Reading: How to Set Up Your Own Spark Environments

Estimated time needed: 5 minutes

After completing this reading, you will be able to set up and use Apache Spark on your computer. You will then create an environment to develop and test Spark applications.

Spark is a strong tool that lets you work with large data sets on multiple computers simultaneously.

Here's a simple guide to help you begin:

1. Prerequisites:

- **Java**: Spark is built on Java, so you'll need to have Java installed. Spark requires Java 8 or later versions.

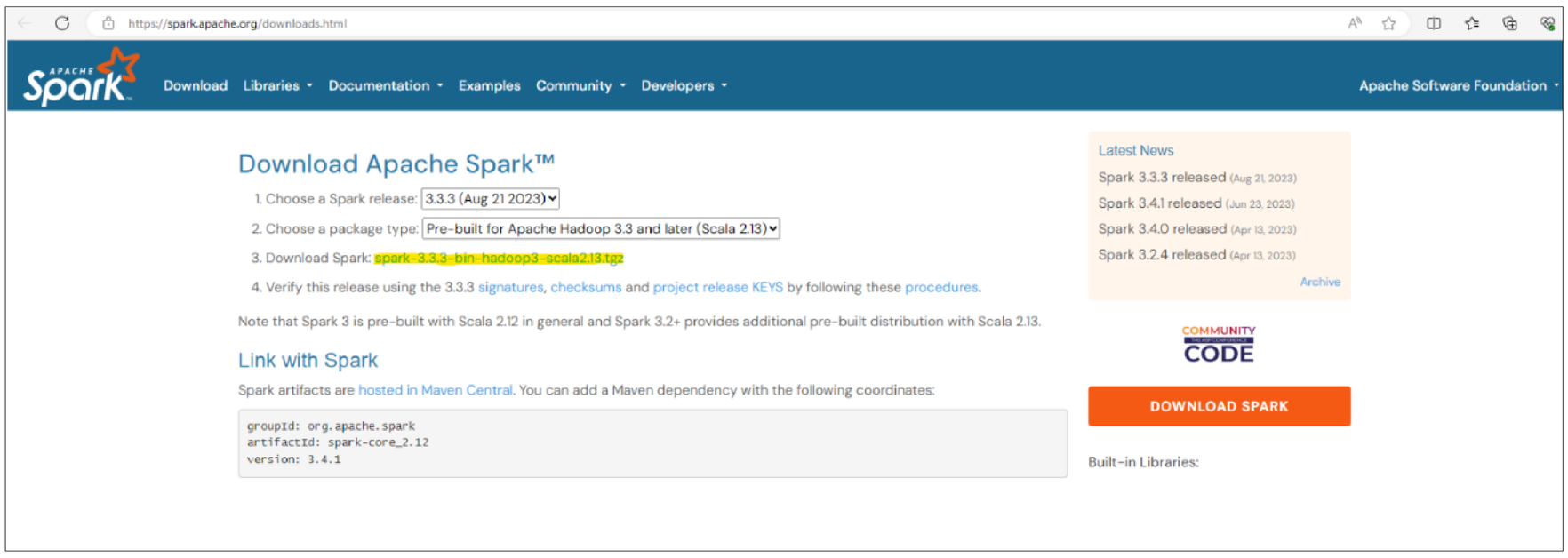
Please open the terminal (in Mac) or command prompt (in Windows), type `java -version`, and press return or enter.

```
C:\Users\DELL>java -version
java version "11.0.17" 2022-10-18 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.17+10-LTS-269)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.17+10-LTS-269, mixed mode)
```

- **Python (optional)**: While Spark is primarily written in Scala, it provides APIs for multiple languages, including Python. You can use Scala, Java, Python, or R to work with Spark.
- **Hadoop (optional)**: Spark can run on top of the Hadoop Distributed File System (HDFS), but you don't necessarily need Hadoop for local development.

2. Download Spark:

Go to the official Spark website (https://spark.apache.org/downloads.html) and download the latest version of Spark. Choose the prebuilt package for Hadoop with the appropriate version of Spark, Scala, and Hadoop.



3. Set up environment variables:

You need to set a couple of environment variables to make Spark work correctly:

- **SPARK_HOME**: Point this variable to the directory where you extracted Spark.
- **PATH**: Add `%SPARK_HOME%\bin` to your PATH to easily access Spark commands.

4. Configuration (Optional):

In the `conf` directory within your Spark installation, you'll find various configuration files. The most important is `spark-defaults.conf`, where you can set Spark properties. However, for local development, the default configurations are often sufficient.

5. Starting Spark:

Interactive Shell (Scala or Python): You can start the interactive Spark shell using the following commands:

- **Scala**: Run `spark-shell` in your terminal.
- **Python**: Run `pyspark` in your terminal.

```
C:\Users\DELL>pyspark
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/08/30 10:03:25 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
asses where applicable
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.2.2
      /_/

Using Python version 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021 11:48:03)
Spark context Web UI available at http://DESKTOP-LQII568:4040
Spark context available as 'sc' (master = local[*], app id = local-1693370006329).
SparkSession available as 'spark'.
>>>
```

Submitting applications: You can submit Spark applications in a similar way:

- **Scala**: `spark-submit --class <main-class> --master local <path-to-jar>`
- **Python**: `spark-submit --master local <path-to-python-script>`

6. Writing Spark applications:

Spark applications are typically written using the Spark APIs. You can use Resilient Distributed Datasets (RDDs) or DataFrames and data sets for more structured and optimized operations.

Here's a simple example using Python and DataFrames.

Input file: `data.csv`

Sample data:

Name	Score
A	10
B	15
A	20
B	5
A	30

```
from pyspark.sql import SparkSession
# Create a Spark session
spark = SparkSession.builder.appName("MySparkApp").getOrCreate()
# Load data
data = spark.read.csv("", header=True, inferSchema=True)
data.csv
# Perform operations
result = data.groupBy("name ").agg({"score": "sum"})
# Show result
result.show()
# Stop the Spark session
spark.stop()
```

7. Monitoring:

Spark provides a web-based interface (by default at http://localhost:4040) to monitor your Spark applications and their progress.

8. Clean up:

Make sure to stop the Spark session after you're done releasing resources.

```
spark.stop()
```

This is a basic guide to getting started with Spark on your own machines. For more advanced configurations and optimizations, you can refer to the official Spark documentation: https://spark.apache.org/documentation.html.

Author(s)

- Raghul Ramesh