Processing Big Data with Azure Data Lake

Lab 1 - Getting Started with Azure Data Lake

Overview

In this lab, you will create an Azure Data Lake Analytics account and an associated Azure Data Lake store. You will then upload some data files to the Azure Data Lake store and create a U-SQL job to process the data.

What You'll Need

To complete the labs, you will need the following:

- A web browser
- A Microsoft account
- A Microsoft Azure subscription
- A Windows, Linux, or Mac OS X computer
- The lab files for this course

Note: To set up the required environment for the lab, follow the instructions in the Setup document for this course. Specifically, you must have signed up for an Azure subscription.

Creating an Azure Data Lake Analytics Account

In this exercise, you will create an Azure Data Lake Analytics Account and associated Azure Data Lake store.

Note: The Microsoft Azure portal is continually improved in response to customer feedback. The steps in this exercise reflect the user interface of the Microsoft Azure portal at the time of writing, but may not match the latest design of the portal exactly.

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.

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 **Intelligence and 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**: *Create a new resource group with a unique name*
- **Location:** *Select any available region*
- **Data Lake Store:** *Create a new Data Lake Store with a unique name (and make a note of it!)*
- **Pin to dashboard**: *Not selected*

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

Explore Your Data Lake Store

Your Azure Data Lake store will be used to store data and code files, and databases for use by your Azure Data Lake Analytics account.

1. In the Microsoft Azure portal, browse to your Azure Data Lake Store.
2. On the **Overview** page, note the current storage utilization (which should be 0 bytes).
3. View the Data Explorer page, which shows the current contents of your store. It should contain two folders (**catalog** and **system**). Later, you will use this tool to upload and download data to your store.

Explore Your Data Lake Analytics Account

Your Azure Data Lake Analytics account provides an on-demand service for performing big data processing jobs.

1. In the Azure portal, browse to your Azure Data Lake Analytics account.
2. In the Overview page note the information available.
3. Click **Data Explorer** and view the current contents of your storage account. It should contain two folders (**catalog** and **system**). Later, you will use this tool to upload data to your store.

Running Azure Data Lake Analytics Jobs

In this exercise, you will run a simple Azure Data Lake Analytics Job to process a web server log file.

View and Upload a Source Data File

In this lab, you will use Azure Data Lake Analytics to process web server log files. Initially, you will run some simple jobs to process a single log file.

1. In the folder where you extracted the lab files, open the **iislogs** folder and then use a text editor to view the **2008-01.txt** file.
2. Review the contents of the file, noting that it contains some header rows (prefixed with a # character) and some space-delimited web server request records for the month of January in 2008. Then close the file without saving any changes.
3. In the Azure portal, view the **Data Explorer** page for your Azure Data Lake Analytics account, and create a new folder named **iislogs** in the root of your Azure Data Lake store.
4. Open the newly created **iislogs** folder. Then click **Upload**, and upload the **2008-01.txt** file you viewed previously.

Create a Job

Now that you have uploaded the source data to your Azure Data Lake store, you can use a U-SQL query to read and process the data.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click the **New Job** page.
2. In the **New U-SQL Job** blade, in the Job Name box, type **Read Log File**. Then in the code window, enter the following code:

```
@log = EXTRACT entry string
       FROM "/iislogs/2008-01.txt"
       USING Extractors.Text();

OUTPUT @log
TO "/output/log.txt"
USING Outputters.Text();
```

   This code uses the built-in **Text** *extractor* to read the contents of the **2008-01.txt** log file. The default field delimiter for the **Text** extractor is a comma, which the source data does not contain, so the code reads each line of text in the log file, and then uses the default **Text** *outputter* to write the data to an output file.

3. 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.
4. When the job has finished, click the **Output** tab and select **log.txt** to see a preview of the results, which should be essentially the same as the original source data.

Add a SELECT Statement to Filter the Data

The job you have created does not perform any useful work. You will now adapt it to filter the data, removing the comment rows.

1. Return to the New SQL Job blade, and modify the query as shown here:

```
@log = EXTRACT entry string
       FROM "/iislogs/2008-01.txt"
       USING Extractors.Text();

@cleaned = SELECT entry
           FROM @log
           WHERE entry.Substring(0,1) != "#";

OUTPUT @cleaned
TO "/output/cleaned.txt"
USING Outputters.Text();
```

   This query uses a SELECT statement to filter the data retrieved by the extractor. Note that the WHERE clause in the SELECT statement uses the C# **Substring** function to filter out rows that start with a "#" character. The ability to blend C# and SQL is what makes U-SQL such a flexible and extensible language for data processing.

2. Click **Submit Job** and observe the job details as it is run.
3. When the job has finished, click the **Output** tab and select **cleaned.txt** to see a preview of the results, which no longer contains comment rows.

   Note that the preview automatically displays the data as a table, detecting spaces as the delimiter. However, the output data is plain text.

4. Download the output file and open it in a text editor, and note that each row in the data is represented as a single string.

