Installation Guide For Setting Up a Deep Learning Stack

By: Joojay Huyn May 2, 2016

The following installation guide describes how to set up an environment for developing deep learning programs to run on Apache Spark locally and deploying them on an AWS Spark EC2 cluster (with HDFS installed). The deep learning programs classify program traces as benign or malicious using a dataset derived from Michael Bartling's "mongoBUfeb26_2016 malobs2016" malware dataset.

Install Java 1.7 or later (preferably Java 1.8) and IntelliJ IDEA:

If you already have Java 1.7 or later, IntelliJ IDEA, and Git, you may skip this step. Note that even though you may have the above software already downloaded and installed, it may be a good idea to update to the latest version.

Install Java 1.7 or Java 1.8 (preferably Java 1.8) with these instructions:
http://deeplearning4j.org/quickstart#Java
OR you can go to this link: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Install IntelliJ IDEA with these instructions: http://deeplearning4j.org/quickstart#IntelliJ
OR you can go to this link: https://www.jetbrains.com/idea/download/

After IntelliJ IDEA is successfully installed, configure it to use your desired JDK by clicking Configure > Project Defaults > Project Structure (from the *Welcome to IntelliJ IDEA* screen the menu can be accessed in the bottom right, and from the full project window, use the File > Project Structure). Under Project SDK, set this to the correct JDK by specifying the desired Java home directory.

Install Scala: (this step is optional)

Download and install scala with these instructions: http://nd4j.org/getstarted.html#scala
With scala installed, you can quickly test out scala code snippets in scala's REPL (interactive shell, similar to the Python interpreter/shell)

Install SBT:

Download and install SBT using these platform-specific instructions:

- MAC OS X: http://www.scala-sbt.org/release/docs/Installing-sbt-on-Mac.html
- Linux: http://www.scala-sbt.org/release/docs/Installing-sbt-on-Linux.html
- Windows: http://www.scala-sbt.org/release/docs/Installing-sbt-on-Windows.html

The sbt command will be used to build/rebuild/remove jar artifacts that will be submitted to Spark (locally or remote cluster) for execution. Alternatively, it may be possible to rebuild these artifacts from IntelliJ IDEA through the SBT plugin, but this functionality is not enabled out of the box and I have not yet figured out how to configure IntelliJ IDEA.

Install IntelliJ IDEA Scala and SBT Plugin:

Install the IntelliJ scala and sbt plugin with these instructions:
http://nd4j.org/getstarted.html#scala
Click IntelliJ IDEA (top left corner) > Preferences > Plugins, search for "Scala" and "SBT" in all repositories, and install as needed. Sometimes, the scala plugin already comes with sbt.

Download Spark 1.6.0:

Go to the Spark Download page: http://spark.apache.org/downloads.html

In "Choose a Spark release:", select 1.6.0. (1.6.1 has some issues with the script that sets up a Spark EC2 cluster on AWS)

In "Choose a package type:", select "Pre-built for Hadoop 2.6 and later".

In "Choose a download type:", select "Direct Download"

Click the link to the right of "Download Spark:" to download Apache Spark 1.6.0

In the downloads folder of your local machine's file system, uncompress the downloaded Apache Spark .tgz file and move the resulting spark directory to a desired destination. (I chose the /Applications folder on my Mac OS X)

The logging statements that Spark programs print in the shell can be distracting. To control verbosity of logging, follow the next steps in the next few sentences. In the conf directory of the spark directory, create a file in the conf directory called log4j.properties. The Spark developers already included a template for this file called log4j.properties.template. To make the logging less verbose, make a copy of conf/log4j.properties.template called conf/log4j.properties and find the following line:

```
log4j.rootCategory=INFO, console
```

Then, lower the log level so that it shows only the WARN messages and above by changing it to the following:

```
log4j.rootCategory=WARN, console
```

Download Source Code for Deep Learning Programs in DL4S-SparkV2 Project:

Open up a terminal and navigate to a desired directory to hold the DL4S-SparkV2 project folder. This scala project is in an IntelliJ SBT project format. Then type:

```
$ git clone https://github.com/joojayhuyn/DL4S-SparkV2.git
```

This creates a project directory called DL4S-SparkV2 in the current directory you are in.

If you don't already have Git installed, install it with these instructions:
http://deeplearning4j.org/quickstart#Git
OR you can go to this link: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Import Spark Scala Project:

Open up the IntelliJ IDEA application. If you have any IntelliJ projects currently opened, close them by clicking File > Close Project.

In the Welcome to IntelliJ IDEA screen, click "Import Project". Select the DL4S-SparkV2 folder created by the git clone command. Then, choose "Import project from external model SBT".

