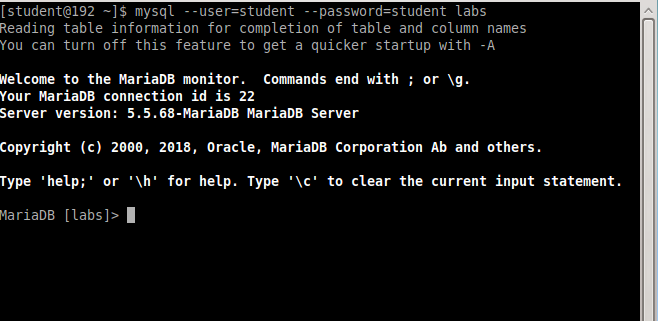
# Lab 1:Data Ingestion with Sqoop for RDBMS (MariaDB)

1. In a terminal window, log in to MariaDB and select the database labs.

Database: labs

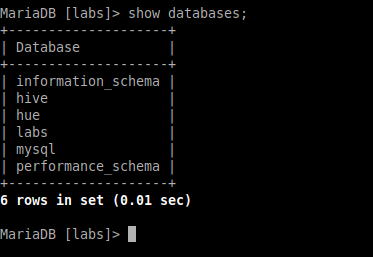
$ mysql --user=student --password=student labs



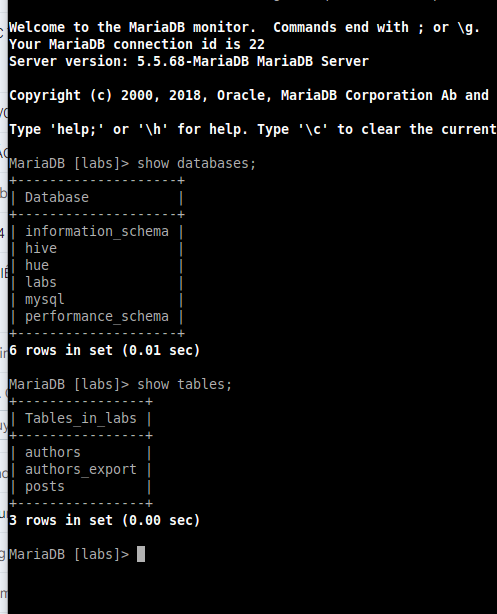
2. If the login is successful, the "MariaDB [labs]>" prompt appears and a screen waiting for

commands is displayed. Enter a command to check which database exists here.

MariaDB> show databases;

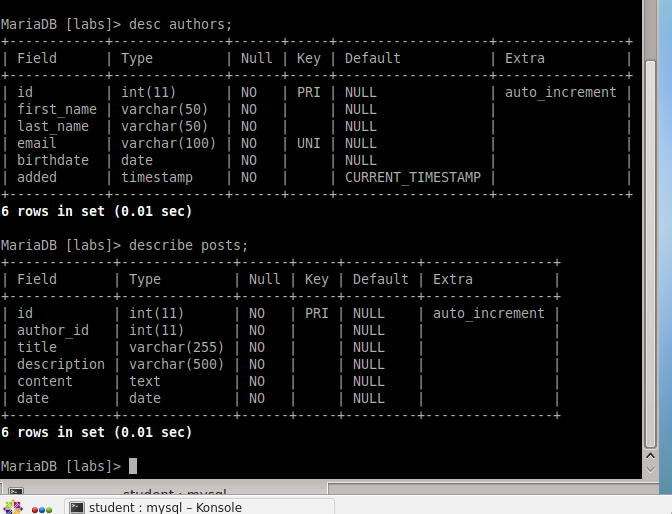


3. Next, enter the command to review the table in labs.



MariaDB> desc authors;

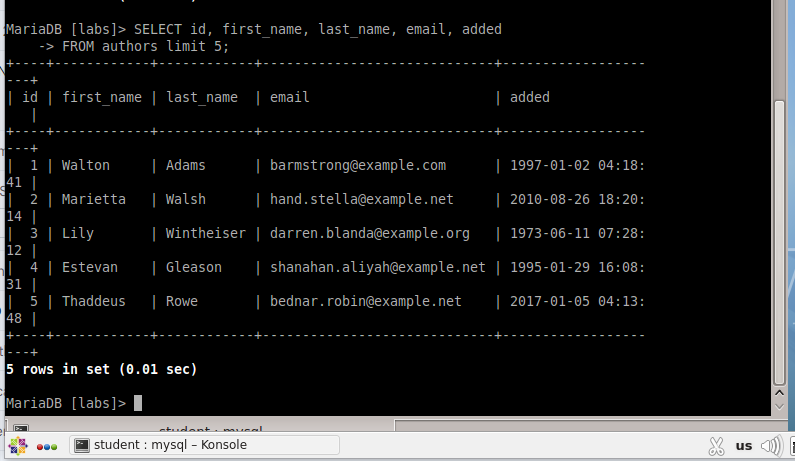
MariaDB> describe posts;



4. Review the structure of the authors, posts tables and review some records.

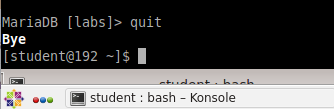
MariaDB> SELECT id, first\_name, last\_name, email, added

FROM authors limit 5;



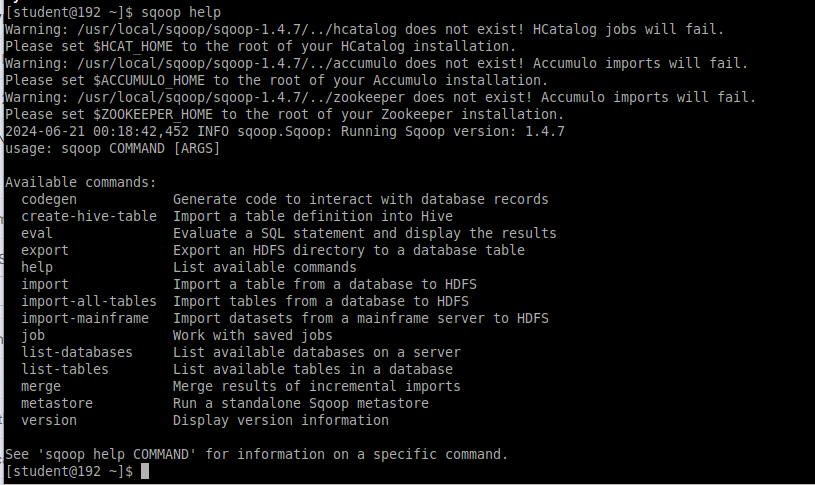
5. Type quit to exit MariaDB and press Enter.

