Introducción a GIT

Proyecto de software

UNAJ

Antes de comenzar

Debemos instalar GIT desde https://git-scm.com/downloads y además crear una cuenta en Github (https://github.com/) si es que no la tenemos aún.

> *NOTA: En caso de tener dificultades con la instalación de Git ver : https://medium.com/laboratoria-how-to/c%C3%B3mo-instalar-git-368c78187b51*

Configuración de cuenta.

Para Git reconozca tu cuenta debemos configurarla, para eso abrimos la consola de Git y ejecutamos...

```
$ git config --global user.name "John Doe"
---
$ git config --global user.email johndoe@example.com
```

Comenzamos

Clonar repositorio

Para comenzar clonaremos el siguiente repositorio de Github:
https://github.com/luksolivera/proyecto_software_unaj.git en alguna directorio de nuestra maquina.

```
$ git clone https://github.com/luksolivera/proyecto_software_unaj.git
```

El comando `git clone` clona un repositorio de Git a un directorio local.

Crear un nuevo archivo

Nos posicionamos dentro del directorio y creamos un archivo de texto que llamaremos "Agenda.txt"

```
$ cd proyecto_software_unaj/

$ nano Agenda.txt
```

agregaremos unos numeros de telefono a nuestra agenda.

```
$ cat Agenda.txt

Carlos = 123456789
```

podemos visualizar el cambio que realizamos al repositorio ejecutando el comando `git status`.

> *El comando `git status` : Nos indica el estado del repositorio, por ejemplo cuales están modificados, cuales no están siendo seguidos por GIT, entre otras características.*

```
$ git status

On branch master
```

```
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Agenda.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Git nos indica que tenemos un archivo que no se ha añadido, es decir se encuentra en estado modificado pero todavia no esta preparado.

Para añadirlo, realizamos el siguiente comando.

```
$ git add Agenda.txt

---

$ 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:   Agenda.txt
```

> `git add + path` : *Agrega al repositorio los archivos que indiquemos.*

> `git add -A` : *Agrega al repositorio TODOS los archivos y carpetas que estén en nuestro proyecto, los cuales GIT no está siguiendo.*

en este instante el archivo `Agenda.txt` se encuentra añadido.

Modificar un archivo

Abrimos el archivo de `Agenda.txt` y agregamos un nuevo número.

```
$ cat Agenda.txt

Carlos = 123456789
Lucas = 987654321
```

analizamos el estado del repositorio

```
$ 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:   Agenda.txt

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:   Agenda.txt
```

Git nos informa que tenemos un archivo preparado y un archivo que fue modificado, es decir, si confirmaremos en este momento los cambios, no se guardarán los cambios en el repositorio.

> - *Para ver qué has cambiado pero aún no has preparado, escribe `git diff` sin más parámetros*

> - *Si quieres ver lo que has preparado y será incluido en la próxima confirmación, puedes usar `git diff --staged`. Este comando compara tus cambios preparados con la última instantánea confirmada.*

> - *Puedes usar `git diff +Path` para ver los cambios en el archivo que ya están preparados y los cambios que no lo están. Si nuestro ambiente es como este:*

```
$ git diff Agenda.txt

warning: LF will be replaced by CRLF in Agenda.txt.
The file will have its original line endings in your working directory.
diff --git a/Agenda.txt b/Agenda.txt
index 8087229..d7b07ba 100644
--- a/Agenda.txt
+++ b/Agenda.txt
@@ -1 +1,2 @@
 Carlos = 123456789
+Lucas = 987654321
```

Confirmar los cambios

Realizaremos la confirmación de los cambios.

> *El comando `git commit` confirma nuestros cambios. habitualmente se agrega `-m` para realizar un comentario a la confirmación. Utilizando `-am` en el commit estamos realizando el comando `git add` a los archivos que fueron modificados.*

```
$ 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:   Agenda.txt

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:   Agenda.txt
```

```
$ git add -A
```

```
$ git commit -m "commit de agenda"

