

Belajar Git untuk Pemula



ditulis oleh Jaka Maulana Ibrahim

Belajar Git Untuk Pemula

Ditulis oleh Jaka Maulana Ibrahim

GIT merupakan Sistem Kontrol Versi terdistribusi yang digunakan untuk melacak perubahan dalam kode sumber selama pengembangan perangkat lunak. Dengan git, beberapa pengembangan dapat bekerja secara bersamaan pada proyek yang sama tanpa konflik, memungkinkan kolaborasi yang lebih efisien dan pengelolaan versi kode yang lebih baik.

Git ini merupakan sebuah brand dari type software Version Control System. Sama seperti GIT, SVN dan Mercurial.

VSC adalah teknologi yang memungkinkan anda untuk mengelola perubahan pada file dan proyek secara terstruktur dan terorganisir. VSC memungkinkan untuk merekam perubahan pada file dan proyek, melacak perubahan yang telah dilakukan, mengembalikan versi sebelumnya jika diperlukan, berkolaborasi dengan tim lain dalam mengelola proyek. VCS sangat penting dalam pengembangan perangkat lunak, karena memungkinkan anda untuk mengelola perubahan dan memastikan bahwa semua anggota tim memiliki versi yang sama dari proyek.

Git sendiri di ciptakan oleh Linus Torvaldo yap orang yang sama yang menciptakan Linux. Linus mulai mengembangkan GIT pada tahun 2005 bulan april. Untuk membantu mengontrol source code karnel Linux. Menggantikan peran BitKeeper yang pada tahun 2002 dihentikan. Dan juga pada momen yang sama ada yang membuat Version Control System sejenis, yaitu Mercurial.

Banyak sekali VCS selain GIT yang terkenal di antaranya ada Subversion , TFVC , Mercurial , CVS Perforce , VSS , IBM DevOps Code ClearCase , Zip file backups, Raw network sharing. Dan lain-lain. Tetapi perkembangan dari tahun 2015 sampai tahun 2022. terdata bahwa pengguna git sudah tumbuh dari 69.3% sampai 93.9% yang artinya disini bahwa sebagian besar programmer sudah beralih menggunakan git untuk VCS nya.

Untuk menginstall git pada linux turunan debian/ubuntu cukup untuk ketik :

“sudo apt install git“

untuk memastikan apakah git sudah terinstall dengan benar atau belum dapat menuliskan perintah :

“git -version” nantinya akan ada keluar output seperti “git version 2.34.1“

Daftar Isi

- Pendahuluan	02
- Daftar Isi	03
- Git macam-macam perintah dasar	04
Menginisialisasi project dengan git init dan melakukan gitclone	
- Menambahkan file baru,dan melakukan git add	05
- GIT Reset perubahan file dengan git reset	07
- GIT Melakukan commit,mempraktekan diff dan log	08
- GIT Melakukan unggah file dengan git push	12
- GIT Melakukan unduh file dengan git pull	14
- GIT Bermain main dengan fetch dan branch	16
- GIT Menyatukan branch satu dengan lainnya, git merge	19
- GIT Menandai milestone project dengan git tag	22
- Daftar pustaka	24

Git Macam-macam perintah GIT Dasar

git memiliki kode perintah yang di gunakan untuk perintah tertentu dalam system untuk melakukan beberapa proses git. Perintah ini sering digunakan di terminal untuk menjalankan systemnya. Berikut Macam-macam perintah GIT Dasar :

(1) Create = Digunakan Untuk membuat repo git

- git Init = untuk menginisialisasi

disini git akan membuat sebuah folder tersembunyi untuk menyimpan sebuah perubahan dari proyek yang telah kita kembangkan atau kita ubah.

- git clone <URL Projeknya> = untuk mengambil proyek dari proyek orang lain yang ada di git , file nya akan muncul di file baru maintainance

(2) Make a Change = untuk melakukan perubahan dan pengecekan prubahan

- git add <File> = menambahkan file tertentu ke staging area untuk dipersiapkan commit.

- git add . = Menambahkan semu perubahan (file baru, modifikasi) di direktori kerja ke staging area

- git commit -m "<Message>" = menyimpan snapshot dari staging area ke dalam riwayat repository dengan pesan deskriptif.

- git reset <File> = menghapus file dari staging area tapi tidak menghapus perubahan dari working directory. Cocok jika ingin rivisi sebelum commit.

- git reset --hard =Menghapus semua perubahan (baik di staging area maupun working directory) dan kembali ke commit terakhir. Hati-hati karena ini tidak dapat di batalkan.

(3) Observe untuk mengobservasi proyek

- git status = Menampilkan status working directory dan staging area: file mana yang diubah, belum ditambahkan, atau siap di-commit.

- git diff = Menampilkan perbedaan isi file yang telah diubah tapi belum di-staging

- git show = digunakan untuk **menampilkan informasi detail dari objek Git tertentu**, seperti *commit* terakhir (perubahan yang sudah di-*commit*, penulis, tanggal, dan pesan *commit*), *tag*, atau *tree*. Jika dijalankan tanpa argumen, perintah ini akan menampilkan detail dari *commit* terakhir.

- git log = Menampilkan riwayat commit lengkap dengan hash, pesan commit, tanggal, dan penulis. Bisa ditambahkan argumen untuk filtering.

(4) Sync untuk push pull ke server

- git push = mengirimkan commit lokal ke remote repository. Biasanya digunakan setelah git commit.

- git pull = mengambil dan menggabungkan (merge) perubahan dari remote repository ke local. Sama dengan git fetch lalu git merge

- git fetch = mengambil update dari remote repository, tanpa menggabungkannya langsung. Cocok untuk mengecek dulu sebelum merge.

(5) Branch untuk melakukan versioning

- git branch = Menampilkan daftar branch yang ada atau membuat branch baru (git branch <nama-branch>).

- git checkout = berpindah ke branch tertentu.juga bisa digunakan untuk mengembalikan file di *working directory* ke versi terakhir yang tercatat di *commit* terakhir, membatalkan perubahan yang belum di-*staging*.

- git merge = Menggabungkan perubahan dari branch tertentu ke branch saat ini.

- git tag = memberi label pada commit tertentu, biasanya digunakan untuk menandai versi (v1.0, v2.0 ,dll)

Menginisialisasi project dengan git init dan mencoba clone

sekarang kita akan mencoba menginisialisasi proyek dengan git init dan cara mengunduh proyek dengan git clone. Keduanya adalah titik awal saat bekerja dengan Git, tetapi digunakan dalam situasi yang sangat berbeda.

git init : digunakan untuk membuat repositori baru dari nol di dalam folder proyek yang sudah ada di komputer

git clone : digunakan untuk mengunduh salinan repositori yang sudah ada dari server (seperti GitHub) ke komputer anda.

Cara Menginisialisasi Project baru dengan git init

Gunakan Perintah ini ketika sudah memiliki folder berisi file proyek di komputer, tetapi belum dilacak oleh git. Git init akan mengubah folder tersebut menjadi sebuah repositori git.

1. memulai sebuah proyek baru dari awal di komputer local.
2. ingin menambahkan kontrol versi (GIT) ke proyek yang sudah ada.

Langkah-langkah git init

1. buka terminal di Vscode sajah biar lebih mudah.
2. masuk ke direktori / folder utama proyek menggunakan perintah “cd” (change directory).
3. jalankan perintah git init.
4. Git akan memberikan respons seperti Ini “git init Reinitialized existing Git repository in /home/jaka/Documents/Codepolitan/belajargit/filehtml/.git/” artinya git telah berhasil membuat sebuah sub-folder tersembunyi bernama .git. Folder inilah yang berisi semua riwayat perubahan dan konfigurasi repositori Anda.
5. cek kembali foldernya dengan menggunakan ls -a (untuk melihat semua file yang di hidden).



