edX Study Lab4

(이삭엔지니어링 인턴 김형근 : hkkim@isaac-eng.com)

Lab 4 – Orchestrating Big Data Workflows
(문서 내용 중 옥색으로 칠한 부분은, 기존의 문서에서 CDH 환경에 맞도록 내용이 변경된
부분이다. CDH 5.15에서 실습하였음.)

Overview

Often, you will want to combine Hive, Pig, and other Hadoop jobs in a workflow. Oozie is an orchestration
engine that you can use to define a workflow of data processing actions. In this lab, you will create and
run an Oozie workflow to process web server log data.

You will then use Sqoop to transfer the processed log data to Azure SQL Database. HDInsight provides a
powerful platform for transforming and cleansing data; but while Hive offers a SQL-like interface through
which to query data, many organizations prefer to store data in a relational database from where it can be
accessed by applications and users who need to query it. Sqoop provides support for bi-directional data
transfer between a Hadoop data store (in the case of HDInsight, Azure blob storage) and a range of
databases that can be accessed using JDBC.

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, Linux, or Apple Mac OS X computer on which the Azure Storage Explorer
  and a SQL Server command line tool have been installed.
- 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.

Running an Oozie Workflow

Oozie provides a way to orchestrate a series of actions in a workflow. In this exercise, you will use Oozie to
coordinate Pig and Hive jobs to cleanse and summarize web server log data.

View the Source Data

1. Use a text editor to view the **log.txt** file in the **HDILabs\Lab04\data\iislogs\src** folder where you
   extracted the lab files for this course. Note that the file contains IIS web server log records.
2. Close the file without saving any changes.

Examine the Oozie Workflow

1. In the **HDILabs\Lab04\data\iislogs\oozie** folder, open **workflow.xml** in a text editor (or an application that can display XML). 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>
      <job-xml>hive-site.xml</job-xml> //진짜 해결법은 찾지못함
      <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 **HDILabs\Lab04\data\iislogs\oozie** folder, open **DropHiveTables.txt** in a text editor. 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 **HDILabs\Lab04\data\iislogs\oozie** 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>
      <job-xml>hive-site.xml</job-xml> //진짜 해결법은 찾지못함
      <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 **HDILabs\Lab04\data\iislogs\oozie** 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>
      <job-xml>hive-site.xml</job-xml> //진짜 해결법은 찾지못함
      <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="end"/>
    <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 end.

10. Keep workflow.xml open, and in the **HDILabs\Lab04\iislogs\oozie** 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 **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.

12. Close workflow.xml without saving any changes.

* workflow.xml에서 'SummarizeData'가 작동하지 않아서 아래 구문을 Hue HiveQuery 인터페이스에 직접 집어넣었음

```
SummarizeDataScript = SummarizeData.txt
```

INSERT OVERWRITE TABLE edx_hkkim.summarizedlogs
SELECT log_date, COUNT(*), SUM(sc_bytes), SUM(cs_bytes)
FROM edx_hkkim.cleansedlogs
GROUP BY log_date
ORDER BY log_date ASC;

Edit Oozie Job Properties

1. In the **HDILabs\Lab04\data** folder, open **job.properties** in a text editor and note that this file contains the configuration properties for the Oozie workflow. These properties include the values for the parameters in the workflow.xml file, such as file paths and the names of script files as shown here:

```
nameNode=hdfs://master1.isaac-eng.com:8020

jobTracker=master1.isaac-eng.com:8032
queueName=default
oozie.use.system.libpath=true
DropTableScript=DropHiveTables.txt
CleanseDataScript=CleanseData.txt
CreateTableScript=CreateHiveTables.txt
SummarizeDataScript=SummarizeData.txt

stagingFolder=hdfs://iseHA1/user/hkkim/data/i
islogs/src

cleansedTable=edx_hkkim.cleansedlogs

cleansedFolder=hdfs://iseHA1/user/hkkim/data/
iislogs/cleansed

summaryTable= edx_hkkim.summarizedlogs

summaryFolder=hdfs://iseHA1/user/hkkim/data/ii
slogs/summarized

user.name=hkkim

oozie.wf.application.path=hdfs://iseHA1/user/
hkkim/data/iislogs/oozie
```

2. Edit job.properties, updating the following properties to reflect your settings:
    - **nameNode**: Edit the URL to reflect the blob container and storage account used by your cluster.
    - **user.name**: Specify the SSH user name for your cluster.
3. Save the changes to job.properties and close the text editor.

Upload Files to HDFS

Now you are ready to upload the source data and workflow files to Azure storage. The instructions here assume you will use Azure Storage Explorer to do this, but you can use any Azure Storage tool you prefer.

Upload the **HDILabs\Lab04\data***all_things* to /user/hkkim/data folder

Connect to the Cluster

connect 10.100.3.210 (master1.isaac-eng.com)

View the Uploaded Files

Now that you have a remote console for your cluster, you can verify that the various files for your workflow have been uploaded to the shared cluster storage.

1. In the SSH console window for your cluster, enter the following command to view the contents of the **/data** folder in the HDFS file system.

```
hdfs dfs –ls /user/hkkim/data
```

2. Verify that the folder contains the **job.properties** file you uploaded.
3. In the SSH console window for your cluster, enter the following command to view the contents of the **/data/iislogs** folder in the HDFS file system.

```
hdfs dfs –ls /user/hkkim/data/iislogs
```

4. Verify that the folder contains folders named **src** and **oozie**.
5. In the SSH console window for your cluster, enter the following command to view the contents of the **/data/iislogs/src** folder in the HDFS file system.

```
hdfs dfs –ls /user/hkkim/data/iislogs/src
```

6. Verify that the folder contains the **log.txt** file you uploaded.
7. In the SSH console window for your cluster, enter the following command to view the contents of the **/data/iislogs/oozie** folder in the HDFS file system.

```
hdfs dfs –ls /user/hkkim/data/iislogs/oozie
```

8. Verify that the folder contains the **workflow.xml** file you uploaded, along with the various text files containing Pig and Hive scripts.

