

3.3 Notebooks Analíticos

[Concepto y alternativas. ¿Por qué Jupyter?](#)

[Reproducibilidad de resultados y método científico](#)

[El Método Científico en el Data Science](#)

[Reproducibilidad de resultados](#)

[Nuevas herramientas para Data Science: Notebooks](#)

[Concepto](#)

[Jupyter](#)

[Instalación](#)

[Instalación de Jupyter \(Classic\)](#)

[conda](#)

[pip](#)

[Instalación de Jupyter \(Lab\)](#)

[conda](#)

[pip](#)

[Introducción y uso de Jupyter](#)

[Jupyter Server](#)

[Funcionalidades](#)

[Utilización de los Notebooks](#)

[Utilización con diferentes lenguajes](#)

[R en Jupyter Notebooks](#)

[JupyterLab](#)

[Instalación](#)

[Lanzamiento de JupyterLab](#)

[Test de Conocimientos](#)

[Ejercicio práctico](#)

Concepto y alternativas. ¿Por qué Jupyter?

Reproducibilidad de resultados y método científico

El Método Científico en el Data Science

Desde que comienza a generalizarse el concepto de Data Science alrededor del año 2015, este ha estado muy ligado al método científico. Tomando la definición original de [Wikipedia](#):

*Data science is a multidisciplinary field that uses **scientific methods**, processes, algorithms and system to extract knowledge and insights from structured and unstructured data. Data science is the same concept as data mining and big data: "use the most powerful hardware, the most powerful programming systems, and the most efficient algorithms to solve problems"*

El método científico se basa principalmente en plantear hipótesis y comprobar su veracidad utilizando para ello experimentos. Este enfoque metodológico del que se hablará más adelante en el curso provoca que surjan herramientas de trabajo específicas que difieren de aquellas herramientas clásicas que se utilizan en Desarrollo de Software.

Puede decirse que en el caso del Data Science, el uso de la programación y el desarrollo de software es un medio para alcanzar un objetivo (la extracción de valor de los datos) y no el fin en sí mismo (desarrollar aplicaciones de software) durante la mayor parte del proceso.

Reproducibilidad de resultados

El Data Science se basa en la extracción de valor de los datos. Para que esta extracción de valor se produzca, es necesario entender muy bien los métodos utilizados para el desarrollo de las soluciones que se proponen en Data Science.

Si bien al desarrollar aplicaciones al uso, entender el procedimiento para añadir una nueva funcionalidad es relativamente sencillo, en el caso de las soluciones que incorporan el uso de Datos, la complejidad inherente a las soluciones hace necesarias explicaciones más detalladas de los procedimientos y los algoritmos utilizados.

La reproducibilidad de los resultados se basa en que las hipótesis obtenidas en base a los datos mediante la realización de experimentos deben poder explicarse y comprenderse por parte de otros profesionales del Data Science e idealmente también por perfiles ejecutivos con capacidad de tomar decisiones. Para obtener esta reproducibilidad de los resultados es interesante explorar el uso de herramientas que permitan al Científico de Datos combinar la explicación de la metodología seguida para llegar al resultado, con la propia implementación de la solución, y que además permite compartir el resultado de forma sencilla.

Nuevas herramientas para Data Science: Notebooks

Concepto

En lugar de los IDE tradicionales, surgen paralelamente unos entornos de trabajo llamados ‘Notebooks’ con las siguientes características:

- Permiten intercalar código con texto e imágenes.
- Permiten generar reportes en .html o .pdf en los que se describe un proceso de forma que estos sean fáciles de compartir.
- Permiten escribir expresiones matemáticas utilizando [LaTeX](#).
- Permiten desarrollar de forma iterativa, ejecutando fragmentos de código de forma modular, obteniendo un output.

Jupyter

[Jupyter](#) es el gran proyecto de Notebook analítico y de código abierto, el más maduro y el que ha sentado los estándares de computación interactiva.

