SPA Conference 2016

Real-World Big Data in Action

Nick Rozanski
Eoin Woods
Chris Cooper-Bland

Number 2 in an occasional series

Prerequisites

Install Java (the full JDK, not the JRE) if not installed already

- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- For Mac El Capitain, these commands will work if you have Homebrew:

```
$ brew tap caskroom/cask
$ brew unlink brew-cask
$ brew cask install java
```

- For Debian and Ubuntu, these commands should work:

```
$ sudo apt-get update
$ sudo apt-get install default-jdk
```

- I'm not entirely sure about this, I already had the JRE…

Install Python 2.7 if not installed already

- https://www.python.org/downloads/
- you don't need to know Python programming for this session, but need it to run some of the tools
- you may need to add C:\Python27 to %PATH% (see later slide for instructions)

Install git if not installed already

- https://git-scm.com/downloads (or `apt`, `yum` etc for Linux)
- Windows and Mac versions are on the USB stick in the `downloads` folder

Windows Setup

Install the Visual C++ Runtime

- Run `vcredist_x64.exe` (it's on the on the USB stick in the `downloads\windows` directory)
- *I downloaded this from https://www.microsoft.com/en-us/download/details.aspx?id=13523*

Create a `spa16` User in Windows

- You need to add a user with a name without spaces
- You can't do this from the Windows UI (since it demands a first and last name)
- Run a Windows Command Prompt as Administrator and enter the command:

```
C:\Windows\System32>net user spa16 /add
```

- Use Settings -> Accounts to make this account an Administrator (Family & other users -> Change account type)
- Log out of Windows, and log back in again as the `spa16` user

Clone the Project Files

Start a Shell / Command Prompt

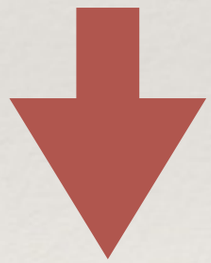
- OS X: Applications → Terminal; Linux: Start Terminal from the toolbar; Windows: run a Command Prompt

Create the Project Subdirectory

```
$ mkdir -p $HOME/SPA_2016/   ← for OS X and Linux
C:\> mkdir \SPA_2016         ← for Windows
```

Clone the Project files

```
$ cd $HOME/SPA_2016   ← for OS X and Linux
C:\> cd \SPA_2016     ← for Windows
$ git clone https://github.com/rozanski/bcs_spa16.git .
```



Copy the Additional Directories from the USB Stick

- Copy the `datasets` and `download` directories from the USB stick into the `SPA_2016` directory

Install Hadoop

For OS X and Linux

- The Unix Hadoop software is in the `downloads/osx-linux` directory on the USB stick
- Extract `hadoop*.tar` into `~/SPA_2016/hadoop`

For Windows

- The standard Hadoop binaries do not work in Windows
- A custom Hadoop build for Windows is in the `downloads/windows` directory on the USB stick
- It includes Windows DLLs `hdfs.dll` and `hadoop.dll`, and batch scripts for starting the various programs (which generally don't work…)
- Extract `hadoop*.tar` into `C:\SPA_2016\hadoop`

Check Extract

- Check you have directories `hadoop/bin`, `hadoop/sbin`, `hadoop/logs`, `hadoop/sbin`…

Install Spark and Hive

Install Spark

- The Spark software (all platforms) is in the `downloads/` directory on the USB stick
- Extract `spark*.tar` into `$SPA_2016/spark` (OS X, Linux) or `%SPA_2016%\spark` (Windows)
- You should end up with directories `spark/bin`, `spark/conf`, `spark/logs`, `spark/sbin`…

Install Spark CSV Support

- This is also in the `downloads/` directory on the USB stick
- Copy `spark-csv*.jar` into `$SPA_2016/spark/lib`

Install Hive

- The Hive software (all platforms) is in the `downloads/` directory on the USB stick
- Extract `apache-hive*.tar` into `$SPA_2016/hive` (OS X, Linux) or `%SPA_2016%\hive` (Windows)
- You should end up with directories `hive/bin`, `conf`…

Set Environment Variables (OS X, Linux)

Check the script `$HOME/SPA_2016/env.src`

- This attempts to derive `$JAVA_HOME` for your environment
  - It is configured for the latest Java version (1.8.0_91)
- It sets `$SPA_HOME` to the root directory for your project files
- It sets various environment variables for Hadoop, Spark and Hive

