**RANGKUMAN GIT & GITHUB**

### ****VIDEO 1 APA ITU GIT DAN GITHUB?****

1. **Pengertian Git dan GitHub**

**Git** merupakan *Version Control System* yang memungkinkan pengembang mengelola perubahan dan bahkan mengembalikan perubahan dalam *source code* selama pengembangan perangkat lunak. **GitHub** adalah layanan cloud untuk menyimpan dan mengelola *project* atau repository. **Git** bisa berjalan tanpa **GitHub** dan sebaliknya. Sederhananya, **GitHub** adalah website yang ada **Git** di dalamnya.

1. **Manfaat Menggunakan Git dan GitHub**
2. Memungkinkan pengembang untuk melacak dan mengelola perubahan kode secara efisien.
3. Memudahkan kerja tim dalam proyek pengembangan perangkat lunak.
4. Menyimpan salinan kode secara online untuk mencegah kehilangan data.
5. **Istilah dalam Git dan GitHub**
6. **Repository**: Tempat penyimpanan proyek yang berisi semua file dan riwayat revisi.
7. **Commit**: Menyimpan riwayat perubahan yang dilakukan pada file.
8. **Hash: Penanda pada commit.**
9. **Branch: Percabangan pada commit.**
10. **Push dan Pull**: Mengirim dan mengambil perubahan antara repositori lokal (**Git**) dan **GitHub.** Dan untuk melakukannya, harus di clone dulu repository-nya.

### VIDEO 2 BEKERJA DENGAN GITHUB

1. **Mengakses GitHub**
2. Masuk ke website GitHub
3. Login atau sign in
4. **Membuat Repository**
5. Ke halaman beranda
6. Klik logo buku hijau di bagian kiri
7. Isi nama repository, pilih akses dan bisa menambahkan file README kalau diperlukan
8. Tekan “create repository”
9. **Menambahkan File di Repository**
10. Tekan add file dan pilih “create new file”
11. Beri nama file yang ingin kamu buat
12. Isi filenya sesuai yang dibutuhkan
13. Tekan “commit change” dan isi commitnya
14. **Mengedit File di Repository**
15. Tekan file yang kamu ingin edit
16. Tekan simbol pensil
17. Mulai edit
18. Tekan “commit change” dan isi jika sudah selesai mengedit file
19. Kamu bisa mengecek perubahan dengan menekan tombol history

### VIDEO 3 GITHUB : BRANCH

1. **Pengertian Branch**

Branch adalah cabang dari alur pengembangan utama (biasanya main atau master) yang memungkinkan pengembang bekerja pada fitur atau perbaikan secara terpisah tanpa memengaruhi kode utama.

1. **Manfaat dan Kegunaan Branch**
2. Membuat sebuah perubahan tanpa mengganggu cabang utama
3. Untuk pengerjaan lebih dari satu orang
4. **Membuat Branch Baru**
5. Pilih ikon branch pada bagian kiri yang bertuliskan “main”
6. Tambahkan branch dengan mengetikkan nama branchnya dan tekan “create new branch”
7. Sekarang kamu telah menggunakan branch yang baru di buat
8. Edit file
9. Pergi ke bagian insights, kemudian pilih network untuk mengecek branch
10. **Merge Branch dan Pull Request**
11. Setelah membuat branch baru dan menambah atau mengedit sebuah file, akan ada opsi “compare & pull request”
12. Tekan tombol tersebut dan otomatis akan diarahkan untuk pull request. Isi bagian-bagiannya dan tekan “create pull request”
13. GitHub akan mengecek apakah ada merge conflict, jika tidak maka bisa digabungkan
14. Tekan tombol “confirm merge”
15. Branch baru tadi sekarang telah bergabung dengan branch utama.
16. Cek bagian insight untuk melihat diagramnya
17. Jika melakukan merge lebih dari satu branch, maka harus satu-persatu. Jika ada merge conflict maka harus diedit terlebih dahulu.
18. **Istilah Baru**
19. **Checkout :** Berpindah branch atau commit.
20. **Pull Request :** Meminta pemilik repository untuk mengambil/menyetujui perubahan
21. **Merge :** Menggabungkan dua branch
22. **Merge Conflict :** Baris yang sama diubah oleh 2 branch yang berbeda.

