

Introduction to Hadoop

SMU, 2019

Nikita Neveditsin

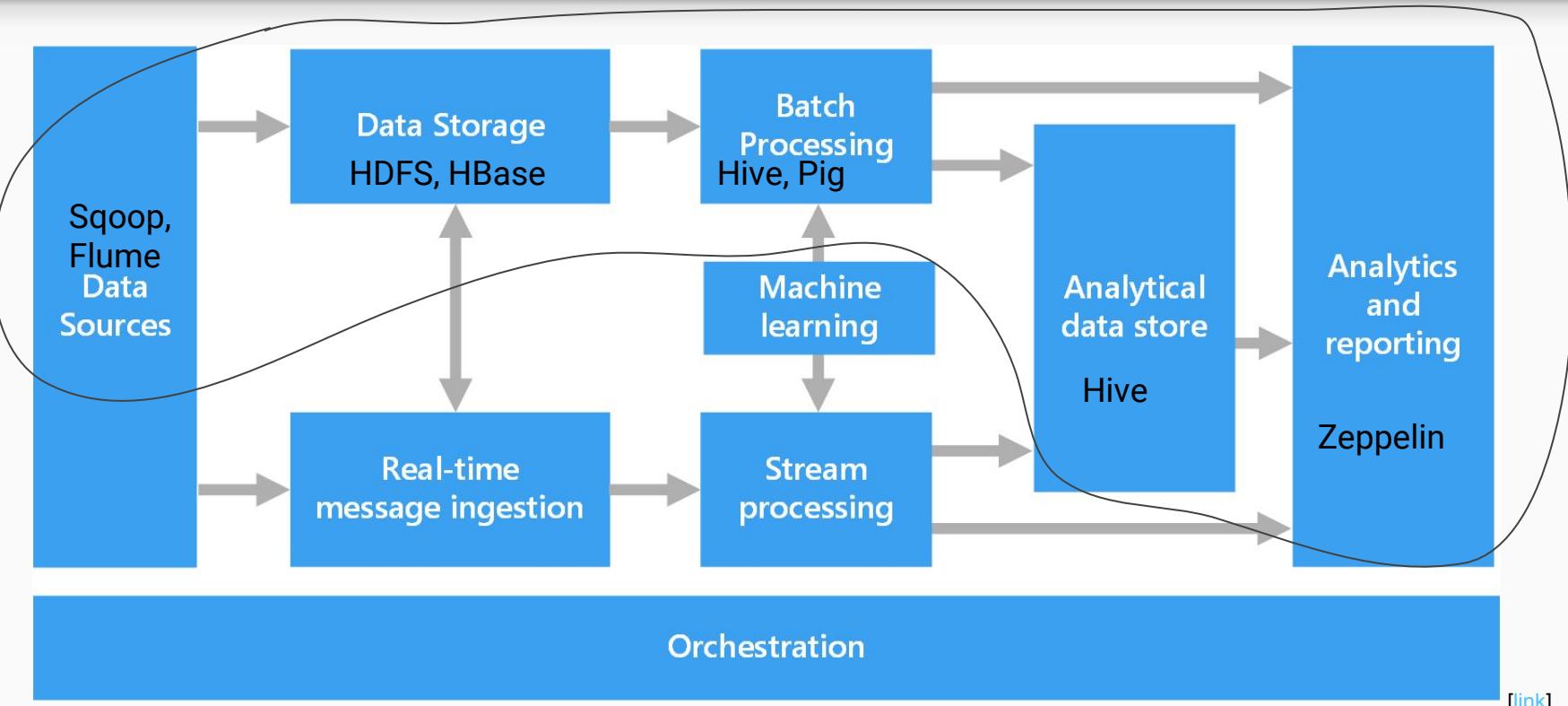
nikita.neveditsin@smu.ca

Workshop outline

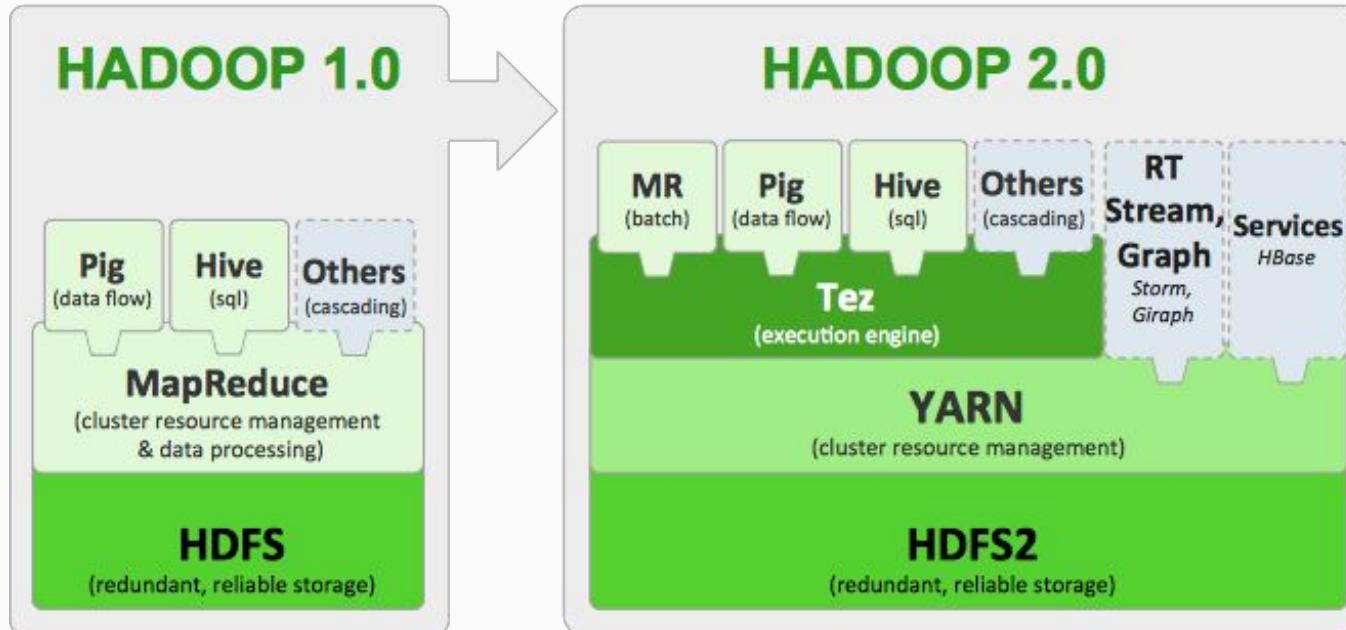
Target: learn basic components of Hadoop. By the end of the workshop students should be able to load data into Hadoop, work with HDFS, understand how Hive works, perform basic data visualizations with Zeppelin

1. HDP Interface: Ambari overview, CLI
2. HDFS: working with files and basic commands
 - Exercise
3. Sqoop
 - Exercise
4. Hive
 - Exercises
5. Zeppelin
 - Exercise

From lecture:



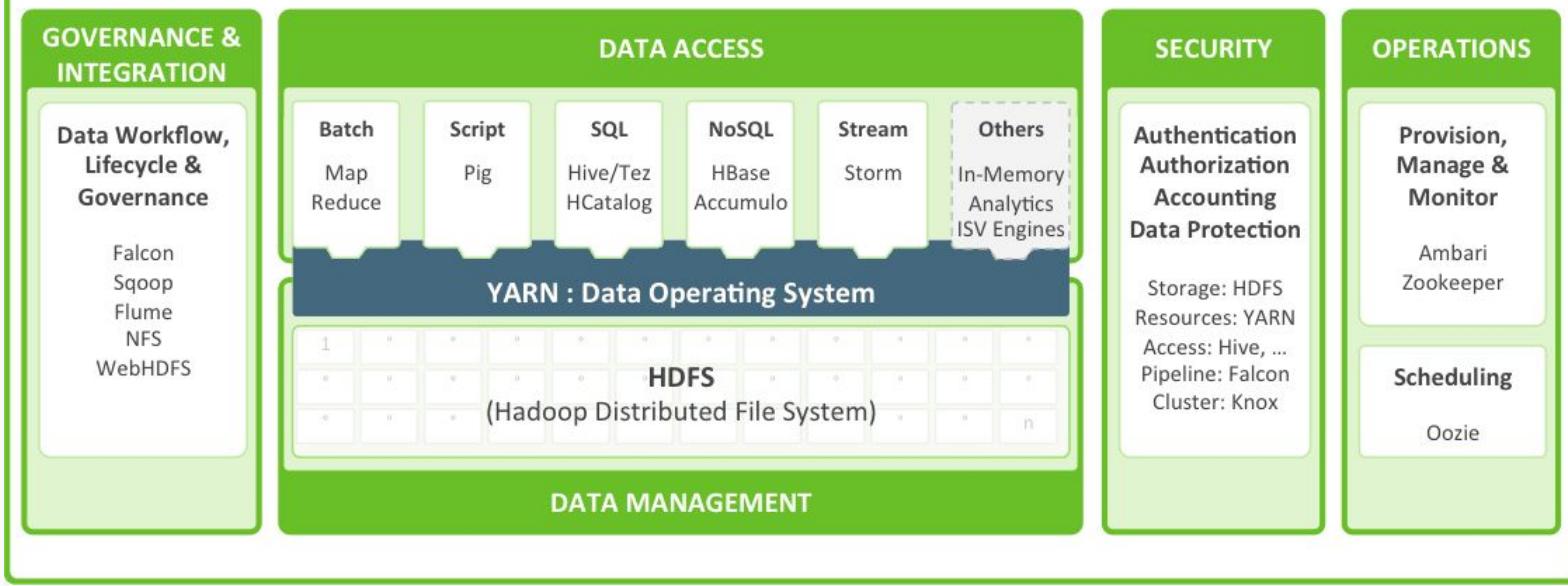
Hadoop 2 architecture



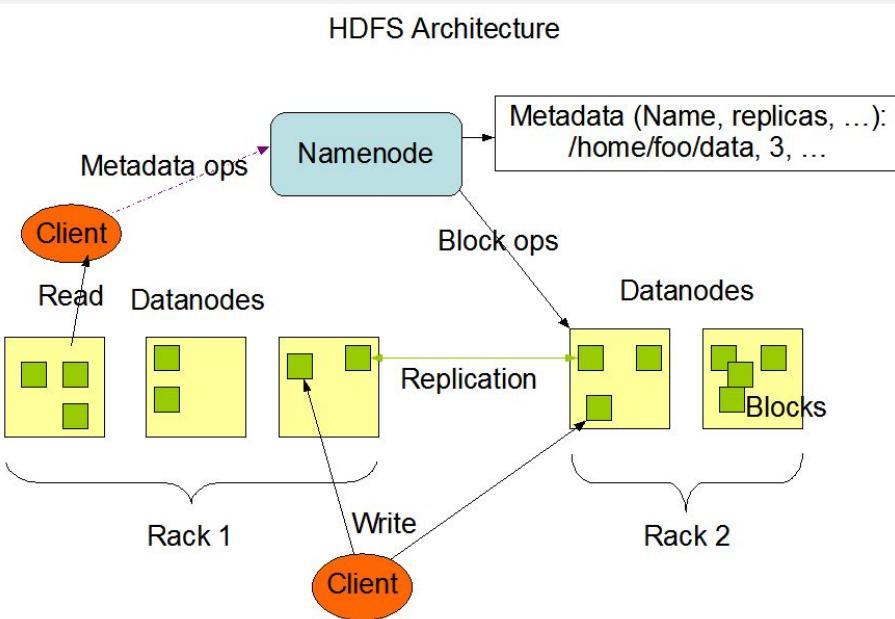
[link]

Hortonworks: overview

Hortonworks Data Platform



HDFS Architecture (from Lecture)

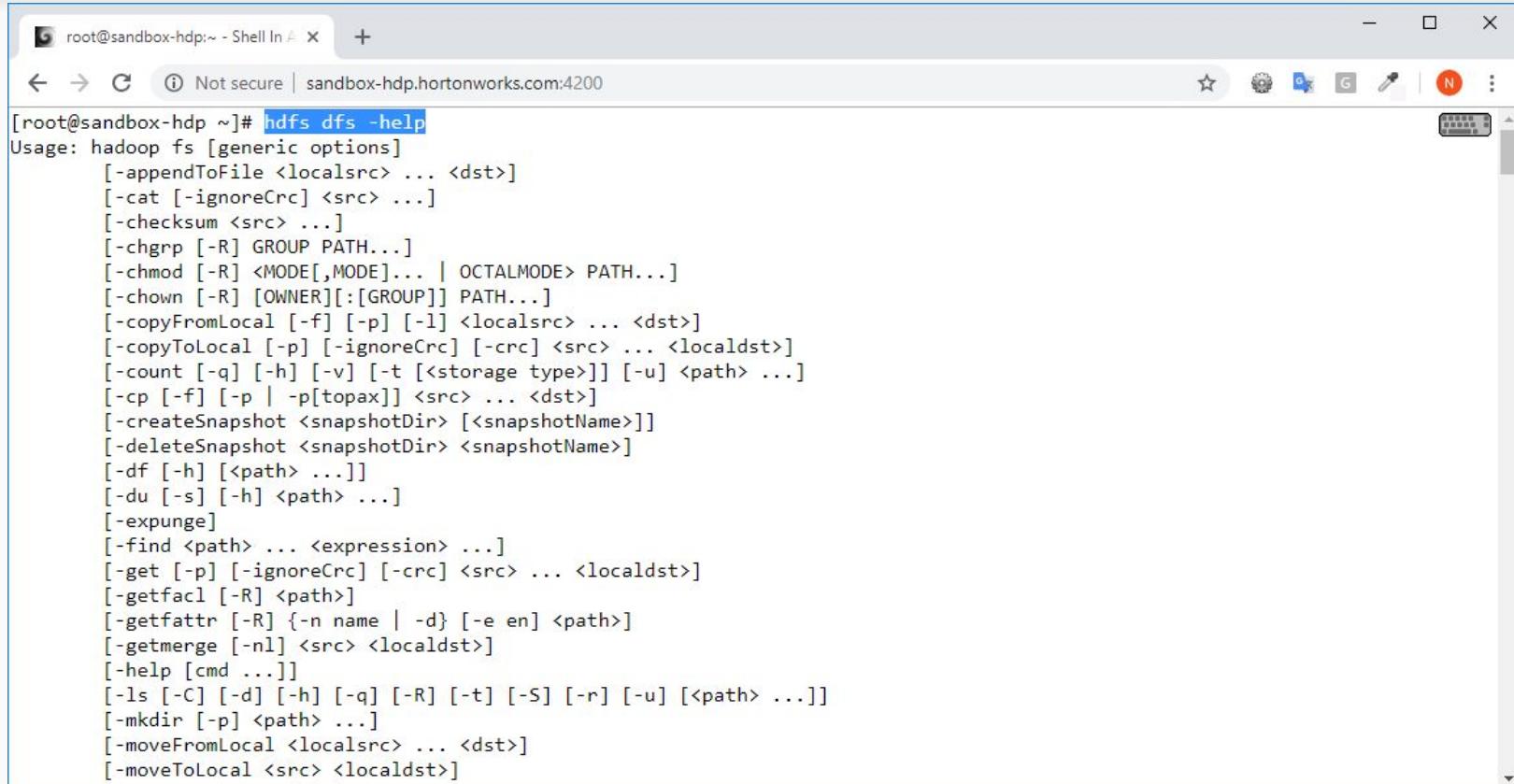


Two major types on **nodes** (servers):

- **Namenode** stores metadata i.e. number of data blocks, replicas and other details. This metadata is available in memory in the master for faster retrieval of data
- **Datanodes** store actual data in HDFS. They perform read and write operations as per the request of the client [I]

[link]

