MAKALAH

Git & GitHub

PEMROGRAMAN WEB 1

DOSEN PENGAMPU:

Dr. Harja Santanapurba, M.Kom

Novan Alkaf B. S., S.Kom., M.T

Ihdalhubbi Maulida, M.Kom

DISUSUN OLEH:

Rizqa Aulia faiha	2410131220008
Rizqa Oktavia Ramadhani	2410131120010
Siti Noor Hayati	2410131120012

PROGRAM STUDI PENDIDIKAN KOMPUTER

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

UNIVERSITAS LAMBUNG MANGKURAT

BANJARMASIN

2025

DAFTAR ISI

PEMBAHASAN

1. Git init

```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/aaaaa
$ git init
```

Perintah git init memiliki peran yang sangat penting dalam alur kerja Git karena berfungsi untuk menginisialisasi sebuah repository Git baru di direktori lokal. Ketika seseorang menjalankan git init, Git akan membuat sebuah subdirektori tersembunyi bernama .git, yang berisi seluruh informasi penting terkait repository tersebut, seperti riwayat commit, konfigurasi, informasi branch, dan semua data yang diperlukan untuk sistem kontrol versi berjalan dengan baik. Tanpa adanya folder .git, Git tidak akan mampu melacak perubahan apa pun yang terjadi di dalam proyek. Dengan kata lain, git init mengubah sebuah folder biasa menjadi sebuah repository Git yang siap untuk melacak setiap perubahan file, mencatat histori pengembangan, serta mengelola berbagai cabang (branch) pengembangan.

Penggunaan git init biasanya menjadi langkah awal saat memulai proyek baru yang belum pernah dikelola oleh Git sebelumnya. Setelah repository berhasil diinisialisasi, pengguna bisa mulai menambahkan file ke dalam staging area menggunakan git add, melakukan penyimpanan perubahan dengan git commit, membuat branch baru untuk fitur-fitur tertentu, serta memanfaatkan seluruh kekuatan sistem version control Git. Selain itu, git init juga berguna dalam situasi lain, misalnya ketika ingin mengelola proyek yang sudah ada tanpa version control sebelumnya, sehingga proyek tersebut bisa langsung dipantau dan dikendalikan oleh Git.

2. Git Clone

```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d
$ git clone https://github.com/rizqa-oktavia-ramadhani01/rizqaduo-github.io
Cloning into 'rizqaduo-github.io'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 9 (delta 2), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), 5.31 KiB | 906.00 KiB/s, done.
Resolving deltas: 100% (2/2), done.
```

Perintah git clone digunakan untuk mengunduh atau menyalin sebuah repository Git yang sudah ada dari server atau lokasi remote ke dalam direktori lokal. Dengan kata lain, perintah ini memungkinkan pengguna untuk membuat salinan lengkap dari repository Git yang ada, termasuk seluruh riwayat commit, branch, dan file yang ada di dalamnya, sehingga pengguna bisa mulai bekerja dengan proyek tersebut di komputer lokal mereka.

Ketika menjalankan git clone, Git akan menyalin seluruh isi repository beserta semua informasi yang dibutuhkan untuk bekerja dengan repository tersebut secara penuh. Hal ini termasuk semua commit sebelumnya, cabang-cabang (branch) yang ada, dan konfigurasi remote repository. Ini sangat berguna ketika seseorang ingin berkolaborasi dalam proyek yang sudah ada atau ketika ingin mendapatkan salinan lokal dari repository untuk dipelajari atau dimodifikasi.

Sebagai contoh, jika Anda ingin bekerja pada sebuah proyek yang ada di GitHub, Anda bisa menggunakan git clone dengan URL dari repository tersebut. Git akan menyalin seluruh isi repository ke direktori lokal Anda, sehingga Anda bisa mulai mengedit file, melakukan commit, atau berkolaborasi dengan tim pengembang lainnya.

