

*Estos Apuntes se basan un laboratorio (Que iré escalando) de Apache-Spark y Docker y controlando por PySpark , Jupyter-Notebook . Probando diferentes configuraciones de Pytho-Venv , Docker ,en Cluster etc. En esta ocasión usare un portátil Host el cual ejecuta sus nodos Master y Worker en Host ,es decir modo Standalone .*

Tenemos un entorno virtual **Python3.6-Venv** , ya creado anteriormente como un ecosistema con **Apache-Spark , PySpark , Jupyter** .

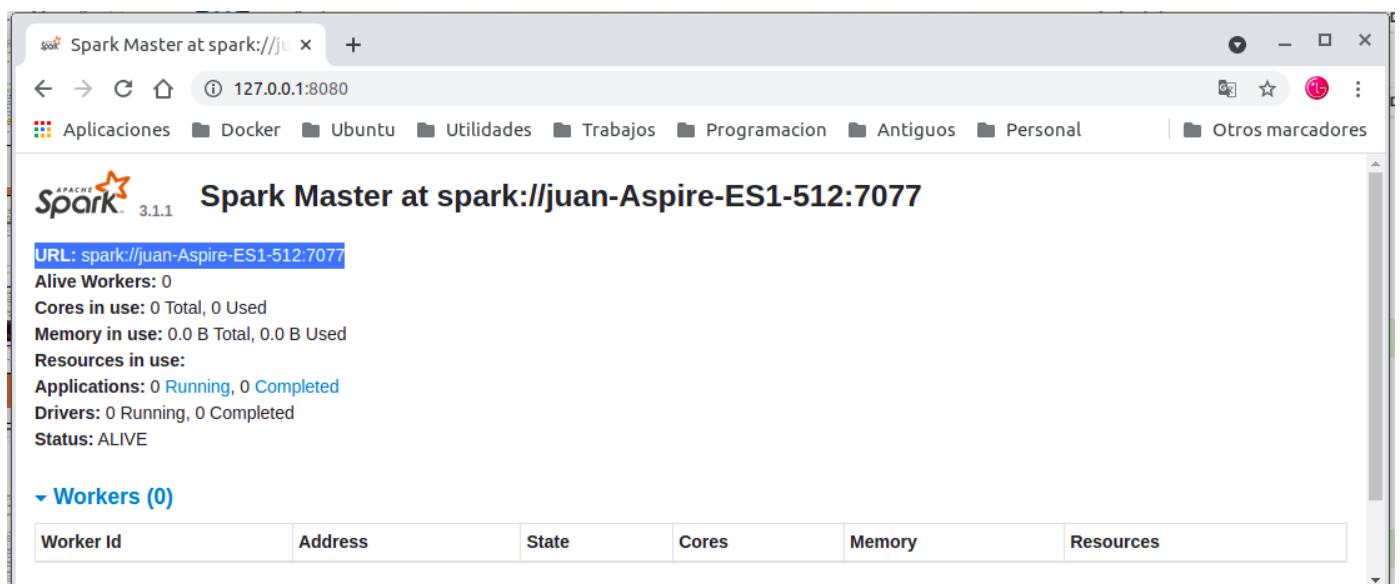
*Levantamos entorno virtual **venv** (Activate)-(Deactivate)*

`root@juan-Aspire-ES1-512:/# source my_pyspark/bin/activate`  
 (my\_pyspark) `root@juan-Aspire-ES1-512:/# deactivate`

Arrancamos **Spark** en modo **Standalone** es decir se ejecuta sus , **nodos Master** y (aunque el **Worker** se encuentre un contenedor **Docker** ) esta en el Host

*Arrancamos Spark en modo Standalone es decir se ejecuta sus nodos Master y Worker en Host.*

(my\_pyspark) `root@juan-Aspire-ES1-512:/# ./spark/sbin/start-master.sh -h 192.168.1.36`  
 starting org.apache.spark.deploy.master.Master, logging to /my\_pyspark/spark/logs/spark-root-  
 org.apache.spark.deploy.master.Master-1-juan-Aspire-ES1-512.out



<http://127.0.0.1:8080/>

Creamos el **contenedor Docker** tenemos diferentes formas de abordar el tema , usar una **imagen del repositorio oficial de Docker** , usar o crear un **Dockerfile** o hacerlo manualmente nosotros usaremos esta ultima.