HDFS Interface



The screenshot shows a terminal window titled "root@sandbox-hdp:~ - Shell In A Box" with the URL "Not secure | sandbox-hdp.hortonworks.com:4200". The window displays the output of the command `hdfs dfs -help`, which lists various HDFS command-line options:

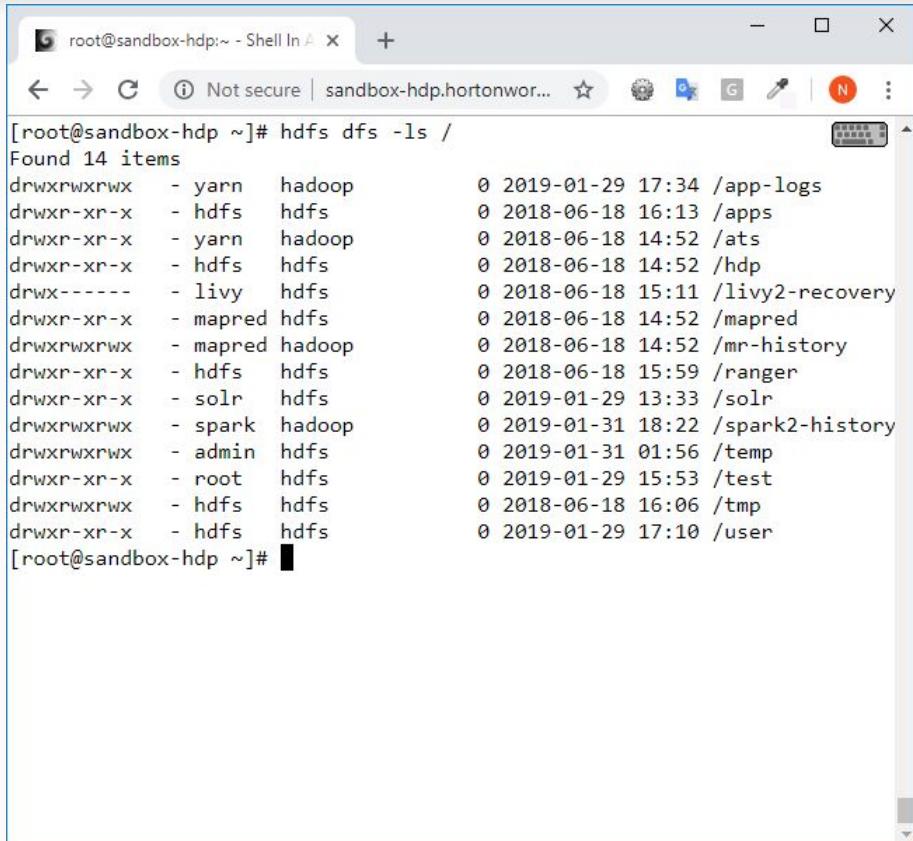
```
[root@sandbox-hdp ~]# hdfs dfs -help
Usage: hadoop fs [generic options]
      [-appendToFile <localsrc> ... <dst>]
      [-cat [-ignoreCrc] <src> ...]
      [-checksum <src> ...]
      [-chgrp [-R] GROUP PATH...]
      [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
      [-chown [-R] [OWNER][:[GROUP]] PATH...]
      [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
      [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-count [-q] [-h] [-v] [-t [<storage type>]] [-u] <path> ...]
      [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
      [-createSnapshot <snapshotDir> [<snapshotName>]]
      [-deleteSnapshot <snapshotDir> <snapshotName>]
      [-df [-h] [<path> ...]]
      [-du [-s] [-h] <path> ...]
      [-expunge]
      [-find <path> ... <expression> ...]
      [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
      [-getfacl [-R] <path>]
      [-getattr [-R] {-n name | -d} [-e en] <path>]
      [-getmerge [-n1] <src> <localdst>]
      [-help [cmd ...]]
      [-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [<path> ...]]
      [-mkdir [-p] <path> ...]
      [-moveFromLocal <localsrc> ... <dst>]
      [-moveToLocal <src> <localdst>]
```

HDFS Interface (cont-d)

Syntax:

hadoop fs -command args

- A lot of unix-like commands (**ls**, **cp**, **rm**, **mv**, **mkdir**, etc.)



```
[root@sandbox-hdp ~]# hdfs dfs -ls /
Found 14 items
drwxrwxrwx  - yarn   hadoop      0 2019-01-29 17:34 /app-logs
drwxr-xr-x  - hdfs   hdfs       0 2018-06-18 16:13 /apps
drwxr-xr-x  - yarn   hadoop      0 2018-06-18 14:52 /ats
drwxr-xr-x  - hdfs   hdfs       0 2018-06-18 14:52 /hdp
drwx-----  - livy   hdfs      0 2018-06-18 15:11 /livy2-recovery
drwxr-xr-x  - mapred hdfs      0 2018-06-18 14:52 /mapred
drwxrwxrwx  - mapred hadoop    0 2018-06-18 14:52 /mr-history
drwxr-xr-x  - hdfs   hdfs      0 2018-06-18 15:59 /ranger
drwxr-xr-x  - solr   hdfs      0 2019-01-29 13:33 /solr
drwxrwxrwx  - spark   hadoop    0 2019-01-31 18:22 /spark2-history
drwxrwxrwx  - admin   hdfs      0 2019-01-31 01:56 /temp
drwxr-xr-x  - root   hdfs      0 2019-01-29 15:53 /test
drwxrwxrwx  - hdfs   hdfs      0 2018-06-18 16:06 /tmp
drwxr-xr-x  - hdfs   hdfs      0 2019-01-29 17:10 /user
[root@sandbox-hdp ~]#
```

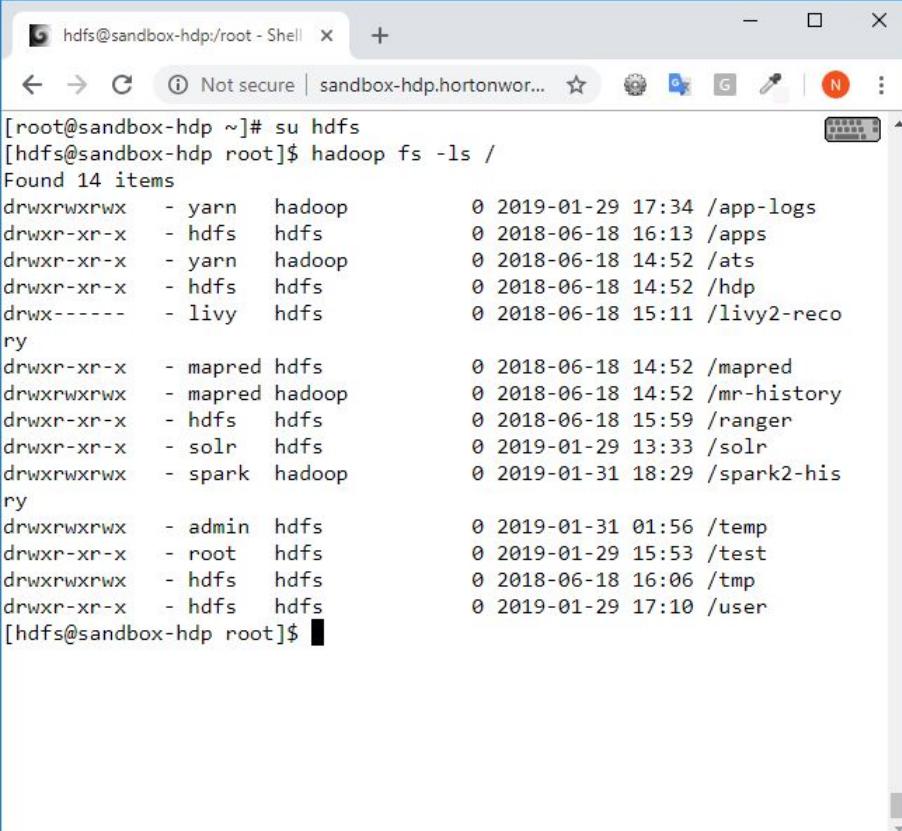
HDFS: superuser

hdfs has “hdfs” superuser
(hdfs user can do
everything inside the file
system)

From sandbox type:

su hdfs

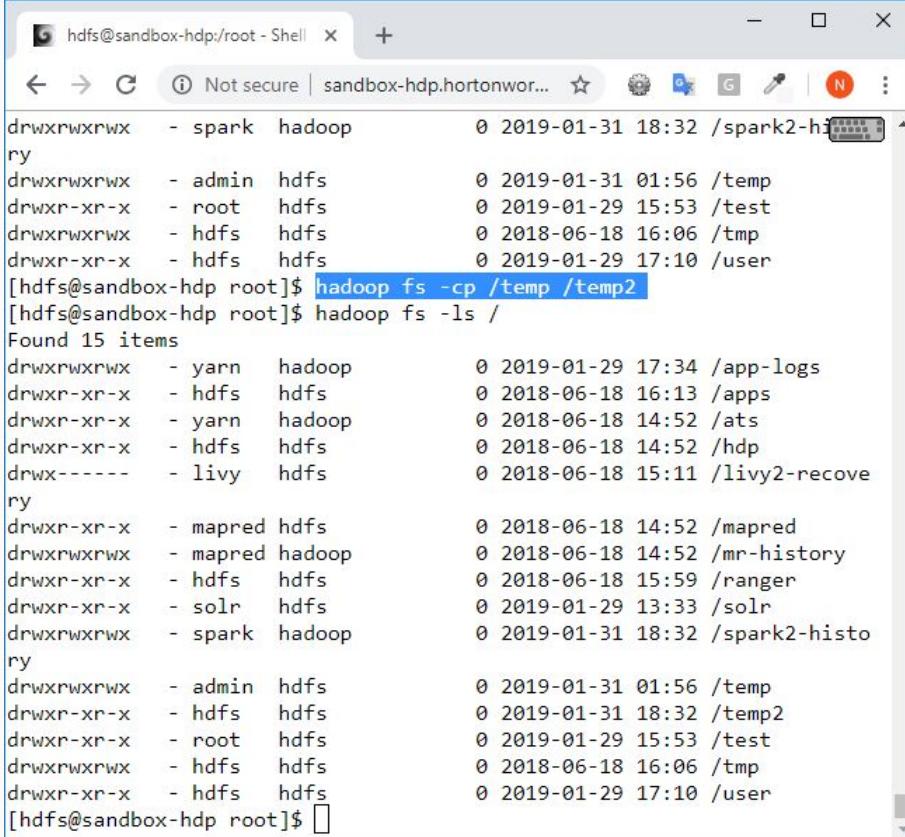
This will switch your user
from **root** to **hdfs**



```
[root@sandbox-hdp ~]# su hdfs
[hdfs@sandbox-hdp root]$ hadoop fs -ls /
Found 14 items
drwxrwxrwx  - yarn   hadoop      0 2019-01-29 17:34 /app-logs
drwxr-xr-x  - hdfs   hdfs       0 2018-06-18 16:13 /apps
drwxr-xr-x  - yarn   hadoop      0 2018-06-18 14:52 /ats
drwxr-xr-x  - hdfs   hdfs       0 2018-06-18 14:52 /hdp
drwx-----  - livy   hdfs      0 2018-06-18 15:11 /livy2-reco
ry
drwxr-xr-x  - mapred hdfs      0 2018-06-18 14:52 /mapred
drwxrwxrwx  - mapred hadoop    0 2018-06-18 14:52 /mr-history
drwxr-xr-x  - hdfs   hdfs      0 2018-06-18 15:59 /ranger
drwxr-xr-x  - solr   hdfs      0 2019-01-29 13:33 /solr
drwxrwxrwx  - spark   hadoop    0 2019-01-31 18:29 /spark2-his
ry
drwxrwxrwx  - admin   hdfs      0 2019-01-31 01:56 /temp
drwxr-xr-x  - root    hdfs      0 2019-01-29 15:53 /test
drwxrwxrwx  - hdfs   hdfs      0 2018-06-18 16:06 /tmp
drwxr-xr-x  - hdfs   hdfs      0 2019-01-29 17:10 /user
[hdfs@sandbox-hdp root]$
```

HDFS: copy files

`hadoop fs -cp src dst`



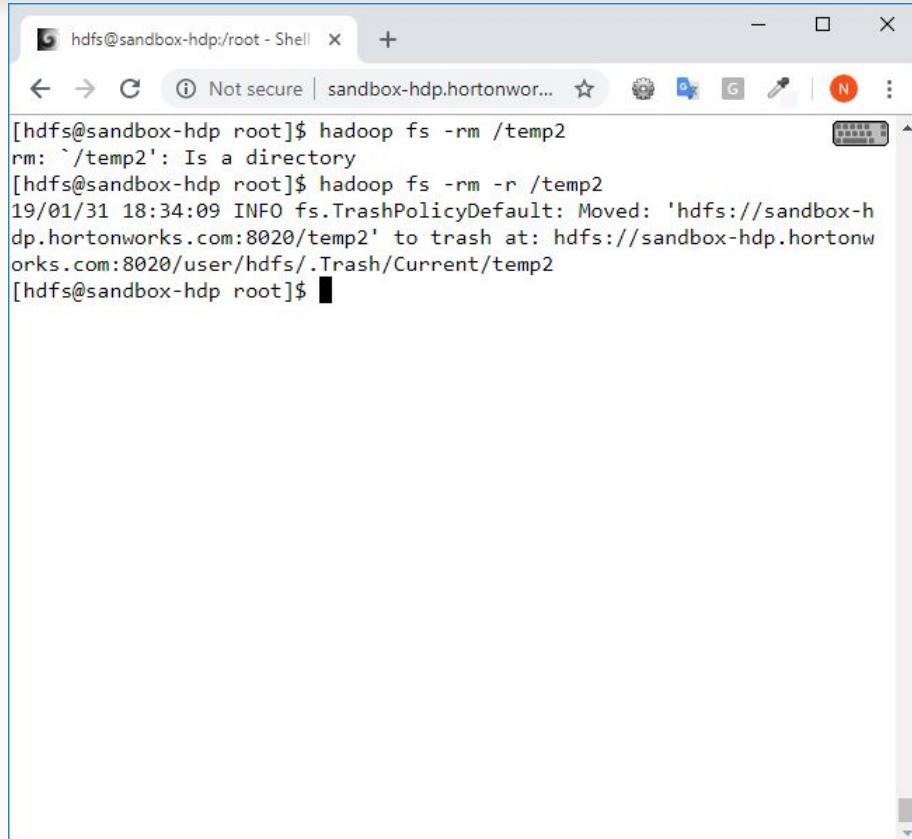
The screenshot shows a terminal window titled "hdfs@sandbox-hdp:/root - Shell". The user has run the command `hadoop fs -ls /` which lists 15 items in the root directory. The user then runs `hadoop fs -cp /temp /temp2` to copy the contents of the /temp directory to /temp2. Finally, the user runs `hadoop fs -ls /` again to verify the contents.

```
[hdfs@sandbox-hdp root]$ hadoop fs -ls /
Found 15 items
drwxrwxrwx - spark  hadoop      0 2019-01-31 18:32 /spark2-histo
ry
drwxrwxrwx - admin   hdfs      0 2019-01-31 01:56 /temp
drwxr-xr-x - root    hdfs      0 2019-01-29 15:53 /test
drwxrwxrwx - hdfs   hdfs      0 2018-06-18 16:06 /tmp
drwxr-xr-x - hdfs   hdfs      0 2019-01-29 17:10 /user
[hdfs@sandbox-hdp root]$ hadoop fs -cp /temp /temp2
[hdfs@sandbox-hdp root]$ hadoop fs -ls /
Found 15 items
drwxrwxrwx - yarn    hadoop     0 2019-01-29 17:34 /app-logs
drwxr-xr-x - hdfs   hdfs      0 2018-06-18 16:13 /apps
drwxr-xr-x - yarn    hadoop     0 2018-06-18 14:52 /ats
drwxr-xr-x - hdfs   hdfs      0 2018-06-18 14:52 /hdp
drwx----- - livy    hdfs      0 2018-06-18 15:11 /livy2-recove
ry
drwxr-xr-x - mapred  hdfs      0 2018-06-18 14:52 /mapred
drwxrwxrwx - mapred  hadoop     0 2018-06-18 14:52 /mr-history
drwxr-xr-x - hdfs   hdfs      0 2018-06-18 15:59 /ranger
drwxr-xr-x - solr   hdfs      0 2019-01-29 13:33 /solr
drwxrwxrwx - spark   hadoop     0 2019-01-31 18:32 /spark2-histo
ry
drwxrwxrwx - admin   hdfs      0 2019-01-31 01:56 /temp
drwxr-xr-x - hdfs   hdfs      0 2019-01-31 18:32 /temp2
drwxr-xr-x - root    hdfs      0 2019-01-29 15:53 /test
drwxrwxrwx - hdfs   hdfs      0 2018-06-18 16:06 /tmp
drwxr-xr-x - hdfs   hdfs      0 2019-01-29 17:10 /user
[hdfs@sandbox-hdp root]$
```

