**PRAKTIKUM I**

**VCS pada Git dan GitHub**

**SUB-CAPAIAN PEMBELAJARAN MATA KULIAH:**

Mahasiswa mampu mendemonstrasikan (C3), mendiskusikan (A2) dan mengerjakan (P3) Version Control System (VCS) dengan menggunakan Git dan GitHub dengan benar.

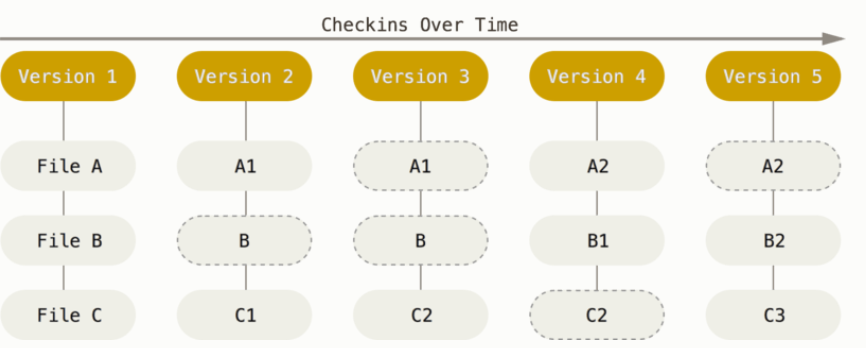
**POKOK BAHASAN:**

1. VCS dengan Git
2. VCS dengan GitHub
   1. **Tujuan Praktikum**

Setelah melakukan praktikum, mahasiswa diharapkan untuk dapat:

1. Menjelaskan dan menerapkan konsep VCS
2. Menjelasakan dan menerapkan konsep VCS pada Git dan GitHub
3. Memaksimalkan kreativitasnya dalam menerjemahkan algoritma ke dalam bahasa python dengan memanfaat konsep VCS pada Git dan GitHub untuk menyelesaikan permasalahan.
   1. **VCS pada Git**

Git merupakan salah satu software untuk melakukan Version Control System (VCS) atau source code management. Git dikembangkan oleh Linux Torvald (2005). Git termasuk VCS berjenis distributed. Dengan menggunakan Git, kita hanya mengelola setiap perubahan secara lokal. Perbedaan Git dengan VCS lainnya adalah bagaimana cara Git berpikir tentang datanya. Data dalam Git diibaratkan sebagai sekumpulan snapshot dari file system. Setiap kali anda melakukan commit atau menyimpan keadaan proyek and di git, anda seperti mengambil sebuah gambar tentang bagaimana tampilan semua berkas anda pada saat itu, dan menyimpan acuan kepada snapshot tersebut. jika file-file itu tidak berubah, Git tidak menyimpan file tersebut lagi, tapi hanya menautkan ke file yang sama persis sebelumnya yang telah tersimpan. Git berpikir tentang datanya lebih seperti sebuah aliran snapshot, sebagaimana pada gambar di bawah ini.



Ada 3 kondisi/keadaan/state dalam Git Repo:

1. Working tree/working directory: folder tempat kita mengerjakan project
2. Staging Area: pada staging area, kita sudah memberi tahu Git kalau kita sudah melakukan perubahan dan berisi informasi tentang apa yang menjadi commit anda selanjutnya.
3. History/Git Directory/Repository: Apapun yang sudah kita commit, akan masuk ke area history. Git directory juga menyimpan metadata dan basis data objek untuk project yang sedang dibangun.

Alur kerja dasar Git adalah sebagai berikut: pertama, anda mengubah file dari working directory. Ketika anda menyimpan berkasanya, anda menambahkan snapshot darinya ke staging area. Ketika anda melakukan commit, file-file yang ada pada staging are diambil dan snapshot tersebut disimpan secara tetap kedalam Git directory.

Diagram

Description automatically generated with medium confidence

Dalam bekerja menggunakan Git, ada dua jenis keadaan file yaitu Tracked dan Untracked. Tracked adalah file yang ada di snapshot terakhir; file tersebut bisa dalam status ‘unmodified’, ‘modified’, dan ‘staged’. File yang memiliki keadaan ‘untracked’ adalah file apapun di working tree yang tidak ada di snapshot terakhir dan tidak ada di staging area. Saat pertama kali mengkloning repositoru, semua file anda akan dalam keadaan ‘tracked’ dan ‘unmodified’ karena anda baru memeriksanya dan belum mengedit apapun. Saat anda mengubah file, Git melihatnya sebagai modifikasi, karena anda telah mengubahnya sejak commit terakhir anda. Anda bisa melakukan ‘stage’ pada file yang telah dimodifikasi ini dan kemudian melakukan commit pada semua file yang sudah di ‘stage’. Siklus hidup status dan keadaan dari file pada Git diilustrasikan pada Gambar dibawah.

