


Git Dasar

Eko Kurniawan Khannedy

Eko Kurniawan Khannedy

- Technical architect at one of the biggest ecommerce company in Indonesia
- 10+ years experiences
- www.programmerzamannow.com
- youtube.com/c/ProgrammerZamanNow



Eko Kurniawan Khannedy

- Telegram : [@khannedy](https://t.me/khannedy)
- Facebook : [fb.com/ProgrammerZamanNow](https://www.facebook.com/ProgrammerZamanNow)
- Instagram : [instagram.com/programmerzamannow](https://www.instagram.com/programmerzamannow)
- Youtube : [youtube.com/c/ProgrammerZamanNow](https://www.youtube.com/c/ProgrammerZamanNow)
- Telegram Channel : t.me/ProgrammerZamanNow
- Email : echo.khannedy@gmail.com

Agenda

- Pengenalan Version Control
- Pengenalan Git
- Repository
- The Three Tree
- Working Directory
- Staging Index
- Commit
- Reset Commit
- Dan lain-lain

Pengenalan Version Control

Sebelum Ada Version Control System

- Saat kita mengerjakan pekerjaan, kita sering sekali melakukan revisi. Misal saja kita membuat dokumen proposal atau skripsi
- Biasanya kita akan simpan dokumen pertama dengan nama document_1, setelah mendapat revisi, kita akan simpan dengan nama document_2, jika masih ada revisi, kita akan simpan dengan nama document_3, dan seterusnya
- Kenapa kita lakukan hal tersebut? Agar kita bisa mengetahui perubahan yang terjadi antar revisi document, dan jika sewaktu-waktu kita perlu menggunakan revisi yang sebelumnya, kita bisa menggunakannya dengan mudah

Version Control System

- Version Control adalah sebuah system yang merekam perubahan pada file dari waktu ke waktu, sehingga kita bisa melihat versi sebelumnya jika diinginkan
- Version Control sangat populer dikalangan programmer, karena programmer selalu membuat kode program dalam bentuk kode tulisan. Dengan version control, programmer bisa merekam semua perubahan yang terjadi, sehingga jika terjadi kesalahan, programmer bisa dengan mudah kembali ke versi sebelumnya
- Tapi tidak hanya untuk file dalam bentuk text, jika misal kita adalah seorang Graphic Designer, kita juga bisa memanfaatkan Version Control untuk merekam perubahan dari file gambar atau layout, sehingga kita tidak perlu membackup tiap perubahan secara manual

Tipe Version Control

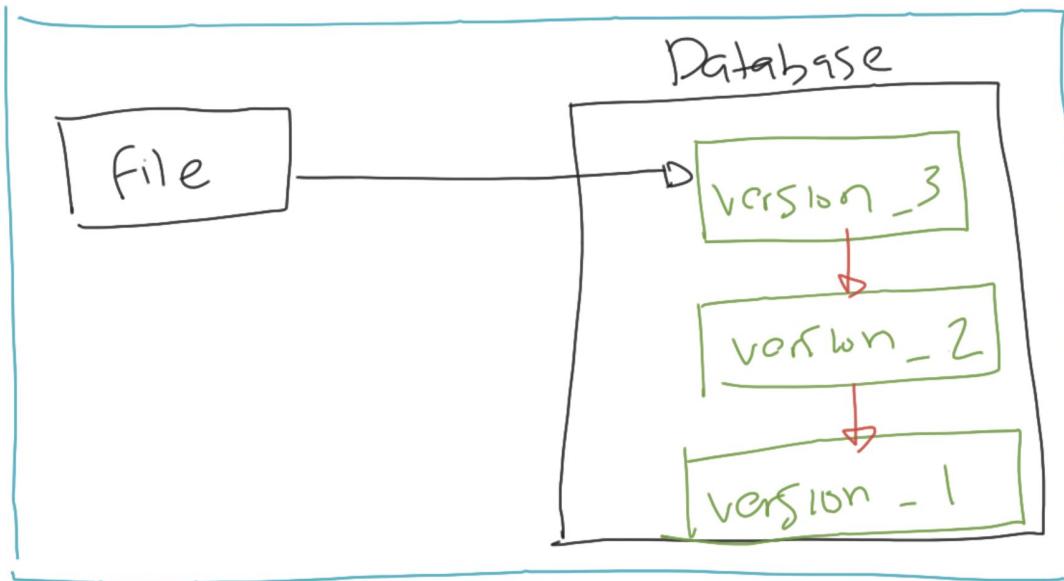
Secara garis besar, version control dibagi menjadi 3 jenis

- Local Version Control
- Centralized Version Control, dan
- Distributed Version Control

Local Version Control

- Local version control merupakan version control yang berjalan hanya di local komputer
- Pendekatan ini biasa digunakan karena sederhana dan tidak butuh server, karena semua riwayat pekerjaan disimpan di local komputer
- Setiap perubahan versi yang terjadi pada file hanya disimpan di local komputer

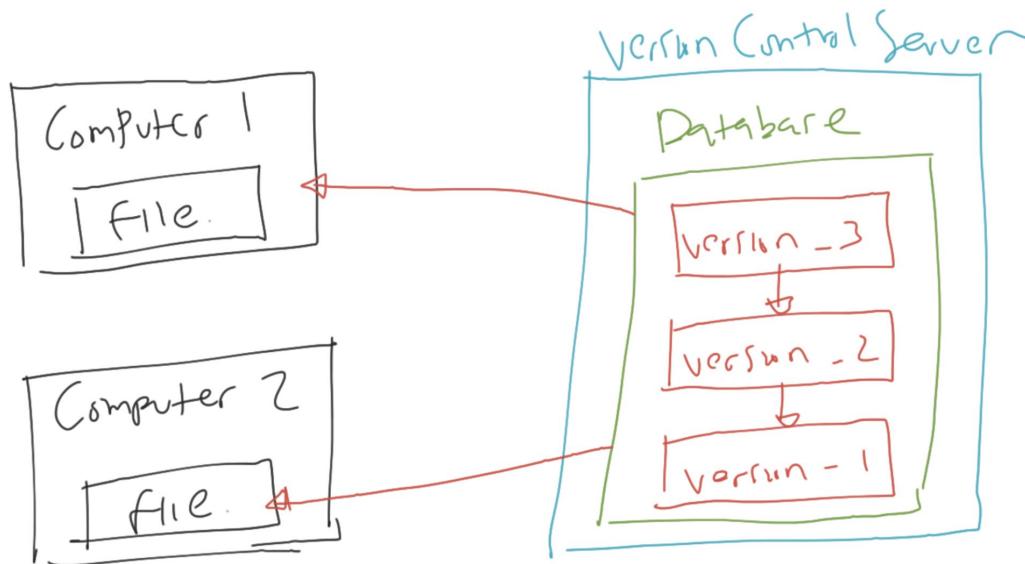
Diagram Local Version Control



Centralized Version Control

- Masalah yang terjadi pada Local Version Control adalah, jika komputer rusak, maka seluruh data bisa hilang.
- Selain itu agak sulit untuk berkolaborasi dengan pengguna lain jika file hanya ada di satu komputer
- Untuk menangani masalah ini, kita bisa menggunakan Centralized Version Control
- Centralized Version Control menyimpan seluruh data revisi di Server
- Pengguna bisa mengakses data ke Server untuk melihat file
- Kekurangannya adalah, jika pengguna offline, mereka tidak bisa melihat riwayat file karena semua riwayat hanya ada di Server
- Jika Server down, maka seluruh pengguna tidak bisa melakukan perubahan dan melihat revisi file
- Contoh Centralized Version Control adalah Subversion

