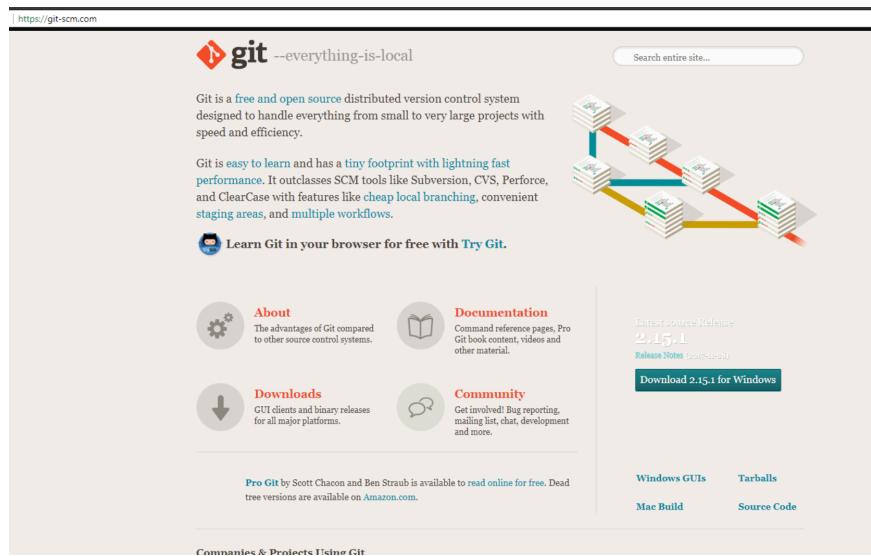


Instalación y configuración de Git

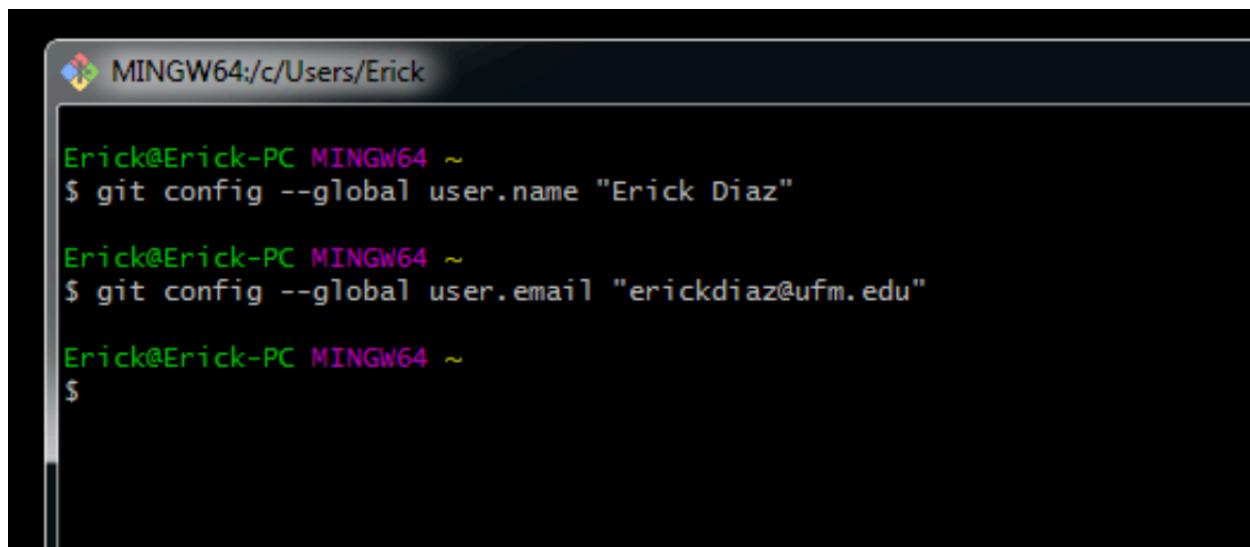
El primer paso es descargar GIT en nuestra computadora, para esto vamos al sitio oficial de Git:
<https://git-scm.com/>



Una vez descargado instalamos Git como cualquier aplicación de nuestra computadora, al finalizar la instalación abrimos la línea de comandos de Git para personalizar el entorno, este paso se realiza solo una vez, Git mantendrá estas configuraciones y te permite cambiarlas en cualquier momento.

