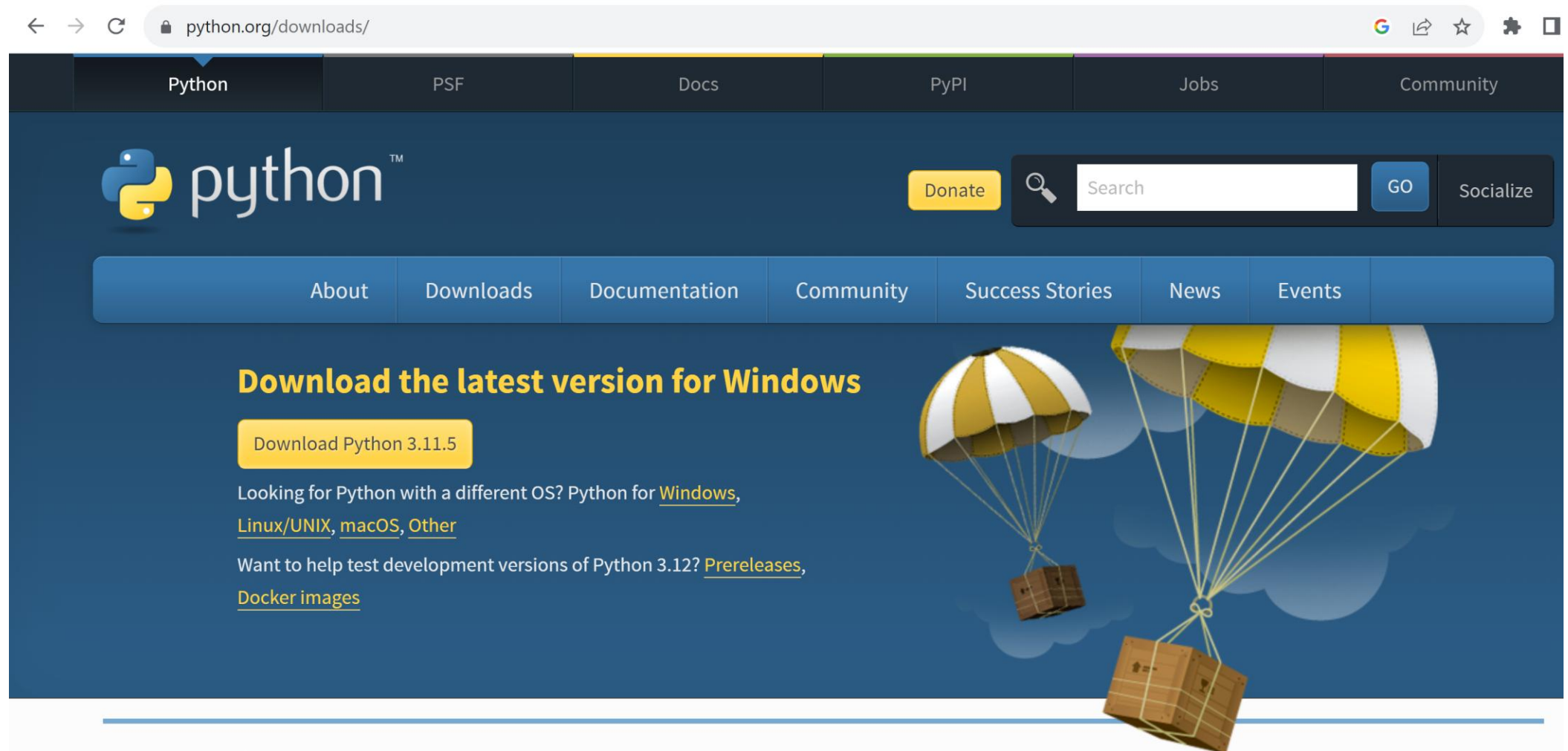


Installing Jupyter Notebook for Spark

Arnaud.nauwynck@gmail.com

Step 1 : install python (simplest method, not using AnaConda)



Step 2 : install jupyter

Choose class Jupyter Notebook... OK



JupyterLab

Install JupyterLab with `pip`:

```
pip install jupyterlab
```

Note: If you install JupyterLab with conda or mamba, we recommend using [the conda-forge channel](#).