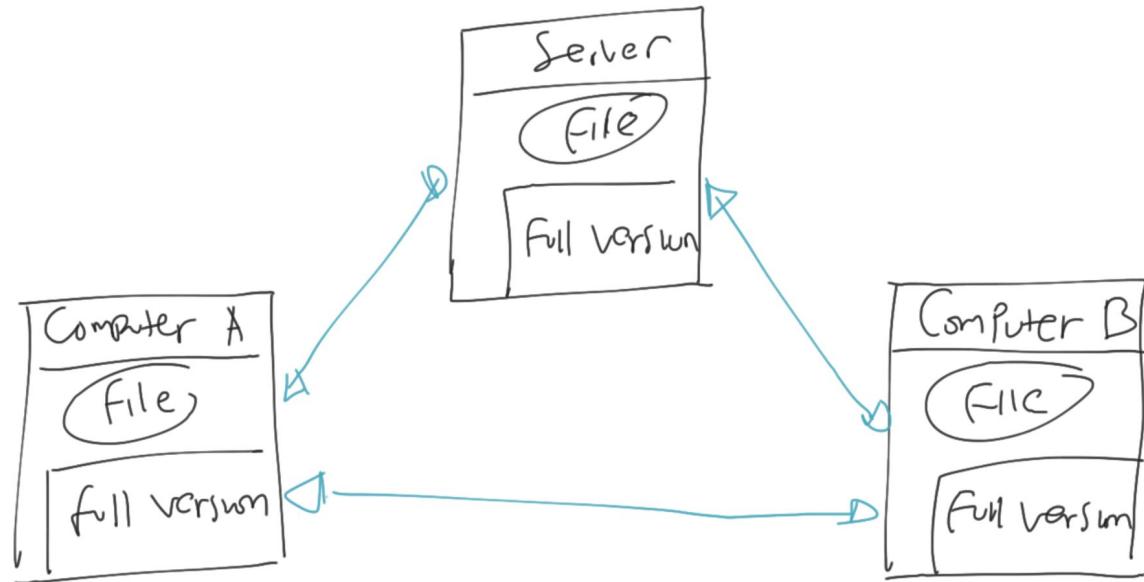
Diagram Centralized Version Control



Distributed Version Control

- Distributed Version Control adalah alternatif lain dari Centralized Version Control
- Dalam DVCs, client tidak hanya mengambil file versi terakhir, namun seluruh riwayat revisi di copy seluruhnya
- Hal ini menjadikan jika terjadi masalah dengan Server, client masih tetap bekerja, memanipulasi file, sampai melihat riwayat perubahan
- Bahkan dalam DVCs, Server bisa lebih dari satu, karena tiap server isinya sama, full backup data
- Contoh DVCs adalah Git, Mercurial dan lain-lain

Diagram Distributed Version Control



Pengenalan Git

Sejarah Git

- Git muncul dengan latar belakang OpenSource project Linux Kernel
- tahun 1991-2002, Linux Kernel di develop hanya dengan memanfaatkan patch dan archived files
- Tahun 2002, Linux Kernel mulai menggunakan DVCs bernama BitKeeper
- Di tahun 2005, hubungan antara perusahaan pemilik BitKeeper dengan komunitas Linux Kernel kurang baik, sehingga pembuat Linux, Linus Torvalds mulai membuat DVCs sendiri
- Git pertama kali dikenalkan tahun 2005, semakin kesini Git semakin populer dan sekarang menjadi DVCs yang paling populer di dunia
- Git sangat cepat, ringan dan baik dalam me-manage project dengan ukuran besar

Pengenalan Git

- Jadi, Git adalah salah satu DVCs yang ada
- Git tidak membutuhkan server untuk melakukan perubahan atau melihat riwayat revisi, hal ini dikarenakan dalam Git, semua riwayat project akan selalu di duplikasi, baik itu di server ataupun di local computer
- Artinya sebenarnya Git juga bisa digunakan sebagai Local Version Control
- Setiap perubahan yang terjadi di Git akan selalu dibuat signature (tanda) nya menggunakan algoritma hashing SHA-1. Hal ini menjadikan perubahan sekecil apapun pasti bisa dideteksi oleh git.
- Semua hal yang terjadi di git secara otomatis akan dicatat, hal ini menjadikan perubahan apapun di Git, pasti selalu bisa dikembalikan ke versi sebelumnya

Menginstall Git

- Git adalah aplikasi OpenSource dan Gratis, kita bisa download aplikasi Git dengan bebas
- Git tersedia untuk berbagai sistem operasi, seperti Windows, Mac dan Linux
- Kita bisa download Git di : <https://git-scm.com/downloads>

Memastikan Git Berjalan

- Git merupakan aplikasi berbasis terminal / command line, oleh karena itu, untuk menggunakan Git, kita perlu membuat terminal / command line
- Untuk mengecek versi Git yang terinstall di local computer kita, kita bisa gunakan perintah :
git --version

```
→ ~ git --version
git version 2.24.3 (Apple Git-128)
→ ~ |
```

Tool Pembantu

- Visual Studio Code : <https://code.visualstudio.com/>
- Install di PATH

Getting Started

>Install code |

Shell Command: **Install 'code' command in PATH**

recently used 

Configuration

Configuration

- Setelah selesai menginstall git, hal yang pertama kali kita lakukan adalah melakukan konfigurasi
- Yang paling utama yang perlu kita konfigurasi diawal adalah user name dan user email

```
→ ~ git config --global user.name "Eko Kurniawan Khannedy"
→ ~ git config --global user.email "echo.khannedy@gmail.com"
→ ~ |
```



Menggunakan Visual Studio Code

- Agar mempermudah, kita akan menjadikan Visual Studio Code sebagai default editor untuk Git dan default diff tool

```
→ ~ git config --global core.editor "code --wait"
→ ~ |
```

```
→ ~ git config --global diff.tool "default-difftool"
→ ~ git config --global difftool.default-difftool.cmd "code --wait --diff \$LOCAL \$REMOTE"
→ ~ |
```

Melihat Seluruh Configuration

```
→ ~ git config --list --show-origin  
→ ~ |
```

Repository

Repository

- Repository merupakan sebutan project di Git
- Kita bisa membuat folder kosong atau folder yang sudah berisi file, lalu membuatnya sebagai Git Repository
- Atau kita bisa melakukan clone Git Repository yang sudah ada dari Server Git

Membuat Repository

- Untuk membuat repository, kita hanya perlu menggunakan perintah :
`git init`
- Dan dilakukan dari dalam folder yang akan kita jadikan sebagai Git Repository
- Setelah membuat Git Repository, kita bisa lihat ada folder baru dengan nama `.git`
- `.git` merupakan folder yang berisikan database Git, jangan sampai kita mengubah data yang terdapat pada folder `.git`