Vamos a configurar la identidad en Git, para esto hay establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque las confirmaciones de cambios (commits) en Git usan esta información. Para configurar la identidad utilizamos la herramienta **git config** de la siguiente manera:

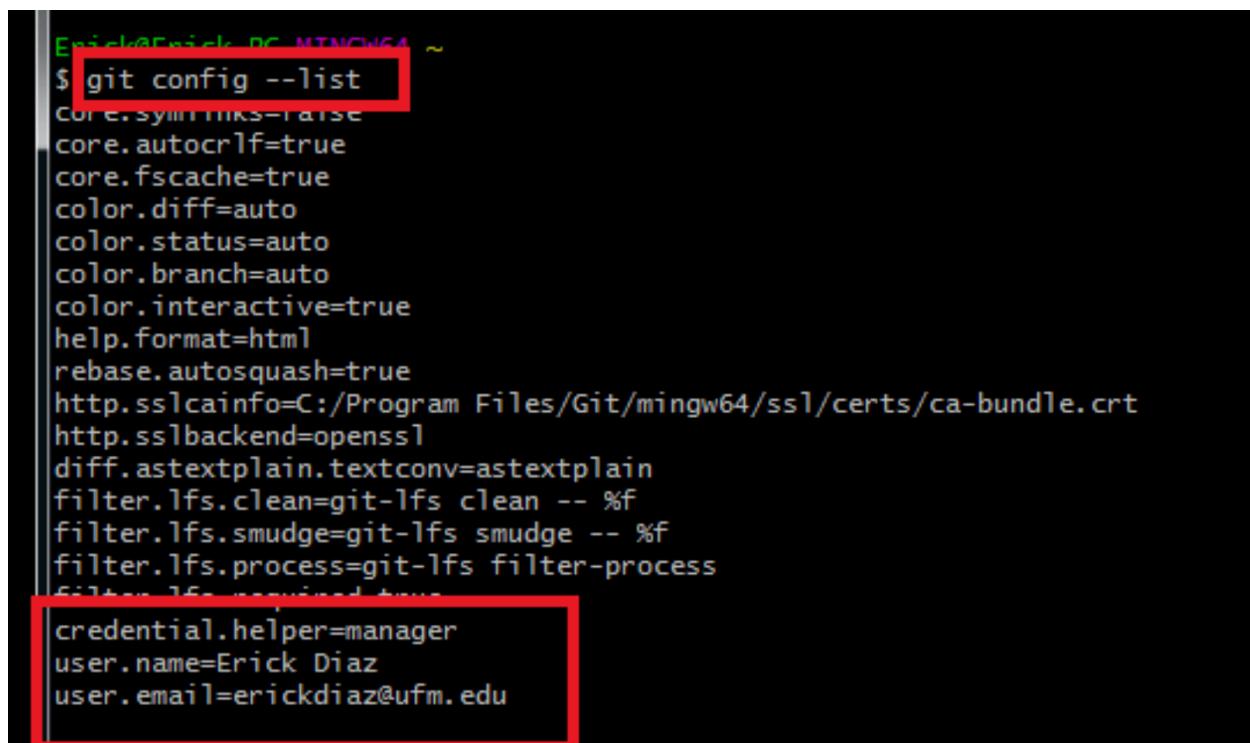
```
$ git config --global user.name "Erick Diaz"  
$ git config --global user.email erickdiaz@ufm.edu"
```



```
MINGW64:/c/Users/Erick
Erick@Erick-PC MINGW64 ~
$ git config --global user.name "Erick Diaz"
Erick@Erick-PC MINGW64 ~
$ git config --global user.email "erickdiaz@ufm.edu"
Erick@Erick-PC MINGW64 ~
$
```

Luego de ingresar estos comandos podemos comprobar la configuración, para esto podemos utilizar el comando **git config --list** para listar todas las propiedades de git

```
$ git config --list
```