Once installed, launch JupyterLab with:

```
jupyter lab
```

Jupyter Notebook

Install the classic Jupyter Notebook with:

```
pip install notebook
```

To run the notebook:

```
jupyter notebook
```

Star Jupyter Notebook

jupyter notebook

```
C:\Users\arnaud>jupyter notebook
[I 2023-09-06 23:09:00.357 ServerApp] Package notebook took 0.0000s to import
[I 2023-09-06 23:09:00.482 ServerApp] Package jupyter_lsp took 0.1187s to import
[W 2023-09-06 23:09:00.482 ServerApp] A `_jupyter_server_extension_points` function was not found in jupyter_lsp.
Instead, a `_jupyter_server_extension_paths` function was found and will be used for now. This function name will
be deprecated in future releases of Jupyter Server.
[I 2023-09-06 23:09:00.544 ServerApp] Package jupyter_server_terminals took 0.0598s to import
[I 2023-09-06 23:09:00.544 ServerApp] Package jupyterlab took 0.0000s to import
```

[[truncated logs Also contains errors??]]

Jupyter Notebook ..

```
[I 2023-09-06 23:09:01.879 ServerApp] jupyterlab | extension was successfully loaded.  
[I 2023-09-06 23:09:01.879 ServerApp] notebook | extension was successfully loaded.  
[I 2023-09-06 23:09:01.895 ServerApp] Serving notebooks from local directory: C:\Users\arnaud  
[I 2023-09-06 23:09:01.895 ServerApp] Jupyter Server 2.7.3 is running at:  
[I 2023-09-06 23:09:01.895 ServerApp] http://localhost:8888/tree?token=a9195d7e950f42e4e26a7b93bbebf3664d1794b94959c5e3  
[I 2023-09-06 23:09:01.911 ServerApp] http://127.0.0.1:8888/tree?token=a9195d7e950f42e4e26a7b93bbebf3664d1794b94959c5e3  
[I 2023-09-06 23:09:01.911 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).  
[C 2023-09-06 23:09:01.974 ServerApp]
```

To access the server, open this file in a browser:

