git_started

June 21, 2017

1 Git started. Manual sencillo de git para viajeros en el tiempo

---

by [@juan_qenk](https://twitter.com/juan_qenk)

Imaginad que recibimos una invitación de Stephen Hawking para una fiesta que se celebró en Cambridge el 28 de junio de 2009. No se ha perdido la invitación en el correo, se mandó a proposito una vez finalizada la fiesta. La fiesta a la cual habeis sido invitados es una fiesta para viajeros en el tiempo.

Me parece una falta de respeto no pasar siquiera a saludar. Los buenos modales no son algo que el tejido espacio-tiempo deban coartar. Dispongo de lo necesario para esta breve historia del tiempo: linux, conocimiento inicial en python y atuendo de la época.

Para el desarrollo de un software de tal calibre es necesario trabajar con un sistema de control de versiones. Git es el más usado en la actualidad, ideal para el viajero en el tiempo. Es posible que tu equipo lo tenga instalado, pero si no es así, puedes usar los siguientes comandos:

```
En Debian/Ubuntu y derivados:
sudo apt-get install git

En Fedora y derivados:
sudo yum install git

En Arch y derivados:
sudo pacman -S git
```

Una vez instalado, estas son las versiones con las que vamos a viajar.

```
In [2]: git --version
        bash --version
```

```
git version 2.11.0
GNU bash, versión 4.4.7(1)-release (x86_64-pc-linux-gnu)
Copyright (C) 2016 Free Software Foundation, Inc.
Licencia GPLv3+: GPL de GNU versión 3 o posterior <http://gnu.org/licenses/gpl.html>

Esto es software libre; siéntase libre para cambiarlo y redistribuirlo.
No existe NINGUNA GARANTÍA, en la medida permitida por la ley.
```

Yo ya os he presentado git, pero él también quiere conoceros. No olvidéis completar la configuración inicial de git con vuestro nombre de usuario e email.

```
git config --global user.name viajerotemporal
git config --global user.email viajerotemporal@ejemplo.com
```

Podeis comprobar vuestras credenciales escribiendo:

```
git config --list
```

Si en algún momento os asalta una duda, no dudéis en consultar la guía del autoestopista del tiempo

```
man git
```

Es este un buen momento para crear vuestro entorno de trabajo. Es peligroso eso de ir programando viajes temporales sin orden, corremos el riesgo de equivocarnos y acabar como brasas en una barbacoa medieval. Hemos llamado gitstarted a nuestro directorio de trabajo, de momento vacío. No por mucho tiempo.

```
In [5]: mkdir gitstarted
        cd gitstarted
        pwd
```

Nuestro directorio de trabajo necesita algo de contenido, se siente solo. Cualquier proyecto que pretenda hacerle twerking a las leyes de la física debe estar organizado al detalle y los dos archivos que más nos ayudarán a acotar y definir el uso son README.md y licence.txt.

Podemos ir rellenando el README sobre la marcha:

```
In [7]: echo '# git started  ' > README.md

In [8]: echo '## Manual para impacientes  ' >> README.md
        echo '- Instalación en linux  ' >> README.md
        echo '- Configuración inicial  ' >> README.md
```

Hasta el momento hemos conseguido completar los primeros pasos de la descripción de nuestro programa.

```
In [9]: cat README.md

# git started
## Manual para impacientes
- Instalación en linux
- Configuración inicial
```

En el caso de que tengamos todos los puntos claros y sepamos como vamos a recomendar la instalación, es más fácil hacerlo con nuestro editor de textos de confianza, aquel a quién confiamos nuestros secretos más turbios, por quién lanzamos flames, el cual nos convierte en trols. O cualquier otro en su defecto, como

```
In [2]: gedit
```

De la elección de la licencia no hablaré hoy. Un asunto tan importante merece su propio espacio. Siendo conciso y en concreto; el viaje en el tiempo es libre o no es.