Run the Script

```
$ source $HOME/SPA_2016/env.src
```

- If there are no errors, and `$JAVA_HOME` and `$SPA_HOME` have been set correctly, you (probably) don't need to change it
- You will run the script at the start of each exercise

Set Environment Variables (Windows)

Set System Environment Variables

- You need Administrator access to do this
- Use Settings -> System Properties -> Advanced -> Environment Variables…
- You may need to restart

%JAVA_HOME%

- Set it to something like `C:\PROGRA~1\Java\JDK18~1.0_9`
- to get the 8.3 path type `for %I in (.) do echo %~sI`

%PATH%

- Add the following paths to %PATH%:

```
C:\WINDOWS\SYSTEM32
C:\SPA_2016\hadoop\bin
C:\SPA_2016\hadoop\sbin
```

Run the Environment script

```
C:\> env.cmd
```

- You will run the script at the start of each exercise

Configure the Big Data Software

Sample Configuration Files

- Sample configuration files for the session are in directories under `$SPA_2016/config` (OS X, Linux) or `%SPA_2016%\config` (Windows)
- You will be copying the files into the software directories, and then editing them for your setup
- You will need to change `YOURNAME` to your operating system hostname (eg `nick`)
- You will need to change `/YOURHOME` to your home directory (eg `/Users/nick`)

Copy Hadoop Config Files

- from: `config/hadoop_etc_hadoop`
- to: `hadoop/etc/hadoop`

Copy Spark Config Files

- from: `config/spark_conf`
- to: `spark/conf`

Copy Hive Config Files

- from: `config/hive_conf`
- to: `hive/conf`

Configure Hadoop (Linux, OS X)

Hadoop Configuration Files

- Hadoop configuration files are XML files
- Edit them as shown below
- Change `/YOURNAME` to your operating system hostname (eg `spa16`)
- Change `/YOURHOME` to your home directory

$SPA_2016/hadoop/etc/hadoop/core-site.xml

`fs.defaultFS`	`hdfs://localhost:9000`	NameNode URI
`hadoop.tmp.dir`	`/YOURHOME/SPA_2016/data/hadoop/tmp`	Hadoop temporary directory
`hadoop.proxyuser.YOURNAME.hosts`	`*` (*asterisk*)	used to configure connect connections; you adopt your operating system username and group when logged into Hadoop or Hive
`hadoop.proxyuser.YOURNAME.groups`	`*` (*asterisk*)	

$SPA_2016/hadoop/etc/hadoop/hdfs-site.xml

`dfs.datanode.data.dir`	`file:/YOURHOME/SPA_2016/data/hadoop/hdfs/datanode`	Path on the local filesystem where the DataNode stores its blocks
`dfs.namenode.name.dir`	`file:/YOURHOME/SPA_2016/data/hadoop/hdfs/namenode`	Path on the local filesystem where the NameNode stores the namespace and transaction logs

Configure Hadoop (Windows)

Hadoop Configuration Files

- Hadoop configuration files are XML files
- Edit them as shown below (use forward-slashes even though this is Windows)
- Don't forget to change `YOURNAME` to your operating system hostname

C:\SPA_2016\hadoop\etc\hadoop\core-site.xml

`fs.defaultFS`	`hdfs://localhost:9000`	NameNode URI
`hadoop.tmp.dir`	`/SPA_2016/data/hadoop/tmp`	Hadoop temporary directory
`hadoop.proxyuser.YOURNAME.hosts`	`*` (*asterisk*)	used to configure connect connections; you adopt your operating system username and group when logged into Hadoop or Hive
`hadoop.proxyuser.YOURNAME.groups`	`*` (*asterisk*)	

C:\SPA_2016\hadoop\etc\hadoop\hdfs-site.xml

`dfs.datanode.data.dir`	`file:/SPA_2016/data/hadoop/hdfs/datanode`	Path on the local filesystem where the DataNode stores its blocks
`dfs.namenode.name.dir`	`file:/SPA_2016/data/hadoop/hdfs/namenode`	Path on the local filesystem where the NameNode stores the namespace and transaction logs

Configure Spark

Spark Configuration Files

- Edit Spark configuration script and files as shown below
- for Windows, replace /YOURHOME/SPA_2016 with /SPA_2016