Diagram, timeline

Description automatically generated

Beberapa vocabulary penting dari Git adalah sebagai berikut:

Graphical user interface

Description automatically generated

Graphical user interface, text, application

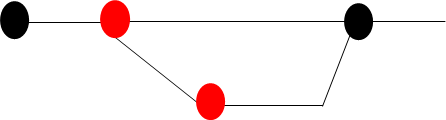
Description automatically generated

Keterangan:

* nilai **HASH** adalah kumpulan kata sepanjang 40 karakter dari karakter heksadesimal (0-9 dan a-f) dan dihitung berdasarkan isi dari sebuah berkas atau struktur direktori dalam Git. Nilai hash menjadi penanda unik dari tiap commit.
* Pada Git juga terdapat istilah **HEAD**, yaitu penunjuk ke referensi cabang saat ini.
* Commit biasanya dilambangkan dengan simbol titik.

Merge dalam Git merupakan perintah untuk menggabungkan dua cabang/branch. Ada 2 tipe merge:

1. Fast-forward: terdapat direct path



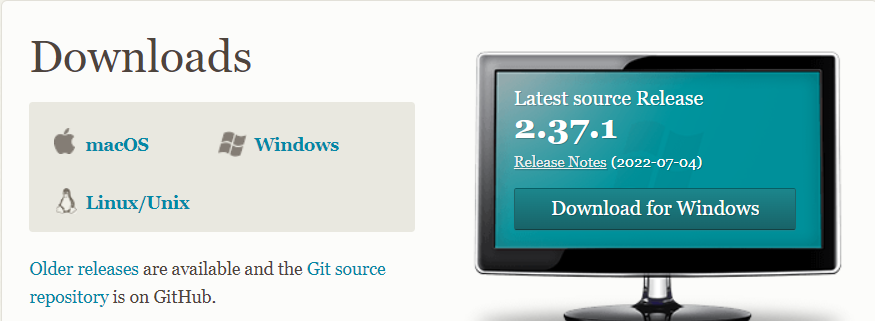
1. Three-way merge: tidak terdapat direct path, harus membuat commit baru



* 1. **Instalasi Git**

Langkah pertama yang dilakukan adalah mempersiapkan environment lokal Git pada OS baik Windows/MacOs/Linux. Pada modul praktikum ini menggunakan windows sebagai OS. Langkah-langkah penginstalan Git adalah sebagai berikut:

1. Unduh Git melalui tautan <https://git-scm.com/downloads>, pilih OS dan/atau bit sesuai dengan OS dan/atau bit pada laptop/komputer anda.



1. Setelah terunduh, jalankan installer Git

A picture containing application

Description automatically generated

1. Klik Next untuk melanjutkan proses penginstalan dan lakukan pemilihan dan pengaturan komponen atau pilih sesuai rekomendasi Git. Ketika diminta untuk memilih teks editor pilih teks editor yang kamu gunakan untuk Git. Modul ini menggunakan Visual Studio Code sebagai teks editor.
2. Jika proses instalasi telah selesai maka akan muncul direktori Git pada Drive C:\Program Files\
3. Untuk membuka Git Bash, anda bisa mencari menulis Git Bash pada kolom search pada taskbar windows. Pilih Git Bash. Git Bash merupakan aplikasi khusus untuk Microsoft Windows yang menyediakan emulasi untuk Git command line. Anda bisa mengoperasikan perintah-perintah Git pada GitBash. Selain Git Bash anda juga bisa menggunakan command-prompt bawaan dari windows.

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Untuk memastikan command-prompt bisa menjalankan perintah git ketik **git --version**. Jika hasilnya menunjukkan versi dari git maka command-prompt telah bisa dipakai untuk melakukan perintah git.

Text

Description automatically generated

* 1. **Percobaan Praktikum VCS pada Git**

Pada bagian ini, kita akan mengakses Git menggunakan console / command prompt agar kita semakin mengerti konsep dari VCS pada Git. Berikut beberapa operasi dasar dari Git:

|  |  |
| --- | --- |
| **Operasi dasar git** | **Fungsi** |
| **$ git config** | mengatur konfigurasi Git pada tingkat proyek global atau lokal |
| **$ git init** | menginisialisasi repo git di komputer |
| **$ git add <file(s)>** atau **git add .** | menambahkan file pada staging area |
| **$ git status** | mengetahui status pada repository |
| **$ git commit** | menyimpan perubahan |
| **$ git branch** | membuat branch/cabang |
| **$ git help** | mencari tahu beberapa perintah dalam git |
| **$ git checkout** | pindah ke branch/cabang, atau commit tertentu |
| **$ git log** | melihat apa saja yang sudah pernah dilakukan |
| **$ git clone** | membuat kloning atau salinan repo di direktori baru/di lokasi lain |
| **$ git push** | mengunggah konten di repository lokal ke repository remote |
| **$ git pull** | mengambil dan mengunduh konten dari repositoriremote dan segera perbarui repositori lokal agar sesuai dengan konten itu |
| **$ git remote** | membuat, melihat, dan menghapus koneksi ke repositori lain |
| **$ git merge** | mengintegrasikan perubahan dari branchcabang lain |
| **$ git diff** | menjalankan fungsi diff pada sumber data Git |
| **$ git remove** | menghapus file yang ditracked dari indeks Git |

**Catatan**: jika pada proses penginstalan Git, anda tidak memberikan nama untuk cabang/branch utama, maka cabang/branch utama akan diberi nama ‘master’.