### VIDEO 4 FORK

1. **Pengertian Fork**

**Fork** merupakan istilah untuk menduplikat repository milik orang lain tanpa mengganggu repository yang di duplikat. Github memungkinkan adanya **Fork,** namun tetap memberikan credit bagi pemilik repository. **Fork tidak sama dengan Clone** karena **Fork itu menduplikat repository milik orang lain, sedangkan clone menduplikat repository dari remote ke komputer lokal.**

1. **Cara Melakukan Fork di GitHub**
2. Kunjungi akun milik orang lain dan masuk ke repositorynya
3. Edit langsung atau klik tombol “fork”. Jika sudah, otomatis akan kembali ke akun kalian dan repository sudah terduplikat
4. Bisa langsung mengedit atau menambahkan file baru yang nantinya bisa di pull request ke pemilik aslinya
5. Sekarang, tinggal menunggu pemilik repository untuk menerima pull requestnya. Walaupun tidak diterima, hasil forknya tetap ada di akun kita.

.

### VIDEO 5 BEKERJA DENGAN GIT

1. **Menginstall Git**
2. Pergi ke website git-scm.com
3. Download dan install Git
4. Pada halaman instalasi, sesuaikan Git dengan kebutuhan
5. Buka git dan cek apakah sudah terdeteksi masuk atau belum
6. **Istilah-istilah pada Git**
7. **Area :** *Working tree*(tempat bekerja), *staging area* (file yang sudah diubah akan masuk ke sini), dan *History* (tempat commit ditaruh)
8. **Directory :** tempat kita berada saat ini
9. **Cara membuat repository dan menambahkan file untuk di commit**
10. Buat folder di file manager
11. Klik kanan dan open with Git Bash
12. Ketik git init
13. Ketik code . untuk langsung menuju code editormu
14. Tambahkan file sesuai keinginan
15. Ketik git status untuk melihat perubahan
16. Ketik git add <nama file> untuk menambahkan file tertentu ke repository. Atau, ketik git add . untuk menambahkan semua file yang masih ada di working tree dan belum masuk ke repository
17. Ketik git commit <nama file> untuk menambahkan file ke bagian history atau git commit untuk menambahkan semuanya
18. Jika ada perubahan pada file, maka status file akan berubah menjadi modified dan harus di tambahkan lagi ke history dengan mengetikkan git commit -am “pesan”

## 🎥 Video 6: GIT BRANCH & MERGE

Video ini membahas konsep dan praktik penggunaan **branch** dan **merge** dalam Git, yang merupakan fitur penting untuk pengembangan perangkat lunak secara terstruktur dan kolaboratif.

### 1. ****Pengertian Branch****

* **Definisi**: Branch adalah cabang dari alur pengembangan utama yang memungkinkan pengembang bekerja pada fitur atau perbaikan secara terpisah tanpa memengaruhi kode utama.
* **Manfaat**:
  + Memungkinkan pengembangan fitur baru tanpa mengganggu kode yang sudah stabil.
  + Memfasilitasi kolaborasi antar tim dengan bekerja pada branch masing-masing.
  + Memudahkan pengujian dan review sebelum menggabungkan perubahan ke branch utama.

### 2. ****Membuat dan Berpindah ke Branch Baru****

* **Perintah**:
* git checkout -b nama\_branch
* **Penjelasan**: Perintah ini membuat branch baru dengan nama yang ditentukan dan langsung berpindah ke branch tersebut.

### 3. ****Melihat Daftar Branch****

* **Perintah**:
* git branch
* **Penjelasan**: Menampilkan daftar semua branch yang ada di repository lokal, dengan tanda asterisk (\*) menunjukkan branch yang sedang aktif.

### 4. ****Melakukan Perubahan dan Commit di Branch****

* **Langkah-langkah**:
  1. Lakukan perubahan pada kode di branch aktif.
  2. Tambahkan perubahan ke staging area:
  3. git add .
  4. Commit perubahan dengan pesan yang sesuai:
  5. git commit -m "Deskripsi perubahan"

### 5. ****Berpindah ke Branch Lain****

* **Perintah**:
* git checkout nama\_branch
* **Penjelasan**: Digunakan untuk berpindah ke branch yang sudah ada. Setelah berpindah, perubahan yang dilakukan akan tercatat di branch tersebut.