HDFS: remove files

[hadoop fs -rm file](#)

- **-r** if need to remove directory

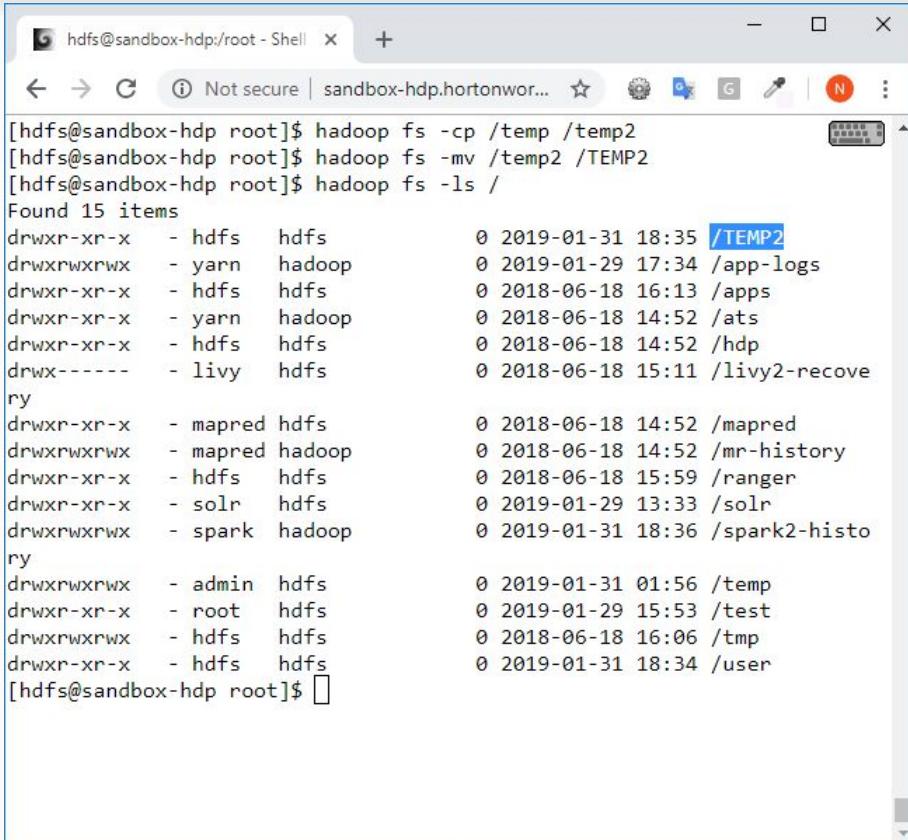


The screenshot shows a terminal window titled "hdbs@sandbox-hdp:root - Shell". The URL bar indicates "Not secure | sandbox-hdp.hortonwor...". The terminal output is as follows:

```
[hdbs@sandbox-hdp root]$ hadoop fs -rm /temp2
rm: `/temp2': Is a directory
[hdbs@sandbox-hdp root]$ hadoop fs -rm -r /temp2
19/01/31 18:34:09 INFO fs.TrashPolicyDefault: Moved: 'hdfs://sandbox-hdp.hortonworks.com:8020/temp2' to trash at: hdfs://sandbox-hdp.hortonworks.com:8020/user/hdfs/.Trash/Current/temp2
[hdbs@sandbox-hdp root]$ █
```

HDFS: move or rename

`hadoop fs -mv file1 file2`



The screenshot shows a terminal window titled "hdfs@sandbox-hdp/root - Shell". The window contains the following command history and output:

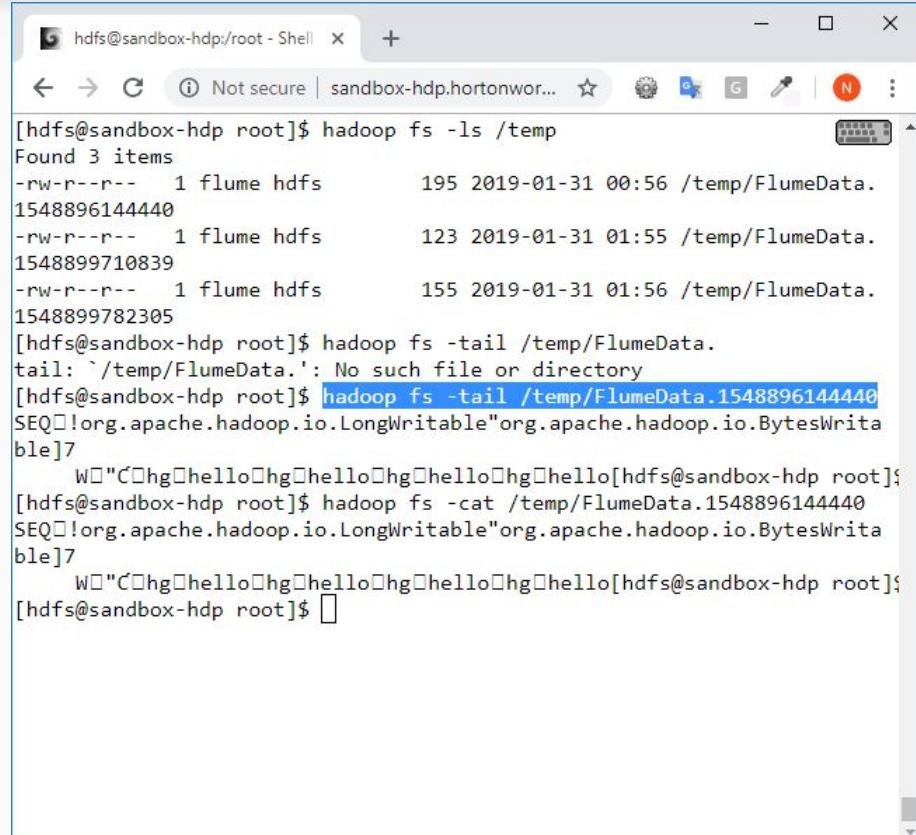
```
[hdfs@sandbox-hdp root]$ hadoop fs -cp /temp /temp2
[hdfs@sandbox-hdp root]$ hadoop fs -mv /temp2 /TEMP2
[hdfs@sandbox-hdp root]$ hadoop fs -ls /
Found 15 items
drwxr-xr-x  - hdfs  hdfs          0 2019-01-31 18:35 /TEMP2
drwxrwxrwx  - yarn   hadoop        0 2019-01-29 17:34 /app-logs
drwxr-xr-x  - hdfs  hdfs          0 2018-06-18 16:13 /apps
drwxr-xr-x  - yarn   hadoop        0 2018-06-18 14:52 /ats
drwxr-xr-x  - hdfs  hdfs          0 2018-06-18 14:52 /hdp
drwx-----  - livy   hdfs          0 2018-06-18 15:11 /livy2-recovery
drwxr-xr-x  - mapred hdfs         0 2018-06-18 14:52 /mapred
drwxrwxrwx  - mapred hadoop       0 2018-06-18 14:52 /mr-history
drwxr-xr-x  - hdfs  hdfs          0 2018-06-18 15:59 /ranger
drwxr-xr-x  - solr   hdfs          0 2019-01-29 13:33 /solr
drwxrwxrwx  - spark  hadoop       0 2019-01-31 18:36 /spark2-history
drwxrwxrwx  - admin   hdfs         0 2019-01-31 01:56 /temp
drwxr-xr-x  - root    hdfs         0 2019-01-29 15:53 /test
drwxrwxrwx  - hdfs  hdfs          0 2018-06-18 16:06 /tmp
drwxr-xr-x  - hdfs  hdfs          0 2019-01-31 18:34 /user
[hdfs@sandbox-hdp root]$
```

HDFS: cat and tail

[hadoop fs -cat file](#)

[hadoop fs -tail file](#)

Like in Linux/*NIX

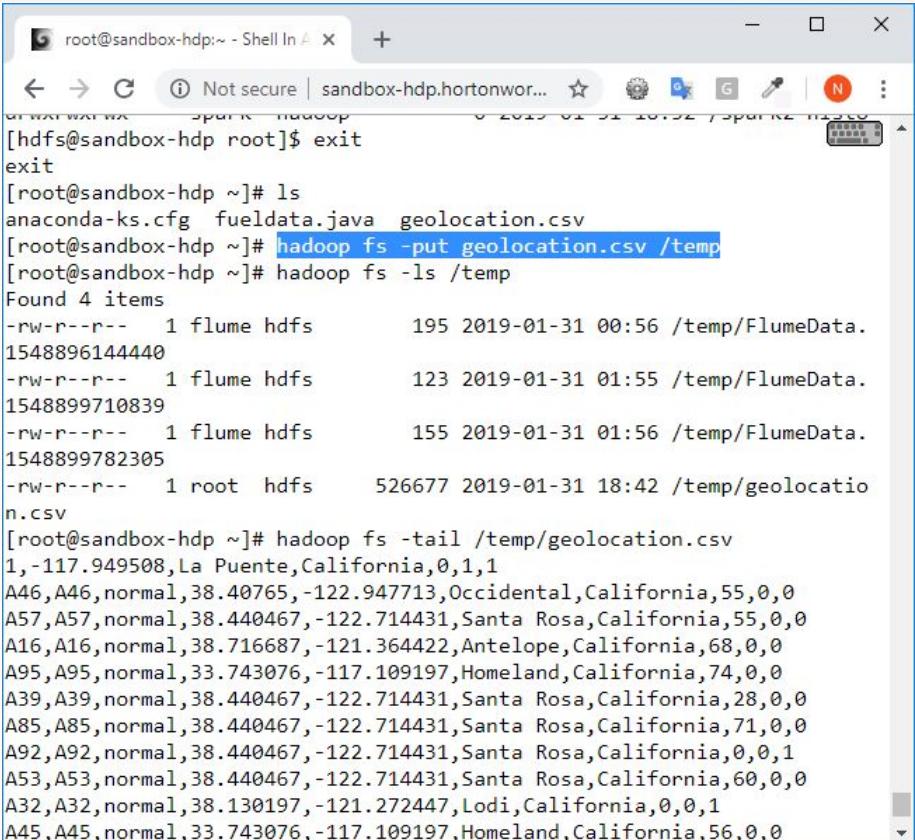


The screenshot shows a web browser window with a terminal-like interface. The address bar says "Not secure | sandbox-hdp.hortonwor...". The terminal output is as follows:

```
[hdfs@sandbox-hdp root]$ hadoop fs -ls /temp
Found 3 items
-rw-r--r-- 1 flume hdfs      195 2019-01-31 00:56 /temp/FlumeData.1548896144440
-rw-r--r-- 1 flume hdfs      123 2019-01-31 01:55 /temp/FlumeData.1548899710839
-rw-r--r-- 1 flume hdfs      155 2019-01-31 01:56 /temp/FlumeData.1548899782305
[hdfs@sandbox-hdp root]$ hadoop fs -tail /temp/FlumeData.
tail: '/temp/FlumeData.': No such file or directory
[hdfs@sandbox-hdp root]$ hadoop fs -tail /temp/FlumeData.1548896144440
SEQ!org.apache.hadoop.io.LongWritable"org.apache.hadoop.io.BytesWritable]7
W\0"C\0hg\0hello\0hg\0hello\0hg\0hello\0hg\0hello[hdfs@sandbox-hdp root]$
[hdfs@sandbox-hdp root]$ hadoop fs -cat /temp/FlumeData.1548896144440
SEQ!org.apache.hadoop.io.LongWritable"org.apache.hadoop.io.BytesWritable]7
W\0"C\0hg\0hello\0hg\0hello\0hg\0hello\0hg\0hello[hdfs@sandbox-hdp root]$
[hdfs@sandbox-hdp root]$ 
```

HDFS: copy from local file system

`hadoop fs -put local_file HDFS_path`

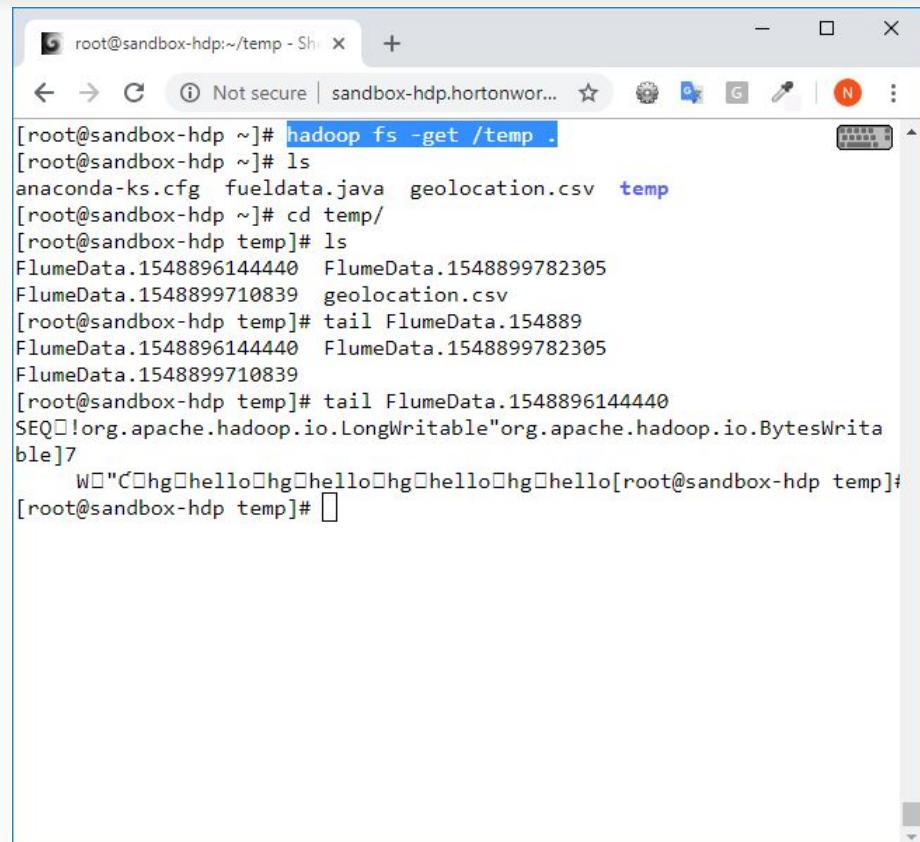


The screenshot shows a terminal window titled "root@sandbox-hdp:~ - Shell In A..." with the command history and output as follows:

```
[root@sandbox-hdp ~]$ exit  
exit  
[root@sandbox-hdp ~]# ls  
anaconda-ks.cfg fueldata.java geolocation.csv  
[root@sandbox-hdp ~]# hadoop fs -put geolocation.csv /temp  
[root@sandbox-hdp ~]# hadoop fs -ls /temp  
Found 4 items  
-rw-r--r-- 1 flume hdfs 195 2019-01-31 00:56 /temp/FlumeData.  
1548896144440  
-rw-r--r-- 1 flume hdfs 123 2019-01-31 01:55 /temp/FlumeData.  
1548899710839  
-rw-r--r-- 1 flume hdfs 155 2019-01-31 01:56 /temp/FlumeData.  
1548899782305  
-rw-r--r-- 1 root hdfs 526677 2019-01-31 18:42 /temp/geolocation.csv  
[root@sandbox-hdp ~]# hadoop fs -tail /temp/geolocation.csv  
1,-117.949508,La Puente,California,0,1,1  
A46,A46,normal,38.40765,-122.947713,Occidental,California,55,0,0  
A57,A57,normal,38.440467,-122.714431,Santa Rosa,California,55,0,0  
A16,A16,normal,38.716687,-121.364422,Antelope,California,68,0,0  
A95,A95,normal,33.743076,-117.109197,Homeland,California,74,0,0  
A39,A39,normal,38.440467,-122.714431,Santa Rosa,California,28,0,0  
A85,A85,normal,38.440467,-122.714431,Santa Rosa,California,71,0,0  
A92,A92,normal,38.440467,-122.714431,Santa Rosa,California,0,0,1  
A53,A53,normal,38.440467,-122.714431,Santa Rosa,California,60,0,0  
A32,A32,normal,38.130197,-121.272447,Lodi,California,0,0,1  
A45,A45,normal,33.743076,-117.109197,Homeland,California,56,0,0
```

