 3. Name the web activity Send Failure Email. 4. Click the Save Icon. |  |
| We now want to link the failure branch of the copy activity with the Send Failure Email activity | 1. Drag from the red square of the S3 to Azure Blob activity to the Send Failure Email activity. 2. Click the Failure Email web activity. |  |
| We now are going to fill out the information needed to send the email. Note we use built-in parameter notation to reference values from the pipleline as well our our custom parameter. The body field names match the names setup in the Azure Logic email app to fill out the template.  The header type needs to be set to JSON here for the Azure email logic app. | 1. Click the Settings tab. 2. Fill out the following information: URL: Paste your Azure Logic App URL created in Lab Module 1. Method: Post Body{"dataFactoryName":"@{pipeline().DataFactory}","successtype":"Failed","message":"@{activity('S3 to Azure Blob').error.message}","pipelineName":"@{pipeline().Pipeline}","recipient":"@{pipeline().parameters.recipient}"} 3. Click the + Add Header button and for the Header: Content-Type Expression: Application/json 4. Click the Save icon. |  |

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