

# Apa itu Git dan Kenapa Penting bagi Programmer?

Git adalah salah satu *tool* yang sering digunakan dalam proyek pengembangan software.

Git bahkan menjadi *tool* yang wajib dipahami oleh programmer, karena banyak digunakan di mana-mana.

Pada kesempatan ini kita akan belajar Git dari dasar.

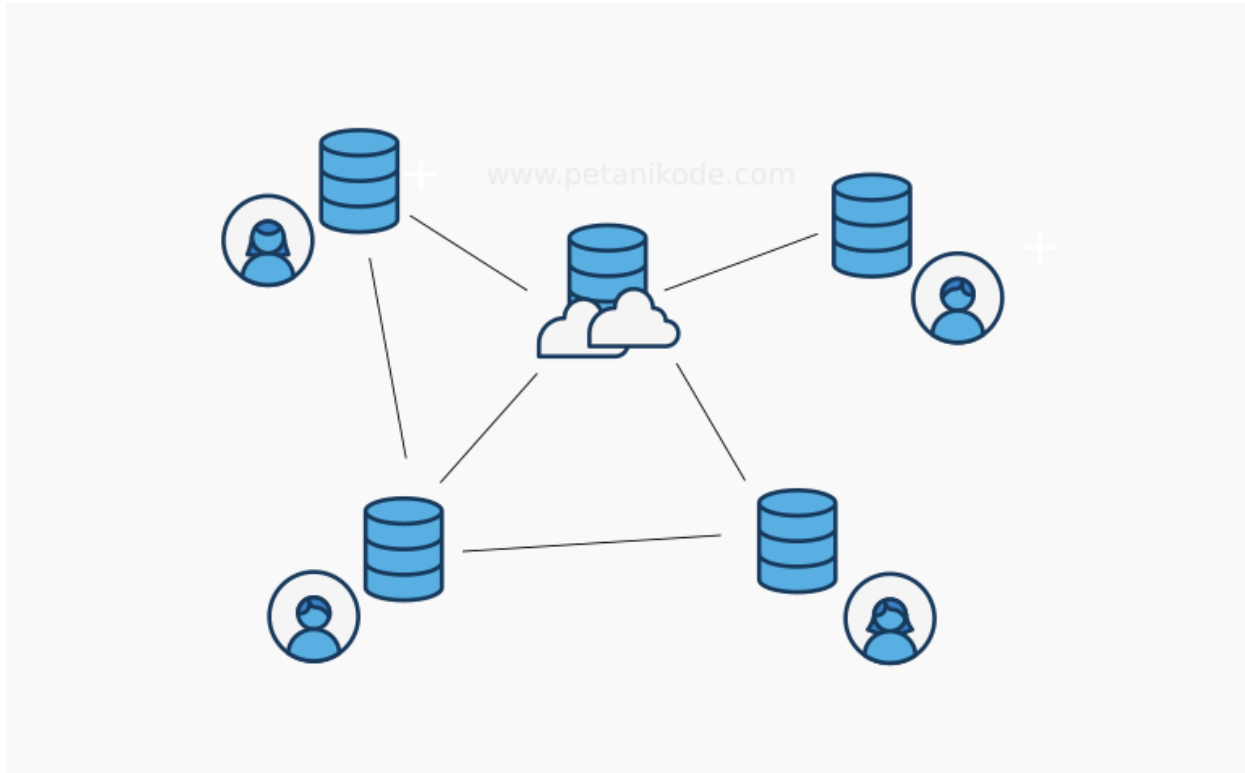
Artikel ini hanya akan membahas pengenalan Git saja. Untuk mempelajari Git lebih lanjut, saya sudah menyediakan link di bagian akhir.

## Mengenal Git

Git adalah salah satu sistem pengontrol versi (*Version Control System*) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds.

Pengontrol versi bertugas mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang maupun sendiri.

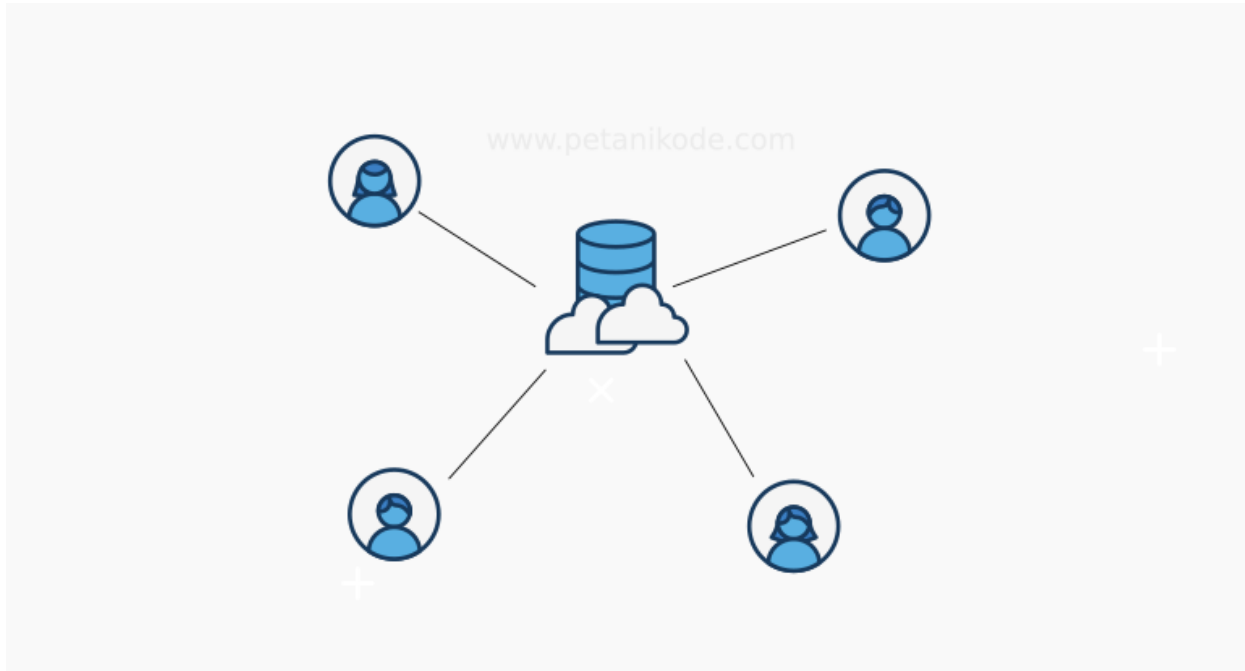
Git dikenal juga dengan *distributed revision control* (VCS terdistribusi), artinya penyimpanan database Git tidak hanya berada dalam satu tempat saja.



