Pro Git, el libro oficial de Git

2.3. Viendo el histórico de confirmaciones

Después de haber hecho varias confirmaciones, o si has clonado un repositorio que ya tenía un histórico de confirmaciones, probablemente quieras mirar atrás para ver qué modificaciones se han llevado a cabo. La herramienta más básica y potente para hacer esto es el comando `git log`.

Estos ejemplos usan un proyecto muy sencillo llamado simplegit que suelo usar para hacer demostraciones. Para clonar el proyecto, ejecuta:


```
$ git clone git://github.com/schacon/simplegit-progit.git
```

Cuando ejecutes `git log` sobre este proyecto, deberías ver una salida similar a esta:

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

Por defecto, si no pasas ningún argumento, `git log` lista las confirmaciones hechas sobre ese repositorio en orden cronológico inverso. Es decir, las confirmaciones más recientes se muestran al principio. Como puedes ver, este comando lista cada confirmación con su suma de comprobación SHA-1, el nombre y dirección de correo del autor, la fecha y el mensaje de confirmación.

El comando `git log` proporciona gran cantidad de opciones para mostrarte exactamente lo que buscas. Aquí veremos algunas de las más usadas.

Una de las opciones más útiles es `-p`, que muestra las diferencias introducidas en cada confirmación. También puedes usar la opción `-2`, que hace que se muestren únicamente las dos últimas entradas del histórico:

```
$ git log -p -2
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

diff --git a/Rakefile b/Rakefile
index a874b73..8f94139 100644
--- a/Rakefile
+++ b/Rakefile
@@ -5,7 +5,7 @@ require 'rake/gempackagetask'
 spec = Gem::Specification.new do |s|
-    s.version   =   "0.1.0"
+    s.version   =   "0.1.1"
     s.author    =   "Scott Chacon"

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

diff --git a/lib/simplegit.rb b/lib/simplegit.rb
index a0a60ae..47c6340 100644
--- a/lib/simplegit.rb
+++ b/lib/simplegit.rb
@@ -18,8 +18,3 @@ class SimpleGit
     end

 end
-
-if $0 == __FILE__
-  git = SimpleGit.new
-  puts git.show
-end
\ No newline at end of file
```

Esta opción muestra la misma información, pero añadiendo tras cada entrada las diferencias que le corresponden. Esto resulta muy útil para revisiones de código, o para visualizar rápidamente lo que ha pasado en las confirmaciones enviadas por un colaborador.

También puedes usar con `git log` una serie de opciones de resumen. Por ejemplo, si quieres ver algunas estadísticas de cada confirmación, puedes usar la opción `--stat`:

```
$ git log --stat
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

 Rakefile |    2 +-
 1 files changed, 1 insertions(+), 1 deletions(-)

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

 lib/simplegit.rb |    5 -----
 1 files changed, 0 insertions(+), 5 deletions(-)

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit

 README           |    6 ++++++
 Rakefile         |   23 +++++++++++++++++++++
 lib/simplegit.rb |   25 +++++++++++++++++++++++++
 3 files changed, 54 insertions(+), 0 deletions(-)
```

Como puedes ver, la opción `--stat` imprime tras cada confirmación una lista de archivos modificados, indicando cuántos han sido modificados y cuántas líneas han sido añadidas y eliminadas para cada uno de ellos, y un resumen de toda esta información.

Otra opción realmente útil es `--pretty`, que modifica el formato de la salida. Tienes unos cuantos estilos disponibles. La opción `oneline` imprime cada confirmación en una única línea, lo que puede resultar útil si estás analizando gran cantidad de confirmaciones. Otras opciones son `short`, `full` y `fuller`, que muestran la salida en un formato parecido, pero añadiendo menos o más información, respectivamente:

```
$ git log --pretty=oneline
ca82a6dff817ec66f44342007202690a93763949 changed the version number
085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test code
a11bef06a3f659402fe7563abf99ad00de2209e6 first commit
```

La opción más interesante es `format`, que te permite especificar tu propio formato. Esto resulta especialmente útil si estás generando una salida para que sea analizada por otro programa — como especificas el formato

explícitamente, sabes que no cambiará en futuras actualizaciones de Git —:

```
$ git log --pretty=format:"%h - %an, %ar : %s"
ca82a6d - Scott Chacon, 11 months ago : changed the version number
085bb3b - Scott Chacon, 11 months ago : removed unnecessary test code
a11bef0 - Scott Chacon, 11 months ago : first commit
```

La Tabla 2-1 lista algunas de las opciones más útiles aceptadas por `format`.