3. Git status

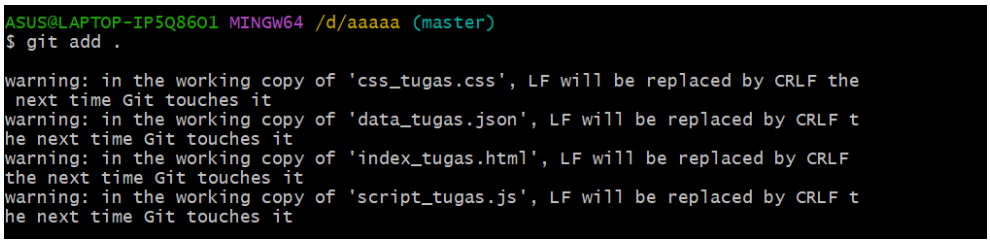
```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/rizqaduo-github.io (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Perintah git status digunakan untuk menampilkan informasi terbaru tentang keadaan repository Git Anda. Ini adalah salah satu perintah pertama yang

sering digunakan saat bekerja dengan Git karena memberikan gambaran umum mengenai apa yang terjadi pada file dalam proyek. Dengan menggunakan git status, Anda dapat melihat file yang sudah dimodifikasi namun belum ditambahkan ke staging area, yang berarti file tersebut belum siap untuk di-commit.

Git status juga menampilkan file-file yang sudah berada di staging area, yaitu perubahan yang telah Anda tandai dan siap untuk dimasukkan ke dalam commit berikutnya. Tak hanya itu, perintah ini memberi tahu Anda cabang (branch) mana yang sedang aktif serta status sinkronisasi antara repository lokal dan remote. Secara keseluruhan, git status sangat membantu untuk memverifikasi kondisi file dalam proyek sebelum melanjutkan ke langkah berikutnya, seperti menambahkan file ke staging area, melakukan commit, atau menghapus file yang tidak diperlukan.

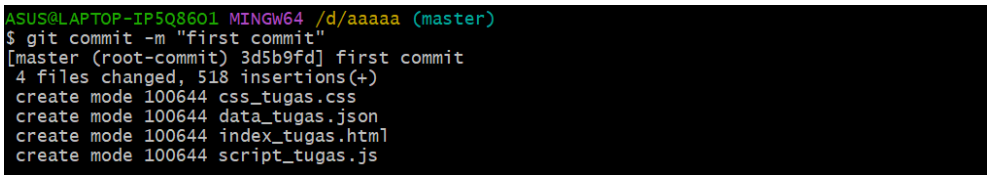
4. Git add



Perintah git add digunakan untuk menambahkan perubahan pada file ke dalam staging area, yaitu tempat sementara di mana perubahan disiapkan sebelum dimasukkan ke dalam commit. Perintah ini sangat penting karena memungkinkan pengembang untuk memilih secara selektif perubahan mana yang ingin dimasukkan ke dalam commit berikutnya. Misalnya, jika Anda telah melakukan perubahan pada beberapa file namun hanya ingin menyertakan sebagian dari perubahan tersebut dalam commit, Anda dapat menggunakan git add untuk memilih file yang relevan. Proses penggunaannya dimulai setelah Anda mengubah file—pada tahap ini, perubahan belum tercatat oleh Git. Untuk menyiapkan perubahan tersebut agar dapat di-commit, Anda perlu menambahkan file ke staging area

menggunakan perintah git add. Setelah file berada di staging area, barulah file tersebut siap untuk di-commit.

5. Git commit



Perintah git commit digunakan untuk menyimpan perubahan yang telah ditambahkan ke staging area ke dalam repository lokal Git. Commit berfungsi sebagai titik referensi yang merekam snapshot dari proyek pada saat itu, memungkinkan pengembang untuk melacak dan mengelola riwayat perubahan dengan lebih mudah. Setiap commit dapat disertai dengan pesan yang menjelaskan perubahan yang dilakukan. Pesan ini sangat penting, terutama dalam proyek kolaboratif, karena memberikan konteks yang jelas tentang tujuan perubahan tersebut. Proses commit dimulai setelah file yang dimodifikasi ditambahkan ke staging area menggunakan git add. Kemudian, dengan menjalankan git commit, perubahan tersebut akan dicatat dalam riwayat proyek, dan Git akan memberikan ID unik (hash) untuk setiap commit. ID ini memudahkan pengguna untuk menelusuri, merujuk, atau bahkan mengembalikan kondisi proyek ke commit tertentu jika diperlukan. Penulisan pesan commit yang jelas dan deskriptif sangat penting untuk menjaga dokumentasi perubahan dalam proyek jangka panjang.

Sebagai contoh, perintah git commit -m "Menambahkan fitur login pengguna" akan menciptakan commit dengan pesan yang menjelaskan bahwa perubahan yang dilakukan terkait dengan penambahan fitur login pada aplikasi. Pesan semacam ini memberikan gambaran singkat namun informatif tentang maksud dari commit tersebut. Selain opsi dasar, terdapat pula opsi tambahan seperti git commit -a, yang secara otomatis menambahkan semua file yang telah dimodifikasi ke staging area sebelum melakukan commit, serta git commit --amend, yang digunakan untuk mengubah commit terakhir, baik untuk memperbaiki pesan commit maupun menambahkan file yang terlupa.

Dengan demikian, git commit tidak hanya menyimpan perubahan, tetapi juga memainkan peran penting dalam pengelolaan versi dan dokumentasi proyek.

6. Git push

Perintah git push digunakan untuk mengirimkan commit yang ada di repository lokal ke repository remote, seperti GitHub, GitLab, atau Bitbucket. Setelah Anda melakukan commit, perubahan tersebut hanya tersimpan secara lokal di komputer Anda. Agar perubahan tersebut dapat diakses oleh orang lain atau disimpan secara terpusat di remote repository, Anda perlu menggunakan git push. Perintah ini akan mengirimkan semua commit dari branch lokal ke branch yang sama pada remote repository. Proses ini memastikan bahwa versi terbaru dari proyek Anda tersedia di remote dan dapat diakses atau di-clone oleh kolaborator lainnya. Jika branch lokal yang Anda gunakan belum ada di remote repository, maka Git akan secara otomatis membuat branch baru dengan nama yang sama di remote tersebut.

Sebagai contoh, perintah git push origin master digunakan untuk mengirimkan commit dari branch master di repository lokal ke branch master di remote repository yang bernama origin. Dalam banyak kasus, origin adalah nama default yang diberikan untuk remote repository pertama yang dikloning atau ditambahkan, namun Anda dapat menggantinya dengan nama remote lain sesuai kebutuhan. Dengan menggunakan git push, dapat menjaga sinkronisasi antara repository lokal dan remote, serta memastikan bahwa setiap perubahan yang telah dibuat dapat dicadangkan dan dibagikan secara efektif.

7. Git pull