#### Crear un contenedor docker

```
root@juan-Aspire-ES1-512:/# docker run -it --name spark_docker -m 1024M --cpus 1 ubuntu
root@c59e3ef5557f:/#
```

#### Deshabilitar el modo interactivo ¿para que no pregunte?

```
root@c59e3ef5557f:/# export DEBIAN_FRONTEND=noninteractive
```

#### Instalar las dependencias(Java, Python y Nano)

```
root@c59e3ef5557f:/# apt update && apt install -y openjdk-8-jdk python nano
```

#### Descarga Spark:(descomprimir el fichero.tgz)-(crear)Mover a la carpeta /spark

```
root@juan-Aspire-ES1-512:/# curl -O https://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
root@juan-Aspire-ES1-512:/# tar xvf spark-3.1.1-bin-hadoop3.2.tgz
```

#### Copiar el paquete entero de Apache Spark <ID\_CONTENEDOR> /opt del cotenedor

```
root@juan-Aspire-ES1-512:/# docker cp spark-3.1.1-bin-hadoop3.2 c59e3ef5557f:/opt
```

#### Renombra Carpeta

```
root@c59e3ef5557f:/opt# mv spark-3.1.1-bin-hadoop3.2 spark
```

#### Crea un enlace simbólico

```
root@c59e3ef5557f:/opt# ln -s /opt/spark/sbin
```

#### Establecer entorno de Spark(Abra su archivo de configuración de **bashrc**)Activa los cambios

```
root@1bf0b87d3b85:/# nano ~/.bashrc
export SPARK_HOME=/opt/spark
export PATH=$SPARK_HOME/bin:$PATH
root@1bf0b87d3b85:/# source ~/.bashrc
```

#### Arrancamos el worker situado en Contenedor-Docker

```
root@1bf0b87d3b85:/opt# ./spark/sbin/start-worker.sh spark://192.168.1.36:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark--
org.apache.spark.deploy.worker.Worker-1-1bf0b87d3b85.out
```

Ahora veremos en **Spark-Master** el **Worker activo** con su **ID** , **Address 172.17.0.2:46473** este puerto habrá que asignarle un puerto fijo y vemos los recursos usados por el **Worker** .

URL: spark://192.168.1.36:7077  
 Alive Workers: 1  
 Cores in use: 1 Total, 0 Used  
 Memory in use: 1024.0 MiB Total, 0.0 B Used  
 Resources in use:  
 Applications: 0 Running, 0 Completed  
 Drivers: 0 Running, 0 Completed  
 Status: ALIVE

## Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210706122551-172.17.0.2-46473	172.17.0.2:46473	ALIVE	1 (0 Used)	1024.0 MiB (0.0 B Used)	

## 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
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

<http://127.0.0.1:8080/>

## Arrancamos un Jupyter-Notebook

(my\_pyspark) root@juan-Aspire-ES1-512:/# jupyter notebook --allow-root

```

(my_pyspark) root@juan-Aspire-ES1-512:/# jupyter notebook --allow-root
[I 10:03:17.331 NotebookApp] Serving notebooks from local directory: /
[I 10:03:17.331 NotebookApp] Jupyter Notebook 6.4.0 is running at:
[I 10:03:17.331 NotebookApp] http://localhost:8888/?token=17cafcaaad4e02e72275b1332a26fed7334588b6df7e3c92
[I 10:03:17.331 NotebookApp] or http://127.0.0.1:8888/?token=17cafcaaad4e02e72275b1332a26fed7334588b6df7e3c92
[I 10:03:17.331 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 10:03:17.388 NotebookApp]

To access the notebook, open this file in a browser:
  
```

<http://localhost:8888/?token=17cafcaaad4e02e72275b1332a26fed7334588b6df7e3c92>

Recomiendo cortar-pegar en nuestro navegador en ciertas ocasiones e tenido algún problema