MariaDB> quit



6. Run the following command to check the basic options of sqoop.

$sqoop help



7. To see detailed options for each sub-command, enter the desired subcommand after help. To see detailed options for import, run the command as follows.

$sqoop help import

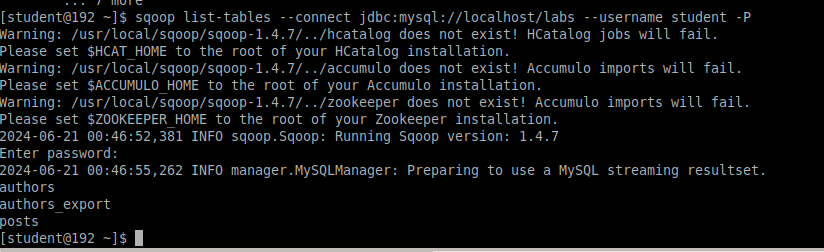


8. Run the list of databases in MariaDB and tables in database labs with the following

command.

$sqoop list-databases --connect jdbc:mysql://localhost --username student – password student

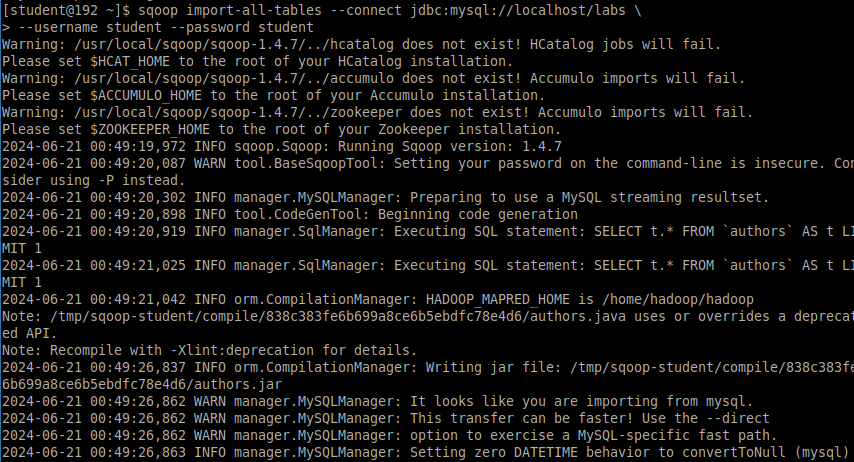
$sqoop list-tables --connect jdbc:mysql://localhost/labs --username student -P

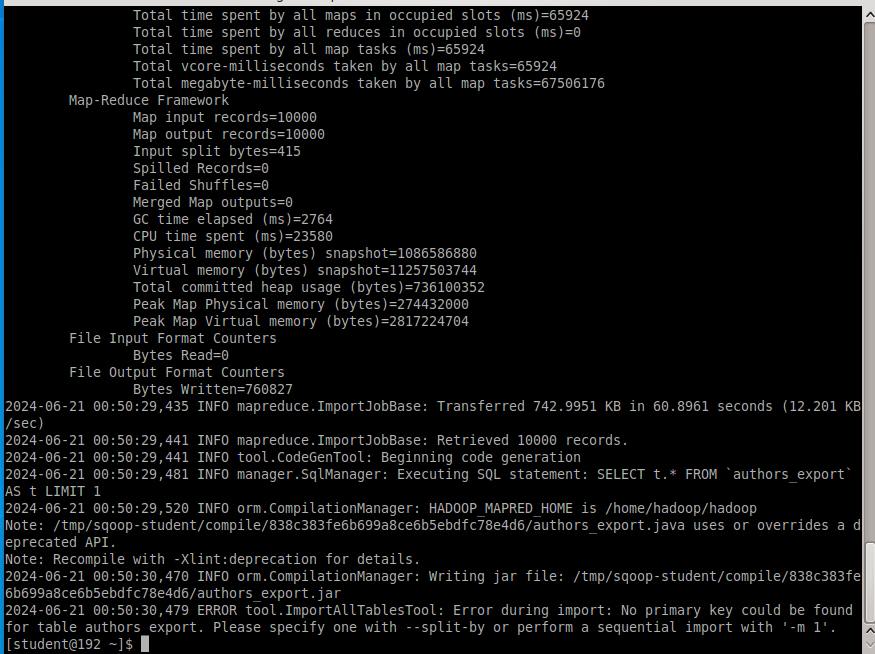


9. Import all tables in labs database using the import-all-tables command.

$sqoop import-all-tables --connect jdbc:mysql://localhost/labs \

--username student --password student



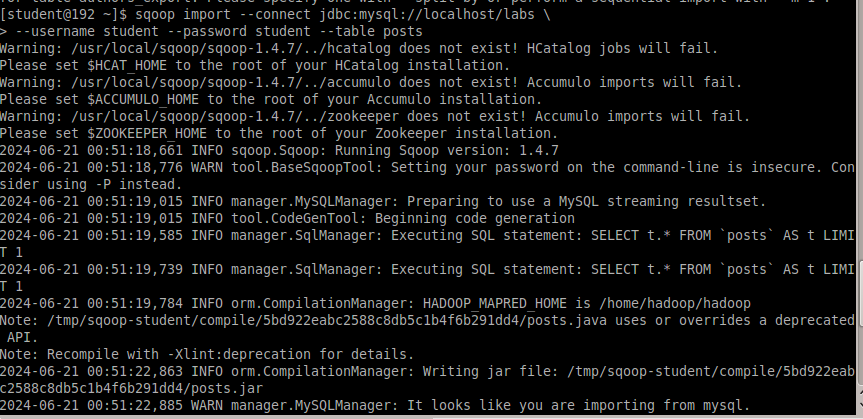


10. Execute the command to fetch the posts table from the labs database using Sqoop and

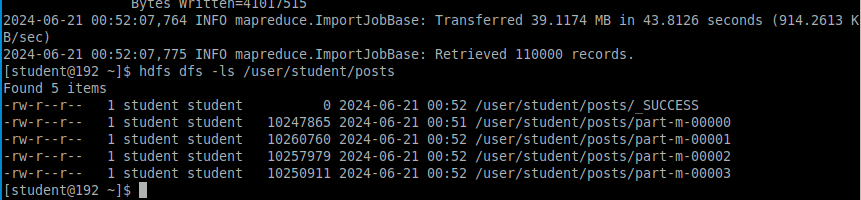
store it in HDFS.

