Full Tutorial Git VCS

oleh Maulana Ifandika
https://ifandika.github.io
27-3-2025

| Git

- VCS = Version Control System.
- Suatu system yang menyimpan dan mengelola
  rekaman dari source code.
- Free.

| Tipe Version Control

• Local Version Control
  (Version control yang berjalan pada local komputer)

• Centralized Version Control (subversion)
  (Version control yang berjalan pada local dan di server(database),
  di local hanya versi yang terbaru dan di server full versinya)

• Distributed Version Control (Git dll)
  (alternatif dari CVCs, dalam DVCs tidak hanya versi terakhirnya
  saja namun semua riwayat di copy, sehingga jika server down maka masih biasa berkerja)

- Git = Software VCS
- https://git-scm.com/
- Windows | Mac OS | Linux/Unix

| Sejarah Git dan tentang Git
- Latar belakang OpenSource projek Linux Kernel
- Tahun 1991-2001 Linux Kernel hanya di develop hanya dengan menfaatkan patch
  dan archived files
- 2002 Linux Kernel memulai dengan DVCs bernama bitkeeper
- 2005 hubungan dengan perusahaan bitkeeper kurang baik, sehingga pembuat
  linux, Linus Torvalds membuat DVCs sendiri
- Git di perkenalkan pertama kali pada tahun 2005
- Git menggunakan algoritma hashing SHA-1
- Git lebih mengedepankan perubahan yang terjadi di file tersebut

---

| Istilah pada git

- Repo (repository, sebutan projek di Git, folder kosong yang akan kita jadikan git repo)
- Branch (cabang bebas dari commit seperti master, main dll)
- Hash (penanda unik pada commit berupa

- Angka dan huruf, Git menggunakan
- Algoritma SHA-1, hash harus di jaga agar data tetap valid jika hash yang di belakang rusak maka data yang di depan tidak valid/rusak)
- Commit (rekaman / snapshot pada repo)
- Snapshot (berisikan semua perubahan yang terjadi pada semua file yang kita commit, dan setiap snapshot akan menghasilkan hash)
- Remote (sumber / sever yang memiliki repo github, gitlab)
- Checkout (berpindah ke sebuah commit)
- Clone (menggambil repo dari remote)
- Push (mengirim commit ke repo server)
- Pull (mengambil commit dari repo server)
- Merge (menggabungka branch)
- HEAD (pointer menuju hash paling akhir/versi terbaru)
- .git (adalah isi dari repository kita)

| 3 Area pada git / The Tree States

- Working directory [READ] | Tempat file yang belum di track / new file.
- Staging index/staging area [GREEN] | Tempat file yang sudah di track dan siap untuk commit.
- Commit [BLACK] | File sudah di commit, dan aman di repo.

| Command git dan fungsinya

+ Instalasi
  - Install git di terminal.
    sudo/apt install git (debian/ubuntu)
  - Menampilkan git / seperti help.
    git
  - Menampilkan version git)
    git --version

+ Konfigurasi
  - Isi nama untuk git.
    git config --global user.name "nama anda"

  - Isi email untuk git.
    git config --global user.email "email anda"

  - Lihat seluruh konfigurasi.
    git config --list --show-origin
    git config --list

  - Shortcut untuk command git.
    git config --global alias.<nama shortcut> comand-git
    contoh: git config --global alias.ko commit
    contoh: git config --global alias.logone "log --oneline"

- Buat shortcut git secara sementara.
  alias nama-shortcut="command git"
  contoh: alias log="git log"

+ configurasi text editor default VScode
- 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"

+ Konfigurasi remote ke github
- untuk mengclone repo dari remote ke local)
  git clone <https repo pada github kalian> 'ini hanya berlaku ketika menggunakan https'

- Cara untuk kloning repo pada git local ke remote, tapi pada remote harus membuat repo yang nama folder sama dan pada file README tidak di centang.
  git remote add origin <https repo pada github>

- Untuk konfigurasi push ke git remote ke branch master , agar saat push tidak configurasi lagi.
  git push -u origin master

