

VERSION 1.2

14 Februari 2025



# PEMROGRAMAN BERORIENTASI OBJEK

MODUL 0 – SETUP IDE, JDK, GIT & GITBASH

DISUSUN OLEH:

WIRA YUDHA AJI PRATAMA

KEN ARYO BIMANTORO

DIAUDIT OLEH:

Ir. Galih Wasis Wicaksono, S.Kom, M.Cs.

PRESENTED BY: TIM LAB. IT

UNIVERSITAS MUHAMMADIYAH MALANG

## **PENDAHULUAN**

### **PERSIAPAN**

1. Device masing - masing
2. Koneksi internet
3. Web browser

### **TARGET MODUL**

1. Mahasiswa dapat menginstal dan mengkonfigurasi Java Development Kit (JDK) serta Integrated Development Environment (IDE) IntelliJ IDEA.
2. Mahasiswa dapat menginstal dan menggunakan Git serta Git Bash untuk version control.
3. Mahasiswa memahami dasar-dasar penggunaan Git.
4. Mahasiswa dapat mengatur environment variables yang diperlukan untuk menjalankan Java dan Git dengan benar di sistem operasi yang digunakan.

### **KEYWORDS**

IDE, IntelliJ IDEA, JDK, GIT, GitBash

### **TABLE OF CONTENTS**

<b>PENDAHULUAN.....</b>	<b>1</b>
PERSIAPAN.....	1
TARGET MODUL.....	1
KEYWORDS.....	1
TABLE OF CONTENTS.....	1
<b>IDE (INTEGRATED DEVELOPMENT ENVIRONMENT).....</b>	<b>3</b>
TEORI.....	3
Apa itu IDE?.....	3
MATERI.....	4
DOWNLOAD IntelliJ IDEA Community Edition.....	4
INSTALASI IntelliJ IDEA Community Edition.....	5
TIPS.....	6
<b>JDK (JAVA DEVELOPMENT KIT).....</b>	<b>7</b>
TEORI.....	7
MATERI.....	7
INSTALASI.....	7
TIPS.....	9

<b>GIT &amp; GITBASH.....</b>	<b>10</b>
TEORI.....	10
Apa itu Git & GitBash?.....	10
Git.....	10
GitBash.....	11
MATERI.....	11
INSTALASI.....	11
PRAKTEK.....	15
Update repository.....	21
TIPS.....	23
<b>SUMMARY AKHIR MODUL.....</b>	<b>24</b>

## IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

### TEORI

#### Apa itu IDE?



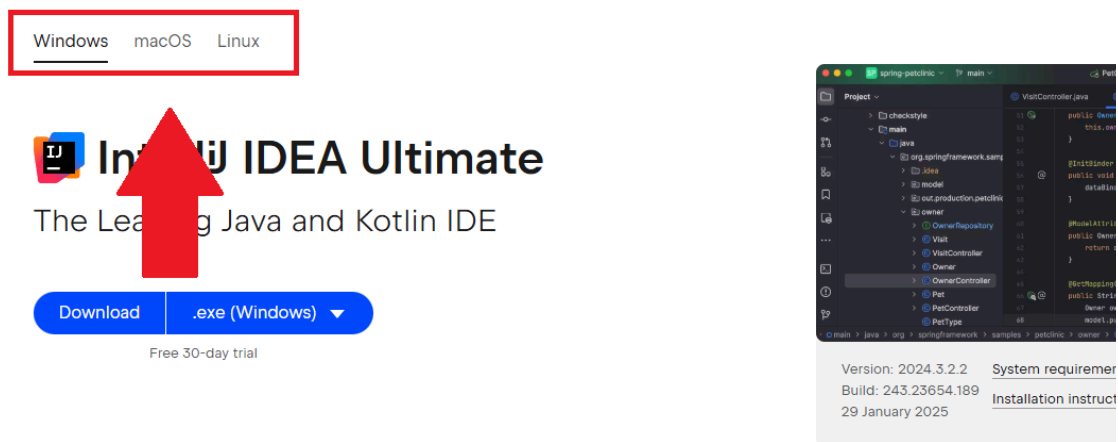
**IDE (Integrated Development Environment)** adalah **perangkat lunak** yang digunakan untuk **membantu pengembangan program** dengan menyediakan berbagai fitur seperti **editor kode**, **debugger**, dan **compiler** dalam satu aplikasi. Dengan IDE, proses **menulis**, **menguji**, dan **menjalankan** kode menjadi lebih mudah dan efisien.

Dalam modul ini, kita akan menggunakan **IntelliJ IDEA Community Edition**, sebuah IDE yang populer untuk pengembangan Java. IntelliJ IDEA menawarkan fitur cerdas seperti **auto-completion**, **debugging tools**, dan **integrasi dengan Git**, sehingga sangat membantu dalam proses belajar dan pengembangan proyek berbasis **Java**.

**MATERI****DOWNLOAD IntelliJ IDEA Community Edition**

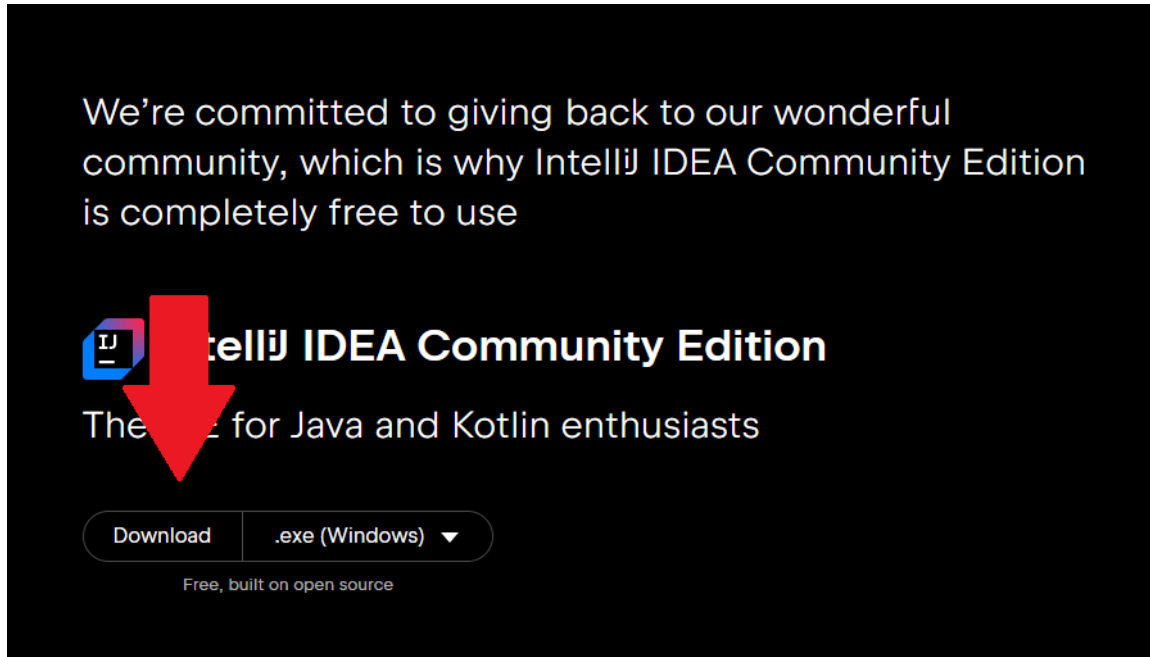
# IntelliJ IDEA

- 
- Kunjungi laman <https://www.jetbrains.com/idea/download/?section=windows> atau cari “**Download intellij idea community edition**” di mesin pencarian favorit anda.
- Pilih varian sesuai Sistem Operasi anda, jika Windows maka pilih Windows.



