Lab 2: ETL Using Azure Databricks

Technologies showcased: Azure Portal

Author: Mike Dobing

Date: 11/6/2018

Table of Contents

[Summary 3](#_Toc517343754)

[Pre-requisites 3](#_Toc517343755)

[Scenario 4](#_Toc517343756)

[Part 1 – Create A Databricks Workspace 4](#_Toc517343757)

[Part 2 – Create a Spark cluster in Databricks 8](#_Toc517343758)

[Part 4 – Create an Azure Data Lake Store account 9](#_Toc517343759)

[Part 5 – Create an Azure Active Directory service principal 11](#_Toc517343760)

[Part 6 – Get application ID and authentication key for the service principal 14](#_Toc517343761)

[Part 7 – Upload data to Data Lake Store 16](#_Toc517343762)

[Part 8 – Associate service principal with Azure Data Lake Store 17](#_Toc517343763)

[Part 9 – Extract data From Data Lake Store 20](#_Toc517343764)

[Part 10 – Transform data IN Azure Databricks 23](#_Toc517343765)

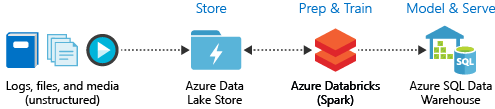
[Part 11 – Load data into Azure SQL Data Warehouse 26](#_Toc517343766)

[Part 12 – Clean up resources 29](#_Toc517343767)

## Summary

In this lab you will perform an ETL (extract, transform, and load data) operation using Azure Databricks. You extract data from Azure Data Lake Store into Azure Databricks, run transformations on the data in Azure Databricks, and then load the transformed data into Azure SQL Data Warehouse.

The steps in this tutorial use the SQL Data Warehouse connector for Azure Databricks to transfer data to Azure Databricks. This connector, in turn, uses Azure Blob Storage as temporary storage for the data being transferred between an Azure Databricks cluster and Azure SQL Data Warehouse.



Steps -

* Create an Azure Databricks workspace
* Create a Spark cluster in Azure Databricks
* Create an Azure Data Lake Store account
* Upload data to Azure Data Lake Store
* Create a notebook in Azure Databricks
* Extract data from Data Lake Store
* Transform data in Azure Databricks
* Load data into Azure SQL Data Warehouse

## Pre-requisites

* Azure Subscription with rights to use/deploy Azure services, and X of Azure credit
* Prereqs complete from lab 1
* Files from here - https://github.com/Azure/usql/blob/master/Examples/Samples/Data/json/radiowebsite/small\_radio\_json.json
* Web browser (Edge/Chrome recommended)
* SQL Server Management Studio Installed

## 

## Scenario

|  |  |  |
| --- | --- | --- |
| Part 1 – Create A Databricks Workspace | | |
| **Scenario** | | |
| First, we are going to create a Databricks Workspace, which allows you create and access Databricks resources | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | In the Azure portal, select **Create a resource** > **Data + Analytics** > **Azure Databricks**. |  |
|  | Under **Azure Databricks Service**, provide the values to create a Databricks workspace. |  |
|  | Provide the following rules –   | Property | Description | | --- | --- | | **Workspace name** | Provide a name for your Databricks  workspace | | **Subscription** | From the drop-down, select your Azure  subscription. | | **Resource group** | Specify whether you want to create a new resource group or use an existing one. A resource group is a container that holds related resources for an Azure solution. For more information, see [Azure Resource Group overview](https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview). | | **Location** | Select **West Europe**. | | **Pricing Tier** | Choose **Standard** |   Select **Pin to dashboard** and then select **Create**.  The account creation takes a few minutes. During account creation, the portal displays the **Submitting deployment for Azure Databricks** tile on the right side. You may need to scroll right on your dashboard to see the tile. There is also a progress bar displayed near the top of the screen. You can watch either area for progress. |  |

