

LAPORAN PRAKTIKUM
REKAYASA PERANGKAT LUNAK

Judul : Git Lanjut Bagian 3 dan Git Lanjut Bagian 4



Disusun Oleh:

Alfath Roziq Widhayaka – L0122012

PROGRAM STUDI INFORMATIKA
FAKULTAS INFORMASI DAN SAINS DATA
UNIVERSITAS SEBELAS MARET

2023

A. Git Lanjut Bagian 3

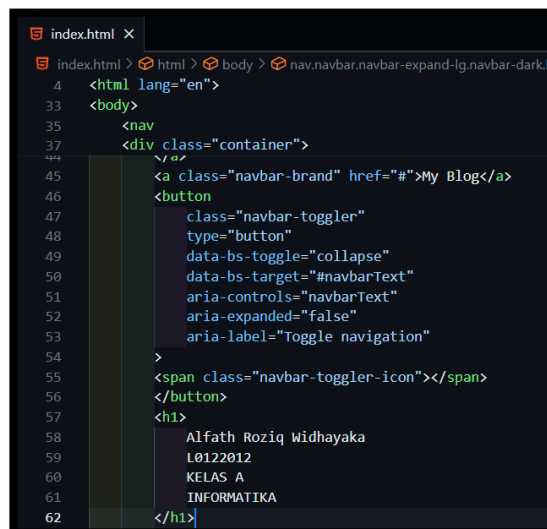
1. Perintah Untuk Membatalkan Revisi

Pada praktikum sebelumnya, sudah belajar cara melihat perbedaan di setiap revisi. Sekarang akan belajar, cara membatalkan sebuah revisi. Terkadang pada perubahan yang dilakukan terjadi kesalahan dan ingin mengembalikannya seperti keadaan sebelumnya. Maka perlu menyuruh git untuk mengembalikannya. Ada beberapa perintah yang digunakan diantaranya : **git checkout**, **git reset**, dan **git revert**.

a. Membatalkan perubahan

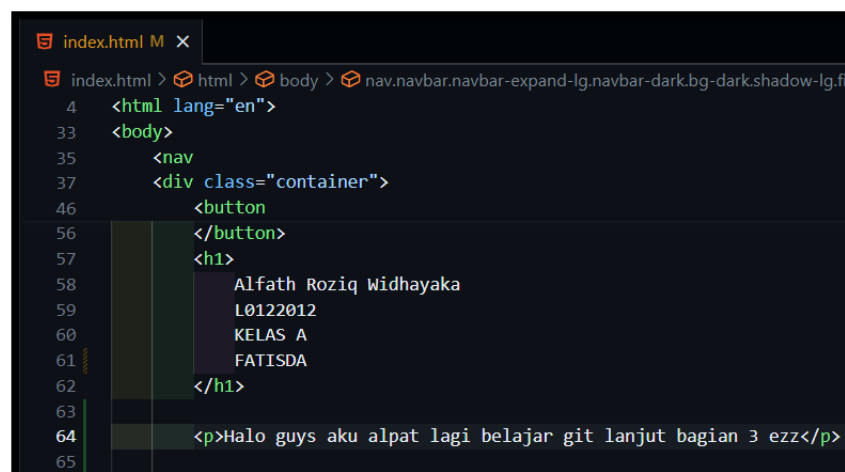
Jika revisi belum staged ataupun committed, bisa mengembalikannya menggunakan perintah **git checkout nama_file.html**. Contoh: Misalkan saya akan merubah isi dari file index.html pada repositori proyek-01.

Sebelum diubah :



```
index.html X
index.html > html > body > nav.navbar.navbar-expand-lg.navbar-dark.l
4  <html lang="en">
33 <body>
35   <nav
37     <div class="container">
44     </div>
45     <a class="navbar-brand" href="#">My Blog</a>
46     <button
47       class="navbar-toggler"
48       type="button"
49       data-bs-toggle="collapse"
50       data-bs-target="#navbarText"
51       aria-controls="navbarText"
52       aria-expanded="false"
53       aria-label="Toggle navigation"
54     >
55     <span class="navbar-toggler-icon"></span>
56   </button>
57   <h1>
58     Alfath Roziq Widhayaka
59     L0122012
60     KELAS A
61     INFORMATIKA
62   </h1>
```

Setelah diubah :



```
index.html M X
index.html > html > body > nav.navbar.navbar-expand-lg.navbar-dark.bg-dark.shadow-lg.fi
4  <html lang="en">
33 <body>
35   <nav
37     <div class="container">
46     <button
56   </button>
57   <h1>
58     Alfath Roziq Widhayaka
59     L0122012
60     KELAS A
61     FATISDA
62   </h1>
63
64   <p>Halo guys aku alpat lagi belajar git lanjut bagian 3 ezz</p>
65
```

Hasil git diff :

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git diff
diff --git a/index.html b/index.html
index be3522e..31bfb3 100644
--- a/index.html
+++ b/index.html
@@ -58,8 +58,11 @@
     Alfath Roziq Widhayaka
     L0122012
     KELAS A
-    INFORMATIKA
+    FATHISDA UNS
+
+</h1>
+
+<p>Halo guys aku alpat lagi belajar git lanjut bagian 3 ezz</p>
+
+<div class="collapse navbar-collapse text-right" id="navbarText">
+  <ul class="navbar-nav nav-underline mx-auto">
+    <li class="nav-item mx-2">
```

Berdasarkan perintah **git diff** di atas dapat dilihat perbedaan isi file index.html. Tulisan berwarna merah diganti dengan tulisan berwarna hijau dalam file index.html. Sekarang akan membatalkan perubahan tersebut. Karena belum melakukan stage dan commit, maka bisa menggunakan perintah **git checkout index.html** :

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git checkout index.html
Updated 1 path from the index
```

Perubahan yang baru saja lakukan akan dibatalkan. Untuk mengeceknya, silahkan coba periksa file yang sudah dirubah tadi atau cek dengan perintah **git status**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git status
On branch main
nothing to commit, working tree clean
```

Hati-hati! Terkadang perintah ini sangat berbahaya, karena akan menghapus perubahan yang baru saja dilakukan. Bila sudah merubah banyak hal, maka itu akan sia-sia setelah menjalankan perintah ini

b. Membatalkan perubahan file yang sudah dalam kondisi staged

Kondisi staged merupakan kondisi file yang sudah di add (**git add**), namun belum disimpan (**git commit**) ke dalam Git. Sebagai contoh, lakukan perubahan lagi di file index.html seperti pada contoh sebelumnya.

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Prakti
kum/Praktikum 10 dan 11/proyek-01 (main)
$ git diff
diff --git a/index.html b/index.html
index be3522e..05682b3 100644
--- a/index.html
+++ b/index.html
@@ -58,8 +58,11 @@
     Alfath Roziq widhayaka
     L0122012
     KELAS A
-    INFORMATIKA
+    FATISDA
+
+</h1>
+
+<p>Halo guys belajar lagi ini!!</p>
+
+<div class="collapse navbar-collapse text-right" id="navbarText">
+  <ul class="navbar-nav nav-underline mx-auto">
+    <li class="nav-item mx-2">
```