### 6. ****Menggabungkan Branch (Merge)****

* **Langkah-langkah**:
  1. Berpindah ke branch tujuan (misalnya main):
  2. git checkout main
  3. Menggabungkan branch lain ke branch aktif:
  4. git merge nama\_branch
* **Penjelasan**: Perintah git merge digunakan untuk menggabungkan perubahan dari nama\_branch ke branch yang sedang aktif.

### 7. ****Menghapus Branch yang Sudah Digabung****

* **Perintah**:
* git branch -d nama\_branch
* **Penjelasan**: Menghapus branch yang sudah tidak diperlukan. Perintah -d akan menolak penghapusan jika branch belum digabung (merge) ke branch lain, sedangkan -D akan memaksa penghapusan.

Video ini memberikan pemahaman mendalam tentang penggunaan branch dan merge dalam Git, yang merupakan praktik penting dalam pengembangan perangkat lunak modern.

Jika kamu memiliki link video berikutnya dari playlist ini, silakan kirimkan, dan saya akan melanjutkan dengan merangkum materi dari video tersebut.

Berikut ringkasan materi dari video ketujuh dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**, berjudul **"#7 GIT MERGE CONFLICT"**:

## 🎥 Video 7: GIT MERGE CONFLICT

Video ini membahas tentang **merge conflict** dalam Git, yaitu situasi di mana Git tidak dapat secara otomatis menggabungkan perubahan dari dua branch yang berbeda karena adanya konflik pada bagian kode yang sama.

### 1. ****Pengertian Merge Conflict****

* **Definisi**: Merge conflict terjadi ketika dua branch yang akan digabung memiliki perubahan pada bagian kode yang sama, dan Git tidak dapat menentukan perubahan mana yang harus dipertahankan.
* **Penyebab Umum**:
  + Perubahan pada baris kode yang sama di dua branch berbeda.
  + Penghapusan file di satu branch dan perubahan pada file yang sama di branch lain.

### 2. ****Menangani Merge Conflict****

* **Langkah-langkah**:
  1. Lakukan merge antara dua branch:
  2. git merge nama\_branch
  3. Jika terjadi konflik, Git akan menandai file yang bermasalah dan menghentikan proses merge.
  4. Buka file yang mengalami konflik dan cari bagian yang ditandai dengan <<<<<<<, =======, dan >>>>>>>.
  5. Pilih perubahan yang ingin dipertahankan dan hapus tanda-tanda konflik tersebut.
  6. Setelah menyelesaikan konflik, tambahkan file yang telah diperbaiki ke staging area:
  7. git add nama\_file
  8. Lanjutkan proses merge dengan commit:
  9. git commit

### 3. ****Tips untuk Menghindari Merge Conflict****

* **Komunikasi Tim**: Selalu berkoordinasi dengan anggota tim mengenai bagian kode yang sedang dikerjakan.
* **Sering Melakukan Pull**: Secara rutin lakukan git pull untuk mendapatkan perubahan terbaru dari repository pusat.
* **Pecah Perubahan Menjadi Kecil**: Lakukan commit untuk perubahan kecil dan sering, sehingga memudahkan dalam mengidentifikasi dan menyelesaikan konflik.

Video ini memberikan pemahaman tentang bagaimana menghadapi dan menyelesaikan merge conflict dalam Git, yang merupakan keterampilan penting dalam pengembangan perangkat lunak kolaboratif.

Jika kamu memiliki link video berikutnya dari playlist ini, silakan kirimkan, dan saya akan melanjutkan dengan merangkum materi dari video tersebut.

Terima kasih atas klarifikasinya. Berikut adalah ringkasan materi dari video **"#8 GIT REMOTE"** oleh **Web Programming UNPAS**:

## 🎥 Video 8: GIT REMOTE

Video ini membahas konsep dan perintah dasar untuk bekerja dengan remote repository di Git, yang memungkinkan kolaborasi dalam pengembangan perangkat lunak.([YouTube](https://www.youtube.com/watch?pp=ygUNI2dpdHJlbW90ZWFkZA%3D%3D&v=6D_YArepsuo&utm_source=chatgpt.com))

Pembelajaran saat Jam Pembelajaran PPL di jam pelajaran PPL

1. **Mengirimkan file ke GitHub (Remote)**
2. Ketik git remote add origin <https://github.com/Fa-rain/belajarBarengXIRPL1.git> (sesuaikan, bila sudah terhubung selanjutnya tidak perlu di remote lagi kecuali jika ingin mengganti akun atau repository) Jika pertama kali, perlu autentikasi terlebih dahulu
3. Ketik git push -u origin master (sesuaikan bagian masternya karena itu adalah branch)
4. Refresh website github untuk mengecek hasil

### 1. ****Pengertian Remote Repository****

* Remote repository adalah versi dari proyek Git yang disimpan di server, seperti GitHub, GitLab, atau Bitbucket.
* Berfungsi untuk berbagi kode dengan tim atau publik, serta menyimpan salinan cadangan dari proyek.