|  |  |  |
| --- | --- | --- |
| Part 2 – Create a Spark cluster in Databricks | | |
| **Scenario** | | |
|  | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | In the Azure portal, go to the Databricks workspace that you created, and then select Launch Workspace.  You are redirected to the Azure Databricks portal. From the portal, select Cluster. |  |
|  | In the **New cluster** page, provide the values to create a cluster. |  |
|  | Accept all other default values other than the following:   * Enter a name for the cluster. * For this article, create a cluster with **4.0** runtime. * Make sure you select the **Terminate after *\_\_* minutes of inactivity** checkbox. Provide a duration (in minutes) to terminate the cluster, if the cluster is not being used.   Select **Create cluster**. Once the cluster is running, you can attach notebooks to the cluster and run Spark jobs. |  |

|  |  |  |
| --- | --- | --- |
| Part 4 – Create an Azure Data Lake Store account | | |
| **Scenario** | | |
| In this section, you create an Azure Data Lake Store account and associate an Azure Active Directory service principal with it. Later in this tutorial, you use this service principal in Azure Databricks to access Azure Data Lake Store | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | From the Azure portal, select Create a resource > Storage > Data Lake Store.  In the New Data Lake Store blade, provide the values as shown in the following screenshot:  Use WEST EUROPE for the region and your own Resource Group  Click CREATE |  |
|  | Provide the following values –   | Property | Description | | --- | --- | | **Name** | Enter a unique name for the Data Lake Store account. | | **Subscription** | From the drop-down, select your Azure subscription. | | **Resource group** | For this tutorial, select the same resource group you used while creating the Azure Databricks workspace. | | **Location** | Select **West Europe** | | **Pricing package** | Select **Pay-as-you-go**. | | **Encryption Settings** | Keep the default settings. | |  |
| Part 5 – Create an Azure Active Directory service principal | | |
| **Scenario** | | |
| This topic creates the azure active directory principal that will act as the service principal for databricks | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | From the Azure portal, select All services, and then search for Azure Active Directory.  Select App registrations. |  |
|  | Select New application registration. |  |
|  | Provide a name and URL for the application. Select Web app / API for the type of application you want to create. Provide a sign-on URL, and then select Create. |  |
|  | To access the Data Lake Store account from Azure Databricks, you must have the following values for the Azure Active Directory service principal you created:   * Application ID * Authentication key * Tenant ID   In the following sections, you retrieve these values for the Azure Active Directory service principal you created earlier. |  |

|  |  |  |
| --- | --- | --- |
| Part 6 – Get application ID and authentication key for the service principal | | |
| **Scenario** | | |
| When programmatically logging in, you need the ID for your application and an authentication key. To get those values, use the following steps | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | From App registrations in Azure Active Directory, select your application. |  |
|  | Copy the Application ID and store it in your application code. Some sample applications refer to this value as the client ID. |  |
|  | To generate an authentication key, select Settings |  |
|  | To generate an authentication key, select Keys. |  |
|  | Provide a description of the key, and a duration for the key. When done, select Save |  |
|  | After saving the key, the value of the key is displayed. Copy this value because you are not able to retrieve the key later. You provide the key value with the application ID to log in as the application. Store the key value where your application can retrieve it |  |

