

V Big Data and Machine Learning Bootcamp

Data Processing

PRACTICA (3/3)

Punto 2.1

INSTALACION DE ZEPPELIN 0.8.2

Esta instalación la haré en un MAC.

Vamos a la página web <https://www.zppelin.apache.org> y pulsamos en "Download", "zeppelin-0-8-2-bin.all.tgz". Hago la descarga en el home del ordenador (/Users/fernandojarilla/)

Abrimos un terminal y vamos al home

```
$ cd /users/fernandojarilla
```

Aquí tenemos el fichero zeppelin-0.8.2-bin-all.tgz. Lo descomprimos

```
$ tar -xvzf zeppelin-0.8.2-bin-all.tgz
```

Esto me crea el directorio zeppelin-0.8.2-bin-all. Entramos

```
$ cd zeppelin-0.8.2-bin-all
```

Vemos que está descomprimido y tenemos los directorios /bin y /conf que necesitaremos. Desde aquí ejecutaremos

```
$ /bin/zeppelin-daemon.sh start
```

Con esto, lanzamos el proceso y vemos que lo ha ejecutado correctamente

```
iMac:zeppelin-0.8.2-bin-all fernandojarilla$ bin/zeppelin-daemon.sh start
Log dir doesn't exist, create /Users/fernandojarilla/zeppelin-0.8.2-bin-all/logs
Pid dir doesn't exist, create /Users/fernandojarilla/zeppelin-0.8.2-bin-all/run
Zeppelin start [ OK ]
iMac:zeppelin-0.8.2-bin-all fernandojarilla$
```

Foto 1 Arranque correcto de Zeppelin

Para parar el proceso, haríamos

```
$ /bin/zeppelin-daemon.sh stop
```

```
Pid dir doesn't exist, create /Users/fernandojarilla/zeppelin-0.8.2-bin-all/run
Zeppelin start [ OK ]
iMac:zeppelin-0.8.2-bin-all fernandojarilla$ bin/zeppelin-daemon.sh stop
Zeppelin stop [ OK ]
iMac:zeppelin-0.8.2-bin-all fernandojarilla$
```

Foto 2: Parada de Zeppelin

Este proceso lo ha lanzado en localhost en el puerto 8080 (localhost:8080). Para cambiar este puerto, ya que lo tengo ocupado por otro proceso, haremos:

Editamos el fichero **zeppelin-site.xml.template** y cambiamos **zeppelin.server.port** a **8180**

Editamos el fichero **zeppelin-env.sh.template** y añadimos la línea **export ZEPPELIN_PORT=8180**

Copiamos los templates como ficheros de configuración (que usará Zeppelin)

```
cp conf/zeppelin-env.sh.template conf/zeppelin-env.sh
```

```
cp conf/zeppelin-site.xml.template conf/zeppelin-site.xml
```

Ahora volvemos a arrancar zeppelin

```
$ /bin/zeppelin-daemon.sh start
```

Nos vamos al navegador e introducimos **localhost:8180**, que es el puerto donde se ejecuta el Notebook de Zeppelin. Nos aparece la ventana de zeppelin

