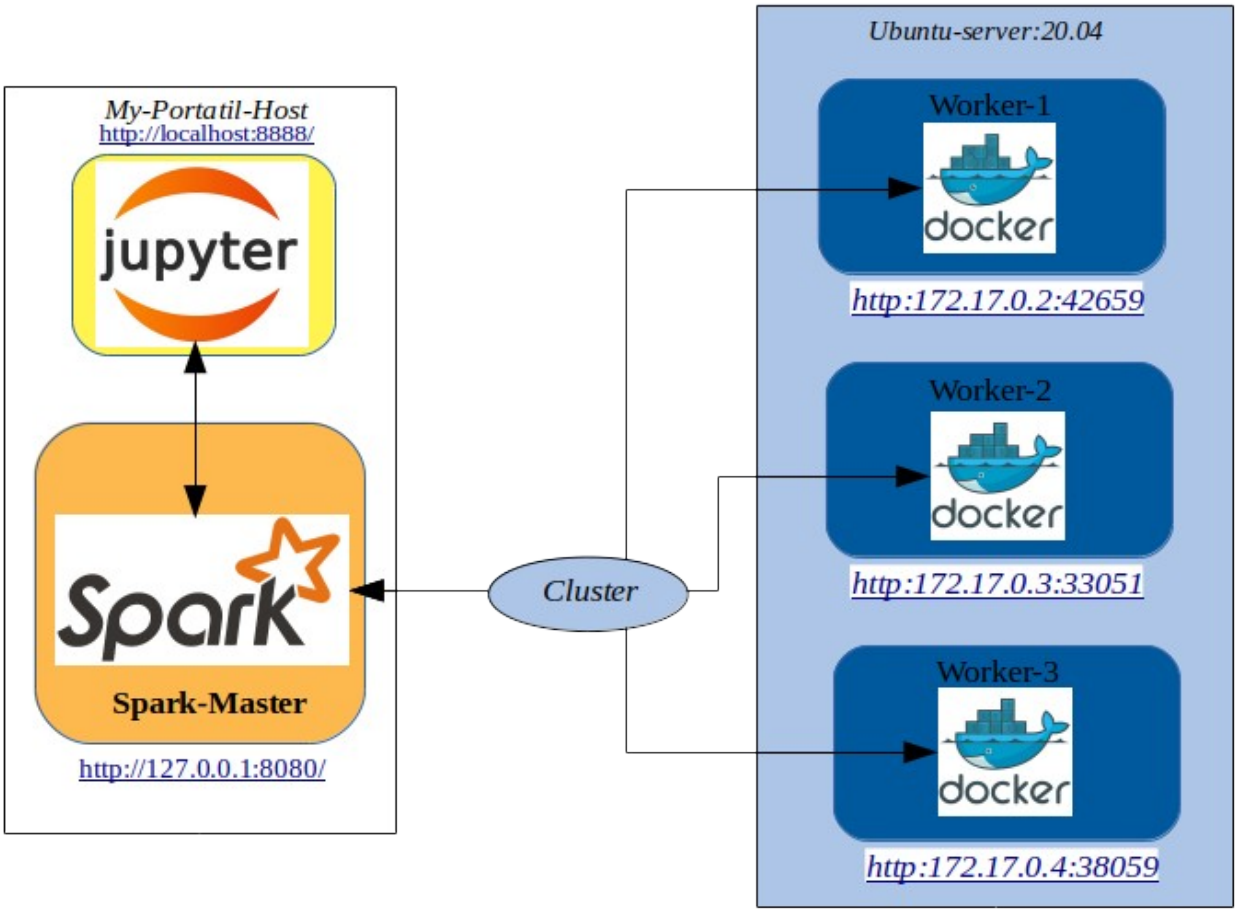


Configuración de Master en Python3.6-Venv , tres Worker en Contenedores Docker y con Jupyter

Doker-Spark-Jupyter-V:02

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*** esta la llamare V:02 .
En esta ocasión usare un ***portátil Host*** el cual ejecuta sus nodo ***Master, Jupter, SparkSession*** y un ***Server-Ubuntu:20.04*** en el cual irán alojados los ***Worker1***,***Worker2***,***Worker3*** ,
Tenemos un entorno virtual ***Python3.6-Venv*** , ya creado anteriormente como un ecosistema con ***Apache-Spark*** , ***PySpark*** , ***Jupyter*** .