|  |  |  |
| --- | --- | --- |
| Part 7 – Upload data to Data Lake Store | | |
| **Scenario** | | |
| In this section, you upload a sample data file to Data Lake Store. You use this file later in Azure Databricks to run some transformations. The sample data (**small\_radio\_json.json**) that you use in this tutorial is available in this [Github repo](https://github.com/Azure/usql/blob/master/Examples/Samples/Data/json/radiowebsite/small_radio_json.json). | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | From the Azure portal, select the Data Lake Store account you created. |  |
|  | Within the Data Explorer, click **Upload**. |  |
|  | In this tutorial, you uploaded the data file to the root of the Data Lake Store. So, the file is now available at | adl://<YOUR\_DATA\_LAKE\_STORE\_ACCOUNT\_NAME>.azuredatalakestore.net/small\_radio\_json.json. |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Part 8 – Associate service principal with Azure Data Lake Store | | |
| **Scenario** | | |
| In this section, you associate the data in Azure Data Lake Store account with the Azure Active Directory service principal you created. This ensures that you can access the Data Lake Store account from Azure Databricks. For the scenario in this article, you read the data in Data Lake Store to populate a table in SQL Data Warehouse. According to [Overview of Access Control in Data Lake Store](https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-access-control#common-scenarios-related-to-permissions), to have read access on a file in Data Lake Store, you must have:   * **Execute** permissions on all the folders in the folder structure leading up to the file. * **Read** permissions on the file itself.   Perform the following steps to grant these permissions. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | From the Azure portal, select the Data Lake Store account you created, and then select Data Explorer. |  |
|  | In this scenario, because the sample data file is at the root of the folder structure, you only need to assign **Execute** permissions at the folder root. To do so, from the root of data explorer, select **Access**. |  |
|  | Under **Access**, select **Add**. |  |
|  | Under Assign permissions, click Select user or group and search for the Azure Active Directory service principal you created earlier.  Select the AAD service principal you want to assign and click **Select** |  |
|  | Under **Assign permissions**, click **Select permissions** > **Execute**. Keep the other default values and select **OK** under **Select permissions** and then under **Assign permissions**. |  |
|  | Go back to the Data Explorer and now click the file on which you want to assign the read permission. Under **File Preview**, select **Access**. |  |
|  | Under **Access** select **Add**. Under **Assign permissions**, click **Select user or group** and search for the Azure Active Directory service principal you created earlier.  Select the AAD service principal you want to assign and click **Select**. |  |
|  | Under **Assign permissions**, click **Select permissions** > **Read**. Select **OK** under **Select permissions** and then under **Assign permissions**.  The service principal now has sufficient permissions to read the sample data file from Azure Data Lake Store. |  |

|  |  |  |
| --- | --- | --- |
| Part 9 – Extract data From Data Lake Store | | |
| **Scenario** | | |
| In this section, you create a notebook in Azure Databricks workspace and then run code snippets to extract data from Data Lake Store into Azure Databricks. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | In the [Azure portal](https://portal.azure.com/), go to the Azure Databricks workspace you created, and then select **Launch Workspace**.  In the left pane, select **Workspace**. From the **Workspace** drop-down, select **Create** > **Notebook**. |  |
|  | In the **Create Notebook** dialog box, enter a name for the notebook. Select **Scala** as the language, and then select the Spark cluster that you created earlier.  Select **Create**. |  |
|  | Add the following snippet in an empty code cell and replace the placeholder values with the values you saved earlier for the Azure Active Directory service principal.  Press **SHIFT + ENTER** to run the code cell. | spark.conf.set("dfs.adls.oauth2.access.token.provider.type", "ClientCredential")  spark.conf.set("dfs.adls.oauth2.client.id", "<APPLICATION-ID>")  spark.conf.set("dfs.adls.oauth2.credential", "<AUTHENTICATION-KEY>")  spark.conf.set("dfs.adls.oauth2.refresh.url", "https://login.microsoftonline.com/<TENANT-ID>/oauth2/token") |
|  | You can now load the sample json file in Data Lake Store as a dataframe in Azure Databricks. Past the following snippet in a new code cell, replace the placeholder value, and then press **SHIFT + ENTER**. | val df = spark.read.json("adl://<DATA LAKE STORE NAME>.azuredatalakestore.net/small\_radio\_json.json") |
|  | Run the following code snippet to see the contents of the data frame. | df.show() |
|  | You see an output similar to the following snippet:  You have now extracted the data from Azure Data Lake Store into Azure Databricks. |  |

|  |  |  |
| --- | --- | --- |
| Part 10 – Transform data IN Azure Databricks | | |
| **Scenario** | | |
| The raw sample data **small\_radio\_json.json** captures the audience for a radio station and has a variety of columns. In this section, you transform the data to only retrieve specific columns in from the dataset. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | Start by retrieving only the columns firstName, lastName, gender, location, and levelfrom the dataframe you already created. | *val specificColumnsDf = df.select("firstname", "lastname", "gender", "location", "level")* |
|  | You get an output as shown in the following snippet: |  |
|  | You can further transform this data to rename the column **level** to **subscription\_type**.  Press **SHIFT + ENTER** to run the code cell. | val renamedColumnsDF = specificColumnsDf.withColumnRenamed("level", "subscription\_type")  renamedColumnsDF.show() |
|  | You get an output as shown in the following snippet. |  |

