Version Control dengan Git

Alat dan Teknik untuk Pengembangan Perangkat Lunak Kolaboratif

UB FILKOM FAKULTAS ILMU KOMPUTER

Kampus Merdeka INDONESIA JAYA

Sub-CPMK

- Mahasiswa mampu mengelola versi dari layanan secara efisien

Referensi

https://drive.google.com/file/d/1y8cLHLqACk8cTOi3rAmySjc5ZPI_7N7l/view?usp=sharing

Outline

- Pendahuluan
- Git
- Penggunaan GIT
- Branching & Merging
- Remote Repository
- Teknik Lanjutan
- Optimasi & Troubleshooting
- Penutup
  - Pentingnya Git dalam Tim
  - Tips Belajar & Referensi

Motivasi

- Backup dan Keamanan
- Eksperimen Tanpa Risiko
- Integrasi dengan Alat Modern
- Transparansi dan Akuntabilitas
- Skalabilitas Proyek
- Melacak Perubahan
- Kolaborasi Tim
- Pengelolaan Versi
- Pemulihan Kesalahan
- Dokumentasi Otomatis

Pendahuluan

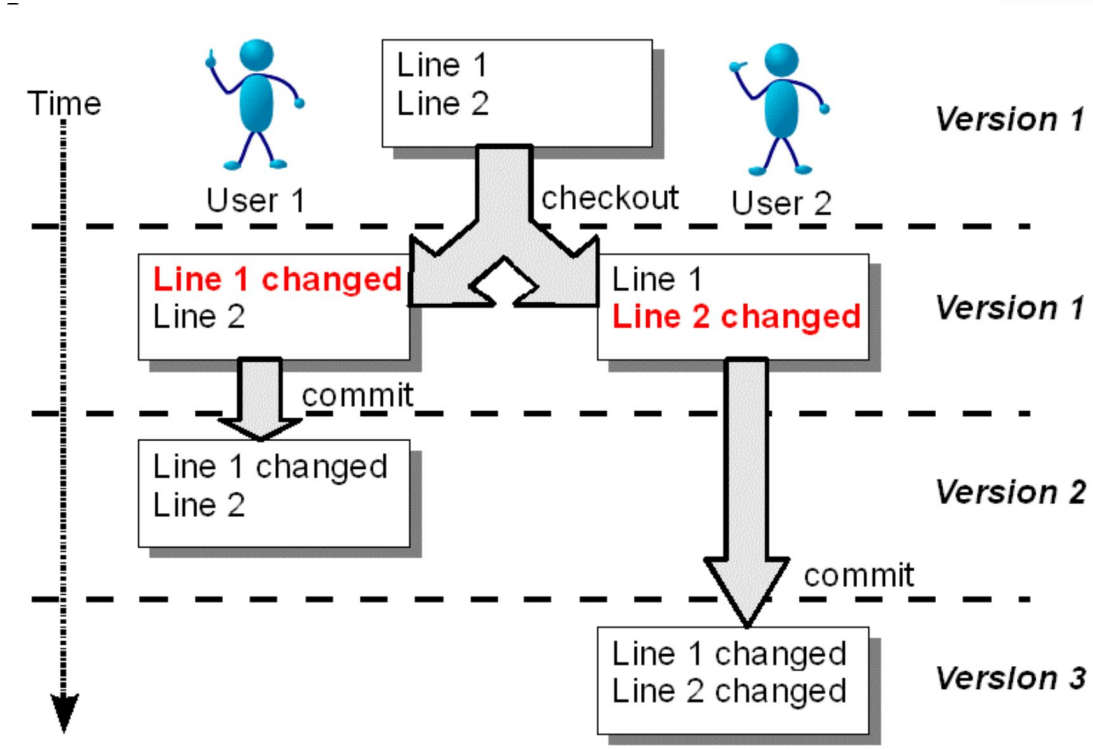


Figure 1: Example of a merge done successful

