

Apache Spark

Máster en Data Science y Big Data

Miguel Ángel Corella
mcorella@geoblink.com

Marzo 2022

Contenido

1. Introducción
2. Hadoop vs. Spark
3. Componentes de Spark
4. Instalación de Spark (python)
5. Ejecución de Spark (python)
6. Referencias

Introducción

Hadoop... ¿la mejor solución?

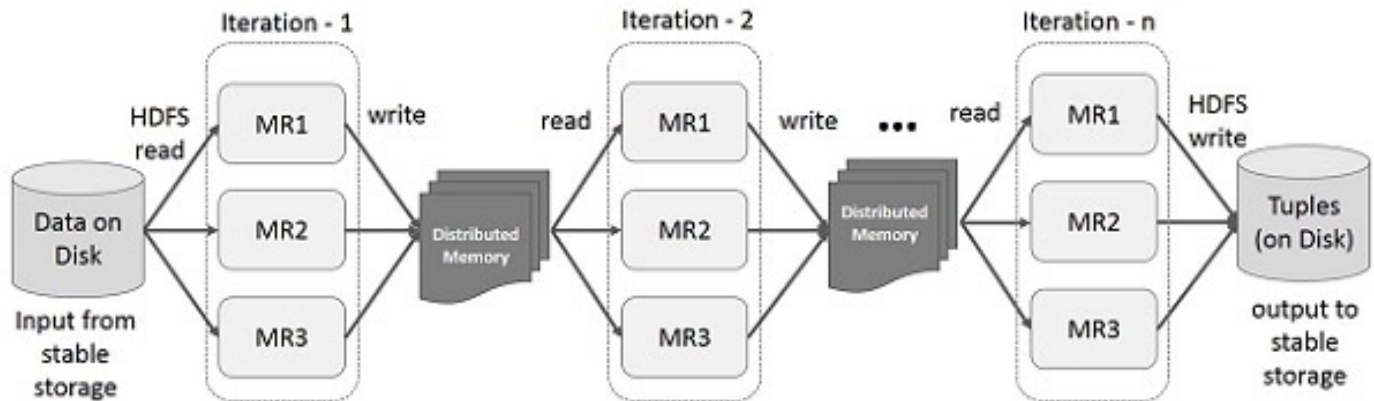
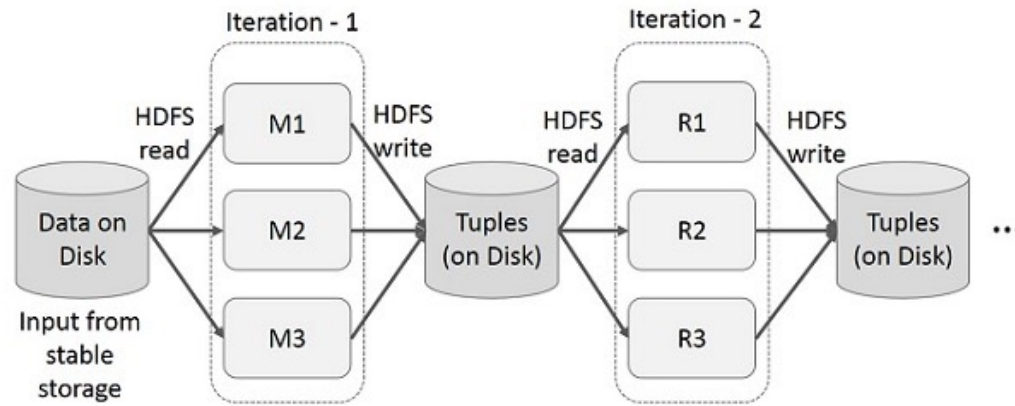
- Apache Hadoop (y sus componentes) ha habilitado el almacenamiento y procesamiento grandes volúmenes de información.
- Es decir, ha dado un **soporte tecnológico** a los problemas de datos sin el cual no podríamos hablar de Big Data.
- Sin embargo, con su uso se han ido detectando ciertas limitaciones en términos de:
 - **Velocidad:** permite ejecutar procesos de forma fiable, pero no es rápido.
 - **Facilidad de uso:** descomponer problemas en MapReduce no es fácil.
 - **Flexibilidad:** obliga a trabajar con un único lenguaje y una única arquitectura.
 - **Homogeneidad:** aunque el ecosistema evoluciona para dar solución a las diferentes carencias de Hadoop, produce a su vez entornos heterogéneos lo que complica su gestión y mantenimiento.

¿Qué es Apache Spark?

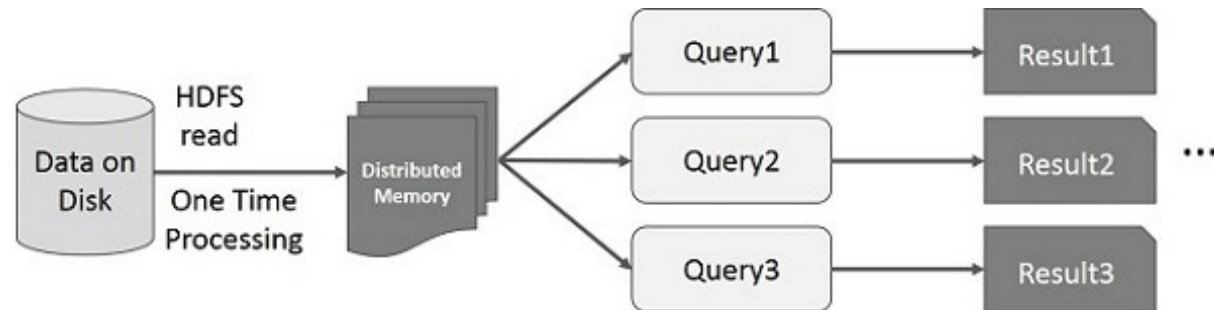
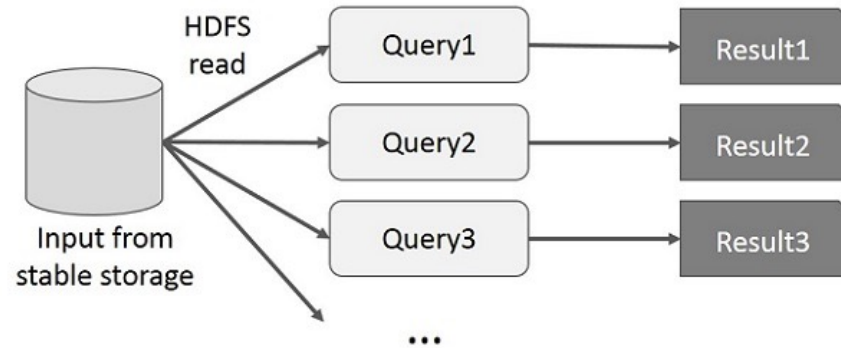
- Es un proyecto *open source* englobado dentro del ecosistema Hadoop (aunque quizá no debería).
- Es un motor de procesamiento (no de almacenamiento) basado en MapReduce con las siguientes características:
 - Basado en la ejecución de tareas *in-memory* y en disco (en lugar de sólo disco).
 - Desarrollado en Scala pero con interfaces “completas” para Java, Python y R.
 - Integrando múltiples formas de trabajo: *batch*, interactivo, *streaming*...
 - Soportando múltiples arquitecturas: *standalone*, ejecución sobre YARN...
- Es el sucesor de “Hadoop MapReduce” principalmente por su:
 - **Velocidad:** gracias a la ejecución en memoria y la optimización de tareas.
 - **Facilidad de uso:** funciones y modelos de programación más allá de MapReduce.
 - **Flexibilidad:** múltiples lenguajes, múltiples arquitecturas, múltiples formas de trabajo.
 - **Homogeneidad:** un único *framework* para todo en lugar de un ecosistema.

