Makalah Git Hub

Pemograman Web 1

Disusun Oleh:

Muhammad jainuddin	(2410131210012)
Mardiyah	(2410131320008)
Mufidah	(2410131220020)

Kelas A2

Dosen Pengampu:

Dr. Harja Santanapurba, M.Kom
Novan Alkaf B. S., S.Kom., M.T
Ihdalhubbi Maulida, M.Kom

UNIVERSITAS LAMBUNG MANGKURAT

FAKULTAS KEGURUAN DAN ILMU PENDIDIKAN

PENDIDIKAN ILMU KOMPPUTER

2025

Daftar isi

Pembahasan

1. Git init

```
1. git init
Menginisialisasi folder sebagai repository Git.
```

Perintah git init adalah langkah pertama dalam membuat repository Git di dalam folder proyek kita. Saat kita menjalankan perintah ini, Git akan membuat sebuah folder tersembunyi bernama .git yang menyimpan semua informasi histori dan konfigurasi proyek. Dengan kata lain, folder biasa akan diubah menjadi repository Git yang siap digunakan untuk pelacakan versi. Biasanya kami menjalankan perintah ini di terminal setelah membuka folder proyek dengan perintah: git init. Setelah berhasil, Git akan mulai mencatat setiap perubahan file dalam folder tersebut. Dalam praktik tugas ini, kami menginisialisasi folder tugas kami agar semua perubahan kode bisa dicatat dan dikembalikan ke versi sebelumnya jika terjadi kesalahan. Ini sangat membantu terutama saat bekerja dalam tim, karena semua proses pengembangan bisa terlacak secara rapi dan sistematis dari awal. Tanpa git init, kita tidak bisa menggunakan fitur Git lainnya seperti commit, branch, dan sebagainya.

2. Git clone

```
2. git clone https://github.com/username/repo.git
Menyalin repository Git dari remote ke lokal.
```

Perintah git clone digunakan untuk menyalin repository yang sudah ada di internet (biasanya di GitHub) ke komputer lokal kita. Ini sangat berguna dalam kerja kelompok atau proyek open source, karena kita bisa langsung mulai mengembangkan tanpa perlu membuat ulang file satu per satu. Dalam tugas ini,

kami mencoba cloning repository dari GitHub menggunakan format perintah: git clone https://github.com/username/nama-repo.git. Setelah proses clone berhasil, seluruh isi repository seperti file, struktur folder, dan riwayat commit akan muncul di komputer lokal kami. Ini sangat menghemat waktu dan memastikan semua anggota tim bekerja di kode sumber yang sama. Selain itu, dengan clone, kami tidak hanya mendapatkan file proyek, tetapi juga mendapatkan seluruh sejarah versi dari repository tersebut. Jadi kami bisa belajar dari riwayat perubahan kode sebelumnya, dan ini sangat membantu memahami perkembangan proyek secara menyeluruh.

3. Git status

```
3. git status
Menampilkan status file di repository: modified, staged, untracked.
```

git status adalah perintah yang sering sekali kami gunakan dalam praktik Git. Perintah ini memberikan informasi lengkap tentang status file dalam repository kita, apakah ada file yang sudah diubah (modified), file baru (untracked), atau file yang sudah siap untuk di-*commit*. Misalnya, setelah kami mengedit file index.html, kami jalankan git status dan Git akan memberitahu bahwa file tersebut belum ditambahkan ke *staging area*. Hal ini sangat membantu agar kami tidak salah langkah, karena bisa jadi ada file penting yang belum dimasukkan ke dalam proses commit. Selain itu, perintah ini juga menampilkan cabang (*branch*) aktif saat ini, sehingga kami tahu kami sedang bekerja di branch yang benar. Dalam konteks tugas, git status memberi kami kejelasan tentang apa yang sudah kami lakukan dan apa yang perlu kami lakukan selanjutnya sebelum melanjutkan ke proses commit atau push ke GitHub.

