Panduan
Git dan Github

1. Pesan Commit

Menulis pesan commit yang baik adalah praktik penting dalam pengembangan perangkat lunak. Pesan commit yang jelas dan konsisten membantu:

- Memahami Sejarah Proyek
  Memudahkan siapa pun (termasuk diri Kita di masa depan) untuk melihat perubahan apa yang terjadi, kapan, dan mengapa.
- Debugging dan Investigasi
  Mempercepat proses mencari tahu kapan bug atau masalah muncul.
- Code Review
  Mempermudah reviewer untuk memahami konteks perubahan.
- Pembuatan Catatan Rilis (Release Notes)
  Pesan commit yang terstruktur dapat digunakan secara otomatis untuk menghasilkan catatan rilis.
- Kolaborasi Tim
  Membangun komunikasi yang lebih baik dalam tim.

Berikut adalah aturan dan panduan untuk menulis pesan commit yang profesional.

1.1. Struktur Pesan Commit

Pesan commit idealnya memiliki dua bagian utama:

a. Baris Subjek (Subject Line)
   Baris pertama yang memberikan ringkasan singkat perubahan.
b. Body Pesan (Body)
   Bagian detail yang menjelaskan konteks, motivasi, atau detail implementasi (jika diperlukan).

Kedua bagian ini dipisahkan oleh satu baris kosong.

```
Ini baris subjek yang ringkas dan jelas

(Baris kosong di sini)

Ini adalah bagian body pesan. Di sini Kita bisa menjelaskan
motivasi
di balik perubahan, masalah apa yang diselesaikan, atau detail
implementasi yang penting. Panjang baris di body sebaiknya
dibatasi
agar mudah dibaca di terminal (misalnya sekitar 72 karakter).

Kita bisa menggunakan poin-poin jika ada beberapa hal yang
perlu
dijelaskan:

* Jelaskan poin pertama
* Jelaskan poin kedua
* Jelaskan poin ketiga

Kita juga bisa menyebutkan isu atau pull request terkait,
misalnya: Menutup isu #123, Lihat PR #456.
```

1.2. Aturan untuk Baris Subjek (Subject Line)

Baris subjek adalah bagian terpenting karena seringkali ini yang pertama kali dilihat.

a. Singkat dan Padat:

   Batasi panjang baris subjek, idealnya tidak lebih dari 50 karakter. Ini penting agar baris subjek tidak terpotong saat ditampilkan di berbagai alat Git (misalnya, git log --oneline).

b. Gunakan Imperatif (Perintah)

   Tulis baris subjek seolah-olah Kita sedang memberikan perintah. Contoh:

   - Benar: "Tambahkan fitur login", "Perbaiki bug validasi email", "Perbarui dokumentasi README".
   - Salah: "Menambahkan fitur login", "Memperbaiki bug validasi email", "Updated README documentation".

c. Huruf Kapital di Awal

   Mulai baris subjek dengan huruf kapital.

   - Benar: Tambahkan fitur…
   - Salah: tambahkan fitur...

d. Tanpa Tanda Baca di Akhir

   Jangan akhiri baris subjek dengan titik (.).

   - Benar: Perbaiki masalah konfigurasi
   - Salah: Perbaiki masalah konfigurasi.

e. Jelaskan Apa Perubahannya

   Baris subjek harus dengan jelas menyatakan apa yang dilakukan oleh commit ini.

1.3. Aturan untuk Body Pesan (Body)

Bagian body bersifat opsional tetapi sangat dianjurkan untuk perubahan yang memerlukan penjelasan lebih lanjut.

a. Pisahkan dengan Baris Kosong

   Pastikan ada satu baris kosong antara baris subjek dan body. Ini penting agar Git memproses baris pertama sebagai subjek dan sisanya sebagai body.

b. Bungkus Baris (Wrap Lines)

   Batasi panjang setiap baris di body, praktik umum adalah sekitar 72 karakter. Ini membuat pesan mudah dibaca di berbagai ukuran terminal.

c. Jelaskan Mengapa dan Bagaimana

   Gunakan body untuk menjelaskan motivasi di balik perubahan

(mengapa Kita melakukan perubahan ini) dan memberikan konteks tambahan. Kita juga bisa menjelaskan bagaimana Kita menyelesaikan masalah jika implementasinya kompleks. Jangan hanya mengulang apa yang sudah jelas dari kode itu sendiri.

d. Sertakan Detail Relevan

   Jika ada detail penting tentang perubahan, seperti asumsi yang dibuat, trade-off, atau dampak potensial, sertakan di sini.

e. Sertakan Referensi

   Jika commit ini terkait dengan isu di issue tracker atau pull request lain, sebutkan referensinya di body (misalnya, Menutup isu #123, Lihat PR #456).