Berikut langkah-langkah untuk menggunakan Git:

1. Inisialisasi Repo

Langkah-langkah untuk menginisialisasi Repository adalah sebagai berikut:

* Membuat folder yang akan menjadi repository
* Masuk ke folder yang dimaksud dengan menggunakan Git Bash atau command-prompt
* Menginisialisasi repository pada folder tersebut dengan menggunakan perintah **git init**

Text

Description automatically generated

Setelah mengetik **git init**, folder tadi akan otomatis menjadi repo, segala perubahan akan dipantau oleh git, dengan branch utama bernama master.

Setelah inisialisasi, jika kita buka folder tersebut akan muncul folder tersembunyi yang bernama ‘.git’.

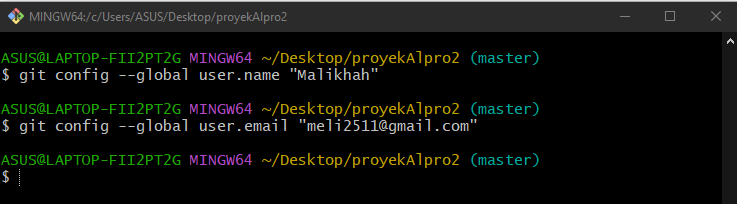
Graphical user interface, application

Description automatically generated

1. Mengisi identitas di Git dengan menggunakan perintah:

**$git config --global user.name “<nama pemilik repo>”**

**$git config --global user.email <email pemilik repo>**

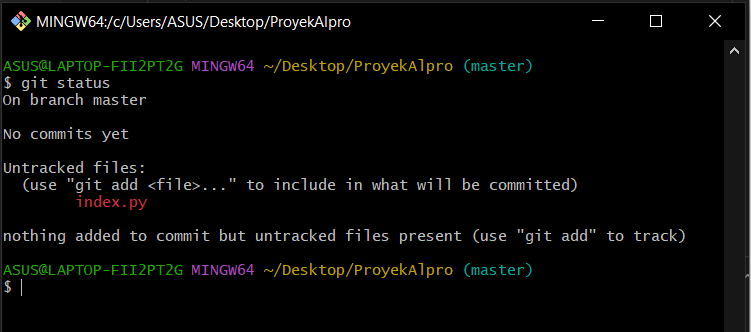


1. Setelah menginisialisasi git dan mengisi konfigurasi identitas, langkah selanjutnya adalah membuat file di dalam folder tersebut. Misalkan kita mendapat tugas membuat file python yang bisa memeriksa apakah karakter yang kita masukkan angka atau tidak. Buka code editor (ex: Notepad++, Visual Studio Code, Sublime, dll). Modul ini menggunakan Visual Studio Code (VSC) sebagai code editor. Ingat! VSC adalah code editor, sedangkan VCS adalah Version Control System. Setelah membuka VSC, pilih File lalu pilih Open Folder, pilih Folder repository yang sudah dibuat. Buat file baru, misalkan index.py dan tulis kode untuk memeriksa apakah suatu karakter angka atau bukan.

Text

Description automatically generated

1. Setelah itu coba ketik **git status** pada git bash



Pada git bash, akan muncul pesan bahwa anda sedang berada di branch ‘master’ dan anda belum pernah melakukan commit. Ada 1 file yang berstatus ‘untracked’.

1. Jika anda sudah yakin dengan pekerjaan anda dan agar file yang anda buat selalu dipantau oleh Git, maka pada Git Bash ketik **git add <nama\_file>**atau **git add .**. kemudian setelah itu ketik **git status** pada git bash. Akan muncul pesan bahwa file ‘index.py’ sudah masuk ke staging area dan siap untuk di commit.

Text

Description automatically generated

1. Untuk melakukan commit, agar file anda tersimpan di Git Directory, ketik : **git commit -m “<pesan commit**>”. Setelah itu ketik **git status**.

Text

Description automatically generated

File ‘index.py’ telah berhasil di commit di branch master dan tidak ada file yang perlu di commit lagi, karena semua perubahan sudah di commit.

1. Ketika anda mengetik **git log**, akan muncul apa saja yang sudah pernah anda lakukan pada git directory tersebut.

Text

Description automatically generated

Keterangan git log diatas adalah sebagai berikut: commit dilakukan sekali dengan nilai hash **74f35cb143c36473c99c9bbacfc9ed536f64e549** yang merupakan nilai unik yang menandakan commit yang kita lakukan, Head saat ini berada pada branch ‘master’. Pada commit, selain terdapat kode hash unik, juga terdapat keterangan Author, kapan dilakukan perubahan, serta pesan perubahan/commit.

1. Misalkan anda mendapat tugas baru lagi, membuat program python untuk memeriksa apakah angka yang dimasukkan adalah angka ganjil atau genap, namun anda tidak ingin branch ‘master’. Dalam kasus ini, anda bisa membuat branch baru. Untuk membuat branch baru ketik **git branch <nama\_branch>**. Untuk melihat semua branch yang ada, anda bisa mengetikkan **git branch**.