Hadoop vs. Spark

Operaciones iterativas



Operaciones interactivas

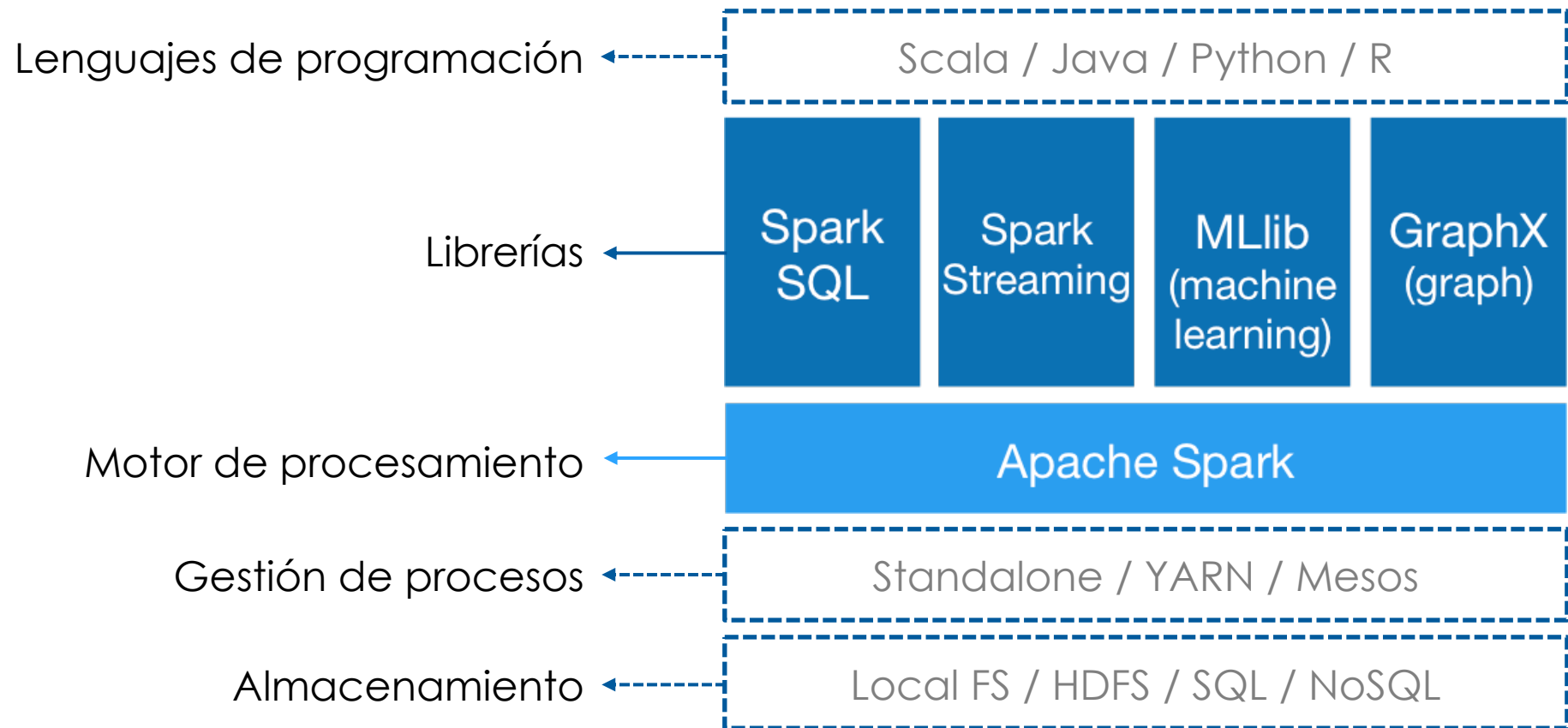


Hadoop vs. Spark

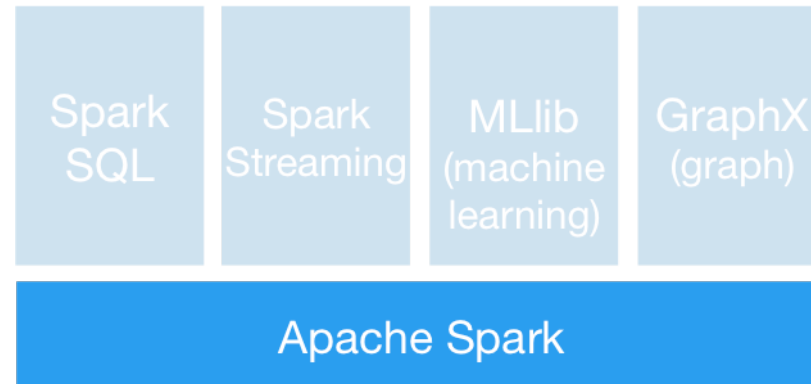
	Hadoop	Spark
Almacenamiento	Sólo disco	Memoria y disco
Operaciones	map, reduce	map, reduce, join, filter, sample...
Modo de trabajo	Batch	Batch, interactivo, streaming
Lenguajes	Java	Scala, Java, Python, R