Semua orang yang terlibat dalam pengkodean proyek akan menyimpan database Git, sehingga akan memudahkan dalam mengelola proyek baik online maupun offline.

Dalam Git terdapat [merge](#) untuk menyebut aktifitas penggabungan kode.

Sedangkan pada VCS (Version Control System) yang terpusat... database disimpan dalam satu tempat dan setiap perubahan disimpan ke sana.



VCS terpusat memiliki beberapa kekurangan:

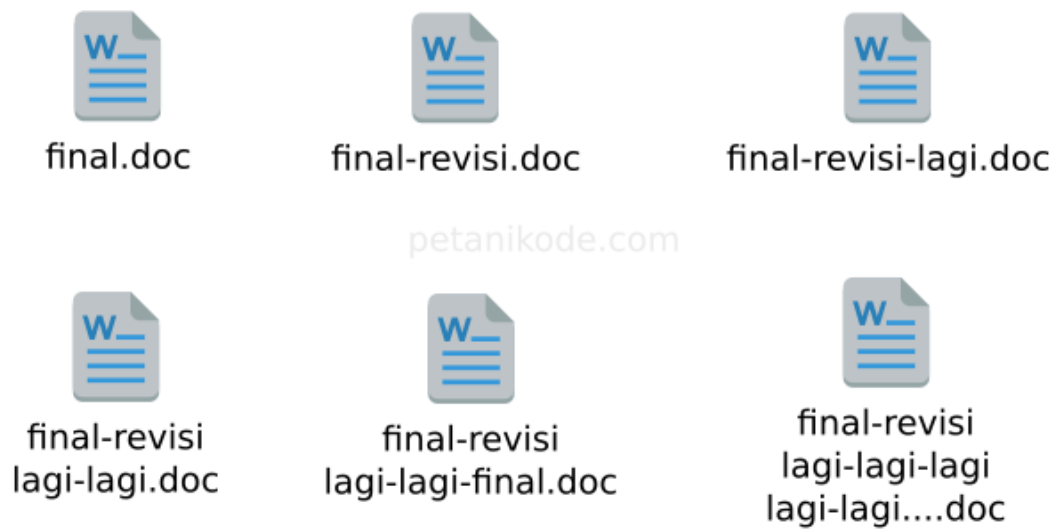
- Semua tim harus terkoneksi ke jaringan untuk mengakses source-code;
- Tersimpan di satu tempat, nanti kalau server bermasalah bagaimana?

Karena itu, Git hadir untuk menutupi kekurangan yang dimiliki oleh VCS terpusat.

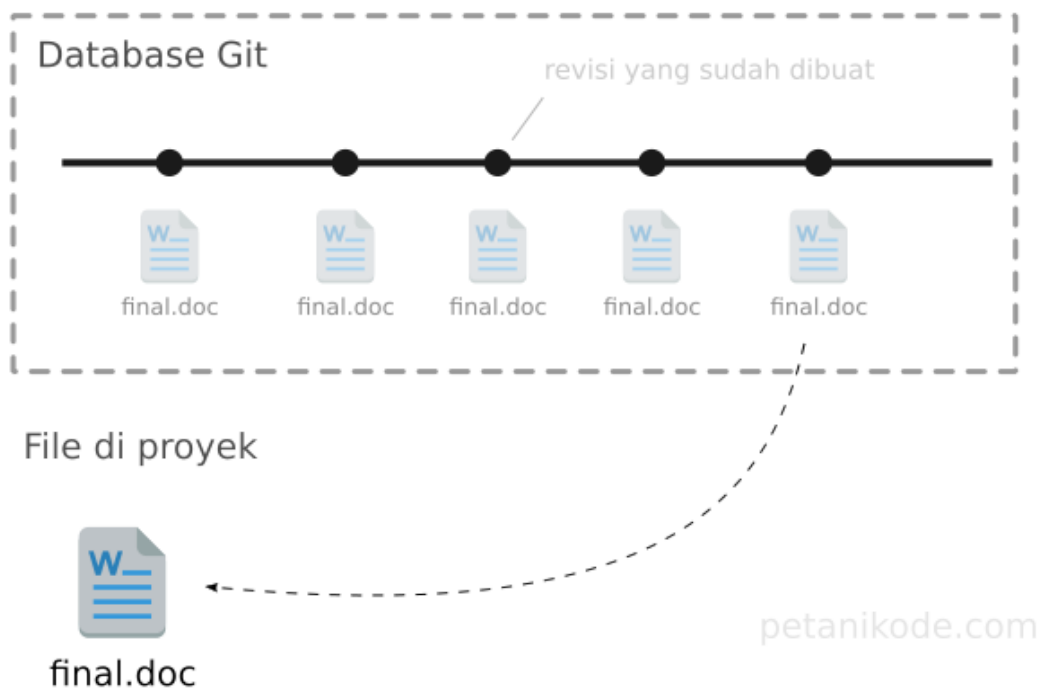
## Apa yang dilakukan oleh Git?

Git sebenarnya akan memantau semua perubahan yang terjadi pada file proyek. Lalu menyimpannya ke dalam database.

Sebelum menggunakan Git:



Setelah menggunakan Git:



Apa perbedaannya?

Saat kita ingin menyimpan semua perubahan pada file, biasanya kita membuat file baru dengan “save as”. Lalu, file akan menumpuk dalam direktori proyek seperti pada ilustrasi di atas.

Tapi setelah menggunakan Git...

Hanya akan ada satu file dalam proyek dan perubahannya disimpan dalam database.

Git hanya akan menyimpan delta perubahannya saja, dia tidak akan menyimpan seluruh isi file yang akan memakan banyak memori.

Git memungkinkan kita kembali ke versi revisi yang kita inginkan.

## **Kenapa Git Penting Bagi Programmer?**

Selain untuk mengontrol versi, git juga digunakan untuk kolaborasi.

Saat ini Git menjadi salah satu *tool* terpopuler yang digunakan pada pengembangan software *open souce* maupun *closed source*.