Text

Description automatically generated

Pada gambar diatas, anda telah berhasil membuat branch baru dengan nama ‘coba’. Ada 2 branch dalam git directory sekarang yaitu ‘coba’ dan ‘master’. Tanda \* menunjuk ke ‘master’, tandanya bahwa head berada di branch ‘master’ (cabang yang aktif adalah cabang ‘master’)

1. Untuk berpindah ke cabang yang baru anda buat, anda bisa mengetikkan **git checkout <nama\_branch>**

Graphical user interface, text

Description automatically generated

Pada gambar diatas, anda telah berpindah ke branch ‘coba’, dan tanda \* menunjuk ke ‘coba’, tandanya bahwa head berada di branch ‘coba (cabang yang aktif adalah cabang ‘coba). Buat file baru untuk mengerjakan tugas anda yang baru, misalkan anda menyimpannya dengan nama ‘cekParitas.py’.

Text

Description automatically generated

1. Tambahkan ke staging area dan commit file baru tersebut.

Text

Description automatically generated

Pada gambar diatas, kita telah menambahkan file baru ke staging area dan melakukan commit file tersebut, semuanya dilakukan pada branch ‘coba’. Ketik **git log** untuk melihat apa saja yang sudah dilakukan pada git directory.

Text

Description automatically generated

Kita telah melakukan 2 commit, dimana satu commit dilakukan pada branch ‘master’ dan commit yang kedua dilakukan pada branch ‘coba’. Head sekarang berada pada branch ‘coba’

1. Jika anda ingin melihat log dalam bantuk graph, anda bisa mengetikkan perintah berikut:

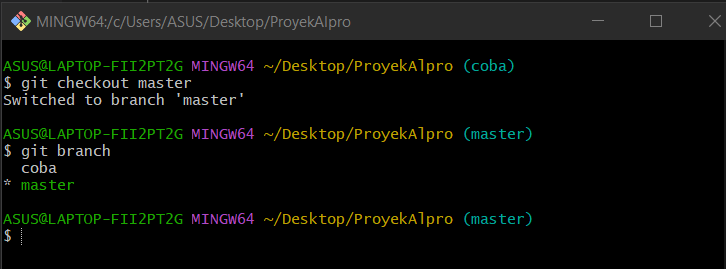
**git log --all --decorate --oneline –graph**

Text

Description automatically generated

Karena commit masih sedikit, maka graph/grafik dari commit tidak terlalu terlihat.

1. Misalkan anda sudah yakin dengan perubahan file baru tersebut dan ingin menggabungkannya (merge) ke dalam branch ‘master’. Maka anda harus checkout ke branch ‘master’ terlebih dahulu. Dengan melakukan checkout ke branch ‘master’, maka head yang sebelumnya berada pada branch ‘coba’ sekarang berpindah ke branch ‘master’.



1. Untuk melakukan merge branch ‘master’ dan ‘coba’ anda bisa mengetikkan **git merge <nama\_cabang\_yang\_ingin\_di\_merge>.** Hasilnya adalah sebagaimana Gambar dibawah. Pada Gambar dibawah, branch ‘coba’ telah berhasil di merge dengan branch ‘master’. Merge jenis ini disebut dengan Fast-forward, karena terdapat direct path antara branch ‘master’ dan branch ‘coba. Pada Fast-Forward, tidak menambahkan perintah commit, hal ini dapat dilihat, ketika kita menjalankan git status, commit masih ada 2, sebagainya bagian (j)

Text

Description automatically generated

1. Jika branch ‘coba’ sudah tidak digunakan lagi anda bisa menghapus branch ‘coba’ dengan menggunakan perintah **git branch -d <nama\_branch>**. -d merupakan perintah bahwa anda akan menghapus branch. Pada gambar di bawah, branch ‘coba’ telah berhasil dihapus.

Text

Description automatically generated

* 1. **GitHub**

Jika dengan Git, anda hanya mampu mengelola setiap perubahan secara lokal, maka dengan GitHub anda bisa mengelola file secara online. GitHub merupakan layanan Git berbasis cloud cloud (online repository). Dengan GitHub anda dapat dengan mudah bekerja denga Git tanpa melakukannya dengan menggunakan command-line atau baris perintah (CLI) karena GitHub berbasis GUI. Berikut perbedaan Git dan GitHub:

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Install software di penyimpanan lokal | Host melalui layanan cloud |
| Dikelola oleh The Linux Foundation | Diakuisisi oleh Mirosoft pada 2018 |
| Software | Service |
| Fokus pada version control dan code sharing | Fokus pada source code hosting terpusat |
| Akses offline | Akses Online |
| Tidak menggunakan fitur user management | Menggunakan user management |
| Desktop interface Bernama “Git GUI” | Desktop interface “GitHub Desktop” |
| Pesaing: Mercurial, Subversion, IBM, Rational Team, Concert, dan ClearCase | Pesaing: GitLab, Atlassian BitBucket |
| Free dan Open source | Free dan pengguna berbayar |

* 1. **Membuat Akun GitHub**

Agar bisa menggunakan dan memanfaatkan layanan yang dimiliki GitHub anda harus memiliki akun GitHub. Langkah-langkahnya adalah sebagai berikut:

1. Buka situs GitHub di <https://github.com>
2. Klik tombol “sign up” yang terdapat pada pojok kanan atas halaman
3. Isi form pendaftaran dengan lengkap
4. Masukkan kode verifikasi yang dikirim ke email yang didaftarkan
5. Lengkapi beberapa pertanyaan-pertanyaan yang diberikan oleh GitHub pada halaman web
6. Pilih “Free” untuk melanjutkan pendaftaran
7. Setelah selesai, maka anda akan menuju ke halaman dashboard GitHub

A screenshot of a computer

Description automatically generated with medium confidence

* 1. **Percobaan Praktikum GitHub**

Pada bagian ini, kita akan mencoba untuk menggunakan GitHub untuk melakukan beberapa fungsi Git. Berikut beberapa percobaan GitHub.