Componentes de Spark

Componentes de Spark

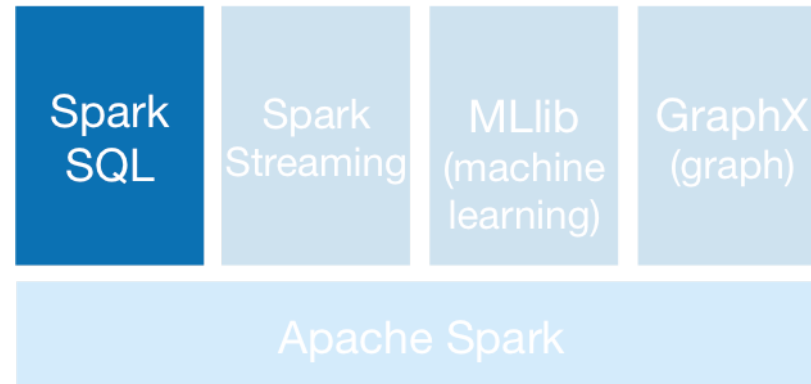


Spark Core



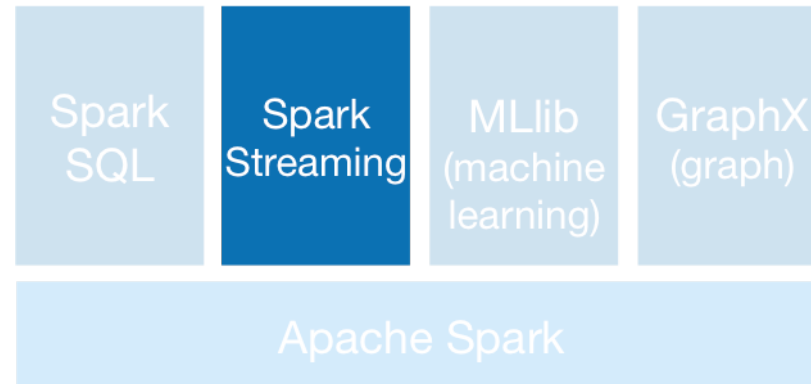
- Es el núcleo de Spark, el motor de ejecución distribuida *in-memory*.
- Ejecución de procesos muy rápida gracias a la carga de datos en memoria.
- Interfaz para múltiples lenguajes de programación:
 - Scala, Java, Python y R.
- Programación interactiva: mediante consolas/intérpretes.
 - Scala, Python y R.
- Independiente de arquitectura: múltiples gestores de procesos.
 - *Standalone*, Apache Hadoop YARN, Apache Mesos.

Spark SQL



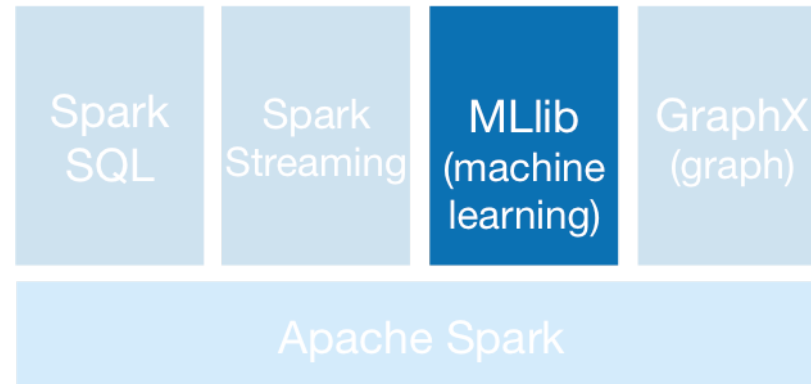
- Crea una capa de abstracción que codifica datos y operaciones con sintaxis SQL.
- Facilita la aproximación de desarrolladores RDBMS al mundo Big Data.
- Define esquemas lógicos (en memoria) que describen la estructura de datos de:
 - Tablas RDMBS.
 - NoSQL.
 - Ficheros de texto plano.
 - JSON.
 - ...

Spark Streaming



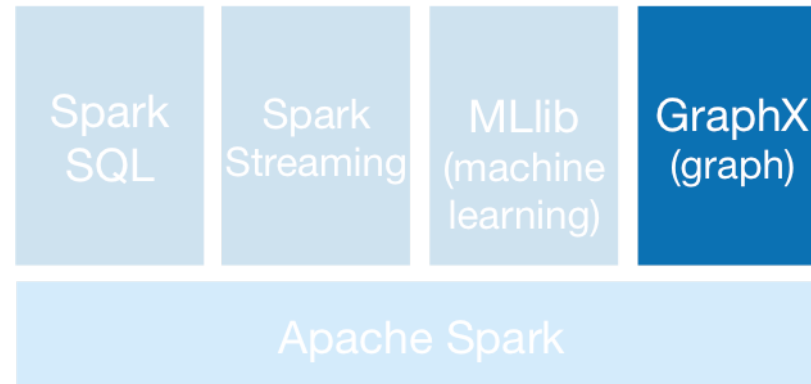
- Tradicionalmente se suele trabajar con datos estáticos (bases de datos, ficheros...).
- Pero existen aplicaciones que requieren de procesamiento en tiempo real:
 - Detección de fraude, filtrado de spam, redes sociales....
- Streaming permite definir procesos que se deben ejecutar en tiempo real sobre bloques de datos que pueden llegar en cualquier momento.
- Estas fuentes de flujos de datos pueden ser muy variadas:
 - Ficheros de texto plano incrementales, TCP/IP, Flume, Kafka, Twitter API, etc...

Spark MLlib



- Ofrece una librería de funciones y algoritmos de *machine learning* que se pueden ejecutar de forma distribuida *in-memory* sobre sets de datos también distribuidos.
 - Estadística descriptiva.
 - Aprendizaje supervisado: clasificación y regresión.
 - Aprendizaje no supervisado: *clustering*.
 - Recomendación: *collaborative filtering*.
 - *Feature engineering*: reducción, extracción y selección de variables.
 - Optimización.
 - ...

Spark GraphX



- Existen problemas para los que las estructuras establecidas por RDBMS no están indicadas (lentitud, dificultad de modelización...).
- Las bases de datos NoSQL basadas en grafos (como Neo4J) han dado solución a este problema.
- Spark GraphX permite llevar a cabo el procesamiento de datos cuya estructura está representada en forma de grafo.
- Es el sucesor de Apache Giraph, que habilitaba el procesamiento de este tipo de estructuras mediante MapReduce.

Instalación de Spark (python)

Modos de instalación de Spark

- Spark ofrece dos modos bien diferenciados de instalación.
- *Standalone*:
 - Necesidades más simples de “configuración”.
 - Ideal para entornos de desarrollo o trabajo con volúmenes de información “intermedios”.
 - Permite/facilita el trabajo con ficheros “locales”.
- *Cluster*:
 - Mayor complejidad de “configuración” (es necesario configurar el cluster).
 - Ideal en entornos de producción.
 - Orientado al trabajo con ficheros “distribuidos”.

Spark *standalone* en local

1. Descargar e instalar una distribución de Python 3 (base, Anaconda, Canopy...) e incluir, si es necesario, el proyecto Jupyter.