[master 1342517] commit de agenda
 1 file changed, 2 insertions(+)
 create mode 100644 Agenda.txt
```

Logs

podemos ver el historial de cambios del repositorio con la siguiente instrucción

> `git log` *muestra el historial de commits que posee el branch, utilizaremos* `git log --graph` *para obtener una mejor visión.*

```
$ git log --graph

* commit 1342517e0356804fc69e583faa46188e9ae1136a (HEAD -> master)
| Author: luksolivera <luksolivera10@gmail.com>
| Date:   Tue Feb 25 16:45:36 2020 -0300
|
|     commit de agenda
|
* commit 288adf2fb0070aa9657d146b47afa8856014b02f (origin/master, origin/HEAD)
| Author: luksolivera <luksolivera10@gmail.com>
| Date:   Tue Feb 25 16:15:48 2020 -0300
|
|     tutorial
|
* commit 25c1efe5e57b674dfb5fcd76a637ee1526b0b484
  Author: luksolivera <31370056+luksolivera@users.noreply.github.com>
  Date:   Sun Feb 16 17:48:18 2020 -0300

      Initial commit
```

Podremos notar el hash del commit, autor, fecha y mensaje que incluyó dentro del commit.

Branches

Git organiza los commit en forma de árbol, de esta forma podemos ir abriendo branch o ramas que parten bien de la rama principal (master) o de otra rama (branch). Una de las ventajas de las ramas es que podemos desarrollar nuevas funcionalidades sin afectar a una versión estable que se encuentre en master.

> *Para concocer mas el funcionamiento de las branch ver: https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell*

Primer paso, vamos a crear una nueva rama en el repositorio.

> `git branch` *nos muestra el listado de las ramas que existen. para crear una rama utilizamos* `git branch <nombre-branch>` .

Creamos una rama con su nombre y apellido.

```
$ git branch ProyectoSoftware

---

$ git branch

  ProyectoSoftware
* master
```

Para realizar un cambio de rama utilizamos el comando `checkout`

> `git checkout <nombre-rama>` *Sirve para moverse entre branches, en este caso vamos al branch que indicamos en el comando.*

```
$ git checkout ProyectoSoftware

Switched to branch 'ProyectoSoftware'
```

Realizar en esta rama :

1. El estado de la rama con git status.
2. El estado del archivo Agenda.txt.
3. Visualizar el log.
4. Modificar el archivo Agenda agregando un nuevo celular y borrando uno que ya existía.
5. Visualizar las diferencias.
6. Confirmar los cambios.
7. Revisar el log.

Resultado esperado:

```
$ git log --graph

* commit 9887a8fba2aa0abbe3e4a6a6c84f1432bff35b2c (HEAD -> ProyectoSoftware)
| Author: luksolivera <luksolivera10@gmail.com>
| Date:   Tue Feb 25 20:04:19 2020 -0300
|
|     primer commit en rama
|
* commit 1342517e0356804fc69e583faa46188e9ae1136a (master, LucasOlivera)
| Author: luksolivera <luksolivera10@gmail.com>
| Date:   Tue Feb 25 16:45:36 2020 -0300
|
|     commit de agenda
|
* commit 288adf2fb0070aa9657d146b47afa8856014b02f (origin/master, origin/HEAD)
| Author: luksolivera <luksolivera10@gmail.com>
| Date:   Tue Feb 25 16:15:48 2020 -0300
|
|     tutorial
|
* commit 25c1efe5e57b674dfb5fcd76a637ee1526b0b484
  Author: luksolivera <31370056+luksolivera@users.noreply.github.com>
  Date:   Sun Feb 16 17:48:18 2020 -0300
```

```
    Initial commit
```

8. Cambiar a la rama master.
9. Visualizar el archivo Agenda.
10. Agregar nuevos contactos a la agenda.
11. Cambiar de rama.

    Resultado esperado:

