Big Data Orientation

Lab 4 – Processing Big Data in Microsoft Azure

Overview

Azure includes a range of Big Data technologies that you can use to process data at scale. In this lab, you will explore two of these technologies; Azure Data Lake Analytics, and Azure HDInsight.

Before You Start

To complete this lab, you will need the following:

- A web browser
- A Windows, Linux, or Mac OS X computer

Important: Before you can perform the lab exercises, you must complete the previous labs in this course.

Exercise 1: Processing Data with Azure Data Lake Analytics

In this exercise, you will create an Azure Data Lake Analytics Account and use it to process data in Azure Storage.

Create an Azure Data Lake Analytics Account

Before you can use Azure Data Lake Analytics to process data, you must create an Azure Data Lake Analytics account, and associate it with at least one Azure Data Lake store. In this case, you will associate your Azure Data Lake Analytics account with the Azure Data Lake Store you created in the first lab of this course.

1. In a web browser, navigate to http://portal.azure.com, and if prompted, sign in using the Microsoft account that is associated with your Azure subscription.
2. In the Microsoft Azure portal, in the Hub Menu, click **New**. Then in the **Data + Analytics** menu, click **Data Lake Analytics**.
3. In the **New Data Lake Analytics Account** blade, enter the following settings, and then click **Create**:

- **Name**: *Enter a unique name (and make a note of it!)*
- **Subscription**: *Select your Azure subscription*
- **Resource Group**: *Use the existing resource group that contains your Azure Data Lake Store*
- **Location:** *Select the region where your Azure Data Lake Store is located.*
- **Data Lake Store:** *Select your existing Azure Data Lake Store*
- **Pin to dashboard**: *Not selected*

4. In the Azure portal, view Notifications to verify that deployment has started. Then wait for the resource to be deployed (this can take a few minutes.)

Link an Azure Storage Account

Azure Data Lake Analytics can access data in the Data Lake Store you associated it with. You can also link additional Data Lake Stores and Azure Storage accounts and process data that is stored in them. In this procedure, you will link your existing Azure Storage account to your Azure Data Lake Analytics account.

1. Click **All Resources**, and then click the Azure Data Lake Analytics account you just provisioned.
2. In the blade for your Data Lake Analytics account, view the **Data Sources** page, and then click **Add Data Source**.
3. In the Add Data Source blade, enter the following details and click **Add**:
   - **Storage Type**: Azure Storage
   - **Selection Method**: Select Account
   - **Azure Storage**: Select your Azure Storage account
4. On the **Data Sources** page, verify that both your Azure Data Lake Store and Azure Storage account are listed.

Process Data

Azure Data Lake Analytics enables you to process data by writing U-SQL code. U-SQL is similar to Transact-SQL, but also includes support for C# syntax and data types.

1. In the blade for your Data Lake Analytics account, view the Data Explorer page.
2. Under **Storage accounts**, expand your Azure Storage account and its **bigdata** container, and select the data folder. This should contain the three files that you uploaded in the first lab of this course (**customers.txt**, **products.txt**, and **reviews.txt**).
3. Click **products.txt** to preview the data it contains (if a preview is not shown, you can download the file and open it locally). Note that this file contains a tab-delimited list of products. Note also that some of the product details contain the text "NULL" – this is used in the **Color**, **Size**, and **Weight** columns to indicate that there is no known value for this product.
4. In the **File Preview** pane, click **Properties** to view the properties of the file, and note the WASB PATH property, which is used to access this file in Azure Storage. The path is in the form **wasbs://bigdata@<*storageacct*>/data/products.txt** (where <*storageacct*> is the name of your Azure storage account.
5. Return to the blade for your Azure Data Lake Store, and on the **Overview** page, click **+ New Job**.
6. In the **New U-SQL Job** blade, change the **Job Name** to **Process Products**.
7. In the query pane, enter the following U-SQL code (you can copy and paste this from **u-sql.txt** in the lab folder) – replace <*storageacct*> with the name of your Azure storage account:

```
@products = EXTRACT ProductID string,
                    ProductName string,
                    ProductNumber string,                    Color
string,
                    StandardCost decimal,
                    ListPrice decimal,
                    Size string,
                    NetWeight string
             FROM "wasbs://bigdata@<storageacct>/data/products.txt"
             USING Extractors.Tsv(skipFirstNRows:1);

@cleanProducts = SELECT ProductID,
                 ProductName,
                 ProductNumber,
                 (Color == "NULL") ? "None" : Color AS Color,
                 StandardCost,
                 ListPrice,
                 ListPrice - StandardCost AS Markup,
                 (Size == "NULL") ? "N/A" : Size AS Size,
                 (NetWeight == "NULL") ? "0.00" : NetWeight AS NetWeight
FROM @products;

OUTPUT @cleanProducts
TO "output/cleanproducts.csv"
ORDER BY ProductID
USING Outputters.Csv(outputHeader:true);
```