Google, Microsoft, Facebook dan berbagai perusahaan raksasa lainnya menggunakan Git.

Jadi, buat kamu yang punya impian ingin bekerja di sana, maka kamu harus bisa Git.

Selain itu, berikut ini ada beberapa manfaat yang akan kamu rasakan setelah bisa menggunakan Git.

1. Bisa menyimpan seluruh versi source code;
2. Bisa paham cara kolaborasi dalam proyek;
3. Bisa ikut berkontribusi ke proyek open-source;
4. Lebih aman digunakan untuk kolaborasi, karena kita bisa tahu apa yang diubah dan siapa yang mengubahnya.
5. Bisa memahami cara *deploy* aplikasi modern;
6. Bisa membuat blog dengan SSG.
7. dan sebagainya...

# Tutorial Git #1: Cara Install Git dan Konfigurasi Awal yang Harus Dilakukan

## 1. Cara Install Git di Linux

Instalasi Git pada Distro keluarga Debian dapat menggunakan perintah `apt`.

```
sudo apt install git
```

atau

```
sudo apt-get install git
```

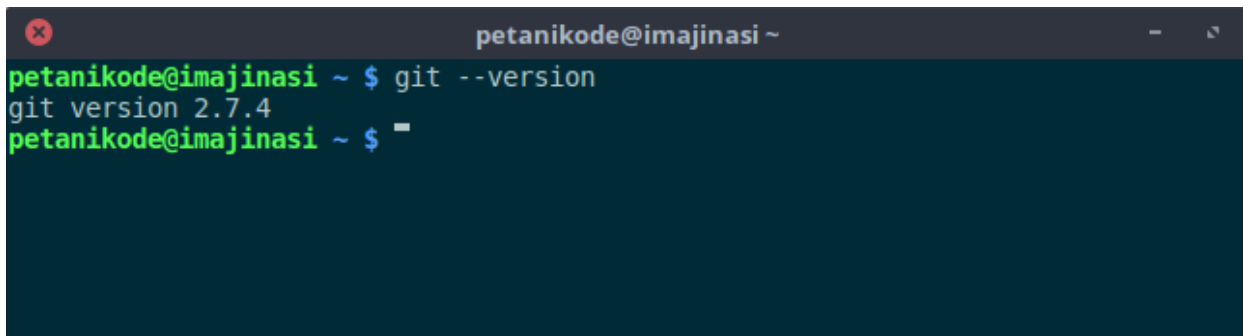
Pada Fedora:

```
yum install git
```

Setelah itu, coba periksa versi yang terinstal dengan perintah:

```
git --version
```

Pada komputer saya, versi yang terinstal adalah versi 2.7.4.

A screenshot of a terminal window with a dark background. The window title is 'petanikode@imajinasi ~'. The prompt is 'petanikode@imajinasi ~ \$'. The command 'git --version' has been entered, and the output 'git version 2.7.4' is displayed on the next line. The prompt is now 'petanikode@imajinasi ~ \$' followed by a cursor.

```
petanikode@imajinasi ~ $ git --version
git version 2.7.4
petanikode@imajinasi ~ $
```

## 2. Cara Install Git di Windows

Instalasi Git di Windows memang tidak seperti di Linux yang ketik perintah langsung terinstal.

Kita harus men-download dulu, kemudian melakukan ritual *next>next>finish*.

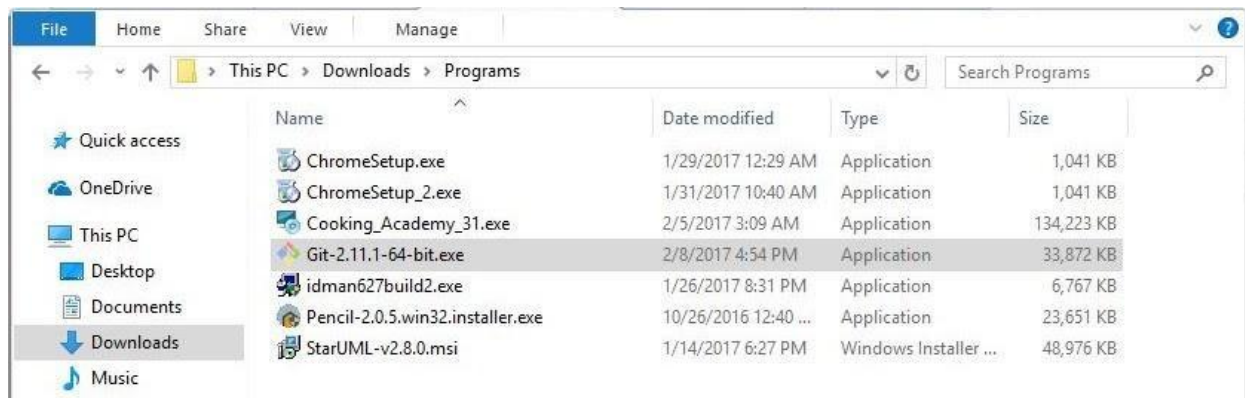
Tapi dalam ritual tersebut, ada pilihan yang harus diperhatikan agar perintah `git` dapat dikenali di CMD.

## Download Git

Silahkan buka website resminya Git ( [git-scm.com](https://git-scm.com)). Kemudian unduh Git sesuai dengan arsitektur komputer kita. Kalau menggunakan 64bit, unduh yang 64bit. Begitu juga kalau menggunakan 32bit.

## Langkah-langkah Install Git di Windows

Baiklah, mari kita mulai ritual instalnya. Silahkan klik 2x file instaler Git yang sudah diunduh.

