

MANUAL DE USO

Para la creación del cluster de Spark es necesario que los workers tengan comunicación con el master. Para tal fin se deben cumplir dos condiciones:

- El *master* (Servidor) debe vincularse con una interfaz, IP o nombre accesible desde los *workers* (*Clientes*).
- Los *workers* deben disponer de la IP del *master* o un nombre que resuelva correctamente a dicha IP, así como conectividad con el mismo.

La dirección de vinculación se le proporciona al *master* a través de la variable de entorno SPARK_MASTER_HOST. Por defecto la imagen tiene un valor de *spark-master*.

La dirección del *master* se le proporciona a cada *worker* a través de la variable de entorno SPARK_MASTER_URL. Por defecto la imagen tiene un valor de *spark://spark-master:7077*.

Para que los contenedores resuelvan el nombre *spark-master* a la IP correcta hay que apoyarse en el servidor DNS para redes definidas por el usuario, una característica introducida a partir de la version 1.10 de Docker Engine. Este servidor DNS proporciona una función de descubrimiento de servicios para cualquier contenedor creado con un nombre o net-alias válido. Las redes definidas por el usuario son redes virtuales que permiten agrupar los contenedores en subredes, según las preferencias del usuario.

Antes de iniciar se debe configurar la red que albergará al cliente y los equipos que servirán de estaciones, la red definida por el usuario se nombrará *sparknet*, y se debe instanciar un contenedor Spark *master* y dos contenedores Spark *workers*.

```
# Se crea la red
```

```
docker network create sparknet
```

```
# Se ejecuta en el puerto 8080
```

```
docker run -d --net=sparknet -p 8080:8080 --name spark-master gradient/spark:2.0.0  
master
```

```
# Se crean las estaciones de trabajo y se indica que es un cliente del servidor
```

```
docker run -d --net=sparknet --name spark-worker1 gradient/spark:2.0.0 worker
```

```
docker run -d --net=sparknet --name spark-worker2 gradient/spark:2.0.0 worker
```

Para comprobar que la resolución de nombres funciona correctamente y existe conectividad entre los *workers* y el *master* se puede ejecutar un comando ping en cada contenedor:

```
# Se ejecuta el servidor y se hace un ping al mismo.
```

```
docker exec spark-master ping -c 1 spark-master
```

```
# Ping desde el cliente hacía el servidor
```

```
docker exec spark-worker2 ping -c 1 spark-master
```

```
docker exec spark-worker2 ping -c 1 spark-master
```

Es posible utilizar nombres personalizados para los contenedores. Solo es necesario sobrescribir las variables de entorno que proporcionan a Spark el nombre del contenedor *master*, por ejemplo:

#Cambiar el nombre del servidor

```
docker run -d --net=sparknet --name maestro -e SPARK_MASTER_HOST=maestro  
gradient/spark:2.0.0 master
```

#Cambiar el nombre del cliente o estaciones de trabajo.

```
docker run -d --net=sparknet --name esclavo -e  
SPARK_MASTER_URL=spark://maestro:7077 gradient/spark:2.0.0 worker
```

Accediendo a la interfaz de usuario de Spark puede comprobarse el estado *demaster y workers*:

The screenshot displays the Spark Master web interface at the URL `spark://spark-master:7077`. It provides a comprehensive overview of the cluster's health and activity.

Cluster Summary:

- URL: `spark://spark-master:7077`
- REST URL: `spark://spark-master:8080 (cluster mode)`
- Alive Workers: 2
- Cores in use: 4 Total, 0 Used
- Memory in use: 2.0 GB Total, 0.0 B Used
- Applications: 0 Running, 4 Completed
- Drivers: 0 Running, 0 Completed
- Status: ALIVE

Workers Table:

Worker Id	Address	State	Cores	Memory
worker-20170422195427-172.19.0.3-45111	172.19.0.3:45111	ALIVE	2 (0 Used)	1024.0 MB (0.0 B Used)
worker-20170422195427-172.19.0.4-38435	172.19.0.4:38435	ALIVE	2 (0 Used)	1024.0 MB (0.0 B Used)

Running Applications Table:

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
----------------	------	-------	-----------------	----------------	------	-------	----------

Completed Applications Table:

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20170422223454-0003	Spark shell	4	1024.0 MB	2017/04/22 22:34:54	root	FINISHED	2.5 min
app-20170422222850-0002	Spark shell	4	1024.0 MB	2017/04/22 22:28:50	root	FINISHED	5.2 min
app-20170422201124-0001	Spark shell	4	1024.0 MB	2017/04/22 20:11:24	root	KILLED	14 min
app-20170422200114-0000	Spark shell	4	1024.0 MB	2017/04/22 20:01:14	root	FINISHED	8.7 min

INICIAR

#Ruta del directorio

```
cd spark-2.0.0-bin-hadoop2.7/
```

Iniciar Servidor

```
docker start spark-master
```

#Iniciar Clientes

```
docker start spark-worker1
```

```
docker start spark-worker2
```

#Iniciar Shell del Cliente

```
docker exec -it spark-worker1 bash
```

Servidor Local

```
spark-shell --master spark://spark-master:7077
```

Welcome to



```
Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_112)
Type in expressions to have them evaluated.
Type :help for more information.
```

EJEMPLO: Contar el número de palabras con más coincidencias dentro de un texto

```
val docs = sc.textFile("biblia.txt")

val lower=docs.map(line => line.toLowerCase)

val stopwords=Set("the","of","my")

words =lower.flatMap(line => line.split("\\s+")).filter(! stopwords.contains(_))

val counts=words.map(word =>(word,1))

val freq =counts.reduceByKey(_ + _)
```

#Resultado

```
(51380, and)
(25295, )
(13643, to)
(12799, that)
(12560, in)
(10263, he)
(9840, shall)
(8987, unto)
(8836, for)
(8708, i)
(8450, his)
(8232, a)
(7300, they)
(6877, is)
(6873, be)
(6049, with)
(5904, not)
(5450, all)
(5202, thou)
(4739, lord)
```