



# 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.

©2013 Alex Xandra Albert Sim

# **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

<b>Penggunaan Git Secara Mandiri</b> . . . . .	<b>1</b>
Dasar Kontrol Versi . . . . .	1
Instalasi Git . . . . .	2
Inisialisasi Repositori Git . . . . .	5
Penambahan File ke Repositori . . . . .	6
Mengubah Isi File . . . . .	9
Mengembalikan File ke Versi Lama . . . . .	12
Pengecekan Status Repositori . . . . .	13
Membaca File Lama, dan Menjalankan Mesin Waktu . . . . .	16
Kesimpulan . . . . .	18
<b>Berkolaborasi dengan Git</b> . . . . .	<b>19</b>
Kolaborasi Klasik: Sistem Terpusat . . . . .	19
Sistem Kolaborasi Terdistribusi . . . . .	20
Kolaborasi pada git . . . . .	21
Kesimpulan . . . . .	29
<b>Penggabungan Versi File (Merging)</b> . . . . .	<b>30</b>
Penggabungan Sederhana . . . . .	30
Penggabungan Lanjutan . . . . .	39
Kesimpulan . . . . .	47
<b>Percabangan (Branching) pada Repositori Git</b> . . . . .	<b>48</b>
Mengapa Bercabang? . . . . .	48
Dasar Percabangan . . . . .	49
Penggabungan Cabang . . . . .	55
Bekerja dengan Beberapa Cabang . . . . .	57
Penutup . . . . .	58
<b>Mengubah Teks Editor Standar pada Git</b> . . . . .	<b>59</b>

# Penggunaan Git Secara Mandiri

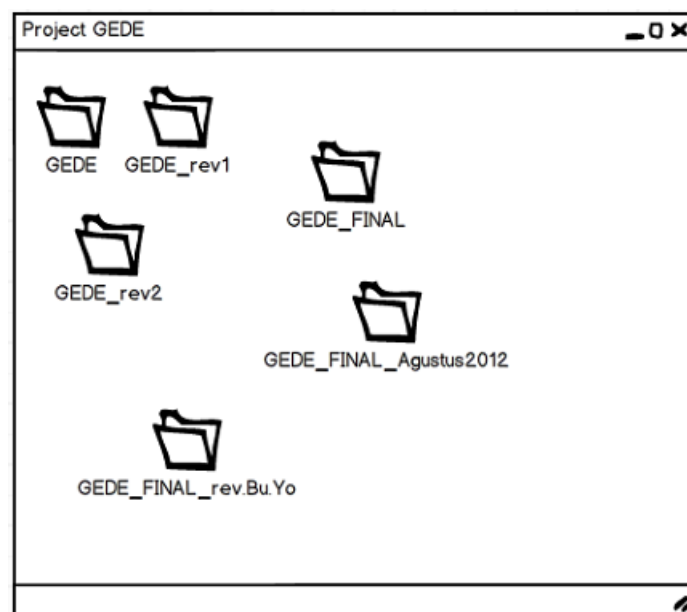
Dalam melakukan pemrograman, perubahan spesifikasi atau kebutuhan adalah hal yang tidak dapat dihindari. Tidak ada program yang dapat dituliskan dengan sempurna pada percobaan pertama. Hal ini menyebabkan pengembang perangkat lunak sangat dekat dengan sistem kontrol versi, baik secara manual maupun menggunakan 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:

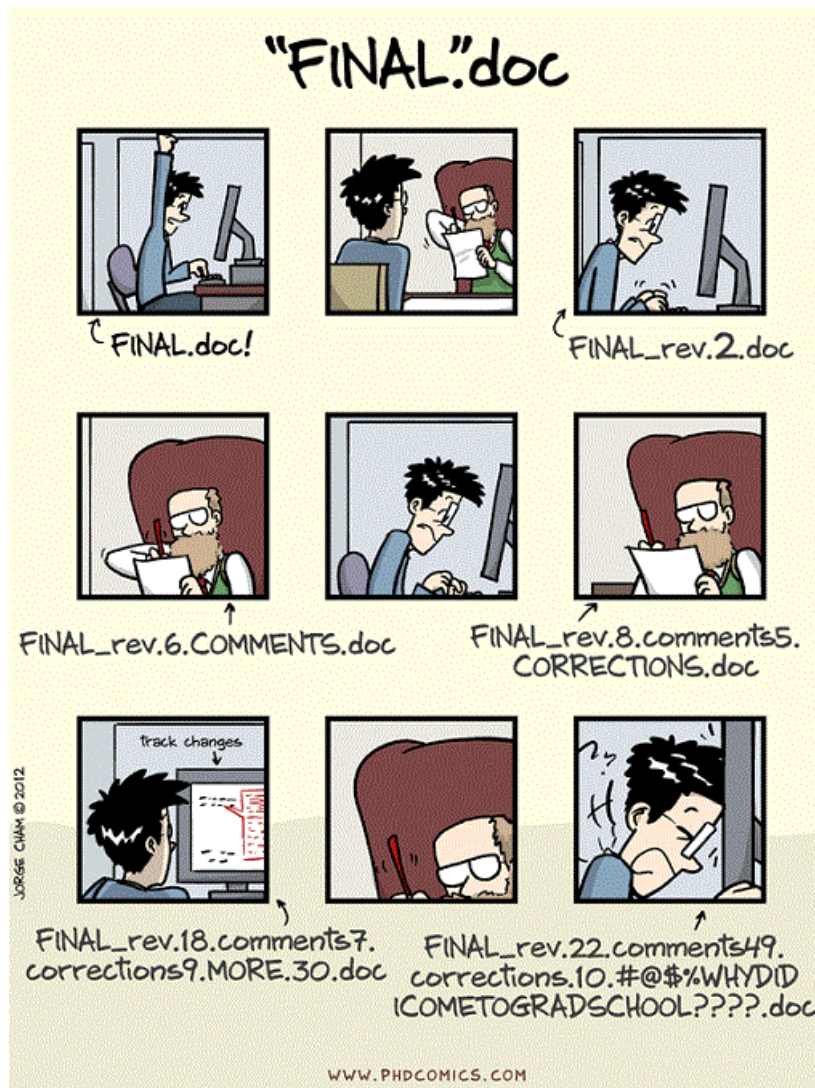
1. Menyimpan versi lama dari kode, maupun
2. Menggabungkan perubahan-perubahan kode dari versi lama (misal: untuk mengembalikan fitur yang telah dihapus) ataupun menggabungkan perubahan dari orang lain (misal: menggabungkan fitur yang dikembangkan oleh anggota tim lain).

Tanpa menggunakan sistem kontrol versi, yang sering saya temukan (dan dulunya saya gunakan, sebelum mengetahui tentang kontrol versi) ialah penggunaan 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 bazaar, dikembangkan untuk menyelesaikan masalah-masalah di atas. Karena tidak ingin membahas terlalu banyak, artikel ini hanya akan menjelaskan penggunaan git, karena kelihatannya git merupakan perangkat lunak kontrol versi yang paling populer untuk sekarang (mengingat popularitas [Github](https://github.com) dan penggunaan 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](https://git-scm.com) lalu lakukan download

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

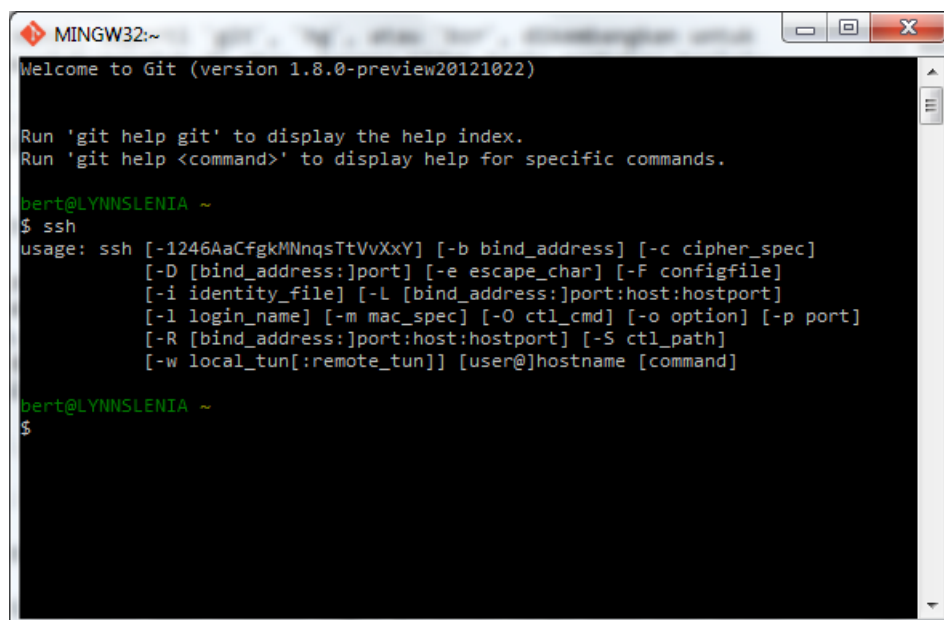
```
1 yum install git
```

pada repositori berbasis RPM, atau perintah

```
1 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](https://git-scm.com), karena pada paket yang tersedia di website tersebut telah diikuti 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):



```
MINGW32:~
Welcome to Git (version 1.8.0-preview20121022)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

bert@LYNNSLENIA ~
$ ssh
usage: ssh [-1246AaCfGkMMnqSttVvXxY] [-b bind_address] [-c cipher_spec]
          [-D [bind_address:]port] [-e escape_char] [-F configfile]
          [-i identity_file] [-L [bind_address:]port:host:hostport]
          [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
          [-R [bind_address:]port:host:hostport] [-S ctl_path]
          [-w local_tun[:remote_tun]] [user@]hostname [command]

bert@LYNNSLENIA ~
$
```

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:

```

1 bert@LYNNSLENIA ~
2 $ git
3 usage: git [--version] [--exec-path[=<path>]] [--html-path] [--man-path] [-\
4 -info-path]
5         [-p|--paginate|--no-pager] [--no-replace-objects] [--bare]
6         [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
7         [-c name=value] [--help]
8         <command> [<args>]
9
10 The most commonly used git commands are:
11     add      Add file contents to the index
12     bisect   Find by binary search the change that introduced a bug
13     branch   List, create, or delete branches
14     checkout Checkout a branch or paths to the working tree
15     clone    Clone a repository into a new directory
16     commit   Record changes to the repository
17     diff     Show changes between commits, commit and working tree, etc
18     fetch    Download objects and refs from another repository
19     grep     Print lines matching a pattern
20     init     Create an empty git repository or reinitialize an existing one
21     log      Show commit logs
22     merge    Join two or more development histories together
23     mv       Move or rename a file, a directory, or a symlink
24     pull     Fetch from and merge with another repository or a local branch
25     push     Update remote refs along with associated objects
26     rebase   Forward-port local commits to the updated upstream head
27     reset    Reset current HEAD to the specified state
28     rm       Remove files from the working tree and from the index
29     show     Show various types of objects
30     status   Show the working tree status
31     tag      Create, list, delete or verify a tag object signed with GPG
32
33 See 'git help <command>' for more information on a specific command.
34
35 bert@LYNNSLENIA ~
36 $

```

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:

```
1 bert@LYNNSLENIA ~
2 $ mkdir Desktop/projects/git-tutor
3
4 bert@LYNNSLENIA ~
5 $ cd Desktop/projects/git-tutor/
6
7 bert@LYNNSLENIA ~/Desktop/projects/git-tutor
8 $ ls
9
10 bert@LYNNSLENIA ~/Desktop/projects/git-tutor
11 $
```

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:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor
2 $ echo "ini adalah sebuah cerita" > cerita.txt
3
4 bert@LYNNSLENIA ~/Desktop/projects/git-tutor
5 $ ls
6 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:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor
2 $ git init
3 Initialized empty Git repository in c:/Users/bert/Desktop/projects/git-tuto\
4 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`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ ls -a
3 .  ..  .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`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git add .
3 warning: LF will be replaced by CRLF in cerita.txt.
4 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:

```
1 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:

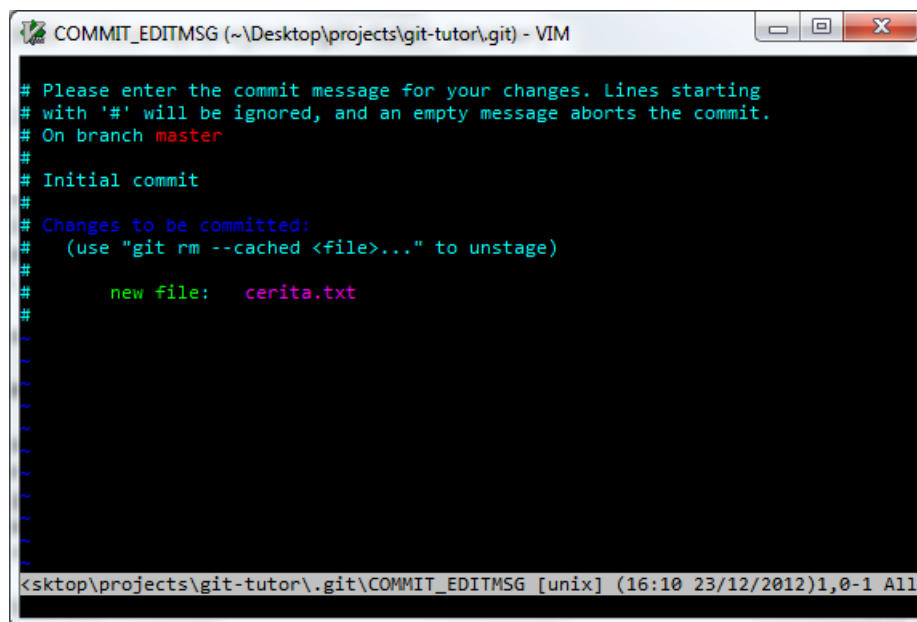
```
1 git add cerita.txt
```

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 perintah *commit* telah dijalankan. Sederhananya, memberikan perintah *commit* berarti berkata kepada git “Oi git, file yang tadi ditambahkan dan diubah itu dicatat ya. Masukin ke daftar sejarah.”

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

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ 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.

