Processing Big Data with Hadoop in Azure HDInsight

Lab 4 – Orchestrating Big Data Workflows

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 CLI 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.

Provisioning an Azure HDInsight Cluster

Before you can process data with HDInsight, you will need an HDInsight cluster.

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. Use the steps provided in Lab 1 to provision a Hadoop HDInsight cluster on Linux and an associated storage account.
2. Make a note of the details of your HDInsight cluster configuration.

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** 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\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>
      <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\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\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>
      <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\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>
      <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\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.

Edit Oozie Job Properties

1. In the **HDILabs\Lab04** 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=wasb://<container>@<storage_account>.blob.core.windows.net
jobTracker=jobtrackerhost:9010
queueName=default
oozie.use.system.libpath=true
DropTableScript=DropHiveTables.txt
CleanseDataScript=CleanseData.txt
CreateTableScript=CreateHiveTables.txt
SummarizeDataScript=SummarizeData.txt
stagingFolder=wasb:///data/iislogs/src
cleansedTable=cleansedlogs
cleansedFolder=wasb:///data/iislogs/cleansed
summaryTable=summarizedlogs
summaryFolder=wasb:///data/iislogs/summarized
user.name=<user_name>
oozie.wf.application.path=wasb:///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 Azure Storage

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

1. Open a command window, and enter the following command to switch the Azure CLI to *resource manager* mode.

   ```
   azure config mode arm
   ```

2. Enter the following command to log into your Azure subscription:

   ```
   azure login
   ```

3. Follow the instructions that are displayed to browse to the Azure device login site and enter the authentication code provided. Then sign into your Azure subscription using your Microsoft account.
4. Enter the following command to view your Azure resources:

   ```
   azure resource list
   ```

5. Verify that your HDInsight cluster and the related storage account are both listed. Note that the information provided includes the resource group name as well as the individual resource names.
6. Enter the following command on a single line to determine the connection string for your Azure storage account, specifying the storage account and resource group names you noted earlier:

   ```
   azure storage account connectionstring show storage_account -g
   resource_group
   ```

7. Note the connection string, copying it to the clipboard if your command line tool supports it.
8. If you are working on a Windows client computer, enter the following command to set a system variable for the connection string:

   ```
   SET AZURE_STORAGE_CONNECTION_STRING=your_connection_string
   ```

   If you are using a Linux or Mac OS X client computer, enter the following command to set a system variable for the connection string (enter the connection string in quotation marks):

   ```
   export AZURE_STORAGE_CONNECTION_STRING="your_connection_string"
   ```

9. Enter the following command to upload the **log.txt** file to a blob named **data/iislogs/src/log.txt** in the container used by your HDInsight cluster; replacing *log* with the local path to log.txt (for example *c:\HDILabs\Lab04\log.txt* or *HDILabs/Lab04/log.txt*) and *container* with the name of the default storage container used by your cluster:

   ```
   azure storage blob upload log container data/iislogs/src/log.txt
   ```

10. Enter the following command to upload the **job.properties** file to a blob named **data/iislogs/job.properties** in the container used by your HDInsight cluster; replacing *job* with the local path to job.properties (for example *c:\HDILabs\Lab04\ job.properties* or

HDILabs/Lab04/ job.properties) and *container* with the name of the default storage container used by your cluster:

```
azure storage blob upload job container data/iislogs/job.properties
```

11. Wait for the file to be uploaded.
12. Enter the following command to upload the **workflow.xml** file to a blob named **data/iislogs/oozie/workflow.xml** in the container used by your HDInsight cluster; replacing *xml* with the local path to job.properties (for example *c:\HDILabs\Lab04\oozie\workflow.xml* or *HDILabs/Lab04/*oozie/*workflow.xml)* and *container* with the name of the default storage container used by your cluster:

```
azure storage blob upload xml container data/iislogs/oozie/workflow.xml
```

13. Wait for the file to be uploaded.
14. Repeat the previous two steps to upload the following files in the local **oozie** folder to blobs named **data/iislogs/oozie/*file_name*** (where *file_name* is the name of the file) in the container used by your cluster:
    - CleanseData.txt
    - CreateHiveTables.txt
    - DropHiveTables.txt
    - SummarizeData.txt
15. Keep the command window open, you will return to it later in the lab.