$ sqoop import --connect jdbc:mysql://localhost/labs \

--username student --password student --table posts



When this command is executed, the posts directory is created under the /user/student home directory of HDFS and data is stored as follows



11.Create a target directory in HDFS to import table data into.

$hdfs dfs -mkdir /mywarehouse



12. Import the authors table and save it to the HDFS directory we created above using ',' to

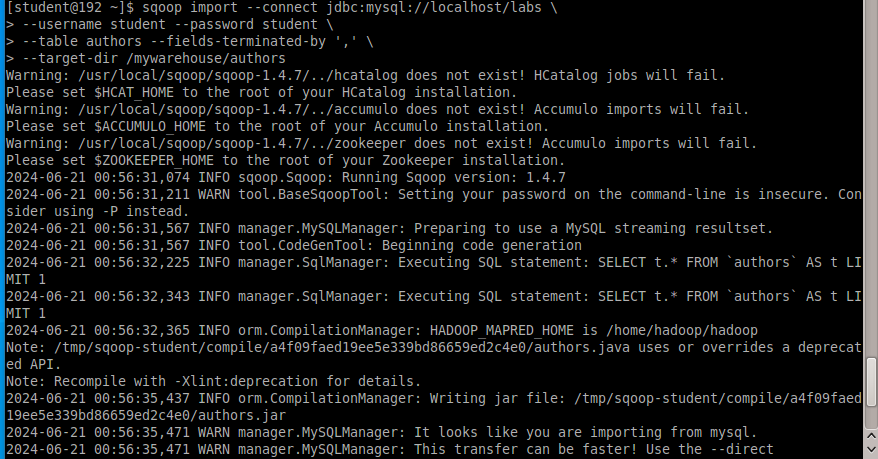
delimit the fields.

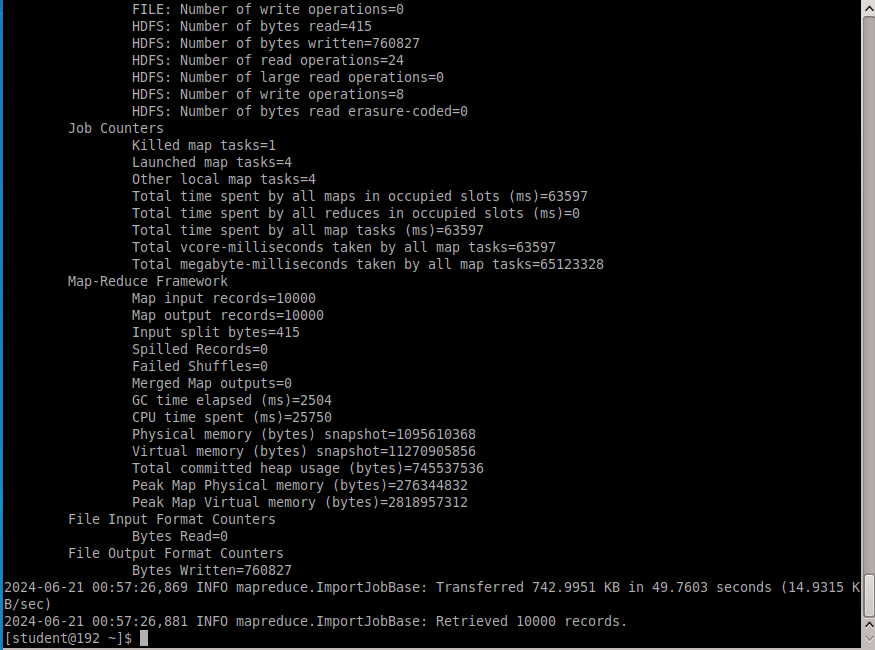
$ sqoop import --connect jdbc:mysql://localhost/labs \