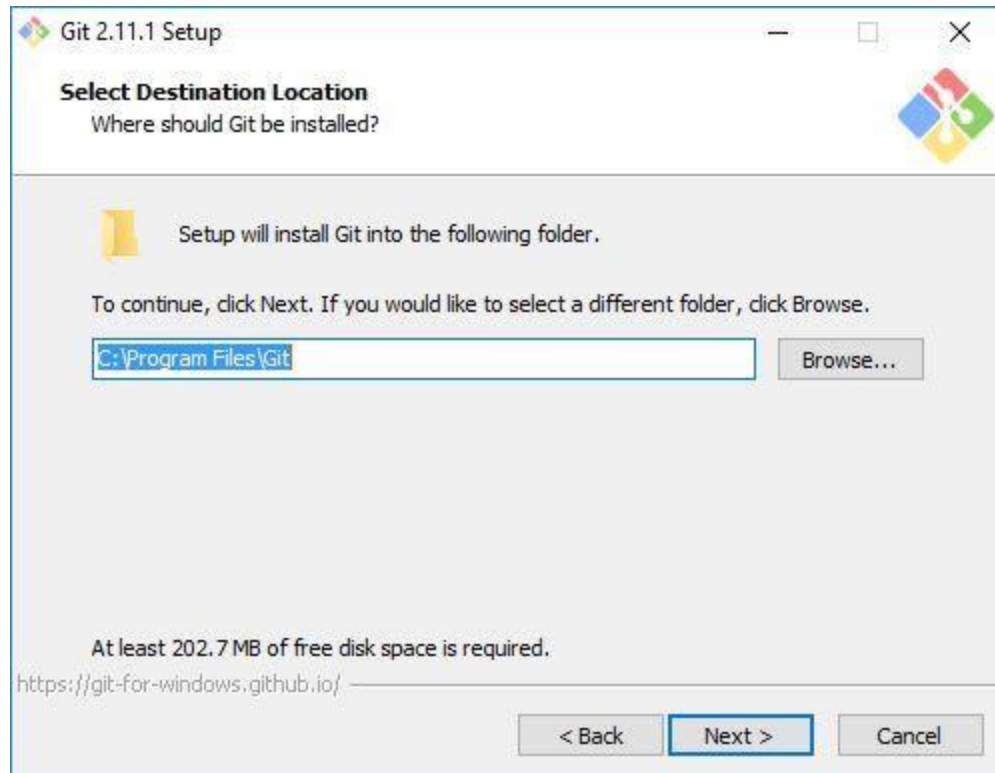


Maka akan muncul informasi lisensi Git, klik *Next* > untuk melanjutkan.

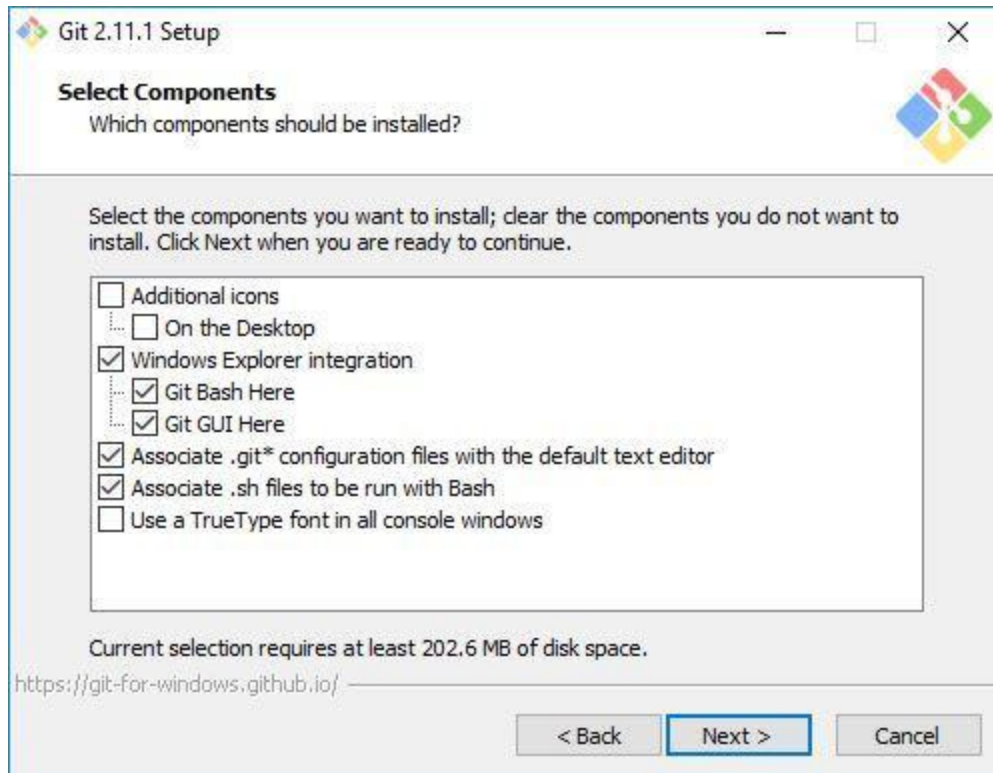




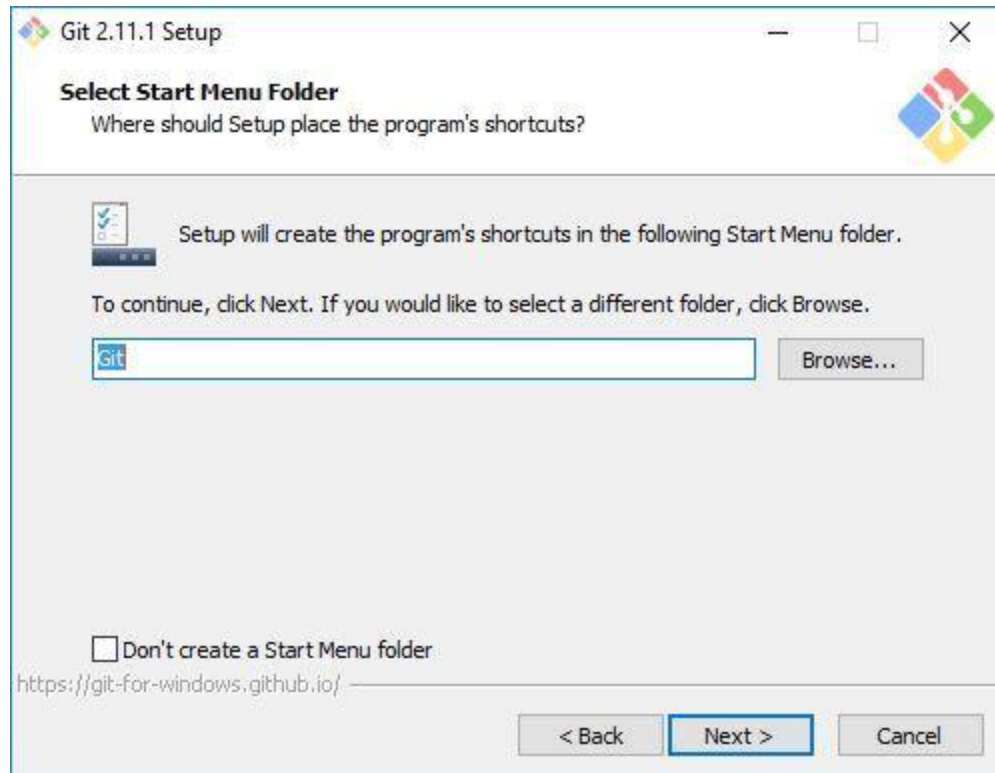
Selanjutnya menentukan lokasi instalasi. Biarkan saja apa adanya, kemudian klik *Next >*.



