Orchestrating Big Data Solutions with Azure Data Factory

Lab 4 – Transforming Data with Hive

Note: If you prefer to transform data using U-SQL, an alternative version of this lab is provided in which you can use Azure Data Lake Analytics to run a U-SQL script.

Overview

In this lab, you will use Azure Data Factory to implement a pipeline that uses Hive to transform web server log data before copying it to Azure SQL Database.

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
- The Azure resources created in the previous labs

Important: If you have not completed Lab 1, or you have deleted the storage account, SQL database, and Azure Data Factory resources you created, complete Lab 1 now.

Exercise 1: Preparing for Data Transformation

In this exercise, you will prepare the environment for your data transformation pipeline, and explore the source data.

Upload the Source Data and Hive Script

The source data for this pipeline is a monthly log file from a web server. Your pipeline will use a Hive script to aggregate the log data to create daily summaries.

1. In the **iislogs** subfolder of the folder where you extracted the lab files for this course, open the **2016** subfolder and note that it contains six subfolders (**01** to **06**). These folders contain web server log files for the months of January to June in 2016.
2. In the **01** folder, open **log.txt** in a text editor, and examine the data it contains. After some initial header rows, the log files contain details of web server requests. After you have viewed the structure of the data, close the text editor without saving any changes.

3. In the **iislogs** folder, use a text editor to open the **SummarizeLogs.hql** script file, and note that it contains the following Hive script:

```
DROP TABLE IF EXISTS rawlogs;
DROP TABLE IF EXISTS logsummary;

CREATE EXTERNAL TABLE rawlogs
(log_date DATE,
 log_time STRING,
 c_ip STRING,
 cs_username STRING,
 s_ip STRING,
 s_port STRING,
 cs_method STRING,
 cs_uri_stem STRING,
 cs_uri_query STRING,
 sc_status STRING,
 sc_bytes INT,
 cs_bytes INT,
 time_taken INT,
 cs_user_agent STRING,
 cs_referrer STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
STORED AS TEXTFILE LOCATION
'${hiveconf:log_folder}/${hiveconf:year}/${hiveconf:month}/'
tblproperties ("skip.header.line.count"="4");

CREATE EXTERNAL TABLE logsummary
  (log_date DATE,
   requests INT,
   bytes_in FLOAT,
   bytes_out FLOAT)
PARTITIONED BY (year INT, month INT)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE LOCATION '${hiveconf:summary_folder}';

INSERT OVERWRITE TABLE logsummary PARTITION(year, month)
SELECT log_date,
       COUNT(*) AS requests,
       SUM(cs_bytes) AS inbound_bytes,
       SUM(sc_bytes) AS outbound_bytes,
       YEAR(log_date),
       MONTH(log_date)
FROM rawlogs
GROUP BY log_date;
```

This script:

a. Drops Hive tables named **rawlogs** and **logsummary** if they already exist.
b. Creates a Hive table named **rawlogs** on a specified folder that represents the year and month and contains text-based data files.
c. Creates a Hive table named **logsummary** that is based on a specified folder and is partitioned by year and month.

    d. Uses a query to insert data aggregated by day from the **rawlogs** table into the **logsummary** table.
4. Close the Hive script without saving any changes.
5. Start Azure Storage Explorer, and if you are not already signed in, sign into your Azure subscription.
6. Expand your storage account and the **Blob Containers** folder, and then double-click the **adf-data** blob container you created in a previous lab.
7. In the **Upload** drop-down list, click **Folder**. Then upload the **iislogs** folder as a block blob to a new folder named **iislogs** in the root of the container.

Prepare the Database

The data your pipeline will copy contains daily summaries of the web server log file data. You will copy this data to a database table named **hivelogs**.

Note: The following steps assume that you are using the cross-platform SQL Server command line interface (mssql). You may use an alternative SQL Server client tool if you prefer.

