

# Praktikum 10

## Git Lanjut Bagian II

### Membuat Repositori Baru dalam Proyek

Repositori (*repository*) dalam bahasa Indonesia artinya gudang. Repositori merupakan istilah yang digunakan untuk direktori proyek yang menggunakan Git. Jika kita memiliki sebuah direktori dengan nama **proyek-01** dan di dalamnya sudah menggunakan git, maka kita sudah punya repositori bernama **proyek-01**.

### Membuat Repositori

Pembuatan repositori dapat dilakukan dengan perintah **git init nama-dir**. Contoh:

```
git init proyek-01
```

Perintah tersebut akan membuat direktori bernama **proyek-01**. Kalau direktorinya sudah ada, maka Git akan melakukan inisialisasi di dalam direktori tersebut.


Perintah **git init** akan membuat sebuah direktori bernama **.git** di dalam proyek kita. Direktori ini digunakan Git sebagai database untuk menyimpan perubahan yang kita lakukan. Hati-hati, kalau kita menghapus direktori ini, maka semua rekaman atau catatan yang dilakukan oleh Git akan hilang.

### Contoh-contoh lain

Perintah berikut ini akan membuat repositori pada direktori saat ini (*working directory*).

```
git init .
```

Tanda titik (.) artinya kita akan membuat repository pada direktori tempat kita berada saat ini.



```
petanikode@imajinasi ~/proyek-ku
petanikode@imajinasi ~ $ mkdir proyek-ku
petanikode@imajinasi ~ $ cd proyek-ku/
petanikode@imajinasi ~/proyek-ku $ git init .
Initialized empty Git repository in /home/petanikode/proyek-ku/.git/
petanikode@imajinasi ~/proyek-ku $
```

Perintah berikut ini akan membuat repositori pada direktori `/var/www/html/proyekweb/`.

```
git init /var/www/html/proyekweb
```

## .gitignore

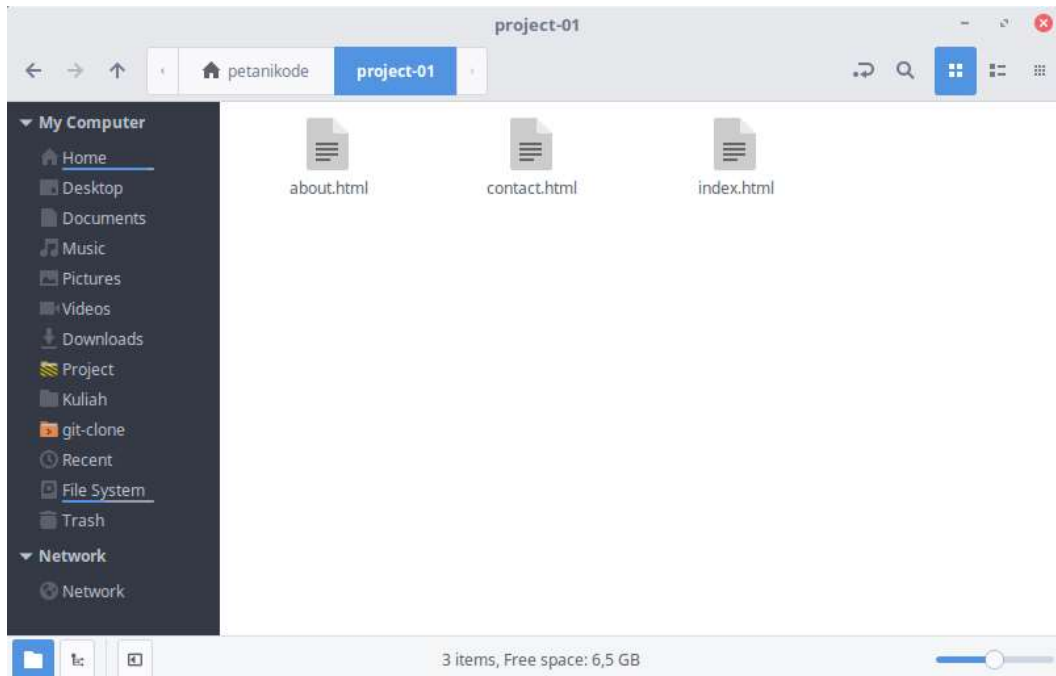
**.gitignore** merupakan sebuah file yang berisi daftar nama-nama file dan direktori yang akan diabaikan oleh Git. Perubahan apapun yang kita lakukan terhadap file dan direktori yang sudah masuk ke dalam daftar **.gitignore** tidak akan dicatat oleh Git. Cara menggunakan **.gitignore**, buat saja sebuah file bernama **.gitignore** dalam root direktori proyek/repositori. File: **.gitignore**

```
/vendor/  
/upload/  
/cache  
test.php
```