If you happen to download an Apache Spark version other than 1.6.0, navigate to the build.sbt file, find the following lines:

```
"org.apache.spark" %% "spark-core" % "1.6.0",
"org.apache.spark" %% "spark-mllib" % "1.6.0",
```

Make sure the version number x.x.x matches the version of Spark you downloaded. Make changes as necessary.

Obtain Malware dataset:

Contact the UT ECE Spark Research Lab to obtain the MongoDB backup files (/data/joojayhuyn-spring2016-malware-datasets/mongoBUfeb26_2016/), raw data txt files (/data/joojayhuyn-spring2016-malware-datasets/Mal.txt and / data/joojayhuyn-spring2016-malware-datasets/Ben.txt), or csv files (/data/joojayhuyn-spring2016-malware-datasets/Mal.csv and /data/joojayhuyn-spring2016-malware-datasets/Ben.csv). These files are located in the central repository (/data/joojayhuyn-spring2016-malware-datasets/).

Run Program on local machine:

Open the howToRun.txt file in the DL4S-SparkV2 project directory. At the top, this file shows a template on how to run the main programs of this project. To run a main program (E.g. SVMDriver) on your local machine, modify a command under the "On local machine:" section. Obviously in this command, you will have to modify:

- The path to the Spark directory to match the location on your machine
- The path to the DL4S-SparkV2 jar file (this should be /path/to/DL4S-SparkV2/target/scala-2.10/dl4s-sparkv2_2.10-1.0.jar)
- Directory containing both the Ben.csv and Mal.csv files

Note that howToRun.txt shows you how to run your program on Spark from the command line, using the spark-submit command. You may wonder whether you can instead use IntelliJ IDEA directly to run/debug your program on Spark. While this is certainly very desirable, I have not yet figured out how.

When making changes to source code, navigate to the DL4S-SparkV2 project directory (make sure you are at the top level of this project directory) and type:

```
$ sbt clean package
```

to recompile your project and repackage the executable jar file to contain the modified source code. Then, you can run the latest version of the main program. Note that "sbt clean package" creates an executable jar file with only the source code in the project directory. An uber jar file would contain not only the source code in the project directory but also the jar files that the build.sbt file points to. I have not been able to successfully build an uber jar file. I think the commands involved are "sbt assembly" and "sbt build".

Install and setup Spark EC2 Cluster on AWS with HDFS: (Drawn from this source: https://www.cs.duke.edu/courses/fall15/compsci290.1/TA_Material/jungkang/how_to_run_spark_app.pdf)

Assuming that you have done the following:

- An AWS account already set up
- Created an EC2 key pair

- Downloaded the private key to your local machine
- Set the permissions for the private key file (myPrivateKeyFile.pem) to 400
- Moved the private key file to the ~/.ssh directory

Create an AWS access key from the AWS console (if you have not done so already) and set the environment variables AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY to your Amazon access key ID and secret access key. For example, I added the following lines to my ~/.bash_profile file:

```
export AWS_ACCESS_KEY_ID=<the aws access key id>
export AWS_SECRET_ACCESS_KEY=<the aws secret access key>
```

In the terminal type:

```
$ /path/to/spark-1.6.0-bin-hadoop2.6/ec2/spark-ec2 \
--key-pair=<name of key pair>
--identity-file=/path/to/.pem \
--region=<the default region for your key> \
--instance-type=m4.xlarge \
--slaves=4 \
launch <desired name for spark cluster>
```

(See setupSparkEC2Cluster.sh in DL4S-SparkV2 project directory for an example command. Pick the desired number of slaves for this cluster and desired instance-type. Note that if the instance-type is too small, you may run out of memory when running the Spark program)
After the cluster is setup, be sure to copy the and save the master node's ip address and public dns address to a file. The master's ip address and public dns address will come in handy.

SSH into the master node with this command:

```
$ ssh –i /path/to/.pem ec2-user@<master ip address>
```

The logging statements that Spark programs print in the shell can be distracting. To control verbosity of logging, follow the next steps in the next few sentences. (These next steps may require a "sudo" command.) In the /root/spark/conf/ directory, create a file called log4j.properties. The Spark developers already included a template for this file called log4j.properties.template. To make the logging less verbose, make a copy of conf/log4j.properties.template called conf/log4j.properties (you may need to change permissions of this file with a chmod) and find the following line:

```
log4j.rootCategory=INFO, console
```

Then, lower the log level so that it shows only the WARN messages and above by changing it to the following:

```
log4j.rootCategory=WARN, console
```