Kode : Git Init

```
→ belajar-git-dasar git init
Initialized empty Git repository in /Users/khannedy/Developments/BELAJAR/belajar-git-dasar/.git/
→ belajar-git-dasar git status
```

Kode : Git Status

```
→ belajar-git-dasar git:(master) git status
```

```
On branch master
```

```
No commits yet
```

```
nothing to commit (create/copy files and use "git add" to track)
```

```
→ belajar-git-dasar git:(master) █
```

Workflow

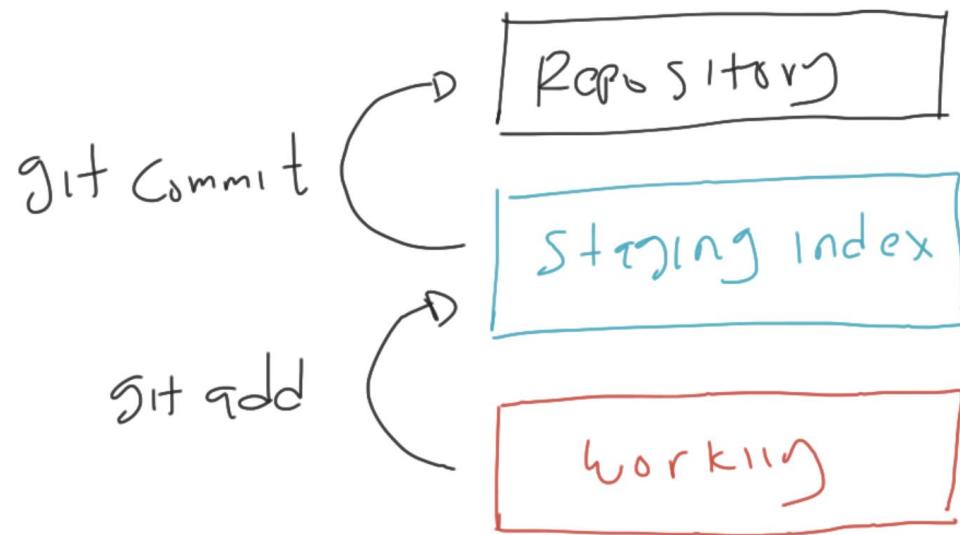
The Three States

- Harap diperhatikan, ini adalah hal utama yang wajib dimengerti di Git agar selanjutnya bisa mengerti dengan baik
- Git memiliki tiga state terhadap file kita: modified, staged dan committed
- Modified artinya kita mengubah (menambah, mengedit, menghapus) file, namun belum disimpan secara permanen ke repository
- Staged artinya kita menandai modifikasi yang kita lakukan terhadap file akan disimpan secara permanen ke repository
- Committed artinya data sudah aman disimpan di repository

Three Section

- Tiga state sebelumnya di dalam Git dilakukan di section yang berbeda-beda, yaitu Working Directory, Staging Area dan Repository
- Saat kita melakukan modifikasi file, itu dilakukan di working directory
- Staging Area merupakan section dimana file sudah disiapkan untuk disimpan secara permanen, di Staging Area semua informasi perubahan file akan disimpan
- Repository merupakan tempat dimana semua file dan database riwayat versi file disimpan

Diagram Three Tree



Workflow

- Sekarang kita sudah tahu tentang arsitektur Three Tree, sekarang pertanyaannya, bagaimana alur kerja menggunakan Git
- Secara sederhana, setiap perubahan akan kita lakukan di working directory
- Jika ada yang mau kita siapkan untuk disimpan secara permanen, kita akan bawa perubahan tersebut ke staging index
- Selanjutnya, kita bisa melakukan penyimpanan versi baru secara permanen ke repository

Diagram Workflow (1)

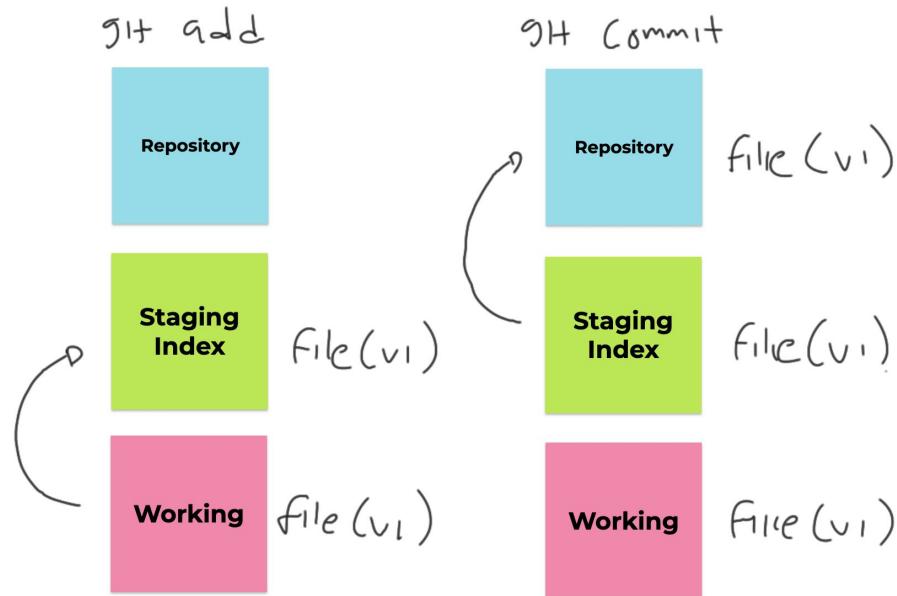
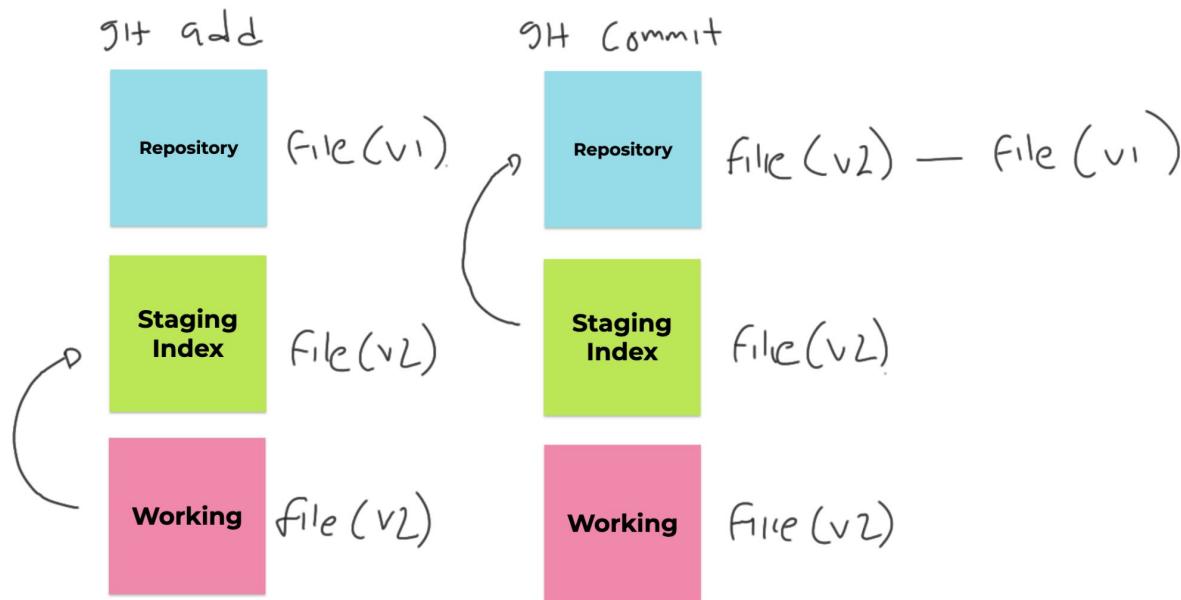