1. In the Azure portal, click **All Resources**, and then click your Azure SQL Database.
2. On the database blade, click **Tools**. Then on the **Tools** blade, click **</> Query editor**. This opens the web-based query interface for your Azure SQL Database.
3. In the toolbar for the query editor, click **Login**, and then log into your database using SQL Server authentication and entering the login name and password you specified when provisioning the Azure SQL Database server.
4. Enter the following command, and then click **Run** to create a table named **dbo.hivelogs**:

```
CREATE TABLE dbo.hivelogs (log_date date, requests int, bytes_in float,
bytes_out float);
```

Exercise 2: Creating Linked Services

The pipeline for your data transformation will use the existing linked services for Azure Blob storage and Azure SQL database that you created in a previous lab. It will also use a new linked service for an on-demand Azure HDInsight cluster, on which the Hive script to transform the data will be run.

Verify Existing Linked Services

You require linked services for the blob store account where the log data is stored, and the Azure SQL Database containing the table you want to load. You should have created these linked services in the previous lab.

1. In the Microsoft Azure portal, browse to the blade for your data factory, and click the **Author and deploy** tile.
2. In the pane on the left, expand **Linked Services** and note that linked services named **blob-store** and **sql-database** are already defined for the blob store and SQL database – these were created by the **Copy Data** wizard in a previous lab.
3. Click the **blob-store** linked service to view its JSON definition, which should look like this:

```
{
 "name": "blob-store",
 "properties": {
  "hubName": "adf_name_hub",
  "type": "AzureStorage",
```

```
  "typeProperties": {
  "connectionString":
"DefaultEndpointsProtocol=https;AccountName=your_store;AccountKey=***"
  }
 }
}
```

4. Click the **sql-database** linked service to view its JSON definition, which should look like this:

```
{
 "name": "sql-database",
 "properties": {
  "hubName": "adf_name_hub",
  "type": "AzureSqlDatabase",
  "typeProperties": {
  "connectionString": "Data
Source=your_server.database.windows.net;Initial
Catalog=DataDB;Integrated Security=False;User
ID=SQLUser;Password=**********;Connect Timeout=30;Encrypt=True"
  }
 }
}
```

Create an HDInsight Linked Service

In addition to the blob store and SQL database linked services, your pipeline will need an HDInsight cluster on which to run the Hive script. You will create an on-demand HDInsight linked service that automatically provisions, uses, and then deletes the cluster each time it is needed.

1. In the **More** menu, click **New compute**, and then click **On demand HDInsight cluster** to create a new JSON document for an on-demand HDInsight cluster.
2. In the new JSON document, replace the default code with the following code, which you can copy and paste from **hdi-cluster.json** in the folder where you extracted the lab files:

```
{
  "name": "hdi-cluster",
  "properties": {
    "type": "HDInsightOnDemand",
    "typeProperties": {
      "version": "3.2",
      "clusterSize": 1,
      "timeToLive": "00:05:00",
      "osType": "Linux",
      "linkedServiceName": "blob-store"
    }
  }
}
```

   This JSON defines an on-demand Linux HDInsight 3.2 cluster with a single node that stores its metadata in the blob storage account defined by the **blob-store** linked service. The cluster will be automatically deleted after 5 minutes of inactivity.

3. Click **Deploy** to deploy the linked service definition to your Azure Data Factory.

Exercise 3: Creating Datasets

The pipeline for your data transformation requires three datasets; one to define the source data, one to define the results of the aggregation transformation that you will implement in Hive, and one to define the destination table in Azure SQL Database.

Create the Source Dataset

The source dataset defines the data in the web server log files in Azure blob storage.

