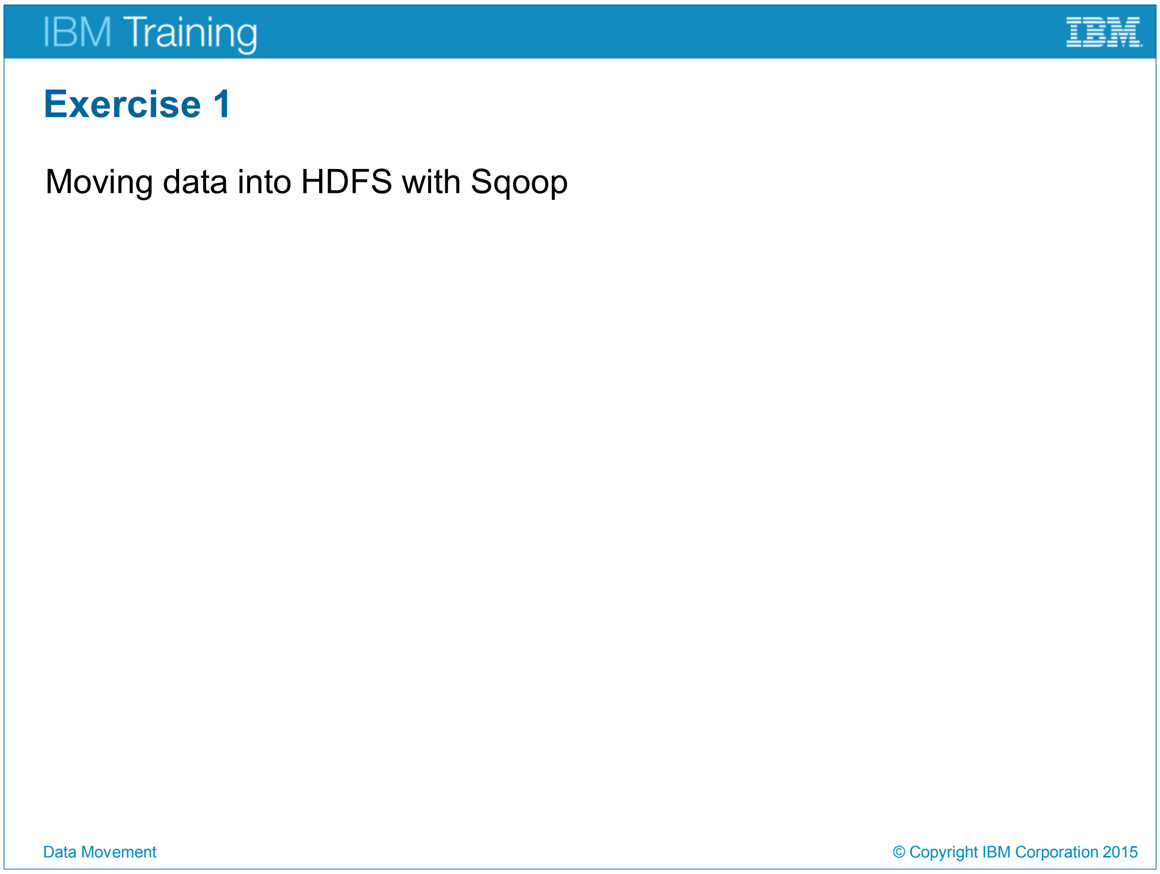

IBM Training

Exercise 1

Moving data into HDFS with Sqoop

Data Movement

© Copyright IBM Corporation 2015

Exercise 1: Moving data into HDFS with Sqoop

Exercise 1: Moving data into HDFS with Sqoop

Purpose:
You will learn how to move data into an HDFS cluster from a relational database. The relational database that will be used is MySQL since that is already installed in an ODP cluster.

You will connect into the MySQL database and create some data records in internal database storage format.

Sqoop will be used to import that data into HDFS (file movements are in relation to Hadoop and HDFS).

Additional exercises are available in the BigDataUniversity.com course on Moving Data into Hadoop (http://bigdatauniversity.com/bdu-wp/bdu-course/moving-data-into-hadoop).

Task 1. Login to your lab environment and connect to the MySQL database.

If you have problems connecting to the MySQL database, it may be for one of several reasons:

- The MySQL database may be not running. Appropriate commands to check status, start/stop, restart the MySQL server (needs root / administrative privileges) are:

```
sudo service mysqld status
sudo service mysqld start
sudo service mysqld stop
sudo service mysqld restart
```

- The database driver may be out-of-date for your version of MySQL:

  You can download the latest MySQL Connector/J driver for Linux from http://dev.mysql.com/downloads/connector/j and, after unpacking the tar file (tar xvf), place the mysql-connector-java-*.*.*-bin.jar file into /usr/iop/4.0.0.0/sqoop/lib directory.

- As noted previously, you may need to verify that your hostname and IP address are setup correctly, and make changes if they are needed as noted in earlier units. Note that if you have shut down your lab environment anytime, or overnight, this verification of hostname and IP address should be repeated.