Diagram Workflow (2)



Hash

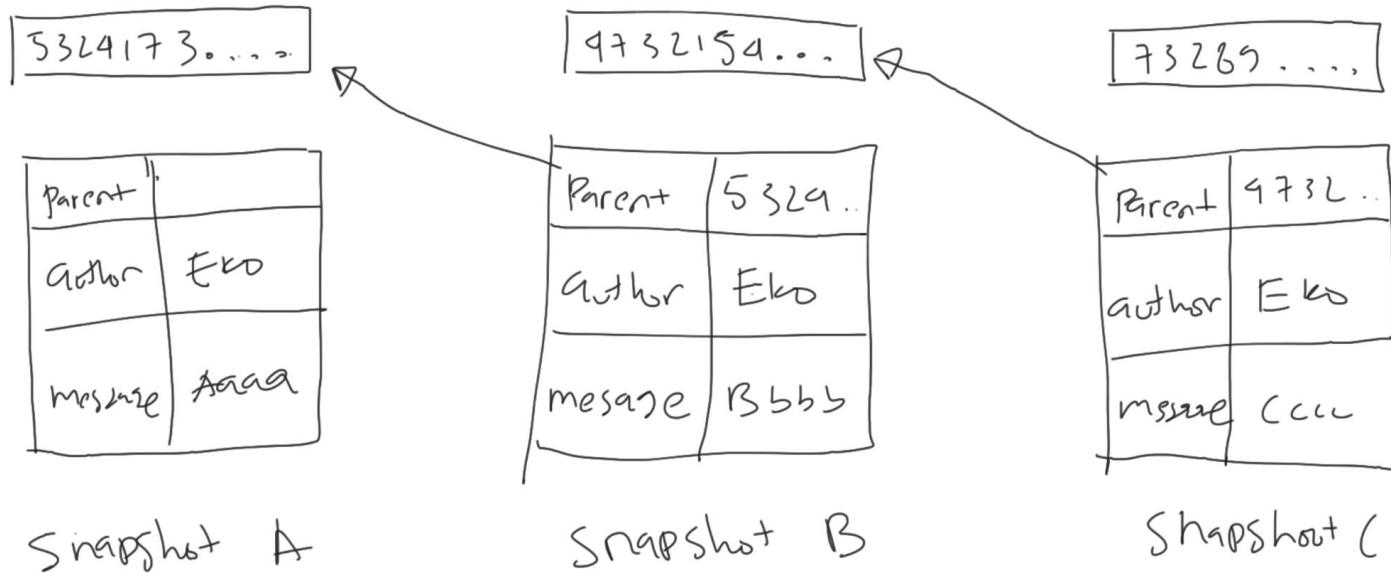
Snapshot

- Pada materi sebelumnya kita selalu menyebutkan versi pada perubahan file
- Sebenarnya perubahan yang dilakukan bisa jadi dilakukan secara bersamaan untuk beberapa file, hal ini berarti sebenarnya tidak ada yang namanya versi file
- Semua perubahan yang terjadi akan direkam, dan kita sebut namanya adalah snapshot
- Snapshot berisikan semua perubahan yang terjadi di semua file yang kita commit
- Setiap snapshot akan menghasilkan hash

Hash

- Setiap snapshot yang kita lakukan, semua akan menghasilkan hash sebagai identitas snapshot nya
- Hash merupakan checksum untuk menghitung perubahan yang terjadi
- Git menggunakan algoritma SHA-1 untuk menghitung hash
- Hash dibutuhkan untuk menjaga data integrity, sehingga tidak tiap snapshot yang sudah kita lakukan tidak bisa diubah, hal ini karena akan secara otomatis merusak hash yang sudah dibuat
- Contoh hash Git : 30534aeabde65981732c6c469b7583483d379b00

Diagram Snapshot



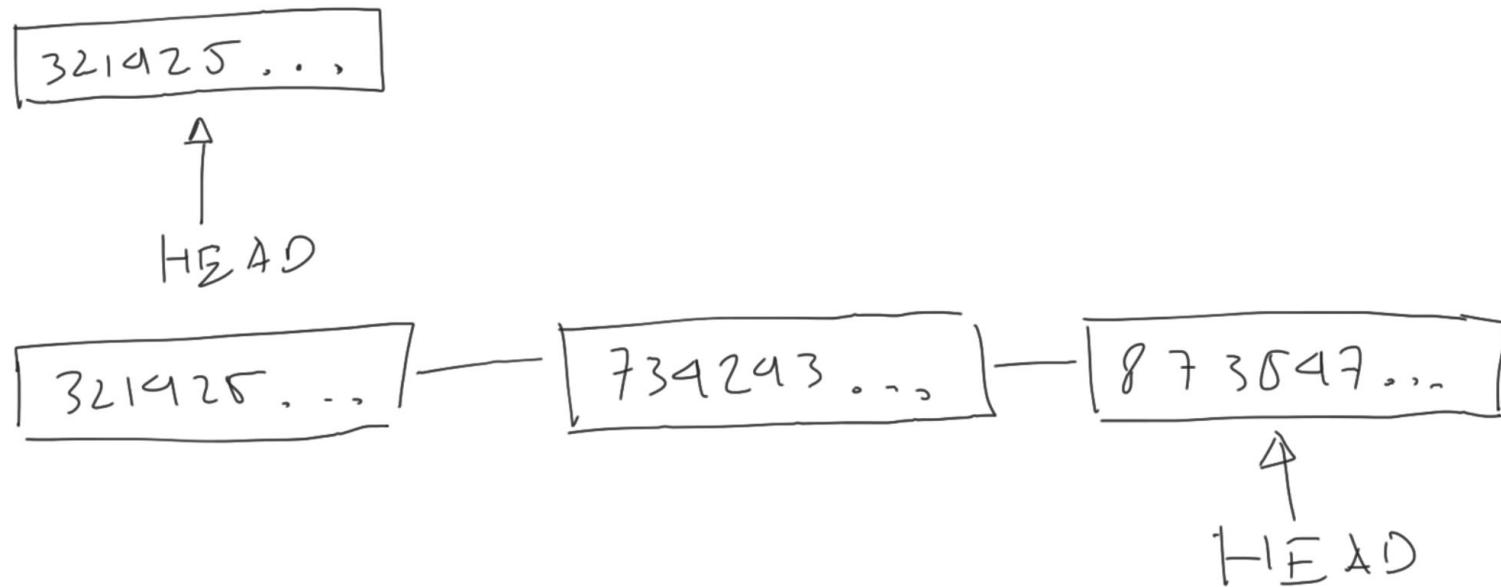
Perhitungan Hash

- Perhitungan hash dilakukan tidak hanya dari perubahan file
- Namun juga dari parent, author dan message
- Artinya perubahan yang terjadi pada snapshot sebelumnya, maka akan menimbulkan efek berantai, karena semua hash selanjutnya akan berubah
- Oleh karena itu, hal tersebut tidak bisa dilakukan di Git