Por último, no olvidéis ni compatáis el pasaporte. Los documentos personales es bueno llevarlos siempre sin mostrarlos mucho. Evitaremos ser suplantados por androides.

```
In [15]: touch .info_privada.txt
         ls -a

.  ..  .info_privada.txt  README.md
```

Tenemos el entorno anhelando contenido, es el momento de inyectarle un poco de git. Al hacerlo habremos ganado control sobre el desarrollo de nuestros programas.

Trabajando con git guardamos los cambios que vamos haciendo de tal forma que nos resultará más fácil volver a una versión anterior. En ese sentido, lleva algo de viaje en el tiempo implícito. En adelante podremos crear ramas para añadir o modificar características sin que estos cambios estropeen el programa principal.

```
In [16]: git init

Initialized empty Git repository in /home/juan/programacion/gitstarted/.git/
```

Hemos iniciado git en nuestro directorio de trabajo con éxito. Ahora podemos comprobar que se ha creado una carpeta oculta donde se irán guardando los cambios que hagamos.

```
In [18]: pwd
         ls -a

/home/juan/programacion/gitstarted
.  ..  .git  .info_privada.txt  README.md
```

El arbol de directorios y archivos puede quedar como se muestra a continuación. Podéis instalar el programa tree fácilmente haciendo

```
sudo apt install tree
```

y visualizarlo con

```
tree -a
```

Recordad que tenemos información privada en nuestro directorio de traqbajo. Git entenderá que forma parte del código y, si estamos subiendo los cambios en un servidor remoto como github, los hará públicos. No queremos que así sea. Para evitarlo solo tenemos que crear un archivo .gitignore y añadir lineas con los archivos o carpetas que queremos que sean ignorados por git.

```
In [21]: echo '.info_privada.txt' > .gitignore
```

Nos hemos tomado un poco más tiempo del habitual preparando el entorno de trabajo pero nos ayudará a tener un codigo organizado y accesible. Además, que importa dedicar unos minutos extra cuando pretendemos crear una máquina del tiempo que nos permitirá viajar al pasado.

Trabajar con git es trabajar en tres espacios virtuales: el directorio de trabajo, el stage(index) y history o repository. Haremos cambios de nuestro código en el directorio de trabajo. Estos cambios los añadiremos al stage (git add) donde irán reuniéndose. Una vez queramos guardar esos cambios, los pasamos a history añadiendo un comentario (git commit)

```
Nota: git status es un comando útil de git que nos dirá el estado de este flujo de trabajo en u
```

```
In [23]: git add .
```

Así hemos añadido todos los archivos al stage. Podemos comprobar qué archivos hemos añadido con el siguiente comando:

```
In [24]: git status
```

```
En la rama master

Commit inicial

Cambios para hacer commit:
  (use ńgit rm --cached <archivo>...ż para sacar del stage)

        nuevo archivo: .gitignore
        nuevo archivo: README.md
```

Hagamos ahora, como sugiere git status, un commit. Al hacerlo pasarmos estos cambios al history, donde estarán acompañados por una referencia y un comentario.

```
In [25]: git commit -m 'Commit inicial. Añadidos los primeros archivos'
```

```
[master (root-commit) 295d086] Commit inicial. Añadido el README.md
 2 files changed, 5 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 README.md
```