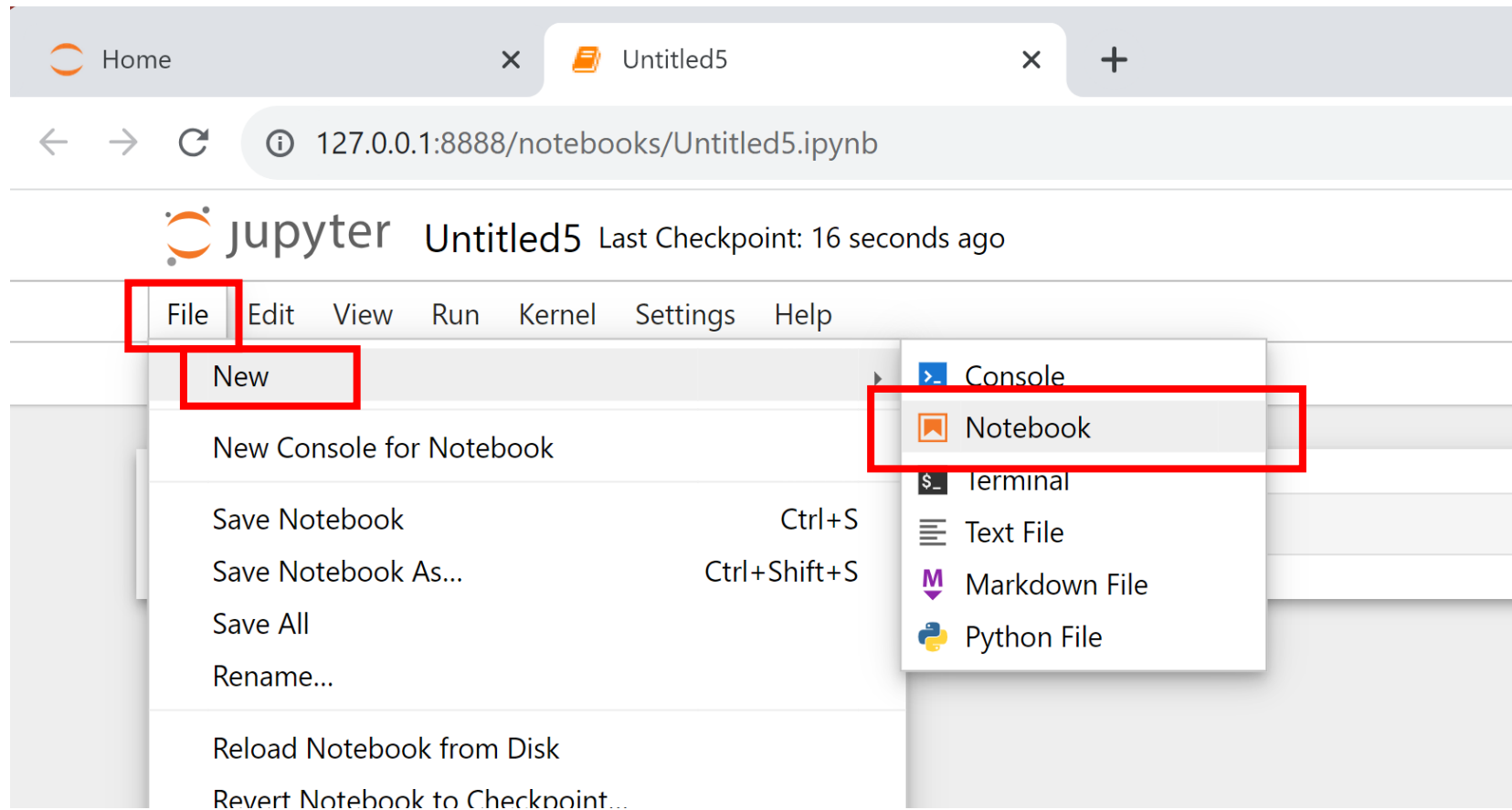
`file:///C:/Users/arnaud/AppData/Roaming/jupyter/runtime/jpserver-636-open.html`

Or copy and paste one of these URLs:

`http://localhost:8888/tree?token=a9195d7e950f42e4e26a7b93bbebf3664d1794b94959c5e3`

`http://127.0.0.1:8888/tree?token=a9195d7e950f42e4e26a7b93bbebf3664d1794b94959c5e3`

Creating a new notebook



Step 3: jupyter needs kernels

← → ↺ docs.jupyter.org/en/stable/projects/kernels.html



Try Jupyter Usage Projects Community Contributing More ▾

Section Navigation

Jupyter User Interfaces

Kernels (Programming Languages)

Education

Execution

Deployment and infrastructure

Formatting and Conversion

IPython

Core Building Blocks

Incubator Projects

Architecture

Project Documentation

Release Notes

🏠 > Projects > Kernels...

Kernels (Programming Languages)

