Notas de Git de Rodolfo en CLI

August 17, 2016

1 Initializando el repositorio

Supondré que git ya está instalado y configurado.

Digamos que tenemos un repositorio que ya existe, podemos inicializar una copia local,

```
$ git init
Initialized empty Git repository in /home/rho/Documents/GitHub/.git/
$git remote add origin git@github.com:Rhodolfo/Ejemplo.git
$git remote -v
origin git@github.com:Rhodolfo/Ejemplo.git (fetch)
origin git@github.com:Rhodolfo/Ejemplo.git (push)
```

Este mensaje simplemente nos dice que el repositorio local se ha creado correctamente y que está atado al repositorio remoto.

Alternativamente, podemos clonar un repositorio,

```
$ git clone git@github.com:Rhodolfo/Ejemplo.git Ejemplo
Cloning into 'Ejemplo'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100\% (3/3), done.
Checking connectivity... done.
```

2 Guardando Cambios

En el repositorio local empecé a escribir estas notas, al hacer este cambio el repositorio local y el remoto difieren,

```
$ git status
On branch master
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:   README.md

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

notas.tex

no changes added to commit (use "git add" and/or "git commit -a")
```

Git nos dice que se ha cambiado README.md, también indica que hemos creado un nuevo archivo notas.tex.

Podemos pedirle a git que guarde estos cambios,

```
$ git add README.md
$ git add notas.tex
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file$..." to unstage)

modified:   README.md
new file:   notas.tex
```

Ahora tiene sentido sincronizar nuestros cambios con el remoto,

```
$git commit -m "Cambio en README.md, nuevo archivo notas.tex"
[master 1626734] Cambio en README.md, nuevo archivo notas.tex
 2 files changed, 59 insertions(+)
 create mode 100644 notas.tex
```

Si estamos contentos,

```
$ git push origin master
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100\% (3/3), done.
Writing objects: 100\% (4/4), 1.09 KiB | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To git@github.com:Rhodolfo/Ejemplo.git
   67c1f32..1626734  master -$ master
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

Uno puede hacer la operación inversa, en este caso los cambios del repositorio remoto se actualizaran al local,

```
$ git pull origin master
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 5 (delta 1), pack-reused 0
Unpacking objects: 100% (5/5), done.
From github.com:Rhodolfo/Ejemplo
 * branch            master     -> FETCH_HEAD
   7cb7566..41d09a5  master     -> origin/master
Updating 7cb7566..41d09a5
Fast-forward
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
```

Pull 'jala' el remoto al local, push 'empuja' el local al remoto.

3 Ramas

Normalmente uno no cambia el repositorio principal del proyecto, conviene crear una 'rama,' las ramas son copias de un repositorio. Las ramas permiten trabajo independiente entre distintos dessarrolladores, uno puede crear, combinar y borrar ramas.

Para crear una rama llamada 'uno,'

```
$ git branch uno
M       notas.tex
Switched to branch 'uno'
```

Para ver una lista de ramas,

```
$ git branch
  dos
* master
  tres
  uno
```

en este caso existen las ramas uno, dos, tres y master. El usuario está trabajando sobre la rama master.

Para cambiar de rama se usa checkout,

```
$ git checkout uno
Switched to branch 'uno'
```

Uno puede hacer git push y git pull de la misma manera que con la rama master,

```
$ git push origin uno
$ git pull origin uno
```

NOTA IMPORTANTE: Al hacer un pull o un push uno debería asegurarse de que las ramas locales y remotas correspondan, pulls deben ser hechos de la rama 'uno' a la rama 'uno', pulls deben ser hechos de la rama 'uno' a la rama 'uno.' Si las ramas difieren, se empezará una operación de merge (combinación).

Con esto en mente, el modo de trabajo es exactamente igual que con la rama master,

```
$ git add README.md
$ git commit -m "Cambio en rama uno"
[uno 6fbb706] Cambio en rama uno
 1 file changed, 2 insertions(+)
$ git push origin uno
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 346 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:Rhodolfo/Ejemplo.git
 * [new branch]      uno -$ uno
$ git push origin uno
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 346 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@github.com:Rhodolfo/Ejemplo.git
 * [new branch]      uno -> uno
```

la rama master no es especial de ninguna manera, simplemente es la rama original, puede ser borrada y cambiada de nombre.

Personalmente, yo tomo la rama principal y creo una rama personal del proyecto original. Al finalizar mis cambios, me encargo de hacer un merge (combinar ramas).

4 Colaboración y Conflictos

Digamos que soy otro desarrollador y estoy trabajando sobre la rama 'dos', donde también se ha cambiado el README.

Podemos hacer un merge, el cuál junta los cambios y pide al usuario que resuelva conflictos. Pero antes debemos bajar la rama 'uno' a nuestro repositorio local,

```
$ git checkout -b uno
$ git branch uno
$ git pull origin uno
```

creamos la rama uno localmente si no existe, cambiamos a ella y hacemos un pull para sincronizar con el repositorio remoto.

Luego vamos de nuevo al repositorio dos,

```
$ git checkout dos
```

Trabajando desde la rama dos:

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

Tenemos un conflicto en el archivo README.md entre las ramas uno y dos.

```
$ git status
On branch dos
Your branch is up-to-date with 'origin/dos'.
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file$..." to mark resolution)

        both modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Tenemos que entrar a editar README.md, en este caso tendrá los cambios de las ramas uno y dos, escojeremos a mano como queremos que se quede el archivo.

Al estar contentos podemos hacer un git add, luego commit y un push como antes.

```
$ git add README.md
$ git status
On branch dos
Your branch is up-to-date with 'origin/dos'.
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:

modified:   README.md

$ git commit -m "Dos unido a Uno en rama Dos"
[dos 955b403] Dos unido a Uno en rama Dos
```

esto ya está listo para hacer un push de manera usual,

```
$ git push origin dos
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 347 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To git@github.com:Rhodolfo/Ejemplo.git
   d809992..955b403  dos -$ dos
```

Y ya tenemos ambos cambios guardados en la rama dos.

De igual manera se pueden combinar cualesquiera ramas, incluyendo la rama master.