Taller de Git 2016

Tutorial de ejemplo

El siguiente tutorial servirá de una pequeña guía para ayudar a entender los conceptos básicos del uso de `git` como *sistema de control de versiones* (**cvs**) a nivel local en nuestro ordenador.

Índice

Configuración inicial

Antes de nada configuramos nuestro nombre y email de usuario en la configuración de git.

```
$ git config --global user.name "Ismael Taboada"
$ git config --global user.email "ismaeljose.taboada@alumnos.uva.es"
```

Comandos iniciales

Lo primero que vamos a hacer, es crear nuestro repositorio. Para ello vamos a iniciarlo en el directorio que más nos apetezca.

Comenzamos creando un directorio:

```
$ mkdir repositorio
```

Accedemos a él y iniciamos nuestro repositorio:

```
$ cd repositorio
$ git init
Initialized empty Git repository in ~/repositorio/.git/
```

Vale, una vez creado vamos a empezar a crear nuestro proyecto. Seguro que muchos de vosotros tenéis vuestros lenguajes preferidos. Bien pues vamos a crear un `Hola Mundo` en cualquiera de ellos. Yo elegiré *Python*:

```
$ touch hello.py
$ vim hello.py
```

El contenido de este script será:

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

print 'Hello World'
```

Que al interpretarlo nos dará:

```
$ python hello.py
```

Bien no hay mucho misterio.

Preparando cambios

Pero este nuevo fichero aún no está en el repositorio como tal. Si echamos una mirada al estado de este:

```
$ git status
En la rama master

Commit inicial

Archivos sin seguimiento:
  (use «git add <archivo>...» para incluir lo que se ha de ejecutar)

    hello.py

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)
```

Vale, pues vamos a hacer que nuestro código entre dentro del repositorio.

```
$ git add .
```

Si revisamos ahora su estado:

```
$ git status
En la rama master

Commit inicial

Cambios para hacer commit:
  (use «git rm --cached <archivo>...« para eliminar stage)

    new file:   hello.py
```

Como vemos, ya está preparado.

Registrando cambios

Vale, ahora tenemos seguimiento sobre los ficheros anteriores. Eso significa que estamos preparados para el primer registro de cambios o `commit`.

```
$ git commit -m 'Primer commit'
```

Pero compliquemos nuestro programa. Añadamos un poco de funciones, vamos a crear un fichero aparte en *Python*:

impresion.py

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

def imprimir():
    print 'Hello World'
```

Y modifiquemos nuestro programa original:

hello.py

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

from impresion import *

imprimir()
```

Tras ejectuar nuestro código en nuestro directorio quedarían tres archivos:

```
$ ls
hello.py  impresion.py  impresion.pyc
```

Los cuales añadimos al repositorio con `git add`. Pero nos damos cuenta que tener un fichero `.pyc` no es adecuado, ya que este se genera cada vez que se ejecute y en sí, no es necesario. Así que decidimos borrarlo de nuestro repositorio, e intentar evitar que estos se tomen en cuenta al hacer cambios en él.

Para ello utilizamos:

```
$ git rm impresion.pyc
error: the following file has changes staged in the index:
    impresion.pyc
```

```
(use --cached to keep the file, or -f to force removal)
```

Esto nos da un error, porque el repositorio ya lo tiene en cache para ser registrado junto a los otros dos archivos, así que lo volvemos a intentar con:

```
$ git rm --cached impresion.pyc
rm 'impresion.pyc'
$ git status
En la rama master

Commit inicial

Cambios para hacer commit:
  (use «git rm --cached <archivo>...« para eliminar stage)

    new file:   hello.py
    new file:   impresion.py

Archivos sin seguimiento:
  (use «git add <archivo>...» para incluir lo que se ha de ejecutar)

    impresion.pyc
```

Aún así el respositorio lo tiene en cuenta por si hubiese cambios, y pide que se le haga seguimiento también. Entonces, veamos como podemos evitar esto.

Fácil, utilizamos el fichero `.gitignore`:

.gitignore

```
# Evitar python pyc
*.pyc
```

Esto nos servirá para evitar que en un futuro estos tipos de ficheros no aparezcan en el seguimiento.

```
$ git status
En la rama master

Commit inicial

Cambios para hacer commit:
  (use «git rm --cached <archivo>...« para eliminar stage)

    new file:   hello.py
    new file:   impresion.py

Archivos sin seguimiento:
  (use «git add <archivo>...» para incluir lo que se ha de ejecutar)

    .gitignore
```

Genial, ahora ya no molestará más. Eso si, no nos olvidemos de nuestro `.gitignore`.

Para terminar, registremos ambos los cambios con el mensaje `Lo complicado funciona`.

Hasta aquí todo bien. Reviesemos la evolución del proyecto con:

```
$ git log --graph --all
```

```
* commit a3a226522c7cd678ebdea9f68b405fbb980e72c8
| Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
| Date:   Wed Apr 27 10:08:49 2016 +0200
|
|     Lo complicado funciona
|
* commit db67a0ab83f1145dba4b68c7efa840555bd60bd5
  Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
  Date:   Wed Apr 27 10:06:57 2016 +0200

      Primer commit
(END)
```

Branchs

Pero una única rama no suele ser buena idea para desarrollar en un repositorio. Sobre todo cuando se trabaja en equipo. Así que, como si fuesemos Tarzanes ó Janes en la jungla vamos a balancearnos un poco por estas.

Preparamos una nueva rama sobre la que trabajar.