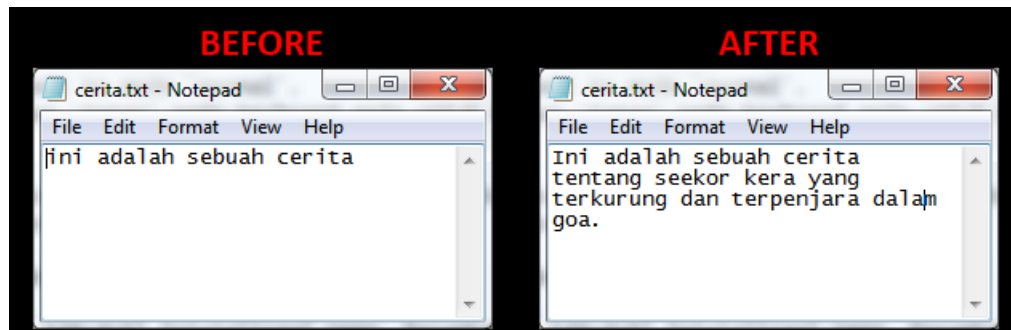
Jika langkah di atas diikuti dengan benar, maka kita akan dibawa kembali ke `git bash`, dengan pesan berikut:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git commit
3 [master (root-commit) 1d4cdc9] Inisialisasi repo. Penambahan cerita.txt.
4 warning: LF will be replaced by CRLF in cerita.txt.
5 The file will have its original line endings in your working directory.
6 1 file changed, 1 insertion(+)
7 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:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git commit
3 # On branch master
4 # Changes not staged for commit:
5 #   (use "git add <file>..." to update what will be committed)
6 #   (use "git checkout -- <file>..." to discard changes in working director\
7 y)
8 #
9 #       modified:   cerita.txt
10 #
11 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`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git commit -a
3 [master 61c4707] Kapitalisasi dan melengkapi kalimat.
4 1 file changed, 1 insertion(+), 1 deletion(-)
```



## git commit -a

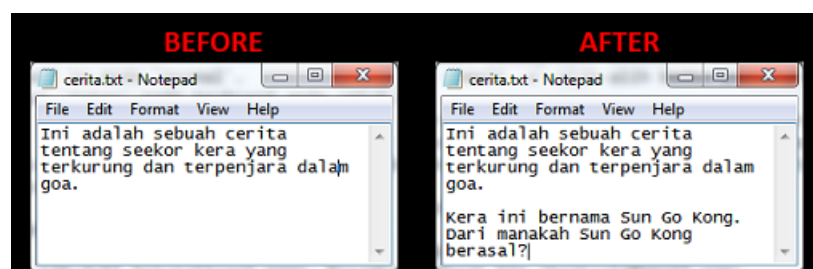
Perintah `git commit -a` ini adalah merupakan perintah singkat untuk memanggil `git add` dan `git commit` dalam satu perintah. Karena harus menjalankan perintah tersebut setiap kali melakukan modifikasi, maka kita dapat langsung menjalankan `git commit -a` alih-alih kedua perintah tersebut.

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`:

```

1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git log
3 commit 61c47074ee583dbdd16fa9568019e80d864fb403
4 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
5 Date:   Sun Dec 23 16:36:46 2012 +0700
6
7     Kapitalisasi dan melengkapi kalimat.
8
9 commit 1d4cdc9350570230d352ef19aededf06769b0698
10 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
11 Date:   Sun Dec 23 16:10:33 2012 +0700
12
13     Inisialisasi repo. Penambahan cerita.txt.
```

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



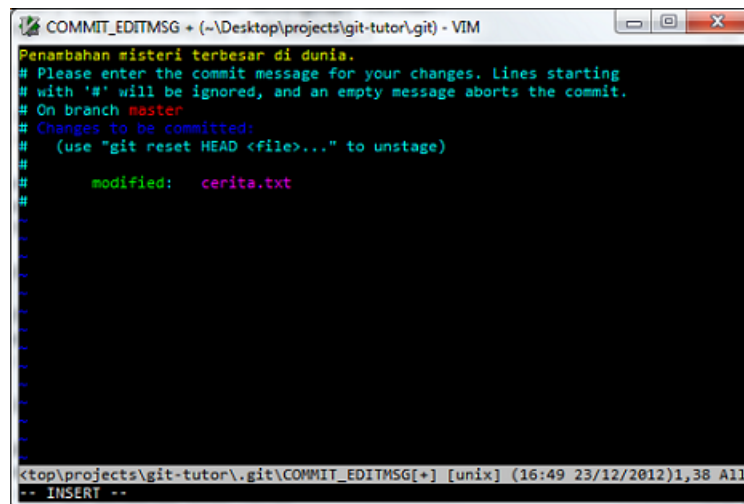
Kera sakti~ Tak pernah berhenti bertindak sesuka hati~

dan lakukan `commit` sekali lagi:

```

1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git commit -a
```

dengan pesan commit:



Ya. Paling besar.

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

```

1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git log
3 commit 28dabb1c54a086cce567ecb890b10339416bcbfa
4 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
5 Date: Sun Dec 23 16:49:21 2012 +0700
6
7     Penambahan misteri terbesar di dunia.
8
9 commit 61c47074ee583dbdd16fa9568019e80d864fb403
10 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
11 Date: Sun Dec 23 16:36:46 2012 +0700
12
13     Kapitalisasi dan melengkapi kalimat.
14
15 commit 1d4cdc9350570230d352ef19aededf06769b0698
16 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
17 Date: Sun Dec 23 16:10:33 2012 +0700
18
19     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!!!”

Sabar nak. Mari kita lihat kenapa.

## 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.

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ rm cerita.txt
3
4 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
5 $ ls
6
7 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
8 $
```

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:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git checkout HEAD -- cerita.txt
3
4 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
5 $ ls
6 cerita.txt
7
8 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
9 $ cat cerita.txt
10 Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
11 dalam goa.
12
13 Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
14 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
15 $
```



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. Penggunaan 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:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ ls
3 cerita.txt
4
5 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
6 $ echo "Seekor kera, terpuruk, terpenjara dalam goa. Di gunung suci sunyi
7 tempat hukuman para dewa." > lagu-intro.txt
8
9 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
10 $ ls
11 cerita.txt  lagu-intro.txt
12
13 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
14 $ git add .
15 warning: LF will be replaced by CRLF in lagu-intro.txt.
16 The file will have its original line endings in your working directory.
17
18 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
19 $ git commit -m "Penambahan lagu intro"
20 [master 03d0628] Penambahan lagu intro.
21 warning: LF will be replaced by CRLF in lagu-intro.txt.
22 The file will have its original line endings in your working directory.
23 1 file changed, 1 insertion(+)
24 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`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ ls
3 cerita.txt lagu-intro.txt
4
5 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
6 $ notepad cerita.txt
7
8 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
9 $ mv lagu-intro.txt lagu-intro-awal.txt
10
11 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
12 $ ls
13 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`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git status
3 # On branch master
4 # Changes not staged for commit:
5 #   (use "git add/rm <file>..." to update what will be committed)
6 #   (use "git checkout -- <file>..." to discard changes in working director\
7 y)
8 #
9 #       modified:   cerita.txt
10 #       deleted:    lagu-intro.txt
11 #
12 # Untracked files:
13 #   (use "git add <file>..." to include in what will be committed)
14 #
15 #       lagu-intro-awal.txt
16 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`:

```

1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git diff cerita.txt
3 diff --git a/cerita.txt b/cerita.txt
4 index 846114d..dbcb596 100644
5 --- a/cerita.txt
6 +++ b/cerita.txt
7 @@ -1,3 +1,3 @@
8  Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara\
9  dala
10
11 -Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
12 \ No newline at end of file
13 +Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal???
14 \ No newline at end of file
15 (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:

```

1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git add lagu-intro-awal.txt
3 warning: LF will be replaced by CRLF in lagu-intro-awal.txt.
4 The file will have its original line endings in your working directory.
5
6 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
7 $ git commit -m "Dramatisasi cerita dan perubahan nama file lagu."
8 [master 306f422] Dramatisasi cerita dan perubahan nama file lagu.
9 warning: LF will be replaced by CRLF in lagu-intro-awal.txt.
10 The file will have its original line endings in your working directory.
11 1 file changed, 1 insertion(+)
12 create mode 100644 lagu-intro-awal.txt
13
14 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)

```

```
15 $ git log
16 commit 306f42258f4bfee95d10396777391ae013bc6edd
17 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
18 Date: Sun Dec 23 18:22:30 2012 +0700
19
20     Dramatisasi cerita dan perubahan nama file lagu.
21
22 commit 03d06284462f7fc43b610d522678f4f22cdd9a40
23 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
24 Date: Sun Dec 23 18:08:10 2012 +0700
25
26     Penambahan lagu intro.
27
28 commit 28dabb1c54a086cce567ecb890b10339416bcbfa
29 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
30 Date: Sun Dec 23 16:49:21 2012 +0700
31
32     Penambahan misteri terbesar di dunia.
33
34 commit 61c47074ee583dbdd16fa9568019e80d864fb403
35 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
36 Date: Sun Dec 23 16:36:46 2012 +0700
37
38     Kapitalisasi dan melengkapi kalimat.
39
40 commit 1d4cdc9350570230d352ef19aededf06769b0698
41 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
42 Date: Sun Dec 23 16:10:33 2012 +0700
43
44     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 penggunaan akan ada pada bagian selanjutnya.

## 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:

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

contoh penggunaan:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git show 1d4cdc:cerita.txt
3 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 checkout` yang telah dijelaskan sebelumnya.

Contohnya, kita bergerak mundur ke masa lalu:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ ls
3 cerita.txt lagu-intro-awal.txt
4
5 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
6 $ cat cerita.txt
7 Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
8 dalam
9 goa.
10
11 Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal????
12
13 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
14 $ git checkout 61c470 cerita.txt
15
16 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
17 $ cat cerita.txt
18 Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
19 dalam
20 goa.
21 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
22 $ git checkout 1d4cdc cerita.txt
23
24 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
25 $ cat cerita.txt
26 ini adalah sebuah cerita
```

dan kemudian maju kembali ke masa depan:

```
1 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
2 $ git checkout 03d0628 cerita.txt
3
4 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
5 $ cat cerita.txt
6 Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
7 dalam
8 goa.
9
10 Kera ini bernama Sun Go Kong. Dari manakah Sun Go Kong berasal?
11 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
12 $ git checkout HEAD cerita.txt
13
14 bert@LYNNSLENIA ~/Desktop/projects/git-tutor (master)
15 $ cat cerita.txt
16 Ini adalah sebuah cerita tentang seekor kera yang terkurung dan terpenjara \
17 dalam
18 goa.
19
20 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 penggunaan `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.

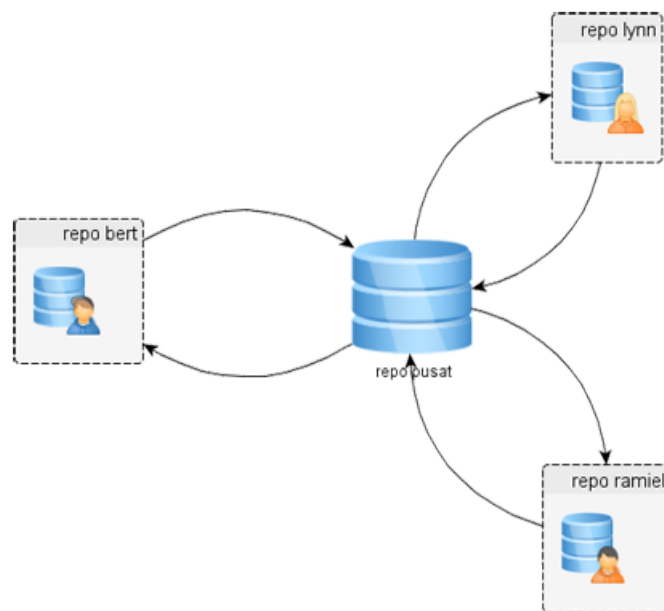
Selamat mencoba dan bereksperimen dengan `git`!

# Berkolaborasi dengan Git

Kegunaan utama sebuah sistem kontrol versi ialah untuk mengelola kode yang kita miliki, dari versi ke versi. Tapi tentunya kontrol versi tidak akan begitu berguna jika tidak dapat digunakan secara kolaborasi, yaitu mengelola kode yang digunakan oleh *tim*. Seluruh kontrol versi yang ada, mulai dari *cvs* sampai dengan *git* memiliki fitur untuk membantu kolaborasi. Apa saja fitur-fitur kolaborasi yang ditawarkan oleh *git*? Bagaimana perbedaan *git* dengan kontrol versi lainnya dalam hal ini? Mari kita lihat.

## Kolaborasi Klasik: Sistem Terpusat

Model kolaborasi yang ada sejak lama ialah model kolaborasi dengan sistem terpusat. Seperti namanya, dalam model kolaborasi ini kita memiliki sebuah sistem pusat, yang adalah sumber dari seluruh data yang ada. Dalam konteks pengembangan, dalam sistem pusat ini tersimpan kode yang menjadi acuan dari seluruh anggota tim. Ketika ingin menambahkan, menghapus, atau mengubah kode, anggota tim harus mengambil kode dari pusat terlebih dahulu, dan kemudian mengirimkan kode kembali ke pusat setelah perubahan dilakukan.



Model Kolaborasi Terpusat

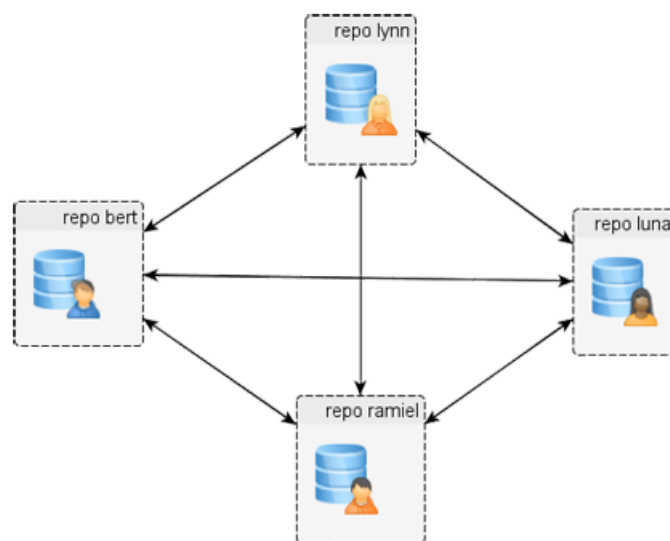
Model seperti ini merupakan model yang paling sederhana, sehingga mudah dimengerti dan banyak digunakan. Kontrol versi klasik seperti *svn* dan *cvs* biasanya menggunakan model kolaborasi seperti ini. Sayangnya, terdapat beberapa kekurangan dari model kolaborasi ini, misalnya:

1. **Bergantung kepada repositori pusat.** Bahkan untuk melakukan *commit* kita harus terkoneksi ke pusat terlebih dahulu. Kita otomatis tidak dapat bekerja dengan tenang ketika repositori tidak dapat diakses (dan percayalah, hal ini sering terjadi, entah karena Internet yang (tidak) anti-lelet atau *server* bermasalah).
2. **Satu titik pusat kegagalan.** Bayangkan jika suatu hari kantor anda diserang teroris, dan *server* pusat anda dibom. BAM. Seluruh sejarah revisi menghilang. Kode mungkin saja aman dan dapat diambil dari klien, tetapi seluruh revisi yang ada tidak lagi dapat dikembalikan.
3. **Branching dan merging SANGAT KOMPLEKS.** Bagian ini agak sulit dijelaskan tanpa merasakan langsung, tetapi kompleksitas proses *branching* dan *merging* pada *cvs* maupun *svn* telah melegenda. Tanyakan kepada tetua yang pernah menggunakan *svn* atau *cvs* apakah ada yang *senang* melakukan *branching* atau *merging* pada kedua sistem tersebut. Kalau ada yang senang, kemungkinan dia adalah masokis.

