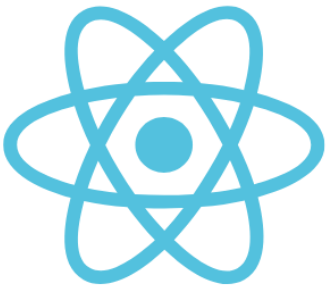
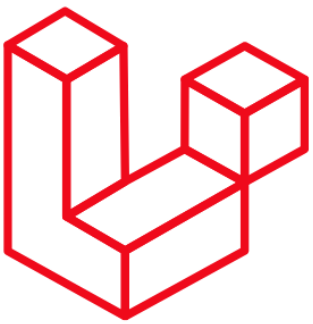
Program Studi Sarjana
Informatika
UNIVERSITAS ATMA JAYA YOGYAKARTA

MODUL PEMROGRAMAN APLIKASI WEB T.A. 25/26

MODUL 02

GITHUB 2





REQUIREMENT

1. Web Browser (Google Chrome/Chromium (Ms Edge, Opera) atau Firefox)
2. Sistem Operasi: Windows (versi 7 atau lebih baru), macOS, atau Linux.
3. Akun GitHub aktif (sudah dibuat di modul 1).
4. Software yang harus sudah terpasang:
   - Git (sudah diinstal sesuai modul 1).
   - Visual Studio Code (VSCode)
   - GitHub Desktop

PENJELASAN

TUJUAN

Setelah menyelesaikan modul ini, praktikan diharapkan mampu :

1. Memahami konsep branching dalam GitHub.
2. Membuat, berpindah, dan menghapus branch.
3. Menggabungkan branch menggunakan merge.
4. Menangani konflik yang muncul saat proses merge.
5. Bekerja secara kolaboratif dengan strategi branching.

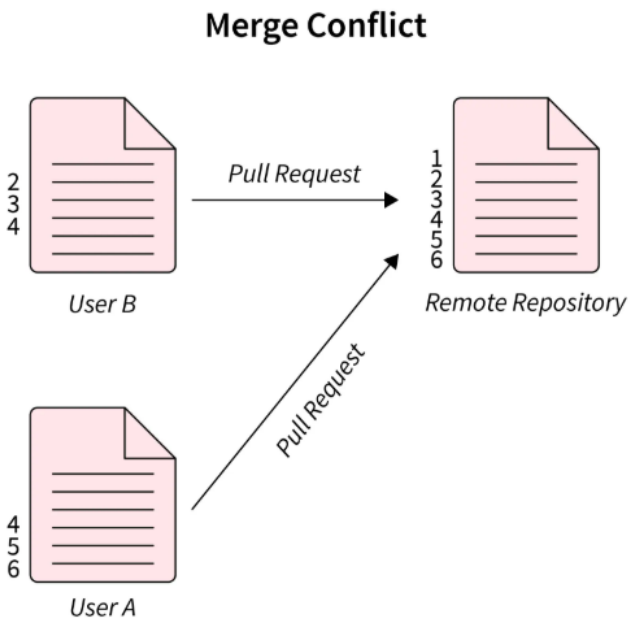
Dalam pengembangan perangkat lunak, tidak semua pekerjaan dapat dilakukan hanya dalam satu alur utama (main/master). Setiap anggota tim sering kali perlu mengerjakan fitur, perbaikan bug, atau eksperimen secara terpisah tanpa merusak kode utama yang sudah stabil. Di sinilah konsep branch, merge, dan konflik menjadi penting untuk dipahami.

Branching memungkinkan developer membuat jalur pengembangan baru yang terpisah dari branch utama. Dengan cara ini, pekerjaan bisa dilakukan secara mandiri tanpa mengganggu stabilitas kode yang sudah ada. Branch juga mempermudah pengelolaan proyek besar karena setiap fitur bisa ditangani dalam cabang masing-masing.

Merging adalah proses penggabungan hasil kerja dari suatu branch kembali ke branch utama. Dengan merge, setiap kontribusi anggota tim dapat digabungkan sehingga membentuk versi terbaru dari proyek. Proses ini menjadi kunci kolaborasi karena memungkinkan semua orang bekerja secara paralel lalu menyatukan hasilnya.

Conflict adalah kondisi ketika dua atau lebih developer mengubah bagian kode yang sama pada file yang sama. Konflik ini adalah hal yang wajar dalam kolaborasi tim. Git menyediakan

mekanisme untuk mendeteksi konflik, memberi tanda pada bagian kode yang bermasalah, dan membantu developer memilih atau menggabungkan perubahan yang paling sesuai.



Gambar 1 Merge Conflict

Konsep branch, merge, dan konflik merupakan dasar penting dalam praktik kolaborasi nyata di industri perangkat lunak. Dengan memahaminya, mahasiswa tidak hanya belajar coding secara individu, tetapi juga siap menghadapi dinamika kerja tim, membagi tugas, dan mengintegrasikan hasil kerja secara profesional.

Branching

Branching adalah proses membuat cabang baru dari alur utama (main/master) pada repository. Dengan branch, seorang developer dapat mengembangkan fitur, memperbaiki bug, atau melakukan eksperimen tanpa mengganggu kode utama yang stabil.

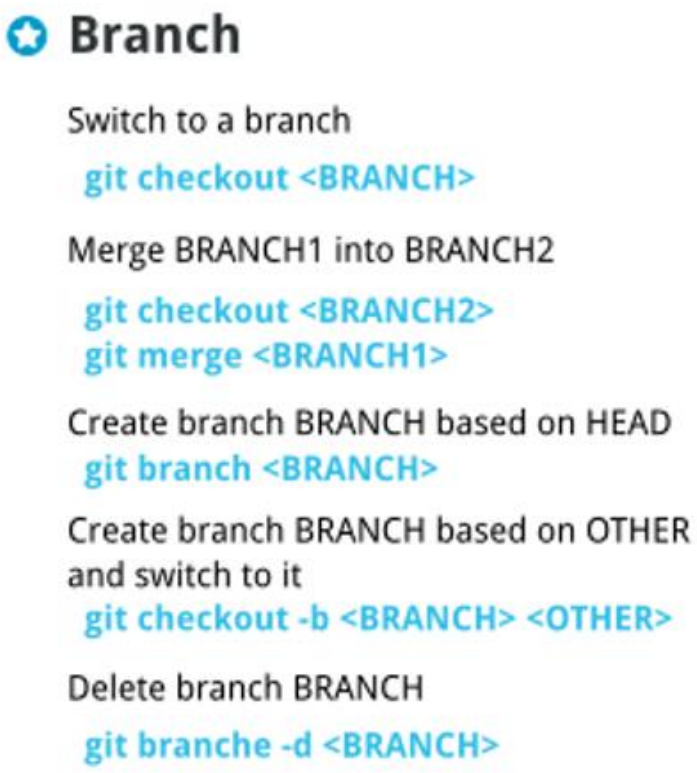
Contoh:

- Branch main = kode stabil siap digunakan.
- Branch fitur-login = percobaan menambah halaman login.
- Branch bugfix-header = perbaikan error pada bagian header.

Dengan adanya branch, pekerjaan bisa berjalan paralel: setiap anggota tim bisa mengerjakan tugas masing-masing dalam branch terpisah.

Tips Branching

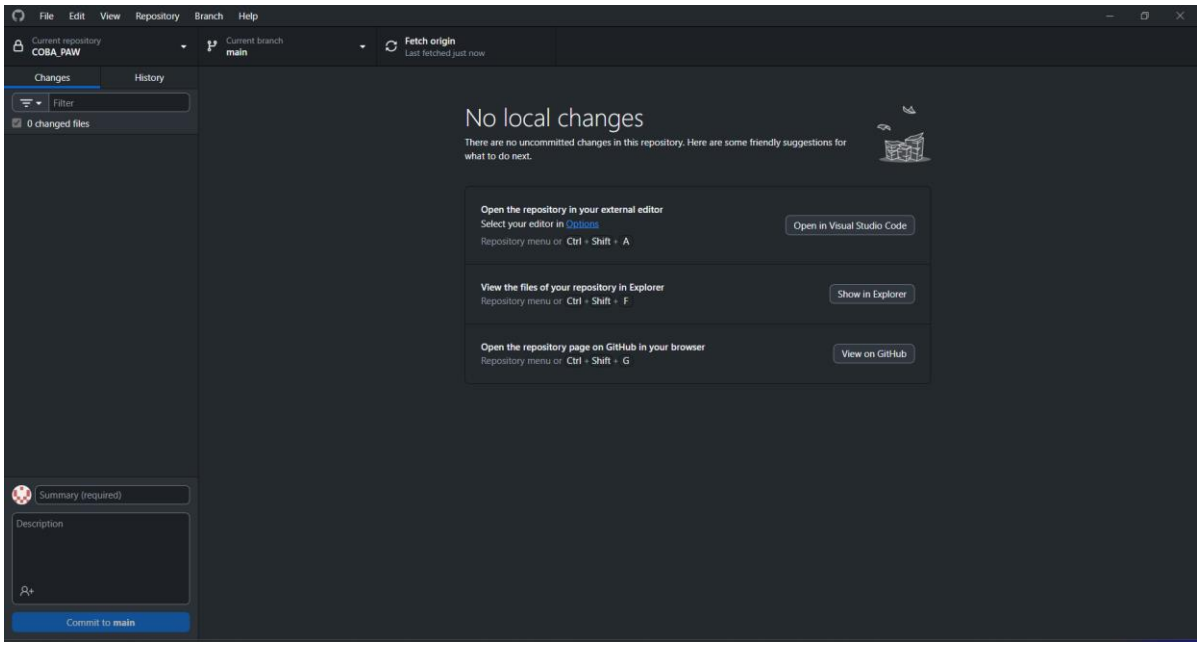
- Selalu buat branch baru untuk fitur/bug yang berbeda.
- Gunakan nama branch yang jelas: fitur-login, bugfix-navbar, update-style.
- Jangan langsung edit di branch main.
- Pastikan melakukan pull terbaru sebelum membuat branch agar tidak tertinggal dari repo utama.