Selanjutnya pemilihan komoponen, biarkan saja seperti ini kemudian klik *Next* >.



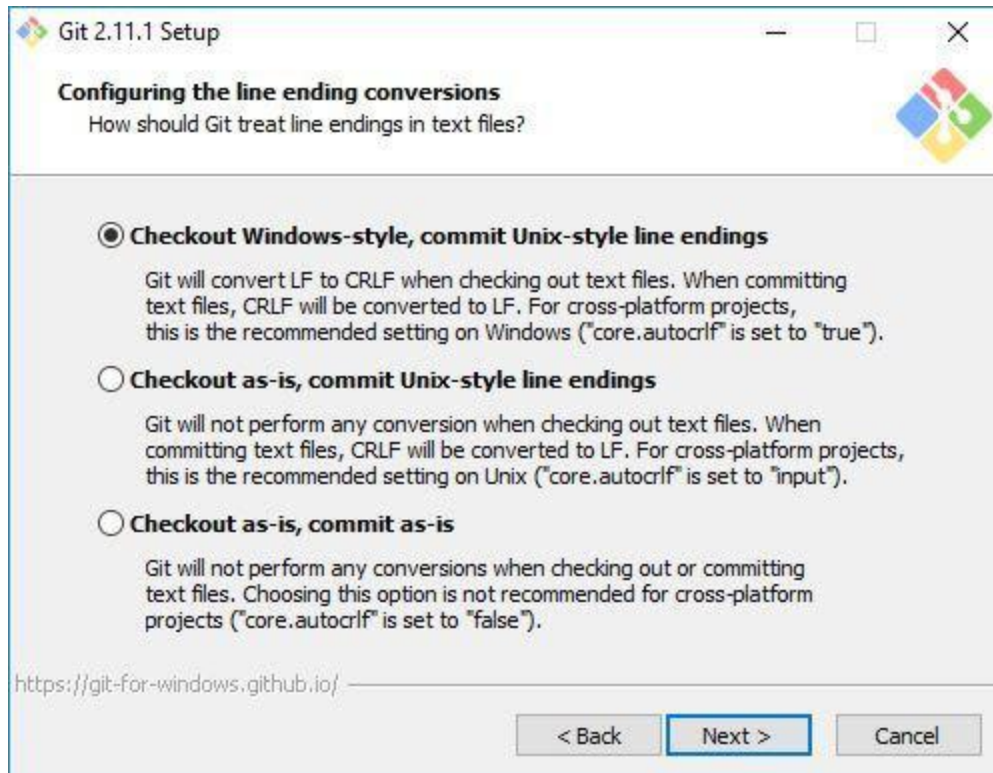
Selanjutnya pemilihan direktori start menu, klik *Next >*.



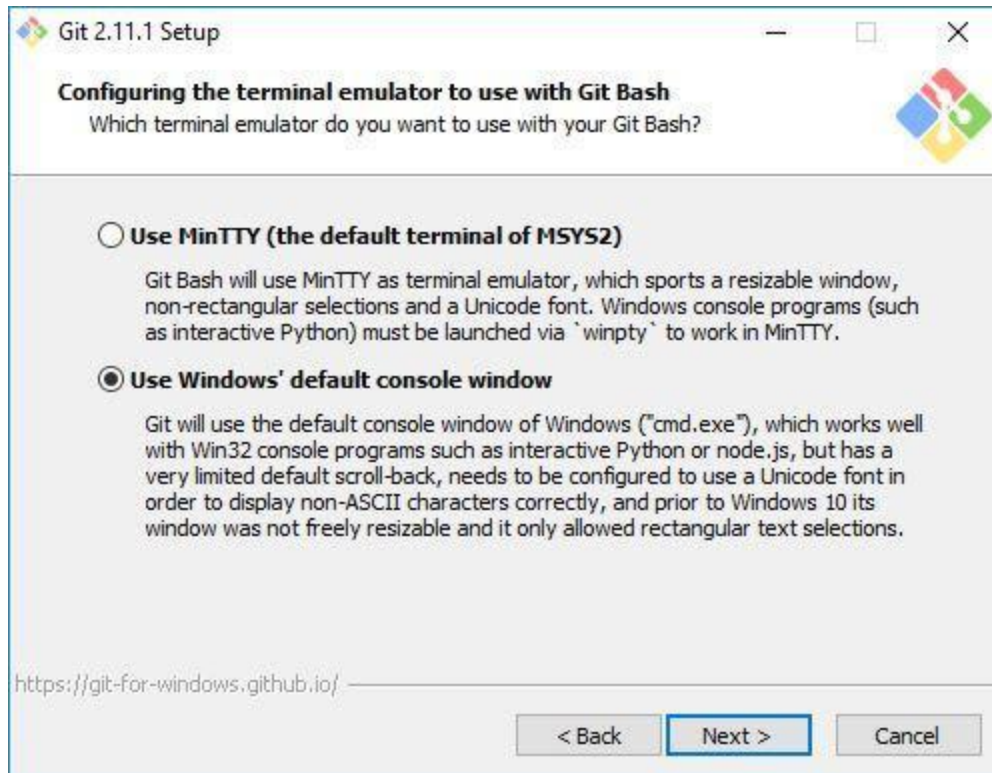
Selanjutnya pengaturan *PATH Environment*. Pilih yang tengah agar perintah `git` dapat di kenali di *Command Prompt* (CMD). Setelah itu klik *Next >*.



