

Git

Conceptos

Repositorio, repository

Conjunto de ficheros, ramas, referencias...

Repositorio local

Aquél que utiliza normalmente un usuario. En algunos comandos se denota por `''`

Repositorio remoto

Repositorio no local, p.e. aquél del que se copió el local

origin

Referencia al repositorio remoto origen del local

Revisión, parche, commit

Conjunto de ficheros en un estado dado que se guarda en el repositorio local
git guarda fotos (snapshots) de ficheros; subversion guarda deltas.

Commit ID

Identificador hexadecimal único

Rama

Lista de commits secuenciales

Rama local

Rama en repositorio local

Rama remota

Rama en repositorio remoto que se observa desde el repositorio local

Rama master, master branch

Rama en que se guarda la versión más estable

Rama de tópico, topic branch

Rama temporal para trabajar un aspecto concreto

Árbol de trabajo, working tree

Checkout de una rama local

Rama actual, current branch

Rama local que está en el árbol de trabajo

Cabeza de rama, branch head, cabeza, head

Último commit

HEAD

Cabeza de la rama actual

HEAD^ HEAD~1: penúltimo cambio guardado, parent head

HEAD~2 HEAD^^, HEAD~3 HEAD^^^, ...: antepenúltimo, ...

Repositorio

Operación		Comando
Crear	Desde cero	mkdir prj
		cd prj
		git init [--shared] (1)
	Desde ficheros	cd prj
		git init
		git add .
	desde repo remoto	git clone <repo> [<new_dir>]

1. --shared: si vamos a permitir que otros usuarios escriban en el repositorio

<repo>:

- <http://host/ruta/absoluta>
- <ssh://host/ruta/absoluta>

- `file:///ruta/absoluta`, `/ruta/absoluta`

Repositorio: configuración

.git/config

```
[user]
  name = Juan Español
  email = juan@dominio.es
[color]
  branch = auto
  diff = auto
  status = auto
```

.gitignore

Ficheros no sometidos a control de versiones

```
*.gif
*.config
```

Ficheros

Operación	Comando
Meter en git	git add <file>...
Guardar	git commit [-m '...'] <file>...
diff con rama actual	git diff [<commit>] <file> (1)
Ver lista cambios	git log <file>
Autor de los cambios	git blame [<commit>] <file>
Descartar modificaciones	git checkout [<commit>] <file>... (2)

1. <commit>: por defecto HEAD

2. <commit>: sobrescribir con la versión de un commit dado; por defecto HEAD

Árbol de trabajo

Operación	Comando
Descartar cambios	git reset --hard
diff con otra rama	git diff <rama>

Revisiones

Operación		Comando
Crear	Todo pendiente	git commit -a [-m '...']
	Ficheros	git commit <file>...
Listar		git log [<file>...]
Mostrar		git show <commit>
	Respecto a w.t.	git diff <commit>
Corregir (1)		git commit --amend [<file>...]
Revertir (2)		git revert <commit>

-m: comentario

1. No hacerlo si se ha propagado a ramas remotas
2. Puede propagarse

Etiquetas

Una etiqueta es un nombre simbólico asignado a un commit, un alias.

Utilidad:

- identificar hitos de desarrollo
- indentificar versiones entregadas a cliente

Operación	Comando
Crear	git tag <tag> [<commit>]
Listar	git tag -l
Eliminar	git tag -d <tag>
Renombrar	git tag <old_tag> <new_tag>; git tag -d <old_tag>
Commit de tag	git rev-parse <tag>

Ramas

Un diseño típico de ramas de un repositorio puede ser

- rama master, con el código más estable
- ramas de tópico o feature, para desarrollar funcionalidades
- ramas de corrección de bugs

Operación		Comando
Creación	desde cero	git checkout -b <new_branch>
	desde rama actual	git branch <new_branch> [<commit>] (1)
	desde otra rama	git branch <new_branch> <branch> (2)
Listar		git branch [-r] [-a] (3)
Establecer actual		git checkout <branch>
Renombrar		git -m [<branch>] <new_branch> (4)
Eliminar		git branch -d <branch> (5)

1. <commit>: HEAD de la nueva rama
2. <branch>: local o remota
3. por defecto: ramas locales
 - -r: ramas remotas
 - -a: todas
4. <branch>: por defecto la actual
5. la rama actual debe ser otra

Conceptos

Veamos un elemento que no aparecía en sistemas más tradicionales

Index, index file

Conjunto de cambios para el próximo commit

¿Cómo afecta este elemento a las operaciones vistas?