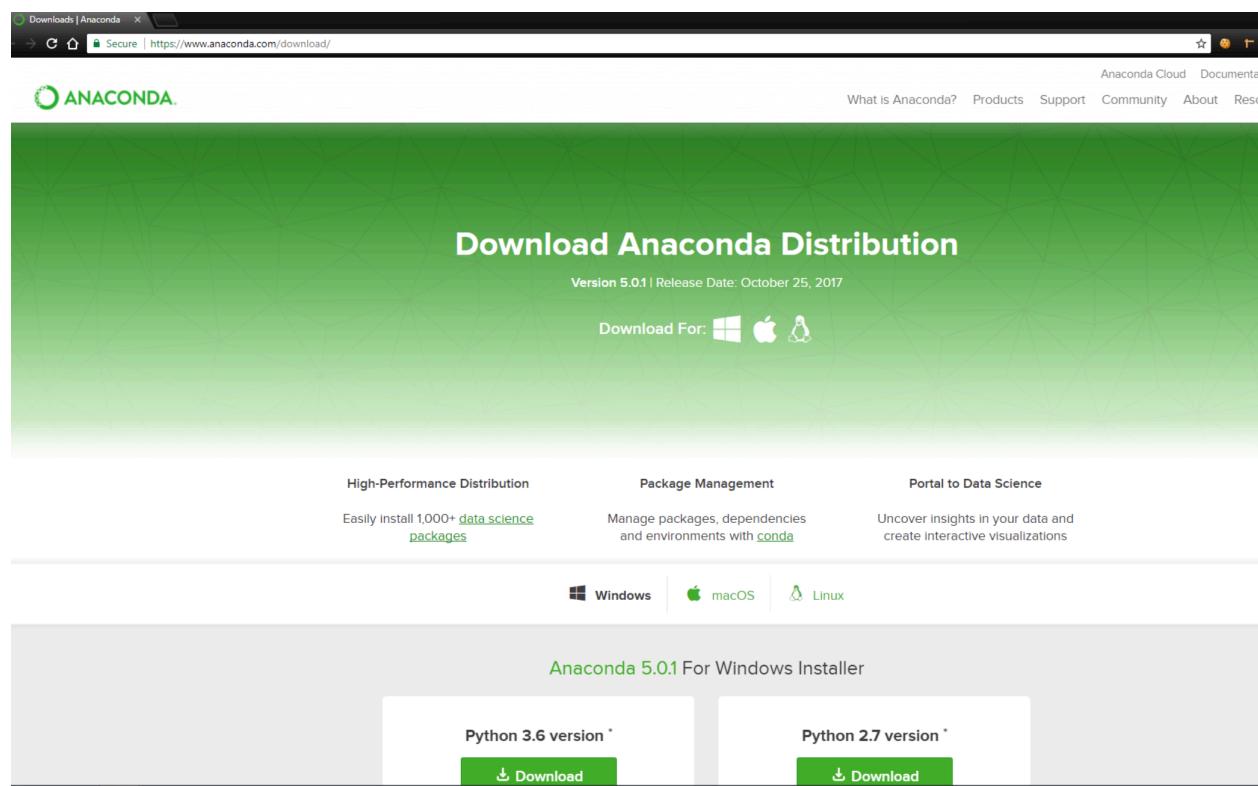


```
Erick@Erick-PC MINGW64 ~
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
user.name=Erick Diaz
user.email=erickdiaz@ufm.edu
```

Instalación de Anaconda

La instalación de Anaconda es sencilla debemos descargar el programa e instalarlo como cualquier otro, el proceso es sencillo y no necesita configuración extra.]

<https://www.anaconda.com/download/>



Creación de ambientes en Anaconda

Para crear un ambiente en Anaconda tenemos 2 opciones, desde la línea de comandos o desde la aplicación **Anaconda Navigator**, como ejemplo crearemos desde la línea de comandos un ambiente llamado **ml_ufm** con la versión de Python 3.6 y los paquetes numpy y también instalaremos Jupyter.

- 1) Para crear un ambiente, el comando es el siguiente:

```
conda create --name myenv
```

NOTA: Reemplazar **myenv** con el nombre que le quiero dar al ambiente

2) Cuando conda te pregunta si quieres seguir con la instalación ingresas y:

```
proceed ([y]/n)?
```

Esto crea el ambiente myenv en /envs/.

Para Crear un ambiente con una versión específica de Python : T

```
conda create -n myenv python=3.4
```

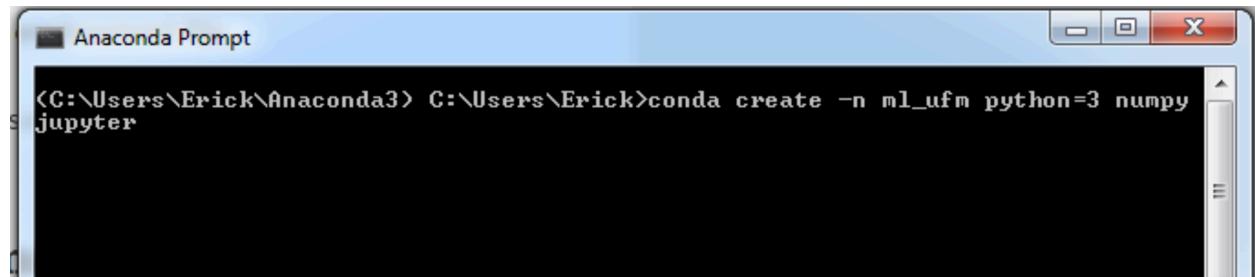
Para crear un ambiente con un paquete específico:

```
conda create -n myenv scipy
```

