

# Praktikum 3

## Git Lanjut Bagian III

### Perintah untuk Membatalkan Revisi

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

### Membatalkan Perubahan

Jika revisi kita belum *staged* ataupun *committed*, kita bisa mengembalikannya menggunakan perintah `git checkout nama_file.html`.

Contoh: Misalkan kita akan merubah isi dari file `index.html` pada repositori `project-01`. Sebelum diubah:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Belajar Git - Project 01</title>
  </head>
  <body>
    <p>Hello Dunia!, Saya sedang belajar Git</p>
  </body>
</html>
```

Setelah diubah:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Belajar Git - Project 01</title>
  </head>
  <body>
    <p>Hello Dunia!, Saya sudah belajar Git</p>
    <p>Belajar git ternyata cukup menyenangkan</p>
  </body>
</html>
```

Hasil `git diff`:

```
$ git diff
diff --git a/index.html b/index.html
index c5082e6..115efcb 100644
--- a/index.html
+++ b/index.html
@@ -5,6 +5,7 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Dunia!, Saya sedang belajar Git</p>
+     <p>Hello Dunia!, Saya sudah belajar Git</p>
+     <p>Belajar git ternyata cukup menyenangkan</p>
   </body>
 </html>
```

Sekarang kita akan membatalkan perubahan tersebut. Karena kita belum melakukan *stage* dan *commit*, maka kita bisa menggunakan perintah:

```
git checkout index.html
```

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

```
$ git status
On branch master
nothing to commit, working directory clean
```

Hati-hati! Terkadang perintah ini sangat berbahaya, karena akan menghapus perubahan yang baru saja dilakukan.

Bila kita sudah merubah banyak hal, maka itu akan sia-sia setelah menjalankan perintah ini.

### 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, kita lakukan perubahan lagi di file `index.html` seperti pada contoh sebelumnya.

```
$ git diff
diff --git a/index.html b/index.html
index c5082e6..c99aa5b 100644
--- a/index.html
+++ b/index.html
@@ -5,6 +5,7 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Dunia!, Saya sedang belajar Git</p>
+     <p>Hello Dunia!, Saya sudah belajar Git</p>
+     <p>Belajar git ternyata gampang-gampang susah</p>
   </body>
 </html>
```

Setelah itu, kita ubah kondisi file menjadi *staged* dengan perintah:

```
git add index.html
```

Cek statunya dulu:

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   index.html
```

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

```
git reset index.html
```

Cek statusnya lagi:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

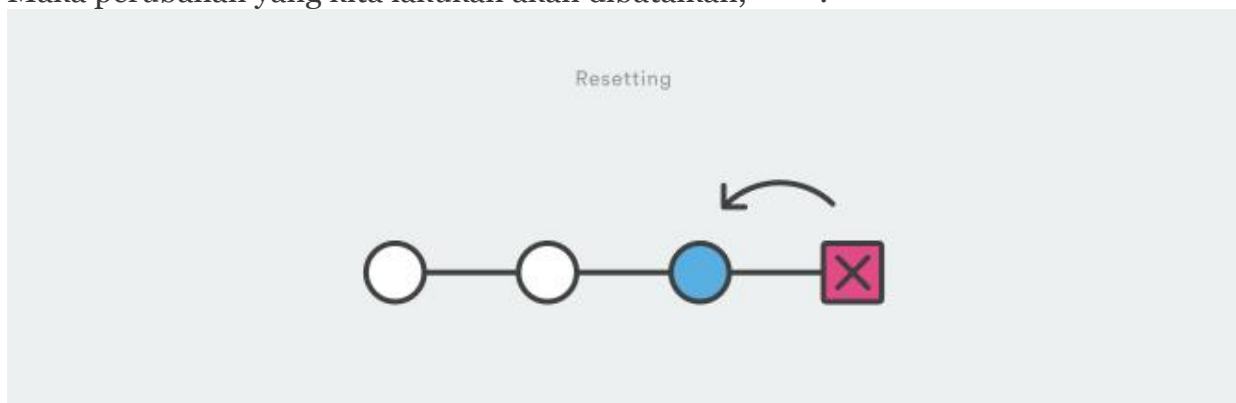
    modified:   index.html

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

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

```
git checkout index.html
```

Maka perubahan yang kita lakukan akan dibatalkan, .



## Membatalkan Perubahan File yang Sudah dalam Kondisi *Committed*

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

Misalkan, kita ubah kembali file `index.html`.

