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 Runtime (JDK) 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

Create Your Project Directory

Start a BASH Shell

- OS X: Applications → Terminal
- Linux: Start Terminal from the toolbar

Create the Project Subdirectory

```
$ mkdir -p $HOME/SPA_2016/
```

Clone the Project files

```
$ cd $HOME/SPA_2016
$ git clone https://github.com/rozanski/bcs_spa16.git .
```

- don't miss out the dot at the end of the command!
- otherwise you will have to:

```
$ mv $HOME/SPA_2016/bcs_spa16/* $HOME/SPA_2016/
```

Set Environment Variables

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…

Install the Big Data Software

Download Hadoop into `$SPA_2016/hadoop`

- https://www.apache.org/dyn/closer.cgi/hadoop/common/
- Select stable source, download and extract the binary tarball
- You should end up with directories `$SPA_2016/spark/bin`, `etc`, `logs`, `sbin`…
- (you could also use `brew` for Mac, `apt` for Linux, but we won't for this demo)

Download Spark into `$SPA_2016/spark`

- https://spark.apache.org/downloads.html
- Choose the package type "pre-built for Hadoop 2.6 and later"
- Download and extract the binary tarball
- You should end up with directories `$SPA_2016/hadoop/bin`, `conf`, `logs`, `sbin`…

Download Hive into `$SPA_2016/hive`

- https://www.apache.org/dyn/closer.cgi/hive/
- Download and extract the latest binary tarball
- You should end up with directories `$SPA_2016/hive/bin`, `conf`…

I will also provide the software on a USB stick

Configure the Big Data Software

Sample Configuration Files

- Sample configuration files for the session are in directories under `$SPA_2016/config`
- You need to copy these files into the software directories, and then edit them for your setup
- You will need to change YOURNAME to your operating hostname (eg `nick`)
- You will need to change YOURHOME to your home directory (eg `/Users/nick`)

Copy `$SPA_2016/config/hadoop_etc_hadoop`

- Copy these files to `$SPA_2016/hadoop/etc/hadoop`

Copy `$SPA_2016/config/spark_conf`

- Copy these files to `$SPA_2016/spark/conf`

Copy `$SPA_2016/config/hive_conf`

- Copy these files to `$SPA_2016/hive/conf`

Configure Hadoop

Hadoop Configuration Files

- Hadoop configuration files are XML files
- Edit them as shown below
- Don't forget to change `YOURNAME` to your operating system hostname

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

`fs.defaultFS`	`hdfs://localhost:9000`	NameNode URI
`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`

`fs.defaultFS`	`hdfs://localhost:9000`	NameNode URI
`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*)	

Configure Spark

Spark Configuration Including CSV Support

- Edit Spark configuration script and files as shown below
- Download `spark-csv` from https://spark-packages.org/package/databricks/spark-csv
- Save the latest JAR into `$SPA_2016/spark/lib`

`$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 download below

Configure Hive

Hive Configuration and Metastore Database

- Edit Hive configuration files as shown below
- Create Hive megastore database (which uses Derby to store Hive metadata)

```
$ source $HOME/SPA_2016/env.src
$ mkdir $SPA_2016/data/hive
$ cd $SPA_2016/data/hive
$ $HIVE_HOME/bin/schematool -initSchema -dbType derby
```

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

`$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.

`$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

Set up Passphraseless SSH

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

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

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

Start (and Stop) Hadoop Server

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”

Check Hadoop is Running

Check Running Processes

```
$ jps | sort -k 2
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
```

Check Log Files

```
egrep 'WARN|ERROR' $SPA_2016/hadoop/logs/hadoop-<user>-namenode-<hostname>.log
egrep 'WARN|ERROR' $SPA_2016/hadoop/logs/hadoop-<user>-datanode-<hostname>.log
egrep 'WARN|ERROR' $SPA_2016/hadoop/logs/hadoop-<user>-secondarynamenode-
           <hostname>.log
```

- 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

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
```

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 can find it in `$SPA_2016/datasets/LFB/load/LFB.csv`
- There is a larger file, `LFB-large.csv`, if you want to play around with more data

Load the data into Hadoop

- Run the following command (split over three lines here for readability):

```
$ $HADOOP_HOME/bin/hadoop fs -put \
   $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
Found 1 items
-rw-r--r--   3 nick supergroup   79888721 2016-05-29 10:53 /user/nick/load/LFB.csv
```

- count the number of lines in the file

```
$ hadoop fs -cat /user/YOURNAME/load/lfb/LFB.csv | wc -l
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--  nick 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 (and Stop) Spark Server

Start Hadoop if not already running

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

Start Spark Server

```
$ 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
```

Check Hadoop and Spark Are Running

Check Running Processes

```
$ jps | sort -k 2
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

- `egrep 'WARN|ERROR' $SPA_2016/spark/logs/*Master*.out`
- `egrep 'WARN|ERROR' $SPA_2016/spark/logs/*Worker*.out`

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
```

- 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('/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!)

