

# Bab I

## Git & GitHub

Git adalah sebuah sistem kontrol versi terdistribusi yang diciptakan oleh Linus Torvalds pada tahun 2005. Git dirancang untuk menjadi cepat, efisien, dan mendukung pengembangan perangkat lunak yang besar dan terdistribusi. Git bekerja dengan menyimpan catatan perubahan-perubahan pada suatu proyek dan memungkinkan kolaborator untuk bekerja secara bersamaan tanpa harus terhubung secara online.

### A. Version Control System (VCS) - GIT

Git adalah software dengan sistem kontrol versi (VCS) yang mengelola dan melacak perubahan pada aplikasi, termasuk kode sumber. Setiap modifikasi dicatat rinci sehingga mudah mengetahui siapa yang mengubah dan kapan, sekaligus berfungsi sebagai cadangan untuk mengembalikan file ke versi sebelumnya bila terjadi kesalahan. Sistem ini penting untuk identifikasi dan perbaikan bug. VCS adalah sistemnya, sedangkan Git adalah alatnya. Analogi sederhananya, penggunaan Git mirip dengan kerja kelompok saat mengerjakan tugas coding bersama di kuliah.

Ketika membuat sebuah project sederhana secara bersama-sama atau kelompok, maka ketua kelompok harus membagi tugas ke beberapa anggota seperti pembagian fitur yang dikerjakan, kemudian masing-masing anggota mengerjakan baginya masing-masing tanpa perlu menunggu bagian lain selesai terlebih dahulu, jika semua bagian telah selesai maka perbagian itu akan dikumpulkan (*pull request*<sup>1</sup>) kepada ketua kelompok untuk digabungkan (*merge*<sup>2</sup>) menjadi satu project utuh.

Dalam penjelasan di kehidupan perkuliahan secara umum, git merupakan alat yang memantau setiap perubahan dalam dokumen yang kita kerjakan, kalau biasanya kita merevisi suatu dokumen itu terdapat banyak salinan/duplikasi berkas dokumen, namun dengan git kita hanya perlu memiliki satu berkas dokumen, jikapun kita ada kesalahan kita hanya perlu kembali ke versi revisi saja, bukan kembali ke berkas dokumen lain.

---

<sup>1</sup> istilah permintaan penggabungan kode dalam git

<sup>2</sup> istilah penggabungan kode dalam git

Sebelum ada GIT	Sesudah ada GIT
 Tugas Akhir ( Baru Mulai )	● <b>Dokumen Utama</b>
 Tugas Akhir ( Revisi 1 )	● <b>Dokumen Utama</b> : Menambah Cover
 Tugas Akhir ( Revisi 2 )	● <b>Dokumen Utama</b> : Menambah Abstrak
 Tugas Akhir ( Revisi 3 )	● <b>Dokumen Utama</b> : Menambah Teori
 Tugas Akhir ( Revisi 4 )	● <b>Dokumen Utama</b> : Merevisi Teori dan Isi
 Tugas Akhir ( Revisi 5 )	● <b>Dokumen Utama</b> : Menambah Gambar
 Tugas Akhir ( Revisi 6 )	● <b>Dokumen Utama</b> : Menambah Daftar Isi
 Tugas Akhir Selesai	● <b>Dokumen Utama</b> : Selesai

Ketika kita ingin menyimpan semua perubahan pada file, biasanya kita membuat salinan baru dengan menggunakan *save as*. Kemudian, salinan tersebut akan bertumpuk di dalam folder proyek seperti yang dijelaskan sebelumnya.

Namun, setelah menggunakan Git, sekarang hanya ada satu file dokumen dalam proyek dan perubahan pada file itu disimpan dalam database. Git hanya menyimpan perbedaan atau perubahan kecil saja, tanpa harus menyimpan seluruh isi file yang bisa memakan banyak ruang penyimpanan. Ini memungkinkan kita untuk kembali ke versi tertentu dari file yang telah direvisi sebelumnya.

## B. Mengenal GitHub

Sebelum kita mendalami berbagai perintah Git, mari kita mengenal GitHub. **GitHub** adalah platform penyimpanan dan kolaborasi proyek perangkat lunak yang menggunakan sistem kontrol versi Git. Secara sederhana, GitHub menyediakan layanan untuk menyimpan, mengelola, dan berkolaborasi pada kode sumber proyek perangkat lunak, yang dapat diakses melalui tautan <https://www.github.com>

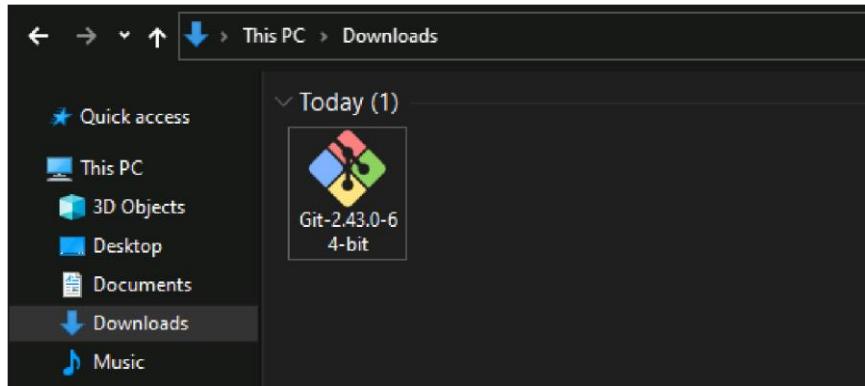
Meskipun sering disebut secara bersamaan, Git dan GitHub memiliki peran yang fundamental berbeda dalam ekosistem pengembangan modern. Git adalah *mesin* kontrol versi yang beroperasi secara lokal, menyediakan fungsionalitas inti untuk mengelola riwayat perubahan. Sebaliknya, GitHub adalah sebuah *platform* sosial berbasis *cloud* yang dibangun di atas fondasi Git. GitHub menyediakan layanan *hosting repository* yang memungkinkan tim untuk menyimpan dan mengelola kode mereka di server, mempermudah kolaborasi jarak jauh [User Query].

Platform GitHub memperluas fungsionalitas Git dasar dengan fitur-fitur manajemen proyek dan kolaborasi yang canggih yang tidak ada pada Git lokal. Fitur-fitur ini termasuk *bug tracking*, *task management*, forum diskusi tim melalui *GitHub Discussions*, dan papan proyek adaptif melalui *GitHub Projects*. *Developer* menggunakan Git untuk operasi personal dan *offline* pada repositori lokal mereka (misalnya, membuat *commit* dan *branch*), dan kemudian menggunakan GitHub sebagai perantara untuk berinteraksi dengan tim dan menyinkronkan pekerjaan mereka (misalnya, melalui *Pull Request* dan *push/pull*).

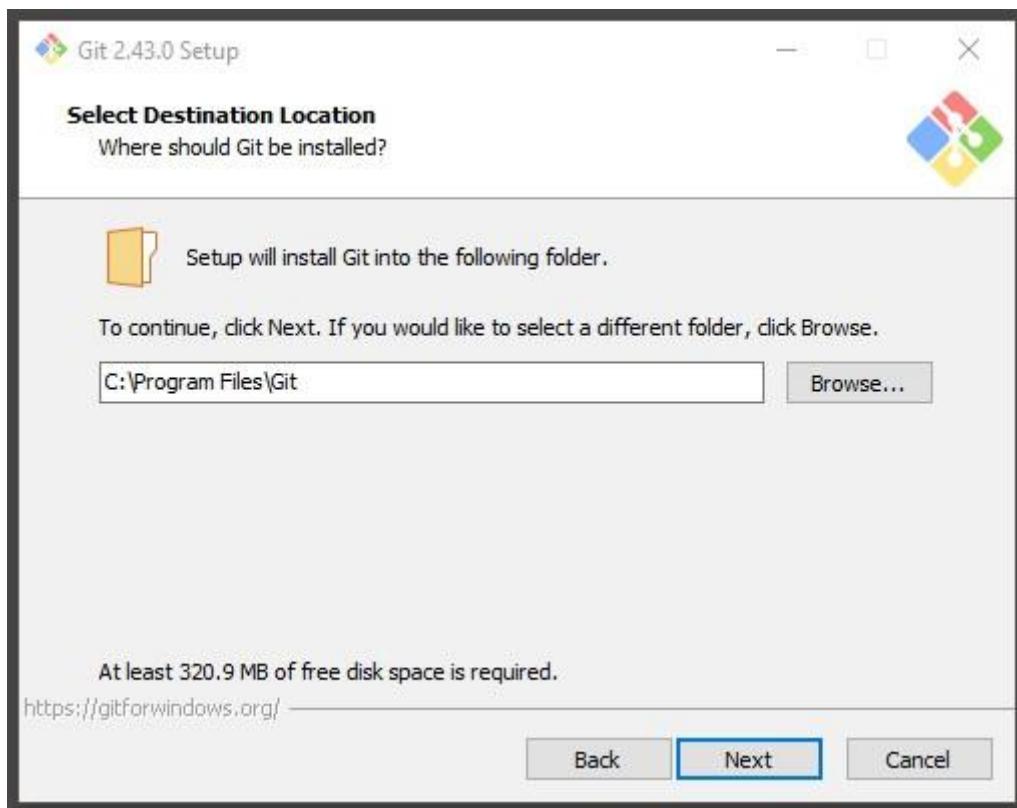
## C. Instalasi Git

Sebelum dapat memulai menggunakan Git, langkah pertama yang perlu dilakukan adalah menginstalnya di sistem operasi yang digunakan.

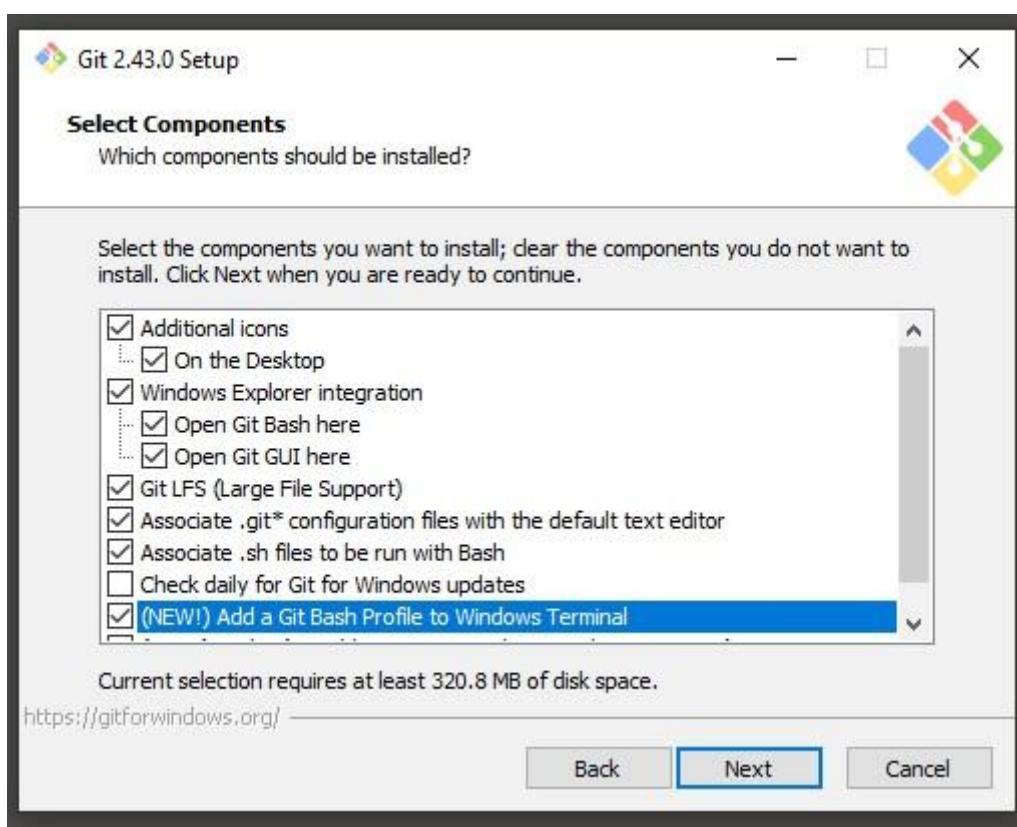
1. Unduh file instalasi git di <https://git-scm.com/downloads>
2. Kemudian taruh di folder mana saja, contoh pada folder *downloads*



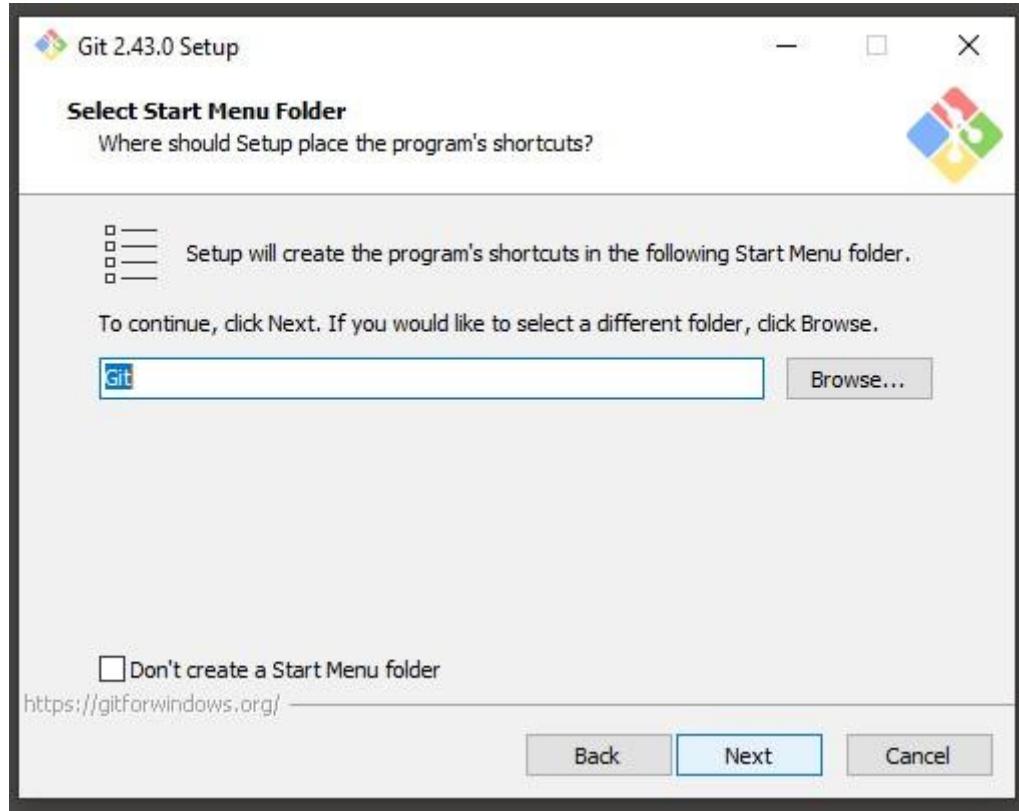
3. Klik dua kali pada file tersebut, maka akan muncul jendela instalasi.
4. Pada pemilihan lokasi instalasi, langsung saja klik ‘Next’



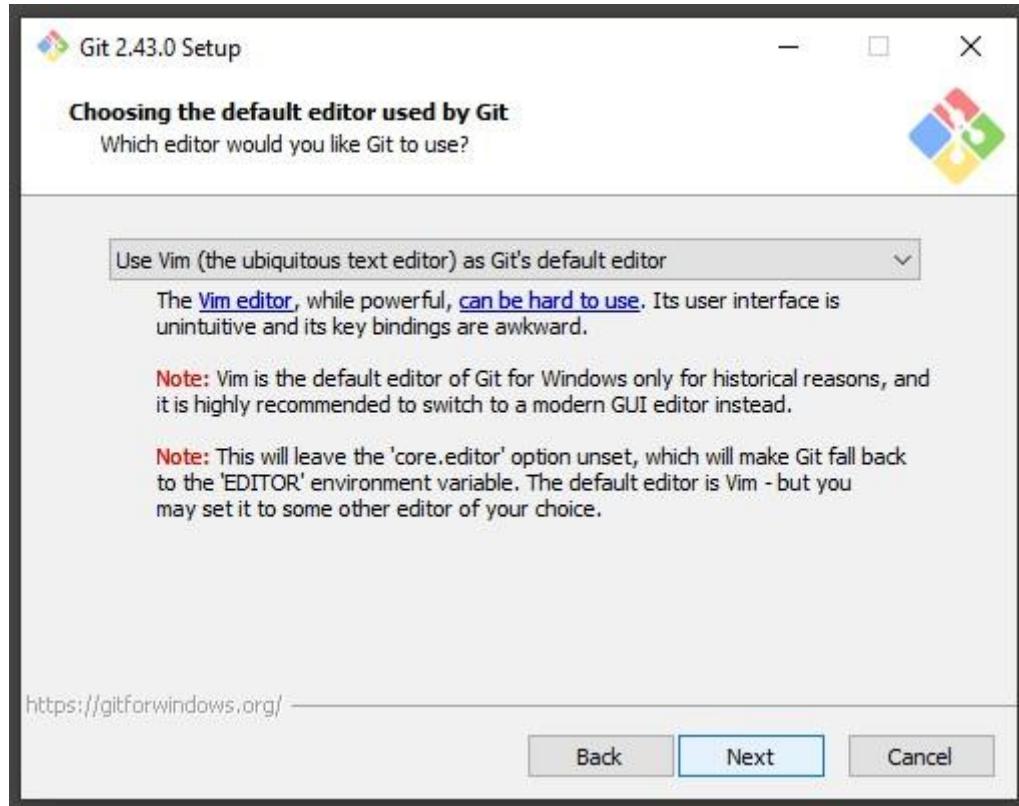
5. Kemudian, pada pemilihan komponen yang akan diinstal, sesuaikan dengan gambar di bawah ini.



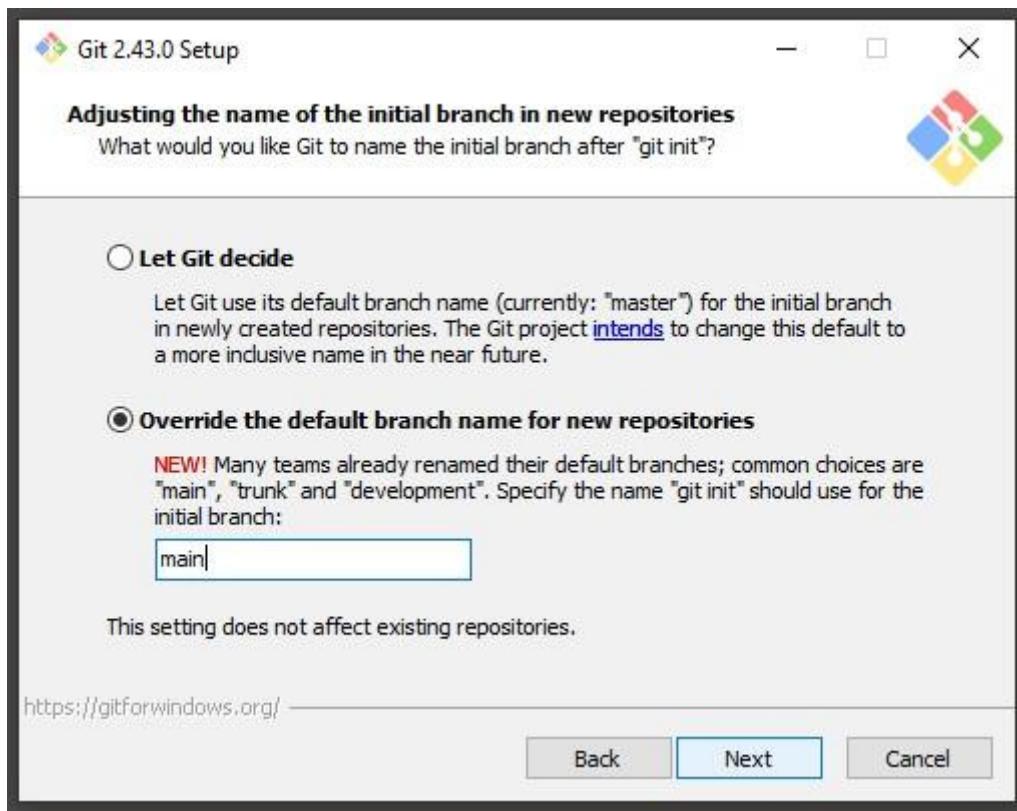
6. Beri nama shortcut icon git pada Start Menu.



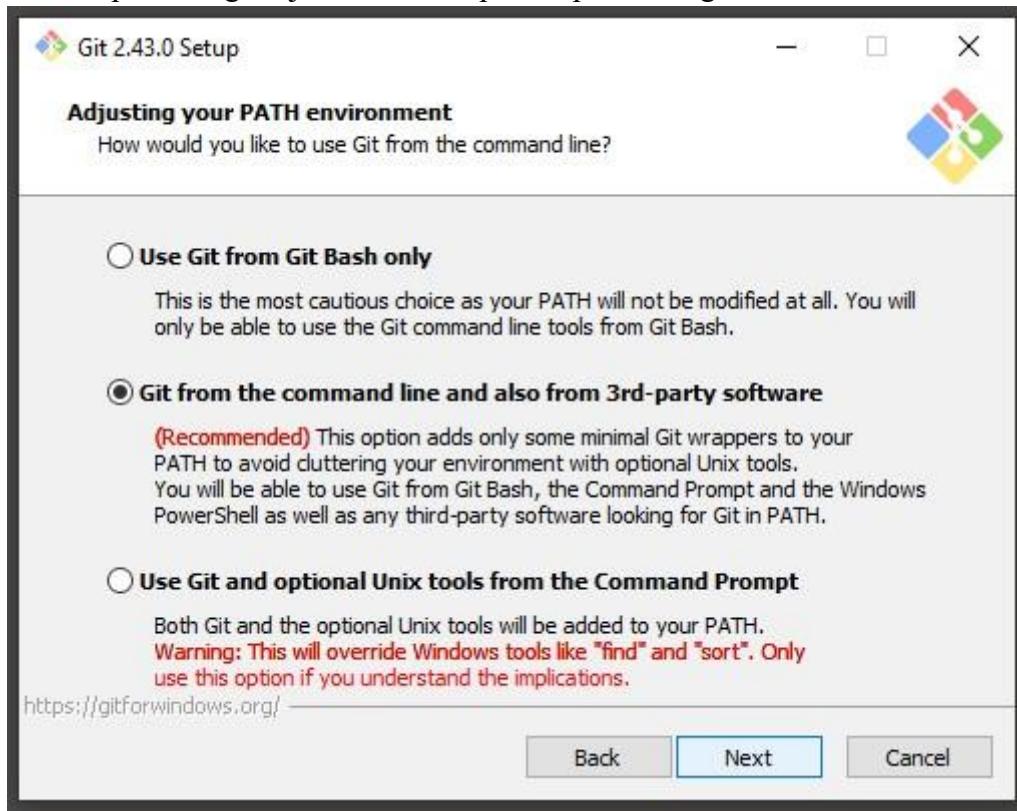
7. Pilih editor vim untuk default git editor



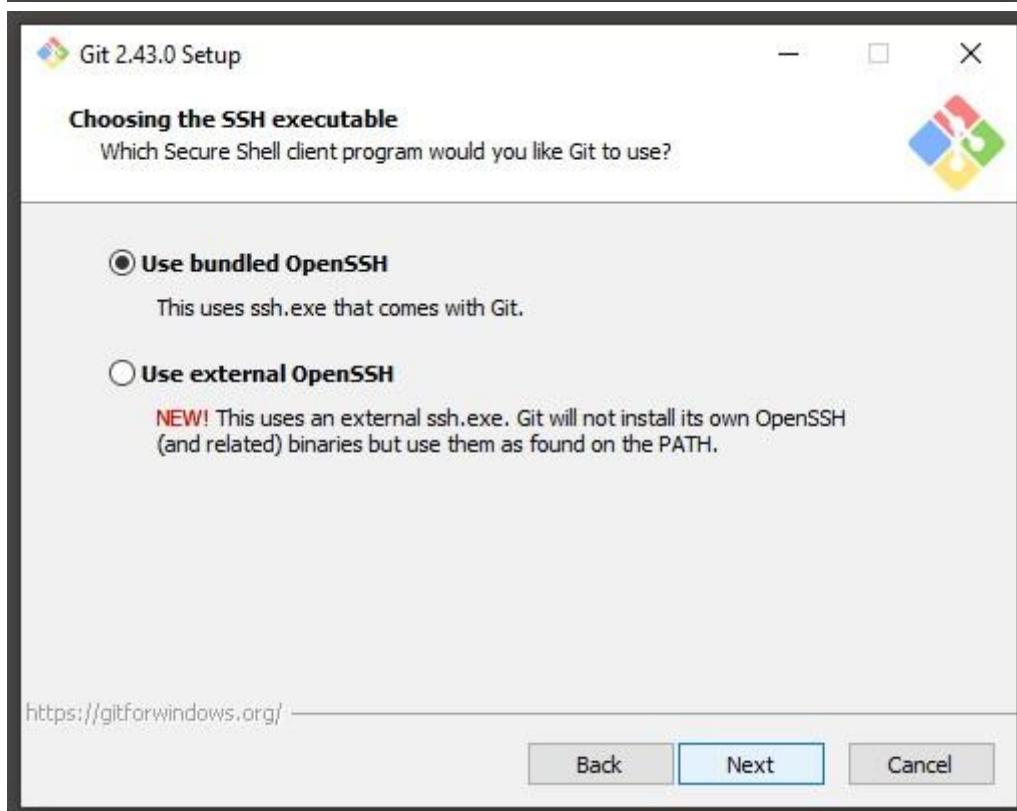
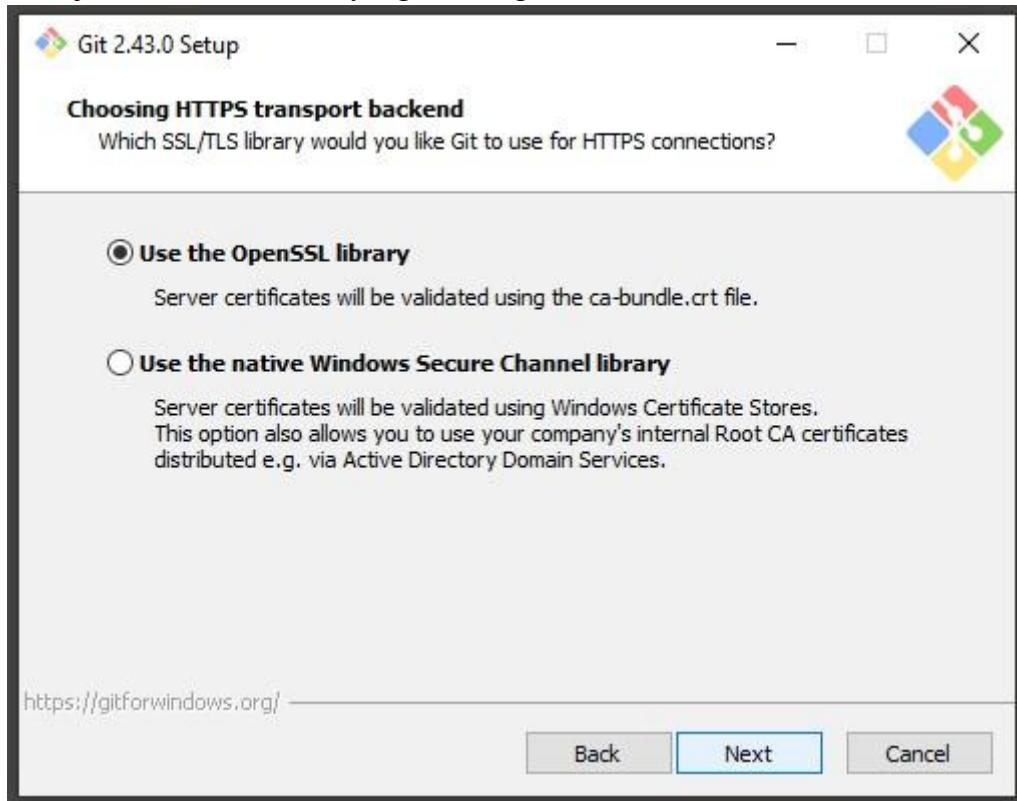
8. Ganti nama default branch ke “main” karena agar menyesuaikan default branch dari github nantinya.



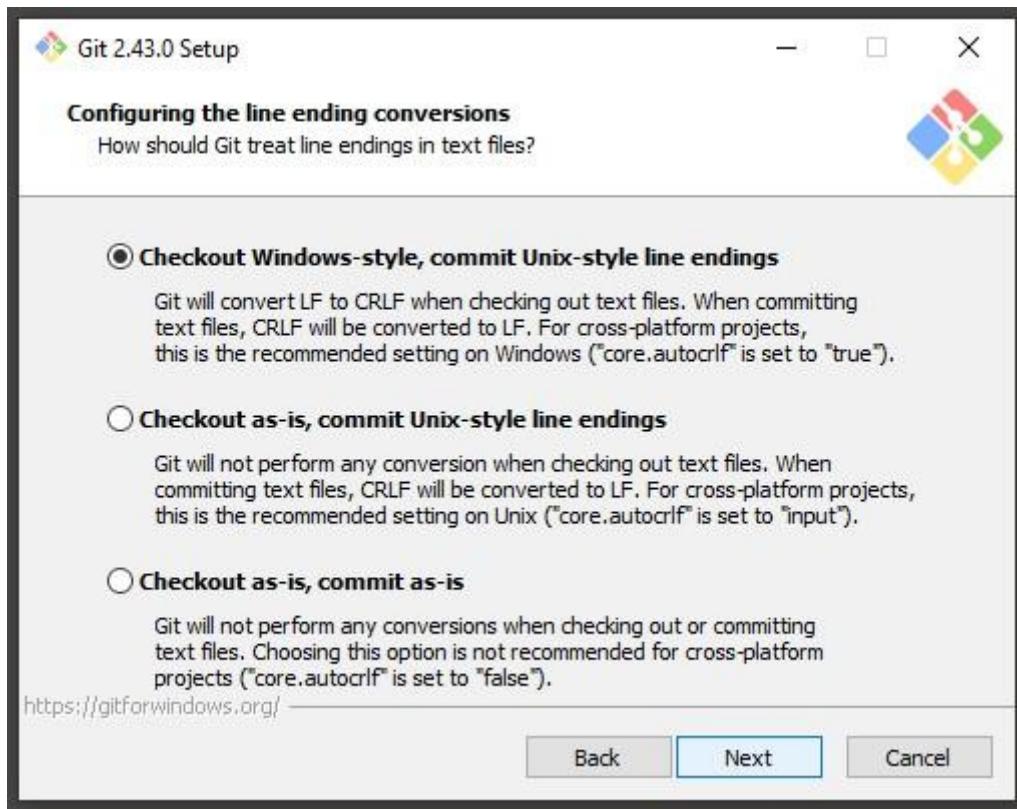
9. Izinkan perintah git dijalankan dari aplikasi pihak ketiga



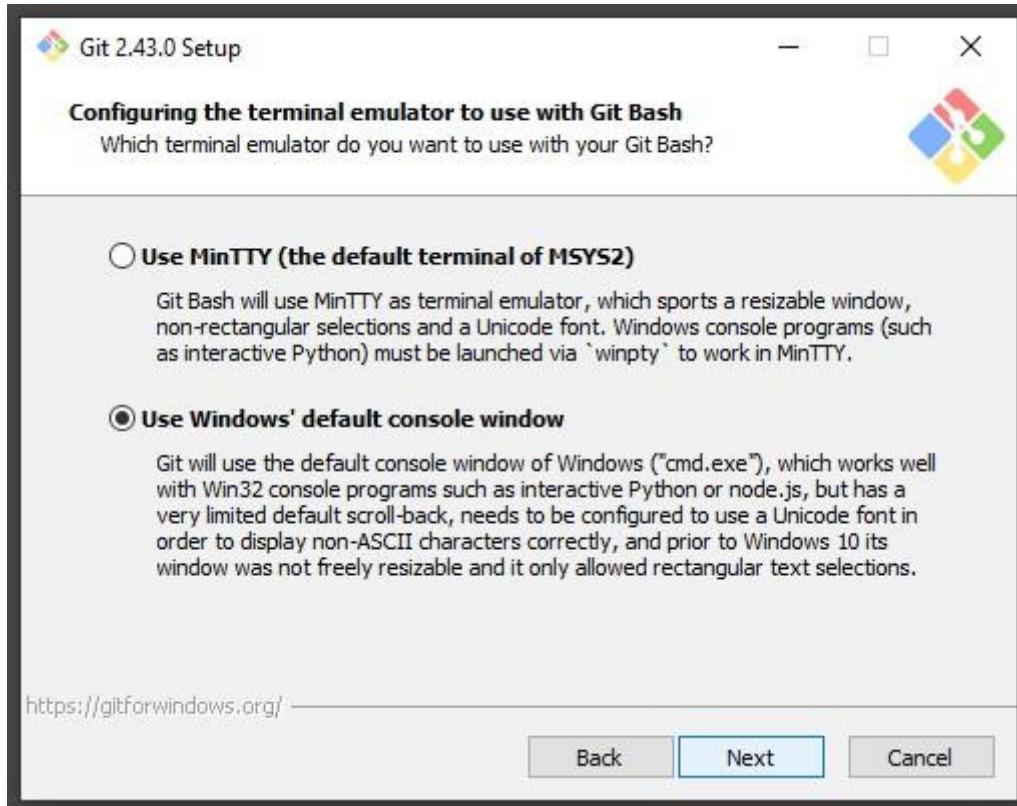
10. Pilih jenis SSH & HTTPS yang akan digunakan



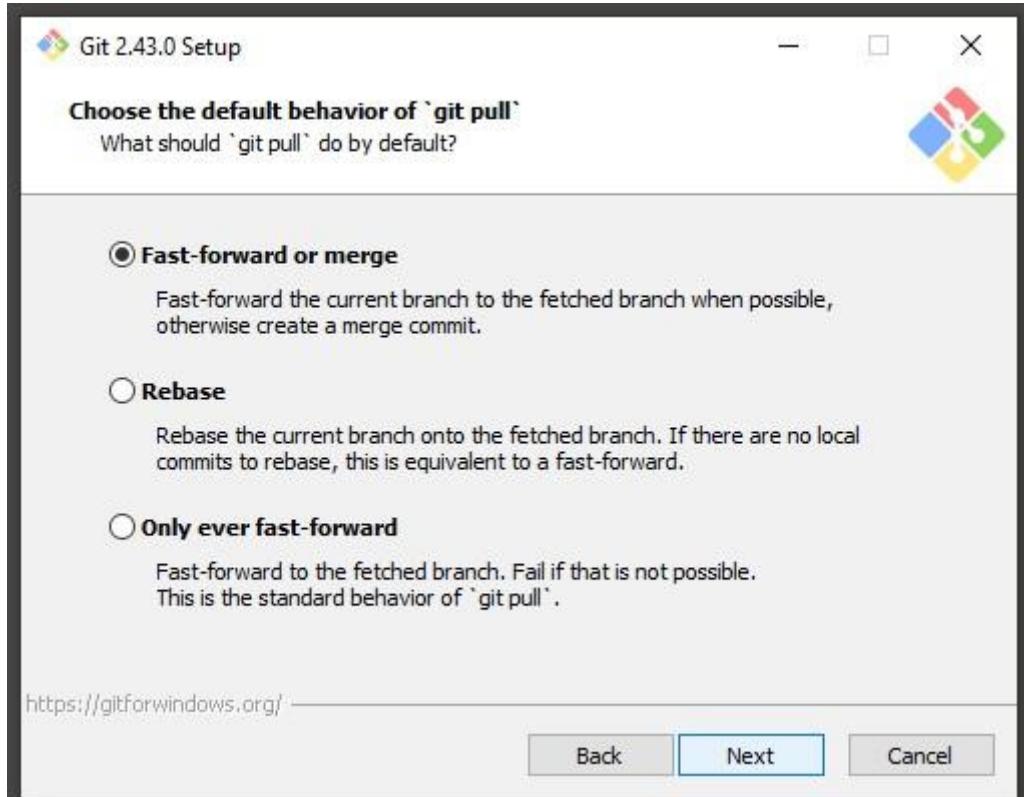
## 11. Konfigurasi *line-ending conversion*



## 12. Jadikan *cmd* menjadi default terminal untuk menjalankan git



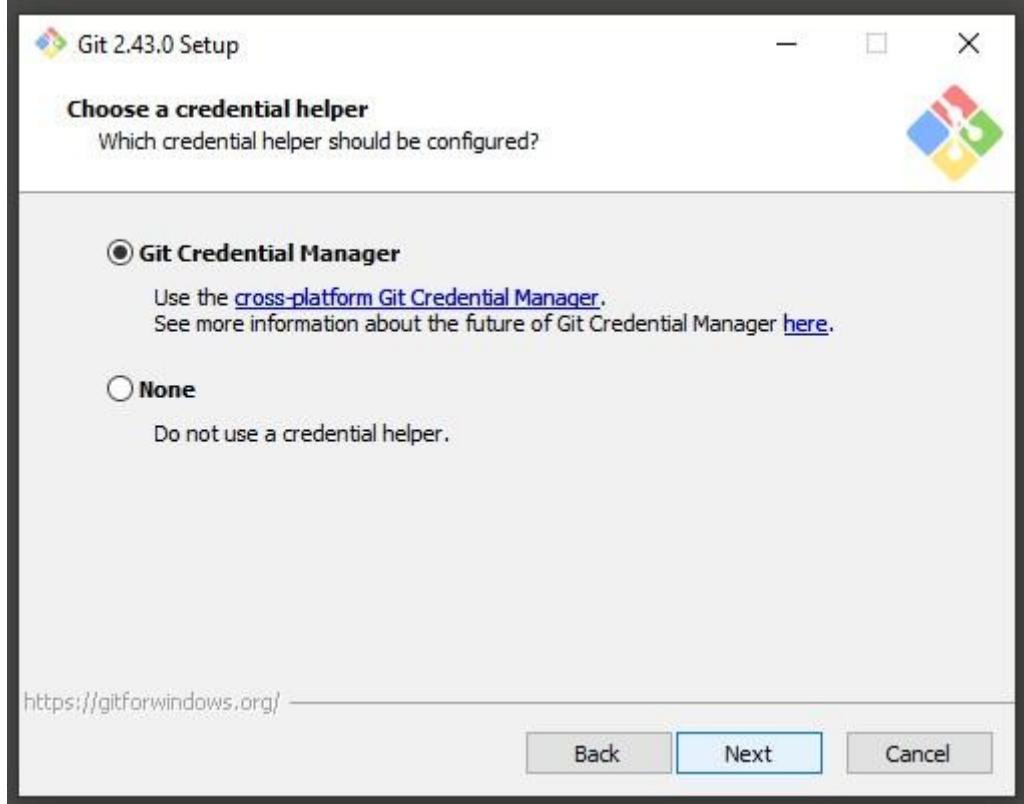
### 13. Konfigurasi pengaturan *git pull* dan *credential manager*



The screenshot shows the 'Choose the default behavior of `git pull`' step of the Git 2.43.0 Setup. It asks, 'What should `git pull` do by default?'. Three options are listed:

- Fast-forward or merge**: Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
- Rebase**: Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
- Only ever fast-forward**: Fast-forward to the fetched branch. Fail if that is not possible. This is the standard behavior of `git pull`.

At the bottom, there is a URL field containing <https://gitforwindows.org/>, and three buttons: Back, Next (highlighted in blue), and Cancel.

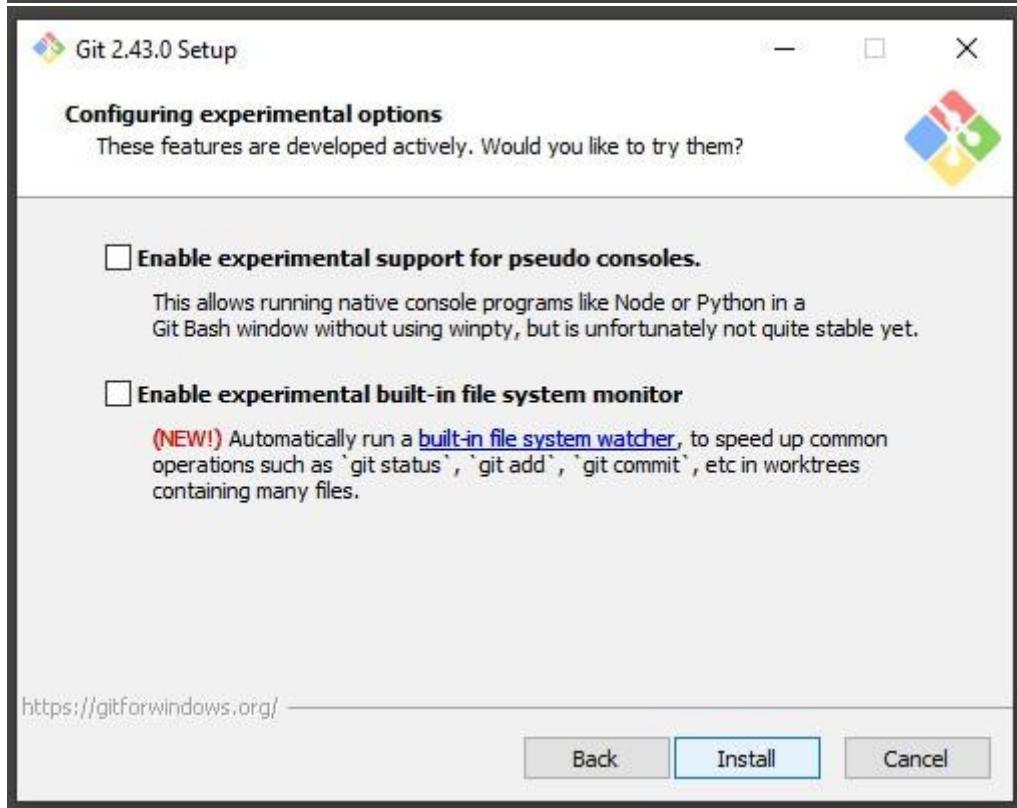
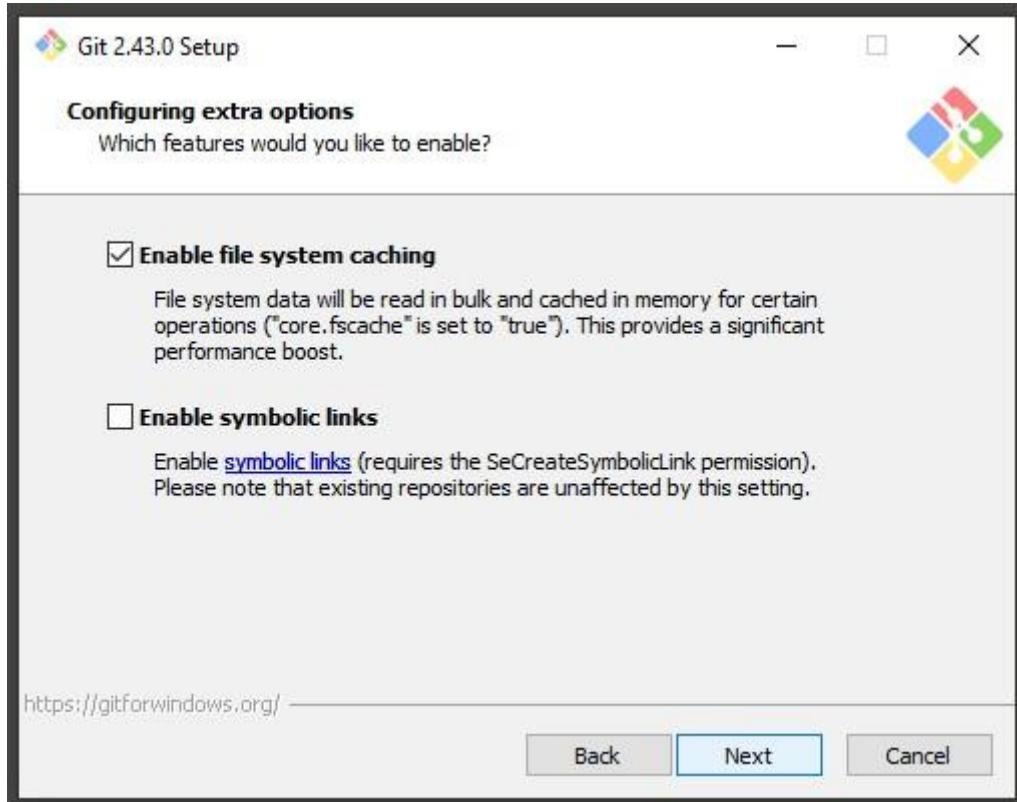
  


The screenshot shows the 'Choose a credential helper' step of the Git 2.43.0 Setup. It asks, 'Which credential helper should be configured?'. Two options are listed:

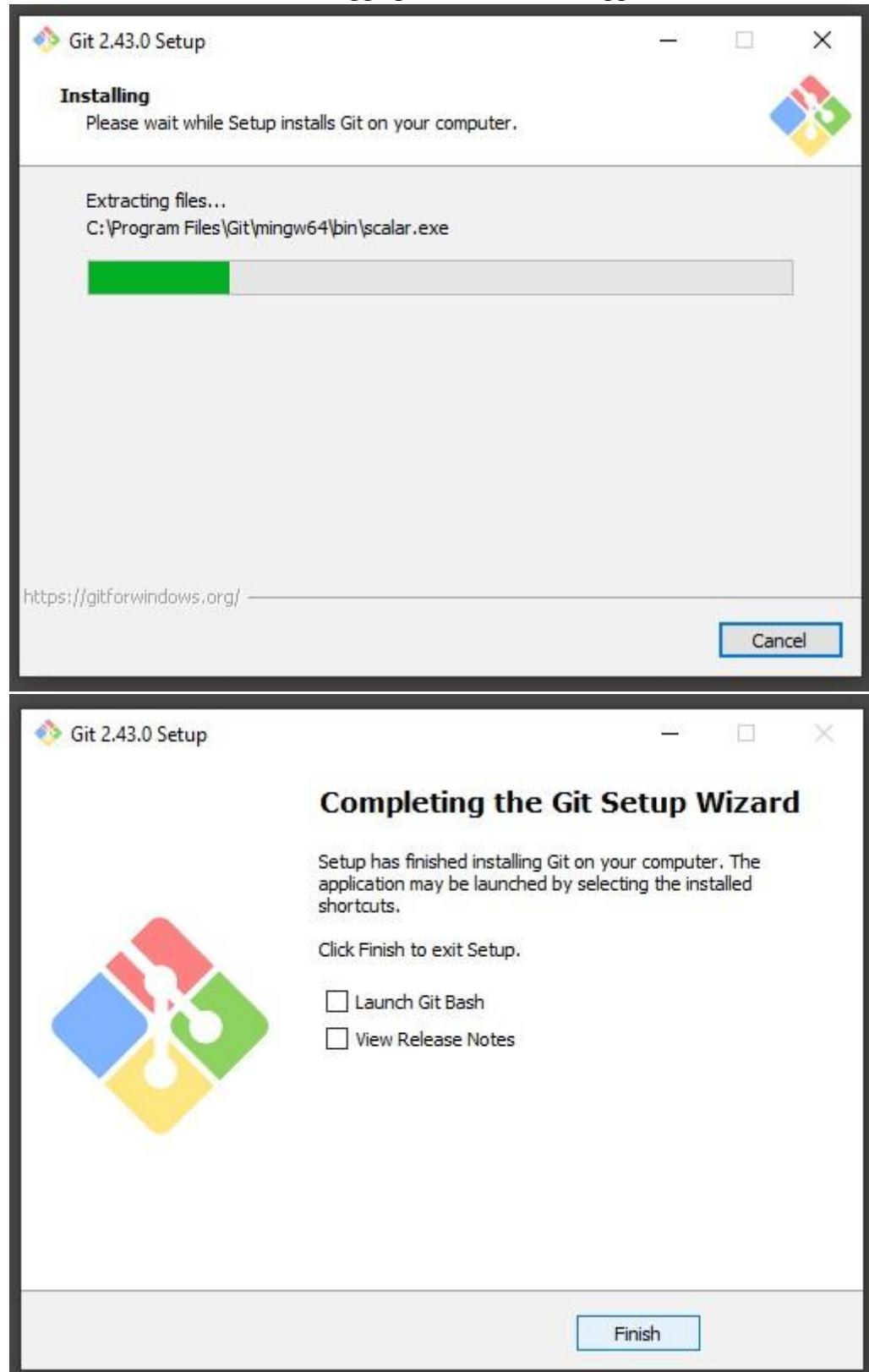
- Git Credential Manager**: Use the [cross-platform Git Credential Manager](#). See more information about the future of Git Credential Manager [here](#).
- None**: Do not use a credential helper.

At the bottom, there is a URL field containing <https://gitforwindows.org/>, and three buttons: Back, Next (highlighted in blue), and Cancel.

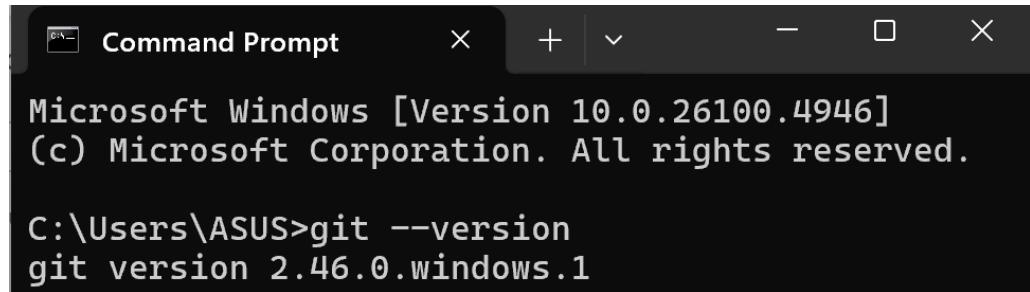
14. Atur pengaturan lainnya menjadi seperti ini.



15. Setelah itu, klik **Install** dan tunggu proses instalasi hingga selesai.



16. Cek apakah berhasil terinstall atau tidak dengan menjalankan perintah `git --version` pada command prompt atau cmd



```
Command Prompt
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>git --version
git version 2.46.0.windows.1
```

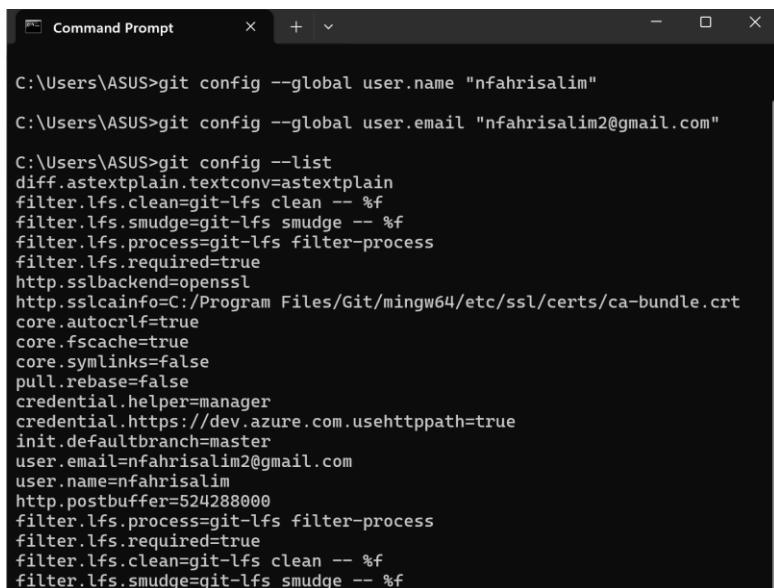
## D. Konfigurasi Git

Ketika mengatur git untuk pertama kalinya di komputer lokal, langkah pertama yang perlu dilakukan adalah melakukan konfigurasi awal. Terdapat beberapa elemen utama yang perlu yaitu tentukan nama pengguna kamu dan disarankan sesuai dengan nama pengguna yang akan terdaftar di github kemudian email aktif dan disarankan terhubung juga dengan akun github.

Jalankan perintah berikut ini melalui terminal/cmd/git bash dan diharapkan memiliki output yang sama seperti di gambar, yang menandakan kamu telah berhasil mengkonfigurasi git.

```
git config --global user.name "username-kamu" git config --
global user.email nama_email@example.com git config -list
```

Lakukan seperti di gambar :



```
Command Prompt
C:\Users\ASUS>git config --global user.name "nfahrisalim"
C:\Users\ASUS>git config --global user.email "nfahrisalim2@gmail.com"
C:\Users\ASUS>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=nfahrisalim2@gmail.com
user.name=nfahrisalim
http.postbuffer=524288000
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
```

## E. Perintah Esensial di Git

Setelah Git berhasil dikonfigurasi di komputer kita, langkah berikutnya adalah memahami perintah-perintah dasar yang paling sering digunakan. Perintah ini sangat penting karena menjadi alur utama dalam bekerja dengan Git, mulai dari menyimpan perubahan hingga membagikannya ke GitHub.

Secara umum, alur kerja Git melewati tiga area, yaitu **working directory** (tempat kita menulis dan mengedit file), **staging area** (tempat sementara untuk menampung perubahan yang dipilih), dan **repository** (tempat penyimpanan permanen dari riwayat perubahan).

Empat perintah inti yang perlu dikuasai adalah sebagai berikut:

1. `git add`

Perintah ini digunakan untuk memasukkan perubahan dari working directory ke staging area. Misalnya, jika kita baru saja mengedit file `home.txt`, maka kita perlu menjalankan:

```
git add home.txt
```

Dengan begitu, Git sudah menandai bahwa file tersebut siap dimasukkan ke commit berikutnya. Jika ingin menambahkan semua file yang berubah sekaligus, kita bisa gunakan:

```
git add .
```

2. `git commit`

Setelah perubahan sudah ada di staging area, kita perlu menyimpannya secara permanen di repository lokal dengan `git commit`. Setiap commit harus disertai pesan singkat yang menjelaskan perubahan.

```
git commit -m "Memperbaiki typo pada file home.txt"
```

Pesan commit ini penting supaya anggota tim lain tahu perubahan apa yang sudah dilakukan.

3. `git push`

Perintah ini berfungsi untuk mengirim commit yang sudah kita buat di repository lokal ke repository GitHub. Dengan begitu, tim lain bisa melihat hasil kerja kita.

```
git push origin main
```

Perintah di atas akan mendorong (push) commit dari cabang `main` di komputer lokal ke cabang `main` di GitHub.

4. `git pull`

Sebelum melakukan push, biasanya kita melakukan git pull terlebih dahulu. Tujuannya adalah mengambil perubahan terbaru dari GitHub agar repository lokal tetap mutakhir dan mengurangi kemungkinan konflik.

```
git pull origin main
```

Dengan cara ini, perubahan dari tim lain akan digabungkan dulu ke dalam repository lokal kita sebelum kita mengirim commit baru.

## F. Membuat Akun GitHub

Pada bagian ini, kita akan membahas langkah-langkah yang diperlukan untuk membuat akun Github, langkah awal yang esensial sebelum memulai perjalanan Anda dalam mengelola kode menggunakan Git dan Github.

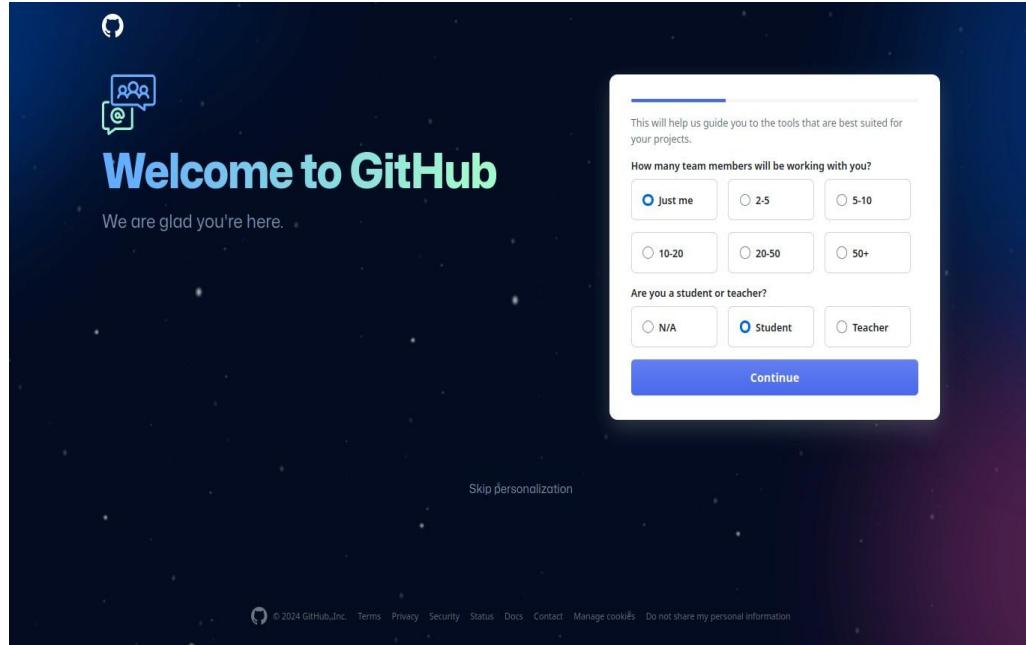
1. Mengakses situs resmi GitHub di <https://github.com>
2. Klik “Sign Up” untuk mendaftar.



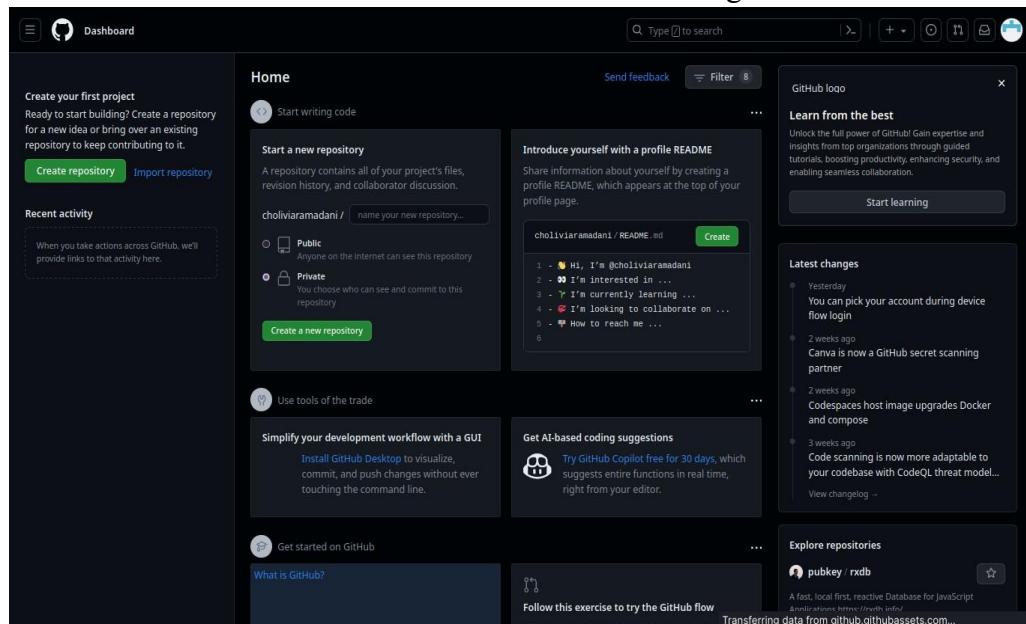
3. Masukan email aktif kamu, password, username yang mau di daftarkan!

The image shows the GitHub sign-up process. On the left, a dark panel displays the 'Create your free account' heading and a brief description of GitHub's core features. On the right, the 'Sign up for GitHub' form is shown. It includes fields for 'Email' (containing 'nfahrisalim@unhas.ac.id'), 'Password' (containing a masked password), and 'Username' (containing 'nfahrisalim'). An error message below the username field states: '⚠ Username nfahrisalim is not available. nfahrisalim-ops, nfahrisalim-det, or nfahrisalim3 are available. Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.' Other form fields include 'Your Country/Region' (set to 'Indonesia') and checkboxes for 'Email preferences' and 'Receive occasional product updates and announcements'. A 'Create account >' button is at the bottom.

- Setelah itu kamu akan disuruh verifikasi captcha dan memasukan OTP dari email.
- Pada bagian ini kamu bisa klik “skip” kalau tidak mau melengkapi



- Jika sudah maka akan diarahkan ke halaman dashboard github kamu



- Setelah itu, selamat kamu telah membuat akun github baru 😊😊

## G. Jenis-jenis Visibilitas Repository

Saat kita membuat repository baru di GitHub, kita akan diminta untuk memilih **jenis visibilitas**. Pilihan ini menentukan siapa saja yang bisa melihat dan mengakses repository tersebut. GitHub menyediakan tiga jenis visibilitas, yaitu **Public, Private, dan Internal**.

## 1. Public Repository

Repositori publik dapat dilihat oleh semua orang di internet. Cocok digunakan untuk proyek open source, portofolio pribadi, atau materi yang memang ingin dibagikan secara luas.

- Keuntungannya adalah memudahkan kolaborasi terbuka.
- Orang lain juga bisa memberikan kontribusi melalui *pull request*.

## 2. Private Repository

Repositori privat hanya bisa diakses oleh pemilik dan kolaborator yang sudah diundang.

- Biasanya digunakan untuk proyek internal, proyek tugas kelompok yang tidak boleh diakses umum, atau kode yang bersifat rahasia.
- Dengan ini, kode tetap aman karena tidak bisa dilihat oleh publik.

## 3. Internal Repository

Jenis ini khusus untuk akun organisasi atau perusahaan. Repository internal dapat diakses oleh semua anggota organisasi, tetapi tidak bisa dilihat oleh orang luar.

- Tujuannya adalah agar anggota tim dalam perusahaan bisa saling berbagi kode (mirip konsep open source, tapi hanya di dalam organisasi).
- Dengan cara ini, tim bisa lebih mudah menemukan kode yang sudah ada dan mengurangi pekerjaan yang berulang.

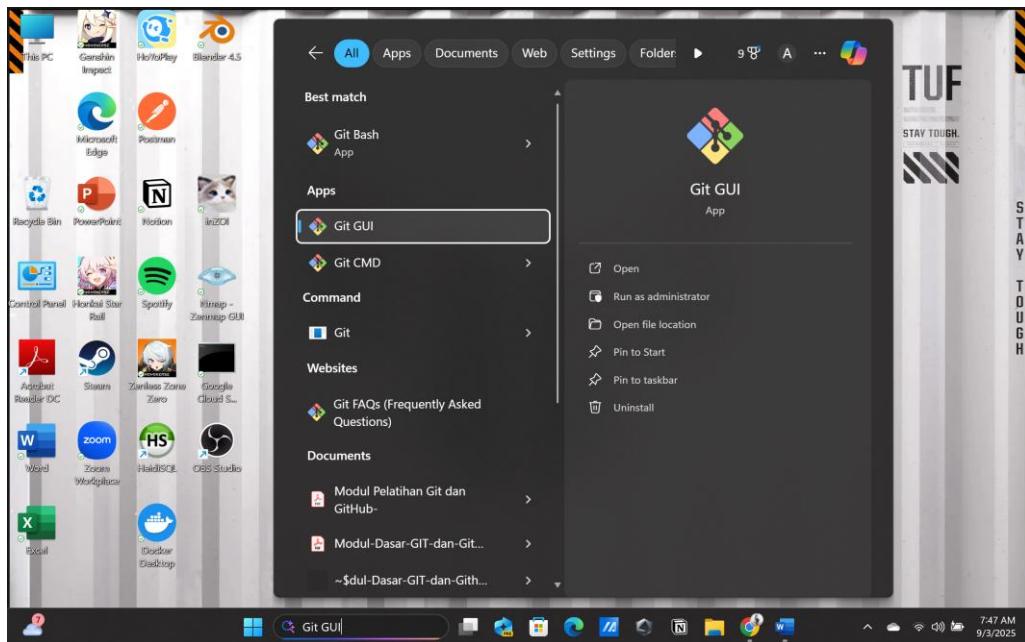
Berikut adalah tabel perbandingan fitur-fitur utama antara repositori publik, privat, dan internal:

Fitur	Repositori Publik	Repositori Privat	Repositori Internal
Visibilitas	Terbuka untuk semua orang di internet	Hanya pemilik dan kolaborator yang diundang	Semua anggota organisasi/perusahaan
Kolaborasi	Siapa saja bisa ikut berkontribusi melalui <i>fork</i> dan <i>pull request</i>	Hanya kolaborator yang diundang yang bisa berkontribusi	Semua anggota perusahaan dapat berkolaborasi
Kasus Penggunaan	Proyek open source, portofolio, demo	Proyek rahasia, kode sensitif, tugas internal	Proyek internal perusahaan, praktik <i>innersource</i>
Keamanan	Kurang aman secara default, karena bisa dilihat publik	Sangat aman, hanya yang diundang yang bisa akses	Aman di dalam lingkup perusahaan
Dampak	Membantu meningkatkan reputasi dan visibilitas pengembang	Melindungi informasi penting dan data sensitif	Mendorong efisiensi, berbagi kode, dan kolaborasi antar tim di perusahaan

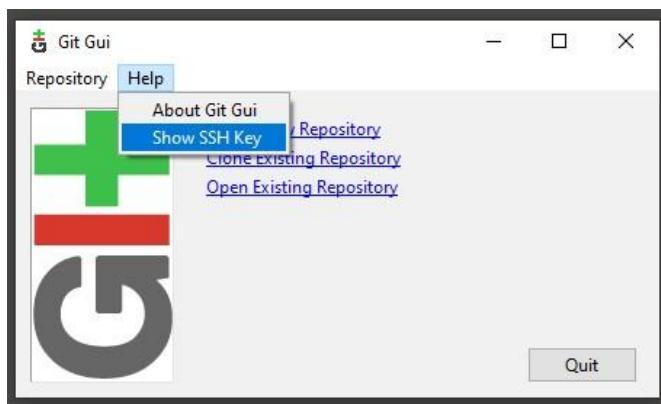
## H. Menghubungkan Git ke GitHub

Pada bagian ini, kita akan membahas langkah-langkah menghubungkan Git dengan GitHub menggunakan metode Secure Shell (SSH). Penggunaan SSH tidak hanya meningkatkan keamanan tetapi juga mempermudah akses ke repository GitHub tanpa perlu memasukkan kata sandi setiap kali berinteraksi. Ikuti langkah-langkah di bawah untuk mengkonfigurasi koneksi Git Anda ke GitHub menggunakan SSH.

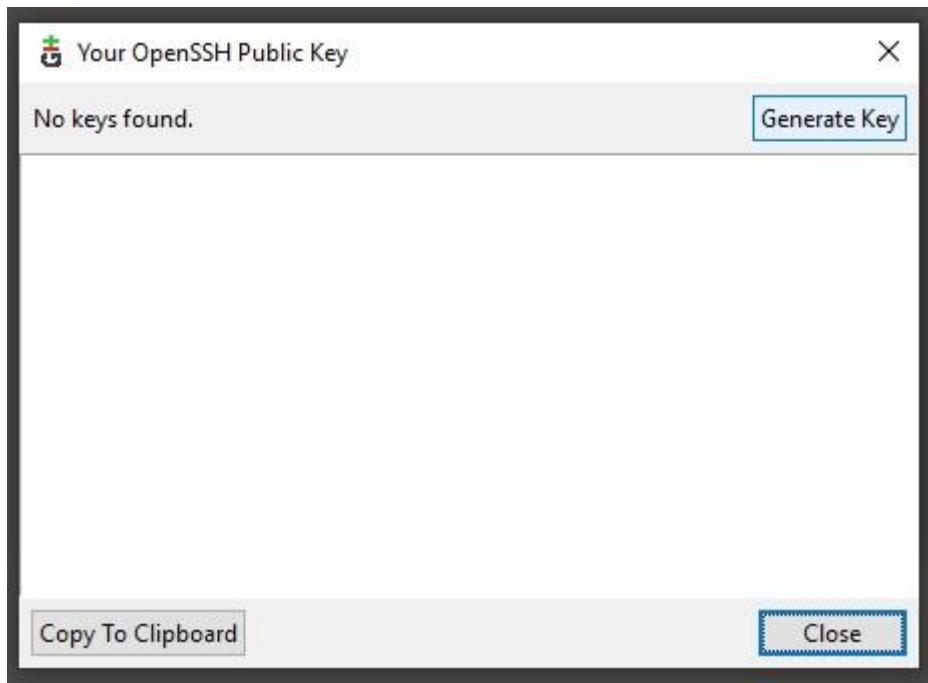
1. Buka GIT GUI di komputer kamu, bisa search aja di pencarian windows



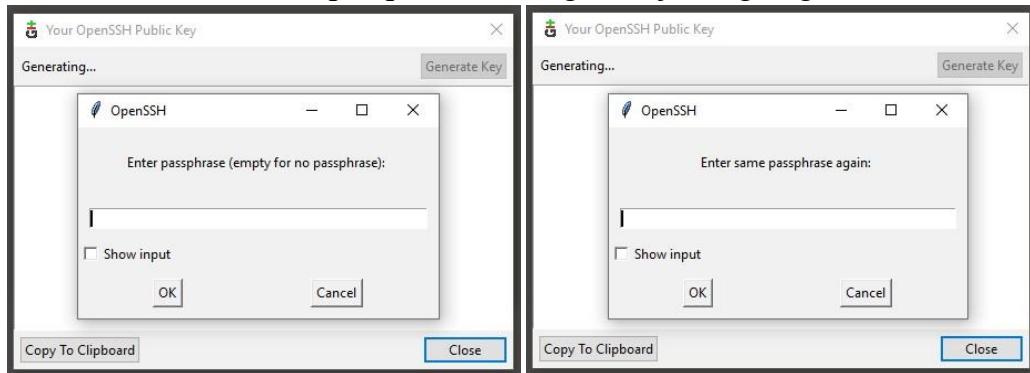
2. Setelah terbuka, klik bagian **help** lalu klik **Show SSH Key**



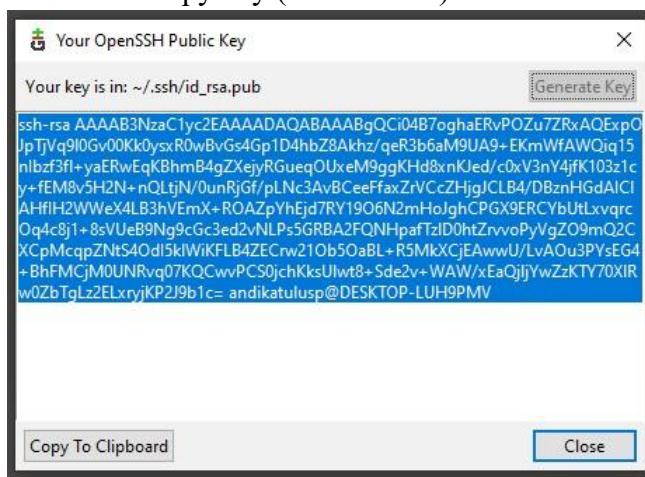
3. Klik tombol **Generate Key** untuk menghasilkan key baru



4. Jika disuruh memasukan *passphrase* kosongkan saja, langsung klik *ok*

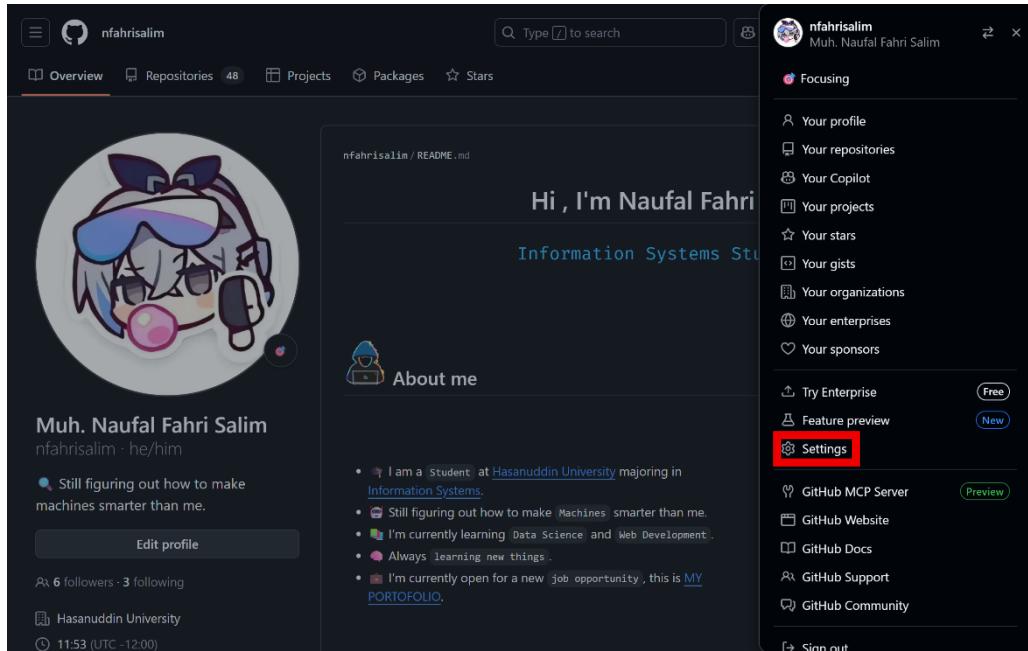


5. Setelah itu copy key ( CTRL + C ) atau tekan tombol *Copy To Clipboard*

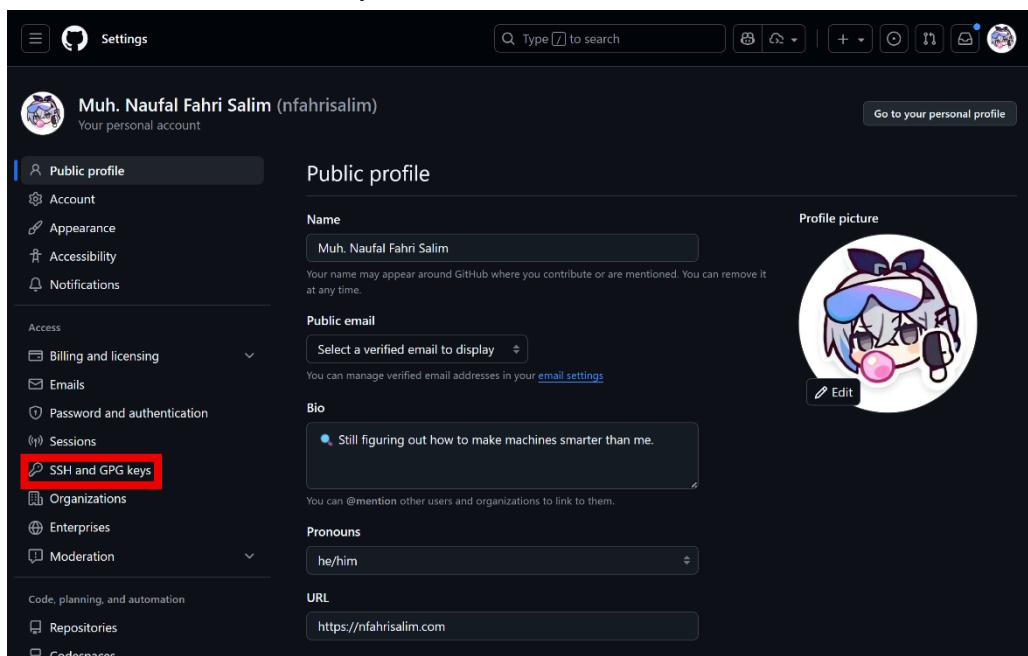


6. Setelah di copy, silahkan masuk ke akun github kamu atau akun github yang akan dihubungkan dengan repository lokal ( komputer kamu )

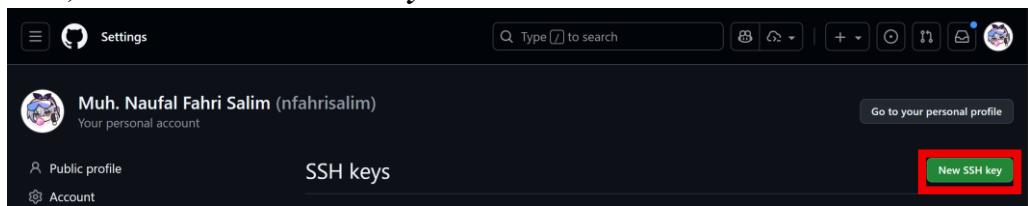
7. Setelah masuk, klik avatar/foto profil github kamu di bagian pojok kanan atas, kemudian pilih menu settings



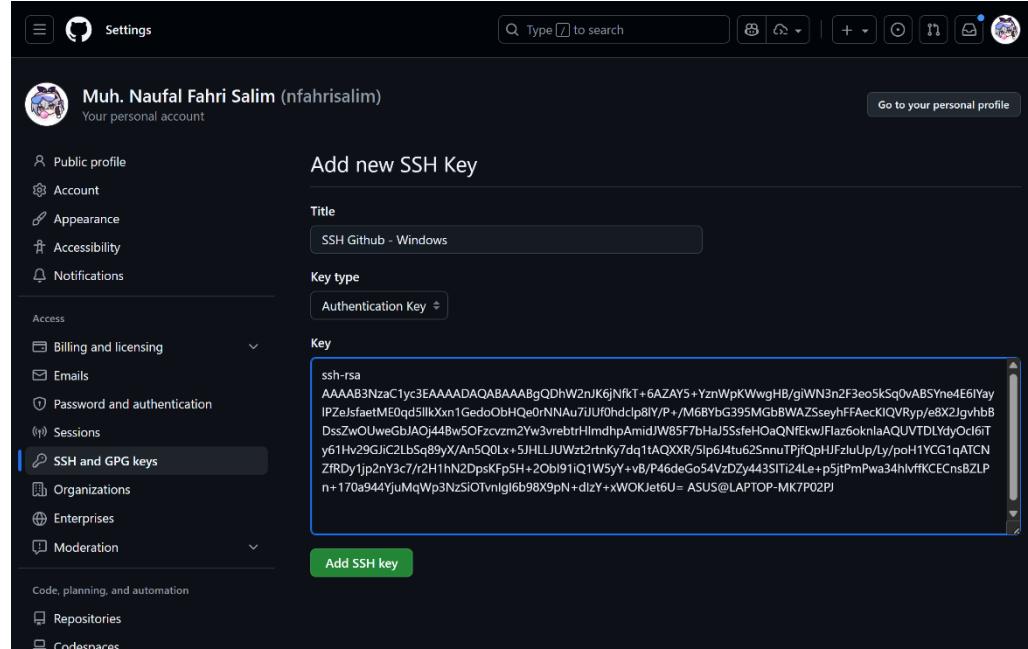
8. Klik menu ***SSH & GPG Keys***



9. Lalu, klik tombol ***New SSH Key***



10. Paste **SSH Key** yang telah kamu copy sebelumnya ke halaman ini



11. Jangan lupa kita cek apakah sudah terhubung atau tidak, maka dari itu kita gunakan perintah berikut ini di CMD atau Terminal

```
ssh -T git@github.com
```

12. Jika hasilnya seperti ini, maka sudah terhubung.

```
Command Prompt
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ASUS>ssh -T git@github.com
Hi nfahrисалим! You've successfully authenticated, but GitHub
does not provide shell access.

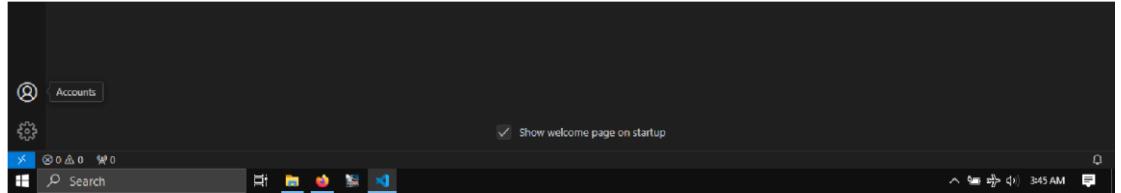
C:\Users\ASUS>
```

## I. Menghubungkan VS code ke GitHub

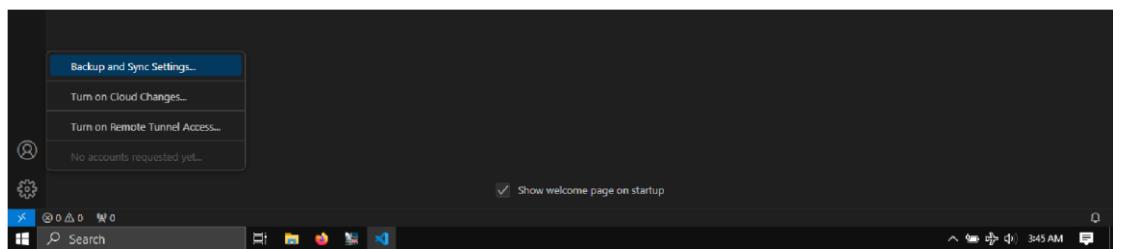
Pada bagian ini, kita akan membahas cara menghubungkan **Visual Studio Code (VS Code)** dengan **GitHub** sebagai platform untuk menyimpan dan mengelola proyek.

1. Pertama kita harus punya dulu akun github.

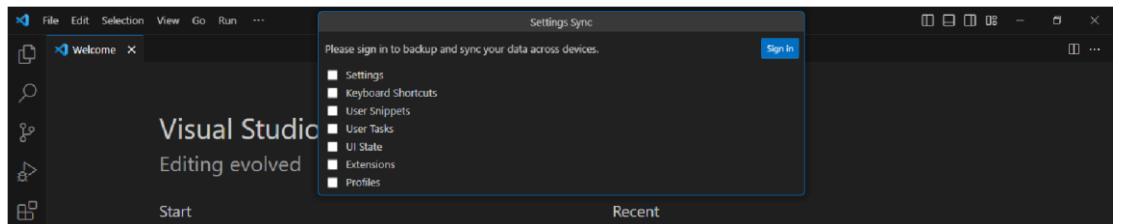
2. Kedua, komputer kita harus sudah terinstall Visual Studio Code.
3. Kalau kedua hal tersebut sudah terpenuhi, silahkan masuk ke akun github.
4. Kemudian buka aplikasi visual studio code dan klik icon account di pojok kiri bawah layar.



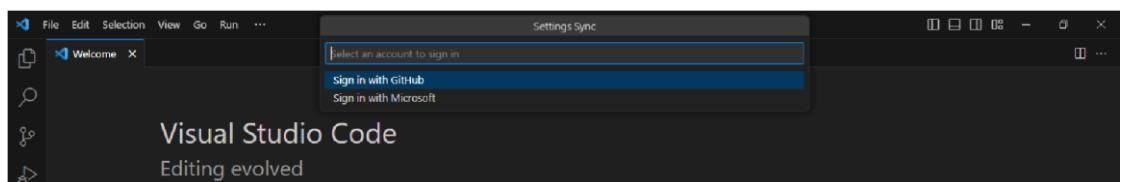
5. Klik menu **Backup and Sync Settings**



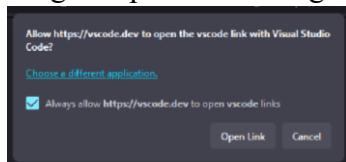
6. Lalu setelah muncul popup bisa klik tombol **sign in**



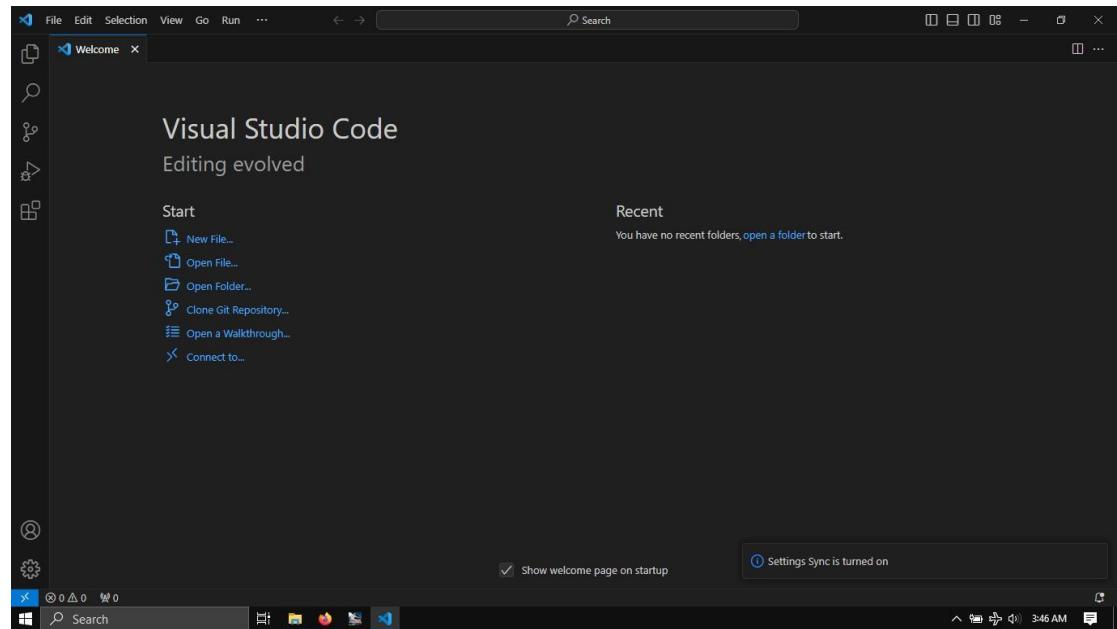
7. Kemudian pilih **Sign in with Github**



8. Jangan lupa klik centang **always allow** dan klik **open link**



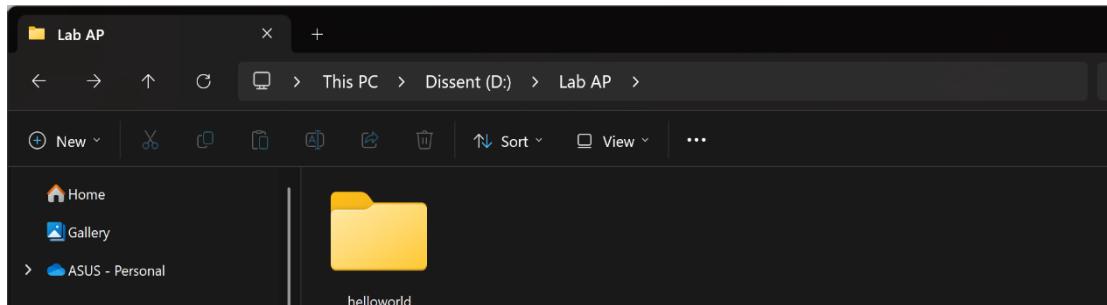
9. Jika diarahkan kembali ke vs code dan terdapat notifikasi **Settings Sync is turned on** di pojok kanan bawah layar, maka telah terhubung.



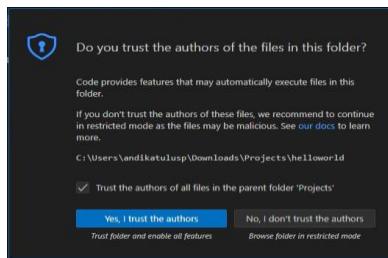
## J. Push Project ke GitHub

Pada bagian ini, kita akan memanfaatkan **Git** dan **GitHub** untuk mengelola proyek pemrograman secara lebih terstruktur dan profesional. Secara sederhana, istilah **push project** berarti mengunggah atau menyimpan kode yang telah kita kerjakan ke penyimpanan berbasis **cloud** melalui GitHub. Dengan cara ini, proyek kamu akan tersimpan dengan aman, dapat diakses dari mana saja, serta memudahkan proses kolaborasi dengan tim.

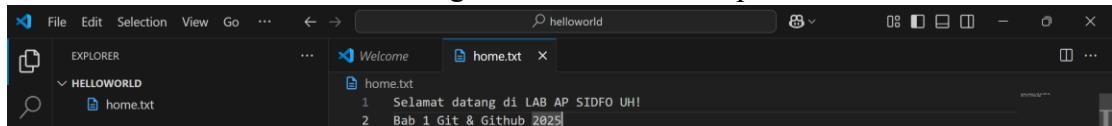
1. Buat sebuah project sederhana baru, disini saya akan membuat project **helloworld** yang berisi berkas *home.txt*



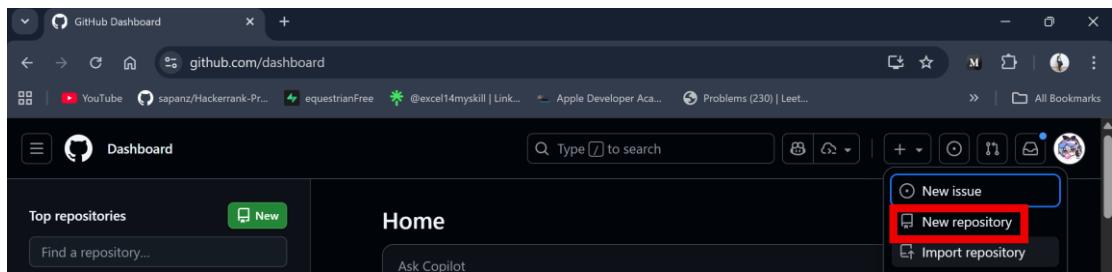
2. Buka folder project di vs code dan jika ada popup **trust folder author**, maka klik saja tombol **Yes, i trust the authors**



3. Kemudian, isi berkas `home.txt` dengan kode sederhana seperti ini.



4. Setelah project sudah siap di push ke github, kita tinggal buat repository baru di github, dengan klik **New Repository**



5. Konfigurasi detail repository sebagai berikut.

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).  
Required fields are marked with an asterisk (\*).

1 General

Owner \* nfahrisalim / Repository name \* helloworld  
 helloworld is available.

Great repository names are short and memorable. How about [upgraded-octo-meme](#)?

Description

0 / 350 characters

2 Configuration

Choose visibility \* Public

Add README Off

Add .gitignore No .gitignore

Add license No license

- Setelah klik tombol **Create Repository** maka akan diarahkan ke halaman langkah-langkah push project ke github.

The screenshot shows the GitHub repository creation interface. At the top, it says "Quick setup — if you've done this kind of thing before" with options for "Set up in Desktop" or "HTTPS / SSH" and a URL "https://github.com/nfahrisalim/helloworld.git". Below this, it says "Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#)". Underneath, there's a section titled "...or create a new repository on the command line" with the following code example:

```
echo "# helloworld" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/nfahrisalim/helloworld.git
git push -u origin main
```

- Kemudian kita buka CMD/Terminal atau Git Bash, lalu jalankan perintah berikut ini.

Inisialisasi Git di Folder Project

```
git init
```

Pindahkan semua file project ke *staging area*<sup>3</sup>

```
git add .
```

Lakukan *commit*<sup>4</sup> dengan disertai pesan commit atau pesan perubahan

```
git commit -m "First Commit"
```

Atur ke **branch**<sup>5</sup> atau cabang mana project akan di push

```
git branch -M main
```

Menghubungkan folder project dengan repository github

```
git remote add origin git@github.com:link_git_kamu
```

Terakhir, lakukan push ke github

```
git push -u origin main
```

- Kurang lebih seperti gambar berikut ini.

<sup>3</sup> Staging area berfungsi sebagai langkah penyaringan sebelum mengirim perubahan ke repository di GitHub. Ini memberikan kontrol tambahan terhadap perubahan yang akan di-commit.

<sup>4</sup> Commit merupakan tindakan untuk menyimpan perubahan pada repositori Git. Setiap commit memiliki pesan yang menjelaskan perubahan yang dilakukan

<sup>5</sup> Branch merupakan cabang yang akan menyimpan berkas project dengan fitur dan versi yang berbeda, ya istilahnya seperti folder didalam folder

```
C:\Windows\System32\cmd > + ^

D:\Lab AP\helloworld>echo "# helloworld" >> README.md

D:\Lab AP\helloworld>git init
Initialized empty Git repository in D:/Lab AP/helloworld/.git/

D:\Lab AP\helloworld>git add .

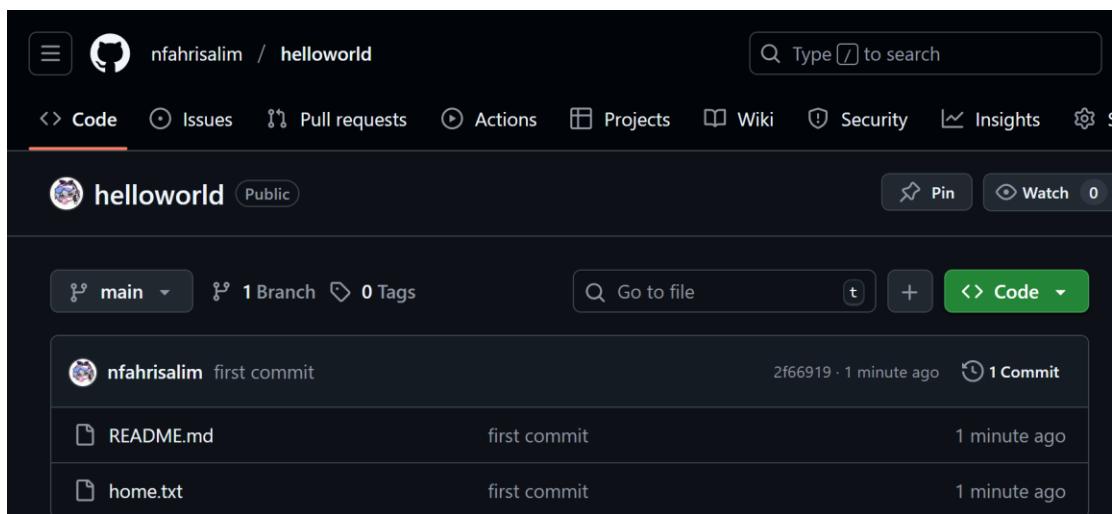
D:\Lab AP\helloworld>git commit -m "first commit"
[master (root-commit) 2f66919] first commit
 2 files changed, 3 insertions(+)
 create mode 100644 README.md
 create mode 100644 home.txt

D:\Lab AP\helloworld>git branch -M main

D:\Lab AP\helloworld>git remote add origin https://github.com/nfahrisalim/helloworld

D:\Lab AP\helloworld>git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 24 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 323 bytes | 323.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/nfahrisalim/helloworld.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

9. Kita cek, apakah kode kita sudah terupload atau belum dengan membuka link repository github kamu



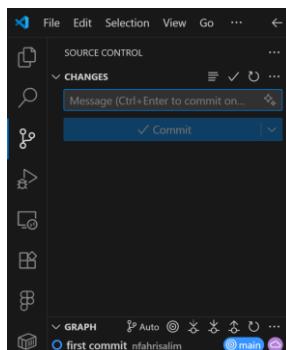
## K. Membuat Perubahan pada Project

Pada bagian ini, kita akan mencoba membuat perubahan pada kodingan kita di komputer lokal, lalu menyimpan perubahan itu di Github melalui GIT.

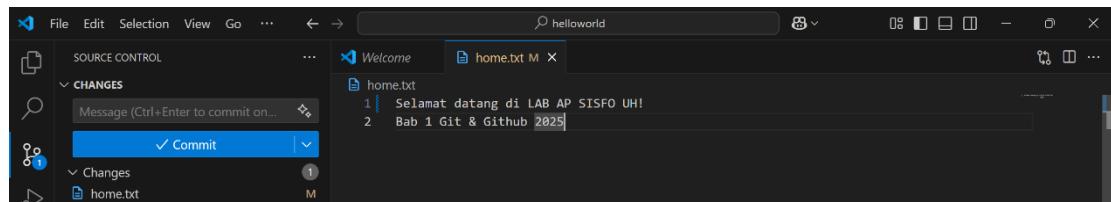
1. Sekarang kita akan fokus untuk manajemen project dengan Git melalui VS Code, buka project kamu di VS Code dan lihat apakah di icon source control terdapat notifikasi jam?



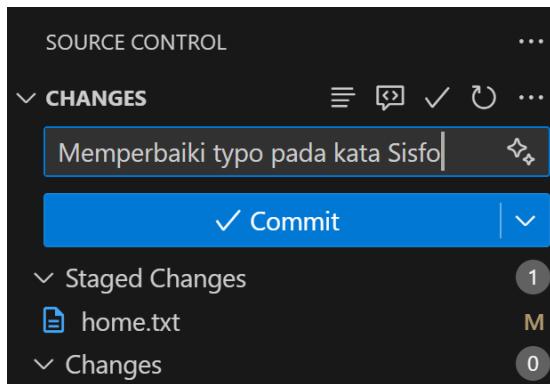
2. Setelah kita klik icon source control, ternyata tidak terjadi apa-apa. Hal itu karena git memeriksa bahwa tidak ada perubahan apapun yang terjadi atau yang perlu disimpan.



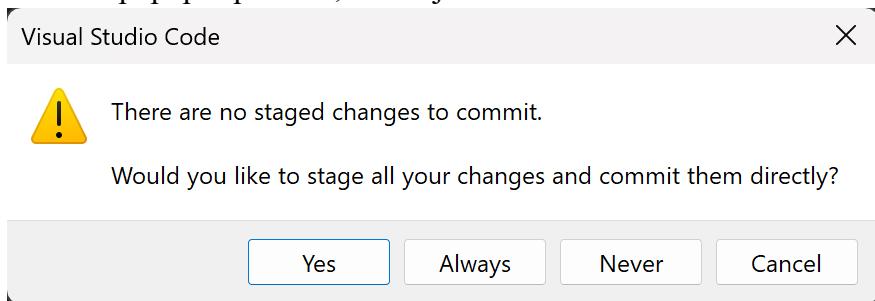
3. Kita akan coba membuat perubahan di kodingan kita, dengan mengedit teks yang ada di file .txt yang berisi kalimat "**Selamat datang di LAB AP SISFO UH! Bab 1 Git & Github 2025**". Perubahan yang dilakukan adalah memperbaiki penulisan kata "**SIDFO**" menjadi "**SISFO**", lalu melihat perubahannya setelah disimpan.



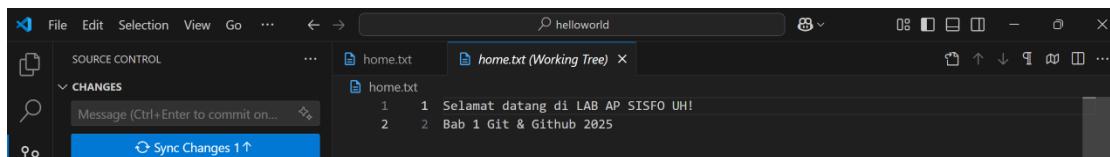
4. Namun perubahan itu hanya disimpan di komputer kita, kodingan yang ada di github belum kita ubah seperti yang terbaru saat ini.
5. Maka dari itu, kita isi pesan perubahannya dan klik tombol **Commit** untuk menyimpan perubahan di github.



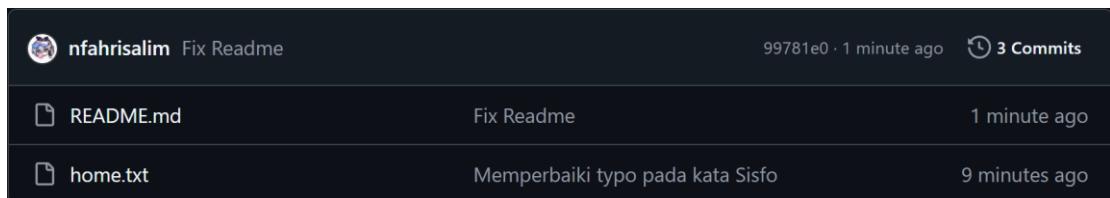
6. Jika ada popup seperti ini, klik saja “Yes”



7. Setelah itu tombol **Commit** berubah menjadi **Sync Changes** artinya ada perubahan yang belum di push (↑) dan jika (↓) ada perubahan yang belum di pull atau diambil. Dan klik tombol tersebut



8. Setelah itu, kita cek kembali apakah perubahan sudah di push di github atau belum dengan cara membuka link repository github kita. Jika sudah maka akan muncul pesan commit yang kita ketik tadi di VS Code.



9. Selesai, kamu bisa membuat perubahan lainnya dan jangan lupa push ke Github.

## L. Membuat Pull Request

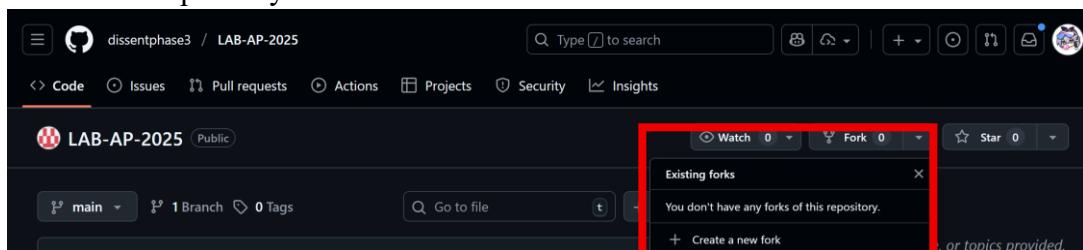
Selain melakukan merge langsung di Git, GitHub menyediakan fitur yang disebut Pull Request (PR). Fitur ini membuat proses penggabungan kode tidak hanya sebatas teknis, tetapi juga menjadi sarana kolaborasi antara anggota tim.

Pull Request dapat diibaratkan sebagai sebuah permintaan izin untuk menggabungkan kode yang telah kita kerjakan di sebuah cabang (branch) ke dalam cabang utama. Namun, fungsinya lebih dari sekadar menggabungkan. Melalui PR, anggota tim dapat meninjau perubahan yang diajukan, melihat perbedaan kode (diff) secara langsung, serta memberikan komentar dan saran sebelum kode benar-benar digabungkan. Bahkan, commit tambahan masih bisa ditambahkan jika diperlukan untuk memperbaiki atau menyempurnakan kode.

Dengan adanya Pull Request, kualitas kode dapat lebih terjamin. Setiap perubahan yang diusulkan akan melewati proses diskusi, ulasan, hingga pemeriksaan otomatis sebelum bergabung ke cabang utama. Hal ini membuat bug atau kesalahan dapat dicegah lebih awal, sehingga repositori tetap terjaga stabilitasnya.

Sebagai contoh, ketika Wawan ingin menambahkan fitur baru pada repository milik Abdul, ia tidak bisa langsung menggabungkannya karena bukan collaborator. Wawan harus membuat sebuah Pull Request terlebih dahulu. Melalui PR tersebut, Abdul dapat meninjau hasil kerja Wawan, memberi masukan, dan memutuskan apakah perubahan tersebut akan digabungkan atau ditolak. Dengan cara ini, GitHub menjadikan proses kolaborasi lebih aman, transparan, dan profesional.

### 1. Membuka Repository dan buat Fork



### 2. Mengisi detail Repo Fork

## Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (\*).

Owner \*



Repository name \*

LAB-AP-2025

✓ LAB-AP-2025 is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Repository LAB AP 2025

Copy the main branch only

Contribute back to dissentphase3/LAB-AP-2025 by adding your own branch. [Learn more.](#)

ⓘ You are creating a fork in your personal account.

**Create fork**

### 3. Setelah itu, klik **Code** dan Clone Repo Fork ke Lokal

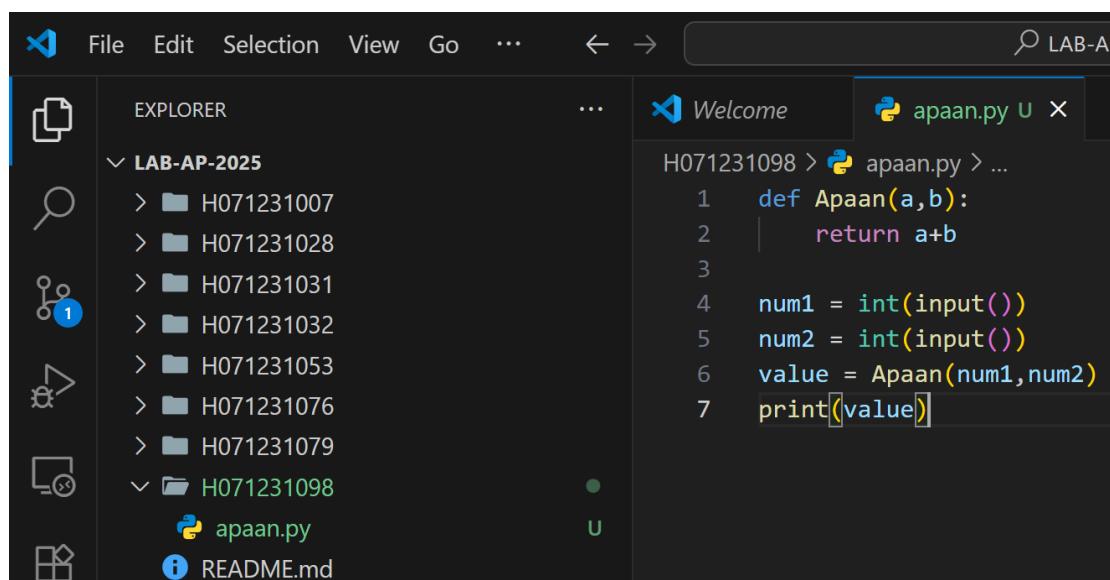
The screenshot shows two GitHub repository pages side-by-side. The top page is for the forked repository, LAB-AP-2025, which was forked from dissentphase3/LAB-AP-2025. The bottom page is for the original repository, LAB-AP-2025, which was forked from dissentphase3/LAB-AP-2025. Both pages have a red box highlighting the 'Code' button in the top navigation bar. On the bottom page, a larger red box highlights the 'Clone' section under the 'Local' tab, showing the HTTPS URL: https://github.com/nfahrisalim/LAB-AP-2025.git.

### 4. Lalu clone ke Komputer Lokal

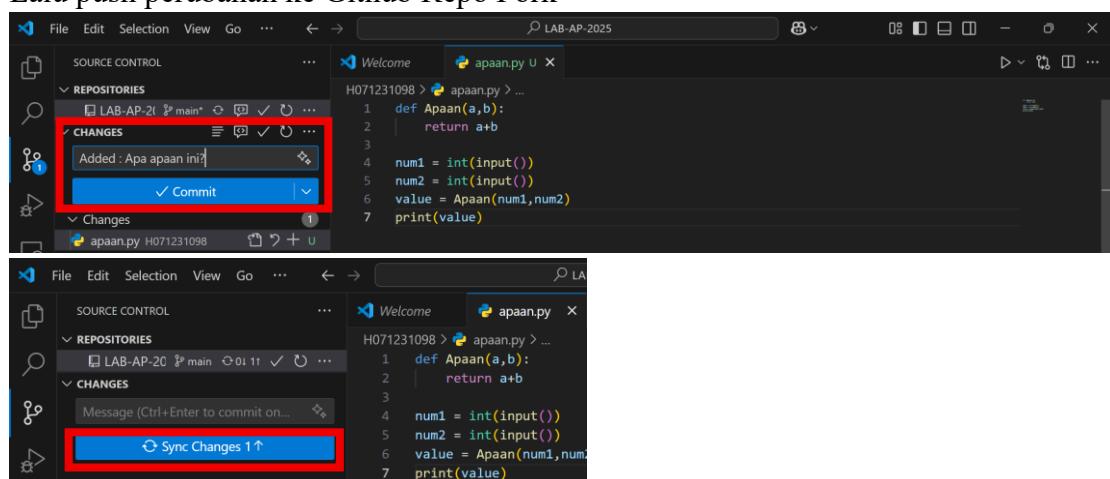
```
C:\Windows\System32\cmd x + v
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

D:\Lab AP>git clone https://github.com/nfahrisalim/LAB-AP-2025.git
Cloning into 'LAB-AP-2025'...
remote: Enumerating objects: 288, done.
remote: Counting objects: 100% (288/288), done.
remote: Compressing objects: 100% (278/278), done.
remote: Total 288 (delta 3), reused 288 (delta 3), pack-reused 0 (from 0)
Receiving objects: 100% (288/288), 82.36 KiB | 160.00 KiB/s, done.
Resolving deltas: 100% (3/3), done.
```

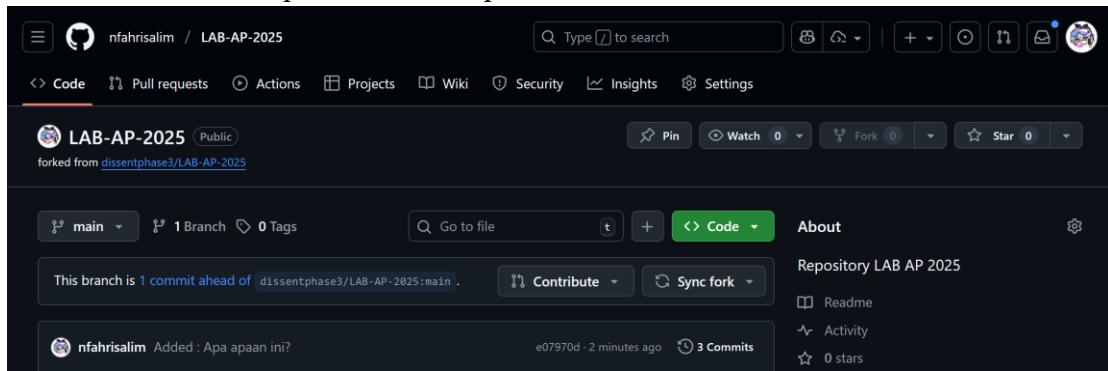
5. Kemudian, saya akan membuat perubahan dengan menambahkan sebuah folder dan mengisinya dengan sebuah file kode Python



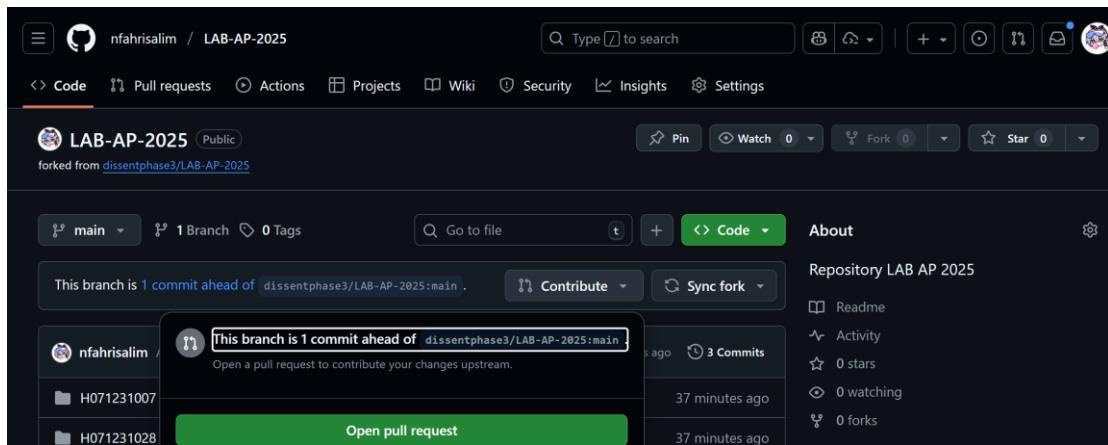
6. Lalu push perubahan ke Github Repo Fork



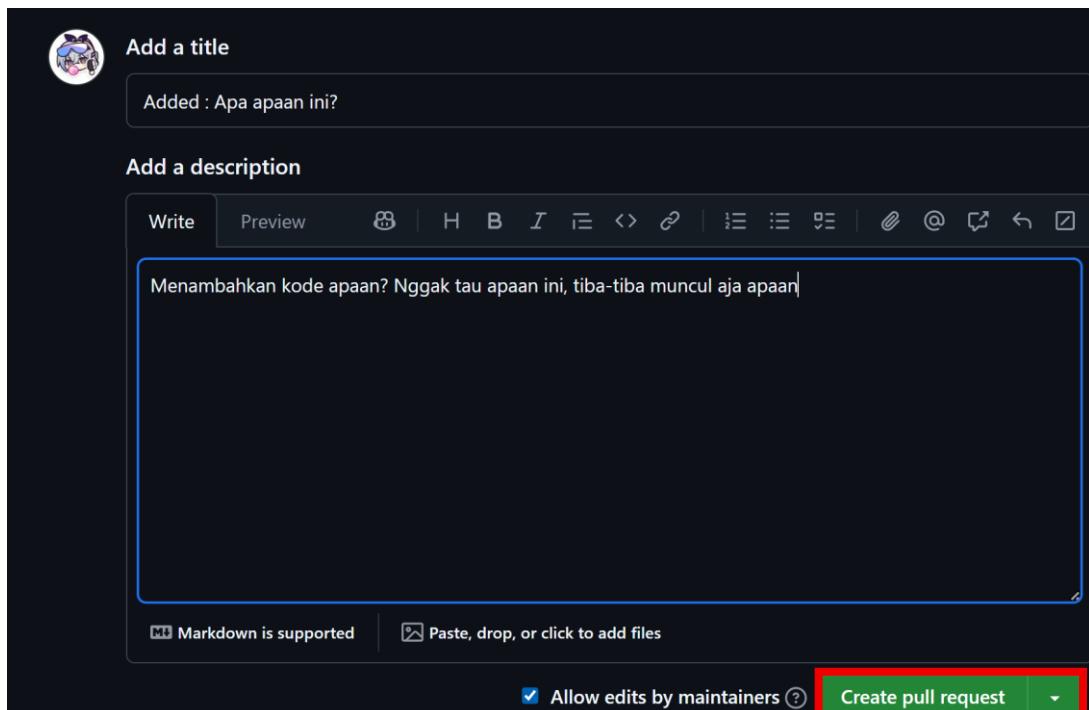
7. Kemudian kita lihat apakah sudah terpush atau tidak



8. Lalu, saya akan mengirimkan perubahan tersebut ke Repo Utama dan mengirimkan pull request. Klik tombol **Contribute** dan klik **Open Pull**



9. Cek kode, tulis deskripsi apa yang dilakukan dan klik Create pull request



10. Setelah berhasil maka akan muncul halaman seperti ini.

The screenshot shows a GitHub Pull Request page. The title of the PR is "Added : Apa apaan ini? #1". The status is "Open" and it shows nfahrisalim wants to merge 1 commit from `dissentphase3:main` into `nfahrisalim:main`. The commit count is 1, and there are 0 checks and 1 file changed. A comment from nfahrisalim says: "Menambahkan kode apaan? Nggak tau apaan ini, tiba-tiba muncul aja apaan". Below the comment, another user added a comment: "Added : Apa apaan ini?". A green box highlights the message "No conflicts with base branch" with the note "Changes can be cleanly merged.". On the right side, there are sections for Reviewers, Assignees, Labels, Projects, and Milestone, all of which are currently empty.