Existen otras alternativas como [Apache Zeppelin](#), con interesante soporte para el lenguaje SQL, y muy similares a Jupyter, aunque en general con menos funcionalidades y menos adopción por parte del mundo corporativo.

Aquellas empresas que optan por soluciones diferentes a Jupyter para computación interactiva, lo hacen principalmente por algún tipo de vendor lock-in, o porque ya venga incluido con software establecido en la compañía, como es el caso de Zeppelin, que se encuentra incorporado en la distribución de Hadoop de Hortonworks.

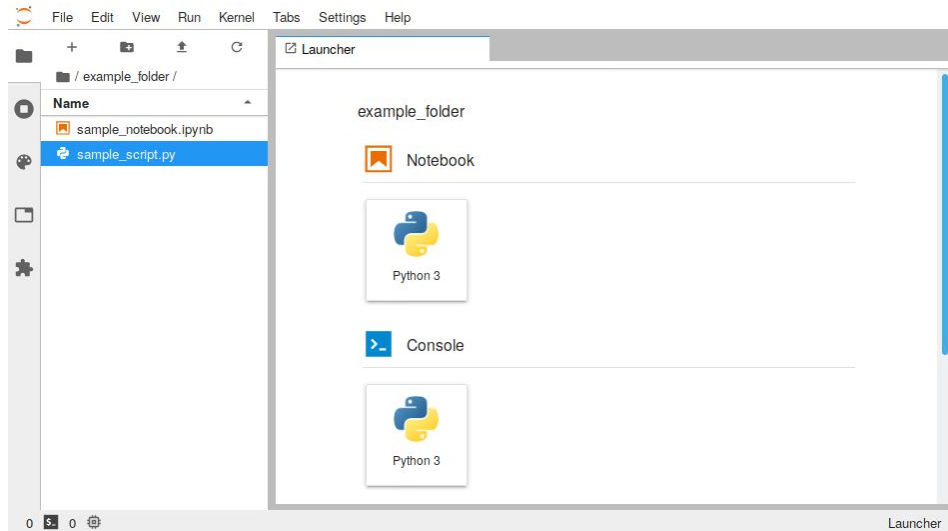
Instalación

Existen varias versiones diferentes de Jupyter:

- Clásica: la versión más madura y extendida. Incluye todas las funcionalidades de Jupyter con un front-end clásico.



- Lab: la versión más reciente. Incluye todas las funcionalidades de Jupyter con un front-end renovado más moderno y amigable para el usuario. Gran parte de la nueva funcionalidad de este front-end se basa en Javascript, por lo que es necesario además instalar Node.js para poder activar ciertas funcionalidades avanzadas que se describirán en apartados posteriores de este documento.



Instalación de Jupyter (Classic)

conda

Para instalar Jupyter en su versión clásica desde conda:

- `conda install jupyter`

Junto con Jupyter se instalarán una serie de requerimientos (dependencias) necesarias para poder utilizar y gestionar Jupyter. La lista completa de estos requerimientos se sale del alcance de este curso y no es necesaria.

pip

Para instalar Jupyter en su versión clásica desde pip:

- `pip install jupyter`

De la misma forma que en el caso de conda, pip instalará aquellas dependencia que considere oportunas.

Instalación de Jupyter (Lab)

conda

Para instalar Jupyter en su versión Lab desde conda, **activando previamente el entorno donde se quiera instalar JupyterLab**:

- `conda install jupyterlab nodejs`

Es necesario instalar también Node.js al no ser un requerimiento obligatorio de Jupyter pero muy recomendable para poder instalar extensiones.

pip

Para instalar Jupyter en su versión Lab desde pip:

- `pip install jupyterlab`

En el caso de pip, la instalación de Node.js no es tan sencilla como desde conda (una de las ventajas de conda es la posibilidad de instalar software que no ha de ser basado necesariamente en Python, como Java, Javascript, R, etc.). La instalación de [Node.js](#) se propone realizarla de forma manual.