Pada contoh file **.gitignore** di atas, kita memasukan direktori **vendor**, **upload**, **cache** dan file **test.php**. File dan direktori tersebut akan diabaikan oleh Git. Pembuatan file **.gitignore** sebaiknya dilakukan di awal pembuatan repositori.

## Simpan Perubahan Revisi dengan Git Commit

Pada tutorial sebelumnya, kita sudah membuat repositori kosong. Belum ada apa-apa di sana. Sekarang coba tambahkan sebuah file baru. Sebagai contoh, Kita akan menambahkan tiga file HTML kosong.



Setelah ditambahkan, coba ketik perintah `git status` untuk melihat status repositorinya.

```
petanikode@imajinasi ~/project-01
petanikode@imajinasi ~/project-01 $ git status
On branch master

Initial commit

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

        about.html
        contact.html
        index.html

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

Berdasarkan keterangan di atas, saat ini kita berada cabang (*branch*) *master* dan ada tiga file yang belum ditambahkan ke Git.

## Tiga Kelompok Kondisi File dalam Git

Sebelum kita membuat revisi, kita akan berkenalan dulu dengan tiga kondisi file dalam Git.

### 1. Modified

*Modified* adalah kondisi dimana revisi atau perubahan sudah dilakukan, tetapi belum ditandai dan belum disimpan di *version control*. Contohnya pada gambar di atas, ada tiga file HTML yang dalam kondisi *modified*.

## 2. Staged

*Staged* adalah kondisi dimana revisi sudah ditandai, tetapi belum disimpan di *version control*. Untuk mengubah kondisi file dari *modified* ke *staged* gunakan perintah `git add nama_file`. Contoh:

```
git add index.html
```

## 3. Committed

Committed adalah kondisi dimana revisi sudah disimpan di *version control*. perintah untuk mengubah kondisi file dari *staged* ke *committed* adalah `git commit`.

## Membuat Revisi Pertama

Baiklah, sekarang kita akan sudah tahu kondisi-kondisi file dalam Git. Selanjutnya, silahkan ubah kondisi tiga file HTML tadi menjadi *staged* dengan perintah `git add`.

```
git add index.html
git add about.html
git add contact.html
```

Atau kita bisa melakukannya seperti ini:

```
git add index.html about.html contact.html
```

atau:

```
git add *.html
```

Atau seperti ini (semua file dan direktori):

```
git add .
```

Setelah itu, cobalah ketik perintah `git status` lagi. Kondisi filenya sekarang akan menjadi *staged*.

```
petanikode@imajinasi ~/project-01

nothing added to commit but untracked files present (use "git add" to track)
petanikode@imajinasi ~/project-01 $ git add *.html
petanikode@imajinasi ~/project-01 $ git status
On branch master

Initial commit

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

        new file:   about.html
        new file:   contact.html
        new file:   index.html

petanikode@imajinasi ~/project-01 $
```

Setelah itu, ubah kondisi file tersebut ke *committed* agar semua perubahan disimpan oleh Git.

```
git commit -m "Commit pertama"
```

Setelah itu, coba cek dengan perintah `git status` lagi.

