Exercise 1

Using Sqoop to Import Relational Data into Hadoop

Contents

Lab 1 Configuring Flume for Data Loading

This exercise gives you the opportunity to use Sqoop to extract data from a relational database table and import that data into Hadoop.

After completing this hands-on lab, you'll be able to:

- Execute the Sqoop import command to extract data from a relational table and copy that data into Hadoop

Allow 20 minutes to complete this lab.

This version of the lab was updated and tested on the InfoSphere BigInsights 4.1 Quick Start Edition. Throughout this lab you will be using the following account login information. If your passwords are different, please note the difference.

	Username	Password
VM image setup screen	root	password
Ambari	admin	admin

Sqoop allows you to move data between a relational database system and Hadoop. Sqoop is able to import data from a relational table into Hadoop and is also able to export data from Hadoop into a relational database table.

This exercise focuses on importing data into Hadoop.

Sqoop is able to work with any relational database system that supports JDBC. To simplify things and since it was installed as a part of BigInsights, this exercise uses the MySQL relational database system.

1.1 Getting Started

It is assumed that you have downloaded the VM image (BigInsights QuickStart 4.1) and completed the setup and configuration for VM Workstation 12 Player.

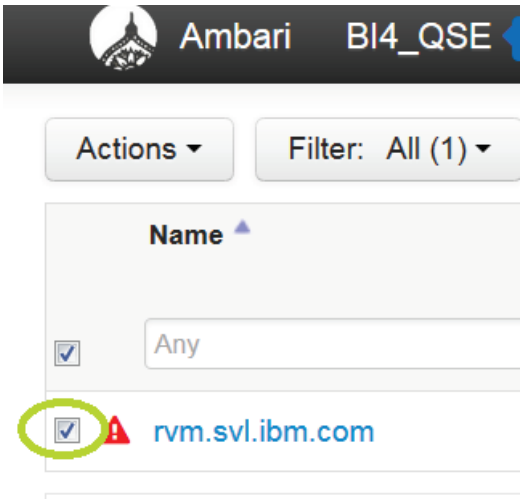
First, you want to start the BigInsights components.

__1. Open a Web browser and navigate to **http://rvm.svl.ibm.com:8080**, and sign in using the Ambari user id and password specified at the beginning of this document.

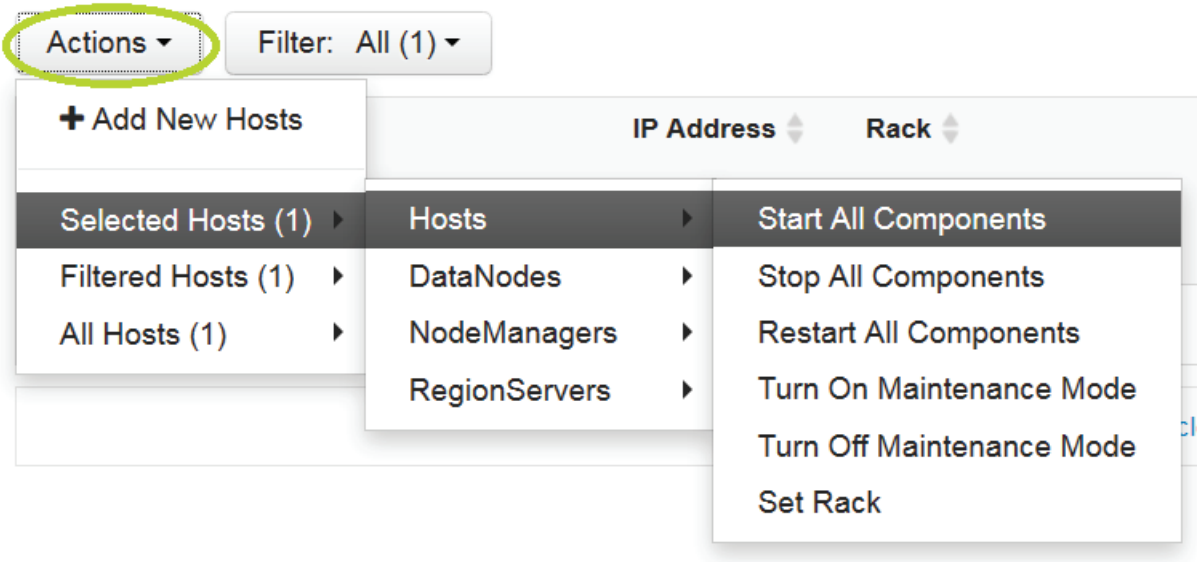
Notice that most of the BigInsights components listed at the left are in a Stopped state as indicated by the red, triangular warning icon.