$SPA_2016/spark/conf/spark-env.sh

export HADOOP_CONF_DIR=/YOURHOME/SPA_2016/hadoop/etc/hadoop	Tells Spark where the Hadoop configuration files can be found
export SPARK_LOCAL_DIRS=/YOURHOME/SPA_2016/data/spark	Tells Spark where to put its local storage

$SPA_2016/spark/conf/slaves

localhost (or IP address)	Tells Spark that a slave (Worker) will be running on the local computer

$SPA_2016/spark/conf/spark-defaults.conf

spark.jars.packages com.databricks:spark-csv_2.11:1.4.0	All on one line. Tells Spark that a slave (Worker) will be running on the local computer. Make sure the version (11.1.4.0) matches the version of the JAR you downloaded previously

Configure Hive

Hive Configuration and Metastore Database

- Edit Hive configuration files as shown below
- for Windows, replace `/YOURHOME/SPA_2016` with `/SPA_2016`

$SPA_2016/hive/conf/hive-site.xml

`fs.defjavax.jdo.option. ConnectionURLaultFS`	`jdbc:derby:;databaseName=/YOURHOME/SPA_2016/ data/hive/metastore_db; create=true`	JDBC connect string for the metastore
`hive.execution.engine`	`spark`	Hive execution engine. Options are: mr (map reduce, default, deprecated), tez, spark.
`datanucleus.autoCreateTables`	`TRUE`	automatically create metadata tables on first invocation of hive server

$SPA_2016/hive/conf/spark-defaults.conf

`spark.master` spark://master:7077	Tells Hive the URL of the Spark Master server
`spark.jars.packages com.databricks:spark-csv_2.11:1.4.0`	Tells Hive to load Spark CSV support

Create Hive Metastore Database

- The Hive megastore database uses Derby to store Hive metadata
- In production, you would use a robust multi-threaded database like MySQL or SQL Server

ALREADY DONE FOR YOU

OS X and Linux

- Run these commands

```
$ source $HOME/SPA_2016/env.src
$ spa_2016.bash init_metastore
```

- You should have a directory `$SPA_2016/data/hive/metastore_db/`

Windows

- Extract the tar file `metastore_db.tgz` from the `downloads` directory into the `\SPA_2016\data\hive` directory
- If you don't have a suitable extraction program (eg Winzip) you can use 7zip (there is an installer in the `downloads` directory)
- You should have a directory `\SPA_2016\data\hive\metastore_db/`

Set up Passphraseless SSH (OS X, Linux)

Spark and SSH

- SSH is Secure Shell, a cryptographically secure way of running services over an insecure network (for example, logging in to another computer)
- Spark uses SSH to communicate between nodes (in an enterprise installation, these will run on many different computers)
- For the exercise we are going to set up SSH without a password

Check for Passphraseless SSH

- Type the command:

```
$ ssh localhost
```

- If you are prompted for a passphrase, you will need to set up passphraseless SSH

Set up Passphraseless SSH

- Type these commands:

```
$ ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
$ cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

- If you don't do this, you will be prompted for your password whenever you start up / shut down Hadoop or Spark

Real-World Big Data in Action

Hadoop Exercises

A Big Data Virtual Filesystem

Initialise Hadoop Filesystem (OS X, Linux)

Format the HDFS Filesystem

- Equivalent to formatting an operating system filesystem partition
- Warning: this destroys all HDFS data!

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/bin/hdfs namenode -format
```

- Confirm there are no `WARN` or `ERROR` messages
- Do this before starting Hadoop
- If it goes wrong, delete the contents of the data directory before retrying

Check it has worked

```
$ ls $SPA_2016/data/hadoop/hdfs/
namenode
```

- The directory `$SPA_2016/data/hadoop` stores the Hadoop physical operating system files for the namenode and datanode

Initialise Hadoop Filesystem (Windows)

Format the HDFS Filesystem

- Equivalent to formatting an operating system filesystem partition
- Warning: this destroys all HDFS data!

```
C:\> env.cmd
C:\> C:\SPA_2016\hadoop\bin\hadoop.cmd namenode -format
```

- Confirm there are no `WARN` or `ERROR` messages (ignore the DEPRECATED message)
- Do this before starting Hadoop