1. Membuat repository. Pada pojok kanan atas, pilih tanda + dan pilih ‘New Repository’.

A screenshot of a computer

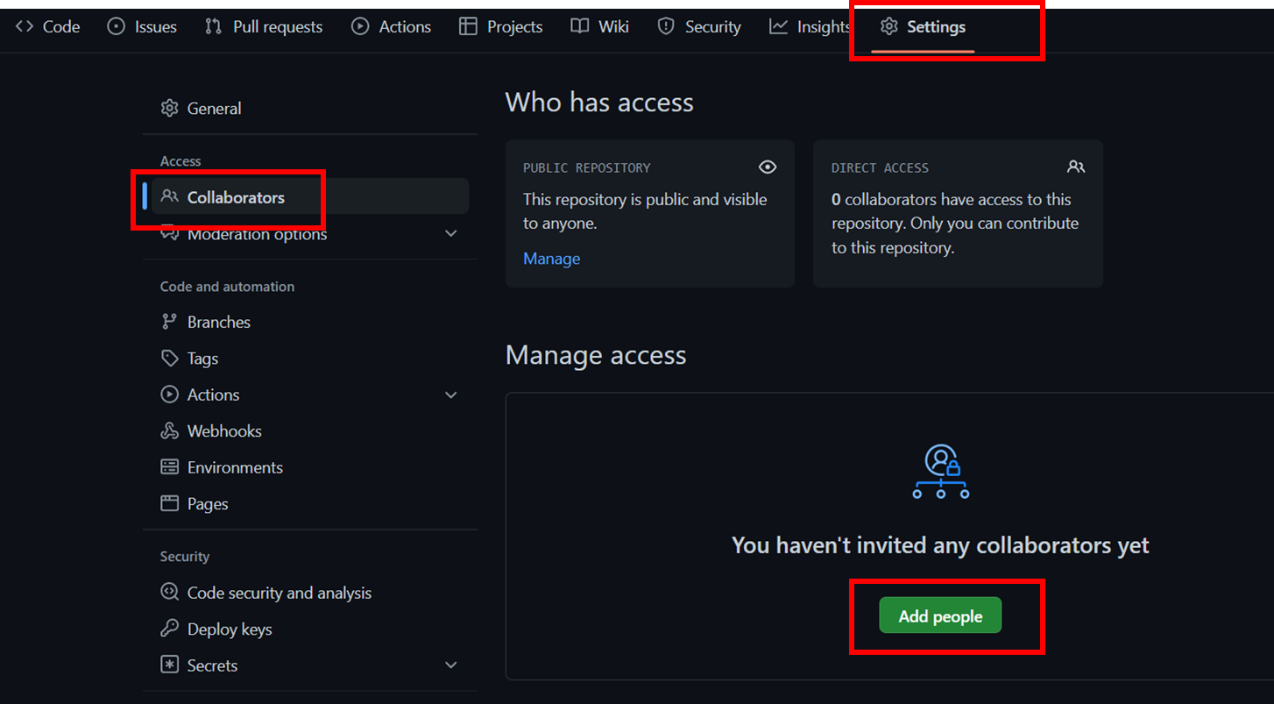
Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Setelah project tersimpan di GitHub, anda bisa memberikan atau membagikan URL project anda ke orang lain. Setiap project di GitHub dapat diakses melalui HTTPS melalui : **https://github.com/<user>/<project\_name>** atau melalui SSH : **git@github.com:<user>/<project\_name>.**

1. Menambahkan kolaborator. Pada Project anda, klik Settings 🡪 Collaborators, lalu tambahkan kolaborator untuk project anda.

