**Tentang GIt**

**Git** adalah sistem kontrol revisi didistribusikan dengan penekanan pada kecepatan.

Setiap Git direktori kerja adalah sebuah repositori matang penuh dengan sejarah lengkap dan kemampuan pelacakan revisi penuh, tidak tergantung pada akses jaringan atau server pusat.

**Setiap klon Git-matang adalah repositori penuh** dengan sejarah lengkap dan kemampuan pelacakan revisi penuh, tidak tergantung pada akses jaringan atau server pusat. **Bercabang dan penggabungan yang cepat** dan mudah dilakukan.

Git digunakan untuk mengendalikan versi dari file, seperti alat seperti Mercurial , Bazaar , Subversion , CVS ,Perforce , dan Visual SourceSafe .

* **Distributed pembangunan.** Seperti kebanyakan modern sistem kontrol versi lain, setiap pengembang Git memberikan salinan lokal dari seluruh sejarah perkembangan, dan perubahan yang disalin dari satu seperti repositori yang lain. Perubahan ini diimpor sebagai cabang pengembangan tambahan, dan dapat digabungkan dalam sebagai cara lokal cabang. dikembangkan sama Repositori dapat dengan mudah diakses melalui protokol Git efisien (opsional dibungkus dalam ssh untuk otentikasi dan keamanan) atau hanya menggunakan HTTP - Anda dapat mempublikasikan repositori Anda di mana saja *tanpa* konfigurasi webserver khusus yang dibutuhkan.
* **Kuat dukungan untuk pembangunan non-linear.** Git mendukung cepat dan mudah bercabang dan penggabungan, dan termasuk alat-alat yang kuat untuk memvisualisasikan dan navigasi linier sejarah pembangunan non-.
* **Efisien penanganan proyek-proyek besar.** Git sangat cepat dan skala baik bahkan ketika bekerja dengan proyek-proyek besar dan sejarah panjang juga. Hal ini umumnya merupakan urutan besarnya lebih cepat dari kontrol versi lain kebanyakan sistem, beberapa lipat lebih cepat pada beberapa operasi. Ini menggunakan format yang dikemas sangat efisien untuk jangka panjang revisi penyimpanan yang saat ini puncak setiap sistem kontrol versi lain sumber terbuka.
* **otentikasi kriptografi sejarah.** Sejarah Git disimpan sedemikian rupa sehingga nama revisi tertentu (suatu “komitmen” dalam hal Git) tergantung pada sejarah perkembangan lengkap menuju ke yang melakukan itu. Setelah dipublikasikan, tidak Dimungkinkan untuk mengubah versi lama tanpa ketahuan. Juga, tag dapat cryptographically ditandatangani.
* **desain Toolkit.** Mengikuti tradisi Unix, Git adalah kumpulan alat kecil yang ditulis dalam C, dan sejumlah skrip yang menyediakan pembungkus nyaman. Git menyediakan alat untuk kedua penggunaan manusia mudah dan mudah scripting untuk melakukan operasi pintar baru.

Selain menyediakan sistem kontrol versi, proyek Git menyediakan toolkit tingkat rendah generik bagi sejarah penyimpanan pohon dan manajemen konten direktori,. Tradisional toolkit ini disebut *pipa saluran air.* Selain user interface datang dengan Git sendiri, beberapa proyek lainnya (begitu- *porselen* disebut) menawarkan antarmuka kontrol versi yang kompatibel - lihat alat terkait daftar.

**2.3 Dasar-dasar Git - Melihat Sejarah Commit**

**Melihat Sejarah Commit**

Setelah Anda membuat beberapa commit, atau jika Anda sudah menduplikasi sebuah repositori dengan sejumlah sejarah commit yang telah terjadi, Anda mungkin akan mau untuk melihat ke belakang untuk mengetahui apa yang sudah pernah terjadi. Alat paling dasar dan tepat untuk melakukan ini adalah perintah git log.

Contoh berikut menggunakan sebuah proyek sangat sederhana yang disebut simplegit yang sering saya gunakan untuk keperluan demonstrasi. Untuk mengambil proyek ini, lakukan

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

Ketika Anda jalankan git log dalam proyek ini, Anda akan mendapat keluaran yang mirip seperti berikut:

$ 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

Secara standar, dengan tanpa argumen, git log menampilkan daftar commit yang pernah dibuat di dalam repositori ini terurut secara kronologis terbalik. Yaitu, commit terbaru muncul paling atas. Seperti yang dapat Anda lihat, perintah ini menampilkan setiap commit dengan nlai checksum SHA-1, nama dan email dari pengubah, tanggal perubahan dilakukan, dan pesan commitnya.

