

JOBSHEET 4

Version Control System

1. Tujuan

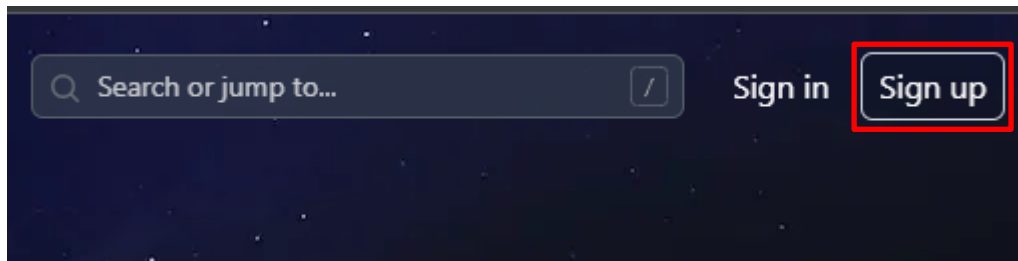
- Mahasiswa mampu menggunakan Git
- Mahasiswa mampu berkolaborasi dengan menggunakan Github

2. Praktikum

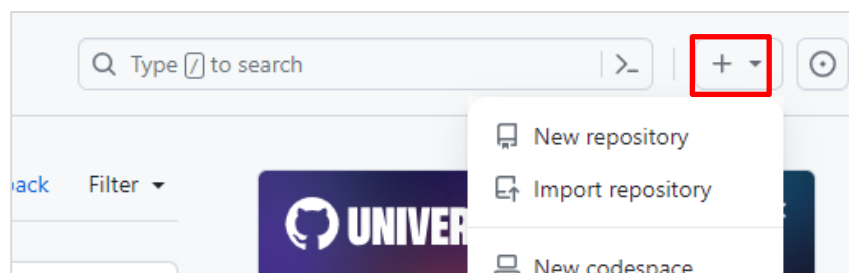
2.1 Percobaan 1: Menggunakan Github

Waktu percobaan : 60 menit

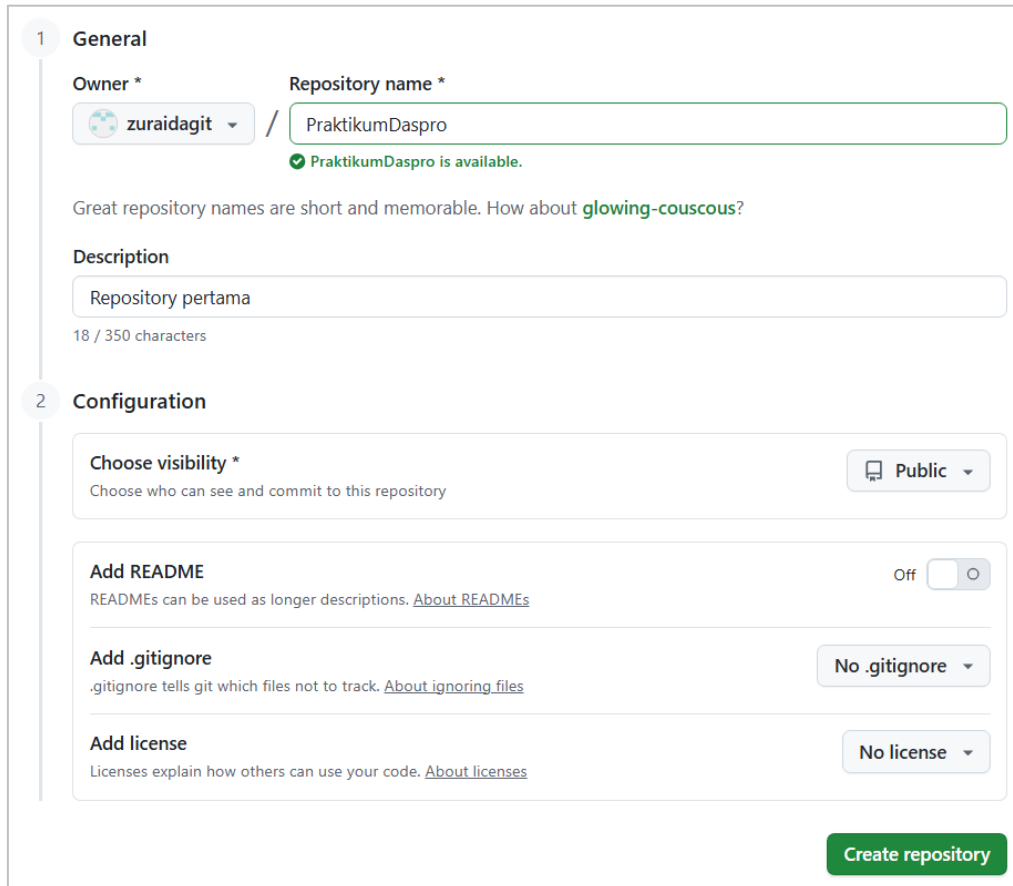
1. Buka situs web GitHub di <https://github.com>.
2. Klik tombol **"Sign up"** (Daftar) untuk membuat akun GitHub.



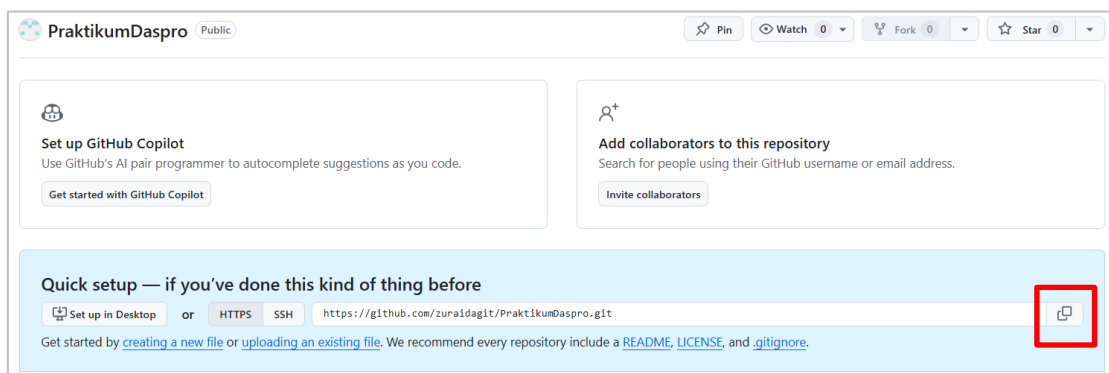
3. Ikuti petunjuk untuk mengisi informasi yang diperlukan dan lakukan verifikasi via email.
4. Setelah masuk ke akun GitHub Anda, klik tombol "+", lalu pilih **"New repository"** (Repositori baru).



5. Isi **nama repositori**, **deskripsi (opsional)**, dan **konfigurasi lainnya**. Anda dapat membuat repository public atau private sesuai kebutuhan. Pada praktikum ini, pilih visibility **Public**

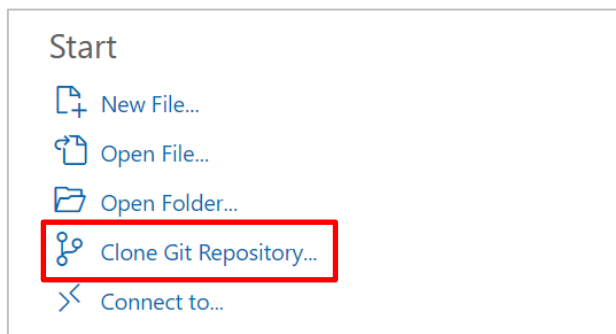


6. Klik tombol **“Create repository”**.
7. Repository baru telah dibuat di Github. Copy **Url** repository dengan click icon **Copy** yang nantinya akan digunakan dalam proses cloning.

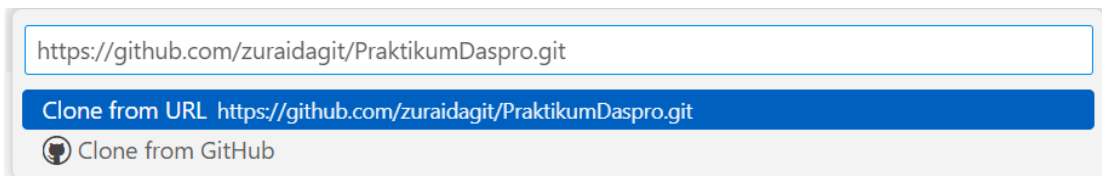


8. Untuk mengelola repositori secara lokal, Anda perlu meng-clone repository yang ada di Github ke komputer Anda. Lakukan installasi **git client** terlebih dahulu. Git client dapat didownload di <https://git-scm.com/downloads>

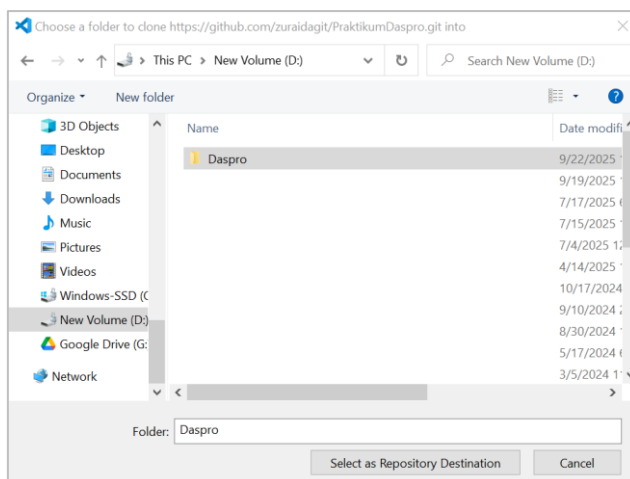
9. Untuk melakukan clone, buka VS Code. Pilih **Clone Git Repository**



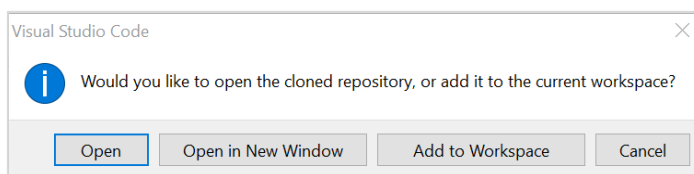
10. Paste repository Url dari Github, lalu enter



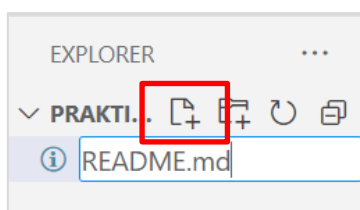
11. Pilih “Repository Destination”



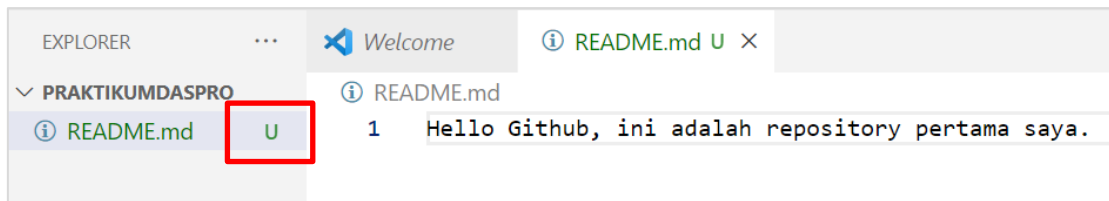
12. Pilih Open atau Open in New Window



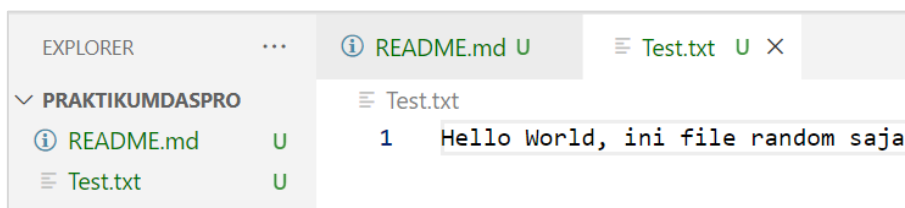
13. Anda dapat membuat file baru atau memodifikasi file pada repository sesuai kebutuhan. Lakukan percobaan dengan menambahkan file baru **“README.md”**.



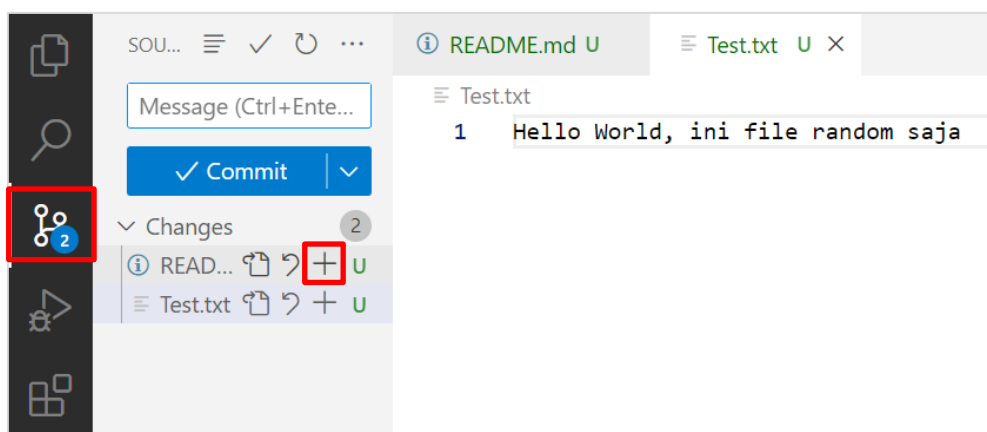
14. Isikan file **"README.md"** dengan kalimat seperti berikut. Perhatikan bahwa icon **U** menunjukkan bahwa file tersebut "Untracked" atau belum di-track oleh Git.



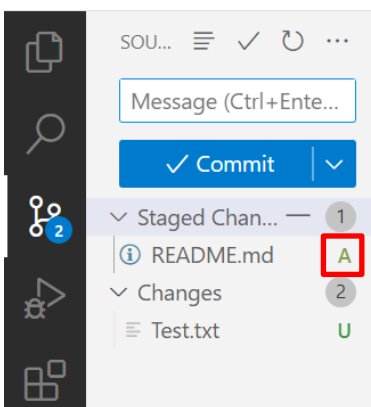
15. Untuk menyimpan perubahan di local repository, langkah pertama adalah melakukan **Add**. Proses Add berarti menambahkan file ke **staging area** yaitu tempate sementara di mana perubahan disiapkan sebelum disimpan permanen ke local repository melalui **Commit**. Dengan kata lain, **Add** berarti Anda sedang memilih perubahan mana saja yang akan di-commit.
16. Tambahkan file Test.txt sebagai perbandingan.



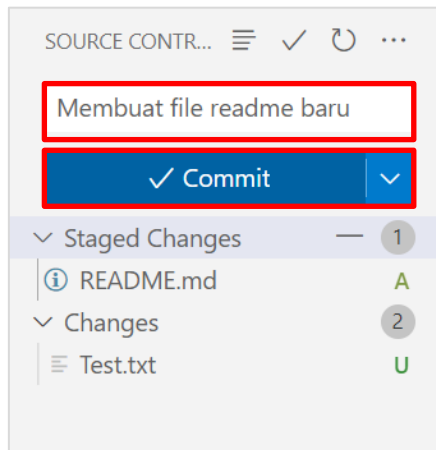
17. Pilih icon Source Control di sebelah kiri lalu klik icon **+** untuk **Add** file **README.md**. Biarkan file Test.txt tetap Untracked.



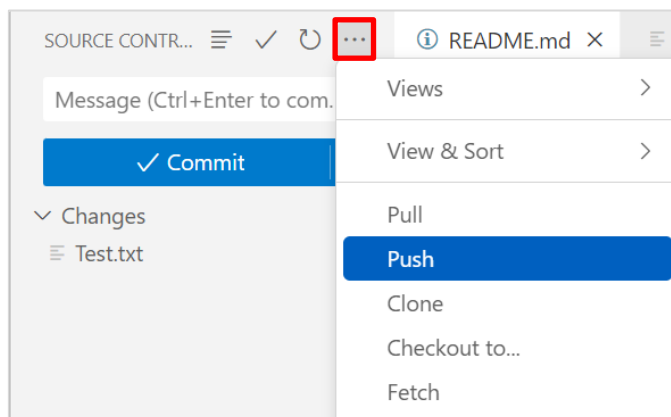
18. Status file README.md akan berubah menjadi **A** atau **Added**



19. Untuk menyimpan perubahan (membuat version baru) di local repository, isi **Commit Message** kemudian klik **Commit**. Biasakan mengisi commit message dengan catatan yang jelas, misalnya mengenai perubahan yang dilakukan.



20. Pada tahap ini, version baru telah dibuat tetapi hanya di local repository Anda. Untuk memperbarui repository di GitHub dengan versions yang sudah Anda simpan di local repository, lakukan **Push**.



21. Saat melakukan push ke repository, jika Git meminta username atau password, silakan gunakan username dan password Git Anda.

Jika muncul error seperti berikut:

```
remote: Permission to <username>/<nama-repository>.git denied
fatal: unable to access 'https://github.com/<username>/<nama-repository>.git': The requested URL returned error: 403
```

maka yang perlu dilakukan adalah membuat token untuk push ke repository. **Jika error tidak muncul, silakan loncat ke langkah 25.**

22. Untuk membuat token, klik akun Anda di Github → Settings → Developer Settings → Personal access tokens → Tokens (classic) → Generate a personal access token. Isi Note, Expiration, dan Select scopes. Jika sudah, klik tombol **Generate token**. **Simpan token tersebut karena tidak bisa dilihat kembali untuk digunakan pada push yang selanjutnya.**

Note

contoh token

What's this token for?

Expiration *

Custom... 02/09/2024

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

☒ repo Full control of private repositories

☒ repo:status Access commit status

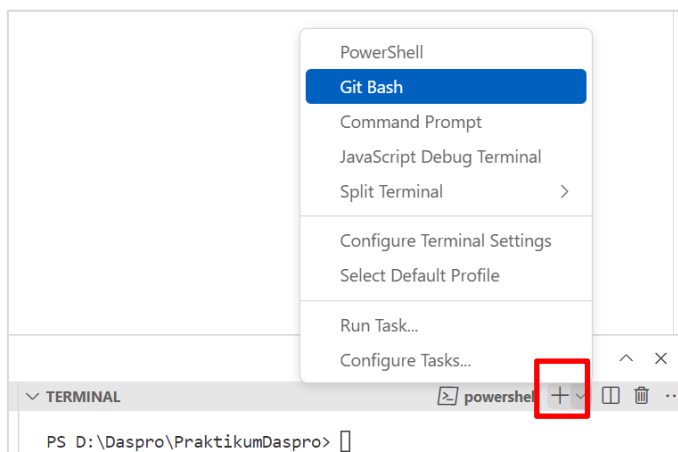
☒ repo_deployment Access deployment status

☒ public_repo Access public repositories

☒ repo:invite Access repository invitations

☒ security_events Read and write security events

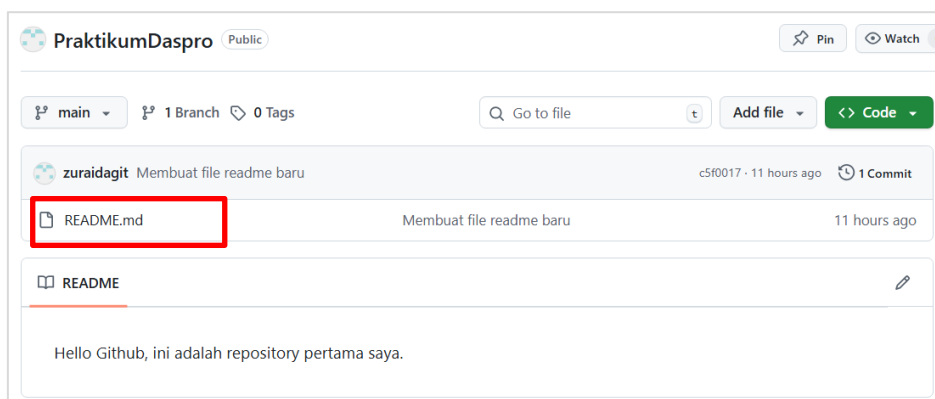
23. Buka terminal Git Bash



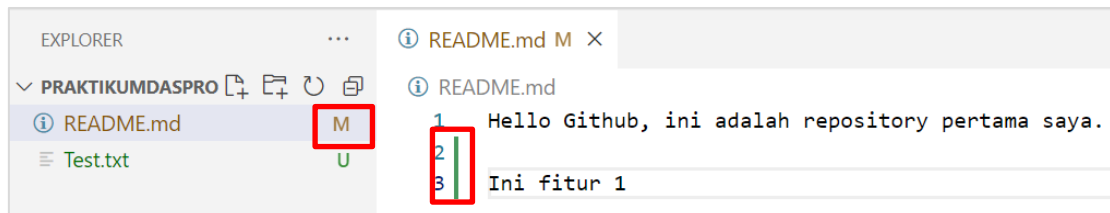
24. Di terminal git bash, jalankan command:

git push <https://<token>@github.com/<username>/<nama-repository>.git>

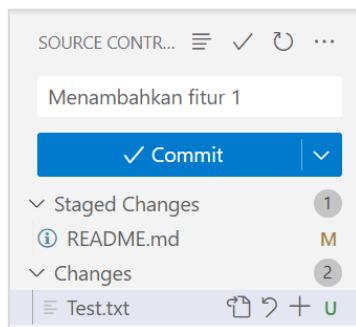
25. Cek halaman Github Anda. File README.md yang dibuat di local sekarang telah tersedia di repository Github. Klik filename untuk menampilkan isi file.



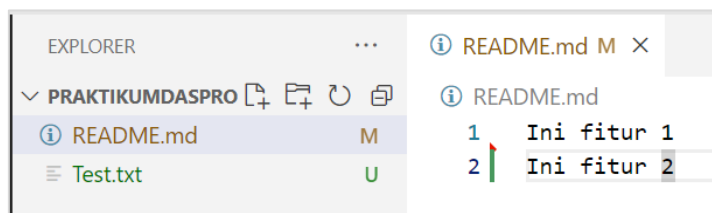
26. Jika Git meminta Anda untuk melakukan login setiap kali **push**, gunakan command:
`git config --global user.email "<email@example.com>"`
`git config --global user.name "<username>"`
27. Untuk memperjelas Add, Commit, dan Push. Coba lakukan perubahan pada file README.md lagi. Jika file di-Save, maka status dari file README.md akan berubah jadi **M** atau **Modified** artinya terdapat perubahan pada file, tapi belum masuk ke staging Area (belum di-Add). Baris yang baru ditambahkan ditandai dengan line berwarna hijau pada line number.



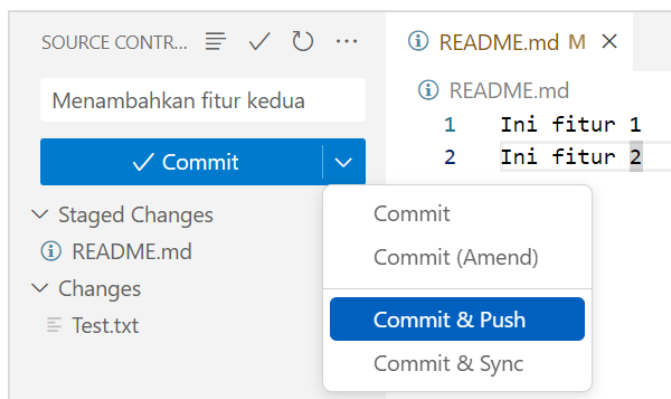
28. Lakukan Add dan Commit file README.md dengan cara yang sama seperti sebelumnya.



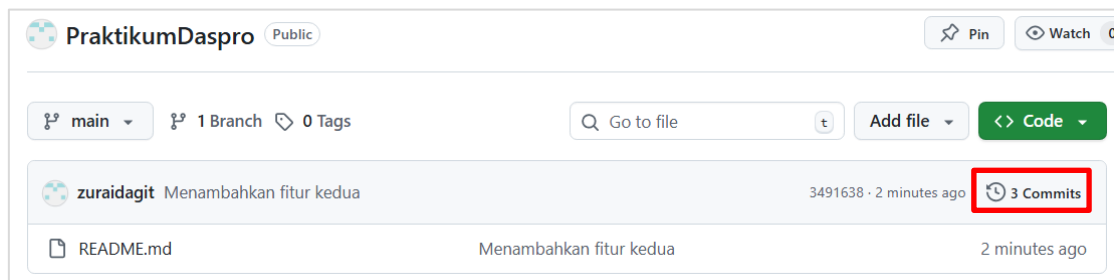
29. Lakukan perubahan lagi pada file README.md.



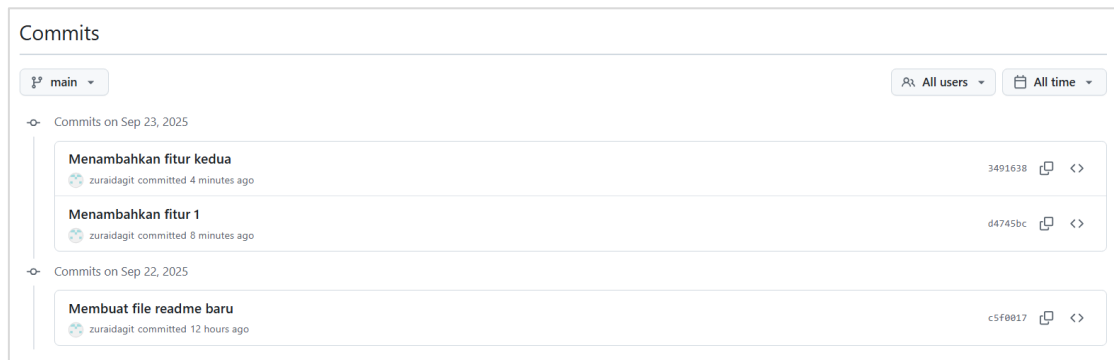
30. Lakukan Add dan Commit kembali. Kali ini, coba lakukan **Commit dan Push** sekaligus. Commit sekaligus Push sebaiknya hanya dilakukan jika Anda mengelola repository pribadi (tidak berkolaborasi dengan orang lain)



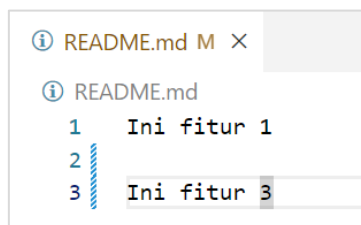
31. Cek Github repository, pilih link pada kotak merah untuk menampilkan riwayat commit.



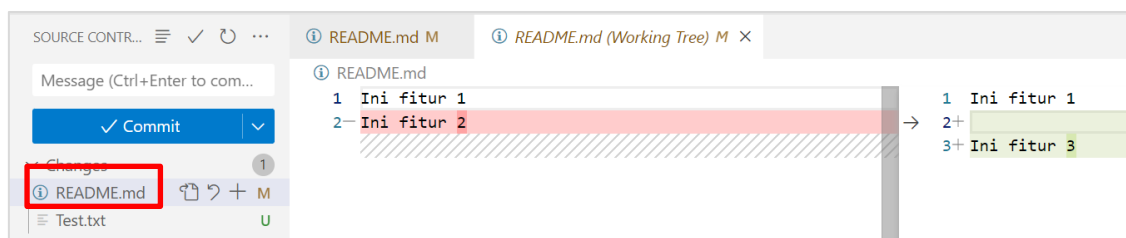
32. Anda dapat melihat setiap version yang dibuat setiap kali Commit dilakukan.



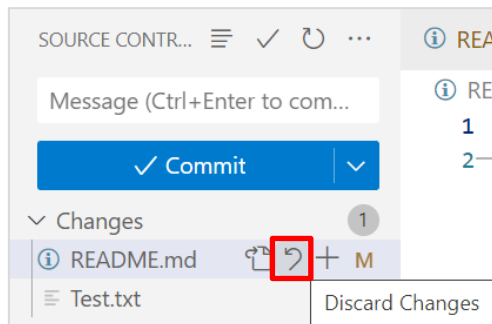
33. Silakan kembali ke VS Code dan coba lakukan perubahan lagi pada file README.md seperti contoh berikut:



34. Untuk membandingkan perubahan yang baru dilakukan dengan commit terakhir, klik nama file pada Source Control maka akan ditampilkan perbandingannya. Baris berwarna merah adalah baris yang dihapus sementara baris berwarna hijau adalah baris yang ditambahkan.

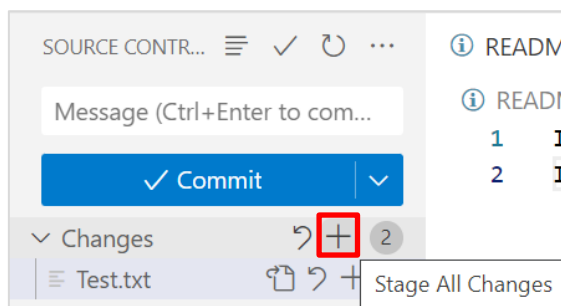


35. Misalnya Anda ingin membatalkan perubahan ke versi commit terakhir, klik icon Undo. Pilih **Discard Changes** jika muncul confirmation pop up.



Pertanyaan

1. Cek apakah file Test.txt ada di Github. Jika tidak, mengapa?
2. Jelaskan fungsi dari Add, Commit, dan Push
3. Add Test.txt. Untuk Add semua perubahan yang dilakukan, Anda bisa menggunakan icon **+** di sisi kanan **Changes**. Lakukan **Stage All Changes** kemudian **Commit & Push**.



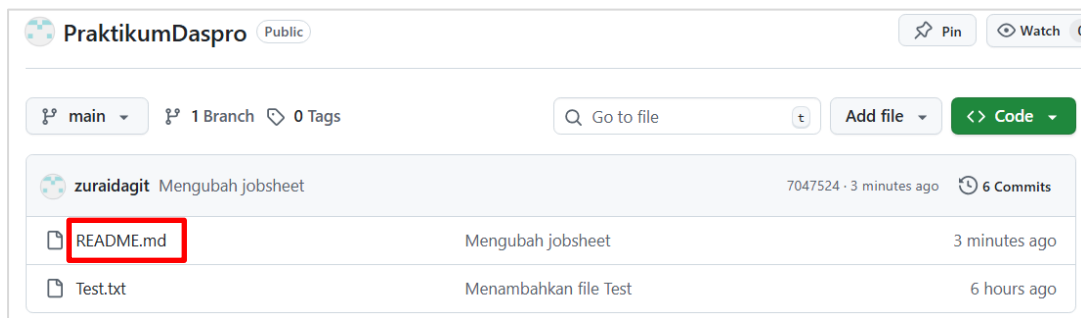
2.2 Percobaan 2 : Dasar Kolaborasi di Github

Waktu percobaan : 30 menit

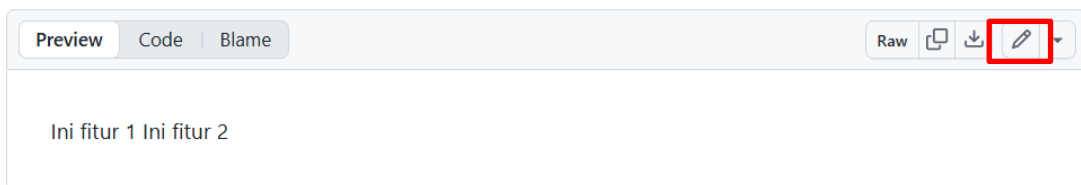
Selain sebagai repository pribadi, Github sering dimanfaatkan sebagai sarana kolaborasi untuk mengerjakan project bersama dengan tim. Lakukan percobaan berikut untuk melihat bagaimana kolaborasi dilakukan dengan Github.

1. Perlu diperhatikan bahwa ketika Anda berkolaborasi dengan tim pada sebuah project, lakukan **Pull** untuk memperbaharui versi project pada local repository Anda dengan versi terbaru di Github. Best practice nya, lakukan Pull setiap Anda mau mulai bekerja dan setiap kali akan melakukan Push.

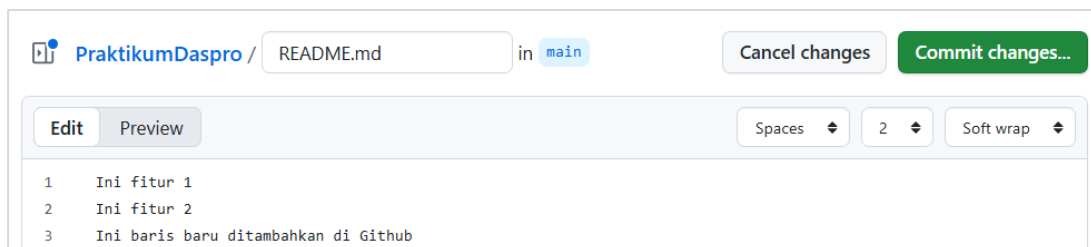
- Sebagai contoh, lakukan perubahan pada file README.md di Github. Klik pada nama file.



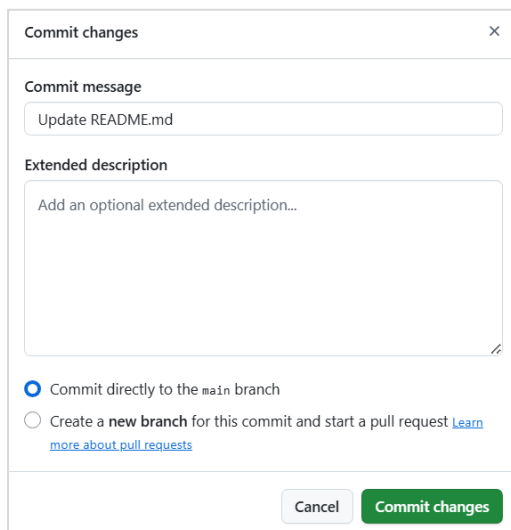
- Klik icon Edit



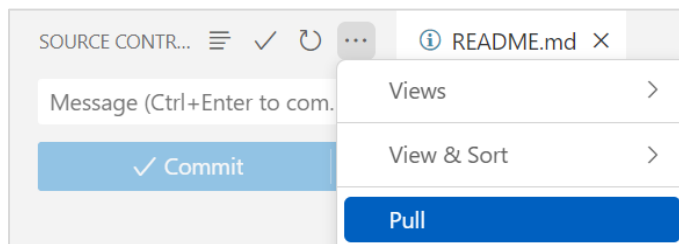
- Ubah file kemudian **Commit changes**



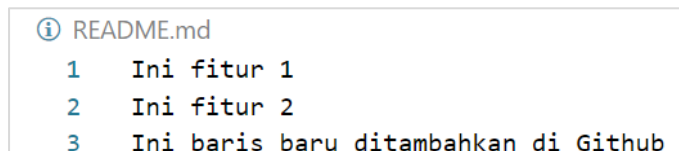
- Klik **Commit changes**



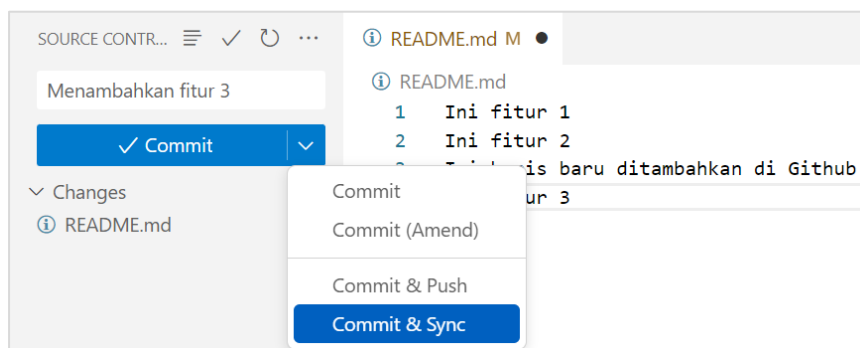
6. Untuk mengambil/mengunduh perubahan di Github, klik More (...) → **Pull**.



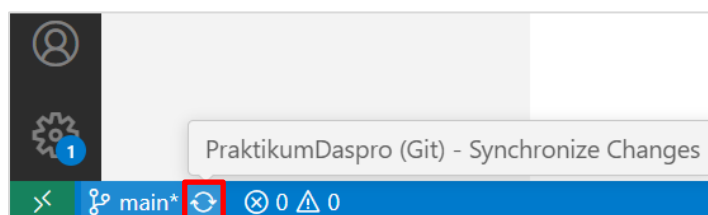
7. Setelah dilakukan Pull maka file README.md di local repository juga akan diupdate.



8. Anda juga dapat memilih opsi Commit & Sync. Sync berarti melakukan sinkronisasi local repository dan Github, jadi akan dilakukan Pull dari Github kemudian dilakukan Push ke Github secara langsung. Untuk mencobanya, tambahkan baris baru di README.md kemudian lakukan **Add** lalu **Commit & Sync**.



9. Opsi lainnya adalah menggunakan icon **Sync** di status bar.

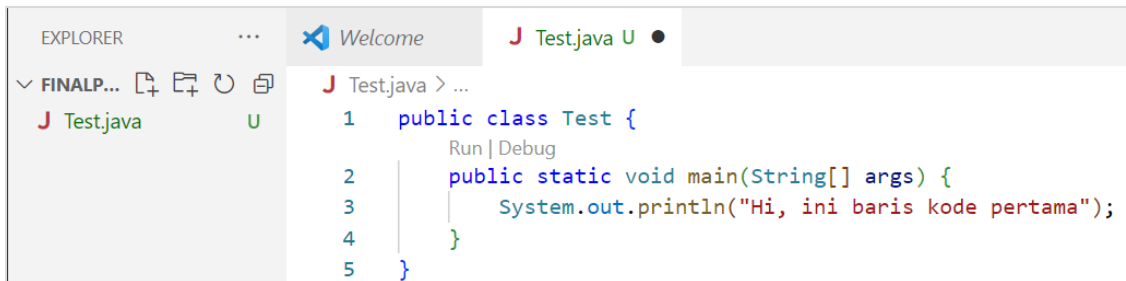


10. Sync sebaiknya juga hanya dilakukan jika repository dikelola secara pribadi. Jika bekerja dengan tim, lakukan **Pull** dan **Push** secara terpisah untuk meng-handle konflik yang mungkin terjadi dengan lebih baik.