1. In the **More** menu, click **New dataset**, and then click **Azure Blob storage** to create a new JSON document for an Azure Blob store dataset.
2. In the new JSON document, replace the default code with the following code, which you can copy and paste from **hive-iis-log.json** in the folder where you extracted the lab files:

```
{
  "name": "hive-iislog-txt",
  "properties": {
    "structure": [
      {
        "name": "log_date",
        "type": "String"
      },
      {
        "name": "log_time",
        "type": "String"
      },
      {
        "name": "c_ip",
        "type": "String"
      },
      {
        "name": "cs_username",
        "type": "String"
      },
      {
        "name": "s_ip",
        "type": "String"
      },
      {
        "name": "s_port",
        "type": "String"
      },
      {
        "name": "cs_method",
        "type": "String"
      },
      {
        "name": "cs_uri_stem",
        "type": "String"
      },
      {
        "name": "cs_uri_query",
        "type": "String"
      },
      {
```

```
      "name": "sc_status",
      "type": "String"
    },
    {
      "name": "sc_bytes",
      "type": "Int32"
    },
    {
      "name": "cs_bytes",
      "type": "Int32"
    },
    {
      "name": "time_taken",
      "type": "Int32"
    },
    {
      "name": "cs_user_agent",
      "type": "String"
    },
    {
      "name": "cs_referrer",
      "type": "String"
    }
  ],
  "type": "AzureBlob",
  "linkedServiceName": "blob-store",
  "typeProperties": {
    "folderPath": "adf-data/iislogs/{Year}/{Month}/",
    "format": {
      "type": "TextFormat",
      "columnDelimiter": " "
    },
    "partitionedBy": [
      {
        "name": "Year",
        "value": {
          "type": "DateTime",
          "date": "SliceStart",
          "format": "yyyy"
        }
      },
      {
        "name": "Month",
        "value": {
          "type": "DateTime",
          "date": "SliceStart",
          "format": "MM"
        }
      }
    ]
  },
  "availability": {
    "frequency": "Month",
    "interval": 1
  },
```

```
    "external": true,
    "policy": {
      "validation": {
        "minimumSizeMB": 0.01
      }
    }
  }
}
```

This JSON defines a schema for the space-delimited log files in the **iislogs/*Year*/*Month*** folder hierarchy in the **adf-data** container of the Azure storage account represented by your **blob-store** linked service. New data will be available every month.

3. Click **Deploy** to deploy the dataset definition to your Azure Data Factory.

Create a Dataset for the Summarized Data File

The Hive job transforms the source data by aggregating it, and stores the results in a text file in Azure blob storage.

1. In the **More** menu, click **New dataset**, and then click **Azure Blob storage** to create a new JSON document for an Azure Blob store dataset.
2. In the new JSON document, replace the default code with the following code, which you can copy and paste from **hive-summary.json** in the folder where you extracted the lab files:

```
{
  "name": "hive-summary",
  "properties": {
    "structure": [
      {
        "name": "log_date",
        "type": "String"
      },
      {
        "name": "requests",
        "type": "Int64"
      },
      {
        "name": "bytes_in",
        "type": "Decimal"
      },
      {
        "name": "bytes_out",
        "type": "Decimal"
      }
    ],
    "type": "AzureBlob",
    "linkedServiceName": "blob-store",
    "typeProperties": {
      "folderPath": "adf-
data/iislogs/summary/year={Year}/month={Month}/",
      "partitionedBy": [
        {
          "name": "Year",
          "value": {
            "type": "DateTime",
```

```
            "date": "SliceStart",
            "format": "yyyy"
          }
        },
        {
          "name": "Month",
          "value": {
            "type": "DateTime",
            "date": "SliceStart",
            "format": "%M"
          }
        }
      ]
    },
    "availability": {
      "frequency": "Month",
      "interval": 1
    }
  }
}
```

This JSON defines a schema for the files generated by the Hive script. These files are the results of an INSERT operation into a partitioned table, so the folder hierarchy for the table includes subfolders for each of the partitioning keys – in this case, a folder named **year=*YYYY*** for each year containing subfolders named **month=*M*** for each month. (Note that the format code **%M** is used to indicate that the month will be expressed with no leading zero – for example 1, 2, 3, and so on, as generated by Hive when partitioning a table. This contrasts with the **MM** format code used in previous datasets, where the month is expressed using two digits – for example, 01, 02, 03, and so on.)

