Arrancando con GIT

El objetivo de este apunte es mostrar de una forma teórica/práctica qué es GIT y responder a las preguntas más que naturalmente aparecen al empezar a utilizarlo.

¿Para qué me sirve?

- Compartir código con otras personas.
- Tener un historial de los cambios realizados en el código.

¿Por qué lo usamos en diseño?

Para aprender cómo diseñar un sistema necesitamos un contexto, un problema que resolver que nos permita poner nuestros conocimientos a prueba.

En diseño elegimos un problema que se resuelva con sistemas software porque nos parece que el software presenta los problemas más complicados y, por lo tanto, las discusiones más interesantes.

Como vamos a estar creando soluciones software para acompañar nuestro proceso de aprendizaje, nos vienen bien las ventajas que nos da un sistema de versionado de código (como por ejemplo git).

¿Por qué?

- **Vamos a estar trabajando con otras personas**
  Entonces alguna forma de compartir el código vamos a precisar. Un sistema de versionado nos permite hacer esto de forma prolija, ya que podemos tener un historial de cambios, saber quién los hizo y cuándo, con comentarios que acompañan a los mismos, y además, algo muy importante, es que nos permite resolver los conflictos que surjan cuando dos o más personas modifiquen el mismo archivo (vamos a ver algo sobre esto más adelante).
- **Vamos a manejar proyectos complejos**
  No es lo mismo si tenemos un único archivo con pocas líneas de código, que si vamos a estar contribuyendo en proyectos medianamente grandes. A medida que nuestro sistema crece se vuelve cada vez más importante mantener una buena trazabilidad e introducir el cambio pero de manera controlada.

Esas son las principales ventajas, aunque podemos mencionar muchas otras.

Por último no queremos dejar pasar esta oportunidad de animarte a que, si nunca antes habías usado git, aproveches para aprender sobre esta herramienta que es ampliamente utilizada en el mundo de sistemas (tanto industrial como open source).

¿Cómo lo instalo?

- https://git-scm.com/downloads

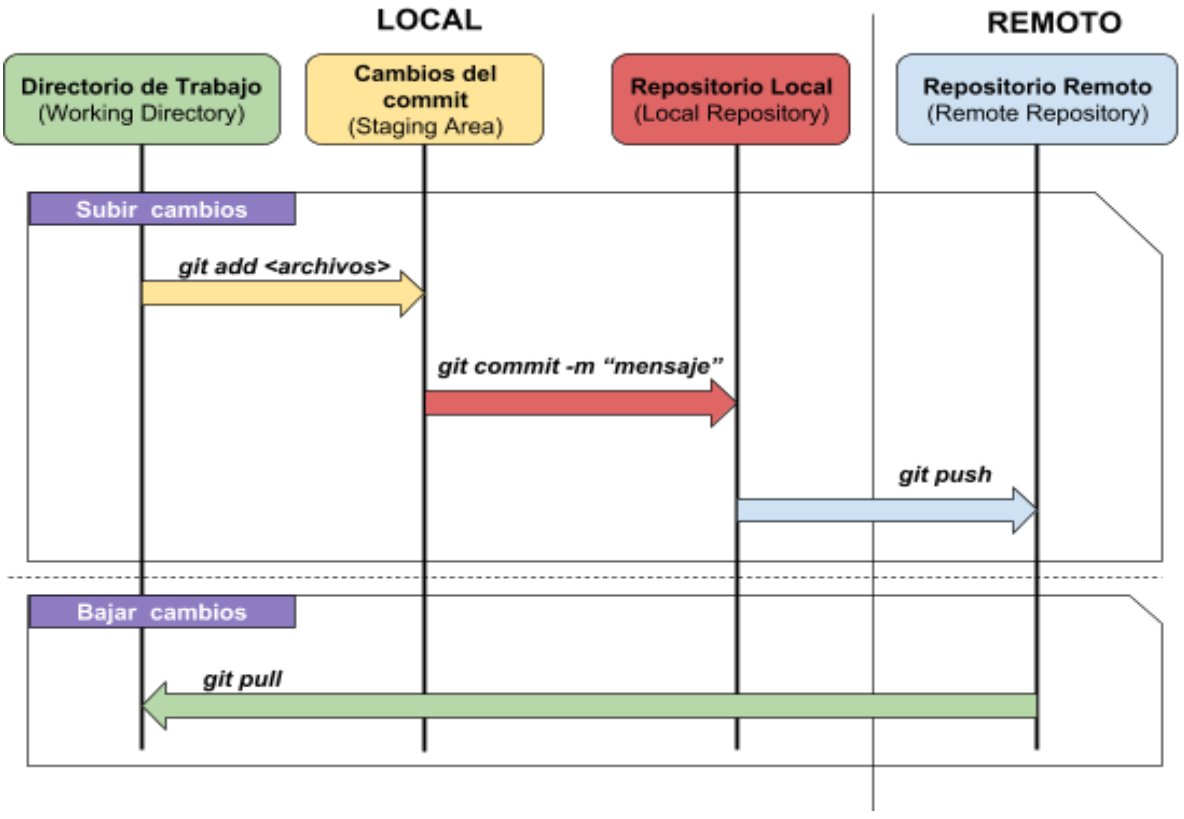
Importante: ¿Cómo sé si instale correctamente GIT?