--username student --password student \

--table authors --fields-terminated-by ',' \

--target-dir /mywarehouse/authors

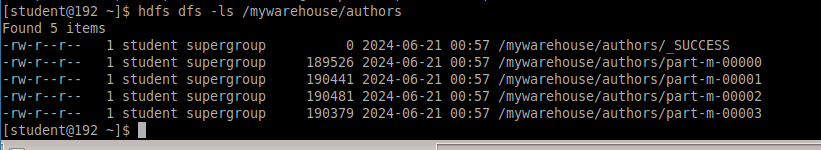


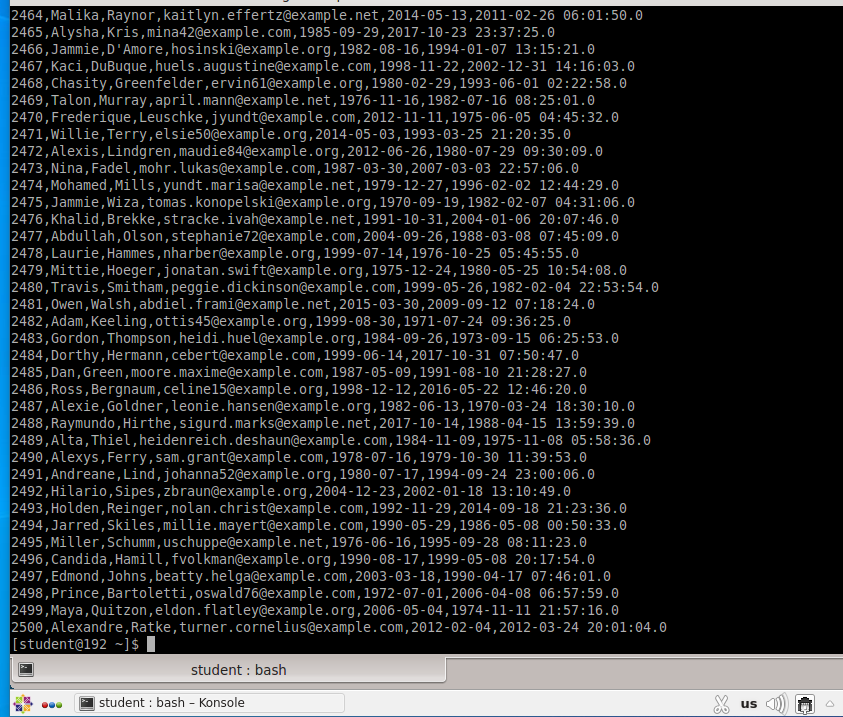


13. Review that the command worked with hdfs commands for target-dir.

$ hdfs dfs -ls /mywarehouse/authors

$ hdfs dfs -cat /mywarehouse/authors/part-m-00000





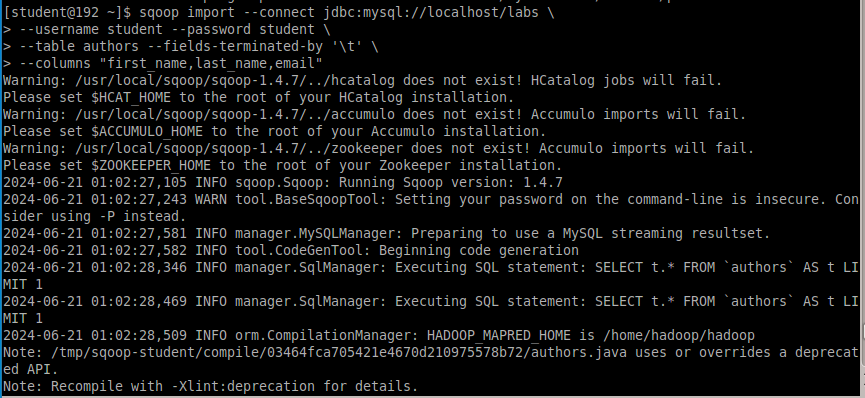
14. Import the only specified columns with –columns for authors in hdfs home directory. The

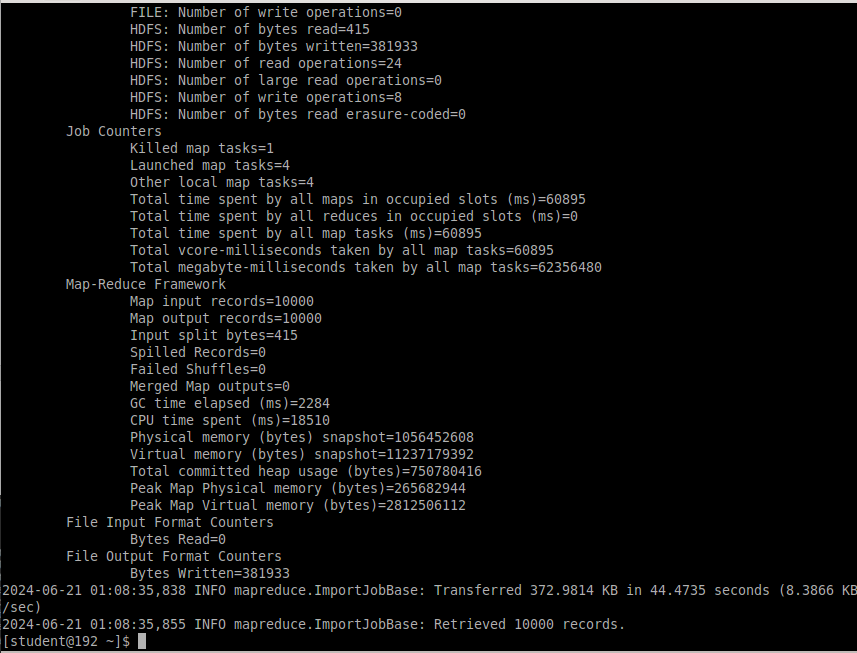
imported columns are first\_name, last\_name, email.

$ sqoop import --connect jdbc:mysql://localhost/labs --username student --password

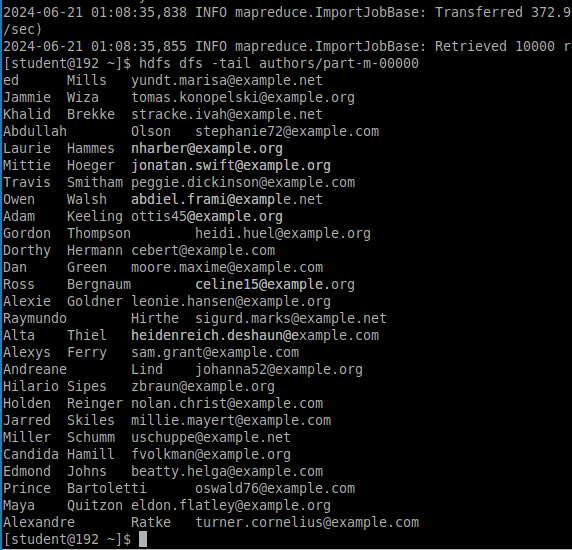
student --table authors --fields-terminated-by '\t' --columns "first\_name, last\_name,