3. Click **Deploy** to deploy the dataset definition to your Azure Data Factory.

Create the Database Table Dataset

The summarized data is copied to the **dbo.hivelogs** table in Azure SQL Database.

1. In the **More** menu, click **New dataset**, and then click **Azure SQL** to create a new JSON document for an Azure SQL Database dataset.
2. In the new JSON document, replace the default code with the following code, which you can copy and paste from **dbo-hivelogs.json** in the folder where you extracted the lab files:

```
{
    "name": "dbo-hivelogs",
    "properties": {
        "type": "AzureSqlTable",
        "linkedServiceName": "sql-database",
        "structure": [
            {
                "name": "log_date",
                "type": "Datetime"
            },
            {
                "name": "requests",
                "type": "Int32"
            },
```

```
            {
                "name": "bytes_in",
                "type": "Decimal"
            },
            {
                "name": "bytes_out",
                "type": "Decimal"
            }
        ],
        "typeProperties": {
            "tableName": "dbo.hivelogs"
        },
        "availability": {
            "frequency": "Month",
            "interval": 1
        }
    }
}
```

This JSON defines a schema for the **dbo.hivelogs** table you created previously in the database defined by the **sql-database** linked service.

3. Click **Deploy** to deploy the dataset definition to your Azure Data Factory.
4. In the pane on the left, expand the **Datasets** folder and verify that the **hive-iislog-txt**, **hive-ummary**, and **dbo-hivelogs** datasets are listed.

Exercise 4: Creating and Running the Pipeline

Now that you have defined the linked services and datasets for your data flow, you can create a pipeline to encapsulate it.

Create a Pipeline

In this lab, your pipeline will consist of an **HDInsightHive** action to summarize the web server log data, followed by a **Copy** action to copy the summarized results to Azure SQL Database.

1. In the **More** menu, click **New pipeline** to create a new JSON document for a pipeline.
2. In the new JSON document, replace the default code with the following code, which you can copy and paste from **summarize-logs-hive.json** in the folder where you extracted the lab files:

Important: Replace ***<storage_acct>*** with the name of your Azure storage account:

```
{
  "name": "Summarize Logs - Hive",
  "properties": {
    "activities": [
      {
        "type": "HDInsightHive",
        "typeProperties": {
          "scriptPath": "adf-data/iislogs/SummarizeLogs.hql",
          "scriptLinkedService": "blob-store",
          "defines": {
            "log_folder": "wasb://adf-
data@<storage_acct>.blob.core.windows.net/iislogs",
```

```
            "summary_folder": "wasb://adf-
data@<storage_acct>.blob.core.windows.net/iislogs/summary/",
            "year": "$$Text.Format('{0:yyyy}',SliceStart)",
            "month": "$$Text.Format('{0:MM}',SliceStart)"
          }
        },
        "inputs": [
          {
            "name": "hive-iislog-txt"
          }
        ],
        "outputs": [
          {
            "name": "hive-summary"
          }
        ],
        "policy": {
          "timeout": "01:00:00",
          "concurrency": 2,
          "executionPriorityOrder": "OldestFirst",
          "retry": 2
        },
        "scheduler": {
          "frequency": "Month",
          "interval": 1
        },
        "name": "Hive script to summarize logs",
        "linkedServiceName": "hdi-cluster"
      },
      {
        "type": "Copy",
        "typeProperties": {
          "source": {
            "type": "BlobSource",
            "recursive": false
          },
          "sink": {
            "type": "SqlSink",
            "writeBatchSize": 0,
            "writeBatchTimeout": "00:00:00"
          },
          "translator": {
            "type": "TabularTranslator",
            "columnMappings":
"log_date:log_date,requests:requests,bytes_in:bytes_in,bytes_out:bytes_
out"
          }
        },
        "inputs": [
          {
            "name": "hive-summary"
          }
        ],
        "outputs": [
          {
```

```
            "name": "dbo-hivelogs"
          }
        ],
        "policy": {
          "timeout": "01:00:00",
          "concurrency": 2,
          "executionPriorityOrder": "OldestFirst",
          "retry": 2
        },
        "scheduler": {
          "frequency": "Month",
          "interval": 1
        },
        "name": "Copy summarized logs to SQL"
      }
    ],
    "start": "2016-01-01T00:00:00Z",
    "end": "2016-06-01T23:59:59Z",
    "pipelineMode": "Scheduled"
  }
}
```