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[14:59:20] [cost 0.025s] cd Documents/Codepolitan/belajargit/filehtml
[14:59:22] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> pwd
/home/jaka/Documents/Codepolitan/belajargit/filehtml
[14:59:24] [cost 0.027s] pwd
[14:59:26] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> ls
main.html
[14:59:27] [cost 0.028s] ls
[15:05:38] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> git init
Reinitialized existing Git repository in /home/jaka/Documents/Codepolitan/belajargit/filehtml/.git/
[15:06:01] [cost 0.038s] git init
[15:06:03] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> ls
main.html
[15:06:04] [cost 0.029s] ls
[15:07:20] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> ls -a
. . .git main.html
[15:07:22] [cost 0.029s] ls -a
```

Gambar 1 : Contoh melakukan git init di terminal vscode

Cara mengunduh proyek dengan git clone

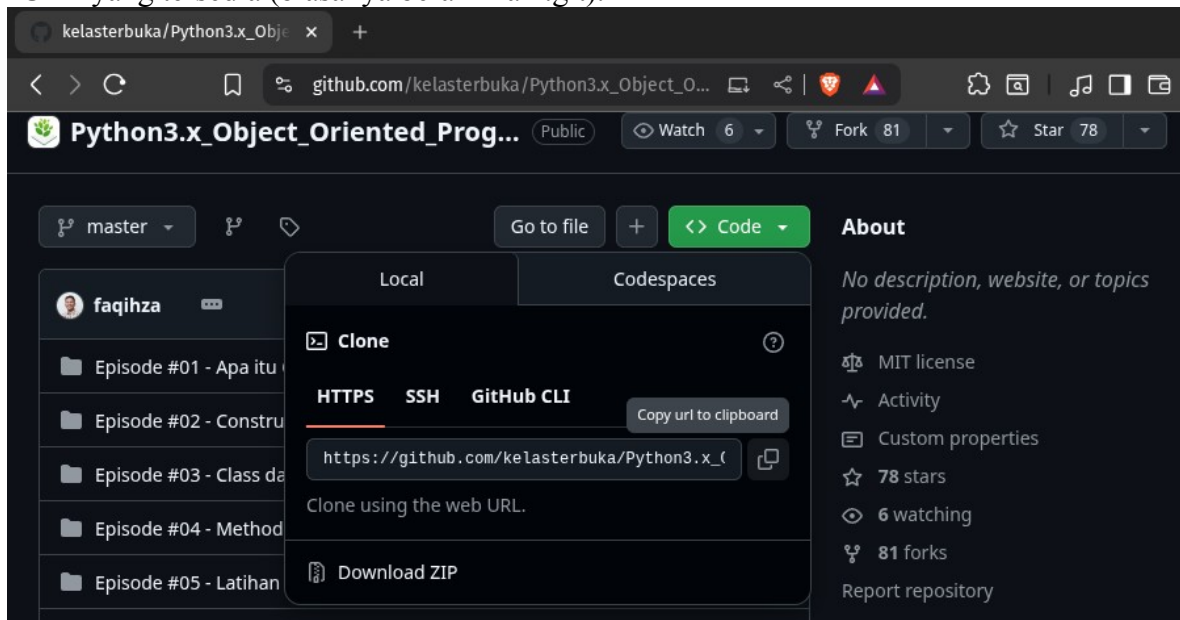
gunakan perintah ini ketika ingin bekerja pada sebuah proyek yang sudah ada di platform seperti GitHub, GitLab, atau Bitbucket. Git Clone akan mengunduh seluruh proyek beserta riwayatnya ke komputer.

Gunakan ketika :

- ingin berkontribusi pada proyek open-source.
- ingin mengunduh proyek milik tim untuk mulai bekerja.
- ingin membuat salinan proyek sendiri di komputer lain.

Langkah -langkah :

1. Dapatkan URL repositori. Di halaman Github (atau sejenisnya), klik tombol hijau “code” dan salin URL yang tersedia (biasanya berakhiran .git).



2. Buka terminal di vscode saja biar mudah.
3. masuk ke direktori tempat anda ingin menyimpan proyek tersebut atau membuat folder baru untuk menyimpan proyek tersebut.
4. jalankan perintah “git clone diikuti dengan URL yang sudah anda salin.
5. cek kembali folder dengan perintah “LS” apakah file sudah terunduh dengan benar atau belum.
6. Selesai Git akan membuat folder baru dengan nama yang sama seperti nama repositori (nama-repositori), mengunduh semua file, dan secara otomatis menghubungkan repositori lokal Anda ke repositori di server (yang disebut origin). Anda bisa langsung masuk ke folder tersebut dan mulai bekerja.

```
[15:42:37] [cost 0.011s] mkdir belajargitclone
[15:42:39] [~/Documents/Codepolitan/belajargit] >>> cd belajargitclone
[15:42:48] [cost 0.010s] cd belajargitclone

[15:42:49] [~/Documents/Codepolitan/belajargit/belajargitclone] >>> ls
[15:42:50] [cost 0.012s] ls

[15:42:53] [~/Documents/Codepolitan/belajargit/belajargitclone] >>> pwd
/home/jaka/Documents/Codepolitan/belajargit/belajargitclone
[15:42:54] [cost 0.010s] pwd