HEAD

- HEAD merupakan pointer menuju hash yang paling akhir
- Karena kadang sangat menyulitkan jika harus menuliskan hash value, jika kita akan menuju ke hash paling baru, kita bisa gunakan kata HEAD

Diagram HEAD



Menambah File

Menambah File

- Untuk menambah file baru ke Repository, kita cukup tambahkan file nya saja
- Secara otomatis file yang kita tambahkan awalnya akan berada di working directory
- Secara default saat menambah file baru, file tersebut tidak akan di track perubahannya
- Agar perubahan di track, kita harus pindahkan dari working directory ke staging index

Kode : Untracked Files

```
→ belajar-git-dasar git:(master) code .  
→ belajar-git-dasar git:(master) git status
```

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)
 file1.txt

nothing added to commit but untracked files present (use "git add" to track)

```
→ belajar-git-dasar git:(master) x
```

Kode : Memindahkan ke Staging Index

```
→ belajar-git-dasar git:(master) ✘ git add file1.txt  
→ belajar-git-dasar git:(master) ✘ git status
```

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)
new file: file1.txt

```
→ belajar-git-dasar git:(master) ✘ █
```

Kode : Commit ke Repository

```
→ belajar-git-dasar git:(master) ✘ git commit -m "Menambah file1.txt"
[master (root-commit) 7453f5e] Menambah file1.txt
 1 file changed, 1 insertion(+)
  create mode 100644 file1.txt
→ belajar-git-dasar git:(master) git status
On branch master
nothing to commit, working tree clean
→ belajar-git-dasar git:(master) █
```

Tugas

- Tambah file2.txt, lalu commit ke Repository
- Tambah file3.txt, lalu commit ke Repository

Mengubah File

Mengubah File

- Untuk melakukan perubahan file, kita cukup lakukan perubahan file terhadap file yang sudah ada di Repository
- Secara otomatis git bisa mendeteksi perubahan
- Sama seperti dengan menambah file, jika perubahan ingin kita simpan secara permanen, kita bisa pindahkan ke staging index, lalu commit ke Repository

Kode : Git Status

```
→ belajar-git-dasar git:(master) git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file3.txt

no changes added to commit (use "git add" and/or "git commit -a")
→ belajar-git-dasar git:(master) x
```

Melihat Perubahan File

- Ketika kita melakukan perubahan, Git secara otomatis mendeteksi bahwa file tersebut berubah
- Jika kita ingin melihat perubahan yang terjadi, kita juga bisa menggunakan Git untuk melihat perubahannya dengan perintah :
`git diff`

Kode : Melihat Perubahan File

```
diff --git a/file3.txt b/file3.txt
index a2a9337..aaaf8c5a 100644
--- a/file3.txt
+++ b/file3.txt
@@ -1 +1,2 @@
-Ini adalah file 3
\ No newline at end of file
+Ini adalah file 3
+Menambah baris di file3.txt
\ No newline at end of file
(END)
```

Tugas

- Commit perubahan file3.txt ke Repository
- Ubah file1.txt dan file2.txt secara bersamaan, lalu commit semuanya ke Repository

Menghapus File

Menghapus File

- Untuk menghapus file di Repository, kita cukup lakukan delete file nya di folder nya
- Secara otomatis Git akan mendeteksi file yang hilang
- Sama seperti menambah dan menghapus, jika ingin simpan secara permanen di Repository, kita harus menambahkan operasi tersebut ke Staging Index, lalu commit ke Repository

Kode : Git Status

```
→ belajar-git-dasar git:(master) git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    file3.txt

no changes added to commit (use "git add" and/or "git commit -a")
→ belajar-git-dasar git:(master) x
```

Kode : Git Commit

```
→ belajar-git-dasar git:(master) ✘ git add file3.txt
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    file3.txt

→ belajar-git-dasar git:(master) ✘ git commit -m "menghapus file3.txt"
[master ae4a58c] menghapus file3.txt
 1 file changed, 2 deletions(-)
 delete mode 100644 file3.txt
→ belajar-git-dasar git:(master) █
```

Membatalkan Perubahan

Membatalkan Penambahan File

- Jika kita menambah file di Working Directory, lalu misal kita ingin membatalkan perubahannya
- Caranya cukup sederhana, kita hanya perlu menghapus file tersebut, atau bisa menggunakan perintah:
`git clean -f`

Membatalkan Perubahan File

- Jika kita ingin membatalkan perubahan file yang telah kita lakukan, kita bisa menggunakan perintah :
`git restore namafile`



Kode : Git Restore

```
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

```
→ belajar-git-dasar git:(master) ✘ git restore file1.txt
```

```
→ belajar-git-dasar git:(master) git status
```

```
On branch master
```

```
nothing to commit, working tree clean
```

```
→ belajar-git-dasar git:(master) █
```

Membatalkan Penghapusan File

- Penghapusan file sebenarnya sama dengan perubahan file, jadi jika kita ingin membatalkan penghapusan file, kita bisa gunakan perintah yang sama dengan membatalkan perubahan file :
`git restore namafile`



Kode : Git Restore

```
→ belajar-git-dasar git:(master) git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
→ belajar-git-dasar git:(master) ✘ git restore file1.txt
→ belajar-git-dasar git:(master) git status
On branch master
nothing to commit, working tree clean
→ belajar-git-dasar git:(master) █
```

Membatalkan dari Staging Index

- Perintah git restore hanya bisa dilakukan untuk membatalkan perubahan yang terjadi di Working Directory
- Artinya jika perubahan terlanjur kita masukkan ke Staging Index, maka untuk membatalkannya tidak bisa kita lakukan secara langsung dari Staging Index
- Kita perlu mengembalikan posisi dari Staging Index ke Working Directory terlebih dahulu, caranya kita bisa gunakan perintah :
git restore --staged namafile

Kode : Git Restore di Staging Index

```
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    file1.txt

→ belajar-git-dasar git:(master) ✘ git restore file1.txt
error: pathspec 'file1.txt' did not match any file(s) known to git
→ belajar-git-dasar git:(master) ✘
```



Kode : Git Restore ke Working Directory

```
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    file1.txt

→ belajar-git-dasar git:(master) ✘ git restore --staged file1.txt
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    file1.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Membatalkan Yang Sudah di Commit

- Bagaimana jika perubahan yang kita lakukan terlanjur di commit?
- Tidak ada cara yang bisa kita lakukan jika perubahan sudah terlanjur di commit
- Yang bisa kita lakukan adalah dengan dua cara, Rollback Commit atau Revert Commit
- Kedua cara tersebut akan kita bahas di materi sendiri-sendir