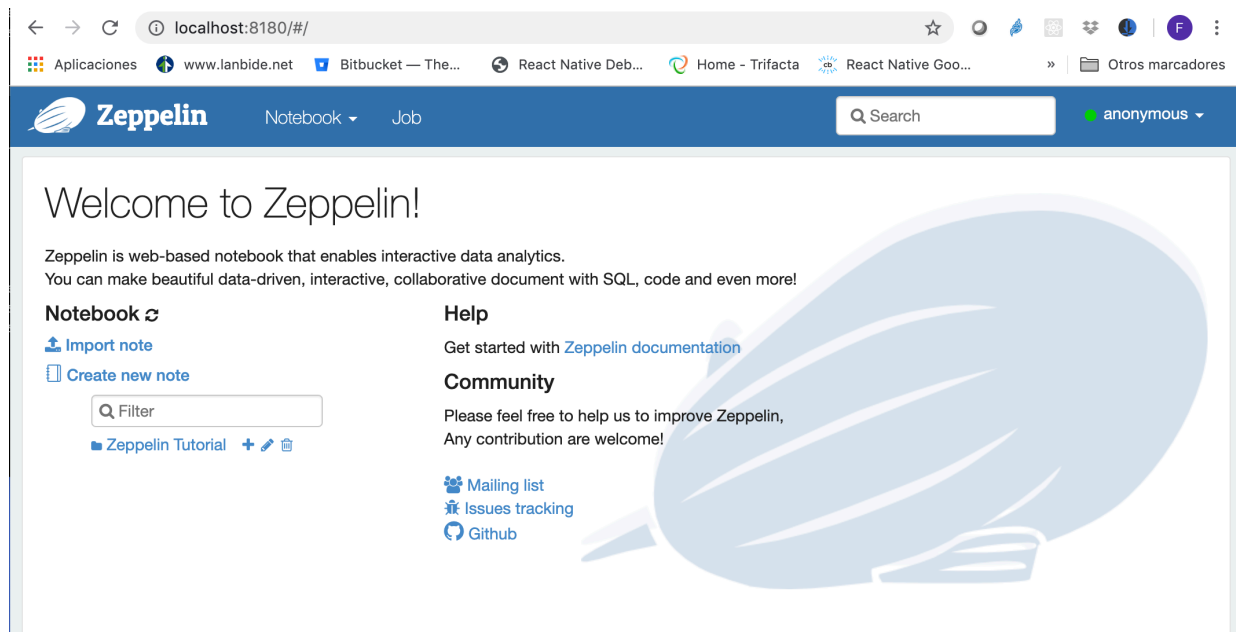


Foto 3: Ventana de inicio de Zeppelin

Vamos a lanzar una aplicación y para ello pulsamos en "**Create new note**"

Le llamamos Practica1 y nos aparece el notebook donde podremos lanzar comandos de scala. Probamos un simple `println("Hola mundo")`

Para poder trabajar con Spark 2.x, debemos vincularlo a Zeppelin, pues este viene vinculado a Spark 1.x Para ello, desplegamos el menu "**anonymous**" y pulsamos sobre "**Interpreter**"

Dentro de los Interpreters, pulsamos "**+ Create**"

Como "**Interpreter Name**", ponemos **Spark2**

Como "**Interpreter Group**", seleccionamos **Spark**

Seleccionamos, al final del documento, "**Save**"

Create new interpreter

Interpreter Name

Interpreter group

Option

The interpreter will be instantiated Globally in shared process [i](#)

☐ Connect to existing process

☐ Set permission

Properties

name	value	action	description
args	<input type="text"/>	<input type="button" value="x"/>	spark commandline args
master	<input type="text" value="local[*]"/>	<input type="button" value="x"/>	Spark master uri. ex) spark://masterhost:7077
spark.app.name	<input type="text" value="Zeppelin"/>	<input type="button" value="x"/>	The name of spark application.
spark.cores.max	<input type="text"/>	<input type="button" value="x"/>	Total number of cores to use. Empty value uses all available core.
spark.executor.memory	<input type="text"/>	<input type="button" value="x"/>	Executor memory per worker instance. ex) 512m, 32g
zeppelin.R.cmd	<input type="text" value="R"/>	<input type="button" value="x"/>	R repl path
zeppelin.R.image.width	<input type="text" value="100%"/>	<input type="button" value="x"/>	
zeppelin.R.knitr	<input checked="" type="checkbox"/>	<input type="button" value="x"/>	whether use knitr or not
zeppelin.R.render.options	<input type="text" value="out.format = 'html', comment = NA, echo = FALSE, results = 'asis', message = F, warning = F, fig.retina = 2"/>	<input type="button" value="x"/>	