[15:43:21] [~/Documents/Codepolitan/belajargit/belajargitclone] >>> git clone https://github.com/kelasterbuka/Python3.x_Object_Oriented_Programming.git
Cloning into 'Python3.x_Object_Oriented_Programming'...
remote: Enumerating objects: 56, done.
remote: Total 56 (delta 0), reused 0 (delta 0), pack-reused 56 (from 1)
Receiving objects: 100% (56/56), 12.33 KiB | 12.33 MiB/s, done.
Resolving deltas: 100% (3/3), done.
[15:43:35] [cost 1.816s] git clone https://github.com/kelasterbuka/Python3.x_Object_Oriented_Programming.git
[15:43:39] [~/Documents/Codepolitan/belajargit/belajargitclone] >>> ls
Python3.x_Object_Oriented_Programming
[15:43:40] [cost 0.013s] ls
```

Contoh gambar proses git clone menggunakan terminal vs code.

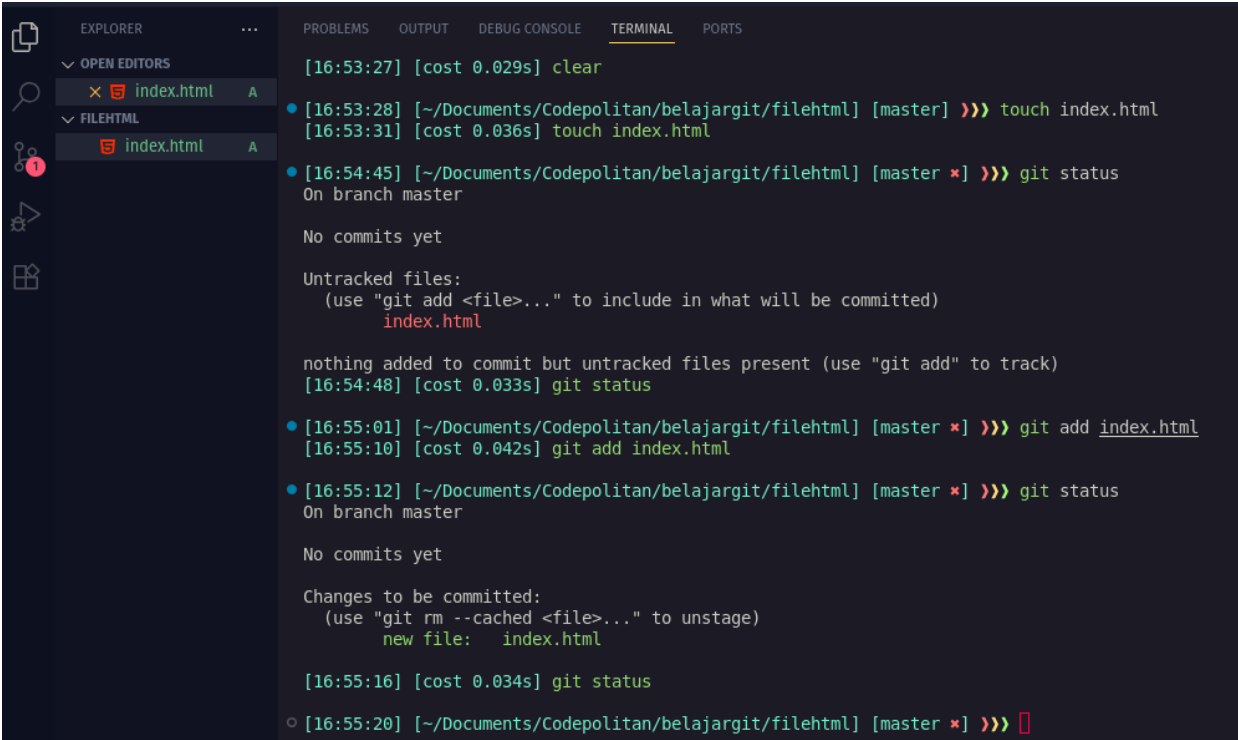
GIT Menambahkan file baru, dan melakukan git add

mari kita lanjutkan proses setelah sebuah proyek diinisialisasi dengan git init, langkah selanjutnya adalah membuat atau memodifikasi file, lalu memberikan Git file mana yang perubahannya ingin disimpan. Proses ini menggunakan perintah “git add”.

Perintah git add berfungsi untuk menambahkan perubahan dari working directory (folder kerja) ke traging area. Anggap saja staging area adalah sebuah “panggung” atau area transit tempat mengumpulkan semua perubahan yang siap untuk disimpan secara permanen dalam satu commit.

Langkah – langkah menambah file baru

1. pilih sebuah file baru dalam folder proyek. Bisa membuatnya menggunakan Vscode atau dari terminal dengan perintah “touch index.html”.
2. buat sajah proyek sederhana didalam index.html untuk latihan git sajah.
3. sebelum menambahkan file, sangat disarankan untuk selalu memeriksa status repository dengan perintah “git status”.
4. git mungkin akan menampilkan bahwa ada file yang belum dilacak (untracked files). Ini berarti git tahu ada file baru, tetapi belum memantaunya.
5. sekarang, gunakan perintah “git add” untuk mempersiapkan file tersebut agar bisa di-commit
 - jika hanya menambah 1 file sajah gunakan nama filenya secara langsung
bash : git add index.html
 - jika menambah semua file atau banyak file baru makan cukup dengan “.” sajah
bash : git add .
4. cek status kembali dengan git status, kita akan melihat bahwa status file index.html telah berubah dari untracked files menjadi changes to be committed. Ini menandakan file anda sudah berhasil masuk ke staging area dan siap untuk di commit.



```

[16:53:27] [cost 0.029s] clear
[16:53:28] [~/Documents/Codepolitan/belajargit/filehtml] [master] >>> touch index.html
[16:53:31] [cost 0.036s] touch index.html
[16:54:45] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.html

nothing added to commit but untracked files present (use "git add" to track)
[16:54:48] [cost 0.033s] git status
[16:55:01] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>> git add index.html
[16:55:10] [cost 0.042s] git add index.html
[16:55:12] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html

[16:55:16] [cost 0.034s] git status
[16:55:20] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>>
  
```

Contoh penggunaan git add di terminal vcs

GIT Reset perubahan file dengan git reset

perintah git reset adalah alat yang sangat kuat untuk membatalkan perubahan, tetapi penggunaannya bisa sedikit berbeda tergantung pada apa yang ingin anda capai. Fungsi utamanya adalah untuk “mengatur ulang” kondisi, baik itu di staging area maupun di riwayat commit.

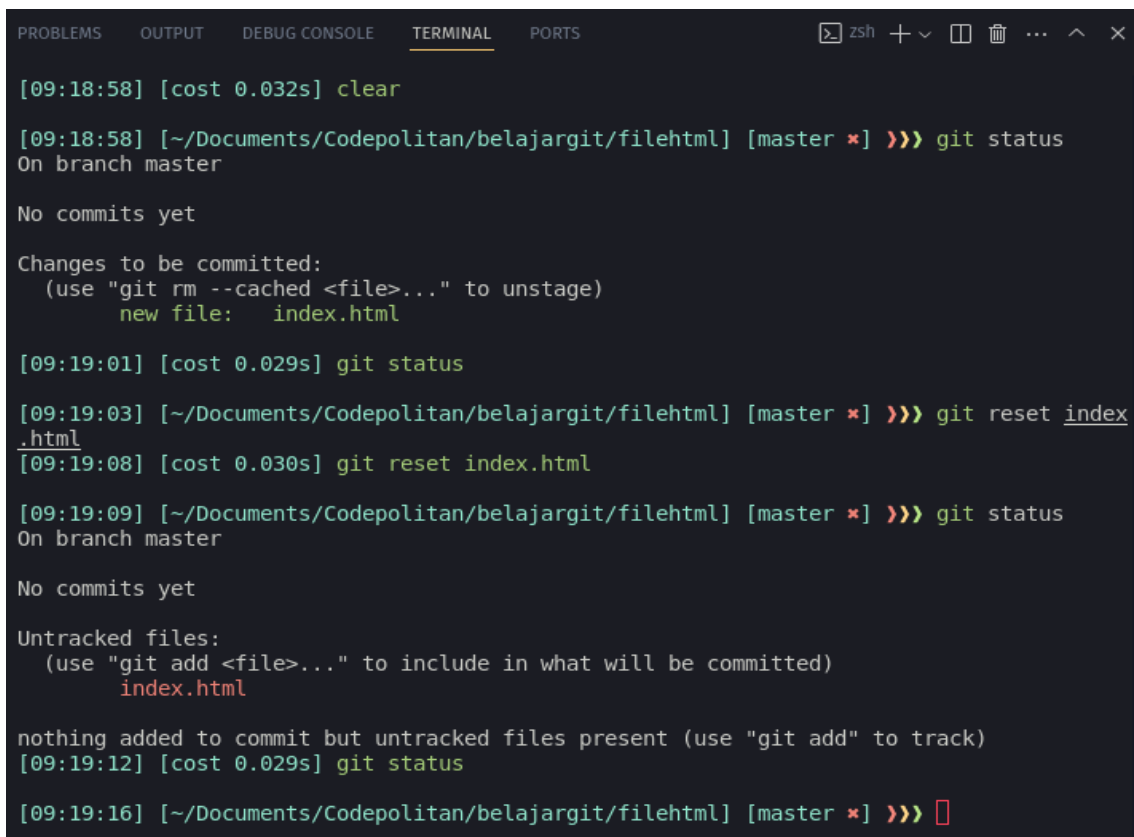
Membatalkan add (Unstaging a file)

ini adalah penggunaan “git reset” yang paling aman dan paling umum. Saat menggunakan ini ketika sudah menjalankan git add pada sebuah file, tetapi kemudian sadar ada yang salah dan tidak ingin memasukan perubahan tersebut ke dalam commit selanjutnya.

Perintah ini akan mengeluarkan file dari steging area, tetapi tidak akan menghapus perubahan yang sudah ada buat pada file di working directory.

Langkah-langkah

1. kita akan kembali memfokuskan pada file “index.html” pada saat ini kita sudah melakukan “git add” pada file index.html
2. cek status untuk melihat apakah file “index.html” masih berada di area steging.
3. untuk menghilangkan atau menghapus file “index.html” kita hanya perlu melakukan perintah “git reset” caranya “git reset index.html” maka file yang index.html yang sudah ada di staging akan menghilang dari area staging.