Setelah itu, ubah kondisi file menjadi staged dengan perintah **git add index.html**

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Prakti
kum/Praktikum 10 dan 11/proyek-01 (main)
$ git add index.html

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Prakti
kum/Praktikum 10 dan 11/proyek-01 (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
```

Nah, file index.html sudah masuk ke dalam kondisi staged. Untuk mengubahnya menjadi kondisi **modified**, bisa menggunakan perintah **git reset index.html**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Prakti
kum/Praktikum 10 dan 11/proyek-01 (main)
$ git reset index.html
Unstaged changes after reset:
M       index.html
```

Setelah dilakukan perintah **git reset** maka file index.html yang sudah berada di kondisi staged kini berubah lagi dan kembali ke dalam kondisi modified yang dilambangkan dengan huruf M. Perintah **git reset** sukses dilakukan dengan diikuti output berikut.

Unstaged changes after reset:

M index.html

Cek statusnya lagi :

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
```

Sekarang file index.html sudah dalam kondisi modified, bisa membatalkan perubahannya dengan perintah **git checkout** seperti contoh sebelumnya.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git checkout index.html
Updated 1 path from the index
```

Maka perubahan yang lakukan akan dibatalkan. Perintah **git checkout** digunakan untuk membatalkan revisi atau perubahan pada file yang belum masuk ke dalam tahap staging. Dengan dijalankan perintah **git checkout index.html** maka perubahan yang sebelumnya dilakukan akan dibatalkan dan isi file akan sama seperti semula.

c. Membatalkan perubahan file yang sudah dalam kondisi Committed

Sekarang bagaimana kalau filenya sudah dalam kondisi committed dan ingin mengembalikannya? Untuk melakukan ini, harus mengetahui nomer commit, kemudian mengembalikan perubahannya seperti pada nomer commit tersebut.

Misalkan, ubah kembali file **index.html**

```
MINGW64/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git diff
diff --git a/index.html b/index.html
index be3522e..d0e0d10 100644
--- a/index.html
+++ b/index.html
@@ -56,9 +56,9 @@
</button>
<h1>
    Alfath Roziq Widhayaka
    L0122012
    KELAS A
    INFORMATIKA
+   MANUSIA
+   KEREN
+   KECE PARAH
</h1>
<div class="collapse navbar-collapse text-right" id="navbarText">
  <ul class="navbar-nav nav-underline mx-auto">

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ !
```

Kemudian melakukan commit.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git add index.html

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git commit -m "Belajar git 3 lagi bre"
[main 10aab09] Belajar git 3 lagi bre
1 file changed, 3 insertions(+), 3 deletions(-)
```

Untuk melakukan commit, terlebih dahulu dilakukan perintah **git add** untuk mengubah kondisi file modified menjadi staged. Setelah berada dalam kondisi staged, maka lakukan commit untuk mengubah kondisi file menuju kondisi committed.

Sekarang akan melihat nomer commit dengan perintah **git log**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git log
commit 10aab0990ce73ac2b0d9182de43db30ab46dd62c (HEAD -> main)
Author: alfathroziqq <alfathroziqq94@gmail.com>
Date: Thu Jun 6 11:55:11 2024 +0700

    Belajar git 3 lagi bre

commit 5279f7ed7d0efcd6e068862b425f63339c745a26 (origin/main)
Author: alfathroziqq <alfathroziqq94@gmail.com>
Date: Thu Jun 6 11:29:01 2024 +0700

    commit pertama
```

Commit yang baru saja dilakukan

Ccommit yang sebelumnya, kita ingin kembali kesini

akan mengembalikan kondisi file `index.html`, seperti pada commit sebelumnya.

Maka bisa menggunakan perintah **git checkout <id>**.

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git checkout 5279f7ed7d0efcd6e068862b425f63339c745a26
Note: switching to '5279f7ed7d0efcd6e068862b425f63339c745a26'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 5279f7e commit pertama
```

Seperti mesin waktu, sudah mengembalikan keadaan file **index.html** seperti keadaan saat commit tersebut. Namun, saat ini kondisi **index.html** dalam keadaan staged. bisa kembalikan ke dalam kondisi modified dengan perintah **git reset**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 ((5279f7e...))
$ git reset index.html

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 ((5279f7e...))
$ git status
HEAD detached at 5279f7e
nothing to commit, working tree clean
```

Pada contoh tersebut, sudah berhasil mengembalikan file **index.html** ke dalam keadaan seperti commit sebelumnya.

Apabila ingin mengembalikan seluruh file dalam commit, cukup melakukan checkout ke nomer commit saja, tanpa diikuti nama file. Contoh:

```
git checkout ac6d798f98bac5fad693ef8159f957c5b0805c23
```

Catatan: Perintah ini akan mengembalikan semua file dalam kondisi pada nomer commit yang diberikan, namun bersifat temporer.

d. Kembali ke 3 commit sebelumnya

Untuk kembali ke 3 commit sebelumnya, bisa menggunakan perintah berikut.

```
git checkout HEAD~3 index.html
```

e. Membatalkan semua perubahan yang telah ada

Jika ingin mengembalikan semua file ke suatu commit, bisa melakukannya dengan perintah:

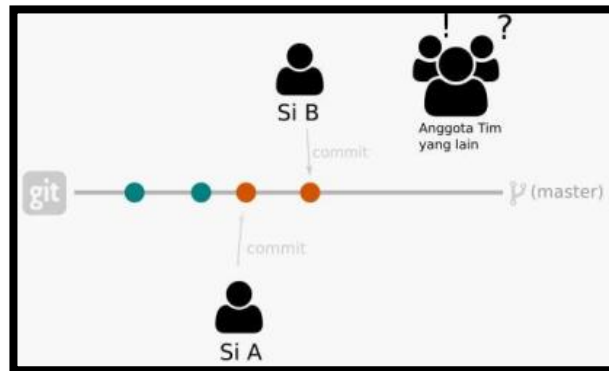
```
git revert -n <nomer commit>
```

Contoh :

```
git revert -n 2400ba0e258bd6a144caa273012b130d6baa5e42
```

2. Menggunakan Percabangan Untuk Mencegah Konflik

Bayangkan anda sedang bekerja dengan tim pada suatu repositori Git. Repositori ini dikerjakan secara bersama-sama. Kadang akan terjadi konflik, karena kode yang tulis berbeda dengan yang lain. Misalnya, Si A menulis kode untuk fitur X dengan algoritma yang ia ketahui. Sedangkan si B menulis dengan algoritma yang berbeda. Lalu mereka melakukan commit, dan kode sumber jadi berantakan. Anggota tim yang lain menjadi pusing.