<http://www.python.org/downloads>



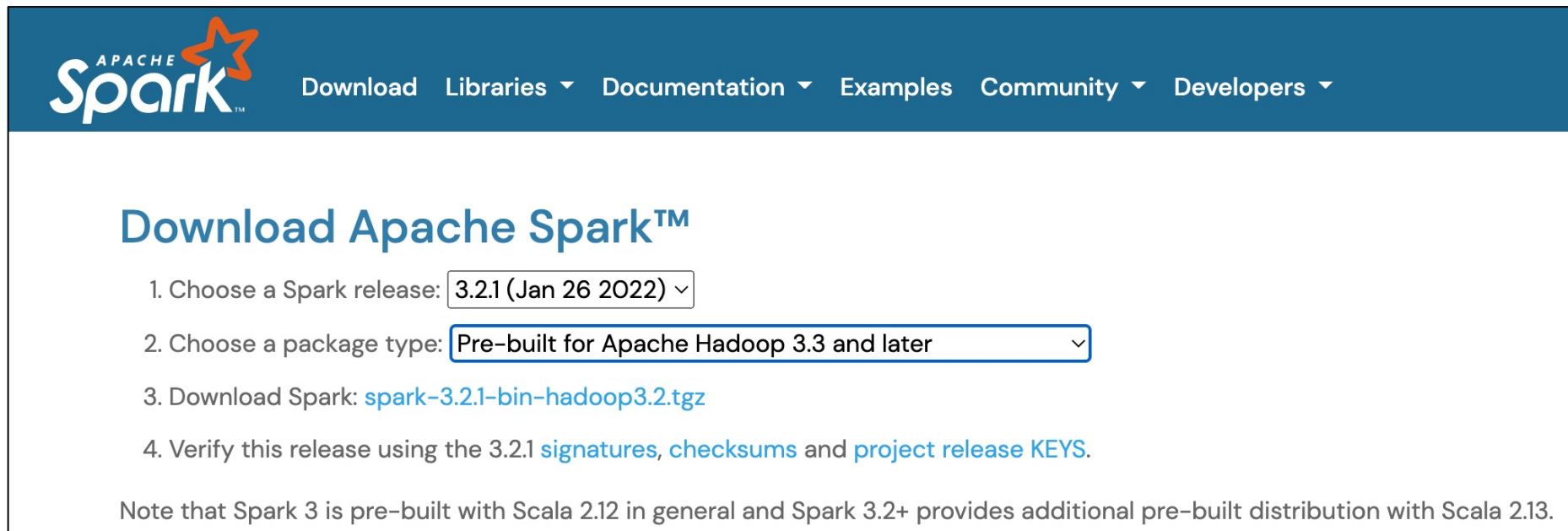
<http://jupyter.org/install.html>



<https://www.continuum.io/downloads>

Spark *standalone* en local

2. Descargar la última versión de Apache Spark disponible.



The screenshot shows the Apache Spark download page. At the top is a dark blue header with the Apache Spark logo on the left and navigation links: Download, Libraries, Documentation, Examples, Community, and Developers. Below the header, the main heading is "Download Apache Spark™". There are four numbered steps: 1. Choose a Spark release: 3.2.1 (Jan 26 2022) with a dropdown arrow. 2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later with a dropdown arrow. 3. Download Spark: spark-3.2.1-bin-hadoop3.2.tgz. 4. Verify this release using the 3.2.1 signatures, checksums and project release KEYS. At the bottom, a note states: "Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13."

APACHE
Spark™

Download Libraries ▾ Documentation ▾ Examples Community ▾ Developers ▾

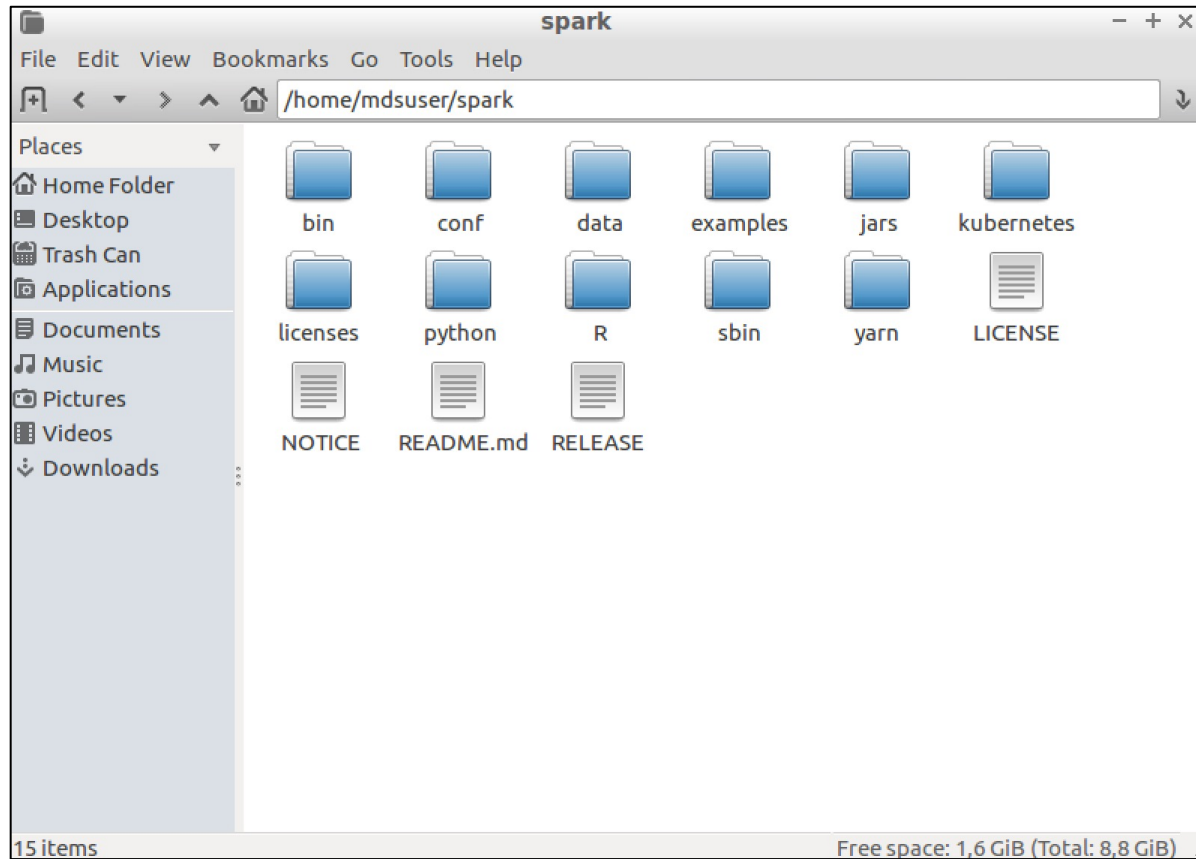
Download Apache Spark™

1. Choose a Spark release: 3.2.1 (Jan 26 2022) ▾
2. Choose a package type: Pre-built for Apache Hadoop 3.3 and later ▾
3. Download Spark: [spark-3.2.1-bin-hadoop3.2.tgz](#)
4. Verify this release using the 3.2.1 [signatures](#), [checksums](#) and [project release KEYS](#).