email





Result in



15. Import the only matching row with –where statement. The imported rows are the first

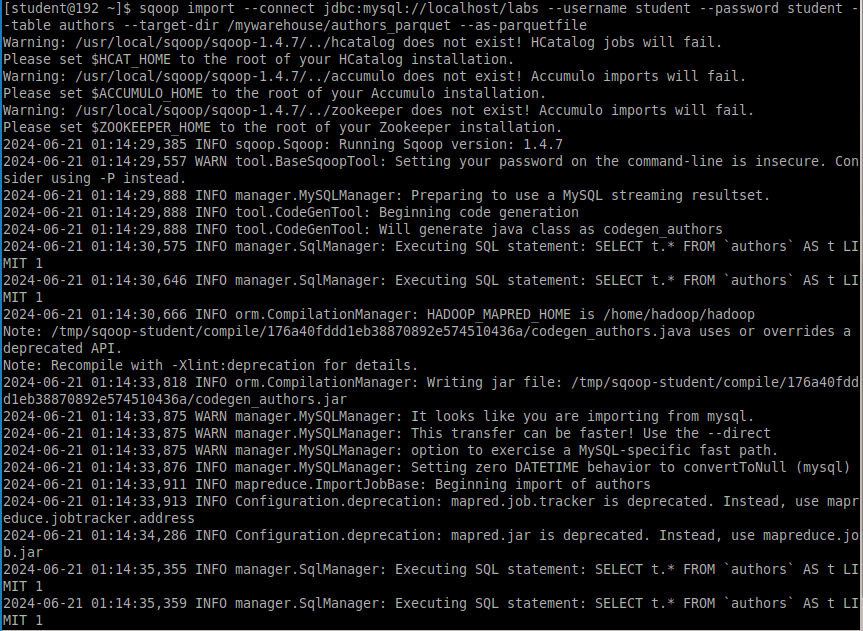
named 'Dorthy' in the authors table.

$ sqoop import --connect jdbc:mysql://localhost/test --username student –password student --table authors --fields-terminated-by '\t' --where "first\_name='Dorthy'" –- target-dir authors\_Dorthy

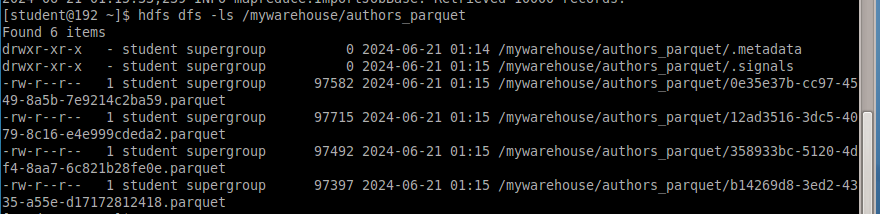
16. Import a table using an alternate file format instead of text format. Import the authors

table to Parquet format.

$sqoop import --connect jdbc:mysql://localhost/labs --username student --password student --table authors --target-dir /mywarehouse/authors\_parquet --as-parquetfile

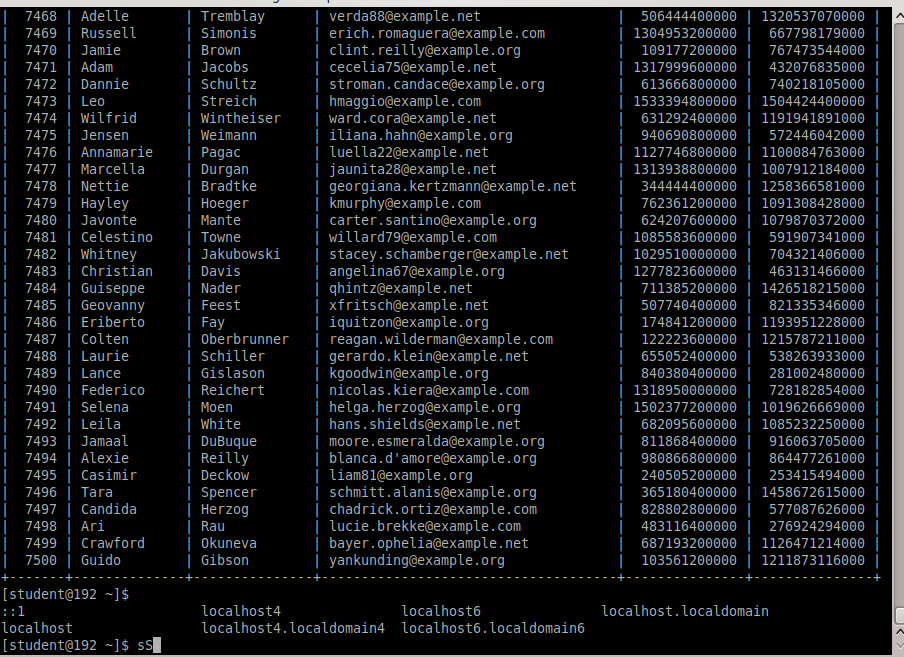


17. view the results of the import commands by listing the contents in HDFS (target-dir).



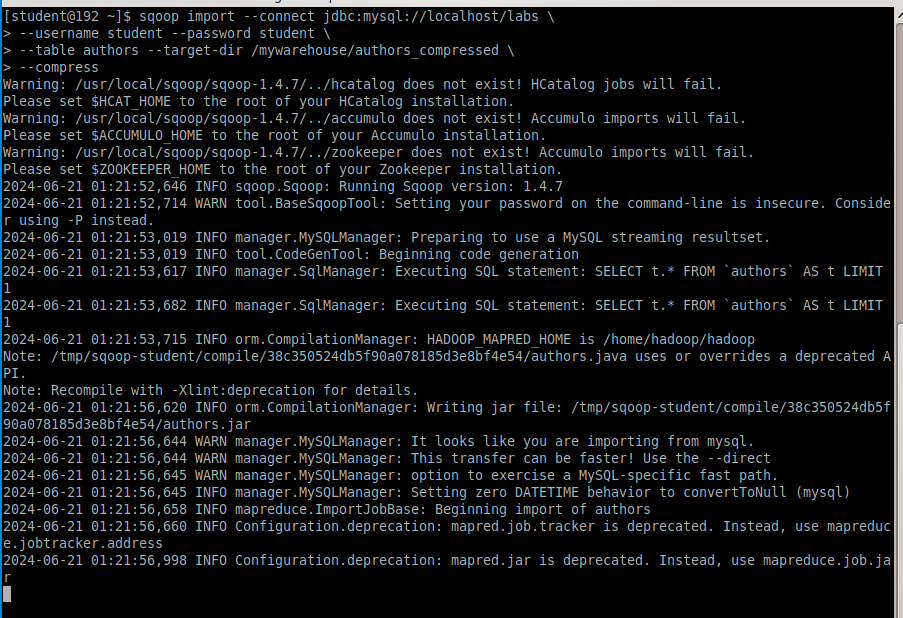
$ hdfs dfs -get /mywarehouse/authors\_parquet/ e35e37b-cc97-4549-8a5b-7e9214c2ba59.parquet