```
$ git diff
diff --git a/index.html b/index.html
index c5082e6..3c150a8 100644
--- a/index.html
+++ b/index.html
@@ -5,6 +5,7 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Dunia!, Saya sedang belajar Git</p>
+     <p>Hello Dunia!, Saya sudah belajar Git</p>
+     <p>Belajar Git Greget!</p>
   </body>
 </html>
```

Kemudian kita melakukan *commit*.

```
git add index.html
git commit -m "belajar git greget!"
```

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

```
petanikode@imajinasi ~/project-01
commit ac6d798f98bac5fad693ef8159f957c5b0805c23
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Tue Feb 21 15:46:07 2017 +0800

    belajar git greget!

commit b05f7d05c9298f2cd11b870369f3cf4b2350eca7
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Tue Feb 21 15:13:12 2017 +0800

    perubahan sudah dilakukan

commit 962e7d6c954f6478de9b890aef4fb4d1790c56e1
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Tue Feb 14 19:27:31 2017 +0800

    ditambahkan isi

commit 1a78e03c2ae7ef9999ab05296e4a025e3696f3b6
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Tue Feb 14 19:26:15 2017 +0800

    Revert "ditambahkan isi"
```

Kita akan mengembalikan kondisi file `index.html`, seperti pada *commit* sebelumnya. Maka kita bisa menggunakan perintah:

```
git checkout b05f7d05c9298f2cd11b870369f3cf4b2350eca7 index.html
```

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