En la terminal (linux / windows) ejecuto el comando
git --version

Debería mostrar algo similar a lo siguiente:
git version 2.14.1

¿Cómo funciona?

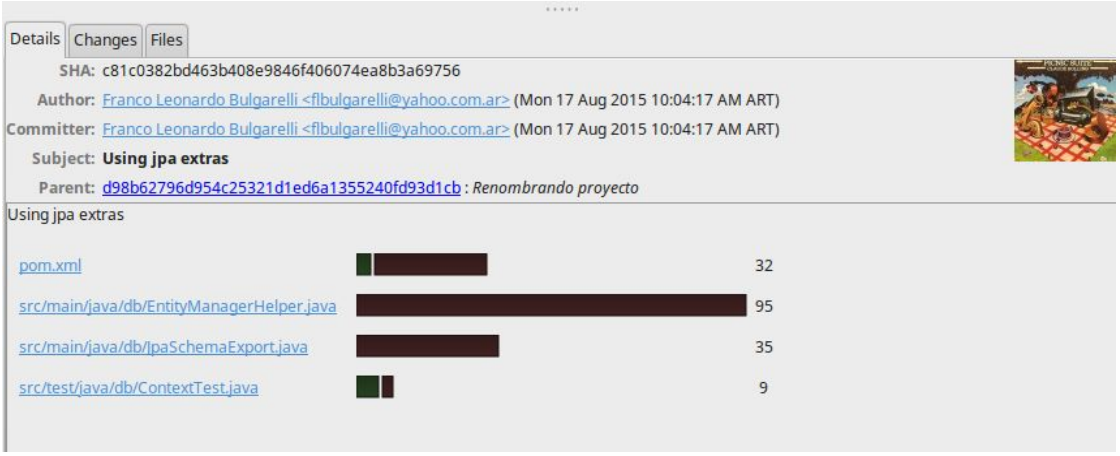
1. Creas/borras/modificas archivos en una carpeta asociada a un repositorio.
2. Seleccionas los archivos que van a ser parte de un commit.
3. Confirmas el commit para agregarlo al repositorio.
4. Subis los commits de tu repositorio local al repositorio remoto.



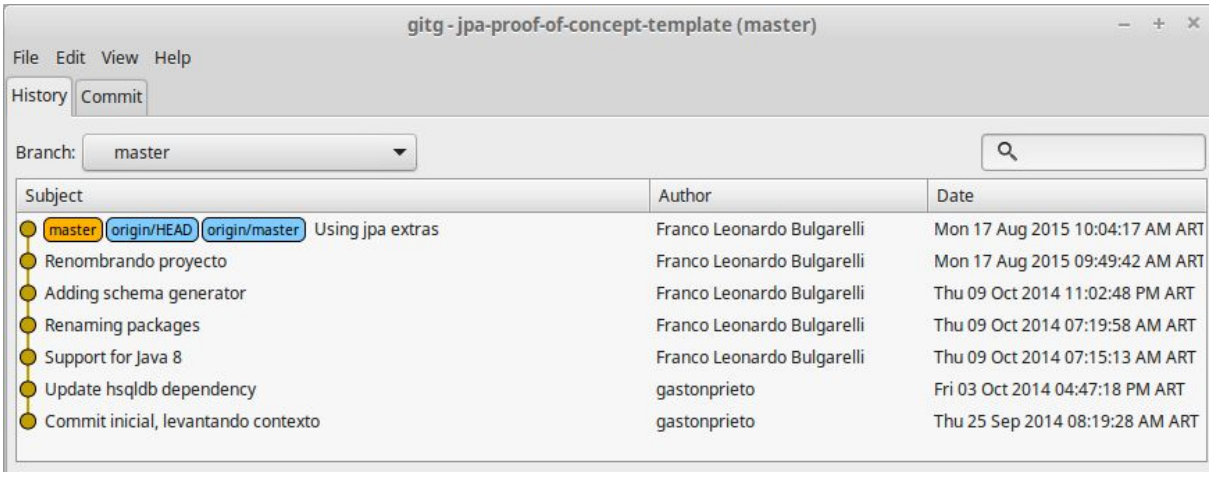
Momento, cerebrito… ¿Repositorio? ¿Commit? ¿Qué es eso?

