Panduan Sederhana Setting-Up Knowledge Managemnt Sytems

Git, Github, Quarto, Obsidian, dan Zotero

Armein Z R Langi

2025-02-13

Table of contents

# Pendahuluan

## File, Directory, Repository, Vault

Di dunia digital kita menghasilkan FILE, yang berisikan entah data biner atau data teks (text). Data teks sebenarnta juga data biner, tetapi dibatasio oleg kode teks seperti ASCII atau UTF-8. Fiile data biner dan teks untuk dibaca mesin, sedangkan data teks biasanya untuk di baca juga oleh manUsia.

Proses yang kita lakukan pada level paling dasar adalah mengubah file input menjadi file output. Atau file versi lama menjadi file versi baru. Sedemikian penting file itu śehingga kita menyimpanya ke dalam sebuah repositori atay “lebari besi” vault. Hampir selalu ada lebih dari satu files, sehinnga kita membri nama atau label pada file itu. Nama file adalah data teks, sehungga bisa dibaca manusia. Nama juga digunakan sebagi entry daro daftar isu (directoy. Sebuah directiory berisikan pohon dari folfers (Map), dan sebuah fiolder berisikan folder lain atu file.

## Github: Bank Repositori

Seperti uang yang lebih aman dan berguna disimpan, repository kiat bisa direpliksi di Github, dimana isi repository lokal sama dengan repository naster g=di Guthub. Dalam kondisi yang tidak kita harpkan, komputer kita rysa atau hilabg, kita bisa dsenfab nudah mereplikasi repository kiat di komputrer ayng baru, dengan mrngclone reoisutir ynag d=tersu=imaon di Bnak GIthub

## Git GithuB dan KNowlegdde nanagemen

Dalam berkreasi menghasilkan tulisan, paper, disertasi, konten, software kita sering menghadapi kebingungan, kelelahan, dan kebuntuan akibat sifat kompleksitas yang tinggi.

Berbagai alat bantu digital telah sangat menolong, tetapi ancaman kompleksitas selalu mengancam. Akibatnya kalaupun tugas ini selesai, kualitas tidak maksimal, waktu terlalu lama, dan biaya yang dikeluarkan membengkak.

Jalan keluarnya adalah dengan menguraikan pekerjaan ke dalam potongan, tahapan, untuk dikrjakan sendiri atau parlelisasi.

Git dan Github alat digital yang sangat berguna untuk mengelola berbagai potongan pekerjaan dan mendistribusinya pada banyak orang, dengan tetap menjaga integritas proyek.

Git mengendalikan pnguraian pekerjaan ke dalam potongan-potongan penugasan, mencatat siapa yang mengambil potongan pekerjaan tertentu, lalu memantau penyelesaiainya. Pekerjaan juga dapat diurai ke dalam versi-versi, sehingga sebelum pengembangan versi baru selesai versi lama tetap tersedia.

Git mengenal tiga tempat penyimpanan: lokal repository, staging, dan working space.

Adapun Github menjadi pusat distribusi, sehingga Github menyimpan repositori utama (master atau main) dan setiap individu kontributor memiliki replikasi dari repositori master, yaitu replikasi lokal.

Sebagaimana diperlihatkan pada Gambar [Figure 1](#fig-git-github), Github menjdi pusat dari banyak kontributor dengan repositori lokal masing masing. Git dan Github bekerjasama mengatur potongan mana yang dikerjakan kontributor tertentu, dan mencegah tumpang tindih atau pengerjaan bagian sama pada saat yang sama oleh kntributor yang berbeda.

|  |
| --- |
| Figure 1: Git dan Github |

Dari gambar ini, dapat dimengerti bahwa kontributor bekerja di working space, yang bila hasinya memuaskan, di kirim ke staging area (yang biasanya hidden dari kontrbutor ) untuk menunggu keputusan apakah hendak di commit ke local repository. Untuk proyek personal, proses berhenti cukup di sini

Bila diputuskan untuk untuk masuk ke repository utama, maka proses push diinisiasi, yang biasanya diterima kecuali bisa ada anomali seperti duplikasi kntributor tadi.