```
git reset index.html
```

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

Apabila kita ingin mengembalikan seluruh file dalam *commit*, kita 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.

### Kembali ke 3 Commit sebelumnya

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

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

### Membatalkan Semua Perubahan yang ada

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

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

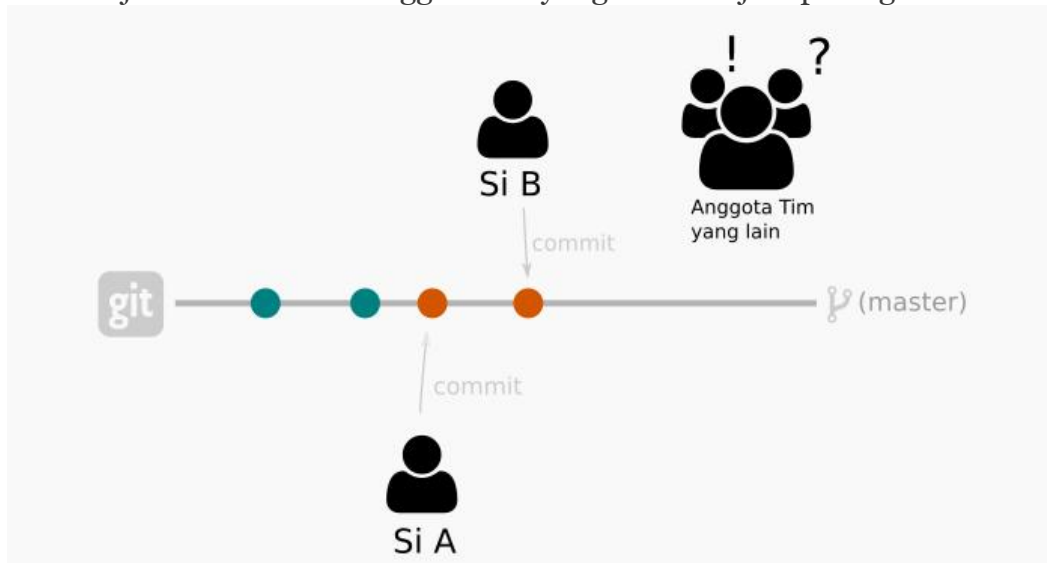
Contoh:

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

### 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 kita 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, kita 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 (*master*).

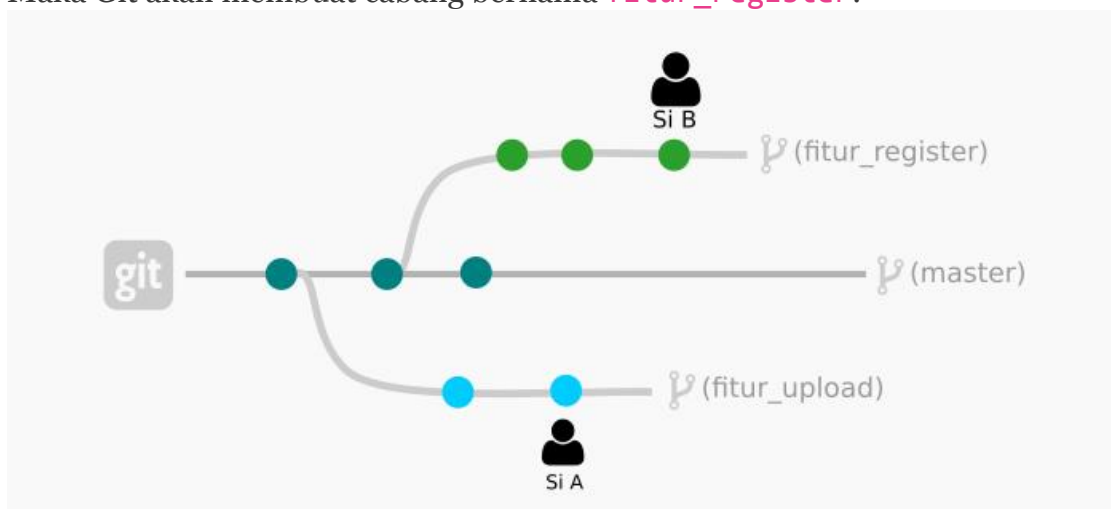
### Cara Membuat Cabang Baru

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 Kita sedang berada di sana.

## Latihan

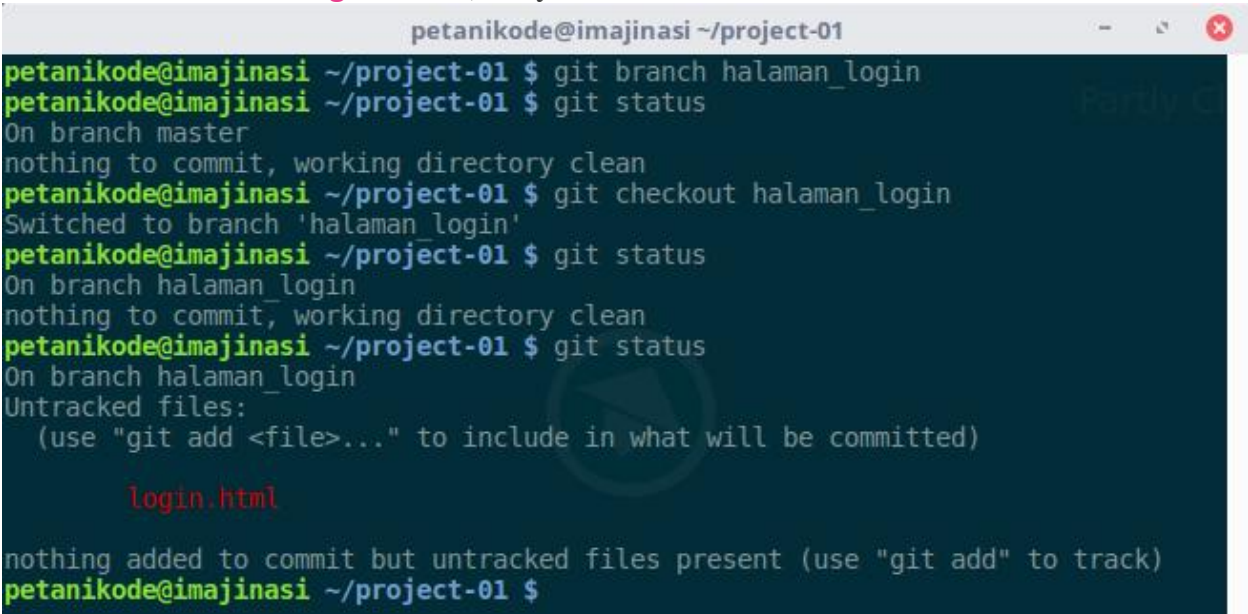
Untuk memantapkan pemahaman tentang percabangan Git, mari kita coba praktik. Pada repositori, buatlah sebuah cabang baru.