This JSON defines a pipeline that includes an **HDInsightHive** action to run the Hive script that transforms the **hive-iislog-txt** dataset to the hive-**summary** dataset, and a **Copy** action to copy the **hive-summary** dataset to the **dbo.hivelogs** table every month. Because the input dataset to the second activity is the output dataset from the first activity, the pipeline will start the second activity only after the first activity has completed successfully.

The **HDInsightHive** action defines four parameters that are passed to the Hive script:

- **log_folder**: The path to the **iislogs** folder where the course data is stored.
- **summary_folder**: The path to the folder where the Hive table for the summarized data should be created.
- **year**: The year for the current slice being processed by the pipeline.
- **month**: The month for the current slice being processed by the pipeline.

3. Click **Deploy** to deploy the pipeline to your Azure Data Factory.

View Pipeline Status

Now that you have deployed your pipeline, you can use the Azure portal to monitor its status.

1. After the pipeline has been deployed, return to the blade for your Azure Data Factory, wait a few minutes for the **Pipelines** tile to indicate the addition of your pipeline (it may already indicate the pipelines you created in the previous lab).
2. Click the **Monitor and Manage** tile. This opens a new tab in your browser.
3. View the pipeline diagram, which should include the **Summarize Logs - Hive** pipeline that transfers data from the **hive-iislog-txt** Azure Blob Storage dataset to the **dbo-hivelogs** Azure SQL database dataset.
4. Right-click the **Summarize and Copy Logs** pipeline and click **Open pipeline**. The diagram should now show the **Hive script to summarize logs** and **Copy summarized logs to SQL** actions in the pipeline and the intermediary **hive-summary** Azure Blob Storage dataset.

5. Above the diagram, click the start time and change it to 0:00 on January 1st 2016. Then click the end time and change it to 0:00 on July 1st 2016. Click **Apply** for the filter to take effect.
6. Wait for 30 minutes or so, as the HDInsight cluster is provisioned and the pipeline activities are run. You can refresh the list of activity windows periodically to see the status of each activity window change from **Waiting**, to **In progress**, and then to **Ready**. You can select each activity window in the list to see more details about its status in the pane on the right.

View the Output Generated by the Pipeline

When the first HDInsight Hive actions has finished, you can view the summarized log files that it generates; and after the first of the copy actions has completed, you can verify that data is being copied to your SQL database.

1. In Azure Storage Explorer, view the **iislogs** folder and verify that a **summary** folder has been created (you may need to refresh the view).
2. In the summary folder, verify that a folder named **year=2016** has been created, and that this folder contains folders named **month=*N*** (where *N* is the month number). Each **month=*N*** folder contains text files containing the daily log summary data for that month.
3. Return to the Query editor for your Azure SQL Database and run the following query:

```
SELECT * FROM dbo.hivelogs ORDER BY log_date;
```

4. Verify that the table now contains aggregated log data. This was copied to the table from the summary log data files generated by the Hive script.
5. Keep viewing the summary log data files and querying the **dbo.hivelogs** table as the pipeline actions are processed. Eventually, data for all six months should have been summarized and copied to the database.