|  |  |  |
| --- | --- | --- |
| Part 11 – Load data into Azure SQL Data Warehouse | | |
| **Scenario** | | |
| In this section, you upload the transformed data into Azure SQL Data Warehouse. Using the Azure SQL Data Warehouse connector for Azure Databricks, you can directly upload a dataframe as a table in SQL data warehouse.  As mentioned earlier, the SQL date warehouse connector uses Azure Blob Storage as a temporary storage to upload data between Azure Databricks and Azure SQL Data Warehouse. So, you start by providing the configuration to connect to the storage account. You must have already created the account as part of the prerequisites for this article. | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | Provide the configuration to access the Azure Storage account from Azure Databricks. | val blobStorage = "<STORAGE ACCOUNT NAME>.blob.core.windows.net"  val blobContainer = "<CONTAINER NAME>"  val blobAccessKey = "<ACCESS KEY>" |
|  | Specify a temporary folder that will be used while moving data between Azure Databricks and Azure SQL Data Warehouse. | val tempDir = "wasbs://" + blobContainer + "@" + blobStorage +"/tempDirs" |
|  | Run the following snippet to store Azure Blob storage access keys in the configuration. This ensures that you do not have to keep the access key in the notebook in plain text. | val acntInfo = "fs.azure.account.key."+ blobStorage  sc.hadoopConfiguration.set(acntInfo, blobAccessKey) |
|  | Provide the values to connect to the Azure SQL Data Warehouse instance. You must have created a SQL data warehouse as part of the prerequisites. | //SQL Data Warehouse related settings  val dwDatabase = "<DATABASE NAME>"  val dwServer = "<DATABASE SERVER NAME>"  val dwUser = "<USER NAME>"  val dwPass = "<PASSWORD>"  val dwJdbcPort = "1433"  val dwJdbcExtraOptions = "encrypt=true;trustServerCertificate=true;hostNameInCertificate=\*.database.windows.net;loginTimeout=30;"  val sqlDwUrl = "jdbc:sqlserver://" + dwServer + ".database.windows.net:" + dwJdbcPort + ";database=" + dwDatabase + ";user=" + dwUser+";password=" + dwPass + ";$dwJdbcExtraOptions"  val sqlDwUrlSmall = "jdbc:sqlserver://" + dwServer + ".database.windows.net:" + dwJdbcPort + ";database=" + dwDatabase + ";user=" + dwUser+";password=" + dwPass |
|  | Run the following snippet to load the transformed dataframe, **renamedColumnsDF**, as a table in SQL data warehouse. This snippet creates a table called **SampleTable** in the SQL database. | spark.conf.set(  "spark.sql.parquet.writeLegacyFormat",  "true")  renamedColumnsDF.write  .format("com.databricks.spark.sqldw")  .option("url", sqlDwUrlSmall)  .option("dbtable", "SampleTable")  .option( "forward\_spark\_azure\_storage\_credentials","True")  .option("tempdir", tempDir)  .mode("overwrite")  .save() |
|  | Connect to the SQL database and verify that you see a **SampleTable**. |  |
|  | Run a select query to verify the contents of the table. It should have the same data as the **renamedColumnsDF** dataframe. |  |

|  |  |  |
| --- | --- | --- |
| Part 12 – Clean up resources | | |
| **Scenario** | | |
| After you have finished running the tutorial, you can terminate the cluster. IF YOU ARE GOING ONTO THE DATA SCIENCE LAB THEN DO NOT COMPLETE THIS STEP | | |
| **Commentary / Notes** | **Click Steps & ‘Bits’** | **Screenshots** |
|  | From the Azure Databricks workspace, from the left pane, select **Clusters**. For the cluster you want to terminate, move the cursor over the ellipsis under **Actions** column, and select the **Terminate** icon. |  |
|  | If you do not manually terminate the cluster it will automatically stop, provided you selected the **Terminate after \_\_ minutes of inactivity** checkbox while creating the cluster. In such a case, the cluster automatically stops if it has been inactive for the specified time. |  |

**END OF LAB 2**

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