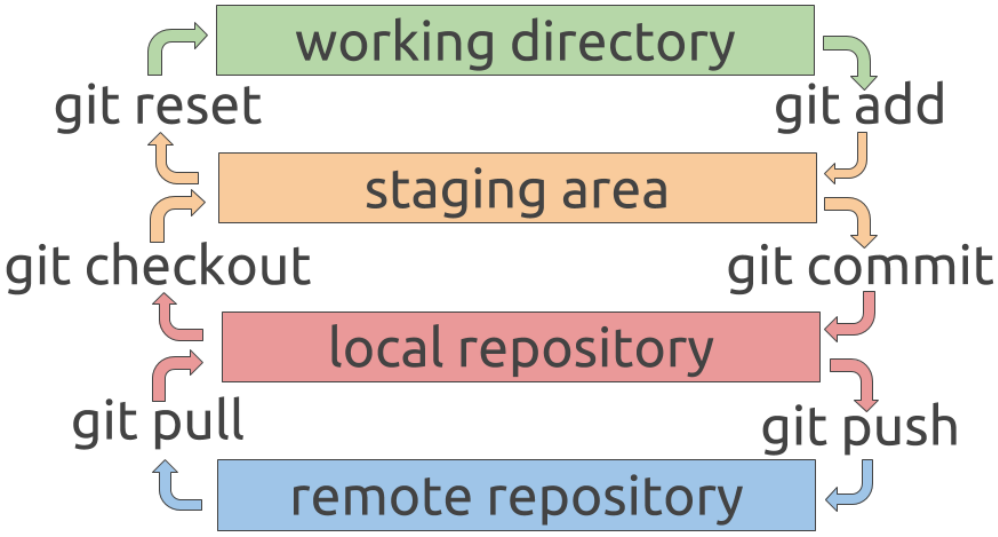
```
In [26]: git status
```

```
En la rama master
nothing to commit, working tree clean
```

Los cambios hechos en este sentido se pueden revertir sin peligro en el otro sentido. La tres áreas estan conectadas por carriles de dos sentidos; circulad sin miedo.

Permitidme añadir un área nueva. Hasta el momento habéis trabajado en vuestro ordenador, en local. Es recomendable guardar una copia en un repositorio remoto. Podéis mandar los cambios incluso a la nube pero vamos a hacerlo con github.

Para ello debemos crear antes el repositorio en nuestra cuenta de github. Una vez creado, podemos añadir la dirección del repositorio remoto a la lista de repositorios de git.




```
In [27]: git remote add origin https://github.com/juanqenk/gitstarted.git
```

Podemos listar los repositorios remotos, donde aparece el que hemos llamado origin:

```
In [28]: git remote -v

origin        https://github.com/juanqenk/gitstarted.git (fetch)
origin        https://github.com/juanqenk/gitstarted.git (push)
```

Como no hemos creado ninguna rama, estamos en la rama master. En ella hemos hecho unos cambios que ya se han guardado en nuestro repositorio local, pero no en nuestro repositorio remoto. Para hacer esto basta con invocar el siguiente comando:

```
git push origin master
```

Ya tenéis los primeros cambios en vuestro repositorio remoto.

Si queréis hacer o deshacer el camino que estamos siguiendo tened en cuenta el siguiente diagrama:

```
In [3]: git status

En la rama master
nothing to commit, working tree clean
```

Dentro del directorio de trabajo he creado nuestro programa para viajar en el tiempo.

```
In [8]: ls

README.md  timetravel.py
```

```
In [9]: python timetravel.py

Estimado viajero en el tiempo, gracias por confiar en TimeTraveler para deshacer sus errores de
Actualmente usted está en Tue Jun 20 19:51:35 2017
Abrochense los cinturones. Usted va a viajar al día 28 - 6 - 2009
Done!
Bienvenido al pasado, esperamos que pase un buen rato.
```

¡Funciona! Genial. Esto del pasado es una pasada. Creo que nos divertiremos aquí. Antes de ir a la fiesta dejadme que guarde los cambios.

```
In [10]: git add .

In [11]: git status

En la rama master
Cambios para hacer commit:
  (use ńgit reset HEAD <archivo>...ż para sacar del stage)

        borrado:       hola.py
        nuevo archivo: timetravel.py


In [12]: git commit -m 'Añadido un archivo que permite asistir a la fiesta de viajeros en el ti

[master 60787ef] Añadido un archivo que permite asistir a la fiesta de viajeros en el tiempo de
 2 files changed, 43 insertions(+), 2 deletions(-)
 delete mode 100644 hola.py
 create mode 100644 timetravel.py


In [13]: git status

En la rama master
nothing to commit, working tree clean
```

La fiesta no ha sido un éxito de público pero Stephen y yo lo hemos pasado un buen rato jugando al scrabble. En realidad, no vino nadie más. Espero que gente divertida siga este tutorial en el futuro y se sume a la fiesta. Adjunto documento gráfico (yo soy quién hace la foto).