```
petanikode@imajinasi ~/project-01

petanikode@imajinasi ~/project-01 $ git commit -m "commit pertama"
[master (root-commit) cf08ca0] commit pertama
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 about.html
 create mode 100644 contact.html
 create mode 100644 index.html
petanikode@imajinasi ~/project-01 $ git status
On branch master
nothing to commit, working directory clean
petanikode@imajinasi ~/project-01 $
```

Selamat, revisi pertama sudah kita buat. Selanjutnya cobalah untuk membuat revisi kedua.

## Membuat Revisi kedua

Ceritanya ada perubahan yang akan kita lakukan pada file `index.html`. Silahkan modifikasi isi file `index.html`. Sebagai contoh Kita mengisinya seperti ini.

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

Setelah itu ketik lagi perintah `git status`.

```
petanikode@imajinasi ~/project-01
nothing to commit, working directory clean
petanikode@imajinasi ~/project-01 $ 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)

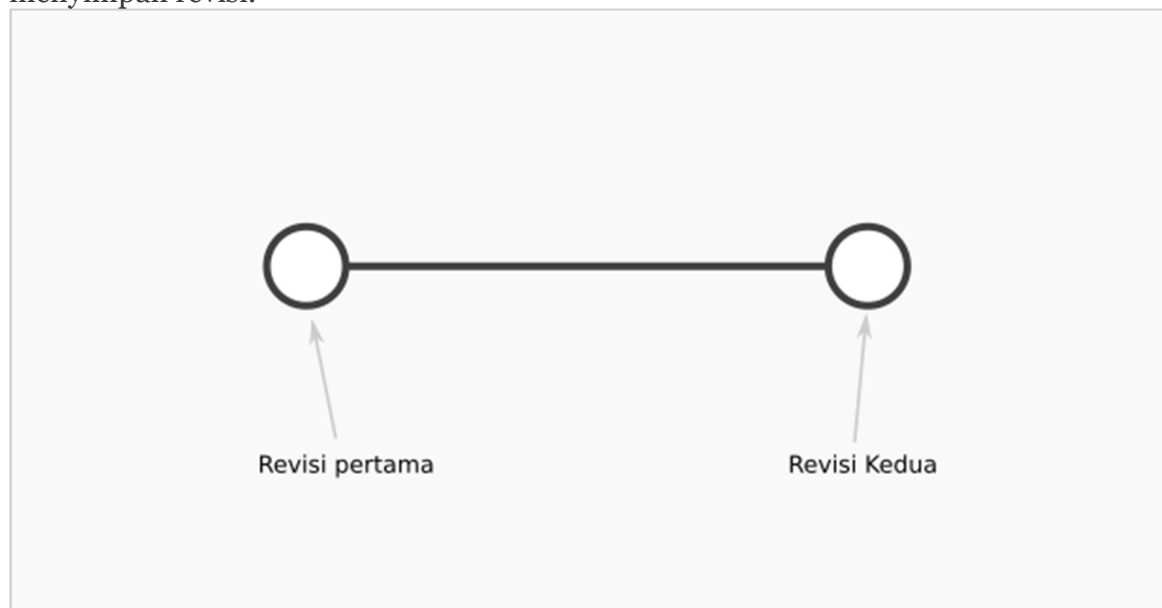
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
petanikode@imajinasi ~/project-01 $
```

Terlihat di sana, file `index.html` sudah dimodifikasi. Kondisinya skarang berada dalam *modified*. Lakukan *commit* lagi seperti revisi pertama.