HDFS: copy from HDFS to local file system

`hadoop fs -get HDFS_path local_file`



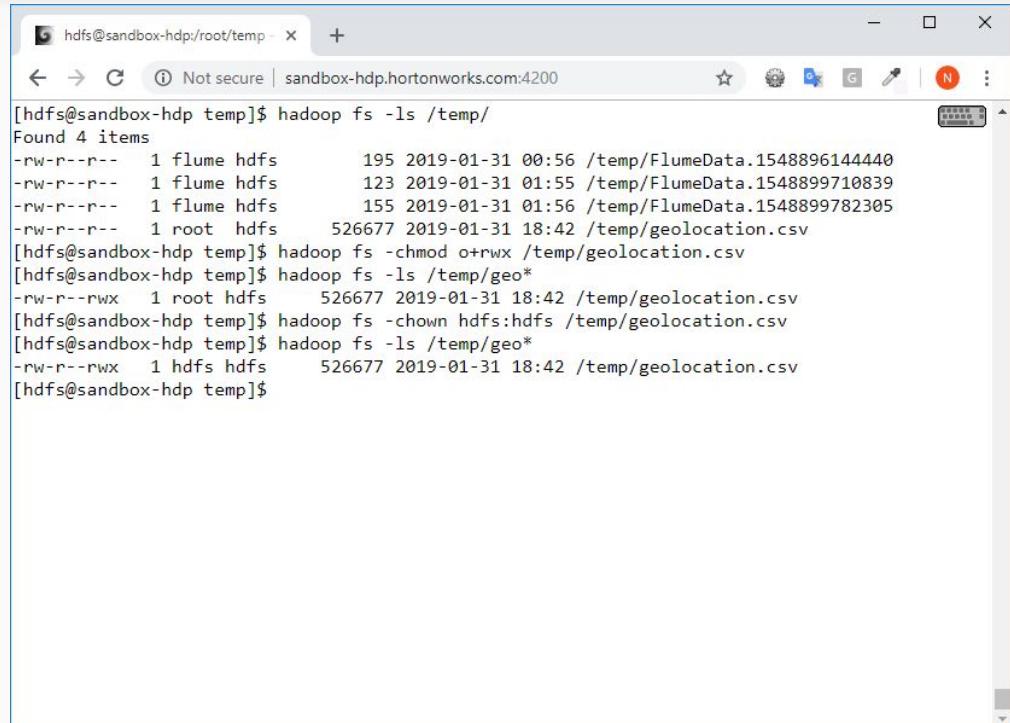
The screenshot shows a terminal window titled "root@sandbox-hdp:~/temp - Sh" with the following session history:

```
[root@sandbox-hdp ~]# hadoop fs -get /temp .
[root@sandbox-hdp ~]# ls
anaconda-ks.cfg  fueldata.java  geolocation.csv  temp
[root@sandbox-hdp ~]# cd temp/
[root@sandbox-hdp temp]# ls
FlumeData.1548896144440  FlumeData.1548899782305
FlumeData.1548899710839  geolocation.csv
[root@sandbox-hdp temp]# tail FlumeData.154889
FlumeData.1548896144440  FlumeData.1548899782305
FlumeData.1548899710839
[root@sandbox-hdp temp]# tail FlumeData.1548896144440
SEQ!org.apache.hadoop.io.LongWritable"org.apache.hadoop.io.BytesWrit
ble]7
W" hg hello hg hello hg hello hg hello[roo
[root@sandbox-hdp temp]#
```

HDFS: permissions

[hadoop fs -chown user:group file](#)

[hadoop fs -chmod o+rwx file](#)



A screenshot of a terminal window titled "hdfs@sandbox-hdp:root/temp". The window shows a series of HDFS commands being run and their outputs. The commands include listing files, changing ownership, changing permissions, and listing files again.

```
[hdfs@sandbox-hdp temp]$ hadoop fs -ls /temp/
Found 4 items
-rw-r--r-- 1 flume hdfs      195 2019-01-31 00:56 /temp/FlumeData.1548896144440
-rw-r--r-- 1 flume hdfs      123 2019-01-31 01:55 /temp/FlumeData.1548899710839
-rw-r--r-- 1 flume hdfs      155 2019-01-31 01:56 /temp/FlumeData.1548899782305
-rw-r--r-- 1 root  hdfs     526677 2019-01-31 18:42 /temp/geolocation.csv
[hdfs@sandbox-hdp temp]$ hadoop fs -chmod o+rwx /temp/geolocation.csv
[hdfs@sandbox-hdp temp]$ hadoop fs -ls /temp/geo*
-rw-r--rwx 1 root  hdfs     526677 2019-01-31 18:42 /temp/geolocation.csv
[hdfs@sandbox-hdp temp]$ hadoop fs -chown hdfs:hdfs /temp/geolocation.csv
[hdfs@sandbox-hdp temp]$ hadoop fs -ls /temp/geo*
-rw-r--rwx 1 hdfs hdfs     526677 2019-01-31 18:42 /temp/geolocation.csv
[hdfs@sandbox-hdp temp]$
```

HDFS: “Files View” in Ambari

The screenshot shows the Ambari interface for managing HDFS files. The top navigation bar includes tabs for Dashboard, Services, Hosts, Alerts, Admin, and a user dropdown for 'admin'. A sidebar on the left provides navigation links for YARN Queue Manager, Files View (which is currently selected), Hive View, Hive View 2.0, Pig View, Tez View, and Workflow Manager. The main content area displays a table of 15 files or folders. The columns are Name, Size, Last Modified, and Owner. The table lists the following entries:

| Name | Size | Last Modified | Owner |
|------------------------------------|------|------------------|--------|
| TEMP2 | -- | 2019-01-31 14:35 | hdfs |
| app-logs | -- | 2019-01-29 13:34 | yarn |
| apps | -- | 2018-06-18 13:13 | hdfs |
| ats | -- | 2018-06-18 11:52 | yarn |
| hdp | -- | 2018-06-18 11:52 | hdfs |
| livy2-recovery | -- | 2018-06-18 12:11 | livy |
| mapred | -- | 2018-06-18 11:52 | mapred |
| mr-history | -- | 2018-06-18 11:52 | mapred |
| ranger | -- | 2018-06-18 12:59 | hdfs |
| sandbox-hdp.hortonworks.com:8080/# | -- | 2019-01-29 09:33 | solr |

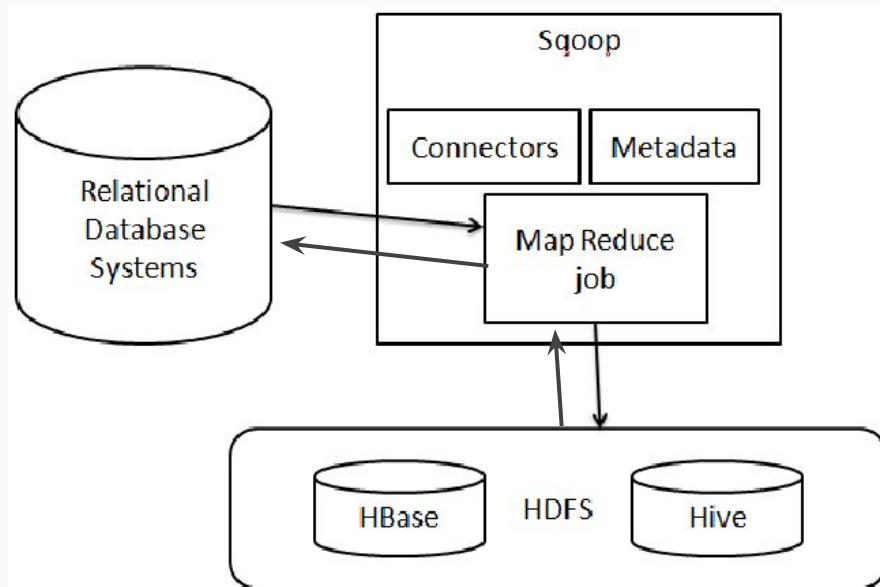
Exercise

1. Download a dataset (US census, 1996):

<http://wolly.cs.smu.ca/tmp/adult.data>

2. Create a folder “**/dsets/census**” in HDFS, put the data there
3. Permissions:
 - The dataset file should have **read/write** permissions for everyone
 - **dsets** folder should have read/write permissions for everyone as well

Sqoop



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

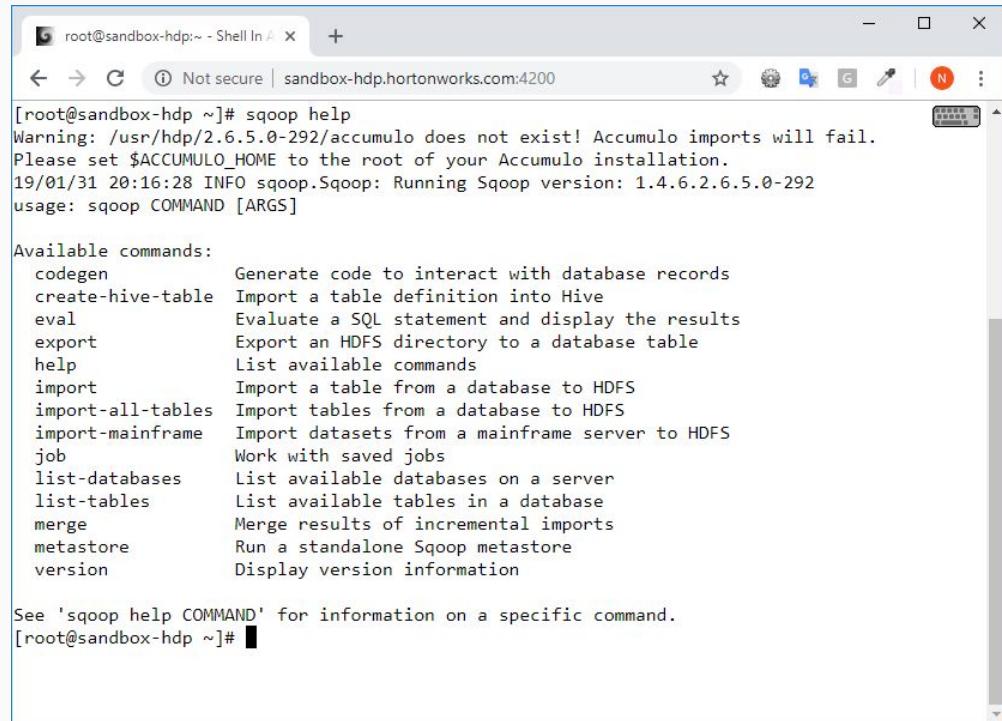
[\[link\]](#)

Sqoop is a command-line tool

sqoop command args

Most commonly used commands:

- import
- export (Note: a table with schema have to be defined in the target database)



The screenshot shows a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "Not secure | sandbox-hdp.hortonworks.com:4200". The terminal displays the following text:

```
[root@sandbox-hdp ~]# sqoop help
Warning: /usr/hdp/2.6.5.0-292/accumulo does not exist! Accumulo imports will fail.
Please set $ACCUMULO_HOME to the root of your Accumulo installation.
19/01/31 20:16:28 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6.2.6.5.0-292
usage: sqoop COMMAND [ARGS]

Available commands:
  codegen          Generate code to interact with database records
  create-hive-table Import a table definition into Hive
  eval             Evaluate a SQL statement and display the results
  export            Export an HDFS directory to a database table
  help              List available commands
  import            Import a table from a database to HDFS
  import-all-tables Import tables from a database to HDFS
  import-mainframe  Import datasets from a mainframe server to HDFS
  job               Work with saved jobs
  list-databases   List available databases on a server
  list-tables      List available tables in a database
  merge             Merge results of incremental imports
  metastore        Run a standalone Sqoop metastore
  version          Display version information

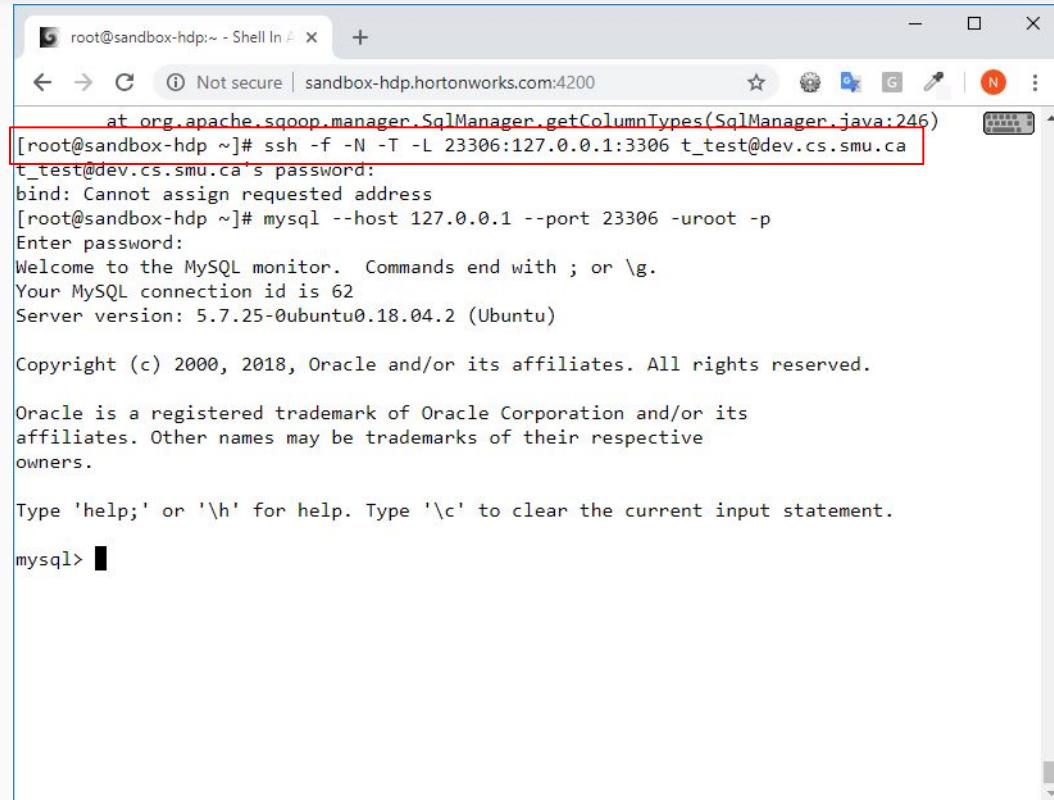
See 'sqoop help COMMAND' for information on a specific command.
[root@sandbox-hdp ~]#
```