8. Review the code in the query, noting that it:
    - Loads the tab-delimited data from the text file into a table variable with appropriate columns, skipping the first row (which contains the column names).
    - Cleans the data by changing NULL **Color** values to "None", changing NULL **Size** values to "N/A", and changing NULL **Weight** values to 0.00.
    - Calculates the markup for each product by subtracting the cost from the list price.
    - Stores the cleaned data in a comma-delimited file.
9. Click **Submit Job** and observe the job details as it is run. After the job has been prepared, a job graph should be displayed, showing the steps used to execute it.
10. When the job has finished, on the **Output** tab and **cleanproducts.csv** to see a preview of the results.
11. Click **Data Explorer** and browse to the **output** folder in your Azure Data Lake Store – this where the **cleanproducts.csv** file was saved.

Exercise 2: Processing Data with HDInsight

In this exercise, you will create an Azure HDInsight cluster and use it to process data in Azure Storage.

Provision an HDInsight Cluster

1. In the Microsoft Azure portal, in the Hub Menu, click **Create a resource**. Then in the **Analytics** menu, click **HDInsight**
2. In the **HDInsight** blade, ensure the *Quick create* option is selected.
3. In the **Basics** blade, enter the following settings, and then click **Next**:

- **Cluster Name**: *Enter a unique name (and make a note of it!)*
- **Subscription**: *Select your Azure subscription*
- **Cluster type**
  - **Cluster Type**: Spark
  - **Cluster Operating System**: Linux
  - **HDInsight Version**: *Choose HDI 3.6 or **earlier**. Do not choose HDI 4.0.*
- **Cluster Login Username**: *Enter a user name of your choice (and make a note of it!)*
- **Cluster Login Password**: *Enter and confirm a strong password (and make a note of it!)*
- **SSH Username**: *Enter a user name of your choice for SSH access (and make a note of it!)*
- **SSH Password**: *Use the same password as the cluster login password*
- **Resource Group:** *Select the resource group you have used throughout the labs in this course.*
- **Location**: *Choose the region where your Azure Storage account is located.*

4. In the **Storage** blade, enter the following settings, and then click **Next**:
   - **Primary storage type**: Azure Storage
   - **Use Existing**: *My Subscriptions*
   - **Select a storage account**: *Select your existing Azure Storage account*
   - **Container**: *The name of your cluster*
   - **Advanced Settings**: *None*
5. In the **Cluster Summary** blade, review the cluster configuration settings. Notice that the Quick Create method creates a default cluster size of 2 Head nodes (D12) and 4 worker nodes (D13). Once you have reviewed the cluster configuration settings, click **Create.**
6. In the Azure portal, view **Notifications** to verify that deployment has started. Then wait for the cluster to be deployed (this can take a long time – often 30 minutes or more. Feel free to catch up on your social media networks while you wait!)

Note: As soon as an HDInsight cluster is running, the credit in your Azure subscription will start to be charged. Free-trial subscriptions include a limited amount of credit limit, which should be enough to complete the labs in this course as long as clusters are deleted when not in use. If you decide not to complete this lab, follow the instructions in the *Cleaning Up* procedure at the end of the lab to delete your cluster to avoid using your Azure credit unnecessarily.

Process Data using Hive

Hive is a SQL-like language that is commonly used in Hadoop and Spark clusters to process big data.

1. In the Azure portal, if it is not already open, browse to the blade for the HDInsight cluster you just created.
2. In the **Overview** blade for your cluster, under **Quick Links**, click **Cluster Dashboards**.
3. In the **Cluster Dashboards** blade, click **Ambari Home** to open a new browser tab containing the cluster dashboard. When prompted, log on using the cluster login and password you specified when provisioning the cluster (be sure to use the cluster login and not the SSH username!)