Gambar 2

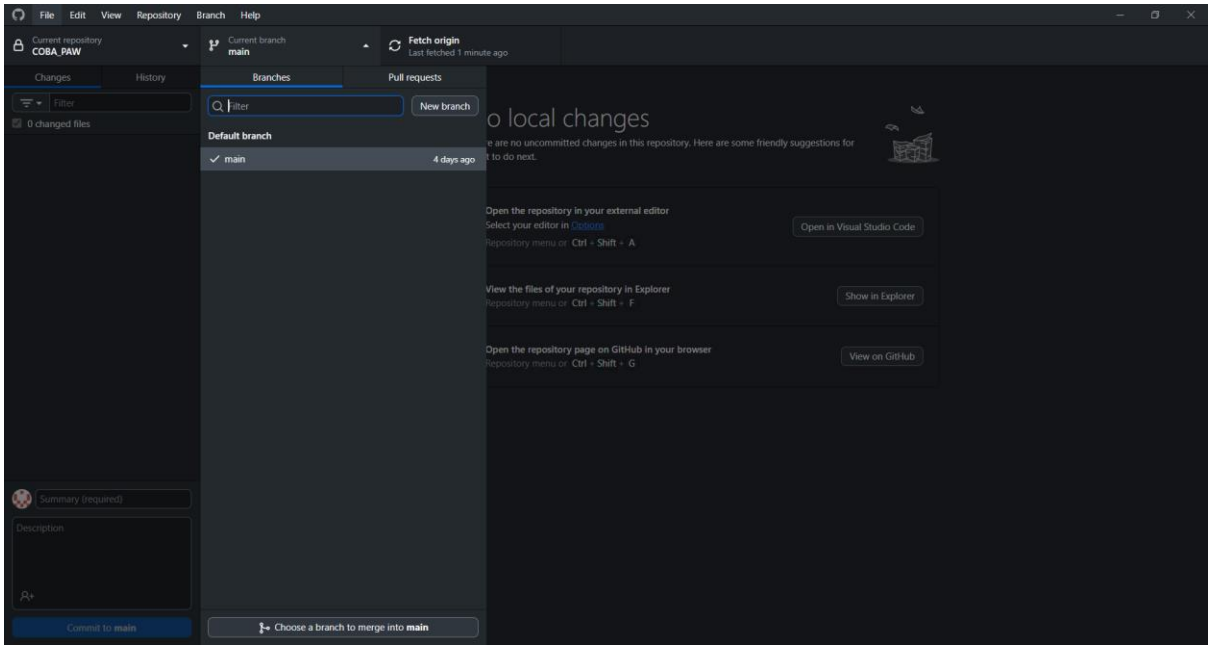
Branching dengan Github Desktop

1. Buka repository di GitHub Desktop.



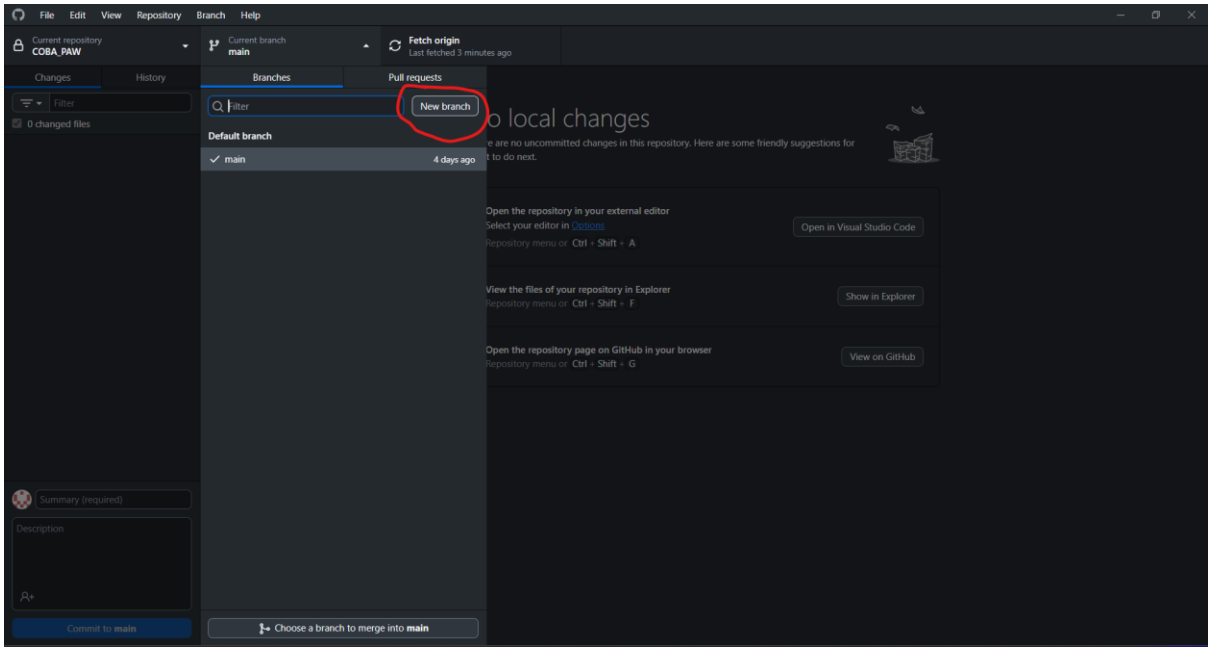
Gambar 3

2. Klik menu Current Branch di bagian atas.

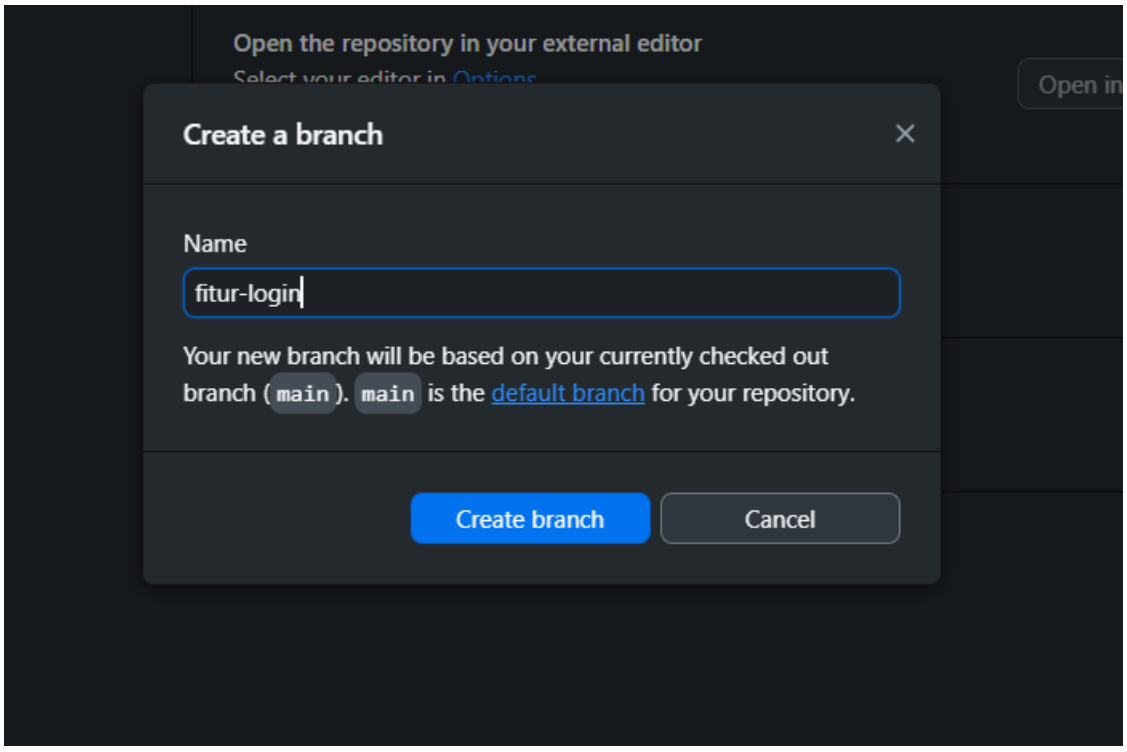


Gambar 4

3. Pilih New Branch…, masukkan nama branch, lalu klik Create Branch.

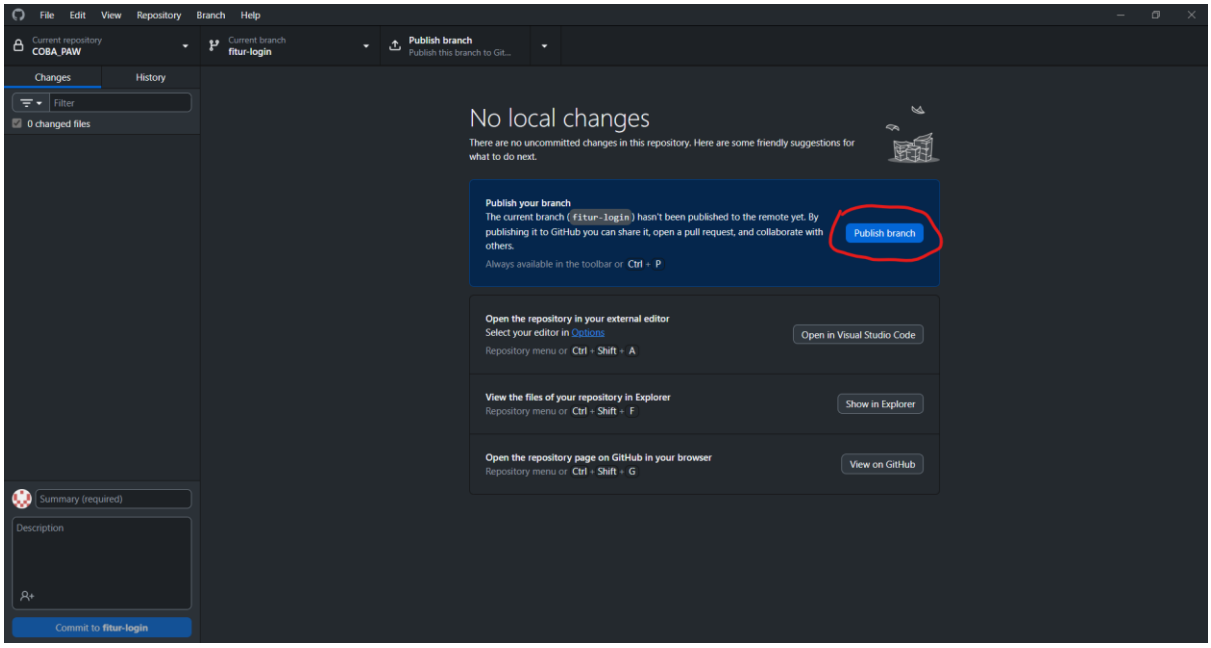


Gambar 5



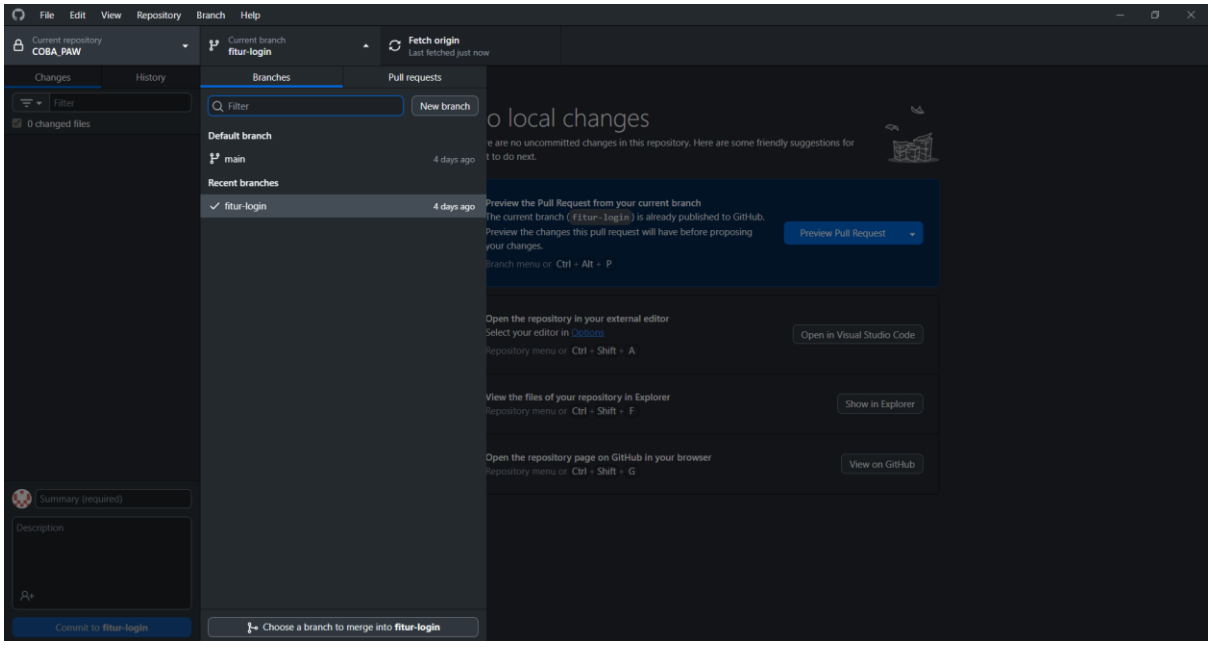
Gambar 6

4. Kemudian publish branch



Gambar 7

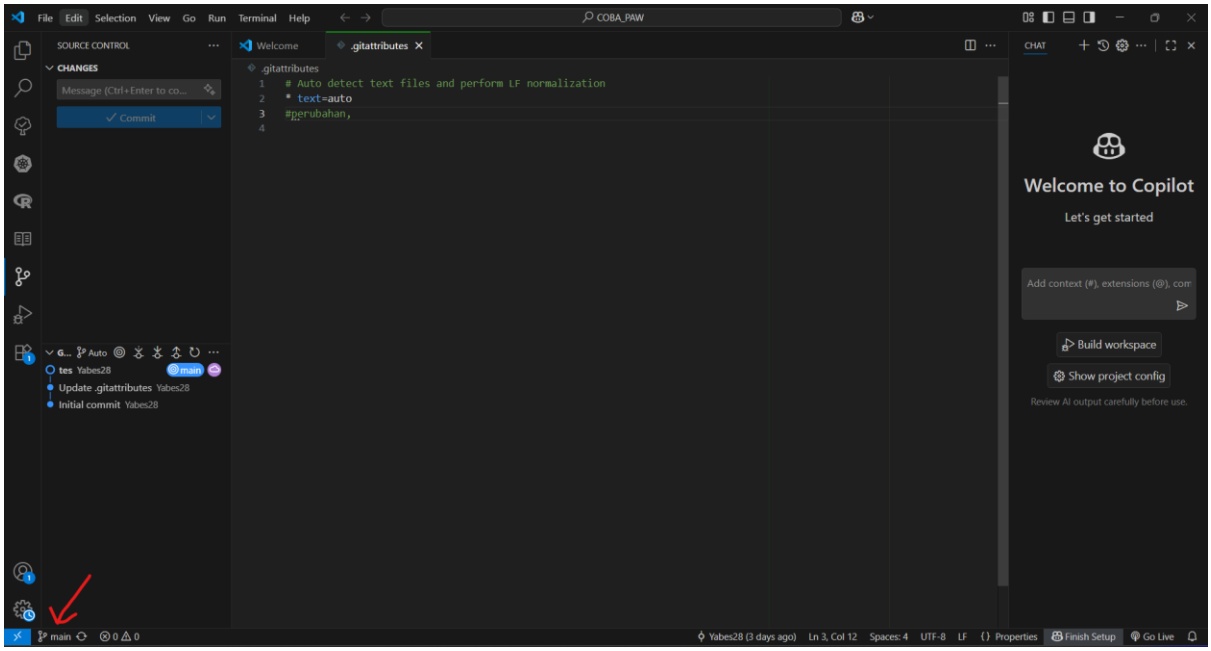
5. Branch baru otomatis aktif, dan semua commit berikutnya akan tersimpan di branch tersebut.



