Operaciones avanzadas y principios de Git

Esta entrada del blog fue organizada con la asistencia de ChatGPT-4o. El Keynote está disponible en https://github.com/lzwjava/Keynotes.

Deshacer

```
$ git commit --amend
$ git add *
$ git status
$ git reset HEAD CONTRIBUTING.md
Cambios no preparados después del reset:
M CONTRIBUTING.md
$ git status
$ git checkout -- CONTRIBUTING.md
```

- Rehacer el commit
- Deshacer los archivos en staging
- Deshacer los cambios en los archivos

Comandos

- `git revert`, revertir, hacer un nuevo commit que anula un commit anterior específico
- `git cherry-pick`, seleccionar commits específicos de múltiples commits
- `git rebase`, reorganizar la base de los commits

filter-branch

- Eliminar un archivo de cada commit
- Quiero eliminarlo, pero aún está en el historial
- ¿Cómo hacer que todos los commits eliminen un archivo?

```
$ git filter-branch --tree-filter 'rm -f passwords.txt' HEAD
Reescritura 6b9b3cf04e7c5686a9cb838c3f36a8cb6a0fc2bd (21/21)
La referencia 'refs/heads/master' fue reescrita
```

Búsqueda

```
$ git grep -n gmtime_r
compat/gmtime.c:3:#undef gmtime_r
compat/gmtime.c:8:return git_gmtime_r(timep, &result);
compat/gmtime.c:11:struct tm *git_gmtime_r(const time_t *timep, struct tm *result)
compat/gmtime.c:16:ret = gmtime_r(timep, result);
compat/mingw.c:606:struct tm *gmtime_r(const time_t *timep, struct tm *result)
compat/mingw.h:162:struct tm *gmtime_r(const time_t *timep, struct tm *result);
date.c:429:if (gmtime_r(&now, &now_tm))
date.c:492:if (gmtime_r(&time, tm)) {
git-compat-util.h:721:struct tm *git_gmtime_r(const time_t *, struct tm *);
git-compat-util.h:723:#define gmtime_r git_gmtime_r
$ git log -SZLIB_BUF_MAX --oneline
e01503b zlib: permitir alimentar más de 4GB de una vez
ef49a7a zlib: zlib solo puede procesar 4GB a la vez
```

.Directorio de Git

- `config` del proyecto
- `info` .gitignore
- `objects` todo el contenido de la base de datos de Git
- `refs` punteros de las ramas
- `HEAD` puntero de la rama actual
- `index` información del área de staging

Objetos de Git

```
$ git init test
Inicializado un repositorio Git vacío en /tmp/test/.git/
$ cd test
$ find .git/objects
.git/objects
.git/objects/info
.git/objects/pack
$ find .git/objects -type f
```

```
$ echo 'test content' | git hash-object -w --stdin
d670460b4b4aece5915caf5c68d12f560a9fe3e4
$ find .git/objects -type f
.git/objects/d6/70460b4b4aece5915caf5c68d12f560a9fe3e4
```

- `hash-object`, el comando para guardar datos en el directorio .git
- `-w`, escribe el objeto, de lo contrario solo devuelve la clave
- `--stdin`, lee desde la entrada estándar
- d670..., checksum de 40 caracteres
- `cat-file`, la navaja suiza para ver objetos en Git

```
$ git cat-file -p d670460b4b4aece5915caf5c68d12f560a9fe3e4
contenido de prueba

$ echo 'versión 1' > test.txt
$ git hash-object -w test.txt
83baae61804e65cc73a7201a7252750c76066a30
$ echo 'versión 2' > test.txt
$ git hash-object -w test.txt
1f7a7a472abf3dd9643fd615f6da379c4acb3e3a
$ find .git/objects -type f
.git/objects/1f/7a7a472abf3dd9643fd615f6da379c4acb3e3a
.git/objects/83/baae61804e65cc73a7201a7252750c76066a30
.git/objects/d6/70460b4b4aece5915caf5c68d12f560a9fe3e4
$ git cat-file -p 83baae61804e65cc73a7201a7252750c76066a30 > test.txt
$ cat test.txt
versión 1
$ git cat-file -p 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a > test.txt
$ cat test.txt
versión 2
$ git cat-file -t 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a
blob
$ git cat-file -p master^{tree}
100644 blob a906cb2a4a904a152e80877d4088654daad0c859 README
100644 blob 8f94139338f9404f26296befa88755fc2598c289 Rakefile
040000 tree 99f1a6d12cb4b6f19c8655fca46c3ecf317074e0 lib
$ git cat-file -p 99f1a6d12cb4b6f19c8655fca46c3ecf317074e0
```

```
100644 blob 47c6340d6459e05787f644c2447d2595f5d3a54b simplegit.rb
Objetos Tree

$ git update-index --add --cacheinfo 100644 \
83baae61804e65cc73a7201a7252750c76066a30 test.txt
update-index
```