Opción	Descripción de la salida
`%H`	Hash de la confirmación
`%h`	Hash de la confirmación abreviado
`%T`	Hash del árbol
`%t`	Hash del árbol abreviado
`%P`	Hashes de las confirmaciones padre
`%p`	Hashes de las confirmaciones padre abreviados
`%an`	Nombre del autor
`%ae`	Dirección de correo del autor
`%ad`	Fecha de autoría (el formato respeta la opción `--date`)
`%ar`	Fecha de autoría, relativa
`%cn`	Nombre del confirmador
`%ce`	Dirección de correo del confirmador
`%cd`	Fecha de confirmación
`%cr`	Fecha de confirmación, relativa
`%s`	Asunto

Puede que te estés preguntando la diferencia entre *autor* (*author*) y *confirmador* (*committer*). El autor es la persona que escribió originalmente el trabajo, mientras que el confirmador es quien lo aplicó. Por tanto, si mandas un parche a un proyecto, y uno de sus miembros lo aplica, ambos recibiréis reconocimiento — tú como autor, y el miembro del proyecto como confirmador —. Veremos esta distinción en mayor profundidad en el Capítulo 5.

Las opciones `oneline` y `format` son especialmente útiles combinadas con otra opción llamada `--graph`. Ésta añade un pequeño gráfico ASCII mostrando tu histórico de ramificaciones y uniones, como podemos ver en nuestra copia del repositorio del proyecto Grit:

```
$ git log --pretty=format:"%h %s" --graph
* 2d3acf9 ignore errors from SIGCHLD on trap
*  5e3ee11 Merge branch `master` of git://github.com/dustin/grit
|\
| * 420eac9 Added a method for getting the current branch.
* | 30e367c timeout code and tests
* | 5a09431 add timeout protection to grit
* | e1193f8 support for heads with slashes in them
|/
* d6016bc require time for xmlschema
*  11d191e Merge branch 'defunkt' into local
```

Éstas son sólo algunas de las opciones para formatear la salida de `git log` — existen muchas más. La Tabla 2-2 lista las opciones vistas hasta ahora, y algunas otras opciones de formateo que pueden resultarte útiles, así como su efecto sobre la salida.

Opción	Descripción
`-p`	Muestra el parche introducido en cada confirmación.
`--stat`	Muestra estadísticas sobre los archivos modificados en cada confirmación.
`--shortstat`	Muestra solamente la línea de resumen de la opción `--stat`.
`--name-only`	Muestra la lista de archivos afectados.
`--name-status`	Muestra la lista de archivos afectados, indicando además si fueron añadidos, modificados o eliminados.
`--abbrev-commit`	Muestra solamente los primeros caracteres de la suma SHA-1, en vez de los 40 caracteres de que se compone.
`--relative-date`	Muestra la fecha en formato relativo (por ejemplo, "2 weeks ago" ("hace 2 semanas")) en lugar del formato completo.
`--graph`	Muestra un gráfico ASCII con la historia de ramificaciones y uniones.
`--pretty`	Muestra las confirmaciones usando un formato alternativo. Posibles opciones son `oneline`, `short`, `full`, `fuller`, y `format` (mediante el cual puedes especificar tu propio formato).

2.3.1. Limitando la salida del histórico

Además de las opciones de formateo, `git log` acepta una serie de opciones para limitar su salida — es decir, opciones que te permiten mostrar únicamente parte de las confirmaciones —. Ya has visto una de ellas, la opción `-2`, que muestra sólo las dos últimas confirmaciones. De hecho, puedes hacer `-<n>`, siendo `n` cualquier entero, para mostrar las últimas `n` confirmaciones. En realidad es poco probable que uses esto con frecuencia, ya que Git por defecto pagina su salida para que veas cada página del histórico por separado.

Sin embargo, las opciones temporales como `--since` (desde) y `--until` (hasta) sí que resultan muy útiles. Por ejemplo, este comando lista todas las confirmaciones hechas durante las dos últimas semanas:

```
$ git log --since=2.weeks
```

Este comando acepta muchos formatos. Puedes indicar una fecha concreta ("2008-01-15"), o relativa, como *"2 years 1 day 3 minutes ago"* ("hace 2 años, 1 día y 3 minutos").