Gambar 8

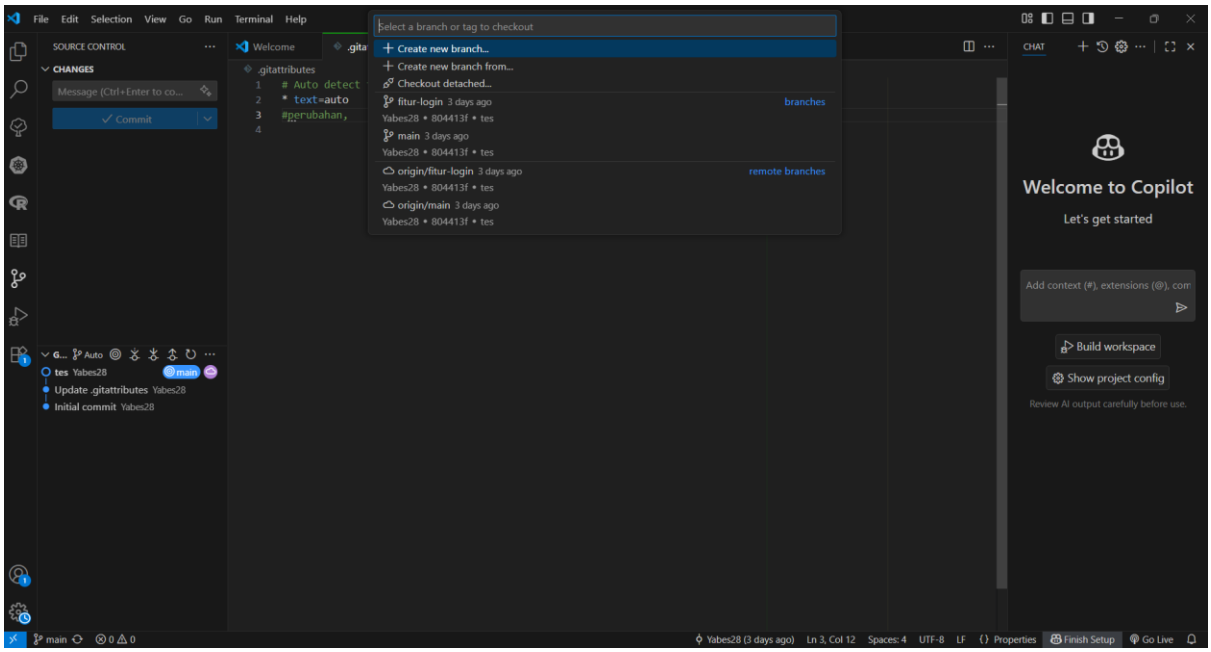
Branching dengan VSCode

1. Buka project di VSCode.
2. Klik nama branch di pojok kiri bawah (biasanya tertulis main).

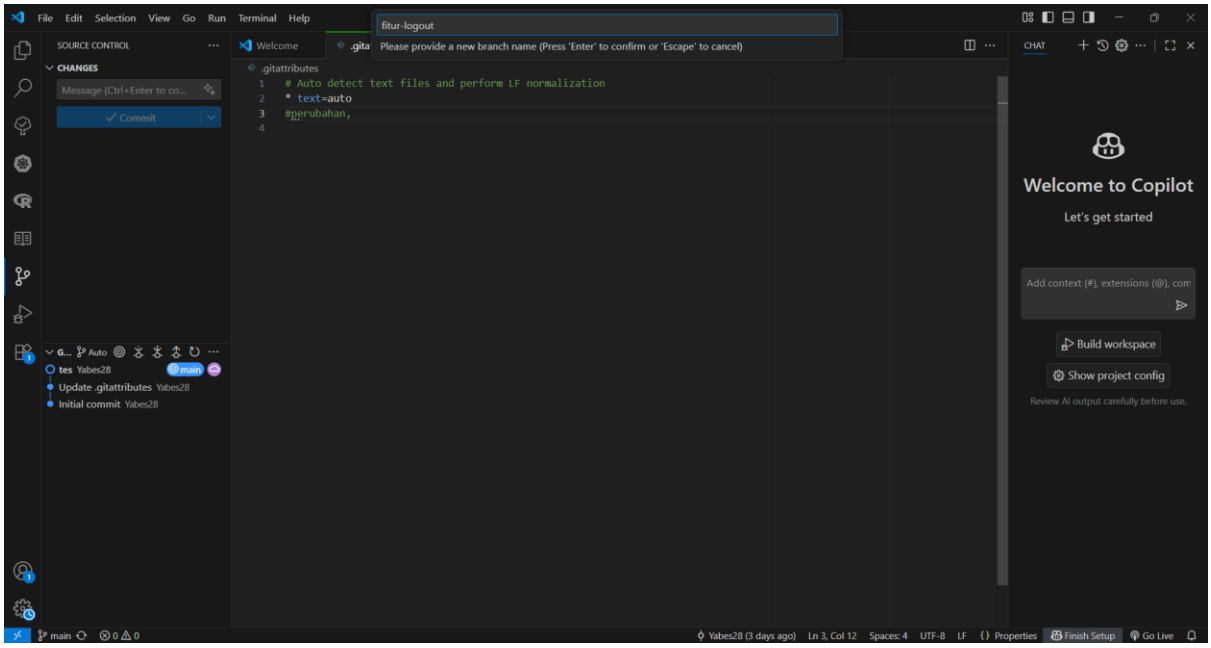


Gambar 9

3. Pilih Create new branch…, beri nama sesuai fitur.

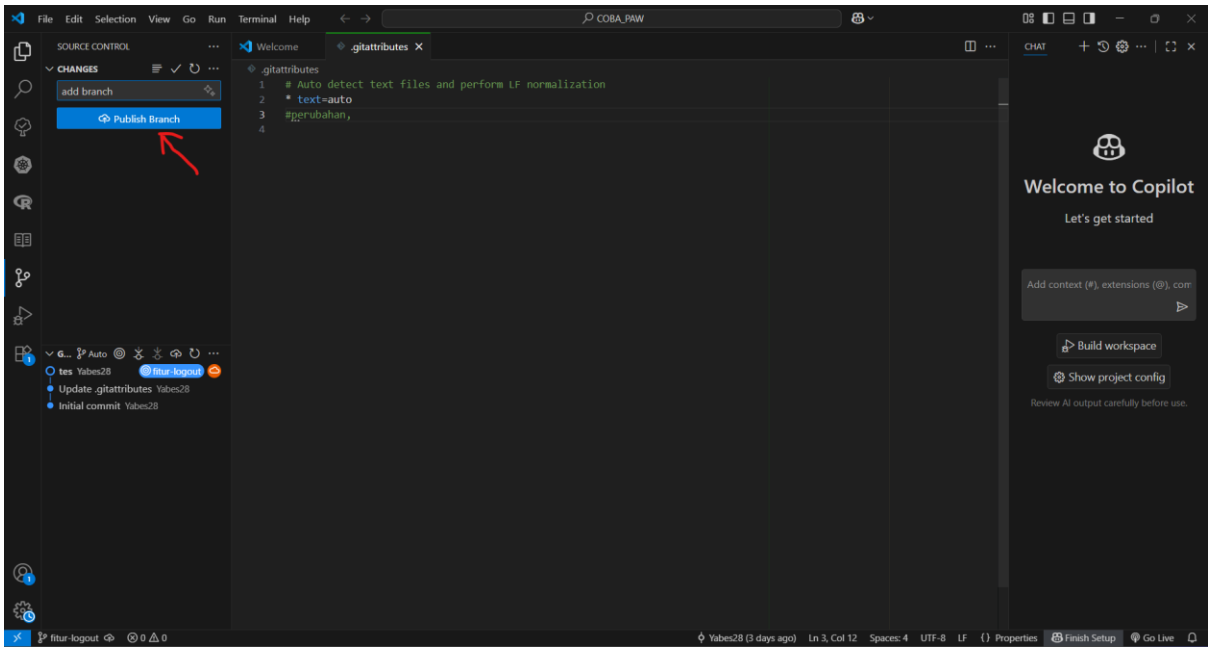


Gambar 10



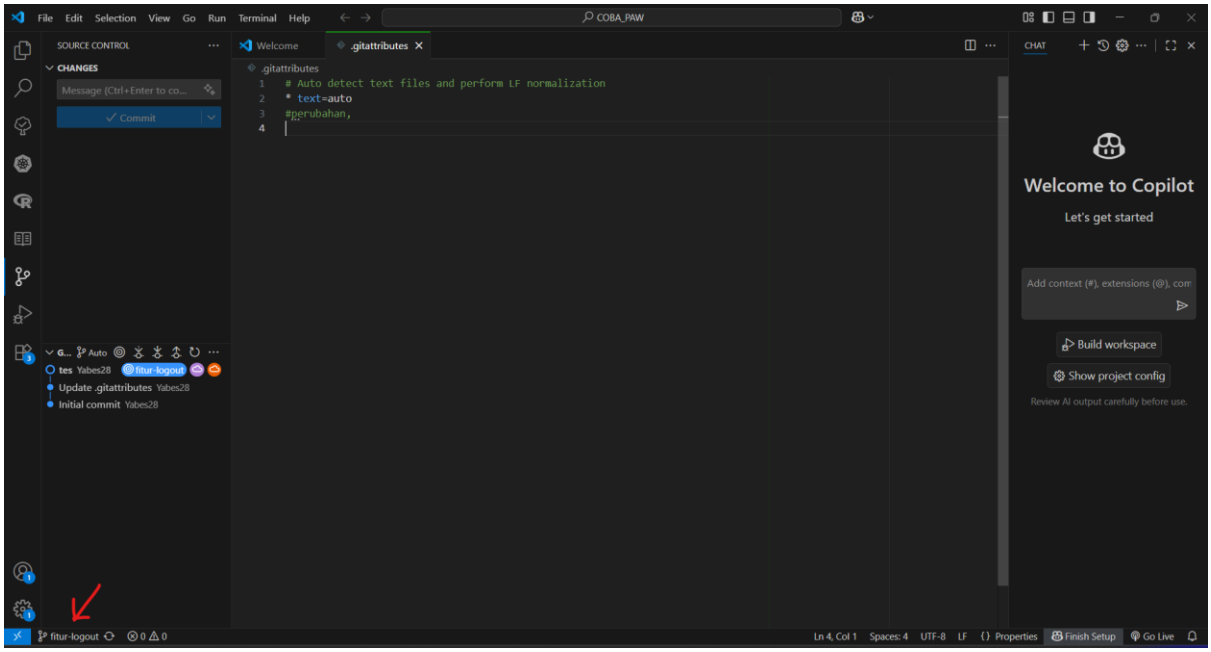
Gambar 11

4. Kemudian publish branch.



Gambar 12

5. Branch aktif akan berubah sesuai nama yang baru dibuat.



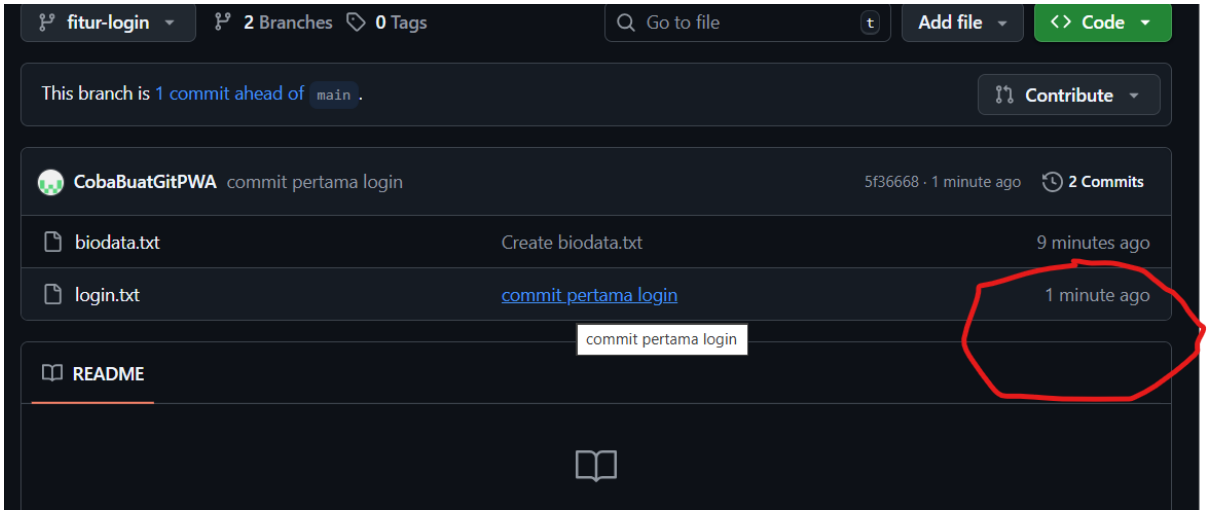
Gambar 13

Merging

Merging adalah proses menggabungkan perubahan dari satu branch ke branch lain. Biasanya, perubahan yang sudah selesai dikerjakan di branch fitur akan digabungkan kembali ke branch utama (main/master) agar kode terbaru bisa digunakan bersama. Dengan merge, hasil kerja tim yang dikerjakan secara paralel dapat disatukan sehingga repository tetap terupdate dengan versi paling baru.