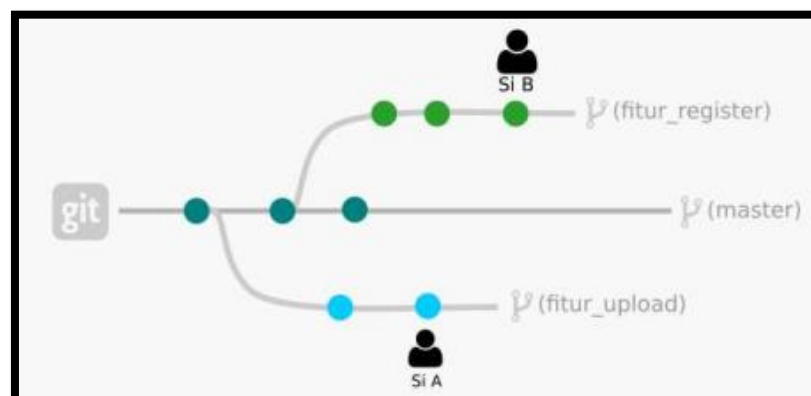
Agar tidak terjadi hal yang seperti ini, harus membuat cabang (branch) tersendiri. Misalnya, si A akan mengerjakan fitur X, maka dia harus membuat cabang sendiri. Si A akan bebas melakukan apapun di cabangnya tanpa mengganggu cabang utama (main).

a. Cara membuat cabang baru

Perintah untuk membuat cabang adalah **git branch**, kemudian diikuti dengan nama cabangnya. Contoh:

```
git branch fitur_register
```

Maka Git akan membuat cabang bernama **fitur_register**.



Sekarang setiap orang memiliki cabangnya masing-masing. Mereka bebas bereksperimen. Untuk melihat cabang apa saja yang ada di repositori, gunakan perintah **git branch**. Contoh:

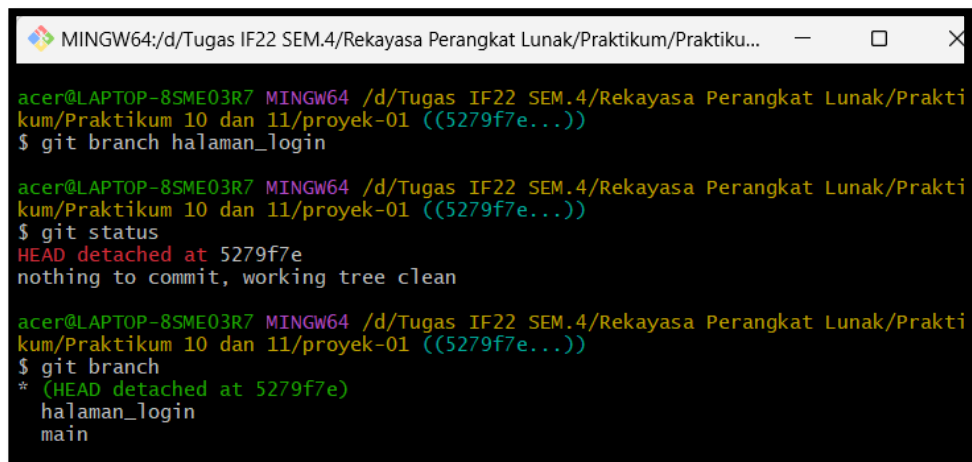
```
$ git branch
  halaman_login
* master
```

Tanda bintang (*) artinya cabang yang sedang aktif atau sedang berada di sana.

b. Latihan

Untuk memantapkan pemahaman tentang percabangan Git, mari coba praktik. Pada repositori, buatlah sebuah cabang baru.

Dengan menjalankan perintah **git branch halaman_login** maka git akan membuat branch baru dengan nama **halaman_login**.

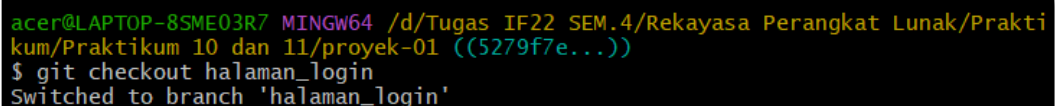


```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 ((5279f7e...))
$ git branch halaman_login

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 ((5279f7e...))
$ git status
HEAD detached at 5279f7e
nothing to commit, working tree clean

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 ((5279f7e...))
$ git branch
* (HEAD detached at 5279f7e)
  halaman_login
  main
```

Setelah itu, pindah ke cabang yang baru saja buat dengan perintah **git checkout halaman_login**.



```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 ((5279f7e...))
$ git checkout halaman_login
Switched to branch 'halaman_login'
```

Perintah **git checkout halaman_login** akan menyebabkan git pindah ke cabang **halaman_login**. Git melakukan checkout dari cabang main dan pindah ke cabang **halaman_login**. Lalu tambahkan file **login.html**, isinya bebas.

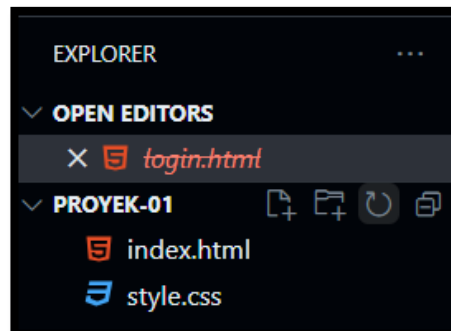
```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git add login.html

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git commit -m "login.html baru"
[halaman_login 5612024] login.html baru
1 file changed, 260 insertions(+)
create mode 100644 login.html
```

Untuk melakukan commit, terlebih dahulu dilakukan perintah **git add** untuk mengubah kondisi file modified menjadi staged. Setelah berada dalam kondisi staged, maka lakukan commit untuk mengubah kondisi file menuju kondisi committed. Revisi pada cabang **halaman_login** sudah disimpan. Sekarang coba kembali ke cabang **main**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git checkout main
Switched to branch 'main'
```

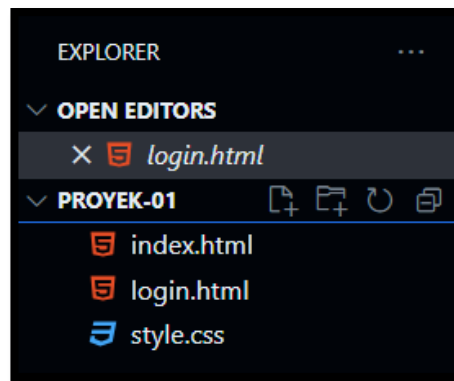
Perintah **git checkout main** akan menyebabkan git pindah ke cabang main. Git melakukan checkout dari cabang **halaman_login** dan pindah ke cabang **main**. Apakah anda menemukan file **login.html**? Pasti tidak (file tercoret)



Sekarang kembali lagi ke cabang **halaman_login**.