Connect to the Cluster

To run the Oozie job, you will need to open an SSH console that is connected to your cluster:

If you are using a Windows client computer:

1. In the Microsoft Azure portal, on the **HDInsight Cluster** blade for your HDInsight cluster, click **Secure Shell**, and then in the **Secure Shell** blade, under **Windows users**, copy the **Host name** (which should be ***your_cluster_name*-ssh.azurehdinsight.net**) to the clipboard.
2. Open PuTTY, and in the **Session** page, paste the host name into the **Host Name** box. Then under **Connection type**, select **SSH** and click **Open**. If a security warning that the host certificate cannot be verified is displayed, click **Yes** to continue.
3. When prompted, enter the SSH username and password you specified when provisioning the cluster (not the cluster login).

If you are using a Mac OS X or Linux client computer:

1. In the Microsoft Azure portal, on the **HDInsight Cluster** blade for your HDInsight cluster, click **Secure Shell**, and then in the **Secure Shell** blade, under **Linux, Unix, and OS X users**, note the command used to connect to the head node.
2. Open a new terminal session, and enter the appropriate command to connect, specifying your SSH user name (not the cluster login) and cluster name as necessary:

```
ssh your_ssh_user_name@your_cluster_name-ssh.azurehdinsight.net
```

3. If you are prompted to connect even though the certificate can't be verified, enter **yes**.
4. When prompted, enter the password for the SSH user name.

Note: If you have previously connected to a cluster with the same name, the certificate for the older cluster will still be stored and a connection may be denied because the new certificate does not match the stored certificate. You can delete the old certificate by using the **ssh-keygen** command, specifying the path of your certificate file (**f**) and the host record to be removed (**R**) - for example:

```
ssh-keygen -f "/home/usr/.ssh/known_hosts" -R clstr-ssh.azurehdinsight.net
```

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/iislogs** folder in the HDFS file system.

   ```
   hdfs dfs -ls /data/iislogs
   ```

2. Verify that the folder contains the **job.properties** file you uploaded, and folders named **src** and **oozie**.
3. In the 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 /data/iislogs/src
   ```

4. Verify that the folder contains the **log.txt** file you uploaded.
5. In the 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 /data/iislogs/oozie
   ```

6. 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 /data/iislogs/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.

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 /data/iislogs/summarized
```

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

```
hdfs dfs -text /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, on the hub menu, click **All resources**, then note the **Location** of your HDInsight cluster.
2. On the hub menu, click **SQL databases**. Then click **Add**, and use the **SQL Database** blade to create a new database with the following settings:
    - **Name**: DataDB
    - **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*
        - **Create V12 server (Latest update)**: Yes
        - **Allow azure services to access server**: Selected
    - **Select source:** Blank database
    - **Pricing tier:** *View all and select* Basic
    - **Optional configuration:** *Leave the default settings*
    - **Resource group:** *Select the existing resource group containing your HDInsight cluster*
    - **Subscription:** *Verify that your Azure subscription is selected*
    - **Pin to dashboard:** Unselected
3. After you have clicked **Create**, wait for the database to be created. Then view all resources and click your SQL Database server.
4. On the **Settings** blade for your server, click **Firewall**; and note that your client IP address has been automatically detected.
5. Create a rule named **My IP**, using your client IP address as the **Start IP** and **End IP** value. Then click **Save** to save the rule (note that if your client IP address changes, you will need to edit this rule to restore access to your database server from your client computer). When the firewall rule change is confirmed, click **OK**.