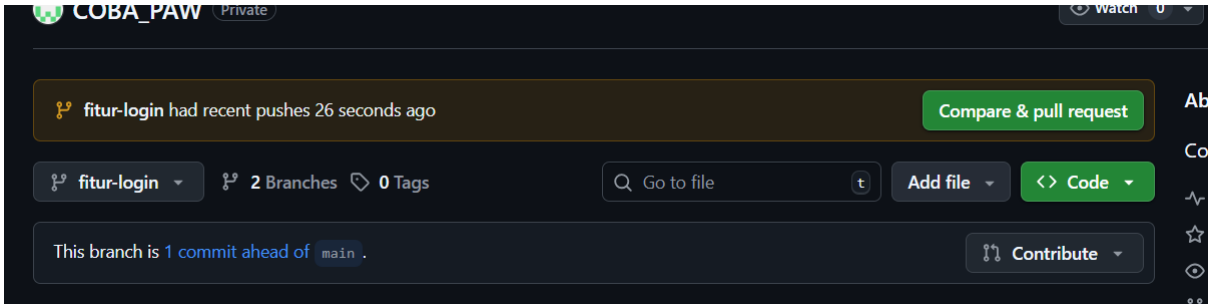
Langkah-langkah melakukan merge

1. Setelah anda melakukan push file pada branch (misal di dalam fitur-login menambahkan login.txt)



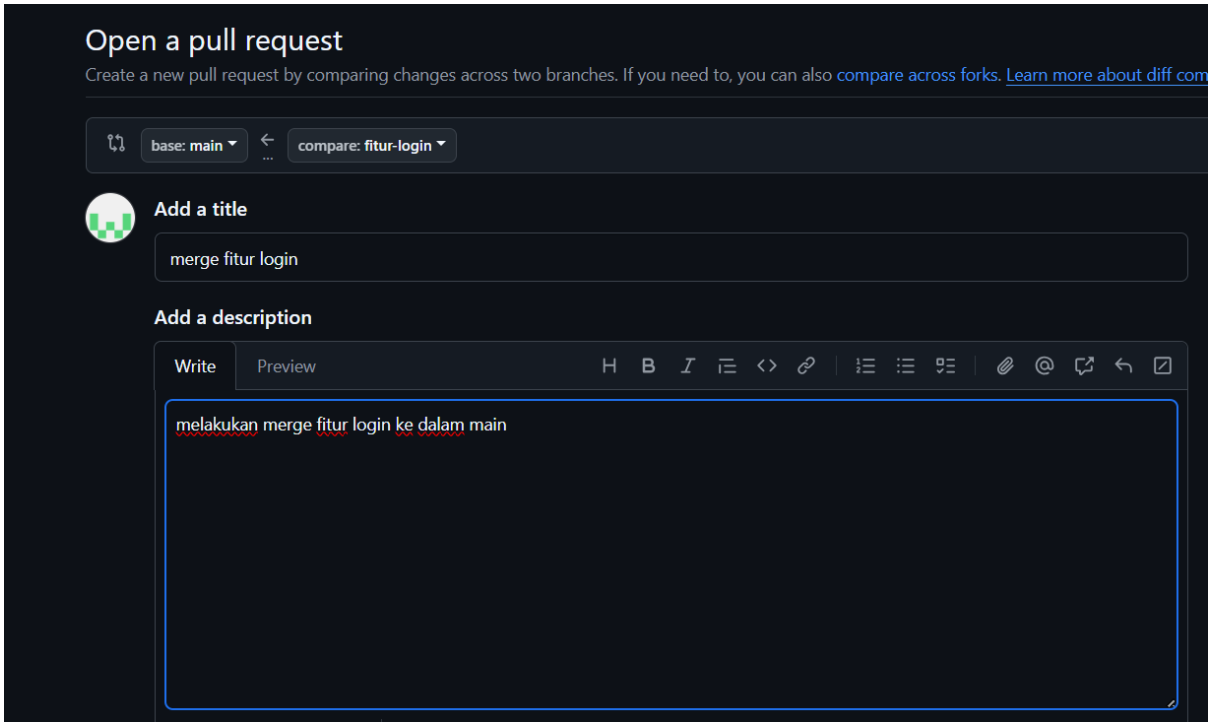
Gambar 14

2. Akan ada permintaan untuk compare & pull request



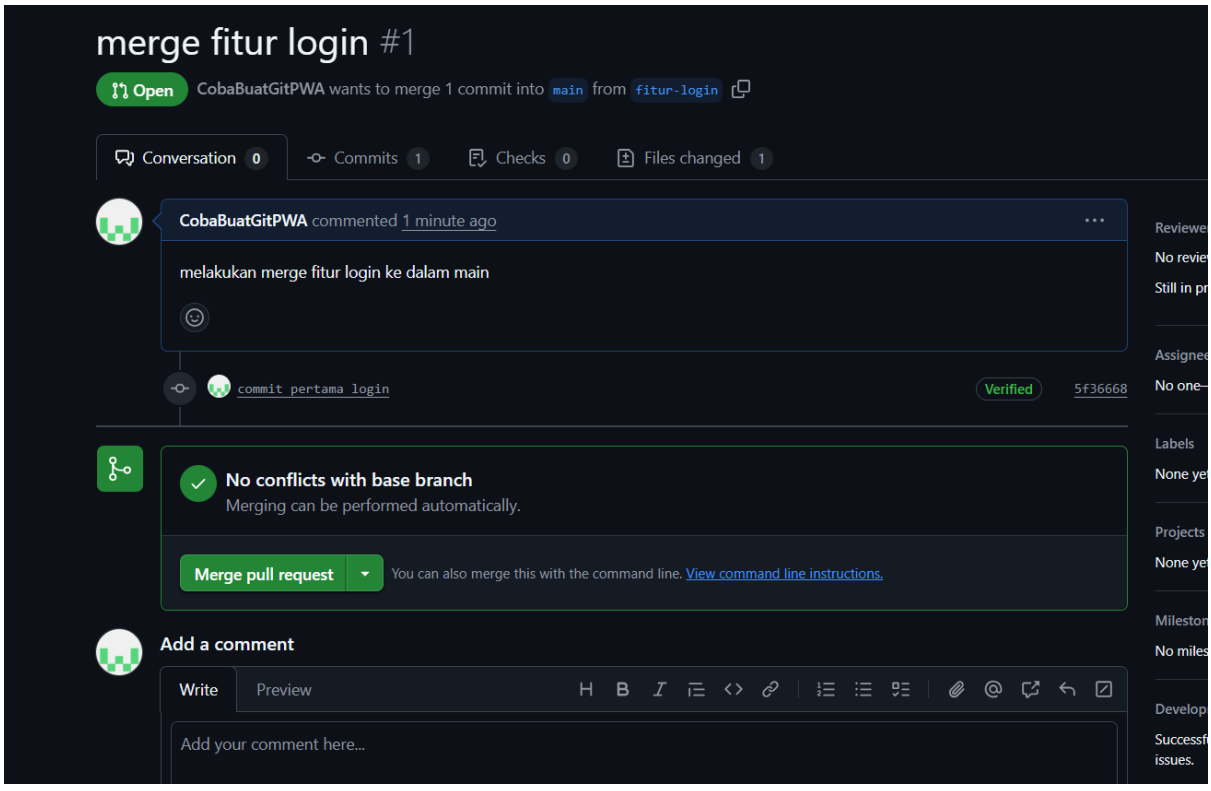
Gambar 15

3. Dalam pull request, **base branch** berarti branch tujuan yang akan menerima perubahan. Sedangkan **compare branch** berarti branch sumber yang berisi perubahan. Ketika melakukan merge berikan judul dan deskripsi merge yang jelas untuk membantu rekan tim memahami konteks merge. Dirasa sudah jelas maka dilanjut untuk pull request.



Gambar 16

4. Jika tidak ada konflik, maka tombol merge akan berwarna hijau dan file bisa langsung dilakukan merge dan dilanjut confirm merge.



Gambar 17

5. Ketika sudah **merged**, berarti proses merge telah berhasil dilakukan, biasanya jika fitur sudah selesai dibuat, branch nya bisa dihapus pada **delete branch** pada bagian bawah

Gambar 18

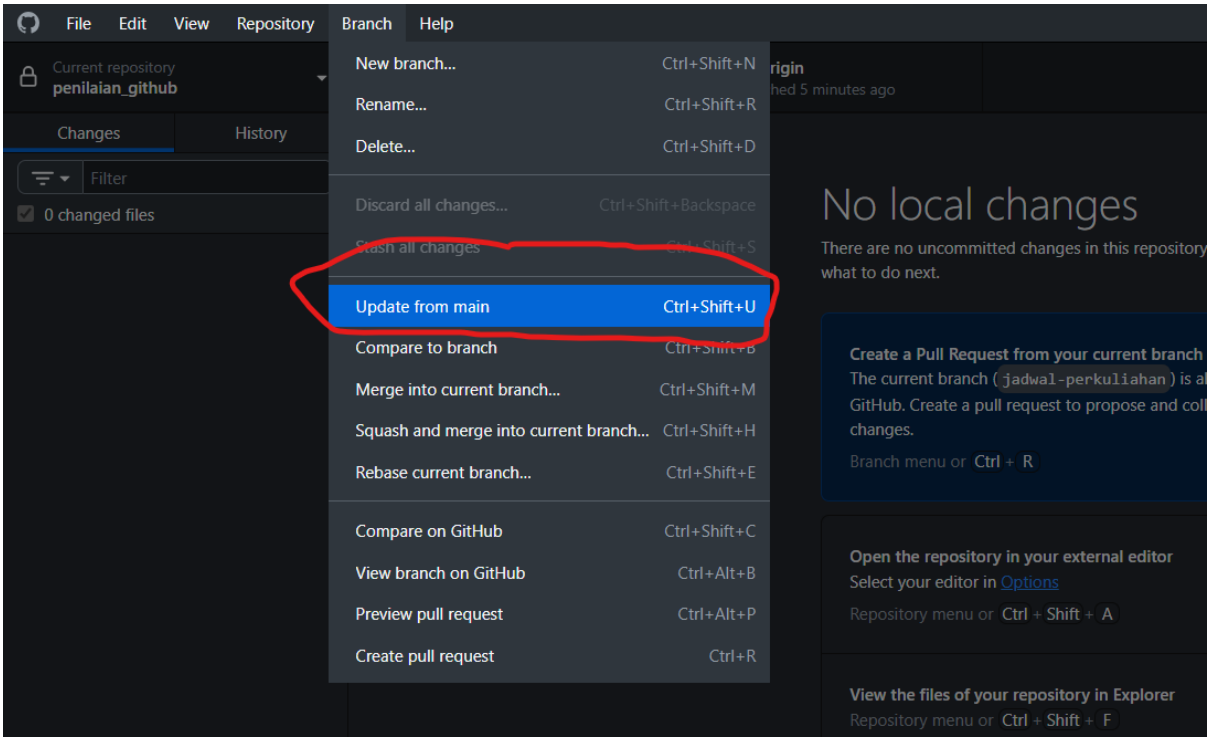
Tips Merging

- Lakukan merge secara berkala agar branch selalu up to date dengan main.
- Sebelum merge, biasakan untuk git pull origin main agar branch lokal tidak tertinggal.
- Gunakan pesan commit merge yang jelas.

Sinkronisasi Branch

Untuk mencegah masalah dan meminimalisir konflik di kemudian hari, sangat penting untuk secara berkala menyinkronkan branch dengan perubahan terbaru dari main. Proses ini memastikan.untuk **branch selalu up to date** dengan kemajuan di dalam main.