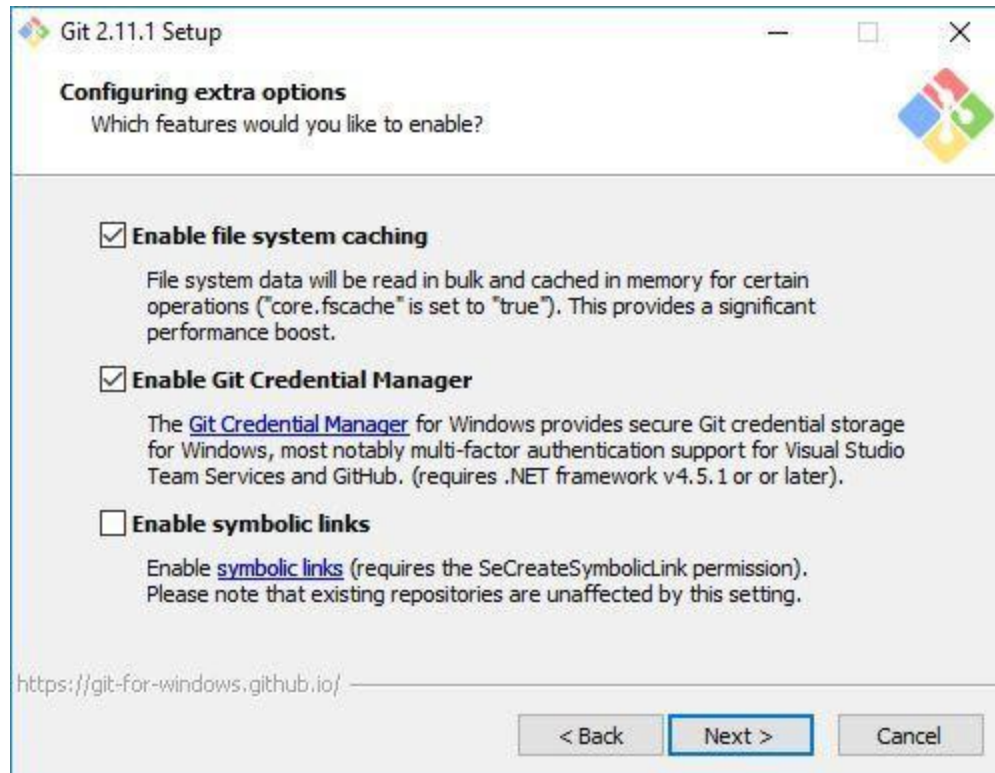
Selanjutnya konversi *line ending*. Biarkan saja seperti ini, kemudian klik *Next >*.



Selanjutnya pemilihan emulator terminal. Pilih saja yang bawah, kemudian klik *Next >*.

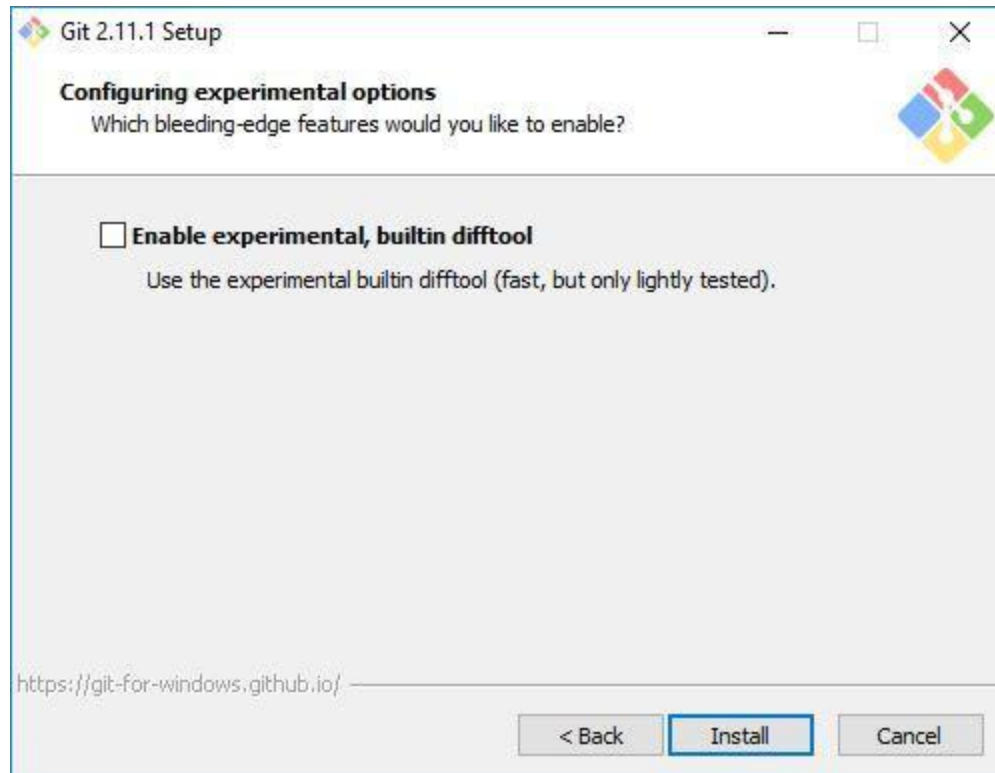


Selanjutnya pemilihan opsi ekstra. Klik saja *Next >*.