### 2. ****Menambahkan Remote Repository****

* Gunakan perintah berikut untuk menambahkan remote repository:([Stack Overflow](https://stackoverflow.com/questions/16330404/how-to-remove-remote-origin-from-a-git-repository?utm_source=chatgpt.com))
* git remote add origin <URL-remote-repository>

Contoh:

git remote add origin https://github.com/username/repository.git

([TheServerSide](https://www.theserverside.com/video/How-to-use-the-git-remote-add-origin-command-to-push-remotely?utm_source=chatgpt.com))

* Perintah ini menghubungkan repositori lokal dengan repositori jarak jauh yang ditentukan.

### 3. ****Melihat Daftar Remote****

* Untuk melihat daftar remote yang telah dikonfigurasi:([GitHub Docs](https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories?utm_source=chatgpt.com))
* git remote -v

Perintah ini menampilkan URL dari remote yang telah ditambahkan.

### 4. ****Menghapus Remote****

* Untuk menghapus remote yang tidak diperlukan:
* git remote remove origin

Perintah ini menghapus remote dengan nama 'origin'.([GitHub Docs](https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories?utm_source=chatgpt.com))

### 5. ****Mengubah URL Remote****

* Untuk mengubah URL dari remote yang ada:
* git remote set-url origin <URL-baru>

Contoh:

git remote set-url origin https://github.com/username/new-repository.git

* Perintah ini berguna jika alamat repositori jarak jauh berubah.

### 6. ****Push dan Pull ke Remote****

* Setelah menambahkan remote, Anda dapat mengirimkan perubahan ke repositori jarak jauh dengan:([GitHub Docs](https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories?utm_source=chatgpt.com))
* git push -u origin master

Perintah ini mengirimkan commit dari branch lokal 'master' ke remote 'origin'.

* Untuk menarik perubahan dari remote:([Stack Overflow](https://stackoverflow.com/questions/16330404/how-to-remove-remote-origin-from-a-git-repository?utm_source=chatgpt.com))
* git pull origin master

Perintah ini menarik perubahan dari branch 'master' di remote 'origin'.

Jika Anda memerlukan ringkasan materi dari video berikutnya dalam playlist ini, silakan kirimkan linknya, dan saya akan dengan senang hati membantu.

Terima kasih atas pertanyaan Anda. Setelah memeriksa, video dengan judul **"GITHUB PAGES / NGOBAR#12 : Web Hosting Gratis"** yang Anda kirimkan merupakan video ke-9 dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**. Video ini membahas tentang penggunaan **GitHub Pages** sebagai layanan web hosting gratis untuk website statis.([YouTube](https://www.youtube.com/watch?v=rgDDWhQe-ow&utm_source=chatgpt.com))

Berikut adalah ringkasan materi dari video tersebut:

## 🎥 Video 9: GITHUB PAGES / NGOBAR#12 – Web Hosting Gratis

Video ini menjelaskan cara menggunakan **GitHub Pages** untuk menghosting website statis secara gratis.

### 1. ****Pengertian GitHub Pages****

* **GitHub Pages** adalah layanan dari GitHub yang memungkinkan pengguna untuk menghosting website statis langsung dari repository GitHub.
* Cocok untuk proyek-proyek seperti portofolio, dokumentasi, atau blog pribadi.

### 2. ****Membuat Repository untuk GitHub Pages****

* Buat repository baru di GitHub dengan nama username.github.io, di mana username adalah nama pengguna GitHub Anda.
* Repository ini akan menjadi tempat untuk menyimpan file website Anda.

### 3. ****Menambahkan File Website****

* Tambahkan file HTML, CSS, dan JavaScript ke repository tersebut.
* Pastikan file utama website Anda bernama index.html.

