

1 Instalación de spark paso a paso en linux e integración con jupyter lab	1
1.1 Instalación de Spark	1
1.2 Iniciando Apache Spark	3
1.3 Detener Spark	6

1 Instalación de spark paso a paso en linux e integración con jupyter lab

<https://www.exploradata.com/instalando-spark-en-ubuntu-20-04/>

Guía sobre cómo instalar y ejecutar Apache spark en una máquina Linux.

Apache Spark funcionan en sistemas windows y UNIX (linux, mac)

1.1 Instalación de Spark

1. Instalar scala

En primer lugar debemos instalar Scala, ya que Spark está implementado en dicho lenguaje de programación y lo necesita para ejecutarse. Podemos hacerlo desde la siguiente url:

```
sudo apt install scala
```

para verificar que se ha instalado correctamente, ejecutamos la instrucción:

```
scala -version
```

Si todo está bien nos devolverá lo siguiente.

```
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
```

2- Descargar spark

```
wget https://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
```

3- Extraemos su contenido y lo movemos a la carpeta /opt y renombramos la carpeta llamándola spark.

```
tar -xvzf spark-3.1.1-bin-hadoop3.2.tgz sudo  
mv spark-3.1.1-bin-hadoop3.2 /opt/spark
```

4- creamos las variables de entorno para facilitar la ejecución de spark

<https://www.zeppelinlinux.es/descripcion-y-uso-de-los-archivos-bashrc-y-etc-bashrc/>

se puede modificar el fichero bashrc de todos los usuarios (/etc/bashrc) o el del usuario actual (~/.bashrc)

```
nano ~/.bashrc
```

Nota: En distribuciones [Debian](#) y basadas en [Debian](#), en /etc encontraremos el archivo bash.bashrc. En otras distribuciones, en /etc encontraremos el archivo con el archivo bashrc.

nos situamos al final del fichero bashrc y añadimos las siguientes líneas

```
export SPARK_HOME=/opt/spark export  
PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Para que se apliquen las modificaciones hay que recargar el archivo ~/.bashrc , ejecutaremos el siguiente comando

```
source ~/.bashrc
```

1.2 Iniciando Apache Spark

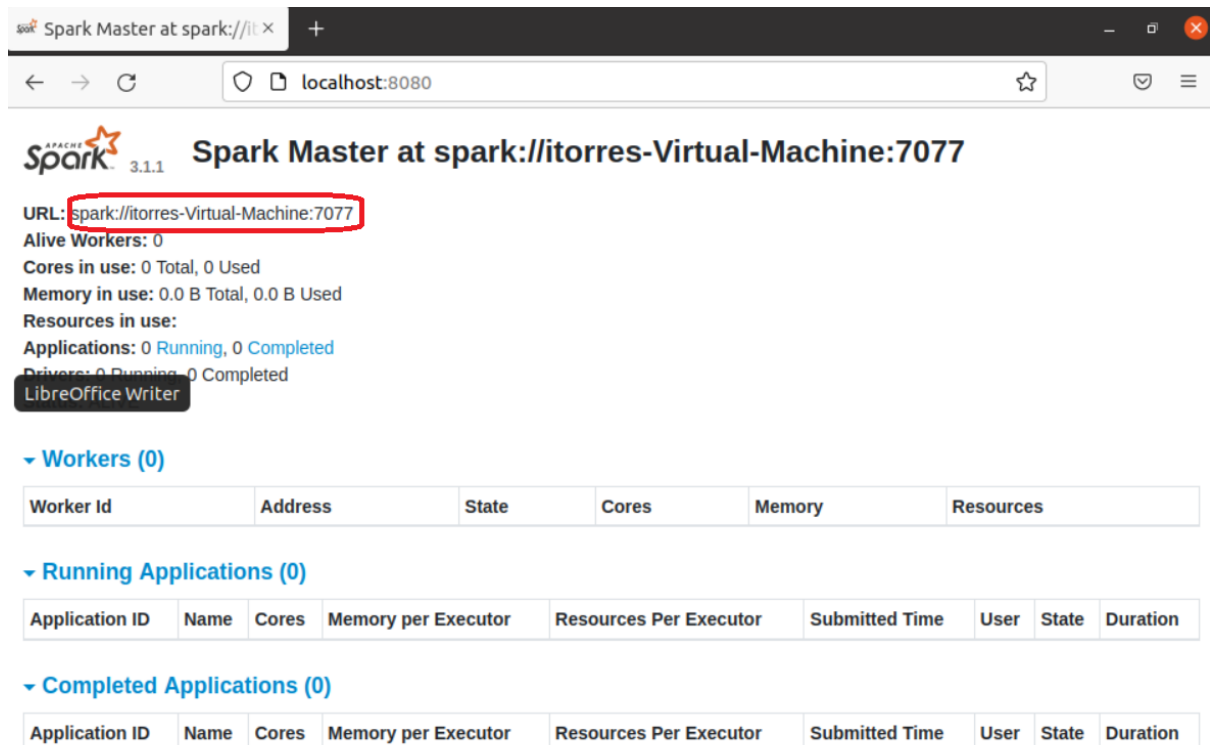
1- Para empezar iniciamos el Master server, si realizaste bien los pasos anteriores no necesitaras el path completo y funcionaria la siguiente instrucción