Untuk melakukan **update from main** bisa melalui github desktop pada menu branch (Pastikan berada di dalam branch yang ingin dilakukan sinkronisasi dengan main)



Gambar 19

Lalu pilih **update from main**

Dengan ini maka branch akan mengambil kemajuan terbaru dari main, lalu kita bisa mulai mengerjakan fitur di dalam branch yang sudah up to date dengan main.

Conflict

Conflict adalah kondisi ketika Git menemukan perubahan berbeda pada baris kode yang sama di dua branch yang akan digabungkan. Git tidak bisa memutuskan perubahan mana yang benar, sehingga developer harus menyelesaikan konflik tersebut secara manual.

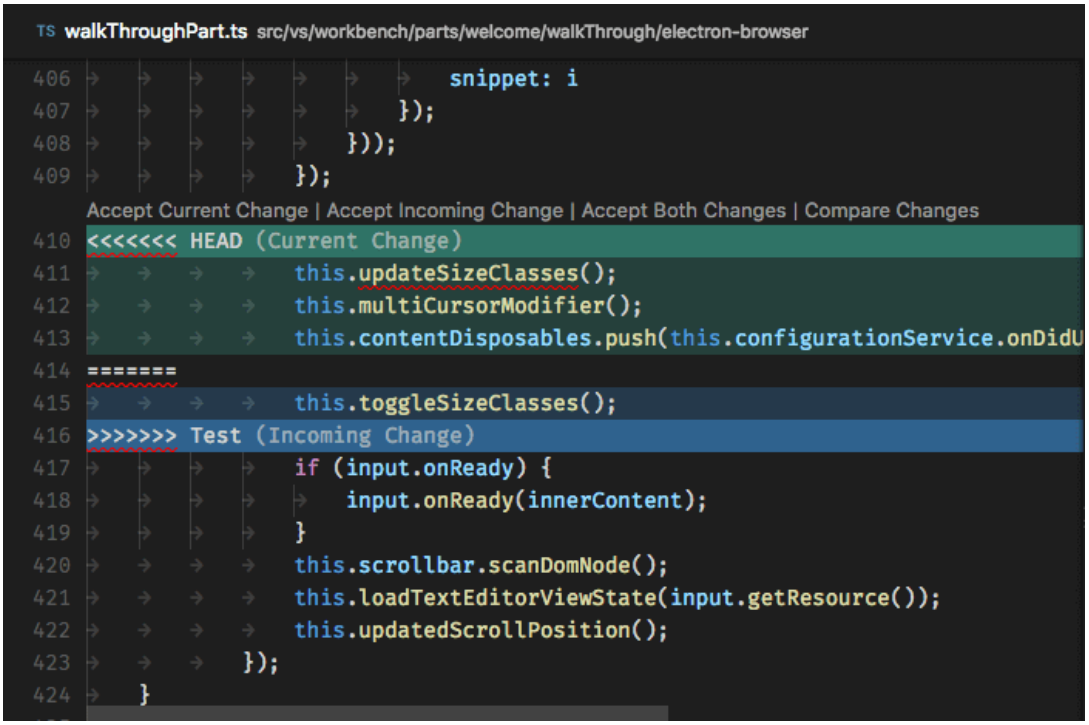
Contoh kasus:

- Branch main mengubah baris ke-10 file index.html.
- Branch fitur-header juga mengubah baris ke-10 file index.html.
- Saat dilakukan merge, Git akan menandai adanya konflik pada baris tersebut.

Tips Menghindari Konflik

- Selalu lakukan git pull sebelum mulai bekerja.
- Sering sinkronisasi dengan branch utama agar tidak terlalu banyak perbedaan.
- Komunikasikan dengan tim bagian mana yang sedang dikerjakan.

Contoh code konflik:

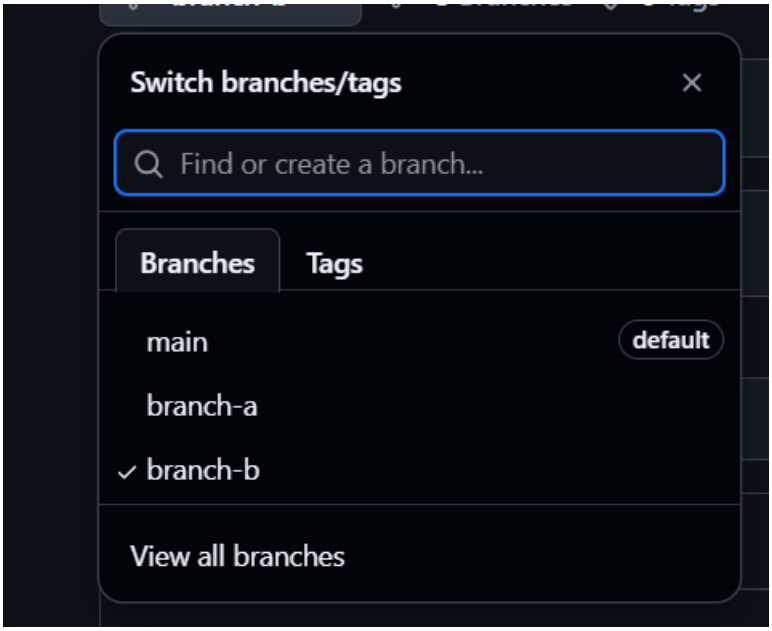


Gambar 20

1. Bagian HEAD → kode dari branch utama.
2. Bagian setelah garis ======= → kode dari branch yang digabung.
3. Untuk kasus ini harus memilih salah satu, atau menggabungkan keduanya secara manual.