Necesitamos una Imagen de Docker ya creada para tal fin, las tenemos el el repositorio de Docker o la generamos nosotros , que es lo que aremos : empezamos por ***crear un contenedor Docker*** a partir de una imagen de Docker (***Ubuntu:18.04***) la usaremos como base de nuestro contenedor la cual modificaremos para crear nuestro ***contenedor*** y apartir de este crear ***nuestra imagen*** de la cual crearemos los Workers que necesitemos**.** 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.

*Creacion de una **Imagen Docker** a partir de un Contenedor*

*Crear un contenedor Docker lo llamaremos (**spark_docker)***

root@serve--hp8100:/# 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@serve--hp8100:/# curl -O https://archive.apache.org/dist/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
root@serve--hp8100:/# 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#

*Crear una **Imagen** a partir de un **Contenedor***

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
40c9653c010c ubuntu:18.04 "bash" 24 hours ago Exited (127) 24 hours ago spark_docker

root@serve--hp8100:/# docker commit 40c9653c010c spark_docker:21.07
sha256:5bb39b6744e6d287621bdd42bb3c8cb2d9f65f73c98b3d0b5d47b1e791529d36

root@serve--hp8100:/# docker images
spark_docker 21.07 **5bb39b6744e6** About a minute ago 783MB .

Arracamos Paython-Venv – Spark-Master

Disponemos de un entorno virtual Python3.6-Venv , ya creado anteriormente como un ecosistema con ***Apache-Spark*** , ***PySpark*** , ***Jupyter*** .en el ***portátil Host***

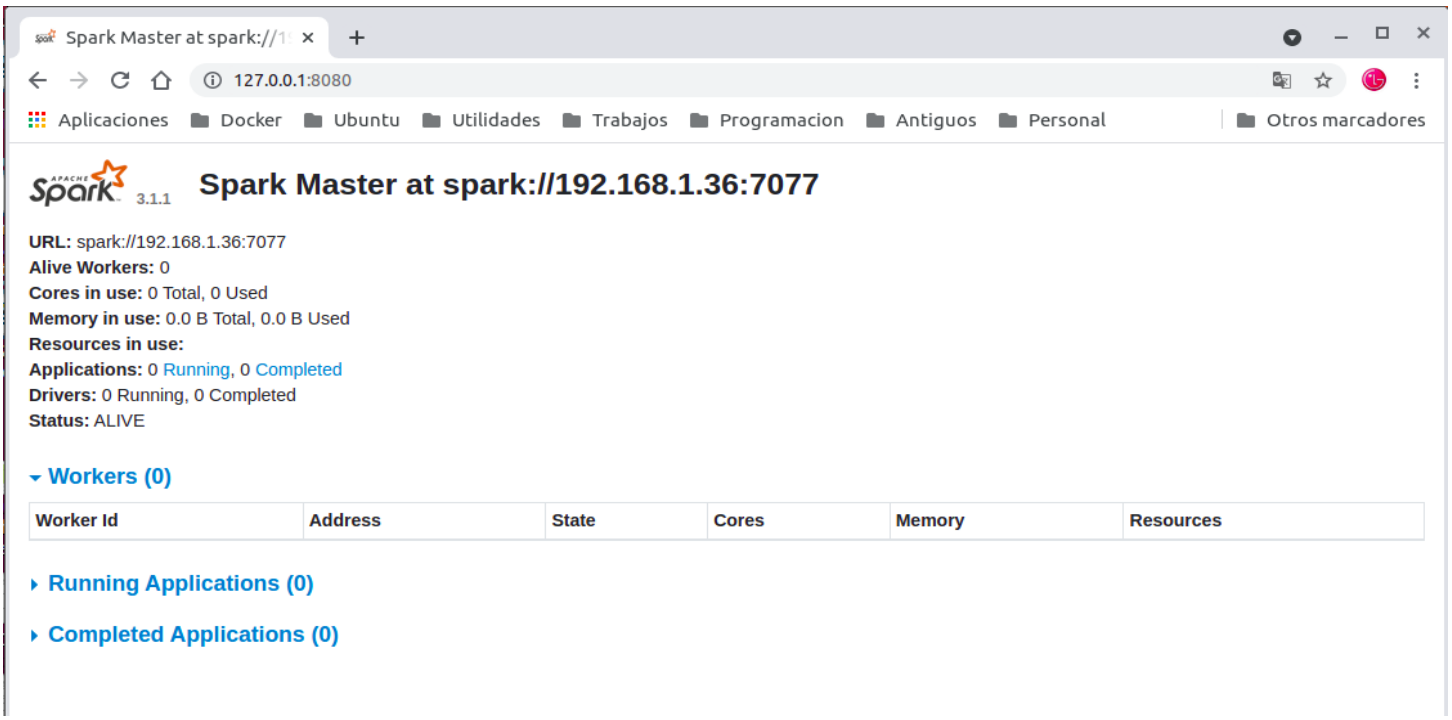
Levantamos entorno virtual ***venv*** *(Activate)-(Deactivate)*

