# Ver el Historial 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”. Para clonar el proyecto, ejecuta:

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

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



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 parámetro, `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 historial:

```
$ 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.platform  =   Gem::Platform::RUBY

     s.name      =   "simplegit"

-    s.version   =   "0.1.0"

+    s.version   =   "0.1.1"

     s.author    =   "Scott Chacon"

     s.email     =   "schacon@gee-mail.com"

     s.summary   =   "A simple gem for using Git in Ruby code."



commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
```

```
Author: Scott Chacon <schacon@gee-mail.com>

Date:   Sat Mar 15 16:40:33 2008 -0700



    removed unnecessary test



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 file changed, 1 insertion(+), 1 deletion(-)

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

    removed unnecessary test

 lib/simplegit.rb | 5 -----
 1 file changed, 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(+)
```

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

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, 6 years ago : changed the version number

085bb3b - Scott Chacon, 6 years ago : removed unnecessary test

a11bef0 - Scott Chacon, 6 years ago : first commit
```

Opciones útiles de `git log --pretty=format` lista algunas de las opciones más útiles aceptadas por `format`.

Tabla 1. Opciones útiles de `git log --pretty=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 |

Tabla 1. Opciones útiles de `git log --pretty=format`

| Opción | Descripción de la salida |
|---|---|
| `%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 con mayor profundidad en Git en entornos distribuidos.

Las opciones `oneline` y `format` son especialmente útiles combinadas con otra opción llamada `--graph`. Ésta añade un pequeño gráfico ASCII mostrando tu historial de ramificaciones y uniones:

```
$ 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
```

Este tipo de salidas serán más interesantes cuando empecemos a hablar sobre ramificaciones y combinaciones en el próximo capítulo.

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

Tabla 2. Opciones típicas de `git log`

| 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). |

## Limitar la Salida del Historial

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 historial 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.)

Otra opción útil es `-S`, la cual recibe una cadena y solo muestra las confirmaciones que cambiaron el código añadiendo o eliminando la cadena. Por ejemplo, si quieres encontrar la última confirmación que añadió o eliminó una referencia a una función específica, puede ejecutar:

```
$ git log -Sfunction_name
```

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 Opciones para limitar la salida de `git log` se listan estas opciones, y algunas otras bastante comunes a modo de referencia.

Tabla 3. Opciones para limitar la salida de `git log`

| 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. |
| `-S` | Muestra sólo aquellas confirmaciones que añaden o eliminen código que corresponda 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 Junio Hamano, y no fueron uniones, 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 attributes are in use

acd3b9e - Enhance hold_lock_file_for_{update,append}() API

f563754 - demonstrate breakage of detached checkout with symbolic link
HEAD

d1a43f2 - reset --hard/read-tree --reset -u: remove unmerged new paths

51a94af - Fix "checkout --track -b newbranch" on detached HEAD

b0ad11e - pull: allow "git pull origin $something:$current_branch" into an
unborn branch
```

De las casi 40.000 confirmaciones en la historia del código fuente de Git, este comando muestra las 6 que cumplen estas condiciones.