- Lihat branch pada remote.
  git remote

- Untuk melihat configurasi git pada remote)
  git remote -v

| Command untuk mengoperasikan git

> ls -l
(untuk melihat isi file .git)

> git init {harus masuk ke dalam folder yang akan di jadikan repo}
(untuk membuat folder menjadi repositori)

> git add name-file
(menambahkan file ke staging index)

> git checkout <branch>
(berpindah branch)

> git branch <nama branch>
(membuat branch baru)

> git status
(melihat status repositori)

> git commit - m "pesan"
(commit file ke repository)

> git log
(menampilkan hasil history commit)

> git log --oneline
(menampilkan hasil history commit dengan pendek)

> git log -- <nama file>
(melihat perubahan pada suatu file)

> git merge name-branch
- Menggabungkan branch.
- Proses merge ketika 2 branch (master/temp) lalu master merge ke temp, maka nama branch yang seterusnya adalah master bukan temp.

> git diff
(melihat perubahan yang terjadi pada file yang kita ubah)

> git clean -f
(membatalkan penambahan file yang ada di working directory)

> git branch --merged
(melihat branch yang sudah di merge)

> git branch -d <nama branch>
(menghapus branch)

> git branch -D <nama branch>
(menghapus branch walau belum di marge branch tersebut)

> git restore nama-file
- Pada working directory.
- Membatalkan perubahan / penghapusan file yang ada di working dirctory.

> git restore --staged nama-file (staging index)
(membatalkan penghapusan / perubahan yang ada di staged index)

> git log --oneline --graph
(menampilkan histori commit dengan grafik secara sederhana)

> git log --all --decorate --oneline --graph
(menampilkan histori commit dengan grafik yang saling terhubung)

> git show code-hash
(melihat perubahan yang terjadi pada commit tertentu)

> git diff hash1 hash2
(membandigkan commit1 dengan commit2, bukan membandingkan hasil dari commit1 sampai ke commit2, namun hanya membandingkan hasil perubahan commit1 dan hasil akhir commit2)

> git difftool hash1 hash2
(melihat perbandingan menggunakan VScode)

> git reset --mode hash
(membatalkan perubahan di commit dengan cara reset commit dengan mode)

mode
• --soft (memindahkan pointer HEAD namun tidak melakukan perubahan di staged index dan working directory)
• --mixed (default)
(memindahkan pointer HEAD dan mengubah staged index menjadi sama seperti repository namun tidak mengubah apapaun di working directory)
• --hard (memindahkan pointer HEAD dan mengubah staged index dan working directory menjadi sama seperti repository)

> git commit --amend -m "pesan"
(amend commit, alternatife dari reset commit dan melakukan perubahan lalu commit ulang, hal ini bisa di lakukan tampa manual, dengan amend commit, namun amend akan merubah code gash commit karena data bertambah)

> git checkout hash -- nama-file
(melihat versi file sebelumnya)

> git checkout branch/hash
(kembali ke versi yang kita inginkan)

> git blame nama-file
(melihat siapa yang mengubah file tersebut)

> git switch <name branch>
(untuk berganti branch)

_____
| Note

setiap perubahan itu harus di commit entah itu menghapus, mengedit, merename dll

tidak ada cara untuk membatalkan perubahan yang sudah di commit namun ada cara yaitu Rollback commit / Revert commit

operasi rename file dalam git adalah yaitu operasi penggabungan menghapus file lalu membuat file baru dengan isi yang sama namun dengan name-file yang berbeda, sehingga terdeteksi sebagai rename file

.gitignore file yang tidak di track oleh git, dan juga isi file nya menuju pada file yang tidak kita inginkan

Jenis-jenis merge

• Fast Forward Merge
• Three-way Merge
"Direct Path" > jalur langsung sebuah branch yang jika di merge akan menghasilkan fast forward merge

| Shortcut git

> git commit -am "pesan"
(cara cepat commit, hanya berlaku ketika file yang modifed)

> git add .
(menambahkan semua file dari working directory ke staging index)