USM Numérica
Licencia y configuración del laboratorio

IPython Notebook v4.0 para Python 3.0

Sebastian Flores, Christopher Cooper, Alberto Rubio, Pablo Bunout.

Introducción a *GIT*

GIT es un sistema de control de versiones, desarrollado y distribuido a partir del año 2005 por Linux Kernel. Una característica fundamental de un controlador de versiones, es que las unidades básicas sobre las que se estructura, denominadas repositorios, además de almacenar información y/o archivos al igual que un directorio o carpeta de un sistema operativo cualquiera, como Linux o Windows, son capaces de generar un historial o línea de tiempo que registre los diversos estados del repositorio y sus contenidos respectivos, otorgándonos la posibilidad de acceder a estos estados a partir de la ejecución de procedimientos específicos a través de *git*.

Otra cualidad importante de *GIT*, es que nos permite vincular repositorios generados localmente en la computadora física, con otros repositorios remotos, existentes en la red (online). Pudiendo subir contenidos desde el repositorio local al remoto o bajar material de manera inversa.

Lo anterior abre un mundo de posibilidades, particularmente en contextos donde el flujo de información sea relevante, por ejemplo en proyectos que requieran la participación de 2 o más personas trabajando en archivos comunes, siendo *GIT* una herramienta útil para sincronizar las versiones de distintos usuarios, realizadas simultáneamente o en distintos tiempos. Un buen uso de las herramientas que nos provee *GIT*, ayudará a sincronizar y/o fusionar archivos evitando la pérdida de información o la superposición de esta.

En este tutorial veremos algunas instrucciones y comandos básicos, que nos permita realizar operaciones fundamentales, tales como crear un repositorio local, clonar un repositorio en línea y sincronizar ambos.

Objetivos

1. Operaciones para crear un repositorio local
2. Operaciones trabajo en repositorio local
3. Sincronización de repositorio local y remoto
4. Clonar directamente un repositorio remoto en un directorio de trabajo local
5. Operaciones para repositorios remotos con más de un usuario

1. Operaciones para crear un Repositorio Local

Para crear un repositorio local, debemos primero generar un directorio sobre el cual queremos guardar los archivos de trabajo y luego identificar este como un repositorio *GIT*. Usamos los siguientes comandos de *bash* para crear un directorio desde la terminal.

```
mkdir <repositorio_local>
```

Para que *GIT* reconozca el directorio creado como un repositorio debemos primero ubicarnos en el nuevo directorio y luego escribir el comando *git init*.

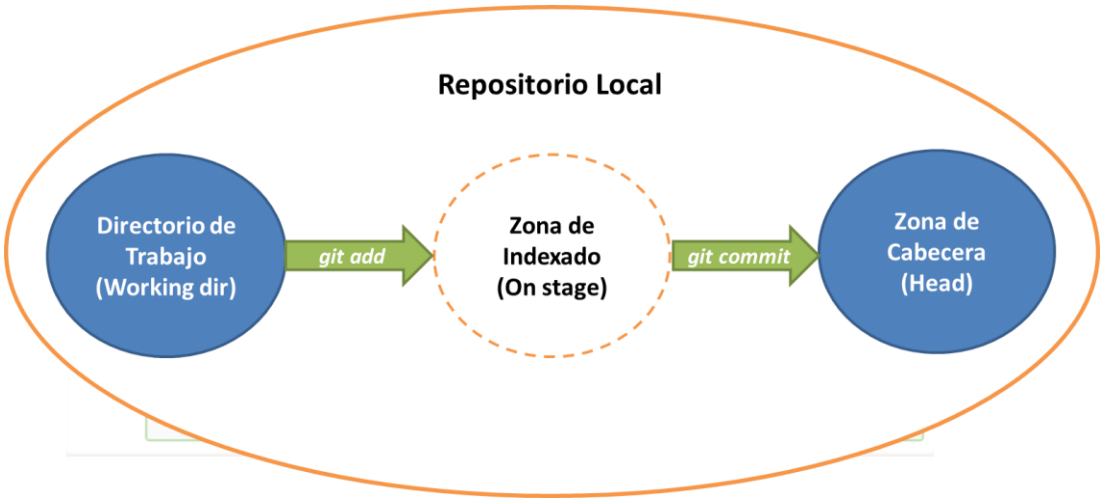
```
cd <repositorio_local>
git init
```

Ahora *GIT* ya reconoce el directorio como un nuevo repositorio.