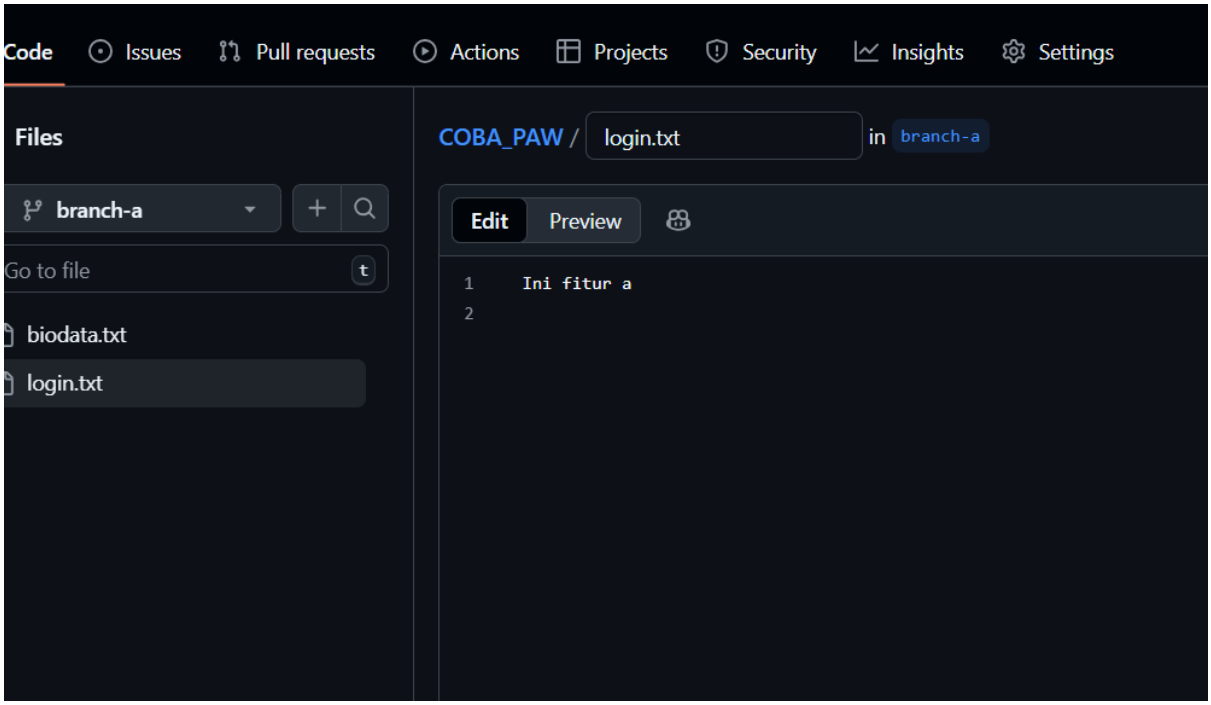
Langkah-langkah menyelesaikan conflict

1. Pertama-tama, buat dulu 2 branch from main, berikan nama “branch-a” dan “branch-b”. Harusnya tampilannya akan seperti di bawah ini