Introducción y uso de Jupyter

Jupyter Server

Jupyter es una aplicación web a la que **se accede desde un navegador**. Para poder acceder, es necesario arrancar un servidor de Jupyter en un equipo. Se recomienda que el servidor se lance en un equipo que tenga suficiente potencia ya que hay muchos procesos de Jupyter que se ejecutan en local, y **visibilidad a los datos**, por ejemplo en uno de los nodos maestros de un clúster Big Data.

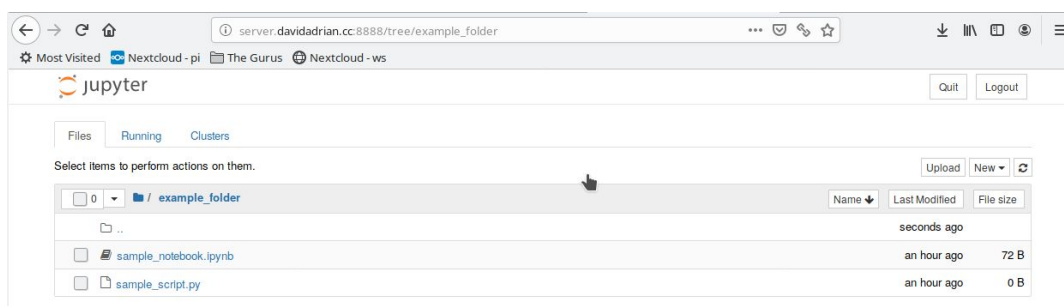
Para lanzar un servidor de Jupyter, se deberá utilizar el siguiente comando, activando previamente el entorno virtual en el caso de que se haya instalado dentro de uno:

- `jupyter-notebook --ip=XX.XX.XX.XX --port=XXXX --no-browser`

Donde la ip será la ip del equipo en cuestión, que tendrá que ser accesible sobre el resto de equipos de la red que vayan a conectarse al servidor como clientes. El puerto, por convenio se suele tomar el 8888 como referencia y asignar los que se encuentran cercanos al mismo, teniendo cuidado de no colisionar con otras aplicaciones.

Funcionalidades

- **Navegación:** existe la posibilidad de navegar por el árbol de directorios aguas abajo de la ruta donde se haya lanzado el servidor, mover archivos, descargarlos al equipo cliente, subir archivos desde el equipo cliente, etc. Esto ahorra tiempo al evitar tener que utilizar la terminal constantemente para subir y descargar archivos desde un servidor remoto, y además facilita este tipo de tareas a aquellos usuarios que no tienen un perfil muy técnico.



- **Text Editor:** Jupyter ofrece un editor de texto básico que permite crear y editar código y archivos de texto plano. Está más orientado a la creación de pequeños scripts y la consulta de archivos de configuración que al desarrollo al no tener capacidades de autocompletado.

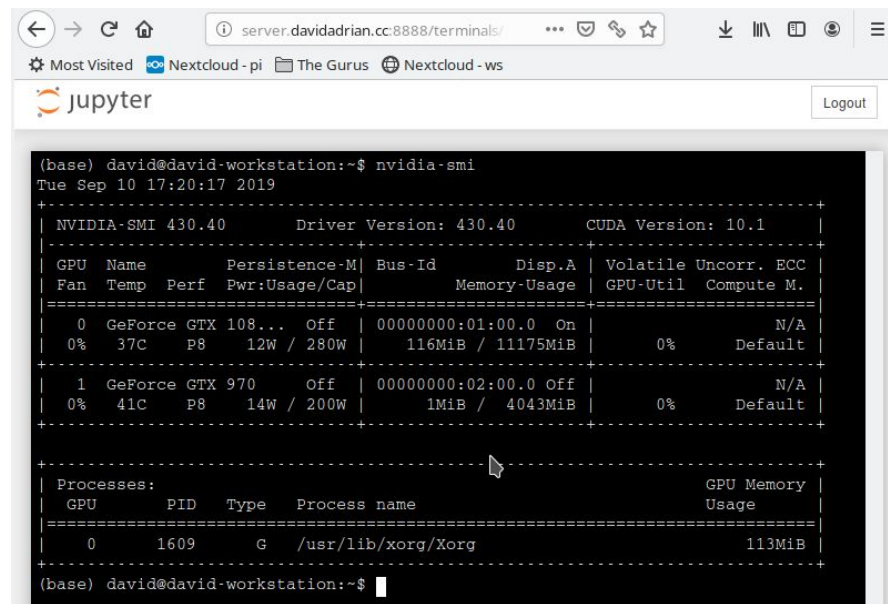


```

1 # This is the Jupyter text editor
2 # It includes syntax highlighting
3
4 def square(x):
5     return x ** 2
6
7 if __name__ == '__main__':
8     squares = [square(x) for x in range(10)]
9     print(squares)
10
11