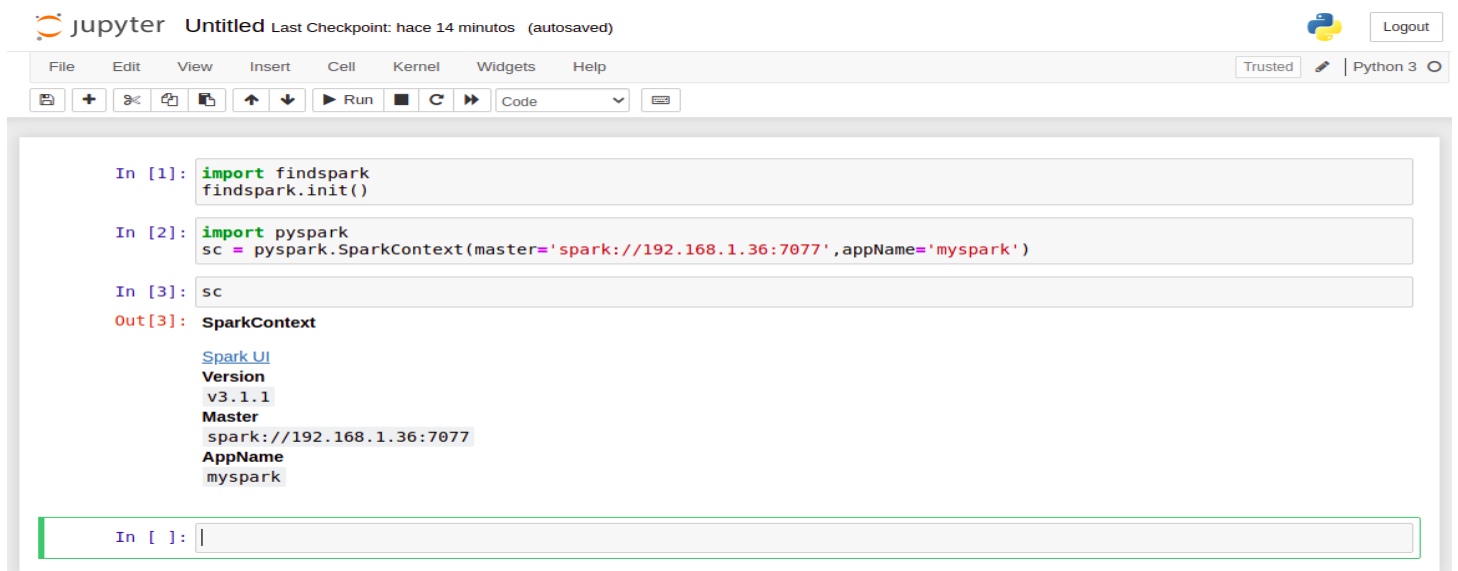
## Crear una aplicación Spark y comenzar: pegando el siguiente script en una celda de Jupyter

```
import findspark
```

```
findspark.init()
```

```
import pyspark
```

```
sc = pyspark.SparkContext(master='spark://192.168.1.36:7077',appName='myspark')
```



The screenshot shows the Jupyter Notebook interface with the following content:

- Header:** jupyter Untitled Last Checkpoint: hace 14 minutos (autosaved) Logout
- Menu:** File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
- Toolbar:** Run, Stop, Refresh, Code, etc.
- Code Cells:**
  - In [1]: `import findspark`  
`findspark.init()`
  - In [2]: `import pyspark`  
`sc = pyspark.SparkContext(master='spark://192.168.1.36:7077',appName='myspark')`
  - In [3]: `sc`
- Output:**
  - Out[3]: `SparkContext`
  - `Spark UI`
  - `Version`  
`v3.1.1`
  - `Master`  
`spark://192.168.1.36:7077`
  - `AppName`  
`myspark`
- Input Field:** In [ ]: |

ID: worker-20210708075039-172.17.0.2-37689  
 Master URL: spark://192.168.1.36:7077  
 Cores: 1 (1 Used)  
 Memory: 1024.0 MiB (1024.0 MiB Used)  
 Resources:

[Back to Master](#)

## Running Executors (1)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
0	RUNNING	1	1024.0 MiB		ID: app-20210708102751-0000 Name: myspark User: root	<a href="#">stdout stderr</a>

En el Spark Worker situado en el Contenedor-Docker vemos la ejecución de nuestro script , por-tanto esta (Running) , ID: worker-20210708075039-172.17.0.2-37689

Consola PySpark , Web UI available at <http://192.168.1.36:4040>



192.168.1.36:4040 como vemos y arancado la Consola PySpark en nuestro Jupyter hemos dado la orden de (**import** pyspark) , arrancamos (SparkContext) , perdón tenia que usar (SparkSesaion) ,para probar Vale .

**Recapitulando :** activamos el entorno virtual Python3.6-Venv , activamos el (start-master.sh) en el Host . En nuestro contenedor Docker activamos el Worker situado en el Host (start-worker.sh) .

En nuestro navegador Wed <http://127.0.0.1:8080/> el cual nos dara la **URL: spark://192.168.1.36:7077** y información de nuestros worker y recurso de nuestro entorno .

Arrancaremos nuestra IDE favorita en mi caso Jupyter-Notebook , import findspark y pyspark y ejeturamos un pequeño script para confirmar la interacciona con Spark a traves de API-PySpark lo cual podemos comprobar en Spark context Web UI available at <http://192.168.1.36:4040>

Con todo esto usar Docker y Python-Venv se trata de aislar todos estos procesos de la maquinas Host o Server se notara mucho cuando escalemos el proceso con la cleacion de cluster