Pendahuluan

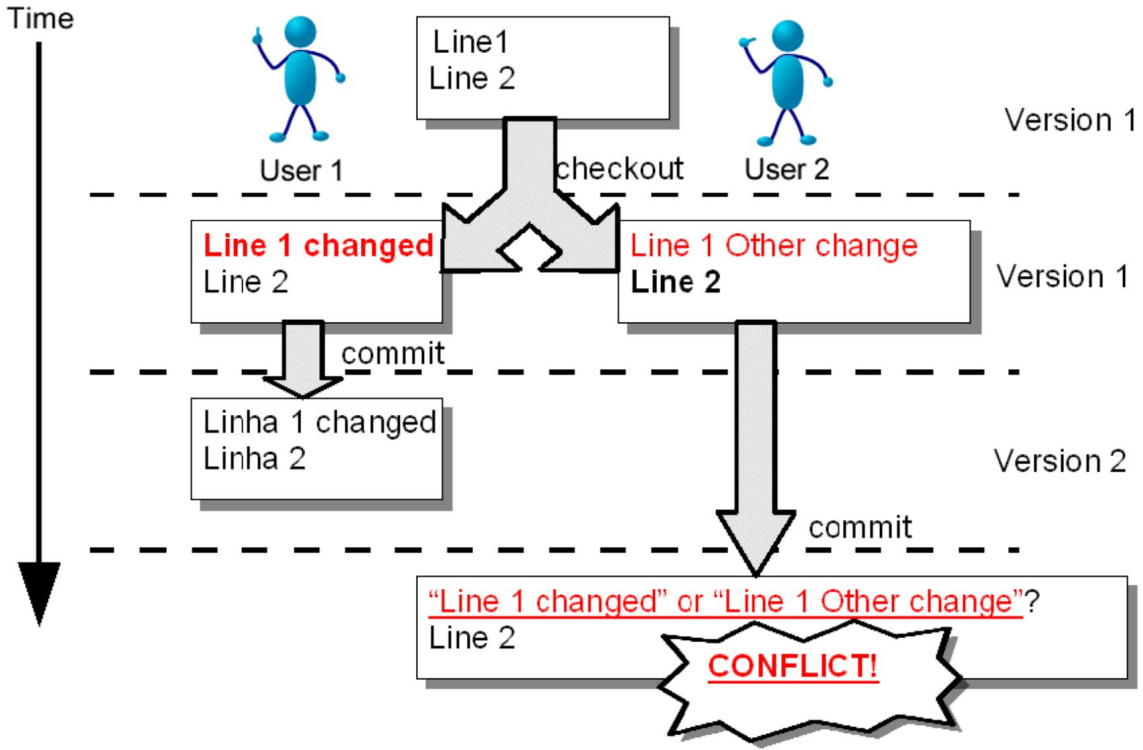


Figure 2: Example of a conflict during an update operation

Pendahuluan jenis-jenis version control

- Version Control Lokal
- Version Control Terpusat (Centralized)
- Version Control Terdistribusi (Distributed)

	Lokal	**Terpusat**	**Terdistribusi**
Kolaborasi	❌	✅	✅
Backup Data	❌	✅ (di server pusat)	✅ (di semua komputer)
Kecepatan Akses	⚡	🐢	⚡
Resiko Kehilangan Data	Tinggi	Sedang	Rendah
Contoh Tools	**Manual Copy**	**SVN, CVS**	**Git, Mercurial**

Apakah ada yang menggunakan Git?

Kenapa termasuk distributed?

Git

- Git adalah sistem version control terdistribusi (Distributed Version Control System) yang digunakan untuk melacak perubahan pada file atau proyek dari waktu ke waktu.
- Git dikembangkan oleh Linus Torvalds pada tahun 2005 .
- Awalnya dibuat untuk mengelola pengembangan kernel Linux.