```
$ git checkout ProyectoSoftware

error: Your local changes to the following files would be overwritten by
checkout:
   Agenda.txt
Please commit your changes or stash them before you switch branches.
Aborting
```

> *Cómo realizamos cambios y los commit de las ramas son distintas, git no nos permite realizar el cambio de rama debido a que podemos perder los cambios realizado.*

12. Confirmamos los cambios.
13. Visualizamos el Log

Merge

Con las actividades realizadas, tenemos dos ramas que se encuentran en diferentes commit y por lo tanto en diferentes versiones de nuestro repositorio, se realiza un merge para unificar las ramas.

> `git merge <NombredeRama>` *: Hace un merge entre dos branches, en este caso la dirección del merge sería entre el branch que indiquemos en el comando, y el branch donde estemos ubicados.*

Nos ubicamos en la rama master y procedemos a realizar el merge.

```
$ git merge ProyectoSoftware

Auto-merging Agenda.txt
CONFLICT (content): Merge conflict in Agenda.txt
Automatic merge failed; fix conflicts and then commit the result
```

Al modificar el mismo archivo en las dos ramas, Git nos informa que ha ocurrido un conflicto y fallo el merge. Si analizamos el archivo Agenda.

```
$ cat Agenda.txt

Carlos = 123456789
<<<<<<< HEAD
Lucas = 987654321
Robert = 15464644
=======
Maria = 987654321
```

```
Jose = 541646544
>>>>>>> ProyectoSoftware
```

Git nos crea esas etiquetas para que sepamos los datos que existen en una rama y los que existen en otra. La sección HEAD apunta a la rama en la que estamos parados.

Procedemos a eliminar la etiqueta y mantener toda la info

```
$ cat Agenda.txt

Carlos = 123456789
Lucas = 987654321
Maria = 987654321
Jose = 541646544
```

Debemos confirmar esta corrección.

```
$ git commit -am "merge"

[master 7c4847d] merge
```

```
$ git log --graph

*   commit 7c4847d82f8739108dfe499a75a9c2fbf81534c9 (HEAD -> master)
|\  Merge: 206a7a1 9887a8f
| | Author: luksolivera <luksolivera10@gmail.com>
| | Date:   Tue Feb 25 20:40:00 2020 -0300
| |
| |     merge
| |
| * commit 9887a8fba2aa0abbe3e4a6a6c84f1432bff35b2c (ProyectoSoftware)
| | Author: luksolivera <luksolivera10@gmail.com>
| | Date:   Tue Feb 25 20:04:19 2020 -0300
| |
| |     primer commit en rama
| |
* | commit 206a7a1f65c08268ab898691e28a5d6f07be311c
|/  Author: luksolivera <luksolivera10@gmail.com>
|   Date:   Tue Feb 25 20:22:01 2020 -0300
|
|       modified agenda
|
* commit 1342517e0356804fc69e583faa46188e9ae1136a (LucasOlivera)
| Author: luksolivera <luksolivera10@gmail.com>
| Date:   Tue Feb 25 16:45:36 2020 -0300
|
|     commit de agenda
|
```

Pull y Push

> `git pull origin <branch>` : *Descargamos los últimos cambios que se encuentran en el repositorio.*

> *`git push origin <branch>`: Subimos los cambios (commits) que realizamos al repositorio.*

> *NOTA: Cuando se trabaja en equipo, siempre se debe realizar un pull antes de un push, debido a que puede ser que un compañero haya realizado cambios y no poseas la última versión.*

```
$ git pull origin master

From https://github.com/luksolivera/proyecto_software_unaj
 * branch            master     -> FETCH_HEAD
Already up to date.
```

Creamos una nueva rama desde master llamada "nombreApellidoFecha" y realizamos el push

```
$ git push origin ProyectoSoftwareFecha

Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'ProyectoSoftwareFecha' on GitHub by visiting:
remote:
https://github.com/luksolivera/proyecto_software_unaj/pull/new/ProyectoSoftwareFecha
remote:
To https://github.com/luksolivera/proyecto_software_unaj.git
 * [new branch]      ProyectoSoftwareFecha -> ProyectoSoftwareFecha
```