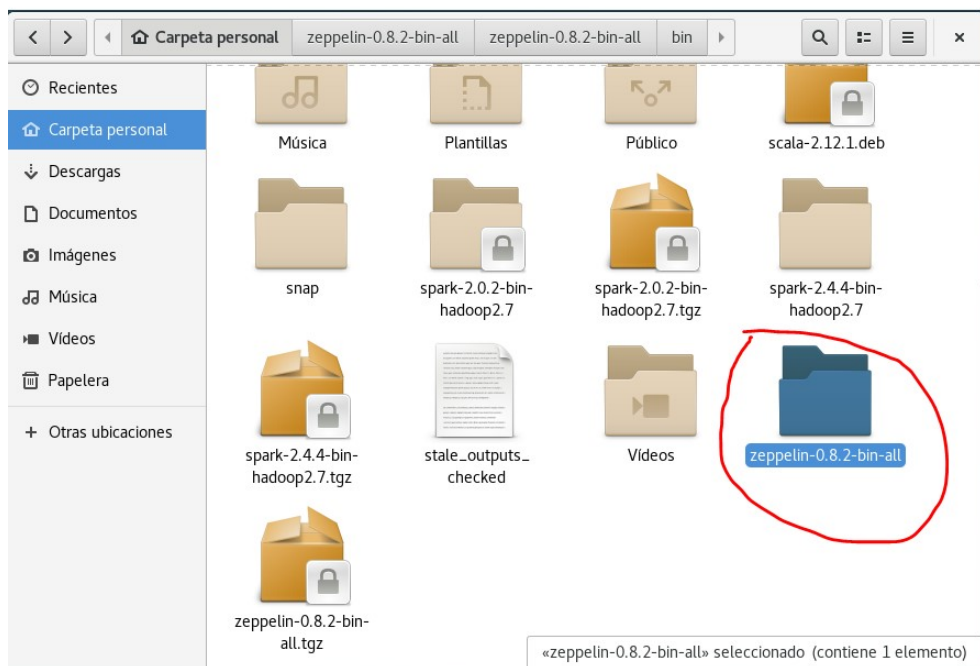


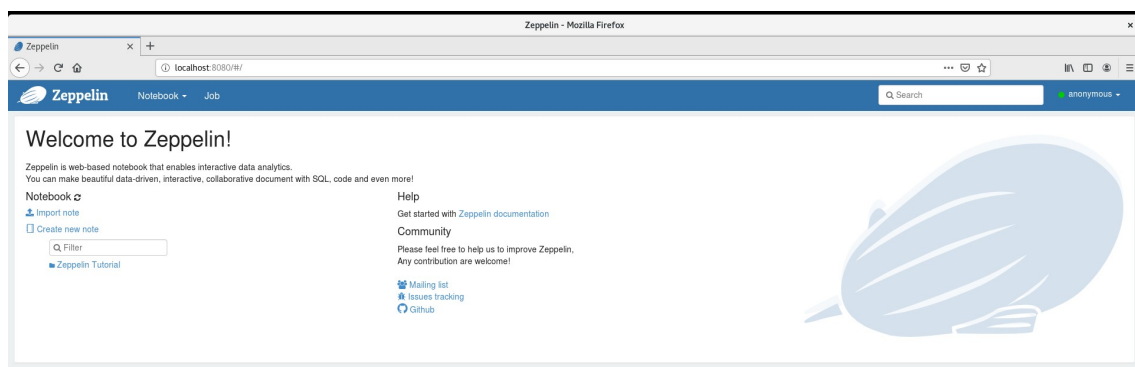
Descomprimos el zeppelin que tenemos en la maquina virtual.