```
git add index.html
git commit -m "ditambahkan isi"
```

Dengan demikian, revisi kedua sudah disipan oleh Git. Mungkin anda belum tahu maksud dari argumen `-m`, argumen tersebut untuk menambahkan pesan setiap menyimpan revisi.



Sekarang Git sudah mencatat dua revisi yang sudah kita lakukan. Kita bisa ibaratkan revisi-revisi ini sebagai *checkpoint* pada Game. Apabila nanti ada kesalahan, kita bisa kembali ke *checkpoint* ini.

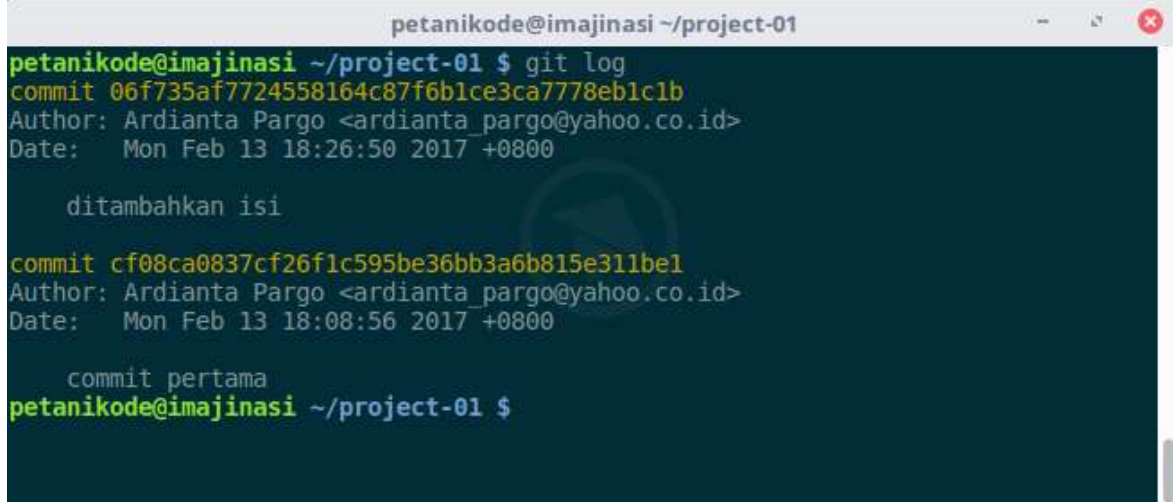
## Melihat Catatan Log Revisi

Pada tutorial sebelumnya, kita sudah membuat dua revisi pada repositori `project-01`. Sekarang bagaimana caranya kita melihat catatan log dari revisi-revisi tersebut?

Git sudah menyediakan perintah **git log** untuk melihat catatan log perubahan pada repositori. Contoh penggunaannya:

```
git log
```

Maka kita akan melihat log perubahan apa saja yang sudah dilakukan dalam repositori.



```
petanikode@imajinasi ~/project-01 $ git log
commit 06f735af7724558164c87f6b1ce3ca7778eb1c1b
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Mon Feb 13 18:26:50 2017 +0800

    ditambahkan isi

commit cf08ca0837cf26f1c595be36bb3a6b815e311be1
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Mon Feb 13 18:08:56 2017 +0800

    commit pertama
petanikode@imajinasi ~/project-01 $
```

Pada gambar di atas, terdapat dua revisi perubahan yang telah dilakukan.

## Log yang Lebih Pendek

Untuk menampilkan log yang lebih pendek, kita bisa menambahkan argumen **--oneline**.

```
git log --oneline
```

Maka akan menghasilkan output:

```
06f735a ditambahkan isi
cf08ca0 commit pertama
```

## Log pada Nomer Revisi/Commit

Untuk melihat log pada revisi tertentu, kita bisa memasukkan nomer revisi/commit.

```
git log cf08ca0837cf26f1c595be36bb3a6b815e311be1
```

Maka akan menghasilkan output:

```
commit cf08ca0837cf26f1c595be36bb3a6b815e311be1
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date: Mon Feb 13 18:08:56 2017 +0800
```

```
commit pertama
```

## Log pada File Tertentu

Untuk melihat revisi pada file tertentu, kita dapat memasukan nama filenya.