Kembali ke topik, *git* pada awalnya dikembangkan untuk digunakan pada pengembangan kernel Linux, yang memiliki sangat banyak kontributor. Fitur kolaborasi tentunya adalah hal yang sangat penting untuk dapat mengakomodasi kasus penggunaan tersebut. Karenanya, *git* dikembangkan dengan asumsi akan terdapat ribuan atau bahkan ratusan ribu kolabolator. Dan memang akhirnya *git* membuat revolusi pada sistem kolaborasi kontrol versi. Mempersembahkan, sistem kolaborasi terdistribusi.

## Sistem Kolaborasi Terdistribusi

Perbedaan utamanya dengan sistem terpusat? Tidak terdapat repositori pusat yang menjadi acuan (duh). Untuk mempermudah, bandingkan ilustrasi kolaborasi terdistribusi di bawah dengan ilustrasi model kolaborasi terpusat sebelumnya.



Model Kolaborasi Terdistribusi

Semua anggota tim melakukan pembaruan kode pada repositorinya sendiri, dan jika ingin menggabungkan kode dengan anggota tim lain, kita dapat langsung berkomunikasi dengan



anggota tim yang bersangkutan, dan kemudian melakukan *merging*. Kenapa melakukan *merging*? Bukankah tadi dikatakan bahwa proses *merging* sangat ribet? *Well*, inilah salah satu kelebihan *git*, proses *branching* dan *merging* dapat dilakukan dengan *sangat mudah*, terutama karena kedua proses ini adalah inti dari sistem kontrol versi terdistribusi (selanjutnya akan dirujuk sebagai *DVCS - Decentralized Version Control Sistem*).

Tapi, tapi, bukankah kita tetap perlu sebuah repositori pusat, untuk menjadi acuan dari program akhir yang dapat selalu berjalan?

– *Semua orang yang belum mengenal DVCS*

Pada dasarnya iya, di satu titik kita akan memerlukan satu repositori final yang menjadi acuan akhir. Pada DVCS, biasanya dilakukan salah satu dari dua hal berikut untuk masalah tersebut:

1. Repositori salah satu anggota (biasanya pemimpin tim) dijadikan sebagai acuan. Perangkat lunak yang dirilis merupakan hasil kompilasi dari repositori ini.
2. Terdapat satu repositori pusat, yang dapat diperbaharui oleh semua anggota tim. Repositori ini kemudian akan dikompilasi secara otomatis, dan terkadang juga menjalankan pengujian. Sistem seperti ini dikenal dengan nama *Continuous Integration (CI)*.

Terlepas dari manapun yang digunakan, pada akhirnya tetap saja DVCS memiliki model terdistribusi - kita dapat saling berbagi kode dengan anggota lainnya tanpa harus terkoneksi dengan repositori pusat. *Commit* juga dapat dilakukan tanpa perlu adanya repositori pusat (perhatikan bahwa pada [bagian sebelumnya](#) kita sama sekali tidak melakukan konfigurasi repositori pusat).

## Kolaborasi pada *git*

Daripada berteori terlalu banyak, kita akan langsung melakukan praktek berkolaborasi dengan *git*. Mari mulai membuat sebuah repositori baru:

```
1 bert@LYNNSLENIA ~
2 $ cd Desktop/
3
4 bert@LYNNSLENIA ~/Desktop
5 $ mkdir polijuice
6
7 bert@LYNNSLENIA ~/Desktop
8 $ cd polijuice/
9
10 bert@LYNNSLENIA ~/Desktop/polijuice
11 $ git init
12 Initialized empty Git repository in c:/Users/bert/Desktop/polijuice/.git/
```

Kemudian kita tambahkan file baru di dalam repositori. Misalkan kita ingin mencatat resep pembuatan [ramuan polijus](#). Buat file `polijus.txt` yang isinya adalah:

```

1  Bagian 1
2
3  1. Masukkan tiga takaran fluxweed ke dalam panci.
4  2. Tambahkan dua ikat knotgrass ke dalam panci.
5  3. Aduk tiga kali, searah jarum jam.
6  4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 60 menit.
7  5. Masukkan empat ekor lintah ke dalam panci.
8  6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk ke\
9  dalam panci.
10 7. Panaskan selama 30 detik, dalam suhu rendah.
11 8. Ayunkan tongkat untuk menyelesaikan ramuan.

```

dan kemudian lakukan penambahan file dan *commit*. Langkah opsional: buat ramuan polijus dan coba minum setelah selesai di langkah 8.

```

1  bert@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git add .
3
4  bert@LYNNSLENIA ~/Desktop/polijuice (master)
5  $ git commit -m "Penambahan resep polijus"
6  [master (root-commit) ecd0cac] Penambahan resep polijus
7  1 file changed, 10 insertions(+)
8  create mode 100644 polijus.txt

```

Selesai menyimpan, maka kita (selanjutnya disebut “Alex”) langsung mencoba membuat ramuan polijus, menggunakan langkah-langkah yang tertulis di atas. Hasilnya tidak menyenangkan. Kaki Alex menjadi gemuk, bercakar, dan berbulu putih, seperti kaki beruang kutub. Sayangnya, bagian lain dari tubuh tidak ikut berubah - bahkan tidak mengalami efek apa-apa (Alex tidak akan keberatan jika beurbah dapat menjadi beruang kutub selama satu jam)! Efek samping ini tentunya adalah tanda bahwa ada sesuatu yang salah pada ramuan tersebut. Alex kemudian meminta bantuan dari Ramiel, profesor ramuan terbaik di dunia. Agar Ramiel dapat melihat resep anda, Alex terlebih dahulu harus mempersiapkan repositori untuk dibagikan ke publik, dengan menggunakan perintah `git daemon`:

```

1  bert@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git daemon --reuseaddr --base-path=. --export-all --verbose
3  [8812] Ready to rumble

```

Setelah mempersiapkan repositori anda, Ramiel kemudian dapat mengambil data dari repositori anda melalui perintah `git clone`:

```

1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop
2 $ git clone git://127.0.0.1/ polijuice
3 Cloning into 'polijuice'...
4 remote: Counting objects: 3, done.
5 remote: Compressing objects: 100% (2/2), done.
6 remote: Total 3 (delta 0), reused 0 (delta 0)
7 Receiving objects: 100% (3/3), done.

```

Apa yang barusan terjadi? Ramiel melakukan pengambilan kode dari repositori anda, yang terletak pada `git://127.0.0.1/`, dengan nama proyek `polijuice`. Meskipun pada contoh ini kita menggunakan protokol `git`, pada dasarnya `git` juga mendukung protokol lain seperti `http`, `https`, maupun `ssh`. Dan untuk alasan keamanan, penggunaan protokol `https` atau `ssh` sangat disarankan. Tulisan ini menggunakan protokol `git` untuk menyederhanakan kasus, dan mempermudah contoh. Kembali ke topik, sintaks dari `git clone` adalah:

```

1 git clone [lokasi-repositori-atau-file-.git]

```

Perhatikan juga bahwa ketika ada yang mengambil kode, maka anda akan diberi notifikasi. *Log* pada terminal anda akan bertambah seperti berikut:

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git daemon --reuseaddr --base-path=. --export-all --verbose
3 [8812] Ready to rumble
4 [8004] Connection from 127.0.0.1:62777
5 [8004] Extended attributes (16 bytes) exist <host=127.0.0.1>
6 [8004] Request upload-pack for '/'

```

Setelah mengambil data, Ramiel terlebih dahulu memastikan bahwa data yang diambil adalah data yang benar:

```

1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop
2 $ cd polijuice/
3
4 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
5 $ ls
6 polijus.txt
7
8 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
9 $ cat polijus.txt
10 Bagian 1
11
12 1. Masukkan tiga takaran fluxweed ke dalam panci.
13 2. Tambahkan dua ikat knotgrass ke dalam panci.
14 3. Aduk tiga kali, searah jarum jam.
15 4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 60 menit.

```

```

16 5. Masukkan empat ekor lintah ke dalam panci.
17 6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk ke\
18    dalam panci.
19 7. Panaskan selama 30 detik, dalam suhu rendah.
20 8. Ayunkan tongkat untuk menyelesaikan ramuan.
21
22 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
23 $ git log
24 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
25 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
26 Date: Sat Dec 29 14:07:43 2012 +0700
27
28     Penambahan resep polijus

```

Perhatikan bahwa meskipun kita sedang berada pada akun Ramiel, data *commit* yang diberikan oleh `git log` tetap mencatat *commit* pertama dilakukan oleh bertzzie@gmail.com. Kembali ke topik, sebagai seorang profesor ramuan, Ramiel langsung menyadari bahwa terdapat kesalahan pada langkah keempat, dan bagian dua belum ditambahkan. Pertama-tama, ramiel melakukan perbaikan pada langkah keempat terlebih dahulu.

4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama ~~60 menit~~ 80 menit jika menggunakan panci timah campuran, 68 menit pada panci kuningan, dan 60 menit pada panci kaleng.

Perubahan pada langkah keempat

Berikut adalah langkah-langkah yang dilakukan:

```

1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2 $ vim polijus.txt
3
4 // lakukan perubahan...
5
6 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
7 $ git diff polijus.txt
8 diff --git a/polijus.txt b/polijus.txt
9 index d58baa9..a0fb16d 100644
10 --- a/polijus.txt
11 +++ b/polijus.txt
12 @@ -3,7 +3,7 @@ Bagian 1
13     1. Masukkan tiga takaran fluxweed ke dalam panci.
14     2. Tambahkan dua ikat knotgrass ke dalam panci.
15     3. Aduk tiga kali, searah jarum jam.
16 -4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 60 menit.
17 +4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 80 menit jika \
18 mengg

```

```

19 5. Masukkan empat ekor lintah ke dalam panci.
20 6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk k\
21 e dal
22 7. Panaskan selama 30 detik, dalam suhu rendah.
23
24 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
25 $ git commit -am "Perbaikan pada langkah 4"
26 [master a0ba939] Perbaikan pada langkah 4
27 1 file changed, 1 insertion(+), 1 deletion(-)

```

Langkah-langkah di atas dapat dilakukan terlepas apakah anda masih menjalankan perintah git daemon atau tidak. Bahkan jika komputer anda sama sekali tidak hidup (sehingga repositori tidak dapat diakses), langkah-langkah di atas tetap dapat dijalankan. Sekarang, mari kita lihat status revisi yang ada:

```

1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2 $ git log
3 commit a0ba93957a475bc54888e9b04a9b716c94cb3856
4 Author: Ramiel <ramiel@bertzzie.com>
5 Date: Sat Dec 29 14:38:40 2012 +0700
6
7     Perbaikan pada langkah 4
8
9 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
10 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
11 Date: Sat Dec 29 14:07:43 2012 +0700
12
13     Penambahan resep polijus

```

git telah berhasil mencatat perubahan tersebut dengan benar. Perhatikan juga bagaimana kita dapat melihat perbedaan repositori Ramiel dengan repositori Alex ketika menjalankan perintah git status:

```

1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2 $ git status
3 # On branch master
4 # Your branch is ahead of 'origin/master' by 1 commit.
5 #
6 nothing to commit, working directory clean

```

Pesan “*Your branch is ahead of ‘origin/master’ by 1 commit.*” memberitahukan bahwa kode pada repositori Ramiel berada 1 *commit* di depan repositori *origin/master*. Dari mana *origin/master* berasal? Apa artinya?

Setiap kali kita melakukan git clone terhadap sebuah repositori, git akan mencatat asal repositori kita, dan memberi nama repositori tersebut *origin*. *master* merupakan nama cabang tempat

kita bekerja sekarang, dan paralelnya pada origin. Hal ini dilakukan untuk mempermudah kita merujuk ke repositori asal. Jika ingin melakukan pengiriman kode misalnya, kita dapat langsung menuliskan `git push origin master`. Penjelasan mengenai percabangan (master dalam kasus ini) akan dilakukan pada artikel-artikel berikutnya.

Ramiel muncul sebagai nama penulis. Sampai titik ini, jika Ramiel ingin memberitahukan mengenai perubahan tersebut, Ramiel dapat mengirimkan pesan ke Alex. Untuk mempermudah, git menyediakan fasilitas untuk menghasilkan pesan yang ingin dikirimkan, melalui perintah `git request-pull`. Tentunya sebelum melakukan `request-pull`, repositori Ramiel harus siap terlebih dahulu:

```

1  ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2  $ git daemon --reuseaddr --base-path=. --export-all --verbose
3  [10980] Ready to rumble
4
5  ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
6  $ git request-pull -p origin/master git://localhost/
7  The following changes since commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe\
8  :
9
10     Penambahan resep polijus (2012-12-29 14:07:43 +0700)
11
12  are available in the git repository at:
13
14     git://localhost/ master
15
16  for you to fetch changes up to a0ba93957a475bc54888e9b04a9b716c94cb3856:
17
18     Perbaikan pada langkah 4 (2012-12-29 14:38:40 +0700)
19
20  -----
21  Ramiel (1):
22     Perbaikan pada langkah 4
23
24     polijus.txt | 2 +-
25     1 file changed, 1 insertion(+), 1 deletion(-)
26
27  diff --git a/polijus.txt b/polijus.txt
28  index d58baa9..a0fb16d 100644
29  --- a/polijus.txt
30  +++ b/polijus.txt
31  @@ -3,7 +3,7 @@ Bagian 1
32     1. Masukkan tiga takaran fluxweed ke dalam panci.
33     2. Tambahkan dua ikat knotgrass ke dalam panci.
34     3. Aduk tiga kali, searah jarum jam.
35  -4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 60 menit.
36  +4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 80 menit jika \

```

```

37 mengg
38 unakan panci timah campuran, 68 menit pada panci kuningan, dan 60 menit pad\
39 a pan
40 ci kaleng.
41 5. Masukkan empat ekor lintah ke dalam panci.
42 6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk k\
43 e dal am panci.
44 7. Panaskan selama 30 detik, dalam suhu rendah.

```

Keluaran dari git request-pull ini kemudian dapat dikirimkan ke Alex, baik melalui email ataupun metode lainnya. Keluaran ini memiliki beberapa informasi utama, yaitu:

1. *Commit* tambahan (pada pesan: "The following changes since commit ecd0ca...")
2. Tempat pengambilan perubahan (pada pesan: "are available in the git repository at:")
3. Perubahan yang terjadi (Setelah "-----")

Tetapi pesan ini tentunya akan menjadi terlalu panjang jika terdapat banyak perubahan, misalnya jika terdapat banyak file yang diubah. Kita dapat mempersingkat teks dengan membuang parameter -p, seperti berikut:

```

1  ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2  $ git request-pull origin/master git://localhost/
3  The following changes since commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe\
4  :
5
6  Penambahan resep polijus (2012-12-29 14:07:43 +0700)
7
8  are available in the git repository at:
9
10  git://localhost/ master
11
12  for you to fetch changes up to a0ba93957a475bc54888e9b04a9b716c94cb3856:
13
14  Perbaikan pada langkah 4 (2012-12-29 14:38:40 +0700)
15
16  -----
17  Ramiel (1):
18      Perbaikan pada langkah 4
19
20  polijus.txt | 2 +-
21  1 file changed, 1 insertion(+), 1 deletion(-)

```

Setelah mendapatkan pesan tersebut, Alex kemudian dapat mengambil perubahan yang dibuat oleh Ramiel dengan perintah git pull, sambil memberikan alamat yang ada:

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git pull git://localhost/ master
3 remote: Counting objects: 5, done.
4 remote: Compressing objects: 100% (2/2), done.
5 remote: Total 3 (delta 1), reused 0 (delta 0)
6 Unpacking objects: 100% (3/3), done.
7 From git://localhost
8  * branch          master      -> FETCH_HEAD
9 Updating ecd0cac..a0ba939
10 Fast-forward
11  polijus.txt | 2 +-
12  1 file changed, 1 insertion(+), 1 deletion(-)
13
14 bert@LYNNSLENIA ~/Desktop/polijuice (master)
15 $ git log
16 commit a0ba93957a475bc54888e9b04a9b716c94cb3856
17 Author: Ramiel <ramiel@bertzzie.com>
18 Date:   Sat Dec 29 14:38:40 2012 +0700
19
20     Perbaikan pada langkah 4
21
22 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
23 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
24 Date:   Sat Dec 29 14:07:43 2012 +0700
25
26     Penambahan resep polijus

```

Perhatikan juga bahwa status revisi pada repositori Alex telah berubah, sama dengan yang ada pada repositori Ramiel. Satu hal yang tersisa adalah, repositori Ramiel belum mengetahui bahwa perubahan yang dikirimkannya telah diterima oleh repositori Alex. Hal ini dapat diketahui dengan menjalankan `git status` pada repositori Ramiel:

```

1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2 $ git status
3 # On branch master
4 # Your branch is ahead of 'origin/master' by 1 commit.
5 #
6 nothing to commit, working directory clean

```

Perhatikan bahwa repositori Ramiel masih mengira ia berada satu *commit* di depan *origin/master*. Pada kenyataannya, *origin/master* telah memiliki revisi yang sama dengan repositori Ramiel. Idealnya, Ramiel harus melakukan `git pull` lagi terhadap repositori Alex, untuk melakukan sinkronasi kode (meskipun langkah ini tidak harus dilakukan setiap waktu, `git` cukup mampu untuk melakukan *merging* ketika terjadi masalah seperti ini). Karena sedang memberikan contoh, maka kita akan langsung melakukan `git pull` sekarang:



```
1 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
2 $ git pull origin
3 From git://127.0.0.1
4   ecd0cac..a0ba939  master    -> origin/master
5 Already up-to-date.
6
7 ramiel@LYNNSLENIA ~/Users/ramiel/Desktop/polijuice (master)
8 $ git status
9 # On branch master
10 nothing to commit, working directory clean
```

Dan repositori Ramiel sekarang telah tersinkronasi dengan baik.

## Kesimpulan

Sampai titik ini kita telah mempelajari bagaimana git bekerja secara terdistribusi, meskipun masih terdapat beberapa pertanyaan yang belum terjawab, misalnya: bagaimana jika Ramiel dan Alex melakukan perubahan pada file yang sama, dalam *commit* yang sama? Apa yang akan terjadi kalau Ramiel menambahkan file baru dalam satu *commit*, sementara Alex mengubah sebuah file? Resep bagian dua dari ramuan polijus itu apa? Apakah Alex berhasil kembali menjadi manusia seutuhnya?

Pertanyaan-pertanyaan di atas akan kita jawab pada bagian berikutnya. Sementara ini, mari kita lihat kembali alur kerja yang umumnya kita gunakan ketika berkolaborasi menggunakan git:

1. Bagikan repositori dengan perintah `git daemon`.
2. Jika sudah lama tidak dilakukan, ambil terlebih dahulu kode dari repositori anggota tim dengan perintah `git pull`.
3. Lakukan pekerjaan anda. Ubah file, tambahkan, kurangi, dll.
4. Lakukan *commit* secara lokal.
5. Jalankan langkah 3-4 sampai pekerjaan selesai.
6. Ketika sudah selesai melakukan pekerjaan, minta anggota lain mengambil perubahan yang anda lakukan dengan perintah `git request-pull`.
7. Sinkronasikan repositori anda dengan repositori anggota tim menggunakan perintah `git pull`.

Tidak usah takut jika langkah-langkah di atas terlihat tidak efisien dan rumit. Hal ini hanya dikarenakan oleh pembahasan kita yang belum lengkap. Pada akhir seluruh seri tulisan ini, akan terdapat langkah-langkah yang lebih sederhana dan efisien dalam berkolaborasi. Untuk sementara, setelah membaca tulisan ini anda seharusnya dapat melakukan hal-hal berikut:

1. Mempersiapkan repositori anda untuk berbagi kode dengan anggota tim lain.
2. Mengambil kode dari repositori lain.
3. Mengirimkan *pull request*

Selamat bersenang-senang dengan fitur kolaborasi dari git!

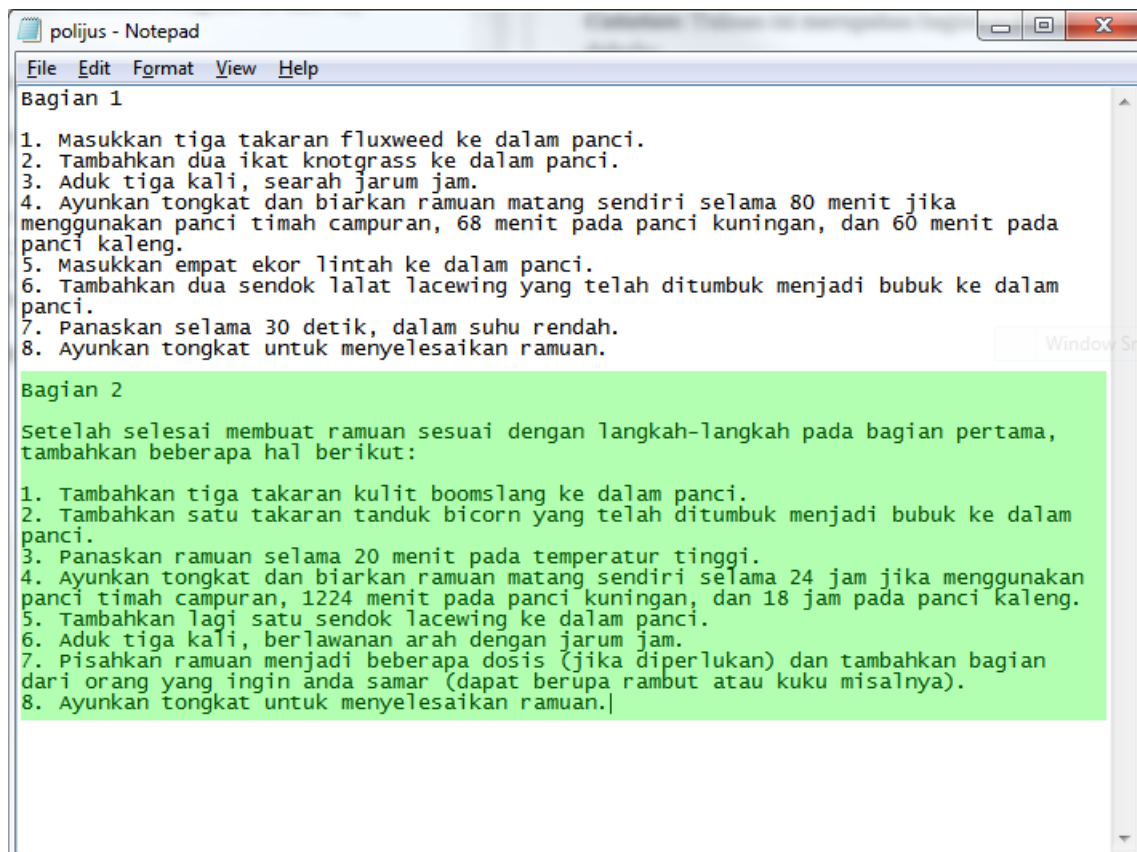
# Penggabungan Versi File (Merging)

Bagian penting dalam penggunaan kontrol versi adalah kolaborasi, yaitu bekerja sama dengan banyak orang untuk mengembangkan sistem dalam satu kode yang sama. Tulisan ini akan menjelaskan mengenai bagaimana mengatasi konflik yang dapat terjadi ketika kita memiliki banyak kontributor untuk kode yang sama. Konflik yang mungkin terjadi misalnya: beberapa orang melakukan perubahan pada kode yang sama, ataupun penambahan kode pada bagian yang berbeda.

Untuk ilustrasi, kita akan melanjutkan kisah penulisan ramuan polijus yang masih menggantung dari bagian 2. Pada akhir bagian 2, Ramiel dan Alex telah berhasil menyelesaikan bagian pertama dari ramuan polijus, yang akan digunakan untuk mengembalikan Alex menjadi manusia seutuhnya. Pada tulisan ini, kita akan melihat bagaimana bagian 2 dari ramuan polijus dibuat.

## Penggabungan Sederhana

Selesai membuat resep bagian 1 polijus, Ramiel memutuskan untuk langsung menambahkan bagian 2 dari resep ramuan polijus.



Tambahan Bagian 2 dari Ramuan Polijus

Mari kita lihat perubahan yang telah dilakukan oleh Ramiel sejauh ini dengan menggunakan perintah `git diff`:

```

1  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git diff
3  diff --git a/polijus.txt b/polijus.txt
4  index a0fb16d..fccd214 100644
5  --- a/polijus.txt
6  +++ b/polijus.txt
7  @@ -8,3 +8,16 @@ Bagian 1
8   6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk k\
9   e dal
10  7. Panaskan selama 30 detik, dalam suhu rendah.
11  8. Ayunkan tongkat untuk menyelesaikan ramuan.
12  +
13  +Bagian 2
14  +
15  +Setelah selesai membuat ramuan sesuai dengan langkah-langkah pada bagian p\
16  ertam
17  +
18  +1. Tambahkan tiga takaran kulit boomsling ke dalam panci.
19  +2. Tambahkan satu takaran tanduk bicorn yang telah ditumbuk menjadi bubuk \
20  ke da
21  +3. Panaskan ramuan selama 20 menit pada temperatur tinggi.
22  +4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 24 jam jika me\
23  nggun
24  +5. Tambahkan lagi satu sendok lacewing ke dalam panci.
25  +6. Aduk tiga kali, berlawanan arah dengan jarum jam.
26  +7. Pisahkan ramuan menjadi beberapa dosis (jika diperlukan) dan tambahkan \
27  bagia
28  +8. Ayunkan tongkat untuk menyelesaikan ramuan.
29  \ No newline at end of file

```

Seperti biasa, setelah melakukan perubahan maka Ramiel harus melakukan *commit* terlebih dahulu:

```

1  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git commit -am "Penambahan bagian 2 ramuan"
3  [master ccdfeb8] Penambahan bagian 2 ramuan
4  1 file changed, 13 insertions(+)

```

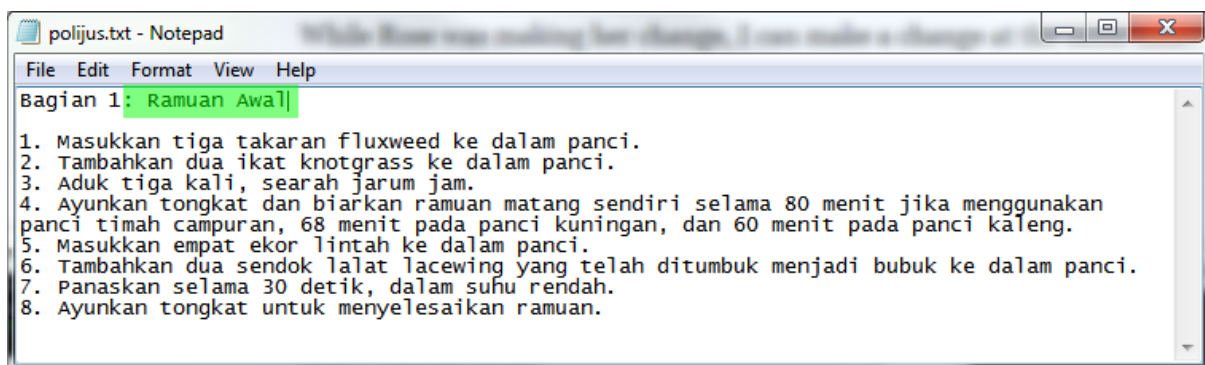
Dan selesai melakukan *commit*, kita dapat melihat catatan *commit* yang telah dilakukan sejauh ini:

```

1  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git log
3  commit ccdfeb8b7096aaf1e5dca90e664c3a41e689e757
4  Author: Ramiel <ramiel@bertzzie.com>
5  Date:   Sun Jan 6 14:10:17 2013 +0700
6
7      Penambahan bagian 2 ramuan
8
9  commit a0ba93957a475bc54888e9b04a9b716c94cb3856
10 Author: Ramiel <ramiel@bertzzie.com>
11 Date:   Sat Dec 29 14:38:40 2012 +0700
12
13     Perbaikan pada langkah 4
14
15 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
16 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
17 Date:   Sat Dec 29 14:07:43 2012 +0700
18
19     Penambahan resep polijus

```

Sementara Ramiel melakukan penambahan bagian dua pada ramuan, di saat yang bersamaan Alex juga melakukan penambahan pada teks dengan memberikan judul pada bagian pertama ramuan:



Penambahan Judul pada Bagian 1

```

1  bert@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git diff
3  diff --git a/polijus.txt b/polijus.txt
4  index a0fb16d..3d7adae 100644
5  --- a/polijus.txt
6  +++ b/polijus.txt
7  @@ -1,4 +1,4 @@
8  -Bagian 1
9  +Bagian 1: Ramuan Awal
10

```

- 11 1. Masukkan tiga takaran fluxweed ke dalam panci.
- 12 2. Tambahkan dua ikat knotgrass ke dalam panci.