4. Git add

```
4. git add .
Menambahkan semua file ke staging area.
```

Perintah git add digunakan untuk menandai file yang akan kita simpan ke dalam

Git melalui proses commit. File yang di-*add* akan masuk ke dalam *staging area*, yaitu area sementara sebelum perubahan benar-benar direkam secara permanen. Dalam tugas ini, setiap kali kami selesai mengedit atau menambahkan file baru, kami harus mengetik git add nama-file atau git add . untuk menambahkan semua file sekaligus. Jika kami lupa menjalankan perintah ini, maka file yang diubah tidak akan tersimpan dalam commit, dan itu bisa mengacaukan proses kerja kelompok. Fungsi git add sangat penting dalam menjaga keteraturan alur kerja, karena memastikan hanya file yang sudah benar-benar siap yang dimasukkan ke commit. Jadi dalam konteks proyek nyata, perintah ini seperti gerbang seleksi agar hanya perubahan yang sesuai dan aman yang masuk ke dalam versi final. Kesalahan dalam add bisa membuat commit tidak lengkap.

5. Git commit

```
5. git commit -m "Pesan commit"
Menyimpan perubahan yang sudah distage ke riwayat proyek.
```

git commit adalah langkah penting dalam pencatatan perubahan. Setelah file dimasukkan ke staging area melalui git add, maka kami bisa menggunakan git commit -m "Pesan" untuk menyimpan snapshot proyek pada titik waktu tersebut. Pesan commit ini wajib ditulis dengan jelas, misalnya: git commit -m "Tambah fitur form registrasi petani". Ini agar ketika kami atau anggota tim melihat riwayat commit, bisa memahami apa yang diubah tanpa harus membuka file satu per satu. Dalam tugas ini, kami melakukan commit setiap selesai mengerjakan bagian tertentu agar tidak bingung ketika ingin kembali ke versi sebelumnya. Commit juga membantu dosen atau pembimbing proyek mengecek perkembangan tugas kami dari waktu ke waktu. Jadi, commit itu seperti menyimpan checkpoint dalam game, sehingga kalau ada error di versi terbaru, kita bisa kembali ke versi sebelumnya yang masih stabil.

6. Git push

```
6. git push origin main
Mengirim commit dari lokal ke remote (GitHub).
```

Perintah git push digunakan untuk mengirim semua commit dari komputer lokal ke repository online seperti GitHub. Ini penting agar perubahan yang kami buat bisa disinkronkan dengan tim, dan bisa diakses dari perangkat lain jika sewaktu-waktu terjadi kerusakan di komputer lokal. Dalam praktik tugas ini, kami menjalankan git push origin main untuk mengirim commit ke branch utama. Namun, push hanya bisa dilakukan setelah kita terkoneksi dengan GitHub menggunakan SSH atau HTTPS. Jika belum dikonfigurasi, maka push akan gagal. Fungsi push ini juga membantu menjaga backup proyek secara otomatis di cloud, jadi kami tidak khawatir kehilangan data. Selain itu, saat bekerja dalam kelompok, push menjadi alat komunikasi antar anggota tim, karena setiap push akan memperbarui kode yang bisa ditarik (pull) oleh rekan lainnya. Jadi, push adalah jembatan antara pekerjaan lokal dan versi online dari proyek kita.

7. Git pull

```
7. git pull origin main
Mengambil dan menggabungkan perubahan dari remote ke lokal.
```

git pull adalah kebalikan dari push. Saat teman satu tim mengirim update ke repository GitHub, kami harus melakukan git pull agar perubahan tersebut masuk ke komputer lokal kami. Dalam tugas ini, perintah ini sangat sering digunakan agar kami semua bekerja pada versi kode yang paling baru. git pull menjalankan dua proses sekaligus, yaitu fetch (mengambil data) dan merge (menggabungkan perubahan). Jadi, setelah pull, file kami bisa langsung disinkronkan dengan yang ada di server GitHub. Namun, terkadang saat file yang kami edit juga diubah oleh anggota tim lain, akan terjadi konflik (*merge conflict*) yang harus kami selesaikan secara manual. Meski begitu, proses ini sangat mengedukasi karena kami jadi

belajar cara menyatukan kerja tim dalam satu proyek. pull membuat kolaborasi menjadi mungkin, tanpa saling menimpa file satu sama lain secara sembarangan.