4. In the Ambari dashboard, the list of services are on the left. In the list of services that are running on the cluster, click **Hive**.
5. Under the summary of Hive service status, next to **Hive View 2.0**. click **Go To View**. After some initialization, the hive query editor will be displayed. Be patient, it may take a minute.
6. In the Hive query editor, enter the following HiveQL code (you can copy and paste this from **hive.txt** in the lab folder) – replace all instances of <*storageacct*> with the name of your Azure storage account:

```
DROP TABLE IF EXISTS customers;

DROP TABLE IF EXISTS cleanedcustomers;

CREATE EXTERNAL TABLE customers
(CustomerID INT,  Title
STRING,
 FirstName STRING,
 MiddleName STRING,
 LastName STRING,
 EmailAddress STRING,
 Phone STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE LOCATION
'wasbs://bigdata@<storageacct>.blob.core.windows.net/customers/'
TBLPROPERTIES ("skip.header.line.count"="1");

LOAD DATA INPATH
'wasbs://bigdata@<storageacct>.blob.core.windows.net/data/customers.txt'
INTO TABLE customers;

CREATE EXTERNAL TABLE cleanedcustomers
(CustomerID INT,
 CustomerName STRING,
 EmailAddress STRING,
 Phone STRING)
 ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION
'wasbs://bigdata@<storageacct>.blob.core.windows.net/output';

INSERT INTO TABLE cleanedcustomers SELECT CustomerID,
regexp_replace(concat(concat(concat(concat(concat(concat(Title,
' '), FirstName),' '), MiddleName), ' '), LastName), 'NULL', '') AS
CustomerName,
         EmailAddress,
         Phone
FROM customers
ORDER BY CustomerID;

SELECT * FROM cleanedcustomers;
```

7. Review the code in the query, noting that it:
    - Creates a table that defines a schema for tab-delimited customer data. The table is based on a new folder named **customers** in your Azure Storage **bigdata** blob container.
    - Loads the **customers.txt** text file from the data folder into the table

> **Note**: this will *move* the **customers.txt** file to the **customers** folder. If you plan to re-run this code, you must first move the file back to the data folder!

- Creates a table named **cleanedcustomers**, which is based on a new folder named **output**.
- Concatenates the various fields that make up the customer's name, removing "NULL" values.
- Loads the modified data into the **cleanedcustomers** table (therefore creating a new text file in the **output** folder).
- Retrieves all rows from the **cleanedcustomers** table.

8. Click **Execute** and wait for the query to complete.
9. Use Azure Storage Explorer or the Data Explorer in the Azure portal to view the contents of the **output** folder in the **bigdata** blob container, and verify that it contains a text file with a name similar to **000000_0**. This contains the processed customer data in comma-delimited format.

Process Data using Spark

Spark clusters support multiple languages, including Python. You can use the Spark libraries for Python to distribute the data processing work across the nodes in the cluster.

1. In the Azure portal, if it is not already open, browse to the blade for the HDInsight cluster you just created and view the **Cluster Dashboards** page.
2. Click Jupyter Notebook to open a new tab containing the Jupyter Notebook environment (if you are prompted to log in, do so using your cluster login and password).

> **Note**: Jupyter Notebooks is a commonly used browser-based coding environment for working with data.

3. In the Jupyter folder tree page, click **Upload**, and then select and upload the **SparkCustomers.ipynb** notebook from the lab folder on your computer where you extracted the lab files for this course. This file contains some Python code that uses Spark libraries to perform further processing on the customer data you cleaned using Hive.
4. After the **SparkCustomers.ipynb** notebook has been uploaded, click it to open it in a new browser tab.
5. Review the notebook, noting that it consists of a series of *cells*. Each cell contains either some text or some Python code.
6. Review the code in the cell under the text **Load the cleaned customer data generated by Hive**, noting that this code loads the output file that was generated by your Hive code in the previous exercise into a dataframe with an appropriate schema. Then edit the code in this cell to replace ***<storageacct>*** with the name of your Azure storage account.
7. With the first code cell still selected, click the ▶| button to run the cell. At the top right of the page, next to **PySpark3**, note that a solid ● icon indicates that the code is running. When it has finished, the icon changes to ○.
8. Review the results produced by the code, which show the first 20 rows of the data generated by Hive.
9. Select the code cell under the comment **Create a new dataframe containing only the name and phone number** and run it. This produces a new dataframe that contains only the **CustomerName** and **Phone** columns.
10. Select and run the code cell under the comment **Now create a Spark SQL table**. There is no visible output from this code, but it creates a temporary table that you can query using SQL.

11. Select and run the code cell under the comment **Use inline SQL to query the table**. This code runs a SQL query against the temporary table to retrieve the phone numbers for all employees with a name that contains “Kath”.

Cleaning Up

Now that you have finished the labs in this course, you can delete the Azure resources you have been using.

Delete the Azure Resource Group

If you no longer need the Azure resources used in this course, you should delete them to avoid incurring unnecessary costs.

1. Close all browser tabs other than the one containing the Azure portal.
2. In the Azure portal, click **Resource Groups**.
3. On the **Resource groups** blade, click the resource group that contains the resources you created in the labs for this course, and then on the **Resource group** blade, click **Delete**. On the confirmation blade, type the name of your resource group, and click **Delete**.
4. Wait for your resource group to be deleted, and then click **All Resources**, and verify that the resources have all been removed.
5. Close the browser.