```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/rizqaduo-github.io (master)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 952 bytes | 68.00 KiB/s, done.
From https://github.com/rizqa-oktavia-ramadhani01/rizqaduo-github.io
   fd393ee..1e3a6bf  master     -> origin/master
```

```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/rizqaduo-github.io (master)
$ git branch
* master

ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/rizqaduo-github.io (master)
$ git pull origin master
From https://github.com/rizqa-oktavia-ramadhani01/rizqaduo-github.io
 * branch            master     -> FETCH_HEAD
Updating fd393ee..1e3a6bf
Fast-forward
 index_tugas.html | 3 ---
 1 file changed, 3 deletions(-)
```

Perintah git pull digunakan untuk menarik perubahan terbaru dari remote repository dan langsung menggabungkannya dengan repository lokal Anda. Perintah ini sangat berguna untuk memastikan bahwa repository lokal tetap sinkron dengan versi terbaru yang ada di remote, terutama ketika bekerja dalam tim. Saat Anda menjalankan git pull, Git akan melakukan dua langkah secara otomatis: pertama, git fetch untuk mengunduh perubahan dari remote repository, dan kedua, git merge untuk menggabungkan perubahan tersebut dengan branch lokal Anda. Dengan demikian, Anda tidak perlu menjalankan kedua perintah itu secara terpisah.

Misalnya, perintah git pull origin master akan mengambil pembaruan dari branch master di remote repository bernama origin, lalu menggabungkannya dengan branch master di repository lokal Anda. Ini penting dilakukan secara rutin, terutama sebelum melakukan perubahan besar, agar Anda bekerja di atas versi kode terbaru dan menghindari konflik yang tidak perlu. Jika terdapat perubahan yang bertentangan antara kode lokal dan remote, Git akan memberi tahu Anda agar konflik tersebut diselesaikan secara manual sebelum proses merge selesai. Dengan menggunakan git pull, dapat menjaga konsistensi kode antara semua kontributor dalam proyek.

8. Git branch