Mengingat pentingnya mengkombinasi personalisasi kontribusi (git) dan kolaborasi (Github) maka dipandang perlu akadmisi memanfaatkan alat bantu ini. Untuk itu tulisan ini dibuat.

Proses pembuatan tulisan ini menggunakan Git dan sedikit Github. Tulisan ini dibuat menggunakan texteditor, dengan format markdwon versi Quarto. This is a book created from markdown and executable code.

## Tejks, Markdown, Quarto

pada akhirnya isi file harus di jonsumsi pengguna. Ada 3 bentuk pernyajian yang umumnya dituntut orang: web pages (untuk dikonsumsi di komputer, PDF (untuk dibagikan dan dikonsmunsi melalui layar atau printer), atau docx untuk diedit swlanjutnya nenggunakan word processor

Drngan menmbahkan kode-kode matking pasda file .Txt, dokumen berubah menh=jai .md.  
kendian bila ditambahjsn atribut yaml, maka foile berubah mnjadi .qmd yabg dunakan Quarto.

Tugas utama Quarto adalah merender file teks markdown .qmd menjsdi pageweb, pdf, docs, serta format lainya seperti latex

# obsidian dan Zotero, dan Quarto

Onsidian adalh aplikasi serta paltfoem untuk menghasilak file yang bernialai tinggi. Dalam bentuk paling sederhana kita daoa memilih seatu fold dalam repository untuk di jadikam vaukt (ruang penyimpn harta). Sekali vaultr itu dibuat, obisidian bsai membuat file .md unrtuk membuta f]konten, seklain itu, file tersebut dapat mengndung btabwk,gambar, persamaan matematika sehingga dapa digunkan untuk menimpan pengethuan yang berharga/

Kualitas file .md dapat ditingkatkan karena Qbisidan memiliki mekaniskme untuk membuat ,ind map (graph yang menbghubungkan isi suatu file dengan isi file lain), sehingga penulis mendapatkan ganmabran relasional yang lebih lengkao.

Dengan sebuah ekstensi (program tambahan) di Obsidian, Obsidian mampu melakukan hal yang sama pad dfile .qmd. File qmd dsaoat mnenggabung kan potongan pengetahuan untk manusia dasn potongna kode software yang daopat dieksekusikomputer dan menayangkan hasdilnya. Quarto merealisasikan konsep literate programming, yang menjadikan penulisan oaoper dan koding dilakuan bersamaabn

See Knuth (1984) for additional discussion of literate programming.

Bandung, 13 Februari 2025

>Armein Z R Langi

# 1. Git dan Github

### 1.0.1 📌 **Konsep Dasar Git**

**Git** adalah sistem kontrol versi (VCS) terdistribusi yang digunakan untuk melacak perubahan dalam kode, memungkinkan kolaborasi, dan mengelola proyek perangkat lunak dengan lebih efisien.

## 1.1 🔹 **1. Apa Itu Git?**

Git adalah alat yang memungkinkan: - **Versi Kontrol**: Melacak perubahan kode dari waktu ke waktu. - **Kolaborasi**: Memungkinkan banyak orang bekerja di proyek yang sama tanpa konflik. - **Distribusi**: Setiap pengembang memiliki salinan lengkap dari repositori.

