2.3 Основы Git - Просмотр истории коммитов

**Просмотр истории коммитов**

После того, как вы создали несколько коммитов или же склонировали репозиторий с уже существующей историей коммитов, вероятно вам понадобится возможность посмотреть что было сделано – историю коммитов. Одним из основных и наиболее мощных инструментов для этого является команда git log.

Следующие несколько примеров используют очень простой проект “simplegit”. Чтобы склонировать проект, используйте команду:

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

Если вы запустите команду git log в папке склонированного проекта, вы увидите следующий вывод:

$ 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

По умолчанию (без аргументов) git log перечисляет коммиты, сделанные в репозитории в обратном к хронологическому порядке – последние коммиты находятся вверху. Из примера можно увидеть, что данная команда перечисляет коммиты с их SHA-1 контрольными суммами, именем и электронной почтой автора, датой создания и сообщением коммита.

Команда git log имеет очень большое количество опций для поиска коммитов по разным критериям. Рассмотрим наиболее популярные из них.

Одним из самых полезных аргументов является -p или --patch, который показывает разницу (выводит **патч**), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр -2 для вывода только двух записей:

$ 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

Эта опция отображает аналогичную информацию но содержит разницу для каждой записи. Очень удобно использовать данную опцию для код ревью или для быстрого просмотра серии внесенных изменений. Так же есть возможность использовать серию опций для обобщения. Например, если вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию --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(+)

Как вы видите, опция --stat печатает под каждым из коммитов список и количество измененных файлов, а также сколько строк в каждом из файлов было добавлено и удалено. В конце можно увидеть суммарную таблицу изменений.

Следующей действительно полезной опцией является --pretty. Эта опция меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция oneline выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции short, full и fuller делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно:

$ git log --pretty=oneline

ca82a6dff817ec66f44342007202690a93763949 changed the version number

085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test

a11bef06a3f659402fe7563abf99ad00de2209e6 first commit

Наиболее интересной опцией является format, которая позволяет указать формат для вывода информации. Особенно это может быть полезным когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления 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

[Полезные опции для git log --pretty=format](https://git-scm.com/book/ru/v2/ch00/rpretty_format) отображает наиболее полезные опции для изменения формата.

| Table 1. Полезные опции для git log --pretty=format | |
| --- | --- |
| **Опция** | **Описания вывода** |
| %H | Хеш коммита |
| %h | Сокращенный хеш коммита |
| %T | Хеш дерева |
| %t | Сокращенный хеш дерева |
| %P | Хеш родителей |
| %p | Сокращенный хеш родителей |
| %an | Имя автора |
| %ae | Электронная почта автора |
| %ad | Дата автора (формат даты можно задать опцией --date=option) |
| %ar | Относительная дата автора |
| %cn | Имя коммитера |
| %ce | Электронная почта коммитера |
| %cd | Дата коммитера |
| %cr | Относительная дата коммитера |
| %s | Содержание |

Вам наверное интересно, какая же разница между **автором** и **коммитером**. Автор – это человек, изначально сделавший работу, а коммитер – это человек, который последним применил эту работу. Другими словами, если вы создадите патч для какого-то проекта, а один из основных членов команды этого проекта применит этот патч, вы оба получите статус участника – вы как автор и основной член команды как коммитер. Более детально мы рассмотрим разицу в разделе [[ch05-distributed-git]](https://git-scm.com/book/ru/v2/ch00/ch05-distributed-git).

Опции oneline и format являются особенно полезными с опцией --graph команды log. С этой опцией вы сможете увидеть небольшой граф в формате ASCII, который показывает текущую ветку и историю слияний:

$ 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

Данный вывод будет нам очень интересен в следующей главе, где мы рассмотрим ветвление и слияние.

Мы рассмотрели только несколько простых опций для форматирования вывода с помощью команды git log — на самом деле их гораздо больше. [Наиболее распространенные опции для команды git log](https://git-scm.com/book/ru/v2/ch00/rlog_options) содержит описание как уже рассмотренных, так и нескольких новых опций, которые могут быть полезными в зависимости от нужного формата вывода.