```

- **Terminal:** Jupyter ofrece conexión ssh con el equipo en el que se ha lanzado el servidor con un único click ofreciendo una terminal con funcionalidad completa.



```

(base) david@david-workstation:~$ nvidia-smi
Tue Sep 10 17:20:17 2019

+-----+
| NVIDIA-SMI 430.40          Driver Version: 430.40          CUDA Version: 10.1          |
+-----+-----+
| GPU Name      Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|=====+=====+
| 0  GeForce GTX 108...    Off | 00000000:01:00.0 On    |          N/A         |
| 0%   37C    P8     12W / 280W | 116MiB / 11175MiB |      0%    Default  |
+-----+-----+
| 1  GeForce GTX 970      Off | 00000000:02:00.0 Off   |          N/A         |
| 0%   41C    P8     14W / 200W | 1MiB / 4043MiB |      0%    Default  |
+-----+-----+

+-----+
| Processes:                                                       GPU Memory |
|  GPU       PID    Type   Process name                               Usage      |
|=====+=====+
|    0       1609     G   /usr/lib/xorg/Xorg                               113MiB      |
+-----+

(base) david@david-workstation:~$

```

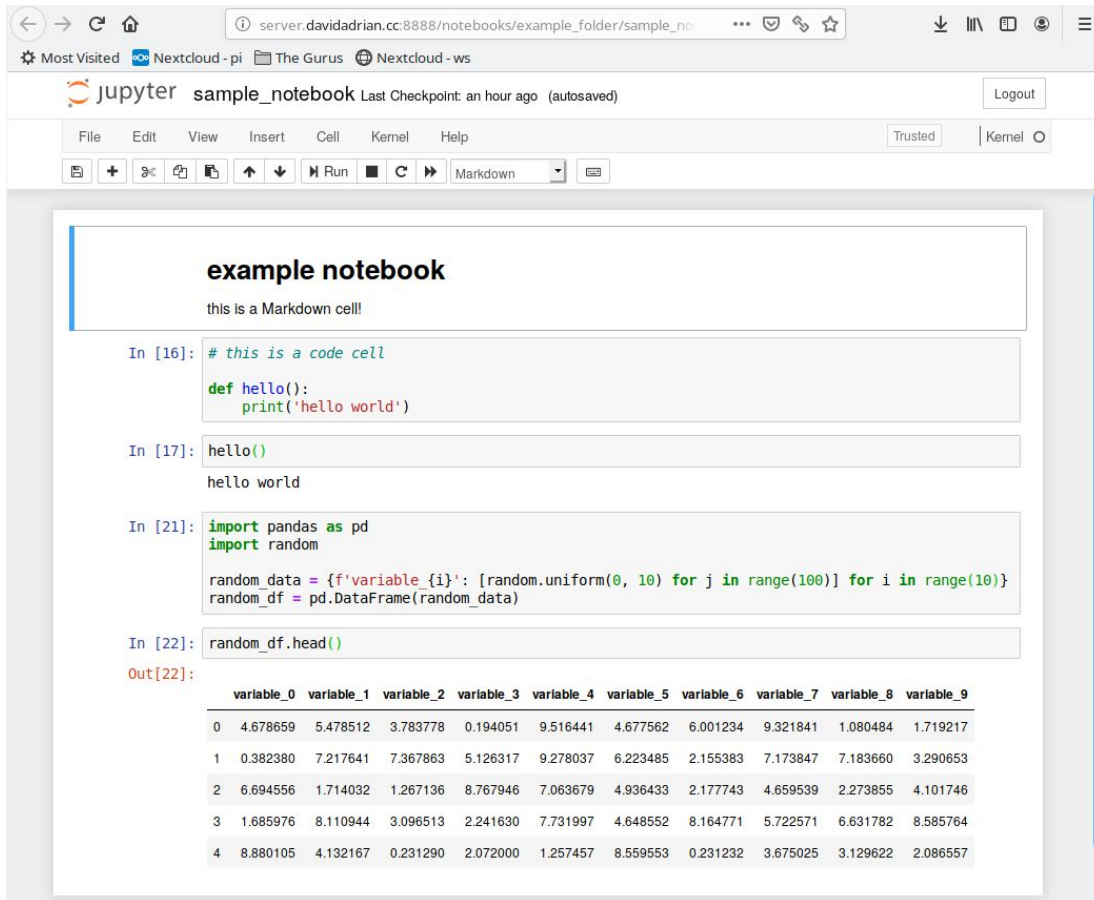
- **Notebooks:** La funcionalidad de creación de Notebooks es la más importante de Jupyter. Los Notebooks se comentarán más en detalles en apartados posteriores de este documento.

