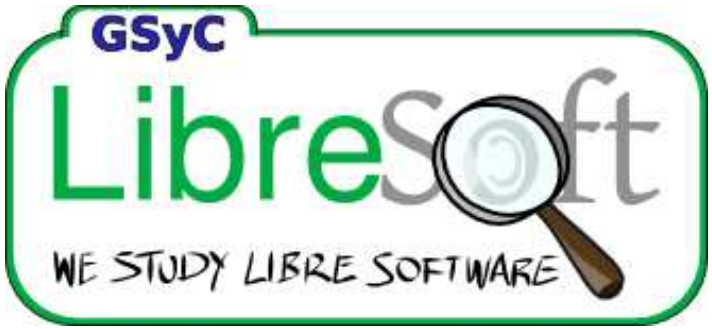
Git

Trabajando con Git

Carlos García Campos

carlosgc@gsyc.escet.urjc.es
GSyC/LibreSoft



Algunas citas

“For the first 10 years of kernel maintenance, we literally used tarballs and patches, which is a much superior source code management system than CVS is[...]” – Linus Torvalds

“When I say I hate CVS with a passion, I have to also say that if there any SVN users in the audience, you might want to leave. Because my hatred of CVS has meant that I see Subversion as being the most pointless project ever started, because the whole slogan for the Subversion for a while was ’CVS done right’ or something like that. And if you start with that kind of slogan, there is nowhere you can go. It’s like, there is no way to do CVS right.” – Linus Torvalds

Antes de empezar

- Le decimos a Git quienes somos:

```
[user]
        name = Tu nombre completo
        email = tu@correo.ext
```

  - De forma global: /.gitconfig
  - De forma local: repo/.git/config
- Fichero de configuración extensible
- También manipulable con *git config*

Operaciones básicas

- Crear un repositorio

```
$ mkdir project
$ cd project
$ git init
```

- Commit

```
$ git add file
$ git commit
```

- Diferencias
  - *git diff*: diferencias entre el árbol de trabajo y el índice
  - *git diff –cached*: diferencias entre HEAD y el índice
- Atajo especial para commit

```
$ git commit -a
```

- Estado

```
$ git status
```

NOTA: el índice es un snapshot del árbol de trabajo que representa el contenido que se verá afectado por *git commit*

Arreglando errores

- Déjame todo como estaba . . . (deshaciendo la historia)

```
$ git reset --hard <commit>
```

- Déjame todo como estaba . . .

```
$ git revert <commit>
```

- Recupérame una versión antigua de un fichero

```
$ git checkout <commit>
```

Explorando la historia

- *git log*

```
commit ce6ddeac2200225b173c52e2509ffcc01b2b4a1d
Author: Carlos Garcia Campos <carlosgc@gnome.org>
Date:   Mon Nov 19 16:14:01 2007 +0100

    Initial commit
```

  - Commit: identificador del commit (sha1)
  - Author: autor del commit (no confundir con committer)
  - Date: fecha del commit
  - Mensaje del commit: (una única línea con una breve descripción del cambio y, opcionalmente, una línea en blanco y una o mas líneas con una descripción mas detallada)

No mas fichero ChangeLog -> Isra feliz :-)

Personalizando la salida de log

- Con estadística de los cambios: *–stat*
- Con el diff de los cambios (parche): *-p*
- Con información de las rutas alteradas: *–name-status*
- Con información completa de autor y committer: *–pretty=fuller*
- Limitando el número de commits a mostrar: *-n<n_commits>*
- Atajo útil: *git show*

Ramas

- Las ramas son muy “baratas” (proceso rápido y limpio)
- Crear una rama

```
$ git branch <branch> [<start-point>]
```

- Listar ramas

```
$ git branch
```

- Cambiar de rama

```
$ git checkout
```

- Crear una rama y cambiar a ella

```
$ git checkout -b <branch> [<start-point>]
```

- Mezclar ramas

```
$ git merge <branch>
```

- Actualizar una rama (desde otra)

```
$ git rebase <branch>
```

Tirando del carro

- Clonar un repositorio (no confundir con checkout)

```
$ git clone <uri>
```

- Actualizar cambios de un repositorio en nuestro clon

```
$ git fetch <uri>
$ git merge origin/master
```

- Atajo útil (uso normal)

```
$ git pull
```

- Actualizar un repositorio remoto

```
$ git push
```

Flujo de trabajo (Resumen)

```
$ git clone <uri>
$ cd project
$ git checkout -b new-feature
algunos cambios
$ git commit -a
mas cambios
$ git commit -a
últimos cambios que dejan mi feature lista
$ git commit -a
$ git checkout master
$ git pull
$ git checkout new-feature
$ git rebase master
$ git checkout master
$ git merge new-feature
$ git push
```

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- Rendimiento!!!

Grandes ventajas en el día a día

- Uso sin red
- Trabajar en varias features distintas a la vez
- División del trabajo en micro-commits
- Facilidad de compartir el trabajo
- Author != Committer
- Historia mas rica
- Interactividad con otros sistemas (git-svn, git-cvs)
- **Rendimiento!!!**

Algunos trucos

- Cherry-picking

```
$ git cherry-pick <commit>
```

- Quita, que tengo que arreglar esto primero

```
$ git stash "work in progress for foo thing"
cambios, commit, push y lo que sea
$ git stash apply
```

- Mezcla, pero no me toques la historia que tengo commits embarazosos

```
$ git merge --squash <branch>
$ git commit -a
$ git push
```

Algunos trucos

- Cherry-picking

```
$ git cherry-pick <commit>
```

- Quita, que tengo que arreglar esto primero

```
$ git stash "work in progress for foo thing"
cambios, commit, push y lo que sea
$ git stash apply
```

- Mezcla, pero no me toques la historia que tengo commits embarazosos

```
$ git merge --squash <branch>
$ git commit -a
$ git push
```

Algunos trucos

- Cherry-picking

```
$ git cherry-pick <commit>
```

- Quita, que tengo que arreglar esto primero

```
$ git stash "work in progress for foo thing"
cambios, commit, push y lo que sea
$ git stash apply
```

- Mezcla, pero no me toques la historia que tengo commits embarazosos

```
$ git merge --squash <branch>
$ git commit -a
$ git push
```

Trabajando con otros

- Enviando parches

```
$ git format-patch <commit-from>..<commit-to>
```

- Aplicando parches recibidos

```
$ cat patch1 patch2 .. patchn > patches.mbox
$ git am patches.mbox
```

Uso Git en la intimidad y nadie lo sabe

- Importar un svn

```
$ git-svn clone <svn-uri> <modulename>
```

- Hacer commit al svn

```
$ git-svn dcommit # Desde master
```

- Actualizar con los cambios del svn

```
$ git-svn rebase # Desde master
```

Flujo de trabajo (Resumen)

```
$ git clone <svn-uri> <modulename>
$ git checkout -b super-feature
commits y mas commits
$ git checkout master
$ git-svn rebase
si hay cambios rebaso también la rama super-feature desde master
$ git merge --squash super-feature
actualizo el ChangeLog
$ git commit -a
$ git-svn dcommit
```