root@juan-Aspire-ES1-512:/# source my_pyspark/bin/activate
(my_pypark) root@juan-Aspire-ES1-512:/# deactivate

Arrancamos Spark ejecuta sus nodos Master my-portatil-Host 192.168.1.36

(my_pyspark) root@juan-Aspire-ES1-512:/# ./spark/sbin/start-master.sh -h 192.168.1.36

*Arranca interface-Wed-**Spar-Master*** *http://127.0.0.1:8080/*



Continuamos con la creación del (Cluster de tres Worker)

Creamos el Contenedor ***Worker-1***

root@serve--hp8100:/# docker run -it --name worker-1 -m 1024M --cpus 1 spark_docker:21.07
root@ef00c9ebbb14:/#

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ef00c9ebbb14	spark_docker:21.07	"bash"	About a minute ago	Up About a minute		worker-1

Arrancamos ***Worker-1***

root@ef00c9ebbb14:/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-***ef00c9ebbb14***.out
root@ef00c9ebbb14:/opt#

*Creamos el Contenedor **Worker-2***

root@serve--hp8100:/# docker run -it --name worker-2 -m 1024M --cpus 1 spark_docker:21.07
root@5cfbc7c97119:/#

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5cfbc7c97119	spark_docker:21.07	"bash"	5 minutes ago	Up 5 minutes		worker-2
ef00c9ebbb14	spark_docker:21.07	"bash"	23 minutes ago	Up 23 minutes		worker-1

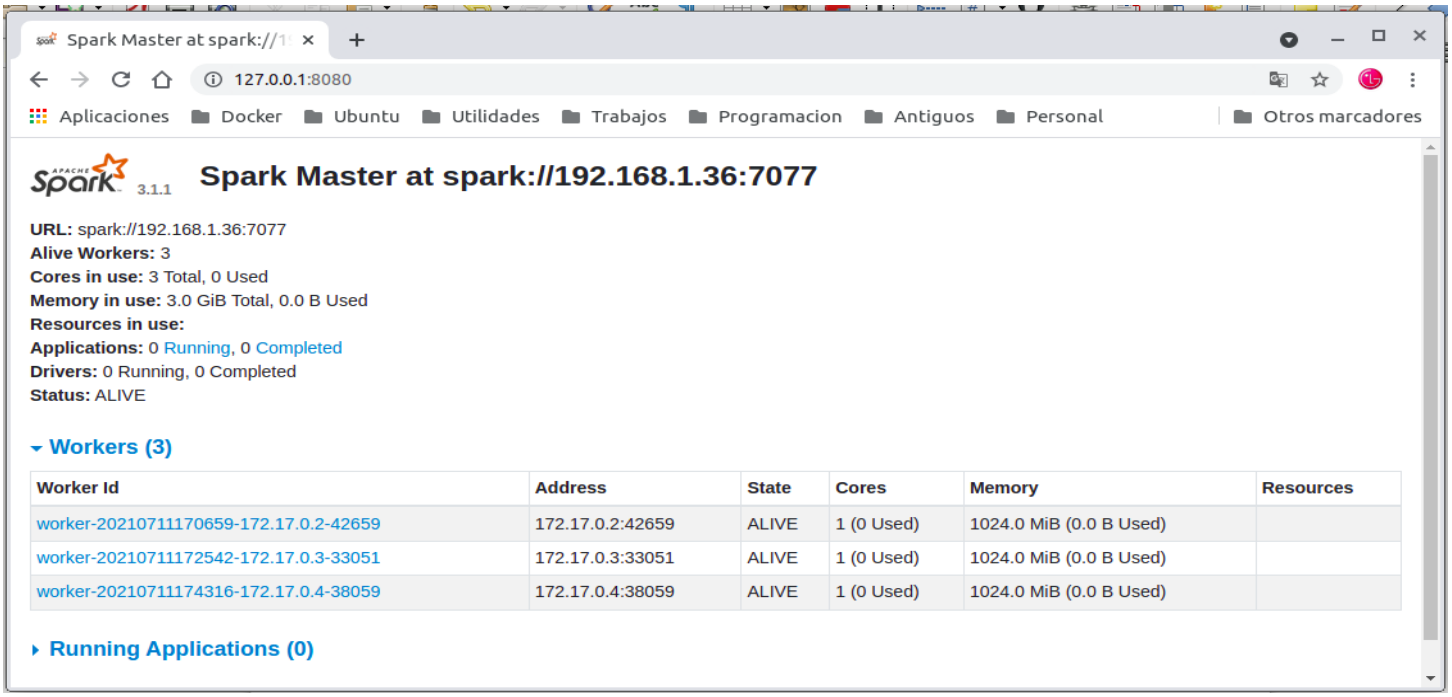
root@5cfbc7c97119:/opt# ./spark/sbin/start-worker.sh spark://192.168.1.36:7077
starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark/logs/spark--
org.apache.spark.deploy.worker.Worker-1-**5cfbc7c97119**.out
root@5cfbc7c97119:/opt#