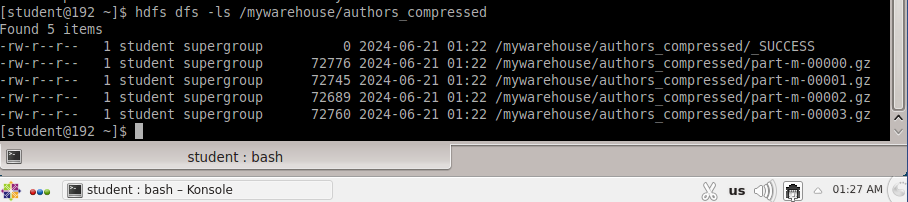
$ parquet-tools show e35e37b-cc97-4549-8a5b-7e9214c2ba59.parquet



18. Import a table using a compression option –compress or -z for authors table.

$ sqoop import --connect jdbc:mysql://localhost/labs --username student --password student --table authors --target-dir /mywarehouse/authors\_compressed –compress

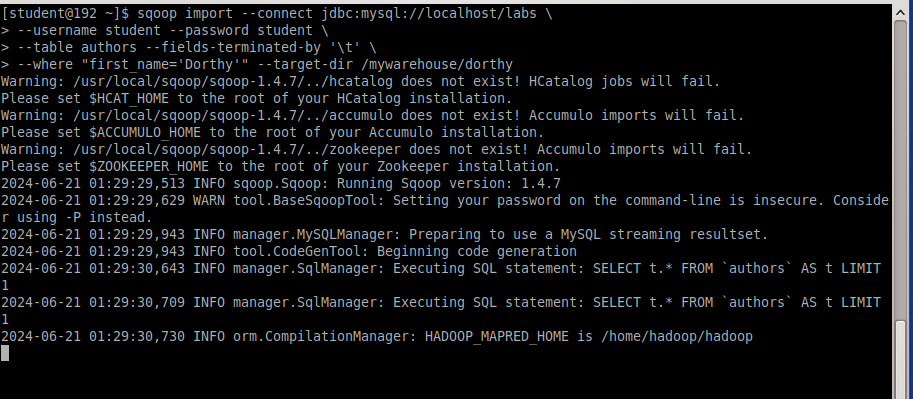




19. First, import the rows whose first name is "Dorthy" performed in step 15, and save it as

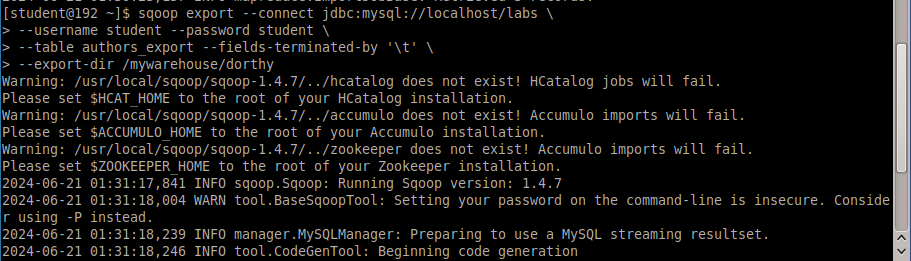
dorthy folder in the hdfs home directory

$ sqoop import --connect jdbc:mysql://localhost/labs --username student –password student --table authors --fields-terminated-by '\t' --where "first\_name='Dorthy'" -- target-dir dorthy



Export the saved dorthy folder as a table to the labs DB of RDBMS.

$sqoop export --connect jdbc:mysql://localhost/labs --username student --password student --table authors\_export --fields-terminated-by '\t' --export-dir dorthy



20.Review the contents of the exported records in MariaDB.



# Lab 2: Data Ingestion with Apache Flume

1. Simple Data Transfer

This Agent allows the user to generate events and subsequently log them to the console. This configuration defines a single agent named agent1.

1.1. Create configuration file

mkdir flume

cd flume

vi transfer.conf

1.2. Agent1 configuration file

The agent1 has a source that listens for data on port 3333, a channel that buffers event data

in memory, and a sink that logs event data to the console.

agent1.sources = netcatSrc

agent1.channels = memChannel

agent1.sinks = log

agent1.sources.netcatSrc.channels = memChannel

agent1.sinks.log.channel = memChannel

agent1.sources.netcatSrc.type = netcat

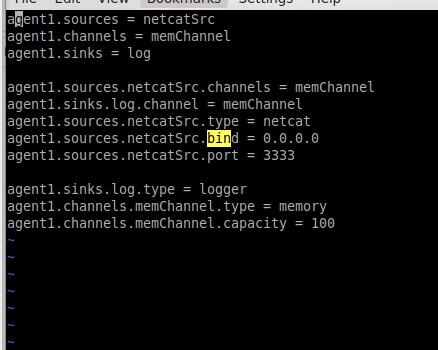
agent1.sources.netcatSrc.bind = 0.0.0.0

agent1.sources.netcatSrc.port = 3333

agent1.sinks.log.type = logger

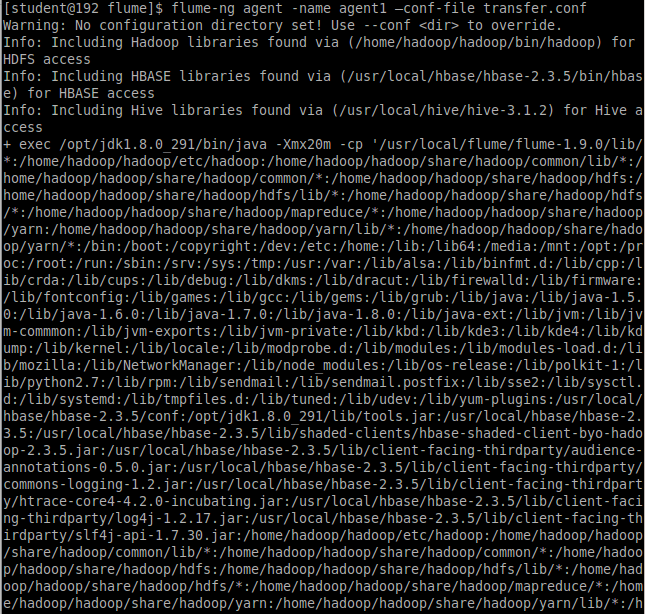
agent1.channels.memChannel.type = memory