Sqoop: simple export example

Step 1:

Make sure that you can connect to the database you are pulling data from

In our example we set an [ssh tunnel](#) to **dev.cs.smu.ca** server to access a database there



The screenshot shows a browser window with a terminal-like interface. The title bar says "root@sandbox-hdp:~ - Shell In". The address bar says "Not secure | sandbox-hdp.hortonworks.com:4200". The terminal content is as follows:

```
at org.apache.sqoop.manager.SqlManager.getColumnTypes(SqlManager.java:246)
[root@sandbox-hdp ~]# ssh -f -N -T -L 23306:127.0.0.1:3306 t_test@dev.cs.smu.ca
t_test@dev.cs.smu.ca's password:
bind: Cannot assign requested address
[root@sandbox-hdp ~]# mysql --host 127.0.0.1 --port 23306 -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 62
Server version: 5.7.25-0ubuntu0.18.04.2 (Ubuntu)

Copyright (c) 2000, 2018, 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.

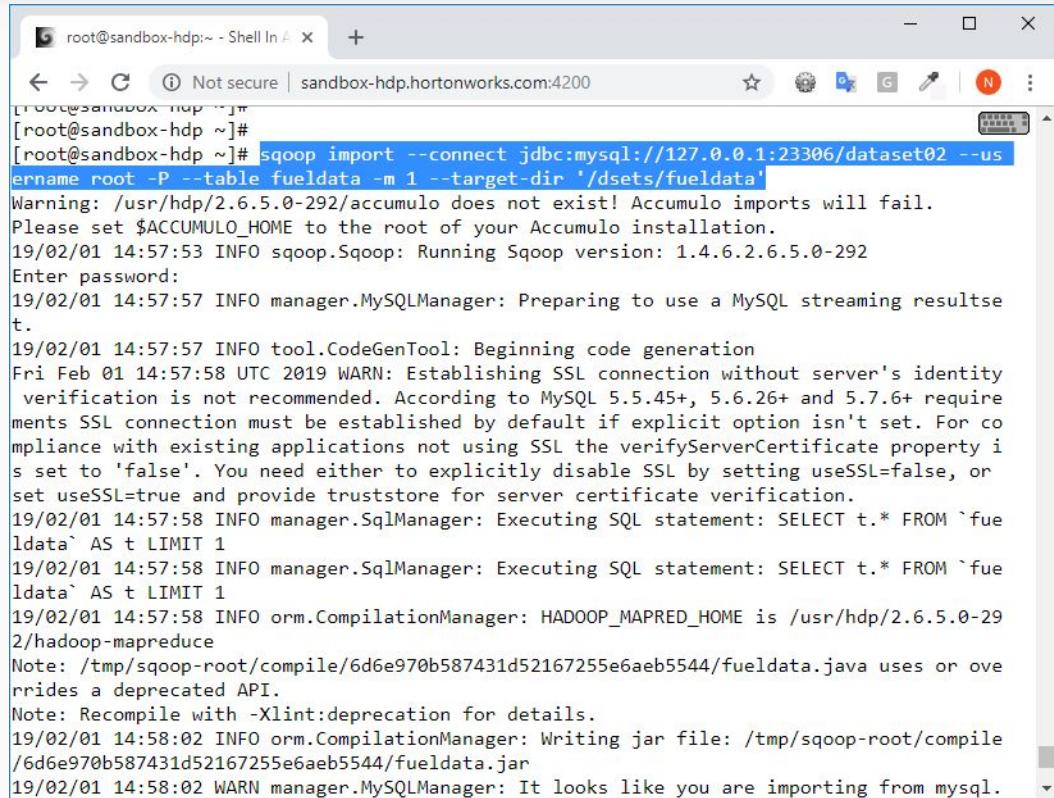
mysql> ■
```

Sqoop: simple export example (cont-d)

Step 2:

Run sqoop import command:

```
sqoop import --connect  
jdbc:mysql://127.0.0.1:23306/dataset02  
--username root -P --table fueldata -m 1  
--target-dir '/dsets/fueldata'
```



The screenshot shows a terminal window titled "root@sandbox-hdp:~ - Shell In" with the URL "sandbox-hdp.hortonworks.com:4200". The command entered is:

```
[root@sandbox-hdp ~]# sqoop import --connect jdbc:mysql://127.0.0.1:23306/dataset02 --username root -P --table fueldata -m 1 --target-dir '/dsets/fueldata'
```

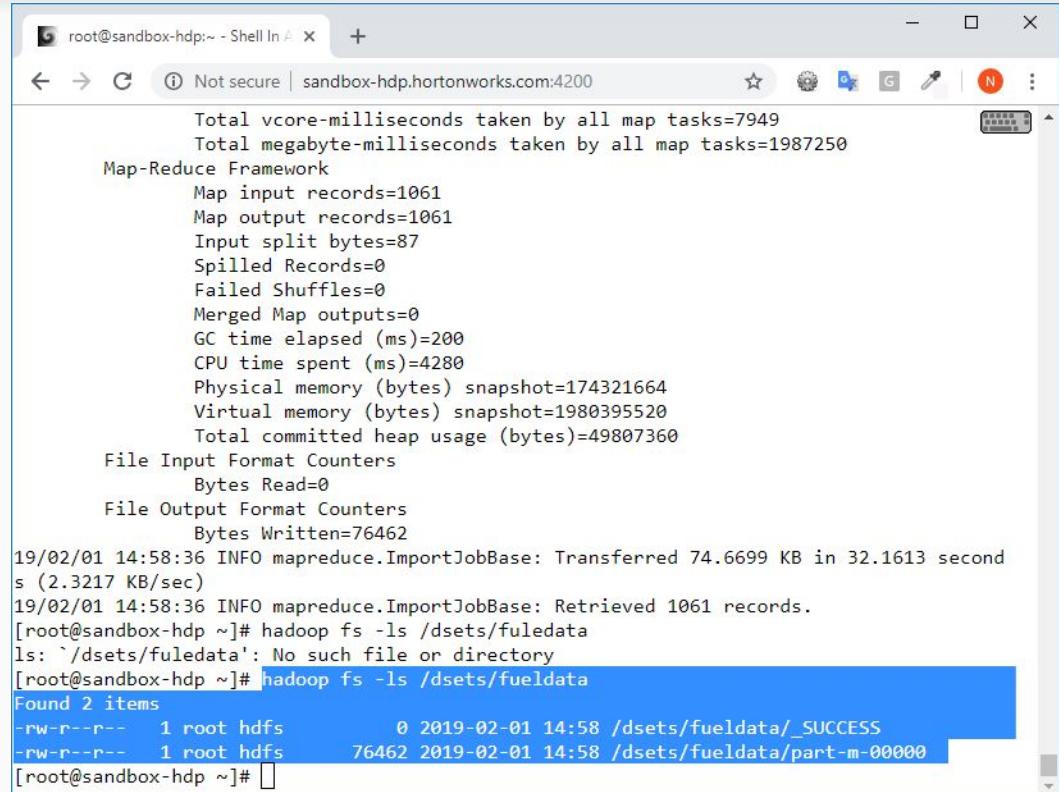
The output of the command is displayed below the command line:

```
Warning: /usr/hdp/2.6.5.0-292/accumulo does not exist! Accumulo imports will fail.  
Please set $ACCUMULO_HOME to the root of your Accumulo installation.  
19/02/01 14:57:53 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6.2.6.5.0-292  
Enter password:  
19/02/01 14:57:57 INFO manager.MySQLManager: Preparing to use a MySQL streaming resultset.  
19/02/01 14:57:57 INFO tool.CodeGenTool: Beginning code generation  
Fri Feb 01 14:57:58 UTC 2019 WARN: Establishing SSL connection without server's identity verification  
is not recommended. According to MySQL 5.5.45+, 5.6.26+ and 5.7.6+ require  
ments SSL connection must be established by default if explicit option isn't set. For co  
mpliance with existing applications not using SSL the verifyServerCertificate property i  
s set to 'false'. You need either to explicitly disable SSL by setting useSSL=false, or  
set useSSL=true and provide truststore for server certificate verification.  
19/02/01 14:57:58 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `fue  
ldata` AS t LIMIT 1  
19/02/01 14:57:58 INFO manager.SqlManager: Executing SQL statement: SELECT t.* FROM `fue  
ldata` AS t LIMIT 1  
19/02/01 14:57:58 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /usr/hdp/2.6.5.0-29  
2/hadoop-mapreduce  
Note: /tmp/sqoop-root/compile/6d6e970b587431d52167255e6aeb5544/fueldata.java uses or o  
verrides a deprecated API.  
Note: Recompile with -Xlint:deprecation for details.  
19/02/01 14:58:02 INFO orm.CompilationManager: Writing jar file: /tmp/sqoop-root/compile  
/6d6e970b587431d52167255e6aeb5544/fueldata.jar  
19/02/01 14:58:02 WARN manager.MySQLManager: It looks like you are importing from mysql.
```

Sqoop: simple export example (cont-d)

You'll see results of the command.

fueldata directory is created and has files there



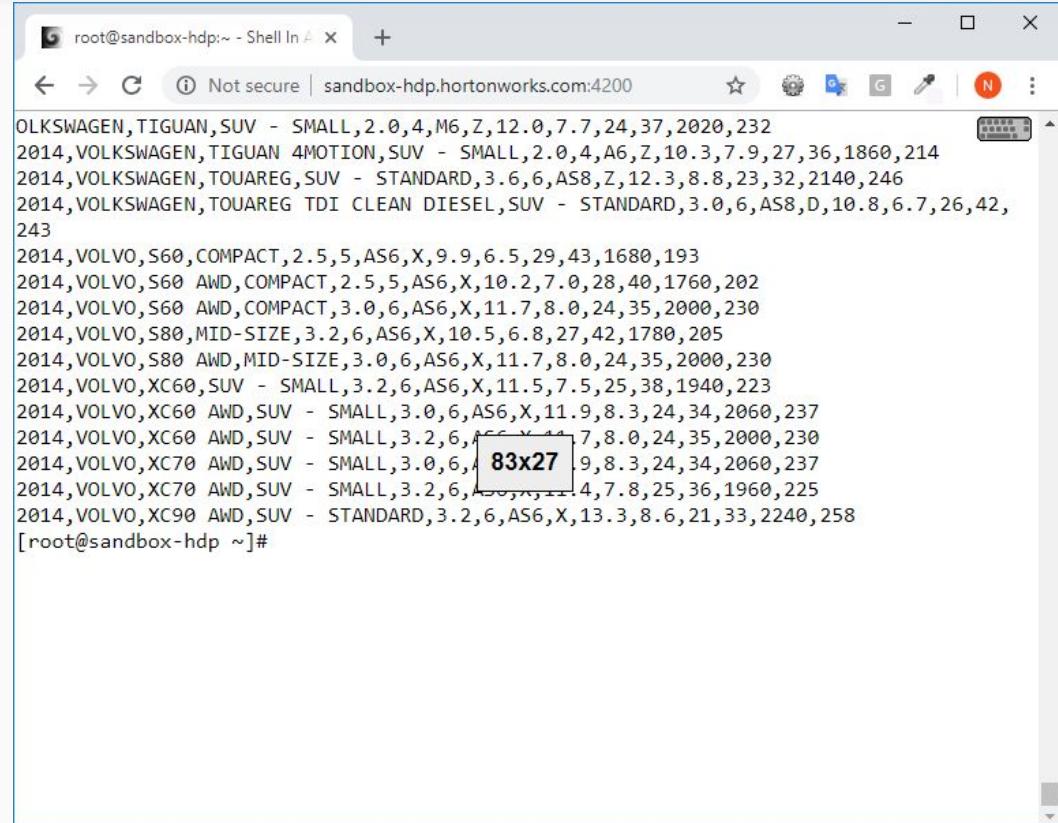
The screenshot shows a terminal window with the following output:

```
Total vcore-milliseconds taken by all map tasks=7949
Total megabyte-milliseconds taken by all map tasks=1987250
Map-Reduce Framework
  Map input records=1061
  Map output records=1061
  Input split bytes=87
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=200
  CPU time spent (ms)=4280
  Physical memory (bytes) snapshot=174321664
  Virtual memory (bytes) snapshot=1980395520
  Total committed heap usage (bytes)=49807360
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=76462
19/02/01 14:58:36 INFO mapreduce.ImportJobBase: Transferred 74.6699 KB in 32.1613 seconds (2.3217 KB/sec)
19/02/01 14:58:36 INFO mapreduce.ImportJobBase: Retrieved 1061 records.
[root@sandbox-hdp ~]# hadoop fs -ls /dsets/fueldata
ls: `/dsets/fueldata': No such file or directory
[root@sandbox-hdp ~]# hadoop fs -ls /dsets/fueldata
Found 2 items
-rw-r--r--  1 root hdfs          0 2019-02-01 14:58 /dsets/fueldata/_SUCCESS
-rw-r--r--  1 root hdfs    76462 2019-02-01 14:58 /dsets/fueldata/part-m-00000
[root@sandbox-hdp ~]#
```

Sqoop: simple export example (cont-d)

By default, the data is saved as
textfile in HDFS (rows are
separated by a newline, columns
are separated by comma or other
separator)

You may also import data
directly to Hive (later this day)



```
OLKSWAGEN, TIGUAN, SUV - SMALL, 2.0, 4, M6, Z, 12.0, 7.7, 24, 37, 2020, 232
2014, VOLKSWAGEN, TIGUAN 4MOTION, SUV - SMALL, 2.0, 4, A6, Z, 10.3, 7.9, 27, 36, 1860, 214
2014, VOLKSWAGEN, TOUAREG, SUV - STANDARD, 3.6, 6, AS8, Z, 12.3, 8.8, 23, 32, 2140, 246
2014, VOLKSWAGEN, TOUAREG TDI CLEAN DIESEL, SUV - STANDARD, 3.0, 6, AS8, D, 10.8, 6.7, 26, 42,
243
2014, VOLVO, S60, COMPACT, 2.5, 5, AS6, X, 9.9, 6.5, 29, 43, 1680, 193
2014, VOLVO, S60 AWD, COMPACT, 2.5, 5, AS6, X, 10.2, 7.0, 28, 40, 1760, 202
2014, VOLVO, S60 AWD, COMPACT, 3.0, 6, AS6, X, 11.7, 8.0, 24, 35, 2000, 230
2014, VOLVO, S80, MID-SIZE, 3.2, 6, AS6, X, 10.5, 6.8, 27, 42, 1780, 205
2014, VOLVO, S80 AWD, MID-SIZE, 3.0, 6, AS6, X, 11.7, 8.0, 24, 35, 2000, 230
2014, VOLVO, XC60, SUV - SMALL, 3.2, 6, AS6, X, 11.5, 7.5, 25, 38, 1940, 223
2014, VOLVO, XC60 AWD, SUV - SMALL, 3.0, 6, AS6, X, 11.9, 8.3, 24, 34, 2060, 237
2014, VOLVO, XC60 AWD, SUV - SMALL, 3.2, 6, AS6, X, 11.5, 7.8, 24, 35, 2000, 230
2014, VOLVO, XC70 AWD, SUV - SMALL, 3.0, 6, AS6, X, 11.9, 8.3, 24, 34, 2060, 237
2014, VOLVO, XC70 AWD, SUV - SMALL, 3.2, 6, AS6, X, 11.5, 7.8, 25, 36, 1960, 225
2014, VOLVO, XC90 AWD, SUV - STANDARD, 3.2, 6, AS6, X, 13.3, 8.6, 21, 33, 2240, 258
[redacted] 83x27 [redacted]
[redacted]
```

Sqoop: simple export example (cont-d)

Command line arguments explanation:

```
sqoop import --connect jdbc:mysql://127.0.0.1:23306/dataset02 --username root -P --table fueldata -m 1  
--target-dir '/dsets/fueldata'
```