Now open up the /root/spark/conf/spark-defaults.conf file, add the following line: (you may need to change permissions of the file with a chmod and you may need a sudo command)
spark.driver.memory <whatever amount of memory is given to the spark.executor.memory line>
(This step may not be necessary, but it helps to avoid OOM errors)

Then, disseminate the configuration directory to worker nodes by typing:

```
$ sudo /root/spark-ec2/copy-dir /root/spark/conf
```

Create HDFS Directory:

In a terminal, ssh into the master node of the Spark EC2 cluster if you haven't already (ssh –i /path/to/.pem ec2-user@<master ip address>). Let's create a directory in HDFS that will hold the input data files with this command: (The –p flag creates parent directories along the path)

```
$ sudo /root/ephemeral-hdfs/bin/hadoop fs –mkdir –p /user/<username that identifies you>
```

Upload files to HDFS:

Open up a terminal and navigate to the directory on your local machine containing the Mal.csv and Ben.csv files. Compress these files with this command in the terminal:

```
$ gzip Mal.csv
$ gzip Ben.csv
```

Transfer these files the home directory of the master node of the Spark EC2 cluster with this command:

```
$ scp –i /path/to/.pem /path/to/Mal.csv.gz ec2-user@<master node ip address>:/home/ec2-user/
$ scp –i /path/to/.pem /path/to/Ben.csv.gz ec2-user@<master node ip address>:/home/ec2-user/
```

Now, ssh into the master node of the Spark EC2 cluster (ssh –i /path/to/.pem ec2-user@<master ip address>), and navigate to the directory (should be /home/ec2-user/) containing the Mal.csv.gz and Ben.csv.gz file.

Uncompress those files with this command:

```
$ gunzip Mal.csv.gz
$ gunzip Ben.csv.gz
```

Copy these files to the desired directory in HDFS on the Spark EC2 cluster with this command:

```
$ sudo /root/ephemeral-hdfs/bin/hadoop fs –put /path/to/Ben.csv /user/<username that identifies you>
$ sudo /root/ephemeral-hdfs/bin/hadoop fs –put /path/to/Mal.csv /user/<username that identifies you>
```

Compile Application Code:

Open up a terminal and navigate to the DL4S-SparkV2 directory on your local machine. Type "sbt clean package" in the terminal to create an executable JAR file of the project. Then, transfer the jar file to the master node with this command:

```
$ scp –i /path/to/.pem /path/to/DL4S-SparkV2/target/scala-2.10/dl4s-sparkv2_2.10-1.0.jar ec2-user@<master node ip address>:/home/ec2-user/
```

Then, open up the howToRun.txt file in the DL4S-SparkVS directory and modify the commands under the "On AWS Spark EC2 Cluster with HDFS:" section to reflect the new Spark EC2 cluster with HDFS you just created. For these commands, you will have to modify:

- The path to the executable jar file that sits on the master node of the Spark EC2 cluster (path should be /home/ec2-user/dl4s-sparkv2_2.10-1.0.jar)
- The HDFS data directory that stores the data files Ben.csv and Mal.csv
- The master node public dns

Again, you may choose to recompile your project by navigating to the DL4S-SparkV2 directory on your local machine and create an executable JAR file with "sbt clean package". Then, transfer the jar file to the master node with the above scp command.

Run Source Code on AWS:

Simply open up the howToRun.txt file in the DL4S-SparkV2 directory and copy a command under the "On AWS Spark EC2 Cluster with HDFS:" section. (By now, these commands should be updated to represent the latest Spark EC2 cluster you have created)

Open up a terminal and ssh into the master node of the Spark EC2 cluster.

Paste the command into the terminal and hit enter to run it.

Run DL4J Program on AWS:

To run the MBartlingDL4J.MLPDriver program, first scp the directory on sparkd (/data/joojayhuyn-spring2016-malware-datasets/dl4j_jars/) containing the necessary dl4j jar files to a desired directory on the AWS Spark EC2 cluster master node. Then in the howToRun.txt file under the “On AWS Spark EC2 Cluster with HDFS:” section, modify the following parts of the command that executes the MBartlingDL4J.MLPDriver program:

- The path to the directory that stores the DL4J jar files. (This directory sits on some designated directory – chosen by you – on the AWS Spark EC2 cluster master node)
- This command references 14 DL4J jar files on 1 executable jar file containing the Spark programs

Open up a terminal and ssh into the master node of the Spark EC2 cluster.

Paste the command that runs the MBartlingDL4J.MLPDriver program into the terminal and hit enter to run it.

Terminate the cluster:

On your local machine, open up a terminal and type:

$ /path/to/spark-1.6.0-bin-hadoop2.6/ec2/spark-ec2 –region=<ec2-region where Spark cluster resides> destroy <cluster name>