Ejemplos:

Creamos el ambiente ml_ufm con Python 3.6 e instalamos numpy, scikit-learn y Jupyter:

```
conda create -n ml_ufm python=3 numpy scikit-learn jupyter
```



```
conda create -n ml_ufm python=3 numpy jupyter
ipykernel:      4.7.0-py36h2f9c1c0_0
ipython:        6.2.1-py36h9cf0123_1
ipython_genutils: 0.2.0-py36h3c5d0ee_0
ipywidgets:     7.1.0-py36_0
jedi:           0.11.1-py36_0
jinja2:          2.10-py36h292fed1_0
jpeg:            9b-hb83a4c4_2
jsonschema:     2.6.0-py36h7636477_0
jupyter:         1.0.0-py36h422fd7e_2
jupyter_client: 5.2.1-py36_0
jupyter_console: 5.2.0-py36h6d89b47_1
jupyter_core:   4.4.0-py36h56e9d50_0
libpng:          1.6.32-h140d38e_4
markupsafe:    1.0-py36h0e26971_1
mistune:         0.8.3-py36_0
mkl:             2018.0.1-h2108138_4
nbconvert:       5.3.1-py36h8dc0fde_0
nbformat:        4.4.0-py36h3a5bc1b_0
notebook:        5.2.2-py36hc48260a_0
numpy:           1.14.0-py36hda99626_0
openssl:         1.0.2n-h74b6da3_0
pandoc:          1.19.2.1-hb2460c7_1
pandocfilters:  1.4.2-py36h3ef6317_1
parso:           0.1.1-py36hae3edee_0
pickleshare:    0.7.4-py36h9de030f_0
pip:              9.0.1-py36h226ae91_4
prompt_toolkit: 1.0.15-py36h60b8ff86_0
pygments:        2.2.0-py36hb010967_0
pyqt:             5.6.0-py36hb5ed885_5
python:          3.6.3-h3b118a2_4
python-dateutil: 2.6.1-py36h509ddcb_1
pyzmq:           16.0.3-py36he714bf5_0
qt:               5.6.2-vc14h6f8c307_12
qtconsole:       4.3.1-py36h99a29a9_0
setuptools:      38.4.0-py36_0
simplegeneric:  0.8.1-py36heab741f_0
sip:              4.18.1-py36h9c25514_2
six:              1.11.0-py36h4db2310_1
sqlite:          3.20.1-h9eeafa9_2
testpath:        0.3.1-py36h2698cfe_0
tornado:         4.5.3-py36_0
traitlets:       4.3.2-py36h096827d_0
vc:               14-h2379b0c_2
vs2015_runtime: 14.0.25123-hd4c4e62_2
wcwidth:         0.1.7-py36h3d5aa90_0
webencodings:  0.5.1-py36h67c50ae_1
wheel:           0.30.0-py36h6c3ec14_1
widgetsnbextension: 3.1.0-py36_0
wincertstore:   0.2-py36h7fe50ca_0
zlib:            1.2.11-h8395fce_2

Proceed <[y/n]>? y_
```

Una vez instalado el ambiente, si lo queremos utilizar debemos activarlo para esto ingresamos el siguiente comando:

```
activate ml_ufm
```

```
#
# To activate this environment, use:
# > activate ml_ufm
#
# To deactivate an active environment, use:
# > deactivate
#
# * for power-users using bash, you must source
#
<C:\Users\Erick\Anaconda3> C:\Users\Erick>activate ml_ufm
```

Sabemos que el ambiente esta activo porque aparece su nombre entre paréntesis al inicio de la línea. Con el ambiente activo ya podemos ejecutar Jupyter y crear nuestro primer notebook:

```
jupyter notebook
```