Note that Spark 3 is pre-built with Scala 2.12 in general and Spark 3.2+ provides additional pre-built distribution with Scala 2.13.

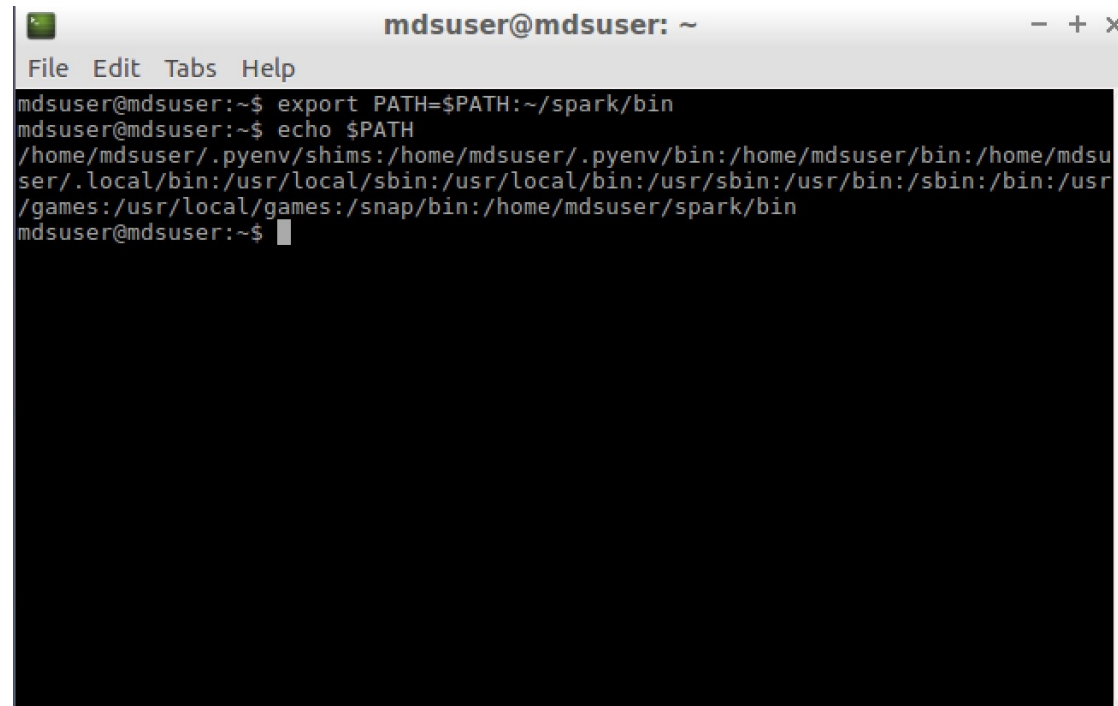
Spark *standalone* en local

3. Descomprimir el fichero en la ruta donde se desee dejar Spark instalado.



Spark *standalone* en local

4. (Opcional) Añadir la ruta de la carpeta “bin” de Spark al PATH del sistema.



```
mdsuser@mdsuser: ~  
File Edit Tabs Help  
mdsuser@mdsuser:~$ export PATH=$PATH:~/spark/bin  
mdsuser@mdsuser:~$ echo $PATH  
/home/mdsuser/.pyenv/shims:/home/mdsuser/.pyenv/bin:/home/mdsuser/bin:/home/mdsuser/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/mdsuser/spark/bin  
mdsuser@mdsuser:~$
```

> export PATH=\$PATH:<ruta a spark/bin>

Ejecución de Spark (python)

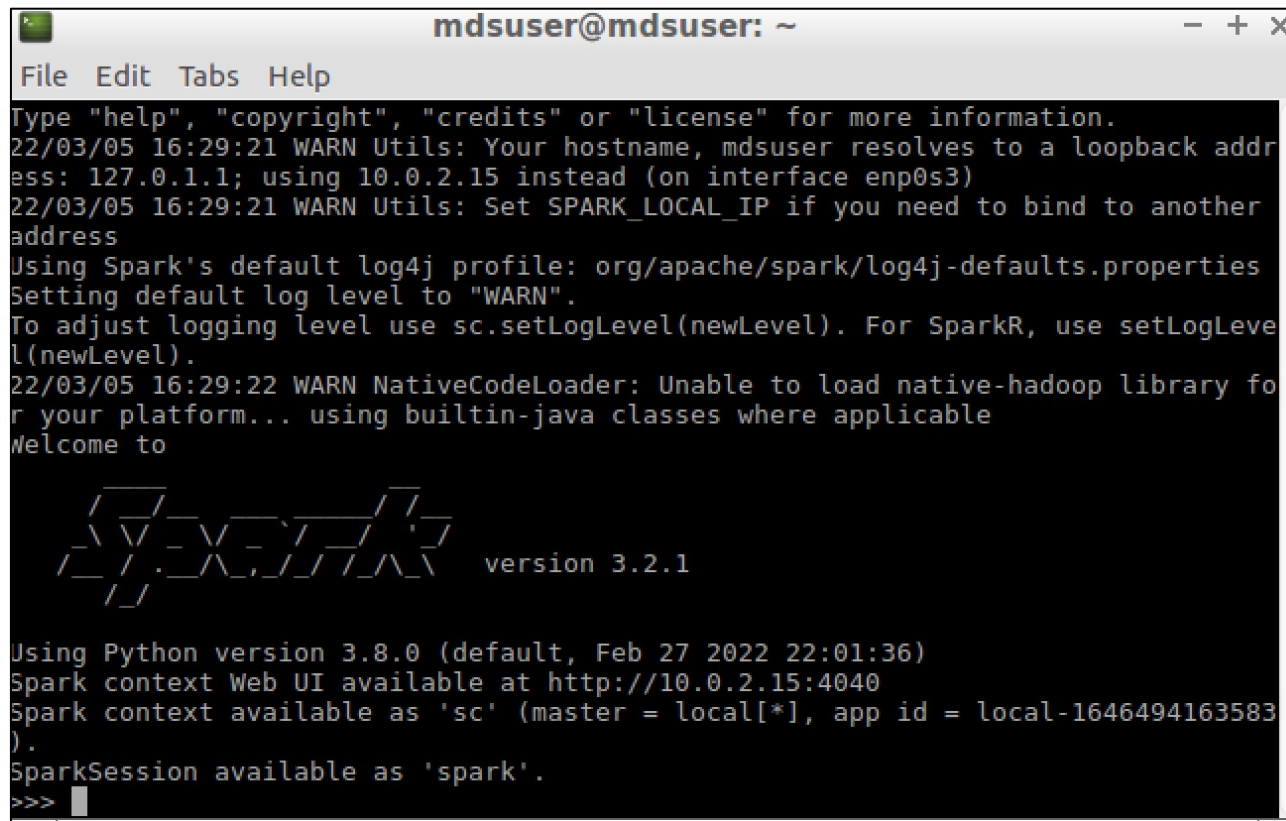
Modos de ejecución de Spark

- Spark ofrece tres formas distintas de ejecución (para python).
- **Spark como kernel:**
 - Ejecución de un kernel de python automáticamente conectado a Spark.
 - Accesible desde consola o Jupyter.
- **Spark como servicio:**
 - Conexión “manual” a un cluster existente de Spark desde código.
 - Uso de Spark equivalente al que haríamos de una base de datos.
- **Spark como job:**
 - Desarrollo de scripts no interactivos sin necesidad de apertura de conexiones.
 - Ejecución completamente desatendida.