- Un Commit es un conjunto de cambios en tus archivos que decidiste que tenían que ser versionados juntos, además le podes poner una descripción para que sepas que

cambiaste.



- Un Repositorio es el lugar donde van a estar todos los commits que forman parte de tu historial.



Dijiste repositorio local y remoto ¿Cuál es la diferencia?

Bueno… GIT trabaja con un repositorio local que está en tu equipo, donde vas a ir agregando tus commits y uno remoto del cual podes subir tus commits o bajarte commits que haya subido alguien.

La verdad que no entendí mucho la diferencia, pero… ¿Me mostras como trabajar?

1. La primera vez que usas GIT vas a tener que configurar tu nombre completo y tu email con los siguientes comandos:
    - ***git config --global user.name "TU NOMBRE"***
    - ***git config --global user.email "TU DIRECCION DE EMAIL"***
2. Si vas a trabajar en grupo necesitas un repositorio remoto al cual todos los integrantes tengan acceso. Hay varios servicios para esto, tres de los más conocidos son:
    - Github (https://github.com)
        i. https://help.github.com/articles/create-a-repo/

        ii. Si sos alumno, podes pedir repositorios privados desde: https://education.github.com/

    - Bitbucket (https://bitbucket.com)
        i. Repositorios privados con límite de 5 usuarios
    - Gitlab (https://gitlab.com/)
        i. Repositorios privados sin restricciones

3. Una vez creada la cuenta y un repositorio en alguno de estos servicios, tenés que bajarte la información del repositorio remoto a tu computadora.

Para eso vamos a utilizar el siguiente comando: ***git clone <url_repositorio_remoto>***
Ejemplo:

> **> git clone https://github.com/gastonprieto/sample.git**
> *Cloning into 'sample'...*
> *remote: Counting objects: 3, done.*
> *remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0*
> *Unpacking objects: 100% (3/3), done.*
> *Checking connectivity... done.*

Una vez realizado esto, habrá una carpeta con el mismo nombre que el repositorio remoto, allí se encontrará nuestro repositorio local y es donde podrás crear/modificar/eliminar archivos para versionarlos.

- ¿Y ahora qué puedo hacer dentro de esa carpeta?

- Modificar tus archivos (agregar, modificar, eliminar) :D
    - Nota: una carpeta vacía no se puede comitear

- Saber si hay cambios que aún no commiteaste: **git status**

> **> git status**
> On branch master
> Your branch is ahead of 'origin/master' by 1 commit.
> (use "git push" to publish your local commits)
>
> *Changes not staged for commit:*
> *(use "git add <file>..." to update what will be committed)*
> *(use "git checkout -- <file>..." to discard changes in working directory)*
> *modified: foo.txt*
>
> *Untracked files:*
> *(use "git add <file>..." to include in what will be committed)*
> *lala.txt*
>
> no changes added to commit (use "git add" and/or "git commit -a")

- Decirle que los cambios hechos hasta el momento de un archivo/path debería ser commiteados: **git add <path>**

  > git add foo.txt

  - Si borraste un archivo para decirle que lo queres borrar del repositorio, tenes dos opciones:
    - ***git rm <path> # si ya lo borraste o lo queres borrar***
    - ***git add -u <path> # solo si ya lo borraste***

- Saber que cosas estoy por commitear: **git status**

  > git status
  On branch master
  Your branch is up-to-date with 'origin/master'.
  Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
  new file: foo.txt

- Realizar el commit de los archivos a los que previamente le hiciste *git add,* haces: **git commit -m "MENSAJE"**

- Ver el historial de commits hechos hasta ahora, para esto tenes el comando: ***git log***

  > git log
  commit e6fd73801f44a72ee73f6863a7d263a89ed7fdb5
  Author: Gaston Prieto <gastonprietoutn@gmail.com>
  Date: Mon Mar 28 00:36:55 2016 -0300
  Initial commit

- De-seleccionar los archivos que querías commitear: ***git reset <path>***
  > git reset foo.txt
  En este caso lo que estoy haciendo es decirle a git que los cambios de foo.txt no van a ser agregados en el commit.