Run the Oozie workflow

1. In the SSH console window for your cluster, enter the following command to download the job.properties file from shared storage to the local file system on the cluster head node:

   ```
   hdfs dfs -get /user/hkkim/data/job.properties
   ```

2. Enter the following command to start the Oozie workflow.

   ```
   oozie job -oozie http://localhost:11000/oozie -config job.properties -
   run
   ```

3. Wait for the Oozie job to start, and then note that its ID is displayed (in the form 0000000-123456789012345-oozie-hdp-W).

4. Enter the following command, replacing *JobID* with the job ID for your Oozie job.

   ```
   oozie job -oozie http://localhost:11000/oozie -info JobID
   ```

5. View the status of the Oozie job while it is running. The job may take several minutes to complete. Re-enter the command every minute or so to observe each action in the workflow run.

-> SummarizeData Workflow만 Hue에서 직접 진행

6. When the last action (*SummarizeData*) has completed, enter the following command to verify that the data for the summary Hive table has been stored in a file named **000000_0** in the **/data/iislogs/summarized** folder:

```
hdfs dfs –ls /user/hkkim/data/iislogs/summarized
```

7. Enter the following command to view the summarized data produced by the Oozie workflow:

```
hdfs dfs -text /user/hkkim/data/iislogs/summarized/000000_0
```

Transfer Data to Azure SQL Database

Now that you have processed the web server log data, you are ready to use Sqoop to transfer the results from HDInsight to Azure SQL Database.

Provision Azure SQL Database

1. In the Microsoft Azure portal, in the menu, click **New**. Then in the **Databases** menu, click **SQL Database**.
2. In the **SQL Database** blade, enter the following settings, and then click **Create**:
   - **Name**: DataDB
   - **Subscription**: *Select your Azure subscription*
   - **Resource Group**: *Select the resource group you created previously*
   - **Select source**: Blank database
   - **Server:** *Create a new server with the following settings*:
     - **Server name**: *Enter a unique name (and make a note of it!)*
     - **Server admin login:** *Enter a user name of your choice (and make a note of it!)*
     - **Password:** *Enter and confirm a strong password (and make a note of it!)*
     - **Region:** *Select your HDInsight cluster location*
     - **Allow azure services to access server**: Selected
   - **Pricing tier:** *View all and select* Basic
   - **Collation**: SQL_Latin1_General_CP1_CI_AS
   - **Pin to dashboard:** Unselected
3. In the Azure portal, view **Notifications** to verify that deployment has started. Then wait for the SQL database to be deployed (this can take a few minutes.)
4. After the database has been created, browse to your Azure SQL server (not the database) and under **Settings**, click **Properties**.
5. Note the fully qualified name of your server (which should take the form *server*.database.windows.net, where *server* is the server name you specified earlier) and the server admin user name (which should be the login you specified earlier).
6. Under **Settings**, click **Firewall**, and in the **Firewall** blade, note that your client IP address has been automatically detected.
7. Click **Add client IP** to create a rule that permits access to the server from your local computer. Then click **Save** to save the rule. When the firewall rule change is confirmed, click **OK**.

> **Note**: If your client IP address changes, you will need to edit this rule to restore access from your client computer's new address. In some cases, your computer's IP address may be abstracted behind a local firewall or router. If you experience errors when trying to connect to your Azure SQL server in subsequent procedures, try creating a firewall rule that permits access to the IP address range **0.0.0.0** to **255.255.255.255** (this permits access from any Internet-connected device, so you should generally not do this for production servers!). If this still does not resolve the problem, your local firewall may be blocking outbound connections on port 1433 –refer to the documentation for your firewall product to resolve this issue.

Start Oracle Database

1. When the SQL Server command line prompt is displayed, enter the username(*selectonly*) and password(*ise1212*) and type the following SQL statement to create a table (you can copy and paste this code from **Create Logdata Table.txt** in the **HDILabs\Lab04** folder.)

    Note: If you are using the SQL CLI tool, you must enter the command on a single line:

    ```
    CREATE TABLE logdata(log_date date, Requests int, InboundBytes int,
    OutboundBytes int);
    ```

2. Keep the SQL Server client tool open. You will return to it in a later procedure.

Transfer the Data to Oracle Database

Now you're ready to use Sqoop to transfer the tab-delimited weather data in the shared storage for your CDH cluster to Oracle Database.

1. In the SSH console for your cluster, enter the following command.
    You can copy this command from **Sqoop Command.txt** in the **HDILabs\Lab04**
    folder. (You must connect to 10.100.3.211. 10.100.3.211 has Sqoop.)

    ```
    sqoop export
    --connect jdbc:oracle:thin:@10.100.3.152:1521:dwd
    --username selectonly --password ise1212
    --direct
    --export-dir /user/hkkim/data/iislogs/summarized --table LOGDATA
    --fields-terminated-by '\t'
    ```

2. Wait for the Sqoop job to complete. This may take some time.
3. In the SQL Server client application, enter the following Transact-SQL statement to verify that the data has been loaded into the weather table you created earlier:

    ```
    SELECT * FROM selectonly.logdata;
    ```

4. Close the SQL Server tool. To exit Oracle, enter the following command:

    ```
    .quit
    ```

5. Close all open command windows and applications.

    Reference : https://github.com/MicrosoftLearning/Processing-Big-Data-with-Hadoop-in-Azure-HDInsight/blob/master/Labs/Lab%204%20-%20Orchestrating%20Big%20Data%20Workflows.pdf