Foto 4: Crear nuevo interprete (Spark). Definiciones

			ly if set true.
zeppelin.spark.enableSupportedVersionCheck	<input checked="" type="checkbox"/>	<input type="button" value="x"/>	Do not change - developer only setting, not for production use
zeppelin.spark.importImplicit	<input checked="" type="checkbox"/>	<input type="button" value="x"/>	Import implicits, UDF collection, and sql if set true. true by default.
zeppelin.spark.maxResult	<input type="text" value="1000"/>	<input type="button" value="x"/>	Max number of Spark SQL result to display.
zeppelin.spark.printREPLOutput	<input checked="" type="checkbox"/>	<input type="button" value="x"/>	Print REPL output
zeppelin.spark.sql.interpolation	<input type="checkbox"/>	<input type="button" value="x"/>	Enable ZeppelinContext variable interpolation into paragraph text
zeppelin.spark.sql.stacktrace	<input type="checkbox"/>	<input type="button" value="x"/>	Show full exception stacktrace for or SQL queries if set to true.
zeppelin.spark.ui.hidden	<input type="checkbox"/>	<input type="button" value="x"/>	Whether to hide spark ui in zeppelin ui
zeppelin.spark.uiWebUrl	<input type="text"/>	<input type="button" value="x"/>	Override Spark UI default URL
zeppelin.spark.useHiveContext	<input checked="" type="checkbox"/>	<input type="button" value="x"/>	Use HiveContext instead of SQL Context if it is true.
zeppelin.spark.useNew	<input checked="" type="checkbox"/>	<input type="button" value="x"/>	Whether use new spark interpreter implementation
<input type="text"/>	<input type="text"/>	<input type="button" value="textarea"/> <input type="button" value="+"/>	


Dependencies

artifact	exclude	action
<input type="text" value="groupId:artifactId:version or local file path"/>	<input type="text" value="(Optional) comma separated groupId:artifactId list"/>	<input type="button" value="+"/>

Foto 5: Crear nuevo interprete, Salvar

Con esto hemos creado lo que queremos, pero no le hemos dado contenido.

Para darle este contenido, vamos a **Interpreters** y buscamos **Spark2**. Ahora lo tenemos. Sobre **Spark2**, pulsamos "edit"


Zeppelin

Notebook ▾
Job

● anonymous ▾

Repository
+ Create

Spark2
%Spark2, %sql, %dep, %pyspark, %ipyspark, %r ●

spark ui
edit
restart
remove

Option

The interpreter will be instantiated Globally ▾ in shared ▾ process ⓘ

☐ Connect to existing process
☐ Set permission

Properties

name	value
args	
master	local[*]
spark.app.name	Zeppelin
spark.cores.max	
spark.executor.memory	
zeppelin.R.cmd	R
zeppelin.R.image.width	100%
zeppelin.R.knitr	true
zeppelin.R.render.options	out.format = 'html', comment = NA, echo = FALSE, results = 'asis', message = F, warning = F, fig.retina = 2
zeppelin.dep.additionalRemoteRepository	spark-packages,http://dl.bintray.com/spark-packages/maven,false;

Foto 6: Editar el nuevo interprete (Spark2)

En este edit, vamos a la parte inferior y encima de Dependencies (en el cuadro que aparece y en el siguiente) ponemos

SPARK_HOME

/users/fernandojarilla/spark-2.4.4-bin-hadoop2.7

zeppelin.pyspark.python	python	x
zeppelin.pyspark.usePython	<input checked="" type="checkbox"/>	x
zeppelin.spark.concurrentSQL	<input type="checkbox"/>	x
zeppelin.spark.enableSupportedVersionCheck	<input checked="" type="checkbox"/>	x
zeppelin.spark.importImplicit	<input checked="" type="checkbox"/>	x
zeppelin.spark.maxResult	1000	x
zeppelin.spark.printREPLOutput	<input checked="" type="checkbox"/>	x
zeppelin.spark.sql.interpolation	<input type="checkbox"/>	x
zeppelin.spark.sql.stacktrace	<input type="checkbox"/>	x
zeppelin.spark.ui.hidden	<input type="checkbox"/>	x
zeppelin.spark.ui.WebUrl		x
zeppelin.spark.useHiveContext	<input checked="" type="checkbox"/>	x
zeppelin.spark.useNew	<input checked="" type="checkbox"/>	x
SPARK_HOME	/users/fernandojarilla/spark-2.4.4-bin-hadoop2.7	textarea +