```
Switched to branch 'halaman_login',
$ git checkout halaman_login
Kembali ke cabang halaman_login
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git checkout halaman_login
```

Perintah **git checkout halaman_login** akan menyebabkan git pindah ke cabang **halaman_login**. Git melakukan checkout dari cabang main dan pindah ke cabang **halaman_login**. Cek lagi, apakah sekarang file login.html sudah ada?



Ternyata ada. Dapat diambil kesimpulan, kalau perubahan pada cabang **halaman_login** tidak akan berpengaruh di cabang **main**.

c. Menggabungkan cabang

Anggaplah sudah selesai membuat fitur login di cabang **halaman_login**. Sekarang ingin Menggabungkannya dengan cabang **main** (utama). Pertama, harus pindah dulu ke cabang **main**.

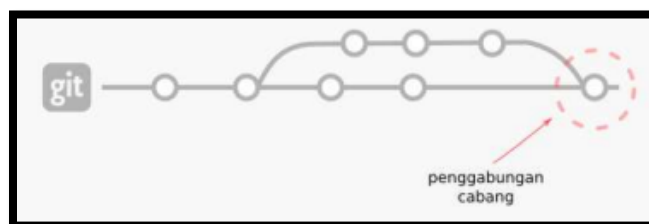
```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git checkout main
Switched to branch 'main'
```

Setelah itu, barulah bisa menggabungkan dengan perintah **git merge**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git merge halaman_login
Already up to date.
```

Perintah **git merge halaman_login** akan menggabungkan cabang **main** dan cabang **halaman_login** sehingga file **login.html** juga akan ada di cabang **main**.

Sekarang lihat, file **login.html** sudah ada di cabang **main**.



Hati-hati! kadang sering terjadi bentrok ketika menggabungkan cabang. saat menjalankan perintah **'git merge'** lalu ada kode yang bentrok.

3. Mengatasi Bentrok

Bentrok biasanya terjadi jika ada dua orang yang mengedit file yang sama. Kenapa bisa begitu, padahal mereka sudah punya cabang masing-masing?

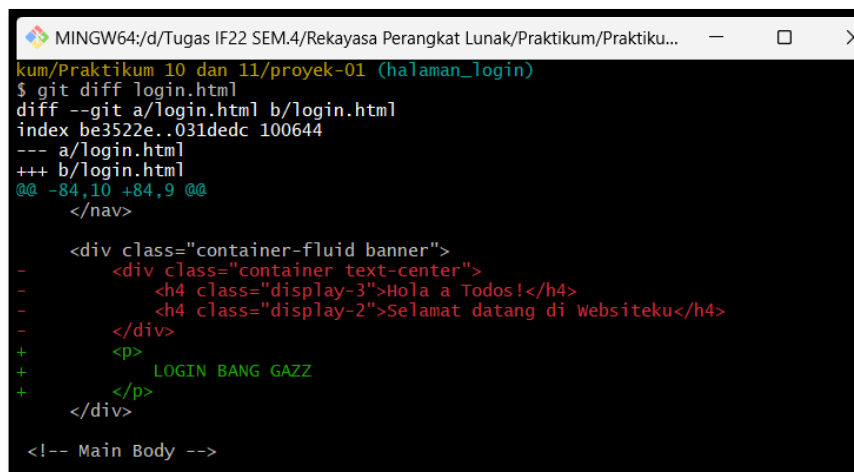
Bisa jadi, di cabang yang mereka kerjakan ada file yang sama dengan cabang lain. Kemudian, saat digabungkan terjadi bentrok. Mengatasi bentrok adalah tugas dari pemilik atau pengelola repostori. Dia harus bertindak adil, kode mana yang harus diambil. Biasanya akan ada proses diskusi dulu dalam mengambil keputusan.

Sekarang akan coba membuat bentrokan.

Pindah dulu ke branch **halaman_login**...

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git checkout halaman_login
Switched to branch 'halaman_login'
```

Setelah itu, edit file login.html atau index.html, karena kedua file tersebut ada di kedua cabang yang akan gabungkan.



```
MINGW64;d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
kum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git diff login.html
diff --git a/login.html b/login.html
index be3522e..031dedc 100644
--- a/login.html
+++ b/login.html
@@ -84,10 +84,9 @@
     </nav>

     <div class="container-fluid banner">
-       <div class="container text-center">
-         <h4 class="display-3">Hola a Todos!</h4>
-         <h4 class="display-2">Selamat datang di Websiteku</h4>
-       </div>
+       <p>
+         LOGIN BANG GAZZ
+       </p>
     </div>

<!-- Main Body -->
```

Setelah itu, lakukan commit lagi:

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git add login.html

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git commit -m "ubah isi login ni"
[halaman_login 531c7a3] ubah isi login ni
1 file changed, 3 insertions(+), 4 deletions(-)
```

Untuk melakukan commit, terlebih dahulu dilakukan perintah **git add** untuk mengubah kondisi file modified menjadi staged. Setelah berada dalam kondisi staged, maka lakukan commit untuk mengubah kondisi file menuju kondisi committed.

Selanjutnya pindah ke cabang main dan lakukan perubahan juga di cabang ini. Ubah file yang sama seperti di cabang **halaman_login**.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (halaman_login)
$ git checkout main
Switched to branch 'main'
```

Setelah itu, lakukan commit di cabang **main**

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git commit -m "ubah lagi ni isi login di main"
[main 3340886] ubah lagi ni isi login di main
1 file changed, 3 insertions(+), 4 deletions(-)
```

Untuk melakukan commit, terlebih dahulu dilakukan perintah git add untuk mengubah kondisi file modified menjadi staged. Setelah berada dalam kondisi staged, maka lakukan commit untuk mengubah kondisi file menuju kondisi committed.

Terakhir, coba gabungkan cabang **halaman_login** dengan cabang **main**, maka akan terjadi bentrok.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git merge halaman_login
Auto-merging login.html
CONFLICT (content): Merge conflict in login.html
Automatic merge failed; fix conflicts and then commit the result.
```

4. Menghapus Cabang

Cabang yang sudah mati atau tidak ada pengembangan lagi, sebaiknya dihapus. Agar repositori bersih dan rapi. Cara menghapus cabang, gunakan perintah git branch dengan argumen -d dan diikuti dengan nama cabangnya.

Contoh :

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main|MERGING)
$ git branch -D halaman_login
Deleted branch halaman_login (was 531c7a3).
```

Perintah **git brach -D halaman_login** akan menghapus branch **halaman_login** dari repository.