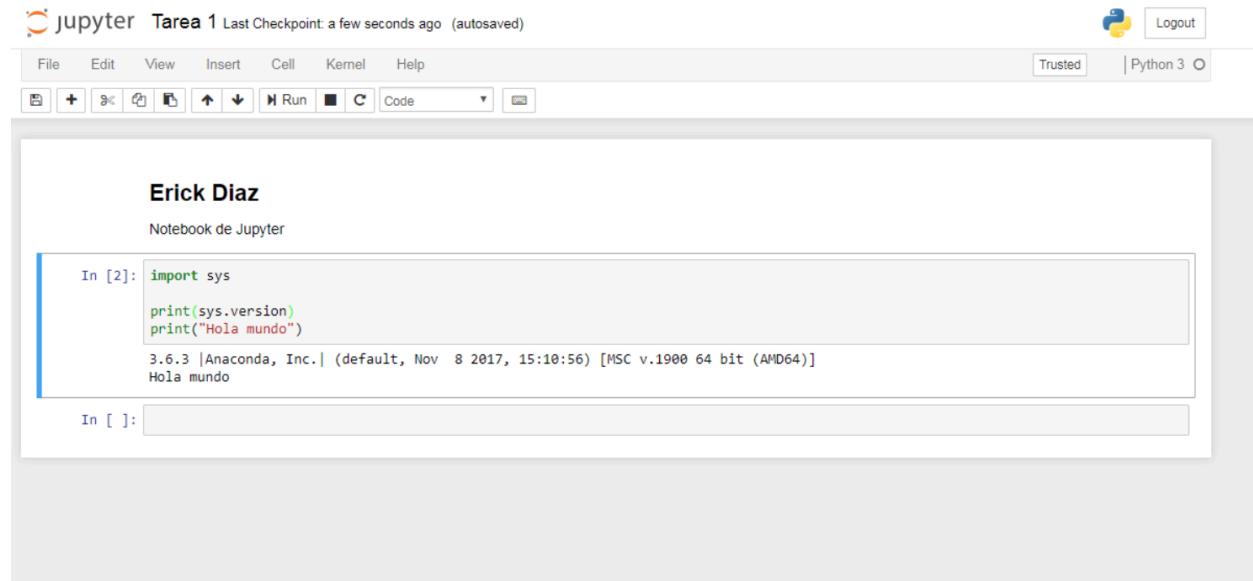
```
(ml_ufm) C:\Users\Erick\Documents>jupyter notebook
```

Al presionar la tecla enter lanzara Jupyter en el navegador web, una vez en Jupyter si queremos crear un nuevo notebook que utilice Python presionamos en botón new y seleccionamos Python 3, que es la versión de Python que instalamos en el ambiente **ml_ufm** :



Jupyter nos permite crear celdas en Markdown, esto es muy útil para hacer anotaciones y documentar el código, el siguiente link es una guía de Markdown:
<https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

De esta forma de debe de ver la tarea, la primera celda esta en Markdown y la siguiente celda es código de Python:



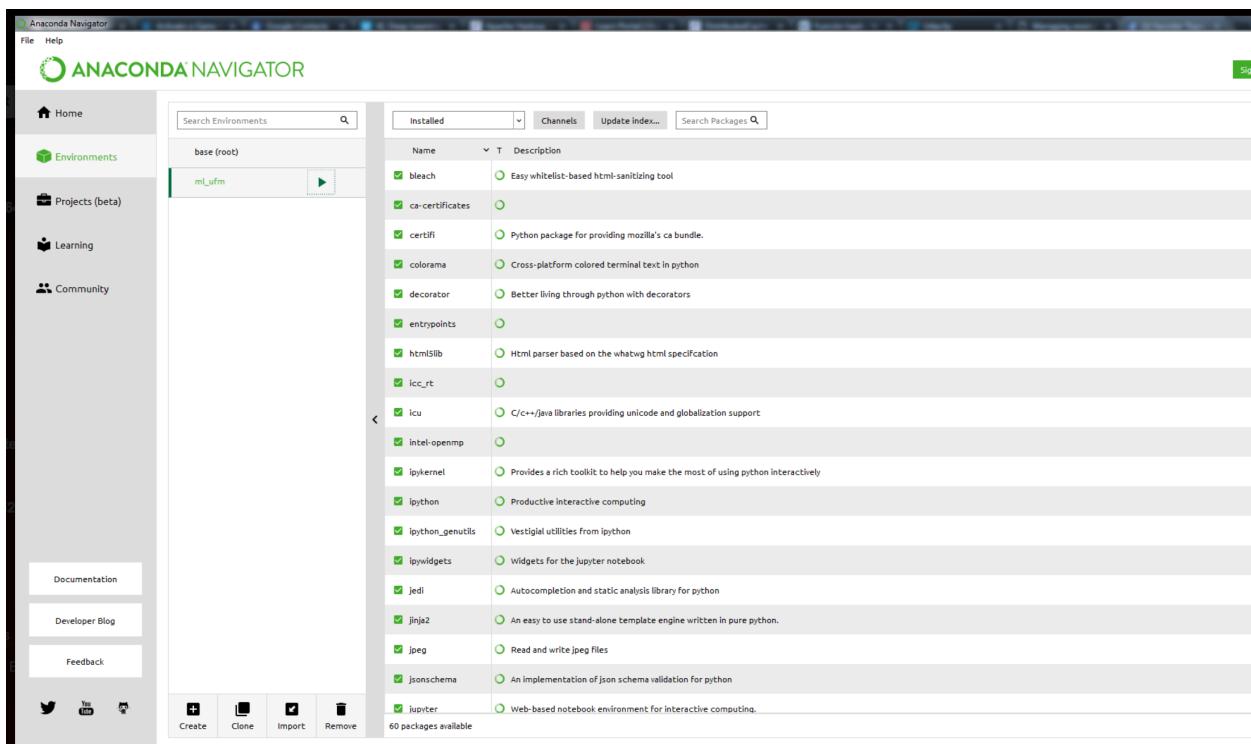
```
In [2]: import sys
print(sys.version)
print("Hola mundo")
3.6.3 |Anaconda, Inc.| (default, Nov  8 2017, 15:10:56) [MSC v.1900 64 bit (AMD64)]
Hola mundo
```