## 1.2 🔹 **2. Arsitektur & Cara Kerja Git**

Git memiliki tiga area utama:

| **Area** | **Fungsi** |
| --- | --- |
| **Working Directory** | Tempat di mana file saat ini diedit. |
| **Staging Area** | Area persiapan sebelum file dikomit. |
| **Repository (Local Repo)** | Tempat penyimpanan perubahan yang telah dikomit. |
| **Remote Repository** | Repositori yang disimpan di server (GitHub, GitLab, dll.). |

⚡ **Alur Kerja Git:** 1. **Edit File** → di **Working Directory**. 2. **Tambah ke Staging Area** → git add file.txt 3. **Commit ke Repository** → git commit -m "Pesan commit" 4. **Push ke Remote Repo** → git push origin main

## 1.3 🔹 **3. Perintah Dasar Git**

| Perintah | Fungsi |
| --- | --- |
| git init | Membuat repositori Git baru. |
| git clone URL | Mengunduh repositori dari remote. |
| git add file.txt | Menambahkan file ke staging area. |
| git commit -m "Pesan" | Menyimpan perubahan ke repositori lokal. |
| git push origin main | Mengunggah perubahan ke repositori remote. |
| git pull origin main | Mengambil perubahan terbaru dari remote. |
| git status | Melihat status perubahan file. |
| git log | Melihat riwayat commit. |
| git branch feature-X | Membuat branch baru. |
| git checkout feature-X | Berpindah ke branch lain. |
| git merge feature-X | Menggabungkan branch ke branch utama. |

## 1.4 🔹 **4. Branching & Merging**

Git memungkinkan pengembang bekerja di fitur yang berbeda tanpa mengganggu kode utama.

1. **Membuat branch baru** → git branch feature-1
2. **Beralih ke branch** → git checkout feature-1
3. **Edit & commit perubahan** → git commit -m "Tambah fitur"
4. **Kembali ke branch utama** → git checkout main
5. **Gabungkan branch** → git merge feature-1

**🔀 Git Flow Umum:** - **main** → Branch stabil (production). - **develop** → Branch pengembangan utama. - **Feature Branches** → Untuk fitur baru, lalu digabungkan ke develop.

## 1.5 🔹 **5. Remote Repository (GitHub, GitLab, Bitbucket)**

Git dapat terhubung ke layanan cloud seperti **GitHub** untuk kolaborasi.

1. **Tambahkan Remote Repo**:

* git remote add origin https://github.com/user/repo.git

1. **Push kode ke GitHub**:

* git push -u origin main

1. **Ambil perubahan dari GitHub**:

* git pull origin main

## 1.6 🎯 **Kesimpulan**

* **Git membantu dalam pengelolaan kode secara terstruktur.**
* **Memungkinkan banyak pengembang bekerja di proyek yang sama.**
* **Branching mempermudah pengembangan fitur tanpa mengganggu kode utama.**
* **Remote repository (seperti GitHub) membuat kolaborasi lebih mudah.**

➡ **Ada bagian yang ingin kamu eksplor lebih dalam?** 🚀

# 2. Contoh Praktis

### 2.0.1 📌 **Contoh Realistis Proyek Pembuatan Software di Linux dengan Git**

Kita akan membuat proyek software sederhana bernama **“MyApp”**, sebuah program CLI yang ditulis dalam **Python**. Proyek ini akan menggunakan **Git** untuk mengelola versi dan berkolaborasi.

## 2.1 🔹 **1. Inisialisasi Proyek**

### 2.1.1 🔥 **Langkah 1: Buat Direktori Proyek**

Di terminal Linux, jalankan:

mkdir MyApp  
cd MyApp

### 2.1.2 🔥 **Langkah 2: Inisialisasi Git**

git init

📌 **Git akan membuat repositori kosong di dalam folder proyek.**

## 2.2 🔹 **2. Menambahkan File Awal**

### 2.2.1 🔥 **Langkah 3: Buat File Program Utama**

Buat file app.py:

touch app.py

Edit app.py:

print("Hello, this is MyApp!")

### 2.2.2 🔥 **Langkah 4: Tambahkan README**

echo "# MyApp" > README.md

### 2.2.3 🔥 **Langkah 5: Tambahkan File ke Git**

git add app.py README.md

📌 **Perintah git add menambahkan file ke Staging Area.**

### 2.2.4 🔥 **Langkah 6: Commit Perubahan**

git commit -m "Inisialisasi proyek dengan file utama dan README"

📌 **Perintah git commit menyimpan perubahan ke dalam riwayat versi.**

## 2.3 🔹 **3. Menambahkan Remote Repository (GitHub)**

Sekarang, kita ingin menyimpan proyek ini di GitHub.

