

Creación de un Cluster en Apache Spark

1) Instalar Docker.

Actualiza el repositorio e instala las dependencias:

```
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

Crea la clave GPG key:

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

Añade al repositorio el enlace de docker:

```
echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Actualiza nuevamente el repositorio:

```
sudo apt-get update
```

Instala Docker Engine y Docker Compose:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Comprueba que todo ha ido bien:

```
sudo docker run hello-world
```

2) Descargar imagen de Ubuntu desde Docker.

```
sudo docker pull ubuntu
sudo docker image ls
```

3) Creamos el contenedor del master:

Nota: -it para activar el modo interactivo y poder trabajar desde la consola
`sudo docker run -it ubuntu`

4) El siguiente paso ha sido instalar las dependencias (Java, Python y Nano) y hemos usado los siguientes comandos:

```
export DEBIAN_FRONTEND=noninteractive
apt update && apt install -y openjdk-8-jdk python3 nano
```

Ahora abrimos 2 pestañas de la terminal más (serán nuestros nodos workers) y le instalamos a cada una:

```
sudo docker run -it ubuntu
export DEBIAN_FRONTEND=noninteractive
apt update && apt install -y openjdk-8-jdk python3 nano
```

Abrimos una nueva pestaña del terminal, para copiar en las 3 máquinas los datos con los que vamos a trabajar (nos movemos a la carpeta donde esté el archivo csv y el paquete de Spark):

```
sudo docker cp data.csv eb0:/opt
sudo docker cp spark-3.3.2-bin-hadoop3 eb0:/opt
```

NOTA: eb0 es el comienzo del nombre de la máquina a la que vamos a copiar (aparece en la pestaña de terminal). Lo mismo para cada máquina.

Ahora si nos movemos por cada máquina y escribimos

```
cd /opt
```

```
ls
```

Veremos que están copiados ambos archivos.

En la máquina principal vamos a la carpeta de Spark:

```
cd spark-3.3.2-bin-hadoop3/sbin
```

```
ls
```

```
./start-master.sh -h 0.0.0.0
```

Le hemos pasado la IP local para que esté en modo escucha en esa dirección.

Ya estaría arrancada la máquina master. Para comprobarlo, abrimos el navegador y escribimos la IP 172.17.0.X:8080 siendo X la IP local (hay que probar 2, 3, 4...).

Le toca el turno a los esclavos. En cada nodo worker, ejecutamos:

```
cd /sbin
```

```
./start-worker.sh spark://172.17.0.2:7077
```

Si actualizamos el navegador, veremos los workers.

Ahora vamos a lanzar un programa para que se ejecute en los nodos esclavos. Desde el nodo maestro, vamos a /bin y creamos un archivo nuevo:

```
nano app.py
```

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
df = spark.read.options(header='True', inferSchema='True').csv("/opt/*.csv")
```

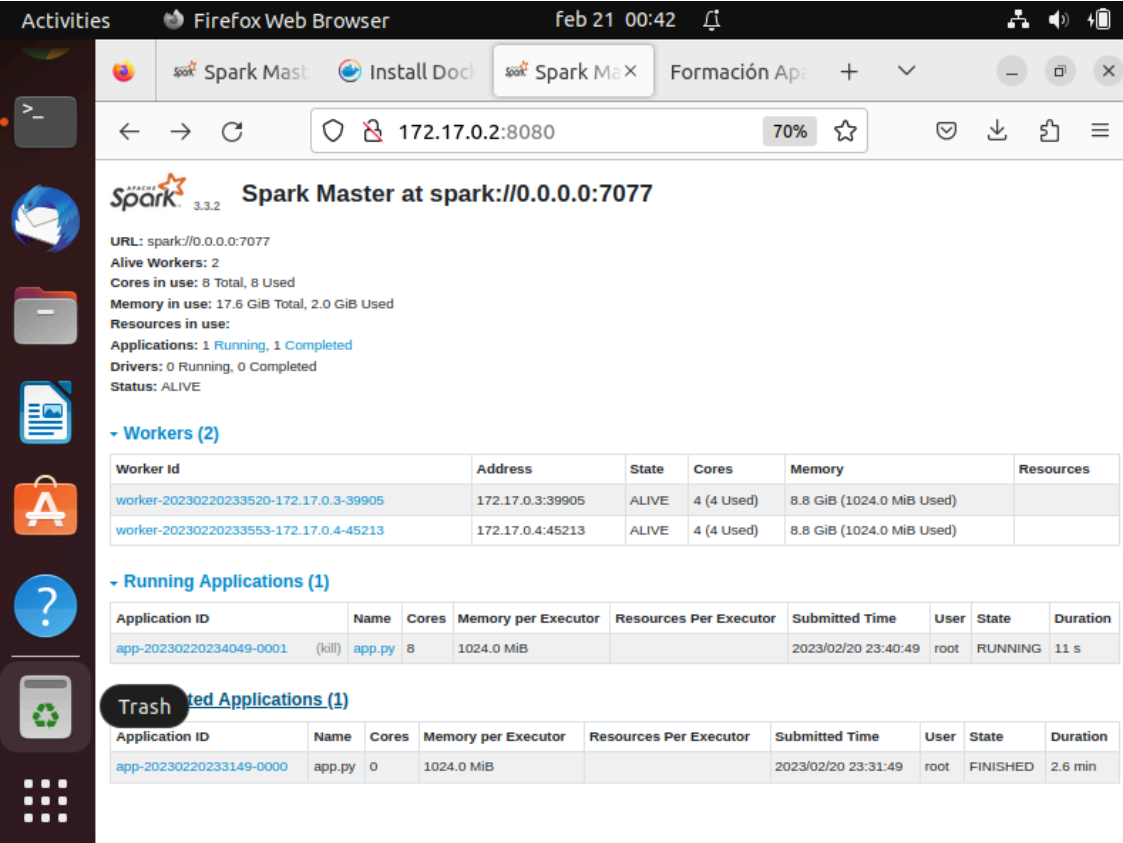
```
df.count()
df.printSchema()
df.select("categories").distinct().show()
def myFunc(s):
    return [ ( s["categories"], 1) ]
```

```
lines=df.rdd.flatMap(myFunc).reduceByKey(lambda a, b: a + b)
```

Guardamos y ejecutamos:

```
./spark-submit --master spark://172.17.0.2:7077 app.py
```

Si actualizamos el navegador, veremos el proceso en ejecución, así como lo que ejecuta cada uno de los nodos esclavos.



The screenshot shows a Firefox Web Browser window with the address bar set to `172.17.0.2:8080`. The page displays the Spark Master at `spark://0.0.0.0:7077`. The UI includes the following sections:

- URL:** `spark://0.0.0.0:7077`
- Alive Workers:** 2
- Cores in use:** 8 Total, 8 Used
- Memory in use:** 17.6 GiB Total, 2.0 GiB Used
- Resources in use:**
- Applications:** 1 Running, 1 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
<code>worker-20230220233520-172.17.0.3-39905</code>	<code>172.17.0.3:39905</code>	ALIVE	4 (4 Used)	8.8 GiB (1024.0 MiB Used)	
<code>worker-20230220233553-172.17.0.4-45213</code>	<code>172.17.0.4:45213</code>	ALIVE	4 (4 Used)	8.8 GiB (1024.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
<code>app-20230220234049-0001</code> (kill)	<code>app.py</code>	8	1024.0 MiB		2023/02/20 23:40:49	root	RUNNING	11 s

Trash Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
<code>app-20230220233149-0000</code>	<code>app.py</code>	0	1024.0 MiB		2023/02/20 23:31:49	root	FINISHED	2.6 min