Configuración del Ambiente en la interfaz Grafica

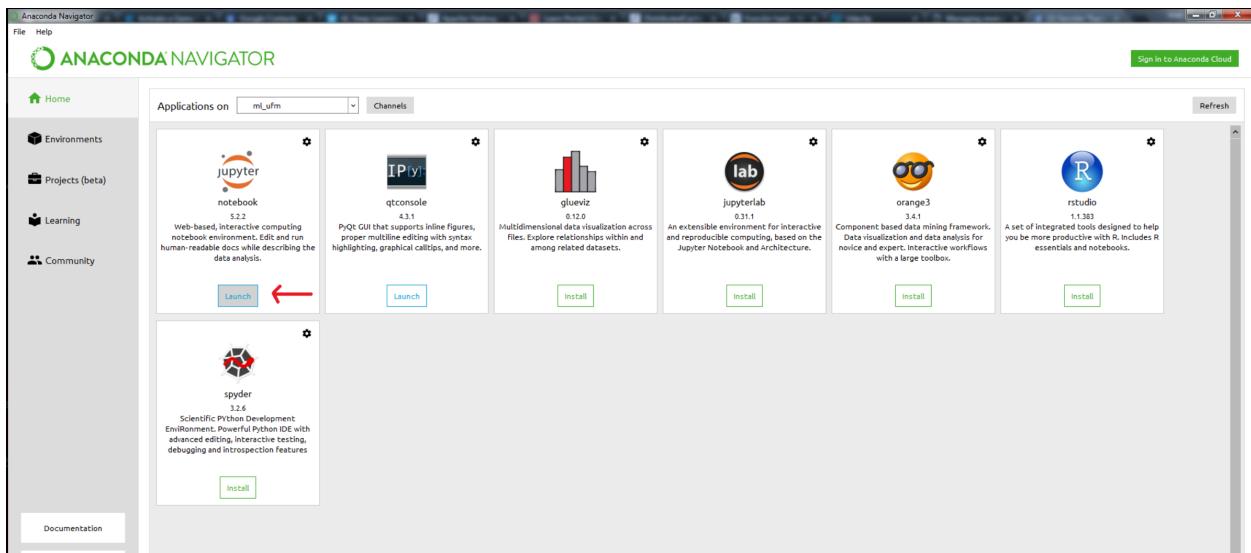
Para cambiar el tipo de celda damos click en el dropdown que esta en la barra de herramientas y seleccionamos el tipo de celda puede ser Markdown o Code.

Si no queremos utilizar la línea de comandos podemos crear el ambiente y ejecutar Jupyter desde la aplicación Anaconda Navigator

En la pestaña de ambientes nos lista los ambientes que tenemos disponibles y lista que paquetes tiene cada ambiente, en la parte inferior de la ventana este el botón para crear un nuevo ambiente.