Una vez descomprimido vamos al directorio donde esta y ejecutamos con start

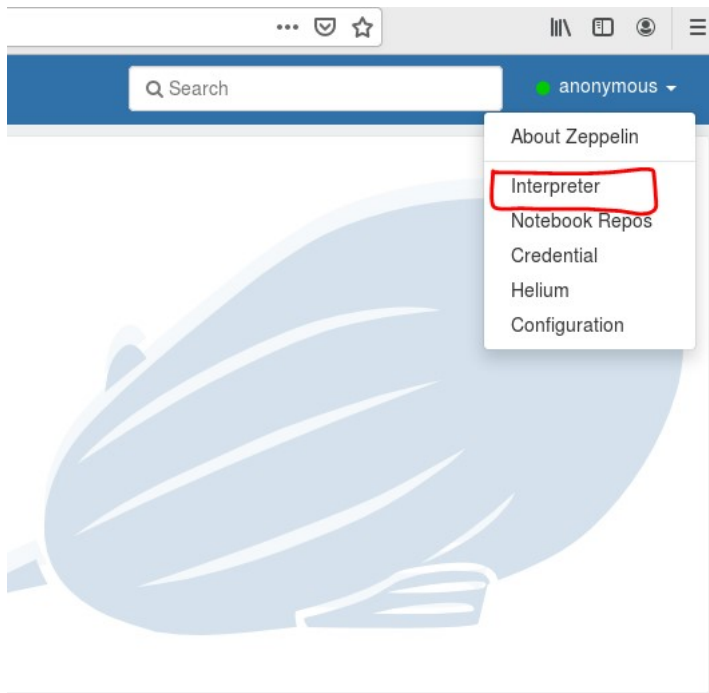
```
keepcoding@debian:~/zeppelin-0.8.2-bin-all/zeppelin-0.8.2-bin-all$ bin/zeppelin-daemon.sh start
Log dir doesn't exist, create /home/keepcoding/zeppelin-0.8.2-bin-all/zeppelin-0.8.2-bin-all/logs
Pid dir doesn't exist, create /home/keepcoding/zeppelin-0.8.2-bin-all/zeppelin-0.8.2-bin-all/run
Zeppelin start [ OK ]
keepcoding@debian:~/zeppelin-0.8.2-bin-all/zeppelin-0.8.2-bin-all$
```

Una vez hecho esto vamos al navegador y vamos a localhost:8080

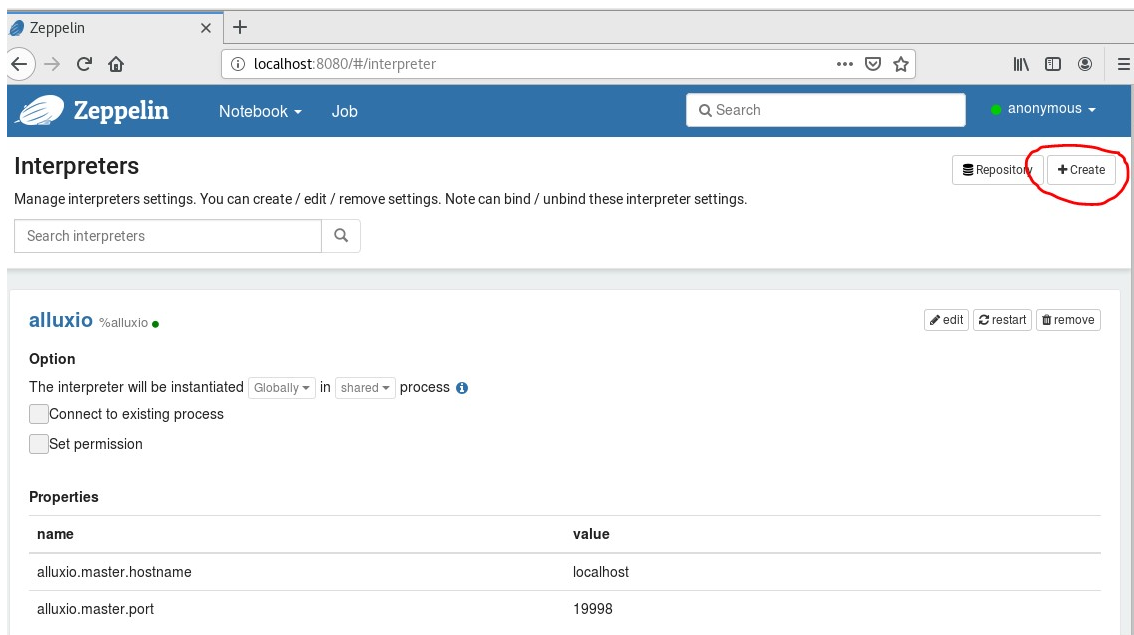


Vemos que ya tenemos el zeppelin, ahora hay que configurarlo.