Spark como kernel

Consola interactiva

> pyspark

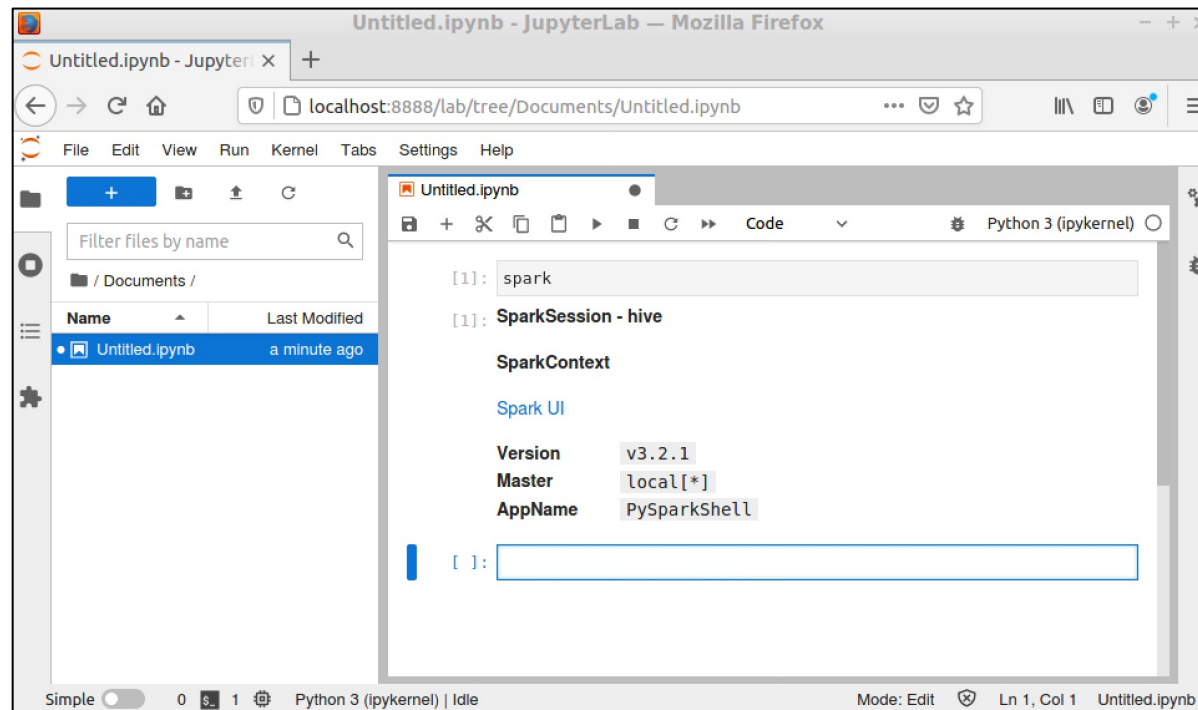


```
mdsuser@mdsuser: ~  
File Edit Tabs Help  
Type "help", "copyright", "credits" or "license" for more information.  
22/03/05 16:29:21 WARN Utils: Your hostname, mdsuser resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)  
22/03/05 16:29:21 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address  
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).  
22/03/05 16:29:22 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
Welcome to  
  
██████████ version 3.2.1  
  
Using Python version 3.8.0 (default, Feb 27 2022 22:01:36)  
Spark context Web UI available at http://10.0.2.15:4040  
Spark context available as 'sc' (master = local[*], app id = local-1646494163583).  
SparkSession available as 'spark'.  
>>>
```

Spark como kernel

Jupyter

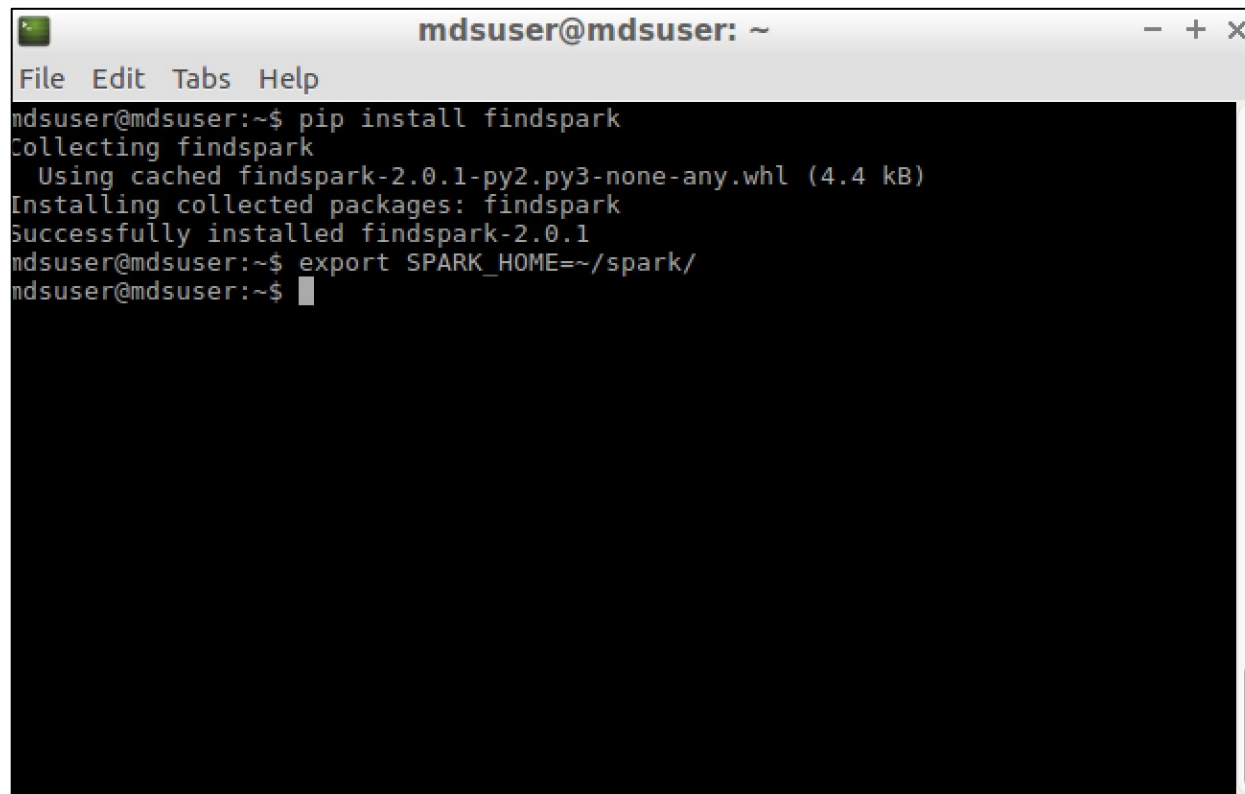
- > export PYSPARK_DRIVER_PYTHON=jupyter
- > export PYSPARK_DRIVER_PYTHON_OPTS=lab
- > pyspark



