Sqoop Exercise

Apache Sqoop(TM) is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases.

Main Website:

* <https://sqoop.apache.org>

Documentation:

* <https://sqoop.apache.org/docs/1.4.0-incubating/SqoopUserGuide.html>

Exercise Requirements

* Cloudera Quickstart VM is Setup
* Access to the MySQL DB on the Quickstart VM
  + On the Cloudera Quickstart VM you can access the local MySQL with the following command:

$ mysql -u root -p

Password: cloudera

Disclaimer:

* This exercise doesn’t show all the possible functionality with Sqoop. It is simply here to give a high level introduction into using sqoop. Like with every big data technology, there is a lot more to learn.

Use Case

For the exercise let’s pick a practical example. Say we want to get demographic information on customer info from a MySQL Database.

Load Data into MySQL

1. Download the mysqlsampledatabase.zip file and unzip it. This will create the mysqlsampledatabase.sql file.
   1. Original data was found at: <http://www.mysqltutorial.org/mysql-sample-database.aspx>

File can be found at: <https://drive.google.com/open?id=1TljwnmecqiavU39nymOlGSX56QnKU7c6>

$ unzip mysqlsampledatabase.zip

1. Import the data into MySQL

$ mysql -u root -p < mysqlsampledatabase.sql

# password: cloudera

1. Check the data exists

$ mysql -u root -p

mysql> show databases;

You should see classicmodels table

mysql> use classicmodels;

mysql> show tables;

Lists all tables available

Import data into Hadoop with Sqoop

When using sqoop, you will run commands from the Command Line using the “sqoop” command.

You can view all the options using the following command from the Command Line:

$ sqoop help

To test your connection, we can first list all the databases in the database. To do this you will need to use the “sqoop list-databases” command and provide some additional arguments:

* --connect {connection\_string}
  + Example: jdbc:mysql://localhost/classicmodels
    - Breakdown:
      * jdbc:mysql:// - The protocol to use connecting to the database.
      * localhost - The host name of where the db is running
      * classicmodels - The database name to use as the default database
* --username {mysql\_username}
* --password {mysql\_user\_password}

With including the options, we end up with the following command. This will list all the available databases.

$ sqoop list-databases \

--connect jdbc:mysql://localhost/classicmodels \

--username root \

--password cloudera

We can also list the tables with a similar command: “sqoop list-tables”. We will have to include the same options in order to connect:

$ sqoop list-tables \

--connect jdbc:mysql://localhost/classicmodels \

--username root \

--password cloudera

Now let's try importing some data into Hadoop from the MySQL database. We will keep it simple and just import data from one table and load it into HDFS:

To do this, we will need to run the “sqoop import” command and include the same connection details we included above. But in addition to that we will have to include some more arguments to let sqoop know what table specifically to import and how to do it:

* --table {table\_to\_import}
* --num-mappers {number\_of\_mappers}
  + **Sqoop, under the covers, runs a MapReduce job.** You can specify the number of mappers as an argument.
  + In our simple example, we’re not specifying what column the job should partition based on, so we have to use 1 mapper. This will be explained more later.
* --target-dir {hdfs\_directory\_to\_dump\_data}

$ sqoop import \

--connect jdbc:mysql://localhost/classicmodels \

--username root \

--password cloudera \

--table customers \

--num-mappers 1 \

--target-dir /user/cloudera/sqoop-mysql/customers

After running the job, you will see counters pertaining to the job that was ran. This includes the “Map input records” counter which shows how many rows were imported. You should also see a partition file in the /user/cloudera/sqoop-mysql/customers directory containing all the data.

Import data into Hive with Sqoop

As useful feature of Sqoop is that it integrates with Hive. This means we can import data into Hive from a relational database and keep the schema of the relational database. We will be going through the same process that we did in the previous section, but instead use Hive as the intermediary storage service.