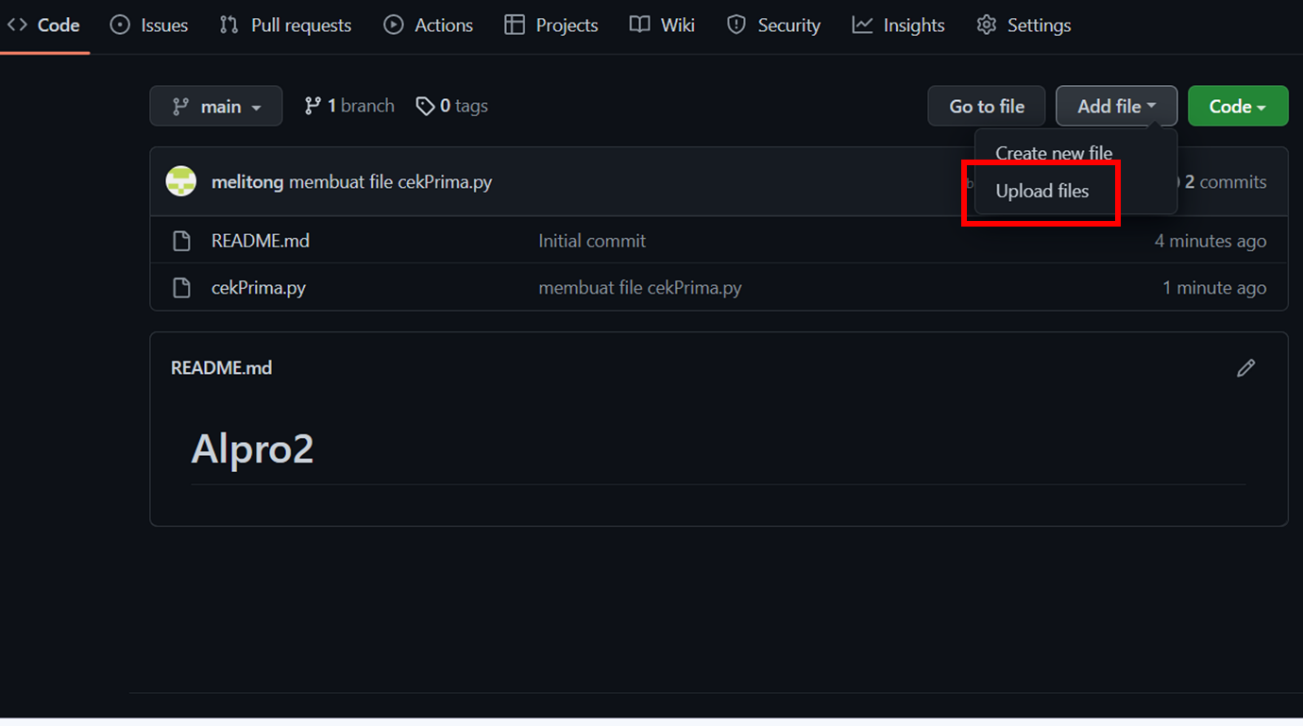


1. Membuat File baru. Pada Project Anda, pilih Add File 🡪 Create New File. Setelah mengedit file, jangan lupa untuk melakukan commit agar perubahan tersimpan di GitHub.

Graphical user interface, text, application, email, Teams

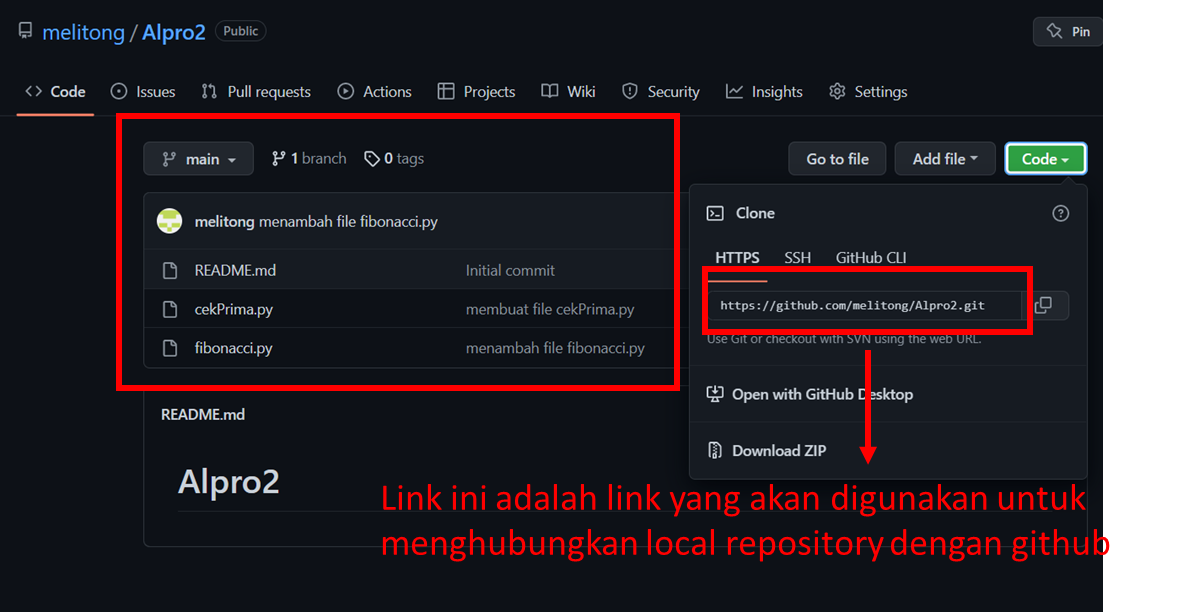
Description automatically generated

1. Mengunggah file dari lokal. Pada GitHub, anda juga bisa mengunggah file dari komputer anda. Pilih Add File 🡪 Upload Files.