```
start-master.sh
```

En caso que no encuentre el fichero se puede ejecutar con el path completo `./sbin/start-master.sh`

Una vez iniciado, el maestro imprimirá una URL `spark://HOST:PORT`, que puedes usar para conectar trabajadores a él, o pasar como argumento "maestro" a `SparkContext`. También puedes encontrar esta URL en la interfaz web del maestro, que es `http://localhost:8080` por defecto.

Arrancamos el navegador para verificar que se ha arrancado spark e introducimos la dirección `localhost:8080`



Spark Master at spark://itorres-Virtual-Machine:7077

URL: `spark://itorres-Virtual-Machine:7077`

Alive Workers: 0

Cores in use: 0 Total, 0 Used

Memory in use: 0.0 B Total, 0.0 B Used

Resources in use:

Applications: 0 Running, 0 Completed

Drivers: 0 Running, 0 Completed

LibreOffice Writer

▼ Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

▼ Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

2- Ahora iniciaremos el Spark Worker Process.

Del mismo modo, puede iniciar uno o más trabajadores y conectarlos al maestro a través de:

```
start-slave.sh
```

```
start-worker.sh spark://itorres-Virtual-Machine:7077
```

o

```
start-worker.sh spark://localhost:7077
```

verificamos que se ha iniciado el worker y se ha añadido al cluster de spark (http://localhost:8080 por defecto). Deberías ver el nuevo nodo listado allí, junto con su número de CPUs y memoria (menos un gigabyte que queda para el SO).

Finalmente, las siguientes opciones de configuración pueden ser pasadas al maestro y al trabajador:

- -h HOST, --host HOST: Nombre de host para escuchar
- -p PORT, --port PORT : Puerto de escucha del servicio (por defecto: 7077 para el maestro, aleatorio para el trabajador)
- --webui-port PORT Puerto para la interfaz web (por defecto: 8080 para el maestro, 8081 para el trabajador)
- -c CORES, --cores CORES: Total de núcleos de CPU que las aplicaciones Spark pueden utilizar en la máquina (por defecto: todos los disponibles); sólo en worker
- -m MEM, --memory MEM: Cantidad total de memoria que las aplicaciones Spark pueden utilizar en la máquina, en un formato como 1000M o 2G (por defecto: la RAM total de su máquina menos 1 GiB); sólo en worker
- -d DIR, --work-dir DIR : Directorio que se utilizará para el espacio de memoria y los registros de salida de los trabajos (por defecto: SPARK_HOME/work); sólo en worker
- --properties-file FILE : Ruta a un archivo de propiedades de Spark para cargar (por defecto: conf/spark-defaults.conf)


```
$ pyspark
```

```
Welcome to
```

```
  ____
 /  __ \   _ __   ___
 \  __ <  / _ `\/ __ \
  \  __ < /  __/ /  __/
   \  __ < \___/ \___/   version 3.1.1
    \  __ <
     \  __ <
```

```
Using Python version 3.9.4 (default, Apr  4 2021 19:38:44)
Spark context Web UI available at http://172.16.200.3:4041
Spark context available as 'sc' (master = local[*], app id = local-1621421056899).
SparkSession available as 'spark'.
```

- para salir del shell de spark basta con introducir el comando `exit()`

1.3 Detener Spark

Primero detenemos los workers

```
stop-worker.sh
```

una vez detenido el worker, detenemos el master

```
stop-master.sh
```

Integracion de spark -Scala con jupyter lab

<https://programmerclick.com/article/86911715560/>

<https://www.bmc.com/blogs/jupyter-notebooks-apache-spark/>

Jupyter proporciona soporte para muchos lenguajes de programación distintos de Python, como R, Go, Scala, etc., pero todos deben ser instalados manualmente por el usuario. Aquí está la instalación del kernel scala y el kernel spark

1- verificar los kernels existentes

```
$ jupyter kernelspec list
```

```
root@master:~# jupyter kernelspec list
Available kernels:
  python3      /usr/local/anacond/share/jupyter/kernels/python3
  python2      /usr/local/share/jupyter/kernels/python2
```

En este caso el kernel de scala no está instalado para jupyter

2 Instalamos **scala** para jupyter

Este paso se realiza si scala no está ya instalado para jupyter

```
tar -xvf jupyter-scala_2.10.5-0.2.0-SNAPSHOT.tar.xz -C /usr/local/
bash /usr/local/jupyter-scala_2.10.5-0.2.0-SNAPSHOT/bin/jupyter-scala
```

volvemos a verificar los kernels disponibles (ver punto anterior) y nos tiene que aparecer que scala ya está disponible

Integracion de spark -python con jupyter lab

El único requisito para obtener la referencia de Jupyter Notebook PySpark es añadir las siguientes variables de entorno en tu archivo .bashrc o .zshrc, que apuntan PySpark a Jupyter.

Variables de entorno para jupyter notebook (

```
export PYSPARK_DRIVER_PYTHON='jupyter'
export PYSPARK_DRIVER_PYTHON_OPTS='notebook --no-browser --port=8889'
```

Variables de entorno para jupyter lab

```
export PYSPARK_DRIVER_PYTHON='jupyter'
export PYSPARK_DRIVER_PYTHON_OPTS='lab --no-browser --allow-root --port=8889'
```

Las variables de entorno creadas, incluyendo las de spark serían:

```
export SPARK_HOME=/opt/spark export
PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export PYSPARK_DRIVER_PYTHON='jupyter'
export PYSPARK_DRIVER_PYTHON_OPTS='lab --no-browser --allow-root --port=889'
```

Si todo ha ido bien, al ejecutar pySpark se abriría jupyter lab y ya podríamos empezar a trabajar con spark

```
$ pyspark
```