IST769 Lab C

# Hadoop

In this lab we will learn about Hadoop. Specifically, HDFS, YARN, Hive SQL on Hadoop, and Spark.

### Learning Outcomes

At the end of this lab, you should be able to:

* Load data into HDFS from the command line,
* Create internal and external Hive tables,
* Create a Spark session configured to integrate with Hive.
* Execute Hive commands in PySpark

### Pre-Requisites

Before you begin:

* Open a terminal window in the lab environment
* Set the current working directory to **advanced-databases**
* Start the following services required by Hadoop: **jupyter namenode datanode hive-server hive-metastore hive-metastore-postgresql**

### Tools Used In this Lab

The following tools will be used in this lab:

1. To access the Linux shell where the Hadoop and Beeline clients are installed, we must connect to the running instance of the **hive-server** service. Type the following at the PowerShell Prompt:  
   PS:> docker-compose exec hive-server bash
2. To see a list of Hadoop commands, from the Linux shell, type:  
   root@hive-server: # hadoop fs
3. To connect the Hive beeline client from the Linux shell, type:  
   root@hive-server: # beeline -u jdbc:hive2://hive-server:10000/default
4. To access HFDS over the web from the windows host:  
   <http://localhost:50070>
5. To access Jupyter Lab from your Windows host:  
   <http://localhost:8888>   
   The password is **SU2orange!**

# Lab Problem Set

Try to complete the following exercises. Refer to your notes, and the class lecture video to help you if you get stuck! You will struggle, but also learn a lot as you complete these exercises!  
Answer each of the following using the Problem Set Submission Form, included with this lab.

1. Connect to the Linux shell on the **hive-server** (this is where the Hadoop client has been installed for you.) On this server you will see the **/datasets** folder is mounted. Load the:
   1. customers/customers.csv,
   2. customers/surveys.csv, and
   3. tweets/tweets.psv into HDFS.

Specifically:

Source HDFS Location

customers/customers.csv /user/root/labc/customers/customers.csv

customers/surveys.csv /user/root/labc/surveys/surveys.csv

tweets/tweets.psv /user/root/labc/tweets/tweets.psv

Record the Hadoop commands you entered to complete this task. provide a screenshot of evidence these files are in HDFS. The screen shot can use the Hadoop client output or the HDFS website.  
List of Commands used:

To create directories:

!sh hadoop fs -mkdir /user/root/labc/customers/customers

!sh hadoop fs -mkdir /user/root/labc/surveys

!sh hadoop fs -mkdir /user/root/labc/tweets

To load the data into HDFS:

hadoop fs -put /datasets/customers/customers.csv /user/root/labc/customers/customers.csv

hadoop fs -put /datasets/customers/surveys.csv /user/root/labc/surveys/surveys.csv

hadoop fs -put /datasets/tweets/tweets.psv /user/root/labc/tweets/tweets.psv

**Screenshots of HDFS website:**

1. Create a Hive database called **labc**. In the **labc** database create an external hive table for the **tweets**. Your external table will point to the existing location on HDFS.   
   **NOTE:** You will need to view the tweets.psv file to see the format of the file before you can create the table schema correctly.

After you create the table write a SELECT query to display all of the tweets for a user a single user of your choice. Please include the HQL code you wrote to create and query the **tweets** table. Along with screenshots of a **describe tweets** command output along with your SELECT query output.

1. In the **labc** database, let’s create an internal hive table for **customers**. After you create the table, use the LOAD command to move the data from the current HDFS location into the Hive data warehouse.

**NOTE 1:** if you screw up you will need to drop table and reload the file back into HDFS from step 1.

**NOTE 2:** there is a header row in this file, you might need to search the Hive docs on the web for how to exclude this first row.

When you have created the table and imported the data, provide the HQL code you entered to complete the task and provide screenshots of the **describe customers** command, a SELECT output to show data is there, and a screenshot on Web HDFS to show the data is located in **/user/hive/warehouse**.

1. Like the previous step, import the surveys.csv into a Hive internal table in the **labc** database called **surveys**. When you have created the table and imported the data, provide all the commands you entered to complete the task, a screenshots of the table description, the select statement output, and Web HDFS location.
2. Open Jupyter Lab. Create a new notebook called **labc.** Copy over the code from an example to create a spark session connected to Hive.   
     
   In a separate cell, write Spark SQL code to join customers to surveys on email address. Include all rows and columns and show output in the notebook. Provide a screenshot of the notebook cell with a reasonable amount of output (doesn’t need to be the entire set of rows and columns as that will be too large).
3. The marketing department would like a dataset of customers / surveys for analysis. In a separate cell in the **labc** Juypter Notebook, write a Spark SQL query to create a hive table called **marketing** in **AVRO** file format from a SELECT query that once again joins customers and surveys on email addresses. Include the following columns in the new table: **Household Income**, **Education**, **Marital Status**, **Gender**, **City** and **State**.  
     
   Provide a screenshot of the Jupyter cell and output that creates the new table, and another of the cell and output of executing a SELECT on the table.
4. Stupid marketing doesn’t know what they want! Now they would like the same query in the previous step, only output as a Comma-Delimited file instead of a Hive table. In a new Jupyter Lab cell, write Spark SQL to execute the Hive query but save the output back to HDFS in the folder **/user/root/marketing**.  
     
   Provide a screenshot of the Spark code cell and its output, as well as a screenshot of the file on Web HDFS.

**IMPORTANT:** When you are finished with the lab, execute:

PS:> docker-compose stop

To turn off all running services, then shut down your Azure Lab instance.