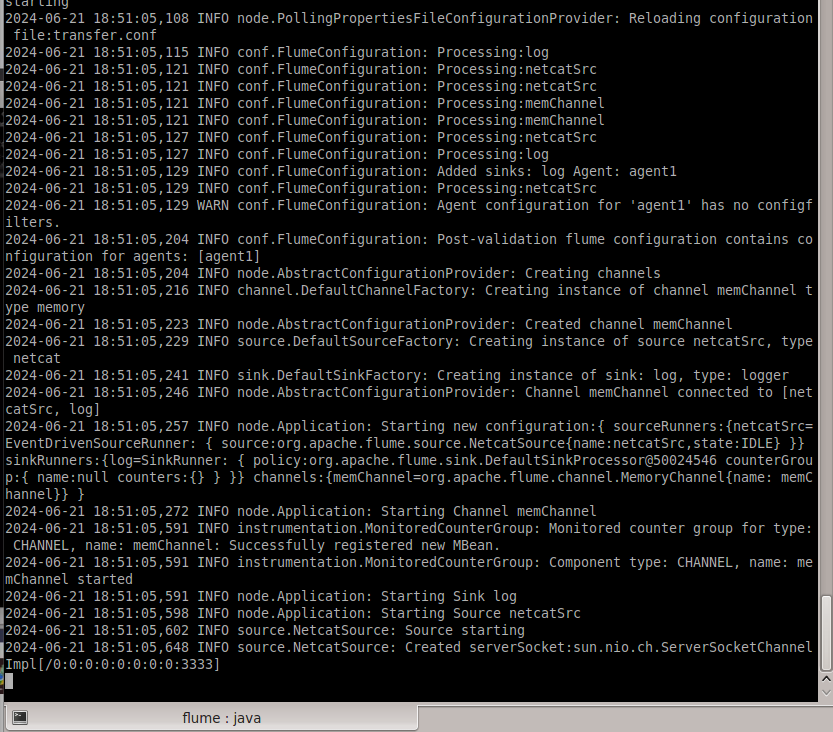
agent1.channels.memChannel.capacity = 100



1.3. Flume agent1 execution

flume-ng agent -name agent1 –conf-file transfer.conf



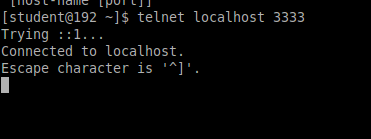


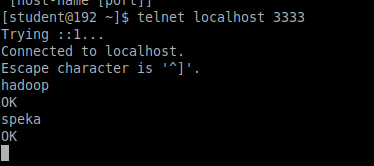
1.4. Open another terminal window and execute the telent command.

telnet localhost 3333

Typing whatever you want...

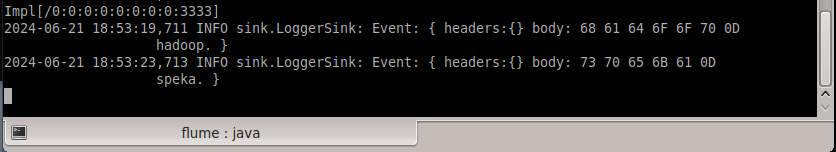
Hadoop





1.5. Check that the message sent to telnet in step 4 is output from the terminal where the

flume agent was executed in step 3



1.6. telnet close with command after ctrl + quit.

^] (ctrl+])

telnet> close



2. Basic Data Transfer with spool directory

This agent 2 is to save the files coming into the spool directory to the local directory.

2.1. Create configuration file

vi transfer\_spool.conf

2.2. Agent2 configuration file

agent2.sources = dirSrc

agent2.channels = memChannel

agent2.sinks = fileSink

agent2.sources.dirSrc.channels = memChannel

agent2.sinks.fileSink.channel = memChannel

agent2.sources.dirSrc.type = spoolDir

agent2.sources.dirSrc.spoolDir = /home/student/data/spool

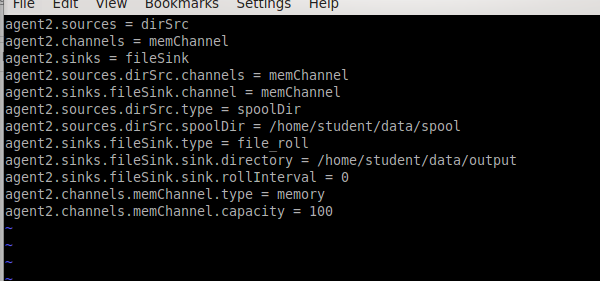
agent2.sinks.fileSink.type = file\_roll

agent2.sinks.fileSink.sink.directory = /home/student/data/output

agent2.sinks.fileSink.sink.rollInterval = 0

agent2.channels.memChannel.type = memory

agent2.channels.memChannel.capacity = 100



File spool ko có

2.3. Flume agent2 execution

flume-ng agent -name agent2 –conf-file transfer\_spool.conf

2.4. Open another terminal window and copy two sql files to spool directory.

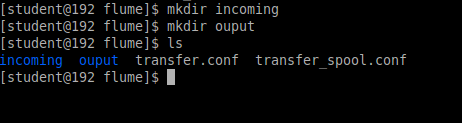
mkdir -p flume/incoming flume/output

cd /home/student/flume/incoming

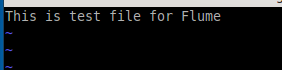
cp ~/Data/\*.txt .

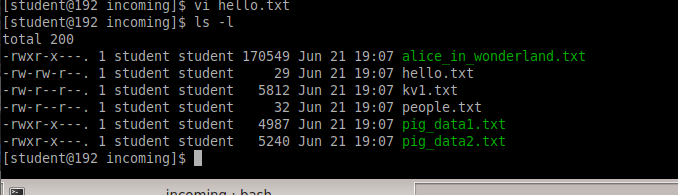
vi hello.txt

This is test file for Flume.









2.5. You can check the message that pig\_data1.txt, pig\_data2.txt, alice\_in\_wonderland.txt,

and hello.txt copied to the spool directory in step 4 are transmitted to the terminal

where Agent 2 is running.