- `update-index`, el comando para crear el árbol
- `--add`, crea el índice
- `--cacheinfo`, lee desde la base de datos de Git
- `100644`, el modo del archivo, para archivos normales; 100755 para archivos ejecutables; 120000 para enlaces simbólicos
- 83baae, es el contenido anterior, versión 1
- `\`, divide un comando en dos líneas

write-tree

```
$ git write-tree
d8329fc1cc938780ffdd9f94e0d364e0ea74f579
$ git cat-file -p d8329fc1cc938780ffdd9f94e0d364e0ea74f579
100644 blob 83baae61804e65cc73a7201a7252750c76066a30 test.txt
$ git cat-file -t d8329fc1cc938780ffdd9f94e0d364e0ea74f579
tree
```

- `write-tree`, escribe el contenido del área de preparación (staging area) en un Tree Object.

read-tree

```
$ echo 'nuevo archivo' > new.txt
$ git update-index test.txt
$ git update-index --add new.txt
$ git write-tree
0155eb4229851634a0f03eb265b69f5a2d56f341
$ git cat-file -p 0155eb4229851634a0f03eb265b69f5a2d56f341
100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt
100644 blob 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a test.txt
$ git read-tree --prefix=bak d8329fc1cc938780ffdd9f94e0d364e0ea74f579
```

```
$ git write-tree
3c4e9cd789d88d8d89c1073707c3585e41b0e614
$ git cat-file -p 3c4e9cd789d88d8d89c1073707c3585e41b0e614
040000 tree d8329fc1cc938780ffdd9f94e0d364e0ea74f579 bak
100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt
100644 blob 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a test.txt
```

Objetos de Commit

```
$ echo 'primer commit' | git commit-tree d8329f
d5ffe1aa4b7b089eee03dccea5e0439ad6d72037
$ git cat-file -p d5ffe1aa4b7b089eee03dccea5e0439ad6d72037
tree d8329fc1cc938780ffdd9f94
```

e0d364e0ea74f579 author lzwjava lzwjava@gmail.com 1462090215 +0800 committer lzwjava lzwjava@gmail.com 1462090215 +0800 primer commit

```
- El valor hash será diferente porque tanto la fecha de creación como el autor son distintos.
```

$ echo ‘segundo commit’ | git commit-tree 0155eb -p d5ffe1aa4 e946d6367f07de45cac242dca7cd002f5eaa72
$ echo ‘tercer commit’ | git commit-tree 3c4e9c -p e946d6 09490bf051c34b3693dfd5c7fb63dfe27b295904
$ git log –stat 09490 commit 09490bf051c34b3693dfd5c7fb63dfe27b295904 Author: lzwjava lzwjava@gmail.com Date: Sun May 1 16:38:55 2016 +0800 tercer commit bak/test.txt | 1 + 1 archivo cambiado, 1 inserción(+) commit e946d6367f07de45cac242dca7cd002f5eaa72b1 Author: lzwjava lzwjava@gmail.com Date: Sun May 1 16:37:01 2016 +0800 segundo commit new.txt | 1 + test.txt | 2 +- 2 archivos cambiados, 2 inserciones(+), 1 eliminación(-) commit d5ffe1aa4b7b089eee03dccea5e0439ad6d72037 Author: lzwjava lzwjava@gmail.com Date: Sun May 1 16:10:15 2016 +0800 primer commit test.txt | 1 + 1 archivo cambiado, 1 inserción(+)

```
- `commit-tree`, el primer parámetro apunta a un blob o a un tree
- `-p` apunta al nodo padre
```

tercer commit segundo commit primer commit árbol árbol árbol “versión 2” “nuevo archivo” “versión 1” 09490b e946d6 d5ffe1 d8329f 0155eb 3c4e9c 83baae fa49b0 1f7a7a test.txt new.txt test.txt new.txt test.txt bak

```
**El principio de las ramas**
```

$ find .git/refs
.git/refs .git/refs/heads .git/refs/tags $ find .git/refs -type f
$ echo “09490bf051c34b3693dfd5c7fb63dfe27b295904” > .git/refs/heads/master $ git log –pretty=oneline master 09490bf051c34b3693dfd5c7fb63dfe27b295904 tercer commit e946d6367f07de45cac242dca7cd002f5eaa72b1 segundo commit d5ffe1aa4b7b089eee03dccea5e0439ad6d primer commit $ git update-ref refs/heads/master 09490bf051c34b3693dfd5c7fb63dfe27b295904
$ git update-ref refs/heads/test e946d6367f07de45cac242dca7cd002f5eaa72b1 $ git branch
* master test