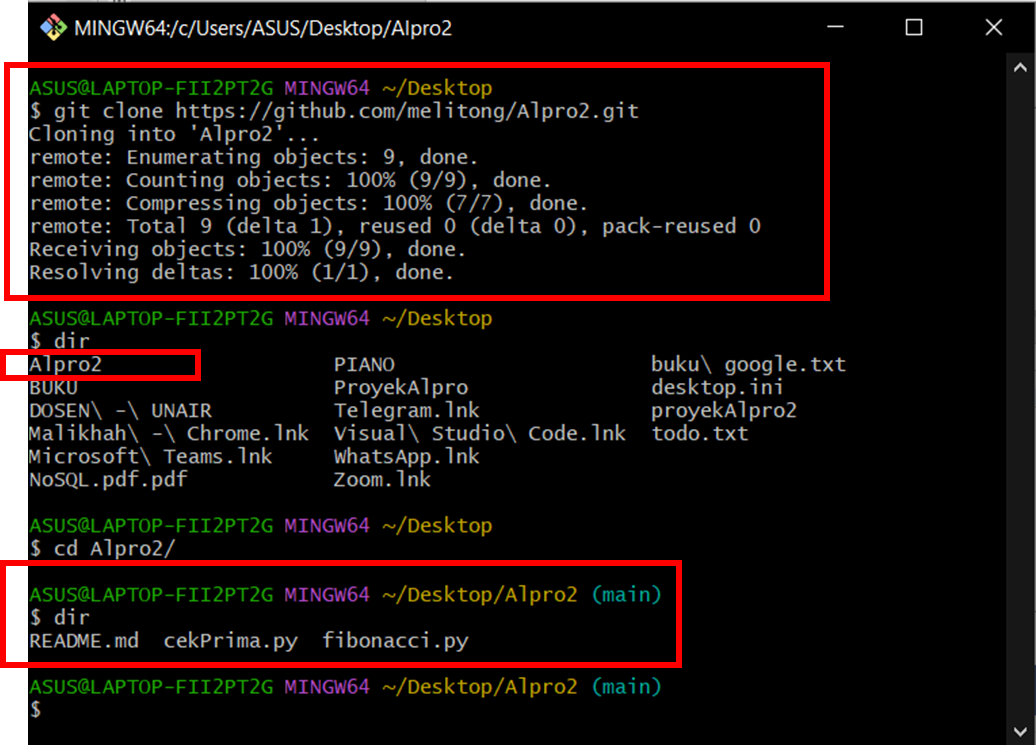
1. Melakukan duplikasi atau clone GitHub

Salah satu fitur GitHub adalah integrasi dengan aplikasi Git yang dapat memudahkan kita untuk melakukan proses clone atau duplikasi sebuah konten pada repositori Github. Dengan melakukan clone GitHub, semua file di repo GitHub akan ada di local repository.



Untuk melakukan clone, anda harus menentukan dimana anda ingin menyimpan local repository, kemudian buka Git Bash dan ketik: **git clone <link\_GitHub>.**

Setelah melakukan perintah git clone, pada folder tersebut akan ada folder baru dengan nama yang sama dengan project GitHub anda. Semua file yang ada di project GitHub telah tersimpan di local repository.



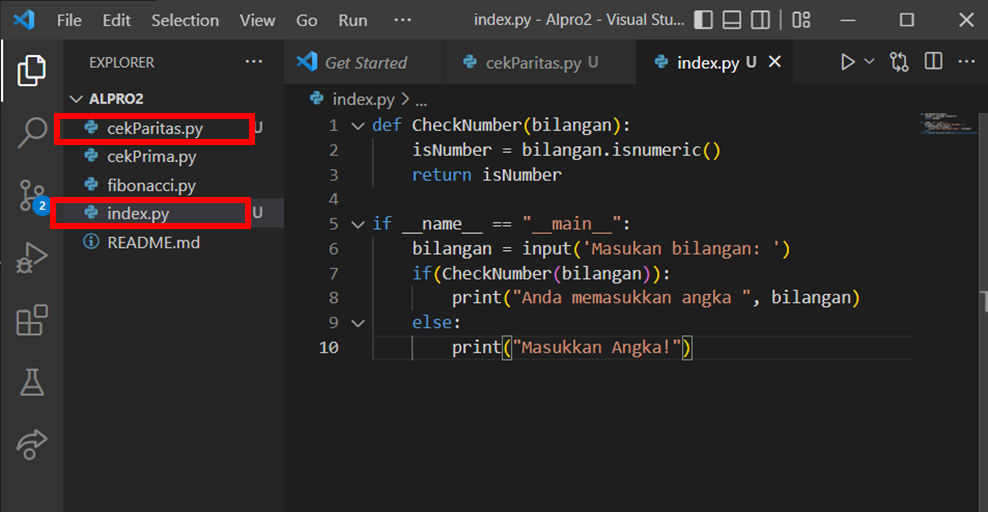
1. Push file dari repository lokal

Setelah melakukan clone GitHub pada local repository, anda bisa melakukan push file (mengunggah file) dari local repository ke GitHub. Buat dan simpanlah file baru di local repository, kemudian ketik pada Git Bash:

**git add .**

**git commit –m “<pesan\_commit>”**

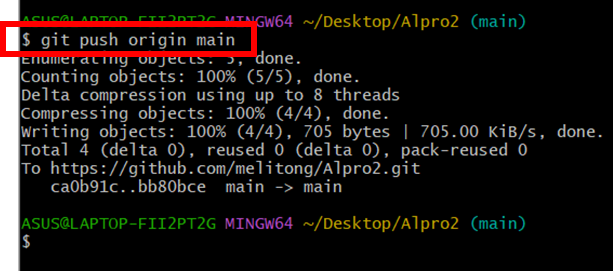
Setelah melakukan perintah diatas, perubahan akan tersimpan di local repository.



