Processing Big Data with Hadoop in Azure HDInsight

Lab 4B – Using Oozie

Overview

In this lab, you will use Oozie to define a workflow of data processing operations that summarize data in Internet Information Services (IIS) web server log files and load the results to a table in Azure SQL Database for further analysis.

What You'll Need

To complete the labs, you will need the following:

- A web browser
- A Microsoft account
- A Microsoft Azure subscription
- A Microsoft Windows computer with the following software installed:
    - Microsoft Azure PowerShell
    - Microsoft Power BI Desktop
- 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, installed and configured Azure PowerShell, imported the publisher settings for your Azure subscription into PowerShell, and installed Microsoft Power BI Desktop.

When working with cloud services, transient network errors can occasionally cause scripts to fail. If a script fails, and you believe that you have entered all of the required variables correctly; wait a few minutes and run the script again.

Provisioning HDInsight and Azure SQL Database

In this lab, you will build an Oozie workflow that uses Azure HDInsight to process IIS web server log data, and then transfers the processed data to Azure SQL Database. Before running the workflow, you need to provision the required Azure services.

Provision an Azure Storage Account and HDInsight Cluster

Note: If you already have an HDInsight cluster and associated storage account, you can skip this task.

1. In the C:\HDILabs\Lab04B folder, rename **Provision HDInsight.txt** to **Provision HDInsight.ps1** (you may need to modify the *View* options for the folder to see file extensions). Then right-click **Provision HDInsight.ps1** and click **Run with PowerShell** to run the script (if you are prompted to change the execution policy, enter **Y**).
2. The script takes around 15-20 minutes to complete; so now is a really good time to go and make a cup of coffee! When the script has finished make a note of the details of your HDInsight cluster configuration. Then press ENTER to end the script.

Provision Azure SQL Database

Note: If you already have an Azure SQL Database named **AnalysisDB** in the same region as your HDInsight cluster, you can skip this task.

1. In the C:\HDILabs\Lab04B folder, rename **Provision SQL Database.txt** to **Provision SQL Database.ps1** (you may need to modify the *View* options for the folder to see file extensions). Then right-click **Provision SQL Database.ps1** and click **Run with PowerShell** to run the script (if you are prompted to change the execution policy, enter **Y**).
2. When the script has finished make a note of the details of your Azure SQL Database server name and login credentials. Then press ENTER to end the script.

   Note: The script creates a firewall rule for your Azure SQL Database server, allowing access from any Internet-connected computer. This is not considered a best practice, and should generally not be done for production servers. You can change the firewall rules for your Azure SQL Database server in the Azure portal.

Running an Oozie Workflow

Now that you have the required infrastructure, you are ready to initiate an Oozie workflow to process the log data.

Examine the Oozie Workflow

1. In the C:\HDILabs\Lab04B\oozieworkflow folder, open **workflow.xml** in Visual Studio or Notepad. This XML document defines an Oozie workflow that consists of multiple actions.
2. Note the following element, which directs the workflow to start at the **PrepareHive** action:

```
<start to="PrepareHive"/>
```

3. Review the following element, which defines the **PrepareHive** action:

```
<action name='PrepareHive'>
    <hive xmlns="uri:oozie:hive-action:0.2">
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.job.queue.name</name>
          <value>default</value>
        </property>
      </configuration>
      <script>${DropTableScript}</script>
        <param>CLEANSED_TABLE_NAME=${cleansedTable}</param>
        <param>SUMMARY_TABLE_NAME=${summaryTable}</param>
    </hive>
    <ok to="CleanseData"/>
    <error to="fail"/>
  </action>
```

The **PrepareHive** action is a Hive action that runs the HiveQL script specified in a workflow configuration setting named **DropTableScript**. Two parameters named **CLEANSED_TABLE_NAME** and **SUMMARY_TABLE_NAME** are passed to the script based on the values in the **cleansedTable** and **summaryTable** workflow configuration settings. If the action succeeds, the workflow moves on to the **CleanseData** action.

4. Keep workflow.xml open, and in the C:\HDILabs\Lab04B\oozieworkflow folder, open **DropHiveTables.txt** in Notepad. This is the script that will be specified in the **DropTableScript** workflow configuration setting. Note that it contains HiveQL statements to drop the two tables specified by the parameters if they exist. Then close DropHiveTables.txt without saving any changes.
5. In workflow.xml, review the following element, which defines the **CleanseData** action:

```
<action name="CleanseData">
    <pig>
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <script>${CleanseDataScript}</script>
      <param>StagingFolder=${stagingFolder}</param>
      <param>CleansedFolder=${cleansedFolder}</param>
    </pig>
    <ok to="CreateTables"/>
    <error to="fail"/>
  </action>
```

The **CleanseData** action is a Pig action that runs the Pig Latin script specified in a workflow configuration setting named **CleanseDataScript**. Two parameters named **StagingFolder** and **CleansedFolder** are passed to the script based on the values in the **stagingFolder** and **cleansedFolder** workflow configuration settings. If the action succeeds, the workflow moves on to the **CreateTables** action.