Fungsi Utama :

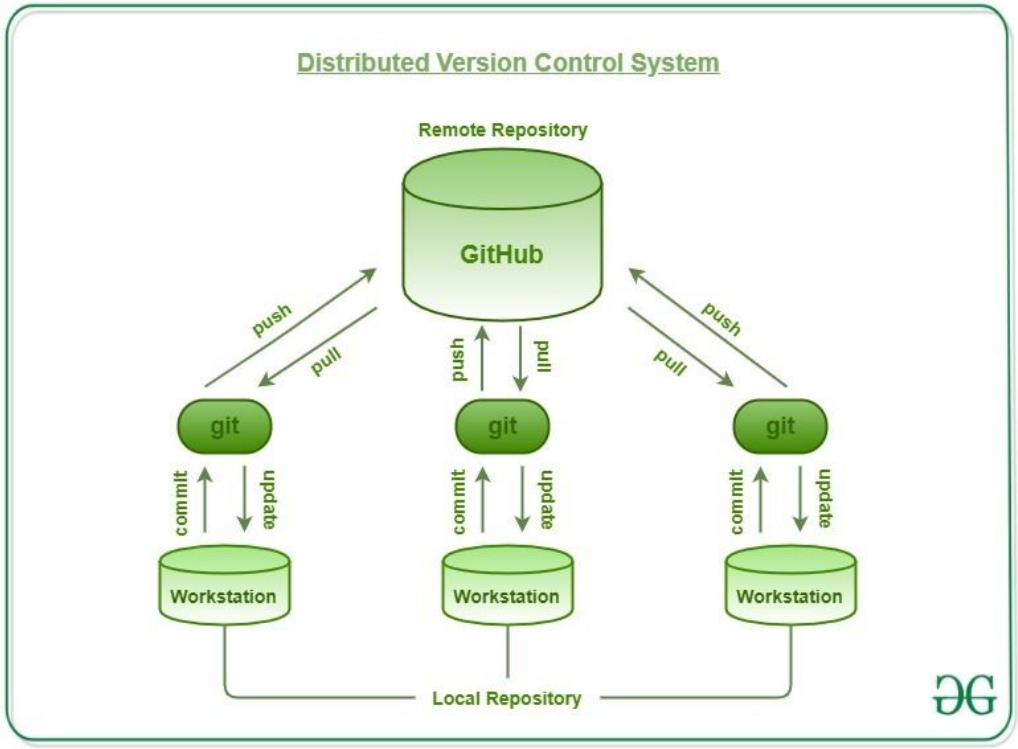
- Melacak sejarah perubahan kode.
- Memungkinkan kolaborasi tim dalam pengembangan perangkat lunak.
- Mengelola versi proyek secara efisien.

Git: motivasi

Mengapa Git Dibutuhkan?

- **Melacak Perubahan**: Mengetahui siapa yang membuat perubahan dan kapan.
- **Kolaborasi Tim:** Memudahkan kerja sama dalam tim besar.
- **Backup dan Keamanan:** Data tersimpan di banyak tempat (distributed).
- **Eksperimen Tanpa Risiko:** Branching memungkinkan eksperimen tanpa merusak kode utama.
- **Skalabilitas:** Mendukung proyek dari skala kecil hingga besar.

GIT: usecase


Distributed Version Control System
Remote Repository
GitHub
push
pull
push
pull
push
pull
git
git
git
commit
update
commit
update
commit
update
Workstation
Workstation
Workstation
Local Repository