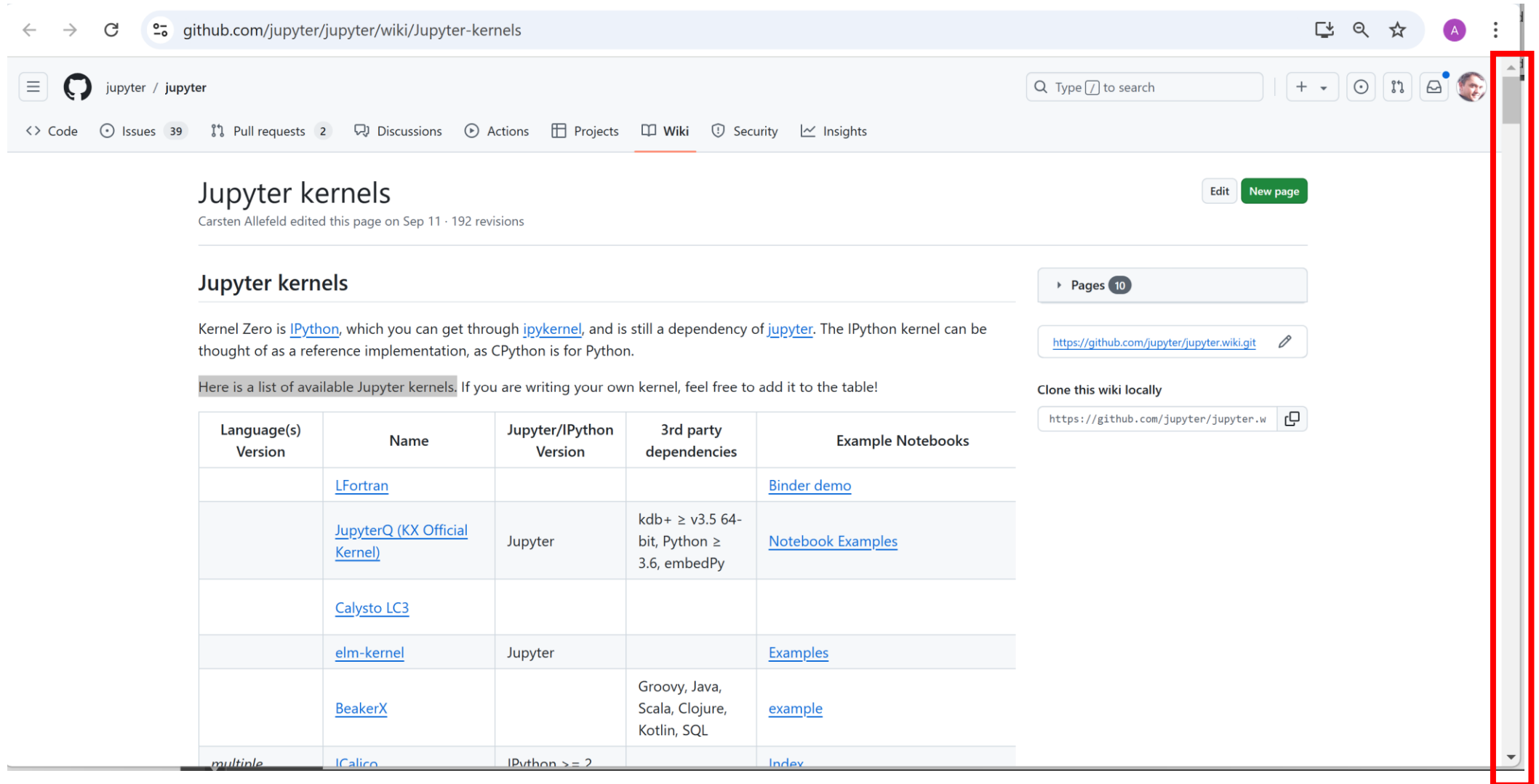
The Jupyter team maintains the [IPython](#) project which is shipped as a default kernel (as [ipykernel](#)) in a number of Jupyter clients. Many other languages, in addition to Python, may be used in the notebook.

The community maintains many other language kernels, and new kernels become available often. Please see the [list of available kernels](#) for additional languages and [kernel installation instructions](#) to begin using these language kernels.

Kernels

Kernels are *programming language specific* processes that run independently and interact with the Jupyter Applications and their user interfaces. [ipykernel](#) is the reference Jupyter kernel built on top of [IPython](#), providing a powerful environment for interactive computing in Python.