2. Operaciones de trabajo en Repositorio Local

Antes de explicar los procedimientos y comandos que nos permiten subir o bajar información desde y hacia un repositorio, actualizar nuevos estados de archivos almacenados en él y/o sincronizar repositorios locales con otros remotos, es importante entender que GIT proporciona 3 zonas virtuales, las que cumplen distintas tareas en función de las operaciones mencionadas recientemente. Estas son el *directorio de trabajo*, una *zona de indexado* y por último una zona que actúa como *cabecera* del repositorio, las que pasamos a detallar en orden respectivo.

El *directorio de trabajo* corresponde a la zona sobre la cual se ha iniciado el repositorio local. Aquí es donde se almacenan los archivos locales y se registran las modificaciones realizadas a los últimos. Hasta el momento esto no tiene ninguna diferencia con la manera de operar en un sistema operativo, que nos permite generar un directorio o carpeta y guardar archivos en ella, actualizándolos de forma manual.



La segunda es una zona intermedia, en la cual disponemos los archivos modificados en el directorio local, seleccionándolos con el comando *git add*.

Finalmente la tercera zona es sobre la cual *GIT* genera un historial o la línea de tiempo, usando el comando *git commit*, que actualiza los archivos previamente seleccionados, registrando las modificaciones respectivas en el repositorio local.

Describimos los procedimientos mencionados anteriormente:
Ya hemos iniciado un repositorio local sobre un directorio creado recientemente, si queremos añadir algún archivo creándolo desde la terminal, podemos escribir los comandos de *bash*:

```
touch <archivo.tipo>
```

Una vez creado el archivo en el directorio de trabajo, para que sea reconocido en el repositorio local debemos seleccionarlo de la siguiente manera.

```
git add <archivo.tipo>
```

En caso de que existan muchos archivos sobre los cuales hemos realizado modificaciones, una forma de seleccionarlos sería escribir sus nombres uno al lado de otro de forma sucesiva.

```
git add <archivo_1.tipo> ... <archivo_n.tipo>
```

Otra manera de hacer más eficiente esto último, sería utilizando el comando `git add -u`, el cual selecciona automáticamente todos los archivos modificados desde el último *commit*, descartando aquellos archivos sobre los cuales no se han realizado modificaciones presentes.

Un comando alternativo que permite seleccionar todo lo existente en el directorio de trabajo es `git add .`, haciéndolo sin discriminar entre los archivos modificados y los que no, es usando `.`

Luego de indexar los archivos seleccionados, debemos disponerlos en la tercera zona, que es la que finalmente GIT reconoce y sobre la cual genera el historial de versiones.

```
git commit -m <"mensaje actualización">
```

Ya tenemos entonces los archivos y actualizaciones en nuestro repositorio local al disponerlos en el encabezado de este. Otro aspecto útil de *git*, es que cada vez que hacemos un *commit*, este queda registrado con una etiqueta o *id*, podemos acceder a esta información a través de *git log*. Si queremos modificar la etiqueta de forma manual, escribimos lo siguiente:

```
git tag <etiqueta nueva> <etiqueta antigua>
```

2.1 Ejercicio Práctica

- Crear un directorio donde generar el repositorio local
- Crear un repositorio local en el directorio
- Generar un archivo de texto en el directorio de trabajo
- Actualizar el archivo creado en el repositorio local *git add + git commit*
- Modificar la etiqueta de la actualización por *"mi primer commit"*
- Visualizar antes y posteriormente la etiqueta con *git log*

3. Sincronización de Repositorio Local y Remoto

Cuando sincronizamos un repositorio local con uno remoto o existente en la red (online), tenemos la posibilidad de intercambiar información entre estos, es decir subir material nuevo desde el repositorio local al remoto y viceversa.

