Kontrol Versi

dengan git

Alex Xandra Albert Sim

Kontrol Versi dengan Git

Alex Xandra Albert Sim

This book is for sale at http://leanpub.com/kontrol-versi-git

This version was published on 2013-07-11

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Alex Xandra Albert Sim by spreading the word about this book on Twitter!

The suggested tweet for this book is:

Baru membeli buku ”Kontrol Versi dengan Git”! Cek bukunya di https://leanpub.com/kontrol-versi-git

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#

Contents

Penggunaan Git Secara Mandiri

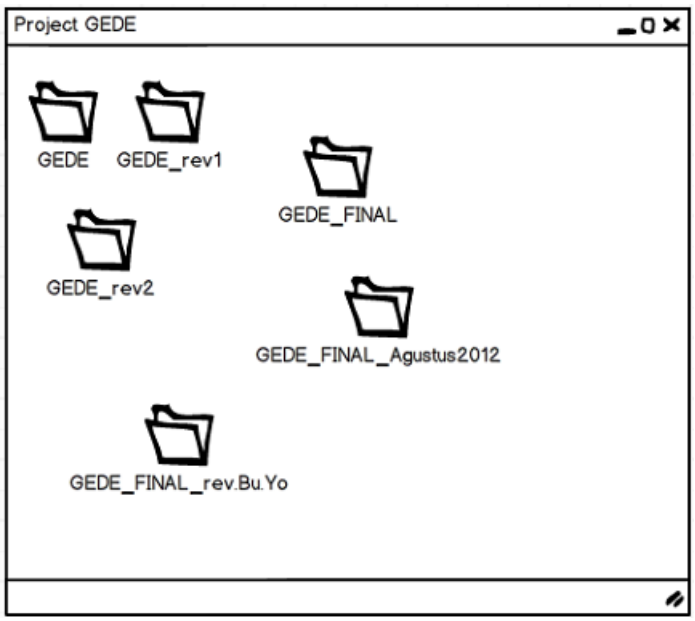
Dalam melakukan pemrograman, perubahan spesifikasi atau kebutuhan adalah hal yang tidak dapat dihindari. Tidak ada program yang dapat dituliskan dengan sempurna pada percobaan pertama. Hal ini menyebabkan pengembang perangkat lunak sangat dekat dengan sistem kontrol versi, baik secara manual maupun menggunakan perangkat lunak khusus. Bagian ini akan membahas tentang sistem kontrol versi, kegunaannya, serta contoh kasus menggunakan `git`, salah satu perangkat lunak populer untuk kontrol versi.

Dasar Kontrol Versi

Kegunaan utama dari sistem kontrol versi ialah sebagai alat untuk manajemen kode program. Terdapat dua kegunaan utama dari sistem ini, yaitu:

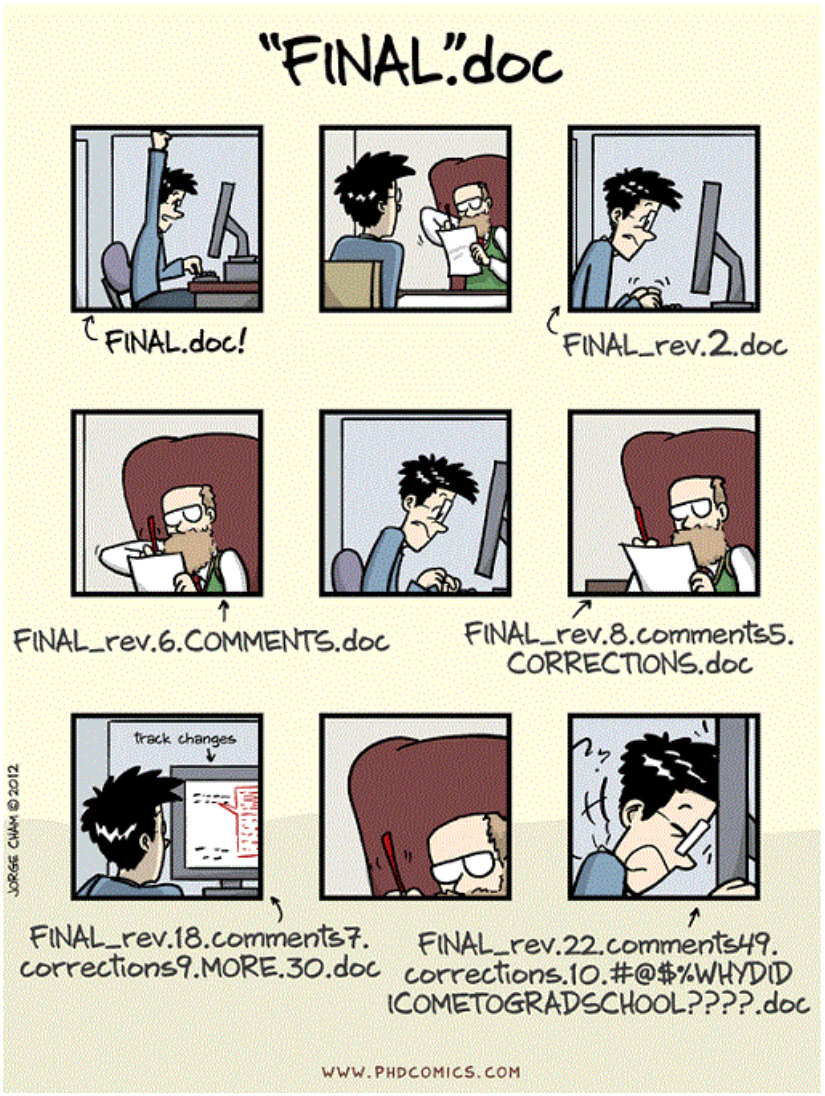
1. Menyimpan versi lama dari kode, maupun
2. Menggabungkan perubahan-perubahan kode dari versi lama (misal: untuk mengembalikan fitur yang telah dihapus) ataupun menggabungkan perubahan dari orang lain (misal: menggabungkan fitur yang dikembangkan oleh anggota tim lain).