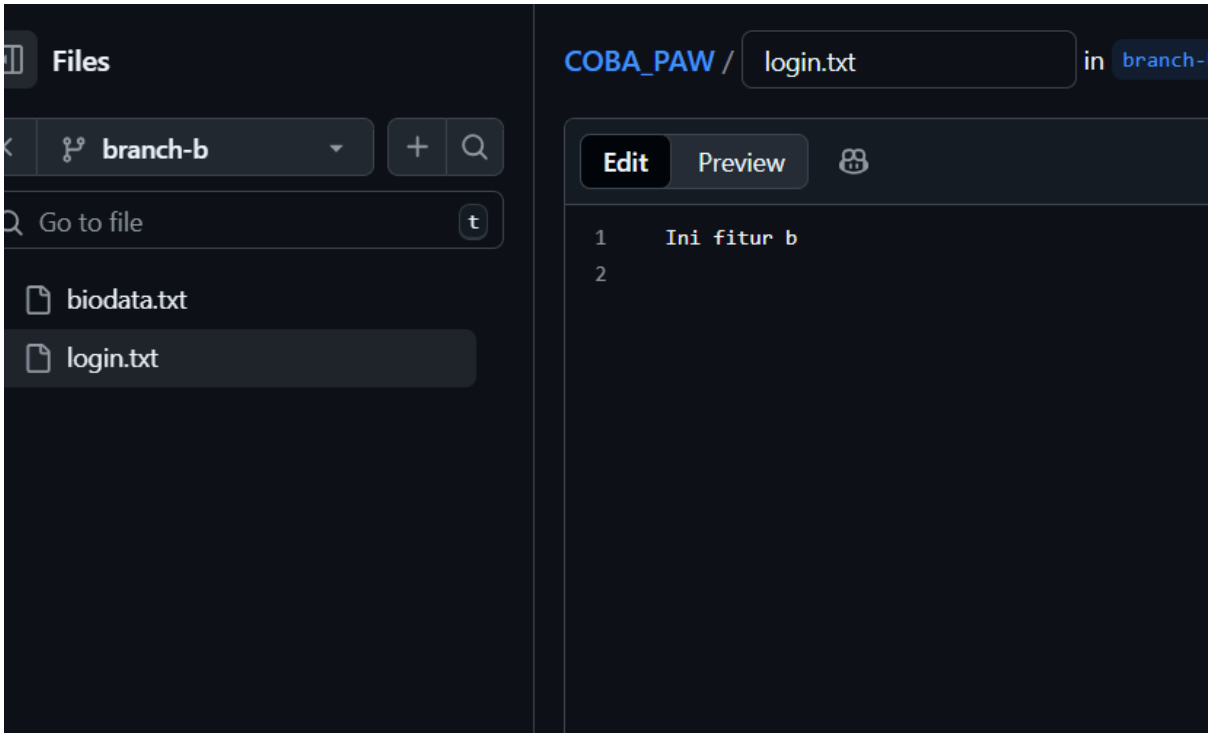
Gambar 21

2. Di dalam “branch-a”, gantilah isinya menjadi **ini fitur a**



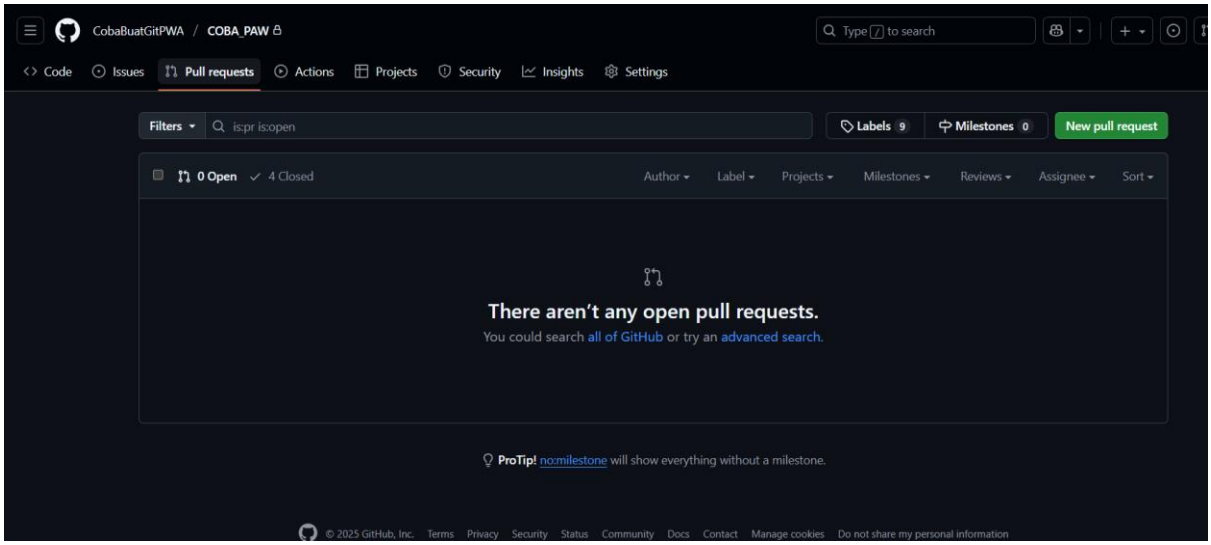
Gambar 22

3. Untuk “branch-b”, gantilah isinya menjadi **ini fitur b**



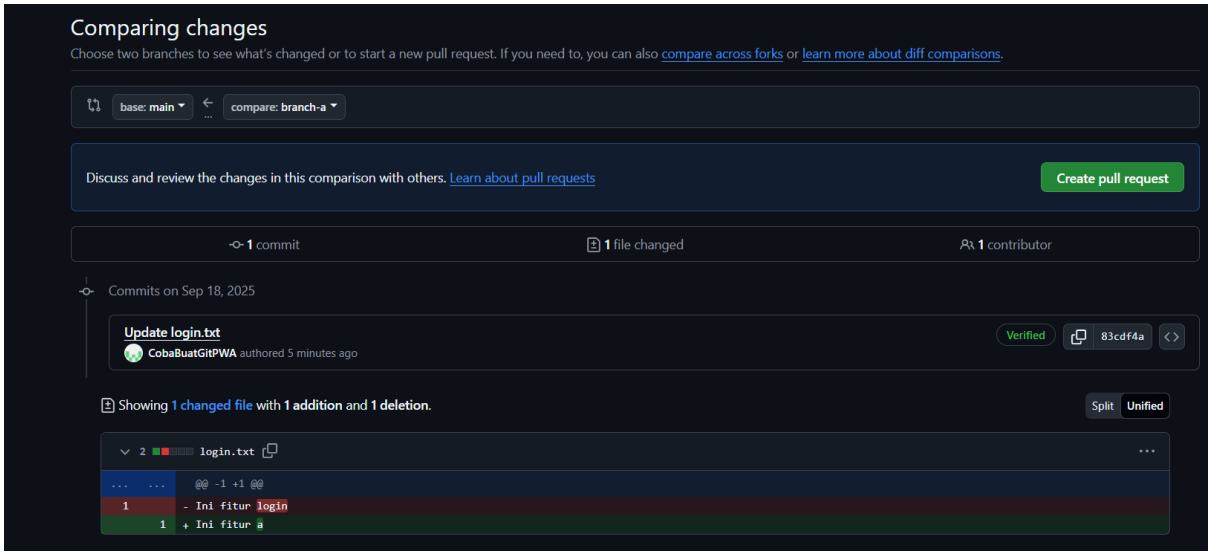
Gambar 23

4. Masuk ke dalam menu pull request, lalu **new pull request**



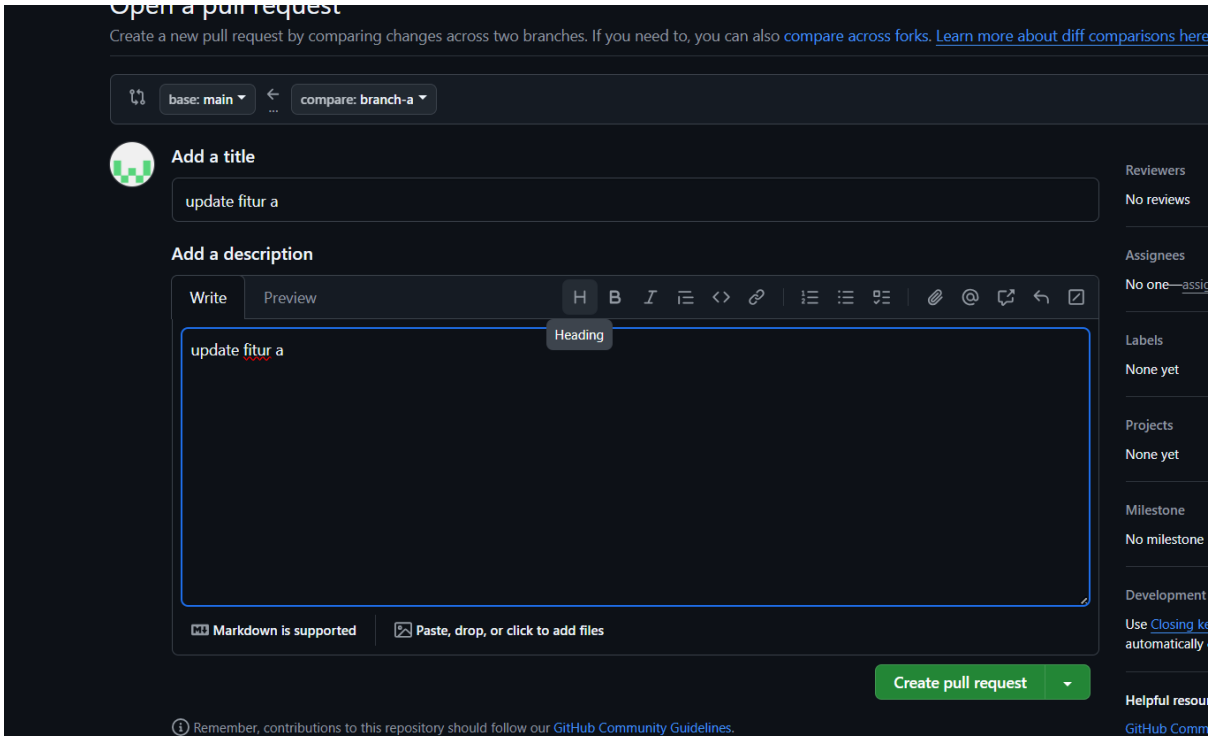
Gambar 24

5. Create pull request terlebih dahulu untuk branch-a, lalu **create pull request**



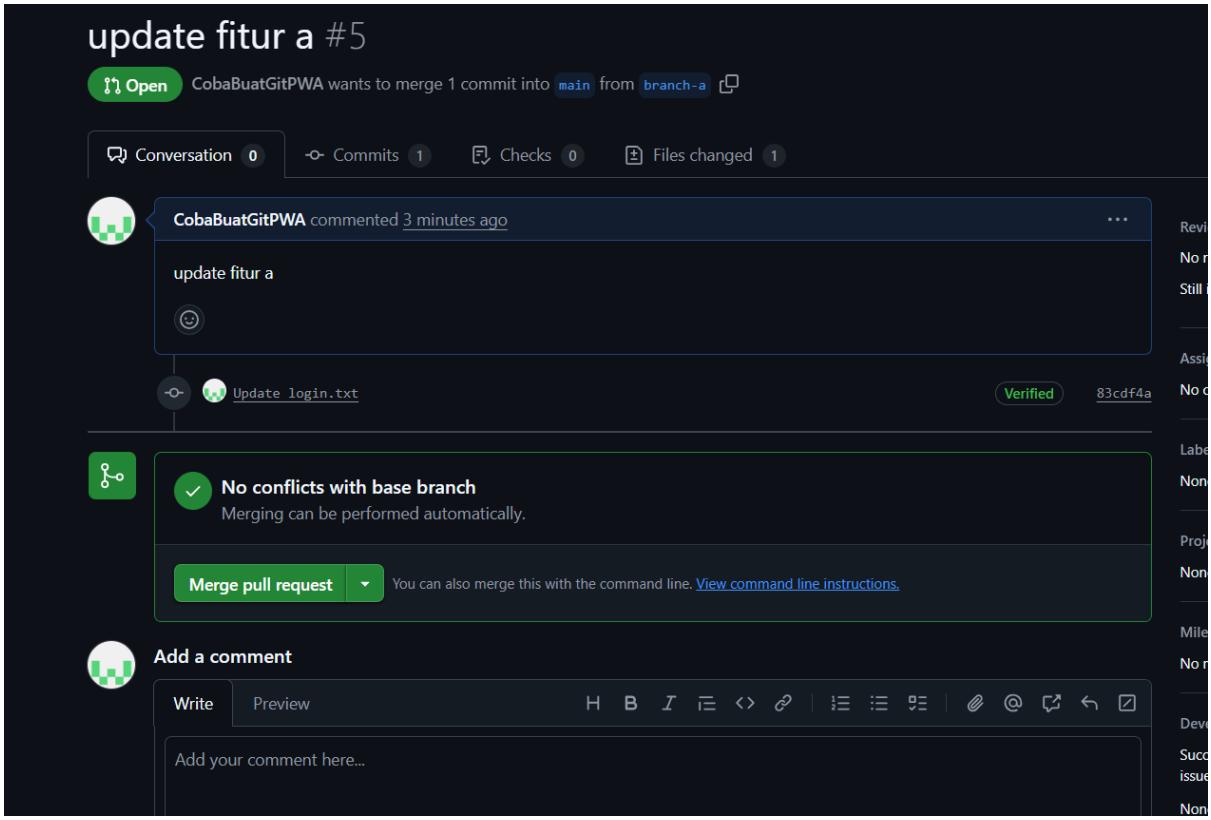
Gambar 25

6. Pastikan base nya main dan compare dengan branch-a, lalu **create pull request**



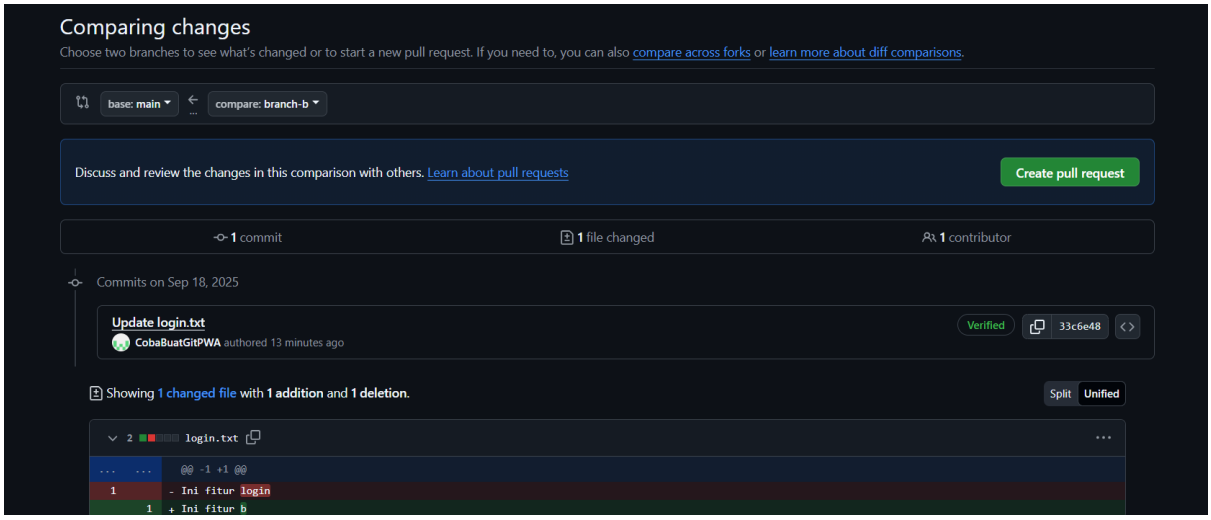
Gambar 26

7. Tombol merge pasti akan berwarna hijau karena belum ada perubahan pada main, sehingga git akan menambahkan perubahan baru tersebut ke dalam main. Langsung saja **merge pull request**.