Commit Log

Commit Log

- Git adalah distributed version control, artinya walaupun kita Repository di local komputer kita, semua riwayat perubahan disimpan di komputer kita
- Kekurangannya menjadi makin lama Repository akan semakin besar ukurannya, namun keuntungannya, kita bisa melihat semua riwayat commit, atau disebut Commit Log
- Untuk melihat Commit Log, kita bisa gunakan perintah :
`git log`



Kode : Git Log

```
commit ae4a58c9c89de0ab52915185d5ea3cab8df69c40 (HEAD -> master)
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 10:49:30 2021 +0700
```

menghapus file3.txt

```
commit 1b5f5645b4f13233ce228b57db7babe4e4b7ef61
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 09:11:13 2021 +0700
```

perubahan di file1.txt dan file2.txt

```
commit 2118ac318ee242d5101934900ada8aaee0de7826
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 09:10:20 2021 +0700
```

perubahan di file3.txt

```
commit 59272c69f9388aa712e3e34d248dda33a8c008be
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 09:05:14 2021 +0700
```

menambah file3.txt

```
commit 59272c69f9388aa712e3e34d248dda33a8c008be
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 09:05:14 2021 +0700
```

menambah file3.txt

```
commit 4f15c9b5550720f8d908380a0e59a94bf42f7edf
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 09:05:07 2021 +0700
```

menambah file2.txt

```
commit 7453f5ea0c123a2b292e1e8ce52edbb7a7fbafcc
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
Date:   Sat Jun 19 09:00:34 2021 +0700
```

Menambah file1.txt
(END)

Log Sederhana

- Kadang kita hanya ingin melihat commit log message nya saja atau istilahnya adalah versi sederhananya saja
- Untuk melakukan itu, kita bisa gunakan perintah :
`git log --oneline`

Kode : Git Log Oneline

```
ae4a58c (HEAD -> master) menghapus file3.txt  
1b5f564 perubahan di file1.txt dan file2.txt  
2118ac3 perubahan di file3.txt  
59272c6 menambah file3.txt  
4f15c9b menambah file2.txt  
7453f5e Menambah file1.txt  
(END)
```

Graph

- Saat nanti kita sudah belajar tentang Git Branching, kadang kita ingin melihat commit log dengan hubungannya dengan commit log sebelumnya
- Hal ini bisa kita lakukan menggunakan perintah :
`git log --oneline --graph`

Kode : Git Log Graph

```
* ae4a58c (HEAD -> master) menghapus file3.txt
* 1b5f564 perubahan di file1.txt dan file2.txt
* 2118ac3 perubahan di file3.txt
* 59272c6 menambah file3.txt
* 4f15c9b menambah file2.txt
* 7453f5e Menambah file1.txt
```

(END)



Kode : Contoh Git Graph Kompleks

```
* 6137d45 (HEAD -> master, origin/master) Start development 0.0.9-SNAPSHOT
*   7812110 Merge pull request #41 from bliblidotcom/release/0.0.8
|\ \
| * d9887b1 (tag: 0.0.8, origin/release/0.0.8, release/0.0.8) Release 0.0.8
| * e698832 Support RequestParam with Collection (multiple values)
|/
* 81e07ff Update to maven central
*   c6c5a92 Merge pull request #40 from bliblidotcom/feature/github-packages
|\ \
| * f01f0f4 Change to github package
|/
* ac7a927 Start Development 0.0.8-SNAPSHOT
*   297174b Merge branch 'master' of github.com:bliblidotcom/blibli-backend-framework
|\ \
| *   f224b89 Merge pull request #39 from bliblidotcom/bugfix/sleuth-not-detected-by-api-client
| | \
*   | b469d34 (tag: 0.0.7, origin/bugfix/sleuth-not-detected-by-api-client) Release 0.0.7
| | /
|/
*   | 01bad61 Sleuth not detected by api client
|/
```

Melihat Detail Commit

- Kadang kita ingin melihat detail perubahan yang terjadi pada sebuah commit
- Untuk melakukan itu, kita bisa gunakan perintah :
`git show hash`

Kode : Git Show

```
commit 1b5f5645b4f13233ce228b57db7babe4e4b7ef61
```

```
Author: Eko Kurniawan Khannedy <echo.khannedy@gmail.com>
```

```
Date: Sat Jun 19 09:11:13 2021 +0700
```

```
perubahan di file1.txt dan file2.txt
```

```
diff --git a/file1.txt b/file1.txt
```

```
index c7a8c45..f61cc80 100644
```

```
--- a/file1.txt
```

```
+++ b/file1.txt
```

```
@@ -1 +1,2 @@
```

```
-Ini adalah file 1
```

```
\ No newline at end of file
```

```
+Ini adalah file 1
```

```
+Menambah baris di file1.txt
```

```
\ No newline at end of file
```

```
diff --git a/file2.txt b/file2.txt
```

```
index 756daa2..8d28041 100644
```

```
--- a/file2.txt
```

```
+++ b/file2.txt
```

```
@@ -1 +1,2 @@
```

```
-Ini adalah file 2
```

```
\ No newline at end of file
```

```
+Ini adalah file 2
```

```
+Menambah baris di file2.txt
```

```
\ No newline at end of file
```

```
(END)
```

Compare Commit

Compare Commit

- Git memiliki fitur untuk membandingkan antara commit dengan commit lainnya
- Namun jangan sampai salah pengertian, membandingkan disini adalah membandingkan snapshot hasil commit, bukan perubahan yang terjadi antara commit
- Misal pada commit sebelumnya kita pernah menambah file3.txt, namun jika kita bandingkan antara commit pertama dan terakhir (HEAD), hasilnya hanyalah perbandingan antara file1 dan file2, tidak ada file3
- Hal ini dikarenakan membandingkan commit bukanlah membandingkan perubahan yang pernah terjadi, melainkan membandingkan hasil di commit
- Untuk membandingkan commit, kita bisa gunakan perintah :
`git diff hash1 hash2`

Kode : Git Diff

```
git diff 7453f5e HEAD
```

```
diff --git a/file1.txt b/file1.txt
index c7a8c45..f61cc80 100644
--- a/file1.txt
+++ b/file1.txt
@@ -1 +1,2 @@
-Ini adalah file 1
\ No newline at end of file
+Ini adalah file 1
+Menambah baris di file1.txt
\ No newline at end of file
diff --git a/file2.txt b/file2.txt
new file mode 100644
index 0000000..8d28041
--- /dev/null
+++ b/file2.txt
@@ -0,0 +1,2 @@
+Ini adalah file 2
+Menambah baris di file2.txt
\ No newline at end of file
(END)
```

Difftool

- Sebelumnya kita sudah melakukan pengaturan menggunakan Visual Studio Code untuk melihat diff
- Jika kita ingin menggunakan visual studio code untuk melihat perbedaan antar commit, kita bisa gunakan perintah :
`git difftool hash1 hash2`