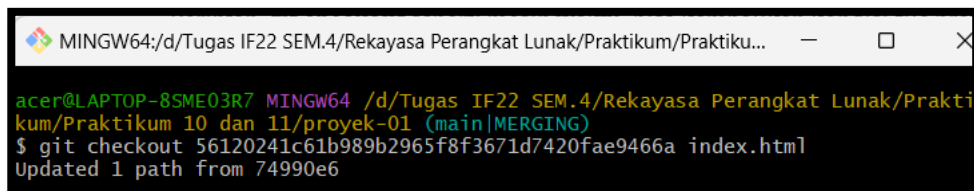
B. Git Lanjut Bagian 4

1. Perbedaan Git Checkout, Git Reset, dan Git Revert

Pada praktikum git lanjut bagian 3, sudah membahas cara membatalkan revisi menggunakan perintah **git checkout**, **git reset**, dan **git revert**. Sekilas, tiga perintah tersebut melakukan hal yang sama. Akan tetapi ada perbedaannya. Berikut perbedaannya.

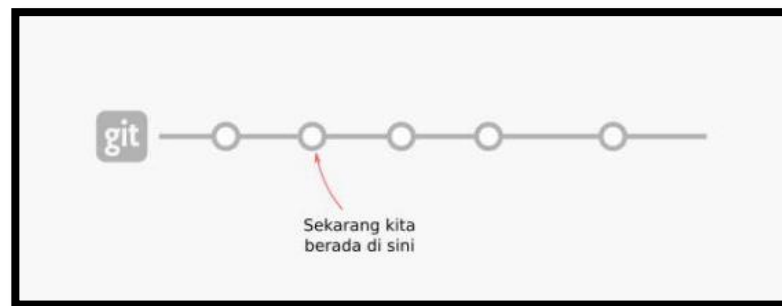
a. Git Checkout

Perintah **git checkout** seperti mesin waktu, bisa mengembalikan kondisi file proyek seperti waktu yang dituju.



```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main|MERGING)
$ git checkout 56120241c61b989b2965f8f3671d7420fae9466a index.html
Updated 1 path from 74990e6
```

Maka semua file akan dikembalikan seperti keadaan pada nomer commit tersebut.

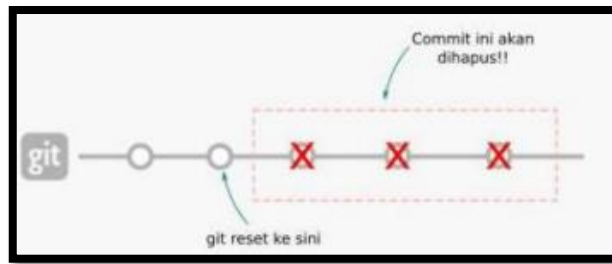


Akan tetapi, ini bersifat temporer (sementara). Pengembalian ini tidak disimpan dalam database Git. Bisa sebut perintah **git checkout** sebagai perintah untuk mengecek kondisi file di setiap commit.

b. Git Reset

Perintah **git reset** sering disebut sebagai perintah berbahaya yang dapat menghancurkan catatan histori perubahan.

Hati-hati! Perintah ini membuat tidak bisa kembali lagi ke masa depan. Mau tidak mau, harus menulis ulang histori.



Perintah ini memiliki tiga argumen atau opsi utama, yaitu **--soft**, **--mixed**, dan **--hard**.

- **--soft** akan mengembalikan dengan kondisi file dalam keadaan *staged*
- **--mixed** akan mengembalikan dengan kondisi file dalam keadaan *modified*
- **--hard** akan mengembalikan dengan kondisi file dalam keadaan *committed*

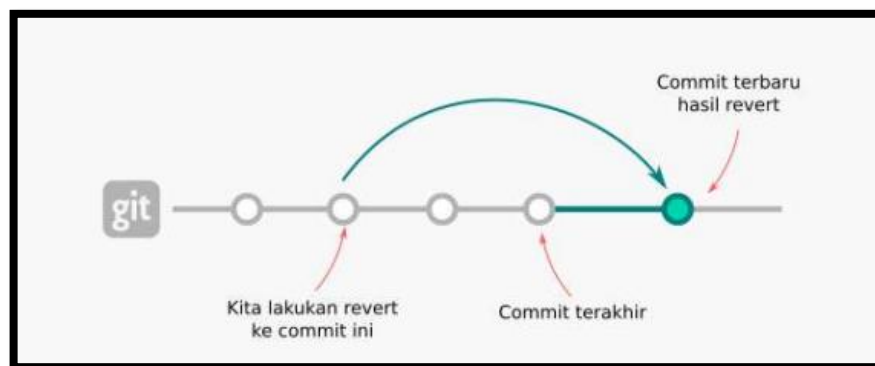
Contoh penggunaan:

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Prakti
$ git reset --hard 56120241c61b989b2965f8f3671d7420fae9466a
HEAD is now at 5612024 login.html baru
```

Hati-hati! Jangan lakukan git reset pada repositori yang sudah di bagikan ke publik, karena dapat merusaknya.

c. Git Revert

Revert artinya mengembalikan. Perintah ini lebih aman daripada **git reset**, karena tidak akan menghapus catatan sejarah revisi. **Revert** akan akan mengambil kondisi file yang ada di masa lalu, kemudian menggabungkannya dengan commit terakhir.



coba pada repository masingmasing. Sebelumnya, berikut ini adalah kondisi repositori yang dijadikan bahan percobaan.

Kali ini coba revert ke commit pertama pada log yang ada pada git log. Tuliskan **git revert <id>**.

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01
Revert "commit pertama"

This reverts commit 5279f7ed7d0efcd6e068862b425f63339c745a26.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Changes to be committed:
#   deleted:   index.html
#   deleted:   style.css
#
```

Gambar diatas merupakan hasil dari revert menjadi ke commit pertama pada folder proyek-01.

Ketik **git log** untuk melihat perubahan yang terjadi pada log.

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ git log
commit 29c510678136025a430be6e9b7d2a48ba575629c (HEAD -> main)
Author: alfathroziqq <alfathroziqq94@gmail.com>
Date: Thu Jun 6 18:37:19 2024 +0700

    Revert "commit pertama"

    This reverts commit 5279f7ed7d0efcd6e068862b425f63339c745a26.