Spark como servicio

Preparación del entorno

- > pip install findspark
- > export SPARK_HOME=<ruta a spark>



```
mdsuser@mdsuser: ~  
File Edit Tabs Help  
mdsuser@mdsuser:~$ pip install findspark  
Collecting findspark  
  Using cached findspark-2.0.1-py2.py3-none-any.whl (4.4 kB)  
Installing collected packages: findspark  
Successfully installed findspark-2.0.1  
mdsuser@mdsuser:~$ export SPARK_HOME=~/spark/  
mdsuser@mdsuser:~$
```

Spark como servicio

Conexión a Spark

```
# Import findspark
import findspark

# Spark initialization
findspark.init()

# Import required Spark classes
from pyspark import SparkContext, SparkConf

# Configuration properties
conf = SparkConf().setAppName('<application_name>')
conf = conf.setMaster('local[*]')
sc = SparkContext.getOrCreate(conf = conf)

# -----
# Application code
# -----

# Close Spark context
sc.stop()
```

Spark como job

Preparación del código del proceso

```
# Import required Spark classes
from pyspark import SparkContext, SparkConf

# Configuration properties
conf = SparkConf().setAppName('<application_name>')
conf = conf.setMaster('local[*]')
sc = SparkContext.getOrCreate(conf = conf)

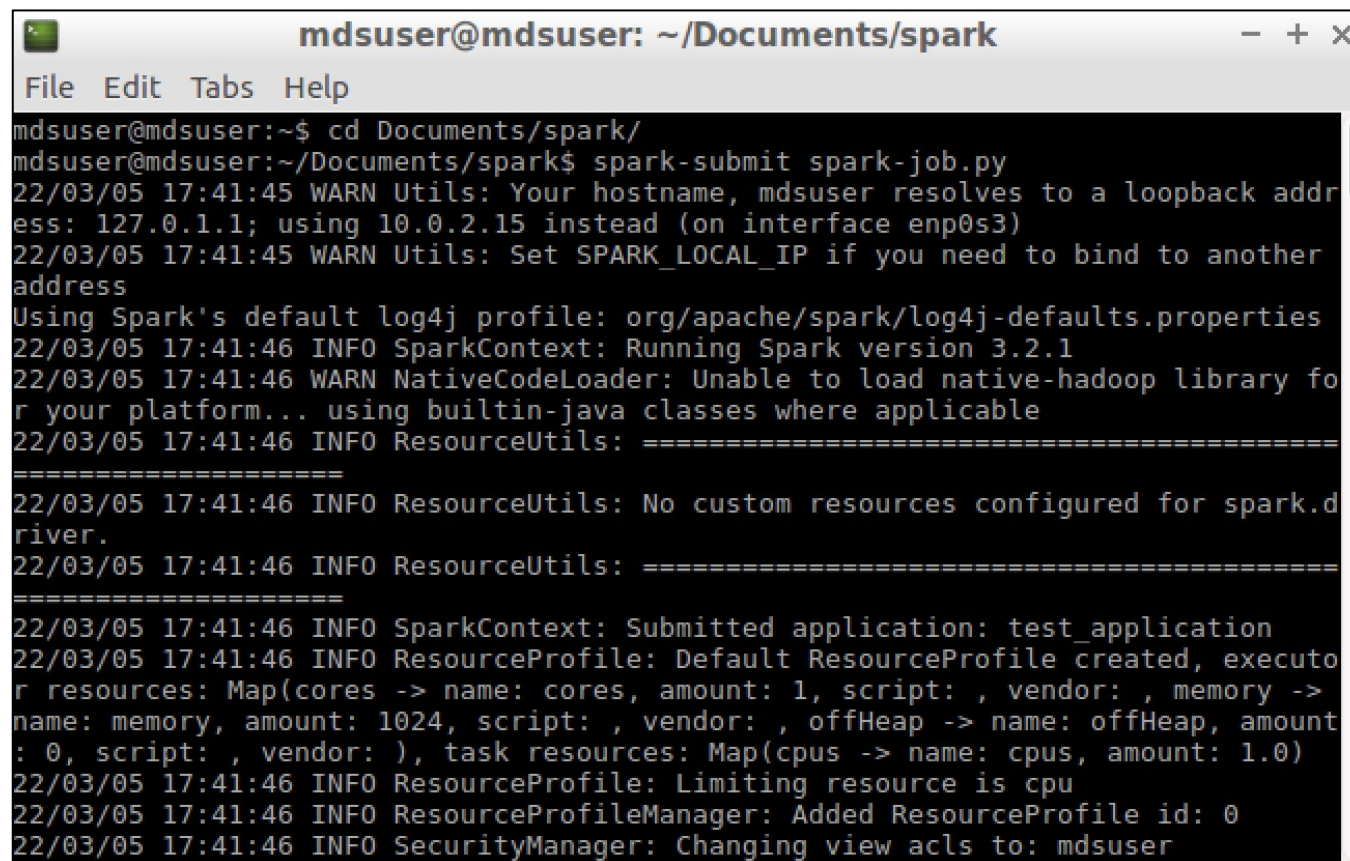
# -----
# Application code
# -----

# Close Spark context
sc.stop()
```