```

[09:18:58] [cost 0.032s] clear

[09:18:58] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html

[09:19:01] [cost 0.029s] git status

[09:19:03] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> git reset index.html
[09:19:08] [cost 0.030s] git reset index.html

[09:19:09] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)
[09:19:12] [cost 0.029s] git status

[09:19:16] [~/Documents/Codepolitan/belajargit/filehtml] [master ✖] >>> 

```

4. lakukan “git status” untuk memastikan apakah perintah “git reset” sudah berfungsi atau belum. Contoh gambar penggunaan fungsi git reset

5. **WARNING!!!** : jika ingin menghapus semua file yang berada di staging area kita cukup mengetikan perintah “git reset --hard” berhati-hatilah dengan ini karena dapat menghapus semuanya

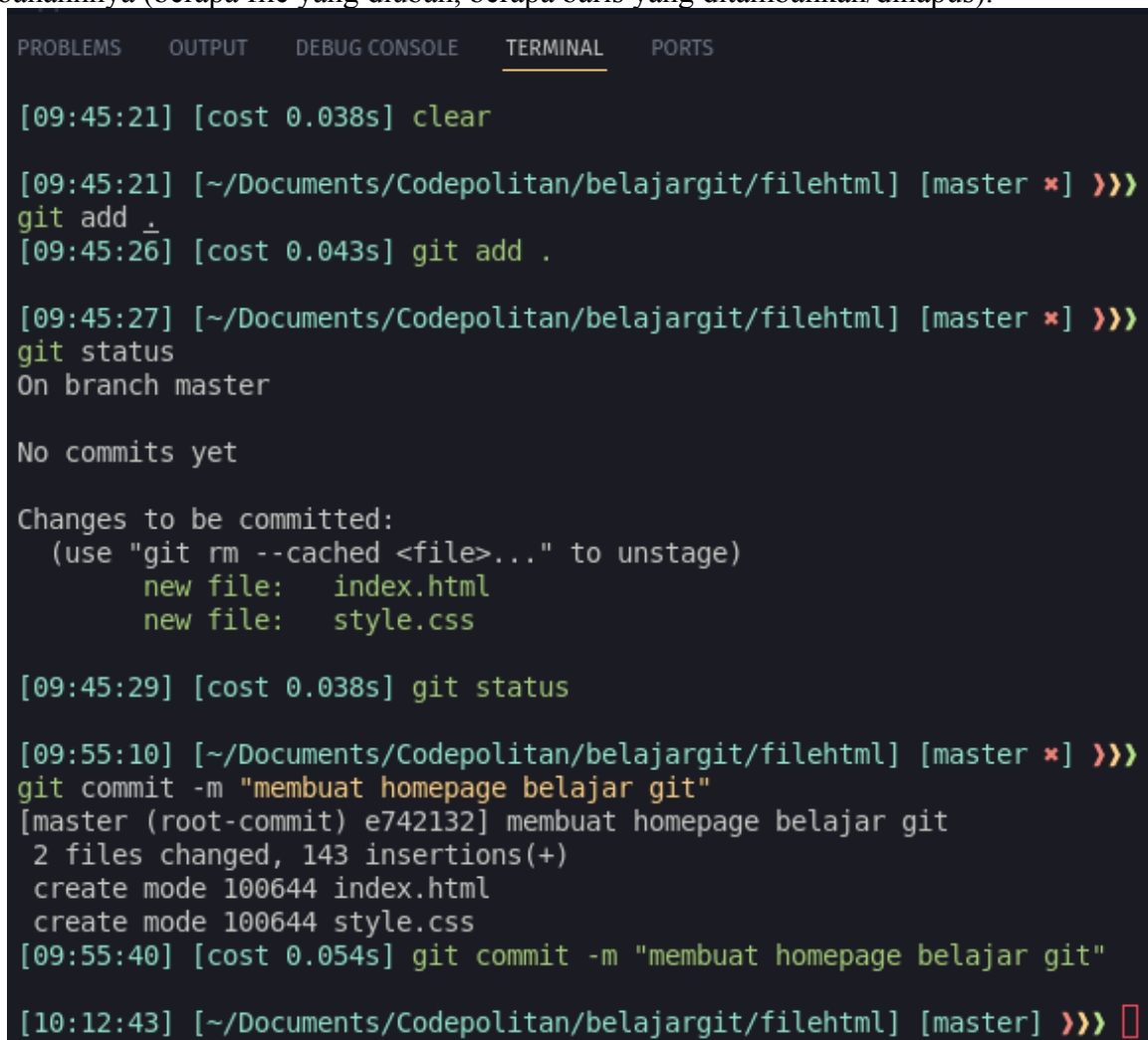
GIT Melakukan commit, mempraktekan diff dan log

perintah git commit berfungsi untuk mengambil “foto” atau snapshot dari semua perubahan yang telah disiapkan di staging area (yang sudah di-git add) dan menyimpannya secara permanen ke dalam riwayat repository lokal.

Setiap commit memiliki ID unik (disebut dengan hash) dan wajib disertai dengan pesan yang deskriptif. Pesan ini sangat penting untuk memahami perubahan apa yang terjadi pada commit tersebut.

Langkah – langkah :

1. kembali lagi pada file “index.html” di sini saya menambahkan file “style.css” kedalamnya. Untuk menambahkannya ke area staging kita perlukan perintah “git add .” ini akan memindahkan semua file yang ada ke area staging.
2. cek dengan “git status” apakah kedua file berada pada area staging.
3. lakukan perintah //git commit -m “Membuat homepage belajar git”// perintah -m singkatan dari message untuk menulis pesan commit langsung dari terminal.
4. git akan memberikan konfirmasi bahwa commit berhasil dibuat, berserta ringkasan perubahannnya (berapa file yang diubah, berapa baris yang ditambahkan/dihapus).



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[09:45:21] [cost 0.038s] clear

[09:45:21] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>>
git add .
[09:45:26] [cost 0.043s] git add .

[09:45:27] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>>
git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   index.html
    new file:   style.css

[09:45:29] [cost 0.038s] git status

[09:55:10] [~/Documents/Codepolitan/belajargit/filehtml] [master *] >>>
git commit -m "membuat homepage belajar git"
[master (root-commit) e742132] membuat homepage belajar git
2 files changed, 143 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
[09:55:40] [cost 0.054s] git commit -m "membuat homepage belajar git"

[10:12:43] [~/Documents/Codepolitan/belajargit/filehtml] [master] >>> 
  
```

Contoh : penggunaan git commit -m pada terminal vcs

Melihat perbedaan dengan git diff. perintah git diff sangat berguna untuk melihat perbedaan spesifik pada isi file. Ada 2 skenario utama penggunaan.

Langkah – langkah

A : melihat perubahan yang belum di staging

gunakan perintah “git diff” tanpa argumen apa pun untuk melihat perbedaan antara file di working directory dengan versi terakhir yang ada di staging area (atau commit terakhir jika staging area kosong).