First lets create the database in Hive to house the data we want to ingest:

$ hive

hive> CREATE DATABASE classicmodels;

Now that we have the database available, we can run the same “sqoop import“ command to ingest data into Hive. Instead of including the “--target-dir” option, we will need to include different ones to specify to you Hive:

* --hive-import
  + Argument to specify that we’re importing data into Hive
* --hive-database {hive\_database\_name}
* --hive-table {hive\_table\_name\_to\_store\_data}

With these options we end up with the following command:

$ sqoop import \

--connect jdbc:mysql://localhost/classicmodels \

--username root \

--password cloudera \

--table customers \

--num-mappers 1 \

--hive-import \

--hive-database classicmodels \

--hive-table customers

Internally, what this will do is import the data into a subdirectory under the user’s HDFS home directory (hdfs:///user/{username}/{table\_name}) and then run Hives “LOAD DATA” command to load the data into Hive. So if you ever see an error like:

*ERROR tool.ImportTool: Encountered IOException running import job: org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory hdfs://quickstart.cloudera:8020/user/cloudera/customers already exists*

This could mean that there is an import already happening, or a previous import has failed. You can easily clear this up by deleting the intermediary data:

$ hadoop fs -rm -r -skipTrash /user/cloudera/customers

Now that we have the data in Hive, we can transform the data in the same way we transformed it before, just using a Hive query:

$ hive

hive> CREATE TABLE classicmodels.customers\_transformed AS SELECT customerNumber, customerName, contactLastName, contactFirstName, phone, addressLine1, addressLine2, city, state, postalCode, country, salesRepEmployeeNumber, creditLimit FROM classicmodels.customers WHERE city = "San Francisco";

Now that we have our final data, we can export it back to MySQL.

First lets create a table in MySQL that will house the data:

$ mysql -u root -p

mysql> CREATE TABLE `classicmodels`.`customers\_transformed` (

`customerNumber` int(11) NOT NULL,

`customerName` varchar(50) NOT NULL,

`contactLastName` varchar(50) NOT NULL,

`contactFirstName` varchar(50) NOT NULL,

`phone` varchar(50) NOT NULL,

`addressLine1` varchar(50) NOT NULL,

`addressLine2` varchar(50) DEFAULT NULL,

`city` varchar(50) NOT NULL,

`state` varchar(50) DEFAULT NULL,

`postalCode` varchar(15) DEFAULT NULL,

`country` varchar(50) NOT NULL,

`salesRepEmployeeNumber` int(11) DEFAULT NULL,

`creditLimit` double DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

Now we can start the export process. While import works directly with Hive, exporting sadly does not. But you can still export Hive, just in a more manual way. Instead of pointing to the hive table, you would need to point to where all the base data is stored in Hive: in the Hive Warehouse (/user/hive/warehouse/{database\_name}.db/{table\_name}). So the export process is very similar to what we were doing in the previous section. But, we will need to add a few more arguments to handle how Hive stores the data:

* --input-fields-terminated-by {value}
  + This is how Sqoop identifies when a field ends in a file that’s created by Hive
  + Default value in Hive: \001
* --lines-terminated-by {value}
  + This is how Sqoop identifies when a line/row ends in a file that’s created by Hive
  + Default value in Hive: \n

With these arguments we end up with the following command to do the export:

$ sqoop export \

--connect jdbc:mysql://localhost/classicmodels \

--username root \

--password cloudera \

--table customers\_transformed \

--num-mappers 1 \

--input-fields-terminated-by "\\001" \

--lines-terminated-by "\\n" \

--export-dir /user/hive/warehouse/classicmodels.db/customers\_transformed

After the export is complete you should see the data in the MySQL table.

Import More Securely

One problem with the above import and export process with sqoop, is that you need to include the password in the command. This can be dangerous since the password will then be stored in the bash history for anyone to read. Another method is to store the password in a file on HDFS:

Create file:

$ printf "cloudera" > root\_pwd.txt

Copy File into HDFS and give read-only permissions to the current user to lock down security further:

$ hadoop fs -put root\_pwd.txt /user/cloudera/

$ hadoop fs -chmod 400 /user/cloudera/root\_pwd.txt

Now that we have the file in HDFS we can use a different argument it in place of the --pasword argument to use the password file:

* --password-file {hdfs\_path\_to\_file}

$ sqoop import \

--connect jdbc:mysql://localhost/classicmodels \

--username root \

--password-file /user/cloudera/root\_pwd.txt \

--table customers \

--num-mappers 1 \

--hive-import \

--hive-database classicmodels \

--hive-table customers