Tanpa menggunakan sistem kontrol versi, yang sering saya temukan (dan dulunya saya gunakan, sebelum mengetahui tentang kontrol versi) ialah pengunaan direktori untuk memisahkan beberapa versi program, seperti berikut:



BU YO CEREWET BANGET!

yang menyebabkan kita berakir seperti ini:



Sumber: PHD Comics.

Dan kedua ilustrasi di atas hanya menjelaskan masalah "penyimpanan versi lama", belum sampai ke penggabungan kode. Penggabungan kode, baik dengan versi lama maupun dengan kode orang lain kemungkinan besar adalah salah satu penyebab utama sakit kepala sebagian besar programmer yang ada.

Sistem kontrol versi, seperti `git`, `hg`, atau `bzr`, dikembangkan untuk menyelesaikan masalah-masalah di atas. Karena tidak ingin membahas terlalu banyak, artikel ini hanya akan menjelaskan pengunaan `git`, karena kelihatannya `git` merupakan perangkat lunak kontrol versi yang paling populer untuk sekarang (mengingat popularitas Github dan pengunaan `git` pada kernel Linux).

Instalasi Git

`git` berjalan pada semua sistem operasi populer (Mac, Windows, Linux). Jika anda menggunakan Windows atau Mac, masuk ke situs utama `git` pada git-scm.com lalu lakukan download

dan instalasi software tersebut. Pengguna Linux dapat melakukan instalasi melalui repositori distribusi yang dilakukan, melalui perintah sejenis:

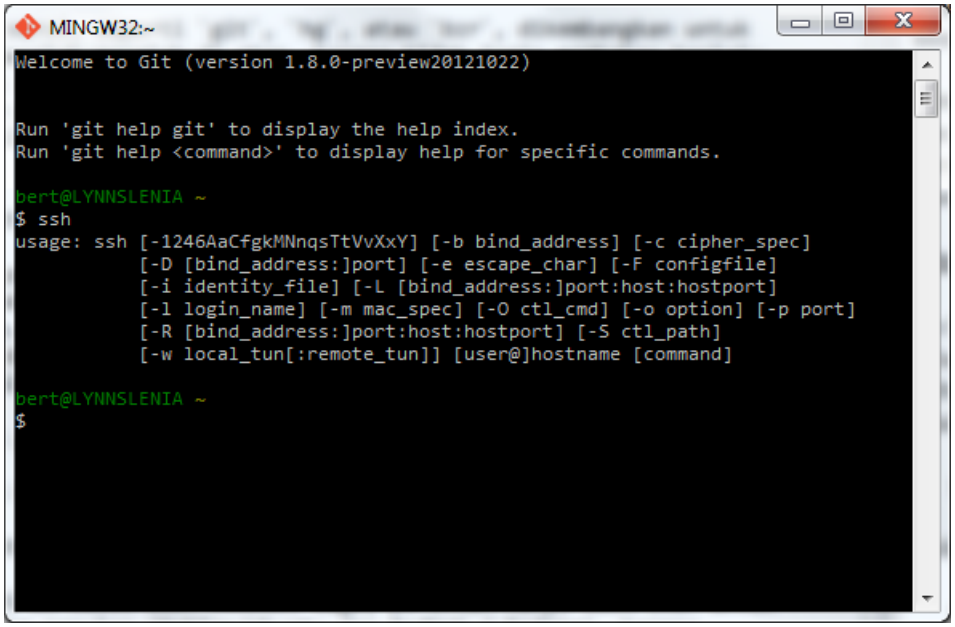
```
yum install git
```

pada repositori berbasis RPM, atau perintah

```
apt-get install git
```

untuk repositori berbasis deb. Kembali lagi, perintah hanya diberikan untuk distribusi paling populer (Debian/Ubuntu dan RedHat / Fedora), karena keterbatasan ruang. Jika anda menggunakan distrusi lain (seperti Gentoo atau Arch), maka diasumsikan anda telah mengetahui cara instalasi `git` atau perangkat lunak lain pada umumnya.