- ficheros
- index
- revisiones
- stash

Ficheros

Operación		Comando
Añadir a git		git add <file>...
	Recursivamente	git add <dir>
Renombrar		git mv <file> <new_file> (1)
Eliminar de w.t. e index		git rm <file> (1)

1. la operación queda en el index

Index

Operación	Comando
Añadir fichero	git add <file>...
Listar	git status
Eliminar fichero	git reset HEAD <file>...
diff index - commit	git diff --cached [<commit>] [<file>...]
diff w.t. - index	git diff [<file>...]
Guardar	git commit

<commit>: por defecto HEAD

Revisión

Operación		Comando
Deshacer (1)	descarta index	git reset [--mixed] [<commit>]
	deja cambios en index	git reset --soft [<commit>]
	reset ambos a nuevo HEAD (2)	git reset --hard [<commit>]

(1) cambiamos el commit referenciado por HEAD <commit>: por defecto HEAD

Warning

2. ¡los commit y modificaciones posteriores al nuevo HEAD se pierden!

Stash

git tiene un array de portapapeles al que mover modificaciones en w.t. e index

- gestionar interrupciones

Operaciones		Comando
Mover cambios en w.t. e index		git stash [save [<comment>]]
Listar		git stash list
Mostrar		git stash show [<stash>]
Aplicar a rama actual		git stash apply [--index] [<stash>] (1)
" y descartar		git stash pop [<stash>]
Descartar	todos	git stash clear
	uno	git stash drop [<stash>]

<stash>: índice en la lista. Por defecto 0

1. --index: restaurar también el index

Merge

Vamos a juntar el trabajo hecho en distintas ramas.

Prerrequisitos

- index y w.t. deben estar limpios (1)
 - git diff --cached: Ø
 - git status: no hay ficheros modificados
- activar la rama destino como rama actual
- git merge <branch>
- también, git pull . <branch>

1. estrictamente, de los ficheros distintos a HEAD que lleguen

merge: resolución de problemas

Árbol sucio

Síntoma	"error: Entry '_<file>' not uptodate. Cannot merge."
Causa	Árbol de trabajo sucio
Solución	Guardar cambios o descartarlos, repetir merge

Conflicto de contenido

Síntoma	"CONFLICT (content): Merge conflict in <file>"
Causa	Un fichero tiene segmentos incompatibles en cada rama
Solución	Editar fichero y hacer commit

```
>>>>>>
a
-----
b
<<<<<<<
```

Warning

Probar el programa, la ausencia de conflictos no garantiza corrección

Trabajar con repositorios remotos

Operación		Comando
Actualiza repositorio remoto		git push [<remote>] [<branch>]
Añadir ref. a repositorio remoto	todas las ramas	git remote add <remote> <repo> (1)
	algunas ramas	git remote add <remote> <repo> -t <branch>
Actualizar copia local de rep. remoto		git fetch [<remote>]
Mezclar rama remota con actual		git merge <remote>/<branch> (2)
Actualizar y mezclar "		git pull [<remote>] <branch>

<remote>: por defecto origin

1. <remote>: nombre con el que se va a referenciar el repositorio remoto
2. P.e., alpha/master, level1/drivers

push: problemas

Rama local no actualizada	
Síntoma	"! [rejected] <branch> -> <branch> (non-fast forward)"
Causa	La rama remote no es descendiente directo de la actual
Solución	pull; push