Creamos el Contenedor Worker-3

root@serve--hp8100:/# docker run -it --name worker-3 -m 1024M --cpus 1 spark_docker:21.07
root@eb42b6da656f:/#

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
eb42b6da656f	spark_docker:21.07	"bash"	About a minute ago	Up About a minute		worker-3
5cfbc7c97119	spark_docker:21.07	"bash"	23 minutes ago	Up 23 minutes		worker-2
ef00c9ebbb14	spark_docker:21.07	"bash"	41 minutes ago	Up 41 minutes		worker-1

root@eb42b6da656f:/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-**eb42b6da656f**.out
root@eb42b6da656f:/opt#



*Arrancamos un **Jupyter-Notebook***

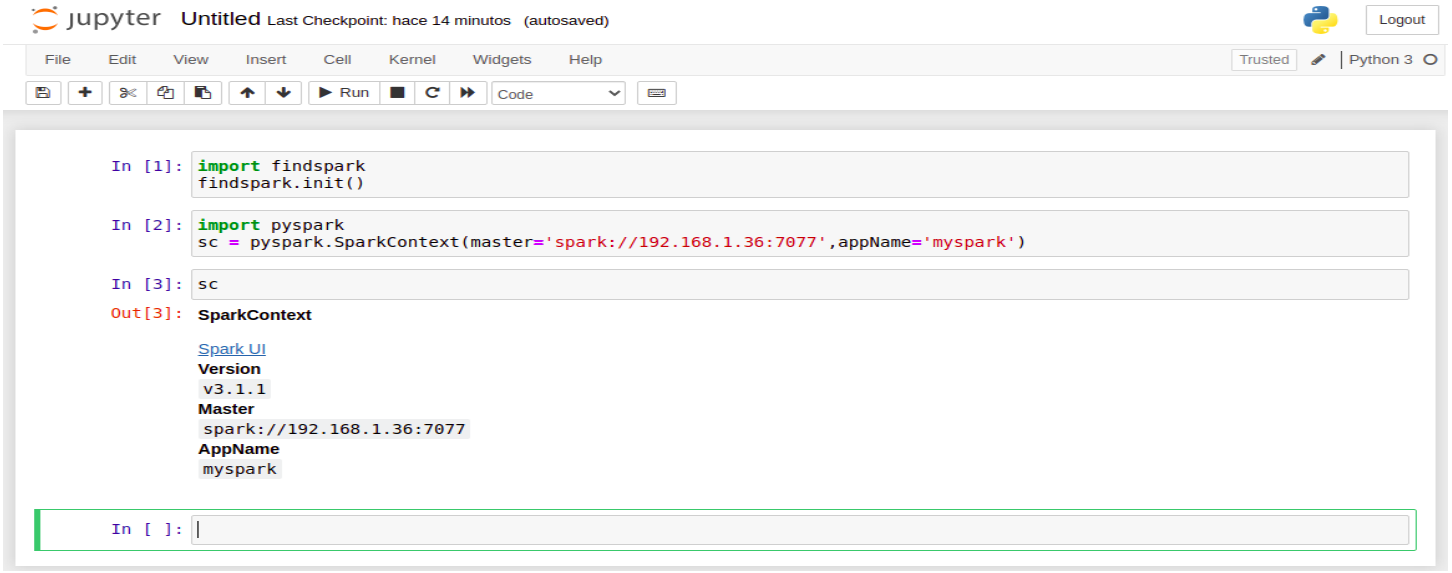
(my_pyspark) root@juan-Aspire-ES1-512:# jupyter notebook --allow-root

```
(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:
```