También puedes filtrar la lista para que muestre sólo aquellas confirmaciones que cumplen ciertos criterios. La opción `--author` te permite filtrar por autor, y `--grep` te permite buscar palabras clave entre los mensajes de confirmación. (Ten en cuenta que si quieres aplicar ambas opciones simultáneamente, tienes que añadir `--all-match`, o el comando mostrará las confirmaciones que cumplan cualquiera de las dos, no necesariamente las dos a la vez.)

La última opción verdaderamente útil para filtrar la salida de `git log` es especificar una ruta. Si especificas la ruta de un directorio o archivo, puedes limitar la salida a aquellas confirmaciones que introdujeron un cambio en dichos archivos. Ésta debe ser siempre la última opción, y suele ir precedida de dos guiones (`--`) para separar la ruta del resto de opciones.

En la Tabla 2-3 se listan estas opciones, y algunas otras bastante comunes, a modo de referencia.

Opción	Descripción
`-(n)`	Muestra solamente las últimas n confirmaciones
`--since, --after`	Muestra aquellas confirmaciones hechas después de la fecha especificada.
`--until, --before`	Muestra aquellas confirmaciones hechas antes de la fecha especificada.
`--author`	Muestra sólo aquellas confirmaciones cuyo autor coincide con la cadena especificada.
`--committer`	Muestra sólo aquellas confirmaciones cuyo confirmador coincide con la cadena especificada.

Por ejemplo, si quieres ver cuáles de las confirmaciones hechas sobre archivos de prueba del código fuente de Git fueron enviadas por el usuario `gitster`, y no fueron fusiones, en el mes de octubre de 2008, ejecutarías algo así:

```
$ git log --pretty="%h - %s" --author=gitster --since="2008-10-01" \
   --before="2008-11-01" --no-merges -- t/
5610e3b - Fix testcase failure when extended attribute
acd3b9e - Enhance hold_lock_file_for_{update,append}()
f563754 - demonstrate breakage of detached checkout wi
d1a43f2 - reset --hard/read-tree --reset -u: remove un
51a94af - Fix "checkout --track -b newbranch" on detac
b0ad11e - pull: allow "git pull origin $something:$cur
```

De las casi 20.000 confirmaciones en la historia del código fuente de Git, este comando muestra las 6 que cumplen estas condiciones.

2.3.2. Usando un interfaz gráfico para visualizar el histórico

Si deseas utilizar una herramienta más gráfica para visualizar el histórico de confirmaciones, puede que quieras echarle un ojo a un programa Tcl/Tk llamado gitk que se distribuye junto con Git. Gitk es básicamente un `git log` visual, y acepta casi todas las opciones de filtrado que acepta `git log`. Si tecleas `gitk` en la línea de comandos dentro de tu proyecto, deberías ver algo como lo de la Figura 2-2.

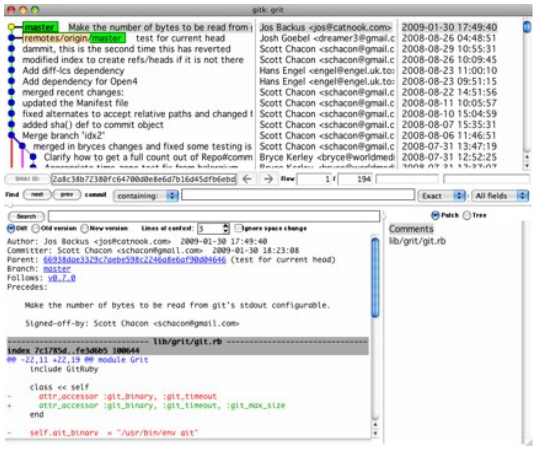

Figura 2.2 El visualizador de histórico gitk

Puedes ver el histórico de confirmaciones en la mitad superior de la ventana, junto con un gráfico de ascendencia. El visor de diferencias de la mitad inferior muestra las modificaciones introducidas en cada confirmación que selecciones.

Anterior
2.2. Guardando cambios en el repositorio

Siguiente
2.4. Deshaciendo cosas