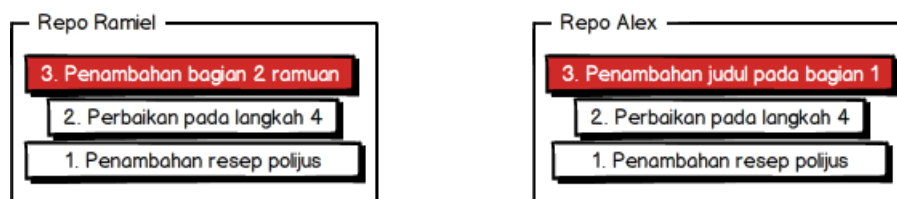
Yang pastinya dilanjutkan dengan *commit* untuk menyimpan perubahan kode pada repository Alex.

```
1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git commit -am "Penambahan judul pada bagian 1"
3 [master 8183a11] Penambahan judul pada bagian 1
4 1 file changed, 1 insertion(+), 1 deletion(-)
```

Sampai di titik ini repository Alex telah memiliki isi yang berbeda dengan repository Ramiel. Cek catatan *commit* untuk melihat perbedaannya:

```
1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git log
3 commit 8183a1159ad1e181b4d5065e4f4efe2b3f4dc5ff
4 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
5 Date: Sun Jan 6 14:16:31 2013 +0700
6
7     Penambahan judul pada bagian 1
8
9 commit a0ba93957a475bc54888e9b04a9b716c94cb3856
10 Author: Ramiel <ramiel@bertzzie.com>
11 Date: Sat Dec 29 14:38:40 2012 +0700
12
13     Perbaikan pada langkah 4
14
15 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
16 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
17 Date: Sat Dec 29 14:07:43 2012 +0700
18
19     Penambahan resep polijus
```

Untuk mempermudah pengertian, gambar berikut memperlihatkan perbedaan sejarah pada repository Ramiel dan Alex:



Perbedaan repo Ramiel dan Alex

Sampai titik ini, kita dapat bahwa pada *commit* ketiga, kedua repositori memiliki isi yang berbeda. Karena kedua perubahan memang diperlukan, tentunya kita tidak dapat langsung membuang salah satu perubahan tersebut. Yang akan kita lakukan ialah menggabungkan kedua *commit* tersebut, sehingga didapatkan kode terbaru yang adalah merupakan hasil gabungan dari kedua repositori tersebut. Penggabungan ini dikenal dengan nama *merging* pada DVCS.

Baik Alex maupun Ramiel dapat melakukan perubahan kode sebanyak yang mereka inginkan, tanpa mengetahui apa yang dilakukan oleh satu sama lain. Tentunya sampai di satu titik, salah satu dari mereka harus mengirimkan *pull request* agar perubahan yang dilakukan oleh kedua belah pihak dapat ditambahkan. Karena pada bagian sebelumnya kita telah menentukan repositori Alex sebagai pusat, maka Ramiel-lah yang harus melakukan penggabungan dari repositori Alex. Terlebih dahulu Ramiel harus menjalankan perintah `git status` untuk melihat seberapa jauh ia berbeda dari repositori Alex:

```
1 ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git status
3 # On branch master
4 # Your branch is ahead of 'origin/master' by 1 commit.
5 #
6 nothing to commit, working directory clean
```

Dapat dilihat bahwa git mencatat repositori Ramiel berada 1 *commit* di depan repositori Alex, tetapi perhatikan bahwa perubahan pada repositori Alex tidak dicatat oleh git! Lantas apa yang terjadi jika kita melakukan `git pull` sekarang? Mari kita coba jalankan.

## Perbedaan 1 Commit

Perbedaan 1 *commit* kita ketahui melalui kalimat `Your branch is ahead of 'origin/master' by 1 commit.` pada hasil eksekusi perintah `git status`.

```
1 ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git pull origin master
3 remote: Counting objects: 5, done.
4 remote: Compressing objects: 100% (2/2), done.
5 remote: Total 3 (delta 1), reused 0 (delta 0)
6 Unpacking objects: 100% (3/3), done.
7 From git://127.0.0.1
8  * branch                master      -> FETCH_HEAD
9 Auto-merging polijus.txt
10 Merge made by the 'recursive' strategy.
11  polijus.txt | 4 ++--
12  1 file changed, 2 insertions(+), 2 deletions(-)
13
```

```
14 ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
15 $ git log
16 commit d81ffc79e7a183472da436171471e81201a26c77
17 Merge: ccdfeb8 8183a11
18 Author: Ramiel <ramiel@bertzzie.com>
19 Date: Sun Jan 6 14:40:17 2013 +0700
20
21 Merge branch 'master' of git://127.0.0.1
22
23 commit 8183a1159ad1e181b4d5065e4f4efe2b3f4dc5ff
24 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
25 Date: Sun Jan 6 14:16:31 2013 +0700
26
27 Penambahan judul pada bagian 1
28
29 commit ccdfeb8b7096aaf1e5dca90e664c3a41e689e757
30 Author: Ramiel <ramiel@bertzzie.com>
31 Date: Sun Jan 6 14:10:17 2013 +0700
32
33 Penambahan bagian 2 ramuan
34
35 commit a0ba93957a475bc54888e9b04a9b716c94cb3856
36 Author: Ramiel <ramiel@bertzzie.com>
37 Date: Sat Dec 29 14:38:40 2012 +0700
38
39 Perbaikan pada langkah 4
40
41 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
42 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
43 Date: Sat Dec 29 14:07:43 2012 +0700
44
45 Penambahan resep polijus
```

Apa yang terjadi? Ketika kita melakukan `git pull`, secara otomatis git melakukan penggabungan dari perbedaan kode antara repositori Ramiel dan Alex. Untuk memastikan perubahan benar-benar terjadi, mari kita lihat isi dari file `polijus.txt` sekarang:

```
1  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ cat polijus.txt
3  Bagian 1: Ramuan Awal
4
5  1. Masukkan tiga takaran fluxweed ke dalam panci.
6  2. Tambahkan dua ikat knotgrass ke dalam panci.
7  3. Aduk tiga kali, searah jarum jam.
8  4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 80 menit jika m\
9  enggu
10 nakan panci timah campuran, 68 menit pada panci kuningan, dan 60 menit pada\
11 panc
12 i kaleng.
13 5. Masukkan empat ekor lintah ke dalam panci.
14 6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk ke\
15 dala
16 m panci.
17 7. Panaskan selama 30 detik, dalam suhu rendah.
18 8. Ayunkan tongkat untuk menyelesaikan ramuan.
19
20 Bagian 2
21
22 Setelah selesai membuat ramuan sesuai dengan langkah-langkah pada bagian pe\
23 rtama
24 , tambahkan beberapa hal berikut:
25
26 1. Tambahkan tiga takaran kulit boomsling ke dalam panci.
27 2. Tambahkan satu takaran tanduk bicorn yang telah ditumbuk menjadi bubuk k\
28 e dal
29 am panci.
30 3. Panaskan ramuan selama 20 menit pada temperatur tinggi.
31 4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 24 jam jika men\
32 gguna
33 kan panci timah campuran, 1224 menit pada panci kuningan, dan 18 jam pada p\
34 anci
35 kaleng.
36 5. Tambahkan lagi satu sendok lacewing ke dalam panci.
37 6. Aduk tiga kali, berlawanan arah dengan jarum jam.
38 7. Pisahkan ramuan menjadi beberapa dosis (jika diperlukan) dan tambahkan b\
39 agian
40 dari orang yang ingin anda samar (dapat berupa rambut atau kuku misalnya).
41 8. Ayunkan tongkat untuk menyelesaikan ramuan.
```

Ternyata benar bahwa penggabungan telah dilakukan secara otomatis. Tentunya setelah penggabungan ini, repositori Ramiel akan berada semakin jauh dari repositori Alex:



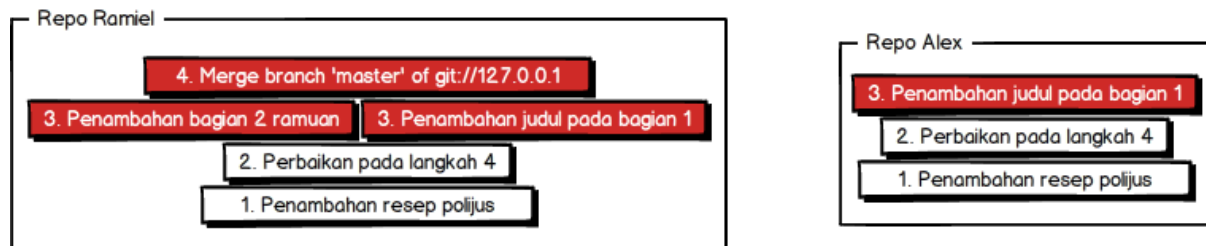
```

1 ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git status
3 # On branch master
4 # Your branch is ahead of 'origin/master' by 3 commits.
5 #
6 nothing to commit, working directory clean

```

Dari 1 *commit* di depan menjadi 3 *commit* di depan. Perbedaan 3 *commit* tersebut disebabkan karena ketertinggalan kita terhadap origin tidak penting, karena ketertinggalan tersebut dapat dengan mudah diselesaikan dengan *merging*. Yang diperlukan ialah seberapa banyak repositori origin (dalam kasus ini, repositori Alex) harus mengambil perubahan yang telah kita lakukan, karena hasil akhir kode akan diambil dari repositori origin.

Sampai di titik ini, kedua repositori akan berbentuk kira-kira seperti gambar berikut:



Repositori sebelum digabungkan ke pusat

dan tentunya Alex harus menggabungkan kembali repositori pusat dengan perubahan yang telah dilakukan oleh Ramiel. Untuk mempersingkat tulisan, langkah melakukan *pull request* akan kita lewatkan. Jika anda belum mengerti, silahkan baca mengenai *pull request* pada [bagian "Berkolaborasi dengan Git"](#).

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git pull git://localhost/ master
3 remote: Counting objects: 10, done.
4 remote: Compressing objects: 100% (4/4), done.
5 remote: Total 6 (delta 1), reused 0 (delta 0)
6 Unpacking objects: 100% (6/6), done.
7 From git://localhost
8  * branch          master      -> FETCH_HEAD
9 Updating 8183a11..d81ffc7
10 Fast-forward
11  polijus.txt | 13 ++++++++
12  1 file changed, 13 insertions(+)
13
14 bert@LYNNSLENIA ~/Desktop/polijuice (master)
15 $ git log
16 commit d81ffc79e7a183472da436171471e81201a26c77
17 Merge: ccdfeb8 8183a11
18 Author: Ramiel <ramiel@bertzzie.com>

```

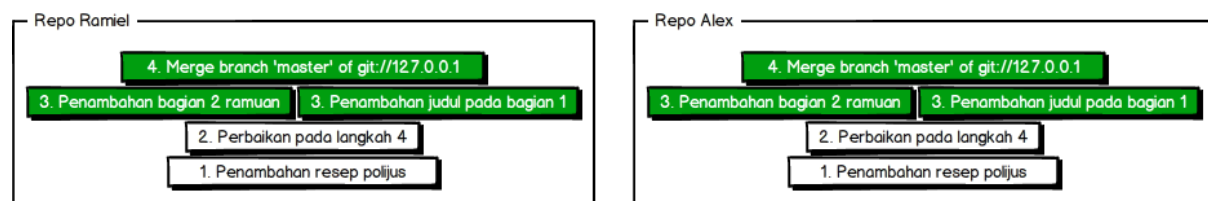
```

19 Date: Sun Jan 6 14:40:17 2013 +0700
20
21 Merge branch 'master' of git://127.0.0.1
22
23 commit 8183a1159ad1e181b4d5065e4f4efe2b3f4dc5ff
24 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
25 Date: Sun Jan 6 14:16:31 2013 +0700
26
27 Penambahan judul pada bagian 1
28
29 commit ccdfeb8b7096aaf1e5dca90e664c3a41e689e757
30 Author: Ramiel <ramiel@bertzzie.com>
31 Date: Sun Jan 6 14:10:17 2013 +0700
32
33 Penambahan bagian 2 ramuan
34
35 commit a0ba93957a475bc54888e9b04a9b716c94cb3856
36 Author: Ramiel <ramiel@bertzzie.com>
37 Date: Sat Dec 29 14:38:40 2012 +0700
38
39 Perbaikan pada langkah 4
40
41 commit ecd0cacf60d4e56f2e4ebd9de2776fc6ac7e3bbe
42 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
43 Date: Sat Dec 29 14:07:43 2012 +0700
44
45 Penambahan resep polijus
46
47 bert@LYNNSLENIA ~/Desktop/polijuice (master)
48 $ cat polijus.txt
49 Bagian 1: Ramuan Awal
50
51 1. Masukkan tiga takaran fluxweed ke dalam panci.
52 2. Tambahkan dua ikat knotgrass ke dalam panci.
53 3. Aduk tiga kali, searah jarum jam.
54 4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 80 menit jika m\
55 enggu
56 nakan panci timah campuran, 68 menit pada panci kuningan, dan 60 menit pada\
57 panc
58 i kaleng.
59 5. Masukkan empat ekor lintah ke dalam panci.
60 6. Tambahkan dua sendok lalat lacewing yang telah ditumbuk menjadi bubuk ke\
61 dala
62 m panci.
63 7. Panaskan selama 30 detik, dalam suhu rendah.
64 8. Ayunkan tongkat untuk menyelesaikan ramuan.

```

65  
 66 Bagian 2  
 67  
 68 Setelah selesai membuat ramuan sesuai dengan langkah-langkah pada bagian pe\  
 69 rtama  
 70 , tambahkan beberapa hal berikut:  
 71  
 72 1. Tambahkan tiga takaran kulit boomslang ke dalam panci.  
 73 2. Tambahkan satu takaran tanduk bicorn yang telah ditumbuk menjadi bubuk k\  
 74 e dal  
 75 am panci.  
 76 3. Panaskan ramuan selama 20 menit pada temperatur tinggi.  
 77 4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 24 jam jika men\  
 78 gguna  
 79 kan panci timah campuran, 1224 menit pada panci kuningan, dan 18 jam pada p\  
 80 anci  
 81 kaleng.  
 82 5. Tambahkan lagi satu sendok lacewing ke dalam panci.  
 83 6. Aduk tiga kali, berlawanan arah dengan jarum jam.  
 84 7. Pisahkan ramuan menjadi beberapa dosis (jika diperlukan) dan tambahkan b\  
 85 agian  
 86 dari orang yang ingin anda samar (dapat berupa rambut atau kuku misalnya).  
 87 8. Ayunkan tongkat untuk menyelesaikan ramuan.

Dapat dilihat bahwa ketika Alex melakukan *pull*, git lagi-lagi secara otomatis menjadikan repositori Alex menjadi repositori terbaru, dengan kode yang sama dengan Ramiel. Kedua repositori sampai pada titik ini telah menjadi sama, seperti yang dapat dilihat pada gambar.



Repositori akhirnya sama!

Sampai sekarang, perbedaan pada kode diselesaikan secara otomatis oleh git. Hal ini dikarenakan perubahan yang kita lakukan ialah perubahan yang mudah dan sederhana: Ramiel dan Alex melakukan perubahan pada bagian yang berbeda dari `polijus.txt`. Sayangnya, pada dunia nyata sekenario ini tidak selalu terjadi. Seringkali akan ada keadaan di mana perubahan akan dilakukan terhadap kode di tempat yang sama. Apa yang akan terjadi ketika kita menemui hal itu?

## Penggabungan Lanjutan

Karena penambahan judul pada bagian 1 telah dibuat, maka Ramiel pun berpikir untuk menambahkan judul pada bagian 2 juga, untuk melengkapi resep. Sebagai profesor yang bertanggung

jawab, Ramiel menuliskan judul yang serius untuk bagian 2.

#### Bagian 2: Bahan Tambahan

Setelah selesai membuat ramuan sesuai dengan langkah-langkah pada bagian pertama, tambahkan beberapa hal berikut:

1. Tambahkan tiga takaran kulit boomslang ke dalam panci.
2. Tambahkan satu takaran tanduk bicorn yang telah ditumbuk menjadi bubuk ke dalam panci.
3. Panaskan ramuan selama 20 menit pada temperatur tinggi.
4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 24 jam jika menggunakan panci timah campuran, 1224 menit pada panci kuningan, dan 18 jam pada panci kaleng.
5. Tambahkan lagi satu sendok lacewing ke dalam panci.
6. Aduk tiga kali, berlawanan arah dengan jarum jam.
7. Pisahkan ramuan menjadi beberapa dosis (jika diperlukan) dan tambahkan bagian dari orang yang ingin anda samar (dapat berupa rambut atau kuku misalnya).
8. Ayunkan tongkat untuk menyelesaikan ramuan.

#### Tambahan Judul Bagian 2 oleh Ramiel