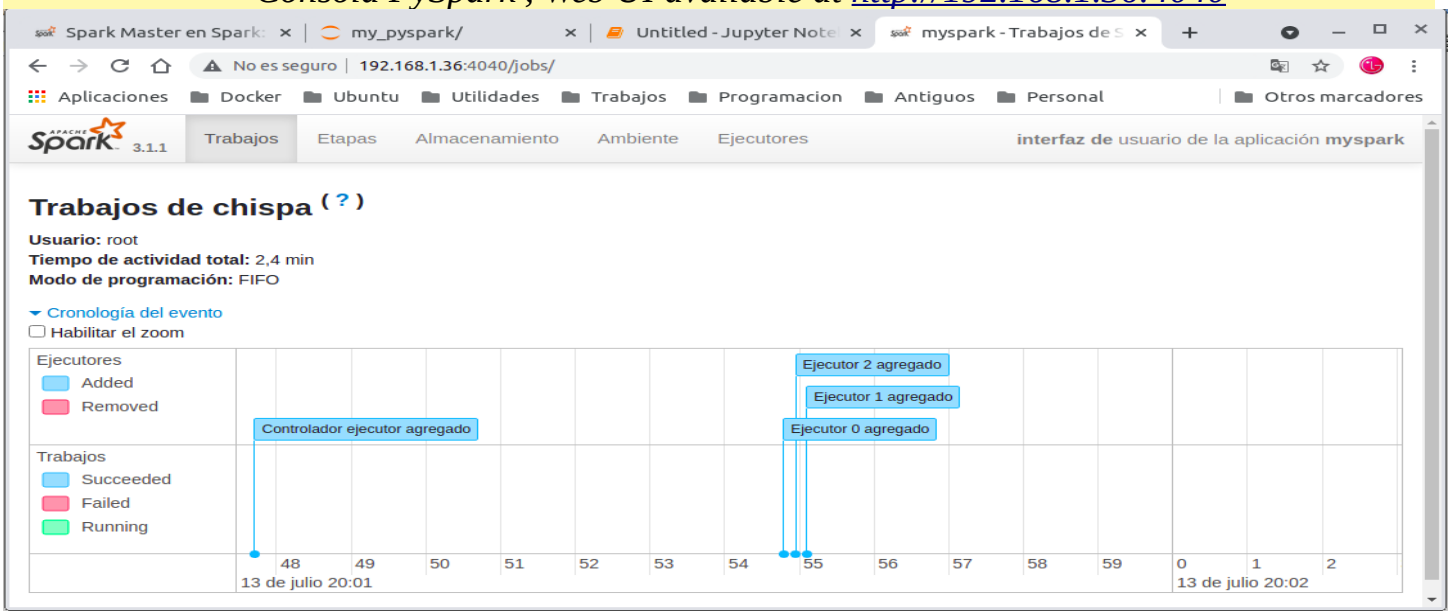
http://127.0.0.1:8888/?token=17cafcaaad4e02e72275b1332a26fed7334588b6df7e3c92