Step 3: list of community jupyter-kernels



github.com/jupyter/jupyter/wiki/Jupyter-kernels

jupyter / jupyter

Code Issues 39 Pull requests 2 Discussions Actions Projects Wiki Security Insights

Jupyter kernels

Carsten Allefeld edited this page on Sep 11 · 192 revisions

Kernel Zero is [IPython](#), which you can get through [ipykernel](#), and is still a dependency of [jupyter](#). The IPython kernel can be thought of as a reference implementation, as CPython is for Python.

Here is a list of available Jupyter kernels. If you are writing your own kernel, feel free to add it to the table!

Language(s) Version	Name	Jupyter/IPython Version	3rd party dependencies	Example Notebooks
	LFortran			Binder demo
	JupyterQ (KX Official Kernel)	Jupyter	kdb+ ≥ v3.5 64-bit, Python ≥ 3.6, embedPy	Notebook Examples
	Calysto LC3			
	elm-kernel	Jupyter		Examples
	BeakerX		Groovy, Java, Scala, Clojure, Kotlin, SQL	example
multiple	Calico	IPython >= 2		Index

Pages 10

<https://github.com/jupyter/jupyter.wiki.git>

Clone this wiki locally

<https://github.com/jupyter/jupyter.w>

Jupyter Kernels "*spark*"

Java 11+, Groovy , Javascript , Kotlin , Scala , Apache Spark , and more	Ganymede	Jupyter >= 4.0	JShell , Apache Maven Resolver	Examples
Pyspark (Python 2 & 3), Spark (Scala), SparkR (R)	sparkmagic	Jupyter >=4.0	Livy	Notebooks , Docker Images
Scala, Python, R	Apache Toree (formerly Spark Kernel)	Jupyter	Spark >= 1.5	Example
Scala >= 2.10	almond (old name: Jupyter-scala)	IPython >= 3.0		examples
Python >= 3.5, scala >= 2.11	spylon-kernel	ipykernel >= 4.5	Apache Spark >= 2.0	Example

Jupyter Kernel(s)

to execute
python
>>> 1+1



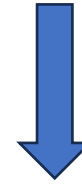
ipykernel (built-in)

to execute
spark python code
>>> spark.sql("...")



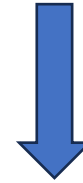
pyspark ...

to execute
spark scala code
scala> spark.sql("...")



spylon kernel
(deprecated?)

to execute
scala (no spark)
scala> 1+1

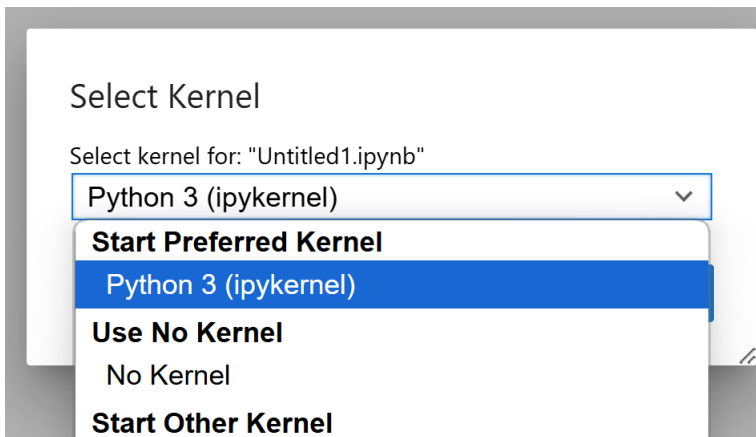


almond kernel

to execute
spark scala code
scala> spark.sql("...")



almond kernel
+ module "**almonnd-spark**"



Step 3 alternative : almond kernel
+ "almond-spark" module

for using spark with scala language

http://almond.sh



almond

A Scala kernel for Jupyter

[TRY IT ONLINE](#)[TRY IT WITH DOCKER](#)[INSTALL](#)