```
git branch halaman_login
```

Setelah itu, pindah ke cabang yang baru saja kita buat dengan perintah:

```
git checkout halaman_login
```

Lalu tambahkan file `login.html`, isinya terserah anda.



```
petanikode@imajinasi ~/project-01
petanikode@imajinasi ~/project-01 $ git branch halaman_login
petanikode@imajinasi ~/project-01 $ git status
On branch master
nothing to commit, working directory clean
petanikode@imajinasi ~/project-01 $ git checkout halaman_login
Switched to branch 'halaman_login'
petanikode@imajinasi ~/project-01 $ git status
On branch halaman_login
nothing to commit, working directory clean
petanikode@imajinasi ~/project-01 $ git status
On branch halaman_login
Untracked files:
  (use "git add <file>..." to include in what will be committed)

  login.html

nothing added to commit but untracked files present (use "git add" to track)
petanikode@imajinasi ~/project-01 $
```

Tips: Jangan lupa untuk menggunakan perintah `git status` untuk melihat status repositori.

Kita sudah menambahkan file `login.html`. Selanjutnya kita lakukan *commit*.

```
git add login.html
git commit -m "membuat file login.html"
```

Revisi kita pada cabang `halaman_login` sudah disimpan. Sekarang coba kembali ke cabang `master`.

```
git checkout master
```

Apakah anda menemukan file `login.html`? Pasti tidak!  
Sekarang kembali lagi ke cabang `halaman_login`.

```
git checkout halaman_login
```

Cek lagi, apakah sekarang file `login.html` sudah ada?

```
project-01/  
├─ index.html  
└─ login.html
```

Ternyata ada. Kita bisa mengambil kesimpulan, kalau perubahan pada cabang `halaman_login` tidak akan berpengaruh di cabang `master`.

## Menggabungkan Cabang

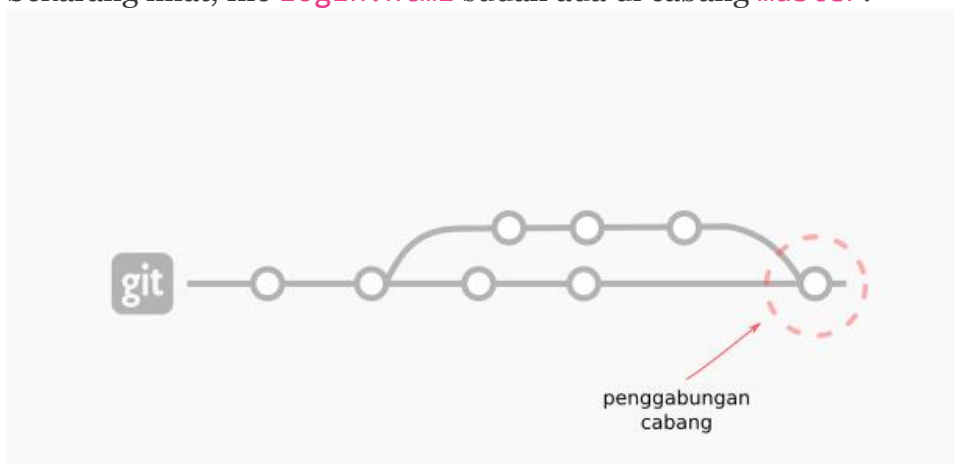
Anggaplah kita sudah selesai membuat fitur login di cabang `halaman_login`. Sekarang kita ingin Menggabungkannya dengan cabang `master` (utama). Pertama, kita harus pindah dulu ke cabang `master`.

```
git checkout master
```

Setelah itu, barulah kita bisa menggabungkan dengan perintah `git merge`.

```
git merge halaman_login
```

Sekarang lihat, file `login.html` sudah ada di cabang `master`.



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



## 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 repostiri. Dia harus bertindak adil, kode mana yang harus diambil. Biasanya akan ada proses diskusi dulu dalam mengambil keputusan.

Sekarang kita akan coba membuat bentrokan.

Pindah dulu ke branch `halaman_login`...