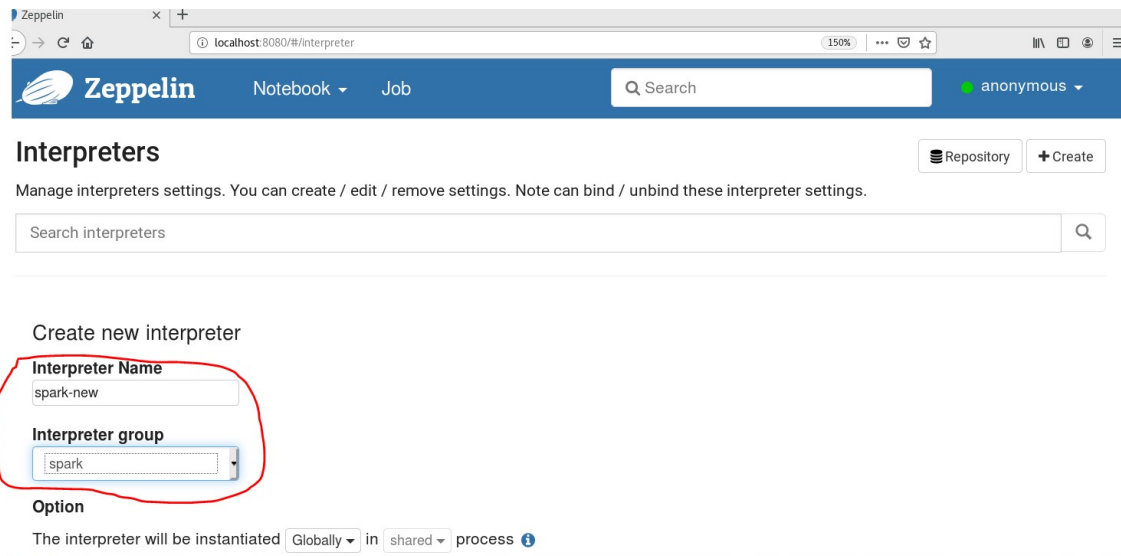
Vamos al desplegable de anonymous y pinchamos en interpreter.



Vamos a crear uno nuevo, para ello pulsamos créate



Le ponemos el nombre que queramos y ponemos en el interpreter group spark.



Zeppelin

Notebook Job

Search

anonymous

## Interpreters

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

Search interpreters

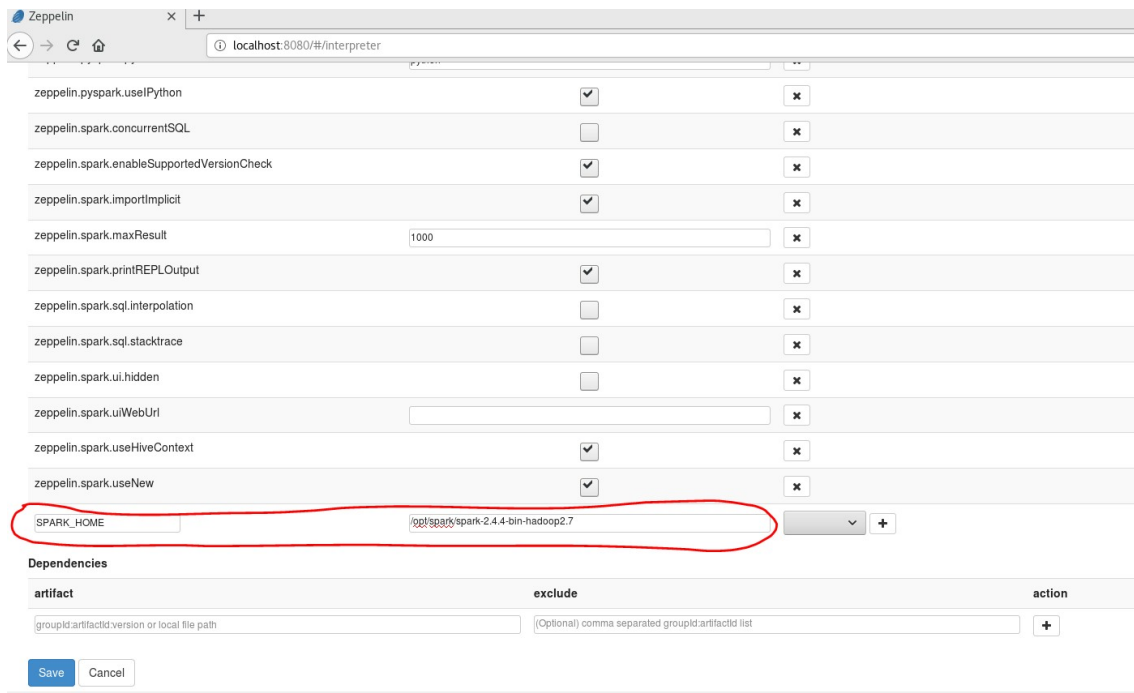
Create new interpreter

**Interpreter Name**  
spark-new

**Interpreter group**  
spark

**Option**  
The interpreter will be instantiated Globally in shared process

En la parte de abajo antes de guardar tenemos que poner SPARK\_HOME y la ruta a nuestra carpeta spark y una vez hecho pulsamos en Save y ya hemos creado el interprete