TODO ... WORK IN PROGRESS

Step 3 alternative : spylon kernel
for spark + scala

Step 3 alternative : install jupyter spylon_kernel
(for using Spark with Scala language)

python -m spylon_kernel install --user

(or `pip install spylon_kernel --user`)

```
C:\Users\arnaud>python -m spylon_kernel install --user
0.00s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
[InstallKernelSpec] Installed kernelspec spylon-kernel in C:\Users\arnaud\AppData\Roaming\jupyter\kernels\spylon-k
ernel
```

Step 3 ... testing spylon



Trusted

JupyterLab Python 3 (ipykernel)

Select Kernel

Select kernel for: "Untitled6.ipynb"

Python 3 (ipykernel)

Start Preferred Kernel
Python 3 (ipykernel)

Use No Kernel
No Kernel

Start Other Kernel
spylon-kernel

Use Kernel from Preferred Session

Use Kernel from Other Session
Untitled4.ipynb
Untitled5.ipynb
Untitled6.ipynb

Trusted

JupyterLab spylon-kernel

Step 3 ... Testing Spylon .. Write Scala

The diagram illustrates the steps to run Scala code in a Databricks notebook:

- Code Cell:** A code cell containing the following Scala code:

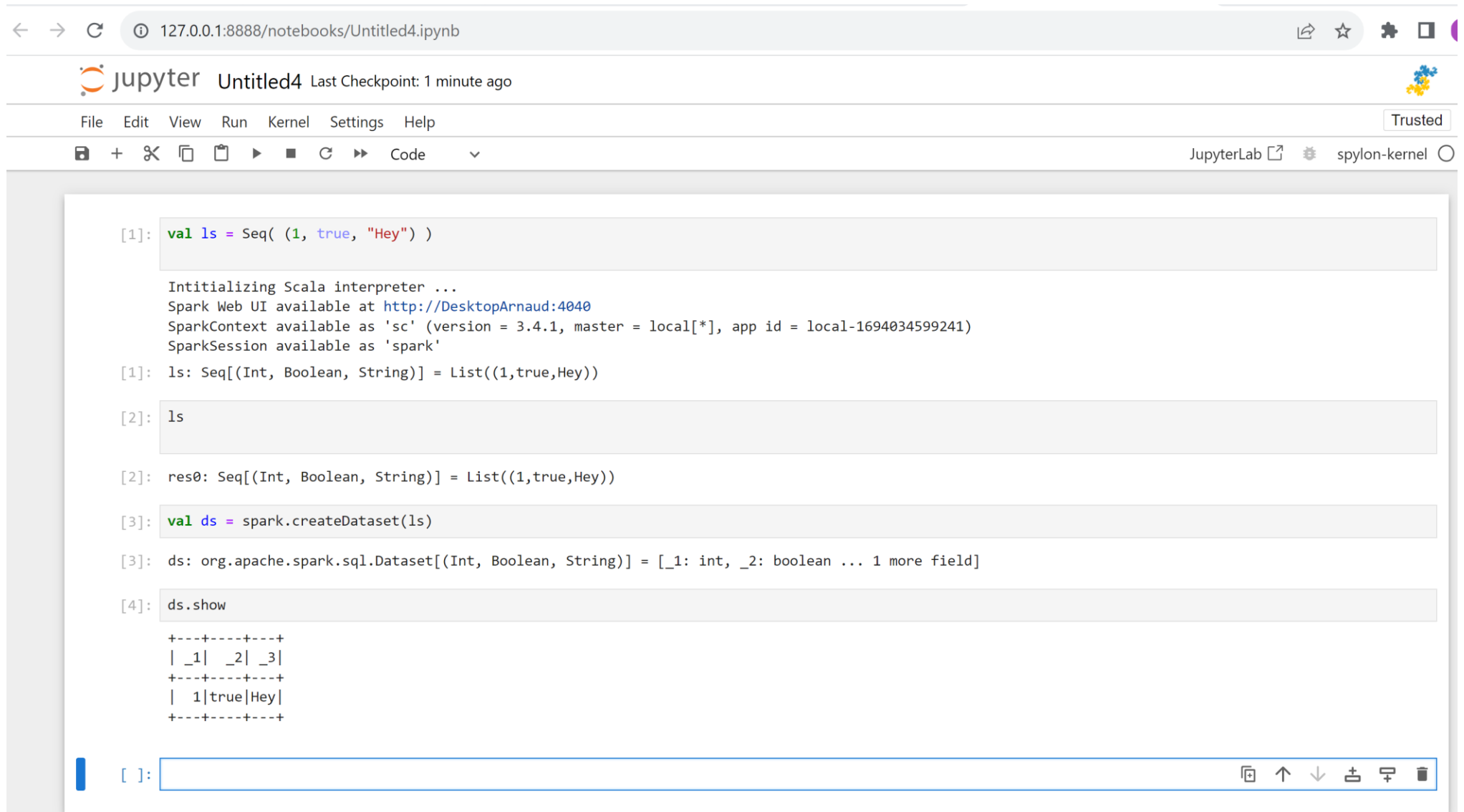
```
[1]: for(i <- 0 to 5) println(s"Scala code.. ${i}")
```
- Run Button:** A button labeled "Run this cell and advance (Shift+Enter)" is shown next to the code cell.
- Output:** The output of the code cell is displayed below the code, showing the following text:

```
[1]: for(i <- 0 to 5) println(s"Scala code.. ${i}")  
Intitializing Scala interpreter ...  
Spark Web UI available at http://DesktopArnaud:4041  
SparkContext available as 'sc' (version = 3.4.1, master = local[*], app id = local-1694035899154)  
SparkSession available as 'spark'  
Scala code.. 0  
Scala code.. 1  
Scala code.. 2  
Scala code.. 3  
Scala code.. 4  
Scala code.. 5
```