En la pestaña “Home” podemos seleccionar el ambiente y ejecutar Jupyter



Creación de repositorio en GitHub

En este ejercicio vamos a crear un nuevo repositorio en GitHub y vamos a agregar el notebook de Jupyter que creamos.

Para esto se utiliza la página de GitHub y en nuestra maquina vamos a utilizar los siguientes 4 comandos:

- **git clone** Este comando se utiliza para clonar un repositorio remoto a un directorio local.
 - <https://git-scm.com/docs/git-clone>
- **git add** Utilizamos este comando para preparar el contenido para el siguiente commit, es decir que preparamos que archivos que han sido modificados queremos que se reflejen sus cambios en el commit.
 - <https://git-scm.com/docs/git-add>
- **git commit** guarda el estado actual del index con un mensaje que el usuario define describiendo los cambios que realizo, este comando se ejecuta después de git add, cuando ya tenemos todos los cambios listos.
 - <https://git-scm.com/docs/git-commit>
- **git push** este comando se utiliza para actualizar el repositorio remoto.
 - <https://git-scm.com/docs/git-push>

Fundamentos de Git; <https://git-scm.com/book/es/v1/Fundamentos-de-Git-Guardando-cambios-en-el-repositorio>

Creando el nuevo repositorio remoto en GitHub

Para esto iniciamos sesión en GitHub, buscamos el botón con el signo + justo a la para de nuestra imagen en GitHub esto va a desplegar un menú donde podemos crear el repositorio.

The screenshot shows the GitHub homepage with a dark theme. At the top right, there is a user icon and a menu with a '+' sign. A dropdown menu is open, showing options: 'New repository' (which is highlighted in blue), 'Import repository', 'New gist', and 'New organization'. Below this, there are sections for 'Repositories you contribute to' (with two repositories listed) and 'Your repositories' (showing 46 repositories). On the left, there are news items and a search bar. The main area displays several recent activity items, such as a star event for 'jwilder/nginx-proxy' and a follow event for 'DigitalCraftGr'.

Para crear el repositorio debemos definir un nombre dejar que se Publica ya que si queremos que sea Privado esto tiene un costo y seleccionar la opción de iniciar el repositorio con el archivo README, esto facilitara la creación del repositorio.

El archivo readme se utiliza para describir el contenido del repositorio, el propósito y como utilizarlo ese archivo esta en Markdown.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner Repository name

 ErickDiaz  / 

Great repository names are short and memorable. Need inspiration? How about [jubilant-fiesta](#).

Description (optional)

 **Public**
Anyone can see this repository. You choose who can commit.

 **Private**
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**  | Add a license: **None**  

Create repository

Si los pasos anteriores están bien debemos ver una pantalla como esta, GitHub por default muestra el contenido del archivo Readme.

GitHub tiene una interfaz para editar el archivo Readme, y al guardas los cambios automáticamente hace commit de estos.

ErickDiaz / ufm_ml_tarea1

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

No description, website, or topics provided. Edit Add topics

1 commit 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download ▾

ErickDiaz Initial commit Latest commit e466a65 just now

README.md Initial commit just now

README.md

ufm_ml_tarea1

Agregando archivos a nuestro repositorio

El primer paso es obtener la URL del repositorio para clonarlo en nuestra maquina.

Create new file Upload files Find file Clone or download ▾

Clone with HTTPS ⓘ Use SSH

Use Git or checkout with SVN using the web URL.

https://github.com/ErickDiaz/ufm_ml_tarea1

Copied!

Open in Desktop Download ZIP

Entramos a la línea de comandos, nos movemos a la carpeta donde queremos clonar el Repositorio en este caso es “Mis Documentos” e ingresamos el siguiente comando:

```
git clone <url_de_nuestro_repositorio>
```

Nos va a solicitar nuestro usuario y contraseña, cuando termina de clonar el repositorio debemos de poder ver una carpeta con el nombre del repositorio y el archivo README.md

```
LM-GUA-22002173:Documents erickdiaz$ git clone https://github.com/ErickDiaz/ufm_ml_tarea1.git
Cloning into 'ufm_ml_tarea1'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
LM-GUA-22002173:Documents erickdiaz$ ls
ufm_ml_tarea1  metrics */ to sql script(create table) for
```

Name	Date Modified
ufm_ml_tarea1	Today, 5:26 PM
README.md	Today, 5:26 PM

Ya clonamos el repositorio remoto, ahora en nuestra copia local ya podemos hacer cambios sin afectar la copia remota, el repositorio remoto se actualiza al momento de ejecutar el comando push. En este caso ya había creado el notebook de Jupyter por lo que solo voy a copiar y pegar el archivo en la carpeta del repositorio.