1. ubah file index.html tanpa menjalankannya misalnya, menambah paragraf baru.
2. jalankan git diff. Output akan berupa penjelasan apa saja yang diganti atau di tambahkan :

```
diff --git a/index.html b/index.html
index 37da86f..92de3d9 100644
--- a/index.html
+++ b/index.html
@@ -23,5 +23,8 @@
     <li>Menguasai perintah dasar Git</li>
   </ul>

+   <h2>Penulis Favorit</h2>
+   <p>Beberapa penulis favorit saya adalah Khalil Gibran, Carl Sagan,
+   dan Ustadz Yazid bin Abdul Qadir Jawas.</p>
+
+ </body>
+ </html>
\ No newline at end of file
```

contoh gambar git diff terlihat pada colom + ada paragraf yang ditambahkan pada index.html

B : melihat perubahan yang sudah di staging

gunakan git diff --staged (atau --cached) untuk melihat perbedaan antara file yang sudah masuk ke staging area dengan versi terakhir yang ada di commit.

1. tambahkan perubahan tadi ke staging area, caranya dengan “git add .”
2. jalankan perintah “git diff --staged. Outputnya akan memberitahu apa saja yang telah di tambahkan.

```
diff --git a/index.html b/index.html
index 37da86f..92de3d9 100644
--- a/index.html
+++ b/index.html
@@ -23,5 +23,8 @@
     <li>Menguasai perintah dasar Git</li>
   </ul>

+   <h2>Penulis Favorit</h2>
+   <p>Beberapa penulis favorit saya adalah Khalil Gibran, Carl Sagan,
+   dan Ustadz Yazid bin Abdul Qadir Jawas.</p>
+
+ </body>
+ </html>
\ No newline at end of file
(END)
```

Contoh gambar git diff --staged perubahan ada di kolom (+) beberapa paragraf telah ditambahkan.

3. jangan lupa untuk di commit kembali.

Melihat Riwayat commit dengan git log. Perintah git log digunakan untuk menampilkan seluruh riwayat commit yang telah anda buat, dari yang paling baru hingga yang paling lama.

Langkah – langkah :

1. cukup jalankan perintah git log di terminal.
2. Lihat hasilnya. Setiap entri *commit* akan menampilkan: Hash: ID unik dari *commit* (contoh: 1a2b3c4...), Author: Siapa yang membuat *commit*, Date: Kapan *commit* dibuat, Pesan Commit: Pesan yang Anda tulis saat melakukan *commit*.
3. tekan “q” untuk keluar dari tampilan log.
4. gunakan git log –oneline untuk melihat riwayat dalam format yang lebih ringkas.

```
commit e55c0a81d1395b7ee093248a2d64f897a7ecc968 (HEAD -> master)
Author: jakajah <jaka.mibrahim12@gmail.com>
Date:   Fri Jul 11 10:43:31 2025 +0700

    membuat homepage belajar git

commit e74213269a24e1186b6e89b9b075a8c19fd9b6c9
Author: jakajah <jaka.mibrahim12@gmail.com>
Date:   Fri Jul 11 09:55:40 2025 +0700

    membuat homepage belajar git
(END)
```

Gambar contoh output untuk perintah git log

perintah git show untuk menampilkan berbagai enis objek git secara detail yang paling umum dan sering digunakan adalah untuk menampilkan detail sebuah commit dalam terminal yang sudah berada fi folder proyek kita cukup ketikan sajah git show maka akan keluar detail enis objek yang sedang ingin kita kembangkan.

```
commit e55c0a81d1395b7ee093248a2d64f897a7ecc968 (HEAD -> master)
Author: jakajah <jaka.mibrahim12@gmail.com>
Date:   Fri Jul 11 10:43:31 2025 +0700

    membuat homepage belajar git

diff --git a/index.html b/index.html
index 37da86f..92de3d9 100644
--- a/index.html
+++ b/index.html
@@ -23,5 +23,8 @@
     <li>Menguasai perintah dasar Git</li>
   </ul>

+   <h2>Penulis Favorit</h2>
+   <p>Beberapa penulis favorit saya adalah Khalil Gibran, Carl Sagan, dan Ustadz Ya
zid bin Abdul Qadir Jawas.</p>
+
   </body>
 </html>
\ No newline at end of file
(END)
```

GIT Melakukan unggah file dengan git push

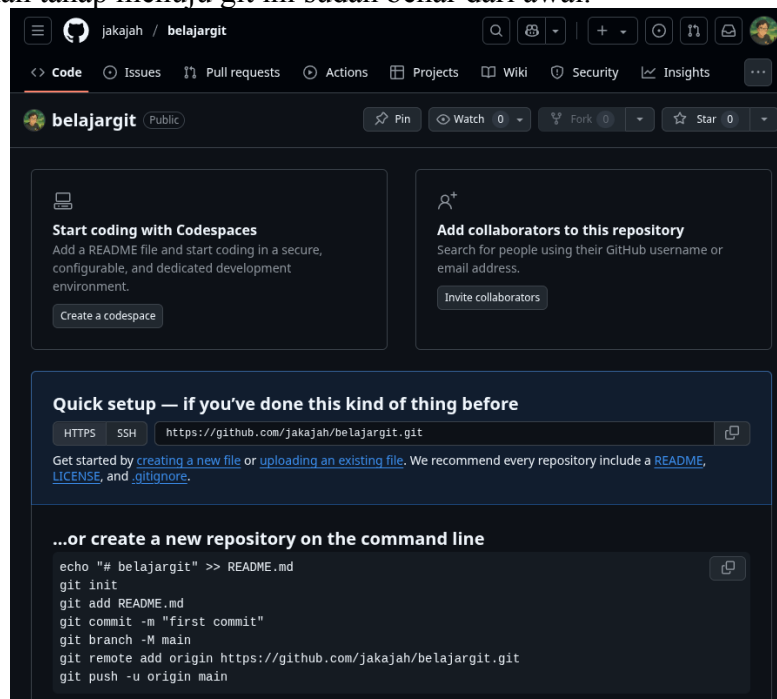
git push adalah perintah untuk mengunggah atau “mendorong” riwayat commit dari repositori lokal ke remote repositori (seperti GitHub, Gitlab, atau Bitbucket). Ini adalah cara membagikan pekerjaan atau membantu cadangan secara online.

Sebelum bisa push, pastikan terlebih dahulu perubahan sudah di commit, kita hanya bisa mendorong commit, bukan file yang belum disimpan. Pastikan sudah menjalankan git add dan git commit. Pastikan juga Repositori Lokal terhubung ke remote, Repositori lokal harus tahu ke mana harus mengirimkan perubahannya. Jika kita memulainya dengan git clone, koneksi ini otomatis dibuat. Jika memulainya dengan git init. Kita harus menghubungkannya secara manual dengan git remote add.

Sebelum kita mulai langkah-langkah ini kita memerlukan server dan kita akan menggunakan server git instan salah satunya yaitu Github, silahkan teman-teman mendaftar terlebih dahulu ke github.

Langkah -langkah :

1. buat repositori baru pada github. setelah itu pastikan file index.html kita sudah pada tahap git commit. Dan pastikan tahap menuju git ini sudah benar dari awal.



2. copy link pada quick setup kita copy yang HTTPS nya.

3. ketik git branch -m main , untuk memilih menggunakan branch utama

4. pada terminal kita ketikan “git remote add origin <https://github.com/jakajah/belajargit.git>” (isi https sesuai dengan link repositori masing-masing.