Annotations and instructions:

- A red arrow points from the text "Type SCALA code ... Shift+Enter" to the "Run this cell and advance (Shift+Enter)" button.
- A red arrow points from the text "Get Spark process (Scala interpreter)" to the "Intitializing Scala interpreter ..." output line.

Step 3 ... Testing Spylon .. Write Scala



The screenshot shows a JupyterLab notebook titled 'Untitled4' with a 'Trusted' security status. The browser address bar shows the URL '127.0.0.1:8888/notebooks/Untitled4.ipynb'. The notebook interface includes a top menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. Below the menu is a toolbar with icons for file operations and execution. The main area contains a Scala code cell with the following content:

```
[1]: val ls = Seq( (1, true, "Hey") )

Initializing Scala interpreter ...
Spark Web UI available at http://DesktopArnaud:4040
SparkContext available as 'sc' (version = 3.4.1, master = local[*], app id = local-1694034599241)
SparkSession available as 'spark'

[1]: ls: Seq[(Int, Boolean, String)] = List((1,true,Hey))

[2]: ls

[2]: res0: Seq[(Int, Boolean, String)] = List((1,true,Hey))

[3]: val ds = spark.createDataset(ls)

[3]: ds: org.apache.spark.sql.Dataset[(Int, Boolean, String)] = [_1: int, _2: boolean ... 1 more field]

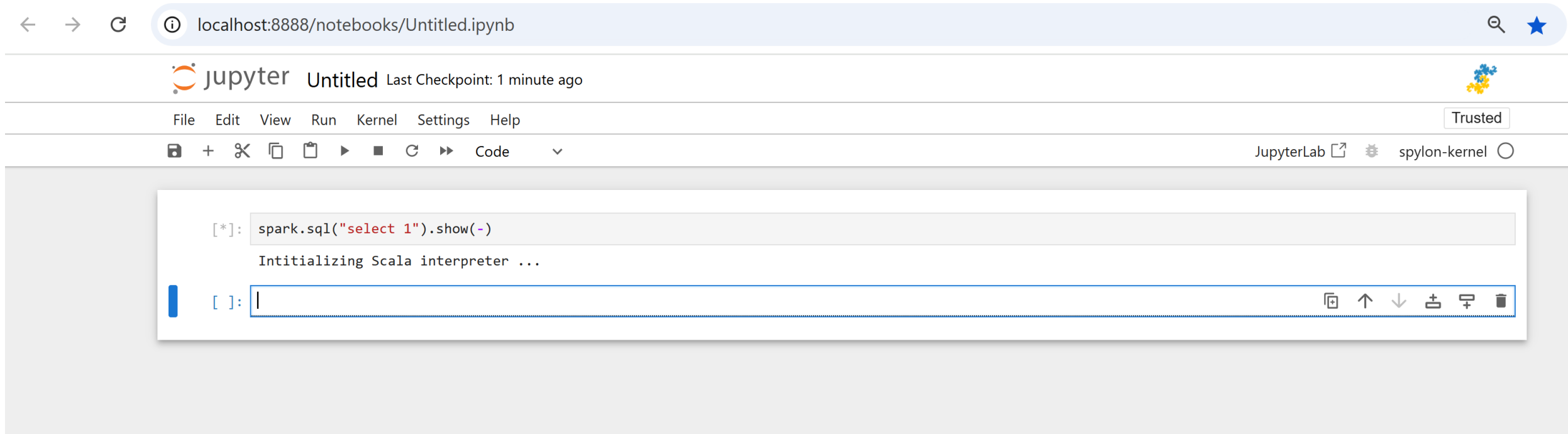
[4]: ds.show
```