Check it has worked

```
C:\> dir \SPA_2016\data\hadoop\hdfs
namenode
```

- The directory `C:\hadoop_data\hadoop` stores the Hadoop physical operating system files for the namenode and datanode

Start Hadoop Server (OS X, Linux)

Start Hadoop

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start namenode
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start secondarynamenode
$ $HADOOP_PREFIX/sbin/hadoop-daemon.sh start datanode
$ $HADOOP_PREFIX/sbin/yarn-daemon.sh start resourcemanager
$ $HADOOP_PREFIX/sbin/yarn-daemon.sh start nodemanager
```

- can also just run

```
$ $HADOOP_PREFIX/sbin/start-dfs.sh
$ $HADOOP_PREFIX/sbin/start-yarn.sh
```

To Stop Hadoop at any time

```
$ source $HOME/SPA_2016/env.src
$ $HADOOP_PREFIX/sbin/stop-dfs.sh
```

- Ignore messages like “Unable to load native-hadoop library for your platform”

Start Hadoop Server (Windows)

Start Hadoop

- Start a Command Prompt, then type

```
C:\> start-dfs.cmd
C:\> start-yarn.cmd
```

- Four command windows should open:
  - Hadoop namenode and datanode
  - YARN nodemanager and resource manager
- Check that the servers are all running and none terminated with errors
- If nothing starts, check %PATH% (should include `hadoop\bin` and `hadoop\sbin`)
- A nearly full disk may cause startup to fail

To Stop Hadoop at any time

```
C:\> stop-dfs.cmd
C:\> stop-yarn.cmd
```

- or just click on each window and type Ctrl-C

Check Hadoop is Running

Check Running Processes

```
$ jps | sort -k 2                ← for OS X and Linux
C:\> %JAVA_HOME%\bin\jps         ← for Windows
nnnnn DataNode                   ← Hadoop datanode
nnnnn NameNode                   ← Hadoop namenode
nnnnn NodeManager                ← Hadoop YARN node manager
nnnnn ResourceManager            ← Hadoop YARN resource manager
nnnnn SecondaryNameNode          ← Hadoop secondary namenode (may not have this)
```

Check Log Files

- Review logs files in `hadoop\logs` directory (and
- check there are no ERROR messages (a few WARN messages is usually ok)

Check Web Interfaces

- Hadoop Web UI http://localhost:50070
- try Utilities → Browse the Filesystem (it's empty at the moment)

Hadoop Command Line (OS X and Linux)

Hadoop Command Line

- many Unix shell file manipulation commands (`ls`, `mkdir`, `rm` etc) have Hadoop equivalents using `hadoop fs -<command>`
- for example: `$HADOOP_PREFIX/bin/hadoop fs -ls /user`
- see https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html

Create Your User Directories on the Hadoop Filesystem

```
$ source $SPA_2016/env.src
$ $HADOOP_PREFIX/bin/hadoop fs -mkdir -p /user/YOURNAME/load/lfb
$ $HADOOP_PREFIX/bin/hadoop fs -mkdir -p /user/YOURNAME/load/lhp
```

- replace `YOURNAME` with your operating system user name (user identities map 1-1 from the operating system)

Check it's Worked

```
$ $HADOOP_PREFIX/bin/hadoop fs -ls /user/YOURNAME/load
```

Hadoop Command Line (Windows)

Hadoop Command Line

- many Unix shell file manipulation commands (`ls`, `mkdir`, `rm` etc) have Hadoop equivalents using `%SPA_2016%\hadoop\bin\hdfs.cmd dfs -<command>`
- for example: `%SPA_2016%\hadoop\bin\hdfs.cmd -ls /user`
- see https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html
- Note this script is in `hadoop/bin`, not `hadoop/sbin`

Create Your User Directories on the Hadoop Filesystem

```
C:\> hdfs.cmd dfs -mkdir -p /user/YOURNAME/load/lfb
C:\> hdfs.cmd dfs -mkdir -p /user/YOURNAME/load/lhp
```

- replace `YOURNAME` with your operating system user name (user identities map 1-1 from the operating system)

Check it's Worked