5. untuk melihat sudah terintegrasi atau belum kita lihat dengan “git remote -v”

(WARNING : biasanya pada tahap ini sering terjadi diminta login ke github melalui terminal, silahkan kalian login dulu ke github melalui vscode nya, atau mengintegrasikan vscode ke github)

6. untuk melakukan push kita lakukan “git push -u origin main” (-u membantu adalah singkatan – set-upstream) berfungsi untuk membuat dan mengingat koneksi permanen antara beanch loal dan banch di remot. Dengan -u nanti nya ketika kita ingin nge push kembali kita hanya perlu mengetikan “git push”

```

[08:50:36] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git branch -M main
[08:50:41] [cost 0.042s] git branch -M main

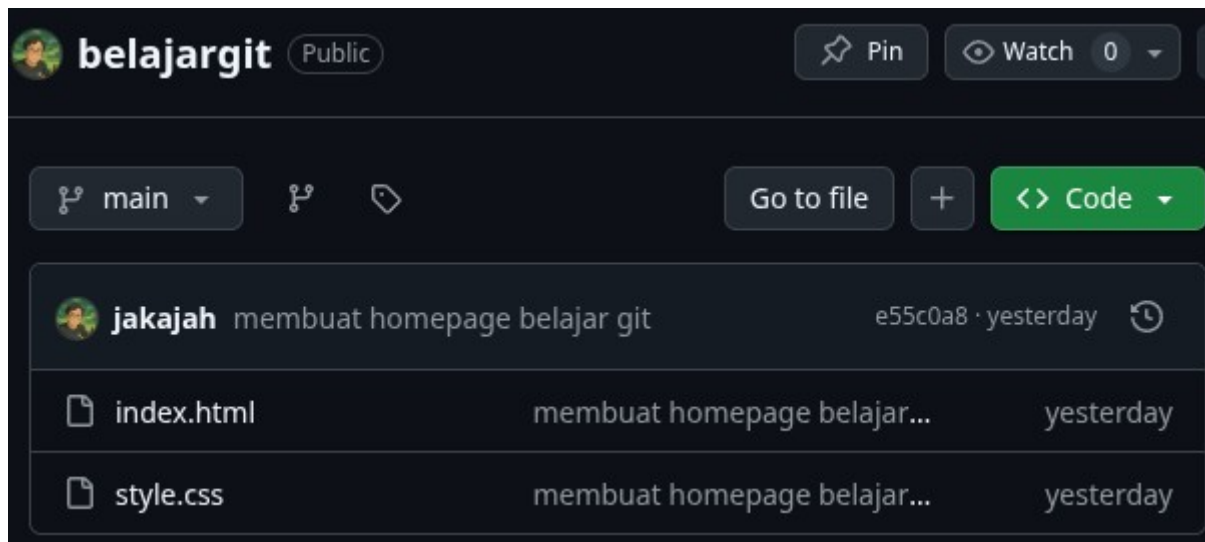
[08:50:55] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git remote add origin https://github.com/jakajah/belajargit.git
[08:51:11] [cost 0.044s] git remote add origin https://github.com/jakajah/belajargit.git

[08:51:28] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git remote -v
origin https://github.com/jakajah/belajargit.git (fetch)
origin https://github.com/jakajah/belajargit.git (push)
[08:51:34] [cost 0.044s] git remote -v

[08:51:46] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git push -u origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 2.28 KiB | 2.28 MiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jakajah/belajargit.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
[08:52:07] [cost 3.523s] git push -u origin main

```

Contoh gambar git push pada terminal vscode



Contoh gambar repositori vscode yang berada di branch main dan berhasil melakukan git push

catatan :

- untuk menghapus branch yang salah pada remote
 git push origin --delete fiturlogin = perintah tersebut untuk menghapus branch remote yang bernama fiturlogin.

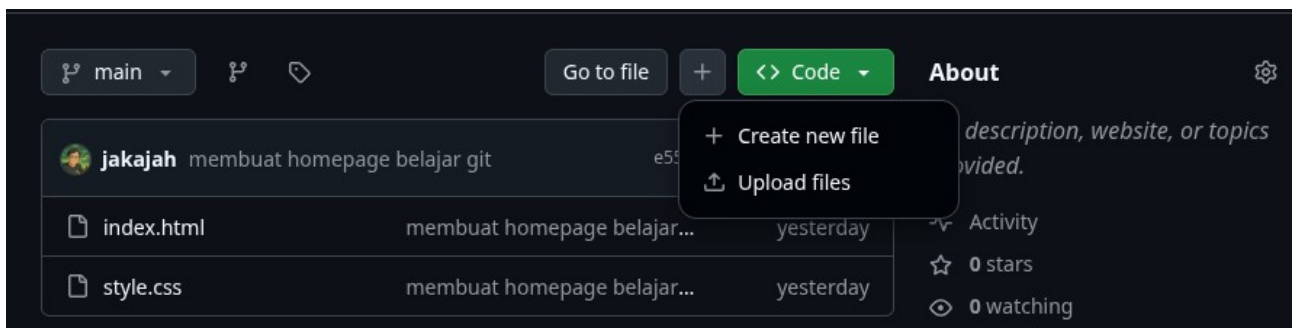
GIT Melakukan unduh file dengan git pull

git pull adalah perintah yang digunakan untuk mengunduh dan mengintegrasikan perubahan dari remote repository (seperti GitHub) ke branch lokal. Secara sederhana git pull adalah gabungan dari dua perintah lain “git fetch (mengambil data)” diikuti oleh “git merge” (menggabungkan data).

Saat kita menjalankan git pull, git melakukan dua hal secara berurutan, **Fetch (Mengambil)**: Git menghubungi *remote repository* (misalnya origin) dan mengunduh semua data baru yang belum Anda miliki, seperti *commit* baru yang dibuat oleh rekan tim Anda. Data ini disimpan di repositori lokal Anda tetapi belum digabungkan ke pekerjaan Anda. **Merge (Menggabungkan)**: Setelah selesai mengunduh, Git secara otomatis mencoba menggabungkan (*merge*) riwayat *commit* yang baru diunduh ke dalam *branch* tempat Anda bekerja saat ini.

Langkah – langkah

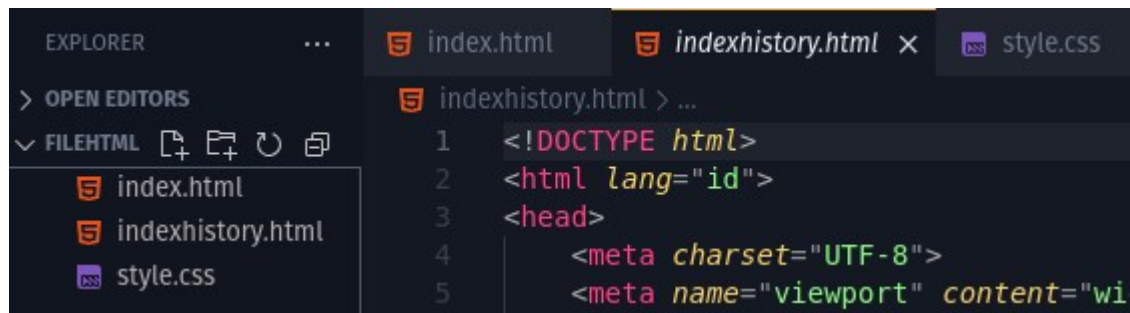
1. masih di repositori yang sama pada github, kita tambahkan create new file lalu tambahkan apa saja file codingan ke inginan kalian seolah olah itu adalah file baru milik teman yang baru di push ke repository github. Setelah itu lakukan commit changes untuk menaruhnya di repositori remote.



2. kembali ke VSC code buka terminal vscode nya lalu ketikan “git pull origin main” maka semua yang ada di branch main akan terunduh di lokal komputer.

```
[09:20:17] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 4.43 KiB | 4.43 MiB/s, done.
From https://github.com/jakajah/belajargit
* branch          main       -> FETCH HEAD
e55c0a8..d49f905  main       -> origin/main
Updating e55c0a8..d49f905
Fast-forward
 indexhistory.html | 282 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 282 insertions(+)
 create mode 100644 indexhistory.html
[09:20:28] [cost 2.765s] git pull origin main

[09:20:54] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> 
```



Contoh gambar melakukan git pull di terminal dan melihat hasil unduhannya di vscode.