1.4. Menggunakan Konvensi Pesan Commit

Untuk konsistensi tim dan potensi otomatisasi, banyak proyek mengadopsi konvensi pesan commit. Salah satu yang paling populer adalah *Conventional Commits*. Konvensi ini menambahkan struktur pada baris subjek:

Konvensi Standar

type(scope?): description

- type: Kategori perubahan (wajib). Contoh umum:
  - feat: Menambahkan fitur baru.
  - fix: Memperbaiki bug.
  - docs: Perubahan pada dokumentasi saja.
  - style: Perubahan format atau gaya kode (spasi, penamaan, dll.).
  - refactor: Memperbaiki struktur kode tanpa mengubah fungsionalitas.
  - test: Menambahkan atau memperbaiki tes.
  - chore: Perubahan pemeliharaan atau rutin (update dependensi, konfigurasi).
- scope (opsional): Menunjukkan bagian spesifik dari proyek yang terpengaruh (misalnya, (api), (ui), (database)). Gunakan tanda kurung ().
- description: Baris subjek seperti yang dijelaskan di atas (imperatif, huruf kapital awal, ringkas).

Konvensi Standar Indonesia

tipe(lingkup?): deskripsi

- tipe: Kategori perubahan (wajib). Contoh umum:
  - fitur: Menambahkan fitur baru.
  - perbaikan: Memperbaiki bug.
  - dok: Perubahan pada dokumentasi saja.

    - gaya: Perubahan format atau gaya kode (spasi, penamaan, semikolon, dll., yang tidak memengaruhi makna kode).
    - refaktor: Memperbaiki struktur kode tanpa mengubah fungsionalitas. Perubahan kode yang tidak menambah fitur atau memperbaiki bug.
    - tes: Menambahkan atau memperbaiki tes.
    - rutin: Perubahan pemeliharaan atau rutin (update dependensi, konfigurasi).
    - ci: Perubahan pada file dan skrip konfigurasi CI (Continuous Integration).
    - build: Perubahan yang mempengaruhi sistem build atau dependensi eksternal.
    - kinerja: Perubahan kode yang meningkatkan kinerja.
- lingkup (opsional): Menunjukkan bagian spesifik dari proyek yang terpengaruh (misalnya, (api), (ui), (database)). Gunakan tanda kurung ().
- deskriptif: Baris subjek seperti yang dijelaskan di atas (imperatif, huruf kapital awal, ringkas).

Contoh Lengkap dengan Konvensi dalam Bahasa Indonesia:

```
feat(profil): Tambahkan opsi ubah foto profil

Menambahkan fungsionalitas yang memungkinkan pengguna mengunggah
dan mengganti foto profil mereka melalui halaman pengaturan akun.

Implementasi menggunakan API unggah gambar baru dan menyimpan foto
di direktori penyimpanan publik. Proses validasi ukuran dan format
gambar juga ditambahkan.

Menutup isu #201
```

```
fitur(pengguna): Tambahkan opsi ganti bahasa

Menambahkan fungsionalitas yang memungkinkan pengguna memilih bahasa
antarmuka aplikasi (saat ini mendukung Bahasa Indonesia dan
Inggris).

Pilihan bahasa disimpan di preferensi pengguna dan diterapkan
secara dinamis saat aplikasi dimuat ulang.

Menutup isu #78
```

```
perbaikan: Perbaiki bug form validasi tanggal

Validasi form tanggal tidak berfungsi dengan benar pada browser
Firefox.

Masalah terletak pada format parsing tanggal di JavaScript.
```

```
Mengubah format menjadi 'YYYY-MM-DD' menyelesaikan masalah ini
di semua browser utama.
```

Berikut adalah contoh cara menulis pesan commit Git melalui Command Prompt/Terminal, menggunakan dua metode yang umum:

Metode 1: Menggunakan Flag `-m` (Untuk Pesan Singkat)

Ini adalah cara cepat untuk menulis pesan commit satu baris (biasanya hanya baris subjek). Cocok untuk perubahan kecil dan sederhana yang tidak memerlukan penjelasan detail.

```
git commit -m "feat: Tambahkan tombol simpan pada form"
```

Penjelasan:

git commit: Perintah untuk membuat commit baru.