commit 56120241c61b989b2965f8f3671d7420fae9466a
Author: alfathroziqq <alfathroziqq94@gmail.com>
Date: Thu Jun 6 13:15:35 2024 +0700

    login.html baru

commit 5279f7ed7d0efcd6e068862b425f63339c745a26
Author: alfathroziqq <alfathroziqq94@gmail.com>
Date: Thu Jun 6 11:29:01 2024 +0700

    commit pertama

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/proyek-01 (main)
$ |
```

Kesimpulan:

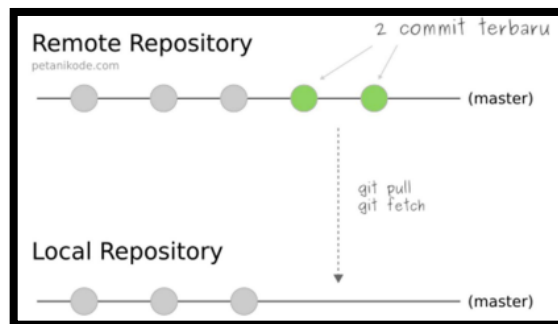
- Perintah **git checkout** mengembalikan file dalam kondisi sebelumnya, tapi bersifat sementara.
- Perintah **git reset**, akan mengembalikan file ke kondisi sebelumnya, kemudian menghapus catatan sejarah commit berikutnya.
- Perintah **git revert** mengembalikan file dengan tidak menghapus sejarah commit.

2. Studi Kasus : Kapan menggunakan git pull dan git fetch

Perintah **git pull** dan **git fetch** adalah dua perintah untuk mengambil commit terbaru dari remote repository. Apa perbedaan dari kedua perintah tersebut? kapan waktu yang tepat menggunakan git pull dan git fetch?

a. Apa perbedaan Git Pull dengan Git Fetch

Sebelum menjawab pertanyaan “kapan waktu yang tepat menggunakan **git pull** dan **git fetch**?” bahas dulu perbedaannya. Perintah **git pull** dan **git fetch** fungsinya sama. Yaitu mengambil commit terbaru dari *remote repository*



Perbedaannya perintah **git pull** dan **git fetch** :

- Perintah **git pull** akan mengambil commit terbaru lalu otomatis menggabungkan (merge) dengan branch yang aktif.
- Sedangkan **git fetch** akan mengambil commit saja. Perintah **git fetch** tidak akan langsung melakukan *merge*.

b. Cara menggunakan git pull dan git fetch

Perintah git pull dan git fetch dapat digunakan seperti ini:

```
git pull origin master  
git fetch origin master
```

Artinya akan mengambil commit dari branch main pada repository remote. origin adalah nama remote-nya. Untuk penjelasan yang lebih detail, bisa membaca di dokumentasi dengan mengetik perintah **git pull --help** atau **git fetch --help**.

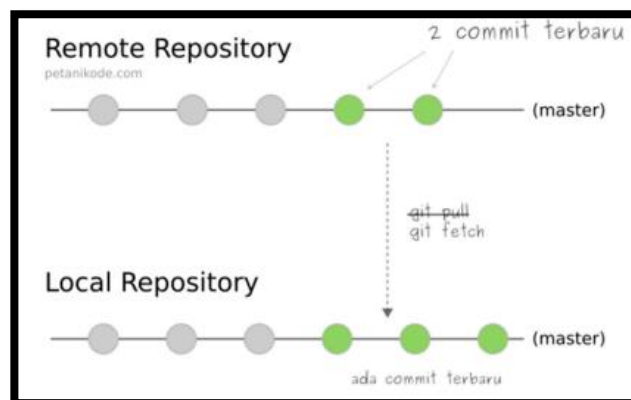
```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Prakti
kum/Praktikum 10 dan 11/cobaBuatRPL (main)
$ git pull -help
usage: git pull [<options>] [<repository> [<refspec>...]]

-v, --[no-]verbose      be more verbose
-q, --[no-]quiet        be more quiet
--[no-]progress        force progress reporting
--[no-]recurse-submodules[=<on-demand>] control for recursive fetching of submodules

Options related to merging
-r, --[no-]rebase[=(false|true|merges|interactive)]
    incorporate changes by rebasing rather than merging
-n                                     do not show a diffstat at the end of the merge
--[no-]stat                          show a diffstat at the end of the merge
--[no-]log[=<n>]                      add (at most <n>) entries from shortlog to merge commi
t message
--[no-]signoff[=...]                add a Signed-off-by trailer
--[no-]squash                       create a single commit instead of doing a merge
--[no-]commit                       perform a commit if the merge succeeds (default)
--[no-]edit                         edit message before committing
--[no-]cleanup <mode>              how to strip spaces and #comments from message
```

c. Kapan waktu yang tepat menggunakan git pull dan git fetch

Apabila sudah melakukan commit di repository lokal, maka yang digunakan adalah git fetch.



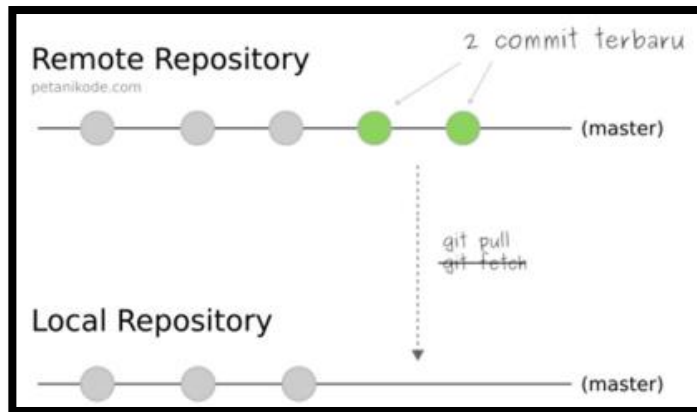
Menggunakan Git Fetch

Kenapa menggunakan **git fetch**?

Karena untuk mencegah terjadinya bentrok.

Perintah **git fetch** akan mengambil commit terbaru dan menyimpannya di branch origin/main.

Sedangkan apabila tidak pernah melakukan apa-apa di lokal repository, bisa menggunakan **git pull**.

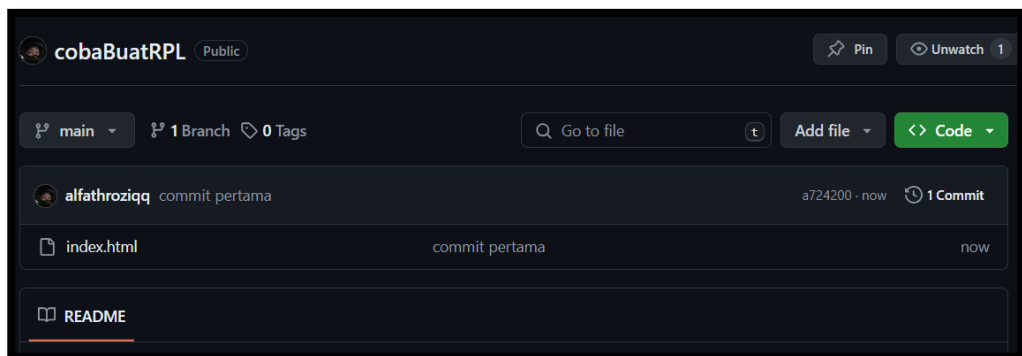


Menggunakan **Git Fetch**

Perintah **git pull** akan mengambil commit terbaru ke branch origin/main dan langsung menggabungkannya dengan branch main (lokal).

d. Studi kasus penggunaan git pull dan git fetch

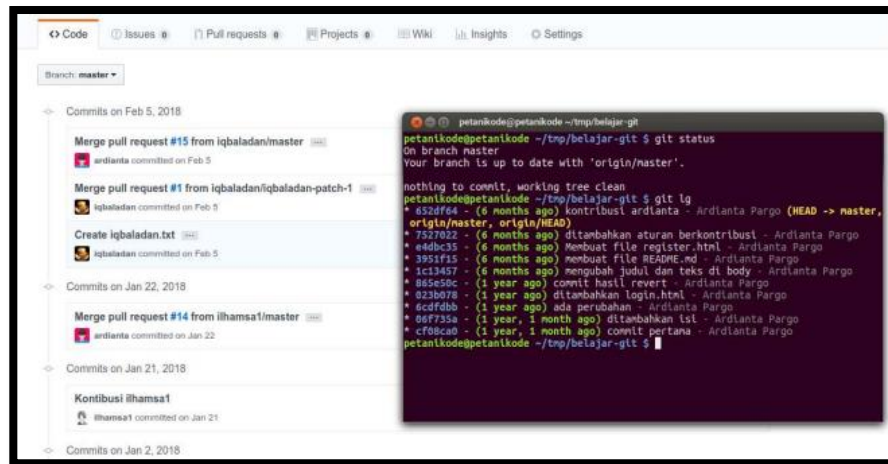
Untuk mencoba penggunaan **git pull** dan **git fetch** akan menggunakan repository **cobaBuatRPL** yang sudah saya dibuat di Github.



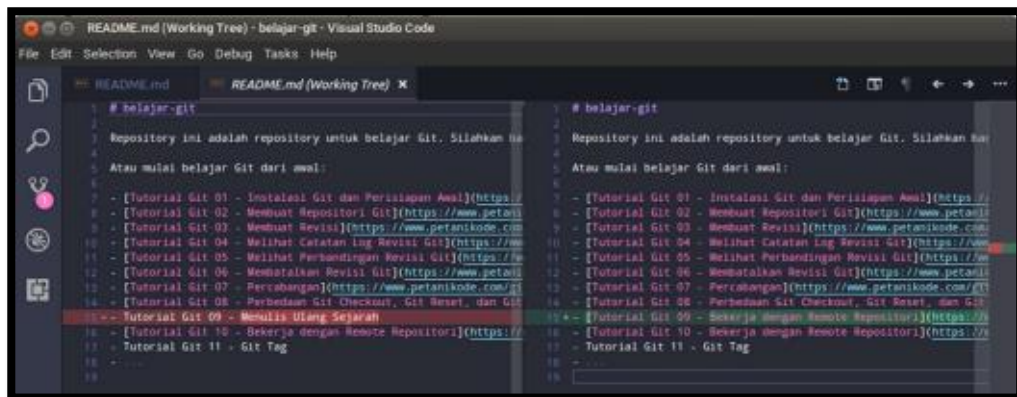
Repositori di Github

Saat ini sudah terdapat 14 kontributor di dalam repositori tersebut. Sedangkan pada repositori lokal, saya belum melakukan apa pun.

Ini perbedaan commit repo lokal dengan remote:



Perbedaan commit di repo remote dengan lokal Pada repository remote sudah terjadi 48 commit, sedangkan pada lokal hanya 10 saja. Sekarang saya akan melakukan sebuah commit di repository lokal. Saya akan mengubah file README.md menjadi seperti ini:



Jadi sekarang di repository lokal ada 11 commit.

Lalu perintah yang mana yang harus gunakan untuk mengambil 48 commit di Github? Tentu saja akan menggunakan git fetch, karena saya sudah melakukan perubahan di repo lokal.