Khusus untuk sistem operasi Windows, pastikan instalasi anda diambil dari git-scm.com, karena pada paket yang tersedia di website tersebut telah diikutkan juga OpenSSH, yang akan sangat berguna jika ingin berkolaborasi dengan programmer lain. Verifikasi dapat dilakukan dengan **menjalankan** `git bash` **melalui Start Menu**, dan kemudian mengetikkan `ssh`, seperti berikut (perhatikan ikon yang muncul, gambar menggunakan `git bash`, bukan `cmd.exe`):



yeah, harus dari command line.

Jika belum berhasil mendapatkan hasil yang tepat, lakukan instalasi OpenSSH terlebih dahulu. Meskipun belum akan digunakan pada bagian ini, OpenSSH merupakan sebuah perangkat lunak penting yang akan selalu digunakan bersamaan dengan `git`. Bagian selanjutnya dari tulisan ini akan memerlukan OpenSSH. Instalasi OpenSSH dapat dilakukan dengan mengikuti langkah-langkah pada website berikut.

Selain perintah `ssh`, pastikan juga bahwa perintah `git` memberikan respon yang benar, seperti kode berikut:

```
bert@LYNNSLENIA ~
$ git
usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [-\
-info-path]
           [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [-c name=value] [--help]
           <command> [<args>]

The most commonly used git commands are:
   add        Add file contents to the index
   bisect     Find by binary search the change that introduced a bug
   branch     List, create, or delete branches
   checkout   Checkout a branch or paths to the working tree
   clone      Clone a repository into a new directory
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   fetch      Download objects and refs from another repository
   grep       Print lines matching a pattern
   init       Create an empty git repository or reinitialize an existing one
   log        Show commit logs
   merge      Join two or more development histories together
   mv         Move or rename a file, a directory, or a symlink
   pull       Fetch from and merge with another repository or a local branch
   push       Update remote refs along with associated objects
   rebase     Forward-port local commits to the updated upstream head
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index
   show       Show various types of objects
   status     Show the working tree status
   tag        Create, list, delete or verify a tag object signed with GPG

See 'git help <command>' for more information on a specific command.

bert@LYNNSLENIA ~
$
```

Tulisan ini juga akan selalu menggunakan *command line*, karena perintah-perintah yang dipelajari pada *command line* dapat digunakan pada seluruh sistem operasi - tidak tergantung kepada perangkat lunak yang digunakan.

Jika telah berhasil menjalankan `ssh` dan `git` serta mendapatkan respon yang benar, sesuai dengan gambar dan contoh kode yang diberikan sebelumnya, mari kita lanjutkan ke bagian berikutnya.

Inisialisasi Repositori Git

Untuk dapat menggunakan sistem kontrol versi, terlebih dahulu kita harus mempersiapkan **repositori**. Sebuah repositori menyimpan seluruh versi dari kode program kita. Tidak usah takut, karena repositori tidak akan memakan banyak ruang *hard disk*, karena penyimpanan tidak dilakukan terhadap keseluruhan file. Repositori hanya akan menyimpan *perubahan* yang terjadi pada kode kita dari satu versi ke versi lainnya. Bahasa kerennya, repositori hanya menyimpan **delta** dari kode pada setiap versinya.

Pada zaman dahulu kala (di saat kontrol versi yang populer adalah `cvs` dan programmer pada umumnya berjanggut putih), membangun repositori kode baru adalah hal yang sangat sulit dilakukan. Kita harus memiliki sebuah *server* khusus yang dapat diakses oleh seluruh anggota tim. Jika server tidak dapat diakses karena jaringan rusak atau internet putus, maka kita tidak dapat menggunakan sistem kontrol versi (dan harus kembali ke metode direktori, atau tidak bekerja).

Untungnya, `git` merupakan sistem kontrol versi terdistribusi, yang berarti `git` dapat dijalankan tanpa perlu adanya repositori terpusat. Yang kita perlukan untuk membuat repositori ialah mengetikkan perintah tertentu di direktori utama kode kita.