6. Keep workflow.xml open, and in the C:\HDILabs\Lab04B\oozieworkflow folder, open **CleanseData.txt** in Notepad. This is the script that will be specified in the **CleanseDataScript** workflow configuration setting. Note that it contains Pig Latin statements to load the data in the path specified by the **stagingFolder** parameter as a single character array for each row, filter it to remove rows that begin with a "#" character, and store the results in the path defined by the **cleansedFolder** parameter. Then close CleanseData.txt without saving any changes.
7. In workflow.xml, review the following element, which defines the **CreateTables** action:

```
<action name='CreateTables'>
    <hive xmlns="uri:oozie:hive-action:0.2">
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.job.queue.name</name>
          <value>default</value>
        </property>
      </configuration>
      <script>${CreateTableScript}</script>
      <param>CLEANSED_TABLE_NAME=${cleansedTable}</param>
      <param>CLEANSED_TABLE_LOCATION=${cleansedFolder}</param>
      <param>SUMMARY_TABLE_NAME=${summaryTable}</param>
      <param>SUMMARY_TABLE_LOCATION=${summaryFolder}</param>
    </hive>
    <ok to="SummarizeData"/>
```

```
      <error to="fail"/>
    </action>
```

The **CreateTables** action is a Hive action that runs the HiveQL script specified in a workflow configuration setting named **CreateTableScript**. Four parameters named **CLEANSED_TABLE_NAME**, **CLEANSED_TABLE_LOCATION**, **SUMMARY_TABLE_NAME**, and **SUMMARY_TABLE_LOCATION** are passed to the script based on the values in the **cleansedTable**, **cleansedFolder**, **summaryTable**, and **summaryFolder** workflow configuration settings. If the action succeeds, the workflow moves on to the **SummarizeData** action.

8. Keep workflow.xml open, and in the C:\HDILabs\Lab04B\oozieworkflow folder, open **CreateHiveTables.txt** in Notepad. This is the script that will be specified in the **CreateTableScript** workflow configuration setting. Note that it contains HiveQL statements to create Hive tables based on the table name and location parameters. Then close CreateHiveTables.txt without saving any changes.
9. In workflow.xml, review the following element, which defines the **SummarizeData** action:

```
<action name='SummarizeData'>
    <hive xmlns="uri:oozie:hive-action:0.2">
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.job.queue.name</name>
          <value>default</value>
        </property>
      </configuration>
      <script>${SummarizeDataScript}</script>
      <param>CLEANSED_TABLE_NAME=${cleansedTable}</param>
      <param>SUMMARY_TABLE_NAME=${summaryTable}</param>
    </hive>
    <ok to="TransferData"/>
    <error to="fail"/>
  </action>
```

The **SummarizeData** action is a Hive action that runs the HiveQL script specified in a workflow configuration setting named **SummarizeDataScript**. Two parameters named **CLEANSED_TABLE_NAME** and **SUMMARY_TABLE_NAME** are passed to the script based on the values in the **cleansedTable** and **summaryTable** workflow configuration settings. If the action succeeds, the workflow moves on to the **TransferData** action.

10. Keep workflow.xml open, and in the C:\HDILabs\Lab04B\oozieworkflow folder, open **SummarizeData.txt** in Notepad. This is the script that will be specified in the **SummarizeDataScript** workflow configuration setting. Note that it contains HiveQL statements to insert data into the table specified by the **SUMMARY_TABLE_NAME** parameter based on the results of a query from the table specified by the **CLEANSED_TABLE_NAME** parameter. Then close SummarizeData.txt without saving any changes.
11. In workflow.xml, review the following element, which defines the **TransferData** action:

```
<action name="TransferData">
    <sqoop xmlns="uri:oozie:sqoop-action:0.2">
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <configuration>
        <property>
          <name>mapred.compress.map.output</name>
```

```
            <value>true</value>
          </property>
        </configuration>
        <arg>export</arg>
        <arg>--connect</arg>
        <arg>${sqlConnectionString}</arg>
        <arg>--table</arg>
        <arg>${sqlTable}</arg>
        <arg>--export-dir</arg>
        <arg>${sourceDir}</arg>
        <arg>--input-fields-terminated-by</arg>
        <arg>\t</arg>
      </sqoop>
      <ok to="end"/>
      <error to="fail"/>
    </action>
```

The **TransferData** action is a Sqoop action that exports data from the storage location specified by the **sourceDir** workflow configuration setting to a table in a database specified by the **sqlTable** and **sqlConnection** workflow configuration settings. If the action succeeds, the workflow moves on to the **end** terminator.