Zeppelin

localhost:8080/#/interpreter

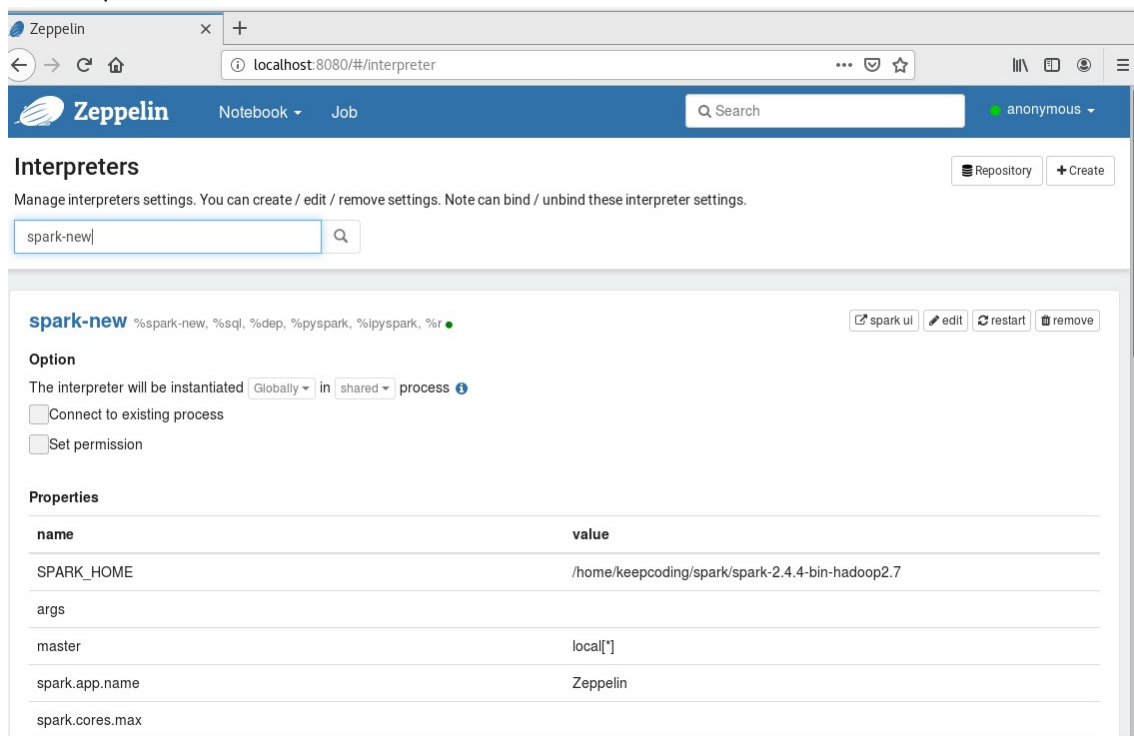
zeppelin.pyspark.usePython	<input checked="" type="checkbox"/>	<input type="checkbox"/>
zeppelin.spark.concurrentSQL	<input type="checkbox"/>	<input type="checkbox"/>
zeppelin.spark.enableSupportedVersionCheck	<input checked="" type="checkbox"/>	<input type="checkbox"/>
zeppelin.spark.importImplicit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

Dependencies

artifact	exclude	action
groupId:artifactId:version or local file path	(Optional) comma separated groupId:artifactId list	<input data-bbox="1257 1585 1284 1608" type="button" value="+"/>

Save Cancel

Si queremos ver que esta bien creado en interpreters buscamos el que acabamos de hacer y vemos que esta bien.