```

1  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git commit -am "Tambahan judul bagian 2"
3  [master faad830] Tambahan judul bagian 2
4    1 file changed, 1 insertion(+), 1 deletion(-)
5
6  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
7  $ git log
8  commit faad8302e9a8d7ae7a9bc28dfd18ac45a358f252
9  Author: Ramiel <ramiel@bertzzie.com>
10 Date:   Sun Jan 6 16:11:54 2013 +0700
11
12     Tambahan judul bagian 2
13
14  commit d81ffc79e7a183472da436171471e81201a26c77
15  Merge: ccdfeb8 8183a11
16  Author: Ramiel <ramiel@bertzzie.com>
17  Date:   Sun Jan 6 14:40:17 2013 +0700
18
19     Merge branch 'master' of git://127.0.0.1
20
21  // dilewatkan untuk mempersingkat...
```

Sialnya, pada saat yang bersamaan Alex juga melakukan perubahan pada bagian yang sama: menambahkan judul pada bagian 2, dengan sedikit humor tentunya.

**Bagian 2: Anti-Beruang-Kutub**

Setelah selesai membuat ramuan sesuai dengan langkah-langkah pada bagian pertama, tambahkan beberapa hal berikut:

1. Tambahkan tiga takaran kulit boomslang ke dalam panci.
2. Tambahkan satu takaran tanduk bicorn yang telah ditumbuk menjadi bubuk ke dalam panci.
3. Panaskan ramuan selama 20 menit pada temperatur tinggi.
4. Ayunkan tongkat dan biarkan ramuan matang sendiri selama 24 jam jika menggunakan panci timah campuran, 1224 menit pada panci kuningan, dan 18 jam pada panci kaleng.
5. Tambahkan lagi satu sendok lacewing ke dalam panci.
6. Aduk tiga kali, berlawanan arah dengan jarum jam.
7. Pisahkan ramuan menjadi beberapa dosis (jika diperlukan) dan tambahkan bagian dari orang yang ingin anda samar (dapat berupa rambut atau kuku misalnya).
8. Ayunkan tongkat untuk menyelesaikan ramuan.

**Judul Bagian 2 oleh Alex**

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git commit -am "Judul bab 2 :D"
3 [master b018d56] Judul bab 2 :D
4 1 file changed, 1 insertion(+), 1 deletion(-)
5
6 bert@LYNNSLENIA ~/Desktop/polijuice (master)
7 $ git log
8 commit b018d568771961531386f37908ef4eb6e9b04082
9 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
10 Date: Sun Jan 6 16:18:58 2013 +0700
11
12 Judul bab 2 :D
13
14 commit d81ffc79e7a183472da436171471e81201a26c77
15 Merge: ccdfeb8 8183a11
16 Author: Ramiel <ramiel@bertzzie.com>
17 Date: Sun Jan 6 14:40:17 2013 +0700
18
19 Merge branch 'master' of git://127.0.0.1

```

Dan sekarang kita mendapatkan konflik. Perubahan dilakukan pada baris yang sama dalam resep! Bagaimana kita akan memperbaikinya? Misalkan dalam skenario ini Ramiel telah mengirimkan *pull request* kepada Alex, sehingga Alex akan mengambil perubahan dari Ramiel. Mari kita lihat apa yang terjadi.

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master)
2 $ git pull git://localhost/ master
3 remote: Counting objects: 5, done.
4 remote: Compressing objects: 100% (2/2), done.
5 remote: Total 3 (delta 1), reused 0 (delta 0)
6 Unpacking objects: 100% (3/3), done.
7 From git://localhost
8 * branch master -> FETCH_HEAD
9 Auto-merging polijus.txt

```

```

10 CONFLICT (content): Merge conflict in polijus.txt
11 Automatic merge failed; fix conflicts and then commit the result.
12
13 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
14 $ git status
15 # On branch master
16 # You have unmerged paths.
17 #   (fix conflicts and run "git commit")
18 #
19 # Unmerged paths:
20 #   (use "git add <file>..." to mark resolution)
21 #
22 #       both modified:       polijus.txt
23 #
24 no changes added to commit (use "git add" and/or "git commit -a")

```

Oops, terjadi konflik yang tidak dapat diselesaikan secara otomatis. Karena perubahan yang terjadi pada baris yang sama, tentunya git tidak dapat mengetahui baris mana yang harus digunakan oleh kode. Pada titik ini, kita dapat melihat perbedaan dari kedua perubahan dengan perintah `git diff`:

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
2 $ git diff polijus.txt
3 diff --cc polijus.txt
4 index 64fba2a,567db6f..0000000
5 --- a/polijus.txt
6 +++ b/polijus.txt
7 @@@ -9,7 -9,7 +9,11 @@@ Bagian 1: Ramuan Awa
8     7. Panaskan selama 30 detik, dalam suhu rendah.
9     8. Ayunkan tongkat untuk menyelesaikan ramuan.
10
11 ++<<<<<<< HEAD
12   +Bagian 2: Anti-Beruang-Kutub
13   ++=====
14   + Bagian 2: Bahan Tambahan
15   ++>>>>>>> faad8302e9a8d7ae7a9bc28dfd18ac45a358f252
16
17   Setelah selesai membuat ramuan sesuai dengan langkah-langkah pada bagian \
18   perta

```

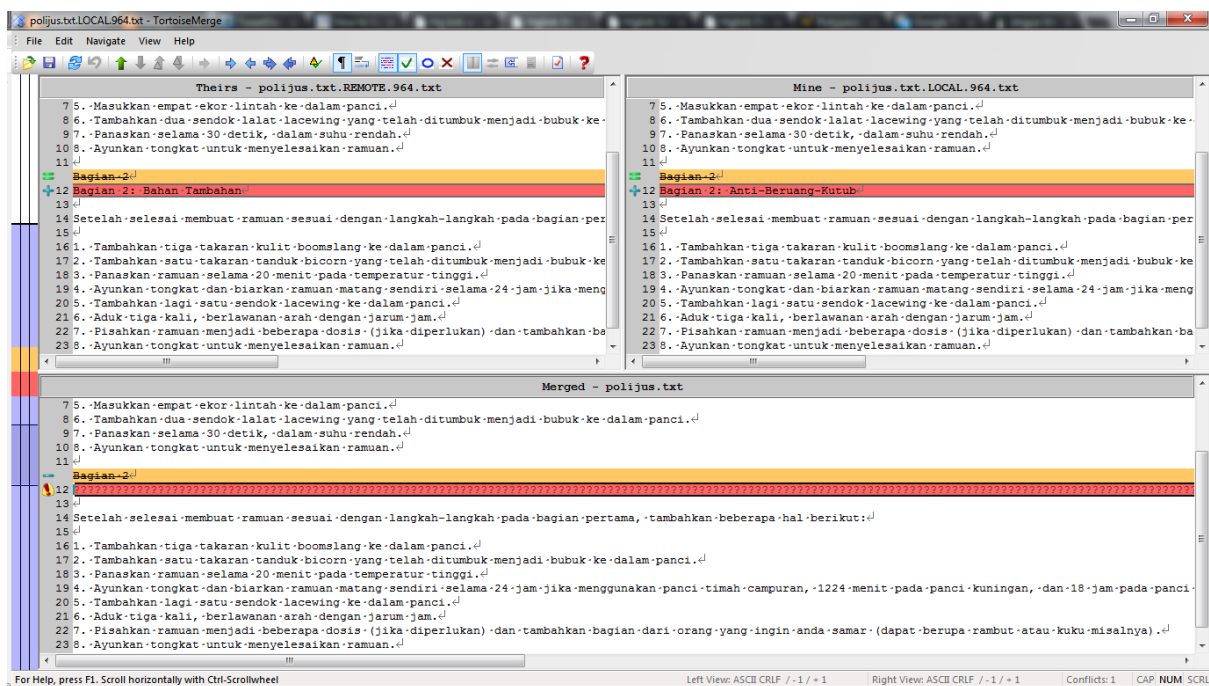
dapat dilihat bahwa perbedaan utama terdapat pada revisi HEAD dan faad83, dengan isi judul Bagian 2 yang berbeda. Untuk mengatasi konflik yang ada, kita dapat menggunakan perintah `git mergetool` seperti berikut:

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
2 $ git mergetool
3 merge tool candidates: tortoisemerge emerge vimdiff
4 Merging:
5 polijus.txt
6
7 Normal merge conflict for 'polijus.txt':
8   {local}: modified file
9   {remote}: modified file
10 Hit return to start merge resolution tool (tortoisemerge):

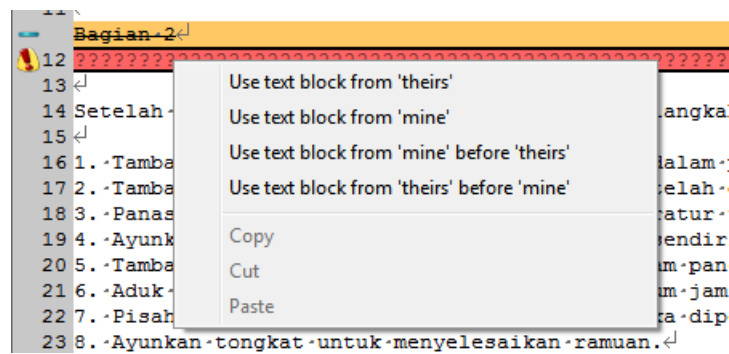
```

Perhatikan bahwa git memberikan kita pilihan *tool* yang dapat digunakan untuk menyelesaikan konflik. Pilihan *tool* ini akan dikeluarkan sesuai dengan apa yang ada dalam sistem. Karena dalam sistem Alex terdapat tortoisemerge, emerge, dan vimdiff, maka git hanya akan menampilkan ketiga *tool* tersebut. Kita akan menggunakan tortoisemerge (tidak ada pilihan khusus, hanya karena tortoisemerge adalah yang pertama kali dimunculkan). Setelah memilih tortoisemerge (dengan menekan Enter), maka kita akan dihadapkan dengan tampilan berikut:



### Antarmuka Tortoisemerge

Panel sebelah kiri atas memperlihatkan perubahan kode yang ada pada Ramiel, sedangkan panel kanan atas memperlihatkan perubahan yang ada pada Alex. Panel bawah menunjukkan bagian yang berkonflik (baris 12), yang ditandai dengan ????. Untuk menyelesaikan konflik, kita dapat melakukan klik kanan pada panel konflik untuk melihat pilihan-pilihan yang ada.

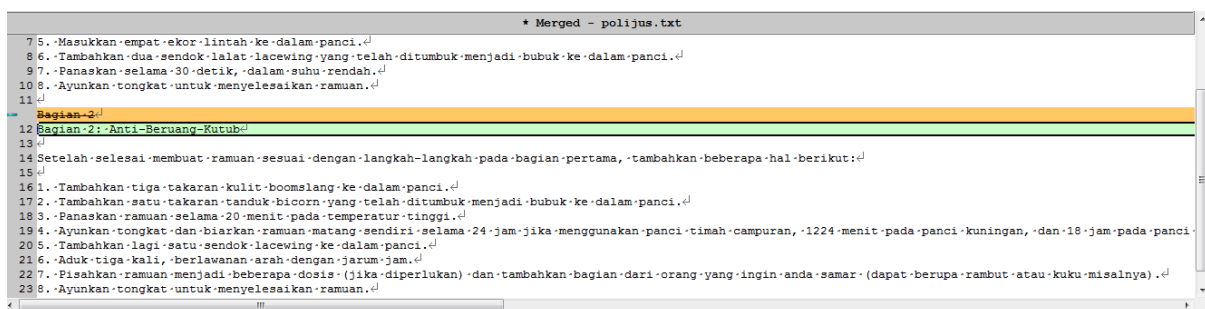


Pilihan aksi

Seperti yang dapat dilihat pada gambar, terdapat empat solusi utama, yang cukup jelas deskripsinya:

1. Menggunakan kode dari Ramiel (Use text block from 'theirs')
2. Menggunakan kode dari Alex (Use text block from 'mine')
3. Menggunakan keduanya, dengan urutan Alex -> Ramiel (Use block from 'mine' before 'theirs')
4. Menggunakan keduanya, dengan urutan Ramiel -> Alex (Use block from 'theirs' before 'mine')

Karena menganggap judul Ramiel membosankan, maka Alex memilih judulnya untuk digunakan. Setelah memilih pilihan kedua, maka tortoisemerge akan memperlihatkan perubahan akhir yang diinginkan:



TortoiseMerge merged

Lalu tentunya Alex dapat melakukan penyimpanan (Save) dan menutup aplikasi. Mari kita lihat lagi status dari repository setelah selsai melakukan *merge*:



```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
2 $ git status
3 # On branch master
4 # All conflicts fixed but you are still merging.
5 # (use "git commit" to conclude merge)
6 #
7 # Untracked files:
8 # (use "git add <file>..." to include in what will be committed)
9 #
10 #      polijus.txt.orig
11 nothing added to commit but untracked files present (use "git add" to track\
12 )

```

Perhatikan bahwa git mengetahui bahwa konflik sudah diselesaikan, tetapi menganggap kita belum selesai melakukan *merging* (All conflicts fixed but you are still merging.). Git juga secara otomatis menambahkan file `polijus.txt.orig` sebagai file sementara, yang aman dihapus jika kita yakin *merge* yang dilakukan tidak akan mengalami masalah.



## File .orig

File sementara berekstensi `.orig` tersebut dapat dihilangkan secara otomatis jika anda tidak ingin menggunakannya. Silahkan baca [dokumentasi git mergetool](#) untuk keterangan lebih lanjut.

Karena kita yakin bahwa penggabungan ini tidak akan menghasilkan masalah, maka kita akan menghapus file tersebut dan menambahkan commit baru untuk menggabungkan perubahan:

```

1 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
2 $ rm polijus.txt.orig
3
4 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
5 $ ls
6 polijus.txt
7
8 bert@LYNNSLENIA ~/Desktop/polijuice (master|MERGING)
9 $ git commit -m "Merged judul bab 2"
10 [master c3c653e] Merged judul bab 2
11
12 bert@LYNNSLENIA ~/Desktop/polijuice (master)
13 $ git status
14 # On branch master
15 nothing to commit, working directory clean
16

```

```

17 bert@LYNNSLENIA ~/Desktop/polijuice (master)
18 $ git log
19 commit c3c653eaf79583834fc41f46687291f4ebd0308b
20 Merge: b018d56 faad830
21 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
22 Date: Sun Jan 6 16:45:24 2013 +0700
23
24     Merged judul bab 2
25
26 commit b018d568771961531386f37908ef4eb6e9b04082
27 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
28 Date: Sun Jan 6 16:18:58 2013 +0700
29
30     Judul bab 2 :D
31
32 commit faad8302e9a8d7ae7a9bc28dfd18ac45a358f252
33 Author: Ramiel <ramiel@bertzzie.com>
34 Date: Sun Jan 6 16:11:54 2013 +0700
35
36     Tambahan judul bagian 2
37
38 // ...

```

Setelah penggabungan selesai, kita dapat melihat bahwa terdapat catatan baru pada git log yang ditampilkan (Merge: b018d56 faad830) yang menandakan penggabungan kedua *commit* yang kita lakukan. Selesai menggabungkan, tentunya kode sudah kanon dan Ramiel dapat melakukan penarikan kode terbaru lagi.

```

1  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
2  $ git pull origin
3  From git://127.0.0.1
4    a0ba939..c3c653e  master    -> origin/master
5  Already up-to-date.
6
7  ramiel@LYNNSLENIA ~/Desktop/polijuice (master)
8  $ git status
9  # On branch master
10 nothing to commit, working directory clean

```

Akhirnya kode Ramiel dan Alex telah sama, dan ramuan polijus dapat diselesaikan. Ramuan kemudian dibuat dengan benar sekali lagi oleh Alex, *and they live happily ever after*. CUT -

Akhirnya kode Ramiel dan Alex telah sama, dan ramuan polijus dapat diselesaikan. Ramuan kemudian dibuat dengan benar sekali lagi oleh Alex, dan akhirnya ia kembali menjadi manusia seutuhnya.

## Kesimpulan

Pada akhir bagian ini kita telah mempelajari bagaimana menggabungkan perbedaan-perbedaan yang mungkin terjadi ketika menggunakan kontrol versi dalam tim. Perlu dicatat bahwa *merging* tidak harus dilakukan setiap saat. *Merging* dapat dilakukan kapanpun, dan jika anda sedang tidak ingin menyelesaikan konflik *merging*, anda dapat terus menerus bekerja secara lokal, sampai titik di mana penggabungan tersebut ingin anda lakukan.

Untuk bekerja secara lokal dengan efektif dan tidak banyak mengganggu kode utama yang digunakan, kita dapat memanfaatkan fasilitas *branching* dari git. Fasilitas ini akan kita bahas di bagian selanjutnya. Untuk sekarang, coba gunakan git dengan orang lain sesering mungkin, dan selesaikan konflik ketika anda menemukannya (atau bahkan anda dapat bereksperimen untuk membuat konflik dan mencoba menyelesaikannya).

# Percabangan (Branching) pada Repositori Git

Pernakah anda ingin mengimplementasikan satu bagian program, tapi tidak ingin kode utama yang sudah bekerja dengan baik sekarang terganggu? Masih tidak yakin bagaimana akan mengimplementasikan fungsi perhitungan gaji, tapi perlu dipakai sebagai fungsi *dummy* sementara ini? Ingin bereksperimen, tetapi takut “mengotori” kode yang ada sekarang? Apakah anda menjadi terpaksa memiliki dua direktori, satu untuk kode utama dan satu lagi untuk eksperimen? Sudah saatnya meninggalkan hari-hari itu kawan, karena kita akan melihat bagaimana git menangani hal tersebut: *branching* (percabangan).

## Mengapa Bercabang?

Seperti yang telah dijelaskan sebelumnya, percabangan membantu kita dalam melakukan eksperimen, tanpa harus takut merusak kode yang ada sekarang. Implementasi fitur baru yang belum jelas kebutuhannya, percobaan kode baru untuk optimasi, dan berbagai eksperimen lain yang umum dilakukan dapat kita jalankan dalam cabang baru, agar eksperimen tersebut tidak merusak kode utama kita.

Percabangan juga kerap kali digunakan untuk membuat kode baru, yang kemudian akan dikirimkan ke repositori utama. Pembuatan cabang baru untuk setiap pengembang akan memudahkan koordinasi, karena setiap pengembang dapat bekerja secara independen, tanpa harus terganggu oleh kode yang dikerjakan oleh pengembang lainnya. Dalam svn, metode utama untuk mengisolasi kode tiap pengembang adalah dengan percabangan (karena svn tidak terdistribusi seperti git). Meskipun ide ini tidak baru, tetapi git membuat percabangan mudah, cepat, dan menyenangkan (terutama ketika harus melakukan *merging* antar cabang), sehingga percabangan dapat dikatakan sebagai salah satu fitur inti dari git.

Untuk mempermudah pengertian, kita akan langsung menuliskan kode dan mencoba berbagai fitur yang ada dalam percabangan. Misalkan kita sedang berada di dalam sebuah proyek untuk membuat blog, dengan isi repositori sebagai berikut:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ ls
3 about.html  index.html  readme.md
4
5 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
6 $ git log
7 commit b921ec5bcffdb308636112dcfb78615b8a77403a
8 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
9 Date: Thu Jan 10 16:19:02 2013 +0700
```

```
10
11     About page.
12
13 commit 263ab58c58665f96ce771c5709bbfc5fa1d34e78
14 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
15 Date:   Thu Jan 10 16:17:59 2013 +0700
16
17     Inisiasi proyek
```

Dan sampai titik ini, kita masih tidak yakin apakah akan menambahkan post pertama terlebih dahulu, ataukah harus membuat sebuah file TODO sebagai daftar sederhana akan tulisan yang ingin kita tuliskan.

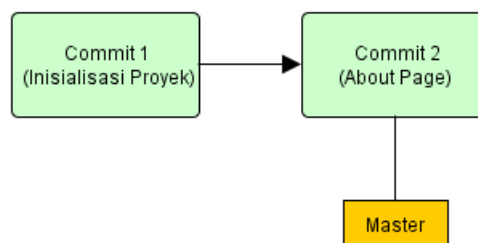
## Dasar Percabangan

### Membuat Cabang Baru

Karena masih belum yakin dengan apa yang harus dilakukan terlebih dahulu, kita mencoba membuat sebuah halaman TODO terlebih dahulu. Tetapi agar tidak membingungkan, kita tidak ingin TODO tersebut “mengotori” kode. Karenanya, kita akan membuat cabang baru untuk bekerja dengan file TODO tersebut. Sebelum mulai, kita akan melihat daftar cabang yang ada sekarang, menggunakan perintah `git branch`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git branch
3 * master
```

Jadi sekarang kita baru memiliki satu cabang, dengan nama “master”. Cabang ini merupakan cabang utama yang secara otomatis dibuat ketika kita menginisialisasi repositori git. Untuk sementara, repositori kita sekarang dapat diilustrasikan seperti berikut:



Repositori Awal

Seperti yang dapat dilihat, kita telah memiliki dua *commit*, dengan hanya satu cabang: master, yang berada pada *commit* terbaru. Sekarang, mari kita buat cabang baru (lagi-lagi) dengan perintah `git branch`:

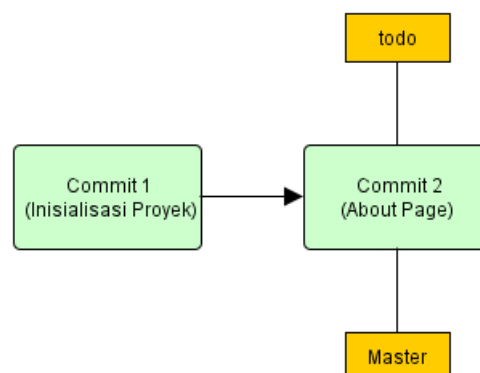
```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git branch todo
3
4 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
5 $ git branch
6 * master
7   todo
```

## Berpindah Cabang

Seperti yang dapat kita lihat, sekarang kita telah memiliki dua cabang, yaitu “master” dan “todo”. Cabang yang sedang aktif dapat dilihat melalui dua cara, yaitu pada teks (nama cabang) sebelum prompt (pada contoh di atas, (master)) ataupun dengan melihat tanda \* pada hasil perintah `git branch`. Mari kita coba berpindah ke cabang `todo`, untuk melihat tanda perubahan cabang. Perintah untuk berpindah cabang ialah `git checkout`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git checkout todo
3 Switched to branch 'todo'
4
5 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
6 $ git branch
7   master
8 * todo
```

Perhatikan bahwa (master) telah berubah menjadi (todo), dan tanda \* hasil respon perintah `git branch` telah berpindah ke samping `todo`. Jadi, sampai tahap ini kita telah memiliki dua cabang, “todo” dan “master”, yang masing-masing berada pada *commit* terbaru. Ilustrasinya:



Repositori dengan dua cabang

Sekarang mari kita coba bekerja pada cabang yang berbeda dan lihat apa yang terjadi.

## Bekerja pada Cabang Lain

Karena sekarang kita sedang berada di dalam cabang “todo”, kita dapat saja langsung melakukan hal yang kita mau. Segala hal yang kita lakukan pada cabang ini tidak akan mempengaruhi cabang “master”.

Misalnya, kita membuat sebuah file baru, bernama TODO, dengan langkah sebagai berikut:

1. Buat file baru bernama TODO:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ touch TODO
```

2. Cek apakah file berhasil dibuat:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ ls
3 TODO about.html index.html readme.md
```

3. Tambahkan TODO ke dalam repositori git:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git add TODO
```

4. Lakukan *commit*:

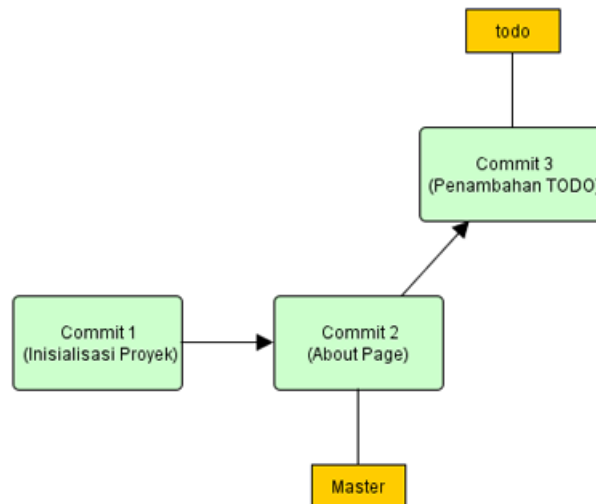
```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git commit -m "Penambahan file TODO"
3 [todo ff5650b] Penambahan file TODO
4 0 files changed
5 create mode 100644 TODO
```

Setelah menambahkan file tersebut, kita dapat berpindah ke cabang “master”, dan kita tidak akan melihat file TODO:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git checkout master
3 Switched to branch 'master'
4
5 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
6 $ ls
7 about.html index.html readme.md
```

Hal ini dikarenakan oleh struktur percabangan git yang unik: setiap cabang memiliki file tersendiri, dan kita tidak akan melihat isi dari cabang lain. Hal ini menyebabkan kita bebas melakukan eksperimen dalam satu cabang, dan kemudian membuang cabang tersebut jika memang hasil eksperimen tidak sesuai dengan keinginan kita.