### 4. ****Mengaktifkan GitHub Pages****

* Masuk ke pengaturan (Settings) repository Anda.
* Di bagian **GitHub Pages**, pilih branch yang akan digunakan (biasanya main atau master).
* Setelah beberapa menit, website Anda akan dapat diakses melalui URL https://username.github.io.

### 5. ****Keuntungan Menggunakan GitHub Pages****

* Gratis dan mudah digunakan.
* Terintegrasi langsung dengan GitHub, memudahkan dalam pengelolaan versi dan kolaborasi.
* Mendukung HTTPS untuk keamanan.

Jika Anda memerlukan bantuan lebih lanjut atau memiliki pertanyaan lain, silakan sampaikan. Saya siap membantu!

Terima kasih atas klarifikasinya. Setelah memeriksa, video dengan judul **"MULTIPLE REMOTES"** yang Anda kirimkan merupakan video ke-10 dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**. Video ini membahas cara bekerja dengan beberapa remote repository dalam Git.

Berikut adalah ringkasan materi dari video tersebut:

## 🎥 Video 10: MULTIPLE REMOTES

Video ini menjelaskan bagaimana cara mengelola beberapa remote repository dalam satu proyek Git.

### 1. ****Pengertian Multiple Remotes****

* Dalam proyek Git, Anda dapat memiliki lebih dari satu remote repository.
* Hal ini berguna untuk berbagai tujuan, seperti:
  + Menyimpan salinan cadangan di server lain.
  + Berbagi kode dengan tim atau kolaborator di platform yang berbeda.
  + Menguji perubahan di lingkungan yang berbeda sebelum di-push ke remote utama.

### 2. ****Menambahkan Remote Baru****

* Untuk menambahkan remote baru, gunakan perintah:
* git remote add <nama\_remote> <url\_remote>

Contoh:

git remote add backup https://github.com/username/backup-repo.git

* Perintah ini menambahkan remote baru dengan nama backup dan URL yang ditentukan.

### 3. ****Melihat Daftar Remote****

* Untuk melihat daftar remote yang telah dikonfigurasi:
* git remote -v
* Perintah ini menampilkan nama dan URL dari setiap remote yang ada.

### 4. ****Menghapus Remote****

* Untuk menghapus remote yang tidak diperlukan:
* git remote remove <nama\_remote>

Contoh:

git remote remove backup

* Perintah ini menghapus remote dengan nama backup.

### 5. ****Push ke Remote Tertentu****

* Untuk mengirim perubahan ke remote tertentu:
* git push <nama\_remote> <nama\_branch>

Contoh:

git push backup master

* Perintah ini mengirim commit dari branch master ke remote backup.

### 6. ****Pull dari Remote Tertentu****

* Untuk menarik perubahan dari remote tertentu:
* git pull <nama\_remote> <nama\_branch>

Contoh:

git pull origin master

* Perintah ini menarik perubahan dari branch master di remote origin.

Jika Anda memerlukan bantuan lebih lanjut atau memiliki pertanyaan lain, silakan sampaikan. Saya siap membantu!

Terima kasih atas kesabaran Anda. Berikut adalah ringkasan materi dari video ke-11 dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**, berjudul **"#11 REMOTE BRANCH"**:

## 🎥 Video 11: REMOTE BRANCH

Video ini membahas tentang **remote branch** dalam Git, yaitu cabang (branch) yang ada di repository jarak jauh (remote repository).

### 1. ****Pengertian Remote Branch****

* **Remote branch** adalah salinan dari branch yang ada di repository lokal, tetapi berada di server remote seperti GitHub.
* Contoh: origin/master adalah branch master yang ada di remote repository bernama origin.

### 2. ****Melihat Daftar Remote Branch****

* Untuk melihat daftar remote branch yang tersedia:
* git branch -r
* Perintah ini menampilkan semua remote branch yang ada.

### 3. ****Melihat Semua Branch (Lokal dan Remote)****

* Untuk melihat semua branch, baik lokal maupun remote:
* git branch -a
* Perintah ini menampilkan daftar lengkap branch yang ada.

### 4. ****Melacak Remote Branch ke Lokal****

* Untuk membuat branch lokal yang melacak remote branch:
* git checkout -b <nama\_branch> <remote>/<nama\_branch>
* Contoh:
* git checkout -b feature/login origin/feature/login
* Perintah ini membuat branch lokal feature/login yang melacak remote branch origin/feature/login.