```
git log index.html
```

Maka akan menghasilkan output:

```
commit 06f735af7724558164c87f6b1ce3ca7778eb1c1b
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date:   Mon Feb 13 18:26:50 2017 +0800

    ditambahkan isi

commit cf08ca0837cf26f1c595be36bb3a6b815e311be1
Author: Ardianta Pargo <ardianta_pargo@yahoo.co.id>
Date:   Mon Feb 13 18:08:56 2017 +0800

    commit pertama
```

Karena file `index.html` sudah direvisi sebanyak dua kali.

## Log Revisi yang dilakukan oleh Author Tertentu

Misalkan dalam repositori dikerjakan oleh banyak orang. Maka kita dapat melihat revisi apa saja yang dilakukan oleh orang tertentu dengan perintah berikut.

```
git log --author='Petani Kode'
```

## Melihat Perbandingan Revisi dengan Git Diff

Pada tutorial sebelumnya, kita sudah belajar cara melihat log revisi di repositori. Sekarang kita kan pelajari perintah `git diff`, fungsinya untuk melihat perbedaan perubahan di revisi.

## Melihat Perbandingan Perubahan yang Dilakukan pada Revisi

Gunakan perintah berikut ini untuk melihat perubahan yang dilakukan pada revisi tertentu.

```
git diff cf08ca0837cf26f1c595be36bb3a6b815e311be1
```

`cf08ca0837cf26f1c595be36bb3a6b815e311be1` adalah nomer revisi yang ingin dilihat.



```
petanikode@imajinasi ~/project-01
petanikode@imajinasi ~/project-01 $ git diff cf08ca0837cf26f1c595be36bb3a6b815e311be1
diff --git a/index.html b/index.html
index e69de29..481cc02 100644
--- a/index.html
+++ b/index.html
@@ -0,0 +1,10 @@
+<!DOCTYPE html>
+<html>
+  <head>
+    <meta charset="utf-8">
+    <title>Belajar Git - Project 01</title>
+  </head>
+  <body>
+    <p>Hello Semua, Saya sedang belajar Git</p>
+  </body>
+</html>
petanikode@imajinasi ~/project-01 $
```

Lihatlah hasil di atas, simbol plus (+) artinya kode yang ditambahkan. Sedangkan kalau ada kode yang dihapus simbolnya akan menggunakan minus (-).

### Contoh:

Ditambahkan:

```
+ <p>ini kode yang ditambahkan</p>
```

Dihapus:

```
- <i>ini kode yang dihapus</i>
```

Dimodifikasi/diubah:

```
- <span>ini kode sebelum diubah</span>
+ <span>ini kode sesudah diubah</span>
```

Sekarang kita akan mencoba merubah isi dari `index.html`.

Sebelum diubah:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
```

```
<title>Belajar Git - Project 01</title>
</head>
<body>
  <p>Hello Semua, 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 sedang belajar Git</p>
  </body>
</html>
```

Setelah itu lakukan jalankan perintah `git diff` lagi.

Apa yang dilakukan `git diff`? Perintah `git diff` akan membandingkan perubahan yang baru saja dilakukan dengan revisi/commit terakhir.

## Melihat Perbandingan pada File

Apa bila kita melakukan banyak perubahan, maka akan banyak sekali tampil output. Karena itu, kita mungkin hanya perlu melihat perubahan untuk file tertentu saja. Untuk melihat perbandingan perubahan pada file tertentu, gunakan perintah berikut.

```
git diff index.html
```

Perintah di atas akan melihat perbedaan perubahan pada file `index.html` saja.

## Melihat Perbandingan antar Revisi/Commit

Perintah untuk membandingkan perubahan pada revisi dengan revisi yang lain adalah sebagai berikut.

```
git diff <nomer commit> <nomer commit>
```

contoh:

```
git diff cf08ca0837cf26f1c595be36bb3a6b815e311be1
06f735af7724558164c87f6b1ce3ca7778eb1c1b
```

## Perbandingan Antar Cabang (Branch)

Kita memang belum masuk ke materi percabangan di Git. Tapi tidak ada salahnya mengetahui cara melihat perbandingan perubahan antar cabang.

```
git diff <nama cabang> <nama cabang>
```

### **Sumber Materi :**

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

### **LAPORAN RESMI**

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