GIT Bermain main dengan fetch dan branch

mari kita "bermain-main" dengan git branch dan git fetch. Memahami keduanya adalah kunci untuk bekerja secara efisien dalam sebuah tim dan mengelola perubahan dengan aman. git branch memungkinkan Anda membuat "cabang" atau lingkungan kerja terisolasi, sedangkan git fetch adalah cara aman untuk melihat pembaruan dari rekan tim tanpa mengganggu pekerjaan Anda.

Bayangkan **branch** sebagai sebuah dunia alternatif dari proyek Anda. Anda bisa membuat *branch* baru untuk mengerjakan sebuah fitur, memperbaiki *bug*, atau sekadar bereksperimen tanpa mengacaukan *branch* utama (*main*) yang stabil.

Perintah Dasar Branch

1. git branch : perintah ini akan menampilkan semua branch yang ada di lokal. Tanda * menunjukan branch yang sedang aktif

```
fiturlogin
* main
(END)
```

2. git branch fitur-login : perintah ini akan membuat branch bernama fitur-login, akan tetapi kita belum berpindah ke dalamnya. Jadi hanya bikin saja seperti perintah mkdir pada linux.

```
[10:42:51] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git
branch fiturlogin
[10:43:00] [cost 0.043s] git branch fiturlogin
```

3. git checkout fitur-login : perintah ini membuat kita berpindah ke branch bernama fitur-login.

```
[10:44:38] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git
checkout fiturlogin
Switched to branch 'fiturlogin'
[10:44:44] [cost 0.044s] git checkout fiturlogin
```

4. git checkout -b fiturlogin : perintah ini akan membuat sekaligus berpindah branch bernama fiturlogin.

```
[14:14:08] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git branch
[14:14:16] [cost 4.930s] git branch

[14:16:15] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git checkout -b f
iturlogin
Switched to a new branch 'fiturlogin'
[14:16:41] [cost 0.039s] git checkout -b fiturlogin

[14:16:42] [~/Documents/Codepolitan/belajargit/filehtml] [fiturlogin *] >>> git branch
[14:16:52] [cost 4.138s] git branch

[14:17:00] [~/Documents/Codepolitan/belajargit/filehtml] [fiturlogin *] >>> 
```

5. `git branch -D fiturlogin` : perintah ini akan menghapus branch bernama fiturlogin, sebelum kita hapus kita harus keluar dulu dari fiturlogin ke branch main.

```
[14:13:54] [~/Documents/Codepolitan/belajargit/filehtml] [fiturlogin *] >>> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
[14:14:03] [cost 0.039s] git checkout main

[14:14:04] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git branch -D fiturlogin
Deleted branch fiturlogin (was d49f905).
[14:14:06] [cost 0.044s] git branch -D fiturlogin

[14:14:08] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git branch
[14:14:16] [cost 4.930s] git branch

[14:14:30] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> 
```

git fetch adalah perintah untuk **mengunduh** semua informasi terbaru (seperti *commit* dan *branch* baru) dari *remote repository* (seperti GitHub) ke komputer lokal.

Poin utamanya adalah git fetch **hanya mengunduh data** dan **tidak secara otomatis menggabungkannya** ke dalam pekerjaan utama. Ini membuatnya menjadi cara yang aman untuk melihat pembaruan tanpa mengganggu *branch* yang sedang dikerjakan.

Anggap git fetch seperti menekan tombol "Refresh" atau "Muat Ulang" pada linimasa media sosial. kita bisa melihat semua postingan terbaru dari teman-teman. Namun, postingan tersebut tidak secara otomatis masuk atau mengubah status yang sedang ditulis. kita hanya mendapatkan informasi terbaru, lalu kita bisa memutuskan apa yang akan dilakukan dengan informasi tersebut.

ini adalah perbedaan yang paling penting untuk dipahami:

git fetch: Hanya mengunduh. Kita bisa meninjau perubahan terlebih dahulu, lalu menggabungkannya secara manual nanti dengan git merge. Ini memberi kontrol penuh.
Proses: Unduh → Tinjau (opsional) → Gabung manual.

git pull: Mengunduh **DAN** langsung menggabungkan. Perintah ini adalah jalan pintas yang menggabungkan git fetch dan git merge dalam satu langkah.
Proses: Unduh + Gabung Otomatis.

Gunakan git fetch ketika ingin: Melihat pekerjaan terbaru dari rekan tim tanpa mengganggu pekerjaan kita saat ini. Memeriksa apakah ada pembaruan di *remote* sebelum kita mulai mengerjakan fitur baru. Memiliki kontrol penuh atas kapan dan bagaimana perubahan dari *remote* diintegrasikan ke dalam *branch* lokal.

Langkah-langkah

1. Sebagai contoh saya sudah membuat branch baru bernama “fiturbaru” pada github dan mengisinya dengan HTML penjelasan tentang branch.

The screenshot shows the GitHub web interface for a repository. At the top, it says 'jakajah Create branch.html' with a commit hash '4810d37' and the word 'now'. Below this, a message states 'This branch is 3 commits ahead of main'. There is a 'Contribute' button. A table below lists the commits:

Name	Last commit message	Last commit da...
branch.html	Create branch.html	now

2. saya akan mencoba menariknya ke local komputer saya dengan perintah git fetch. Kita cek terlebih dahulu branch yang ada di lokal kita , tidak ada branch bernama “fiturbaru”

```
fiturlogin
* main
(END)
```

3. kita pastikan local kita masih terhubung dengan remote jika sudah tidak terhubung kita bisa ketikkan kembali “git remote add origin <https://alamatgithubanda.git>.”

```
[07:45:06] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git remote add origin https://github.com/jakajah/belajargit.git
error: remote origin already exists.
[07:45:22] [cost 0.029s] git remote add origin https://github.com/jakajah/belajargit.git
[07:45:44] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> 
```

4. lalu kita tinggal ketikkan saja git fetch maka branch akan terunduh ke lokal kita, untuk masuk ke branch nya kita ketikkan git checkout fiturbaru (nama branchnya) untuk pindah ke branch baru.

```
[07:45:53] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (4/4), 1.76 KiB | 1.76 MiB/s, done.
From https://github.com/jakajah/belajargit
 f091eec..4810d37 fiturbaru -> origin/fiturbaru
[07:46:15] [cost 1.873s] git fetch
```

5. jika kita cek kembali dengan git branch, branch yang tadi kita unduh tidak dapat terdeteksi. Kita harus pindah terlebih dahulu menggunakan checkout ke branch baru kita unduh dengan git fetch. Maka file serta branch terdeteksi di sistem kita.

```
[07:47:55] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git checkout fiturbaru
Branch 'fiturbaru' set up to track remote branch 'fiturbaru' from 'origin'.
Switched to a new branch 'fiturbaru'
[07:48:38] [cost 0.037s] git checkout fiturbaru
```

```
* fiturbaru
fiturlogin
main
```

Catatan :

jika kita ingin tahu file apa saja yang ada di suatu branch kita cukup mengetikkan perintah :

- git ls-tree main (ini perintah untuk mengetahui file apa saja yang ada di branch main)

GIT Menyatukan branch satu dengan lainnya, git merge

git merge adalah perintah utama di Git untuk menyatukan atau menggabungkan riwayat dari satu *branch* ke *branch* lainnya. Ini adalah proses di mana kita mengambil perubahan dari sebuah *branch* (misalnya, *branch* fitur baru) dan mengintegrasikannya ke dalam *branch* utama (seperti *main*). Bayangkan kita sudah selesai mengerjakan fitur-login di *branch*-nya sendiri. Sekarang, Anda ingin fitur tersebut menjadi bagian dari proyek utama yang ada di *branch* *main*.