```
C:\> hdfs.cmd dfs -ls /user/YOURNAME/load
Found 2 items
drwxr-xr-x   - spa16 supergroup          0 2016-06-19 15:16 /user/spa16/load/lfb
drwxr-xr-x   - spa16 supergroup          0 2016-06-19 15:16 /user/spa16/load/lhp
```

Load Some Data into Hadoop

London Fire Brigade Reported Incidents

- Original from http://data.london.gov.uk/dataset/london-fire-brigade-incident-records
- Covers the period 2013 - 2016
- I loaded it into Excel and converted into a 'Windows Comma-Separated' file
- You will find it in:
  - `$SPA_2016/datasets/lfb/load/lfb.csv` (*OS X,Linux*)
  - `C:\SPA_2016\datasets\lfb\load\lfb.csv` (*Windows*)
- There is a larger file, `lfb-large.csv`, if you want to explore with more data

Load the data into Hadoop

- Run the following command (split over multiple lines here for readability):
- for OS X and Linux:

```
$ $HADOOP_HOME/bin/hadoop fs -put \
   $SPA_2016/datasets/lfb/load/lfb.csv \
   hdfs://localhost:9000/user/YOURNAME/load/lfb
```

- for Windows:

```
C:\> hdfs.cmd dfs -put C:\SPA_2016\datasets\lfb\load\lfb.csv \
   hdfs://localhost:9000/user/YOURNAME/load/lfb
```

Check It Has Loaded Into Hadoop

Browse Hadoop from the Command Line

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/YOURNAME/load/lfb
C:\> hdfs.cmd dfs -ls /user/YOURNAME/load/lfb
Found 1 items
-rw-r--r--   3 spa16 supergroup   79888721 2016-05-29 10:53 /user/spa16/load/lfb.csv
```

- count the number of lines in the file

```
$ hadoop fs -cat /user/YOURNAME/load/lfb/lfb.csv | wc -l
C:\> hadoop.cmd fs -cat /user/YOURNAME/load/lfb/lfb.csv | find /c /v ""
322217
```

Browse Hadoop from your Web Browser

- http://localhost:50070/explorer.html
- look in /user/YOURNAME/load/lfb

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	spa16	supergroup	76.19 MB	6/5/2016, 5:46:50 PM	3	128 MB	lfb.csv

Real-World Big Data in Action

Spark Exercises

A Big Data Processing Engine

Start Spark Server (OS X and Linux)

Start Hadoop if not already running

- see previous slides (for Windows, don't forget to use a Command Prompt)

Start Spark Server

- for Windows, use Cygwin Terminal, not Command Prompt

```
$ source $HOME/SPA_2016/env.src
$ $SPARK_HOME/sbin/start-master.sh
$ $SPARK_HOME/sbin/start-slaves.sh spark://hostname:7077
```

- where `hostname` is the host name (or IP address) of your computer
- enter your password if prompted (Spark uses `ssh`)

Stop Spark Server

```
$ source $HOME/SPA_2016/env.src
$ $SPARK_HOME/sbin/stop-all.sh
```

Start Spark Server (Windows)

Start Hadoop if not already running

- see previous slides (for Windows, don't forget to use a Command Prompt)

Start Spark Server

- use a Command Prompt

```
C:\> cd \SPA_2016
C:\> start spark\bin\spark-class.cmd \
     org.apache.spark.deploy.master.Master
C:\> start spark\bin\spark-class.cmd \
     org.apache.spark.deploy.worker.Worker spark://IPADDRESS:7077
```

- `IPADDRESS` is the IP address of your computer (it's logged by the Master)
- the Worker may need several attempts to connect to the master

Stop Spark Server

- press Control-C in each Spark window or click the red X in the corner

Check Hadoop and Spark Are Running

Check Running Processes

```
$ jps | sort -k 2          ← for OS X and Linux
C:\> %JAVA_HOME%\bin\jps   ← for Windows
nnnnn DataNode             ← Hadoop datanode
nnnnn Master               ← Spark master
nnnnn NameNode             ← Hadoop namenode
nnnnn NodeManager          ← Hadoop YARN node manager
nnnnn ResourceManager      ← Hadoop YARN resource manager
nnnnn SecondaryNameNode    ← Hadoop secondary namenode
nnnnn Worker               ← Spark slave
```

Check Log Files

- Look in `$SPA_2016/logs` or `%SPA_2016%\logs`

Check Web Interfaces

- Hadoop Web UI http://localhost:50070
- Browse Hadoop Filesystem http://localhost:50070/explorer.html#
- Spark Web UI http://localhost:8080

