Article
Joel Espinoza · Oct 28, 2019 5m read

Cómo ejecutar en conjunto InterSystems IRIS, Apache Spark y Jupyter Notebook

¡Hola a todos!

Hoy instalaremos Jupyter Notebook y vamos a conectarlo con Apache Spark e InterSystems IRIS.

Nota: Los siguientes procedimientos los hice en Ubuntu 18.04 y Python 3.6.5.

Introducción

Si estás buscando un bloc de notas que sea reconocido, difundido ampliamente y muy popular entre los usuarios de Python, en lugar de utilizar **Apache Zeppelin**, deberías elegir **Jupyter notebook**. Jupyter notebook es una excelente y muy poderosa herramienta para la "ciencia de datos", que cuenta con una comunidad muy grande, además de muchas funciones y software adicional. Jupyter notebook permite crear y compartir documentos que contienen código en tiempo real, ecuaciones, visualizaciones y texto narrativo. Sus aplicaciones incluyen la limpieza y transformación de los datos, simulaciones numéricas, modelamiento estadístico, visualización de datos, *machine learning* y muchas funciones más. Y lo más importante, existe una gran comunidad que ayuda a resolver los problemas que surjan.

Requerimientos

Si algo no funciona adecuadamente, consulta la sección "Posibles problemas y soluciones" que se encuentra en la parte inferior del artículo.

En primer lugar, asegúrate de que cuentas con Java 8 (**java -version** devuelve la versión "1.8.x"). A continuación, descarga Apache Spark y descomprímelo. Después, ejecuta las siguientes líneas de comando en el terminal:

```
pip3 install jupyter

pip3 install toree

jupyter toree install --sparkhome=/pathtospark/spark-2.3.1-bin-hadoop2.7 --interpreters=PySpark --user
```

Ahora, abre el terminal y ejecuta **vim ~/.bashrc.** Inserta el siguiente código en la parte inferior (estas son variables de entorno):

```
export JAVAHOME=/usr/lib/jvm/  installed java 8
export PATH="$PATH:$JAVAHOME/bin"
export SPARKHOME=/  path to spark/spark-2.3.1-bin-hadoop2.7
export PATH="$PATH:$SPARKHOME/bin"
export PYSPARKDRIVERPYTHON=jupyter
export PYSPARKDRIVERPYTHONOPTS="notebook"
```

```
File  Edit  View  Search  Terminal  Help
 1 .bashrc +
 # this, if it's already enabled in /etc/bash.bashrc and /etc/profile
 # sources /etc/bash.bashrc).
 if ! shopt -oq posix; then
   if [ -f /usr/share/bash-completion/bash_completion ]; then
     . /usr/share/bash-completion/bash_completion
   elif [ -f /etc/bash_completion ]; then
     . /etc/bash_completion
   fi
 fi

 #my variables

 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
 export PATH="$PATH:$JAVA_HOME/bin"
 export SPARK_HOME=/home/guardian/Desktop/spark-2.3.1-bin-hadoop2.7
 export PATH="$PATH:$SPARK_HOME/bin"
 export PYSPARK_DRIVER_PYTHON=jupyter
 export PYSPARK_DRIVER_PYTHON_OPTS="notebook"
 ~
 ~
 NORMAL   .bashrc   +                                          96%   1
```

Por último, ejecuta **source /.bashrc** . Esto para que cargue las nuevas variables.