Sebagian besar variasi opsi dari perintah git log tersedia untuk menunjukkan kepada Anda secara tepat apa yang Anda cari. Di sini, kami akan menunjukkan kepada Anda beberapa dari opsi yang paling sering digunakan.

Salah satu dari opsi yang paling berguna adalah -p, karena menampilkan diff dari setiap commit. Anda juga dapat menggunakan -2, yang membantu membatasi keluarannya hingga 2 entri terakhir:

$ 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

Opsi ini menampilkan informasi log yang sama, namun ditambah informasi diff dari setiap entri. Ini sangat membantu untuk proses tilik-ulang kode atau untuk secara cepat menelusuri apa yang telah terjadi dalam serangkaian commit yang telah ditambahkan oleh rekan kolaborasi. Anda juga dapat menggunakan serangkaian opsi simpulan menggunakan git log. Misalnya, jika Anda ingin melihat statistik dari setiap commit, Anda dapat menggunakan osi --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(-)

Seperti Anda dapat lihat, opsi --stat menampilkan di bawah setiap entri commit sebuah daftar dari berkasi terubah, jumlah berkas yang diubah dan jumlah baris dalam berkas tersebut yang ditambah atau dihapus. Opsi ini juga menambahkan sebuah simpulan dari informasi tadi di bagian akhir. Opsi lain yang juga berguna adalah --pretty. Opsi ini mengubah keluaran log ke dalam bentuk selain dari bentuk standar. Beberapa pilihan bentuk yang telah dibuat sebelumnya dapat Anda gunakan. Pilihan bentuk oneline akan mencetak setiap commit dalam satu baris, yang berguna jika Anda melihat banyak sekali commit. Selain itu, ada pilihan bentuk short, full, dan fuller yang menampilkan keluaran dalam format yang kurang lebih sama tetapi dengan lebih sedikit atau lebih banyak informasi, seperti:

$ git log --pretty=oneline

ca82a6dff817ec66f44342007202690a93763949 changed the version number

085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7 removed unnecessary test code

a11bef06a3f659402fe7563abf99ad00de2209e6 first commit

Yang lebih mearik adalah pilihan bentuk format, yang memungkinkan kita untuk menentukan format keluaran log yang kita inginkan. Ini secara khusus berguna jika Anda membuat keluaran untuk diolah oleh mesin - karena Anda menentukan format secara eksplisit, Anda tahu keluaran tidak akan berubah jika Git dimutakhirkan.

$ 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

Tabel 2-1 memperlihatkan beberapa opsi berguna yang dapat digunakan oleh format.

| **Opsi** | **Penjabaran dari keluaran** |
| --- | --- |
| %H | Hash dari commit |
| %h | Hash dari commit dalam versi pendek |
| %T | Hash dari pohon |
| %t | Hash dari pohon dalam versi pendek |
| %P | Hash dari parent |
| %p | Hash dari parent dalam versi pendek |
| %an | Nama pembuat |
| %ae | Email pembuat |
| %ad | Tanggal pembuat (format juga memperhitungkan opsi -date=) |
| %ar | Tanggal pembuat, relatif |
| %cn | Name pelaku commit |
| %ce | Email pelaku commit |
| %cd | Tanggal pelaku commit |
| %cr | Tanggal pelaku commit, relatif |
| %s | Judul |

Anda mungkin bertanya-tanya apa perbedaan dari *pembuat* dan *pelaku*commit\_. Pembuat adalah orang yang sebetulnya menulis perubahan, sedangkan pelaku commit adalah orang yang terakhir mengaplikasikan perubahan tersebut. Jadi, jika Anda mengirimkan sebuah patch ke sebuah proyek dan salah satu dari anggota inti mengaplikasikan patch tersebut, Anda berdua akan dihitung - Anda sebagai pembuat dan anggota inti sebagai pelaku commit. Perbedaan ini ini akan kita bahas lebih lanjut di Bab 5.