Difftool Visual Studio Code

≡ 1y07mD_file1.txt ↔ lzhjnD_file1.txt X

var > folders > dp > 3nnc32552b9_tw18gsxp5dp40000gn > T > ≡ lzhjnD_file1.txt

1 Ini adalah file 1

1 Ini adalah file 1

2+ Menambah baris di file1.txt

Rename File

Rename File

- Hal yang paling menarik di Git adalah, Git bisa mendeteksi rename file
- Secara sederhana sebenarnya rename file merupakan operasi gabungan antara hapus file, lalu menambah file baru dengan isi yang sama
- Namun Git bisa otomatis mendeteksi jika terjadi perubahan nama file, karena isi file sebagian besar masih sama



Kode : Git Status

```
→ belajar-git-dasar git:(master) git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    file2.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file_2.txt

no changes added to commit (use "git add" and/or "git commit -a")
→ belajar-git-dasar git:(master) x █
```

Kode : Git Status di Staging Index

```
→ belajar-git-dasar git:(master) ✘ git add file2.txt  
→ belajar-git-dasar git:(master) ✘ git add file_2.txt  
→ belajar-git-dasar git:(master) ✘ git status
```

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)
renamed: file2.txt -> file_2.txt

```
→ belajar-git-dasar git:(master) ✘ █
```

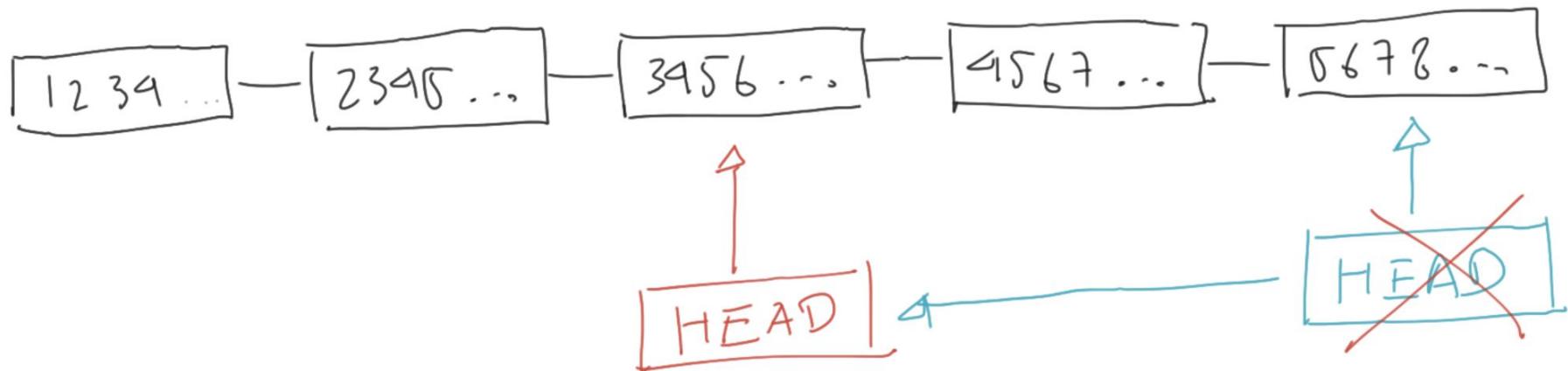
Reset Commit

Reset Commit

- Sebelumnya kita sudah tahu membatalkan perubahan, namun bagaimana jika ternyata perubahan sudah terlanjur kita commit ke Repository?
- Untuk hal seperti itu, kita bisa melakukan reset commit
- Reset commit merupakan mekanisme dimana kita menggeser HEAD pointer ke posisi commit yang kita mau, artinya commit selanjutnya akan dilakukan pada posisi HEAD baru
- Untuk melakukan reset commit, kita bisa gunakan perintah :
`git reset <mode> hash`
- Ada beberapa mode pengaturan melakukan reset commit

Diagram Git Reset

git reset <mode> 3956...



Mode Git Reset

- --soft, memindahkan HEAD pointer, namun tidak melakukan perubahan apapun di Staging Index dan Working Directory
- --mixed (default), memindahkan HEAD pointer, mengubah Staging Index menjadi sama seperti dengan Repository, namun tidak mengubah apapun di Working Directory
- --hard, memindahkan HEAD pointer, dan mengubah Staging Index dan Working Directory sehingga sama dengan Repository

Kode : Git Log

```
b34dc9a (HEAD -> master) mengubah file2.txt menjadi file_2.txt
ae4a58c menghapus file3.txt
1b5f564 perubahan di file1.txt dan file2.txt
2118ac3 perubahan di file3.txt
59272c6 menambah file3.txt
4f15c9b menambah file2.txt
7453f5e Menambah file1.txt
(END)
```

Kode : Git Reset Soft

```
→ belajar-git-dasar git:(master) git reset --soft ae4a58c
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    file2.txt -> file_2.txt

→ belajar-git-dasar git:(master) ✘
```

Rewrite Riwayat Commit

- Jika kita melakukan reset, namun kita belum membuat commit baru
- Kita masih bisa kembali maju lagi ke commit yang paling baru
- Namun jika kita membuat commit baru, secara otomatis commit lama akan ditimpa oleh commit baru

Kode : Git Reset Mixed

```
→ belajar-git-dasar git:(master) git reset --mixed ae4a58c
Unstaged changes after reset:
D      file2.txt
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:   file2.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file_2.txt

no changes added to commit (use "git add" and/or "git commit -a")
→ belajar-git-dasar git:(master) ✘ █
```

Kode : Git Reset Hard

```
→ belajar-git-dasar git:(master) git reset --hard ae4a58c
HEAD is now at ae4a58c menghapus file3.txt
→ belajar-git-dasar git:(master) git status
On branch master
nothing to commit, working tree clean
→ belajar-git-dasar git:(master) █
```

Amend Commit

Amend Commit

- Kadang saat sudah melakukan commit, mungkin ada beberapa hal yang terlupakan
- Biasanya kita akan lakukan reset soft ke commit sebelumnya, lalu tambahkan perubahan yang terlupakan, lalu kita lakukan commit ulang
- Hal tersebut bisa dilakukan tanpa manual melakukan reset, caranya bisa menggunakan perintah :
git commit --amend
- Perlu diingat, amend akan mengubah hash commit karena data perubahan yang dicommit bertambah

Tugas

- Tambah file3.txt
- Commit file3.txt ke Repository
- Ubah file3.txt
- Commit dengan amend ke Repository :
git commit --amend

Kode : Git Commit Amend

```
→ belajar-git-dasar git:(master) git add file3.txt
→ belajar-git-dasar git:(master) ✘ git commit --amend -m "Add file3.txt"
[master a26ec92] Add file3.txt
  Date: Sat Jun 19 22:54:32 2021 +0700
  1 file changed, 2 insertions(+)
  create mode 100644 file3.txt
→ belajar-git-dasar git:(master) █
```

Versi Sebelumnya

Versi Sebelumnya

- Kadang kita sering mengalami masalah dengan file yang sudah kita commit ke Repository
- Git memiliki fitur dimana kita bisa melihat versi file pada commit sebelumnya
- Saat kita ambil versi file sebelumnya, file pada commit tersebut akan berada di Staging Index
- Untuk melakukannya, kita bisa gunakan perintah :
`git checkout hash -- namafile`