### 5. ****Menghapus Remote Branch****

* Untuk menghapus remote branch:
* git push <remote> --delete <nama\_branch>
* Contoh:
* git push origin --delete feature/login
* Perintah ini menghapus branch feature/login dari remote repository origin.

Jika Anda memerlukan bantuan lebih lanjut atau memiliki pertanyaan lain, silakan sampaikan. Saya siap membantu!

Terima kasih atas kesabaran Anda. Berikut adalah ringkasan materi dari video ke-12 dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**, berjudul **"#12 GITIGNORE"**:

## 🎥 Video 12: GITIGNORE

Video ini membahas tentang penggunaan file .gitignore dalam Git, yang berfungsi untuk menentukan file atau direktori mana saja yang harus diabaikan oleh Git agar tidak dimasukkan ke dalam version control.

### 1. ****Pengertian**** .gitignore

* File .gitignore adalah file teks yang berisi daftar pola (pattern) yang menunjukkan file atau direktori mana yang harus diabaikan oleh Git.
* File ini membantu mencegah file sementara, file build, atau file konfigurasi lokal yang tidak perlu dimasukkan ke dalam repository.

### 2. ****Membuat File**** .gitignore

* Untuk membuat file .gitignore, cukup buat file baru dengan nama .gitignore di direktori root repository Anda.
* Tambahkan pola-pola yang sesuai dengan file atau direktori yang ingin diabaikan.([Stack Overflow](https://stackoverflow.com/questions/55880919/how-to-add-well-known-on-github-pages-files-using-html?utm_source=chatgpt.com))

### 3. ****Contoh Isi**** .gitignore

Berikut adalah contoh isi file .gitignore untuk proyek berbasis Node.js:

node\_modules/

npm-debug.log

.DS\_Store

* node\_modules/ – Mengabaikan direktori node\_modules yang berisi dependensi proyek.
* npm-debug.log – Mengabaikan file log dari npm.
* .DS\_Store – Mengabaikan file sistem yang dibuat oleh macOS.

### 4. ****Menambahkan**** .gitignore ****ke Repository****

* Setelah membuat atau memodifikasi file .gitignore, tambahkan file tersebut ke staging area:

git add .gitignore

* Kemudian, commit perubahan tersebut:

git commit -m "Menambahkan file .gitignore"

### 5. ****Mengabaikan File yang Sudah Terlanjur Tersimpan****

* Jika ada file yang sudah terlanjur dimasukkan ke dalam repository dan ingin diabaikan, hapus file tersebut dari index Git tanpa menghapusnya dari sistem file:

git rm --cached <nama\_file>

* Setelah itu, commit perubahan tersebut:([YouTube](https://www.youtube.com/watch?v=ppi_M-FX1CY&utm_source=chatgpt.com))

git commit -m "Menghapus file dari tracking Git"

### 6. ****Menambahkan Pola Umum untuk**** .gitignore

* Untuk proyek tertentu, Anda dapat menggunakan template .gitignore yang sesuai dengan bahasa atau framework yang digunakan.
* GitHub menyediakan koleksi template .gitignore untuk berbagai bahasa pemrograman dan framework di <https://github.com/github/gitignore>.

Jika Anda memerlukan bantuan lebih lanjut atau memiliki pertanyaan lain, silakan sampaikan. Saya siap membantu!

Terima kasih atas kesabaran Anda. Berikut adalah ringkasan materi dari video ke-13 dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**, berjudul **"#13 GIT REBASE"**:

## 🎥 Video 13: GIT REBASE

Video ini membahas tentang penggunaan perintah git rebase dalam Git, yang digunakan untuk menggabungkan perubahan dari satu branch ke branch lain dengan cara yang lebih bersih dan linear dibandingkan dengan git merge.

### 1. ****Pengertian Git Rebase****

* git rebase adalah perintah yang digunakan untuk memindahkan atau menggabungkan urutan commit dari satu branch ke branch lain.
* Berbeda dengan git merge yang membuat commit baru untuk menggabungkan perubahan, git rebase akan memindahkan seluruh commit dari branch sumber ke atas commit branch target, sehingga menghasilkan riwayat commit yang lebih linear.