The output of the `ds.show` command is displayed as a table:

_1	_2	_3
1	true	Hey

The bottom of the notebook shows a new code cell with the prompt `[]:` and a toolbar with icons for copy, paste, and other actions.

Spark SQL "Hello World"



The screenshot displays a JupyterLab web interface in a browser. The address bar shows the URL `localhost:8888/notebooks/Untitled.ipynb`. The JupyterLab header includes the logo, the notebook title "Untitled", and a status message "Last Checkpoint: 1 minute ago". A "Trusted" badge is visible on the right. The main menu bar contains "File", "Edit", "View", "Run", "Kernel", "Settings", and "Help". Below this is a toolbar with icons for saving, adding, deleting, and running code. The central area shows a code cell with the following content:

```
[*]: spark.sql("select 1").show(-)
      Intitalizing Scala interpreter ...
```

Below the code cell is an empty input field for the next cell, with a cursor at the start. To the right of the input field is a small toolbar with icons for undo, redo, and other actions. The bottom of the interface shows the "JupyterLab" logo and the kernel name "spylon-kernel".

may take 1 minute to start ... spark is starting

TroubleShooting ???

```
Cmder
eco
    return f(*a, **kw)
           ^^^^^^^^^^^
File "C:\apps\spark\spark-3.5.0\python\lib\py4j-0.10.9.7-src.zip\py4j\protocol.py", line 330,
in get_return_value
    raise Py4JError(
py4j.protocol.Py4JError: An error occurred while calling None.scala.tools.nsc.interpreter.IMain
. Trace:
py4j.Py4JException: Constructor scala.tools.nsc.interpreter.IMain([class scala.tools.nsc.Settin
gs, class java.io.PrintWriter]) does not exist
    at py4j.reflection.ReflectionEngine.getConstructor(ReflectionEngine.java:180)
    at py4j.reflection.ReflectionEngine.getConstructor(ReflectionEngine.java:197)
    at py4j.Gateway.invoke(Gateway.java:237)
    at py4j.commands.ConstructorCommand.invokeConstructor(ConstructorCommand.java:80)
    at py4j.commands.ConstructorCommand.execute(ConstructorCommand.java:69)
    at py4j.ClientServerConnection.waitForCommands(ClientServerConnection.java:182)
    at py4j.ClientServerConnection.run(ClientServerConnection.java:106)
    at java.base/java.lang.Thread.run(Thread.java:1589)

[I 2024-10-26 09:35:00.917 ServerApp] Saving file at /Untitled.ipynb
|
```

Step 3 alternative : kernel for pyspark
(for using PySpark with Python language)