

Session 3: Lab Document for Sqoop Code Run on AWS EMR

Step 1: Connect to AWS EMR (via PuTTY, etc.). You have already learnt how to connect to AWS EMR in the previous modules.


Note: Change to root user on AWS EMR, if it is not root user by default, by typing the following command on the terminal: **sudo -i**.

Step 2: Practice the following Sqoop commands as you go through the videos in the segments ahead.

Note: We will import the data from the database into directories in Hadoop. Make sure you do not have directories with the same name created already, to avoid conflict issues.

Note: You can follow the following steps using the **hadoop** user in EMR as well. Please make sure that you substitute the corresponding paths with the hadoop directory paths accordingly.

Note: Please make sure that you are changing the <Public DNS> part in the commands below with the Public DNS of the Master Node of your EMR Cluster.

Master public DNS: ec2-3-92-244-104.compute-1.amazonaws.com 
Connect to the Master Node Using SSH

Segment 2 – Importing Data: Importing Specific Rows in Sqoop

Sqoop provides the option to import specific rows as well, based on some where conditions.

Step 1: First, run the where clause in MySQL to determine the number of rows that are getting abstracted out of that query, as shown below:

```
mysql>select count(*) from test.flights_info where  
destination<>origin;
```

Step 2: Before running the sqoop import command, keep in mind that the target directory should not exist at all; otherwise, the Sqoop command will throw an error, as it does not overwrite the already existing job.

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r  
/user/root/flights_where_command
```

Step 3: Second, run the Sqoop import command with the where clause as shown below to import records that satisfy the where condition.

```
[root@ip-172-31-45-59 ~]# sqoop import --connect jdbc:mysql://<Public  
DNS>:3306/test \  
--table flights_info \  
--username root --password 123 \  
--target-dir /user/root/flights_where_command \  
--where "destination<>origin" \  
-m 1
```

Step 4: Verify once, by comparison, whether the record count in MySQL and the count of records imported through the Sqoop command are the same or not

Segment 3 – Importing Data: SQL Queries in Sqoop Import

Sqoop provides the option to import data satisfying SQL query conditions. A Sqoop command for the same is shown below:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_query_command

[root@ip-172-31-45-59 ~]# sqoop import --connect jdbc:mysql://<Public
DNS>:3306/test \
--username root --password 123 \
--query 'select * from test.employee where $CONDITIONS' \
--split-by id \
--target-dir /user/root/flights_query_command
```

Segment 4 – Importing Data: Using Incremental Import in Sqoop

In order to perform incremental import in Sqoop, there should be a column in the corresponding table, which should be either a primary key or a column on which incremental append conditions could be applied.

Through this command, Sqoop provides you with an option to import newly appended records only, based on the last value that is written.

Step 1: Hence, first of all, append 2-3 records in MySQL using an insert clause along with the incremented values of the corresponding primary key column as shown below:

```
mysql>insert into test.employee values (105,'Hitesh','software
engineer',2000);

mysql>insert into test.employee values (106,'Rahul','senior software
engineer',3000);
```

Step 2: Now, run the sqoop import command to fetch only those records that are greater than the last recorded or last seen value, as shown below:

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/employee_incremental_command

[root@ip-172-31-45-59 ~]# sqoop import --connect jdbc:mysql://<Public
DNS>:3306/test \
--table employee --username root --password 123 \
--target-dir /user/root/employee_incremental_command \
--incremental append --check-column id \
--last-value 104 -m 1
```

Step 3: Using the sqoop command, verify once whether all records greater than the last seen value have been imported or not.

Segment 5 – Sqoop Jobs

Step 1: Sqoop provides an option to save the import and export commands so that they can be executed repeatedly, as shown below:

```
[root@ip-172-31-45-59 ~]# sqoop job --create flightsimport -- import
\
--connect jdbc:mysql://<Public DNS>:3306/test \
--table flights_info \
--username root --password 123 \
--target-dir /user/root/flights_job_command -m 1
```

Step 2: You can check whether the job was created or not by running the list option with the sqoop job command, as shown below:

```
[root@ip-172-31-45-59 ~]# sqoop job --list
```

Step 3: Now, you can run the sqoop job by following command. Before executing the job, keep in mind that the target directory should not exist at all

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r
/user/root/flights_job_command
```

Step 4: Now, execute the sqoop job as shown below:

```
[root@ip-172-31-45-59 ~]# sqoop job --exec flightsimport
```

Step 5: Verify the data by checking the target directory once

```
[root@ip-172-31-45-59 ~]# hadoop fs -ls  
/user/root/flights_job_command
```

Segment 6 – Tuning Sqoop

While working with Sqoop in the IT industry, especially in the production environment, password is not passed as a parameter along with the Sqoop command. Rather, it is stored in a text file, with only read permissions to the ID through which the Sqoop command is running in production.

Step 1: In order to store passwords in a text file, please follow the command shown below. Keep in mind that there should not be any '\n' character after the password in the text file; otherwise, the Sqoop import command will keep on failing and you will not be able to debug the issue

```
[root@ip-172-31-45-59 ~]# echo -n "123" > pass.txt
```

Step 2: After storing the password in a local text file, move that file to the Hadoop location and assign chmod 400 permissions to it

```
[root@ip-172-31-45-59 ~]# hadoop fs -put /root/pass.txt  
/user/root/pass.txt  
  
[root@ip-172-31-45-59 ~]# hadoop fs -chmod 400 /user/root/pass.txt
```

Step 3: Before running the sqoop import command, keep in mind that there should not be any target directory with the same name created already. Hence, to be on the safer side, remove the target directory

```
[root@ip-172-31-45-59 ~]# hadoop fs -rm -r /user/root/flights
```

Step 4: Now, run the sqoop import command as shown below and verify the data in the target directory

```
[root@ip-172-31-45-59 ~]# sqoop import \  
--connect jdbc:mysql://<Public DNS>:3306/test \  
--table flights_info \  
--username root --password-file /user/root/pass.txt \  
--target-dir /user/root/flights \  
-m 1
```

Use the following command for verifying the data in the target directory:

```
[root@ip-172-31-45-59 ~]# hadoop fs -ls /user/root/flights/
```