Langkah-langkah

1. pindah ke branch tujuan, pertama pastikan kita berada di branch *main*, karena *main* adalah branch yang akan kita perbarui

- git checkout main

untuk memastikan branch *main* sudah sangat sinkron dengan versi terbaru di remote

- git pull origin main

2. jalankan perintah git merge, sekarang jalankan perintah merge dengan menyebutkan nama branch sumber yang perubahannya ingin di ambil.

- git merge fiturbaru

3. jika tidak ada konflik, git akan menggabungkan perubahan secara otomatis. Sekarang, branch *main* sudah memiliki semua pekerjaan yang sebelumnya ada di *fiturbaru*. Kita bisa memeriksanya dengan melihat file – git ls-tree main.

```
[08:30:06] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
[08:30:11] [cost 0.043s] git checkout main

[08:30:13] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git pull origin main
From https://github.com/jakajah/belajargit
* branch          main          -> FETCH_HEAD
Already up to date.
[08:30:21] [cost 0.865s] git pull origin main

[08:30:23] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git merge fiturbaru
Updating d49f905..4810d37
Fast-forward
 branch.html | 85 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 85 insertions(+)
 create mode 100644 branch.html
[08:30:34] [cost 0.036s] git merge fiturbaru

[08:30:39] [~/Documents/Codepolitan/belajargit/filehtml] [main *] >>> git ls-tree main
100644 blob 8f26db96c3e0275044f7cd86b4e232e3526f94c8    branch.html
100644 blob 92de3d9e55cfd96c65d634907893e3db861fae80    index.html
100644 blob 4d750845731b5ed1d99a48465b3d11cd75abf022    indexhistory.html
100644 blob 38286b2bb7ce4e40a10cea62e49afe97c6756dbb    style.css
[08:30:45] [cost 0.032s] git ls-tree main
```

Contoh gambar git merge pada terminal

GIT Menyelesaikan Konflik pada GIT

Konflik terjadi ketika Git mencoba menggabungkan dua *branch* yang telah mengubah **baris kode yang sama di file yang sama**. Git tidak tahu versi mana yang benar, jadi prosesnya berhenti dan meminta Anda untuk membuat keputusan.

Branch main: file kontak.html berisi <p>Hubungi kami via email.</p>

Branch fitur-kontak: Anda mengubah baris yang sama menjadi <p>Hubungi kami via telepon.</p>

Ketika fitur-kontak di-*merge* ke main, Git akan bingung.

Langkah-langkah Menyelesaikan Konflik

1. **Lakukan Merge dan Lihat Pesan Konflik** Saat Anda menjalankan git merge fitur-kontak, Git akan gagal dan menampilkan pesan:

- CONFLICT (content): Merge conflict in kontak.html Automatic merge failed; fix conflicts and then commit the result.

2. **Identifikasi File yang Konflik** Gunakan git status untuk melihat daftar file yang konflik di bawah bagian "Unmerged paths".

- Unmerged paths:

(use "git add <file>..." to mark resolution)
both modified: kontak.html

3. **Buka dan Edit File Konflik** Buka kontak.html. Anda akan melihat penanda yang ditambahkan Git:

<p>

<<<<<<< HEAD

Hubungi kami via email.

=====

Hubungi kami via telepon.

>>>>>>> fitur-kontak

</p>

<<<<<<< HEAD: Awal dari perubahan di *branch* Anda saat ini (main).

- =====: Pemisah antara dua versi.
- >>>>>>> fitur-kontak: Akhir dari perubahan di *branch* fitur-kontak.

Tugas Anda adalah **menghapus semua penanda tersebut** dan menyisakan hanya versi kode yang Anda inginkan. Misalnya, Anda memutuskan untuk menggabungkan keduanya:

<p>

Hubungi kami via email atau telepon.

</p>

4. Tandai Konflik sebagai Selesai (git add) Setelah menyimpan file, gunakan git add untuk memberitahu Git bahwa Anda telah menyelesaikan konflik pada file tersebut.

- git add kontak.html

5. Buat Commit untuk Menyelesaikan Merge Jalankan git commit untuk membuat *commit* baru yang menyelesaikan proses *merge*. Git biasanya sudah menyiapkan pesan *commit* default.

- git commit

Cukup simpan dan tutup editor yang muncul, dan konflik pun selesai.

GIT Menandai milestone project dengan git tag

git tag adalah perintah untuk memberikan **penanda permanen** atau label pada sebuah *commit* spesifik dalam riwayat proyek Anda. Ini sangat berguna untuk menandai *milestone* penting seperti rilis versi (v1.0, v2.0).

Berbeda dengan *branch* yang terus bergerak maju setiap ada *commit* baru, *tag* akan selalu menunjuk ke *commit* yang sama tempat ia dibuat.

Manajemen Versi: Memberi label yang mudah dibaca manusia (v1.0.0) pada *commit* rilis, memudahkan siapa saja untuk kembali ke versi tersebut.

Penanda Penting: Menandai titik-titik penting dalam sejarah proyek selain rilis, seperti alpha-release atau beta-testing.

Stabilitas: Karena tidak bergerak, *tag* menjamin bahwa v1.0 akan selalu merujuk pada kode yang sama persis.

Cara Menggunakan git tag

1. Melihat Daftar Tag

Untuk melihat semua *tag* yang sudah ada di repositori lokal Anda:

Bash

```
git tag
```

2. Membuat Annotated Tag (Cara Terbaik): Gunakan opsi -a (annotated) untuk nama *tag* dan -m untuk pesan.

```
# Membuat tag v1.0 pada commit terakhir
git tag -a v1.0 -m "Versi 1.0 rilis stabil"
```

```
[11:40:45] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git
tag -a "v1.0"
```

```
#
# Write a message for tag:
# v1.0
# Lines starting with '#' will be ignored.
```

```
Versi 1.0
```


3. Melihat Detail Tag

Gunakan `git show` untuk melihat informasi detail sebuah *tag* beserta *commit* yang ditandainya.

Bash

```
git show v1.0
```

4. Mendorong Tag ke Remote (GitHub)

- **Mendorong Satu Tag Spesifik:**

Bash

```
git push origin v1.0
```

```
[11:48:40] [~/Documents/Codepolitan/belajargit/filehtml] [main] >>> git push origin v1.0
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 161 bytes | 161.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jakajah/belajargit.git
 * [new tag]          v1.0 -> v1.0
[11:48:54] [cost 2.574s] git push origin v1.0
```

5. Menghapus Tag

- **Menghapus Tag Lokal:**

Bash

```
git tag -d v1.0-beta
```

```
[11:42:42] [~/Documents/Codepolitan/belajargit/filehtml] [main ✖] >>> git tag -d v1.0
Deleted tag 'v1.0' (was 9288b14)
[11:42:56] [cost 0.043s] git tag -d v1.0
```

•

- **Menghapus Tag di Remote:** Anda perlu "mendorong penghapusan" ke *remote*.

Bash

```
git push origin --delete v1.0-beta
```

```
[11:43:11] [~/Documents/Codepolitan/belajargit/filehtml] [main ✖] >>> git push origin --delete v1.0
To https://github.com/jakajah/belajargit.git
 - [deleted]          v1.0
[11:43:25] [cost 2.724s] git push origin --delete v1.0
```

Daftar pustaka

Sertifikat ini adalah dokumen resmi dan valid dirilis oleh CODEPOLITAN
untuk member atas nama JAKA MAULANA IBRAHIM