**Interpreters**

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

spark-new

**spark-new** %spark-new, %sql, %dep, %pyspark, %lpyspark, %r

Options: Globally, in shared, process

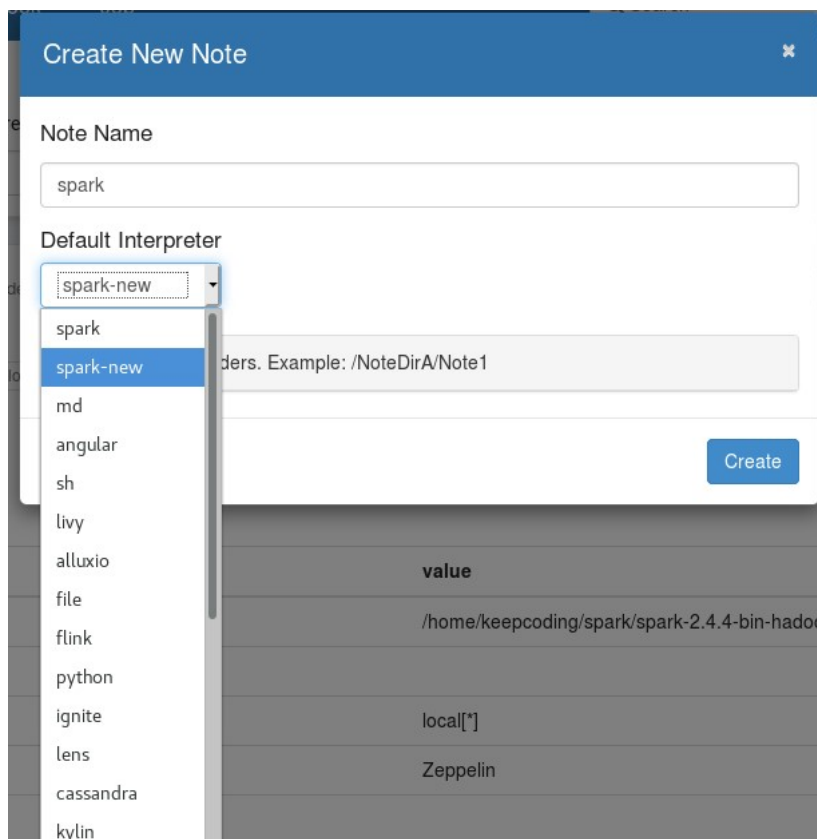
☐ Connect to existing process

☐ Set permission

**Properties**

name	value
SPARK_HOME	/home/keepcoding/spark/spark-2.4.4-bin-hadoop2.7
args	
master	local[*]
spark.app.name	Zeppelin
spark.cores.max	

Ahora vamos a crear un nuevo notebook para realizar la practica, para ello en el interprete buscamos el interprete que acabamos de crear.



**Create New Note**

Note Name: spark

Default Interpreter: spark-new

Buttons: Create

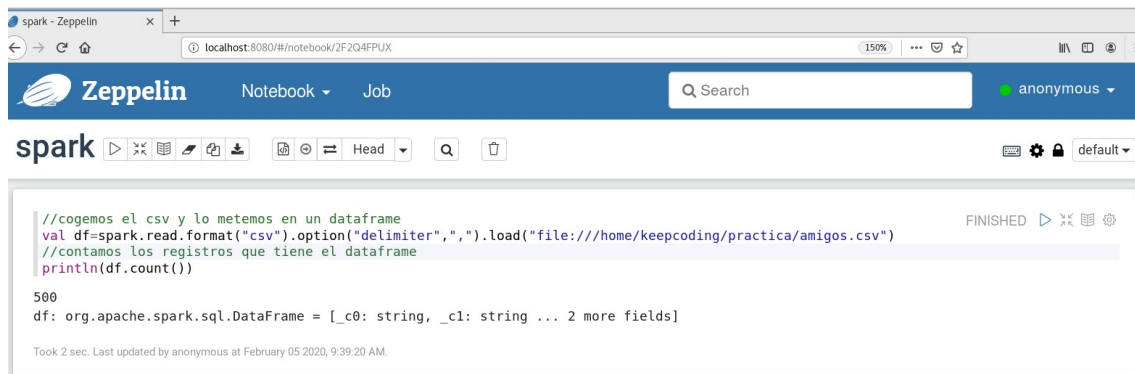
Example: /NoteDirA/Note1

value: /home/keepcoding/spark/spark-2.4.4-bin-hadoo

local[\*]

Zeppelin

Una vez creado vamos a realizar la practica contando los registros de amigos.csv



The screenshot shows the Zeppelin Notebook interface. The top bar includes the Zeppelin logo, 'Notebook' and 'Job' tabs, a search bar, and a user profile 'anonymous'. Below the top bar is a toolbar with various icons for code execution and management. The main area displays a Spark code cell with the following code:

```
//cogemos el csv y lo metemos en un dataframe
val df=spark.read.format("csv").option("delimiter","," ).load("file:///home/keepcoding/practica/amigos.csv")
//contamos los registros que tiene el dataframe
println(df.count())
```

The output of the code is:

```
500
df: org.apache.spark.sql.DataFrame = [_c0: string, _c1: string ... 2 more fields]
```

The execution status is 'FINISHED'.

Below the code and output, it says: 'Took 2 sec. Last updated by anonymous at February 05 2020, 9:39:20 AM.'