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

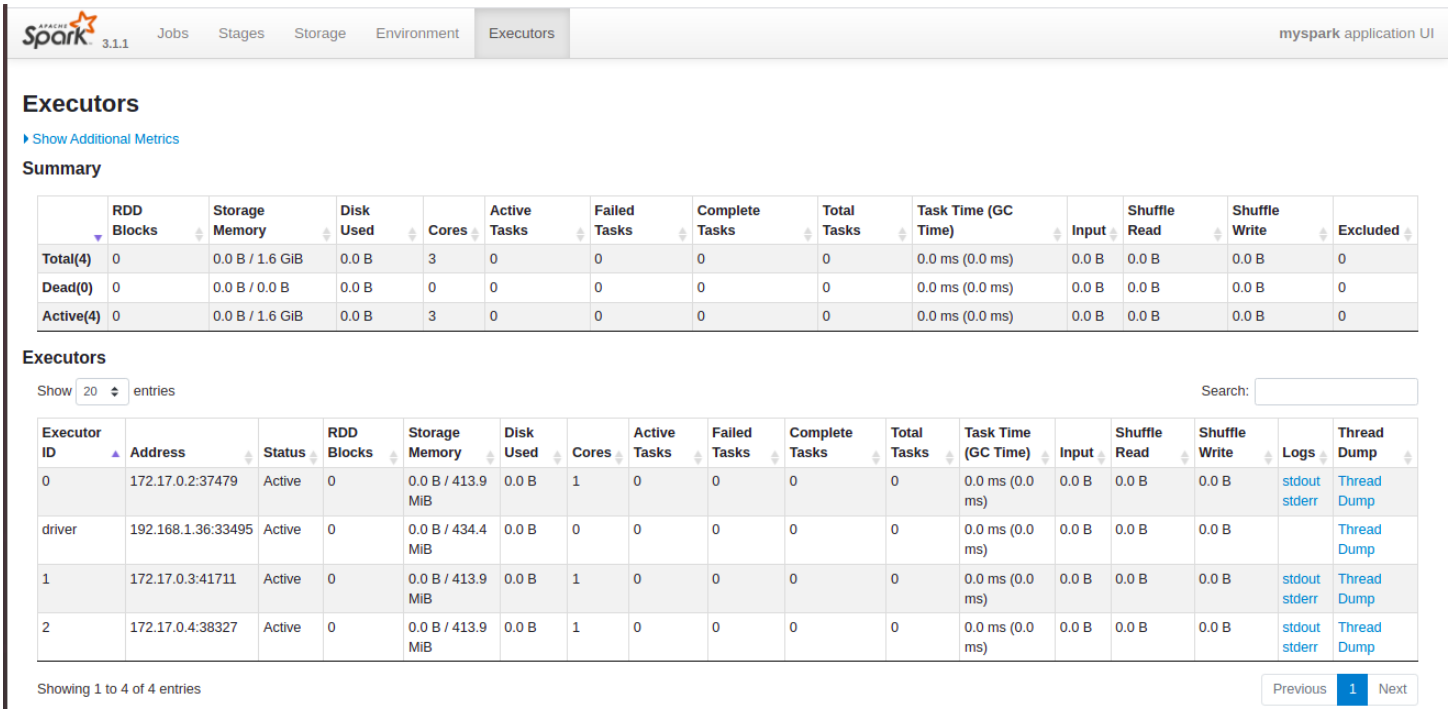
```
import findspark
findspark.init()

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



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) sc .



	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Excluded
Total(4)	0	0.0 B / 1.6 GiB	0.0 B	3	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0
Active(4)	0	0.0 B / 1.6 GiB	0.0 B	3	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	0

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
0	172.17.0.2:37479	Active	0	0.0 B / 413.9 MiB	0.0 B	1	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
driver	192.168.1.36:33495	Active	0	0.0 B / 434.4 MiB	0.0 B	0	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
1	172.17.0.3:41711	Active	0	0.0 B / 413.9 MiB	0.0 B	1	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump
2	172.17.0.4:38327	Active	0	0.0 B / 413.9 MiB	0.0 B	1	0	0	0	0	0.0 ms (0.0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump

En esta captura tenemos los Status , IP , memoria etc en Executurs .

Recapitulando : activamos el entorno virtual ***Python3.6-Venv*** , activamos el (***start-master.sh***) en el Host . En nuestro ***Servidor-Ubuntu*** con tres ***contenedor Docker*** activamos los ***tres Worker*** situado cada contenedor (***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 workers sintuados en los tres contenedores y los recurso de nuestro entorno .

Arrancaremos nuestra **IDE** favorita en mi caso ***Jupyter-Notebook*** , ***import findspark*** y ***pyspark*** y ejetutamos 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

Usar **Docker** y **Python-Venv** se trata de aislar todos estos procesos de la maquinas Host o Server con un entorno mas seguro y relativamente (fácil de usar) esto es el motivo de estos apuntes espero cumplan su objetivos .