- **import** - we **import** data (straightforward)
- After **--connect** we specify a **jdbc connection string** (where 127.0.0.1: **host**, 23306: **port**, dataset02: **database**)
- After **--username** we specify username for the mysql database connection
- **-P** means prompt for a password for the database connection
- After **--table** we specify a table name in our remote mysql database (you can also use **import-all-tables** command to import the whole database)
- After **--target-dir** we specify a directory in HDFS where to store the data
- **-m 1** means that we use **one mapper** rather than many. If you specify many mappers (>1), make sure that the table in database has a primary key (as it splits data by primary key)

Exercise

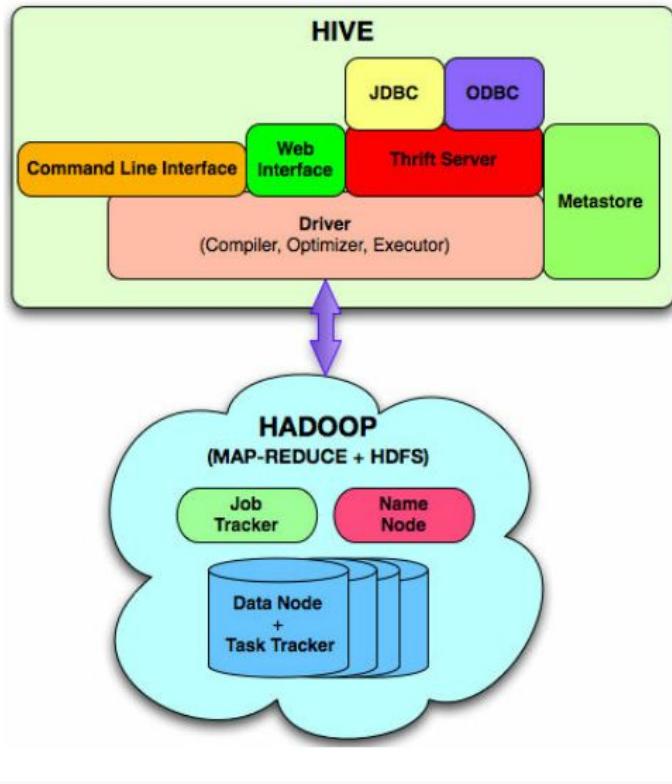
- Transfer the **fueldata** table from **dataset02** database from MySQL at dev.cs.smu.ca to HDFS folder **/dsets/fuel** using [sqoop](#)
- Make sure that it has been transferred correctly (you may use [*hadoop fs -tail*](#) command)

Hint: if you forgot how to create an SSH tunnel:

```
ssh -f -N -T -l 23306:127.0.0.1:3306 username@dev.cs.smu.ca
```

You all can use 23306 port (no birthdays are necessary - you are on your own VM)

Hive Architecture



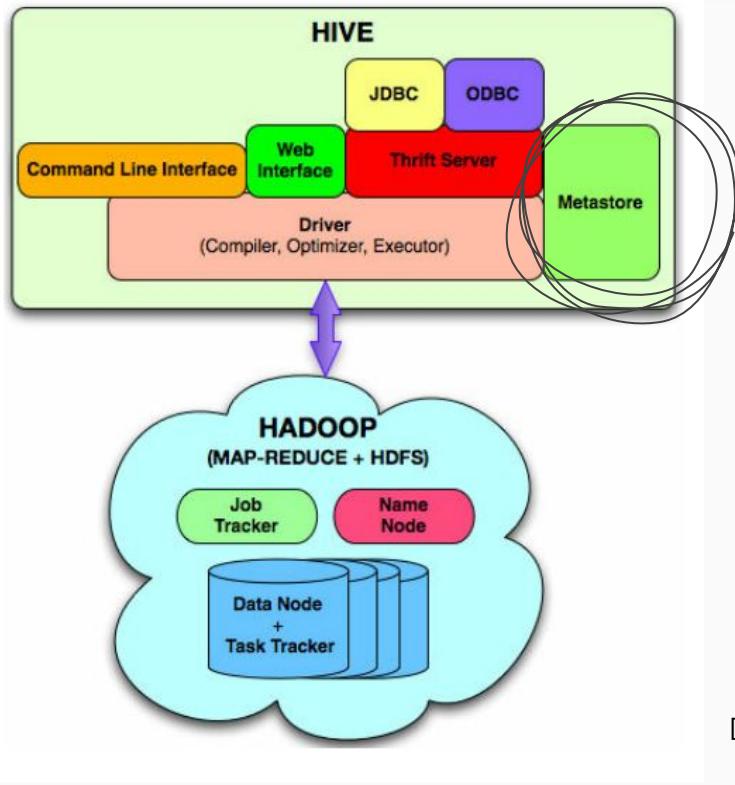
Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis. Introduced initially at Facebook around 2008 [wiki]

Important things:

- Hive is **not a database** but it provides a SQL-like interface to querying data (called **HiveQL**)
- In Hadoop, Hive works on top of HDFS (data is stored in HDFS) and uses MapReduce/Tez to execute queries

[link]

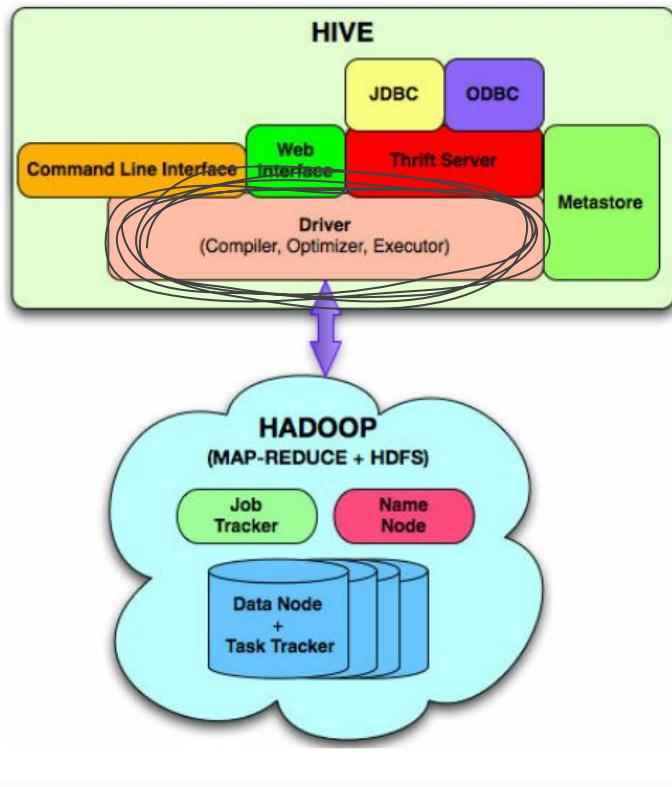
Hive Architecture: metastore



Metastore: Stores metadata for each of the tables such as their schema and location. It also includes the partition metadata which helps the driver to track the progress of various data sets distributed over the cluster. **The data is stored in a traditional RDBMS format (MySQL in our case) [wiki].**

[[link](#)]

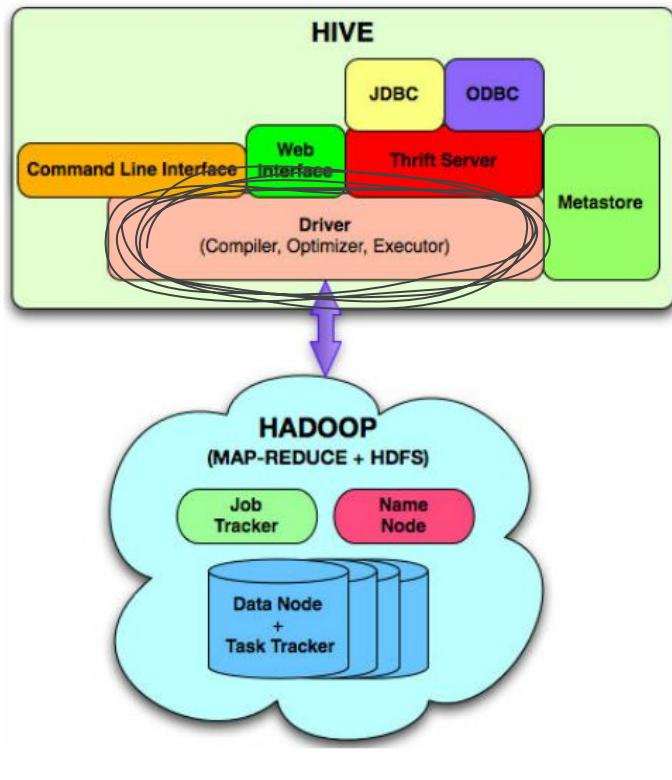
Hive Architecture: driver



Driver: Acts like a controller which receives the **HiveQL** statements. It starts the execution of the statement by creating sessions, and monitors the life cycle and progress of the execution. It stores the necessary metadata generated during the execution of a HiveQL statement. The driver also acts as a collection point of data or query results obtained after the Reduce operation [II]

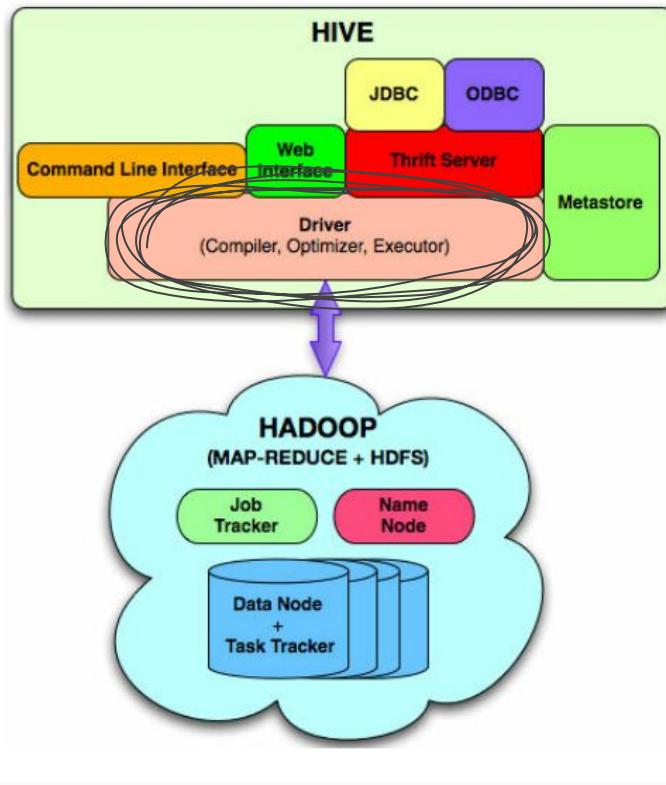
[link]

Hive Architecture: compiler



Compiler: Performs compilation of the HiveQL query, which converts the query to an execution plan. This plan contains the tasks and steps needed to be performed by the Hadoop MapReduce to get the output as translated by the query. The compiler converts the query to an abstract syntax tree (AST). After checking for compatibility and compile time errors, it converts the AST to a directed acyclic graph (DAG). The DAG divides operators to MapReduce stages and tasks based on the input query and data [link] [II]

Hive Architecture: optimizer and executor



Optimizer: Performs various transformations on the execution plan to get an optimized DAG. Transformations can be aggregated together, such as converting a pipeline of joins to a single join, for better performance. It can also split the tasks, such as applying a transformation on data before a reduce operation, to provide better performance and scalability. However, the logic of transformation used for optimization used can be modified or pipelined using another optimizer.

Executor: After compilation and optimization, the executor executes the tasks. It interacts with the job tracker of Hadoop to schedule tasks to be run.

[[link](#)]

Hive vs Relational Database

1. Schema on read:

Traditionally the table's schema is enforced at data load time (schema on write). Hive enforces it at query time (a load operation is simply a quick file move)

2. Updates:

Table updates are only possible by transforming all the data into a new table (i.e. no appends)

3. Transactions :

Hive does not support concurrent accesses to tables and hence application-level concurrency and locking mechanisms are needed.

4. Indexes:

Support provided but relatively immature

[link]

Hive: let's start

The screenshot shows the Apache Ambari web interface. At the top, there is a navigation bar with links: Dashboard, Services, Hosts, Alerts, Admin, and a user dropdown labeled "admin". Below the navigation bar is a main content area. On the left, there is a sidebar with several options: YARN Queue Manager, Files View, Hive View, Hive View 2.0 (which is highlighted with a blue oval), Big View, Tez View, and Workflow Manager. To the right of the sidebar, there is a "NOTIFICATIONS" section with two green buttons: "+ NEW JOB" and "+ NEW TABLE". Below the sidebar, there is a "Browse" dropdown menu. The main content area displays a table titled "default" with 4 tables listed: "fueldata", "intermediate_access_logs", "test", and "users".

| Tables(4) |
|--------------------------|
| fueldata |
| intermediate_access_logs |
| test |
| users |

Hive: let's start (cont-d)

The screenshot shows the Cloudera Manager interface with a blue header bar. The top navigation bar includes 'Dashboard', 'Services', 'Hosts', 'Alerts', 'Admin', and a user dropdown for 'admin'. Below the header, there are buttons for '+ NEW JOB' and '+ NEW TABLE', with '+ NEW TABLE' highlighted by a red box. The main area shows a search bar with 't' and a 'Browse' button. A 'SAVED QUERIES' section is visible. The central part of the screen is a 'TABLE > UPLOAD TABLE' dialog. It has two tabs: 'Select File Format' (selected) and 'Select File Source'. The 'Select File Format' tab contains fields for 'File type' (set to CSV), 'Field Delimiter' (set to ','), 'Escape Character' (set to '\'), 'Quote Character' (set to '\"'), 'Is first row header?' (unchecked), and 'Contains endlines?' (unchecked). The 'Select File Source' tab has radio buttons for 'Upload from HDFS' and 'Upload from Local', with 'Upload from Local' selected. Below this is a 'Select Local File' input field and a large red box highlighting a cloud icon with an upward arrow, which is typically used for uploading files.

Hive: let's start (cont-d)

TABLE > UPLOAD TABLE

Select File Format

File type: XML

Contains endlines?