### 2.3.1 🔥 **Langkah 7: Tambahkan Remote Repository**

git remote add origin https://github.com/username/MyApp.git

📌 **Gantilah username dengan akun GitHub kamu.**

### 2.3.2 🔥 **Langkah 8: Push Kode ke GitHub**

git push -u origin main

📌 **Kode pertama kali diunggah ke GitHub pada branch main.**

## 2.4 🔹 **4. Pengembangan Fitur Baru dengan Branch**

Misalnya, kita ingin menambahkan fitur baru: **menampilkan waktu saat ini**.

### 2.4.1 🔥 **Langkah 9: Buat dan Beralih ke Branch Baru**

git branch feature-time  
git checkout feature-time

📌 **Branch feature-time dibuat untuk pengembangan fitur tanpa mengganggu branch utama.**

### 2.4.2 🔥 **Langkah 10: Edit app.py untuk Menampilkan Waktu**

Edit app.py:

import datetime  
  
print("Hello, this is MyApp!")  
print("Current Time:", datetime.datetime.now())

### 2.4.3 🔥 **Langkah 11: Commit Perubahan di Branch Fitur**

git add app.py  
git commit -m "Menambahkan fitur waktu ke MyApp"

## 2.5 🔹 **5. Menggabungkan Fitur ke main**

Setelah fitur selesai diuji, kita gabungkan ke branch main.

### 2.5.1 🔥 **Langkah 12: Pindah ke Branch main**

git checkout main

### 2.5.2 🔥 **Langkah 13: Gabungkan Fitur**

git merge feature-time

📌 **Branch feature-time digabungkan ke main.**

### 2.5.3 🔥 **Langkah 14: Push Perubahan ke GitHub**

git push origin main

📌 **Kode terbaru sekarang ada di GitHub.**

## 2.6 🔹 **6. Bekerja dengan Tim (Pull Request dan Merge)**

Jika bekerja dalam tim, biasanya: 1. **Developer lain membuat branch fitur baru dan push ke GitHub**: bash git push origin feature-time 2. **Di GitHub, developer membuat Pull Request** dan meminta merge ke main. 3. **Tim melakukan review kode** dan menyetujui merge. 4. **Merge dilakukan di GitHub atau melalui terminal:** bash git checkout main git pull origin main

## 2.7 🔹 **7. Mengelola Bug dan Revisi**

Misalnya, kita menemukan bug di app.py.

### 2.7.1 🔥 **Langkah 15: Buat Branch untuk Perbaikan Bug**

git branch fix-bug  
git checkout fix-bug

Edit app.py untuk memperbaiki bug, lalu:

git add app.py  
git commit -m "Memperbaiki bug dalam fitur waktu"  
git checkout main  
git merge fix-bug  
git push origin main

## 2.8 🔹 **8. Membuat Versi Rilis**

Setelah pengembangan stabil, kita bisa membuat **versi rilis**.

### 2.8.1 🔥 **Langkah 16: Buat Tag untuk Rilis**

git tag -a v1.0 -m "Rilis pertama MyApp"  
git push origin v1.0

📌 **Git akan menandai versi v1.0 untuk rilis.**

## 2.9 🎯 **Kesimpulan**

1. **Gunakan Git untuk mengelola proyek dengan aman.**
2. **Gunakan branch untuk pengembangan fitur dan perbaikan bug.**
3. **Simpan proyek di GitHub untuk kolaborasi dan backup.**
4. **Gunakan tagging (git tag) untuk merilis versi stabil.**

➡ **Mau coba proyek lain atau ada pertanyaan?** 🚀# Summary

In summary, this book has no content whatsoever.

# References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. <https://doi.org/10.1093/comjnl/27.2.97>.