8. Git branch & checkout

```
8. git branch fitur-baru
   git checkout fitur-baru
Membuat dan berpindah ke branch baru.
```

Dalam Git, branch adalah fitur yang sangat berguna untuk mengembangkan fitur baru tanpa merusak kode utama. Misalnya, kami bisa membuat branch baru dengan git branch fitur-login, lalu pindah ke branch tersebut dengan git checkout fitur-login. Di sana, kami bebas bereksperimen dan menulis kode baru. Setelah selesai, hasilnya bisa digabungkan ke branch utama dengan merge. Dalam tugas ini, kami menggunakan branch untuk membagi pekerjaan: satu anggota di branch fitur-pendaftaran, satu lagi di fitur-pemetaan, dan seterusnya. Dengan begitu, masing-masing bisa fokus tanpa mengganggu satu sama lain. Perintah git checkout juga bisa digunakan untuk kembali ke commit tertentu atau berpindah ke branch lain. Jadi, penggunaan branch dan checkout memberikan fleksibilitas dan keamanan dalam pengembangan proyek, serta melatih kami berpikir modular seperti dalam dunia kerja profesional di bidang pemrograman.

Kesimpulan

Setelah mempelajari dan mempraktikkan berbagai perintah dasar Git seperti `git init`, `clone`, `status`, `add`, `commit`, `push`, `pull`, serta penggunaan `branch` dan `checkout`, kami sebagai mahasiswa sangat merasakan pentingnya penguasaan sistem version control dalam proses pengembangan perangkat lunak. Git bukan hanya sekadar alat bantu, tapi sudah menjadi kebutuhan utama dalam menjaga ketertiban dan efektivitas kerja, terutama saat kami bekerja secara kolaboratif dalam satu tim. Dengan adanya Git, kami bisa melacak setiap perubahan kode, menghindari konflik file, dan mempermudah proses penggabungan pekerjaan masing-masing anggota. Misalnya, dengan perintah `git status`, kami tahu kondisi terbaru file, sedangkan `git commit` menjadi penanda resmi setiap perkembangan proyek. Bahkan saat terjadi kesalahan atau bug, Git memungkinkan kami untuk kembali ke versi sebelumnya tanpa harus panik atau mengulang dari awal.

Lebih dari itu, Git juga mengajarkan kami disiplin dalam mengelola proyek. Setiap perubahan harus melalui proses `add` dan `commit` dengan pesan yang jelas dan informatif, sehingga seluruh anggota tim bisa memahami alur pekerjaan dengan mudah. Penggunaan `branch` dan `checkout` sangat membantu dalam membagi tugas secara modular, sehingga masing-masing anggota bisa mengembangkan fitur tanpa mengganggu bagian utama proyek. Selain itu, `push` dan `pull` memperlihatkan bagaimana pentingnya komunikasi kode antaranggota tim agar semua tetap sinkron dan up-to-date. Praktik menggunakan Git ini benar-benar memberikan pengalaman nyata kepada kami dalam bekerja secara profesional, karena tidak hanya fokus pada menulis kode, tapi juga bagaimana mengelolanya dengan rapi, terdokumentasi, dan aman. Kesimpulannya, penggunaan Git tidak hanya mendukung proses teknis, tapi juga membentuk mentalitas kerja tim yang lebih terstruktur dan bertanggung jawab

Daftar Pustaka

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

[2] Git Documentation, "Git - Documentation," 2024. [Online]. Available: https://git-scm.com/docs

[3] GitHub, "GitHub Documentation," 2024. [Online]. Available: https://docs.github.com

[4] Codecademy, "Learn Git," 2023. [Online]. Available: https://www.codecademy.com/learn/learn-git

[5] Atlassian, "Git Tutorials & Training," 2023. [Online]. Available: https://www.atlassian.com/git/tutorials

[6] W3Schools, "Git Tutorial," 2023. [Online]. Available: https://www.w3schools.com/git/

[7] TutorialsPoint, "Git Basics," 2024. [Online]. Available: https://www.tutorialspoint.com/git/index.htm