Utilización de los Notebooks

Los notebooks se componen de unos elementos llamados **celdas**. Las celdas son unidades modulares con diferente funcionalidad:

- Celdas de **código**: las celdas de código, como su nombre indica, contienen código que es ejecutado de forma conjunta. Esto permite estructurar de forma modular el código intercalando otro tipo de celdas.

- Celdas de **Markdown**: las celdas de Markdown son celdas de texto ideales para separar secciones dentro del Notebook y para intercalar explicaciones. En las celdas de Markdown se pueden intercambiar expresiones matemáticas en LaTeX.
- Otros tipos de celdas: existen otros tipos de celdas que o bien se encuentran deprecadas, o bien su uso escapa fuera del alcance de este tutorial.



Utilización con diferentes lenguajes

La opción por defecto para utilizar Jupyter es utilizando Python, pero no es la única opción disponible. Existen multitud de lenguajes que pueden incorporarse para utilizarse de forma interactiva, como por ejemplo (muy importantes en el ecosistema Data):

- R
- Scala
- C++
- Otras versiones de Python (por ejemplo Python 2)

A continuación se explica cómo añadir soporte para estos otros lenguajes en Jupyter.

R en Jupyter Notebooks

Para poder utilizar R integrado en Jupyter, se explica a continuación cómo realizar esta integración utilizando entornos virtuales de conda.

En primer lugar, se creará un entorno virtual nuevo donde se realizará la instalación de R, en este caso se le llamará `r_env` por claridad:

- `conda create -n r_env`

A continuación se activará el entorno:

- `conda activate r_env`

Se instala R en el entorno:

- `conda install r-base r-essentials`

Se instala el kernel de R en el entorno (para permitir la integración con el sistema de kernels de Jupyter):