<https://www.jetbrains.com/idea/nextversion/>

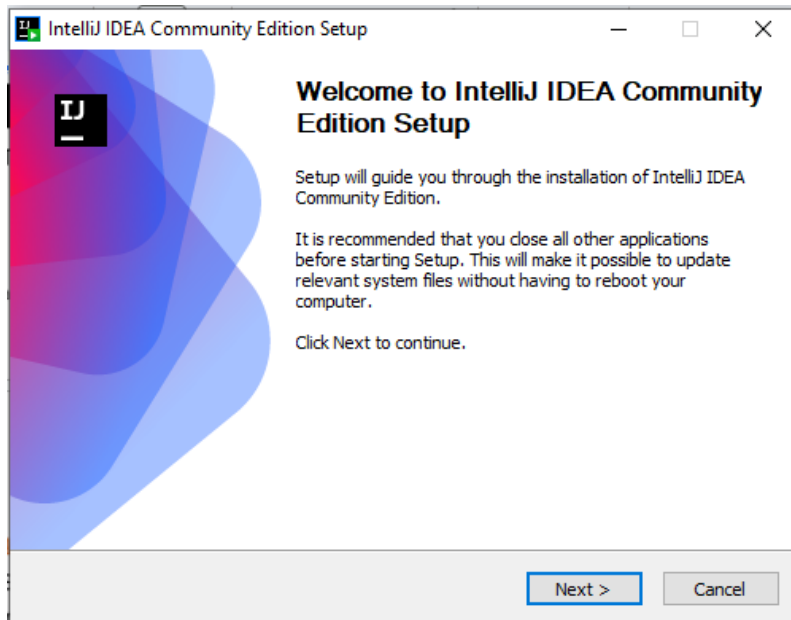
- Lalu scroll ke bawah menuju ke bagian Community Edition, dan klik download.



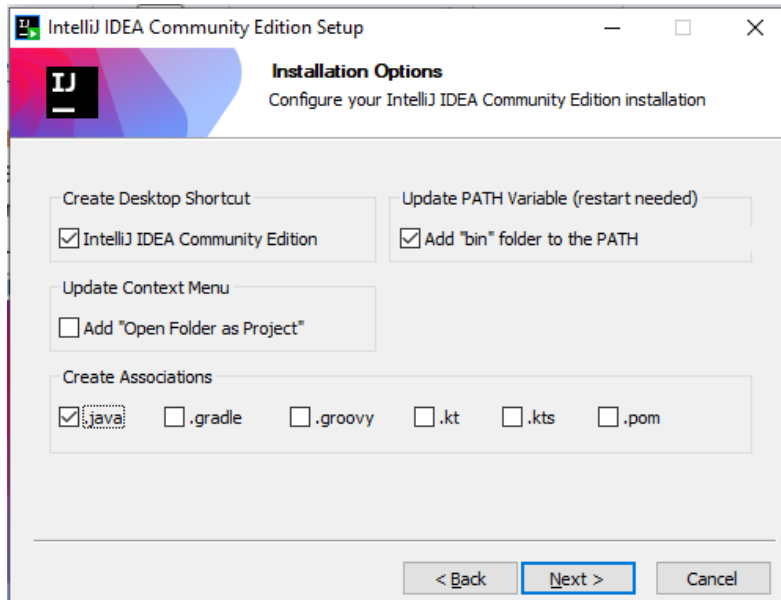
- Tunggu hingga proses download selesai.

#### **INSTALASI IntelliJ IDEA Community Edition**

- Setelah proses download selesai, lalu buka program tersebut untuk memulai proses instalasi.
- Lakukan proses instalasi seperti biasa, klik next saja.

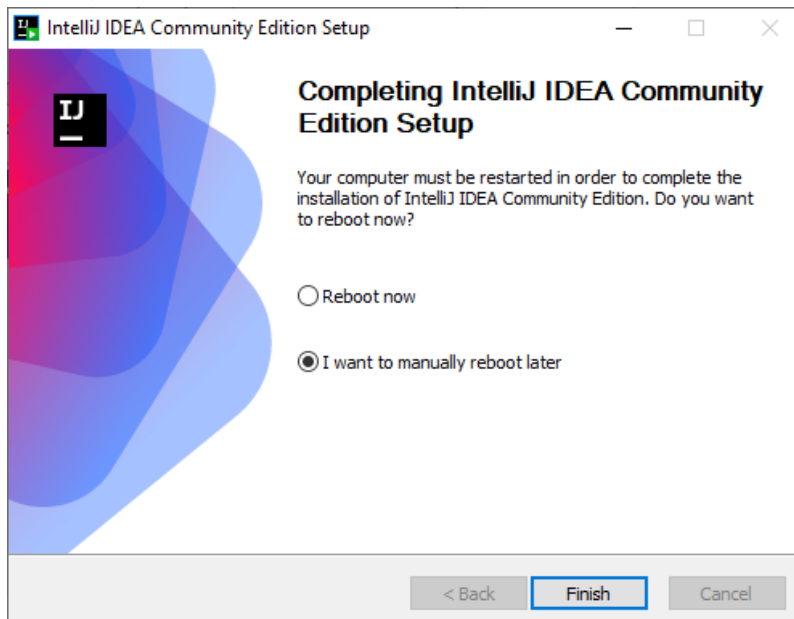


- Pada bagian installation options, pilih seperti ini (ceklis semua juga boleh.)



- Lalu next, install, dan tunggu hingga proses instalasi selesai.
- Setelah proses instalasi selesai, anda bisa pilih “Reboot now” dan klik finish untuk me-restart perangkat anda guna melengkapi proses instalasi.

**Perhatian :** Simpan pekerjaan yang sedang anda buka terlebih dahulu sebelum me-restart perangkat untuk menghindari kehilangan progres pekerjaan anda.



### **TIPS**

Tutorial download IntelliJ:

[Video tutorial](#)

## JDK (JAVA DEVELOPMENT KIT)

### TEORI



**JDK (Java Development Kit)** adalah paket perangkat lunak yang digunakan untuk mengembangkan aplikasi berbasis **Java**. JDK mencakup berbagai komponen penting seperti **JRE (Java Runtime Environment)**, **compiler (javac)**, **debugger**, serta berbagai **pustaka** dan alat pendukung lainnya. Dengan menggunakan JDK, pengembang dapat **menulis**, **mengkompilasi**, dan **menjalankan** program Java dengan optimal.

JDK tersedia dalam berbagai versi yang dirilis oleh vendor-vendor ternama seperti **Oracle**, **OpenJDK**, **Amazon Corretto**, dan **lainnya**. Setiap versi memiliki fitur dan peningkatan yang disesuaikan dengan perkembangan teknologi Java.

### MATERI

#### INSTALASI

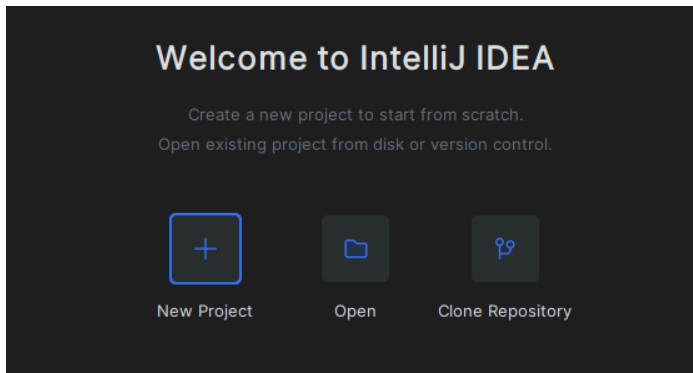
Implementasi **JDK** banyak dibuat oleh vendor-vendor ternama, Anda tidak perlu bingung untuk memilihnya, sebagian besar pasti sudah cukup untuk kebutuhan Anda praktikum di mata kuliah ini. Disini kita akan coba menggunakan **JDK versi 23** yang dibuat oleh **Oracle** saja.