-m "...": Flag -m (message) diikuti oleh pesan commit di dalam tanda kutip ganda ("). Pesan di sini akan menjadi baris subjek commit Kita.

Contoh lain:

```
git commit -m "fix: Perbaiki bug tampilan di mobile"
```

```
git commit -m "dok: Perbarui panduan instalasi"
```

Kita ingin menggunakan flag -m untuk menulis pesan commit yang panjang, termasuk baris subjek dan body pesan.

Secara teknis, ini bisa dilakukan, tetapi tidak disarankan untuk pesan yang benar-benar panjang atau kompleks. Alasannya karena sintaksnya menjadi rumit dan sulit dibaca/ditulis di command line.

Untuk membuat pesan multi-baris menggunakan -m, Kita perlu menggunakan lebih dari satu flag -m:

Flag -m yang pertama akan menjadi baris subjek.

Flag -m berikutnya akan menjadi baris-baris di bagian body. Git secara otomatis akan menambahkan baris kosong di antara subjek dan body.

Sintaksnya adalah:

```
git commit -m "Pesan Subjek" -m "Baris pertama Body" -m "Baris kedua
Body" ...
```

Misalnya Kita ingin membuat pesan commit seperti ini:

```
refaktor: Perbaiki struktur direktori

Memindahkan file-file konfigurasi dari root ke subdirektori
'config'.
Ini untuk meningkatkan keteraturan proyek dan memudahkan manajemen
file.
```

Kita akan menuliskannya di command line sebagai berikut:

```
git commit -m "refaktor: Perbaiki struktur direktori" -m
"Memindahkan file-file konfigurasi dari root ke subdirektori
'config'." -m "Ini untuk meningkatkan keteraturan proyek dan
```

```
memudahkan manajemen file."
```

Metode 2: Menggunakan Editor Default (Untuk Pesan Lengkap/Multi-baris)

Ini adalah metode yang disarankan untuk menulis pesan commit yang profesional, karena memungkinkan Kita untuk menyertakan baris subjek dan body pesan dengan detail. Saat Kita menjalankan git commit tanpa flag -m, Git akan membuka editor teks default Kita.

Jalankan perintah:

```
git commit
```

Git akan membuka editor teks default Kita (misalnya: Nano, Vim, VS Code, atau editor lain yang sudah Kita konfigurasi). Di dalam editor, Kita akan melihat sesuatu seperti ini:

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#   modified:   file1.txt
#   new file:   file2.txt
#
# Changes not staged for commit:
#   modified:   file3.txt
#
```

(Baris-baris yang dimulai dengan # adalah komentar dan akan diabaikan oleh Git. Baris-baris ini memberikan informasi tentang status staging area dan perubahan yang ada).

Tulis pesan commit Kita di baris paling atas, ikuti struktur profesional (subjek, baris kosong, body).

Contoh teks yang Kita tulis di dalam editor:

```
feat(profil): Tambahkan fitur ubah foto profil

Menambahkan fungsionalitas yang memungkinkan pengguna mengunggah
dan mengganti foto profil mereka melalui halaman pengaturan akun.

Implementasi menggunakan API unggah gambar baru dan menyimpan foto
di direktori penyimpanan publik. Proses validasi ukuran dan format
gambar juga ditambahkan.

Menutup isu #201

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#   modified:   file1.txt
```

```
#   new file:   file2.txt
#
# Changes not staged for commit:
#   modified:   file3.txt
#
```

Simpan dan keluar dari editor teks. Langkah ini tergantung pada editor yang Kita gunakan:

Nano: Tekan Ctrl + X, lalu tekan Y untuk menyimpan, lalu tekan Enter untuk mengonfirmasi nama file.

Vim: Tekan tombol Esc, ketik :wq, lalu tekan Enter.

Editor lain: Cari cara menyimpan dan keluar di editor spesifik tersebut.

Setelah Kita menyimpan dan keluar dari editor, Git akan membuat commit dengan pesan yang Kita tulis.

Metode kedua ini mungkin terasa lebih lambat pada awalnya, tetapi memungkinkan Kita untuk menulis pesan commit yang lebih informatif dan terstruktur dengan baik, yang merupakan ciri khas dari praktik Git profesional.

2. dasd