- `conda install r-irkernel`

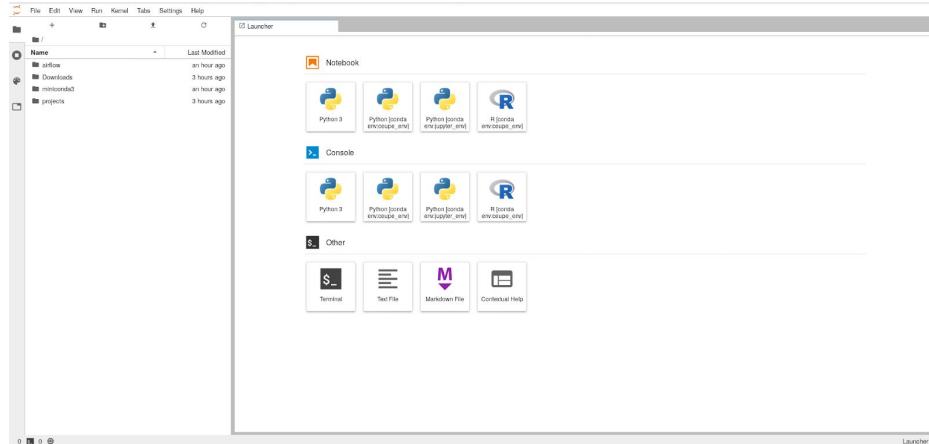
A continuación, se activará el entorno en el que se encuentra instalado jupyter, llamado `jupyter_env` en esta guía por claridad. Antes, se desactivará el entorno de R:

- `conda deactivate`
- `conda activate jupyter_env`

Se instalará la librería `nb_conda_kernels`, que permite que Jupyter detecte de forma automática los kernels de diferentes lenguajes en todos los entornos virtuales existentes en la instalación de conda:

- `conda install nb_conda_kernels`

Una vez hecho esto, tendremos disponible R como lenguaje de programación en Jupyter:



JupyterLab

JupyterLab es la nueva versión de Jupyter, en la que se están centrando la mayoría de los esfuerzos de desarrollo actualmente. Es una versión con un front-end más moderno y que incorpora todas las funcionalidades de la versión clásica de Jupyter.

Es mucho más flexible que la versión clásica de Jupyter, y su principal utilidad para el usuario es la posibilidad de poder distribuir las diferentes secciones (editor de texto, notebook, terminal, visualizador de imágenes, etc.) de la forma deseada:

The screenshot shows the JupyterLab interface with three main panels:

- Files Panel (Left):** Shows a file browser with 'example.ipynb' and 'usa_major_cities.csv'.
- Code Editor (Center):** Contains Python code for visualizing data on a map. The code includes comments and a final plot command.
- Console (Center):** Shows the output of the code, including a table of data for 'usa_major_cities.csv'.
- Map Panel (Right):** Displays a map of the United States with major cities marked. Below the map is a table of population data for various age groups.

Code in the Editor:

```
Visualizing classes with different colors

Often, you may want to classify the numerical values in your data into groups and
visualize them on a map. You can accomplish this with a class break renderer which
splits your data into specific number of groups and uses color to differentiate each
group. You can choose the algorithm that performs the class splits or go with the
default.

Let us visualize the same major cities point dataset using its POPULATION column.

In [12]: df[['ST', 'NAME', 'POPULATION']].head()
Out[12]:
```

	ST	NAME	POPULATION
0	AZ	Somerton	14980
1	CA	Anderson	10239
2	CA	Camp Pendleton South	11869
3	CA	Citrus	11195
4	CA	Commerce	13009

```
In [51]: m3_ua = gis.map('Reno, NV', zoomLevel=4)
m3_ua

In [50]: df.spatial.plot(map_widget=m3_ua,
                        renderer_type='c', # for class breaks renderer
                        method='esriClassifyNaturalBreaks', # classification algor
                        class_count=10, # choose the number of classes
                        col='POPULATION', # numeric column to classify
                        cmap='prism', # color map to pick colors from for each cla
                        alpha=0.7 # specify opacity
)
Out[50]: True
```

Map Panel Data (Population):