Opsi oneline dan format secara khusus berguna dengan opsi log lainnya yang disebut --graph. Opsi ini menambah informasi gambar ASCII yang menunjukkan sejarah pencabangan dan penggabungan, yang kita dapat lihat dari salinan repositori proyek 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

Itulah beberapa opsi dalam memformat keluaran dari git log secara sederhana - masih ada banyak lagi. Tabel 2-2 menjabarkan opsi-opsi yang sejauh ini telah kita bahas dan beberapa opsi format umum lainnya yang mungkin berguna, sejalan dengan bagaimana opsi tersebut mengubah keluaran dari perintah log.

| **Opsi** | **Penjabaran** |
| --- | --- |
| -p | Tampilkan patch yang digunakan di setiap commit |
| --stat | Tampilkan statistik dari berkas terubah di setiap commit |
| --shortstat | Tampilkan opsi --stat dalam satu baris perubahan/penambahan/penghapusan |
| --name-only | Tampilkan daftar berkas yang terubah setelah setiap informasi commit |
| --name-status | Tampilkan daftar berkas yang terubah dan informasi status tertambah/terubah/terhapus |
| --abbrev-commit | Tampilkan beberapa karakter awal dari ceksum SHA-1 |
| --relative-date | Tampilkan tanggal dalam bentuk relatif (misalnya, "2 weeks ago") |
| --graph | Tampilkan gambar ASCII dari sejarah pencabangan dan penggabungan di samping keluaran log |
| --pretty | Tampilkan commit dalam format alternatif. Opsi antara lain oneline, short, full, fuller dan format (dimana kita dapat merumuskan format yang kita inginkan). |

[**Membatasi Keluaran Log**](http://git-scm.com/book/id/Dasar-dasar-Git-Melihat-Sejarah-Commit#Membatasi-Keluaran-Log)

Sebagai tambahan dari opsi format-keluaran, git log juga memiliki opsi pembatasan yang berguna - yaitu opsi yang membuat kita dapat menampilkan sebagian dari commit. Anda telah melihat salah satu opsi pembatasan ini sebelumnya - opsi -2 yang menampilkan 2 commit terakhir. Bahkan jika Anda melakukan -<n>, dengan n adalah integer apapun untuk menampilkan sejumlah n commit terakhir. Dalam kenyataannya, Anda mungkin tidak akan menggunakan opsi ini terlalu sering, karena Git secara standar melakukan pipe dari semua output lewat sebuah pager sehingga Anda melihat hanya sebuah halaman dari keluaran log setiap saat.

Namun demikian, opsi pembatasan waktu seperti --since dan --until akan lebih berguna. Sebagai contoh, perintah berikut akan menampilkan sejumlah commit yang dilakukan dalam 2 minggu terakhir:

$ git log --since=2.weeks

Perintah ini bekerja dengan format lainnya - Anda dapat mencantumkan tanggal tertentu ("2008-01-15") atau tanggal relatif seperti "2 years 1 day 3 minutes ago".

Anda juga dapat menyaring daftar untuk commit yang cocok dengan beberapa kriteria pencarian. Opsi --author membuat Anda dapat menyaring pembuat tertentu, dan opsi --grep membuat Anda dapat mencari keyword di dalam pesan commit. (Mohon diingat bahwa jika Anda ingin mencantumkan kedua opsi author dan grep, Anda harus menambahkan --all-match atau perintah akan mencocokkan yang berisi keduanya saja).

Opsi terakhir yang sangat berguna untuk menyaring git log adalah path. Jika anda mencantumkan direktori atau nama berkas, Anda dapat membatasi keluaran log ke commit yang merubah berkas-berkas tersebut. Ini selalu menjadi opsi terakhir dan biasanya didahului dengan dua tanda hubung (--) untuk memisahkan path dari opsi lainnya.

Dalam tabel 2-3 kita daftarkan opsi pembatasan ini dan opsi umum lainnya untuk acuan Anda.

| **Opsi** | **Penjabaran** |
| --- | --- |
| -(n) | Tampilkan hanya sejumlah n commit terakhir |
| --since, --after | Batasi commit hanya yang dibuat setelah tanggal yang dicantumkan |
| --until, --before | Batasi commit hanya yang dibuat sebelum tanggal yang dicantumkan |
| --author | Hanya tampilkan commit yang entri pembuatnya cocok dengan string yang dicantumkan |
| --committer | Hanya tampilkan commit yang entri pelaku commitnya cocok dengan string yang dicantumkan |

Sebagai contoh, jika Anda ingin melihat commit mana saja yang mengubah berkas test di sejarah kode sumber yang di-commit oleh Junio Hamano dan bukan merupakan penggabungan selama bulan October 2008, Anda dapat menjalankan seperti berikut:

$ 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

Dari sekitar 20,000 commit dalam sejarah kode sumber Git, perintah ini menampilkan hanya 6 yang cocok dengan kriteria di atas.