Kita bisa memanfaatkan **IDE IntelliJ IDEA** untuk membantu dan mempermudah proses instalasi JDK yang kita butuhkan.

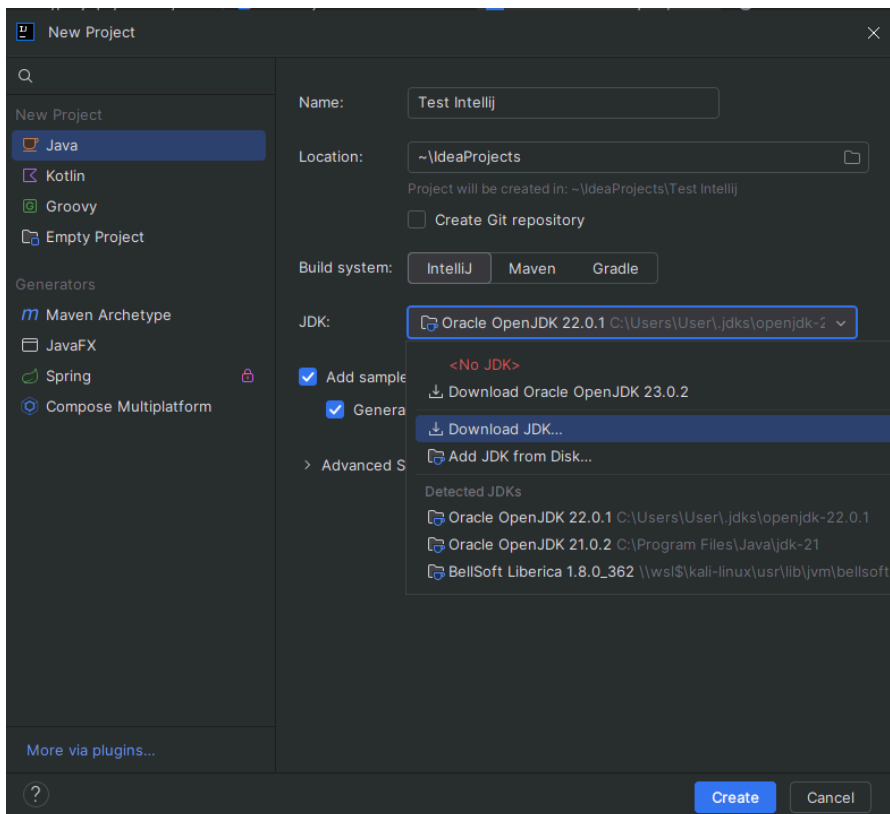
- Buka software **IntelliJ IDEA** yang sudah kita install sebelumnya.



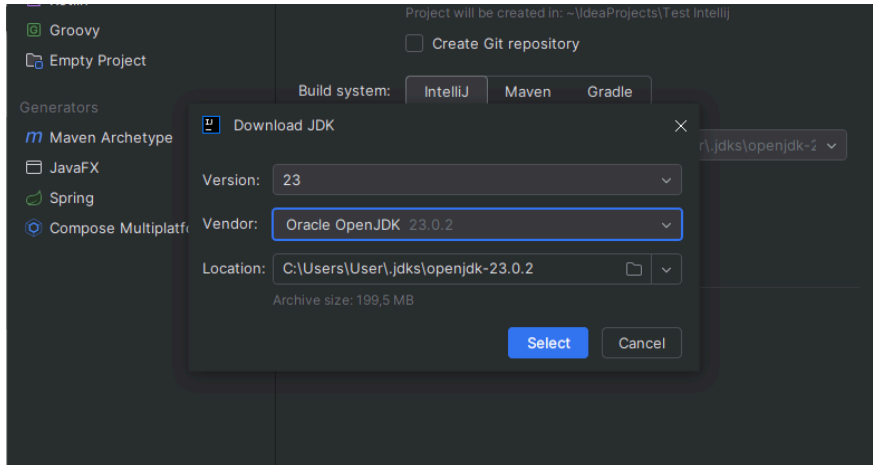
- Lalu coba buat proyek/pekerjaan baru dengan klik tombol **“New Project.”**



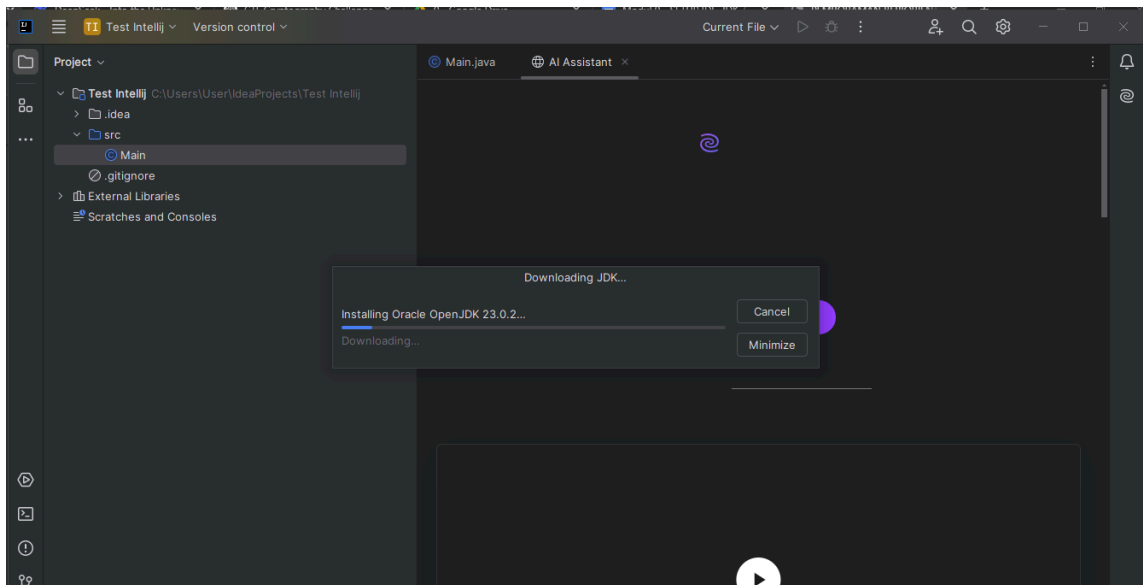
- Lalu lengkapi pilihan dan inputan yang ada, mulai dari **nama proyek**, tempat **penyimpanan** proyek tersebut, sampai **pemilihan JDK**.
- Disini karena kita perlu menginstall **JDK**, maka kita bisa klik pilihan **“Download JDK”** untuk memilih dan mendownload **JDK** yang kita inginkan.



- Maka akan muncul pop up dialog untuk memilih versi, vendor, dan lokasi penyimpanan JDK nya. Anda bisa sesuaikan saja, disini saya akan coba contohkan menggunakan **versi 23** dari **Oracle**.



- Setelah itu lalu klik **“Select”**, dan **“Create”** untuk membuat proyek.



Disini bisa dilihat, pada proses pembuatan proyek baru tersebut maka **JDK** yang kita pilih tadi akan otomatis **di-download** dan **di-install** oleh **IntelliJ IDEA**. Tunggu hingga proses download dan instalasi selesai.

