

## Git Workflow

Instrucciones para entender y aplicar algunos de los comandos de consola necesarios para realizar una buena gestión del sistema de control de versiones local (Git) y remoto (GitHub). Todos estos comandos tienen su homónimo en *front-ends* gráficos de git (como por ejemplo el plugin Egit para Eclipse). Una vez comprendido el funcionamiento de estos comandos es más fácil manejar cualquier *front-end*.

### Instalación y configuración del entorno

1. Instalar git (<http://git-scm.com/download>)
2. Abrir un terminal y configurar las variables git globales.

```
$ git config --global user.name "Your Name Comes Here"
$ git config --global user.email you@yourdomain.example.com
```

3. Dar de alta un usuario GitHub (si es necesario)
4. Seguir todas las instrucciones ([Windows](#), [Linux](#) o [Mac](#)) para configurar correctamente el usuario GitHub en git.

```
$ git config --global github.user username
$ git config --global github.token 0123456789yourf0123456789token
```

### Clonar un repositorio remoto a un directorio local

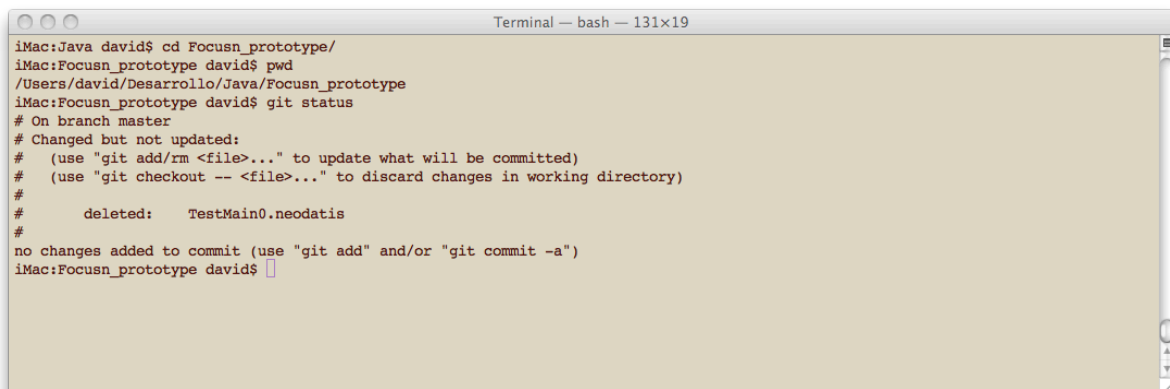
5. Situar el directorio actual al espacio de trabajo (ej: /Users/david/Desarrollo/Java)  
\$ cd <your\_workspace>
6. Clonar el repositorio remoto alojado en GitHub en el espacio de trabajo local.



```
Terminal — bash — 89x10
iMac:Java david$ git clone git@github.com:Flexible-User-Experience/Focusn_prototype.git
Cloning into Focusn_prototype...
remote: Counting objects: 835, done.
remote: Compressing objects: 100% (189/189), done.
remote: Total 835 (delta 629), reused 823 (delta 617)
Receiving objects: 100% (835/835), 11.32 MiB | 436 KiB/s, done.
Resolving deltas: 100% (629/629), done.
iMac:Java david$ pwd
/Users/david/Desarrollo/Java
iMac:Java david$
```

### Consultar estado actual del repositorio (local y remoto)

7. Situar el directorio actual dentro del repositorio local
8. Consultar el estado de git en este repositorio y la rama<sup>1</sup> local

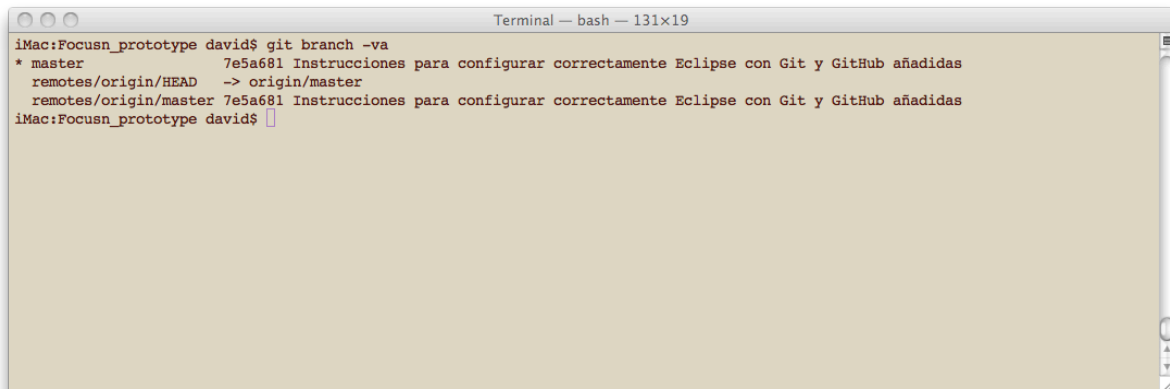


```
Terminal — bash — 131x19
iMac:Java david$ cd Focusn_prototype/
iMac:Focusn_prototype david$ pwd
/Users/david/Desarrollo/Java/Focusn_prototype
iMac:Focusn_prototype david$ git status
# On branch master
#
# Changed but not updated:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       deleted:    TestMain0.neodatis
#
no changes added to commit (use "git add" and/or "git commit -a")
iMac:Focusn_prototype david$
```