Select File Type

CSV

JSON

XML

Select File Source

Upload from HDFS

Select Local File

Drag file to upload or click to browse

Exercise

- Create a table in Hive from the data downloaded into HDFS in the first Exercise (/dsets/census/adult.data)
- You can either use HDFS path or upload it from your local machine (<http://woolly.cs.smu.ca/tmp/adult.data>)
- Don't forget to use first row as headers
- Once uploaded, try the following queries:
 - **SELECT * FROM adult LIMIT 10**
 - **SELECT count(*) FROM adult**

What was done?

| INFORMATION | | VALUE |
|------------------|--|---|
| Database Name | | default |
| Owner | | admin |
| Create Time | | Fri Feb 01 17:17:39 UTC 2019 |
| Last Access Time | | UNKNOWN |
| Retention | | 0 |
| Table Type | | MANAGED_TABLE |
| Location | | hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/adult |
| Parameters | | { "transient_lastDdlTime": "1549041501" } |

What was done? (cont-d)

TABLE > ADULT

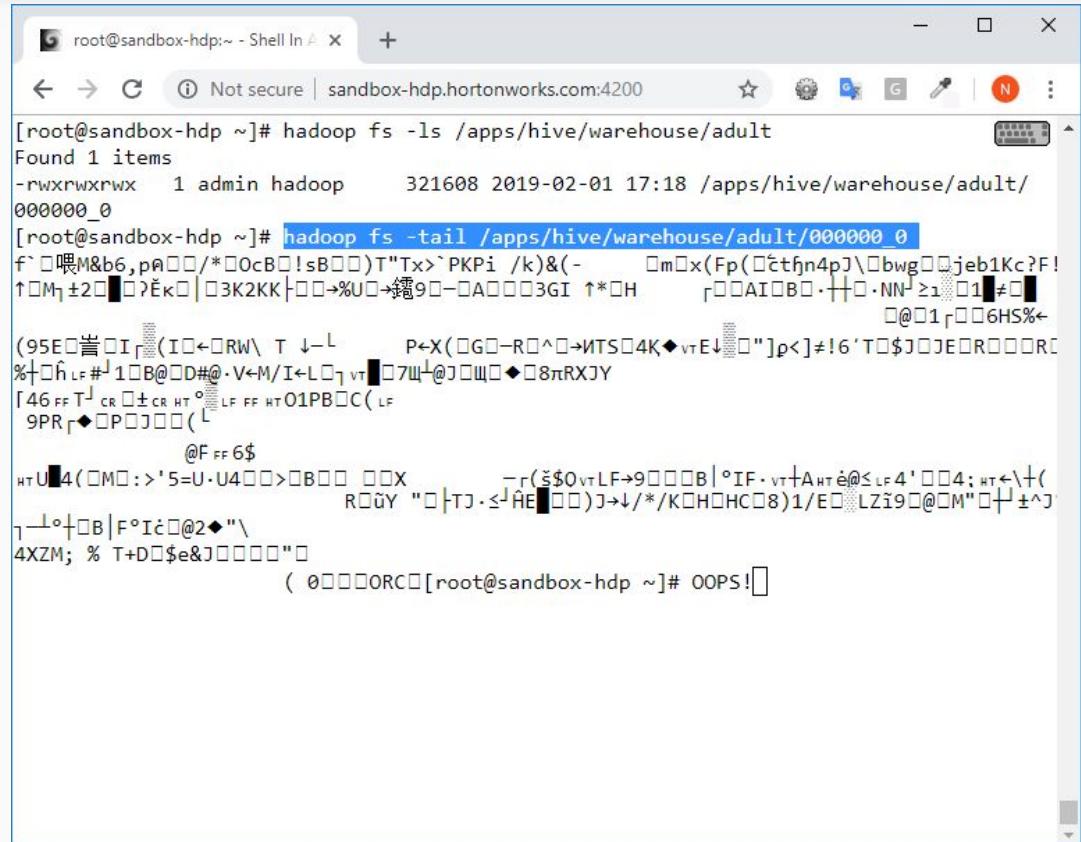
COLUMNS DDL STORAGE INFORMATION DETAILED INFORMATION STATISTICS AUTHORIZATION

```
11   sex string,  
12   `capital-gain` double,  
13   `capital-loss` double,  
14   `hours-per-week` double,  
15   `native-country` string,  
16   `income` string)  
17 ROW FORMAT SERDE  
18   'org.apache.hadoop.hive.ql.io.orc.OrcSerde'  
19 STORED AS INPUTFORMAT  
20   'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'  
21 OUTPUTFORMAT  
22   'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'  
23 LOCATION  
24   'hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/adult'  
25 TBLPROPERTIES (
```



What's in the file?

ORC file was created instead of just a simple text file. That's a default Hive behaviour when we create tables from GUI from .CSV files



The screenshot shows a terminal window titled "root@sandbox-hdp:~ - Shell In A..." with the URL "Not secure | sandbox-hdp.hortonworks.com:4200". The terminal displays the following command and its output:

```
[root@sandbox-hdp ~]# hadoop fs -ls /apps/hive/warehouse/adult
Found 1 items
-rwxrwxrwx 1 admin hadoop 321608 2019-02-01 17:18 /apps/hive/warehouse/adult/
000000_0
[root@sandbox-hdp ~]# hadoop fs -tail /apps/hive/warehouse/adult/000000_0
```

The output of the tail command is heavily redacted, showing only a few characters of the file content. The terminal ends with:

```
( 00000ORC[root@sandbox-hdp ~]# OOPS!
```

What is ORC?

Hive (and some Hadoop ecosystem components) supports different types of files:

- Text files (as with Sqoop)
- AVRO
- ORC
- Parquet
- ..etc

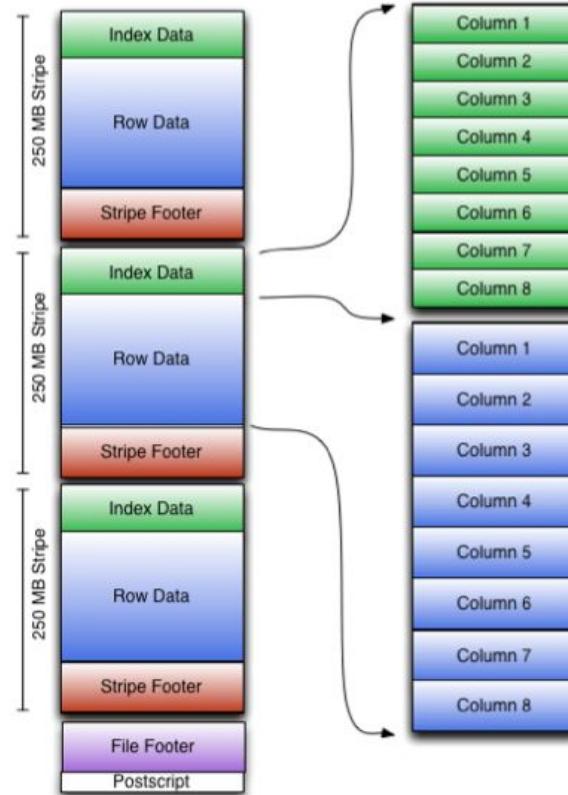
What is ORC? (cont-d)

The Optimized Row Columnar (ORC)

file format provides a highly efficient way to store Hive data

More about the format:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+ORC>



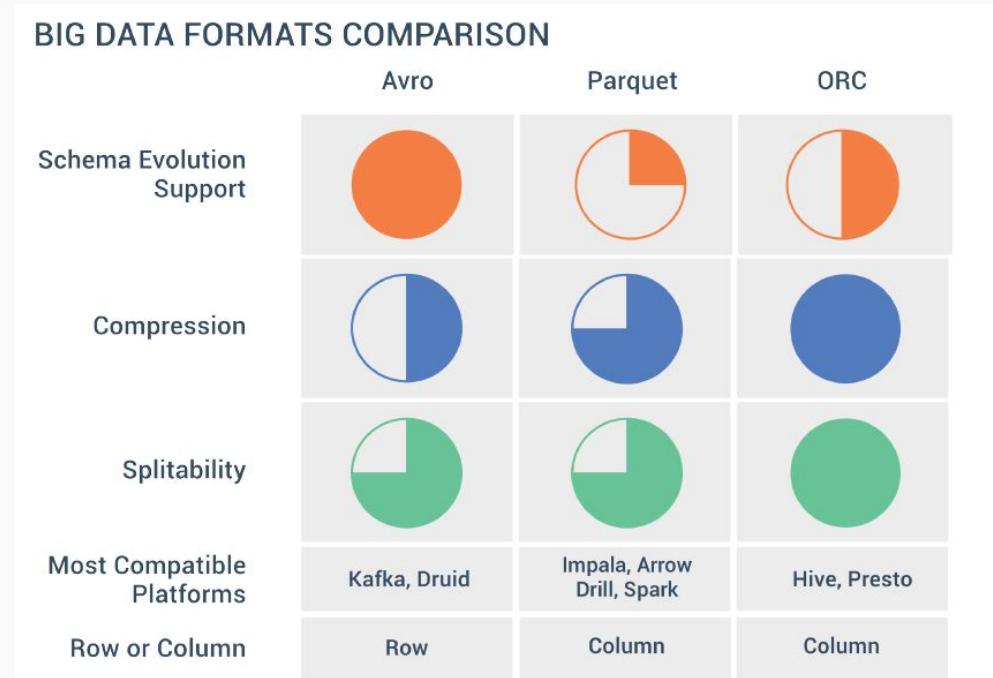
[link]

More on file formats

Can we store data in text format?

Yes!

These formats were created just to speedup applications and to use disk space more efficiently



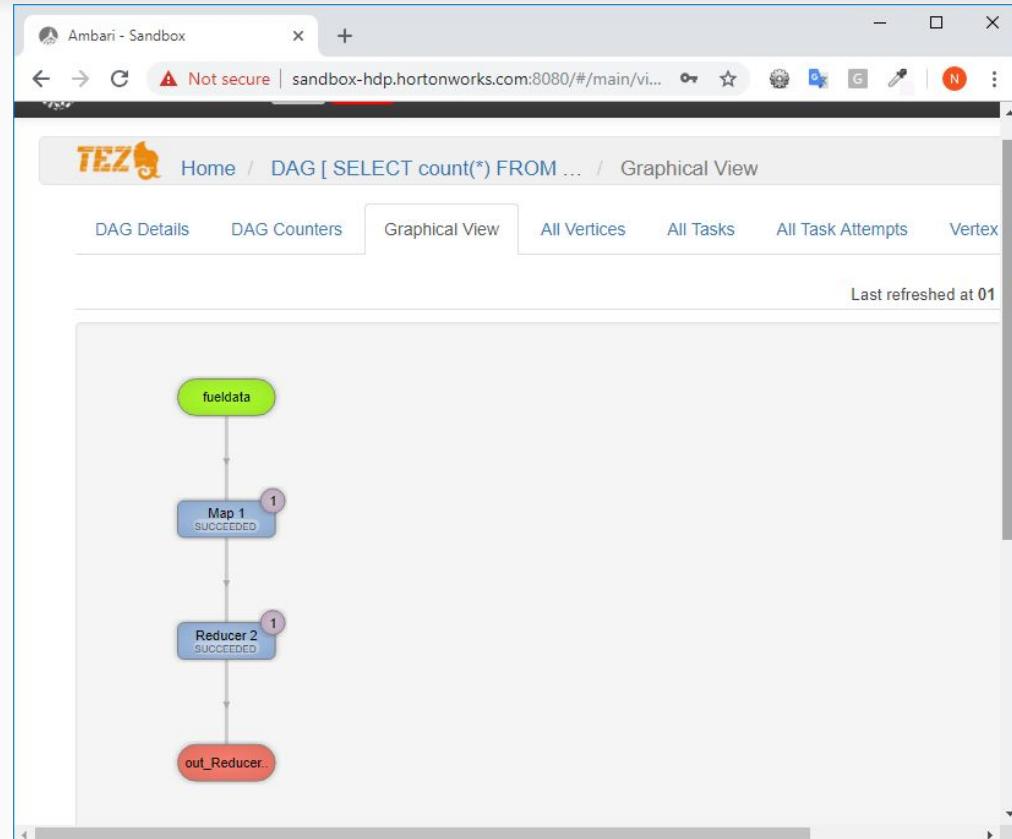
[link]

Hive + sqoop: Exercise

- Transfer the **fueldata** table from **dataset02** database from MySQL at dev.cs.smu.ca to **Hive** using [sqoop](#)
- Command: `sqoop import --connect jdbc:mysql://127.0.0.1:23306/dataset02 --username your_username -P --table fueldata -m 1 --hive-import --create-hive-table --hive-table fueldata --target-dir '/apps/hive/warehouse/fueldata'`
- Important: `--target-dir '/apps/hive/warehouse/fueldata'`: warehouse dir is the Hive's default storage

Hive + sqoop: Exercise

- What is a format of the imported dataset?
- Try: **SELECT count(*) FROM fueldata**
- Go to Tez View in Ambari and find a DAG for the last query



Hive: working with semi-structured data

TABLE > UPLOAD TABLE

Select File Format

File type: JSON

Contains endlines?

Select File Source

Upload from HDFS

Upload from Local

Select Local File

 Drag file to upload or click to browse

Preview

Hive: working with semi-structured data

- Table Preview

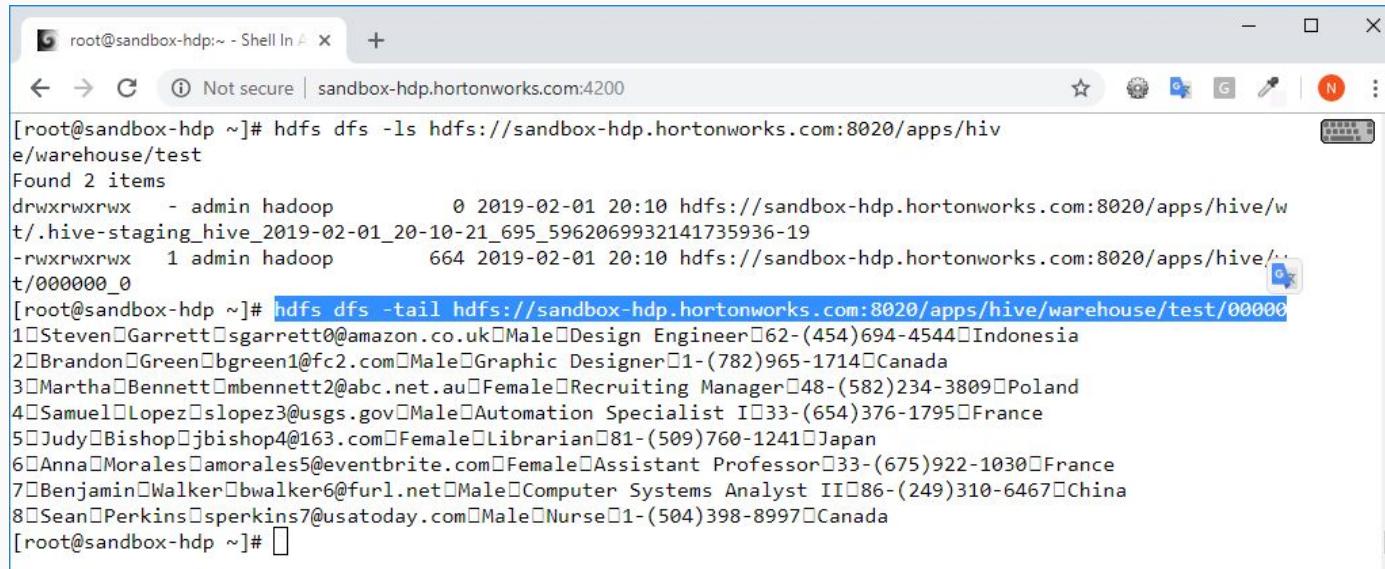
| id | first_name | last_name | email | gender | designation | phone | country |
|-----------|-------------------|------------------|--------------------------|---------------|-----------------------------|------------------|----------------|
| 1 | Steven | Garrett | sgarrett0@amazon.co.uk | Male | Design Engineer | 62-(454)694-4544 | Indonesia |
| 2 | Brandon | Green | bgreen1@fc2.com | Male | Graphic Designer | 1-(782)965-1714 | Canada |
| 3 | Martha | Bennett | mbennett2@abc.net.au | Female | Recruiting Manager | 48-(582)234-3809 | Poland |
| 4 | Samuel | Lopez | slopez3@usgs.gov | Male | Automation Specialist I | 33-(654)376-1795 | France |
| 5 | Judy | Bishop | jbishop4@163.com | Female | Librarian | 81-(509)760-1241 | Japan |
| 6 | Anna | Morales | amorales5@eventbrite.com | Female | Assistant Professor | 33-(675)922-1030 | France |
| 7 | Benjamin | Walker | bwalker6@furl.net | Male | Computer Systems Analyst II | 86-(249)310-6467 | China |
| 8 | Sean | Perkins | sperkins7@usatoday.com | Male | Nurse | 1-(504)398-8997 | Canada |

Hive: working with semi-structured data

The screenshot shows the Apache Hive Table Creation interface. At the top, there is a 'Name' input field containing 'test'. Below it are three tabs: 'COLUMNS', 'ADVANCED', and 'TABLE PROPERTIES'. The 'ADVANCED' tab is selected. Under the 'ADVANCED' tab, there is a 'Settings' section with a gear icon. A dropdown menu is open under 'Enter data type', listing several options: SEQUENCEFILE, TEXTFILE, RCFILE, ORC, AVRO, CUSTOM SerDe, and another ORC option at the bottom. At the bottom of the screen, there are two buttons: a green 'Create' button with a plus sign and an orange 'Cancel' button with a cross.