1. Connect to your lab environment and login as **biadmin**.
2. In a new terminal window, type **`cd`** to change to your home directory, and then type **`mysql`** to start a MySQL command session.

   ```
   [biadmin@ibmclass Desktop]$ cd
   [biadmin@ibmclass ~]$ mysql
   Welcome to the MySQL monitor.  Commands end with ; or \g.
   Your MySQL connection id is 2
   Server version: 5.1.73 Source distribution

   Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

   Oracle is a registered trademark of Oracle Corporation and/or its
   affiliates. Other names may be trademarks of their respective
   owners.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   mysql>
   ```

   You will receive a MySQL prompt: **`mysql>`**
3. To see what databases are currently available in your MySQL server, type **`mysql> SHOW DATABASES;`**.

   ```
   mysql> SHOW DATABASES;
   +--------------------+
   | Database           |
   +--------------------+
   | information_schema |
   | test               |
   +--------------------+
   2 rows in set (0.02 sec)

   mysql>
   ```

   Note: SQL (and hence MySQL) commands are case-insensitive. The convention you are using here is that SQL keywords (reserved words) are written in Upper-Case and words which are not keywords are written here in lower-case.

   The string "mysql>" is the prompt, and should not be entered.

   MySQL commands are terminated by a semi-colon (";").
4. To connect to the test database, type **`mysql> USE test;`**.
5. Type the following command to create a table called **mytable** in this database:

   ```
   mysql> CREATE TABLE test.mytable (id INT,
   name VARCHAR(20));
   ```
6. Type the following command to insert data into your table:

   ```
   mysql> INSERT INTO test.mytable VALUES
   (1,'one'),(2,'two'),(3,'three'),(4,'four'),
   (5,'five'),(6,'six'),(7,'seven');
   ```

7. To verify that your data was actually stored into the database table, type:
   `mysql> SELECT * FROM test.mytable;`

```
mysql> USE test;
Database changed
mysql> CREATE TABLE test.mytable (id INT, name VARCHAR(20));
Query OK, 0 rows affected (0.09 sec)

mysql> INSERT INTO test.mytable VALUES (1,'one'),(2,'two'),(3,'three'),(4,'four'),
    -> (5,'five'),(6,'six'),(7,'seven');
Query OK, 7 rows affected (0.00 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM test.mytable;
+------+-------+
| id   | name  |
+------+-------+
|    1 | one   |
|    2 | two   |
|    3 | three |
|    4 | four  |
|    5 | five  |
|    6 | six   |
|    7 | seven |
+------+-------+
7 rows in set (0.00 sec)

mysql>
```

8. Type **`quit;`** since you will not need the database again for awhile.

Task 2. Import data from the database into HDFS.

Sqoop wants to know how many mappers it should employ. Normally, Sqoop would look at the primary key column to figure out how to split the data across the mappers. Since there was not a primary key defined for this table, you will have to help out.

You could use the *split-by <column-name>* parameter to tell Sqoop which column should be used in place of the primary key column. But since you only have one node in your Hadoop cluster, there is no sense in running more than just a single mapper. Do this by specifying the *-m 1* parameter.

1. Import all rows from **mytable** into the scooper directory in Hadoop using the following command:

   The sqoop statement is (in one line):