Pyspark

Pyspark

- Pyspark allows you to submit Spark commands from a Python shell, in the same way you would invoke Spark programatically
- Pyspark is a wrapper script for `spark-submit`, which is a script you use to launch Spark applications (jar files) on a Spark cluster

Launching Pyspark

- Start Hadoop and Spark
- Start Pyspark:

```
$ $SPARK_HOME/bin/pyspark
C:\> \SPA_2016\env.cmd
C:\> %SPARK_HOME%\bin\pyspark.cmd --master spark://IPADDRESS:7077
```

- You should get the message:

```
SparkContext available as sc, HiveContext available as sqlContext.
```

- You can run any Python command at this point
- You can also call functions in the `pyspark.sql` library

```
>>> help(sqlContext)
```

Data Science Using Spark (1 of 2)

Load the LFB Data from the Hadoop Filesystem into Spark

- Enter the following command at the Pyspark prompt (*on one line, split here for readability*)

```
>>> lfb = sqlContext.read.format("com.databricks.spark.csv").
    option("header", "true").option("inferschema", "true").
    option("mode", "DROPMALFORMED").
    load("hdfs://localhost:9000/user/YOURNAME/load/lfb/lfb.csv")
```

Check it's Loaded

```
>>> print lfb.count()
...
322217
```

Display the data column names

```
>>> lfb.printSchema()
```

Look at Some Data

```
>>> lfb.filter(lfb.IncidentGroup == "Special Service").limit(5).show()
```

Data Science Using Spark (2 of 2)

Incident Counts by Type

```
>>> lfb.groupBy("IncidentGroup").count().show()
```

Incident Counts by Stop Code

```
>>> lfb.groupBy("StopCodeDescription").count().show(truncate=False)
```

Most Dangerous Areas

```
>>> lfb.groupBy("Postcode_district").count(). \
        sort("count", ascending=False).show()
```

And What Happens There

```
>>> lfb.rollup("IncidentGroup","Postcode_district"). \
        count().sort("count", ascending=False).show()
```

"Frequent" Problem Areas

```
>>> for borough in sorted(lfb.freqItems(["IncGeo_BoroughName"]).first()[0]):
          print borough
```

← this line starts with a tab or some spaces (this is Python!)

Hive Exercises

A Big Data data warehousing infrastructure

Start Hive Server (OS X and Linux)

Start Hadoop and Spark if not already running

- see earlier slides

Start Hive Server

```
$ source $HOME/SPA_2016/env.src
$ nohup $HIVE_HOME/bin/hive --service hiveserver2 2>&1 > /dev/null &
```

Stop Hive Server

```
$ killall HiveServer2
```

Use the Start / Stop Script

```
$ source $HOME/SPA_2016/env.src
$ spa_2016.bash start
```

…

```
$ spa_2016.bash stop
```

Start Hive Server (Windows)

Start Hadoop and Spark if not already running

- see earlier slides

Start Hive Server

- use a Command Prompt

```
C:\> cd \SPA_2016
C:\> START %HIVE_HOME%\bin\hive.cmd --service hiveserver2
```

Stop Hive Server

- press Control-C in the Hive window or click the red X in the corner

Use the Start / Stop Script to Start All Servers

- use a Command Prompt

```
C:\> cd \SPA_2016
C:\> spa_2016.cmd start
…
C:\> spa_2016.cmd stop
```

Beeline

Beeline

- Beeline allows you to run Hive SQL queries from a command shell
- Beeline commands can span multiple lines and are terminated by a semicolon `;`
- Exit Beeline by typing `!quit` at the prompt

Run Beeline

```
$ $SPARK_HOME/bin/beeline -u jdbc:hive2:// --color
```

- *Do not run the version of Beeline in `$HIVE_HOME/bin`!*

```
C:\> cd %SPA_2016%
C:\> spa_2016.cmd beeline
```

Check your Hive databases

```
0: jdbc:hive2://> SHOW DATABASES;
+----------------+--+
| database_name  |
+----------------+--+
| default        |
+----------------+--+
```

- You have an empty Hive installation

Data Science Using Hive (1 of 2)

Create your database

- Start beeline and enter the command:

```
0: jdbc:hive2://> create database spa_2016;
0: jdbc:hive2://> show databases;
```