__2. Click on **Hosts** in the top of web page.

__3. Check the **box** next to your host. *"rvm.svl.ibm.com"*



__4. Click Actions -> Selected Hosts (1) -> Hosts -> Start All Components

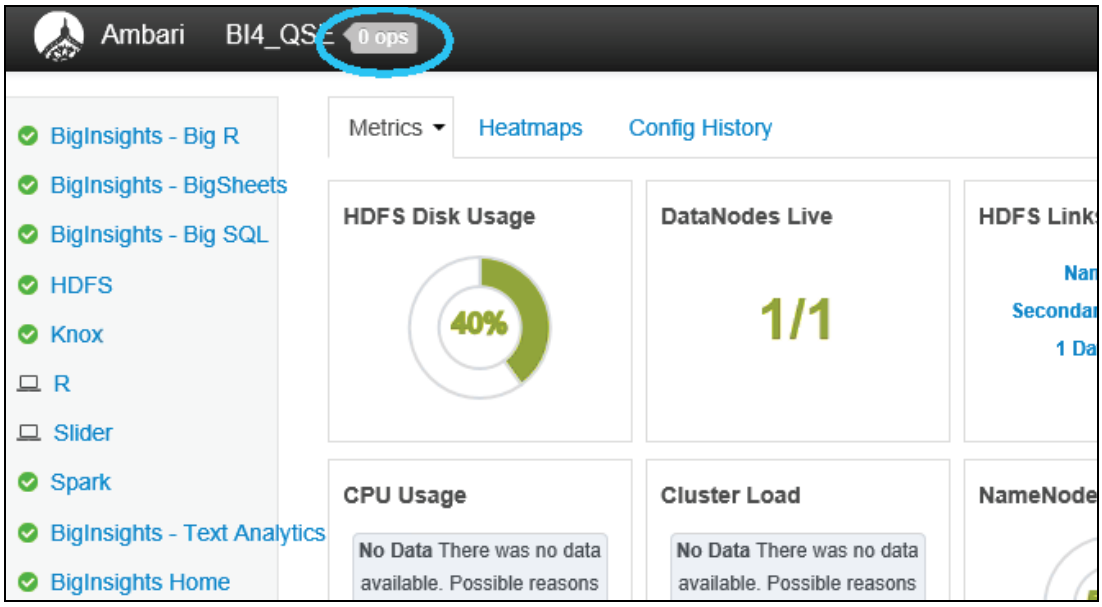


__5. Confirm by hitting **OK** when *"Confirm Bulk Operations"* pops up.

Clicking *Stop All* would stop the components in a similar manner.

Note: Be sure to allow ample time for all the components to start. The first time you start the components and services, it may take approximately 30 minutes or even longer, depending on the physical resources on your machine.

__6. Periodically examine the background operations indicator at the top of the screen. It should show that 1 operation is running. When it updates to 0 ops as shown below, the Start All script is complete and the components should all be running as indicated by the green check mark icons on the left.



Note: If your icons still show red warning signs after the startup, it may be that the Ambari interface did not refresh properly, even though the details in the background operations show 100% and display a successful message. Feel free to click the Admin button at the top right of the window, and then click Sign out. You will be presented with the Ambari login screen. Log back in using the credentials at the beginning of this document and the component list should be updated with the correct, green check-mark icons.

Please refer to ***this link*** for a tutorial on how to enable Shared Folders on the VM.

Please refer to ***this link*** for the download page of the latest MySQL Connector/J driver.

Now that are components are started you may move on to the next section.

1.2 Create a database and table

Important:

Before doing this lab, download the latest MySQL Connector/J driver and place the mysql-connector-java-*.*.**-bin.jar file in the /usr/iop/4.1.0.0/sqoop/lib directory on the BigInsights virtual machine.

You will first have to create a table within the MySQL test database. This is so you will have a source for your MySQL import.

__1. Start a MySQL command line session

```
mysql
```

__2. Check what databases are currently in MySQL.

```
mysql> SHOW DATABASES;
```

__3. We will use the MySQL test database.

```
mysql> USE test;
```

__4. Create a table called *mytable* in this new database.

```
mysql> create table test.mytable (id int, name varchar(20));
```

__5. Insert data into the table.

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

__6. Verify your data...

```
mysql> select * from test.mytable;
```

```
mysql> create table test.mytable (id int, name varchar(20));
Query OK, 0 rows affected (0.01 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 (0.01 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>
```

__7. Disconnect.

```
mysql> quit;
```

1.3 Import data using Sqoop

__1. Make sure you have placed the latest MySQL Connector/J driver file in Sqoop's lib directory.

__2. In a Linux terminal, change to the Sqoop *bin* directory.

```
cd /usr/iop/4.1.0.0/sqoop/bin
```

__3. Import all rows from *mytable* into the /user/hdfs/*sqoopimport1* directory in Hadoop.

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. We also need to be in hdfs and run the sqoop import as root due to permission issues.

```
su hdfs

./sqoop import --connect jdbc:mysql://localhost/test –username root --
table mytable --target-dir sqoopimport1 -m 1
```