```
git checkout halaman_login
```

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

```
$ git diff
diff --git a/login.html b/login.html
index 23a3f5c..eea5658 100644
--- a/login.html
+++ b/login.html
@@ -1,1 @@
-di sini berisi kode untuk halaman login
+<p>di sini berisi kode untuk halaman login<p>
```

Setelah itu, lakukan *commit* lagi:

```
git add login.html
git commit -m "ubah isi login.html"
```

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

Setelah itu, lakukan *commit* di cabang `master`

```
git add login.html
git commit -m "ubah isi login.html di cabang master"
```

Terakhir, coba gabungkan cabang `halaman_login` dengan cabang `master`, maka akan terjadi bentrok.

```
$ 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.
```

Nah, kita disuruh perbaiki kode yang bentrok. Sekarang buka `login.html` dengan teks editor.

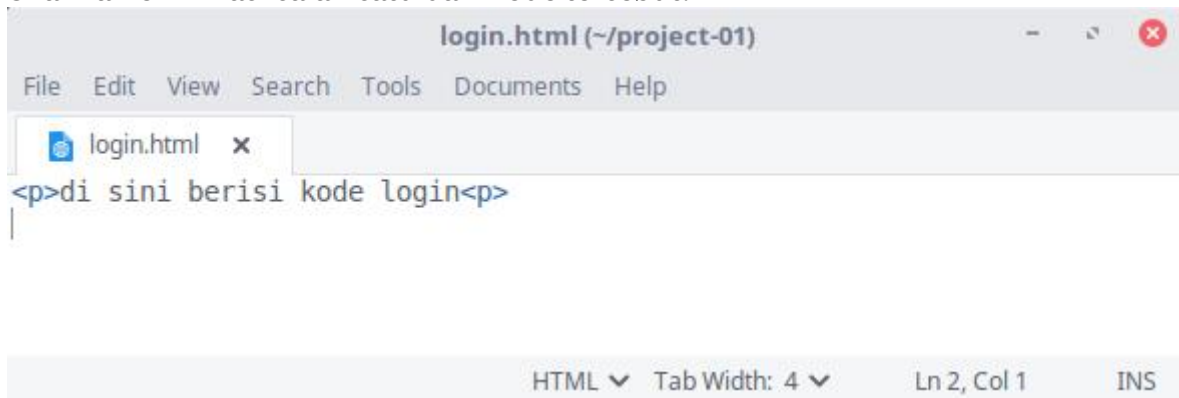


```
*login.html (~/.project-01)
File Edit View Search Tools Documents Help
*login.html x
<<<<<< HEAD
<p>di sini berisi kode login<p>
=====
<p>ok ini bentrokan</p>
<p>di sini berisi kode untuk halaman login<p>
>>>>>> halaman_login
```

HTML Tab Width: 4 Ln 6, Col 22 INS

Kedua kode cabang dipisahkan dengan tanda `=====`. Sekarang.. tugas kita adalah memperbaikinya.

Silahkan eliminasi salah satu dari kode tersebut.



```
login.html (~/.project-01)
File Edit View Search Tools Documents Help
login.html x
<p>di sini berisi kode login<p>
|
```

HTML Tab Width: 4 Ln 2, Col 1 INS

Setelah itu lakukan *commit* untuk menyimpan perubahan ini.

```
git add login.html
git commit -m "perbaiki konflik"
```

## Menghapus Cabang

Cabang yang sudah mati atau tidak ada pengembangan lagi, sebaiknya dihapus. Agar repositori kita bersih dan rapi.

Cara menghapus cabang, gunakan perintah `git branch` dengan argumen `-d` dan diikuti dengan nama cabangnya.

Contoh:

```
git branch -d halaman_login
```

**Sumber Materi :**

1. <https://git-scm.com/doc>
2. <https://www.petanikode.com/tutorial/git/>

**LAPORAN RESMI**

1. Lakukan percobaan pada modul ini dengan cara print screen Percobaan yang anda lakukan
2. Analisa latihan yang telah dilakukan.
3. Berikan kesimpulan dari praktikum ini.