12. In workflow.xml, review the **kill** and **end** elements. These elements are terminators for the workflow. If all actions succeed, the workflow ends with the end terminator. If an action fails, the workflow is redirected to the fail terminator and the most recent error is reported.
13. Close workflow.xml without saving any changes.

Run the Oozie Workflow

1. In the C:\HDILabs\Lab04B folder, rename **Run Oozie Workflow.txt** to **Run Oozie Workflow.ps1** (you may need to modify the *View* options for the folder to see file extensions). Then right-click **Run Oozie Workflow.ps1** and click **Edit** to open the script in the Windows PowerShell interactive script environment (ISE).
2. Change the values assigned to the **$clusterName** and **$hdPassword** variables to match the name of your HDInsight cluster and the password for your HDInsight HTTP user.
   Note: If you did not use the script provided to provision your cluster, you may also need to change the **$storageAccountName**, **$containerName**, and **$hdUser** variables to match the names of your storage account, container, and HTTP user.
3. Change the values assigned to the following variables based on the configuration of your Azure SQL Database server:
   - **$sqlServer**: *Your Azure SQL server name*
   - **$sqlLogin**: *Your SQL login name* – this will be *SQLUser* if you used the script provided to provision your Azure SQL Database
   - **$sqlPassword:** *Your SQL Login Password* (note that you must prefix reserved PowerShell characters such as **$** and **#** with a ` character, so the value *SecurePa`$`$w0rd* assigns the password *SecurePa$$w0rd*).
4. Save the script and then review the rest of the code, noting that it performs the following actions:
   a. Prepares the Azure SQL Database by dropping (if necessary) and creating a table named **logdata**, to which the summarized log data generated by the workflow will be transferred.
   b. Uploads the files in the local **iislogs_gz** subfolder to **/data/iislogs/stagedlogs** in your Azure storage container.
   c. Uploads the files in the local **oozieworkflow** subfolder to **/data/iislogs/oozieworkflow** in your Azure storage container.

   d. Defines the Oozie workflow configuration settings as an XML structure, specifying the values that will be passed to the workflow.
   e. Initiates an Oozie job using the HTTP REST interface.
   f. Waits for the job to complete, displaying the job status every 30 seconds.
5. On the toolbar, click **Run Script**, and wait for the script to finish (which can take several minutes), observing the status information displayed in the console window.
6. When the script has completed, close Windows PowerShell ISE.

Access the Processed Data in Azure SQL Database

1. Start Power BI Desktop and close the welcome page if it opens.

   Note: Power BI Desktop is the released version of the Power BI Designer preview tool used in the demonstrations for this course. The tool has been renamed and updated, and looks cosmetically different from the preview version; but still provides the same functionality as shown in the demonstrations.

2. On the **Home** tab, in the **Get Data** list, click **SQL Server**.
3. In the **Microsoft SQL Database** page, in the **Server** box, type the fully-qualified name of your Azure SQL Database server in the format *your_azure_sql_server_name*.database.windows.net, and in the Database box, type **AnalysisDB**. Then click **OK**.
4. If the **Access a Microsoft SQL Database** dialog box is displayed, view the **Database** page; and in the **Username** box, type your Azure SQL Database login name, in the **Password** box type your Azure SQL Database login password, and click **Connect**.
5. In the **Navigator** window, select the **logdata** table. Then click **Load**.
6. In the **Visualizations** pane, select **Line and Stacked Columns Chart**.
7. In the **Fields** pane, expand the **logdata** table, and select **InboundBytes** and **OutboundBytes**. Then drag **log_date** to the **Shared Axis** area in the **Visualizations** pane, and drag **Requests** to the **Line Values** area in the **Visualizations** pane.
8. View the chart, which shows the summarized web server log data that was generated and transferred to Azure SQL Database by the Oozie workflow.
9. Close Power BI Desktop without saving the report.

Cleaning Up

Now that you have finished this lab, you can delete the HDInsight cluster and storage account.

Delete the HDInsight Cluster and Azure SQL Database Server

If you no longer need the HDInsight cluster and Azure SQL Database server used in this lab, you should delete them to avoid incurring unnecessary costs (or using credits in a free trial subscription). If you used the scripts provided in this lab to provision these services, you can use this procedure to delete them automatically.

1. In the C:\HDILabs\Lab04B folder, rename **Delete HDInsight and SQL DB.txt** to **Delete HDInsight and SQL DB.ps1** (you may need to modify the *View* options for the folder to see file extensions). Then right-click **Delete HDInsight and SQL DB.ps1** and click **Run with PowerShell** to run the script (if you are prompted to change the execution policy, enter **Y**).
2. When prompted, enter the name of your HDInsight cluster and the name of your Azure SQL Database server.
3. When the script finishes (usually after a few minutes), press ENTER to end the script.