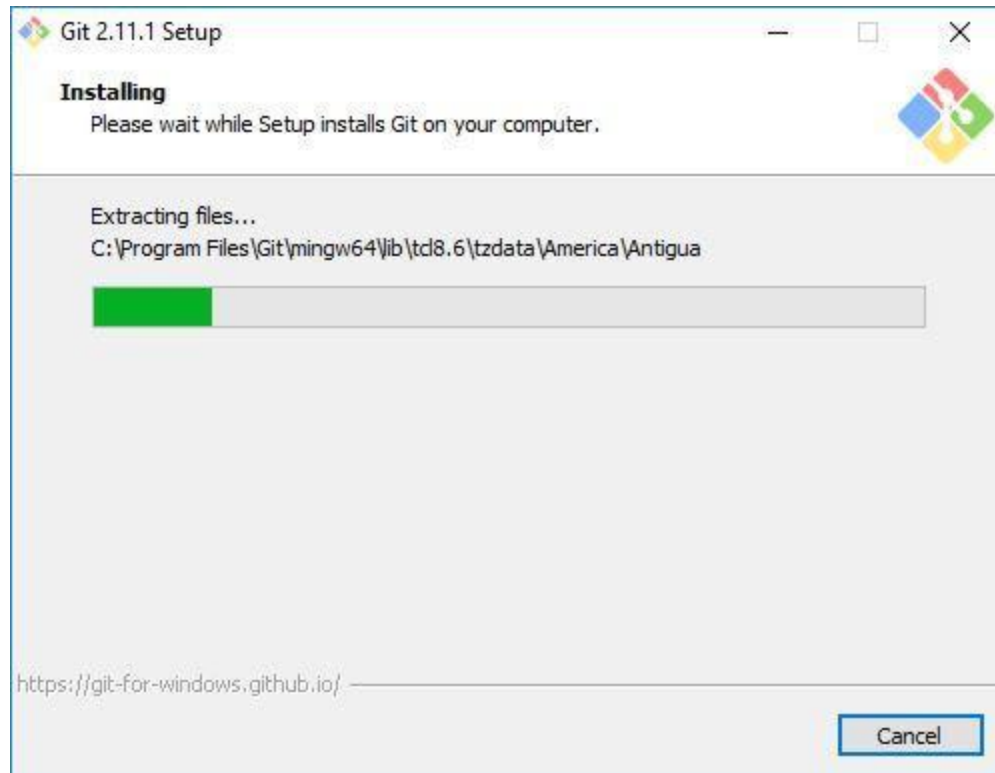


Selanjutnya pemilihan opsi ekspreimental, langsung saja klik *Install* untuk memaulai instalasi.





Tunggu beberapa saat, instalasi sedang dilakukan.



Setelah selesai, kita bisa langsung klik *Finish*.



Selamat, Git sudah terinstal di Windows. Untuk mencobanya, silahkan buka CMD atau PowerShell, kemudian ketik perintah `git --version`.

The image shows a Windows Command Prompt window titled 'cmd - Command Prompt'. The text inside the window is as follows:

```
Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.  
  
C:\Users\Development>git --version  
git version 2.11.1.windows.1  
  
C:\Users\Development>
```

### 3. Konfigurasi Awal yang Harus Dilakukan

Ada beberapa konfigurasi yang harus dipersiapkan sebelum mulai menggunakan Git, seperti *name* dan *email*.

Silahkan lakukan konfigurasi dengan perintah berikut ini.

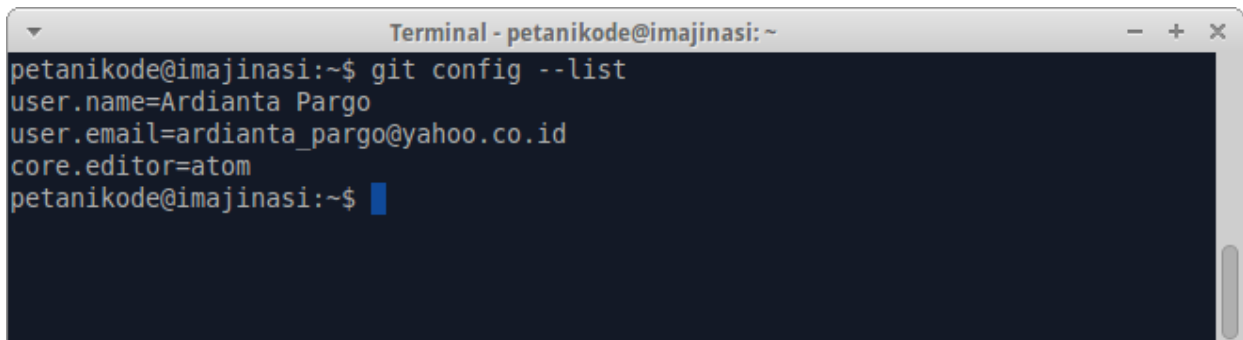
```
git config --global user.name "Petani Kode"
```

```
git config --global user.email contoh@petanikode.com
```

Kemudian periksa konfigurasinya dengan perintah:

```
git config --list
```

Apabila berhasil tampil seperti gambar berikut ini, berarti konfigurasi berhasil.

A screenshot of a terminal window titled "Terminal - petanikode@imajinasi: ~". The terminal shows the command "git config --list" being executed, resulting in the following output: "user.name=Ardianta Pargo", "user.email=ardianta\_pargo@yahoo.co.id", and "core.editor=atom". The prompt "petanikode@imajinasi:~\$" is visible at the bottom of the terminal.

```
petanikode@imajinasi:~$ git config --list
user.name=Ardianta Pargo
user.email=ardianta_pargo@yahoo.co.id
core.editor=atom
petanikode@imajinasi:~$
```

Konfigurasi `core.editor` bersifat opsional. Sedangkan *name* dan *email* wajib.

Jika kamu memiliki akun Github, Gitlab, Bitbucket atau yang lainnya...maka *username* dan *email* harus mengikuti akun tersebut agar mudah diintegrasikan.

# Tutorial Git #2: Cara 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!

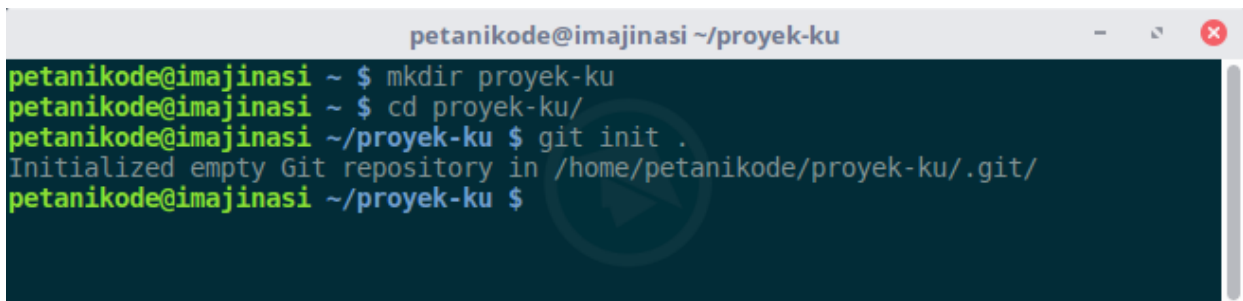
Kalau kita menghapus direktori ini, maka semua rekaman atau catatan yang dilakukan oleh Git akan hilang.