Tugas

Waktu pengerjaan : 60 menit

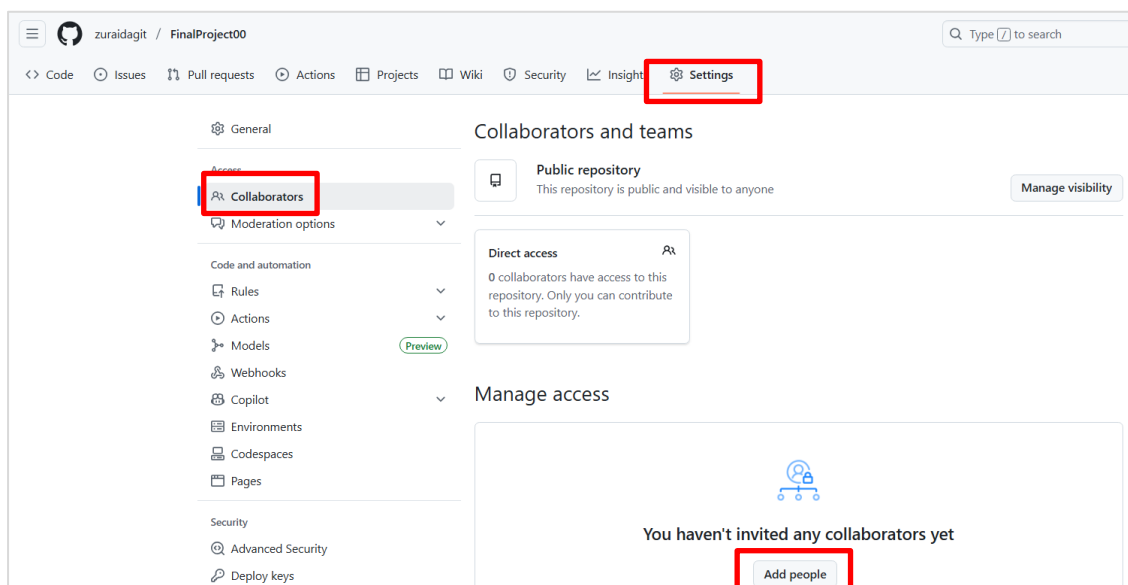
1. Untuk mencoba melakukan kolaborasi, buat repository baru dengan nama FinalProject<NoAbsen>.
2. Lakukan cloning ke local repository seperti pada Percobaan 1.
3. Cobalah tambahkan file baru misalnya Test.java yang berisi fungsi main().



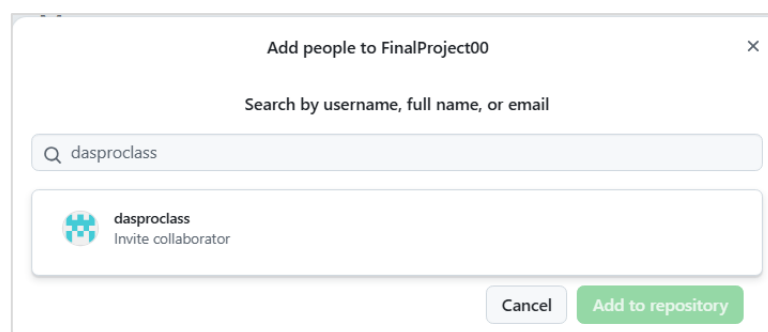
```

1 public class Test {
2     public static void main(String[] args) {
3         System.out.println("Hi, ini baris kode pertama");
4     }
5 }
    
```

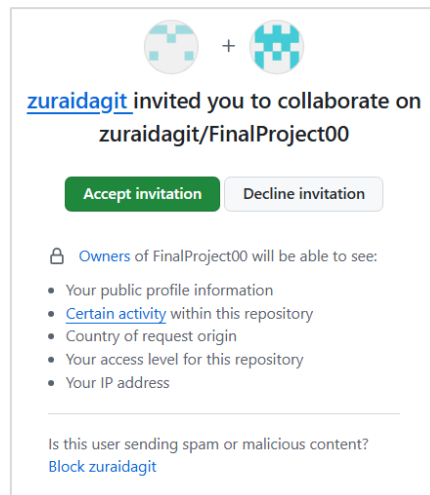
4. Lakukan Add, Commit, dan Push.
5. Anda dapat berkolaborasi dengan anggota tim dengan pilih Settings → Collaborators → Add people



6. Sebagai contoh, tambahkan 1 teman Anda berdasarkan username Github mereka. Invite collaborator kemudian **Add**.



7. Minta teman Anda untuk membuka email yang dikirimkan Github. Klik link “Accept or decline this invitation”. Pilih “Accept invitation”.



8. Minta teman Anda untuk meng-clone repository tersebut ke laptop/PC nya kemudian lakukan perubahan pada file Test.java atau menambahkan file baru. Lakukan Add, Commit, dan Push agar perubahan yang dia lakukan tersimpan di repository Github.
9. Lakukan Pull agar local repository Anda juga menyimpan versi project terbaru.
10. Bekerjalah bergantian dengan meng-clone repository teman Anda kemudian lakukan beberapa percobaan lagi dengan github (add, commit, push, pull) agar Anda terbiasa. Screenshot riwayat commit dari Github.

Catatan: Merge Conflict

Merge Conflict adalah masalah yang terjadi saat Git tidak bisa menggabungkan (merge) dua version secara otomatis, misalnya saat dua orang melakukan perubahan pada baris yang sama (atau berdekatan) pada sebuah file. Di samping itu, merge conflict juga bisa terjadi saat perubahan dilakukan pada file yang ternyata dihapus oleh orang lain.

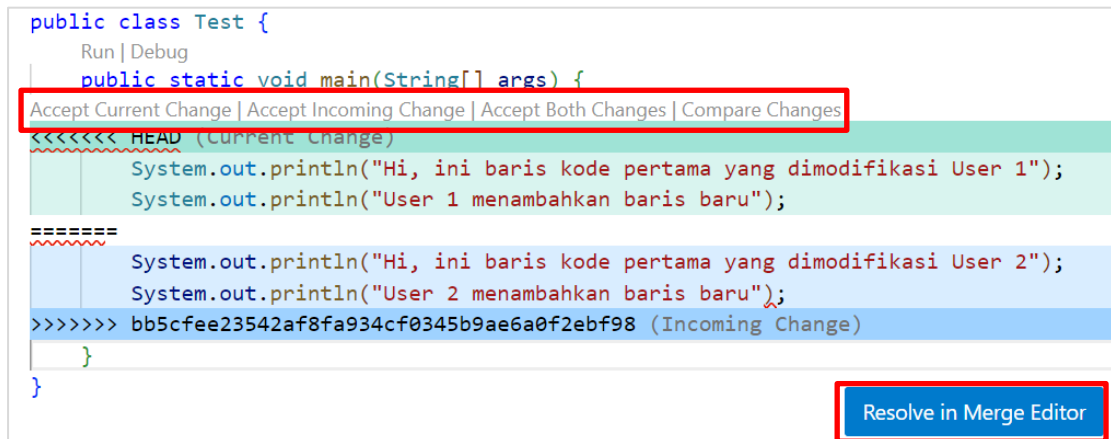
1. Sebagai contoh, misalnya Test.java diubah oleh Anda sebagai User1 sebagai berikut.

```
public class Test {
    Run | Debug
    public static void main(String[] args) {
        System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 1");
        System.out.println("User 1 menambahkan baris baru");
    }
}
```

2. User2 juga mengubah file Test.java (terhadap version yang sama) dan sudah di-push ke Github.

```
public class Test {
    Run | Debug
    public static void main(String[] args) {
        System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 2");
        System.out.println("User 2 menambahkan baris baru");
    }
}
```

3. Saat dilakukan pull akan terjadi merge conflict. Current Change berwarna hijau adalah version di local repository Anda sementara Incoming Change berwarna biru adalah version yang ada di Github.

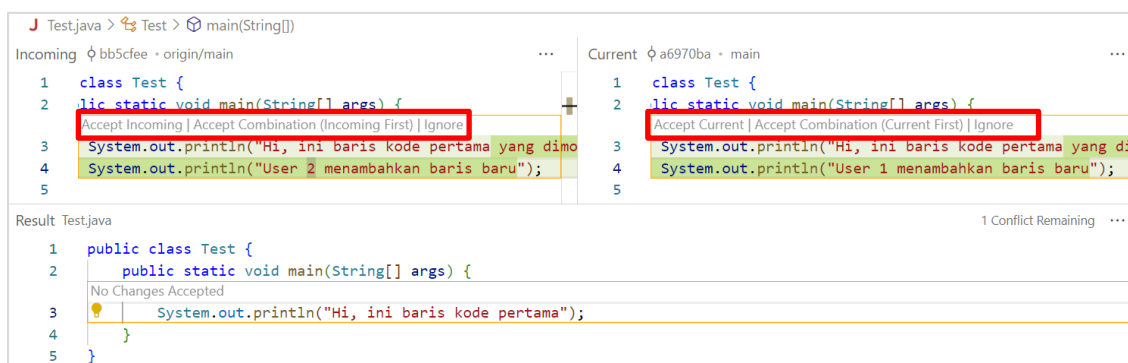


```

public class Test {
    Run | Debug
    public static void main(String[] args) {
        Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
        <<<<<<< HEAD (Current Change)
            System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 1");
            System.out.println("User 1 menambahkan baris baru");
        =====
            System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 2");
            System.out.println("User 2 menambahkan baris baru");
        >>>>>> bb5cfee23542af8fa934cf0345b9ae6a0f2ebf98 (Incoming Change)
    }
}
    
```

Resolve in Merge Editor

4. Anda dapat memilih beberapa opsi, yaitu:
 - **Accept Current Change:** memilih untuk menyimpan version local repository Anda dan mengabaikan version dari Github
 - **Accept Incoming Change:** memilih version yang di Github dan mengabaikan version local repository Anda
 - **Accept Both Changes:** menggabungkan kedua versi. Opsi ini biasanya akan menyebabkan error sehingga perlu diedit kembali secara manual
 - **Compare Changes:** membandingkan kedua version sebelum memilih.
5. Opsi lainnya adalah **Resolve in Merge Editor**.



```

J Test.java > Test > main(String[])
Incoming  bb5cfee · origin/main
1  class Test {
2  public static void main(String[] args) {
3  System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 1");
4  System.out.println("User 2 menambahkan baris baru");
5  }
}

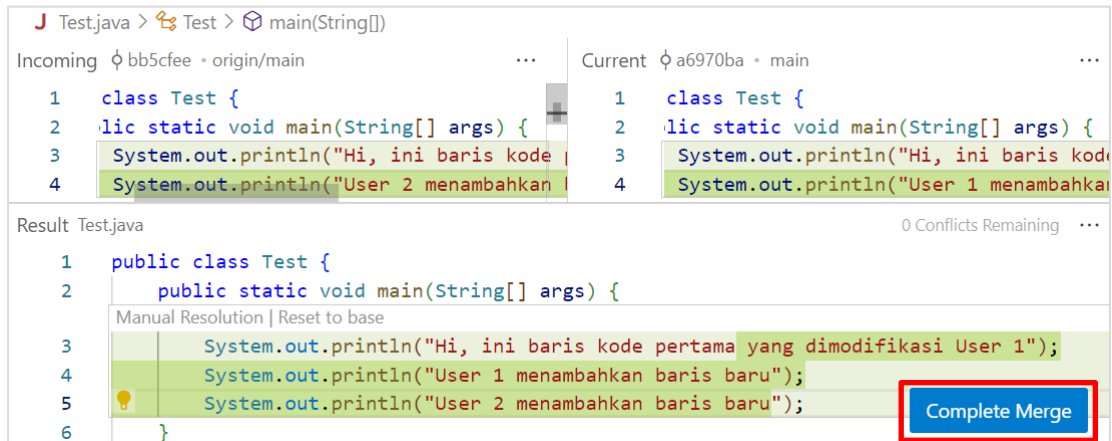
Current  a6970ba · main
1  class Test {
2  public static void main(String[] args) {
3  System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 2");
4  System.out.println("User 1 menambahkan baris baru");
5  }
}

Result Test.java
1  public class Test {
2  public static void main(String[] args) {
3  System.out.println("Hi, ini baris kode pertama");
4  }
5  }
    
```

1 Conflict Remaining

6. Anda dapat memilih version yang diinginkan pada bagian **Incoming** (klik Accept Incoming | Accept Combination(Incoming First) | Ignore) atau bagian **Current** (klik Accept Current | Accept Combination(Current First) | Ignore) kemudian modifikasi secara manual di bagian **Result** jika diperlukan.

7. Jika merge sudah sesuai, klik **Complete Merge**.



```

J Test.java > Test > main(String[])
Incoming  bb5cfee - origin/main
1  class Test {
2  public static void main(String[] args) {
3  System.out.println("Hi, ini baris kode pertama");
4  System.out.println("User 2 menambahkan baris baru");

Current  a6970ba - main
1  class Test {
2  public static void main(String[] args) {
3  System.out.println("Hi, ini baris kode pertama");
4  System.out.println("User 1 menambahkan baris baru");

Result Test.java
1  public class Test {
2  public static void main(String[] args) {
3  System.out.println("Hi, ini baris kode pertama yang dimodifikasi User 1");
4  System.out.println("User 1 menambahkan baris baru");
5  System.out.println("User 2 menambahkan baris baru");
6  }
    
```

8. Lakukan **Commit** kemudian **Push**.