Mari kita mulai membuat repositori baru. Pertama-tama, buat sebuah direktori baru untuk melakukan eksperimen kode. Pada contoh dalam buku ini, direktori dibuat dan disimpan pada `~/Desktop/projects/git-tutor`. Buat direktori tersebut, kemudian masuk ke dalam direktorinya, seperti berikut:

```
bert@LYNNSLENIA ~
$ mkdir Desktop/projects/git-tutor

bert@LYNNSLENIA ~
$ cd Desktop/projects/git-tutor/

bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ ls

bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$
```

Perhatikan bahwa pada awalnya, direktori ini kosong. Kita akan menambahkan kode baru ke dalam direktori ini. Buat sebuah file baru yang bernama `cerita.txt` di dalam direktori tersebut:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ echo "ini adalah sebuah cerita" > cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ ls
cerita.txt
```

File teks dan Source Code

Untuk menyederhanakan tulisan, maka contoh yang diberikan hanya menggunakan file teks. Segala perintah dan konsep yang digunakan dapat juga diterapkan pada kode program, karena pada dasarnya kode program adalah file teks.

dan kemudian masukkan perintah `git init` untuk melakukan inisialisasi repositori:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor
$ git init
Initialized empty Git repository in c:/Users/bert/Desktop/projects/git-tuto\
r/.git/
```

Setelah melakukan inisialisasi, `git` secara otomatis akan membuat direktori `.git` pada repositori kita (lihat potongan kode di bawah). **Jangan lakukan apapun terhadap direktori ini**. Direktori tersebut merupakan direktori yang digunakan oleh `git` untuk menyimpan basis data delta kode kita, dan berbagai metadata lainnya. Mengubah direktori tersebut dapat menyebabkan hilangnya seluruh *history* dari kode sehingga kita tidak lagi dapat mengakses versi lama dari file yang telah dicatat oleh `git`.

Sampai titik ini, direktori `git-tutor` telah berisi sebuah file (`cerita.txt`) dan direktori (`.git`). Cek kembali apakah hal ini sudah benar dengan menjalankan perintah `ls`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls -a
.  ..  .git  cerita.txt
```

Jika sudah tepat maka kita dapat melanjutkan eksperimen dengan menambahkan file baru ke repositori.

Penambahan File ke Repositori

Setelah memiliki repositori, tentunya kita ingin menyimpan sejarah dari kode kita. Penyimpanan sejarah dapat dimulai dari saat pertama: kapan file tersebut dibuat dan ditambahkan ke dalam repositori. Untuk menambahkan file ke dalam repositori, gunakan perintah `git add`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git add .
warning: LF will be replaced by CRLF in cerita.txt.
The file will have its original line endings in your working directory.
```

Kenapa ada Warning?

Peringatan yang diberikan oleh `git` pada contoh di atas tidak perlu diperhatikan. Pada dasarnya, peringatan ini hanya memberitahukan bahwa file akan disimpan oleh git dalam format pengganti baris Unix. Hal ini tidak akan terlalu berpengaruh, karena hal seperti ini biasanya ditangani oleh editor secara otomatis.

Secara sederhana, sintaks dari perintah `git add` adalah sebagai berikut:

```
git add [nama file atau pola]
```

Tetapi perhatikan bahwa pada perintah di atas, kita memasukkan `.` alih-alih nama file. Memasukkan `.` pada nama file dalam perintah `git add` akan memerintahkan `git` untuk menambahkan **semua** file baru dalam repositori. Jika hanya ingin menambahkan satu file (misalkan ada file yang belum yakin akan ditambahkan ke repositori), nama file spesifik dapat dimasukkan:

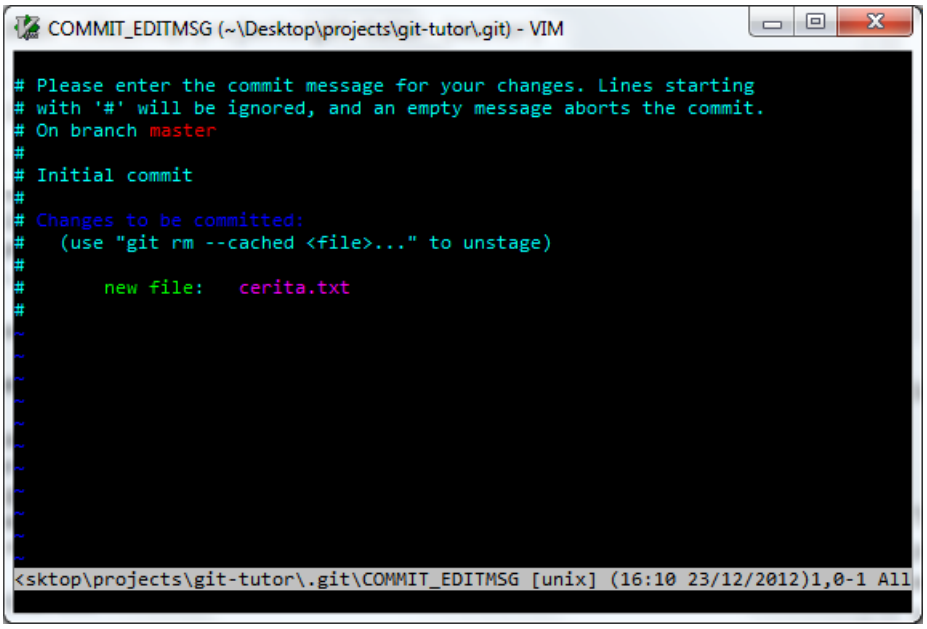
```
git add cerita.txt
```

Setelah menambahkan file ke dalam repositori, kita harus melakukan *commit*. Perintah *commit* memberitahukan kepada `git` untuk menyimpan sejarah dari file yang telah ditambahkan. Pada `git`, penambahan, perubahan, ataupun penghapusan sebuah file baru akan tercatat jika perntah *commit* telah dijalankan. Sederhananya, memberikan perintah *commit* berarti berkata kepada `git` "Oi `git`, file yang tadi ditambahkan dan diubah itu dicatet ya. Masukin ke daftar sejarah."

Mari lakukan *commit* dengan menjalankan perintah `git commit`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
```

Perhatikan bahwa setelah memasukkan perintah `git commit`, anda akan dibawa ke sebuah teks editor untuk mengisikan pesan:



pengisian pesan pada git

Jika bingung, teks editor yang digunakan secara standar ialah vim. Cara mengganti teks editor standar pada `git` dapat dibaca pada bagian lampiran. Untuk sekarang, jika anda bukan pengguna vim, tidak usah bingung, lakukan langkah-langkah berikut untuk memasukkan pesan:

1. Tekan `i` pada keyboard untuk masuk ke dalam mode `insert`.
2. Masukkan pesan yang diinginkan, misalkan: "Inisialisasi repo. Penambahan cerita.txt."
3. Tekan `Esc` pada keyboard untuk kembali ke mode `normal`.
4. Tekan `:wq` dan kemudian `Enter` pada keyboard anda untuk keluar dari vim dan menyimpan data.

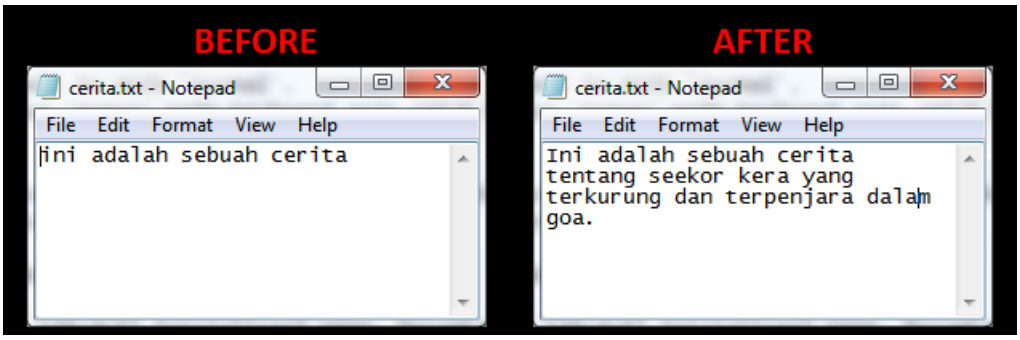
Jika langkah di atas diikuti dengan benar, maka kita akan dibawa kembali ke `git bash`, dengan pesan berikut:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
[master (root-commit) 1d4cdc9] Inisialisasi repo. Penambahan cerita.txt.
warning: LF will be replaced by CRLF in cerita.txt.
The file will have its original line endings in your working directory.
 1 file changed, 1 insertion(+)
 create mode 100644 cerita.txt
```

Selamat, anda telah berhasil melakukan *commit* pertama! Selanjutnya, mari kita coba untuk mengubah isi dari file untuk melihat bagaimana git menangani perubahan file.

Mengubah Isi File

Kegunaan utama kontrol versi (yang tercermin dari namanya) ialah melakukan manajemen perubahan secara otomatis untuk kita. Mari kita lihat apakah `git` benar-benar melakukan hal tersebut. Lakukan perubahan isi pada `cerita.txt`:



karena "Before" dan "After" bukan monopoli produk kecantikan :D

dan kemudian jalankan perintah `git commit` lagi:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working director\
y)
#
#       modified:   cerita.txt
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Perhatikan bahwa `git` secara otomatis mengetahui file mana saja yang berubah, tetapi tidak melakukan pencatatan perubahan tersebut. Untuk memerintahkan `git` mencatat perubahan tersebut, gunakan perintah `git commit -a`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit -a
[master 61c4707] Kapitalisasi dan melengkapi kalimat.
 1 file changed, 1 insertion(+), 1 deletion(-)
```

`git commit -a`

Perintah `git commit -a` ini adalah merupakan perintah singkat untuk memanggil `git add` dan `git commit` dalam satu perintah. Karena harus menjalankan perintah tersebut setiap kali melakukan modifikasi, maka kita dapat langsung menjalankan `git commit -a` alih-alih kedua perintah tersebut.

Selain melakukan perubahan, tentunya terkadang kita ingin mengetahui perubahan-perubahan apa saja yang terjadi selama pengembangan. Untuk melihat daftar perubahan yang telah dilakukan, kita dapat menggunakan perintah `git log`:

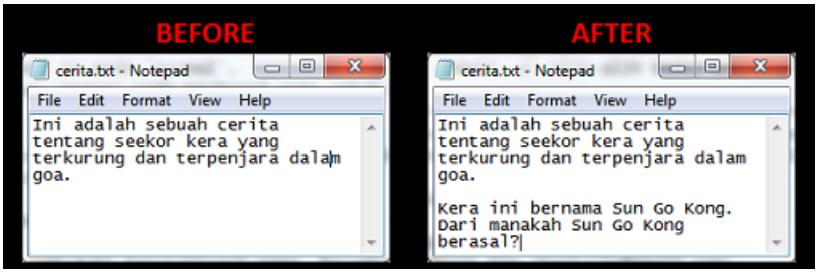
```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git log
commit 61c47074ee583dbdd16fa9568019e80d864fb403
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:36:46 2012 +0700

    Kapitalisasi dan melengkapi kalimat.

commit 1d4cdc9350570230d352ef19aededf06769b0698
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:10:33 2012 +0700

    Inisialisasi repo. Penambahan cerita.txt.
```

Hanya untuk memamerkan fitur `git log` ini, mari lakukan perubahan lagi terhadap `cerita.txt`:

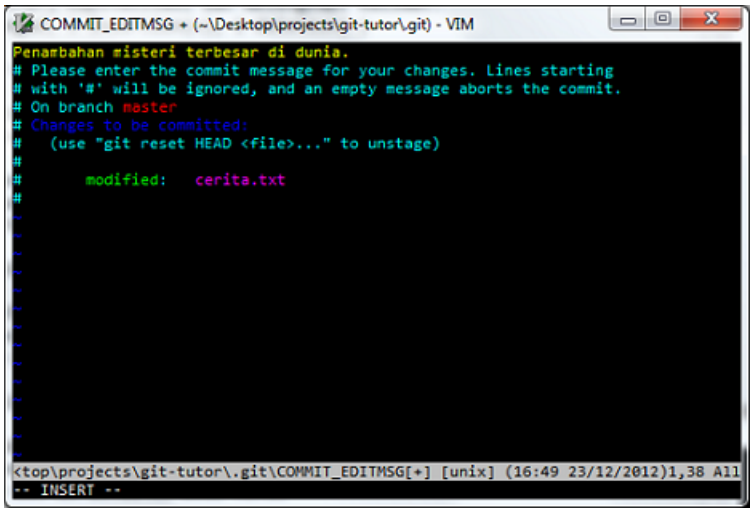


Kera sakti~ Tak pernah berhenti bertindak sesuka hati~

dan lakukan *commit* sekali lagi:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit -a
```

dengan pesan commit:



Ya. Paling besar.

Mari jalankan perintah `git log` sekali lagi, untuk melihat hasil pekerjaan kita sejauh ini:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git log
commit 28dabb1c54a086cce567ecb890b10339416bcbfa
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:49:21 2012 +0700

    Penambahan misteri terbesar di dunia.

commit 61c47074ee583dbdd16fa9568019e80d864fb403
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:36:46 2012 +0700

    Kapitalisasi dan melengkapi kalimat.

commit 1d4cdc9350570230d352ef19aededf06769b0698
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:10:33 2012 +0700

    Inisialisasi repo. Penambahan cerita.txt.
```

Ok, sejauh ini seluruhnya berjalan dengan baik. Sampai tahap ini, pertanyaan yang biasanya paling sering diajukan kepada saya (terkadang secara langsung, terkadang melalui pandangan mata) adalah: “KOK JADI REPOT GINI? TIAP KALI GANTI DIKIT MUSTI COMMIT. DARI DULU-DULU CODING GAK PERLU COMMIT-COMMITAN GINI JUGA GAK PERNAH ADA MASALAH KOK!!1!”

Sabar nak. Mari kita lihat kenapanya.

Mengembalikan File ke Versi Lama

Bayangkan kalau suatu hari, ketika sedang istirahat makan siang, kucing kantor anda melompat ke meja, dan tidur di atas keyboard. Selama tiduran di atas keyboard, kucing tersebut tidak sengaja menekan tombol untuk menghapus file anda.

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ rm cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$
```

Sebelum menggunakan `git` atau sistem kontrol versi lainnya, saat seperti ini adalah saat-saat di mana kita mendadak perduli terhadap sistem *backup*. Kenapa tidak ada sistem *backup* di perusahaan kita? Bukankah *backup* adalah sistem yang paling penting di dunia, khususnya di Indonesia, karena PLN yang tidak punya cukup energi? Kenapa kantor ini memelihara kucing yang bebas keluar masuk? Kenapa kucing suka tidur di atas keyboard? Tak jarang, setelah mengajukan pertanyaan-pertanyaan filosofis yang mendalam tadi, programmer yang mengalami hal tragis tersebut akan mengurung diri dan bertapa di Gunung Hwa Ko, untuk mendapatkan ilmu baru.