Comprueba que funciona adecuadamente

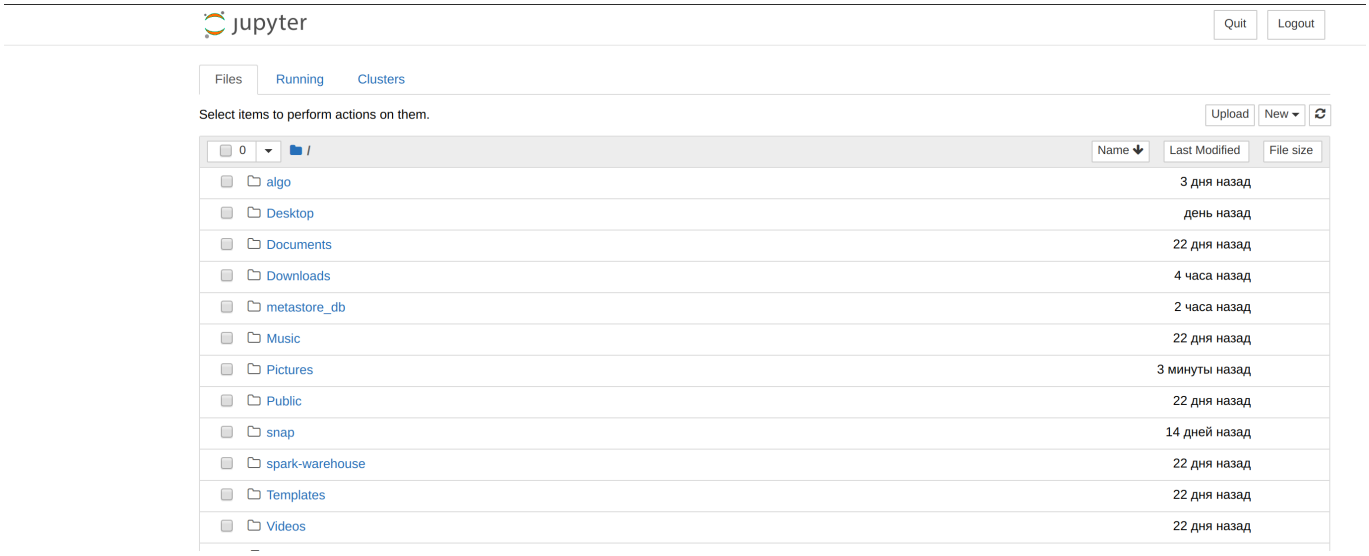
Ahora, vamos a iniciar Jupyter Notebook. Ejecuta **pyspark** en el terminal.

```
File  Edit  View  Search  Terminal  Help

guardian@guardian:~$ pyspark
[I 16:07:42.424 NotebookApp] Serving notebooks from local directory: /home/guard
ian
[I 16:07:42.424 NotebookApp] The Jupyter Notebook is running at:
[I 16:07:42.424 NotebookApp] http://localhost:8888/?token=4d74c58c87b45d7cf2f556
73bbe523a9054c7a574acf3254
[I 16:07:42.424 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
[C 16:07:42.425 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=4d74c58c87b45d7cf2f55673bbe523a9054c7a574ac
f3254
[I 16:07:43.174 NotebookApp] Accepting one-time-token-authenticated connection f
rom 127.0.0.1
```