```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/rizqaduo-github.io (master)
$ git branch
* master
```

Git branch adalah perintah dalam Git yang digunakan untuk mengelola cabang (branch) dalam sebuah repository. Dalam konteks pengembangan perangkat lunak, branch atau cabang berfungsi seperti jalur paralel dari pengembangan proyek. Dengan menggunakan branch, seorang pengembang bisa membuat versi terpisah dari kode utama (biasanya disebut main atau master) untuk mengerjakan fitur baru, melakukan perbaikan bug, atau bereksperimen tanpa mengganggu kode yang sudah stabil. Hal ini memungkinkan tim pengembang untuk bekerja secara independen pada berbagai aspek proyek secara bersamaan. Misalnya, satu orang bisa mengerjakan fitur baru di satu branch, sementara orang lain memperbaiki bug di branch lain, dan ketika semuanya sudah selesai serta stabil, perubahan tersebut bisa digabungkan kembali ke branch utama menggunakan merge.

Secara teknis, ketika Anda menjalankan git branch, Git akan menampilkan daftar semua cabang yang ada di repository lokal, dan menunjukkan cabang mana yang sedang aktif. Untuk membuat branch baru, Anda bisa menggunakan perintah git branch <nama_cabang>, sedangkan untuk berpindah ke branch lain, digunakan git checkout <nama_cabang>. Namun, sejak Git versi 2.23, Git memperkenalkan perintah baru yaitu git switch sebagai cara yang lebih eksplisit dan mudah digunakan untuk berpindah branch. Dengan branch, pengembang dapat mengisolasi pekerjaan mereka, menghindari konflik, dan memastikan bahwa perubahan yang dilakukan tidak merusak fungsionalitas yang sudah ada. Branch juga sangat penting dalam sistem kerja kolaboratif menggunakan platform seperti GitHub, karena memungkinkan pull request dibuat dari branch tertentu, untuk ditinjau dan digabungkan ke branch utama setelah proses review selesai.

Dengan kata lain, git branch adalah alat penting dalam Git yang memberikan fleksibilitas dan kontrol penuh terhadap proses pengembangan proyek. Ini mendukung pola kerja yang terstruktur, memungkinkan pengembangan paralel, dan meminimalisasi risiko kesalahan dalam kode utama. Dalam

praktiknya, manajemen branch yang baik sangat menentukan kelancaran alur kerja tim pengembang perangkat lunak, terutama dalam proyek skala besar yang melibatkan banyak kontributor.

9. Get Checkout

```
ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/aaaaa (master)
$ git checkout -- index_tugas.html

ASUS@LAPTOP-IP5Q86O1 MINGW64 /d/aaaaa (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

Git checkout adalah salah satu perintah penting dalam Git yang digunakan untuk berpindah antar branch, commit, atau bahkan file tertentu di dalam sebuah repository. Perintah ini memungkinkan pengguna untuk "melompat" ke titik tertentu dalam riwayat pengembangan proyek—baik itu ke branch lain, ke commit sebelumnya, atau mengambil versi file tertentu dari commit terdahulu. Salah satu fungsi utamanya adalah untuk berpindah dari satu branch ke branch lainnya. Misalnya, jika Anda sedang berada di branch main dan ingin mulai mengerjakan fitur baru di branch fitur-1, Anda dapat mengetik git checkout fitur-1 untuk berpindah ke branch tersebut. Ini sangat berguna ketika Anda bekerja secara paralel pada beberapa fitur atau perbaikan bug dalam satu proyek.

Selain itu, git checkout juga bisa digunakan untuk mengembalikan file ke kondisi tertentu. Misalnya, jika Anda telah melakukan perubahan pada sebuah file tetapi belum menyimpannya ke dalam staging area atau melakukan commit, dan Anda ingin membatalkan perubahan tersebut, Anda dapat menggunakan perintah seperti git checkout -- nama_file untuk mengembalikan file ke versi terakhir yang dikomit. Ini membantu mencegah kesalahan yang mungkin terjadi selama proses pengembangan. Di sisi lain, jika Anda ingin melihat atau bekerja dengan versi lama dari suatu commit,

Anda bisa menggunakan git checkout <commit_id> untuk berpindah ke commit tersebut dalam mode "detached HEAD", yang berarti Anda sedang melihat revisi tertentu tanpa berada di cabang aktif mana pun. Dalam mode ini, perubahan yang Anda buat tidak akan tercatat dalam branch yang ada kecuali Anda membuat branch baru dari posisi tersebut.