Không có file spool trong máy

2.6. The transferred files are stored as files in the OUTPUT directory

Không có file spool trong máy

3. Using Interceptor

3.1. Create configuration file

vi interceptor.conf

3.2. Agent3 configuration file

agent3.sources = netcatSrc

agent3.channels = memChannel

agent3.sinks = log

agent3.sources.netcatSrc.channels = memChannel

agent1.sinks.log.channel = memChannel

agent1.sources.netcatSrc.type = netcat

agent1.sources.netcatSrc.bind = 0.0.0.0

agent1.sources.netcatSrc.port = 3333

agent1.sinks.log.type = logger

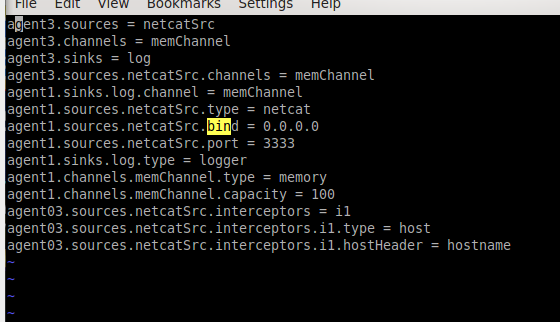
agent1.channels.memChannel.type = memory

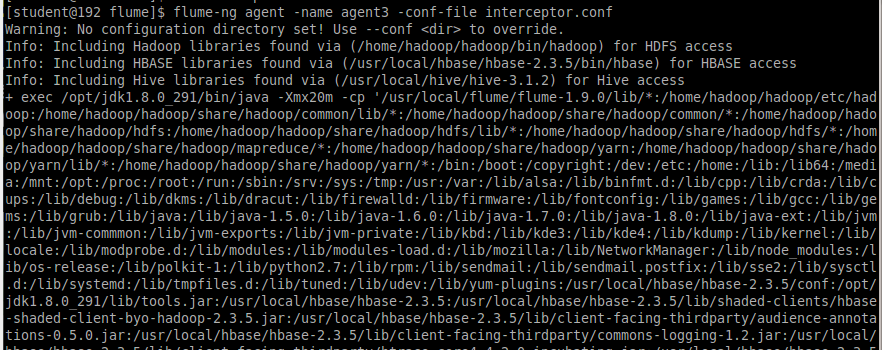
agent1.channels.memChannel.capacity = 100

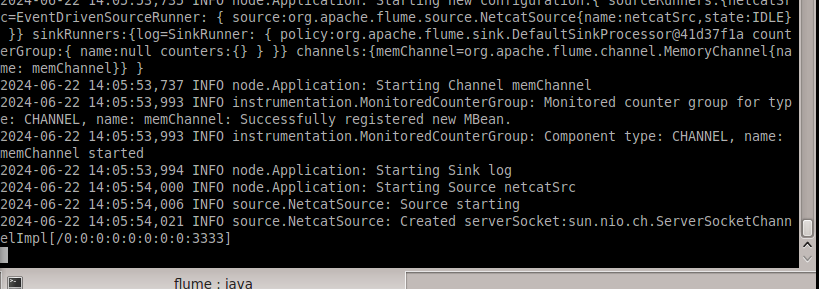
agent03.sources.netcatSrc.interceptors = i1

agent03.sources.netcatSrc.interceptors.i1.type = host

agent03.sources.netcatSrc.interceptors.i1.hostHeader = hostname





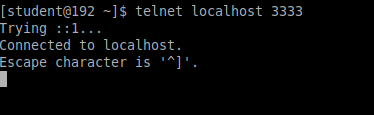


telnet localhost 3333

This is testing Flume with interceptor.

Hadoop

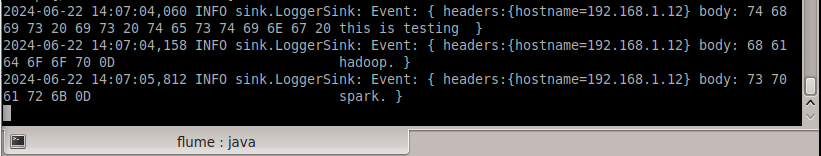
Spark



3.5. The message sent to telnet in step 4 is output from the terminal where the flume

agent was executed in step 3, and it is confirmed that the IP address where the agent

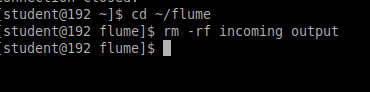
is currently running is inserted into the event header and transmitted

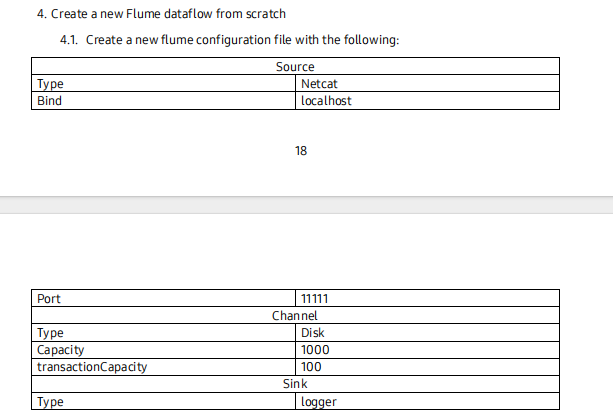


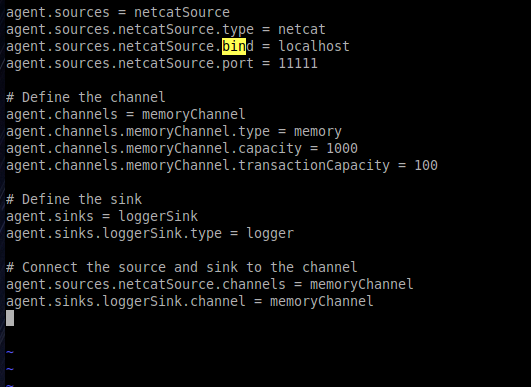
3.6. Delete the temporary directory used for the flume operations.

$cd ~/flume

$rm -rf incoming output







4.2. Start the agent