Para que el repositorio local reconozca la dirección del repositorio remoto, debemos utilizar el siguiente comando, haciéndolo desde el directorio de trabajo.

```
git remote add origin https://dirección repositorio online
```

Por defecto el repositorio local reconocerá al remoto por nombre *origin*, esto nos será útil para abreviar la dirección total del repositorio remoto, lo que tendrá un efecto práctico a la hora de realizar operaciones con diversos comandos de *git*. Luego podemos visualizar las opciones de subir (*push*) y bajar (*fetch*) información con el comando:

```
git remote -v
```

Otro aspecto importante a entender, son las líneas de tiempo o historiales denominada *branch*. Cada repositorio tiene un *branch* principal el que por defecto se denomina *master*. *GIT* nos permite además crear *branchs* paralelos al principal, lo que puede ayudar a realizar modificaciones a los archivos de forma segura, sin que estos se actualicen automáticamente en la rama *master*, debiendo hacer una operación extra para fusionar el *branch* secundario con *master* principal.

Ahora estamos listos para subir archivos al repositorio remoto desde el local, ya hemos mostrado como crear un archivo en el directorio de trabajo y a continuación especificaremos los pasos para actualizarlo en el repositorio local y luego subirlo al remoto.

```
git add <archivo.tipo>
git commit -m <"mensaje actualización">
git push origin master
```

En resumen, el primer comando selecciona el archivo creado en el directorio de trabajo, para colocarlo en la *zona de indexado* u *onstge*.

Luego lo hemos actualizado en el repositorio local, con un mensaje en el que especificamos las modificaciones hechas.
Finalmente subimos estos archivos al repositorio remoto, reconocido por el local a partir de la denominación *origin*, haciéndolo sobre línea de tiempo o rama principal (*branch*), denominada por defecto con el nombre de *master*.

4. Clonar directamente un Repositorio Remoto en un Directorio de Trabajo

Cuando existe un repositorio en la red, el cual tiene sus propios archivos y queremos tener acceso a estos, tenemos la opción de hacerlo desde un repositorio local, trabajando desde nuestro directorio de trabajo.

Para esto existe la opción de clonar el repositorio en la re, escribiendo el siguiente comando desde un directorio de trabajo local específico:

```
git clone https://dirección repositorio online
```

Eso crea automáticamente un repositorio local que reconoce la dirección del repositorio remoto por defecto como *origin* y su rama principal como *master*, al igual que lo visto en la sección anterior.

Si luego trabajamos en los archivos bajados desde la red, desde nuestro directorio local y queremos subir las modificaciones hechas en el repositorio local, repetimos de forma idéntica la secuencia de comandos vista en el último punto.

```
git add <archivo.tipo>
git status
git commit -m <"mensaje actualización">
git push origin master
```

Hemos agregado el comando *git status*, ya que nos permite visualizar cuales archivos de nuestro repositorio local han sido modificados o creado, y cuáles de ellos están en la zona de indexado, lo que es útil a la hora de trabajar con gran cantidad de archivos y directorios.

4.1 Ejercicio Práctica

- Crear un directorio y clonar en el un repositorio remoto con *git clone*
  (copiar la dirección de reconocimiento del repositorio desde la plataforma online)
- Copiar el archivo creado anteriormente y crear otro nuevo
- Indexar los 2 archivos con el comando *git add .*
- Encabezar los archivos indexados con un mensaje que diga *"primer commit"*
- Visualizar el estado de los archivos nuevos y antiguos con sus modificaciones
- Subir el archivo desde el repositorio local al remoto con *git push*

5. Operaciones para Repositorios Remotos con más de un Usuario

En esta sección veremos aspectos importantes en caso de que trabajemos localmente, sincronizados a un repositorio remoto al cual tienen acceso más de un usuario. En este contexto es importante considerar que un usuario puede generar modificaciones locales y subirlas al repositorio remoto. Un problema típico en esta lógica de trabajo es el desfase de las versiones entre los usuarios para iguales archivos, surge entonces la necesidad de actualizar los archivos en nuestro repositorio local, sincronizándolos con las modificaciones hechas por otros usuarios, antes de comenzar a trabajar y realizar modificaciones en ellos. Para esto usamos el siguiente comando *GIT*, ubicados en nuestra dirección de trabajo local.

```
git pull
```

Es una muy buena práctica, cada vez que trabajamos sobre nuestro repositorio local, actualizarlo con el comando anterior, en caso de que algún usuario haya hecho modificaciones en el repositorio remoto.