Name
ufm_ml_tarea1
README.md
Tarea 1.ipynb

Podemos mejorar el archivo léeme desde GitHub o lo modificamos en un editor de texto en nuestra maquina, esta parte es opcional solo para demostrar como hacer commit y push de múltiples archivos.

Original:

```
1 # ufm_ml_tarea1
```

Modificado:

```
1 # UFM ml tarea1
2
3 ### By Erick Diaz
```

Git add

Actualizamos el index para preparar el contenido que vamos a hacer commit en este caso es el archivo nuevo “Tarea 1.ipynb” que actualmente no forma parte de nuestro repositorio y el archivo léeme que forma parte de nuestro repositorio pero fue modificado, los cambios que no agreguemos al índice no se van a ver reflejados en nuestro repositorio remoto.

Para agregar nuestros 2 archivos ingresamos el siguiente comando en la línea de comandos dentro de la carpeta de nuestro repositorio:

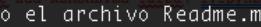
```
git add README.md Tarea\ 1.ipynb
```

```
LM-GUA-22002173:ufm_ml_tareal erickdiaz$ git add README.md Tarea\ 1.ipynb  
LM-GUA-22002173:ufm_ml_tareal erickdiaz$   
Actualizamos el index para preparar el contenido que vamos a hacer commit en este caso
```

Git commit

Ahora ya podemos hacer commit de nuestros cambios para luego subirlos a nuestro repositorio remoto en GitHub, antes de ejecutar git commit debamos tener listos los archivos con git add, de lo contrario va a desplegar un mensaje de error indicando que no hay cambios que guardar. Al hacer commit debemos ingresar un mensaje que describe los cambios que realizamos al repositorio:

```
git commit -m "Se modiflico el archivo Readme.md y se agrego el archivo  
Tarea 1.ipynb"
```

```
LM-GUA-22002173:ufm_ml_tareal erickdiaz$ git commit -m "Se modiflico el archivo Readme.md y se agrega  
o el archivo Tarea 1.ipynb"  
[master b6845e3] Se modiflico el archivo Readme.md y se agrego el archivo Tarea 1.ipynb  
2 files changed, 65 insertions(+), 1 deletion(-)   
create mode 100644 Tarea 1.ipynb  
LM-GUA-22002173:ufm_ml_tareal erickdiaz$ 
```

Git push

Podemos realizar múltiples cambios en nuestro repositorio local y tener realizar varios commits, cuando queremos que estos cambios se vean reflejados en nuestro repositorio remoto para que otros usuarios de GitHub los puedan ver ejecutamos git push

```
git push
```

```
LM-GUA-22002173:ufm_ml_tareal erickdiaz$ git push  
Counting objects: 4, done.  
Delta compression using up to 8 threads.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (4/4), 840 bytes | 840.00 KiB/s, done.  
Total 4 (delta 0), reused 0 (delta 0)  
To https://github.com/ErickDiaz/ufm_ml_tareal.git  
 e466a65..b6845e3 master -> master
```

Si realizamos de forma correcta todos los pasos anteriores en la pagina de GitHub del repositorio deberíamos de poder ver nuestros archivos y el mensaje de commit

No description, website, or topics provided.

2 commits 1 branch 0 releases 1 contributor

Erick Diaz Se modiflico el archivo Readme.md y se agrego el archivo Tarea 1.ipynb

README.md Se modiflico el archivo Readme.md y se agrego el archivo Tarea 1.ipynb

Tarea 1.ipynb Se modiflico el archivo Readme.md y se agrego el archivo Tarea 1.ipynb

README.md

UFM ml tarea1

By Erick Diaz

Git permite visualizar los notebooks de Jupyter por lo que podemos ver el notebook al dar click en el nombre del archivo:

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Branch: master ufm_ml_tarea1 / Tarea 1.ipynb Find file Copy path

Erick Diaz Se modiflico el archivo Readme.md y se agrego el archivo Tarea 1.ipynb b6845e3 6 minutes ago

0 contributors

63 lines (62 sloc) | 1.07 KB Raw Blame History

Erick Diaz

Notebook de Jupyter

```
In [2]: import sys
print(sys.version)
print("Hola mundo")
```

3.6.3 |Anaconda, Inc.| (default, Nov 8 2017, 15:10:56) [MSC v.1900 64 bit (AMD64)]
Hola mundo

```
In [ ]:
```