```
git fetch origin master
```

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
a7242005b570d66c02035dc19b35ecc1034c99cc refs/heads/main

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL (master)
$ git remote -v
origin https://github.com/alfathroziqq/cobaBuatRPL.git (fetch)
origin https://github.com/alfathroziqq/cobaBuatRPL.git (push)

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL (master)
$ git status
On branch master
nothing to commit, working tree clean

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL (master)
$ git fetch origin main
From https://github.com/alfathroziqq/cobaBuatRPL
* branch      main      -> FETCH_HEAD
* [new branch] main      -> origin/main
```

bisa masuk ke branch origin/main dengan perintah git checkout:

```
git checkout origin/master
```

```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
kum/Praktikum 10 dan 11/cobaBuatRPL (master)
$ git checkout origin/main
Note: switching to 'origin/main'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at a724200 commit pertama

acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL ((a724200...))
```

Lalu bisa lihat commit apa saja yang ada di sana dengan git log:

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL ((ff01736...))
$ git log
commit ff0173638e4555c58595cdb4e221c1f32697cce5 (HEAD)
Author: alfathroziqq <alfathroziq94@gmail.com>
Date: Thu Jun 6 20:05:05 2024 +0700

    edit index nih

commit a7242005b570d66c02035dc19b35ecc1034c99cc (origin/main, master)
Author: alfathroziqq <alfathroziq94@gmail.com>
Date: Thu Jun 6 19:49:34 2024 +0700

    commit pertama
```

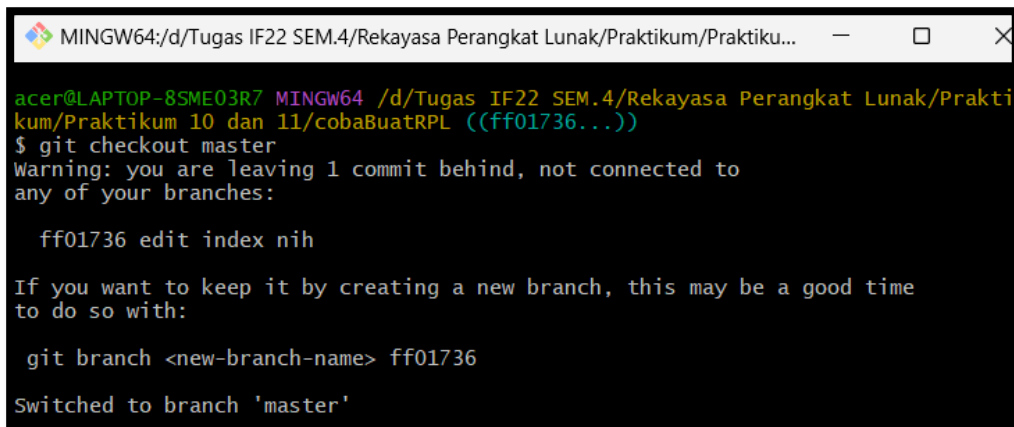
Ada dua branch dalam log tersebut:

- **master** adalah branch master di lokal.
- **origin/master** adalah branch master di remote yang sudah fetch.

Apabila sudah yakin, bisa menggabungkan (merge) dua branch tersebut. Caranya:

```
# pindah dulu ke branch master (lokal)
git checkout master
# merge branch
git merge master origin/master
```

Menggabungkan branch **master** dengan **origin/master**