Untuk melakukan push (mengunggah file) dari local repository ke GitHub, ketik perintah berikut pada Git Bash:

**git push origin <nama\_branch>**

ketika anda melakukan push ke GitHub, GitHuub akan memverifikasi bahwa anda memiliki wewenang untuk melakukan perubahan ke GitHub. Hasilnya dapat dilihat pada Gambar di bawah. Terdapat pesan bahwa anda telah melakukan push ke GitHub repository (remote).

****

Ketika dilihat di GitHub, 2 file baru yang anda push sudah tersimpan di GitHub.

**A screenshot of a computer screen

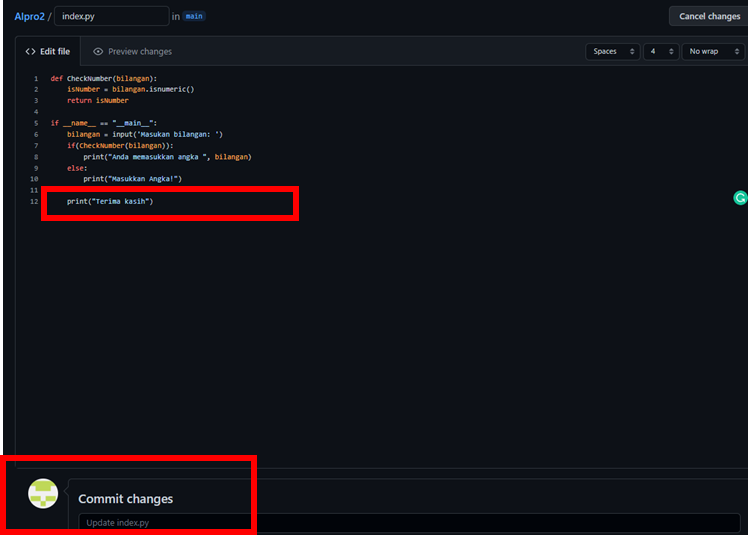
Description automatically generated with medium confidence**

1. Mengubah file melalui GitHub

Anda juga bisa mengubah file melalui GitHub, Pilih file yang ingin diubah lalu pilih Edit this file.

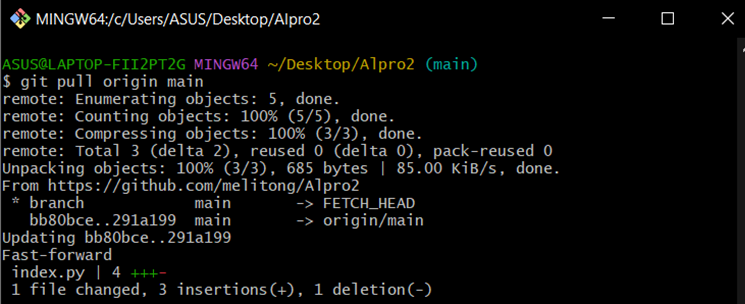


Lakukan perubahan pada file lalu commit perubahan tersebut.

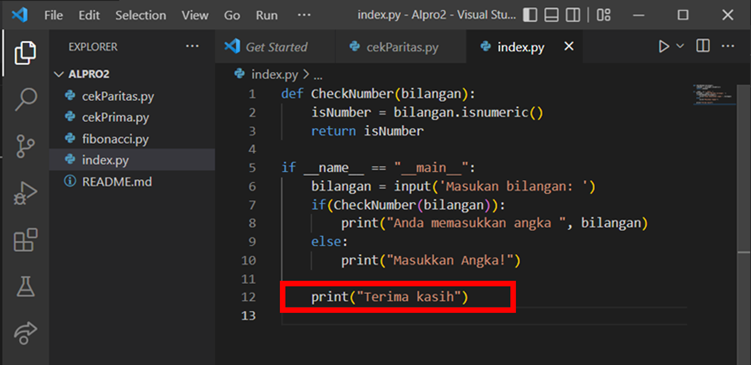


1. Pull file ke repository lokal

Sebelum anda akan mengubah file di local repository, ada baiknya anda melakukan pull file (mendownload file) dari GitHub ke local repository. Hal tersebut dilakukan untuk menghindari adanya conflict. Untuk melakukan pull file ke local repository, ketik perintah berikut pada Git Bash : **git pull origin <nama\_branch>.**

****

Pada bagian (g) anda merubah sebuah file melalui GitHub. Setelah melakukan pull, perubahan itu akan tersimpan di local repository.



**Tugas Praktikum:**

Setelah anda memahami Git dan GitHub dan mempraktekkannya, kerjakanlah tugas praktikum berikut:

1. Buatlah program kalkulator sederhana python dan lakukan perintah-perintah Git untuk melakukan manajemen program, dan simpan repository anda ke GitHub.
2. Buatlah laporan yang berisi dokumentasi dan penjelasan pengerjaan tugas yang menerapkan Git dan menggunakan GitHub secara urut (tulis semua perintah/command Git yang dilakukan secara urut dan sertakan screen shootnya).
3. Laporan juga memuat code snippets dari program yang disimpan pada GitHub.