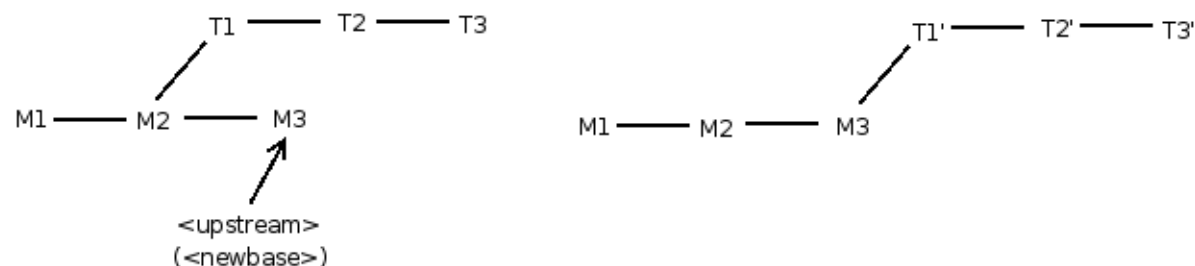
Reescribir la historia

Se pueden "trasplantar ramas" por el árbol de commits.

```
git rebase [--onto <newbase>] <upstream> [<branch>]
```

Git va a intentar aplicar en secuencia, a partir de <newbase>, los commits <upstream>..HEAD (cambios en la rama actual que no están en la rama de <upstream>)

- <newbase>: punto al que trasplantar la rama; por defecto <upstream>
- <branch>: rama en la que hacer el rebase mediante un checkout inicial; por defecto la actual



Pueden producirse conflictos de merge

- resolver y dejar la solución en el index; `git rebase --continue`
- o saltarse el commit problemático: `git rebase --skip`
- o abortar la operación: `git rebase --abort`

Utilidad

- simplificar el árbol
- actualizar frecuentemente una rama de tópico con el origin para que no diverja mucho y los conflictos se revelen y resuelvan pronto

```
git fetch origin master  
git rebase origin/master
```

rebase interactivo

Como preparación para propagar cambios a otras ramas, puede ser conveniente editarlos interactivamente

```
git rebase -i <upstream>
```

Se mueven los commits <upstream>..HEAD a una agenda temporal que podemos manipular

Operación	
Reordenar commits	reordenar las líneas
Unir con anterior	1ra palabra: squash
Descartar commit	Borrar su línea
Aplicar hasta commit	1ra palabra: edit
... continuar	git rebase --continue
Tratar conflicto	Dejar solución en index; git rebase --continue
	o bien, git rebase --skip
Abortar	git rebase --abort

Utilidad

- consolidar una rama de bug en un solo parche
- diseñar commits significativo en una rama de tópico

Propagar cambios selectivamente

```
git fetch [<remote>]  
git log ..<remote>/<branch>  
git cherry-pick <commit>
```

Es importante tratar de que los parches sean

- funcionalmente significativos
- independientes.

Flujo de trabajo

git permite una enorme variedad de estilos de trabajo: centralizado, distribuido, tradicional, ágil...

Un posible flujo de trabajo podría ser:

- pull para actualizar master local
- checkout rama de tópico
- probar el desarrollo y guardar commits con frecuencia
- rebase frecuente de origin/master, probar
- rebase interactivo para ordenar el trabajo, probar
- merge en rama maestra, probar
- propagar hacia arriba: git push origin master

Si se está corrigiendo un bug, el workflow puede ser muy parecido

- crear rama "bug-<n>"

- seguir el mismo workflow
- preparar un único commit

PROBAR el código es importante porque

- los conflictos pueden resolverse mal
- aunque no haya conflictos la mezcla puede no tener sentido como programa de ordenador

Otros temas

- parches
- .git/hooks
 - pre-commit
 - commit-msg
 - pre-rebase
 - post-commit
 - post-receive

Herramientas gráficas

- Tcl/Tk
 - gitk

- git-gui
- GTK
 - giggle
 - gitg
- Mac OS X
 - gitX
- Windows
 - tortoiseGit

Git como servicio

<http://www.github.com>

Bibliografía

- libros
 - <http://progit.org/book> (Scott Chacon)
 - <http://book.git-scm.com> Git community book
- chuletas
 - <http://cheat.errtheblog.com/s/git>
 - <http://github.com/guides/git-cheat-sheet>