Real-World Big Data in Action

Hive Exercises

A Big Data data warehousing infrastructure

Start (and Stop) Hive Server

Start Hadoop if not already running

- see earlier slide

Start Spark Server if not already running

- see earlier slide

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
```

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` !*

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;
```

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
```

Check It's Loaded

- Run these commands in beeline

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

Real-World Big Data in Action

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/LHP/load/*.csv`

Load the data into Hadoop using `hadoop fs`

```
$ $HADOOP_HOME/bin/hadoop fs -put \
   $SPA_2016/datasets/LHP/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/LHP/load_external.hive
```

Next Steps

- London House Prices
- YARN distributed processing

TO DO

Real-World Big Data in Action

Appendix

Further Information and Troubleshooting

Troubleshooting

Hints and Tips

- avoid pathnames with spaces - this will break many commands

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 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(n)`
`select ... order by ...`	`dataFrame.orderBy([col1, …], ascending=True|False)`

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

Real-World Big Data in Action

Appendix

Settings Detail

Hadoop Configuration (1 of 2)

`$HADOOP_PREFIX/etc/hadoop/core-site.xml`

```
   <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
        <description>NameNode URI</description>
    </property>
    <property>
        <name>hadoop.proxyuser.YOURNAME.hosts</name>
        <value>*</value>
    </property>
    <property>
        <name>hadoop.proxyuser.YOURNAME.groups</name>
        <value>*</value>
    </property>
```

- replace `YOURNAME` with your operating system user name (no spaces!)

Hadoop Configuration (2 of 2)

`$HADOOP_PREFIX/etc/hadoop/hdfs-site.xml`

```
•    <property>
          <name>dfs.datanode.data.dir</name>
          <value>file:///YOURHOME/SPA_2016/data/hadoop/hdfs/datanode</
value>
          <description>
              Filesystem paths where the DataNode stores its blocks
          </description>
      </property>
      <property>
          <name>dfs.namenode.name.dir</name>
          <value>file:///YOURHOME/SPA_2016/data/hadoop/hdfs/namenode</
value>
          <description>
              Filesystem path where NameNode stores the namespace and
logs
          </description>
      </property>
```

- change `/YOURHOME` to your home directory (eg `/home/nick` or `/Users/nick`)

Spark Configuration

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

- add the lines:

```
export HADOOP_CONF_DIR=/YOURHOME/SPA_2016/hadoop/etc/hadoop
export SPARK_LOCAL_DIRS=/YOURHOME/SPA_2016/data/spark
```

- where `/YOURHOME` is your home directory
- these tell Spark where to find files on the local filesystem

Create `$SPA_2016/spark/conf/slaves`

- make sure the file includes:

```
localhost
```

Download Spark CSV Support

- Download `spark-csv` from https://spark-packages.org/package/databricks/spark-csv
- Save the latest JAR into `$SPA_2016/spark/lib`

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

- add the line:

```
spark.jars.packages com.databricks:spark-csv_2.11:1.4.0
```

- make sure the version (11.1.4.0) matches the version of the JAR you downloaded!

Hive Configuration(1 of 2)

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

```
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby:;databaseName=/YOURHOME/SPA_2016/data/hive/
metastore_db;create=true</value>
    <description>JDBC connect string for a JDBC metastore</description>
  </property>
  <property>
    <name>hive.execution.engine</name>
    <value>spark</value>
    <description>
    Expects one of [mr, tez, spark].
    Chooses execution engine. Options are: mr (Map reduce, default), tez, spark.
While MR
    remains the default engine for historical reasons, it is itself a historical
engine
    and is deprecated in Hive 2 line. It may be removed without further warning.
    </description>
  </property>
<configuration>
```

Hive Configuration(2 of 2)

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

- Ensure you have this line, which tells Hive where to find Spark

```
spark.master     spark://master:7077
```

- and this line:

```
spark.jars.packages com.databricks:spark-csv_2.11:1.4.0
```

Create Metastore Database

```
$ source $HOME/SPA_2016/env.src
$ mkdir $SPA_2016/data/hive
$ cd $SPA_2016/data/hive
$ $HIVE_HOME/bin/schematool -initSchema -dbType derby
```

- You should have a directory `$SPA_2016/data/hive/metastore_db$/`
- (this may not be necessary!)

Real-World Big Data in Action

Appendix

Big Data on Windows
NOT CURRENTLY WORKING

Windows Prerequisites

Install Cygwin (Windows 64-bit only)

- Provides a BASH shell to run scripts (not programs)
- Download from https://cygwin.com/install.html
- **Note that you must be running 64-bit Windows for Hadoop!**
- **DOESN'T WORK WITH HADOOP?**

Clone the Project Files

Clone the Project files

```
$ cd $HOME/SPA_2016
$ git clone https://github.com/rozanski/bcs_spa16.git .
```

- don't miss out the dot at the end of the command!
- for Cygwin, add the flag `--config core.autocrlf=input` (avoids CRLF issues)
- create the remaining directories:

Cygwin Setup

Extra Packages When Installing Cygwin

- git, openssh, …

Create a SPA 2016 User

- You need to add a user with a name without spaces (eg spa16)
- You can't do this form the UI since it demands a first and last name
- Run a Windows Command Prompt as Administrator

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

- Log out to Windows, and log back in again as the `spa16` user (you won't need to provide a password)
- Start a Cygwin Terminal
- Check you are running as the `spa16` user

```
$ pwd
/home/spa16
```