__4. In the displayed statistics, you can see *Map output records ==7,* which makes sense since there were seven rows in the table.

```
virtuser@rvm:/usr/iop/4.0.0.0/sqoop/bin
15/05/15 13:29:07 INFO mapreduce.Job: Job job_1431721110730_0001 completed successfully
15/05/15 13:29:07 INFO mapreduce.Job: Counters: 30
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=122920
                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)=7070
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=3535
                Total vcore-seconds taken by all map tasks=3535
                Total megabyte-seconds taken by all map tasks=3619840
        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)=62
                CPU time spent (ms)=1120
                Physical memory (bytes) snapshot=181469184
                Virtual memory (bytes) snapshot=1594941440
                Total committed heap usage (bytes)=231735296
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=48
15/05/15 13:29:07 INFO mapreduce.ImportJobBase: Transferred 48 bytes in 22.5305 seconds (2.1304 bytes/sec)
15/05/15 13:29:07 INFO mapreduce.ImportJobBase: Retrieved 7 records.
```

__5. Display the data in Hadoop. (Make sure you are still in hdfs to run hadoop commands unless you've manually change the permissions)

```
hadoop fs -cat /user/hdfs/sqoopimport1/part-m-00000
```

```
[virtuser@rvm bin]$ hadoop fs -cat /user/virtuser/sqoopimport1/part-m-00000
1,one
2,two
3,three
4,four
5,five
6,six
7,seven
```

1.4 One more time

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? Let's look into each of these ideas.

__1. From the command line execute *vi*

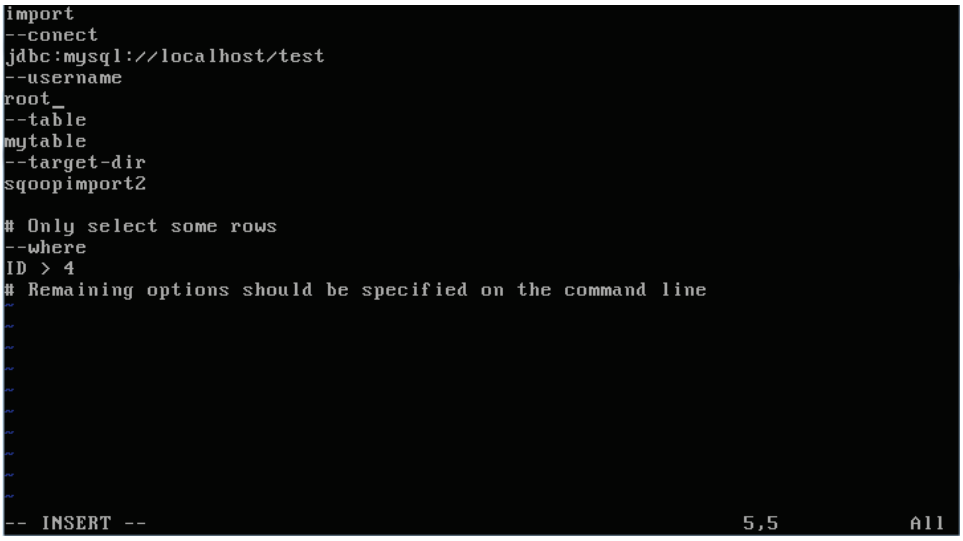
```
cd ~
vi sqoop.params
```

```
[hdfs@rvm ~]$ cd ~
[hdfs@rvm ~]$ vi sqoop.params_
```

__2. Type in your import parameters. Note that you can add comments. Note: Press *"Insert"* to start typing into vi editor, and Press *"Esc"* then *"Shift + z"* two times to exit and save the editor.

```
import
--connect
jdbc:mysql://localhost/test
--username
root
--table
mytable
--target-dir
sqoopimport2

# Only select some rows
--where
ID > 4
# Remaining options should be specified on the command line
```

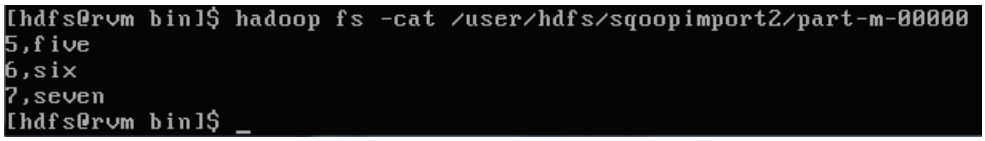


__3. Then open the sqoop bin directory and execute the following.

```
cd /usr/iop/4.1.0.0/sqoop/bin
./sqoop --options-file /home/hdfs/sqoop.params –m 1
```

__4. View your results

```
hadoop fs –cat /user/hdfs/sqoopimport2/part-m-00000
```



__5. You can close all open windows. You will need Hadoop running for your next exercise.

End of exercise

NOTES

NOTES

Please Recycle

IBM Software