Kembali ke topik, sekarang cabang repositori kita akan terlihat seperti berikut:



Repositori dengan dua cabang

Untuk memperjelas perbedaan antara kedua cabang tersebut, mari kita bekerja lebih jauh lagi dalam cabang “todo”, mengikuti langkah-langkah berikut:

1. Pindah ke cabang “todo”:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git checkout todo
3 Switched to branch 'todo'
```

2. Pastikan isi cabang benar:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ ls
3 TODO  about.html  index.html  readme.md
```

3. Lakukan perubahan pada file TODO:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ vim TODO
```

4. *Commit* perubahan tersebut:

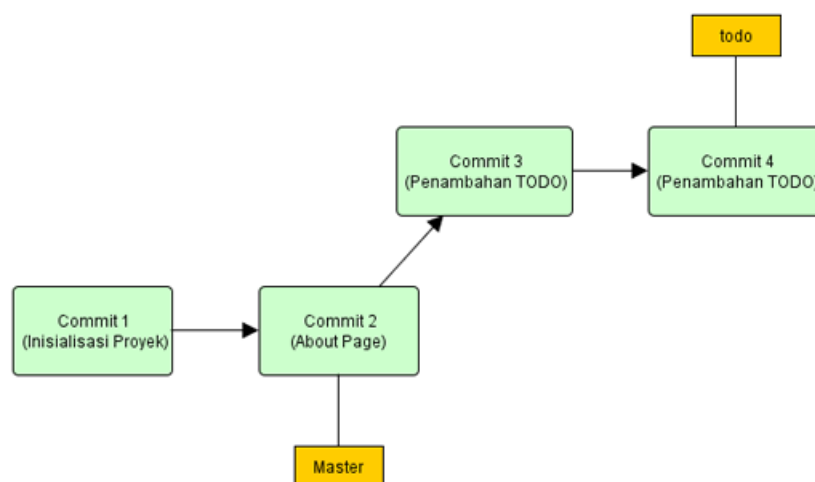
```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git commit -am "Menambahkan 3 TODO awal"
3 [todo 7be3730] Menambahkan 3 TODO awal
4 1 file changed, 5 insertions(+)
```

5. Cek daftar commit untuk melihat bahwa sejarah repositori telah benar:



```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git log
3 commit 7be37301d4188afe5275f12ca4acfe96683f43c2
4 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
5 Date: Thu Jan 10 16:51:55 2013 +0700
6
7     Menambahkan 3 TODO awal
8
9 commit ff5650bec0ecb95cccf1d079b9c6b3b076c4a3d8
10 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
11 Date: Thu Jan 10 16:43:37 2013 +0700
12
13     Penambahan file TODO
14
15 commit b921ec5bcffdb308636112dcfb78615b8a77403a
16 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
17 Date: Thu Jan 10 16:19:02 2013 +0700
18
19     About page.
20
21 commit 263ab58c58665f96ce771c5709bbfc5fa1d34e78
22 Author: Alex Xandra Albert Sim <bertzzie@gmail.com>
23 Date: Thu Jan 10 16:17:59 2013 +0700
24
25     Inisiasi proyek
```

Sekarang kita telah memiliki total sebanyak empat *commit* pada cabang “todo”, sementara cabang “master” masih berada pada *commit* kedua. Ilustrasinya adalah sebagai berikut:



Cabang Repositori setelah TODO dituliskan

Dan kemudian, kita akan menambahkan sebuah file baru pada cabang “master”, untuk memperjelas bagaimana kita dapat bekerja dalam cabang yang berbeda tanpa mengganggu cabang-

cabang lainnya. Mari tambahkan file pada cabang “master”, sesuai dengan langkah-langkah berikut:

1. Pindah ke cabang “master”:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git checkout master
3 Switched to branch 'master'
```

2. Cek untuk memastikan file pada cabang “master” benar:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ ls
3 about.html index.html readme.md
```

3. Buat sebuah file baru, `post1.html`, dan isikan file tersebut:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ vim post1.html
```

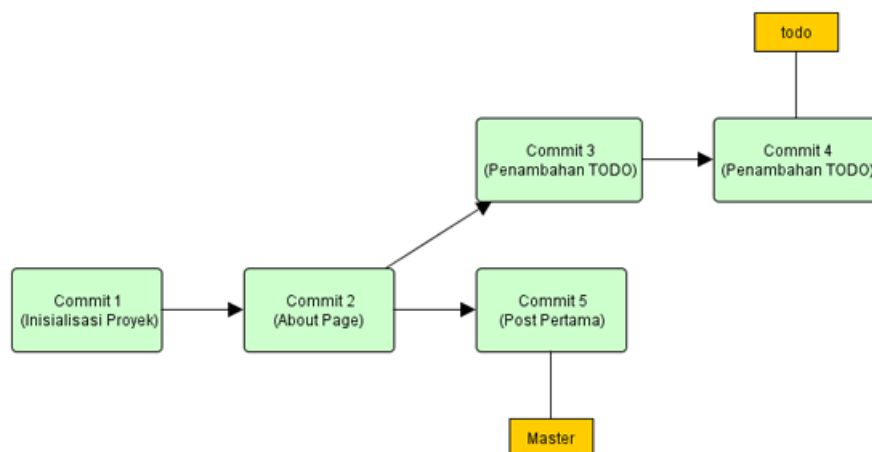
4. Tambahkan file baru ke repositori:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git add post1.html
```

5. *Commit*:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git commit -m "Tambahan post pertama"
3 [master f9ed764] Tambahan post pertama
4 1 file changed, 1 insertion(+)
5 create mode 100644 post1.html
```

Dan sekarang repositori kita telah resmi memiliki dua cabang yang memiliki isi yang berbeda:



Repositori *resmi* bercabang!

Lalu bagaimana kalau ternyata akhirnya yakin bahwa file TODO tersebut sangat berguna dan ingin memasukkannya ke kode utama kembali?

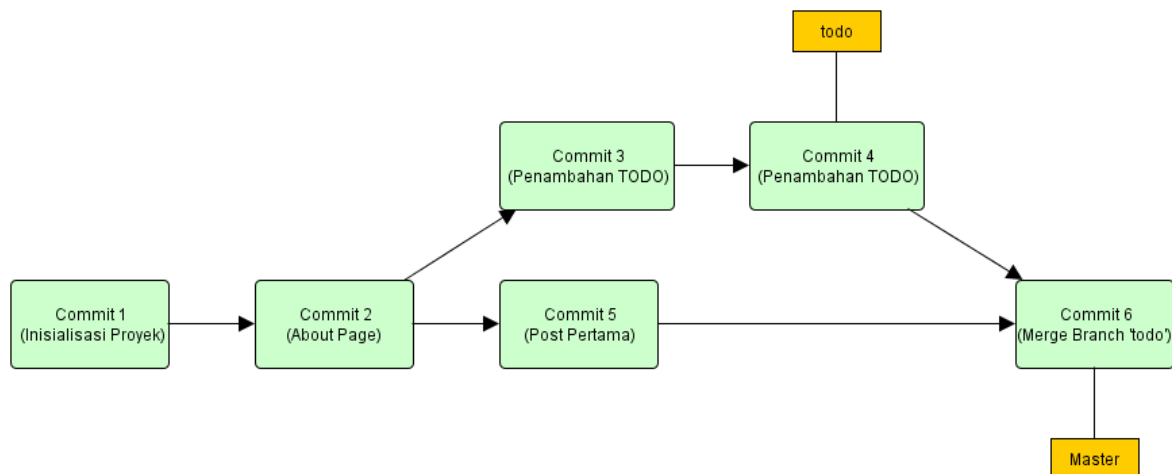
## Penggabungan Cabang

Penggabungan cabang, terutama jika tidak terjadi konflik, sangat mudah untuk dilakukan. Perintah `git merge` akan secara otomatis menggabungkan sebuah cabang ke cabang yang sedang aktif sekarang.

Mari kita coba perintah `git merge`:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git merge todo
3 Merge made by the 'recursive' strategy.
4  TODO | 5 +++++
5  1 file changed, 5 insertions(+)
6  create mode 100644 TODO
7
8 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
9 $ ls
10 TODO  about.html  index.html  post1.html  readme.md
```

Ilustrasi sederhana dari hasil penggabungan kedua cabang tersebut adalah sebagai berikut:



Repositori setelah digabungkan

Sampai pada titik ini, kita bahkan dapat tetap melakukan pekerjaan lanjutan pada cabang “todo”, dan kemudian menggabungkannya lagi ke cabang utama. Dengan tetap bekerja pada cabang “todo”, kita dapat memastikan segala perubahan, baik penambahan maupun pengurangan, yang dilakukan dapat dikembalikan dengan mudah, tanpa harus mempengaruhi *history* cabang utama. Jika terjadi hal-hal yang tidak diinginkan karena perubahan yang dilakukan kita dapat langsung dengan mudah berpindah ke cabang utama dan menghapus cabang “todo”. Sejarah dari cabang utama tidak akan terpengaruh.

Misalkan jika kita ingin menambahkan satu poin baru ke dalam file TODO, tetapi agar aman kita akan melakukannya pada cabang “todo” terlebih dahulu. Langkah-langkah yang harus diambil untuk melakukan hal tersebut yaitu:

## 1. Pindah ke cabang “todo”:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git checkout todo
3 Switched to branch 'todo'
```

## 2. Edit file TODO:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ vim TODO
```

3. Lakukan *commit*, dengan pesan *commit* yang jelas:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git commit -am "Penambahan satu pekerjaan, dan menandai pekerjaan pertam\
3 a sel
4 esai"
5 [todo ccdc1e4] Penambahan satu pekerjaan, dan menandai pekerjaan pertama s\
6 elesai
7
8 1 file changed, 2 insertions(+), 1 deletion(-)
```

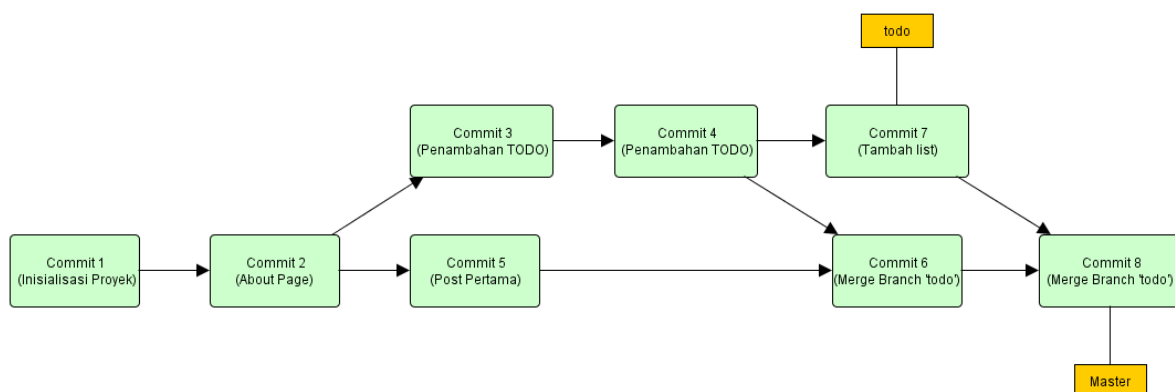
## 4. Pindah kembali ke cabang “master”:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (todo)
2 $ git checkout master
3 Switched to branch 'master'
```

## 5. Lakukan penggabungan:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git merge todo
3 Merge made by the 'recursive' strategy.
4 TODO | 3 ++-
5 1 file changed, 2 insertions(+), 1 deletion(-)
```

Pada akhir langkah, kita akan memiliki repositori kira-kira seperti berikut:



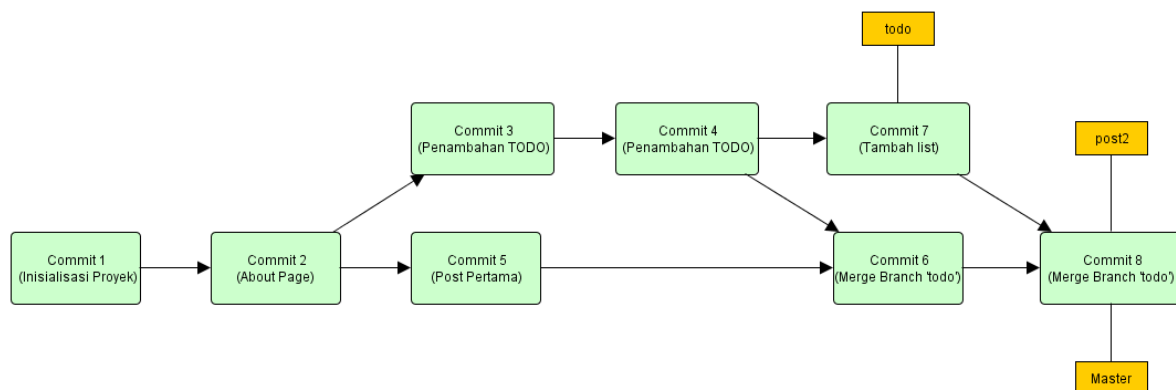
Repositori Akhir

## Bekerja dengan Beberapa Cabang

Prinsip-prinsip yang telah dijelaskan sebelumnya dapat diterapkan dengan cara yang sama untuk banyak cabang. Tidak terdapat batasan jumlah cabang yang dapat disimpan oleh git. Misalnya, kita dapat dengan mudah menambahkan cabang baru ke dalam git dengan perintah `git branch` lagi:

```
1 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
2 $ git branch post2
3
4 bert@LYNNSLENIA ~/Desktop/projects/blog (master)
5 $ git branch
6 * master
7   post2
8   todo
```

Dan sekarang kita memiliki total tiga cabang, seperti gambar berikut:



Repositori Bercabang Banyak

Tentunya, semua aturan mengenai cabang yang sebelumnya telah dibahas juga berlaku untuk cabang baru ini. Idealnya, setiap mengembangkan fitur baru akan lebih baik jika fitur baru tersebut dimasukkan ke dalam cabang baru.

## Konflik

Ketika kita bekerja menggunakan banyak cabang, tentunya konflik antar cabang tidak dapat dihindari. Resolusi konflik yang dilakukan oleh git menggunakan prinsip yang sama dengan teknik resolusi konflik pada beberapa repositori (jika dilihat dari sisi yang berbeda, setiap repositori dapat dikatakan adalah sebuah cabang baru). Metode penggabungan ini telah kita bahas pada [bagian “Penggabungan Lanjutan”](#).

## Penutup

Pada tulisan ini kita telah mempelajari bagaimana cara membuat cabang baru, bekerja di dalam cabang tersebut, berpindah-pindah antar cabang, dan menggabungkan kembali perubahan yang dibuat di cabang ke kode utama. Jika masih belum benar-benar mengerti mengenai percabangan pada git, coba jalankan perintah-perintah yang ada sebelumnya, untuk langsung “mengalami” percabangan dalam git.

Keempat bagian seri mini mengenai git ini seharusnya sudah cukup mengajarkan dasar-dasar git kepada anda, sehingga anda dapat langsung mulai menggunakan git. Jika ada pertanyaan, komentar, atau saran, silahkan tinggalkan pesan di kotak komentar pada bagian bawah tulisan. Akhir kata, *happy coding!*

# Mengubah Teks Editor Standar pada Git

Teks editor standar yang digunakan oleh git adalah vim, sebuah editor yang memerlukan cukup banyak waktu untuk dipelajari. Lampiran ini dibuat untuk membantu pembaca yang tidak terbiasa menggunakan vim untuk mengganti teks editor standar menjadi teks editor favorit pengguna tersebut.

Langsung saja, perintah yang digunakan adalah:

```
1 git config --global core.editor "[nama editor]"
```

Parameter `--global` ialah untuk mengubah lingkup perubahan. Ada tiga parameter yang dapat digunakan untuk merubah lingkup, yaitu:

1. Tanpa parameter: hanya mengubah editor standar pada repositori yang sedang aktif.
2. `--global`: mengubah editor standar untuk seluruh proyek, pada pengguna yang sedang aktif.
3. `--system`: mengubah editor standar untuk seluruh pengguna dalam sistem. Perintah dengan parameter ini akan memerlukan hak akses root atau admin.

Jadi, misalnya untuk sistem operasi **Windows** kita dapat menggunakan perintah berikut untuk mengubah editor standar menjadi notepad:

```
1 git config --global core.editor "notepad"
```

Pada **Linux** dan **Mac OS X** kita dapat mengubah editor standar menjadi nano (atau emacs, atau pico, atau apapun):

```
1 git config --global core.editor "nano"
```

Tentunya sangat disarankan untuk **tidak menggunakan notepad pada Windows**, karena keterbatasan notepad dalam memproses file teks berformat Unix. Gunakan teks editor lain, dan pastikan teks editor tersebut [dapat dijalankan dari command line](#), lalu ganti notepad pada kode di atas dengan file *executable* editor tersebut.