GIT & GITHUB
GUÍA PRÁCTICA

Jesús Arce
info.jess@yahoo.com.ar
Creado: 01 Nov 2019
Actualizado: 17 Mar 2020

Resumen

En esta guía práctica te propongo realizar dos ejercicios:

Ejercicio 1

- Crear un repo local con Git y configurarlo
- Crear un repo remoto (vacío) con GitHub
- Vincular ambos repositorios
- Trabajar en el repo local y "comitear" los cambios
- Subir los cambios al repo remoto

Ejercicio 2

- Crear un repo remoto (vacío) con GitHub
- Descargar por primera vez el repo remoto en nuestra máquina local (clone)
- Trabajar en el repo local y "comitear" los cambios
- Subir los cambios al repo remoto

Pero… Los ejercicios son muy parecidos! ¿Cuál es la diferencia?.

La diferencia principal reside en que en el Ejercicio 1 aprendemos a vincular un repositorio local al repositorio remoto, mientras que en el Ejercicio 2 este paso no es necesario ya que al descargar (clonar) el repositorio remoto, estamos creando un repositorio local *ya sincronizado* automáticamente con el remoto.

El Ejercicio 2 es un caso común de trabajar con un repositorio remoto que ha creado otra persona, entonces lo descargamos para poder utilizarlo localmente.

Bien, y qué necesito para empezar?

Leer la Guía de bolsillo y tener muchas ganas de aprender practicando =)

Ahora bien, pasemos a la acción:

Ejercicio 1:

1- Crear un directorio local donde estará almacenado nuestro proyecto:

`mkdir <nombreDelDirectorio>`

2- Dirigirse a él:

`cd <directorio>`

3- En el directorio, inicializar el repositorio local:

`git init`

Adicionalmente podemos también configurar nuestro **nombre de usuario**

con el comando: `git config user.name <tu_nombre>`

y nuestro email

con el comando: `git config user.email <tu_email>`

4- Crear un repositorio remoto (vacío por el momento) en GitHub

5- Vincular el repositorio local con el repositorio remoto

con el comando: `git remote add origin <url del repo remoto>`

6- Crear archivos y darles seguimiento con los comandos:

`git add <archivo>` //agrega un archivo particular a la zona de Stage

`git add .` //agrega todos los archivos a la zona de Stage

7- Confirmar los cambios haciendo un commit:

`git commit -m "mensaje del commit"` // pasamos de stage al head

8- Enviar los cambios al repositorio remoto:

`git push origin master`

Nota: los pasos 6 al 8 se pueden repetir tantas veces como funcionalidades vayamos sumando al proyecto.

¿Algo anduvo mal?

¿No te salió el Ejercicio 1?, no te preocupes! A continuación, te muestro paso a paso cómo lo resolví en mi máquina:

```
$ mkdir curriculum   //creamos una carpeta en algún lugar de nuestro disco duro

$ cd curriculum/     //ingresamos a la carpeta

$ git init           //inicializamos un repositorio local

Initialized empty Git repository in C:/Users/Yisus/Desktop/curriculum/.git/
```

Ahora que tenemos un repositorio local, podemos pedirle y preguntarle un montón de cosas. Por ejemplo, podríamos preguntarle su estado:

```
$  git status

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

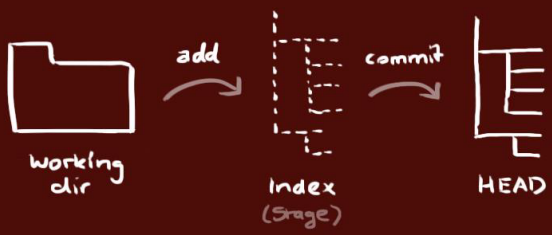
Ya sabemos cuál es su estado. Por lo que ahora procedemos a crear un archivo y ver cómo queda el estado del repo local:

```
$ touch curriculum1.txt    // creamos un archivo

$ git status               // preguntamos el estado

On branch master

No commits yet

Untracked files:

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

      curriculum1.txt

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

Como podemos observar, tenemos un archivo que no tiene seguimiento, por lo que ahora estamos en condiciones de agregar el archivo a la zona intermedia de “Stage”.

Nota: la zona intermedia contiene un índice de todos los archivos que están bajo el control de versiones y listos para ser comiteados.

Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu `Directorio de trabajo` que contiene los archivos, el segundo es el `Index` que actua como una zona intermedia, y el último es el `HEAD` que apunta al último commit realizado.




```
$ git add curriculum1.txt
```

Nota: si queremos agregar todos los archivos en vez de uno solo, simplemente utilizamos el comando `git add .` (no olvidar el punto del final).

```
$ git status
On branch master
No commits yet
Changes to be committed:
 (use "git rm --cached <file>..." to unstage)
      new file:   curriculum1.txt
```

¡Bien hecho! Acabamos de agregar un primer archivo a la zona de “Stage”. Esto le indica a git que queremos darle seguimiento a dicho archivo.

¿Ya terminamos de crear nuestra primera versión?

No. Sólo creamos un repositorio local, y agregamos un archivo a la zona Stage, pero no es suficiente: siempre que queramos crear una nueva versión tendremos que confirmar los cambios realizados. Para ello, simplemente tenemos que confirmar nuestro escenario actual, que está representado por la zona Stage. Es decir, vamos a sacar una “foto” de nuestro escenario con el comando “commit”, de la siguiente manera:

```
$ git commit -m "Primera Versión"
[master (root-commit) dfbac62] Primera Versión
```

```
 1 file changed, 0 insertions(+), 0 deletions(-)

 create mode 100644 curriculum1.txt
```

¡Y ahora sí!
¡Tenemos nuestra primera versión local!

Todo muy lindo hasta acá, pero el problema es que hasta ahora todo está en nuestra máquina. Si se rompe, si lo borramos por accidente, o si simplemente no tenemos nuestra computadora cerca, no podremos acceder a estos contenidos. ¿La solución? ¡Guardar nuestro repositorio en la nube (o en otra máquina)!

Para hacerlo, necesitamos primero vincular nuestro repo local con un repo remoto, al que llamaremos **origin**, mediante el comando:

```
$ git remote add origin https://github.com/MiUsuarioDeGitHub/NombreRepoRemoto.git
```

¡Bien hecho!

Ahora lo que tenemos que hacer es enviar los cambios que estuvimos haciendo en nuestro Repo local, hacia el Repositorio remoto.

Para ellos utilizaremos el comando git push origin master, de la siguiente manera:

```
$ git push origin master

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 226 bytes | 226.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/ProfeJesuArce/curriculumTest.git

 * [new branch]      master -> master
```

Nota: la palabra master hace referencia a la rama principal de nuestro repositorio remoto.

¡EXCELENTE!
Ahora podemos revisar nuestro repositorio remoto y chequear que se encuentre el archivo curriculum.txt

De esta manera podemos trabajar en nuestro proyecto local, haciendo cambios y creando nuevas versiones, para luego subirlas en el repositorio remoto.

Muy bien, ahora quedó más claro. Sigamos adelante…

Ejercicio 2:

1- Crear un directorio local donde estará almacenado nuestro proyecto:

```
mkdir <nombreDelDirectorio>
```

2- Dirigirse a él:

```
cd <directorio>
```

3- En el directorio, descargar el repositorio remoto:

```
git clone https://github.com/<usuario>/<nombreDelRepo.git>
```

> **Nota:** este paso se realiza por única vez al principio. Se creará el repositorio local como un "clon" del repositorio remoto. Además, el repo local quedará vinculado al repo remoto automáticamente. Esto nos permitirá subir los cambios locales y descargar el proyecto actualizado al comenzar el día de trabajo.

4- Crear archivos y darles seguimiento con los comandos:

```
git add <archivo>   //agrega un archivo particular a la zona de Stage

git add .           //agrega todos los archivos a la zona de Stage
```

5- Confirmar los cambios haciendo un commit:

```
git commit -m "mensaje del commit"  // pasamos de stage al head
```

6- Enviar los cambios al repositorio remoto:

```
git push origin master
```

Nota: los pasos 4 al 6 se pueden repetir tantas veces como funcionalidades vayamos sumando al proyecto.

Eso fue fácil pero… Si alguien actualiza el repo Remoto con nuevos cambios, ¿cómo hago para trabajar con esa última versión?

El comando que veremos a continuación es válido tanto para el Ejercicio 1 como para el Ejercicio 2, ya que lo que queremos hacer es obtener la última versión del Repositorio Remoto.

```
git pull origin master
```

Este comando suele utilizarse cuando se trabaja en equipo, ya que un compañero de trabajo puede haber subido nuevos cambios estables al repositorio remoto y nosotros al comenzar el siguiente día de trabajo vamos a necesitar trabajar con el proyecto actualizado. Así que antes de comenzar a trabajar en el proyecto, ejecutamos el comando y listo! Estamos listos para comenzar el día =)

Material adicional:

https://diegomalagon.com/lista-de-comandos-utiles-de-git/

https://rogerdudler.github.io/git-guide/index.es.html

https://rogerdudler.github.io/git-guide/index.es.html

https://learngitbranching.js.org/