Exercise

- Download a json file from <http://wolly.cs.smu.ca/tmp/test.json>
- Create a table in Hive based on this file. If you choose Text format you'll see that the new file in HDFS contains this data:



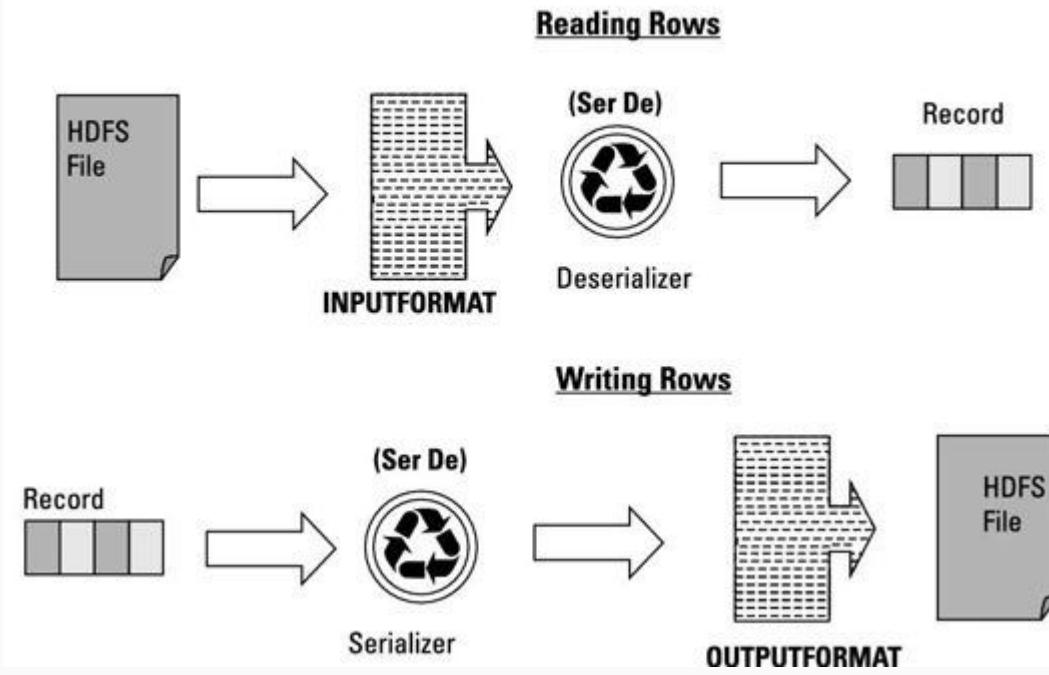
The screenshot shows a terminal window titled "root@sandbox-hdp:~ - Shell In". The URL bar indicates the connection is not secure and points to "sandbox-hdp.hortonworks.com:4200". The terminal output shows the following commands and their results:

```
[root@sandbox-hdp ~]# hdfs dfs -ls hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/test
Found 2 items
drwxrwxrwx  - admin hadoop          0 2019-02-01 20:10 hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/w
t/.hive-staging_hive_2019-02-01_20-10-21_695_5962069932141735936-19
-rwxrwxrwx  1 admin hadoop      664 2019-02-01 20:10 hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/..t/000000_0

[root@sandbox-hdp ~]# hdfs dfs -tail hdfs://sandbox-hdp.hortonworks.com:8020/apps/hive/warehouse/test/00000
1 Steven Garrett sgarrett@amazon.co.uk Male Design Engineer 62-(454)694-4544 Indonesia
2 Brandon Green bgreen1@fc2.com Male Graphic Designer 1-(782)965-1714 Canada
3 Martha Bennett mbennett2@abc.net.au Female Recruiting Manager 48-(582)234-3809 Poland
4 Samuel Lopez slopez3@usgs.gov Male Automation Specialist I 33-(654)376-1795 France
5 Judy Bishop jbishop4@163.com Female Librarian 81-(509)760-1241 Japan
6 Anna Morales amorales5@eventbrite.com Female Assistant Professor 33-(675)922-1030 France
7 Benjamin Walker bwalker6@furl.net Male Computer Systems Analyst II 86-(249)310-6467 China
8 Sean Perkins sperkins7@usatoday.com Male Nurse 1-(504)398-8997 Canada
[root@sandbox-hdp ~]#
```

Hive: SerDe

- SerDe = Serializer + Deserializer
- It is used by Hive to “convert” text (or binary) files to records that Hive works with and vice versa
- You can [write your own SerDe](#) in Java or you can use one of many predefined ones:
 - Avro
 - ORC
 - RegEx
 - Thrift
 - Parquet
 - CSV
 - JsonSerDe



Hive: SerDe: RegEx SerDe example

Worksheet1 * +

DATABASE
Select or search database/schema

```
1 CREATE EXTERNAL TABLE serde_test (
2     age STRING,
3     workclass STRING
4 )
5 ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
6 WITH SERDEPROPERTIES (
7     'input.regex' = '(.+?),(.+?),.*',
8     'output.format.string' = "%1$s %2$s")
9 LOCATION '/dsets/census/';
```

☰

Execute Save As Insert UDF Visual Explain

Hive: SerDe: RegEx SerDe example

```
1 select * from serde_test limit 10
```

Execute Save As Insert UDF Visual Explain

RESULTS LOG VISUAL EXPLAIN TEZ UI

Filter columns

| serde_test.age | serde_test.workclass |
|----------------|----------------------|
| age | workclass |
| 39 | State-gov |
| 50 | Self-emp-not-inc |
| 38 | Private |

Hive: SerDe: RegEx SerDe example explained

```
CREATE EXTERNAL TABLE serde_test (
    age STRING,
    workclass STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    'input.regex' = '(.+?),(.+?),.*',
    'output.format.string' = "%1$s %2$s") -- remember capturing groups  
in regex? Here they are!
LOCATION '/dsets/census/';
```

Exercise: real use case

Purpose: identify Linux usernames that are most likely to be hacked

- Download an **archive** with auth logs from
<http://wolly.cs.smu.ca/tmp/auth.tar.gz>
- Uncompress and put the files into hdfs folder “/dsets/logs”
- Create a Hive table based on the logs (using RegEx SerDe):
 - Name of invalid user
 - Ip address
- Using Hive identify top 10 usernames that hackers try to hack

Hint: example of line with invalid username (format):

```
Failed password for invalid user wi from 184.82.9.233 port 45796 ssh2
```

Zeppelin

The screenshot shows a web browser window with the Zeppelin interface. The title bar includes tabs for "Hortonworks Sandbox with HDP" and "sandbox-hdp.hortonworks.com:9995/#". The main content area features a large blue header with the Zeppelin logo, a search bar, and a user status indicator ("anonymous"). Below the header, a prominent "Welcome to Zeppelin!" message is displayed. To the left, a sidebar contains links for "Notebook" and "Job", along with buttons for "Import note" and "Create new note". A filter input field is also present. To the right, there are sections for "Help" (linking to documentation) and "Community" (inviting contributions and linking to mailing lists, issues tracking, and GitHub). A large, stylized illustration of a hot air balloon is positioned on the right side of the main content area.

Hortonworks Sandbox with HDP × sandbox-hdp.hortonworks.com:9995/#

Not secure | sandbox-hdp.hortonworks.com:9995/#/

Zeppelin Notebook Job Search your Notes anonymous

Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Notebook

- Import note
- Create new note

Filter

- Getting Started
- Labs
- R (SparkR)
- Untitled Note 1
- Zeppelin Tutorial (Basic Features)

Help

Get started with [Zeppelin documentation](#)

Community

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

- Mailing list
- Issues tracking
- Github

Zeppelin

Hortonworks Sandbox with HDP x sandbox-hdp.hortonworks.com:995 x +

Not secure | sandbox-hdp.hortonworks.com:995/#/

Zeppelin Notebook Job Search your Notes anonymous

Welcome to Zeppelin

Zeppelin is web-based notebook that enables i You can make beautiful data-driven, interactive

Notebook Import note Create new note Filter Getting Started Labs R (SparkR) Untitled Note 1 Zeppelin Tutorial (Basic Features) Trash Mailing list Issues tracking Github

Create new note

Note Name

Default Interpreter jdbc

Use '/' to create folders. Example: /NoteDirA/Note1

Create Note

Zeppelin

The screenshot shows a Zeppelin Notebook interface running on an Hortonworks Sandbox with HDP. The browser tab is titled "Hortonworks Sandbox with HDP" and the URL is "sandbox-hdp.hortonworks.com:9995/#/notebook/2E3CPFFXS". The notebook title is "Zepp+Hive". The code cell contains the following:

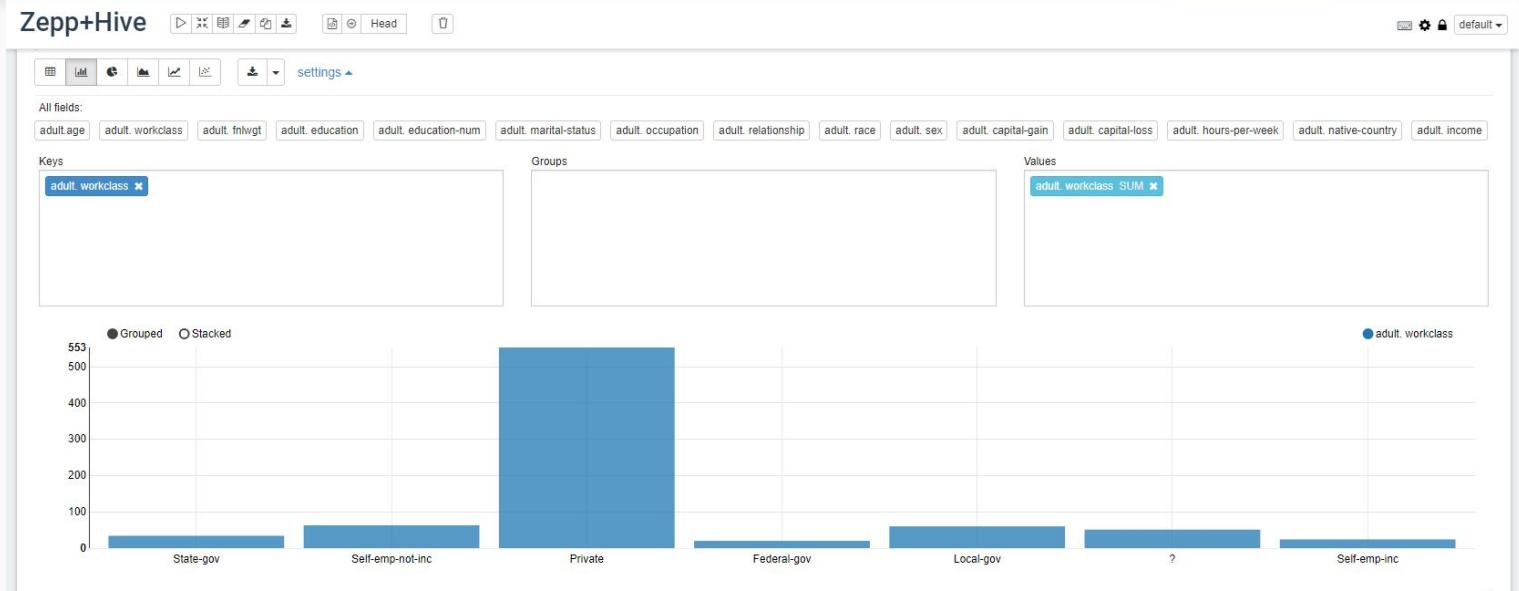
```
%jdbc(hive)
select * from adult
```

The output displays the first six rows of the "adult" table from Hive:

| | adult.age | adult.workclass | adult.fnlwgt | adult.education | adult.education-num | adult.marital-st |
|----|------------------|-----------------|--------------|-----------------|---------------------|------------------|
| 39 | State-gov | 77516.0 | Bachelors | 13.0 | Never-married | |
| 50 | Self-emp-not-inc | 83311.0 | Bachelors | 13.0 | Married-civ-spou | |
| 38 | Private | 215646.0 | HS-grad | 9.0 | Divorced | |
| 53 | Private | 234721.0 | 11th | 7.0 | Married-civ-spou | |
| 28 | Private | 338409.0 | Bachelors | 13.0 | Married-civ-spou | |

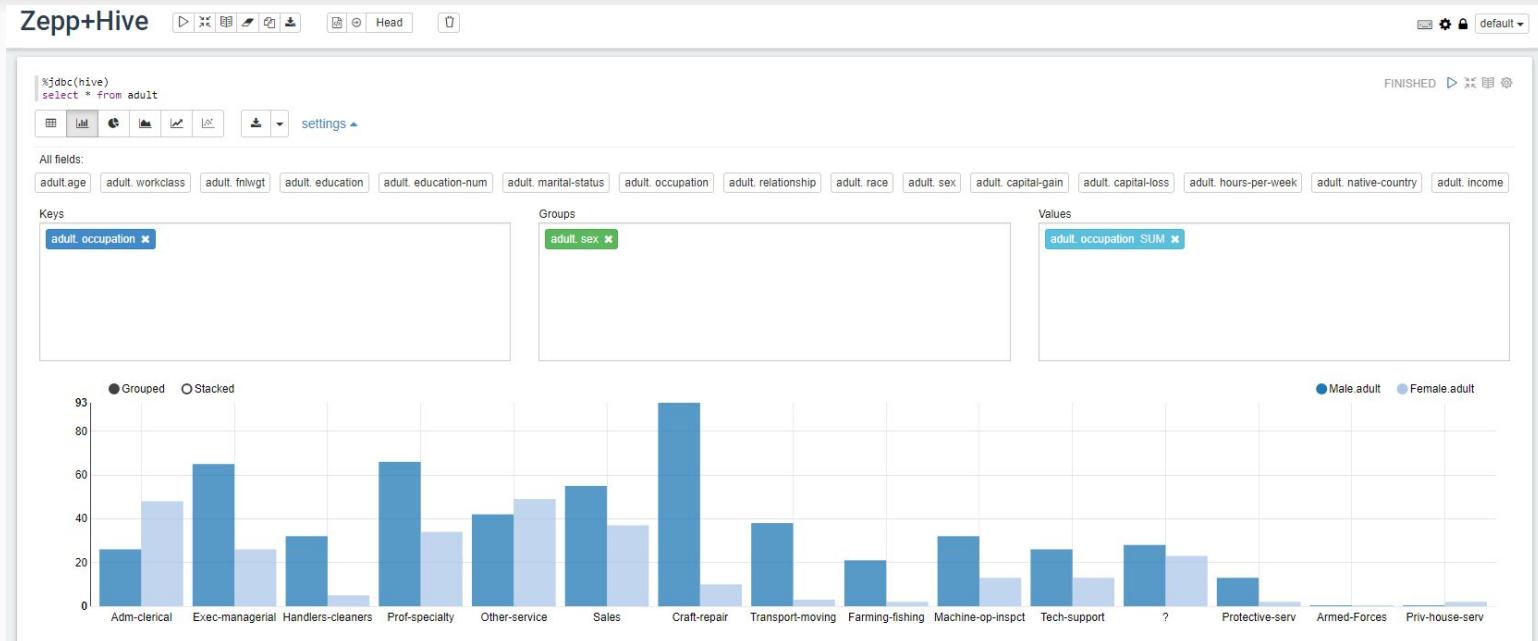
Don't forget about "magic" statement **%jdbc(hive)** before each SQL-like statement

Zeppelin



You generally don't need to write complex SQL statements to get graphs.
Just use UI

Zeppelin



You generally don't need to write complex SQL statements to get some basic insights from data. Just use UI

Exercise

Open Zeppelin, use any available table in Hive from today's exercises (e.g., "adult"). Build multiple graphs that can give some insights from the data:

- Use at least 2 types of graphs
- Try to run different queries

EOW

EOW

Thanks for your attention!