Load the LFB Data into Hive

- Run this script, which creates a Hive external table called `lfb_data`

```
$ $SPARK_HOME/bin/beeline -u jdbc:hive2:// --color < \
      $SPA_2016/datasets/lfb/load_external.hive
C:\> spa_2016.cmd beeline < C:\SPA_2016\datasets\lfb
\load_external.hive
```

Check It's Loaded

- Run these commands in beeline

```
0: jdbc:hive2://> use spa_2016;
0: jdbc:hive2://> select count(*) from lfb_data;
0: jdbc:hive2://> describe lfb_data;
```

- The table should contain 322,217 rows

Data Science Using Hive (2 of 2)

Incident Counts by Type

```
//> select incidentgroup, count(*) from lfb_data group by incidentgroup;
```

Incident Counts by Stop Code

```
//> select stopcodedescription, count(*) from lfb_data
      group by stopcodedescription;
```

Most Dangerous Areas

```
//> select postcode_district, incgeo_boroughname, count(*) as c
     from lfb_data group by postcode_district, incgeo_boroughname
     having c> 1000 order by c desc limit 10;
```

And What Happens There

```
//> select postcode_district, incidentgroup, count(*) as c
     from lfb_data group by postcode_district, incidentgroup
     having c> 1000 order by c desc;
```

"Frequent" Problem Areas

- no Hive equivalent to Spark `freqItems`

Additional Exercises

Even Bigger Big Data

Load Even Bigger Data into Hadoop

London House Prices

- Original from http://data.london.gov.uk/dataset/average-house-prices-borough
- Four CSV files, covers the period 1995 - 2014
- You can find them in `$SPA_2016/datasets/lfb/load/*.csv`

Load the data into Hadoop using `hadoop fs`

```
$ $HADOOP_HOME/bin/hadoop fs -put \
   $SPA_2016/datasets/lfb/load/*.csv \
   hdfs://localhost:9000/user/YOURNAME/load/lhp
```

Load the data from Hadoop into Spark using Pyspark

```
>>> lfb = sqlContext.read.format('com.databricks.spark.csv').
     option('header', 'true').option('inferschema', 'true').
     option('mode', 'DROPMALFORMED').load('/user/YOURNAME/load/lhp/*.csv')
```

Load the data from Hadoop into Hive using Spark Beeline

- Run this script, which creates a Hive external table called `lfb_data`

```
$ $SPARK_HOME/bin/beeline -u jdbc:hive2:// --color < \
      $SPA_2016/datasets/lfb/load_external.hive
```

Data Science With Bigger Data

Average London House Prices by Year

```
//> select year, avg(price) avg_price from lhp_data
      group by year order by year;
```

Average Borough House Prices by Year

```
//> select local_authority, year, avg(price) avg_price from lhp_data
      group by local_authority, year order by local_authority, year;
```

Maximum and Minimum House Prices by Year

```
//> select year, max(price) max_price, min(price) min_price from lhp_data
      group by year order by year;
```

Prices by Property Type and Year

```
//> select property_type, year, avg(price) avg_price from lhp_data
      group by property_type, year order by avg_price desc;
```

Prices by Tenure and Year

```
//> select tenure, year, count(*) count, avg(price) avg_price from lhp_data
      group by tenure, year order by tenure, year;
```

- I assume tenure means freehold or leasehold etc.

Appendix

Further Information and Troubleshooting

Web UIs

Hadoop

- Namenode: http://localhost:50070/
- Datanodes: http://localhost:50075/
- Secondary Namenode: http://localhost:50090/
- YARN Resource Manager: http://localhost:8088/cluster
- see: http://blog.cloudera.com/blog/2009/08/hadoop-default-ports-quick-reference/

Spark

- Spark management interface: http://localhost:8080
- PySpark UI: http://localhost:4040/

Hive

- as defined in hive-site.xml

Troubleshooting

Hints and Tips

- logging out or rebooting in Windows fixes many problems
- avoid pathnames with spaces - this will break many commands
- don't forget to `source env.src` (OS X / Linux) or run `env.bat` (Windows)
- on Windows, make sure that `hadoop\bin` and `%JAVA_HOME%\bin` are in your `%PATH%`
- on Windows you may need to clear the contents of the temporary directory `C:\Users\spa16\AppData\Local\Temp`

