Guía rápida `git`

v1.1

Justin A. Castillo V.

Portolio

Linkedin

Github profile

 Summit

Comandos básicos

`git init`

- Inicia un nuevo repositorio de git.

`git branch`

- Muestra la rama actual.

`git checkout -b branchName`

- Crea la rama `branchName` .

`git checkout branchName`

- Posiciona el `HEAD` [1] en la rama `branchName`

`git add .`

- Añade los archivos a la `staging area` [2].

> **Tip**: se puede añadir un archivo específico al utilizar el comando `git add filename` .

`git commit -m"msg"`

- Añade un comentario específico al commit.

> **Tip:** se puede usar `m` sin comillas para abrir el editor de código por defecto.

`git fetch`

- Descarga las referencias y objetos desde un **remote** [3]

`git pull`

- Trae los nuevos cambios desde **remote**.
- Por defecto, `git` trata de hacer un `merge` "fast forwardable" [4] .

`git push`

- Envía el nuevo `commit` al *remote repository*.

Tip: cuando se cree una nueva rama localmente y se necesite subir los cambios se debe utilizar el flag `git push --set-upstream remoteName branchName` (en muchos casos "remoteName" es *origin*).

`git status`

- Lista el estado de los archivos del proyecto.

Tip: al utilizar el *flag* `-s` se logra visualizar de una forma más compacta el estado.

`git clone URL`

- Clona un proyecto desde un **remote.**

Tip: al utilizar el formato `git clone url directoryName` clona el repository en *directoryName*.

`git diff`

- Muestra exactamente las líneas añadidas y removidas. *Muestra los cambios hechos pero que no se encuentran en la* `staging area` .

Tip: se puede utilizar el flag `--staged` para visualizar los cambios que se encuentran en la `staging area` y podrán ir en el siguiente `commit` .

`git checkout -- fileName.format`

- Remueve los cambios en el archivo especificado

`git rm fileName`

- Quita el seguimiento de un archivo por parte de `git` .

Tip: si el archivo se encuentra en la **staging area** es posible que se deba añadir el flag `-f` para forzar el retiro del seguimiento.

`git log`

- Permite visualizar el historial de `commits` .

Tip: al añadir el flag `-p` se permite visualizar los cambios introducidos en cada `commit` . De igual manera, se puede añadir el flag `-2` para mostras las últimas dos entradas (el número se puede cambiar).

Comandos avanzados

`git log --all --decorate --oneline --graph`

- Muestra un árbol registro de los cambios hechos en cada rama, con información relevante, como el autor, hash del `commit` , etc.

`git branch -d local_branch_name`

- Elimina localmente una rama.

Tip: para eliminar una rama en el **remote** se usa el comando `git push origin --delete remote_branch`

`git merge branch_name`

- Fusiona la rama `branch_name` en la rama actual.

Tip: recordar que para saber el nombre de la rama actual basta con usar el comando `git branch`. De esa manera por ejemplo, si la rama actual fuera `featureX` entonces fusionaríamos el contenido de `branch_name` en `featureX`.

Notas y recomendaciones

- Es conveniente utilizar *git bash* como terminal predeterminada, ya que según mi experiencia puedo decir que es mucho más intuitiva y además se pueden encadenar comandos al añadir `&&` después de escribir cada uno.
- No es importante cuál sea el proveedor cloud para git (*github, gitlab, bitbucket*); los comandos funcionarán de igual manera, es git!.
- Es recomendable no utilizar clientes de `git` en la medida, ya que al utilizar la terminal, se está en un entorno mucho más controlable.

Glosario

1. **HEAD:** referencia al último `commit` realizado en una rama.
2. **Staging area:** zona de *git* donde se encuentran los archivos que fueron creados o modificados y que fueron añadidos mediante el comando `git add .` o `git add filename`, estando listos para `commit`.
3. **Remote:** hosting del repositorio.
4. **Fast Forward:** tipo de fusión que *git* trata de realizar por defecto. Para efectos de prácticidad, la explicación total no se incluirá en ésta guía; más información sobre *Fast forward* aquí