### **TIPS**

Tutorial Instalasi JDK di IntelliJ:

[Video tutorial](#)

## GIT & GITBASH

### TEORI

#### Apa itu Git & GitBash?



**Git** dan **GitBash** adalah dua tools penting dalam **pengembangan software modern**. Mari pahami peran keduanya:

#### Git

**Git** adalah **sistem kontrol versi terdistribusi (DVCS)** yang dirancang untuk **melacak perubahan pada kode sumber** selama **pengembangan software**. Dibuat oleh **Linus Torvalds (pencipta Linux)**, Git membantu developer:

- **Menyimpan riwayat** perubahan kode (history).
- **Berkolaborasi** dalam tim tanpa menimpa perubahan satu sama lain.
- **Mengembalikan versi kode** ke keadaan sebelumnya jika terjadi kesalahan.
- **Membuat branch** (cabang) untuk mengembangkan fitur atau eksperimen tanpa mengganggu kode utama.

Konsep Utama Git:

- **Repository (Repo)**: Direktori penyimpanan proyek beserta riwayat perubahannya.
- **Commit**: "Snapshot" perubahan kode yang disimpan dengan pesan deskriptif.
- **Remote**: Repositori yang tersimpan di server (e.g., **GitHub**, **GitLab**).
- **Push/Pull**: Mengunggah/mengunduh perubahan ke/dari repositori remote.

## GitBash

**GitBash** adalah **terminal emulator** untuk **Windows** yang menyediakan **antarmuka** baris perintah (**Command Line Interface/CLI**) untuk menjalankan **Git** dan perintah **Unix/Linux**.

- **Git** awalnya dirancang untuk **lingkungan Unix**, sehingga **GitBash** memungkinkan pengguna **Windows** menggunakan perintah **Git** dengan nuansa **terminal Linux** (seperti **ls**, **cd**, **mkdir**).
- Dilengkapi dengan **shell Bash (Bourne Again Shell)**, alat dasar untuk interaksi dengan sistem operasi via teks.

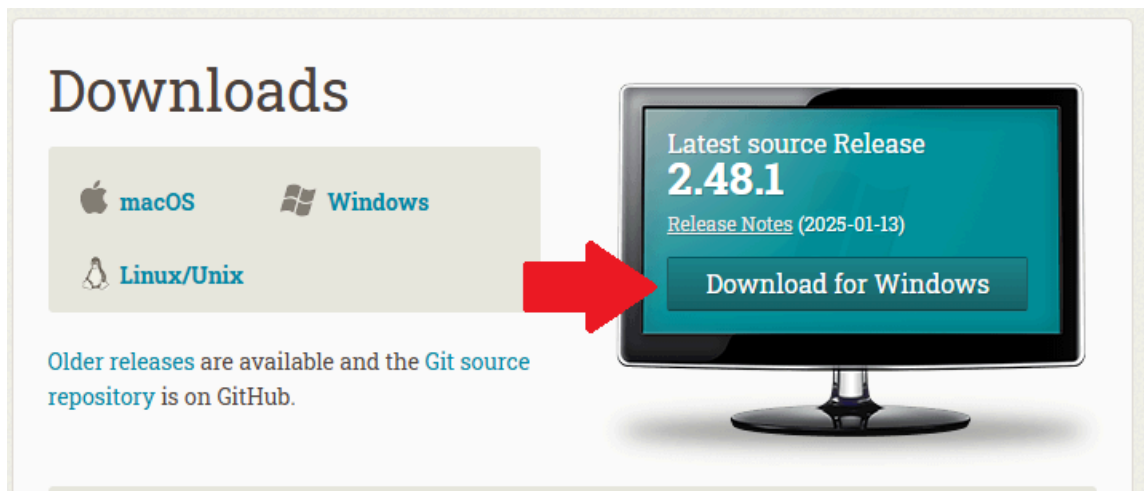
Mengapa **GitBash** Penting?

- Developer sering lebih efisien menggunakan **CLI** untuk **operasi Git** dibanding **GUI**.
- Memastikan konsistensi perintah lintas **OS (Windows, macOS, Linux)**.
- Memungkinkan automasi tugas melalui scripting (e.g., bash script).

## MATERI

### INSTALASI

- Kunjungi laman <https://git-scm.com/downloads> atau cari “**download git bash**” di mesin pencarian favorit anda.
- Pilih varian sesuai dengan **sistem operasi** anda, jika anda mengunjungi laman tersebut menggunakan **Windows**, bisa langsung klik tombol “**Download for Windows**” seperti di gambar ini.



- Kemudian pilih dibagian **standalone installer**, Pilih versi **Git** yang sesuai dengan **arsitektur sistem** operasi Anda (**32-bit atau 64-bit**). Jika tidak yakin, Anda dapat

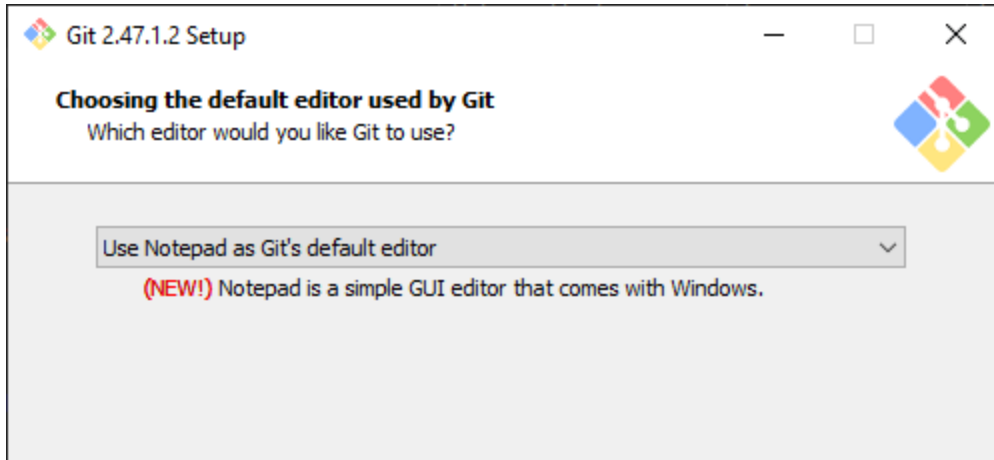
mengecek tipe sistem di pengaturan komputer Anda, salah satunya dengan cara menuju ke **Settings > System > About**.



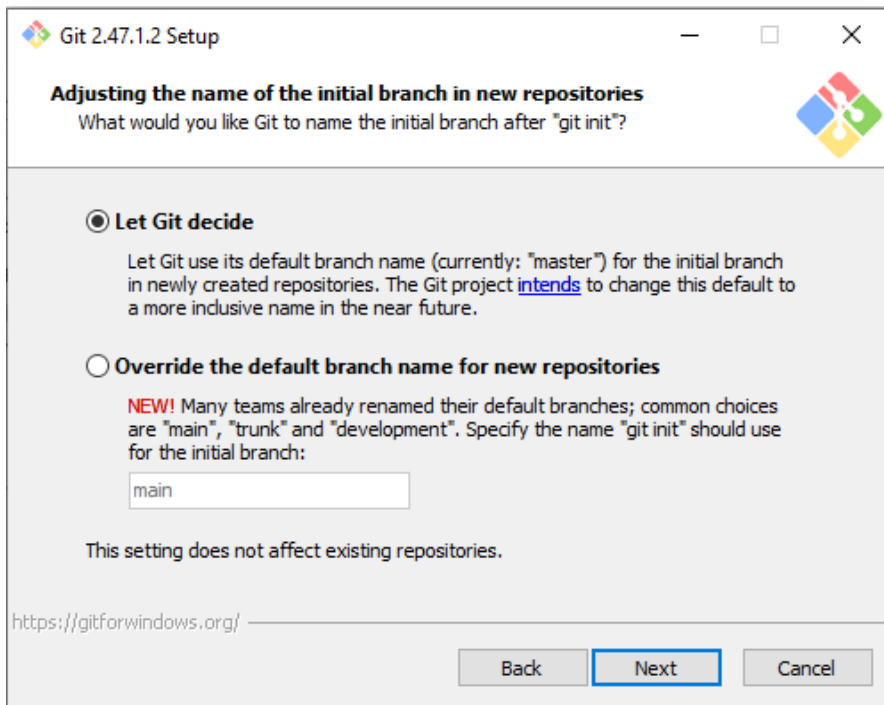
- Setelah proses download selesai, lalu jalankan programnya, dan install seperti biasa, klik next **next** saja.



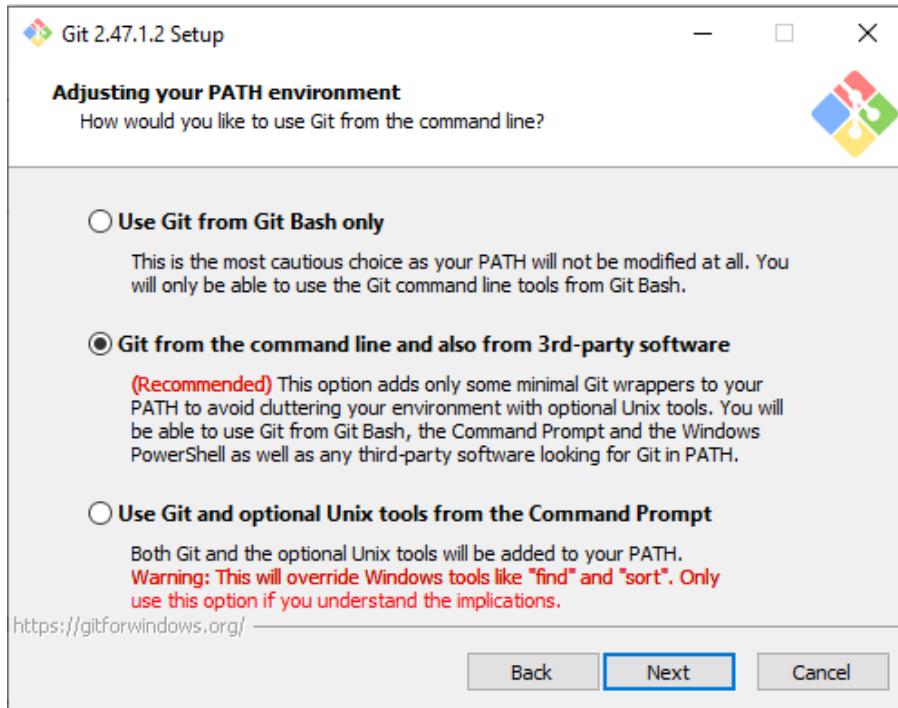
- Pada bagian pemilihan **default editor**, pilih **“Use Notepad as Git’s default editor”**, jika anda paham menggunakan editor yang lainnya untuk git, maka bisa disesuaikan, namun jika tidak yakin, saya sarankan untuk menggunakan **Notepad** saja. Lalu klik **Next**.



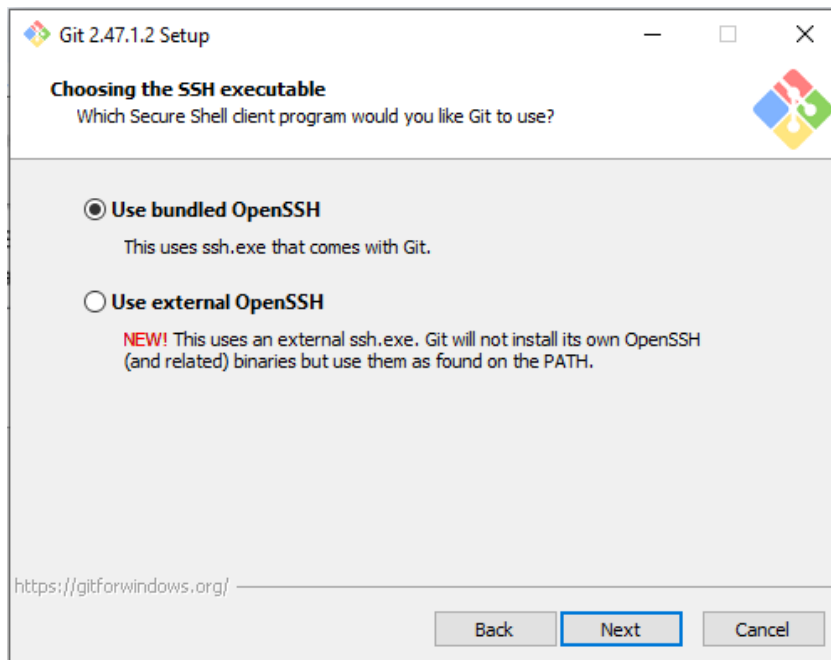
- Selanjutnya pilih **“Let Git decide”**, dan klik **Next**.



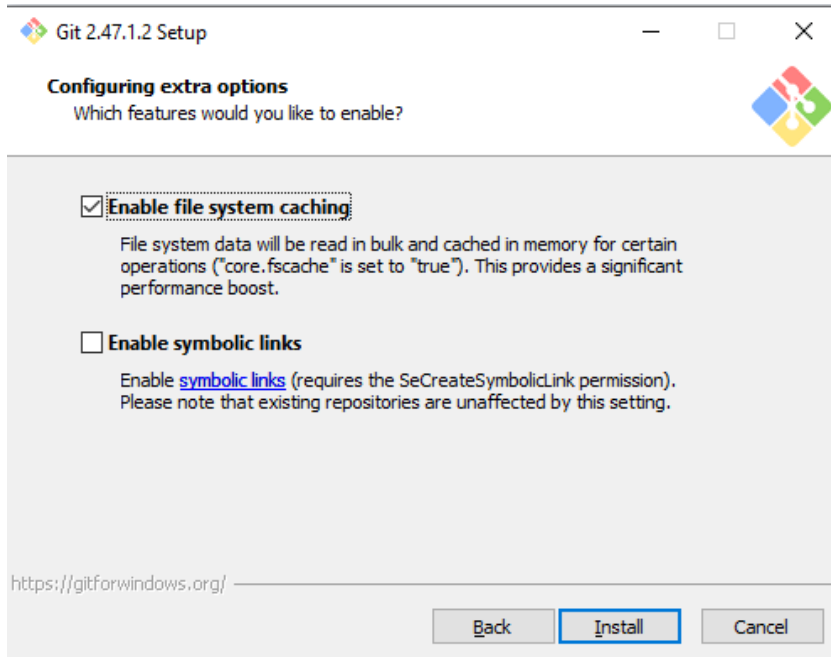
- Selanjutnya pilih pilihan yang recommended saja, yaitu **“Git from the command line and also from 3rd-party software”**.



- Pilih “Use bundled OpenSSH”, lalu Next.

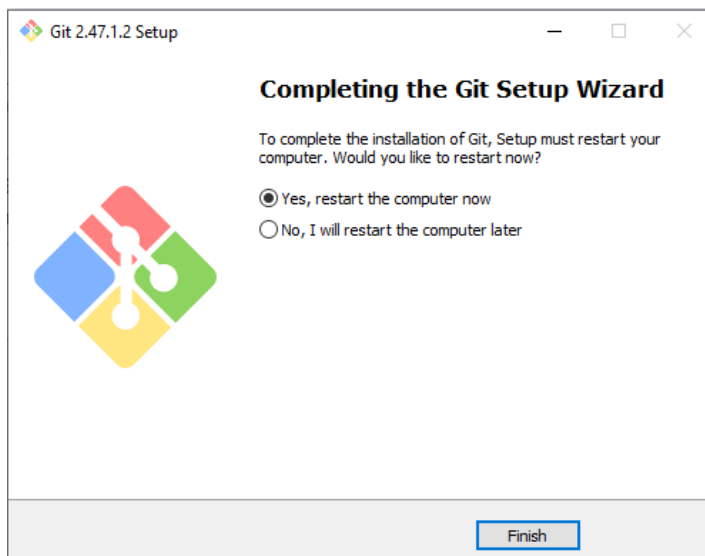


- Selanjutnya klik next next saja untuk menggunakan settingan defaultnya hingga pada bagian akhir seperti ini, lalu lanjut klik **install**.



- Setelah proses instalasi selesai, anda diminta untuk **restart** perangkat untuk melengkapi proses instalasinya, klik **finish**.

Perhatian : Simpan pekerjaan yang sedang anda buka terlebih dahulu sebelum me-restart perangkat untuk menghindari kehilangan progres pekerjaan anda

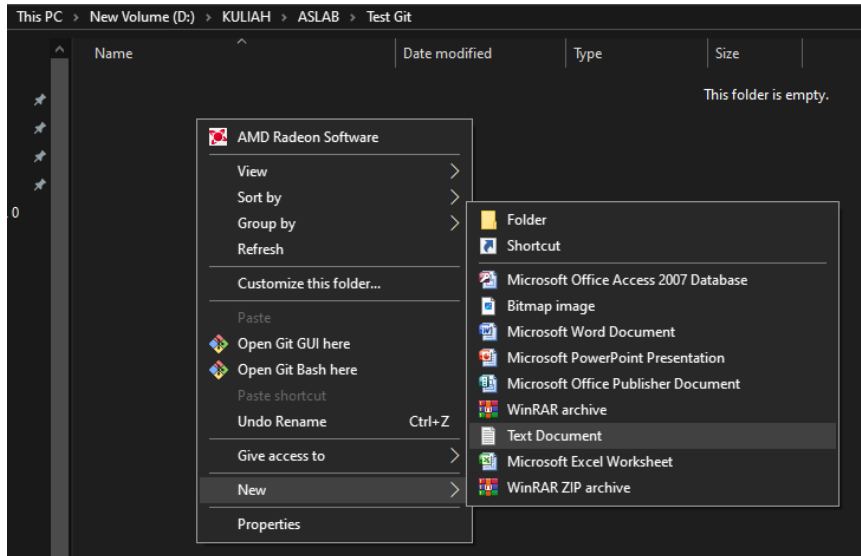


### **PRAKTEK**

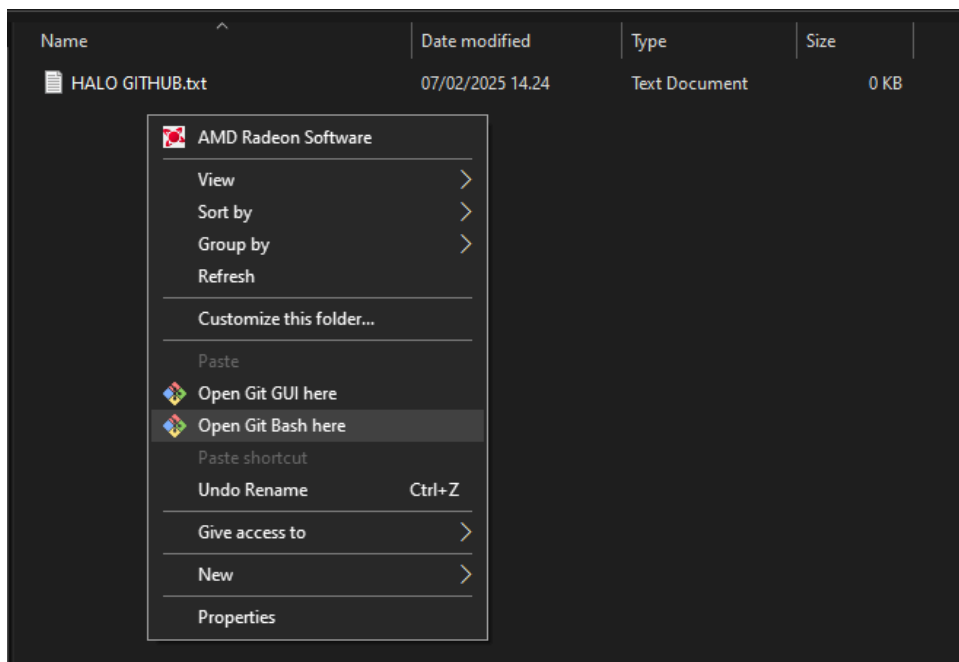
Mari kita coba untuk membuat **repository** baru di **github** dan **push** file simple saja. Pastikan anda sudah mempunyai akun **github** ya, jika belum, bisa buat melalui link berikut <https://github.com> atau cari “**github**” di mesin pencarian favorit anda.



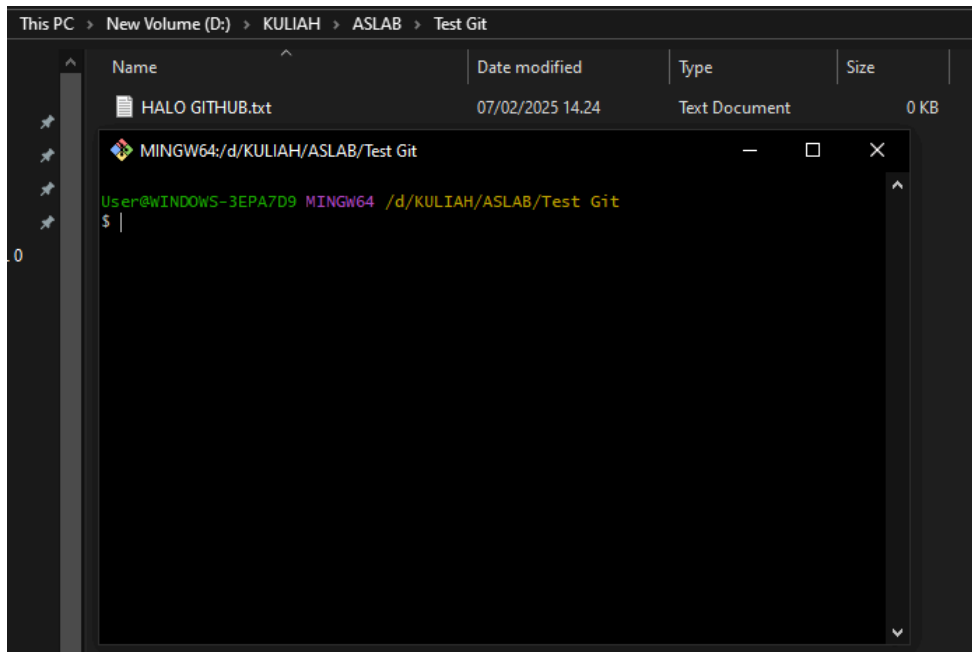
- Sekarang coba buat folder baru lalu buat juga file .txt di dalamnya. Bisa dengan cara seperti ini **Klik kanan > New > Text Document**.



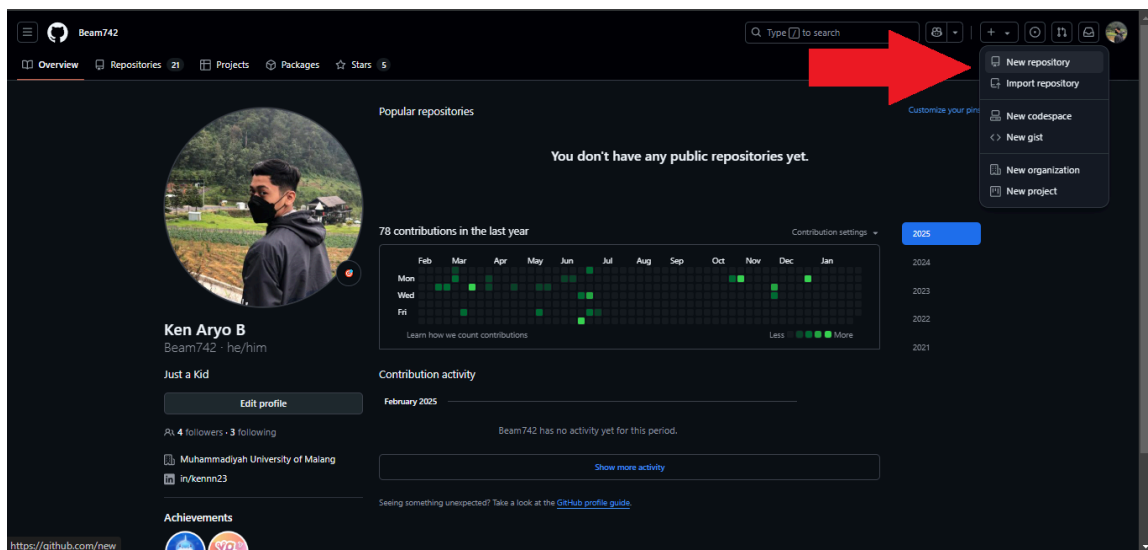
- Setelah itu **klik kanan** di dalam **folder tersebut** dan pilih “**Open Git Bash here**”. Jika menggunakan **Windows 11** dan pilihan tersebut tidak terlihat, maka anda bisa klik pilihan “**Show more options**”.



- Dengan begitu maka akan terbuka **terminal Git bash** yang langsung berlokasi di dalam **folder tersebut**.





- Sebelum lanjut untuk push isi folder tersebut, kita bisa menuju ke **Github** terlebih dahulu untuk membuat **repository baru**. Caranya dengan klik tombol “+” di pojok kanan atas lalu klik “**Create new repository**”.



- Lalu sesuaikan inputannya, terutama untuk nama **repository**. Dan klik “**Create repository**”


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


Owner \*  / Repository name \*

 Your new repository will be created as Test-Github.  
The repository name can only contain ASCII letters, digits, and the characters `.`, `-`, and `_`.

Great repository names are short and memorable. Need inspiration? How about [reimagined-funicular](#)?


Description (optional)

☒  Public  
Anyone on the internet can see this repository. You choose who can commit.


☐  Private  
You choose who can see and commit to this repository.

Initialize this repository with:


☐ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore  
.gitignore template: **None** 

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license  
License: **None** 

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

 You are creating a public repository in your personal account.

[Create repository](#)

- Lalu kita bisa melihat contoh langkah langkah atau perintah yang bisa kita lakukan untuk **push** proyek kita ke **repository**.

Quick setup — if you've done this kind of thing before

 Set up in Desktop or **HTTPS** **SSH** <https://github.com/Beam742/Test-Github.git> 

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# Test-Github" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Beam742/Test-Github.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/Beam742/Test-Github.git
git branch -M main
git push -u origin main
```



Namun, di sini kita hanya akan menjalankan serangkaian perintah berikut ini saja:

### 1. git init

Perintah ini digunakan untuk **menginisialisasi repository Git** di dalam **folder proyek kita**. Setelah menjalankan perintah ini, **Git** akan mulai **melacak** perubahan dalam **folder tersebut**.

**2. git add .** *(beserta titiknya)*

Perintah ini menambahkan semua file yang ada di dalam folder proyek ke dalam **staging area**, yaitu area persiapan sebelum commit. **Tanda titik (.)** berarti **semua file** yang baru atau telah diubah akan ditambahkan.

**3. git commit -m "first commit"**

Perintah ini **menyimpan perubahan** yang sudah ditambahkan ke **staging area** ke dalam **repository lokal**. Pesan **"first commit"** tersebut bisa disesuaikan, yang intinya berfungsi sebagai catatan yang menjelaskan perubahan yang dilakukan.

**4. git branch -M main**

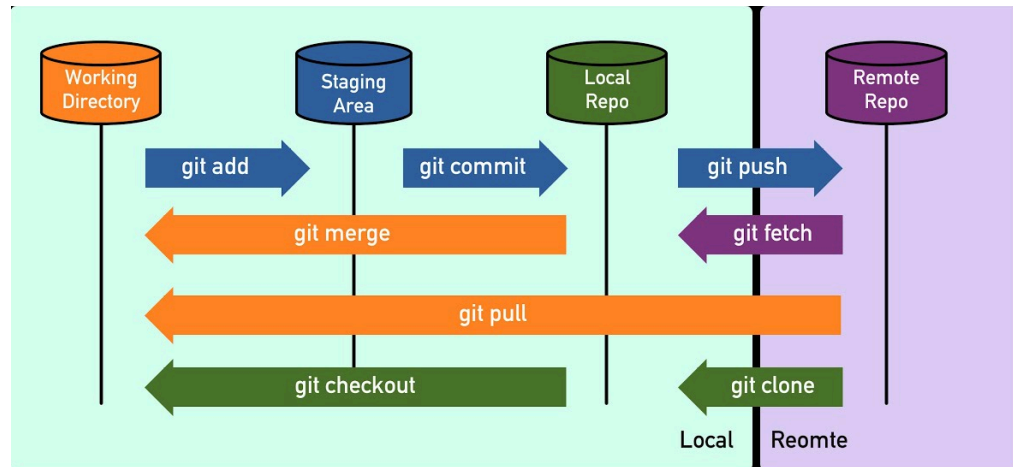
Perintah ini mengubah **nama branch utama** menjadi **main**. Secara **default**, Git biasanya menggunakan nama **master**, tetapi sekarang banyak proyek menggunakan **main** sebagai standar.

**5. git remote add origin** <https://github.com/...> *(ganti sesuai yang tertera di repository anda)*

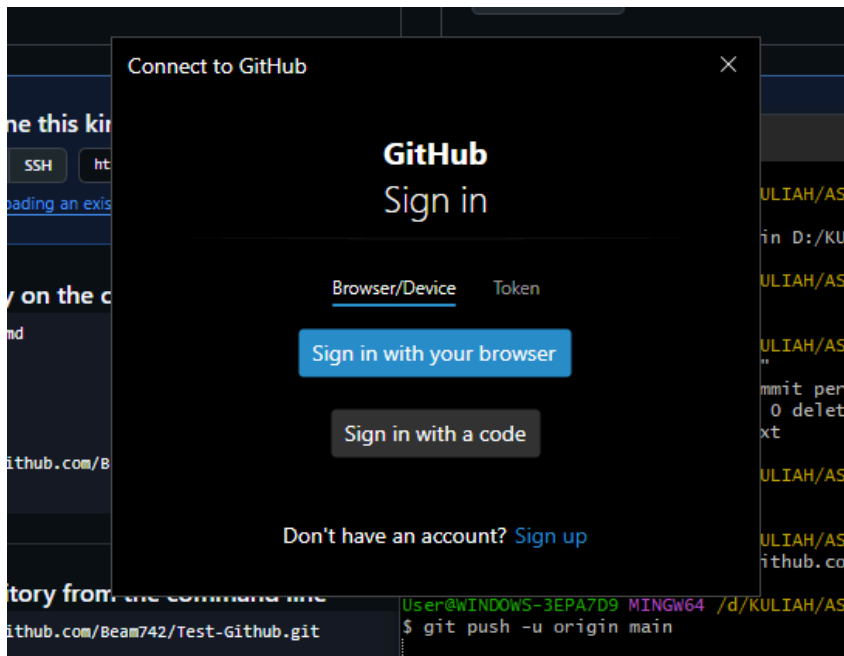
Perintah ini menghubungkan **repository lokal** kita dengan **repository remote** yang ada di **GitHub**. URL yang diberikan adalah alamat repository yang telah kita buat di **GitHub**.

**6. git push -u origin main**

Perintah ini **mengunggah (push)** commit dari **branch main** di repository **lokal** ke **repository remote (GitHub)**. Opsi **-u** (set upstream) digunakan agar branch main di lokal terhubung dengan branch main di remote, sehingga perintah git push berikutnya bisa lebih sederhana tanpa perlu menentukan tujuan **repository** terus menerus.



- Pada saat menjalankan perintah **push** untuk pertama kali, Git akan meminta anda untuk **login** ke **github** terlebih dahulu, disini bisa coba klik “**Sign in with your browser**”.



Lalu lanjutkan proses **login** seperti biasa menggunakan akun **github** anda

- Jika proses **login** berhasil, maka anda bisa menutup halaman login github tersebut dan kembali lihat terminal **git bash** anda, maka akan terlihat bahwa proses push berhasil.