Apply a Schema to the Data

You have seen how to use U-SQL to read and filter text data based on rows of text. Now you will apply a schema to the data, separating it into discrete fields that can be processed individually.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click the **New Job** page.
2. In the **New U-SQL Job** blade, in the Job Name box, type **Process Log Entries**. Then in the code window, enter the following code:

```
@log = EXTRACT date string,
               time string,
               client_ip string,
               username string,
               server_ip string,
               port int,
               method string,
               stem string,
               query string,
               status string,
               server_bytes int,
               client_bytes int,
               time_taken int,
               user_agent string,
               referrer string
        FROM "/iislogs/2008-01.txt"
        USING Extractors.Text(' ', silent:true);

OUTPUT @log
TO "/output/log.csv"
USING Outputters.Csv();
```

   This code uses the built-in **Text** extractor to read the contents of the **2008-01.txt** log file based on a schema that defines multiple columns and their data types. The delimiter for the **Text** extractor is specified as a space, and the extractor is configured to silently drop any rows that do not match the schema.

   The data is then written to the Azure Data Lake store using the built-in **Csv** outputter. This is a specific implementation of the **Text** outputter that saves the data in comma-delimited format.

3. Click **Submit Job** and observe the job details as it is run.
4. When the job has finished, click the **Output** tab and select **log.csv** to see a preview of the results.
5. Download the output file and open it in a text editor or spreadsheet application, and note that each row in the data contains multiple fields.

Apply a Schema to the Data

Now that you have applied a schema to the data, you can write queries that reference individual fields.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click the **Jobs** page and view the details for the jobs that you have run so far in this lab.
2. Click the most recent job, which should be the **Process Log Entries** job you created in the previous procedure.
3. In the **Process Log Entries** blade, click **View Script** to view the U-SQL script for the job. Then click **Duplicate Script** to create a new job based on this script.
4. In the **New U-SQL Job** blade, in the **Job Name** box, type **Summarize Log**. Then in the code window, modify the script as follows:

```
@log = EXTRACT date string,
               time string,
               client_ip string,
               username string,
               server_ip string,
               port int,
               method string,
               stem string,
               query string,
               status string,
               server_bytes int,
               client_bytes int,
               time_taken int,
               user_agent string,
               referrer string
        FROM "/iislogs/2008-01.txt"
        USING Extractors.Text(' ', silent:true);

@dailysummary = SELECT date,
                       COUNT(*) AS hits,
                       SUM(server_bytes) AS bytes_sent,
                       SUM(client_bytes) AS bytes_received
                FROM @log
                GROUP BY date;

OUTPUT @dailysummary
TO "/output/daily_summary.csv"
ORDER BY date
USING Outputters.Csv();
```

   This code uses a SELECT statement with a GROUP BY clause to summarize the total log entries, server bytes, and client bytes by day.

   The results are then written to the Azure Data Lake store using the built-in **Csv** outputter, with an ORDER BY clause that sorts the results into ascending order of date.

5. Click **Submit Job** and observe the job details as it is run.
6. When the job has finished, click the **Output** tab and select **daily_summary.csv** to see a preview of the results.
7. Download the output file and open it in a text editor or spreadsheet application, and note that each row in the data contains a daily summary of hits, bytes sent, and bytes received.

Processing Multiple Files

In this exercise, you will apply what you have learned about processing a single file to data in multiple files.

Upload Additional Source Data Files

In addition to the log files for January 2008 that you have already processed, youwill now process the log data for February to June.

1. In the folder where you extracted the lab files, verify that there are files for an additional five months.
2. In the Azure portal, view the **Data Explorer** page for your Azure Data Lake Analytic account, and open the **iislogs** folder you created earlier.
3. Click **Upload**, and then select the **2008-02.txt**, **2008-03.txt**, **2008-04.txt**, **2008-05.txt**, **2008-06.txt** files (you can hold the CTRL key to select multiple files) and upload them.

Create a Job to Process Multiple Files

Now that your data is spread across multiple files, you can use a wildcard in your query to read data from them all.

1. In the Azure portal, on the blade for your Azure Data Lake Analytics account, click the **Jobs** page and view the details for the jobs that you have run so far in this lab.
2. Click the most recent job, which should be the **Summarize Log** job you created in the previous exercise.
3. In the **Summarize Log** blade, click **View Script** to view the U-SQL script for the job. Then click **Duplicate Script** to create a new job based on this script.
4. In the **New U-SQL Job** blade, in the **Job Name** box, type **Summarize Logs**. Then in the code window, modify the script as follows:

```
@log = EXTRACT date string,
               time string,
               client_ip string,
               username string,
               server_ip string,
               port int,
               method string,
               stem string,
               query string,
               status string,
               server_bytes int,
               client_bytes int,
               time_taken int,
               user_agent string,
               referrer string
        FROM "/iislogs/{*}.txt"
        USING Extractors.Text(' ', silent:true);

@dailysummary = SELECT date,
                       COUNT(*) AS hits,
                       SUM(server_bytes) AS bytes_sent,
                       SUM(client_bytes) AS bytes_received
                FROM @log
                GROUP BY date;
```

```
OUTPUT @dailysummary
TO "/output/six_month_summary.csv"
ORDER BY date
USING Outputters.Csv();
```

This code uses the wildcard placeholder **{*}** to read all files with a .txt extension from the **iislogs** folder.

5. Change the **Parallelism** value to **4** – this query will help scale the query across more vertices (you will learn more about optimizing query performance later in this course).
6. Click **Submit Job** and observe the job details as it is run.
7. When the job has finished, click the **Output** tab and select **six_month_summary.csv** to see a preview of the results.
8. Download the output file and open it in a text editor or spreadsheet application, and note that each row in the data contains a daily summary of hits, bytes sent, and bytes received for January to June.

Note: You will use the resources you created in this lab when performing the next lab, so do not delete them. Ensure that all jobs are stopped to minimize ongoing resource usage costs.