Git: Components

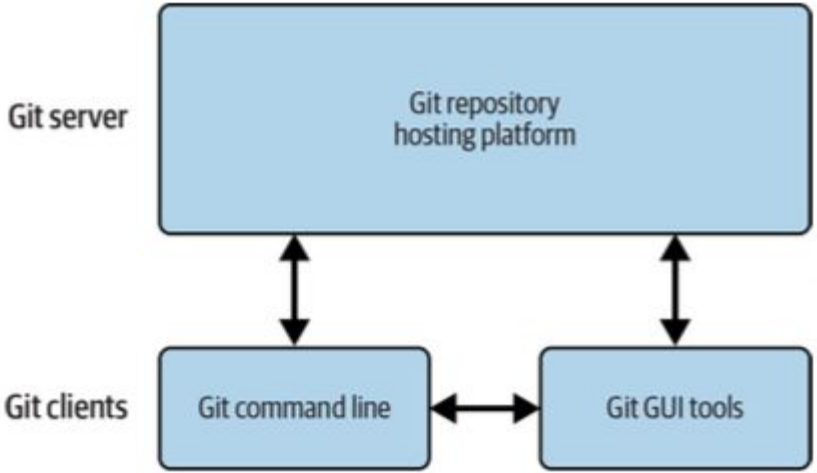


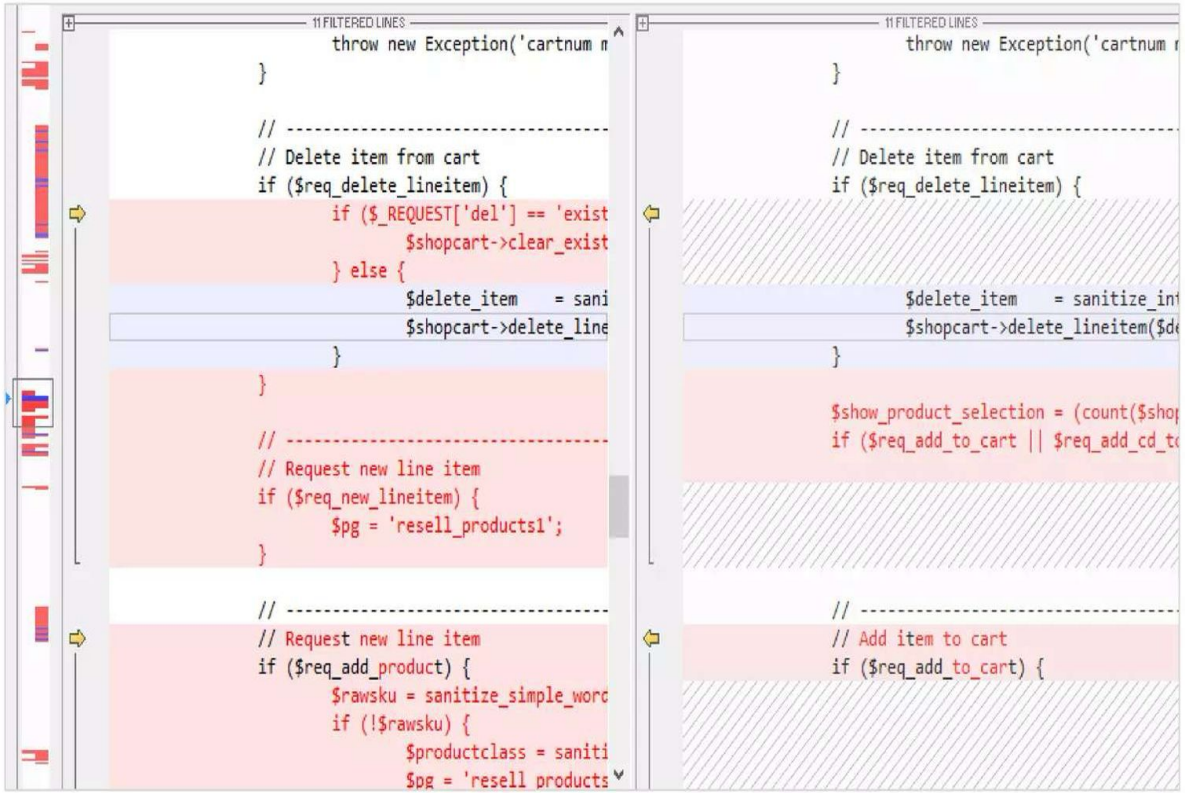
Figure 1-1. Overview of Git components

Operasi dasar (general)

- Add: Place a file under version control
- Check in: Send local changes to the repo
- Check out: Download from a repo to your working copy
- Ignore: Allows files to exist in your working copy but not in the repo
- Revert: Throw away your working copy and restore last revision
- Update / Sync: Update your working copy to the latest revision