Create a Table in the Database

1. In the command line console for your local computer, enter the appropriate command to start the SQL Server client interface and connect to your server:

    If you are using a Windows client computer with the SQL Server client utilities installed, enter the following (case-sensitive) command, replacing *server* with your Azure SQL Database Server name, *login* with your server admin login name, and *password* with your server login password.

    ```
    sqlcmd -S server.database.windows.net -U login -P password -d DataDB
    ```

If you are using a Mac or Linux computer with the cross-platform SQL Server command line interface installed, enter the following (case-sensitive) command, replacing *server* with your Azure SQL Database Server name, *login* with your server admin login name, and *password* with your server login password. Prefix any special characters (such as **$**) with a **** character:

```
mssql -s server.database.windows.net -u login -p password -d DataDB -e
```

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

   If you are using *sqlcmd* on a Windows client computer, enter the following commands:

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

   If you are using *mssql* on a Mac or Linux client computer, enter the following command on a single line:

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

3. Exit the SQL Server command line by entering the following command:

   To exit *sqlcmd* on a Windows client computer:

   ```
   quit
   ```

   To exit *mssql* on a Mac or Linux client computer:

   ```
   .quit
   ```

4. Keep the command window open. You will return to it in a later procedure.

Transfer the Data to Azure SQL Database

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

1. In the SSH console for your cluster, enter the following command on a single line to run a Sqoop job, replacing all instances of *server* with the name of your Azure SQL Database server, *login* with your SQL login name, and *password* with your SQL login password (note that the login is expressed as <*login*>@<*server*>). You can copy this command from **Sqoop Command.txt** in the **HDILabs\Lab04** folder

   ```
   sqoop export
   --connect "jdbc:sqlserver://server.database.windows.net;
              username=login@server;password=password;database=DataDB"
   --table weather
   --export-dir /data/output
   --input-fields-terminated-by \\t
   ```

2. Wait for the Sqoop job to complete. This may take some time.
3. In the command window for your local computer, enter the appropriate command to start the SQL Server client interface and connect to your server:

   If you are using a Windows client computer with the SQL Server client utilities installed, enter the following (case-sensitive) command, replacing *server* with your Azure SQL Database Server name, *login* with your server admin login name, and *password* with your server login password.

   ```
   sqlcmd -S server.database.windows.net -U login -P password -d DataDB
   ```

   If you are using a Mac or Linux computer with the cross-platform SQL Server command line interface installed, enter the following (case-sensitive) command, replacing *server* with your Azure SQL Database Server name, *login* with your server admin login name, and *password* with your server login password. Prefix any special characters (such as **$**) with a **** character:

   ```
   mssql -s server.database.windows.net -u login -p password -d DataDB -e
   ```

4. When the SQL Server command line prompt is displayed, enter the following Transact-SQL statement to verify that the data has been loaded into the weather table you created earlier:

   If you are using *sqlcmd* on a Windows client computer, enter the following commands:

   ```
   SELECT * FROM weather;
   GO
   ```

   If you are using *mssql* on a Mac or Linux client computer, enter the following command on a single line:

   ```
   SELECT * FROM weather;
   ```

5. Exit the SQL Server command line by entering the following command:

   To exit *sqlcmd* on a Windows client computer:

   ```
   quit
   ```

   To exit *mssql* on a Mac or Linux client computer:

   ```
   .quit
   ```

6. Close all open command windows.

Cleaning Up

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

Note: If you are proceeding straight to the next lab, omit this task and use the same cluster and SQL database in the next lab. Otherwise, follow the steps below to delete your 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).

1. In the Microsoft Azure portal, click **Resource Groups**.

2. On the **Resource groups** blade, click the resource group that contains your HDInsight cluster, and then on the **Resource group** blade, click **Delete**. On the confirmation blade, type the name of your resource group, and click **Delete**.
3. Wait for your resource group to be deleted, and then click **All Resources**, and verify that the cluster, the storage account that was created with your cluster, and your SQL server and database, have all been removed.
4. Close the browser.