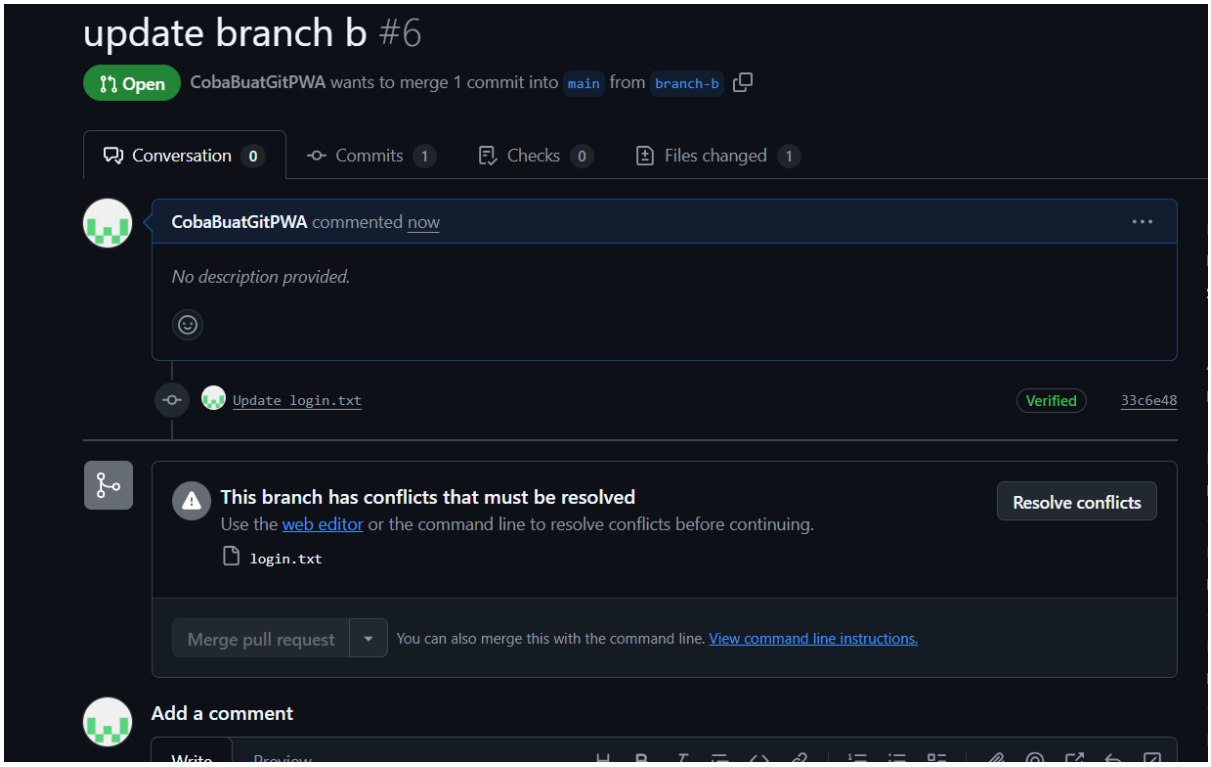
Gambar 27

8. Sekarang lakukan pull request untuk branch-b, pastikan base nya main dan compare dengan branch-b



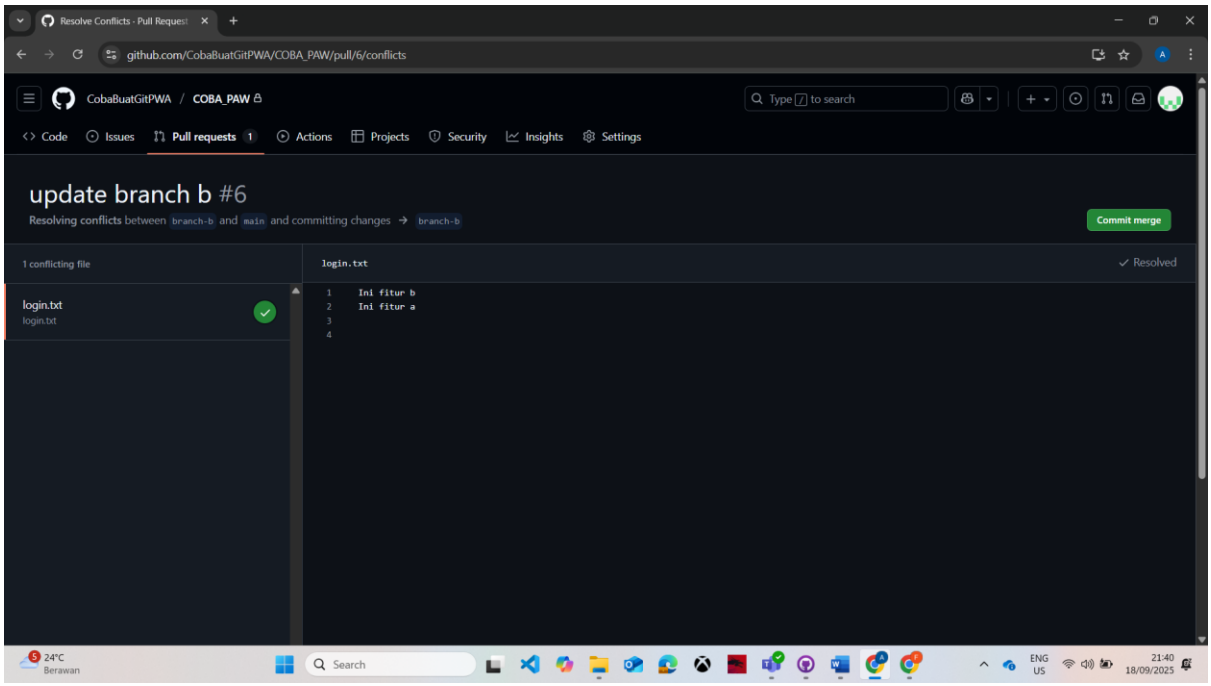
Gambar 28

9. Nah ketika dilakukan create pull request di sini pasti akan mengalami conflict karena git tidak bisa memutuskan otomatis perubahan yang diberikan dan menyerahkannya untuk kita selesaikan secara manual. Klik **resolve conflict**



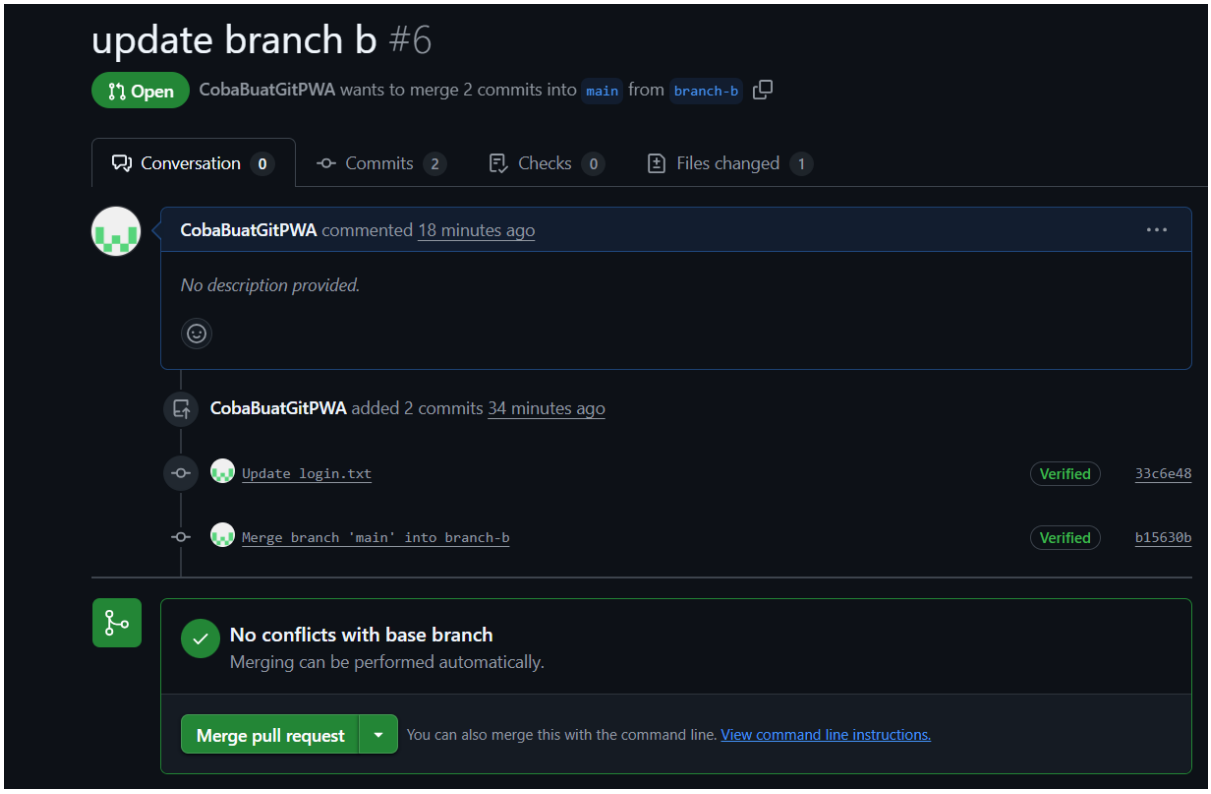
Gambar 29

10. Lalu dalam kasus ini kita akan menggunakan kedua line yaitu **ini fitur a** dan **ini fitur b** dengan cara menghapus tanda panah dan sama dengan. Tampilannya seharusnya akan seperti di bawah ini dan conflict telah terselesaikan. Klik commit merge



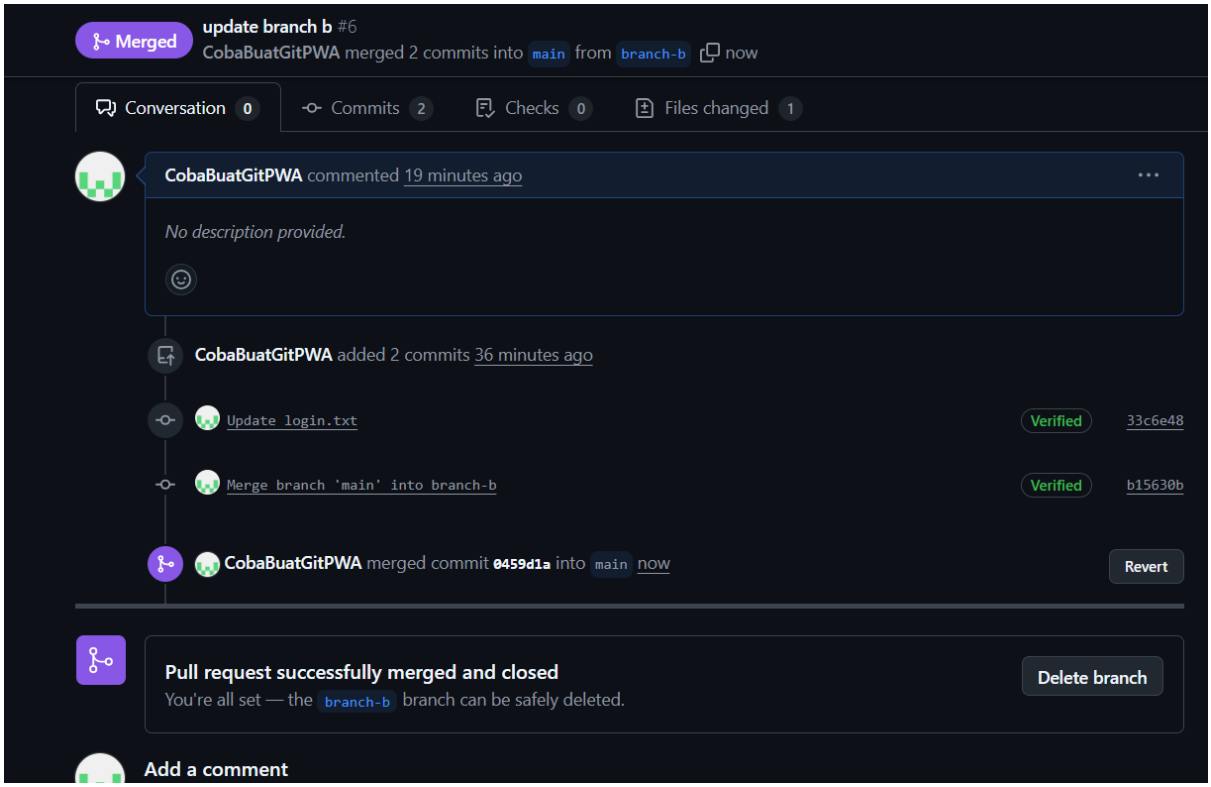
Gambar 30

11. Setelah menyelesaikan conflict, seharusnya merge pull request akan dapat dilakukan.



Gambar 31

12. Kemudian merge pun berhasil dilakukan



Gambar 32

Jadi dari langkah-langkah yang sudah kita lakukan, branch (branch-a dan branch-b) mengubah baris kode yang sama di file yang sama (login.txt) secara terpisah. Git dapat dengan mudah menggabungkan perubahan pertama (branch-a) ke main karena belum ada perubahan tumpang tindih. Namun, saat mencoba menggabungkan perubahan kedua (branch-b), Git tidak bisa memutuskan mana yang benar, sehingga menyerahkan penyelesaian konflik secara manual kepada pengguna. Langkah-langkah yang tadi telah dilakukan merupakan cara resolve conflict dan menyelesaikan merge untuk menggabungkan kedua perubahan secara manual.