### 2. ****Cara Menggunakan Git Rebase****

* Untuk melakukan rebase, pertama-tama pastikan Anda berada di branch yang ingin digabungkan perubahan dari branch lain:

git checkout <nama-branch>

* Kemudian, lakukan rebase terhadap branch target:

git rebase <nama-branch-target>

* Contoh:([GitHub](https://github.com/microsoft/Web-Dev-For-Beginners?utm_source=chatgpt.com))

git checkout feature

git rebase main

* Perintah di atas akan memindahkan seluruh commit dari branch feature ke atas commit terakhir di branch main.

### 3. ****Menangani Konflik saat Rebase****

* Jika terjadi konflik selama proses rebase, Git akan menghentikan proses dan meminta Anda untuk menyelesaikan konflik tersebut.
* Setelah menyelesaikan konflik, tandai file yang telah diperbaiki:

git add <nama-file>

* Kemudian, lanjutkan proses rebase:

git rebase --continue

* Ulangi langkah di atas hingga semua konflik diselesaikan dan rebase selesai.

### 4. ****Menghentikan Proses Rebase****

* Jika Anda ingin membatalkan proses rebase dan kembali ke kondisi sebelum rebase, gunakan perintah:

git rebase --abort

* Perintah ini akan membatalkan rebase dan mengembalikan branch ke kondisi semula.

### 5. ****Keuntungan Menggunakan Git Rebase****

* Menghasilkan riwayat commit yang lebih bersih dan linear.
* Memudahkan dalam melacak perubahan dan memahami alur pengembangan proyek.
* Membantu dalam menghindari commit merge yang tidak perlu.

Jika Anda memerlukan bantuan lebih lanjut atau memiliki pertanyaan lain, silakan sampaikan. Saya siap membantu!

Terima kasih atas kesabaran Anda. Berikut adalah ringkasan materi dari video ke-14 dalam playlist **"GIT & GITHUB"** oleh **Web Programming UNPAS**, berjudul **"#14 GIT REMOTE"**:

## 🎥 Video 14: GIT REMOTE

Video ini membahas tentang penggunaan remote repository dalam Git, yang memungkinkan kolaborasi dan sinkronisasi kode antara repositori lokal dan repositori jarak jauh (remote).

### 1. ****Pengertian Remote Repository****

* **Remote repository** adalah versi repositori yang berada di server, seperti GitHub, GitLab, atau Bitbucket.
* Remote repository memungkinkan kolaborasi antar pengembang dan penyimpanan kode secara terpusat.

### 2. ****Menambahkan Remote Repository****

* Untuk menambahkan remote repository, gunakan perintah:
* git remote add <nama\_remote> <url\_remote>

Contoh:

git remote add origin https://github.com/username/repo.git

Perintah ini menambahkan remote dengan nama origin dan URL yang ditentukan.

### 3. ****Melihat Daftar Remote Repository****

* Untuk melihat daftar remote yang telah dikonfigurasi:
* git remote -v

Perintah ini menampilkan nama dan URL dari setiap remote yang ada.

### 4. ****Menghapus Remote Repository****

* Untuk menghapus remote yang tidak diperlukan:
* git remote remove <nama\_remote>

Contoh:

git remote remove origin

Perintah ini menghapus remote dengan nama origin.

### 5. ****Push Perubahan ke Remote Repository****

* Untuk mengirim perubahan dari branch lokal ke remote repository:
* git push <nama\_remote> <nama\_branch>

Contoh:

git push origin master

Perintah ini mengirim commit dari branch master ke remote origin.

### 6. ****Pull Perubahan dari Remote Repository****

* Untuk menarik perubahan dari remote repository ke branch lokal:
* git pull <nama\_remote> <nama\_branch>

Contoh:

git pull origin master

Perintah ini menarik perubahan dari branch master di remote origin.

### 7. ****Menggunakan Beberapa Remote Repository****

* Anda dapat memiliki lebih dari satu remote repository dalam satu proyek.
* Untuk menambahkan remote baru:
* git remote add <nama\_remote> <url\_remote>

Contoh:

git remote add backup https://github.com/username/backup-repo.git

Perintah ini menambahkan remote baru dengan nama backup.

* Untuk mengirim perubahan ke remote tertentu:
* git push <nama\_remote> <nama\_branch>

Contoh:

git push backup master

Perintah ini mengirim commit dari branch master ke remote backup.

Jika Anda memerlukan bantuan lebih lanjut atau memiliki pertanyaan lain, silakan sampaikan. Saya siap membantu!