Spark como servicio

Ejecución de jobs desatendidos

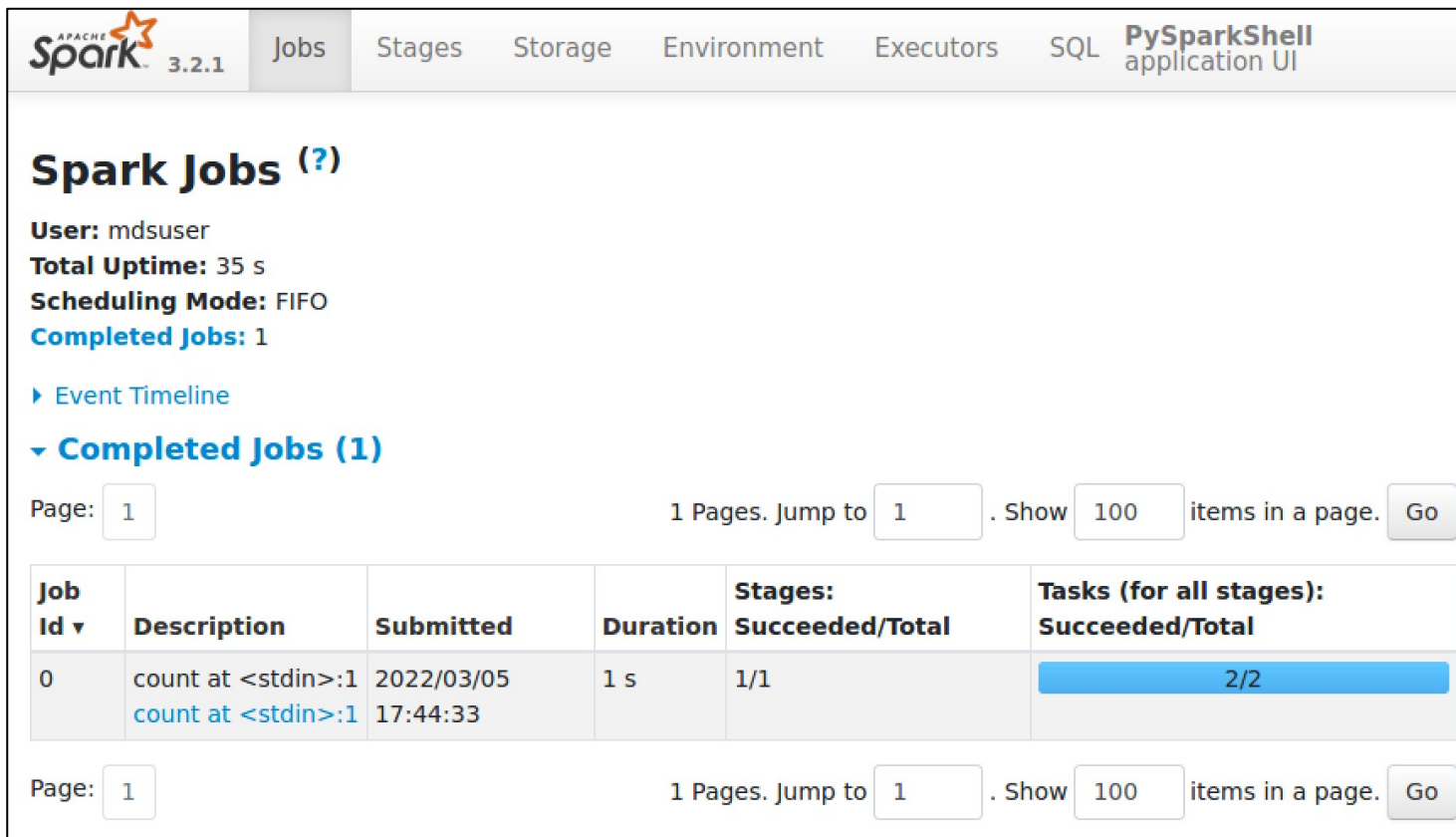
> spark-submit <script>.py

A terminal window titled 'mdsuser@mdsuser: ~/Documents/spark' with a menu bar (File, Edit, Tabs, Help). The terminal shows the execution of 'spark-submit spark-job.py'. It displays various log messages including warnings about hostname resolution, Spark version 3.2.1, and resource configuration details for a submitted application named 'test_application'.

```
mdsuser@mdsuser:~$ cd Documents/spark/  
mdsuser@mdsuser:~/Documents/spark$ spark-submit spark-job.py  
22/03/05 17:41:45 WARN Utils: Your hostname, mdsuser resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)  
22/03/05 17:41:45 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address  
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties  
22/03/05 17:41:46 INFO SparkContext: Running Spark version 3.2.1  
22/03/05 17:41:46 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
22/03/05 17:41:46 INFO ResourceUtils: =====  
=====  
22/03/05 17:41:46 INFO ResourceUtils: No custom resources configured for spark.driver.  
22/03/05 17:41:46 INFO ResourceUtils: =====  
=====  
22/03/05 17:41:46 INFO SparkContext: Submitted application: test_application  
22/03/05 17:41:46 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)  
22/03/05 17:41:46 INFO ResourceProfile: Limiting resource is cpu  
22/03/05 17:41:46 INFO ResourceProfileManager: Added ResourceProfile id: 0  
22/03/05 17:41:46 INFO SecurityManager: Changing view acls to: mdsuser
```


Monitorización de Spark

<http://localhost:4040>



APACHE Spark 3.2.1

Jobs Stages Storage Environment Executors SQL PySparkShell application UI

Spark Jobs (?)

User: mdsuser
Total Uptime: 35 s
Scheduling Mode: FIFO
Completed Jobs: 1

▶ Event Timeline

▼ Completed Jobs (1)

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

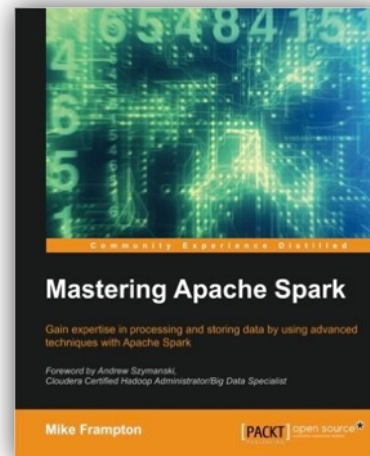
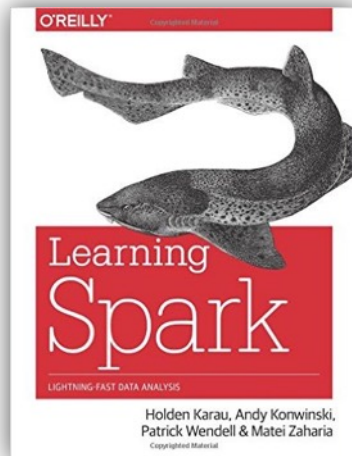
Job Id ▼	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	count at <stdin>:1 count at <stdin>:1	2022/03/05 17:44:33	1 s	1/1	2/2

Page: 1 1 Pages. Jump to 1 . Show 100 items in a page. Go

Referencias

Referencias

- Documentación oficial:
 - <https://spark.apache.org>
- Tutoriales online:
 - https://www.tutorialspoint.com/apache_spark/
- Libros:





Afi Escuela

© 2022 Afi Escuela. Todos los derechos reservados.