Conceptos y comandos de git

Una vez que tengas instalado GIT y esté configurado, podrás disponer y utilizar los siguientes comandos. Aunque puedes utilizar perfectamente bien GIT con algunos cuantos, es bueno conocerlos y su función, por si un día necesitas realizar una acción diferente.

GIT Help

Para conocer TODO lo que puedes hacer con GIT, este comando te permite ver una descripción completa sobre todos los comandos y funciones de GIT

```
$git --help
```

Como podrás observar, puedes hacer combinaciones para conocer la descripción de cada comando. Por ejemplo:

```
$git --help add
```

Como podrás observar, al estar dentro de la terminal, la manera de moverte será tecleando:

"f" - Forward o adelante
"b" - Backward o atrás
"q" - Salir

Iniciando un repositorio

El siguiente paso es empezar a **monitorear** un proyecto. Para ello, nos tenemos que situar en la carpeta de nuestro proyecto.

```
$ cd miproyecto
$ git init
```

git init. Es un comando que le indica a GIT que empieza a seguir todos los cambios del proyecto. Siempre ejecútalo cuando inicies.

Una vez que empiece el rastreo, puedes crear archivos y carpetas. Imaginemos que creamos un README.txt:

```
$ cd mi proyecto
$ touch README.txt
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#    README.txt
nothing added to commit but untracked files present (use "git add" to
track)
```

git status. Este comando indica los nuevos cambios que tiene el repositorio, así como sugerencias de acción para los archivos rastreados. En este ejemplo vemos como aparece el README.txt, el cual está identificado en nuestro espacio de trabajo, más no hemos agregado sus cambios al historial.

Para ello, utilizaremos el comando:

```
$ git add README.txt
```

Con esto, GIT prepara el archivo README.txt para encapsularlo al historial de cambios. Está en una fase de preparación. Si ponemos git status de nuevo, lo veremos más claro:

```
$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#    new file:   README.txt
#
```

Si quisiéramos agregar varios archivos (lo común), podemos utilizar:

```
$ git add -A
```

El último paso para agregarlo al historial de cambios es "commit". Utilizaremos el comando "git commit" para guardarlo en memoria permanente dentro de nuestro repositorio.

```
$git commit -m "Modificamos el título y dejamos una descripción más
clara"
[master e5dbe30] Modificamos el título y dejamos una descripción más
clara
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.txt
```

Listo, acabamos de hacer nuestro primer commit. El significado de -m es message y funciona para crear la descripción ó indicación de lo que significa el commit. Es muy importante ser específico y puntual para que las personas que descarguen tu repositorio sepan que significa ese cambio, actualización, arreglo de bug, etc.

Para poder ver el historial de todos los cambios en el repositorio, utilizaremos un comando llamado git log:

```
$ git log
commit e5dbe30547849e2fa46650e4ae495c47a40e57ab
Author: Miguel Nieva <miguelnieva@conexionew.com>
Date:   Mon May 13 11:50:35 2013 -0500
    Modificamos el título y dejamos una descripción más clara
```

La maravilla de GIT es que puedes tener obtener diferentes datos útiles a partir de este historial de cambios. Por ejemplo, la clave única del commit, el autor, la fecha e incluso, con comandos más avanzados, veremos comparaciones entre cambios.