- Volver a la versión vieja del path, perdiendo los cambios hechos hasta el momento: ***git checkout <path>***
  - Cuidado: Podes perder cambios que te interesen

Bien hasta acá todos estos comandos te sirven para manipular tu repositorio local, ergo, los cambios que hiciste no están en el repositorio remoto.

- ¿Y entonces? Quiero que mi compañero vea mis cambios y yo bajarme los cambios que él haya hecho.

Bueno, vamos por partes, como diría Jack:

Para subir los commits de tu repositorio local lo que tenes que hacer es:

> git push origin HEAD
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 315 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:gastonprieto/sample.git
e6fd738..4d57f9c HEAD -> master

Una vez hecho esto, el repositorio remoto ya se encuentra actualizado y listo para que todos nuestros -amados- compañeros de grupo puedan bajarse esos cambios, aunque a veces esto no sea tan felíz, puede fallar con algo parecido a esto…

> git push origin HEAD
To git@github.com:gastonprieto/sample.git
! [rejected] *HEAD -> master (fetch first)*
error: failed to push some refs to 'git@github.com:gastonprieto/sample.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again*.*
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Si falla no hay que preocuparse, sólo dice que alguien subió cambios al repositorio remoto antes que vos y hay que bajarlos.

- ¿Entonces cómo los bajo?
Fácil, sólo corremos el siguiente comando.

> git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/gastonprieto/sample
e6fd738..4d57f9c master -> origin/master
First, rewinding head to replay your work on top of it...
Fast-forwarded master to 4d57f9cda140f37483fb3b95d455577b0ccc9079.

¿Viste qué era fácil?
Git va a intentar de unificar los cambios que te bajaste del repositorio remoto con los que ya tenías en el repositorio local de manera automática, pero esto también puede fallar (como todo en la vida) y decirnos que no se pueden unificar los cambios, debido a que ambos tienen cambios (probablemente) dentro del mismo archivo y en la misma línea. El error sería algo parecido a esto.

> git pull

Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; **fix conflicts and then commit the result**.

- ¿Pero entonces qué hago?
Nos arremangamos un poquito más -llorando- y abrimos el archivo que está en conflicto.

README.md
<<<<<<< HEAD # sample by fede **=======** # sample by gaston **>>>>>>> cedab6e07df01a63edfb4e95f2127a9801f5888a** Este es un sample para aprender el mínimo uso de GIT

Este paso es un poco más -divertido- manual.
Como notarás las cosas que están en conflicto se encuentran entre:
<<<<<<< HEAD
Tus cambios (lo que tenés en tu repositorio local)
=======
Los de tu compañero (los que están en el repositorio remoto)
>>>>>>> [cosa rara que identifica el commit (hash)]

- ¿Y qué hago ahora entonces con eso?
Bueno, modificas el archivo hasta que quede algo coherente (o sea, una mezcla entre tus cambios y el de tu compañero, dejar sólo lo tuyo o dejar sólo lo de tu compañero).

README.md
sample by gaston y fede Este es un sample para aprender el mínimo uso de GIT

- Ah, ¿listo?
Bueno, no. Si releemos las instrucciones del GIT cuando hicimos push y falló, nos dice que tenemos que hacer un commit. Entonces, tenemos que agregar estos cambios y commitearlos a mi repositorio local. Una vez hecho esto podemos hacer push (si nadie nos ganó de mano antes).

> ***git add README.md***
> ***git commit -m 'resolviendo problemas de merge en README.md'***
[master db7d541] resolviendo problemas de merge en README.md

> git push origin HEAD
Counting objects: 10, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 669 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To git@github.com:gastonprieto/sample.git
cedab6e..db7d541 HEAD -> master

- ¿Y ahora si terminamos con GIT?
Mmmm, no. Tiene muchas más herramientas, pero con esto te va a alcanzar para trabajar.

PD: Nunca hacer:

- ***git <comando de git cualquiera> -f***
- ***git <comando de git cualquiera> --force***
- ***Borrar la carpeta .git que se autogenera***
- ***Borrar el repositorio y clonarlo de nuevo si no te salen las cosas***