   sqoop import --connect jdbc:mysql://localhost/test --table mytable --target-dir scooper -m 1

```
[biadmin@ibmclass ~]$ sqoop import --connect jdbc:mysql://localhost/test --table mytable -
-target-dir scooper -m 1
Warning: /usr/iop/4.0.0.0/sqoop/../accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
15/06/05 18:18:59 INFO sqoop.Sqoop: Running Sqoop version: 1.4.5_IBM_2
15/06/05 18:18:59 ERROR sqoop.ConnFactory: Could not load ManagerFactory
com.ibm.biginsights.ie.sqoop.BIConnectionFactory (not found)
15/06/05 18:18:59 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.
15/06/05 18:18:59 INFO tool.CodeGenTool: Beginning code generation
15/06/05 18:18:59 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM
`mytable` AS t LIMIT 1
15/06/05 18:18:59 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM
`mytable` AS t LIMIT 1
15/06/05 18:18:59 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is
/usr/iop/4.0.0.0/hadoop
Note: /tmp/sqoop-biadmin/compile/edc36d0be586f056b14ba5b6adc97663/mytable.java uses or
overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
15/06/05 18:19:01 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-
biadmin/compile/edc36d0be586f056b14ba5b6adc97663/mytable.jar
15/06/05 18:19:01 WARN manager.MySQLManager: It looks like you are importing from mysql.
15/06/05 18:19:01 WARN manager.MySQLManager: This transfer can be faster! Use the --direct
15/06/05 18:19:01 WARN manager.MySQLManager: option to exercise a MySQL-specific fast
path.
15/06/05 18:19:01 INFO manager.MySQLManager: Setting zero DATETIME behavior to
convertToNull (mysql)
15/06/05 18:19:01 INFO mapreduce.ImportJobBase: Beginning import of mytable
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/iop/4.0.0.0/hadoop/lib/slf4j-log4j12-
1.7.5.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/iop/4.0.0.0/zookeeper/lib/slf4j-log4j12-
1.6.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
15/06/05 18:19:03 INFO impl.TimelineClientImpl: Timeline service address:
http://ibmclass.localdomain:8188/ws/v1/timeline/
15/06/05 18:19:03 INFO client.RMProxy: Connecting to ResourceManager at
ibmclass.localdomain/192.168.244.141:8050
15/06/05 18:19:04 INFO db.DBInputFormat: Using read commited transaction isolation
15/06/05 18:19:04 INFO mapreduce.JobSubmitter: number of splits:1
```

```
15/06/05 18:19:04 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1433542652323_0002
15/06/05 18:19:05 INFO impl.YarnClientImpl: Submitted application
application_1433542652323_0002
15/06/05 18:19:05 INFO mapreduce.Job: The url to track the job:
http://ibmclass.localdomain:8088/proxy/application_1433542652323_0002/
15/06/05 18:19:05 INFO mapreduce.Job: Running job: job_1433542652323_0002
15/06/05 18:19:12 INFO mapreduce.Job: Job job_1433542652323_0002 running in uber mode :
false
15/06/05 18:19:12 INFO mapreduce.Job:  map 0% reduce 0%
15/06/05 18:19:17 INFO mapreduce.Job:  map 100% reduce 0%
15/06/05 18:19:18 INFO mapreduce.Job: Job job_1433542652323_0002 completed successfully
15/06/05 18:19:18 INFO mapreduce.Job: Counters: 30
   File System Counters
          FILE: Number of bytes read=0
          FILE: Number of bytes written=122722
          FILE: Number of read operations=0
          FILE: Number of large read operations=0
          FILE: Number of write operations=0
          HDFS: Number of bytes read=87
          HDFS: Number of bytes written=48
          HDFS: Number of read operations=4
          HDFS: Number of large read operations=0
          HDFS: Number of write operations=2
   Job Counters
          Launched map tasks=1
          Other local map tasks=1
          Total time spent by all maps in occupied slots (ms)=3136
          Total time spent by all reduces in occupied slots (ms)=0
          Total time spent by all map tasks (ms)=3136
          Total vcore-seconds taken by all map tasks=3136
          Total megabyte-seconds taken by all map tasks=3211264
   Map-Reduce Framework
          Map input records=7
          Map output records=7
          Input split bytes=87
          Spilled Records=0
          Failed Shuffles=0
          Merged Map outputs=0
          GC time elapsed (ms)=40
          CPU time spent (ms)=910
          Physical memory (bytes) snapshot=197353472
          Virtual memory (bytes) snapshot=1661890560
          Total committed heap usage (bytes)=230686720
   File Input Format Counters
          Bytes Read=0
   File Output Format Counters
          Bytes Written=48
15/06/05 18:19:18 INFO mapreduce.ImportJobBase: Transferred 48 bytes in 16.2116 seconds
(2.9609 bytes/sec)
15/06/05 18:19:18 INFO mapreduce.ImportJobBase: Retrieved 7 records.
[biadmin@ibmclass ~]$
```

Note that a MapReduce job was created and run. There was one Mapper and no Reducers. There is one output file, and it has 7 records.

2. To see what has been stored in the HDFS file system and the content of the file that was created, type the following commands:

```
hadoop fs –ls –R
hadoop fs –cat scooper/p*
```

```
[biadmin@ibmclass ~]$ hadoop fs -ls -R
drwx------   - biadmin biadmin          0 2015-06-05 18:10 .staging
drwxr-xr-x   - biadmin biadmin          0 2015-06-05 18:10 scooper
-rw-r--r--   3 biadmin biadmin          0 2015-06-05 18:10 scooper/_SUCCESS
-rw-r--r--   3 biadmin biadmin         48 2015-06-05 18:10 scooper/part-m-00000


[biadmin@ibmclass ~]$ hadoop fs -cat scooper/p*
1,one
2,two
3,three
4,four
5,five
6,six
7,seven
```

Task 3. Import data from the database into HDFS using a sqoop script file.

Typing all of those statements into a command line, knowing that when you close the command line window, the command will be lost, seems like a waste. But what if your commands could be saved in a text file? That might be worth something. Also, what if you wanted to limit the rows imported? You will review each of these ideas.

Note that the UNIX rule for parameters applies. When the parameter name is a single letter, a single dash is used (`-m`); but when a parameter name has more than one letter, a double dash is used (`--connect`).

1. From the command line, type `gedit &`.
2. In the document window, type your import parameters as seen below:

   Notice that you can add comments.

```
import
--connect
jdbc:mysql://localhost/test
--table
mytable
--target-dir
sqoopimport2
# Only select some rows
--where
ID > 4
# Remaining options should be specified on the command line
```

3. From the **File** menu, click **Save As**, type **sqoop.params** as the filename, and then click **Save**.

4. From the command line, type the following:

```
sqoop --options-file ~/sqoop.params -m 1
```

5. View your results and take note of how many records were imported.

Results:
You moved data into an HDFS cluster from a relational database. You connected into the MySQL database and created some data records in internal database storage format.

Sqoop was used to import that data into HDFS (file movements are in relation to Hadoop and HDFS).