| Table 2. Наиболее распространенные опции для команды git log | |
| --- | --- |
| **Опция** | **Описание** |
| -p | Показывает патч для каждого коммита. |
| --stat | Показывает статистику измененных файлов для каждого коммита. |
| --shortstat | Отображает только строку с количеством изменений/вставок/удалений для команды --stat. |
| --name-only | Показывает список измененных файлов после информации о коммите. |
| --name-status | Показывает список файлов, которые добавлены/изменены/удалены. |
| --abbrev-commit | Показывает только несколько символов SHA-1 чек-суммы вместо всех 40. |
| --relative-date | Отображает дату в относительном формате (например, "2 weeks ago") вместо стандартного формата даты. |
| --graph | Отображает ASCII граф с ветвлениями и историей слияний. |
| --pretty | Показывает коммиты в альтернативном формате. Возможные варианты опций: oneline, short, full, fuller и format (с помощью последней можно указать свой формат). |
| --oneline | Сокращение для одновременного использования опций --pretty=oneline --abbrev-commit. |

**Ограничение вывода**

В дополнение к опциям форматирования вывода, команда git log принимает несколько опций для ограничения вывода – опций, с помощью которых можно увидеть определенное подмножество коммитов. Вы уже видели одну из таких опций — это опция -2, которая показывает только последние два коммита. В действительности вы можете использовать -<n>, где n – это любое натуральное число и представляет собой n последних коммитов. На практике вы не будете часто использовать эту опцию, потому что Git по умолчанию использует постраничный вывод и вы будете видеть только одну страницу за раз.

Однако, опции для ограничения вывода по времени, такие как --since и --until, являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели:

$ git log --since=2.weeks

Это команда работает с большим количеством форматов — вы можете указать определенную дату вида "2008-01-15" или же относительную дату, например "2 years 1 day 3 minutes ago".

Также вы можете фильтровать список коммитов по заданным параметрам. Опция --author дает возможность фильтровать по автору коммита, а опция --grep искать по ключевым словам в сообщении коммита.

|  |  |
| --- | --- |
| **Note** | Допускается указывать несколько параметров --author и --grep для поиска, которые позволят найти коммиты, сооветствующие **любому** указанному --author и **любому** указанному --grep шаблону; однако, применение опции --all-match заставит искать коммиты соответствующие **всем** указанным --grep шаблонам. |

Следующим действительно полезным фильтром является опция -S, которая принимает аргумент в виде строки и показывает только те коммиты, в которых изменение в коде повлекло за собой добавление или удаление этой строки. Например, если вы хотите найти последний коммит, который добавил или удалил вызов определенной функции, вы можете запустить команду:

$ git log -S function\_name

Последней полезной опцией, которую принимает команда git log как фильтр, является путь. Если вы укажете директорию или имя файла, вы ограничите вывод только теми коммитами, в которых были изменения этих файлов. Эта опция всегда указывается последней после двойного тире (--), что отделяет указываемый путь от опций.

В таблице [Опции для ограничения вывода команды git log](https://git-scm.com/book/ru/v2/ch00/rlimit_options) вы можете увидеть эти и другие распространенные опции.

| Table 3. Опции для ограничения вывода команды git log | |
| --- | --- |
| **Опция** | **Описание** |
| -(n) | Показывает только последние n коммитов. |
| --since, --after | Показывает только те коммиты, которые были сделаны после указанной даты. |
| --until, --before | Показывает только те коммиты, которые были сделаны до указанной даты. |
| --author | Показывает только те коммиты, в которых запись author совпадает с указанной строкой. |
| --committer | Показывает только те коммиты, в которых запись committer совпадает с указанной строкой. |
| --grep | Показывает только коммиты, сообщение которых содержит указанную строку. |
| -S | Показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки. |

Например, если вы хотите увидеть, в каких коммитах произошли изменения в тестовых файлах в исходном коде Git в октябре 2008 года, автором которых был Junio Hamano и которые не были комитами слияния, вы можете запустить следующую команду:

$ 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

Из почти 40,000 коммитов в истории исходного кода Git, эта команда показывает только 6, которые соответствуют этим критериям.

|  |  |
| --- | --- |
| **Tip** | **Preventing the display of merge commits**  В зависимости от используемого порядка работы, история коммитов в вашем репозитории может содержать большое количество коммитов слияния, которые сами по себе не очень информативны. Чтобы исключить их из вывода команды git log используйте опцию --no-merges. |

[prev](https://git-scm.com/book/ru/v2/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%97%D0%B0%D0%BF%D0%B8%D1%81%D1%8C-%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9-%D0%B2-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9) | [next](https://git-scm.com/book/ru/v2/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8-%D0%BE%D1%82%D0%BC%D0%B5%D0%BD%D1%8B)