Dependencies
These dependencies will be added to classpath when interpreter process starts.

artifact	exclude	action
groupid:artifactId:version or local file path	(Optional) comma separated groupid:artifactId list	+

Save Cancel

Foto 7: Linea a añadir

Con esto añadido, pulsamos **"Save"**. Nos pregunta si queremos reiniciar con la nueva configuración y le decimos **"OK"**. Ya estamos en disposición de usar, dentro de zeppelin, Stark 2.x

Para ver que esto es así, creamos una nueva nota (**Notebook / Create new note**) y ya vemos que el **Default Interpreter** que nos sale es **Spark2** (Como queriamos)

Llamamos a la nota, **"PracticaZeppelin"**

Para comprobar si todo ha ido bien, sobre la línea de ejecución pulsamos

```
sc.version
```

La ejecutamos (**"Run"**) comprobamos que nos da

```
res1: String = 2.4.4
```

que es la versión de Spark que tenemos instalada.

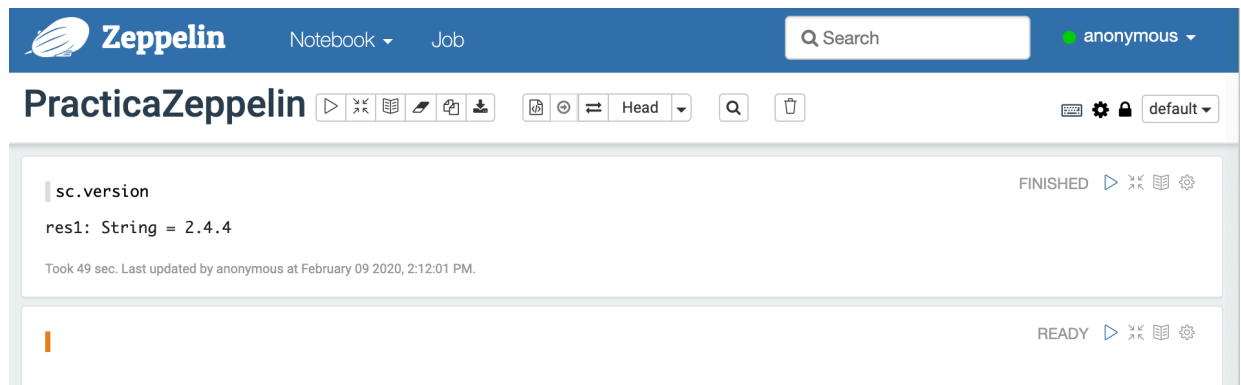


Foto 8: Versión de Spark enlazada con Zeppelin

Punto 2.2

Ahora vamos a hacer el ejercicio pedido sobre Zeppelin. Creamos la aplicación. EL programa quedará:

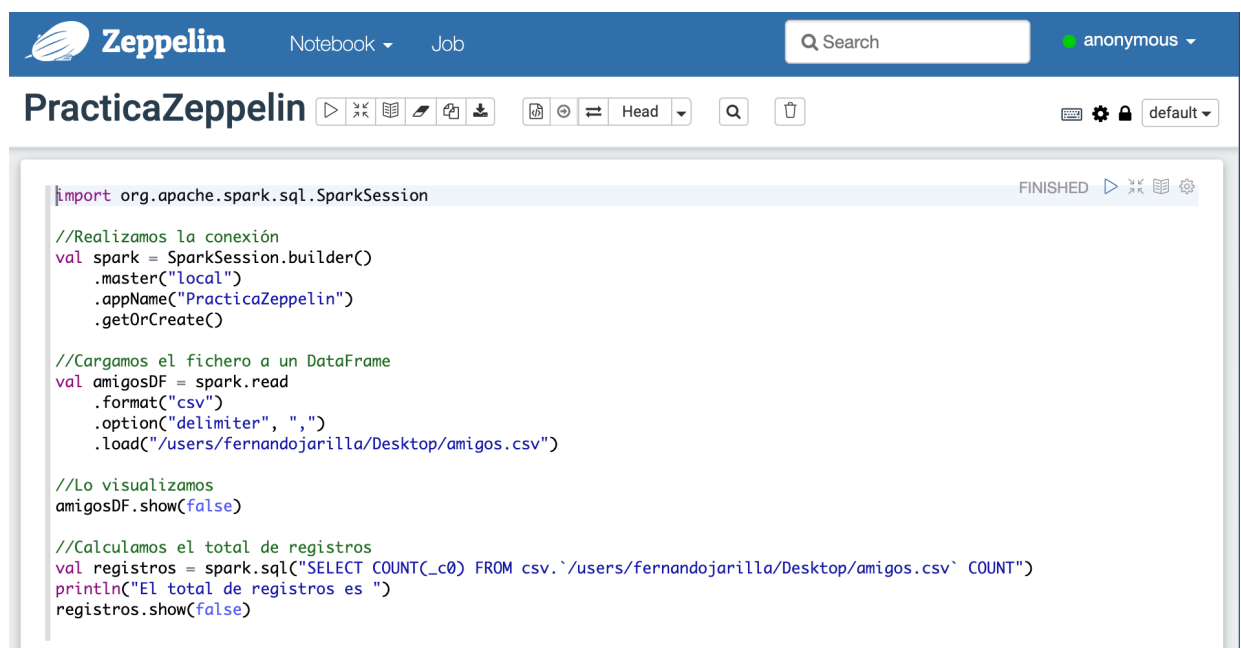



Foto 9: Fuente del programa de la práctica

Lo hacemos correr ("Run"), y comprobamos el resultado


Notebook ▾ Job

anonymous ▾

PracticaZeppelin

▶
⌕
📖
✍️
📄
📥

🔗
⌚
⚖️
Head ▾

🔍
🗑️

🔴
⚙️
🔒
default ▾

FINISHED
▶
🔄
📖
⚙️

```

+---+-----+---+---+
|_c0|_c1      |_c2|_c3|
+---+-----+---+---+
|0  |Will      |33  |385|
|1  |Jean-Luc  |26  |12  |
|2  |Hugh      |55  |221|
|3  |Deanna    |40  |465|
|4  |Quark     |68  |21  |
|5  |Weyoun    |59  |318|
|6  |Gowron    |37  |220|
|7  |Will      |54  |307|
|8  |Jadzia    |38  |380|
|9  |Hugh      |27  |181|
|10 |Odo       |53  |191|
|11 |Ben       |57  |372|
|12 |Keiko     |54  |253|
|13 |Jean-Luc  |56  |444|
|14 |Hugh      |43  |49  |
|15 |Rom       |36  |49  |
|16 |Weyoun    |22  |323|
|17 |Odo       |35  |13  |
|18 |Jean-Luc  |45  |455|
|19 |Geordi    |60  |246|
+---+-----+---+---+
only showing top 20 rows

El total de registros es
+-----+
|count(_c0)|
+-----+
|500      |
+-----+

import org.apache.spark.sql.SparkSession
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@4b046793
amigosDF: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 2 more fields]
registros: org.apache.spark.sql.DataFrame = [count(_c0): bigint]

```

Took 0 sec. Last updated by anonymous at February 09 2020, 2:33:00 PM. (outdated)

Foto 10: Ejecución del programa de la práctica