Lectura: Cómo configurar tus propios entornos de Spark

Tiempo estimado necesario: 5 minutos

Después de completar esta lectura, podrás configurar y usar Apache Spark en tu computadora. Luego crearás un entorno para desarrollar y probar aplicaciones de Spark.

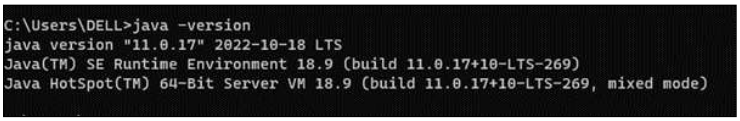
Spark es una herramienta poderosa que te permite trabajar con grandes conjuntos de datos en múltiples computadoras simultáneamente.

Aquí tienes una guía simple para ayudarte a comenzar:

1. Requisitos previos:

- **Java**: Spark está construido sobre Java, por lo que necesitarás tener Java instalado. Spark requiere Java 8 o versiones posteriores.

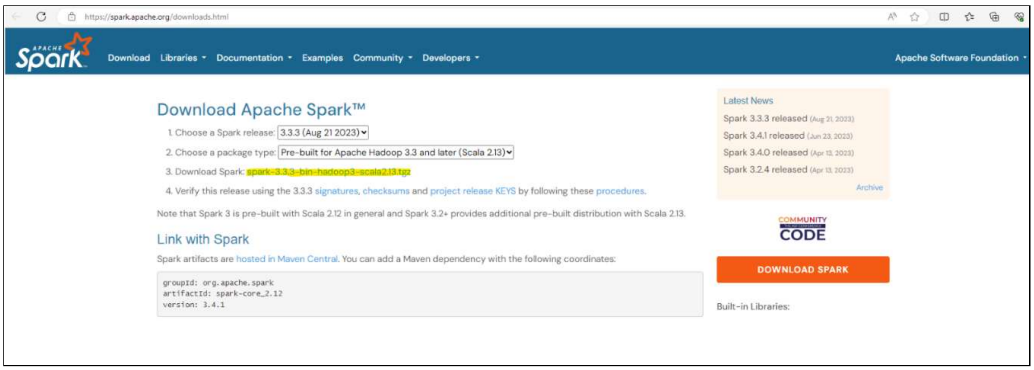
Por favor abre la terminal (en Mac) o el símbolo del sistema (en Windows), escribe `java -version`, y presiona return o enter.



- **Python (opcional)**: Aunque Spark está escrito principalmente en Scala, proporciona APIs para múltiples lenguajes, incluyendo Python. Puedes usar Scala, Java, Python o R para trabajar con Spark.
- **Hadoop (opcional)**: Spark puede ejecutarse sobre el Sistema de Archivos Distribuido de Hadoop (HDFS), pero no necesitas necesariamente Hadoop para el desarrollo local.

2. Descargar Spark:

Ve al sitio web oficial de Spark (https://spark.apache.org/downloads.html) y descarga la última versión de Spark. Elige el paquete preconstruido para Hadoop con la versión adecuada de Spark, Scala y Hadoop.



3. Configurar variables de entorno:

Necesitas configurar un par de variables de entorno para que Spark funcione correctamente:

- **SPARK_HOME**: Apunta esta variable al directorio donde extrajiste Spark.
- **PATH**: Agrega `%SPARK_HOME%\bin` a tu PATH para acceder fácilmente a los comandos de Spark.

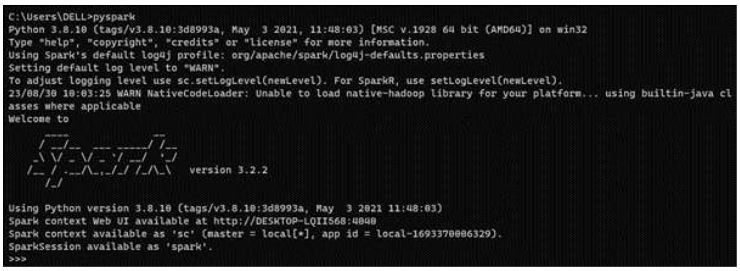
4. Configuración (Opcional):

En el directorio `conf` dentro de tu instalación de Spark, encontrarás varios archivos de configuración. El más importante es `spark-defaults.conf`, donde puedes establecer propiedades de Spark. Sin embargo, para el desarrollo local, las configuraciones predeterminadas suelen ser suficientes.

5. Iniciar Spark:

Shell interactivo (Scala o Python): Puedes iniciar el shell interactivo de Spark utilizando los siguientes comandos:

- **Scala**: Ejecuta `spark-shell` en tu terminal.
- **Python**: Ejecuta `pyspark` en tu terminal.



Envío de aplicaciones: Puedes enviar aplicaciones de Spark de manera similar:

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

6. Escribir aplicaciones de Spark:

Las aplicaciones de Spark se escriben típicamente utilizando las APIs de Spark. Puedes usar Conjuntos de Datos Distribuidos Resilientes (RDDs) o DataFrames y conjuntos de datos para operaciones más estructuradas y optimizadas.

Aquí tienes un ejemplo simple utilizando Python y DataFrames.

Archivo de entrada: `data.csv`

Datos de muestra:

Nombre	Puntaje
A	10
B	15
A	20
B	5
A	30

```
from pyspark.sql import SparkSession
# Crear una sesión de Spark
spark = SparkSession.builder.appName("MySparkApp").getOrCreate()
# Cargar datos
data = spark.read.csv("", header=True, inferSchema=True)
data.csv
# Realizar operaciones
result = data.groupBy("name ").agg({"score": "sum"})
# Mostrar resultado
result.show()
# Detener la sesión de Spark
spark.stop()
```

7. Monitoreo:

Spark proporciona una interfaz web (por defecto en http://localhost:4040) para monitorear tus aplicaciones de Spark y su progreso.

8. Limpieza:

Asegúrate de detener la sesión de Spark después de haber liberado los recursos.

```
spark.stop()
```

Esta es una guía básica para comenzar con Spark en tus propias máquinas. Para configuraciones y optimizaciones más avanzadas, puedes consultar la documentación oficial de Spark: https://spark.apache.org/documentation.html.

Autor(es)

- Raghul Ramesh