Es hora de volver, se hace tarde y los genios necesitan descanso. También le vendrá bien un sueñecito a S.H.

Desgraciadamente se me olvidó programar el regreso al presente. ¡Que desafortunado inconveniente! El bueno de Stephen me ha dejado su ordenador de a bordo. No quiero estropear mi programa de viajes en el tiempo pero debo modificarlo. Lo mejor será crear una rama donde podré añadir esta característica, y si funciona, fusionarla con el programa principal.

Podemos crear la rama con:


```
In [14]: git branch rama
```

y pasar a la rama creada con:

```
In [15]: git checkout rama

Switched to branch 'rama'
```

```
In [16]: git branch -a

  master
* rama
  remotes/origin/master
```

El comando anterior nos da una lista con todas las ramas que tenemos. Se muestra con un asterisco la rama activa.

Antes que mi anfitrión se impaciente voy a crear un script simple para volver al presente.

```
In [17]: touch backtothefuture.py
         gedit backtothefuture.py
```

Es el momento de partir.

```
In [18]: python backtothefuture.py
```

```
Estimado viajero en el tiempo, gracias por confiar en TimeTraveler para volver al presente.
Esperamos que no haya evitado que sus padres se conozcan.
Abrochense los cinturones. Usted va a viajar al día Tue Jun 20 20:19:01 2017
Done!
Bienvenido al presente, le echabamos de menos.
```

Parece que todo funciona correctamente. Guardaré los cambios realizados en esta rama y posteriormente uniré esta con la rama principal.

```
In [19]: git status

En la rama rama
Archivos sin seguimiento:
  (use ńgit add <archivo>...ż para incluir en lo que se ha de confirmar)

        backtothefuture.py

no se ha agregado nada al commit pero existen archivos sin seguimiento (use ńgit addż para dar


In [20]: git add .
         git commit -m 'Añado como volver al presente en la rama'

[rama d42b543] Añado como volver al presente en la rama
 1 file changed, 25 insertions(+)
 create mode 100644 backtothefuture.py
```

Ahora puedo volver a la rama principal.

```
In [21]: git checkout master

Switched to branch 'master'


In [22]: git status

En la rama master
nothing to commit, working tree clean
```

Podemos comprobar que no existe la forma de volver al presente, y no solo porque no se puede ir al presente desde el presente. Ocurre que los cambios los hicimos en la otra rama.

```
In [23]: python backtothefuture.py

python: can't open file 'backtothefuture.py': [Errno 2] No such file or directory
```

Sin embargo desde la rama principal podemos fusionar las demás ramas.

```
In [25]: git merge rama

Updating 60787ef..d42b543
Fast-forward
 backtothefuture.py | 25 +++++++++++++++++++++++++
 1 file changed, 25 insertions(+)
 create mode 100644 backtothefuture.py
```

Y podemos viajar al presente desde el presente.

```
In [26]: python backtothefuture.py

Estimado viajero en el tiempo, gracias por confiar en TimeTraveler para volver al presente.
Esperamos que no haya evitado que sus padres se conozcan.
Abrochense los cinturones. Usted va a viajar al día Tue Jun 20 20:28:21 2017
Done!
Bienvenido al presente, le echabamos de menos.
```

```
In [27]: git status

En la rama master
nothing to commit, working tree clean
```

Si en algún momento queréis consultar todos los cambios que habéis hecho, solo tenéis que hacer:

```
git log
```

Y como hemos fusionado la rama con éxito, podemos borrarla sin problema.

```
In [29]: git branch -d rama

Eliminada la rama rama (era d42b543)
```

```
In [30]: git branch -a

* master
  remotes/origin/master
```

Gracias por llegar conmigo al final de este viaje. Os mando un saludo de vuestro yo del pasado.