	AGE_10_14	AGE_15_19	AGE_20_24	AGE_25_34	AGE_35_44
1	1412	1381	1108	2136	1815
2	727	738	677	1380	1185
3	593	511	2323	2767	746
4	888	988	900	1729	1479
5	1086	1228	1013	1822	1759
6	1078	1085	812	1545	1479
7	234	281	904	2937	1738
8	904	863	720	1363	1233
9	1033	1090	838	1590	1351
10	893	975	845	1550	1358
11	1076	1118	952	1707	1651
12	142	7696	11972	1696	380
13	1186	1216	1002	2154	1716
14	1033	1201	1306	2211	1726
15	14	14	24	44	113

La contrapartida principal de JupyterLab a día de hoy, es que hay muchas librerías de visualización basadas en Javascript cuyo output se encuentra bloqueado en JupyterLab por motivos de seguridad, pudiendo solucionarse estos problemas instalando extensiones o pasando directamente a la versión clásica.

Instalación

Para su correcto funcionamiento y para poder instalar extensiones llegado el caso, es necesario instalar Node.js en el entorno. Por suerte, Node.js se encuentra incluido en los repositorios de conda. Para crear un entorno completo con Jupyter y JupyterLab:

En primer lugar, se crea un nuevo entorno de conda. Se llamará `jupyter_env` por claridad:

- `conda create -n jupyter_env`

A continuación, se activa el entorno y se realiza la instalación de JupyterLab (ya incluye Jupyter) y Node.js:

- `conda activate jupyter_env`
- `conda install jupyterlab nodejs`

Se instala para finalizar la librería `nb_conda_kernels`, que permitirá que Jupyter detecte todos aquellos kernels de los diferentes entornos de conda:

- `conda install nb_conda_kernels`

Lanzamiento de JupyterLab

Para lanzar JupyterLab, estando este instalado en el entorno, se puede utilizar el mismo comando que para el lanzamiento de Jupyter Clásico:

- `jupyter-notebook --ip=XX.XX.XX.XX --port=XXXX --no-browser`

El output de la consola, mostrará algo similar a lo que aparece a continuación:

```
(jupyter_env) gurus@gurus-workstation:~$ jupyter-notebook --ip=192.168.1.153 --port=8911
[I 18:26:30.962 NotebookApp] [nb_conda_kernels] enabled, 6 kernels found
[I 18:26:31.392 NotebookApp] JupyterLab extension loaded from
/home/gurus/miniconda3/envs/jupyter_env/lib/python3.7/site-packages/jupyterlab
[I 18:26:31.393 NotebookApp] JupyterLab application directory is /home/gurus/miniconda3/envs/jupyter_env/share/jupyter/lab
[I 18:26:31.397 NotebookApp] Serving notebooks from local directory: /home/gurus
[I 18:26:31.397 NotebookApp] The Jupyter Notebook is running at:
[I 18:26:31.397 NotebookApp] http://192.168.1.153:8911/?token=ade92ff739744eee4885af228ba305e4f7b274f61ffe6abc
[I 18:26:31.397 NotebookApp] or http://127.0.0.1:8911/?token=ade92ff739744eee4885af228ba305e4f7b274f61ffe6abc
[I 18:26:31.397 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[W 18:26:31.404 NotebookApp] No web browser found: could not locate runnable browser.
[C 18:26:31.405 NotebookApp]

To access the notebook, open this file in a browser:
    file:///home/gurus/.local/share/jupyter/runtime/nbserver-15705-open.html
Or copy and paste one of these URLs:
    http://192.168.1.153:8911/?token=ade92ff739744eee4885af228ba305e4f7b274f61ffe6abc
    or http://127.0.0.1:8911/?token=ade92ff739744eee4885af228ba305e4f7b274f61ffe6abc
[I 18:26:34.891 NotebookApp] 302 GET / (192.168.1.1) 1.67ms
[I 18:26:34.905 NotebookApp] 302 GET /tree? (192.168.1.1) 1.47ms
[W 18:26:39.156 NotebookApp] 401 POST /login?next=%2Ftree%3F (192.168.1.1) 3.51ms
referer=http://server-davidadrian.asuscomm.com:8911/login?next=%2Ftree%3F
[I 18:26:56.926 NotebookApp] Wrote hashed password to /home/gurus/.jupyter/jupyter_notebook_config.json
[I 18:26:56.927 NotebookApp] 302 POST /login?next=%2Ftree%3F (192.168.1.1) 4.05ms
[I 18:27:08.497 NotebookApp] Build is up to date
[I 18:32:12.695 NotebookApp] 302 GET / (192.168.1.1) 1.12ms
```