- Guardar una referencia de puntero
- `update-ref`, utilizado para cambiar o agregar referencias

tercer commit segundo commit primer commit árbol árbol árbol “versión 2” “nuevo archivo” “versión 1” 09490b e946d6 d5ffe1 d8329f 0155eb 3c4e9c 83baae fa49b0 1f7a7a test.txt new.txt test.txt new.txt test.txt bak refs/heads/master refs/heads/test

$ cat .git/HEAD ref: refs/heads/master $ git symbolic-ref HEAD refs/heads/master $ git symbolic-ref HEAD refs/heads/test $ cat .git/HEAD ref: refs/heads/test

symbolic-ref

Principio de las etiquetas (tags)

$ git update-ref refs/tags/v1.0 e946d6367f07de45cac242dca7cd002f5eaa72b1 $ git tag v1.0
$ git tag -a v1.1 e946d6367f07de45cac242dca7cd002f5eaa72b1 -m ‘etiqueta de prueba’ $ cat .git/refs/tags/v1.1 2766532f03289bc5e158629a8b3faffa5f80b8b6 $ git cat-file -p 276653 object e946d6367f07de45cac242dca7cd002f5eaa72b1 type commit tag v1.1 tagger lzwjava lzwjava@gmail.com 1462103203 +0800 etiqueta de prueba

remotes

$ git remote add origin git@github.com:schacon/simplegit-progit.git $ git push origin master Contando objetos: 11, listo. Comprimiendo objetos: 100% (5/5), listo. Escribiendo objetos: 100% (7/7), 716 bytes, listo. Total 7 (delta 2), reusados 4 (delta 1) A git@github.com:schacon/simplegit-progit.git a11bef0..ca82a6d master -> master $ cat .git/refs/remotes/origin/master ca82a6dff817ec66f44342007202690a93763949

- Se colocan en el directorio `refs/remotes`
- La diferencia entre las referencias de Remotes y las ramas es que las primeras son de solo lectura
- Puedes usar `git checkout`, pero HEAD no cambiará, por lo que no puedes modificar las referencias de

$ curl https://raw.githubusercontent.com/mojombo/grit/master/lib/grit/repo.rb > repo.rb $ git checkout master $ git add repo.rb $ git commit -m 'añadido repo.rb' [master 484a592] añadido repo.rb 3 archivos modificados, 709 inserciones(+), 2 eliminaciones(-) delete mode 100644 bak/test.txt create mode 100644 repo.rb rewrite test.txt (100%) $ git cat-file -p master^{tree} 100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt 100644 blob 033b4468fa6b2a9547a70d88d1bbe8bf3f9ed0d5 repo.rb 100644 blob e3f094f522629ae358806b17daf78246c27c007b test.txt $ git cat-file -s 033b4468fa6b2a9547a70d88d1bbe8 22044 $ echo '# testing' » repo.rb $ git commit -am 'modificado repo un poco' [master 2431da6] modificado repo.rb un poco 1 archivo modificado, 1 inserción(+) $ git cat-file -p master^{tree} 100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt 100644 blob b042a60ef7dff760008df33cee372b945b6e884e repo.rb 100644 blob e3f094f522629ae358806b17daf78246c27c007b test.txt $ git cat-file -s b042a60ef7dff760008df33cee372b94 22054 $ git gc Contando objetos: 18, listo. Compresión delta usando hasta 8 hilos. Comprimiendo objetos: 100% (14/14), listo. Escribiendo objetos: 100% (18/18), listo. Total 18 (delta 3), reutilizados 0 (delta 0) $ find .git/objects -type f $ git verify-pack -v .git/objects/pack/pack-978e03944f5c581011e6998cd0e9e30000905586.idx

Refspec

$ git remote add origin https://github.com/schacon/simplegit-progit [remote "origin"] url = https://github.com/schacon/simplegit-progit fetch = +refs/heads/*:refs/remotes/origin/* $ git log origin/master $ git log remotes/origin/master $ git log refs/remotes/origin/master fetch = +refs/heads/master:refs/remotes/origin/master [remote "origin"] url = https://github.com/schacon/simplegit-progit fetch = +refs/heads/master:refs/remotes/origin/master fetch = +refs/heads/qa/*:refs/remotes/origin/qa/* push = refs/heads/master:refs/heads

/qa/master

- `<src>:<dst>`
- `refs/remotes` ubicación local

https://github.com/schacon/simplegit-progit

https://github.com/schacon/simplegit-progit

Referencias

- Pro Git