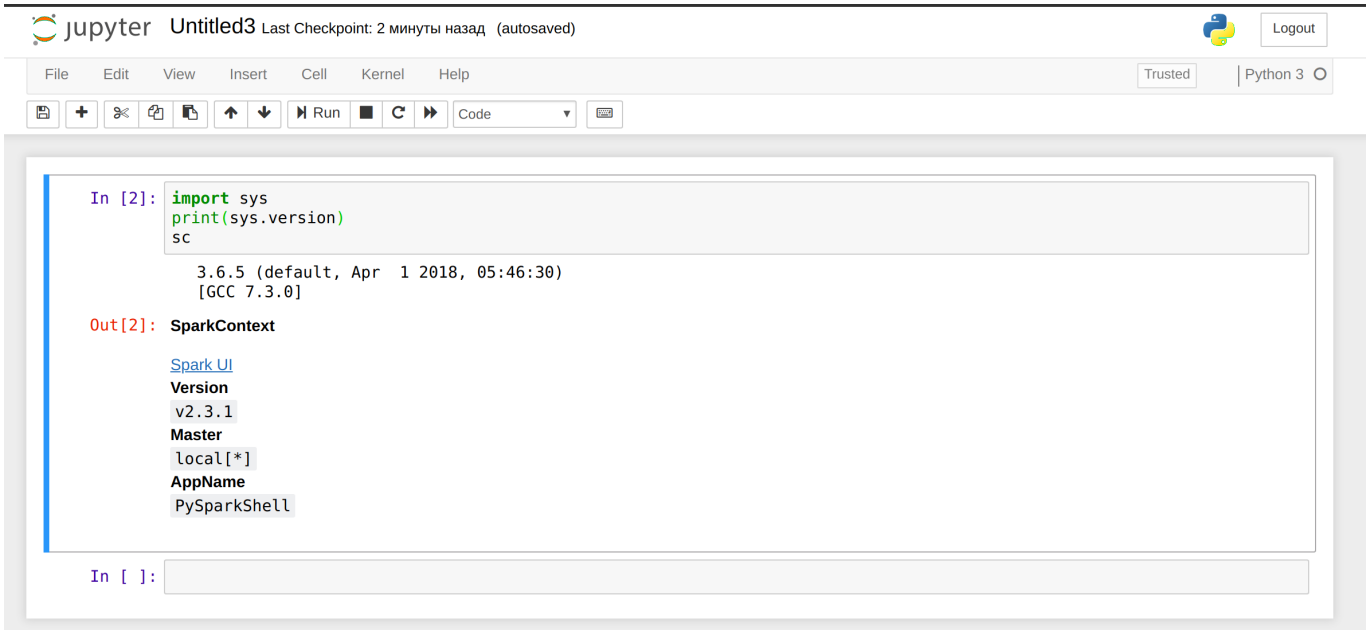
Abre en tu navegador el resultado de la URL. Debería parecerse a esta imagen:



Haz clic en **New**, seleccione **Python 3** e inserta el siguiente código en un párrafo:

```
import sys
print(sys.version)
sc
```

El resultado debería parecerse a esto:



Para Jupyter utilizando Ctrl-c en el terminal.

Nota: Para añadir archivos jar personalizados simplemente mueve los archivos que quieras hacia $SPARKHOME/jars.

Entonces, lo que queremos es trabajar con intersystems-jdbc e intersystems-spark (también necesitaremos una biblioteca jpmml). Vamos a copiar los archivos jar necesarios en Spark. Ejecuta las siguientes líneas de comando en el terminal:

sudo cp /path to intersystems iris/dev/java/lib/JDK18/intersystems-jdbc-3.0.0.jar /path to spark/spark-2.3.1-bin-hadoop2.7/jars

sudo cp /path to intersystems iris/dev/java/lib/JDK18/intersystems-spark-1.0.0.jar /path to spark/spark-2.3.1-bin-hadoop2.7/jars

sudo cp /path to jpmml/jpmml-sparkml-executable-version.jar /path to spark/spark-2.3.1-bin-hadoop2.7/jars

Asegúrate de que funcionan bien. Ejecuta de nuevo **pyspark** en el terminal y también ejecuta el siguiente código (que se mostró en el artículo anterior):

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans
from pyspark.ml import Pipeline
from pyspark.ml.feature import RFormula
from pyspark2pmml import PMMLBuilder

dataFrame=spark.read.format("com.intersystems.spark").\
option("url", "IRIS://localhost:51773/NAMESPACE").option("user", "dev").\
option("password", "123").\
option("dbtable", "DataMining.IrisDataset").load() # load iris dataset

(trainingData, testData) = dataFrame.randomSplit([0.7, 0.3]) # split the data into two sets
assembler = VectorAssembler(inputCols = ["PetalLength", "PetalWidth", "SepalLength", "SepalWidth"],
outputCol="features") # add a new column with features

kmeans = KMeans().setK(3).setSeed(2000) # clustering algorithm that we use

pipeline = Pipeline(stages=[assembler, kmeans]) # First, passed data will run against assembler and after
will run against kmeans.
modelKMeans = pipeline.fit(trainingData) # pass training data

pmmlBuilder = PMMLBuilder(sc, dataFrame, modelKMeans)
pmmlBuilder.buildFile("KMeans.pmml") # create pmml model
```

Este es el resultado que obtuve:

```
In [2]: from pyspark.ml.linalg import Vectors
        from pyspark.ml.feature import VectorAssembler
        from pyspark.ml.clustering import KMeans
        from pyspark.ml import Pipeline
        from pyspark.ml.feature import RFormula
        from pyspark2pmml import PMMLBuilder

        dataFrame=spark.read.format("com.intersystems.spark").\
        option("url", "IRIS://localhost:51773/NEWSAMPLE").option("user", "dev").\
        option("password", "123").\
        option("dbtable", "DataMining.IrisDataset").load() # load iris dataset

        (trainingData, testData) = dataFrame.randomSplit([0.7, 0.3]) # split the data into two sets
        assembler = VectorAssembler(inputCols = ["PetalLength", "PetalWidth", "SepalLength", "SepalWidth"], outputCol="feature

        kmeans = KMeans().setK(3).setSeed(2000) # clustering algorithm that we use

        pipeline = Pipeline(stages=[assembler, kmeans]) # First, passed data will run against assembler and after will run aga
        modelKMeans = pipeline.fit(trainingData) # pass training data

        pmmlBuilder = PMMLBuilder(sc, dataFrame, modelKMeans)
        pmmlBuilder.buildFile("KMeans.pmml") # create pmml model

Out[2]: '/home/guardian/KMeans.pmml'
```

El archivo resultante es un modelo jpmml que se realizó con kmeans. ¡Todo funciona bien!

Posibles problemas y soluciones

- **No se encontró el comando: 'jupyter':**

1. vivim /.bashrc;
2. Añade en la parte inferior **export PATH="$PATH:/.local/bin"** ;
3. En la fuente del terminal /.bashrc.
4. Si esto no soluciona el problema, reinstala pip3 y jupyter.

- **No existe el archivo o directorio env: 'jupyter':**

1. En /.bashrc export PYSPARK_DRIVER_PYTHON=/home/.../.local/bin/jupyter.

- **TypeError: no es posible llamar al objeto 'JavaPackage':**

1. Comprueba que el archivo necesario con la extensión .jar se encuentra en /.../spark-2.3.1-bin-hadoop2.7/jars;
2. Reinicia el bloc de notas.

- **El proceso para la puerta de enlace en Java finalizó antes de que el controlador envíe su número de puerto:**

1. La versión de Java debería ser la 8 (probablemente funciona también con Java 6 o 7, pero no lo he comprobado),
2. **echo $JAVA_HOME** le permitirá conseguir la versión 8 de Java. En caso de que no lo haga, cambia la ruta en /.bashrc.
3. Inserta el comando **sudo update-alternatives --config java** en el terminal y selecciona una versión adecuada de Java.
4. Inserta el comando **sudo update-alternatives --config javac** en el terminal y selecciona una versión adecuada de Java.

- **PermissionError: [Errno 13] Permission denied: '/usr/local/share/jupyter'**

1. Agrega el nombre de usuario al final de tu comando, en el terminal

- **Se produjo un error al ejecutar el comando de Jupyter 'toree': Errno 2], no existe ese archivo o el directorio**

1. Ejecuta el comando sin el sudo.

- **Puede aparecer un error específico si se utilizan variables de sistema como PYSPARK_SUBMIT_ARGS y otras variables de spark/pyspark, o debido a los cambios en /.../spark-2.3.1-bin-hadoop2.7/conf/spark-env.sh**

1. Elimine estas variables y comprueba la configuración de spark-env.sh.

Si todo esta ok, ya tienes tu ambiente Jupyter, Spark e IRIS funcionando!

#API #Compatibility #Best Practices #Beginner #Python #InterSystems IRIS