```
$ git checkout -b develop
$ git branch
* develop
  master
```

Y sobre esta, vamos a modificar nuestros ficheros haciendo la función algo más complicada.

impresion.py

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

def imprimir(texto):
    print texto
```

hello.py

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-
from impresion import *

imprimir('Hello World')
```

Así que probamos que funciona bien, y registramos los cambios:

```
$ git status
En la rama serpientes
Cambios no preparados para el commit:
  (use «git add <archivo>...» para actualizar lo que se ejecutará)
  (use «git checkout -- <archivo>...« para descartar cambios en le directorio de trabajo)

    modificado: hello.py
    modificado: impresion.py

no hay cambios agregados al commit (use «git add» o «git commit -a»)
$ git commit -a -m'Nueva rama'
[serpientes 8052832] Nueva rama
 2 files changed, 4 insertions(+), 6 deletions(-)
```

Fusión de ramas

Bien, llegamos a la última etapa, pero es que nos hemos dado cuenta, que también estaría bien modificarlo en `master`. Así que vamos a cambiarnos a esa rama y vamos a modificar nuestra funcion de `imprimir`:

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

Y modificamos:

impresion.py

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

def imprimir(texto='Hello World'):
    print texto
```

Y ahora registramos los cambios que hemos hecho.

```
$ git commit -a -m 'Cambiada la función de impresión'
[master 958d81f] Cambiada la función de impresión
 1 file changed, 2 insertions(+), 2 deletions(-)
```

Y llego la hora de fusionar ambas ramas.

Como ya estamos en la rama que queríamos fusionar no nos movemos, y hacemos `merge` desde `develop`.

```
$ git merge develop
Automezclado impresion.py
```

```
CONFLICTO(contenido): conflicto de fusión en impresion.py
Automatic merge failed; fix conflicts and then commit the result.
```

Oh no, un **CONFLICTO**, justo lo que queríamos evitar. Pero también nos avisa en que fichero se ha producido el conflicto, en este caso `impresion.py`.

Vamos a intentar arreglarlo. Pero para ello debemos ver que ha pasado dentro de ese fichero, lo abrimos y observamos que el contenido a pasado a ser el siguiente.

```
$ cat impresion.py
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

<<<<<<< HEAD
def imprimir(texto='Hello World'):
=======
def imprimir(texto):
>>>>>>> develop
    print texto
```

Bueno, es sólo una línea. Podemos observar que:

```
<<<<<<< HEAD
def imprimir(texto='Hello World'):
```

Es lo que teníamos en `master` y que:

```
def imprimir(texto):
>>>>>>> develop
```

Aquello que teníamos en la rama `develop`, así que es hora de elegir con que parte del código nos plantamos.

Vamos a quedarnos con la que teníamos en master para que nos funcione bien. Así que eliminamos las líneas de **CONFLICTO** para que nos quede así el fichero:

impresion.py

```
#! /usr/bin/env/python
# -*- coding: utf-8 -*-

def imprimir(texto='Hello World'):
    print texto
```

Y registramos que el **CONFILCTO** se ha resuelto:

```
$ git status
En la rama master
Tiene rutas sin fusionar.
  (solucione los conflictos y ejecute «git commit»)

Cambios para hacer commit:

    modificado: hello.py

Rutas no combinadas:
  (use «git add <archivo>...» para marcar resolución)

    modificado por ambos:impresion.py

$ git commit -a -m 'Resuelto conflicto de impresión'
[master 20b5aaf] Resuelto conflicto de impresión
$ git status
En la rama master
nothing to commit, working directory clean
$ git merge develop
Already up-to-date.
```

Finalmente nuestro código se ha fusionado. Es más podemos ver su evolución con:

```
$ git log --graph --all
```

```
*   commit 971ec52f631ef9ce5bf7b5fad10992eb97eddf7d
|\  Merge: b8ecb47 85454e9
| | Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
| | Date:   Wed Apr 27 10:19:39 2016 +0200
```

```
| |
| |     Resuelto conflicto de impresión
| |
| * commit 85454e93b461eea7fcbc887780967cc5c2a42bc8
| | Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
| | Date:   Wed Apr 27 10:13:00 2016 +0200
| |
| |     Nueva rama
| |
* | commit b8ecb475b73abca18093399a8940505c215b3751
|/  Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
|   Date:   Wed Apr 27 10:13:56 2016 +0200
|
|       Cambiada la función de impresión
|
* commit a3a226522c7cd678ebdea9f68b405fbb980e72c8
| Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
| Date:   Wed Apr 27 10:08:49 2016 +0200
|
|     Lo complicado funciona
|
* commit db67a0ab83f1145dba4b68c7efa840555bd60bd5
  Author: Ismael Taboada <ismaeljose.taboada@alumnos.uva.es>
  Date:   Wed Apr 27 10:06:57 2016 +0200

      Primer commit
(END)
```

Taller organizado por el Grupo Universitario de Informática. Mantente informado de más talleres y eventos en nuestra página web o en las rrss del grupo, Twitter(@gui_uva) y Facebook(Grupo Universitario de Informática).

Contacto: gui@uva.es

Ponentes:

- Ismael Taboada
- Jorge Sanz