- Diff / Change: Specific modification to a document
- Branch: Duplicate copy of code used for feature development
- Merge: Integrate changes from two different branches
- Conflict: Inability to reconcile changes to a document
- Resolve: Manual fixing of conflicted document changes
- Locking: Prevents other developers from making changes

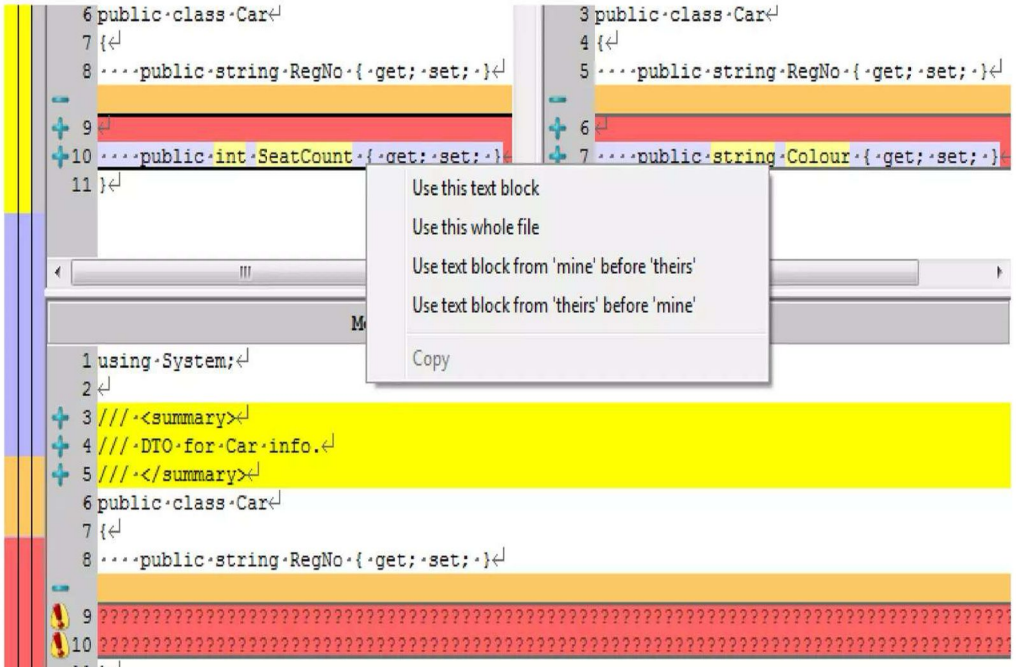
Diff/change/delta



Conflic

```
5   */
6   public class SingerABCD {
7       static final String kSong =
8                   "Row, row, row your raft,\n"
<<<<<<< .mine
10                  + "Swiftly down the stream,\n"
11  =======
12                  + "Gently down the river,\n"
>>>>>>> .r38
14                  + "Merrily, merrily, merrily, merrily,\n"
15                  + "Life is but a dream.\n";
16      public static void main(String[] args)
```

Resolve


public class Car
{
public string RegNo { get; set; }
public int SeatCount { get; set; }
}
public class Car
{
public string RegNo { get; set; }
public string Colour { get; set; }
Use this text block
Use this whole file
Use text block from 'mine' before 'theirs'
Use text block from 'theirs' before 'mine'
Copy
using System;
/// <summary>
/// DTO for Car info.
/// </summary>
public class Car
{
public string RegNo { get; set; }

Penggunaan GIT

Intalasi:

- Instalasi Git di Windows
  - Langkah 1 : Unduh Installer
    - Kunjungi https://git-scm.com .
    - Pilih versi terbaru untuk Windows.
  - Langkah 2 : Jalankan Installer
  - Ikuti wizard instalasi:
    - Pilih lokasi instalasi.
    - Pilih editor default (misalnya Notepad++ atau VS Code).
    - Aktifkan opsi "Git Bash Here" untuk integrasi dengan File Explorer.