---

<sup>1</sup> en la última sección de este documento hay más información sobre las ramas

## 9. Consultar el estado de todas las ramas locales y remotas



```
Terminal — bash — 131x19
iMac:Focusn_prototype david$ git branch -va
* master          7e5a681 Instrucciones para configurar correctamente Eclipse con Git y GitHub añadidas
remotes/origin/HEAD -> origin/master
remotes/origin/master 7e5a681 Instrucciones para configurar correctamente Eclipse con Git y GitHub añadidas
iMac:Focusn_prototype david$
```

### Flujo local

10. Añadir, modificar y eliminar archivos del código hasta el punto deseado en el que se va a tomar una instantánea (*commit*) del estado actual del código.
11. Se recomienda que el código compile correctamente antes de tomar una 'foto' (*commit*). También se recomienda que los intervalos de tiempo entre *commits* sean breves, por ejemplo, al terminar de implementar un método de una clase o al terminar una funcionalidad concreta de la aplicación.
12. Añadir al *commit* los archivos añadidos o modificados con el comando:  

```
$ git add <files>
```
13. Eliminar del *commit* los archivos eliminados con el comando:  

```
$ git rm <files>
```
14. Realizar el *commit* local y añadir su comentario correspondiente. Se recomienda añadir siempre dicho comentario.  

```
$ git commit -m '<your_comment_about_last_changes>'
```

### Flujo remoto

15. Subir todos los *commits* de una rama local al repositorio GitHub remoto. Por defecto la rama local inicial es 'master' y la referencia al repositorio remoto es 'origin'

\$ git push <remote> <your\_local\_branch>

16. La acción del punto anterior nos garantiza que todo nuestro trabajo será guardado en el repositorio remoto (igual que una copia de seguridad)
17. Al realizar un *push* puede pasar que otro usuario haya actualizado el código antes que nosotros, entonces, git nos advertirá de que primero es necesario actualizar nuestra rama local con los cambios del otro usuario. Si las partes de código modificado pertenecen a líneas diferentes, la fusión entre repositorios se realizará automáticamente. Si existen colisiones git ofrecerá los mecanismos para decidir que hacer con los conflictos.
18. Para actualizar nuestra rama local de trabajo con la última versión del código almacenado en el repositorio remoto hay que hacer un *pull*. El comando *pull* encadena los comandos *fetch* y *merge* automáticamente. En caso de no encontrar conflictos entre el código remoto y el código local, *pull* nos actualizará nuestro código local a la última versión del código remoto. Se recomienda ejecutar en dos pasos por separado los comandos *fetch* y *merge*.

\$ git pull <remote>

### Trabajando con ramas

19. Se puede crear una nueva rama local de nuestro trabajo en cualquier momento. La rama nueva partirá del mismo punto que el código actual.

```
$ git branch <new_local_branch>
```

20. Continuamos con nuestro flujo de trabajo habitual descrito entre los puntos 10-14

21. Si hacemos *push* la nueva rama también se replicará en el repositorio remoto.

22. Podemos cambiar de rama local en cualquier momento con el comando:

```
$ git checkout <local_branch>
```

23. Después del punto 22 el código se transformará y recuperará el último estado de la rama a la que hemos saltado (*local\_branch*)

24. Después del punto 23 se puede fusionar en la rama actual (*local\_branch*) el código de la rama nueva (*new\_local\_branch*):

```
$ git merge <new_local_branch>
```

25. Esto también es muy útil para evitar colisiones o conflictos con el repositorio remoto ya que podemos abrir ramas diferentes entre colaboradores.

26. Dentro del ámbito local también es muy útil porque nos permite abrir una nueva rama para, por ejemplo, desarrollar un proceso experimental y, posteriormente, fusionarlo con la rama principal una vez que se haya verificado y terminado de implementar dicho proceso.

27. Las fusiones de ramas locales también se replican en el repositorio remoto después de un *push*.