Install Cygwin (Windows 64-bit only)

- Provides a BASH shell to run scripts and commands (you can't use it to run the Big Dat tools though)
- Download from https://cygwin.com/install.html
- A bit easier to use for debugging than the Windows Command Prompt

Troubleshooting Commands

- what is listening on a port?

```
$ sudo lsof -i -n -P | grep TCP | grep $PORT # OS X
$ sudo netstat -tulpn | grep :$PORT          # Linux
```

- set Hadoop debug level

```
$ $HADOOP_HOME/bin/hadoop daemonlog -setlevel 127.0.0.1:50070 \
    org.apache.hadoop.hdfs.server.namenode.NameNode DEBUG
```

LFB Hive Schema

```
root
 |-- IncidentNumber: string (nullable = true)
 |-- DateOfCall: string (nullable = true)
 |-- TimeOfCall: string (nullable = true)
 |-- IncidentGroup: string (nullable = true)
 |-- StopCodeDescription: string (nullable = true)
 |-- SpecialServiceType: string (nullable = true)
 |-- PropertyCategory: string (nullable = true)
 |-- PropertyType: string (nullable = true)
 |-- AddressQualifier: string (nullable = true)
 |-- Postcode_full: string (nullable = true)
 |-- Postcode_district: string (nullable = true)
 |-- IncGeo_BoroughCode: string (nullable = true)
 |-- IncGeo_BoroughName: string (nullable = true)
 |-- IncGeo_WardCode: string (nullable = true)
 |-- IncGeo_WardName: string (nullable = true)
 |-- Easting_m: string (nullable = true)
 |-- Northing_m: string (nullable = true)
 |-- Easting_rounded: integer (nullable = true)
 |-- Northing_rounded: integer (nullable = true)
 |-- FRS: string (nullable = true)
 |-- IncidentStationGround: string (nullable = true)
 |-- FirstPumpArriving_AttendanceTime: string (nullable = true)
 |-- FirstPumpArriving_DeployedFromStation: string (nullable = true)
 |-- SecondPumpArriving_AttendanceTime: string (nullable = true)
 |-- SecondPumpArriving_DeployedFromStation: string (nullable = true)
 |-- NumStationsWithPumpsAttending: string (nullable = true)
 |-- NumPumpsAttending: string (nullable = true)
```

LHP Hive Schema

```
root
 |-- id: string (nullable = true)
 |-- transaction_id: string (nullable = true)
 |-- price: string (nullable = true)
 |-- date_processed: string (nullable = true)
 |-- quarter: string (nullable = true)
 |-- month: string (nullable = true)
 |-- year: string (nullable = true)
 |-- year_month: string (nullable = true)
 |-- post_code: string (nullable = true)
 |-- property_type: string (nullable = true)
 |-- whether_new: string (nullable = true)
 |-- tenure: string (nullable = true)
 |-- address1: string (nullable = true)
 |-- address2: string (nullable = true)
 |-- address3: string (nullable = true)
 |-- address4: string (nullable = true)
 |-- town: string (nullable = true)
 |-- local_authority: string (nullable = true)
 |-- county: string (nullable = true)
 |-- record_status: string (nullable = true)
 |-- post_code_clean: string (nullable = true)
 |-- inner_outer: string (nullable = true)
 |-- borough_code: string (nullable = true)
 |-- borough_name: string (nullable = true)
 |-- ward_code: string (nullable = true)
 |-- ward_name: string (nullable = true)
 |-- msoa11: string (nullable = true)
 |-- lsoa11: string (nullable = true)
 |-- oa11: string (nullable = true)
```

Spark SQL Cheat Sheet

SQL	Pyspark
`select col1, ... from mutable`	`dataFrame.select(col1, ...)`
`select count(*) from mytable`	`dataFrame.count()`
`select col1, col2, count(*) ... group by...`	`dataFrame.cube(col1, col2, ...)`
`select distinct ...`	`dataFrame.distinct()`
`select ... where ...`	`dataFrame.filter(expression)`
	`dataFrame.groupBy(col1, ...)`
`select ... limit ...`	`dataFrame.limit(`
`select ... order by ...`	`dataFrame.orderB` `ascending=T`

TO DO

etc (MORE WORK ON THIS)

use quotes if columns are reserved words

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame