



## Modul Dasar GIT dan Github Untuk Matakuliah Praktikum



Disusun Oleh :

Husni Faqih, M.Kom (0606018802)

Andika Tulus Pangestu

UNIVERSITAS BINA SARANA INFORMATIKA  
PROGRAM STUDI SISTEM INFORMASI  
KAMPUS KOTA TEGAL

2021

## **DAFTAR ISI**

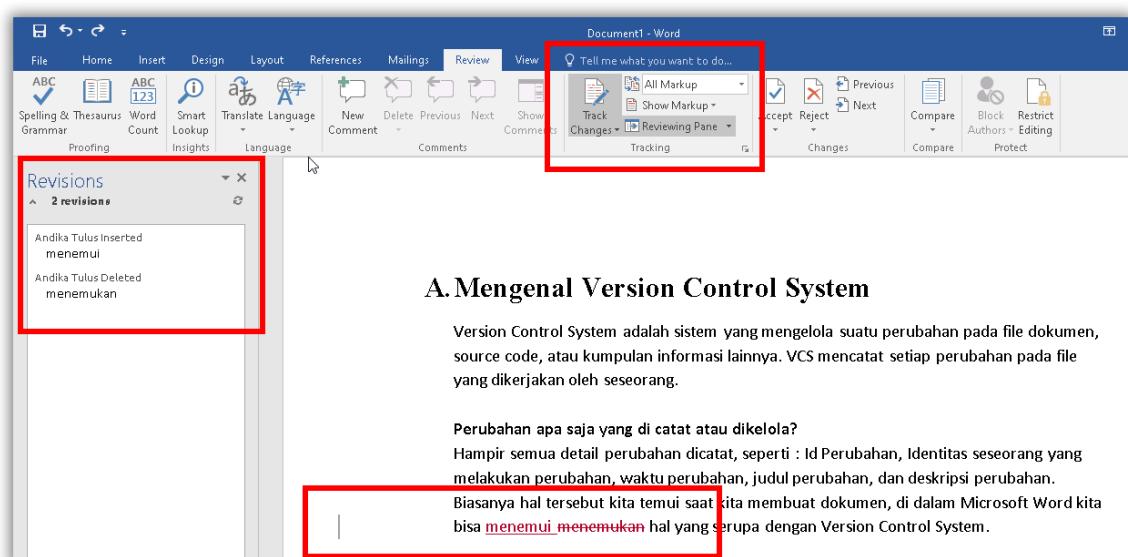
A. Mengenal Version Control System.....	2
B. Mengenal Github .....	8
C. Instalasi Git.....	9
D. Membuat Akun Github .....	29
E. Memahami Alur Kerja Git.....	32
Bekerja Dengan Git dan Github ( Membuat First Project ).....	36
Bekerja Dengan Git dan Github ( Melakukan Perubahan Project ).....	60
Bekerja Dengan Git dan Github ( Kolaborasi Project ).....	70
Bekerja Dengan Git dan Github ( Melakukan Pembaruan & Penggabungan Project ).....	79
Bekerja Dengan Git dan Github ( Visual Studio Code ) .....	84
Bekerja Dengan Git dan Github ( Menggunakan SSH Github ) .....	93
Rangkuman Git dan Github .....	98
Daftar Referensi .....	102

## A.Mengenal Version Control System

Version Control System adalah sistem yang mengelola suatu perubahan pada file dokumen, source code, atau kumpulan informasi lainnya. VCS mencatat setiap perubahan pada file yang dikerjakan oleh seseorang.

### Perubahan apa saja yang dicatat atau dikelola?

Hampir semua detail perubahan dicatat, seperti : Id Perubahan, Identitas seseorang yang melakukan perubahan, waktu perubahan, judul perubahan, dan deskripsi perubahan. Biasanya hal tersebut kita temui saat kita membuat dokumen, di dalam Microsoft Word kita bisa menemukan hal yang serupa dengan Version Control System.



Seperti yang bisa kita lihat, bahwa Version System Control itu melakukan tracking terhadap setiap perubahan yang kita lakukan, dalam gambar diatas kita bisa melihat bahwa system mencatat kata *menemukan* dihapus ( Deleted ) oleh pengetik, dan digantikan ( Insert ) oleh kata *menemui*, itulah gambaran secara jelas bagaimana Version System Control ini bekerja.

Selain sebagai system yang mengelola riwayat perubahan, system ini juga berguna untuk mempermudah pengembangan sebuah program secara kolaboratif atau berkelompok.

Jadi dengan system VCS kita bisa mengerjakan satu berkas secara bersamaan dengan beberapa orang, tanpa menunggu satu persatu selesai dan bergantian mengerjakan.

Dan yang terakhir, Version Control System juga memungkinkan kita untuk kembali ke perubahan sebelumnya. Pada intinya, version control system bisa melakukan tugas seperti melihat riwayat perubahan, mengelola setiap perubahan ( commit, push, checkout, pull ), menuntaskan bagian-bagian berkas secara langsung tanpa menunggu penggerjaan yang lain selesai, dan memungkinkan kita kembali ke perubahan sebelumnya.

Git termasuk dalam jenis Version Control System, dan Github adalah layanan Version Control System berbasis git secara online.

Jenis-jenis Version Control System :

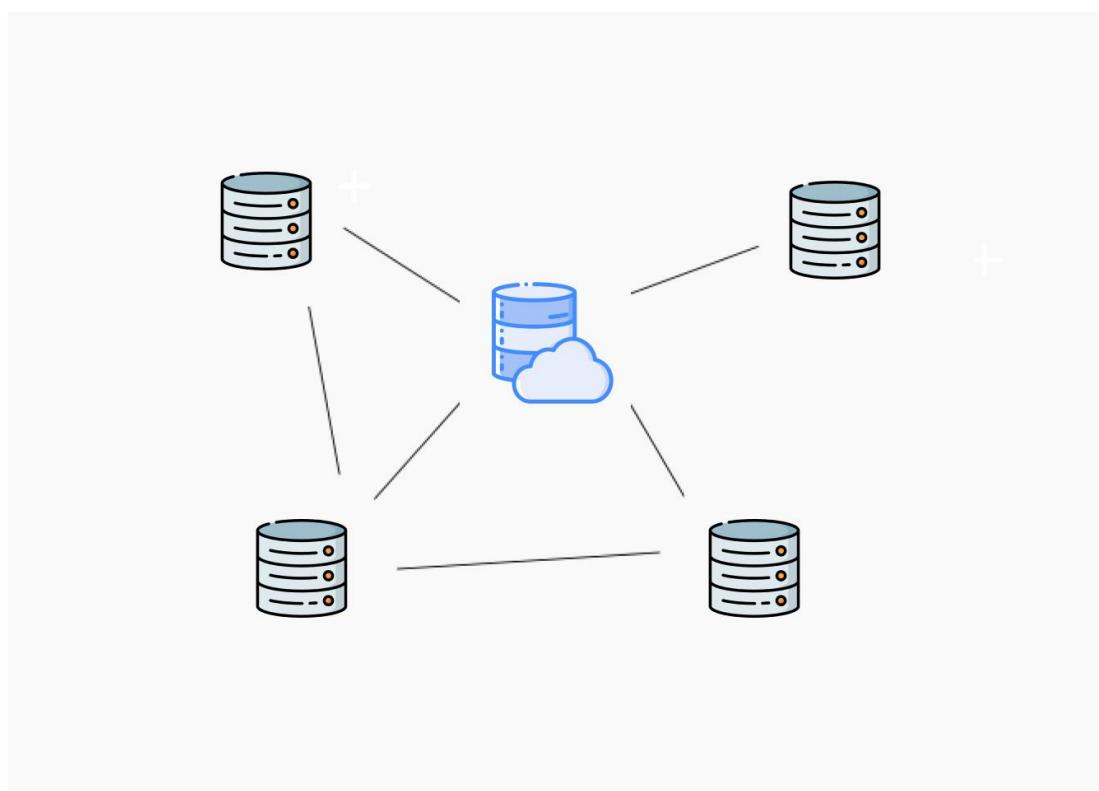
- Mercurial
- Git
- Subversion
- CVS

Itulah ulasan singkat mengenai apa itu Version Control System, dengan kita memahami konsep system tersebut, kita akan sangat terbantu pada saat belajar mengenai Git dan Github.

## A.Mengenal Git

Git adalah sebuah perangkat lunak berbasis Version Control System, dengan git kita mampu mengelola sebuah proyek atau berkas secara terdistribusi, karena perubahan yang tercatat oleh git disimpan didalam database git dan database git tersebut tidak hanya berada pada satu tempat, namun terdapat di berbagai tempat. Artinya bahwa semua orang yang berkontribusi pada proyek tersebut bisa menyimpan database perubahannya, sehingga akan memudahkan dalam mengelola proyek baik online maupun offline.

Dengan git kita bisa melakukan penggabungan beberapa berkas atau bagian pada berkas kedalam satu proyek atau berkas final, hal ini sangat mempermudah developer ketika melakukan pengembangan aplikasi secara berkolaborasi.



Git sebenarnya akan memantau semua perubahan yang terjadi pada file proyek. Lalu menyimpannya ke dalam database.

Misal ketika kita memiliki sebuah project Tugas Akhir dan dokumen tugas akhir tersebut selalu direvisi, dan belum ada sistem VCS atau Git, maka akan terdapat banyak file yang sama dengan isi yang memiliki perubahan berbeda dan menggunakan nama yang berbeda ( File Revisi Menumpuk ).

Ilustrasi Sebelum ada Git :

-  Tugas akhir (Baru Mulai)
-  Tugas akhir (Revisi Ke-1)
-  Tugas akhir (Revisi Ke-2)
-  Tugas akhir (Revisi Ke-3)
-  Tugas akhir (Revisi Ke-4)
-  Tugas akhir (Revisi Ke-5)
-  Tugas akhir (Revisi Ke-6)
-  Tugas akhir (Revisi Ke-7)
-  Tugas akhir (Revisi Ke-8)
-  Tugas akhir (Revisi Ke-9)
-  Tugas akhir (Revisi Ke-10)
-  Tugas akhir (Revisi Ke-11)
-  Tugas akhir (Udah jadi)
-  Tugas akhir Alhamdulillah Lolos
-  Tugas akhir Alhamdulillah Wis Udah

Saat kita ingin menyimpan semua perubahan pada file, biasanya kita membuat file baru dengan “**save as**”. Lalu, file akan menumpuk seperti pada gambar di atas ( Dari Tugas Akhir paling atas hingga paling bawah ).

Pada akhirnya, karena teknologi semakin berkembang maka hadirlah sebuah tools untuk mengontrol atau memanajemen berkas pada sebuah project, tools itu bernama **Git**. Setelah menggunakan Git, maka hanya akan ada satu file dalam project dan perubahannya telah disimpan dalam database.

## Ilustrasi setelah menggunakan git :



Selain memanajemen sebuah berkas, git juga mampu membantu memanajemen proyek kolaborasi atau bisa digunakan untuk bekerja secara tim.

Dengan memanfaatkan Git, kita bisa berkolaborasi dalam tim. Jadi, kita tidak perlu lagi mengerjakan pengembangan sistem maupun hal lainnya yang berbasis teks dengan cara antrian atau menunggu anggota tim pertama selesai mengerjakan kemudian mengirimkan hasil pekerjaan kepada anggota berikutnya untuk melanjutkannya. Tentunya itu akan banyak memakan waktu dan juga tidak efektif.

Misal :

Ketika kita membuat project sebuah Aplikasi hitung luas bangun datar dengan bahasa pemrograman python, dan project

tersebut dikerjakan secara berkelompok atau tim, maka dalam sebuah project tersebut akan dibagi per-bagian aplikasi untuk dikerjakan oleh anggota tim ( kolaborasi ).

Ilustrasi :

- Andika : Bagian Hitung Luas Segitiga
- Adistia : Bagian Hitung Luas Persegi
- Argenta : Bagian Hitung Luas Trapesium

Maka dari itu, pasti di komputer masing-masing anggota tim terdapat source code yang berbeda, dan jika tanpa git dengan sistem pengontrol versinya, maka untuk menggabungkan seluruh source code agar menjadi satu project final atau sebuah aplikasi yang siap digunakan, itu akan sulit. Dengan adanya Version Control dan Git, maka masalah tersebut bisa diatasi.

### **Jadi apa saja manfaat menggunakan Git?**

- Mencatat riwayat perubahan pada berkas atau proyek
- Mengelola berkas pada saat bekerja secara kolaborasi
- Mengelola perubahan berkas atau proyek
- Bisa berkontribusi pada Proyek Sumber Terbuka
- Bisa memperdalam pengembangan aplikasi berbasis CLI ( Command Line Interface )

### **Lalu apa keunggulan Git daripada Aplikasi berbasis VCS lainnya?**

- Cara instalasi aplikasi git sangat sederhana
- Menggunakan sistem VCS Terdistribusi

- Penyimpanan tidak menggunakan database seperti SQL dll.
- Mendukung penggunaan pada proyek skala besar
- Mendukung pengembangan sistem secara non-linear
- Akses penuh dengan perintah baris ( CLI )

### **Apakah ada kekurangannya? Pasti...**

- Tidak begitu optimal untuk pengembangan aplikasi secara individu
- Dukungan terhadap windows terbatas

## **B.Mengenal Github**

Jika dengan git kita hanya mampu mengelola setiap perubahan secara lokal, maka dengan github kita bisa melakukannya secara online. Github adalah layanan git berbasis cloud, selain itu dengan github kita dapat dengan mudah bekerja dengan git tanpa melakukannya dengan baris perintah ( CLI ), karena github merupakan layanan git berbasis GUI.

# Persiapan Environment

## Git dan Github

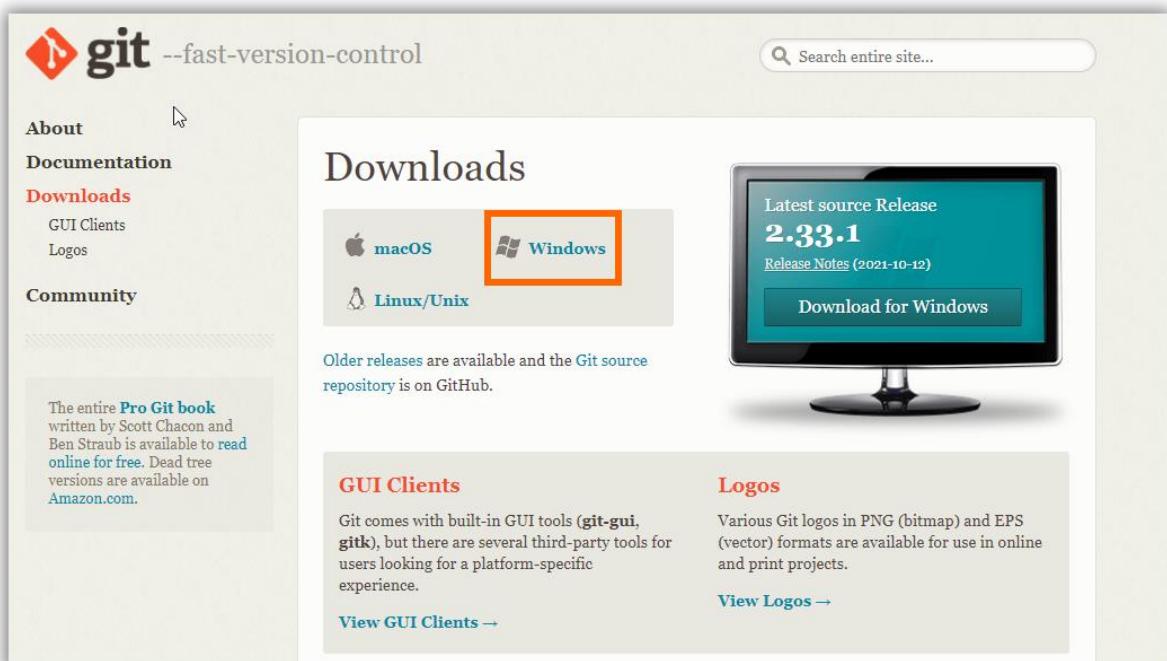
### C.Instalasi Git

Persiapan Environment local Git di Windows, untuk memasang aplikasi git, ada beberapa persyaratan sistem yang harus dipenuhi, yaitu seperti berikut ini :

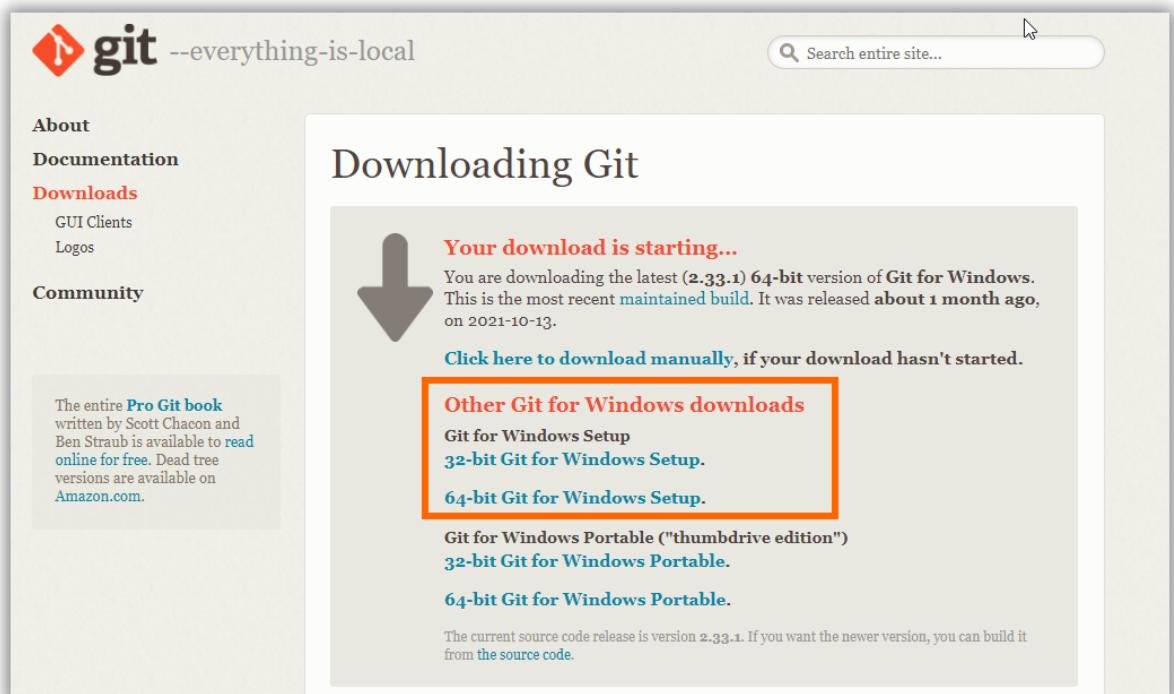
- Laptop atau Komputer dengan OS Windows 7, 8, 10
- Web Browser ( Firefox, Google Chrome, Brave, Edge )

Langkah-langkah pemasangan :

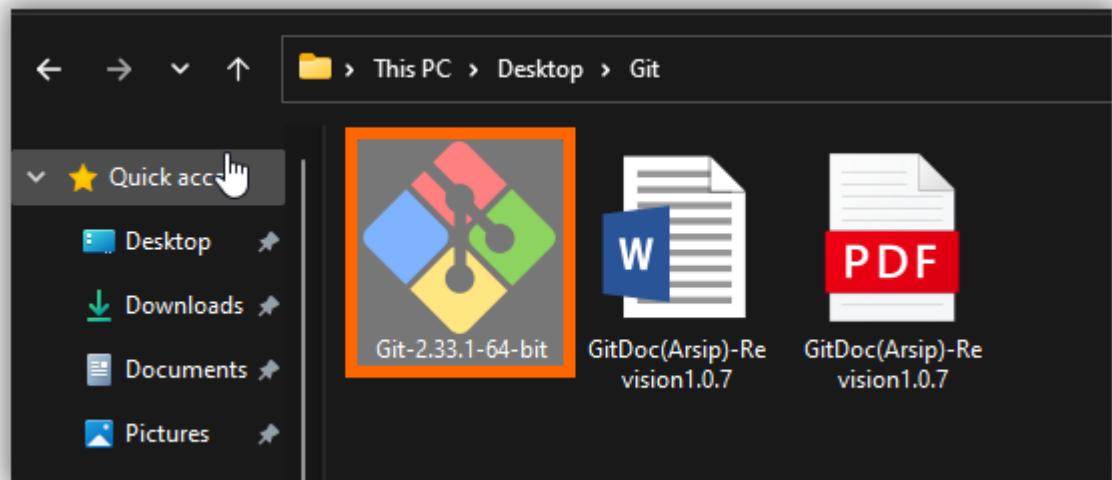
1. Unduh Git dengan membuka tautan <https://git-scm.com/downloads>
2. Pilih “Windows”



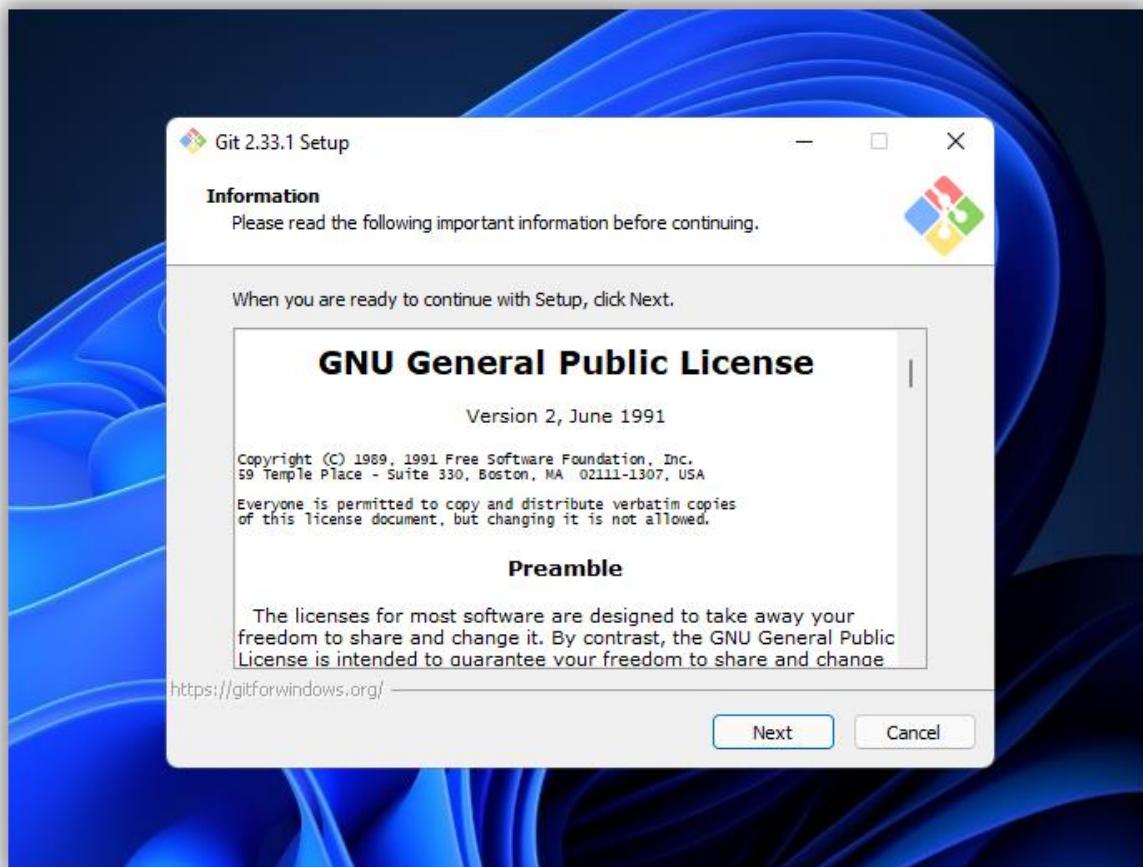
- Setelah itu tunggu hingga muncul berkas aplikasi yang terunduh otomatis atau jika tidak maka pilih unduhan sesuai dengan arsitektur komputer kamu. Kalau menggunakan 64 bit, unduh yang 64 bit. Begitu juga kalau menggunakan 32bit.



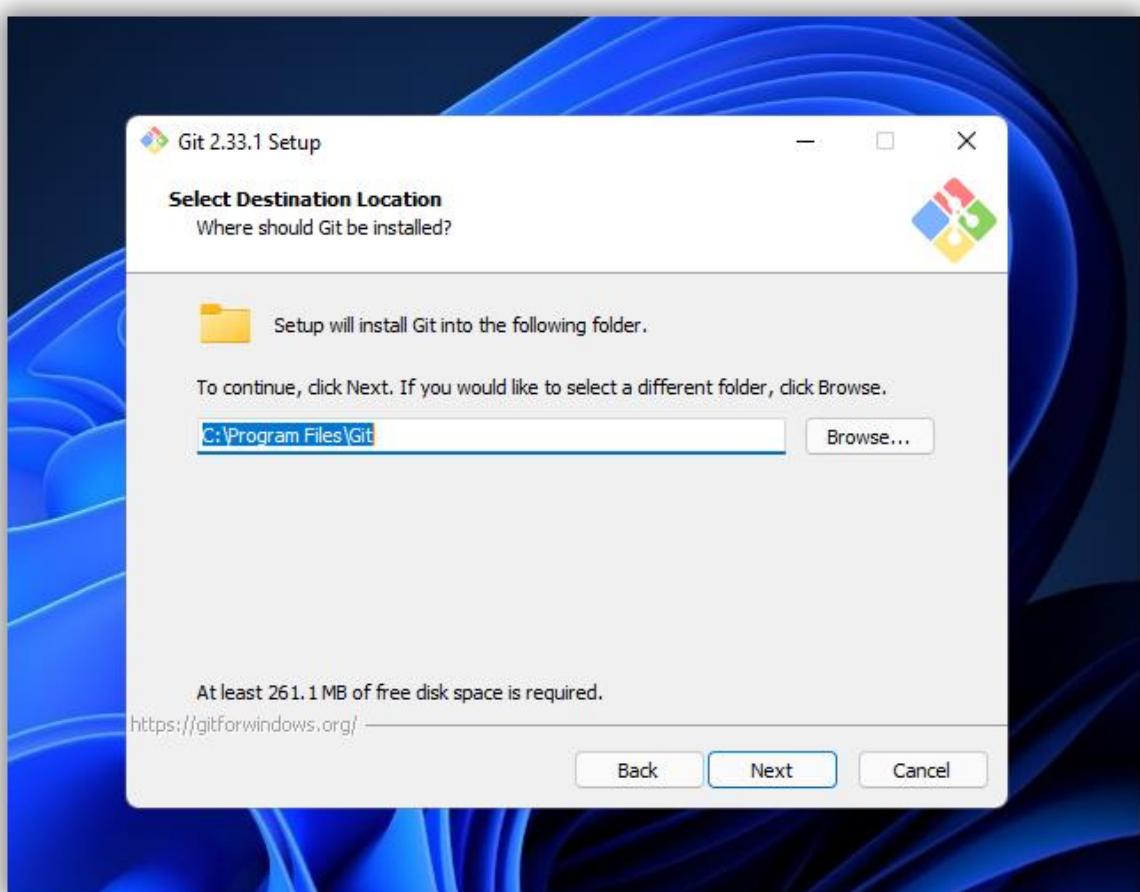
- Setelah terunduh, lalu klik 2x berkas installer Git yang telah diunduh.



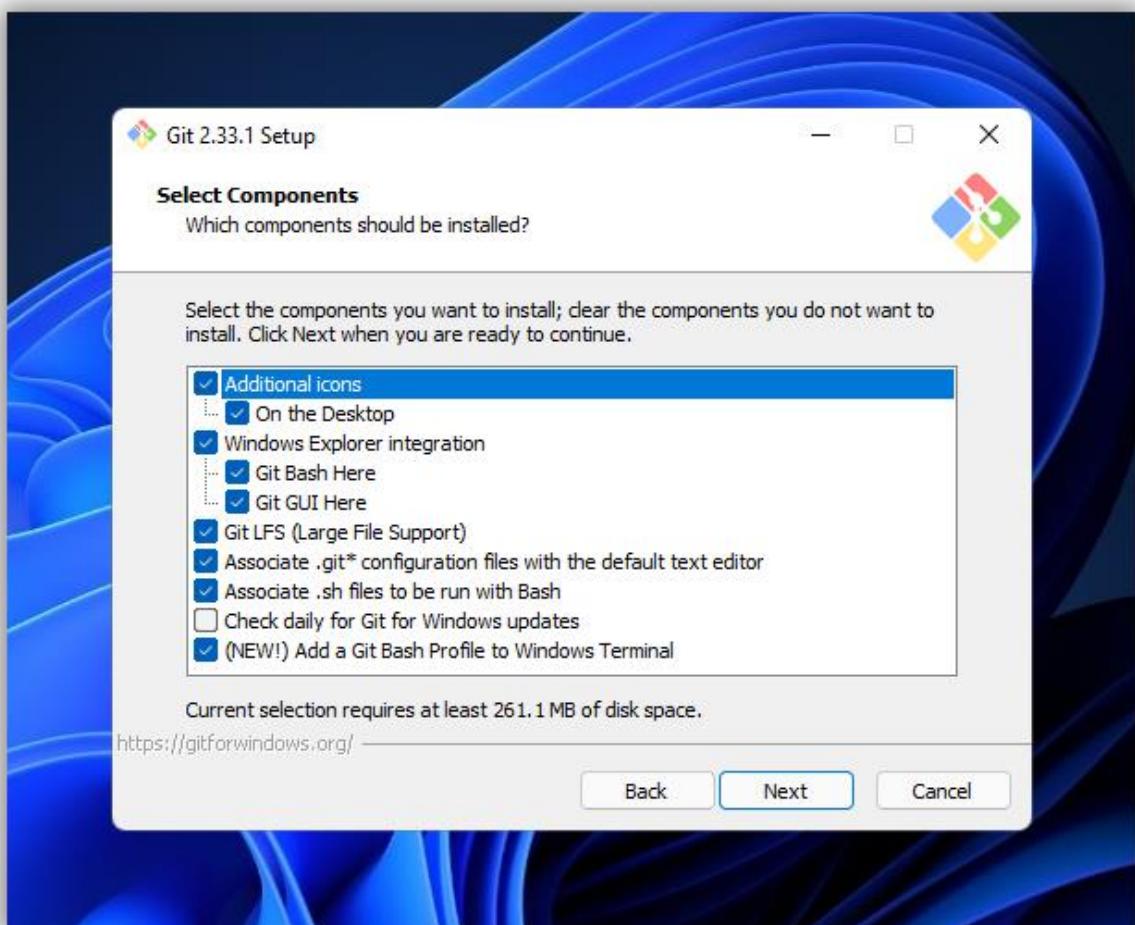
5. Tinjau Lisensi Publik Umum GNU, dan jika kamu sudah siap untuk menginstal, klik **Next**.



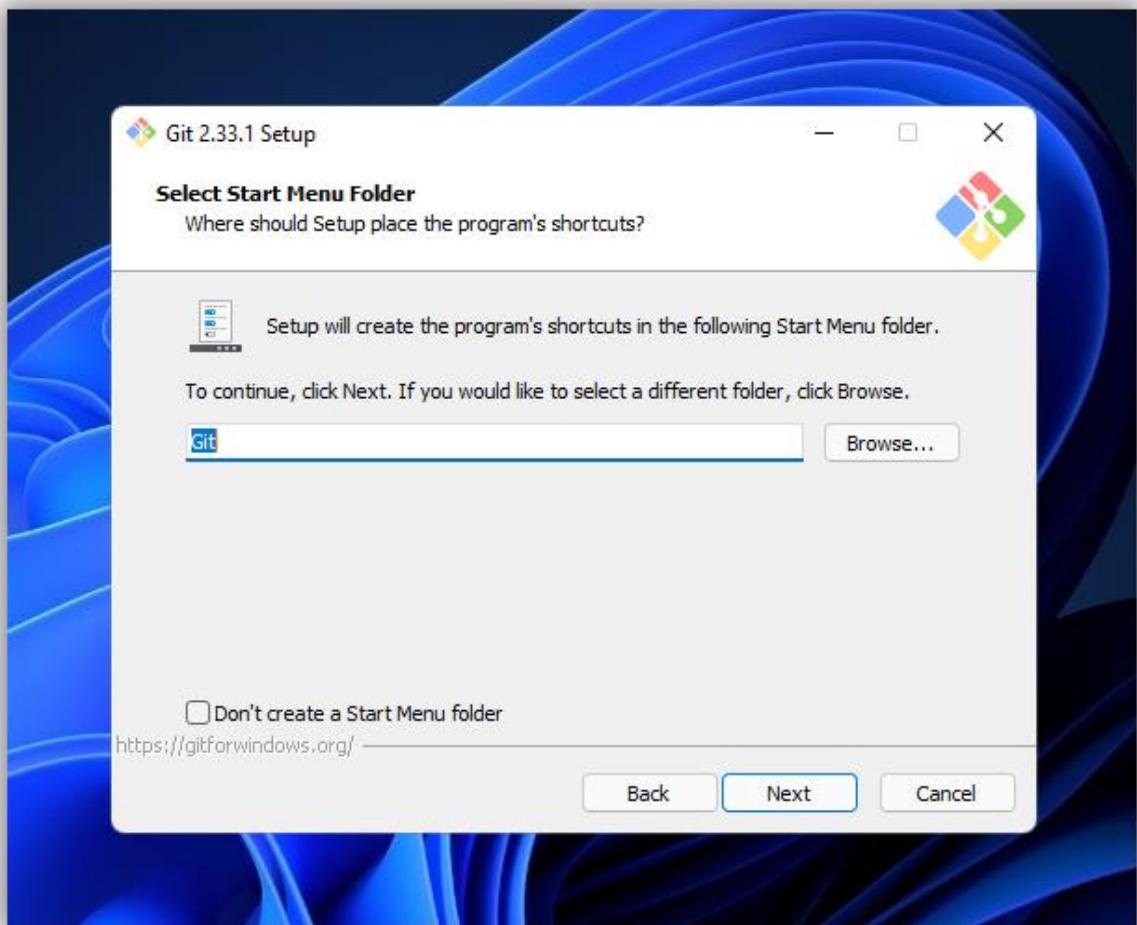
6. Selanjutnya menentukan lokasi instalasi. Biarkan saja secara default, kemudian klik **Next**.



7. Lalu pemilihan komponen, atur saja seperti di bawah ini, lalu **Next**.



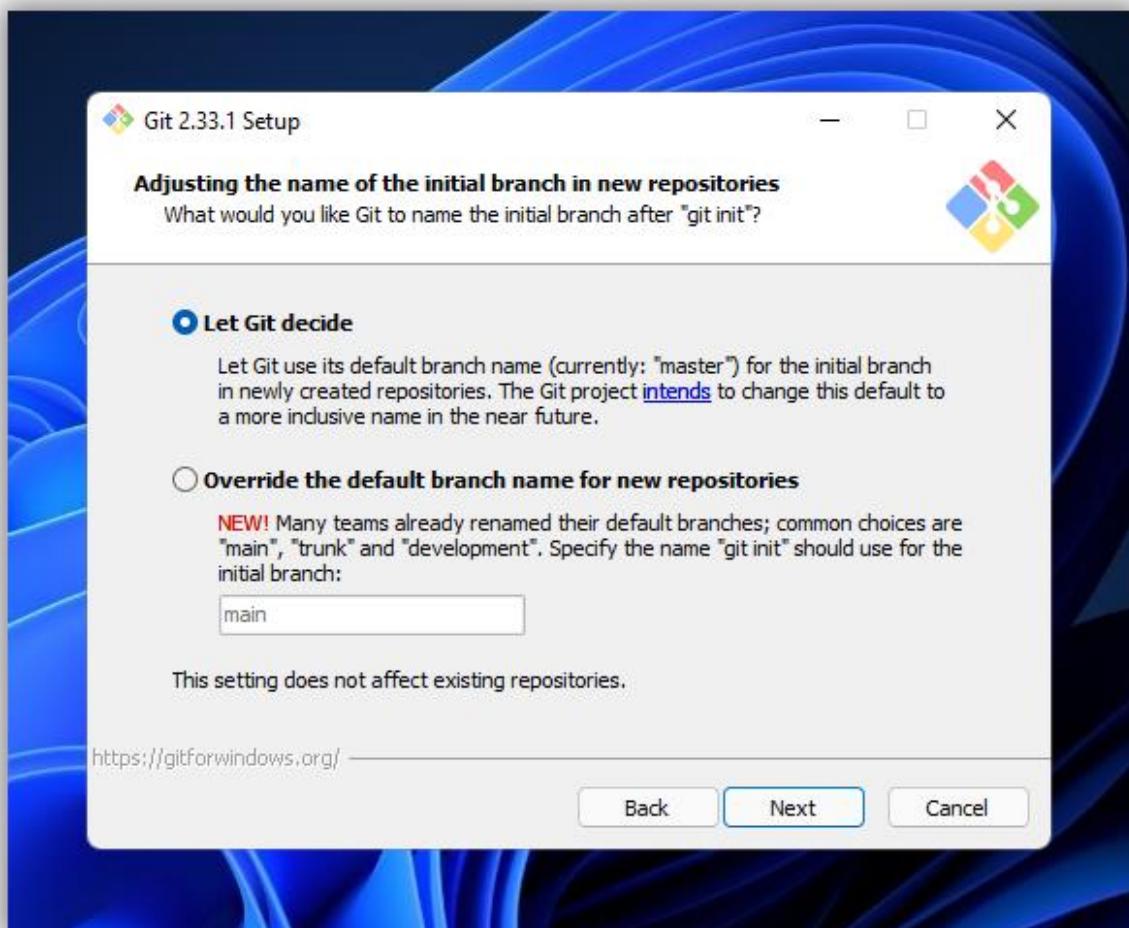
8. Installer akan menawarkan untuk membuat icon start menu di layar Desktop. Cukup klik **Next**.



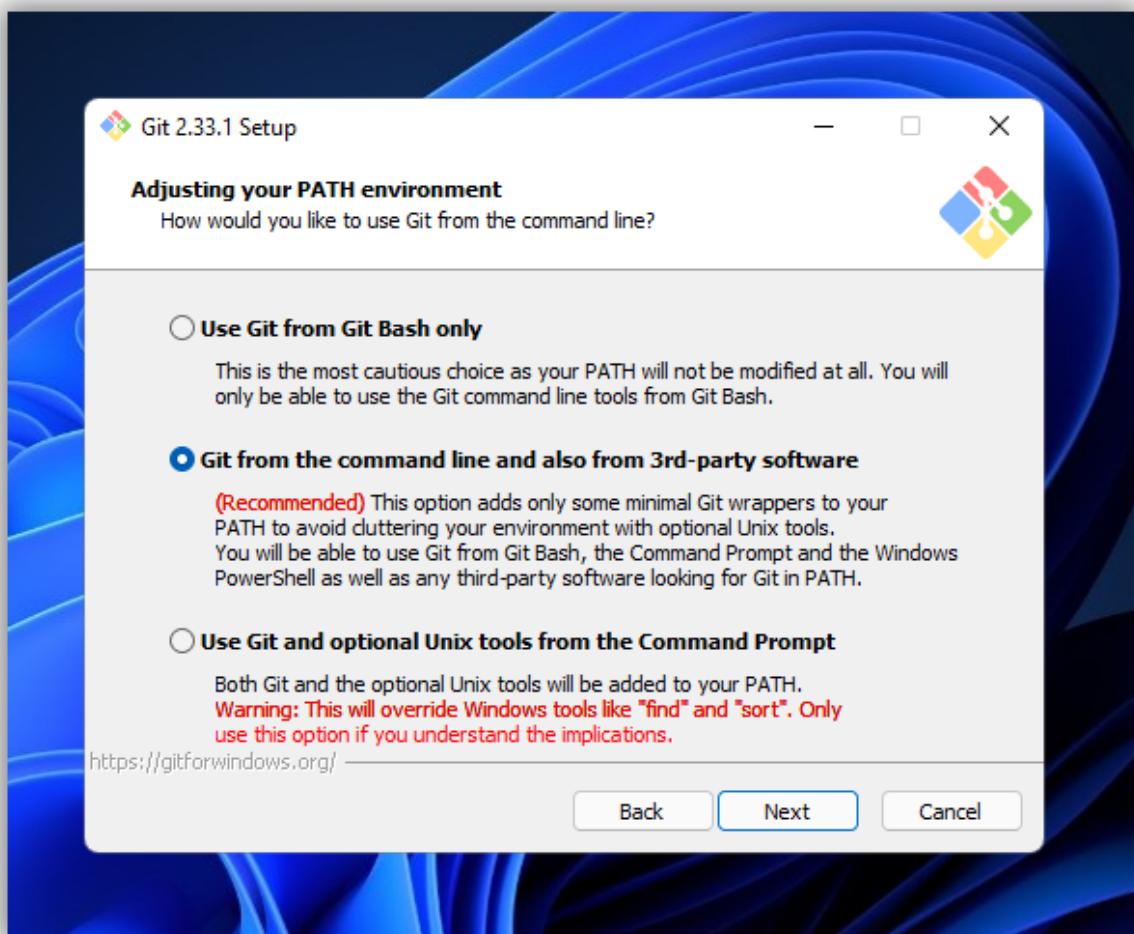
9. Pilih editor teks yang ingin kamu gunakan ( Kita akan menggunakan Visual Studio Code ) untuk Git. Setelah itu klik **Next**.



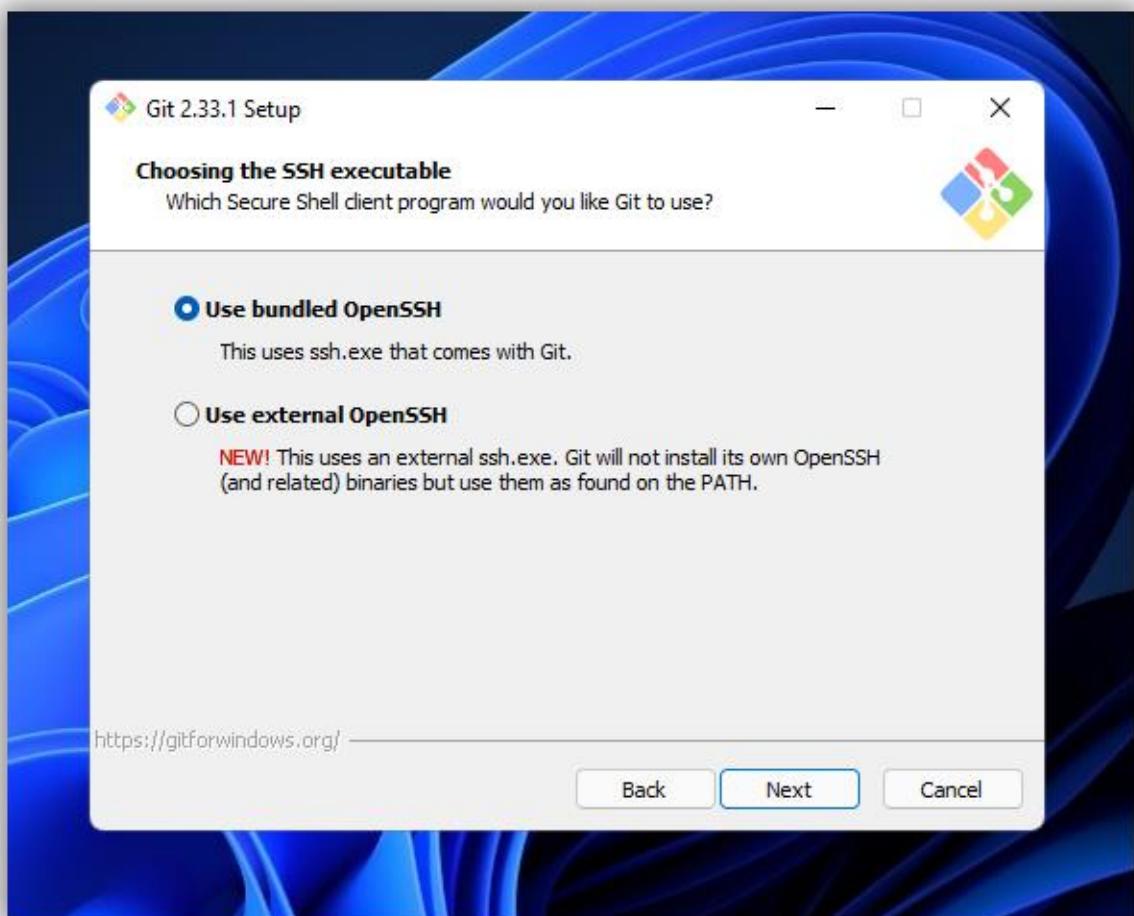
10. Langkah selanjutnya memungkinkan kamu untuk memilih nama yang berbeda untuk branch awal kamu. Standarnya adalah 'master'. Kecuali kamu bekerja dalam tim yang memerlukan nama berbeda, biarkan opsi default dan klik **Next**.



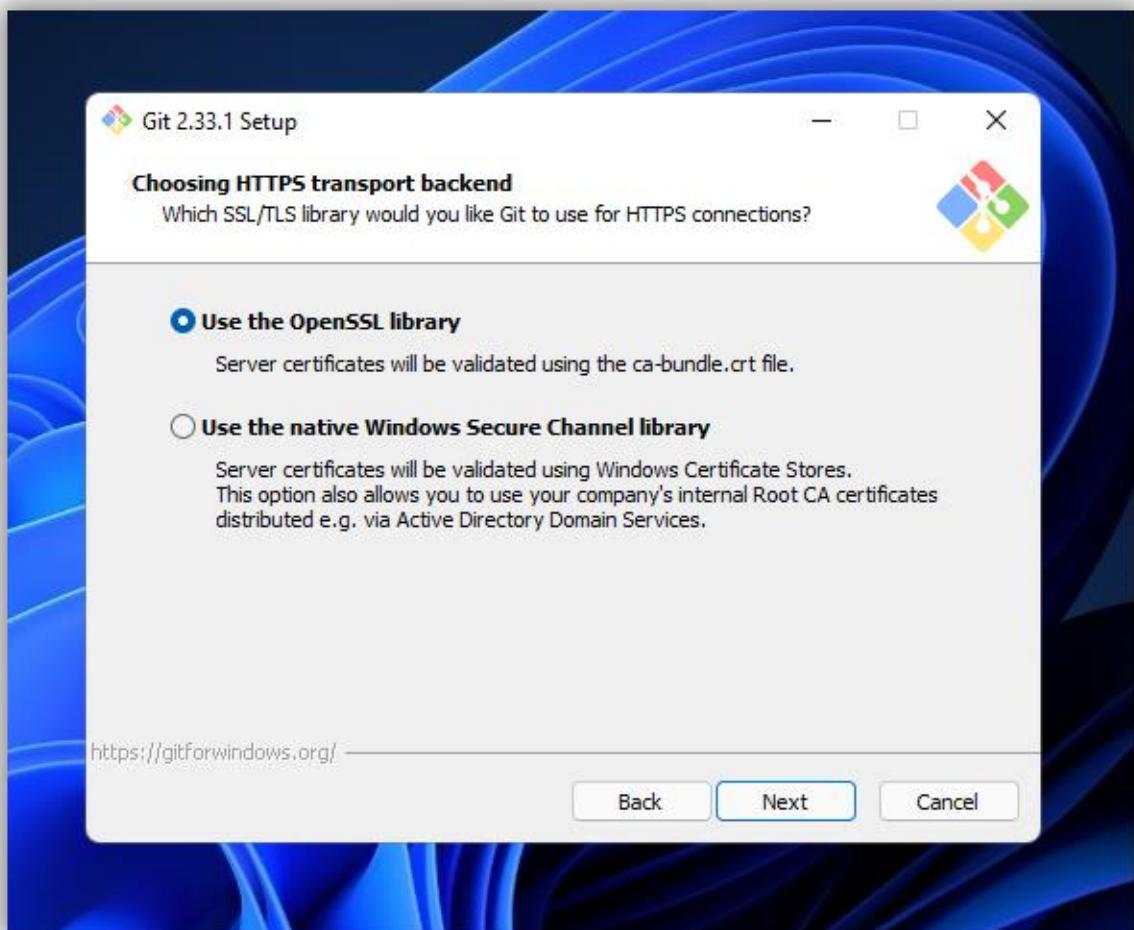
11. Langkah instalasi ini memungkinkan kamu untuk mengubah **lingkungan PATH instalasi**. **PATH** adalah set standar direktori yang disertakan saat kamu menjalankan perintah dari CMD. Biarkan pilihannya di tengah (rekomendasi) dan klik **Next**.



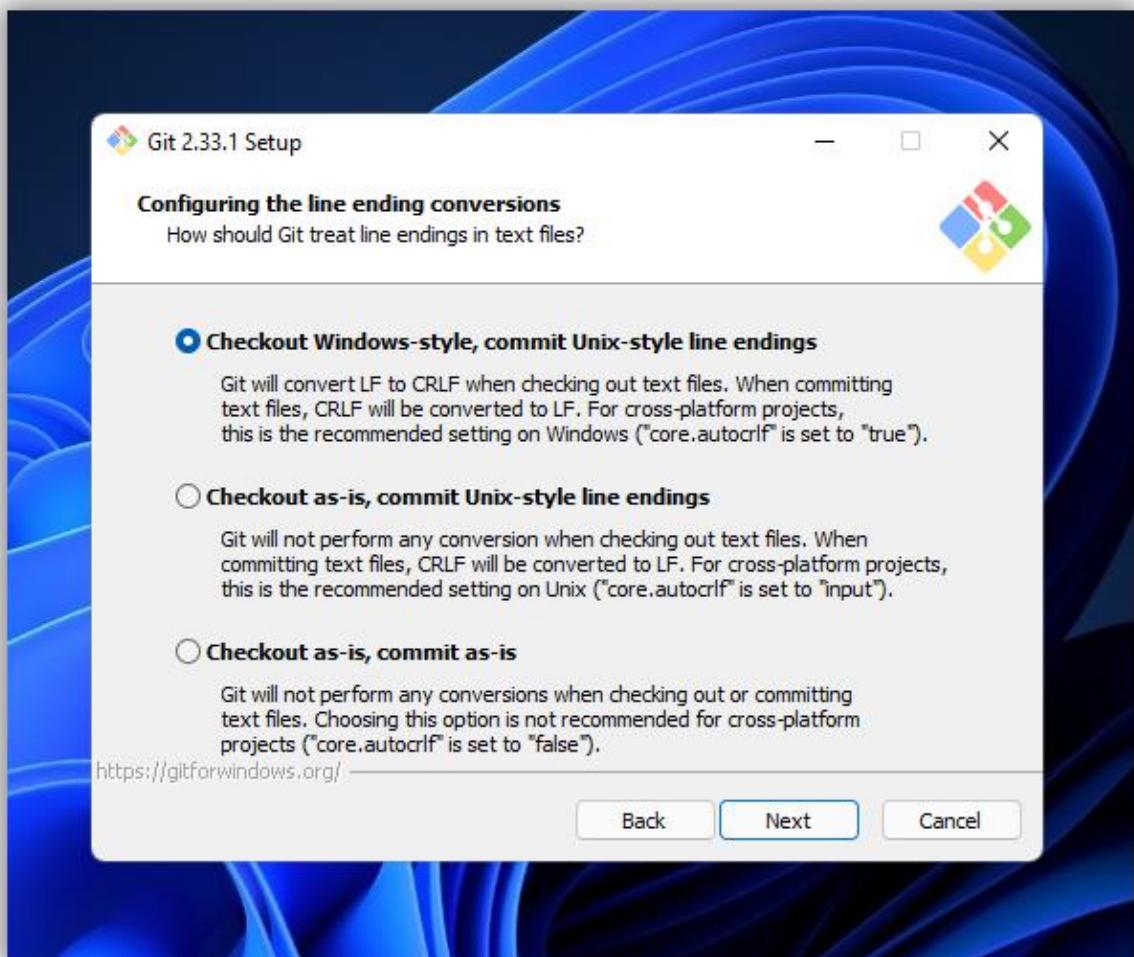
12. Installer sekarang menanyakan klien SSH mana yang ingin kamu gunakan pada git. Git sudah hadir dengan klien SSH-nya sendiri, jadi jika Anda tidak membutuhkan yang spesifik, biarkan opsi default dan klik **Next**.



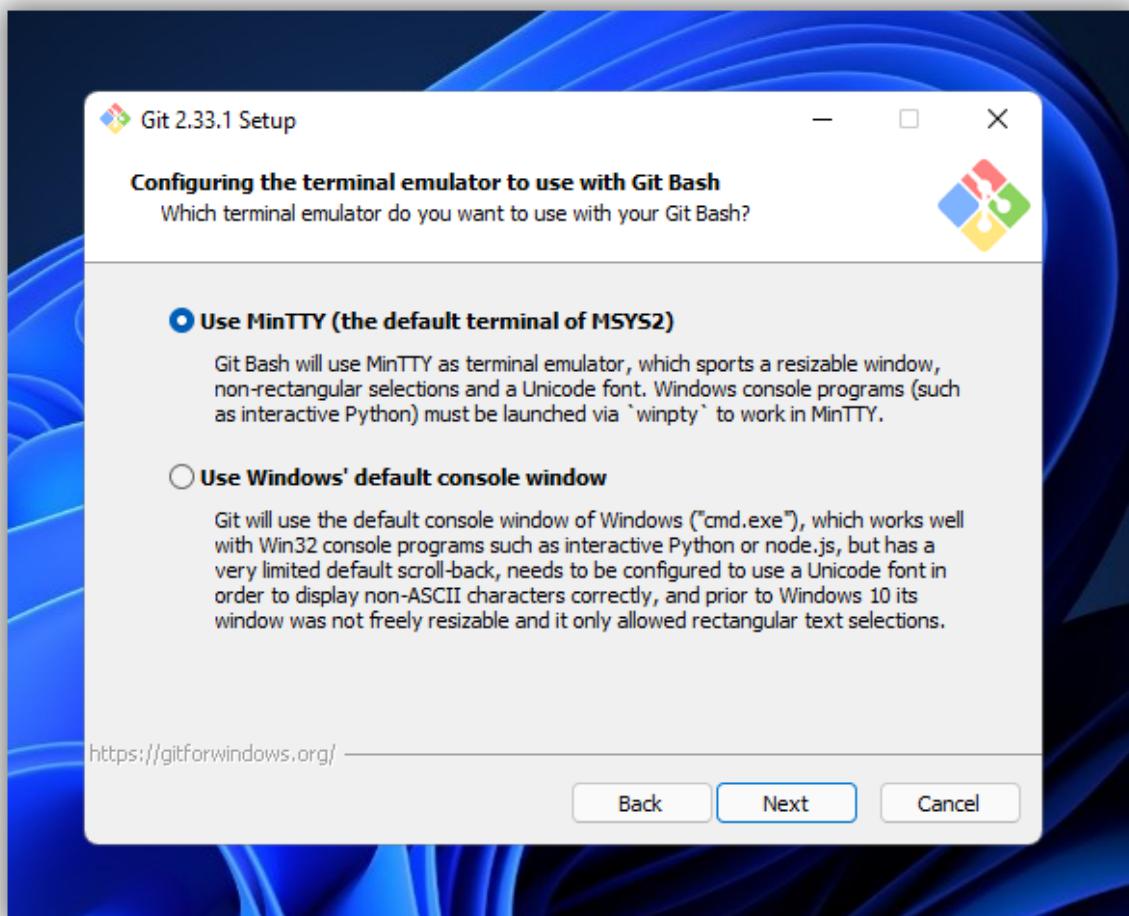
13. Opsi selanjutnya berkaitan dengan sertifikat server. Sebagian besar pengguna harus menggunakan default, klik **Next**.



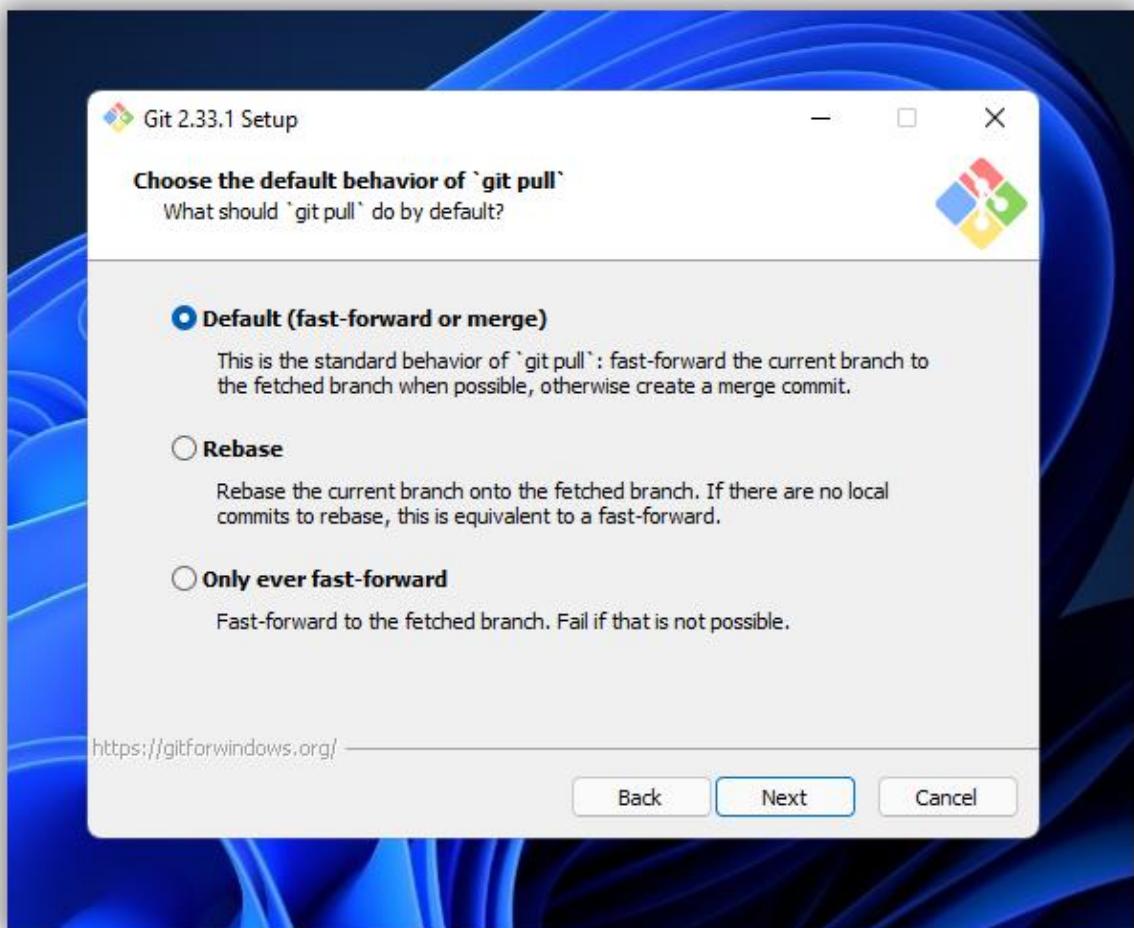
14. Selanjutnya konfigurasi line ending. Biarkan saja seperti ini, kemudian klik **Next**.



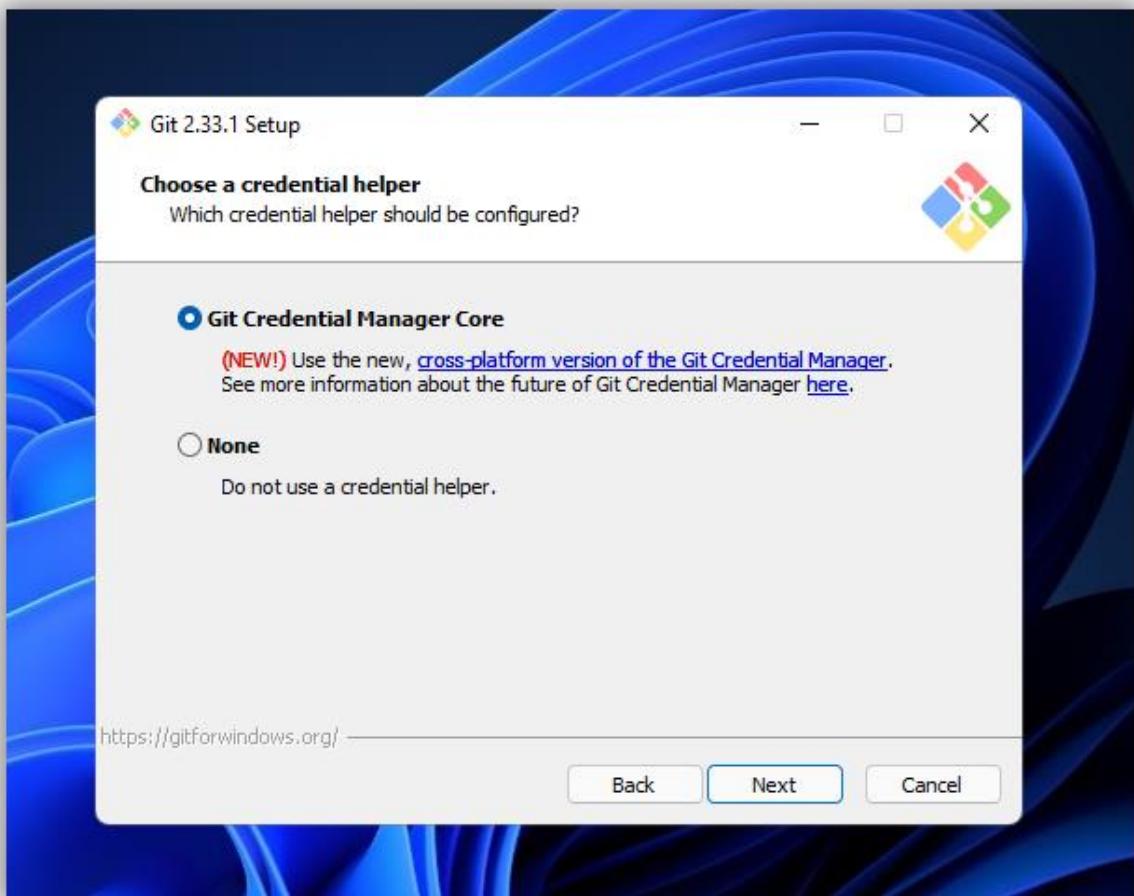
15. Lalu pemilihan terminal emulator. Pilih saja yang paling bawah, kemudian klik **Next**.



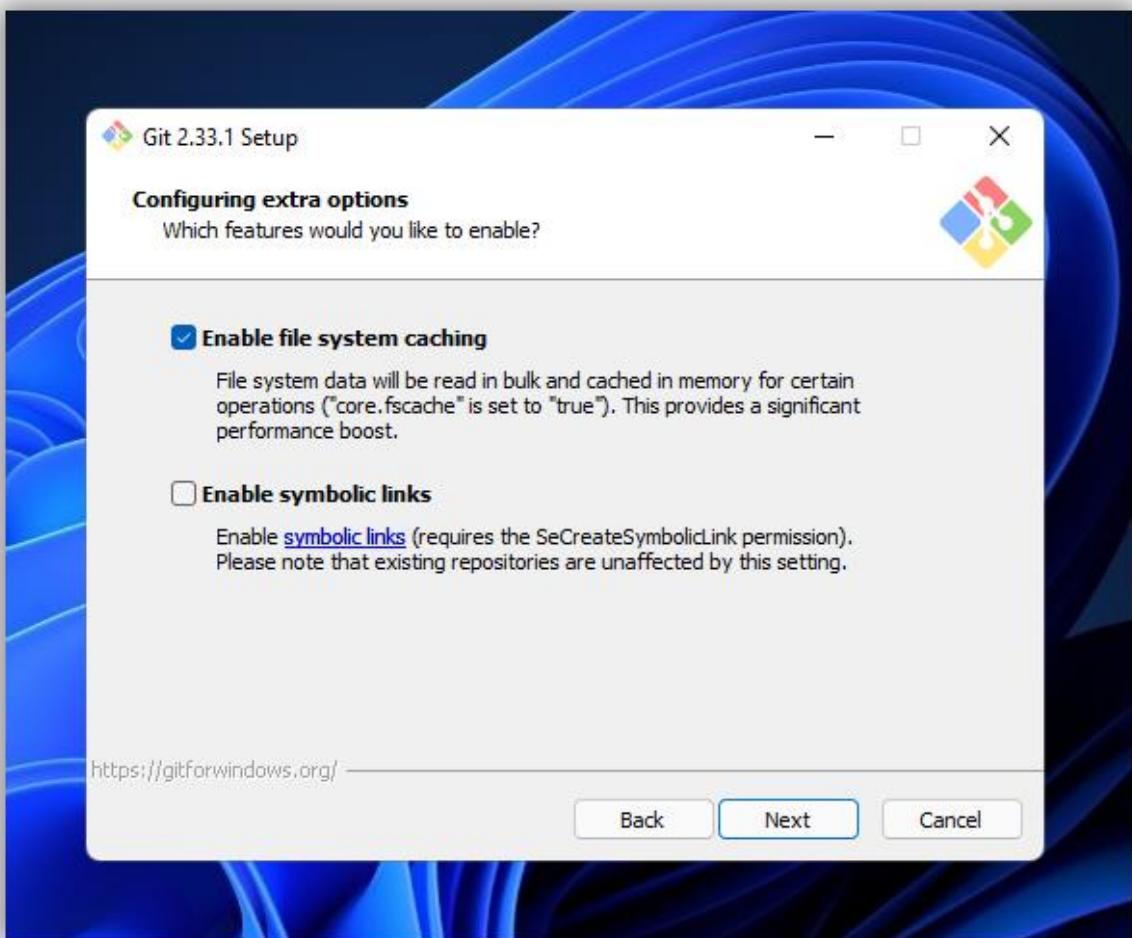
16. Installer sekarang menanyakan apa yang **git pull** harus dilakukan oleh perintah tersebut. Opsi default disarankan **Next** untuk melanjutkan instalasi.



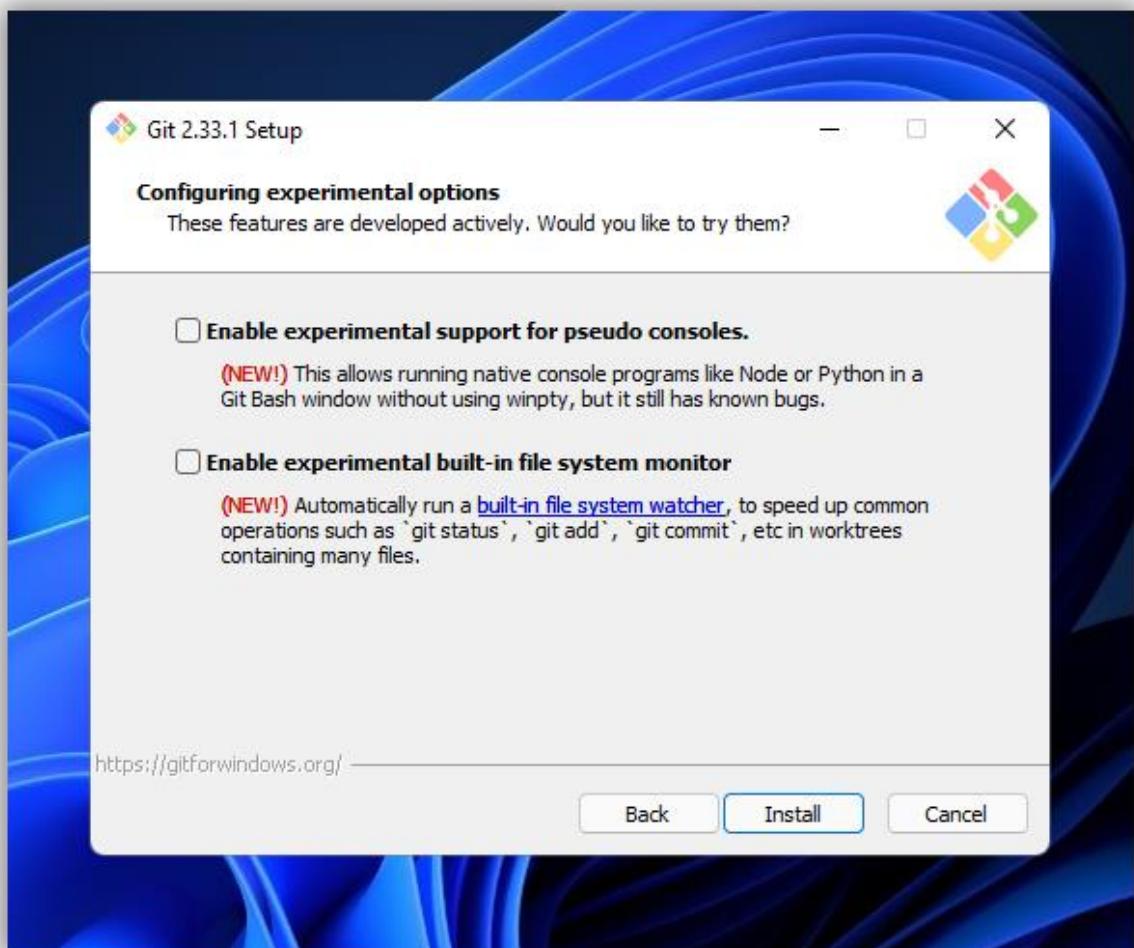
17. Selanjutnya kamu harus memilih alat kredensial mana yang akan digunakan. Git menggunakan alat kredensial untuk mengambil atau menyimpan kredensial. Biarkan opsi default dan klik **Next**.



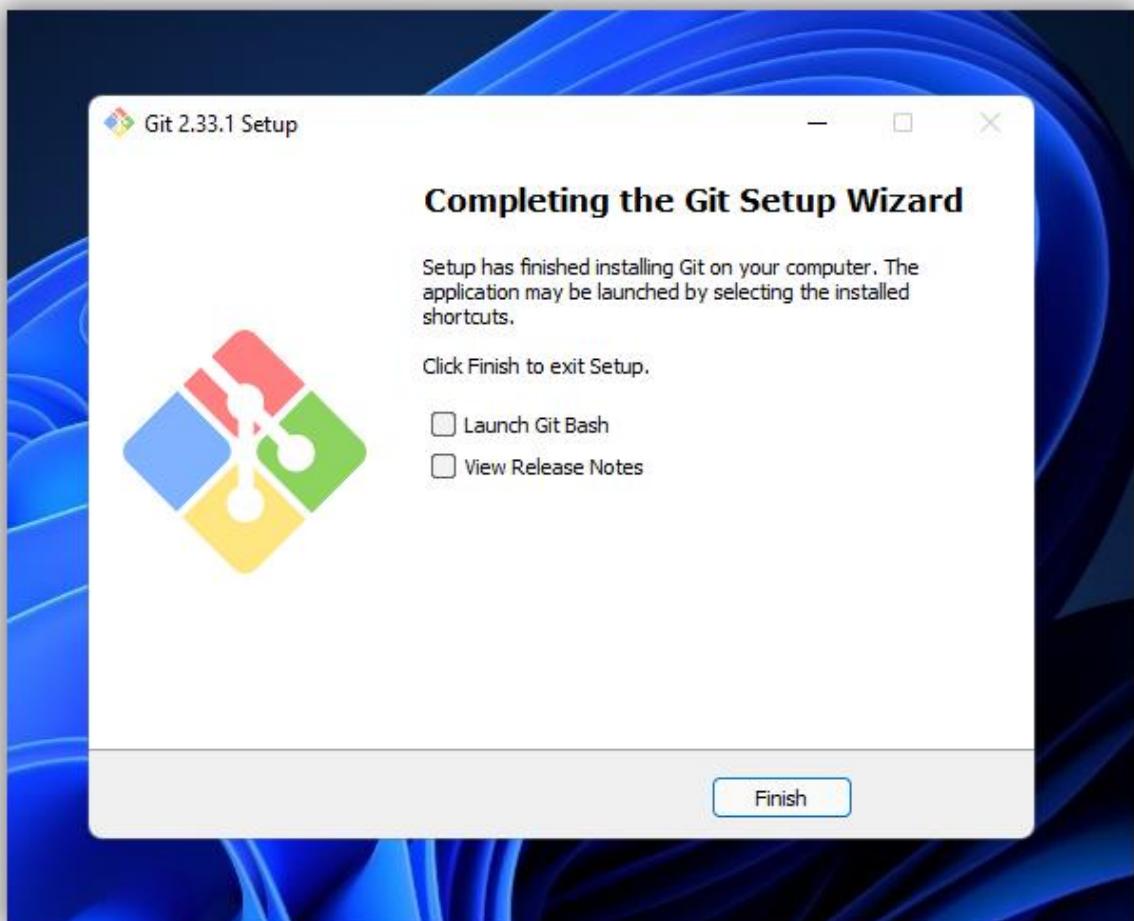
18. Selanjutnya pemilihan opsi ekstra. Klik saja **Next**.



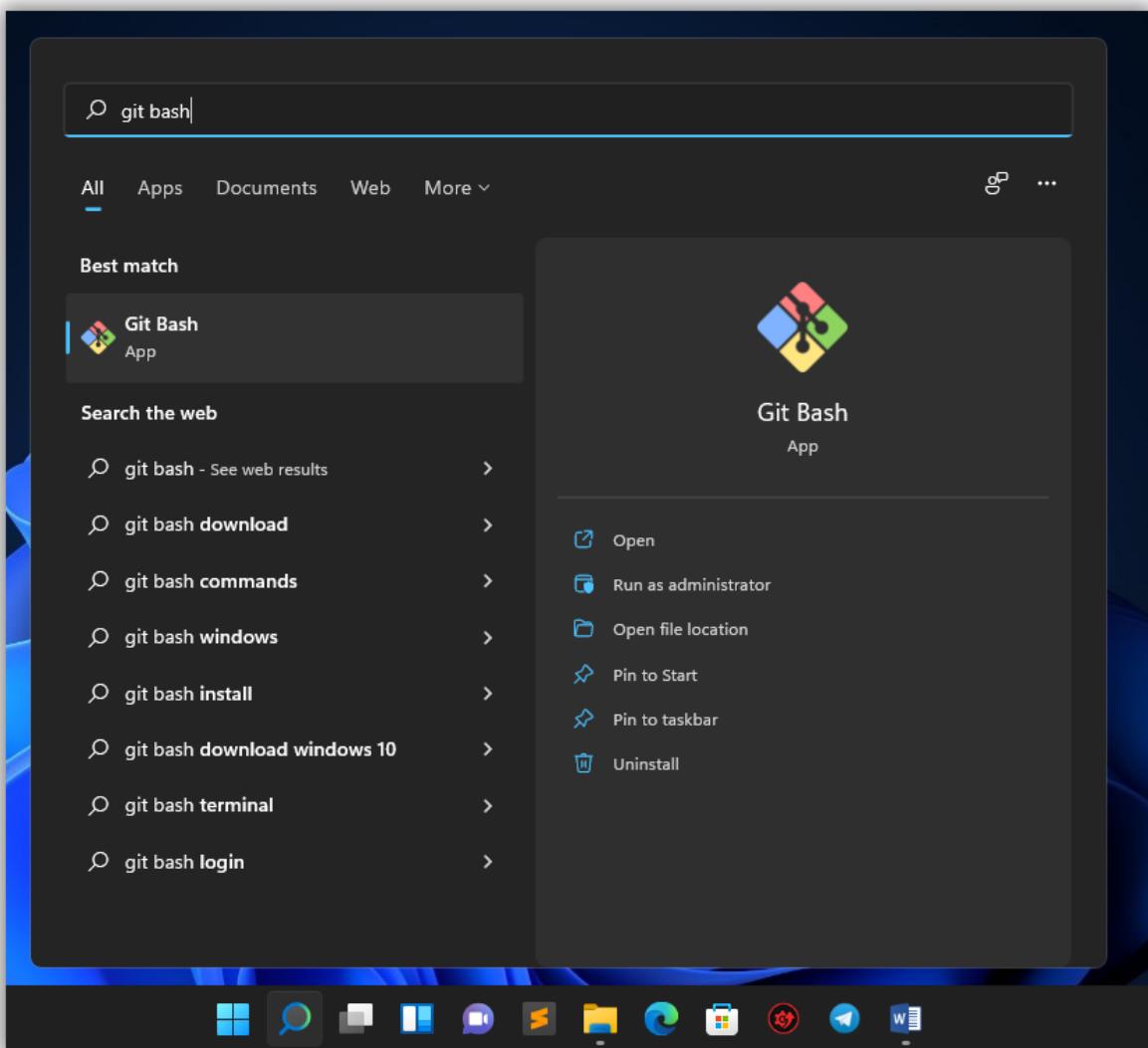
19. Instaler Git mungkin menawarkan untuk menginstal fitur eksperimental, tanpa centang apapun, langsung saja klik **Install**.



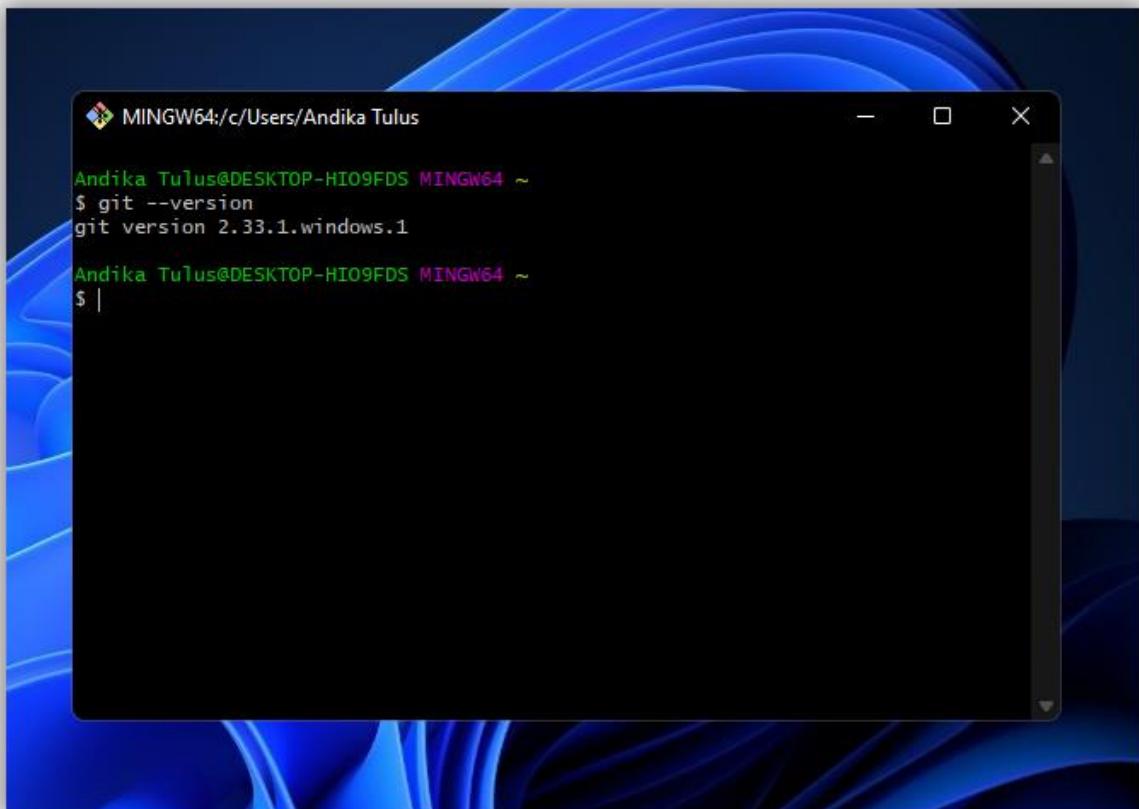
20. Setelah instalasi selesai, centang kotak untuk melihat Catatan Rilis atau Luncurkan Git Bash, lalu klik **Selesai** .



21. Untuk meluncurkan Git Bash, kamu tinggal mencari **Start Menu Windows** lalu **Enter** pada **Git Bash**.



22. kemudian ketik perintah *git -version*



A screenshot of a terminal window titled "MINGW64:/c/Users/Andika Tulus". The window shows the command \$ git --version followed by the output "git version 2.33.1.windows.1". The terminal is set against a blue abstract background.

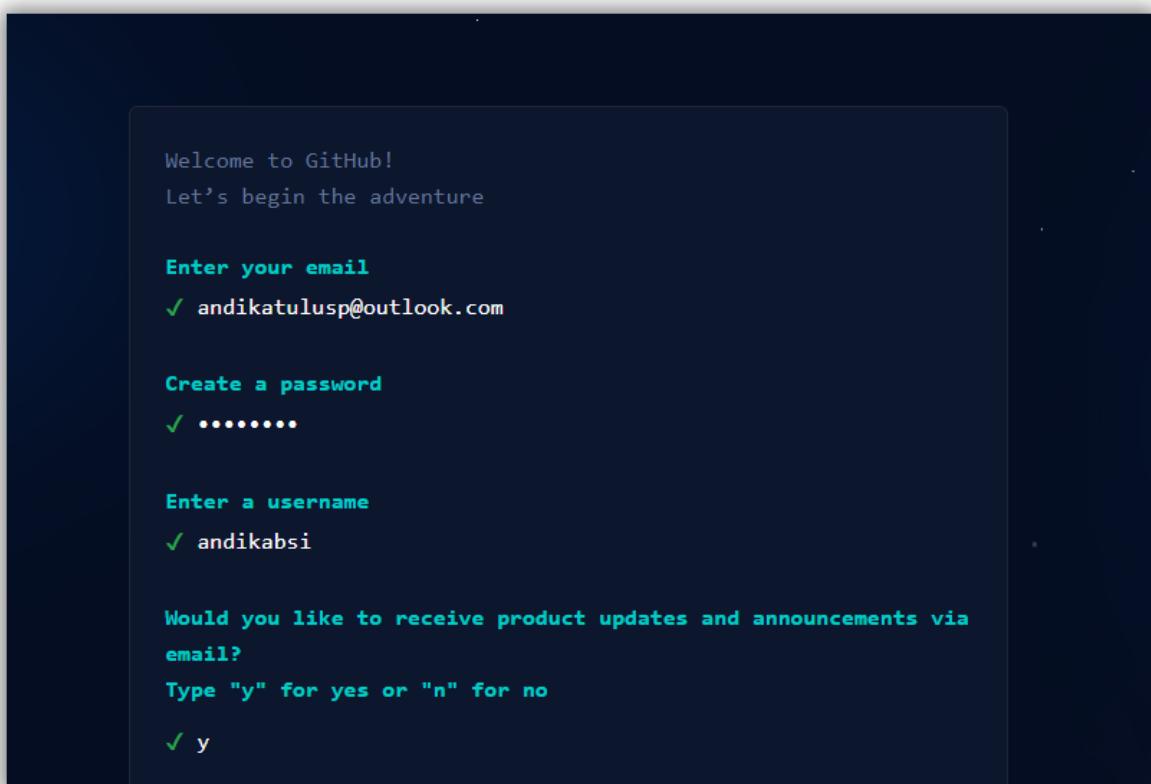
```
MINGW64:/c/Users/Andika Tulus
Andika Tulus@DESKTOP-HI09FDS MINGW64 ~
$ git --version
git version 2.33.1.windows.1
Andika Tulus@DESKTOP-HI09FDS MINGW64 ~
$ |
```

## D.Membuat Akun Github

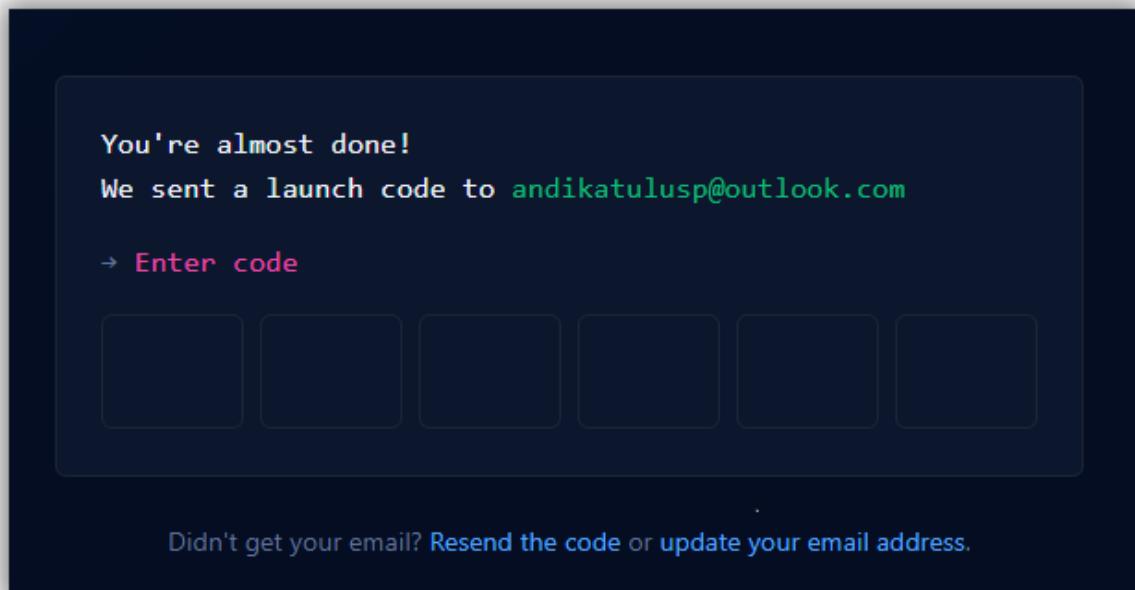
1. Buka Situs Github di <https://github.com>
2. Klik “Sign Up” di pojok kanan atas, untuk mendaftar



3. Lalu lengkapi form pendaftarannya seperti username, password dll.



4. Masukan kode verifikasi yang telah dikirim ke email yang telah didaftarkan.



5. Lengkapi beberapa pertanyaan sperti berikut ini.

Welcome to GitHub

We are glad you're here.

How many team members will be working with you?

This will help us guide you to the tools that are best suited for your projects.

Just me    2 - 5    5 - 10  
10 - 20    20 - 50    50+

Are you a student or teacher?

Student    Teacher

Continue

The tools you need to build what you want.

Soup to nuts, GitHub has it all.

What specific features are you interested in using?

Select all that apply so we can point you to the right GitHub plan.

Collaborative coding  
Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.

Automation and CI/CD  
Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.

Security  
Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.

6. Pilih “Free” untuk melanjutkan pendaftaran akun github.

The screenshot shows the GitHub student benefits landing page. On the left, under the "Free" section, there are four items: "Unlimited public/private repositories", "2,000 Actions minutes/month" (free for public repositories), "500MB of Packages storage" (free for public repositories), and "Community support". On the right, under "Get additional student benefits", there are two sections: "GitHub Pro" and "GitHub Student Developer Pack". The "GitHub Pro" section lists "Protect your branches", "Draft pull requests", "Pages and Wikis", "3,000 CI/CD minutes/month" (free for public repositories), "2GB of Packages storage" (free for public repositories), and "Web-based support". The "GitHub Student Developer Pack" section lists "Free access to the industry's best developer tools" (including DigitalOcean, Microsoft Azure, Heroku, MongoDB, DataDog, Twilio, and Stripe). At the bottom, there are two buttons: "Continue for free" on the left and "Apply for your GitHub student benefits" on the right.

7. Jika berhasil, maka akan tampil dashboard github kamu.

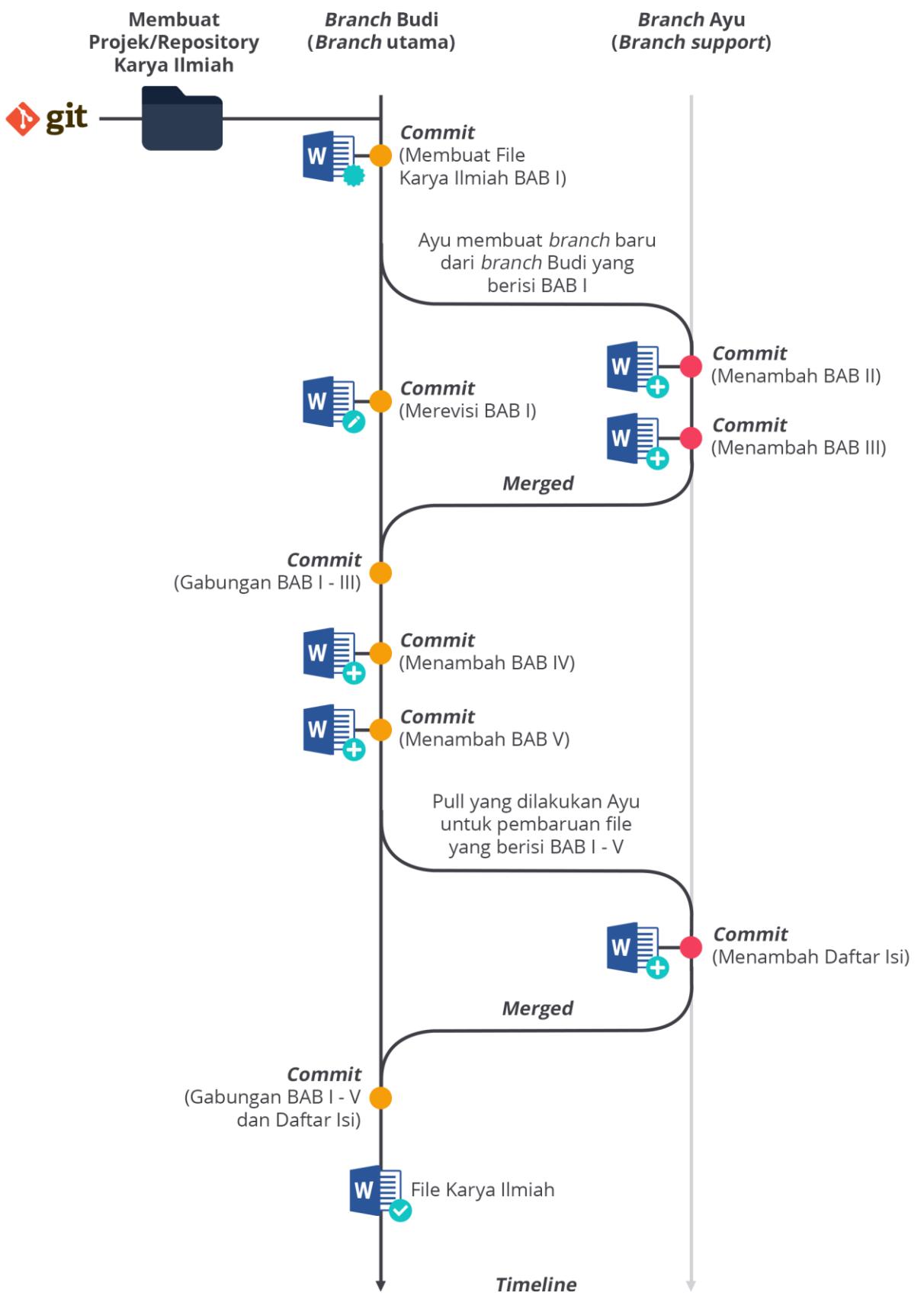
The screenshot shows the GitHub dashboard. The top navigation bar includes "Pull requests", "Issues", "Marketplace", and "Explore". The main area features a "Create your first project" section with a "Create repository" button and an "Import repository" link. Below it is a "Recent activity" section. The central part of the dashboard displays the "All activity" news feed, which includes a callout for "Discover interesting projects and people to populate your personal news feed" and a "ProTip!" note. A notification bubble on the right says "Universe 2021" and "Missed any Universe sessions? All content is available on demand now so you can view on your own time." At the bottom, there are links for "Blog", "About", "Shop", "Contact", "GitHub", and "Pricing", along with "API", "Training", "Status", "Privacy", and "Docs".

## E. Memahami Alur Kerja Git

Pada Git, kita dapat membagi alur kerja dari setiap anggota dalam tim dengan menggunakan fitur *branch* atau cabang kerja. Fitur ini berfungsi untuk menciptakan alur kerja baru sehingga tidak akan mengganggu riwayat atau catatan perubahan pada cabang kerja utama.

Dengan begitu, setiap anggota memiliki alur kerjanya masing-masing dan dapat bekerja secara paralel sampai tiba waktunya untuk menggabungkan setiap pekerjaan dengan menggunakan. Hasil akhir dari gabungan kerja masing-masing anggota dapat kita buatkan *tag* atau versi rilis dari berkas tersebut.

Ilustrasi Alur Kerja Git :



( Source : dicoding.com )

Gambar ilustrasi di atas merupakan contoh penggerjaan proyek pembuatan jurnal karya ilmiah menggunakan Git yang mana proyek ini dikerjakan oleh dua orang, yaitu **Budi** dan **Ayu**.

Sebelum mengerjakan, Budi dan Ayu sepakat untuk membuat dan mengerjakannya dengan menggunakan Git sekaligus melakukan pembagian tugas masing-masing. Hal pertama yang Budi lakukan yaitu membuat *repository* di GitHub, lalu menambahkan akun Github Ayu sebagai *contributor* atau *collaborator* dalam *repository* yang baru saja dibuat dengan nama Proyek Karya Ilmiah.

Pembagian tugas pun telah diberikan di mana Budi akan mengerjakan BAB I, BAB IV, dan BAB V. Sementara itu, Ayu akan mengerjakan BAB II, BAB III, dan Daftar Isi.

Awalnya, *repository* hanya memiliki satu cabang kerja utama, yaitu *branch* Budi. Lalu, setelah Budi melakukan *commit* pertama di mana pada *commit* tersebut Budi membuat File Karya Ilmiah yang berisi BAB I, Ayu melakukan pembuatan *branch* baru yaitu *branch support* dengan *commit* yang baru saja dilakukan Budi, kemudian Ayu mengerjakan bagian tugasnya. Mereka pun mengerjakan proyek tersebut secara paralel.

Pada saat Ayu masih mengerjakan tugasnya, Budi melakukan pembaruan terhadap tugasnya di BAB I (revisi). Setelah Ayu selesai mengerjakan BAB II dan BAB III, mereka melakukan merge untuk menggabungkan BAB I, BAB II, dan BAB III. Lalu Budi melanjutkan tugasnya untuk membuat BAB IV dan BAB V. Pada saat BAB IV dan BAB V selesai dikerjakan, Ayu melakukan *pull* (penarikan hasil kerja) dari *branch* utama atau *branch* Budi lalu melanjutkan mengerjakan Daftar isi.

Setelah selesai mengerjakan daftar isi, dengan fitur merge Budi dan Ayu dapat menggabungkan cabang kerja (*branch*) sehingga hasil kerja dari cabang yang akan di-*merge* akan bergabung ke dalam cabang kerja utama. Dengan begitu, mereka pun

melakukan *merger* atau penggabungan kerja antara pekerjaan Budi. dan pekerjaan Ayu. Dari hasil *merge* inilah akan membentuk file karya ilmiah yang lengkap dan utuh.

Itu hanyalah gambaran cara kerja dari git, banyak istilah yang mungkin asing, tapi tenang. Istilah-istilah tersebut akan kita pelajari pelan-pelan.

# **Bekerja Dengan Git dan Github ( Membuat First Project )**

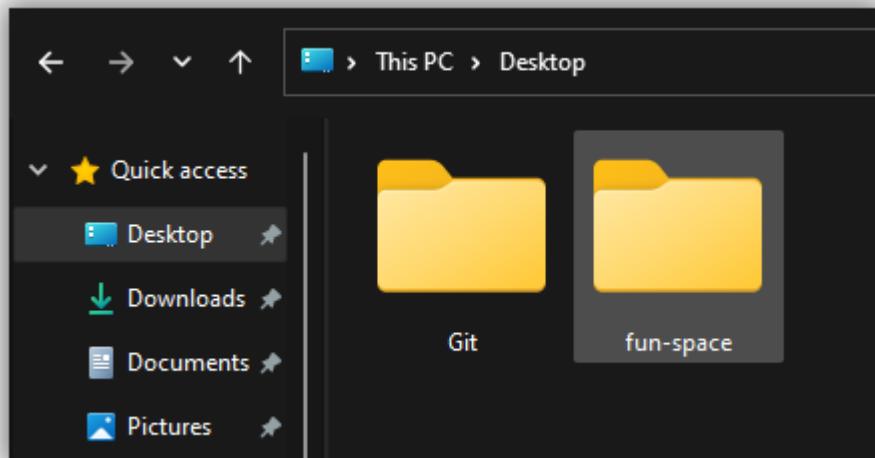
Sebelum kita memulai melakukan segala hal dengan git dan github, kita akan mencoba membuat sebuah program bernama “Fun Space With Python” ( Program ini dibuat untuk menghitung luas bangun datar ), untuk pertama kalinya kita akan membuatnya sendiri ( individu ) dan setelah itu, nanti kita akan melakukan kolaborasi untuk membuat aplikasi ini dengan orang lain.

Persiapan Project Aplikasi :

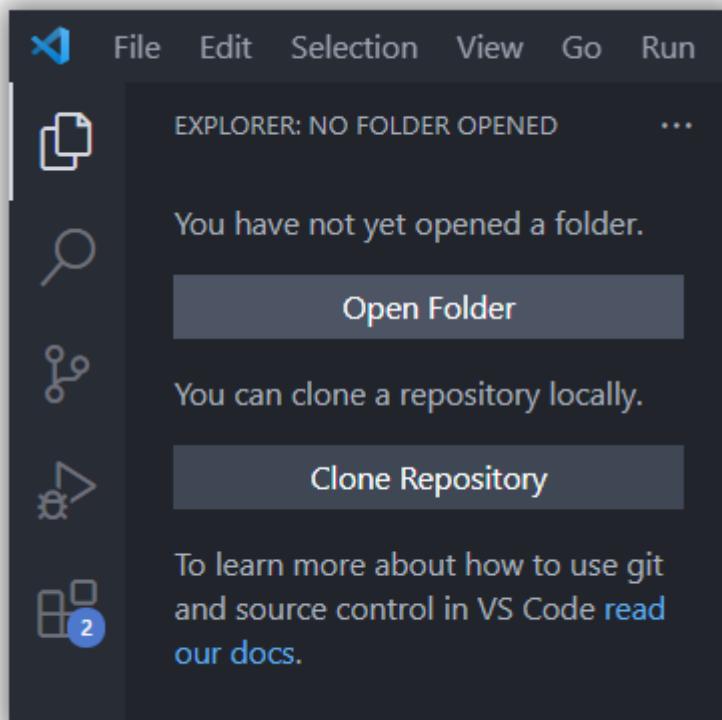
- Laptop dengan OS Windows yang telah terinstall Git
- Telah membuat Akun Github
- Memiliki koneksi internet
- Telah menginstall Kode Editor Visual Studio Code

Langkah – langkah :

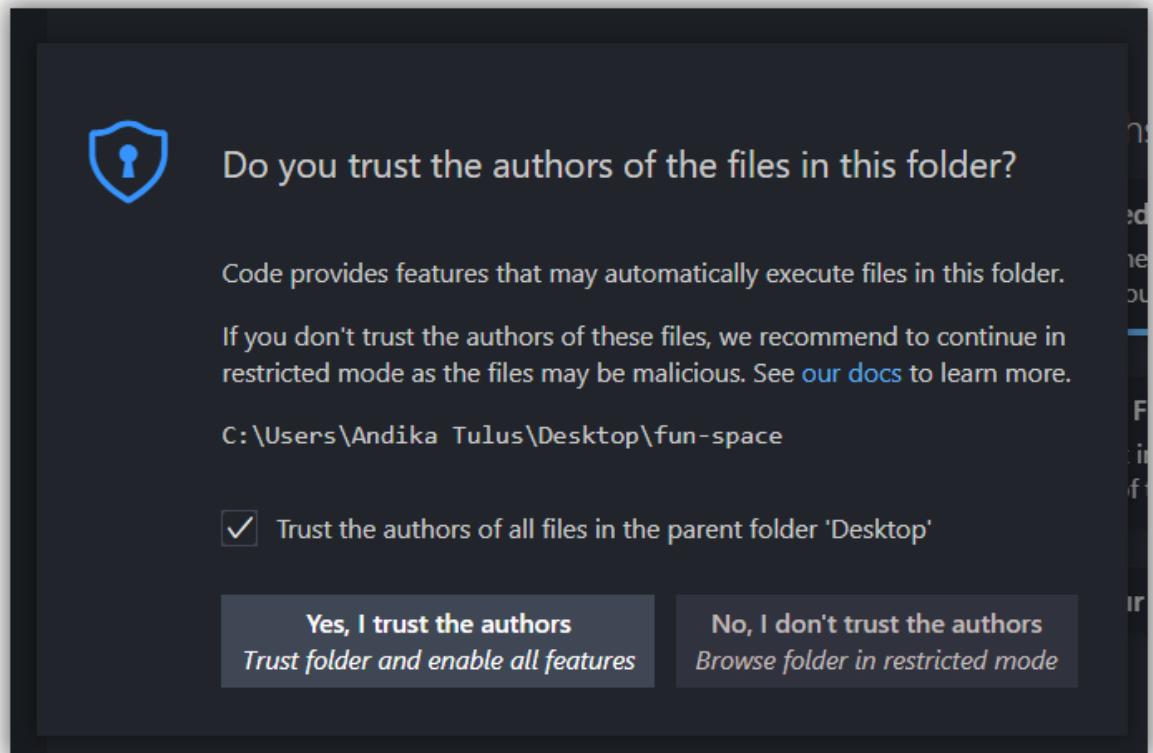
- Buat Folder Project bernama “fun-space” dan Buka di Visual Studio Code



- Klik “Open Folder” lalu pilih folder yang telah kita buat tadi.



- Jika terdapat pop up “Trust Folder Author” maka pilih “Yes, i trust the authors”



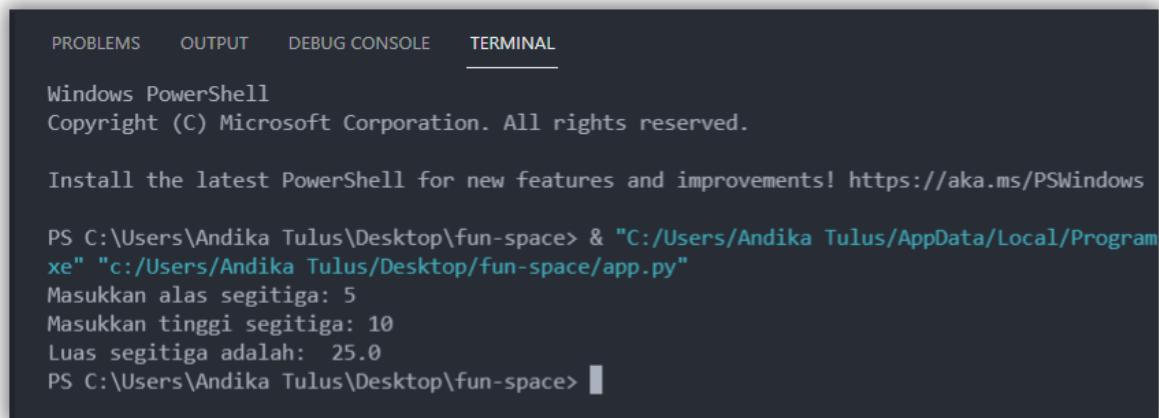
- Buat File baru bernama “app.py” lalu ketik kode seperti berikut.

A screenshot of Visual Studio Code showing a Python script named "app.py". The code calculates the area of a triangle based on user input for base and height. The code is as follows:

```
# Hitung Luas Segitiga sederhana
def luas_segitiga():
    a = int(input("Masukkan alas segitiga: "))
    t = int(input("Masukkan tinggi segitiga: "))
    luas = a * t / 2
    print("Luas segitiga adalah: ", luas)

luas_segitiga()
```

- Kita coba jalankan dan berhasil tanpa eror



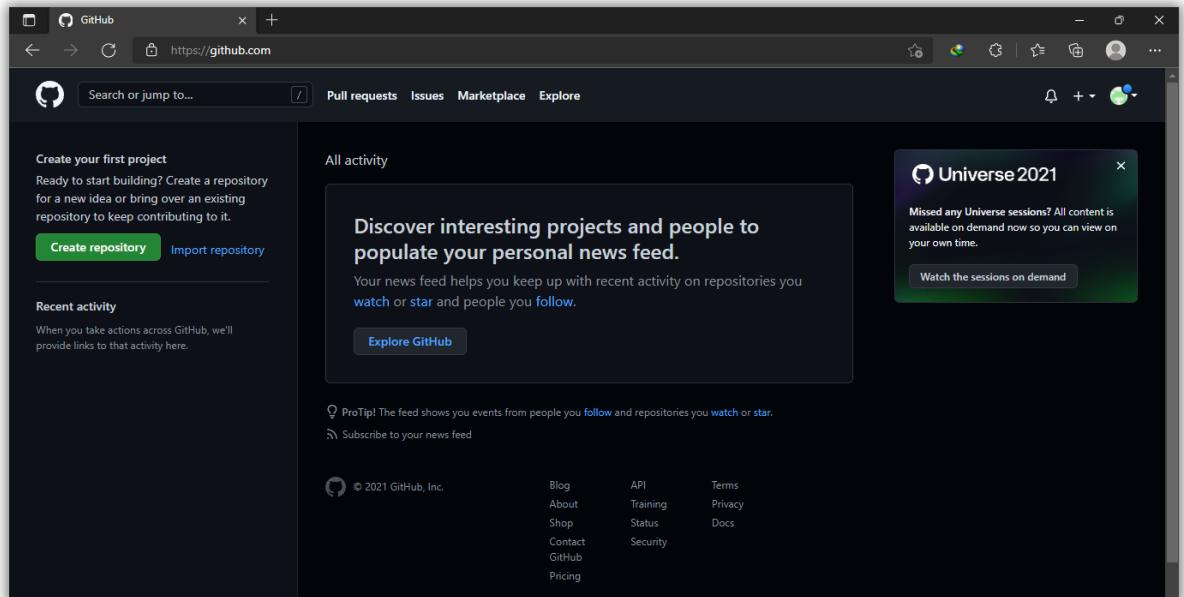
The screenshot shows a terminal window with the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

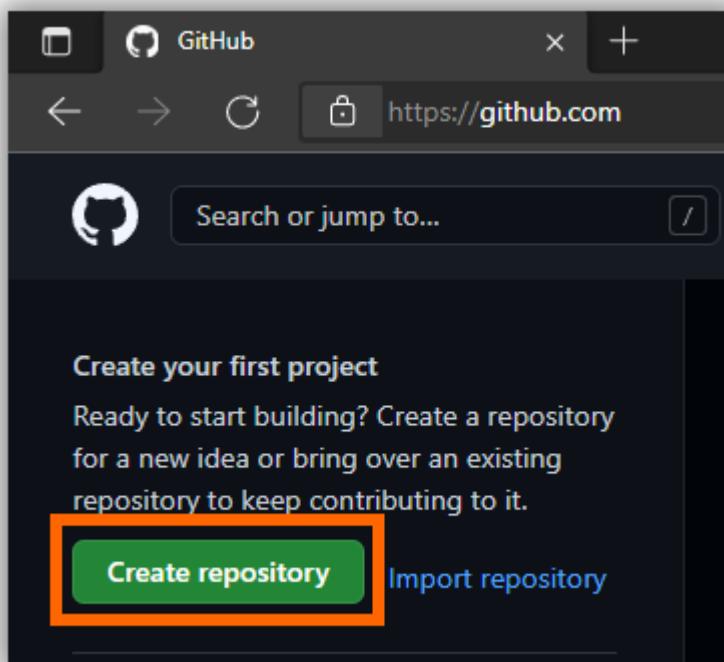
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Andika Tulus\Desktop\fun-space> & "C:/Users/Andika Tulus/AppData/Local/Program
xe" "c:/Users/Andika Tulus/Desktop/fun-space/app.py"
Masukkan alas segitiga: 5
Masukkan tinggi segitiga: 10
Luas segitiga adalah: 25.0
PS C:\Users\Andika Tulus\Desktop\fun-space>
```

- Karena kita ada tugas lain, maka program aplikasi ini tidak bisa dilanjutkan, tapi kita ingin kode aplikasi kita bisa disimpan dengan aman dan suatu saat jika kita mengerjakannya lagi kita bisa tau, apa yang terakhir kali kita lakukan terhadap kode aplikasi kita. Untuk memudahkan hal tersebut mari kita memasuki materi mengenai “Mengelola Source Code dengan Git dan Github”
- Kita buat dulu Repository di Github. **Repository adalah folder untuk menyimpan berkas-berkas project kita secara online, bahkan setiap perubahannya akan disimpan secara otomatis.** (**Seperti membuat Folder di Google Drive atau Onedrive**)
- Untuk membuatnya, kita buka dan masuk menggunakan akun github yang telah didaftarkan di <https://www.github.com>
- Sekarang setelah berhasil masuk, maka kita berada di halaman Dashboard github



- Untuk membuat repository, maka klik “Create Repository”



- Setelah itu akan ada beberapa form untuk memberikan beberapa informasi mengenai Folder (Kita sekarang akan menyebutnya sebagai Repository) yang akan kita buat.
  - a. Repository name  
Ini adalah form untuk memberikan nama repository project kita ( Sering kita lakukan seperti membuat nama

Folder ), disini kita akan memberi nama repositori “**fun-space**”

b. Descriptions

Deskripsi untuk memberikan keterangan singkat tentang repository. Disini kita akan memberikan deskripsi **“Source Code Aplikasi Hitung Luas Bangun Datar”**

c. Repository Visibility

Seperti pada umumnya, kita bisa menyetel repository kita dengan mode **Privat** ( **Hanya bisa diakses oleh pemilik repository** ) atau **Public** ( **Bisa diakses semua orang, dan bisa dilakukan Pull, Fork, Clone dll oleh semua orang.** )

d. Additional Options

Ada beberapa pengaturan tambahan dibawahnya, seperti:

- **Add Readme File** ( Membuat deskripsi repository secara detail dengan file Readme )
- **Add .gitignore** ( Jika ini diaktifkan, maka riwayat perubahan tidak akan dicatat oleh sistem Git )
- **Choose License** ( Memilih jenis lisensi untuk repository project )

Owner \* Repository name \*

 andikabsi / fun-space ✓

Great repository names are short and memorable. Need inspiration? How about [studious-octo-barnacle](#)?

Description (optional)

Source Code Aplikasi Hitung Luas Bangun Datar

 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:**

Skip this step if you're importing an existing repository.

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

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

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

**Create repository**

- Setelah semuanya diisi maka klik tombol “Create Repository”
- Jika berhasil maka akan tampil seperti ini

andikabsi / fun-space Public

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

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

Set up in Desktop or HTTPS SSH <https://github.com/andikabsi/fun-space.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 "# fun-space" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/andikabsi/fun-space.git
git push -u origin main
```

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

```
git remote add origin https://github.com/andikabsi/fun-space.git
git branch -M main
git push -u origin main
```

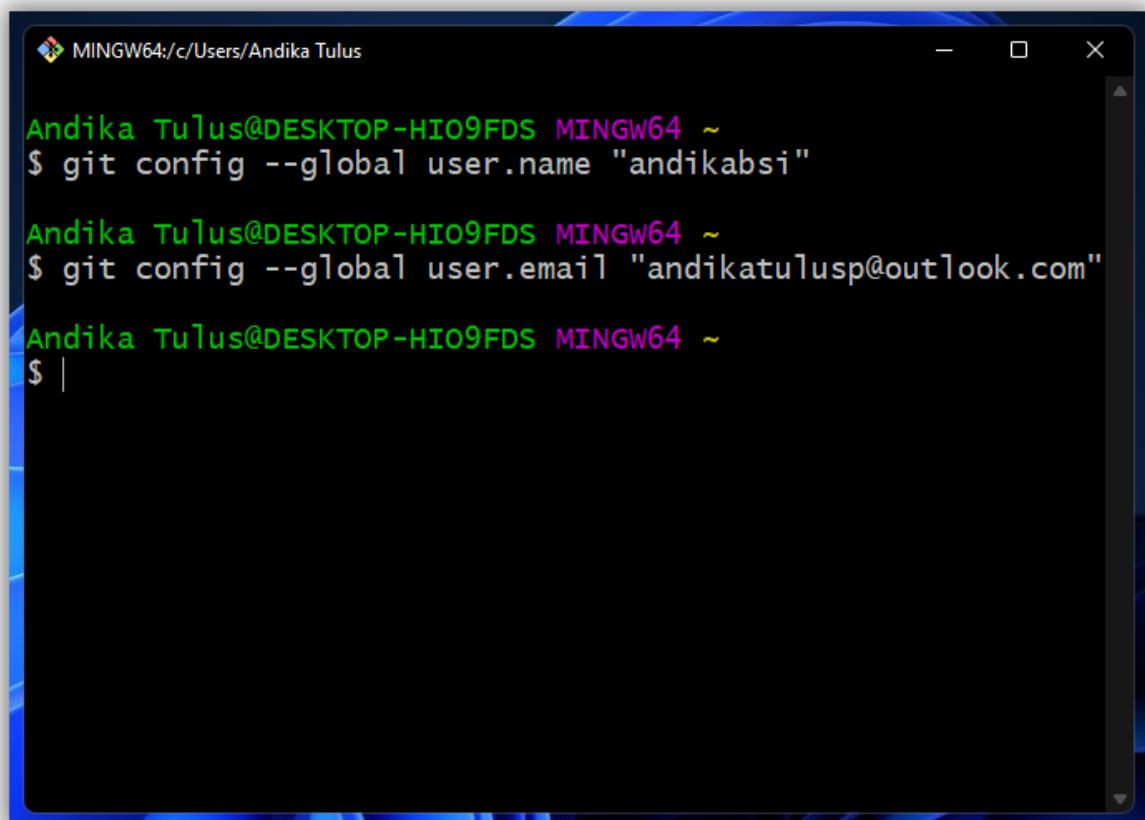
- Sekarang minimize browser kita, dan abaikan saja. Tapi jangan close browsernya.
- **Buka Git bash**, caranya seperti yang diatas kita bahas sebelumnya.
- Setelah terbuka, kita **konfigurasi Git terlebih dahulu**, terdapat beberapa konfigurasi yang harus dipersiapkan sebelum mulai menggunakan Git. Silakan lakukan konfigurasi dengan perintah berikut ini.

```
git config --global user.name "USERNAME AKUN GITHUB"
```

Tekan **ENTER** lalu masukan kembali perintah berikut dan **ENTER**

```
git config --global user.email nama@mail.com
```

Pada [nama@mail.com](mailto:nama@mail.com) silahkan ganti dengan email yang kamu daftarkan di Github.



The screenshot shows a terminal window titled 'MINGW64:/c/Users/Andika Tulus'. It displays three commands entered by the user:

```
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ git config --global user.name "andikabsi"

Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ git config --global user.email "andikatulusp@outlook.com"

Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ |
```

- Jika sudah, mari kita cek. Apakah confignya sudah tersimpan atau belum menggunakan perintah *git config --list*

```
◆ MINGW64:/c/Users/Andika Tulus

Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ git config --global user.name "andikabsi"

Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ git config --global user.email "andikatulusp@outlook.com"

Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ 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/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
core.editor="C:\Users\Andika Tulus\AppData\Local\Programs\Microsoft vs Code\bin\code.cmd" --wait
user.name=andikabsi
user.email=andikatulusp@outlook.com

Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ |
```

- Jika tampil seperti gambar diatas, maka selamat. Konfigurasi git di komputer lokal telah berhasil.
- Sekarang, kita **mulai mengunggah berkas project** aplikasi kita ( Dalam git kita sebut Git Push )

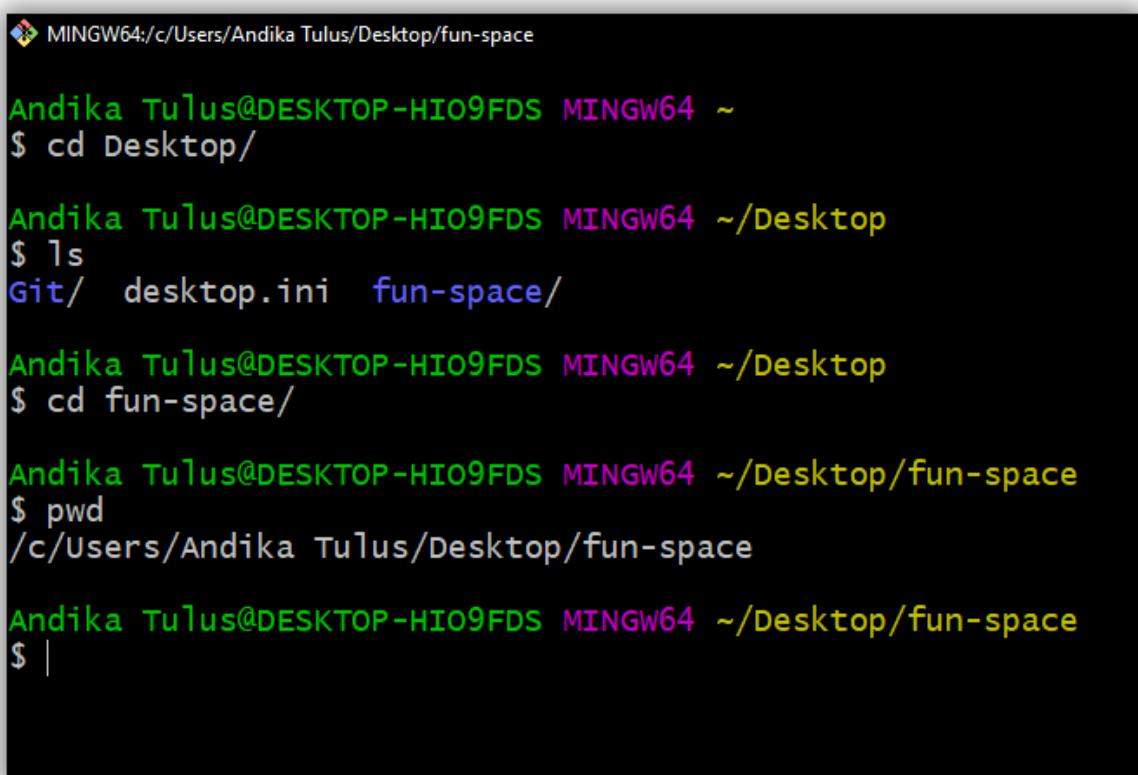
- Masih di Git Bash, kita harus **masuk ke Folder Project kita melalui Git Bash**. Lakukan dengan perintah berikut ini secara satu per satu.

Folder project yang kita buat tadi berada di Desktop.

***cd Desktop*** ( **cd** adalah suatu perintah masuk ke suatu folder )  
***ls*** ( **ls** adalah suatu perintah untuk melihat list folder atau file pada suatu folder )

***cd fun-space***

***pwd*** ( **pwd** adalah suatu perintah untuk melihat keberadaan kita di direktori atau folder saat ini )



```
MINGW64:/c/Users/Andika Tulus/Desktop/fun-space
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~
$ cd Desktop/
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop
$ ls
Git/ desktop.ini fun-space/
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop
$ cd fun-space/
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space
$ pwd
/c/Users/Andika Tulus/Desktop/fun-space
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space
$ |
```

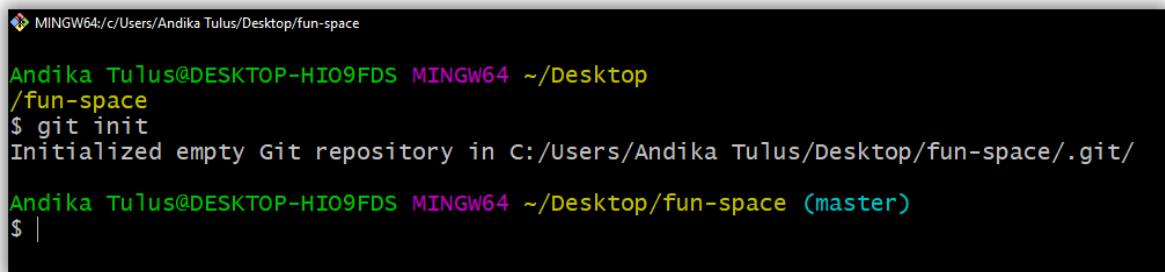
- Setelah masuk ke folder dengan git bash, selanjutnya kita telah membuat Repository secara Online di [github.com](https://github.com), maka dari itu kita juga harus membuat Repository secara Lokal atau Offline di komputer kita, agar nanti bisa dihubungkan ke Repository yang berada di Github. Untuk melakukannya maka

kita harus jalankan perintah berikut ini di git bash yang telah dibuka tadi.

*git init*

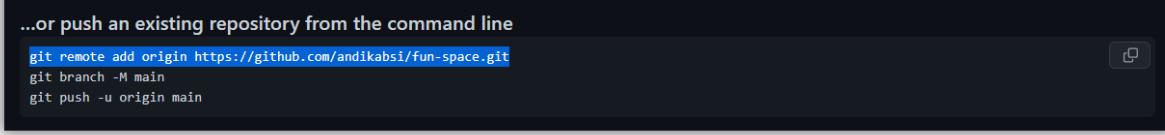
Perintah **git init** berfungsi untuk membuat Repository secara offline di komputer lokal atau para developer biasa menyebutnya sebagai “Initialisasi Repo Lokal”.

Jika berhasil, maka akan terdapat output seperti gambar dibawah ini.



```
MINGW64:/c/Users/Andika Tulus/Desktop/fun-space
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop
/fun-space
$ git init
Initialized empty Git repository in C:/Users/Andika Tulus/Desktop/fun-space/.git/
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (master)
$ |
```

- Selanjutnya, buka Browser yang telah dibuka tadi dan ketik perintah yang seperti ada di gambar ke Git Bash, lalu tekan ENTER.



```
...or push an existing repository from the command line
git remote add origin https://github.com/andikabsi/fun-space.git
git branch -M main
git push -u origin main
```



```
MINGW64:/c/Users/Andika Tulus/Desktop/fun-space
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (master)
$ git remote add origin https://github.com/andikabsi/fun-space.git
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (master)
$ |
```

Perintah ini :

```
git remote add origin https://github.com/andikabsi/fun-space.git
```

Kita mengintruksikan kepada git agar git menghubungkan (**remote**) dan menambahkan (**add**) repository berasal dari (**origin**) link Repository Github kita.

Kesimpulannya :

Kita mengintruksikan git untuk menambahkan akses repository yang berasal dari Github ke komputer lokal kita dengan menggunakan perintah **git remote add origin link-repo-github**

- Lalu, ketik perintah dibawahnya ke Git Bash, ENTER.

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

```
git remote add origin https://github.com/andikabsi/fun-space.git
git branch -M main
git push -u origin main
```

```
MINGW64:/c/Users/Andika Tulus/Desktop/fun-space
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (master)
$ git remote add origin https://github.com/andikabsi/fun-space.git
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (master)
$ git branch -M main
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$
```

Perintah ini :

```
git branch -M main
```

Kita menginstruksikan kepada git agar setelah terhubung ke repository Github, dan git memilih jalur cabang (**git branch**) utama ( **-M main** ) yang ada di repository github.

Kesimpulannya :

Kita menginstruksikan git, bahwa setelah terhubung ke repo github, untuk masuk ke dalam repository melalui jalur cabang utama yaitu cabang atau branch **main**

- Setelah itu, ketik perintah dibawahnya ke Git Bash, dan ENTER.

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

```
git remote add origin https://github.com/andikabsi/fun-space.git
git branch -M main
git push -u origin main
```

```
Andika_Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/andikabsi/fun-space.git'
```

Eror kan? Wkwk.. Dahulu kala, perintah tersebut bisa langsung di eksekusi, tapi akhirnya ada beberapa perubahan kebijakan yaitu kita tidak bisa langsung **push** begitu saja ke Repository Github. Hal tersebut karena kita belum menambahkan berkas project aplikasi kita ke repository lokal.

Jadi kan kita punya dua repository yaitu Lokal dan Online ( **Github** ).

Nah, kita belum mengunggah project kita ke Lokal maupun Online, kita hanya telah membuat Repository di Lokal dan Online, lalu menghubungkannya. Dan sekarang kita akan mengunggah ( **Add** ) project kita ke Lokal terlebih dahulu, lalu kemudian kita simpan riwayat perubahannya ( **Commit** ) dan terakhir kita akan mengunggah ( **Push** ) ke Repository online yang ada di Github.

- Seperti penjelasan diatas, maka sekarang kita akan mengunggah berkas project kita ke lokal dengan perintah **git add** .

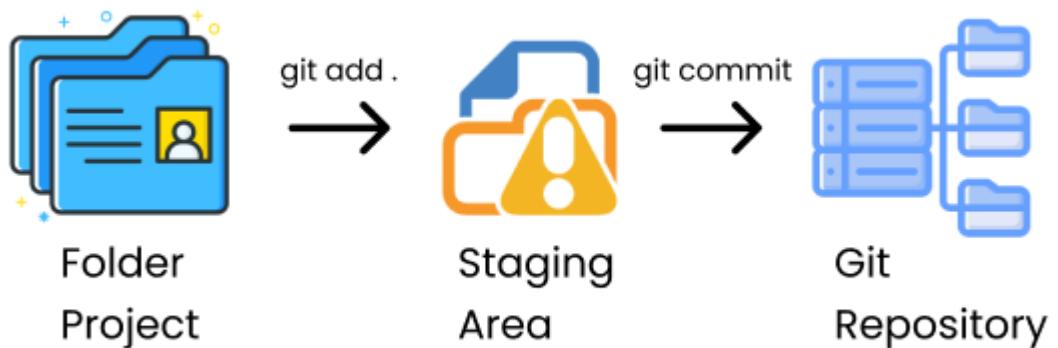
```
Andika Tulus@DESKTOP-HI09FDS MINGW64 ~/Desktop/fun-space (main)
$ git add .
```

Perintah ini :

*git add .*

Kesimpulan :

Perintah tersebut menginstruksikan git untuk menambahkan ( **git add** ) seluruh file ( . ) yang berada di folder project kita (fun-space) ke Staging Area. Staging Area adalah zona tak terlihat dimana seluruh file project bersiap untuk melalui tahap pengecekan terlebih dahulu riwayat perubahannya. Untuk pertama kali, jika kita menjalankan perintah diatas maka tidak akan mendapatkan output apapun. Jadi JANGAN PANIK 😊



- Selanjutnya, yang pasti posisi seluruh berkas project kita sekarang berada di staging area. Maka hal selanjutnya yang harus kita lakukan adalah menyimpan riwayat atau catatan perubahannya, sebelum kita unggah ke repository online. Untuk melakukan hal tersebut, kita akan menggunakan perintah **git commit -m “Judul Perubahan”**, maka akan muncul outputnya.

```
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$ git commit -m "First Github Project"
[main (root-commit) c92ee7c] First Github Project
 1 file changed, 15 insertions(+)
 create mode 100644 app.py
```

Perintah ini :

`git commit -m “judul perubahan”`

Kesimpulannya :

Kita menginstruksikan kepada git agar git menyimpan seluruh perubahan (**git commit**) yang terjadi pada berkas project kita, dan perubahan tersebut bisa kita berikan judul (**-m “judul-perubahan”**) perubahannya. Di kasus kita, karena kita baru

pertama kali dan belum melakukan perubahan apapun pada project kita, maka kita bisa menuliskan judul perubahan berupa “First Github Project”

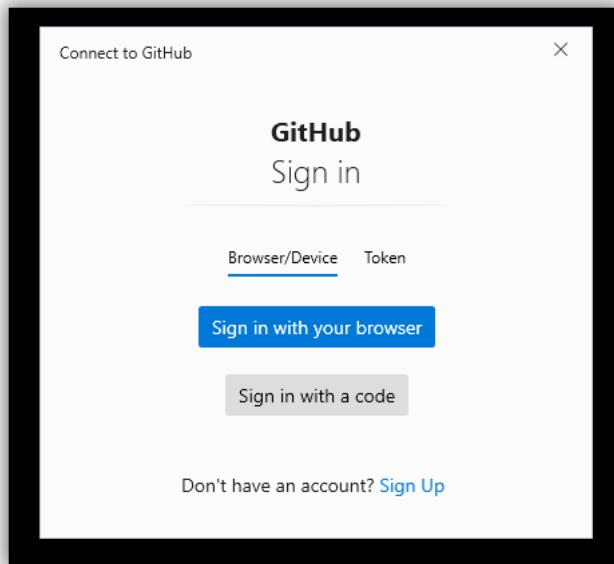
- Setelah menyimpan riwayat perubahannya, selanjutnya kita ke tahap akhir, yaitu mengunggah ( **Push** ) project kita dari Repository lokal ke Repository online ( Github )  
Kita akan menggunakan perintah **git push -u origin main**

Kode tersebut memiliki arti yaitu, kita mengintruksikan git untuk mengunggah ( **push** ) berkas project kita yang telah melalui beberapa tahap sebelumnya, untuk diunggah ke repository semula ( **-u origin** ) yaitu di Repository Github kita dan menaruhnya di branch atau cabang utama ( **main** ).

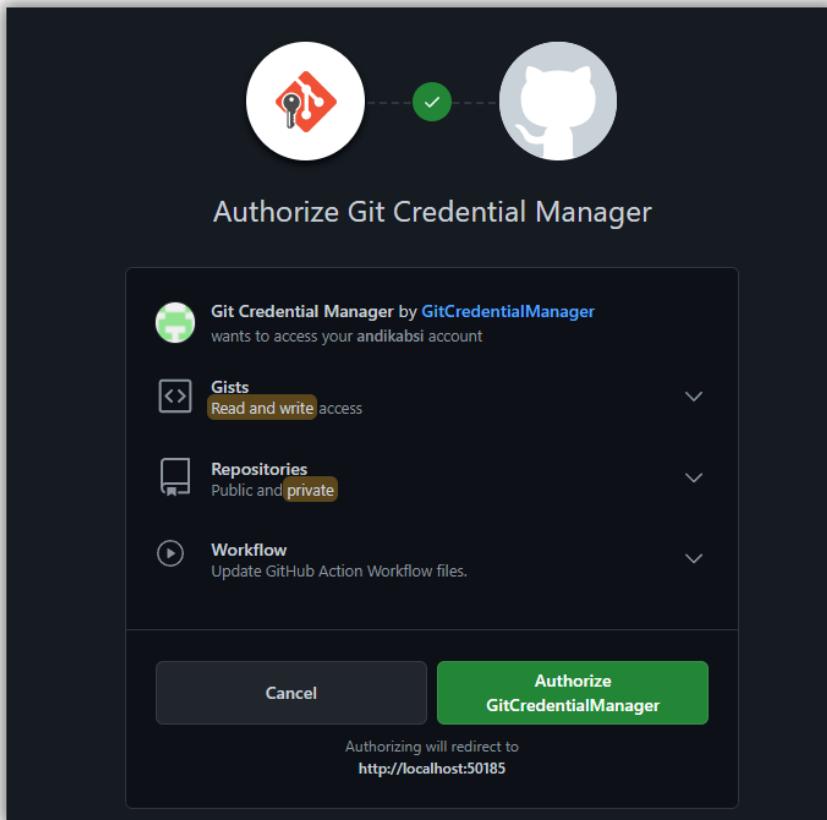
Sehingga semua berkas yang kita unggah, akan berada di penyimpanan awan github, dan dapat diakses semua orang, jika sebelumnya kita telah menyetel repository github kita ke **mode public**.

Coba jalankan perintahnya, jika ada pop-up yang mengintruksikan untuk masuk ke akun github, maka silahkan masukan username & password github kamu.

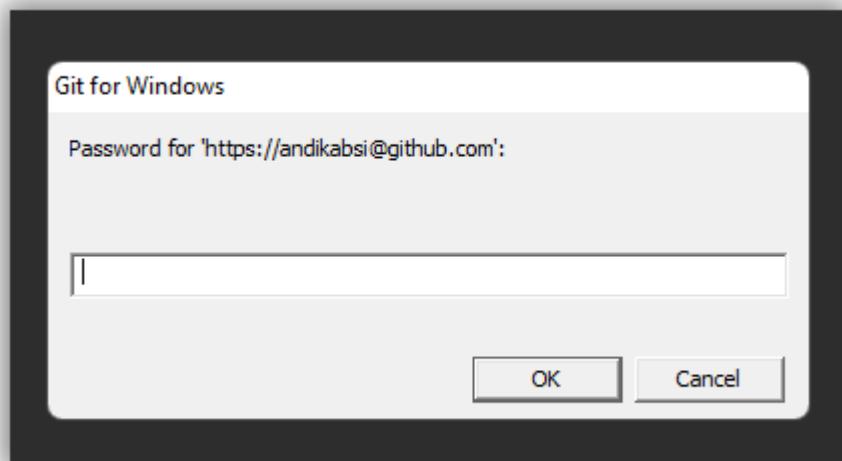
a. Pilih “**Sign in your browser**”



b. Lalu setujui akses Git ke Github



c. Masukan Username dan Password github kamu.



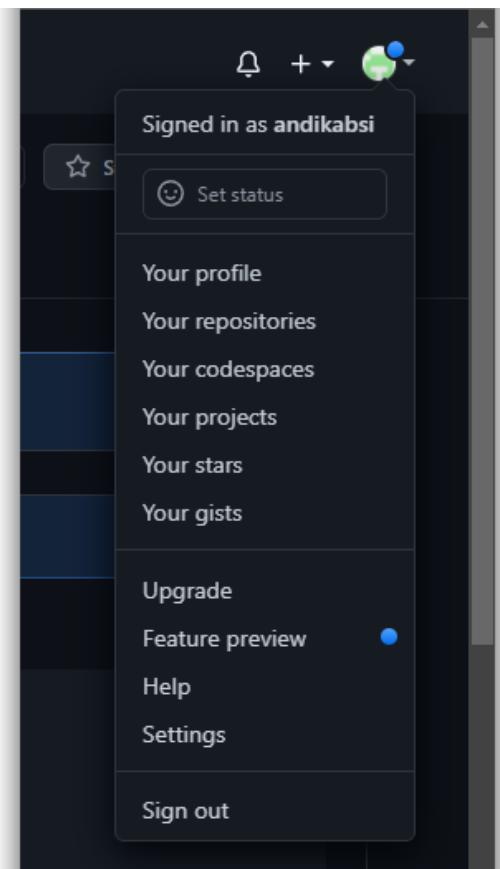
kemungkinan besar terdapat pesan kesalahan, jika tidak berarti anda beruntung.

```
Andika_Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$ git push -u origin main
fatal: An error occurred while sending the request.
fatal: The underlying connection was closed: An unexpected error occurred on a send.
fatal: Unable to read data from the transport connection: An existing connection was forcibly closed by the remote host.
fatal: An existing connection was forcibly closed by the remote host
remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requirements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/andikabsi/fun-space.git/'
```

Coba kita lihat erornya, disitu terlihat bahwa "... *Please use a personal access token instead*" yang berarti kita harus membuat Akses Token terlebih dahulu.

Apa itu github personal akses token? Didalam github, kita memiliki beberapa cara untuk masuk ke akun github, salah satunya adalah menggunakan Personal Akses Token. Dengan kita memiliki dan memberikan token tersebut ke suatu aplikasi, maka aplikasi yang kita gunakan akan diberikan hak akses untuk mengakses github kita. Di tahapan selanjutnya kita coba membuat Personal Akses Token.

- Buka “**Dashboard Github**” dan **Klik Icon/Foto Profil Github** kamu di pojok kanan atas, lalu **posisikan kursor di “Settings”, kemudian klik kanan dan “Open in new tab”**



- Kemudian pilih “**Developer Settings**” di navbar kiri

The screenshot shows the GitHub Profile settings page. On the left, there's a sidebar with various settings categories: Notifications, Scheduled reminders, SSH and GPG keys, Repositories, Packages, Pages, Organizations, Saved replies, Applications, Developer settings (which is selected), Moderation settings, Blocked users, Interaction limits, and Code review limits. The main content area has sections for URL, Twitter username, Company, and Location. A note says you can @mention your company's GitHub organization to link it. There's also a note about optional fields being deletable and a privacy statement link. A green 'Update profile' button is at the bottom of this section. Below it is a 'Contributions' section with a checkbox for including private contributions, which includes a note about getting credit for work on private repos. A 'Generate new token' button is located at the top right of the contributions section.

- Kemudian pilih “Personal access tokens” dan klik “Generate new token”

The screenshot shows the GitHub Developer settings page under the 'Personal access tokens' tab. It includes a note about generating a personal access token for quick API access, another note about token usage for Git over HTTPS or API authentication, and a prominent 'Generate new token' button.

- Isikan form seperti gambar di bawah ini.

The screenshot shows the GitHub 'New personal access token' creation form. It has a note field containing 'BelajarGit', a 'What's this token for?' field, and an 'Expiration' field set to '90 days' with a note that it will expire on Tue, Feb 15 2022.

## Catatan :

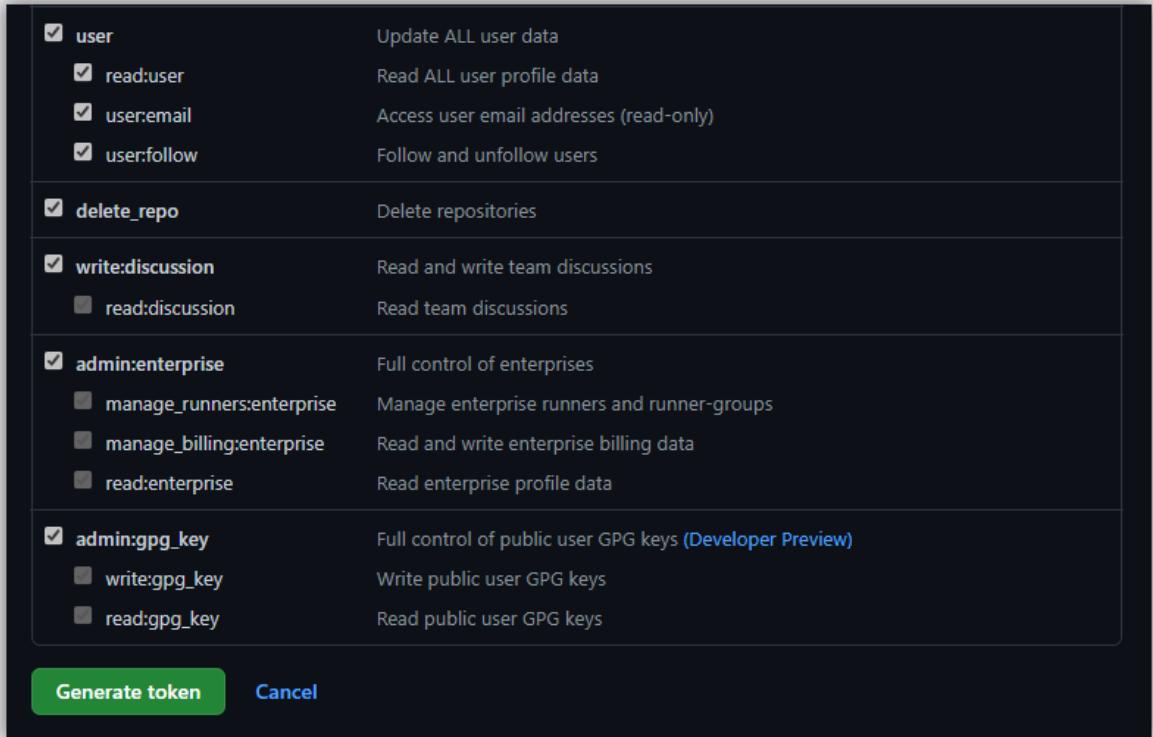
**Note** adalah nama atau catatan dari Token yang akan kita buat, karena kita membuat token untuk belajar Git, maka kita isi dengan kalimat “BelajarGit”

**Expiration** adalah durasi masa berlaku token yang kita buat, dalam hal ini kita akan membuat token yang hanya bisa kita gunakan hingga 90 hari kedepan, setelah lebih dari 90 hari maka kita harus memperbarui tokennya lagi.

- Kemudian centang semuanya dari atas sampai bawah pada “**Select Scopes**”, kemudian klik “Generate Token”

The screenshot shows a list of GitHub OAuth scopes. Each scope has a checkbox next to it, indicating whether it is selected or not. The scopes are grouped into sections: 'repo', 'workflow', 'write:packages', 'delete:packages', 'admin:org', and 'read:org'. Descriptions for each scope are provided to the right of the checkboxes.

Select scopes	
Scopes define the access for personal tokens. <a href="#">Read more about OAuth scopes.</a>	
<input type="checkbox"/> <b>repo</b>	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events
<input checked="" type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input checked="" type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry
<input checked="" type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects



- Copy tokennya dan mungkin bisa di simpan ke dalam notepad.

Settings / Developer settings

**Personal access tokens**

Generate new token    Revoke all

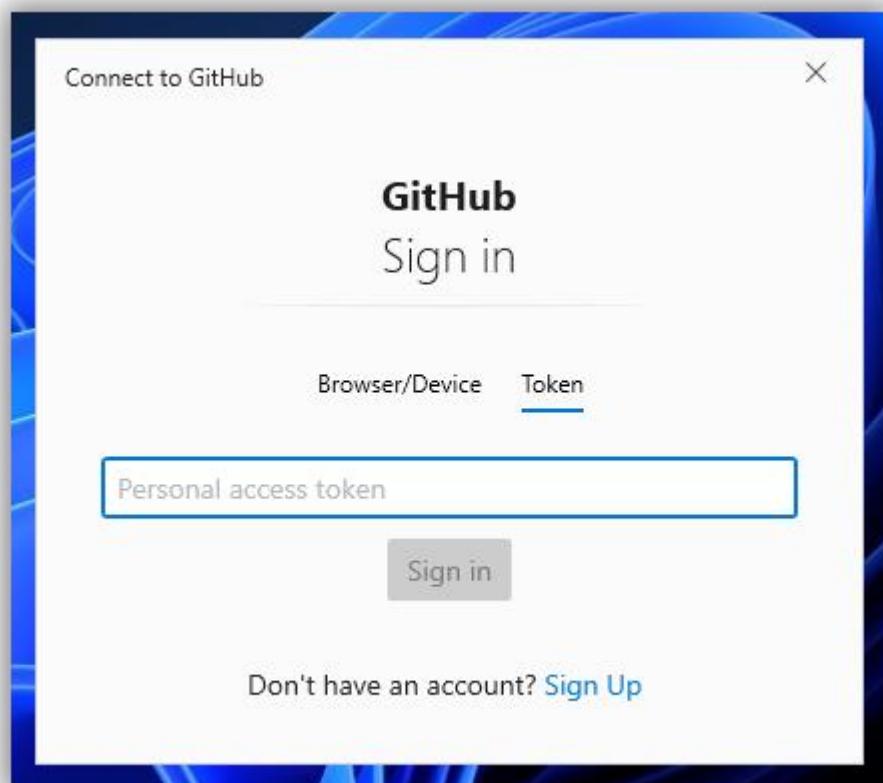
Tokens you have generated that can be used to access the GitHub API.

Make sure to copy your personal access token now. You won't be able to see it again!

ghp\_LqMPZo177SpaeqoJJpTWX2cR5CwJs43ktCVI

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

- Sekarang kita kembali ke Git Bash, dan jalankan kembali perintah **git push -u origin main**, jika terdapat pop-up seperti di bawah ini, maka pilih **Token** dan masukan Token yang telah di copy tadi, setelah itu tekan “**Sign In**”

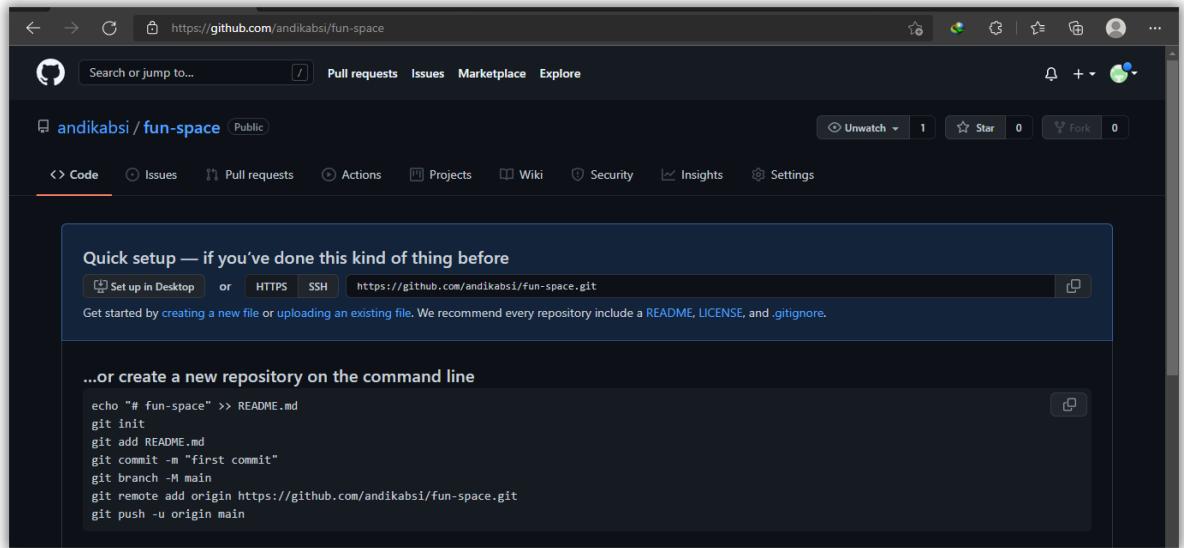


- Jika tampilannya kurang lebih seperti dibawah ini dan tidak ada error, maka kita telah berhasil mengunggah ( **push** ) berkas project aplikasi kita ke online di Github.

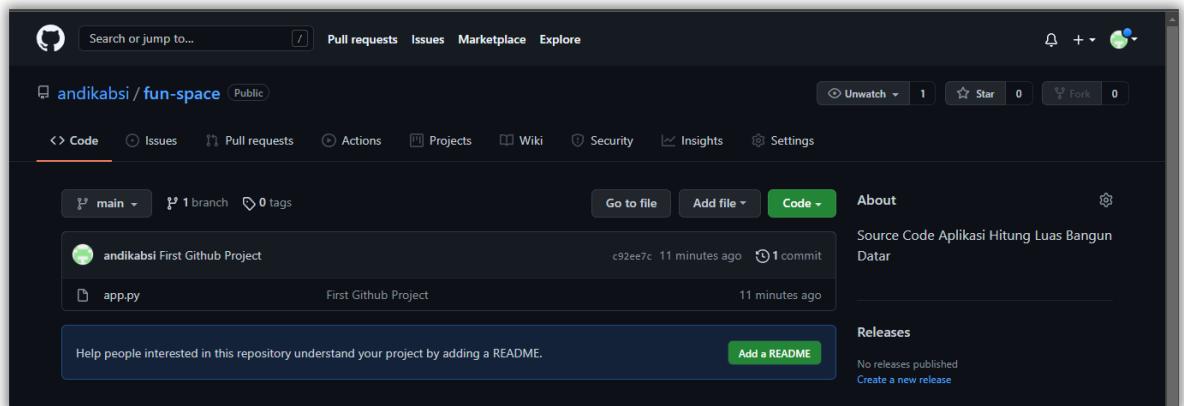
```
Andika_Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 353 bytes | 117.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andikabsi/fun-space.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

- Silahkan kembali ke browser, dan refresh halaman Repository Github kita, maka kita akan melihat perubahannya.

Sebelumnya :



Setelah melakukan Push :



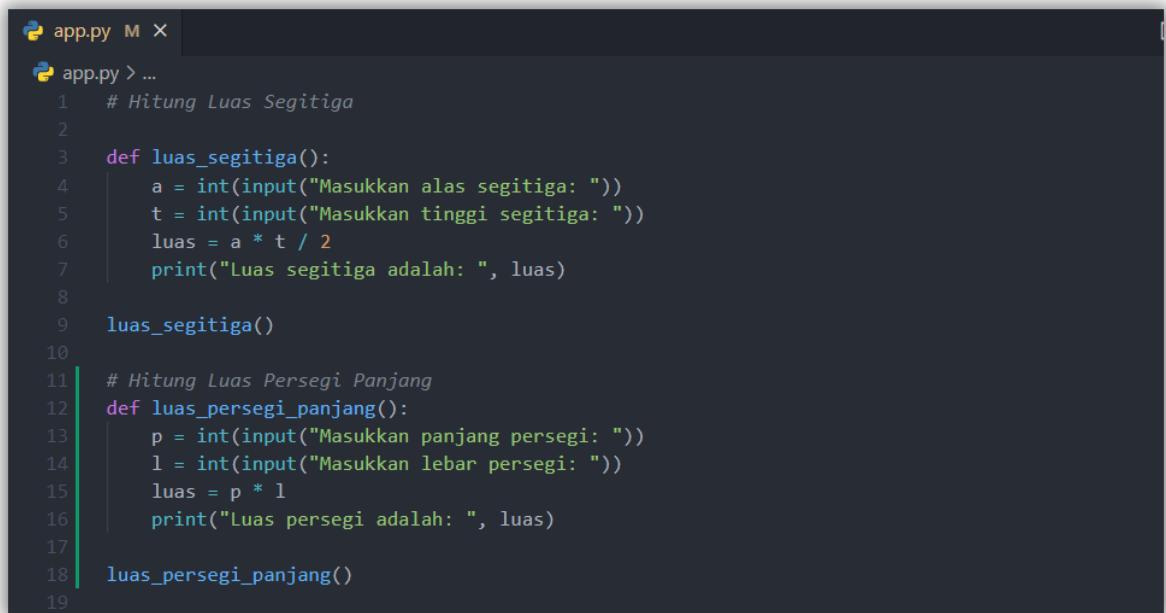
- Baik, kita sekarang telah menambahkan berkas project kita di Repository Github, kamu bisa share link repository ke seseorang agar bisa melihat kode project aplikasi kamu.

Contoh : <https://github.com/andikabsi/fun-space>

# Bekerja Dengan Git dan Github ( Melakukan Perubahan Project )

Pada suatu hari kita akan kembali mengerjakan project Aplikasi Menghitung Luas Bangun, yang kita buat sebelumnya dan kita akan menambahkan beberapa kode untuk menghitung beberapa bangun datar.

- Buka Project kita sebelumnya di Visual Studio Code
- Buka Repository github kita di Browser
- Buka Git Bash
- Pertama, kita tambahkan beberapa baris kode baru, disini kita akan membuat function untuk melakukan perhitungan luas persegi panjang.

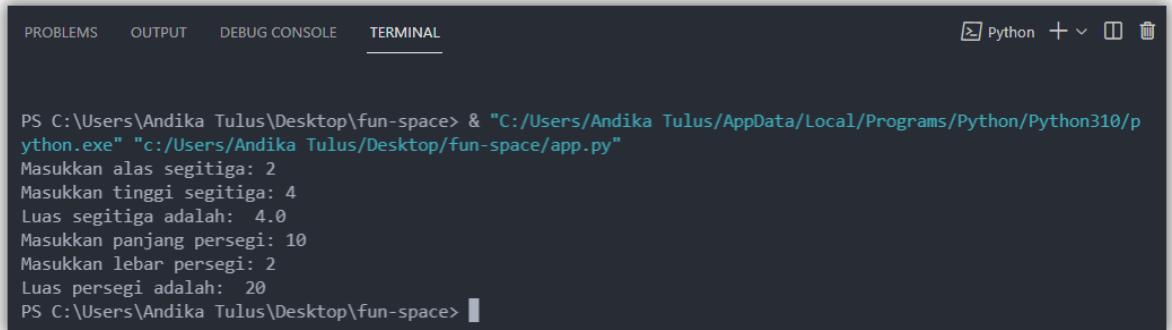


```
app.py M ×
app.py > ...
1 # Hitung Luas Segitiga
2
3 def luas_segitiga():
4     a = int(input("Masukkan alas segitiga: "))
5     t = int(input("Masukkan tinggi segitiga: "))
6     luas = a * t / 2
7     print("Luas segitiga adalah: ", luas)
8
9 luas_segitiga()
10
11 # Hitung Luas Persegi Panjang
12 def luas_persegi_panjang():
13     p = int(input("Masukkan panjang persegi: "))
14     l = int(input("Masukkan lebar persegi: "))
15     luas = p * l
16     print("Luas persegi adalah: ", luas)
17
18 luas_persegi_panjang()
19
```

- Di dalam visual studi code, jika kita telah menghubungkan project kita di Repository lokal dengan di Repository Online ( Github ) maka jika ada baris kode baru, akan telihat garis vertikal ( Kita disini menggunakan tema Visual Studio Code

bernama One Dark Pro ) jadi yang terihat adalah berwarna hijau.

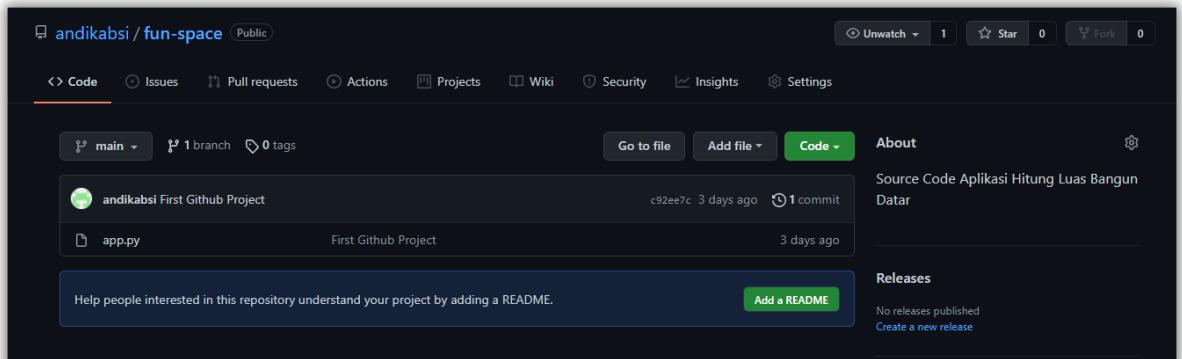
- Kita coba jalankan programnya yang telah kita ubah dan tambahkan kode baru, dan hasilnya berhasil tanpa eror.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python + □ └

PS C:\Users\Andika Tulus\Desktop\fun-space> & "C:/Users/Andika Tulus/AppData/Local/Programs/Python/Python310/python.exe" "c:/Users/Andika Tulus/Desktop/fun-space/app.py"
Masukkan alas segitiga: 2
Masukkan tinggi segitiga: 4
Luas segitiga adalah: 4.0
Masukkan panjang persegi: 10
Masukkan lebar persegi: 2
Luas persegi adalah: 20
PS C:\Users\Andika Tulus\Desktop\fun-space>
```

- Sekarang kita buka repo github kita di browser, dan lihat belum ada perubahan apapun, dan kode aplikasi nya masih sama seperti sebelumnya. Karena kita memang belum **commit** dan **push** kode serta perubahan terbaru dari project kita.



- Sekarang kita akan mencoba untuk melakukan commit dan push perubahannya ke github.
- Buka Git Bash ( Jika kamu telah keluar dari folder project di git bash, silahkan masuk lagi )
- Setelah itu, jalankan perintah **git add .**. Seperti yang telah kita ketahui, bahwa perintah tersebut untuk menambahkan berkas project kita ke staging area untuk bersiap di cek dan dicatat riwayat perubahannya.

```
MINGW64:/c/Users/Andika Tulus/Desktop/fun-space  
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)  
$ git add .  
  
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)  
$
```

- Selanjutnya kita jalankan perintah **git commit -m “Judul Perubahannya”** , karena kita membuat perubahan dengan menambahkan function hitung luas persegi panjang, maka kita beri judul commit “Menambah Hitung Luas Persegi Panjang”

```
MINGW64:/c/Users/Andika Tulus/Desktop/fun-space  
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)  
$ git add .  
  
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)  
$ git commit -m "Menambah Hitung Luas Persegi Panjang"  
[main fb63eb6] Menambah Hitung Luas Persegi Panjang  
 2 files changed, 12 insertions(+), 1 deletion(-)  
  create mode 100644 .vscode/settings.json
```

Fokus pada **git commit**, sekarang kita baca arti dari output tersebut, yaitu kita mencatat perubahan berkas dengan judul perubahan “Menambah Hitung Luas Persegi Panjang”. Lalu git mencatat bahwa terdapat 2 berkas diubah, lalu terdapat 12 baris kode yang ditambahkan, dan 1 baris kode dihapus.

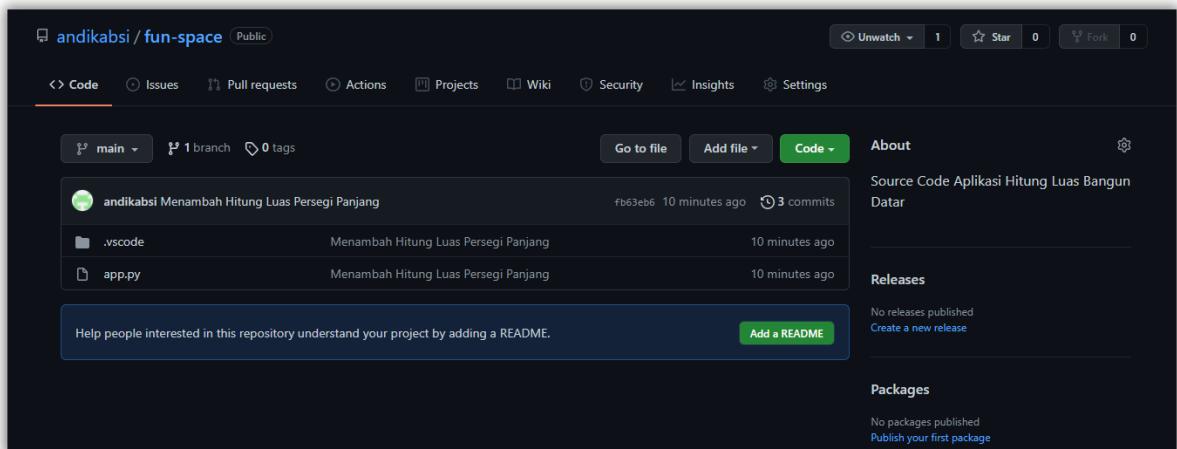
- Setelah kita mencatat perubahannya dengan **git commit**, maka sekarang kita hanya perlu mengunggahnya ke github dengan perintah **git push**

```

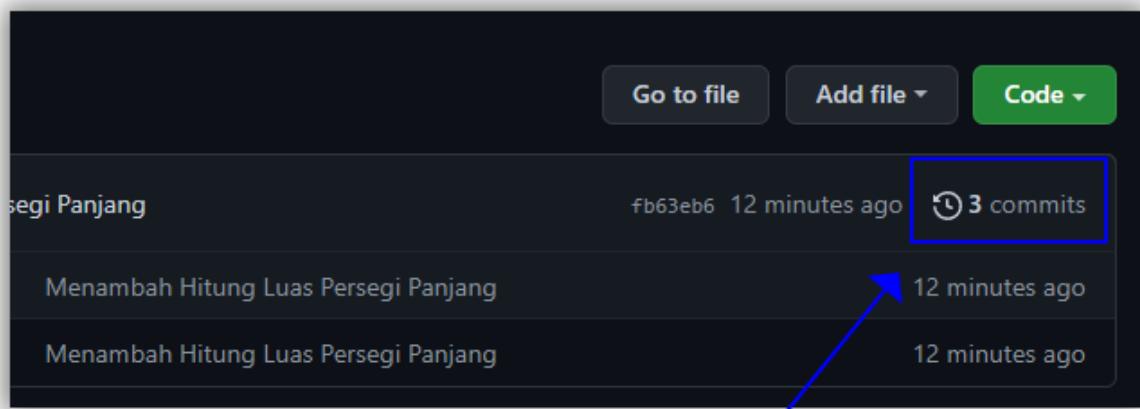
Andika_Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (8/8), 824 bytes | 164.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/andikabsi/fun-space.git
  c92ee7c..fb63eb6  main -> main

```

- Sekarang kita lihat dan refresh repo kita di github, jika berhasil pasti akan berubah, mulai dari Catatan Perubahan Terakhir hingga kodennya.



- Untuk melihat riwayat catatan perubahan pada kode kita, kita bisa tekan **simbol history** dan kita bisa lihat judul perubahan dari awal hingga perubahan terakhir.



The screenshot shows the detailed view of the first commit 'Menambah Hitung Luas Persegi Panjang' on the 'main' branch. It was committed by 'andikabsi' 16 minutes ago. The commit message is 'Menambah Hitung Luas Persegi Panjang'. Below the commit message, there are two more commits listed under 'Commits on Nov 17, 2021':

- Merapihkan Kode** (andikabsi committed 2 days ago)
- First Github Project** (andikabsi committed 3 days ago)

At the bottom are 'Newer' and 'Older' buttons.

- Tekan pada judul perubahan untuk melihat catatan perubahan secara detail, dimana kita bisa lihat baris yang ditambahkan atau dihapus pada source code aplikasi kita.

The screenshot shows the detailed view of the first commit 'Menambah Hitung Luas Persegi Panjang' on the 'main' branch. It was committed by 'andikabsi' 19 minutes ago. The commit message is 'Menambah Hitung Luas Persegi Panjang'. The commit has 1 parent, commit 237ce51, and 12 additions and 1 deletion. The diff shows changes in the '.vscode/settings.json' file:

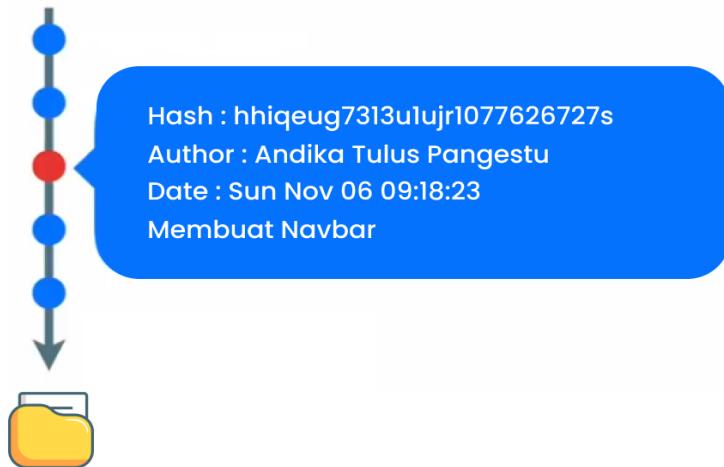
```

1 + {
2 +   "python.formatting.provider": "yapf"
3 + } ⊖

```

```
... @@ -1,4 +1,4 @@
1 - # Hitung Luas Segitiga sederhana
2
3 def luas_segitiga():
4     a = int(input("Masukkan alas segitiga: "))
5
6 @@ -8,6 +8,14 @@
7
8     luas_segitiga()
9
10
11 + # Hitung Luas Persegi Panjang
12 + def luas_persegi_panjang():
13 +     p = int(input("Masukkan panjang persegi: "))
14 +     l = int(input("Masukkan lebar persegi: "))
15 +     luas = p * l
16 +     print("Luas persegi adalah: ", luas)
17 +
18 + luas_persegi_panjang()
19
20
21
```

- Namun, setelah beberapa menit kita ingin membatalkan perubahan, dan kita tidak ingin menambahkan “Hitung Luas Persegi Panjang”, namun kita ingin menambahkan “Hitung Luas Persegi” dahulu.
- Untuk membatalkan perubahan kita akan menggunakan perintah **git checkout**, namun sebelum itu kita harus mengetahui **nomor id commit**. Untuk melihatnya kita bisa menggunakan perintah **git log**, setelah kita melakukan commit sebelumnya.
- Perintah, git log menghasilkan output berupa identitas setiap perubahan yang telah dilakukan. Git hanya akan menyimpan perubahannya saja, dia tidak akan menyimpan seluruh isi file yang akan banyak memakan memori. Dengan Git kita juga bisa kembali ke versi revisi yang kita inginkan.



Detail isi pada sebuah **commit** :

- ID Commit atau Hash
- Author ( Orang yang melakukan perubahan )
- Date ( Tanggal dan waktu file diubah )
- Catatan Perubahan

Ketika kita menjalankan perintah **git log** pada project kita, maka akan mendapatkan output berikut. Terlihat kita telah melakukan 3 perubahan, dimulai dari “First Github Project”.

```
Andika Tulus@DESKTOP-HIO9FDS MINGW64 ~/Desktop/fun-space (main)
$ git log
commit fb63eb6dd849162a8f43a4fc2e1915f6f9977481 (HEAD -> main, origin/main)
Author: andikabsi <andikatulusp@outlook.com>
Date:   Fri Nov 19 21:01:29 2021 +0700

        Menambah Hitung Luas Persegi Panjang

commit 237ce51f8f5f43433e1e20f1765e907bd5d1aaae
Author: andikabsi <andikatulusp@outlook.com>
Date:   Wed Nov 17 10:47:44 2021 +0700

        Merapihkan Kode

commit c92ee7cd41429a4859c6a127b5cd6d6f79528495
Author: andikabsi <andikatulusp@outlook.com>
Date:   Wed Nov 17 09:16:14 2021 +0700

        First Github Project
```

- Karena kita ingin membatalkan perubahan terakhir dari seluruh berkas project, maka kita copy id atau hash dari commit sebelum commit yang terakhir yaitu `237ce51f8f5f43433e1e20f1765e907bd5d1aaae` dengan cara blok id-nya lalu tekan CTRL + SHIFT + C
- Kita akan mengembalikan kondisi file app.py, seperti pada *commit* sebelumnya. Maka kita bisa menggunakan perintah:

```
git checkout 237ce51f8f5f43433e1e20f1765e907bd5d1aaae
```

Maka akan mendapatkan output seperti ini :

```
PS C:\Users\Andika Tulus\Desktop\fun-space> git checkout 237ce51f8f5f43433e1e20f1765e907bd5d1aaae
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

    2bed959 Menambah Hitung Luas Persegi Panjang

If you want to keep it by creating a new branch, this may be a good time
to do so with:

    git branch <new-branch-name> 2bed959

HEAD is now at 237ce51 Merapihkan Kode
```

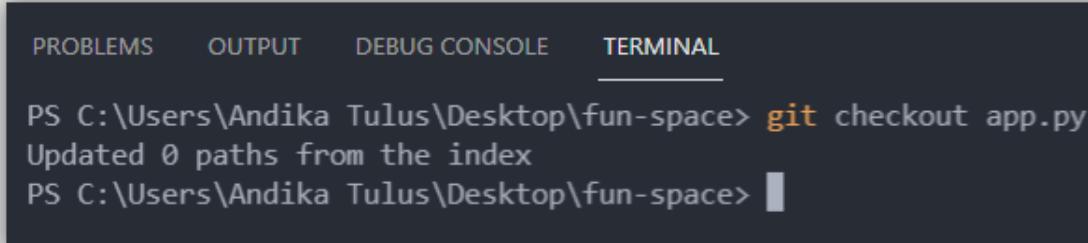
- Seperti mesin waktu, kita sudah mengembalikan keadaan seperti keadaan saat *commit* tersebut. Namun, saat ini kondisi berkas masih dalam posisi *staged* di staging area. Kita bisa kembalikan ke dalam kondisi *modified* dengan perintah `git reset` jika dijalankan, maka tidak akan terdapat output apapun.
- Sekarang kita lihat di visual studio code, dan kode berubah seperti sebelumnya.

```
py app.py > ...
1 # Hitung Luas Segitiga sederhana
2
3 def luas_segitiga():
4     a = int(input("Masukkan alas segitiga: "))
5     t = int(input("Masukkan tinggi segitiga: "))
6     luas = a * t / 2
7     print("Luas segitiga adalah: ", luas)
8
9 luas_segitiga()
10
```

## Catatan :

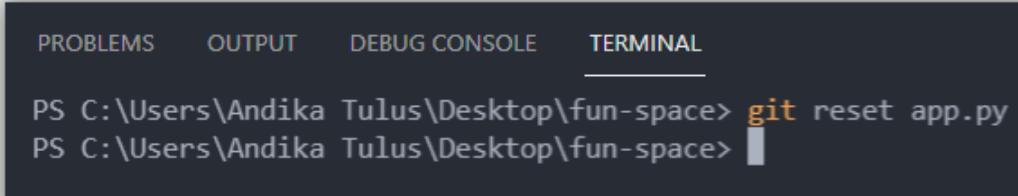
Terdapat beberapa hal yang harus diperhatikan dalam mebatalkan sebuah revisi dengan git, yaitu sebagai berikut.

- Jika revisi kita belum dalam keadaan staged atau pun commit, kita bisa mengembalikan setiap perubahan pada berkas dengan perintah **git checkout namaFile**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Andika Tulus\Desktop\fun-space> git checkout app.py
Updated 0 paths from the index
PS C:\Users\Andika Tulus\Desktop\fun-space>
```

- Jika revisi kita telah masuk ke dalam keadaan staged namun belum melakukan **git commit**, maka untuk membatalkan perubahannya kita menggunakan perintah **git reset namaFile**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Andika Tulus\Desktop\fun-space> git reset app.py
PS C:\Users\Andika Tulus\Desktop\fun-space>
```

- c. Jika revisi kita telah dilakukan commit, dan kita ingin membatalkan revisi tersebut, maka hal pertama yang kita lakukan adalah mengetahui hash commit sebelum dilakukan revisi yang terakhir, untuk melihatnya bisa menggunakan perintah **git log** dan untuk mengembalikan revisi kita bisa menggunakan perintah **git checkout xxx** ( xxx adalah nomor hash commit)

```
PS C:\Users\Andika Tulus\Desktop\fun-space> git checkout 237ce51f8f5f43433e1e20f1765e907bd5d1aaae
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

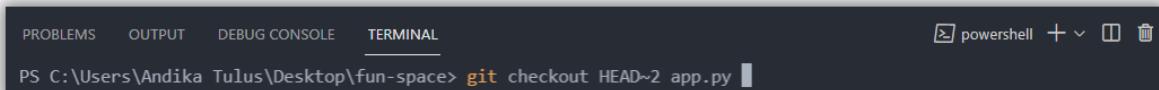
    2bed959 Menambah Hitung Luas Persegi Panjang

If you want to keep it by creating a new branch, this may be a good time
to do so with:

    git branch <new-branch-name> 2bed959

HEAD is now at 237ce51 Merapihkan Kode
```

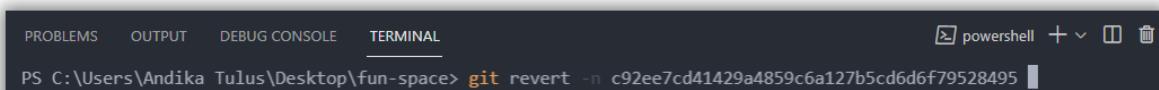
- d. Jika kita ingin mengembalikan perubahan ke commit sebelumnya namun secara spesifik atau terpilih, maka kita bisa menggunakan perintah **git checkout HEAD~2 namaFile** ( Peintah tersebut, kita akan mengembalikan perubahan ke 2 commit sebelumnya )



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Andika Tulus\Desktop\fun-space> git checkout HEAD~2 app.py

- e. Dan terakhir, untuk membatalkan semua perubahan ke suatu commit kita bisa menggunakan perintah **git revert -n nomorHash**



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Andika Tulus\Desktop\fun-space> git revert -n c92ee7cd41429a4859c6a127b5cd6d6f79528495

# Bekerja Dengan Git dan Github ( Kolaborasi Project )

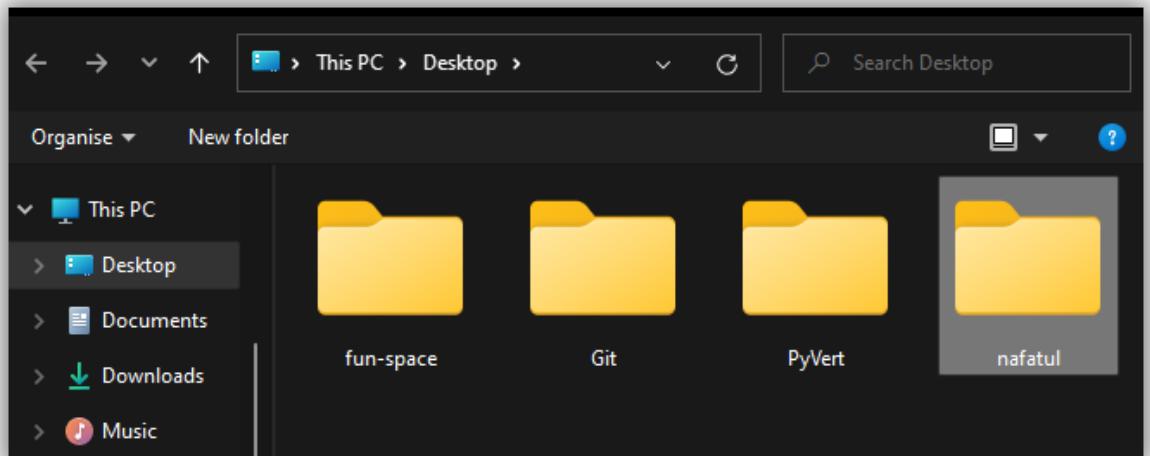
Saya adalah adhistya, saya ingin berkontribusi ke Project yang dibuat andika, yaitu project “Hitung Luas Bangun Datar”, maka dari itu saya akan menggunakan git untuk melakukan kolaborasi dengan andika dalam mengembangkan aplikasi tersebut.

Beberapa hal yang harus kita lakukan dalam berkolaborasi dalam project dengan git, yang pasti dan perlu diketahui bahwa repository yang akan kita kontribusi adalah repository public, jika bersifat privat, maka akan ada pengaturan khusus.

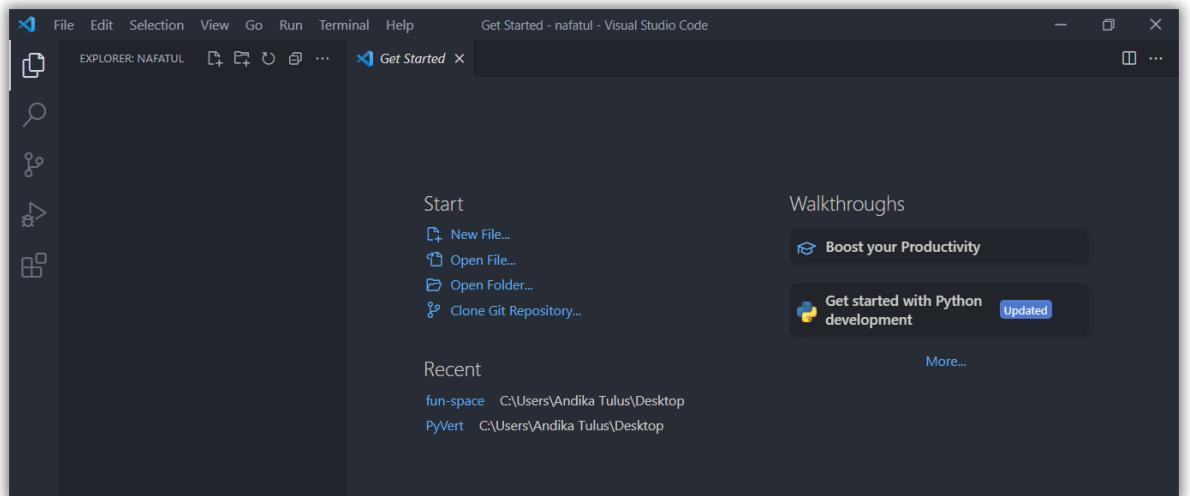
- Buka git bash, dan kita akan mengubah konfigurasi git yang ada di git bash, mengapa kita rubah? Karena saya menggunakan komputer milik andika, jadi saya akan merubah konfigurasinya ke akun saya. Jika kalian menggunakan laptop yang berbeda kamu mungkin perlu maupun tidak sama sekali melakukan hal ini.
- Pertama, saya akan melihat list config atau detail config yang telah dikonfigurasi sebelumnya dengan perintah :  
**git config –list**

```
C:\Users\Andika Tulus>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/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.http://dev.azure.com.usehttppath=true
init.defaultbranch=master
core.editor="C:\Users\Andika Tulus\AppData\Local\Programs\Microsoft VS Code\bin\code.cmd" --wait
user.name=andikabsi
user.email=andikatulusp@outlook.com
```

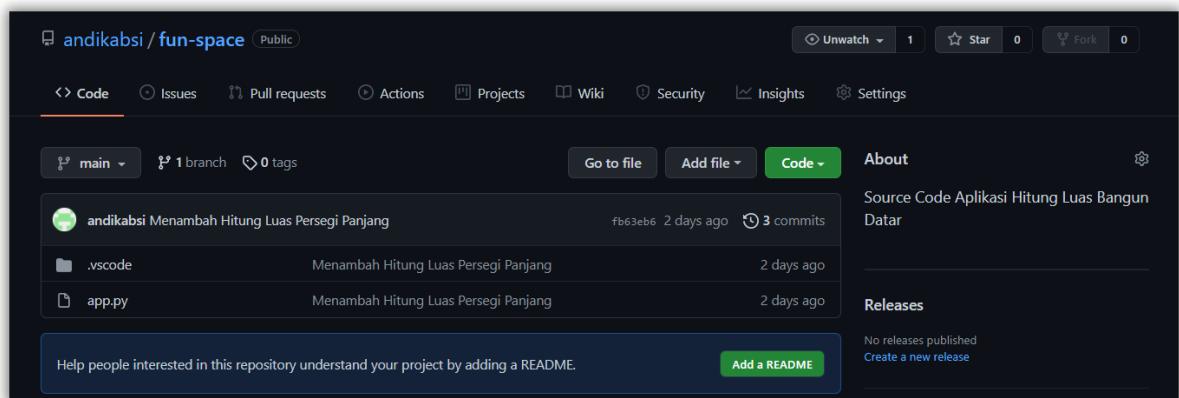
- Kedua, saya akan mengkonfigurasi config git untuk diri saya, dan perlu diketahui bahwa di git terdapat dua lingkup config yaitu config untuk global, dan config untuk repository saat ini. Saya akan menggunakan keduanya.
- Ketiga, saya akan mengkonfigurasi config global terlebih dahulu sesuai akun github saya. Setelah itu cek dengan perintah **git config -list**, jika config telah berubah maka kita berhasil melakukan konfigurasi.
- Selanjutnya kita konfigurasi config untuk repo yang akan kita gunakan untuk berkontribusi ke project. Namun sebelumnya saya harus membuat folder baru untuk project bagian saya.



- Kita buka folder tersebut dengan Visual Studio Code
- Bisa kita lihat, bahwa di dalam folder saya masih kosong.

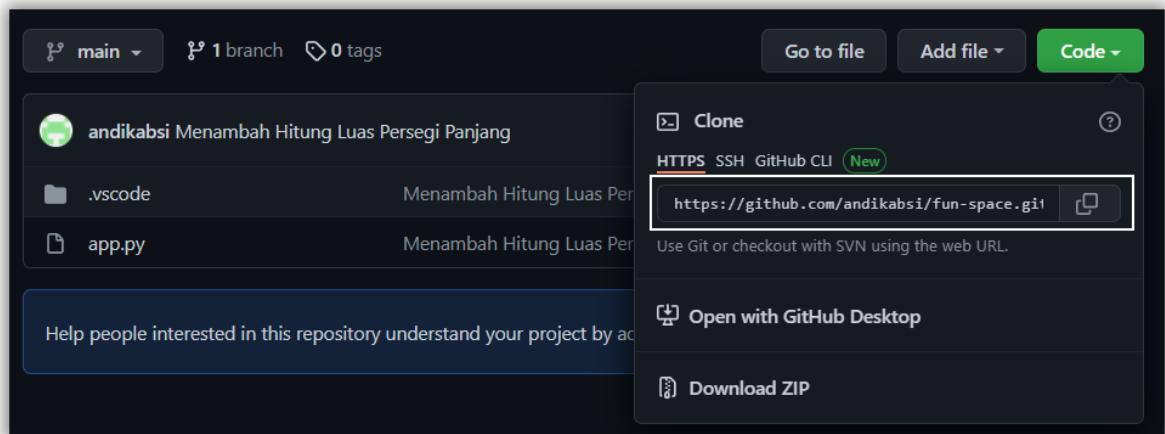


- Selanjutnya kita tidak perlu membuat atau menyalin project “Menghitung Luas Bangun Datar” melalui komputer atau dengan transfer manual. Tapi kita akan menggunakan perintah git yang disebut **git clone**  
 Seperti namanya, kita akan mencloning atau menggandakan repository milik orang lain dan kita akan rekontribusi didalamnya.
- Untuk itu, maka kita perlu membuka tautan repository milik andika di github.

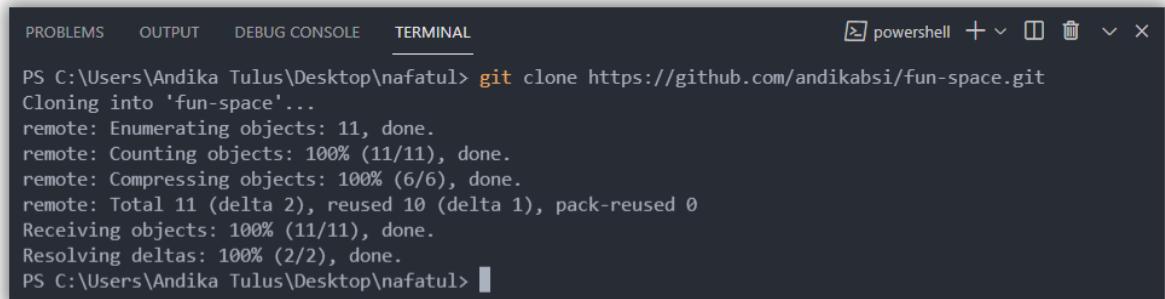


- Klik tombol **Code** yang berwarna hijau dan salin text tersebut, lalu tempel atau paste di terminal vscode.

*git clone link-repository-github*



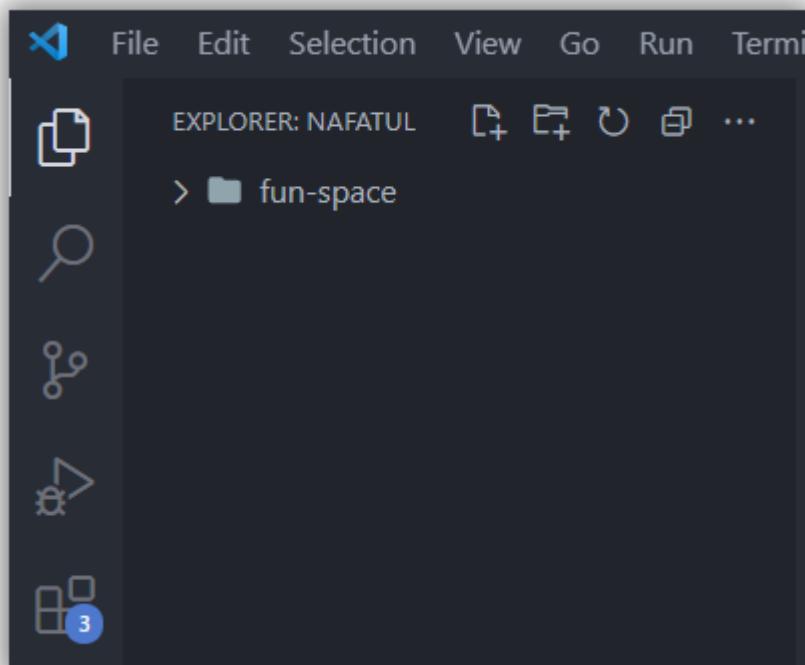
- Jika berhasil maka akan tampil seperti ini.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell + × × ×

PS C:\Users\Andika Tulus\Desktop\nafatul> git clone https://github.com/andikabsi/fun-space.git
Cloning into 'fun-space'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 11 (delta 2), reused 10 (delta 1), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
PS C:\Users\Andika Tulus\Desktop\nafatul>
```

- Akan muncul folder baru, itu adalah folder yang berhasil kita ambil dari repository milik andika dan sekarang folder project sudah ada di folder milik saya, maka dari itu kita tinggal mengkonfigurasi config sesuai repository.



- Sekarang kita buka terminal di VSCode, dan konfigurasi dengan config akun github kita namun tidak menggunakan perintah config global.

```
git config user.name "Adhistya2305"  
git config user.email "email@gmail.com"
```

Sesuaikan dengan username dan email yang terdaftar untuk akun github kamu.

A screenshot of a terminal window titled "powershell". The window shows the following command history:

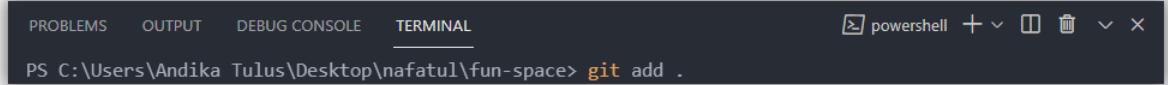
```
PS C:\Users\Andika Tulus\Desktop\nafatul> cd .\fun-space\  
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git config user.name "Adhistya2305"  
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git config user.emil "adhistya60@gmail.com"  
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space>
```

The "TERMINAL" tab is highlighted at the top.

- Sekarang mari kita membuat perubahan di kode project milik Andika, kita akan membuat Function baru, yaitu function untuk menghitung luas lingkaran.

```
app.py    X
fun-space > app.py > ...
11  # Hitung Luas Persegi Panjang
12  def luas_persegi_panjang():
13      p = int(input("Masukkan panjang persegi: "))
14      l = int(input("Masukkan lebar persegi: "))
15      luas = p * l
16      print("Luas persegi adalah: ", luas)
17
18  luas_persegi_panjang()
19
20 # Hitung Luas Lingkaran
21 def luas_lingkaran():
22     r = int(input("Masukkan jari-jari lingkaran: "))
23     luas = 3.14 * r * r
24     print("Luas lingkaran adalah: ", luas)
25
26 luas_lingkaran()
27
```

- Setelah kita jalankan dan tidak ada eror, atau kode kita telah dalam kondisi siap untuk di push maka kita bisa melakukan beberapa perintah berikut.
  - Kita tambahkan ke staging are dengan perintah **git add** .



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git add .
```

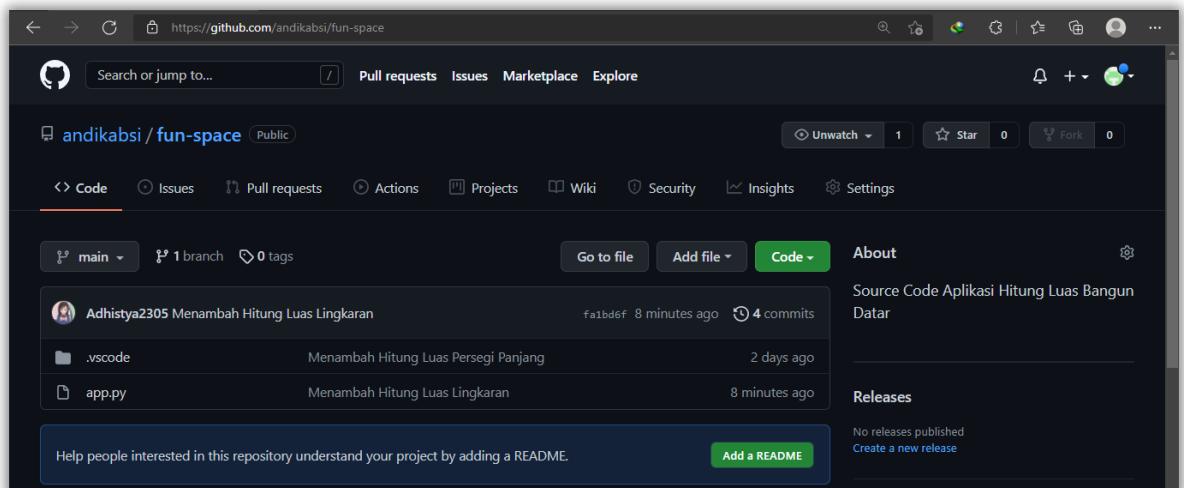
- Lalu, setelah itu kita commit dengan perintah **git commit -m "Menambahkan Hitung Luas Lingkaran"**

```
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git commit -m "Menambahkan Hitung Luas Lingkaran"
[main fa1bd6f] Menambah Hitung Luas Lingkaran
 1 file changed, 8 insertions(+)
```

- Dan terakhir kita push ke repository milik andika langsung dengan perintah **git push origin**

```
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git push origin
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 422 bytes | 84.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/andikabsi/fun-space.git
 fb63eb6..fa1bd6f main -> main
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space>
```

- Sekarang bisa kita lihat bahwa saya telah berkontribusi dan berkolaborasi dengan andika. Saya telah menambahkan "Hitung Luas Lingkaran" di projectnya.



- Dan saya ternyata ada ide baru, saya ingin menambahkan kodingan untuk **Menghitung Pythagoras**, namun saya masih ragu apakah itu akan diperlukan atau tidak oleh andika.
- Maka dari itu saya akan membuat cabang atau dalam git sering kita sebut dengan **branch**  
**Branch berfungsi untuk membuat cabang dari kode sumber utama di dalam repository ( Kita mungkin bisa menggambarkannya seperti membuat folder dalam folder ). Branch merupakan salah satu kerangka kerja utama dari sistem git.**

- Untuk membuat branch kita menggunakan perintah **git checkout -b "namaBranch"**  
Karena kita akan membuat branch untuk fitur pythagoras, maka kita akan melakukan pembuatan branch dengan perintah berikut.

```
git checkout -b fitur_pythagoras
```

Artinya : kita untuk mengintruksikan kepada git agar membuat cabang baru dari cabang utama di repository dengan nama **fitur\_pythagoras**

- Setelah di jalankan perintah tersebut, maka kita telah berganti ke jalur cabang **fitur\_pythagoras**. Jadi jika kita melakukan perubahan apapun dan akan kita push, maka semuanya akan masuk ke cabang **fitur\_pythagoras**, bukan cabang atau branch utama yaitu cabang **main**.

```
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git checkout -b fitur_pythagoras
Switched to a new branch 'fitur_pythagoras'
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space>
```

- Lalu kita akan membuat file baru bernama *pythagoras.py* dan ketikan kode hitung pythagoras.

```

EXPLORER: NAFATUL   ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ...
fun-space > pythagoras.py > ...
1 # Menghitung Teorema Pythagoras
2 def main():
3     a = float(input("Masukkan nilai a: "))
4     b = float(input("Masukkan nilai b: "))
5     c = (a**2 + b**2)**0.5
6     print("Nilai c adalah: ", c)
7
8 main()
9
10

```

- Setelah itu save dan kita akan lakukan **git add .** agar semua berkas masuk ke staging area.

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
powershell + ▾
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git add .

```

- Lalu, kita commit dengan pesan commit “Uji Coba Hitung Pythagoras”

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
powershell + ▾
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git add .
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git commit -m "Uji Coba Hitung Pythagoras"
[fitur_pythagoras bf9a39f] Uji Coba Hitung Pythagoras
 2 files changed, 9 insertions(+), 2 deletions(-)
 create mode 100644 pythagoras.py

```

- Kita tidak bisa langsung melakukan git push seperti sebelumnya, karena sekarang kita berada bukan di branch utama. Untuk itu kita harus melakukan *remote* dahulu dengan branch yang telah kita buat.

Kita akan menggunakan perintah :  
*git push --set-upstream origin fitur\_pythagoras*

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git push
fatal: The current branch fitur_pythagoras has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin fitur_pythagoras

PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git push --set-upstream origin fitur_pythagoras
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 495 bytes | 70.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'fitur_pythagoras' on GitHub by visiting:
remote:     https://github.com/andikabsi/fun-space/pull/new/fitur_pythagoras
remote:
To https://github.com/andikabsi/fun-space.git
 * [new branch]      fitur_pythagoras -> fitur_pythagoras
Branch 'fitur_pythagoras' set up to track remote branch 'fitur_pythagoras' from 'origin'.
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space>
```

- Sekarang kita lihat di Repository github bahwa Branch yang saya buat telah ditambahkan ke repository milik andika.

The screenshot shows the GitHub repository page for 'andikabsi / fun-space'. The repository has 1 star, 0 forks, and 0 issues. It contains 2 branches and 0 tags. The main branch has 4 commits. A recent push from 'fitur\_pythagoras' was made 1 minute ago. There are buttons to 'Compare & pull request', 'Go to file', 'Add file', and 'Code'. A message at the bottom encourages adding a README. On the right, there are sections for 'About' (Source Code Aplikasi Hitung Luas Bangun Datar) and 'Releases' (No releases published, Create a new release). A 'Packages' section is also present.

andikabsi / fun-space Public

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

fitur\_pythagoras had recent pushes 1 minute ago

Compare & pull request

main 2 branches 0 tags Go to file Add file Code

Adhistya2305 Menambah Hitung Luas Lingkaran fa1bd6f 1 hour ago 4 commits

.vscode Menambah Hitung Luas Persegi Panjang 2 days ago

app.py Menambah Hitung Luas Lingkaran 1 hour ago

Help people interested in this repository understand your project by adding a README. Add a README

About

Source Code Aplikasi Hitung Luas Bangun Datar

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

- Dan terakhir kita akan kembali ke branch utama yaitu branch main.

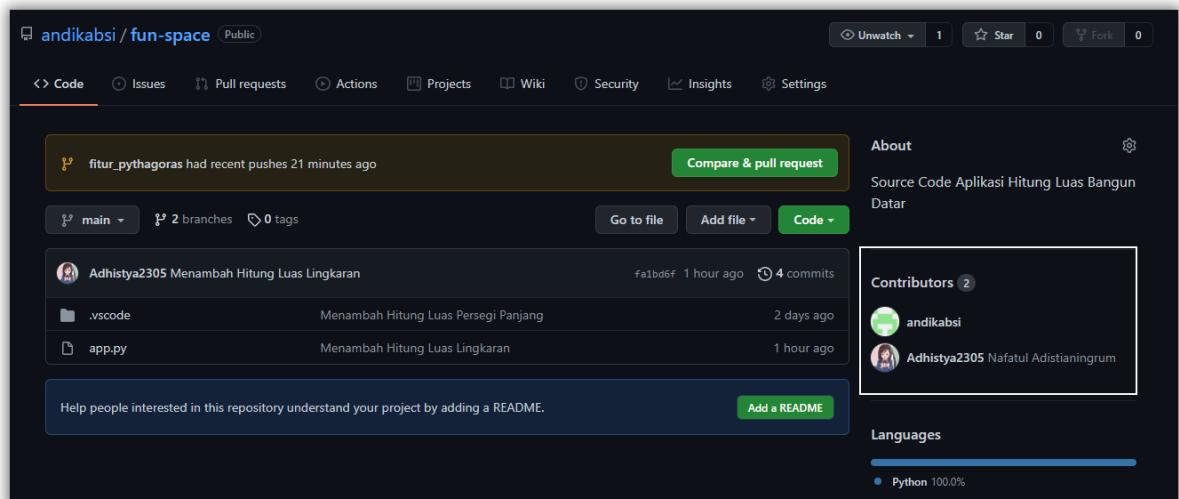
```
PS C:\Users\Andika Tulus\Desktop\nafatul\fun-space> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

# Bekerja Dengan Git dan Github

## ( Melakukan Pembaruan & Penggabungan Project )

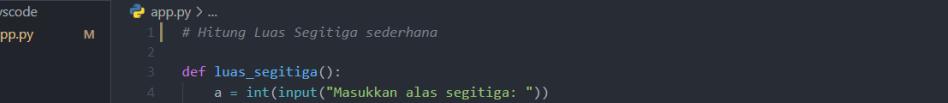
Beberapa hari telah berlalu, andika sekarang sedang cuti kuliah dan sekarang andika akan melanjutkan project “Aplikasi Hitung Luas Bangun Datar”.

Andika telah melihat di repository github nya, bahwa seorang sahabatnya bernama “Adhistya” telah ikut berkontribusi mengembangkan project tersebut.



Maka dari itu yang pasti di repository milik saya ( Andika ) belum terupdate akan segala perubahan yang dilakukan oleh adhistya.

- Pertama yang akan kita lakukan adalah mengambil atau memperbarui segala perubahan yang dilakukan oleh adhistya di github ke repository lokal milik andika.
- Buka folder project kita di VSCode dan buka terminal di VSCode
- Sebelumnya bisa kita lihat, bahwa terakhir kita hanya merapihkan kode dan tidak menambahkan kode apapun.



The screenshot shows a Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** app.py - fun-space - Visual Studio Code
- Left Sidebar:** Includes icons for Explorer, Search, Problems, and others, with "vscode" selected.
- Central Area:** A code editor with the file "app.py" open. The code is as follows:

```
# Hitung Luas Segitiga sederhana
def luas_segitiga():
    a = int(input("Masukkan alas segitiga: "))
    t = int(input("Masukkan tinggi segitiga: "))
    luas = a * t / 2
    print("luas segitiga adalah: ", luas)
luas_segitiga()
```

- Perlu diingat bahwa yang melakukan perubahan seperti meambahkan hitung luas lingkaran dan membuat branch pythagoras adalah adhistya, bukan andika.
  - Maka dari itu, kita akan melakukan **git pull** . Perintah ini digunakan untuk mengambil dan menggabungkan perubahan secara jarak jauh ke repository lokal.

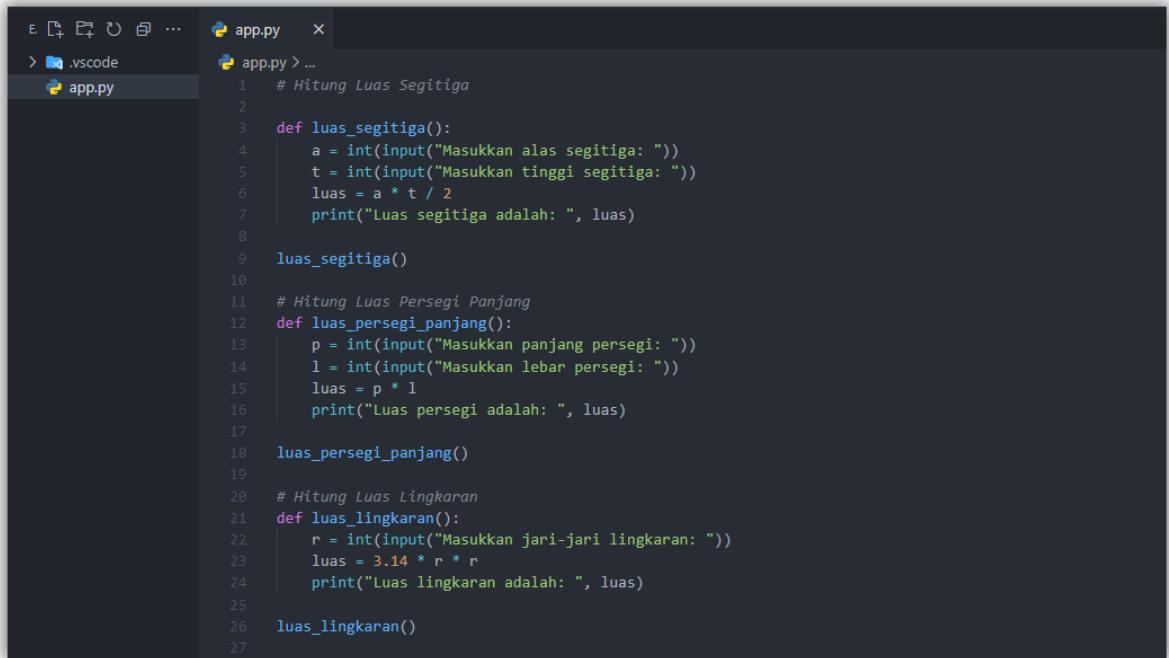
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell + ⌂
PS C:\Users\Andika Tulus\Desktop\fun-space> git pull origin main
From https://github.com/andikabsi/fun-space
 * branch            main      -> FETCH_HEAD
Already up to date.
PS C:\Users\Andika Tulus\Desktop\fun-space>
```

*git pull origin main*

Artinya :

Kita mengintruksikan git untuk mengambil ( pull ) semua perubahan dari repository awal ( origin ) di branch utama ( main ) dan menggabungkannya ke repository lokal kita.

- Kita lihat, kode telah berubah seperti perubahan terakhir yang dilakukan oleh adhistya.



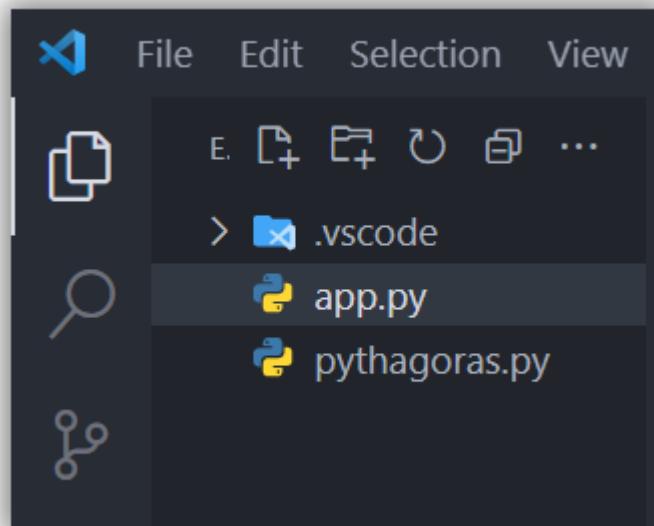
The screenshot shows a dark-themed VS Code interface. In the top bar, there are icons for file operations like new, open, save, and close. The title bar says "app.py". Below the title bar, the file tree shows ".vscode" and "app.py". The main editor area contains the following Python code:

```
1 # Hitung Luas Segitiga
2
3 def luas_segitiga():
4     a = int(input("Masukkan alas segitiga: "))
5     t = int(input("Masukkan tinggi segitiga: "))
6     luas = a * t / 2
7     print("Luas segitiga adalah: ", luas)
8
9 luas_segitiga()
10
11 # Hitung Luas Persegi Panjang
12 def luas_persegi_panjang():
13     p = int(input("Masukkan panjang persegi: "))
14     l = int(input("Masukkan lebar persegi: "))
15     luas = p * l
16     print("Luas persegi adalah: ", luas)
17
18 luas_persegi_panjang()
19
20 # Hitung Luas Lingkaran
21 def luas_lingkaran():
22     r = int(input("Masukkan jari-jari lingkaran: "))
23     luas = 3.14 * r * r
24     print("Luas lingkaran adalah: ", luas)
25
26 luas_lingkaran()
27
```

- Beberapa saat, andika tau bahwa adhistya telah membuat fitur baru yaitu “Menghitung Pythagoras”, namun nafatul membuatnya di cabang ( branch ) lain.
- Lalu andika menyukai fitur yang telah di buat oleh adhistya, dan ingin menerapkannya ke program utama.
- Untuk menggabungkan cabang yang dibuat adhistya ( fitur\_pythagoras) dengan cabang utama ( main ) kita akan menggunakan perintah **git merge**.  
Perintah tersebut untuk menggabungkan cabang lain di dalam repository ke cabang aktif atau cabang utama.

```
PS C:\Users\Andika Tulus\Desktop\fun-space> git merge origin/fitur_pythagoras
Updating fa1bd6f..bf9a39f
Fast-forward
  app.py      | 2 --
  pythagoras.py | 9 ++++++++
  2 files changed, 9 insertions(+), 2 deletions(-)
  create mode 100644 pythagoras.py
PS C:\Users\Andika Tulus\Desktop\fun-space>
```

- Dan setelah berhasil maka cabang yang dibuat adhistya, telah kita gabungkan ke cabang utama.



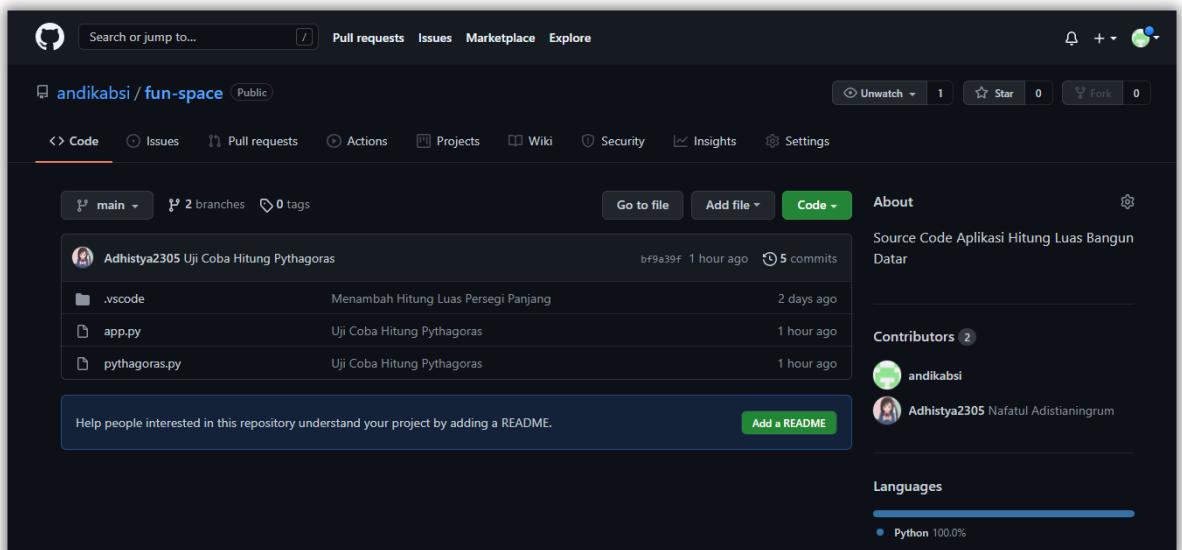
- Sekarang semuanya telah digabungkan di repository lokal dan kita akan gabungkan lalu push semuanya ke repository github.
- Selanjutnya kita guakan perintah **git add .** untuk menambahkan berkas ke staging area
- Lalu kita commit atau simpan perubahannya dengan judul commit “Kolaborasi Pertama”

```
PS C:\Users\Andika Tulus\Desktop\fun-space> git add .
PS C:\Users\Andika Tulus\Desktop\fun-space> git commit -m "Kolaborasi Pertama"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

- Terakhir kita push dan lihat di repository github kita telah digabungkan branch dengan branch sebelumnya, *notifikasi pull request dan compare* telah tiada artinya kita telah berhasil berkolaborasi.

```
PS C:\Users\Andika Tulus\Desktop\fun-space> git commit -m "Kolaborasi Pertama"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\Andika Tulus\Desktop\fun-space> git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/andikabsi/fun-space.git
    fa1bd6f..bf9a39f  main -> main
```



# Bekerja Dengan Git dan Github ( Visual Studio Code )

Sebelumnya kita telah belajar dari dasar git sampai melakukan kolaborasi, namun sebelumnya kita selalu menggunakan perintah git di Terminal, memang pada kenyataanya awal mulai git diperkenalkan sistem ini menggunakan perintah terminal.

Seiring berjalannya waktu dan berkembangnya teknologi, kita bisa melakukannya tanpa perintah di terminal, kita cukup klik-klik saja seperti menggunakan aplikasi pada umumnya.

Untuk lebih jelasnya kita akan praktik secara langsung.

Persyaratan Sistem :

- Komputer dengan OS Windows, Linux, Mac yang telah terpasang git dan telah di konfigurasi
- Terinstall Visual Studio Code

Langkah-langkah :

- Kita akan menggunakan project baru dan berbeda dari sebelumnya, ini dibuat agar kamu tidak bingung.  
Kita akan membuat sebuah games dengan Python yang berjudul “Kertas Batu Gunting”
- Buat Folder baru dan beri nama “pygames”
- Buat File baru beri nama “games.py”
- Lalu, masukan source kode berikut.

```
# Membuat Games Kertas Batu Gunting Kertas
def main():
    print("Pilih Salah Satu!")
    print("1. Kertas")
    print("2. Batu")
```

```

print("3. Gunting")
pilih = int(input("Masukkan Pilihan Anda: "))
import random
comp = random.randint(1,3)
if pilih == 1:
    if comp == 1:
        print("Kamu Memilih Kertas")
        print("Komputer Memilih Kertas")
        print("Seri!")
    elif comp == 2:
        print("Kamu Memilih Kertas")
        print("Komputer Memilih Batu")
        print("Komputer Menang!")
    else:
        print("Kamu Memilih Kertas")
        print("Komputer Memilih Gunting")
        print("Kamu Menang!")
elif pilih == 2:
    if comp == 1:
        print("Kamu Memilih Batu")
        print("Komputer Memilih Kertas")
        print("Kamu Menang!")
    elif comp == 2:
        print("Kamu Memilih Batu")
        print("Komputer Memilih Batu")
        print("Seri!")
    else:
        print("Kamu Memilih Batu")
        print("Komputer Memilih Gunting")
        print("Komputer Menang!")
elif pilih == 3:
    if comp == 1:
        print("Kamu Memilih Gunting")
        print("Komputer Memilih Kertas")
        print("Komputer Menang!")
    elif comp == 2:
        print("Kamu Memilih Gunting")
        print("Komputer Memilih Batu")
        print("Kamu Menang!")
    else:
        print("Kamu Memilih Gunting")
        print("Komputer Memilih Gunting")
        print("Seri!")
else:
    print("Pilihan Tidak Tersedia")

main()

```

- Simpan dan coba jalankan.

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS C:\Users\Andika Tulus\Desktop\pygames> python .\games.py
Pilih Salah Satu!
1. Kertas
2. Batu
3. Gunting
Masukkan Pilihan Anda: 2
Kamu Memilih Batu
Komputer Memilih Batu
Seri!
PS C:\Users\Andika Tulus\Desktop\pygames>

```

- Kita akan push project games tersebut ke github melalui vscode
- Klik icon git di sebelah kiri layar

```

File Edit Selection View Go Run Terminal Help games.py - pygames - Visual Studio Code

SOURCE CONTROL Get Started games.py > main
The folder currently open doesn't have a git repository. You can initialize a repository which will enable source control features powered by git.
Initialize Repository
To learn more about how to use git and source control in VS Code read our docs.

1 # Membuat Games Kertas Batu Gunting Kertas
2 def main():
3     print("Pilih Salah Satu!")
4     print("1. Kertas")
5     print("2. Batu")
6     print("3. Gunting")
7     pilih = int(input("Masukkan Pilihan Anda: "))
8     import random
9     comp = random.randint(1,3)
10    if pilih == 1:
11        if comp == 1:
12            print("Kamu Memilih Kertas")
13            print("Komputer Memilih Kertas")
14            print("Seri!")

```

- Klik lagi di tombol “Publish to Github”

```

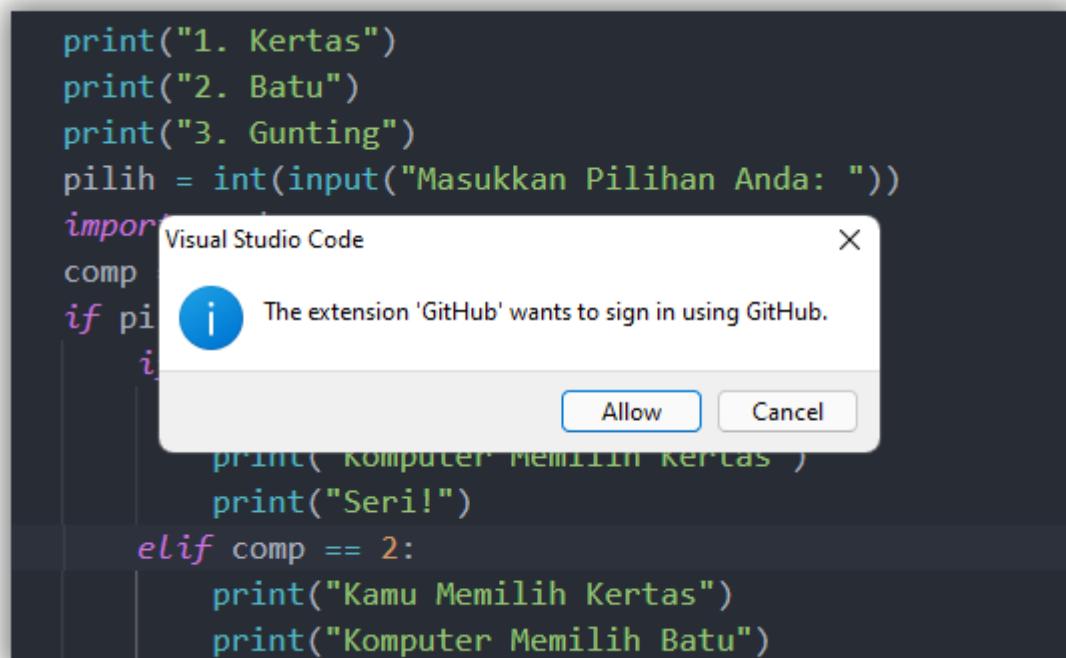
The folder currently open doesn't have a git repository. You can initialize a repository which will enable source control features powered by git.
Initialize Repository
To learn more about how to use git and source control in VS Code read our docs.

You can also directly publish this folder to a GitHub repository. Once published, you'll have access to source control features powered by git and GitHub.
Publish to GitHub

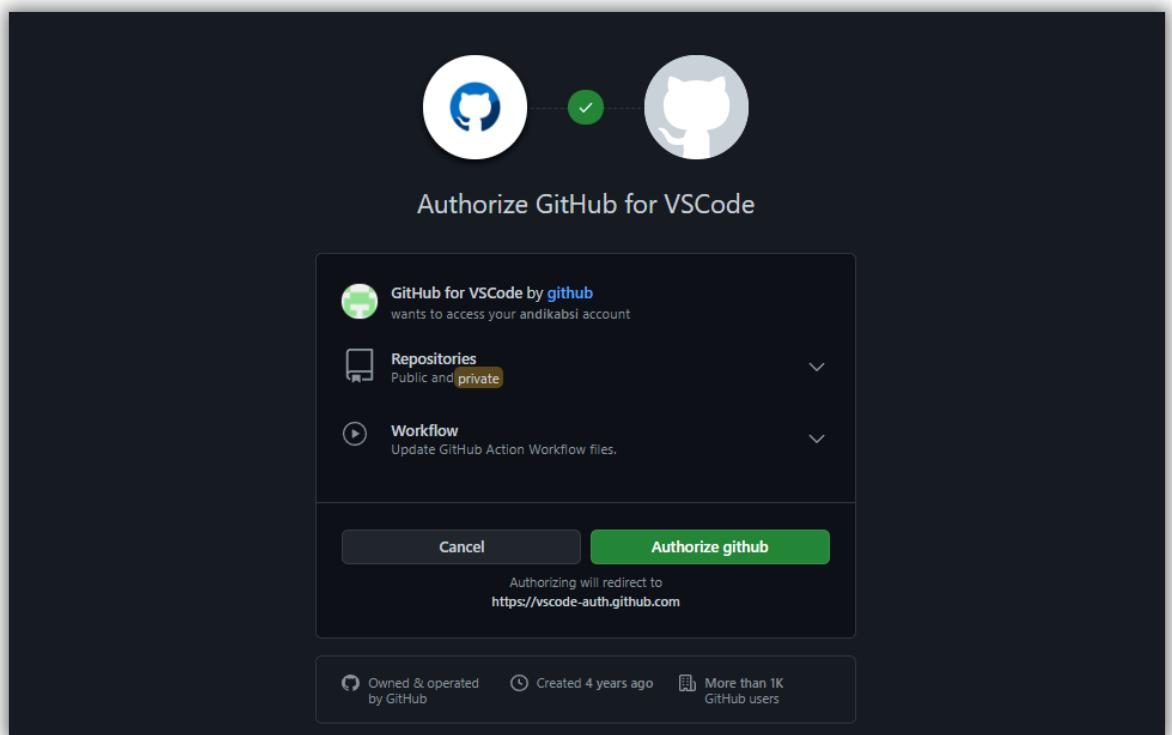
1 # Membuat Games Kertas Batu Gunting Kertas
2 def main():
3     print("Pilih Salah Satu!")
4     print("1. Kertas")
5     print("2. Batu")
6     print("3. Gunting")
7     pilih = int(input("Masukkan Pilihan Anda: "))
8     import random
9     comp = random.randint(1,3)
10    if pilih == 1:
11        if comp == 1:
12            print("Kamu Memilih Kertas")
13            print("Komputer Memilih Kertas")
14            print("Seri!")
15        elif comp == 2:
16            print("Kamu Memilih Kertas")
17            print("Komputer Memilih Batu")
18            print("Komputer Menang!")
19        else:

```

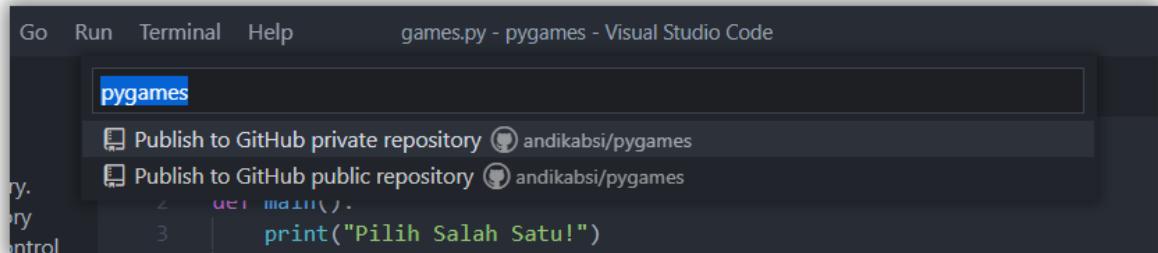
- Jika ada notifikasi seperti ini, klik **allow**



- Maka akan otomatis masuk browser, dan klik **continue**
- Pilih **Authorize Github**

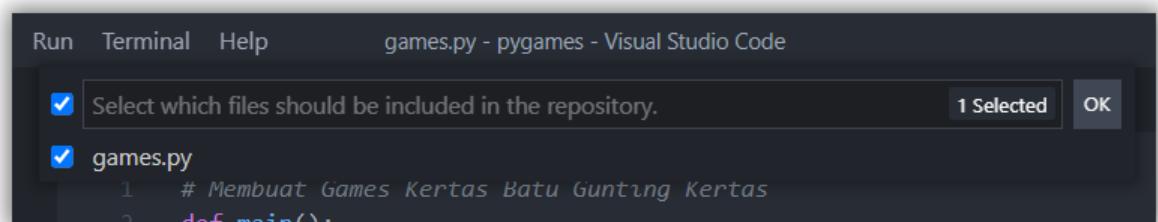


- Setelah otomatis kembali ke Vscode, maka klik **Publish to github** kembali, klik allow, dan klik **Continue**
- Kemudian akan tampil notifikasi seperti berikut.

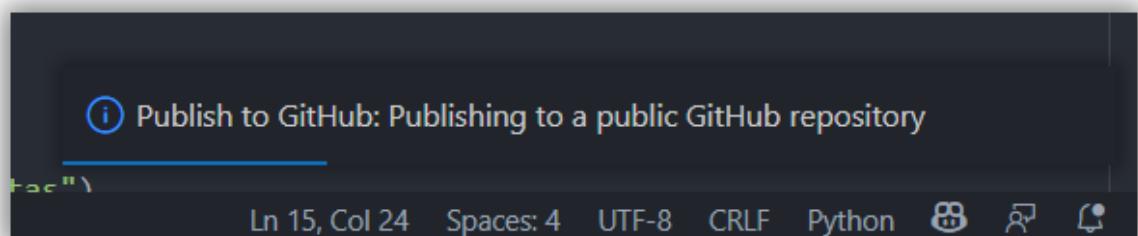


Sekarang kamu bisa memilih, akan di publish ke repository github secara privat atau publik

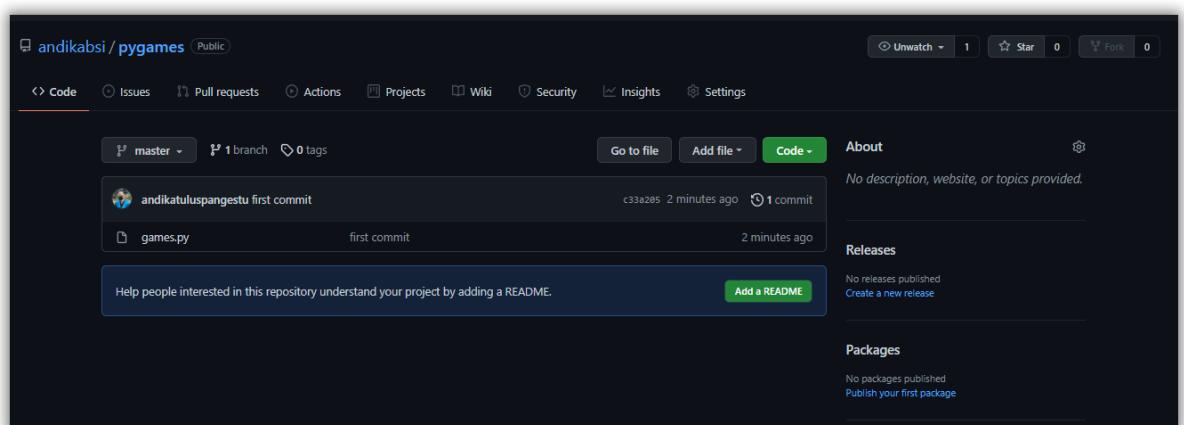
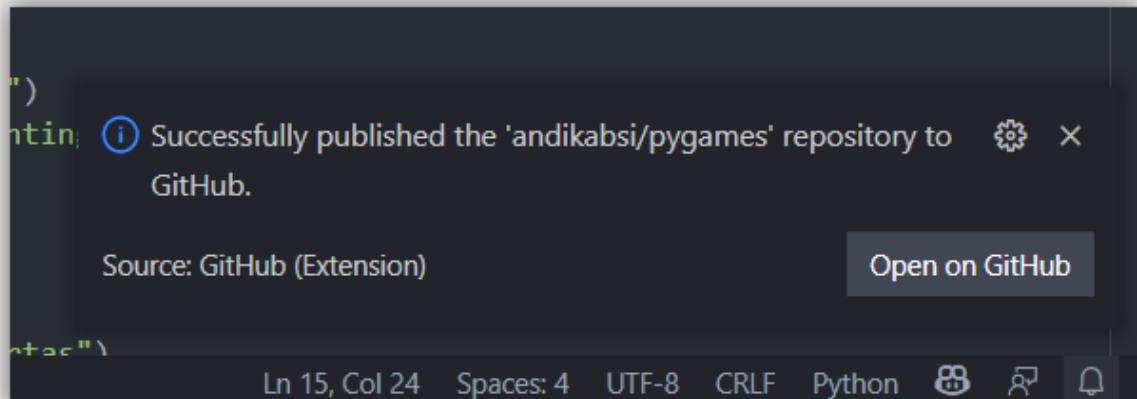
- Saya akan memilih repository publik agar bisa diakses semua orang.
- Pilih berkas yang akan di publish dengan cara centang berkas dan klik **ok**, kita saat ini hanya membuat dan memiliki satu berkas.



- Maka akan tampil notifikasi progress publish project ke github.



- Jika telah selesai dan berhasil, maka akan tampil notifikasi sukses, kamu bisa klik **Open on Github** untuk lihat secara langsung di github.

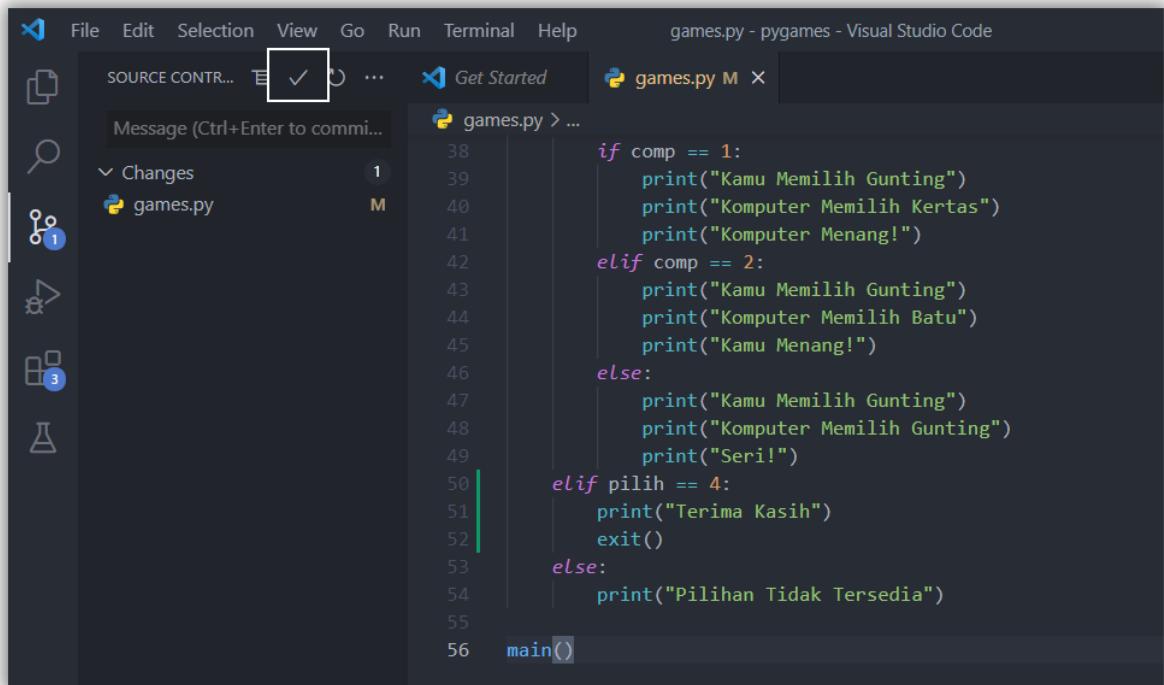


- Sekarang kita akan melakukan perubahan pada kode kita, yaitu kita akan menambahkan pilihan keluar aplikasi.

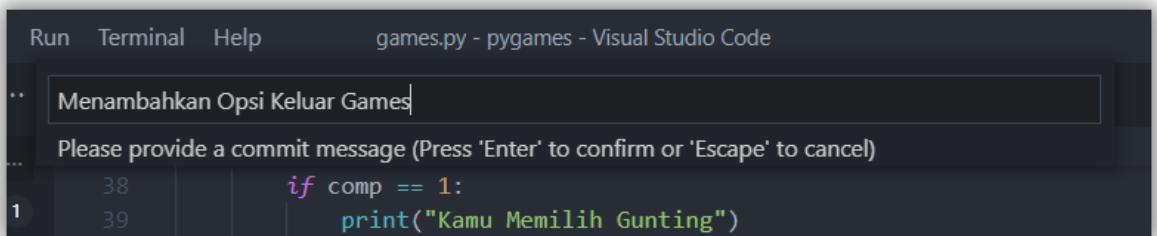
```
py games.py > main
1 # Membuat Games Kertas Batu Gunting Kertas
2 def main():
3     print("Pilih Salah Satu!")
4     print("1. Kertas")
5     print("2. Batu")
6     print("3. Gunting")
7     print("4. Keluar")
8     pilih = int(input("Masukkan Pilihan Anda: "))
9
10    import random
11    comp = random.randint(1,3)
```

```
49         print("Seri!")
50     elif pilih == 4:
51         print("Terima Kasih")
52         exit()
53     else:
54         print("Pilihan Tidak Tersedia")
55
56 main()
```

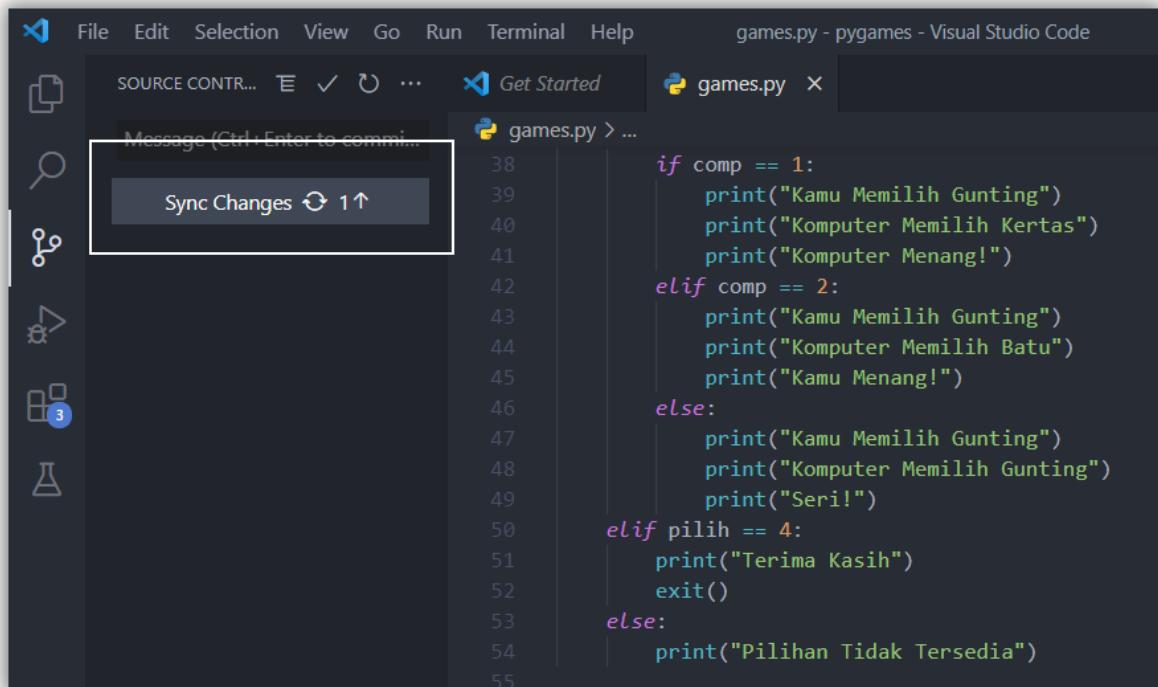
- Simpan dan klik ikon centang



- Masukan Pesan Commit dan Tekan ENTER

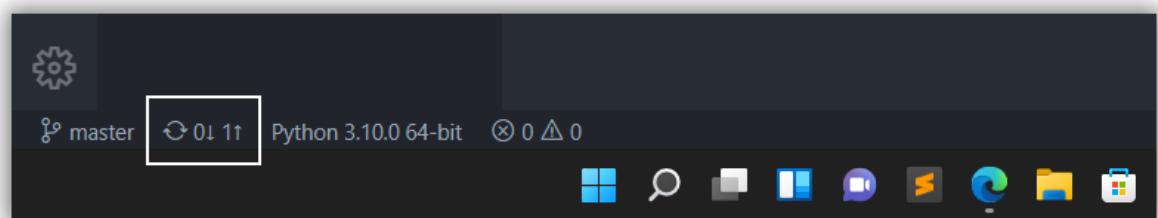


- Lalu tekan **Sync Changes** atau *icon sync* di pojok bawah

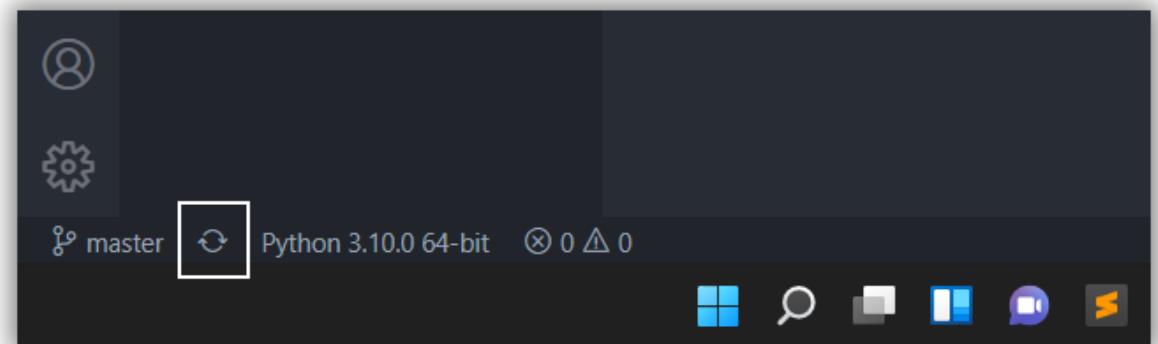


A screenshot of the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. A tab bar shows "Get Started" and "games.py - pygames - Visual Studio Code". On the left is a dark sidebar with icons for file operations like cut, copy, paste, search, and refresh. In the center, a code editor window titled "games.py > ..." displays Python code for a game. The code uses if-elif-else statements to handle user input (1, 2, 3, 4) and computer choice (Gunting, Kertas, Batu). A message box in the center says "Sync Changes" with a circular arrow icon and "1 ↑". The status bar at the bottom shows "master", "Python 3.10.0 64-bit", and a sync icon with "0 ↓ 1 ↑".

```
if comp == 1:
    print("Kamu Memilih Gunting")
    print("Komputer Memilih Kertas")
    print("Komputer Menang!")
elif comp == 2:
    print("Kamu Memilih Gunting")
    print("Komputer Memilih Batu")
    print("Kamu Menang!")
else:
    print("Kamu Memilih Gunting")
    print("Komputer Memilih Gunting")
    print("Seri!")
elif pilih == 4:
    print("Terima Kasih")
    exit()
else:
    print("Pilihan Tidak Tersedia")
```



- Jika sync icon nya adalah 0 maka sinkronisasi atau push pembaruan telah berhasil dilakukan.



- Kita lihat telah berhasil, di repository terjadi perubahan yang baru saja kita lakukan.

The screenshot shows a GitHub repository page for 'andikabsi / pygames'. The repository is public and has one branch ('master') and no tags. A recent commit was made by 'andikatuluspangestu' titled 'Menambahkan Opsi Keluar Games' at 3f7fa85, 5 minutes ago. The commit message is 'Menambahkan Opsi Keluar Games'. Below the commit, there is a note: 'Help people interested in this repository understand your project by adding a README.' with a 'Add a README' button.

- Biasanya kita menggunakan perintah git diff untuk mengetahui perubahan setiap kode, namun di vscode kita bisa melakukan hal tersebut dengan cara klik kotak marker yang berada di dekat nomer baris.

The screenshot shows the VS Code interface with a git diff view for the file 'games.py'. The diff shows one change: the addition of a new option '4. Keluar' to the list of choices. The code snippet below illustrates the change:

```

5      print("2. Batu")
6      print("3. Gunting")
7      print("4. Keluar")
8      pilih = int(input("Masukkan Pilihan Anda: "))
9      import random
10
11      pilih = int(input("Masukkan Pilihan Anda: "))
12      import random
13      comp = random.randint(1,3)
14      if pilih == 1:

```

- Selesai.

## **Bekerja Dengan Git dan Github ( Menggunakan SSH Github )**

Kedepannya kita akan sering menggunakan git dan github, maka dari itu mungkin akan sering juga kita melakukan push, namun tanpa menggunakan SSH kita akan melakukan login ke akun github setiap kita melakukan push perubahan. Untuk itu kita akan belajar mengkonfigurasi SSH Github dengan Git CMD.

- Buka Git Bash
- Lalu, ketikan perintah berikut dan ENTER.

ssh-keygen

- Maka akan terdapat pertanyaan, pertanyaan pertama masukan nama singkat kamu atau apapun terserah, dan pertanyaan selanjutnya hanya tekan enter dan enter, sampai muncul seperti ini.

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
PS C:\Users\Andika Tulus\Desktop\pygames> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Andika Tulus/.ssh/id_rsa): andika
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in andika.
Your public key has been saved in andika.pub.
The key fingerprint is:
SHA256:sPdWY6vWLPOeQS7qdn+9swRZlFWNjqmjc/FEYsYnmw andika tulus@DESKTOP-HI09FDS
The key's randomart image is:
+---[RSA 3072]---+
|       . o*|
|       o + oo=|
|       . E.= +|
|       o . =oo o|
|       . S o=..*|
|       . . ooB+ .|
|       o+*o..|
|       .o=+o.o.|
|       .+oo=+.+|
+---[SHA256]---+
PS C:\Users\Andika Tulus\Desktop\pygames>
```

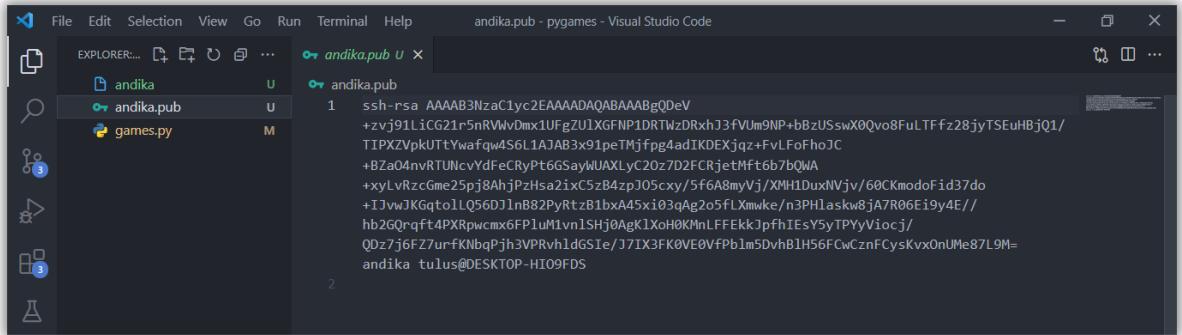
- Setelah itu akan ada file baru ssh di folder project kita.

Name	Date modified	Type	Size
.git	21/11/2021 15:43	File folder	
andika	21/11/2021 15:57	File	3 KB
andika.pub	21/11/2021 15:57	PUB File	1 KB
games	21/11/2021 15:48	PY File	2 KB

Keterangan :

- id\_rsa — ini adalah kunci PRIVATE kamu. Jangan bagikan ini dengan orang lain. Ini adalah rahasia kamu.
- id\_rsa.pub — ini adalah kunci PUBLIC kamu. Ini tidak mengandung rahasia. kamu dapat membaginya dengan orang lain.

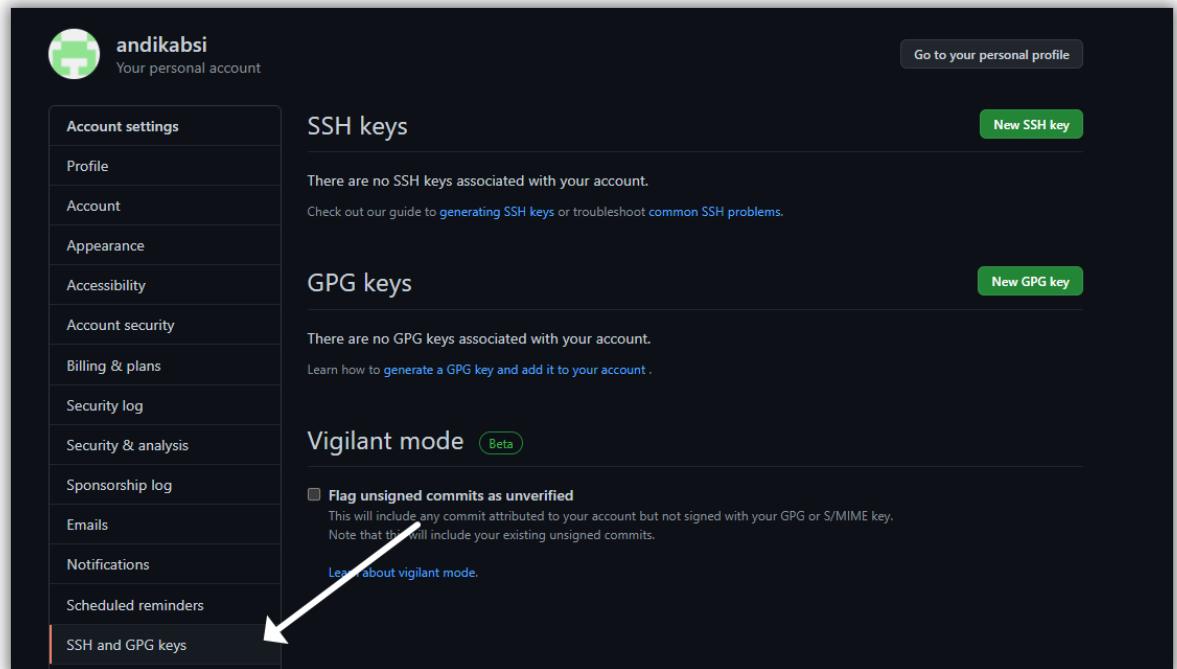
- Selanjutnya, kita tambahkan kunci SSH kita ke GitHub
- Dengan vscode, kita buka kunci publik kita.



```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDeV
+zwj91LiCG21r5RVwDmx1UFgZUJXGFNP1RTWzDRxhJ3fVUm9NP+bBzUSswX0Qvo8FuLTFFz28jyTSeuHBjQ1/
TIPXZVpkUTtYwafqw4S6L1AJAB3x91peTMjfpq4adIKDEXjqz+FvLFoFhoJC
+B2a04nvRTUlcvydfcRyPT6GSaywUAxLyC20z7D2FCRjetMft6b7DQWA
+xyLvRzcGme25pj8ahjPzHsa2ixC5zb4zpJ05cxy/5f6A8myVj/XMH1DuxNVjv/60CKmodoFid37do
+I1vwJKGqto1LQ56Dj1n882PyRtzb1bxA45xi03qAgzo5fLXmwke/n3PHlaskw8jA7R06Ei9y4E// 
hb2GQrqft4PXRpwmcmxf6PluM1vn1Shj0Agk1XoH0KmLnFFFkjpfhIEsYsyTPYyviocj/
QDz7j6FZ7urfkNbqPjh3VPVrh1dGStE/37IX3FK0VE0VFpb1m5DvhB1H56FCwCznFcysKvxOnUMe87L9M=
```

andika tulus@DESKTOP-HIO9FDS

- Sekarang kita buka pengaturan github di browser atau melalui tautan berikut [Your Profile \(github.com\)](https://github.com)
- Pilih menu **SSH and GPG Keys**

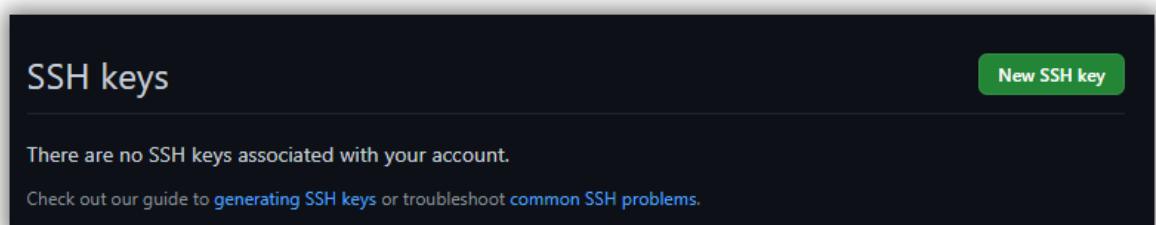


andikabsi  
Your personal account

[Go to your personal profile](#)

<b>Account settings</b>	<b>SSH keys</b>	<a href="#">New SSH key</a>
Profile	There are no SSH keys associated with your account.	
Account	Check out our guide to <a href="#">generating SSH keys</a> or troubleshoot <a href="#">common SSH problems</a> .	
Appearance		
Accessibility		
Account security		
Billing & plans		
Security log		
Security & analysis		
Sponsorship log		
Emails		
Notifications		
Scheduled reminders		
<b>SSH and GPG keys</b>	<b>GPG keys</b>	<a href="#">New GPG key</a>
	<b>Vigilant mode</b> <small>(Beta)</small>	
	<input type="checkbox"/> <b>Flag unsigned commits as unverified</b> This will include any commit attributed to your account but not signed with your GPG or S/MIME key. <small>Note that this will include your existing unsigned commits.</small>	
	<a href="#">Learn about vigilant mode.</a>	

- Klik Tombol Hijau “New SSH Key”



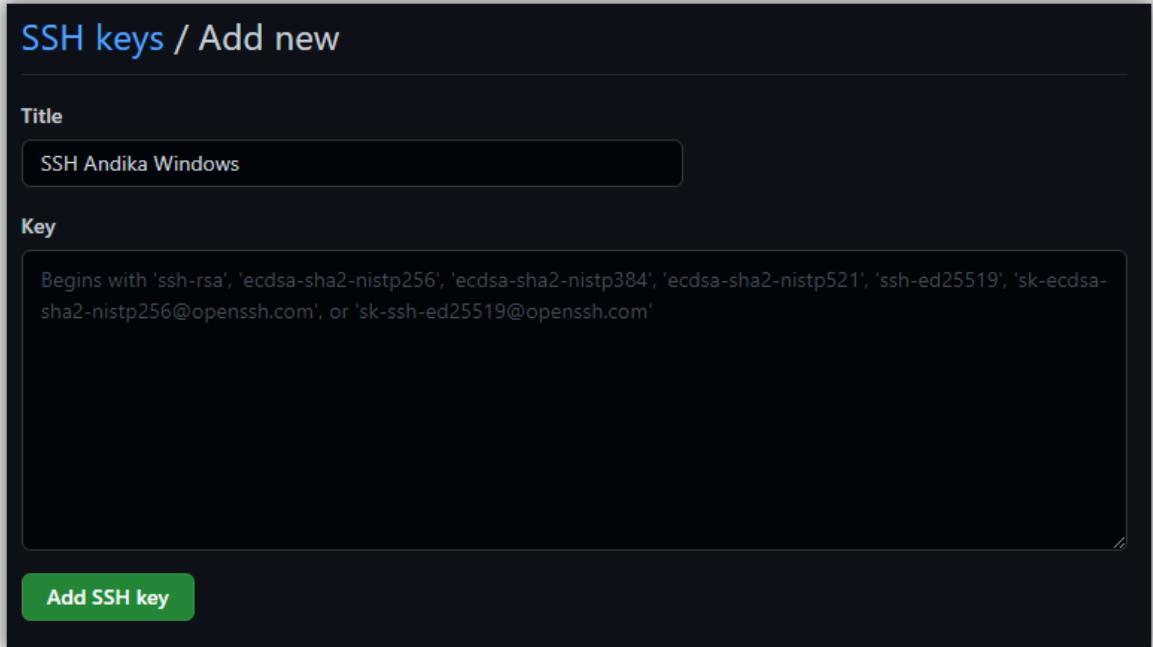
**SSH keys**

[New SSH key](#)

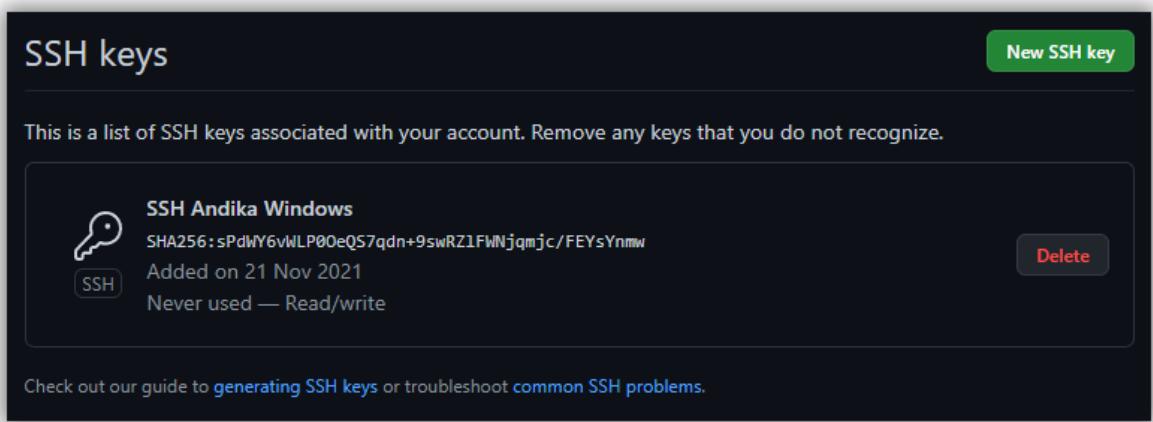
There are no SSH keys associated with your account.

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

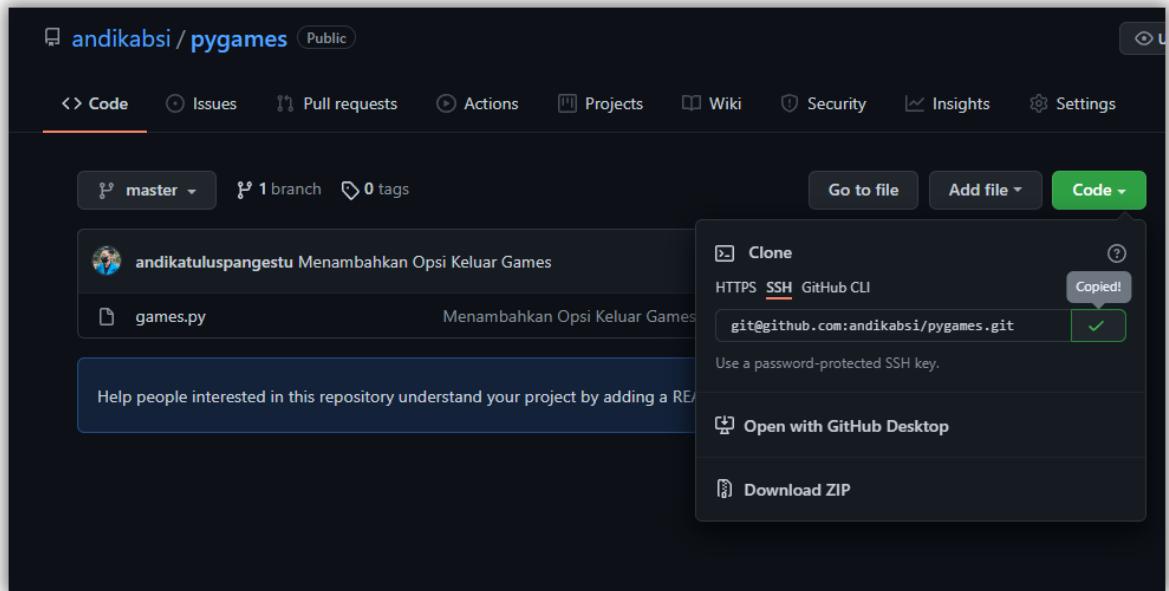
- Masukan Nama SSH dan masukan juga key ssh yang kita buka di vscode tadi, kemudian klik tombol **Add SSH Key**



- Dan kita berhasil membuatnya.



- Untuk mengeceknya maka kita coba **clone** repository kita dengan SSH. Saya akan menclone repo pygames milik kita.



- Kita copy clone SSH nya yaitu :

<git@github.com:andikabsi/pygames.git>

- Lalu kita masuk ke folder project pygames dan paste di git bash dan Enter
- Selesai.

## Rangkuman Git dan Github

- Git didirikan oleh Linus Torvalds di tahun 2005
- Git berguna untuk membantu tim dalam berkolaborasi dalam membangun dan mengembangkan sebuah project, dengan Git, pekerjaan menjadi lebih praktis. Kita dapat melacak perubahan pada berkas yang ada dalam di *repository* atau direktori kerja. Selain itu, kita juga dapat mengelola versi rilis dari sebuah proyek dalam *repository*.
- Penggunaan Git memerlukan adanya ***repository***, yaitu wadah atau tempat penyimpanan proyek di mana setiap berkas yang ada di dalamnya dapat dilacak jika terjadi perubahan. Berdasarkan jenisnya, *repository* terbagi menjadi 2, yaitu *local repository* dan *remote repository*.
- *Local repository* merupakan tempat penyimpanan lokal yang berada di komputer kita. Local repository dapat kita ubah-ubah (hapus, modifikasi, dan tambah) sesuai dengan keinginan kita, sebelum akhirnya nanti di-*push*.
- *Remote Repository* merupakan tempat penyimpanan berkas-berkas pekerjaan atau kenangan yang kita miliki di dalam server. Anda bisa menggunakan berbagai layanan penyimpanan berbasis *cloud* yang sangat populer seperti GitHub. Dengan menggunakan Remote Repository, orang lain dapat mengakses repository yang kita simpan dengan mudah.
- Saat membuat *repository* terdapat pengaturan visibilitas yang terdiri dari 2 yaitu *private* dan *public*.
- *Private repository* merupakan *repository* yang bersifat tertutup/pribadi dan hanya akun–akun yang telah diberikan akses saja yang bisa melihatnya.

- *Public repository* merupakan *repository* yang bersifat terbuka dan semua orang dapat melihat dan menggunakan projek yang ada di *repository* tersebut.
- **Add a README file** atau penambahan *README File* pada *repository* baru. Ketika *repository* telah dibuat, GitHub secara otomatis membuatkan file *Readme* ke dalam *repository*
- **Add .gitignore** atau penambahan berkas *.gitignore* pada *repository*. Ketika *repository* telah terbuat, file *.gitignore* akan tercipta sesuai dengan pilihan *template.gitignore* yang dipilih.
- **Choose a license** atau penambahan berkas *license* pada *repository*. Ketika membuat *repository*, Anda bisa memilih lisensi untuk *repository* tersebut sehingga proyek yang ada di dalamnya dapat digunakan secara bebas oleh orang lain.
- Istilah dalam Git dan Github

Git Repository	Folder penyimpanan project dalam git
Git Branch	Percabangan untuk versi baru dalam sebuah project
Git Clone	Mengambil sebuah <i>repository</i> dan menyimpannya pada direktori lokal.
Git Commit	Catatan riwayat perubahan sebuah berkas
Git Push	Mengunggah hasil perubahan sebuah berkas ke <i>repository</i>
Git Checkout	Melakukan perpindahan dari <i>commit</i> satu ke <i>commit</i> yang dituju.

	Namun hanya bersifat sementara. Git Checkout juga dapat digunakan untuk berpindah dari satu cabang kerja (branch) ke cabang kerja lainnya.
Git Reset	Mengembalikan keadaan suatu commit ke dalam keadaan sebelum terjadi perubahan sesuai dengan <i>commit</i> yang dituju. Namun, git reset akan menghapus beberapa riwayat commit sesudahnya.
Git Revert	Mengembalikan keadaan suatu berkas sebelum terjadi suatu perubahan sesuai dengan tujuan <i>commit</i> yang dituju. Git Revert tidak akan menghilangkan riwayat commit sesudahnya.

- Github adalah layanan berbasis git yang berjalan secara online
- Latihan Membuat Akun GitHub  
Berikut beberapa tahapan yang bisa Anda lakukan untuk membuat akun GitHub:
  - Mengakses alamat <https://github.com/signup>.
  - Melengkapi data-data yang diperlukan.
  - Melakukan verifikasi pendaftaran menggunakan kode unik yang dikirimkan GitHub melalui email.
- Setiap proyek repository memiliki kolaborator dan kontributor. Seorang kolaborator merupakan anggota tim

yang memiliki akses dalam sebuah repositori proyek (*project repository*).

- Seorang kolaborator juga dapat berperan sebagai kontributor. Kontributor adalah siapa saja yang dapat melakukan commit atau menyimpan cuplikan perubahan ke dalam repository (*git push*). Akan tetapi, tidak semua kontributor dapat melakukan *commit* secara langsung. Ini disebabkan karena mereka tidak memiliki akses untuk melakukan push ke dalam repository tersebut (atau tidak berperan sebagai kolaborator). Sehingga mereka perlu melakukan *pull request* untuk melakukan commit pada repository orang lain.

## Daftar Referensi

- Dokumentasi Github  
( <https://docs.github.com> )
- Petanikode – Belajar Git Pemula  
( <https://www.petanikode.com/tutorial/git/> )
- Medium
  - ( <https://medium.com/@fahmiprasetiio/belajar-git-untuk-pemula-7625c686c68f> )
  - ( <https://medium.com/devops-with-valentine/2021-how-to-set-up-your-ssh-key-for-github-on-windows-10-afe6e729a3c0> )
  - ( <https://medium.com/@fachrizalrifahdi/cara-menambahkan-ssh-key-di-github-3444128f11fc> )
- Github Pages  
( <https://rogerdudler.github.io/git-guide/index.id.html> )
- Dicoding Indonesia  
( <https://www.dicoding.com/academies/317/tutorials> )
- Progate  
( <https://progate.com/courses/git> )
- Repository Github untuk latihan
  - ( <https://github.com/andikabsi/fun-space> )
  - ( <https://github.com/andikabsi/pygames> )
  - ( <https://github.com/andikatuluspangestu/python-tiket-kereta-api> )
  - ( <https://github.com/andikabsi> )
  - ( <https://github.com/Adhistya2305> )
  - ( <https://github.com/andikatuluspangestu> )
- Gambar, Ikon, dan ilustrasi
  - ( <https://flaticon.com> )
  - ( <https://freepik.com> )
  - ( <https://iconfinder.com> )



# GitHub

Semoga bermanfaat... Sekian dan terima kasih ☺

Copyright ©2021 Husni Faqih & Andika Tulus Pangestu