Se muestra en rojo en primer lugar que la extensión de JupyterLab ha sido cargada de forma correcta, y posteriormente la URL de autenticación, que incluye un token de seguridad, y que será necesario introducir de forma completa en el navegador, donde se realizará una redirección a:

- `http://IP:PUERTO/tree?`

Esta URL es la de la versión clásica de Jupyter, para acceder a JupyterLab, se modifica en el navegador por la siguiente (es posible pasar de una a otra sin problema e incluso usar las dos al mismo tiempo teniendo cuidado de no colisionar utilizando el mismo Notebook):

- `http://IP:PUERTO/lab`

Test de Conocimientos

- 1) Utiliza habitualmente para realizar sus análisis un entorno con Python 3, Jupyter y diversas librerías analíticas instaladas en el mismo. Su superior le indica que deberá realizar un análisis para otro departamento utilizando Python 2, que posteriormente utilizarán como base para el desarrollo de un modelo propio. ¿Cómo procedería?
 - a) Utilizaría el Python 2 del sistema e instalaría Jupyter utilizando pip.
 - b) Crearía un entorno nuevo usando conda o virtualenv, donde instalaría Python 2 y Jupyter.
 - c) Crearía un entorno nuevo de conda únicamente con Python 2 e instalaría en el entorno de Jupyter la librería 'nb_conda_kernels'.
 - d) **Hablaría con mi superior para alertar de que Python 2 se encuentra deprecado y de que el soporte que recibe cesará en 2020.**
- 2) ¿Cuál de las siguientes afirmaciones **no** es correcta?
 - a) Jupyter y JupyterLab pueden convivir en el mismo entorno.
 - b) Es posible instalar Jupyter sin la necesidad de instalar JupyterLab.
 - c) **Es posible instalar JupyterLab sin la necesidad de instalar Jupyter.**
 - d) Es posible utilizar R en Jupyter.
- 3) ¿De qué forma **no es posible** lanzar Jupyter?
 - a) `jupyter-notebook --ip=XX.XX.XX.XX --port=XXXX --no-browser`
 - b) `jupyter-lab --ip=XX.XX.XX.XX --port=XXXX --no-browser`
 - c) `jupyter-notebook --port=XXXX --ip=XX.XX.XX.XX`
 - d) **`jupyter-notebook XX.XX.XX.XX XXXX`**

Ejercicio práctico

Cree su propio entorno analítico para el máster, con las siguientes características:

1. Un entorno de Jupyter, usando conda, que centralice toda la operación de los Notebooks, y que incorpore JupyterLab, así como la posibilidad de detectar los kernels de todos los entornos de conda existentes en el equipo. El entorno se llamará: `jupyter_env`
2. Un entorno de Python 3 cuyo kernel sea accesible desde el entorno virtual de Jupyter. El entorno se llamará: `py3_env`
3. Un entorno de Python 2 cuyo kernel sea accesible desde el entorno virtual de Jupyter. El entorno se llamará: `py2_env`
4. Un entorno de R cuyo kernel sea accesible desde el entorno virtual de Jupyter. El entorno se llamará: `r_env`

Compruebe la ip de su equipo en la red con el comando `ifconfig` y lance Jupyter para que sea accesible por cualquier equipo conectado a la misma red. Compruebe que se puede acceder desde la red y que los entornos son correctos (existen kernels de Python 3, Python 2 y R, así como JupyterLab).