BLOG

How to install PySpark locally

POSTED BY JAKUB NOWACKI, 11 AUGUST 2017

For both our training as well as analysis and development in SigDelta, we often use Apache Spark's Python API, aka PySpark. Despite the fact, that Python is present in Apache Spark from almost the beginning of the project (version 0.7.0 to be exact), the installation was not exactly the pip-install type of setup Python community is used to.

This has changed recently as, finally, PySpark has been added to Python Package Index PyPI and, thus, it become much easier. In this post I will walk you through all the typical local setup of PySpark to work on your own machine. This will allow you to better start and develop PySpark applications and analysis, follow along tutorials and experiment in general, without the need (and cost) of running a separate cluster. Also, we will give some tips to often neglected Windows audience on how to run PySpark on your favourite system.

Prerequisites

Python

To code anything in Python, you would need Python interpreter first. Since I am mostly doing Data Science with PySpark, I suggest Anaconda by Continuum Analytics, as it will have most of the things you would need in the future. For any new projects I suggest Python 3.

Warning! There is a PySpark issue with Python 3.6 (and up), which has been fixed in Spark 2.1.1. If you for some reason need to use the older version of Spark, make sure you have older Python than 3.6. You can do it either by creating conda environment, e.g.:

```
conda create -n py35 python=3.5 anaconda
```

See this document for details.

Java

Since Spark runs in JVM, you will need Java on your machine. I suggest you get Java Development Kit as you may want to experiment with Java or Scala at the later stage of using Spark as well.

1. Java 8 JDK can be downloaded from the Oracle site.
2. Install Java following the steps on the page.
3. Add `JAVA_HOME` environment variable to your system
   - on *nix, e.g.: `export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64`
   - on Windows, e.g.: `JAVA_HOME: C:\Progra~1\Java\jdk1.8.0_141` see this description for dedails

Other tools

There are no other tools required to initially work with PySpark, nonetheless, some of the below tools may be useful.

For your codes or to get source of other projects you may need Git. It will also work great with keeping your source code changes tracking.

You may need to use some Python IDE in the near future; we suggest PyCharm for Python, or Intellij IDEA for Java and Scala, with Python plugin to use PySpark.

Hadoop binary (only for Windows users)

While Spark does not use Hadoop directly, it uses HDFS client to work with files. On the other hand, HDFS client is not capable of working with NTFS, i.e. the default Windows file system, without a binary compatibility layer in form of DLL file. You can build Hadoop on Windows yourself see this wiki for details), it is quite tricky. So the best way is to get some prebuild version of Hadoop for Windows, for example the one available on GitHub https://github.com/karthikj1/Hadoop-2.7.1-Windows-64-binaries works quite well.

1. Download file https://github.com/karthikj1/Hadoop-2.7.1-Windows-64-binaries/releases/download/v2.7.1/hadoop-2.7.1.tar.gz.
2. Extract it is some place, e.g. `C:\Tools\Hadoop` is a good place to start.
3. Create `HADOOP_HOME` environment variable pointing to your installation folder selected above.
4. Add Hadoop `bin` folder to your Windows `Path` variable as `%HADOOP_HOME%\bin`.
5. You may need to restart your machine for all the processes to pick up the changes.

Installing PySpark via PyPI

The most convenient way of getting Python packages is via PyPI using `pip` or similar command. For a long time though, PySpark was not available this way. Nonetheless, starting from the version 2.1, it is now available to install from the Python repositories. Note that this is good for local execution or connecting to a cluster from your machine as a client, but does not have capacity to setup as Spark standalone cluster: you need the prebuild binaries for that; see the next section about the setup using prebuilt Spark.

Thus, to get the latest PySpark on your python distribution you need to just use the `pip` command, e.g.:

```
pip install pyspark
```

If you work on Anaconda, you may consider using the distribution tools of choice, i.e. `conda`, which you can use as following:

```
conda install -c conda-forge pyspark
```

Note that currently Spark is only available from the `conda-forge` repository. Also, only version 2.1.1 and newer are available this way; if you need older version, use the prebuilt binaries.

Warning! `Pip/conda` install does not fully work on Windows as of yet, but the issue is being solved; see SPARK-18136 for details. Installing PySpark on Anaconda on Windows Subsystem for Linux works fine and it is a viable workaround; I've tested it on Ubuntu 16.04 on Windows without any problems.

Installing PySpark using prebuilt binaries

This is *the classical* way of setting PySpark up, and it' /'s the most versatile way of getting it. It requires a few more steps than the `pip`-based setup, but it is also quite simple, as Spark project provides the built libraries.

1. Get Spark from the project's download site.
   - You can select version but I advise taking the newest one, if you don't have any preferences.
   - You can select Hadoop version but, again, get the newest one 2.7.
2. Extract the archive to a directory, e.g.:
   - on *nix: `$HOME/tools/spark`
   - on Windows: `C:\Tools\spark`
3. Create `SPARK_HOME` environmental variable, e.g:
   - on *nix, e.g.: `export SPARK_HOME=$HOME/tools/spark`
   - on Windows, e.g.: `SPARK_HOME: C:\Tools\spark` see this description for dedails
4. Add Spark paths to PATH and PYTHONPATH environmental variables:
   - on *nix:
   - `export PATH=$PATH:$SPARK_HOME/bin`
   - `PYTHONPATH=$SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-<version>-src.zip:$PYTHONPATH` * on Windows, e.g.:
   - `Path: %Path%; %SPARK_HOME%\bin`
   - `PYTHONPATH: %SPARK_HOME%\python;%SPARK_HOME%\python\lib\py4j-<version>-src.zip:%PYTHONPATH%` * **Warning!** The version of Py4J source package changes between the Spark versions, thus, check what `<version>` you have in your Spark and change the placeholder accordingly.

You can now test Spark by running the below code in the PySpark interpreter:

```
# Start pyspark via provided command
import pyspark

# Below code is Spark 2+
spark = pyspark.sql.SparkSession.builder.appName('test').getOrCreate()

spark.range(10).collect()
```


LATEST BLOG POSTS

- StackOverflow tags with Dask
- Dask - a new elephant in the room
- Scala (and Java) Spark UDFs in Python
- Using language-detector aka large not serializable objects in Spark
- Text analysis in Pandas with some TF-IDF (again)
- Why SQL? Notes from (big) data analysis practice
- Word count is Spark SQL with a pinch of TF-IDF (continued)
- Word count is Spark SQL with a pinch of TF-IDF
- Docker for Data Science
- How to install PySpark locally
- Power BI - Self-service Business Intelligence tool
- Drools far beyond Hello World
- Big Data in the Maturity Stage

CONTACT

How can we help?

Drop us a line and we'll respond as soon as possible.

DROP US A LINE

Title

Email

Your message

Submit

Powered by Cognito Forms.

CALL US

(+48) 22 203 56 00
Nowogrodzka 62C, Warsaw, Poland
office@sigdelta.com
UTC / GMT +1

GET SOCIAL