- Instalasi Git di Linux
  - Untuk Ubuntu/Debian :
  - Buka Terminal dan jalankan:
    - sudo apt update
    - sudo apt install git
- Validasi instalasi:

Konfigurasi GIT

- git config --global user.name "Nama Anda"
- git config --global user.email "email@contoh.com"

Validasi:

- git config --global user.name
- git config --global user.email

Perintah dasar GIT

```
These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects
```

Mengelola Proyek GIT

- Buat repository pada github website
- create a new repository on the command line
  - echo "# ML-based-TLSIDS" >> README.md
  - git init
  - git add README.md
  - git commit -m "first commit"
  - git branch -M main
  - git remote add origin git@github.com:Widhi-yahya/ML-based-TLSIDS.git
  - git push -u origin main
- push an existing repository from the command line
  - git remote add origin git@github.com:Widhi-yahya/ML-based-TLSIDS.git
  - git branch -M main
  - git push -u origin main

Kasus 1: Menambahkan Proyek MK pada GIT

- Masuk ke folder proyek:
  - cd /path/to/your/current/folder
- Inisialisasi Git repo:
  - git init
- Tambahkan file pada repo:
  - git add . (. untuk semua file)
- Commit perubahan
  - git commit -m "Initial commit"
- Tambahkan remote repository:
  - git remote add origin https://github.com/your-username/your-repository.git
- Push perubahan ke Github
  - git push -u origin main

Penjelasan

Git init → create hidden .git file

Pada contoh disini dibuat pada folder my_website, sebagai working directory

```
.
└── my_website
    ├── .git/
    │   └── Hidden git objects
    └── index.html
```

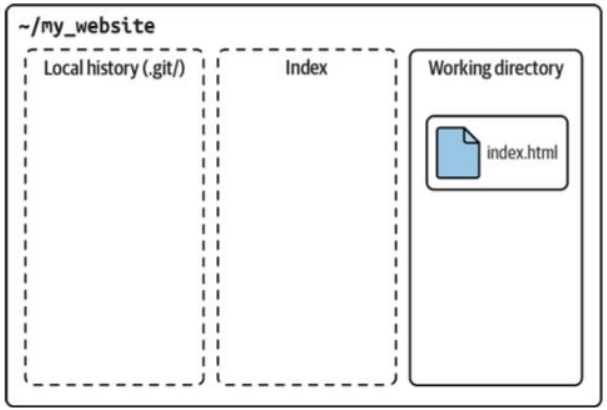
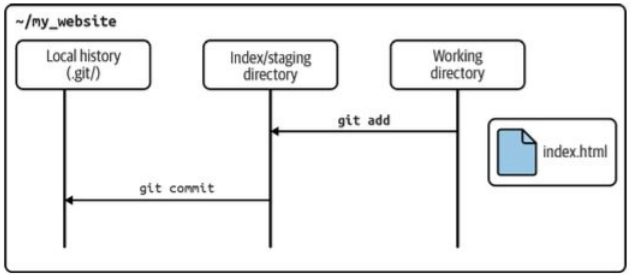
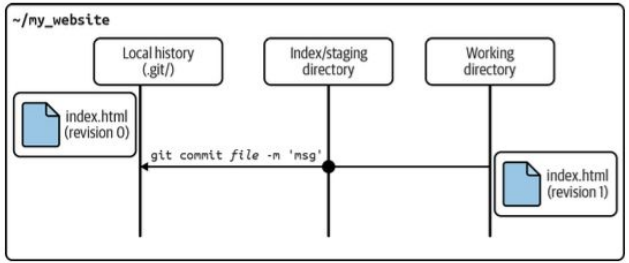


Figure 1-2. Initial repository





Commit terkait update

- Menyelesaikan 2 bug: gunakan 2 commit
- Small commit:
  - Mudah dipahami
  - Mudah roll back
  - Lebih sedikit konflik

Commit/ sering update

Sama seperti sebelumnya.