Kode : Git Status

Misal kita ingin melihat file1.txt sebelum terjadi perubahan di commit 1b5f564, maka kita bisa gunakan perintah : git checkout 2118ac3 -- file1.txt

```
a26ec92 (HEAD -> master) Add file3.txt  
b34dc9a mengubah file2.txt menjadi file_2.txt  
ae4a58c menghapus file3.txt  
1b5f564 perubahan di file1.txt dan file2.txt  
2118ac3 perubahan di file3.txt  
59272c6 menambah file3.txt  
4f15c9b menambah file2.txt  
7453f5e Menambah file1.txt
```

Kode : Git Checkout

```
→ belajar-git-dasar git:(master) git log -1
→ belajar-git-dasar git:(master) git checkout 2118ac3 -- file1.txt
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   file1.txt

→ belajar-git-dasar git:(master) ✘
```

Snapshot Sebelumnya

Snapshot Sebelumnya

- Git juga memiliki fitur seperti mesin waktu, dimana kita bisa kembali pada snapshot sebelumnya
- Kita bisa tentukan kemana tujuan snapshot kita hanya dengan menggunakan hash commit
- Cara jika kita ingin menuju ke snapshot tertentu, cukup gunakan perintah :
git checkout hash
- Jika ingin kembali ke paling awal, kita bisa gunakan perintah :
git checkout namabranch



Git Branch

- Materi branching akan dibahas pada course terpisah, namun secara default saat kita membuat Git Repository, maka secara otomatis Git akan membuat branch
- Untuk melihat nama branch saat ini, kita bisa gunakan perintah :
git branch --show-current

```
→ belajar-git-dasar git:(master) git branch --show-current  
master  
→ belajar-git-dasar git:(master)
```

Kode : Git Checkout

```
→ belajar-git-dasar git:(master) git log -oneLine
→ belajar-git-dasar git:(master) git checkout 7453f5e
Note: switching to '7453f5e'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

Kode : Kembali Ke Commit Terakhir

```
→ belajar-git-dasar git:(7453f5e) git checkout master
Previous HEAD position was 7453f5e Menambah file1.txt
Switched to branch 'master'
→ belajar-git-dasar git:(master) █
```

Revert Commit

Revert Commit

- Git memiliki fitur revert commit, yaitu fitur untuk membatalkan commit yang sudah kita lakukan dengan cara membuat commit baru yang membatalkan commit sebelumnya
- Misal kita sudah melakukan commit data perubahan dari text Eko menjadi Eka, jika kita revert, secara otomatis akan membuat commit baru dengan melakukan perubahan dari Eko ke Eko
- Untuk melakukan revert commit, kita bisa gunakan perintah :
`git revert hash`

Kode : Git Status

Misal kita akan revert commit melakukan rename dari file2.txt menjadi file_2.txt

```
a26ec92 (HEAD -> master) Add file3.txt
b34dc9a mengubah file2.txt menjadi file_2.txt
ae4a58c menghapus file3.txt
1b5f564 perubahan di file1.txt dan file2.txt
2118ac3 perubahan di file3.txt
59272c6 menambah file3.txt
4f15c9b menambah file2.txt
7453f5e Menambah file1.txt
```

Kode : Git Revert

```
→ belajar-git-dasar git:(master) git revert b34dc9a
[master 8b9ed18] Revert "mengubah file2.txt menjadi file_2.txt"
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename file_2.txt => file2.txt (100%)
→ belajar-git-dasar git:(master) █
```



Kode : Git Log Setelah Revert

```
8b9ed18 (HEAD -> master) Revert "mengubah file2.txt menjadi file_2.txt"  
a26ec92 Add file3.txt  
b34dc9a mengubah file2.txt menjadi file_2.txt  
ae4a58c menghapus file3.txt  
1b5f564 perubahan di file1.txt dan file2.txt  
2118ac3 perubahan di file3.txt  
59272c6 menambah file3.txt  
4f15c9b menambah file2.txt  
7453f5e Menambah file1.txt  
(END)
```

Ignore

Ignore

- Kadang saat membuat aplikasi, tidak semua file ingin kita track di Git, contoh seperti file log, hasil kompilasi, kadang itu tidak butuh di track di Git
- Git memiliki fitur ignore, dimana kita bisa meminta Git secara otomatis tidak men-track file di Git
- Caranya kita bisa tambahkan file .gitignore di Repository
- Lalu kita bisa tambahkan tiap baris di file .gitignore berisikan file atau folder yang tidak kita ingin track

Kode : File .gitignore

```
❖ .gitignore U ●  
❖ .gitignore  
1 # Ignore folder log  
2 log/  
3  
4 # Ignore file dengan extension .backup  
5 *.backup  
6  
7 # Ignore file  
8 ignore.txt  
9
```

Kode : Git Status

```
✓ BELAJAR-GIT-DASAR
  ✓ log
    ≡ application.log
    ≡ database.log
  ♦ .gitignore
    ≡ file1.txt
    ≡ file2.txt
    ≡ file3.txt
    ≡ ignore.txt
    ≡ sample.backup
```

```
→ belajar-git-dasar git:(master) ✘ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will
   be committed)

    .gitignore

nothing added to commit but untracked files present
→ belajar-git-dasar git:(master) ✘
```

Blame

Blame

- Saat membuat kode program kadang kita ingin tahu, siapa yang menambahkan baris kode program tersebut, dan apa saja yang ditambahkan
- Git memiliki fitur yang bernama blame, ini digunakan untuk mencari tahu, siapa yang menambah perubahan pada file dan juga untuk mengetahui commit nya
- Caranya kita bisa gunakan perintah :
`git blame namafile`



Kode : Git Blame

```
1b5f5645 (Eko Kurniawan Khannedy 2021-06-19 09:11:13 +0700 1) Ini adalah file 1  
1b5f5645 (Eko Kurniawan Khannedy 2021-06-19 09:11:13 +0700 2) Menambah baris di file1.txt  
(END)
```

Alias

Alias

- Git memiliki fitur yang bernama alias
- Dengan alias, kita bisa menambah nama perintah lain untuk perintah yang sudah ada di git
- Misal kita bisa menambah perintah co, komit untuk nama lain dari commit misalnya
- Atau misal menambah alias logline untuk nama lain dari log --oneline



Kode : Menambah Alias

```
→ belajar-git-dasar git:(master) git config --global alias.ko commit  
→ belajar-git-dasar git:(master) git config --global alias.komit commit  
→ belajar-git-dasar git:(master) git config --global alias.logone "log --oneline"  
→ belajar-git-dasar git:(master)
```

Kode : Menggunakan Alias

```
→ belajar-git-dasar git:(master) git add file4.txt
→ belajar-git-dasar git:(master) ✘ git ko -m "menambah file4"
[master de51b5c] menambah file4
 1 file changed, 1 insertion(+)
 create mode 100644 file4.txt
→ belajar-git-dasar git:(master) git logone
```

Materi Selanjutnya

Materi Selanjutnya

- Git Branching
- Git Remote