```
MINGW64:/d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktiku...
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL ((ff01736...))
$ git checkout master
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

ff01736 edit index nih

If you want to keep it by creating a new branch, this may be a good time
to do so with:

git branch <new-branch-name> ff01736

Switched to branch 'master'
```

Kemudian lihat histori commit nya dengan git log

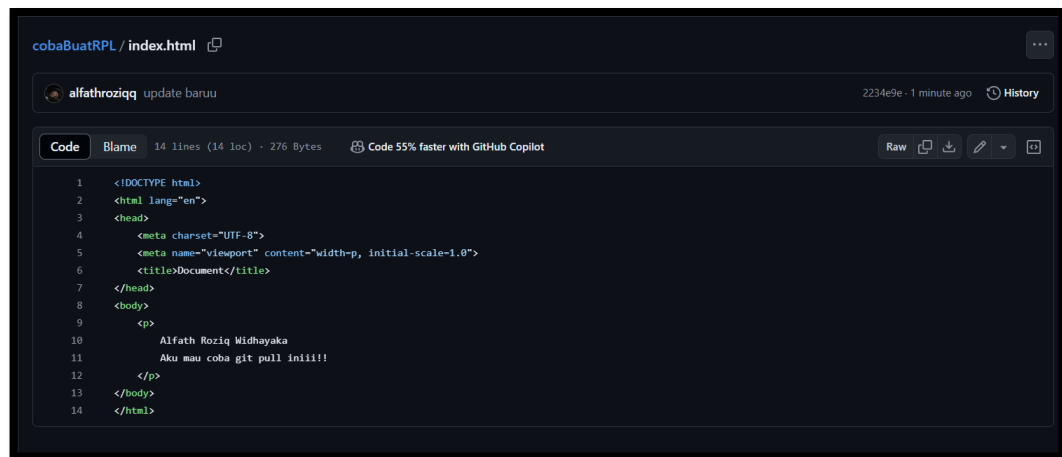
Lalu Bagaimana Penggunaan Git Pull?

Perintah git pull digunakan saat tidak pernah melakukan commit apapun di repo lokal.

Cara kerja git pull:

- Ambil semua commit dari repo remote
- Gabungkan branch main (lokal) dengan origin/main (remote)
- Selesai

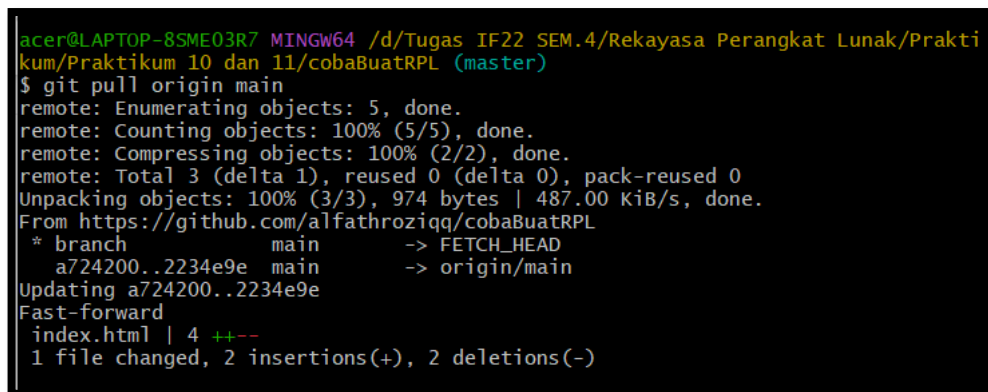
Untuk mencoba **git pull**, saya akan mengubah file **index.html** dari Github.

A screenshot of a GitHub web interface showing a file named 'index.html' in a repository called 'cobaBuatRPL'. The file is owned by 'alfathroziqq' and was updated 'baru' (new) '2234e9e' '1 minute ago'. The file content is an HTML document with a doctype, head section (including charset and viewport), and a body section containing a paragraph about 'Alfath Roziq Midhayaka' and a note about trying git pull. The interface includes tabs for 'Code', 'Blame', and 'History', along with a 'Raw' button and a 'Code 55% faster with GitHub Copilot' badge.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   <p>
10     Alfath Roziq Midhayaka
11     Aku mau coba git pull inii!!!
12   </p>
13 </body>
14 </html>
```

Artinya sekarang di repositori remote (Github) ada 1 commit baru. Bagaimana cara mengambilnya dengan git pull?

Caranya: Gunakan perintah **git pull origin main**.

A terminal window screenshot showing the execution of the 'git pull origin main' command. The output shows the process of enumerating, counting, and compressing objects from the remote repository, followed by a fast-forward update of the local 'main' branch. The final output indicates that 'index.html' was updated with 2 insertions and 2 deletions.

```
acer@LAPTOP-8SME03R7 MINGW64 /d/Tugas IF22 SEM.4/Rekayasa Perangkat Lunak/Praktikum/Praktikum 10 dan 11/cobaBuatRPL (master)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 974 bytes | 487.00 KiB/s, done.
From https://github.com/alfathroziqq/cobaBuatRPL
* branch            main       -> FETCH_HEAD
a724200..2234e9e    main       -> origin/main
Updating a724200..2234e9e
Fast-forward
 index.html | 4 ++--
1 file changed, 2 insertions(+), 2 deletions(-)
```

Maka semua commit terbaru dari branch origin/main di Github akan diambil ke lokal dan langsung digabungkan dengan branch main.

C. Kesimpulan

Pada praktikum **Git Lanjutan bagian 3**, mempelajari berbagai cara untuk membatalkan perubahan dan mengelola revisi di Git menggunakan perintah **git checkout**, **git reset**, dan **git revert**. Perintah ini memungkinkan pembatalan perubahan yang belum di-staged, mengembalikan file dari staged ke modified, serta mengembalikan file ke kondisi commit sebelumnya. Selain itu, bagian ini juga menekankan pentingnya penggunaan percabangan (branching) dalam Git untuk menghindari konflik dan memastikan kolaborasi yang lebih aman dan terstruktur dalam tim. Praktikum ini juga mencakup teknik penggabungan cabang dan cara mengatasi bentrok saat penggabungan.

Pada praktikum **Git Lanjutan bagian 4**, difokuskan pada perbedaan antara perintah **git checkout**, **git reset**, dan **git revert** serta penggunaannya yang tepat. Selain itu, praktikum ini membahas perbedaan antara **git pull** dan **git fetch** serta situasi yang tepat untuk menggunakan masing-masing perintah tersebut. **Git pull** digunakan untuk mengambil dan menggabungkan commit terbaru dari remote repository secara otomatis, sedangkan **git fetch** hanya mengambil commit tanpa langsung menggabungkannya, sehingga lebih aman untuk mencegah konflik ketika sudah ada perubahan di repository lokal.