- Mudah dipahami
- Mudah roll back
- Lebih sedikit konflik

Do not commit

- Broken code
- Half done
  - Breakdown into smaller tasks
- Belum teruji

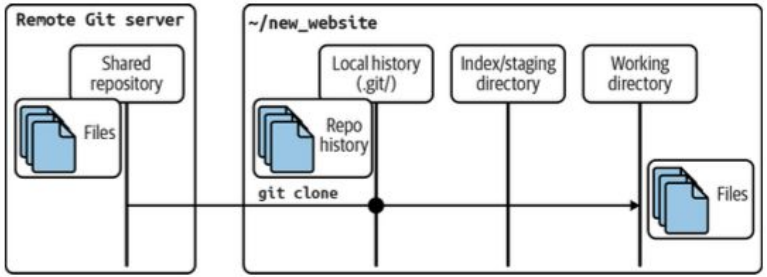
Commit

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

LK 1: Perubahan pada file

1. Ubah file yang sudah dibuat
2. Update repository
3. Lihat log dari git commit

Bekerja dengan Shared Repo: Kolaborasi




```
$ cd ~

$ git clone https://git-hosted-server.com/some-dir/my_website.git new_website
Cloning into 'new_website'...
remote: Enumerating objects: 2, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (103/103), done.
remote: Total 125 (delta 45), reused 65 (delta 18), pack-reused 0
Receiving objects: 100% (125/125), 1.67 MiB | 4.03 MiB/s, done.
Resolving deltas: 100% (45/45), done.
```

Kolaborasi: Tipe

- **Collaborators:** Pemilik repository dapat menambahkan collaborators dengan akses langsung ke repository.Collaborators dapat meng-clone, commit, dan push perubahan secara langsung.
- **Forking and Pull Requests:** Pengguna dapat membuat salinan repository (fork) di akun mereka sendiri. Setelah melakukan perubahan, pengguna dapat mengajukan pull request (PR) ke repository asli.
- **Branch Protection and Pull Requests:** Pemilik repository dapat mengatur branch protection rules untuk memastikan semua perubahan direview sebelum digabungkan. Aturan ini sering digunakan untuk branch utama seperti main atau develop.

Kolaborasi: Forking dan Pull Request

- Fork adalah proses membuat salinan penuh (copy) dari repository milik orang lain ke akun GitHub Anda sendiri.
- Klik “fork” pada halaman Github utama
- Clone seperti biasa:
  - git clone https://github.com/your-username/your-forked-repository.git
  - cd your-forked-repository

Kolaborasi: Branch Protection and Pull Requests

Branch adalah cabang dalam repository yang digunakan untuk mengisolasi pekerjaan pada fitur, bugfix, atau eksperimen tanpa memengaruhi branch utama (biasanya main atau develop).

Motivasi:
Mencerminkan tahapan pengembangan proyek (stabil, pengembangan, rilis).

- Mendukung manajemen versi produk (versi lama vs. baru).
- Mengisolasi fitur atau perbaikan bug agar tidak mengganggu kode utama.
- Mendukung kolaborasi tim dalam proye
- Menjaga sinkronisasi antar repository m

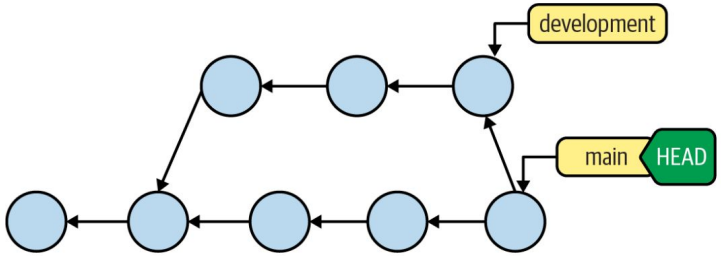


Figure 3-1. Commit graph with branches

LK1: Praktik Penggunaan Branch

Buku:
https://drive.google.com/file/d/1y8cLHLqACk8cTOi3rAmySjc5ZPI_7N7l/view?usp=sharing
Halaman: 61-79

Coba jalankan perintah dan jelaskan fungsi masing-masing operasi berikut:

1. Buat branches
2. List
3. Lihat branch dan lihat commit
4. Switch/checking out
5. Merging
6. Delete

End of slides..