Ilmu baru tersebut adalah `git`.

`git` memungkinkan kita untuk mengembalikan kode ke dalam keadaan sebelumnya, yaitu *commit* terakhir. Kita dapat melakukan pengembalian kode ini dengan menggunakan perintah `git checkout`, seperti berikut:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout HEAD -- cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
dalam goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$
```

Parameter `HEAD` pada perintah yang kita jalankan merupakan parameter untuk memberitahukan `git checkout` bahwa kita ingin mengembalikan kode pada revisi terakhir (`HEAD` dalam istilah `git`). Karena hanya ingin mengembalikan file `cerita.txt`, maka kita harus memberitahukan `git checkout`, melalui parameter `-- cerita.txt`. Perintah `git checkout` juga memiliki banyak kegunaan lainnya selain mengembalikan kode ke revisi tertentu. Pengunaan `git checkout` pada kasus-kasus lainnya akan dijelaskan lebih rinci pada bagian selanjutnya.

Pengecekan Status Repositori

Terkadang, setelah bekerja seharian, kita seringkali lupa apa-apa saja yang telah kita kerjakan. Tidak usah jauh-jauh, terkadang saya bahkan tidak ingat *apa yang harus saya kerjakan hari ini*. Untungnya, `git` memberikan fitur untuk melihat apa saja yang telah kita kerjakan yang belum di-*commit*. Untuk melihat bagaimana fitur ini bekerja, mari lakukan perubahan pada repositori terlebih dahulu. Tambahkan sebuah file baru ke dalam repositori:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ echo "Seekor kera, terpuruk, terpenjara dalam goa. Di gunung suci sunyi
tempat hukuman para dewa." > lagu-intro.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git add .
warning: LF will be replaced by CRLF in lagu-intro.txt.
The file will have its original line endings in your working directory.

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit -m "Penambahan lagu intro"
[master 03d0628] Penambahan lagu intro.
warning: LF will be replaced by CRLF in lagu-intro.txt.
The file will have its original line endings in your working directory.
 1 file changed, 1 insertion(+)
 create mode 100644 lagu-intro.txt
```

Perintah "`git commit -m`"

Perhatikan bahwa pada *commit* kali ini digunakan perintah `git commit -m`, yang berguna untuk memberikan pesan *commit* secara langsung dalam satu perintah.

Kemudian kita akan melakukan edit terhadap `cerita.txt` dan mengganti nama `lagu-intro.txt` menjadi `lagu-intro-awal.txt`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ notepad cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ mv lagu-intro.txt lagu-intro-awal.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro-awal.txt
```

Setelah melakukan perubahan tersebut, kita mengalami amnesia sesaat karena kucing kantor jatuh ke kepala kita (kucing yang menyebalkan!). Karena telah lupa akan perubahan yang dilakukan, kita dapat melihat apa saja yang berubah dengan menggunakan perintah `git status`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working director\
y)
#
#       modified:   cerita.txt
#       deleted:    lagu-intro.txt
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       lagu-intro-awal.txt
no changes added to commit (use "git add" and/or "git commit -a")
```

Perhatikan bahwa terdapat dua bagian dari status yang diberikan:

1. “Changes not staged for commit” menampilkan daftar file yang berubah, tetapi belum di-*commit*. File yang tercatat ini termasuk file yang diubah dan dihapus.
2. “Untracked files” menampilkan file yang belum ditambahkan ke dalam repositori.

Jika ingin melihat apa saja yang diubah pada file `cerita.txt`, kita dapat menggunakan perintah `git diff`:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git diff cerita.txt
diff --git a/cerita.txt b/cerita.txt
index 846114d..dbcb596 100644
--- a/cerita.txt
+++ b/cerita.txt
@@ -1,3 +1,3 @@
 Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara\
 dala

-Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
\ No newline at end of file
+Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???!
\ No newline at end of file
(END)
```

Format yang ditampilkan mungkin agak membingungkan, tetapi tidak usah takut, karena bagian yang perlu diperhatikan hanyalah pada bagian yang bertanda `-` dan `+`. Pada `git bash`, bahkan bagian ini diberi warna (merah untuk `-` dan hijau untuk `+`). Tanda `+`, tentunya berarti bagian yang ditambahkan, dan tanda `-` berarti bagian yang dihapus. Dengan melihat perubahan pada baris yang bersangkutan, kita dapat mengetahui bahwa `?` diubah menjadi `???!` pada akhir baris.

Setelah mengetahui perubahan yang dilakukan, dan menganggap perubahan tersebut aman untuk di-*commit*, kita lalu dapat melakukan *commit* seperti biasa:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git add lagu-intro-awal.txt
warning: LF will be replaced by CRLF in lagu-intro-awal.txt.
The file will have its original line endings in your working directory.

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git commit -m "Dramatisasi cerita dan perubahan nama file lagu."
[master 306f422] Dramatisasi cerita dan perubahan nama file lagu.
warning: LF will be replaced by CRLF in lagu-intro-awal.txt.
The file will have its original line endings in your working directory.
 1 file changed, 1 insertion(+)
 create mode 100644 lagu-intro-awal.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
```

```
$ git log
commit 306f42258f4bfee95d10396777391ae013bc6edd
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 18:22:30 2012 +0700

    Dramatisasi cerita dan perubahan nama file lagu.

commit 03d06284462f7fc43b610d522678f4f22cdd9a40
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 18:08:10 2012 +0700

    Penambahan lagu intro.

commit 28dabb1c54a086cce567ecb890b10339416bcbfa
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:49:21 2012 +0700

    Penambahan misteri terbesar di dunia.

commit 61c47074ee583dbdd16fa9568019e80d864fb403
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:36:46 2012 +0700

    Kapitalisasi dan melengkapi kalimat.

commit 1d4cdc9350570230d352ef19aededf06769b0698
Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
Date:   Sun Dec 23 16:10:33 2012 +0700

    Inisialisasi repo. Penambahan cerita.txt.
```

Sedikit catatan tambahan untuk keluaran dari `git log`, baris `commit` yang berisi angka aneh (misalnya `1d4cdc9350570230d352ef19aededf06769b0698` untuk *commit* paling awal) merupakan nomor *commit* yang diberikan oleh `git` secara otomatis. Nomor ini memang tidak manusiawi, tetapi kita tidak perlu menuliskannya secara lengkap. Cukup hanya menuliskan enam karakter saja, `git` secara otomatis sudah dapat mengetahui nomor yang kita maksud. Contoh pengunaan akan ada pada bagian selanjtunya.

Membaca File Lama, dan Menjalankan Mesin Waktu

Nomor revisi, seperti yang telah dijelaskan sebelumnya, berguna sebagai tanda untuk memisahkan antara satu *commit* dengan *commit* lainnya. Misalnya jika kita ingin melihat isi file `cerita.txt` pada saat awal pertama kali dibuat, kita dapat menggunakan perintah `git show`, yang sintaksnya adalah:

```
git show [nomor revisi]:[nama file]
```

contoh pengunaan:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git show 1d4cdc:cerita.txt
ini adalah sebuah cerita
```

Perhatikan bahwa nomor commit yang dimasukkan hanyalah enam karakter saja. Jika keenam karakter tersebut sama untuk beberapa nomor *commit*, kita baru perlu memasukkan karakter selanjutnya, sampai tidak terdapat konflik nama lagi.

Terakhir, sebelum para pembaca pingsan kecapaian :D, kita dapat bergerak maju dan mundur dengan bebas pada setiap file, sesuai dengan nomor revisi dengan menggunakan `git checkcout` yang telah dijelaskan sebelumnya.

Contohnya, kita bergerak mundur ke masa lalu:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ ls
cerita.txt  lagu-intro-awal.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
dalam
 goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???!

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout 61c470 cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
dalam
 goa.
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout 1d4cdc cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
ini adalah sebuah cerita
```

dan kemudian maju kembali ke masa depan:

```
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout 03d0628 cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
dalam
 goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ git checkout HEAD cerita.txt

bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
$ cat cerita.txt
Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
dalam
 goa.

Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???!
```

Perhatikan bahwa pada saat menggunakan perintah `git checkout`, kita menggunakan `cat` untuk melihat isi file. Hal ini dikarenakan `git checkout` benar-benar mengubah file yang ada pada repositori, berbeda dengan `git show` yang hanya menampilkan file tersebut pada revisi tertentu.

Kesimpulan

Akhirnya bagian awal dari buku ini selesai juga! Seluruh perintah yang diberikan pada bagian ini sudah cukup untuk menggunakan `git` secara lokal, tanpa adanya kontributor. Pada bagian berikutnya akan diberikan langkah-langkah pengunaan `git` dengan kontributor, untuk melihat kemampuan penuh dari `git`. Untuk sekarang, jika masih bingung dengan perintah-perintah `git`, coba jalankan langkah-langkah berikut sebagai latihan:

1. Buat repositori baru
2. Tambahkan banyak file ke dalam repositori
3. Lakukan commit
4. Hapus beberapa file, edit beberapa file, dan tambahkan beberapa file baru.
5. Commit file tersebut
6. Atau lakukan pengembalian ke versi lama jika terjadi kesalahan
7. Lihat isi dari file lama, atau bahkan coba gunakan mesin waktu: pindahkan versi seluruh repositori atau beberapa file saja, dan lalu kembalikan file ke versi terbaru.

Selamat mencoba dan bereksperimen dengan `git`!