Meskipun git checkout sangat powerful dan fleksibel, penggunaannya agak kompleks karena menggabungkan berbagai fungsi (berpindah branch, mengembalikan file, dan mengecek commit). Oleh karena itu, Git sejak versi 2.23 memperkenalkan perintah baru yang lebih spesifik seperti git switch untuk berpindah branch dan git restore untuk mengembalikan file, dengan tujuan menyederhanakan pengalaman pengguna. Meski begitu, git checkout tetap menjadi perintah yang luas digunakan karena fleksibilitasnya. Dalam praktik sehari-hari, pemahaman yang baik tentang git checkout sangat penting agar pengembang dapat berpindah antar konteks kerja dengan aman, mengeksplorasi perubahan tanpa risiko, dan menjaga struktur kerja yang terorganisir dalam sistem kontrol versi Git.

KESIMPULAN

Git adalah sistem kontrol versi terdistribusi yang digunakan untuk melacak perubahan pada file dan mengelola riwayat pengembangan proyek perangkat lunak secara efisien. Dengan Git, pengembang dapat menyimpan versi-versi berbeda dari proyek, bekerja secara paralel melalui cabang (branch), serta menggabungkan perubahan dari berbagai sumber tanpa kehilangan riwayat kerja. Git bekerja secara lokal, artinya semua data riwayat dan pengelolaan versi disimpan di komputer masing-masing pengguna. Sementara itu, GitHub adalah layanan berbasis web yang menyediakan penyimpanan dan manajemen repository Git di cloud.

GitHub memungkinkan kolaborasi antar-pengembang dengan menyediakan fitur tambahan seperti issue tracking, pull request, code review, serta integrasi dengan berbagai layanan CI/CD. Perbedaan utama antara Git dan GitHub adalah bahwa Git merupakan alat untuk mengelola versi proyek di komputer lokal, sedangkan GitHub adalah platform hosting yang memfasilitasi penyimpanan, kolaborasi, dan distribusi proyek Git melalui internet. Singkatnya, Git adalah alatnya, dan GitHub adalah tempat untuk menyimpan dan berbagi proyek Git secara online.

1. Git init: Digunakan untuk membuat repository Git baru di direktori lokal. Ini menginisialisasi folder sebagai repository yang dapat mulai melacak perubahan.
2. Git clone: Digunakan untuk mengunduh salinan repository Git dari remote (misalnya GitHub) ke komputer lokal. Perintah ini menciptakan salinan lengkap dari repository beserta riwayat dan cabang-cabangnya.
3. Git status: Menampilkan status file dalam repository lokal, termasuk file yang dimodifikasi, file baru yang belum dilacak, dan perubahan yang sudah di-staging.

4. Git add: Digunakan untuk menambahkan file yang telah dimodifikasi ke staging area, yang berarti file tersebut siap untuk di-commit.
5. Git commit: Menyimpan perubahan yang ada di staging area ke dalam riwayat repository lokal, dengan opsi untuk menambahkan pesan yang menjelaskan perubahan tersebut.
6. Git push: Mengirimkan commit yang ada di repository lokal ke remote repository, memperbarui versi proyek di server cloud seperti GitHub.
7. Git pull: Menarik perubahan terbaru dari remote repository dan menggabungkannya dengan repository lokal. Ini mengunduh perubahan dan mengintegrasikan perubahan tersebut ke dalam branch lokal.
8. Git branch & checkout: Git branch digunakan untuk membuat atau menampilkan cabang-cabang dalam repository. Git checkout digunakan untuk berpindah antara cabang atau memulihkan file yang hilang ke kondisi sebelumnya.

DAFTAR PUSTAKA

[1] S. Chacon and B. Straub, *Pro Git*, 2nd ed. New York, NY: Apress, 2014. [Online]. Available: https://git-scm.com/book/en/v2

[2] J. Loeliger and M. McCullough, *Version Control with Git*, 2nd ed. Sebastopol, CA: O'Reilly Media, 2012.

[3] Git SCM, "Git Documentation," [Online]. Available: https://git-scm.com/doc. [Accessed: 17-Jun-2025].

[4] GitHub, "GitHub Docs," [Online]. Available: https://docs.github.com. [Accessed: 17-Jun-2025].

[5] R. Ferdiana, *Belajar Git dan GitHub untuk Pemula*, 2020. [Online]. Available: https://github.com/ridwanferdiana/git-github-pemula