Otra necesidad o requerimiento que puede surgir, es que si queremos almacenar nuestro historial de trabajo en un *branch* paralelo al principal *master*, debemos crear esta línea de tiempo usando el comando *checkout -b*, que a su vez nos posiciona a este nuevo *branch* desde nuestro directorio local.

```
git checkout -b <branch_secundario>
```

De este modo, podemos subir las modificaciones hechas a la rama secundaria, que nos permitirá trabajar de forma más segura, en caso de tener problemas de desfase entre el repositorio remoto y el local.

```
git push origin <branch_secundario>
```

El comando *checkout* nos permite cambiar de *branch* según nuestro interés, por ejemplo si queremos volver a la rama principal lo hacemos como sigue:

```
git checkout <master>
```

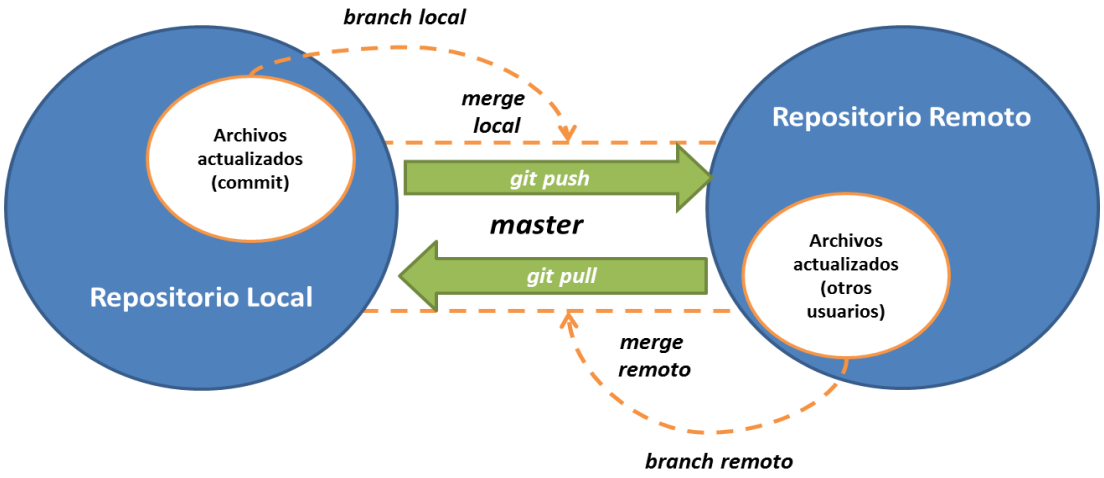
Finalmente, al trabajar sobre un *branch* secundario, debemos fusionar este *branch* con *master*, podemos hacerlo directamente desde nuestro repositorio local o sobre el remoto, esto último dependerá de la plataforma sobre la cual estamos trabajando. Por ejemplo *github* nos ofrece realizar la fusión entre *branchs* haciendo click en *merge*.

Si hacemos la fusión desde nuestro repositorio local, debemos escribir:

```
git merge <branch_secundario>
```

Es recomendable antes de realizar la fusión entre los *branchs*, revisar las diferencias entre ambas, esto puede realizarse con:

```
git diff <source_branch> <target_branch>
```



En caso de que se exista algún error en nuestras operaciones desde el repositorio local, podemos solucionar este problema eliminando las modificaciones hechas y reemplazarlos por las versiones registradas en el último *commit* registrado en el repositorio.

```
git checkout -- <archivo.tipo>
```

Si el asunto es más grave y queremos deshacer todos los cambios locales realizados, podemos bajar la última versión almacenada en el servidor del repositorio remoto con:

```
git fetch origin
```

Y luego reemplazar esta versión por la local.

```
git reset --hard origin/master
```

5.1 Ejercicio de práctica

- Fusionar el repositorio local creado en *ej.1.1* con un remoto usando *git remote add origin (copiar la dirección del repositorio remoto desde la plataforma online)*
- Generar un *branch* paralelo con *git checkout -b*
- Crear un archivo de texto en el *branch* secundario
- Indexar únicamente los archivos modificados usando *git add -u* y luego *git commit*
- Subir las actualizaciones en el repositorio local con *git push origin paralelo*
- Realizar un *merge* entre el *branch* secundario y *master* desde la plataforma online, si es posible