```

MINGW64/d/KULIAH/ASLAB/Test Git

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git
$ git init
Initialized empty Git repository in D:/KULIAH/ASLAB/Test Git/.git/

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (master)
$ git add .

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (master)
$ git commit -m "commit pertamaku"
[master (root-commit) 1825f07] commit pertamaku
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 HALO GITHUB.txt

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (master)
$ git branch -M main

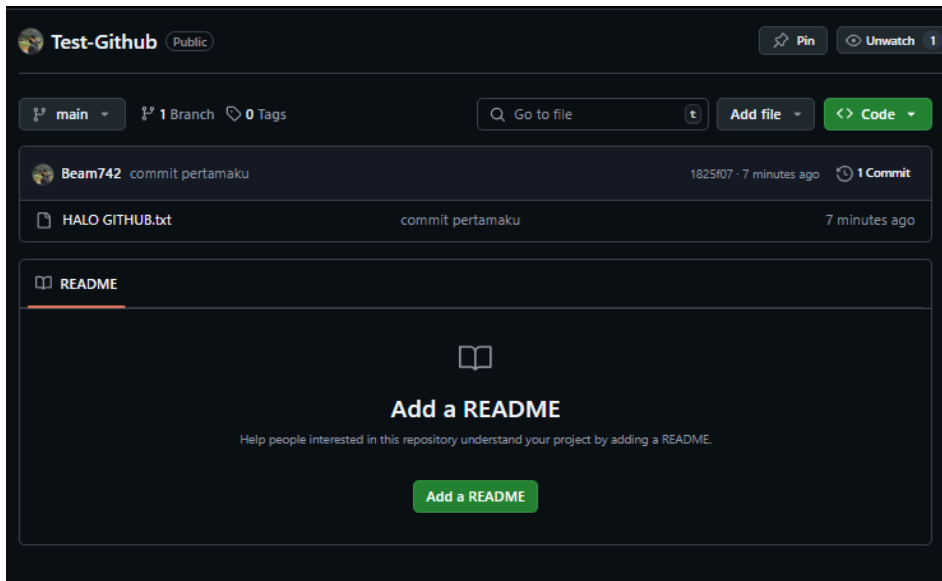
User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)
$ git remote add origin https://github.com/Beam742/Test-Github.git

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 218 bytes | 218.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Beam742/Test-Github.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)

```

- Dan coba refresh halaman repository **github** anda.



Proses push berhasil dilakukan

### Update repository

Jika file **.txt** yang ada pada repository tersebut kita buka, maka masih akan berisi kosongan atau mungkin teks yang pertama kali kita buat. Sekarang mari kita coba untuk memperbarui atau update isi pekerjaan kita.

- Buka file **.txt** yang kita buat di dalam folder tadi dan coba edit isinya lalu simpan.

- Setelah itu kita bisa langsung push perubahan yang ada dengan menggunakan perintah berikut.
  - `git add .`
  - `git commit -m "tuliskan pesan"`
  - `git push`

Perintahnya akan lebih singkat karena kita sudah mengatur alamat repository kita di awal tadi, jadi tidak perlu mengulang perintah sebanyak sebelumnya, seperti pada saat kita baru membuat repository. **Namun** ketika nanti anda membuat sebuah proyek baru lagi **DILUAR** folder yang sudah diinisialisasi dan terdaftar di repository, anda perlu menginisialisasi dan membuat sebuah repository baru lagi.

- Berikut contoh terminal ketika update

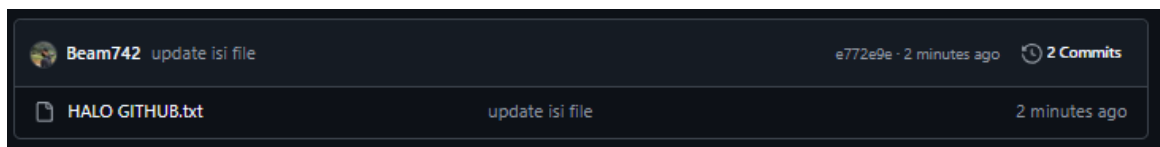
```
User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)
$ git add .

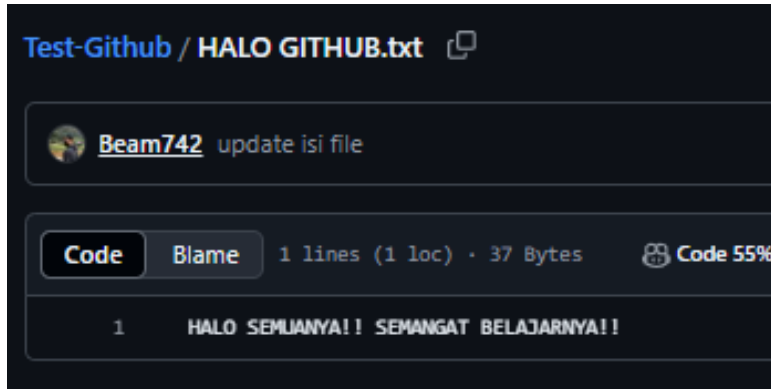
User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)
$ git commit -m "update isi file"
[main e772e9e] update isi file
1 file changed, 1 insertion(+)

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 283 bytes | 283.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Beam742/Test-Github.git
1825f07..e772e9e main -> main

User@WINDOWS-3EPA7D9 MINGW64 /d/KULIAH/ASLAB/Test Git (main)
$ |
```

- Setelah push selesai, sekarang coba anda cek repository anda dan lihat isi dari filenya





Proses update sudah berhasil dilakukan:)

### **TIPS**

Cara membuat akun GitHub:

[Video tutorial](#)

Cara menginstal Git Bash:

[Video tutorial](#)

Panduan Git Bash (bisa dibaca di sini):

[GeeksforGeeks - Working on Git Bash](#)

Tutorial penggunaan Git & Git Bash:

[Video tutorial](#)



---

## SUMMARY AKHIR MODUL

Selamat! Anda sudah menyelesaikan Modul 0 ini, yang merupakan langkah awal sebelum kita masuk ke praktik pemrograman yang sebenarnya. Di modul ini, kita sudah belajar bagaimana menyiapkan IDE, JDK, serta Git & GitBash agar nantinya proses coding bisa berjalan lancar.

Meskipun ini baru tahap persiapan, jangan anggap remeh! Setup yang baik di awal akan membuat perjalanan Anda dalam memahami praktikum Pemrograman Berorientasi Objek (PBO) ini jadi lebih mudah dan menyenangkan. Jadi, pastikan semuanya sudah siap sebelum kita lanjut ke Modul 1, di mana kita akan mulai ngoding beneran!

Tetap semangat, jangan takut buat eksplorasi lebih jauh, dan jangan ragu untuk bertanya kalau ada yang bingung. Sampai ketemu di modul berikutnya!