## 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.

A terminal window titled 'petanikode@imajinasi ~/proyek-ku' showing the following commands and output:

```
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.

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

Pada contoh file `.gitignore` di atas, saya 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.

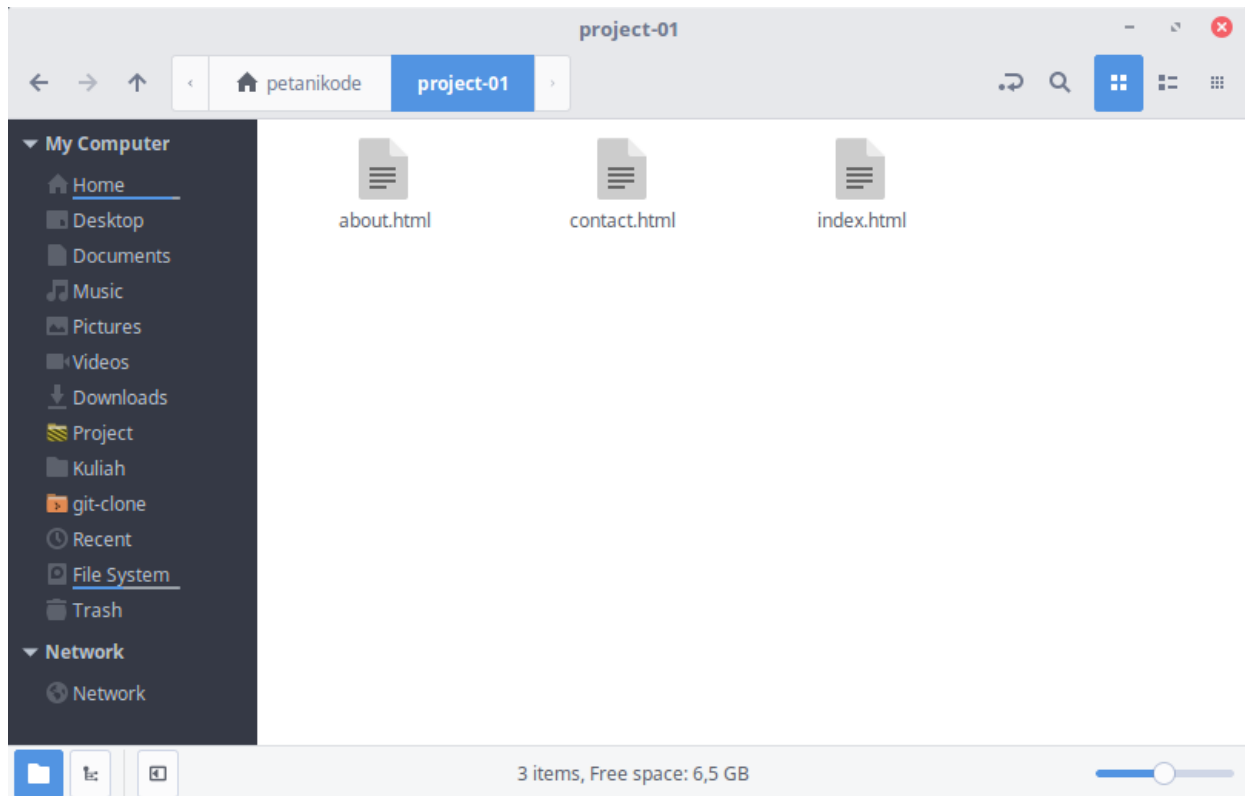
Contoh `.gitignore` pada proyek-proyek tertentu, bisa dilihat di [repositori ini](#).

## Tutorial Git #3: Simpan Perubahan Revisi dengan Git Commit

Pada [tutorial Git yang kedua](#), kita sudah membuat repositori kosong. Belum ada apa-apa di sana.

Sekarang coba tambahkan sebuah file baru.

Sebagai contoh, saya 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
```

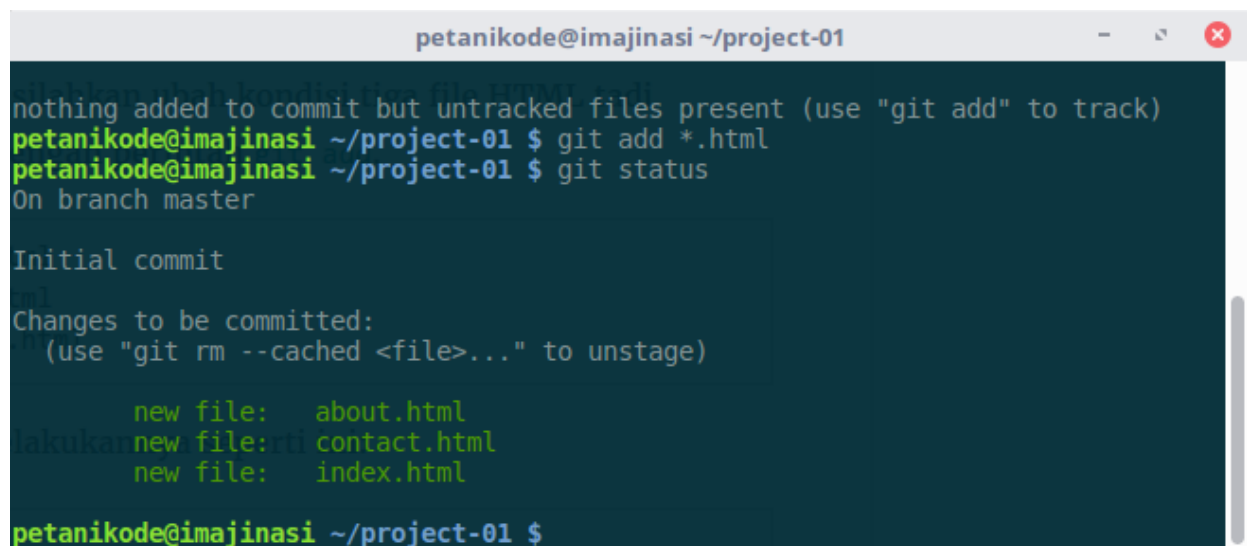
atau:

```
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*.

A terminal window titled 'petanikode@imajinasi ~/project-01' showing the output of 'git status'. The output indicates an initial commit with three new files staged for commit: about.html, contact.html, and index.html. The prompt is 'petanikode@imajinasi ~/project-01 \$'.

```
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 saya mengisinya seperti ini.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Belajar Git - Project 01</title>
```

```
</head>
```

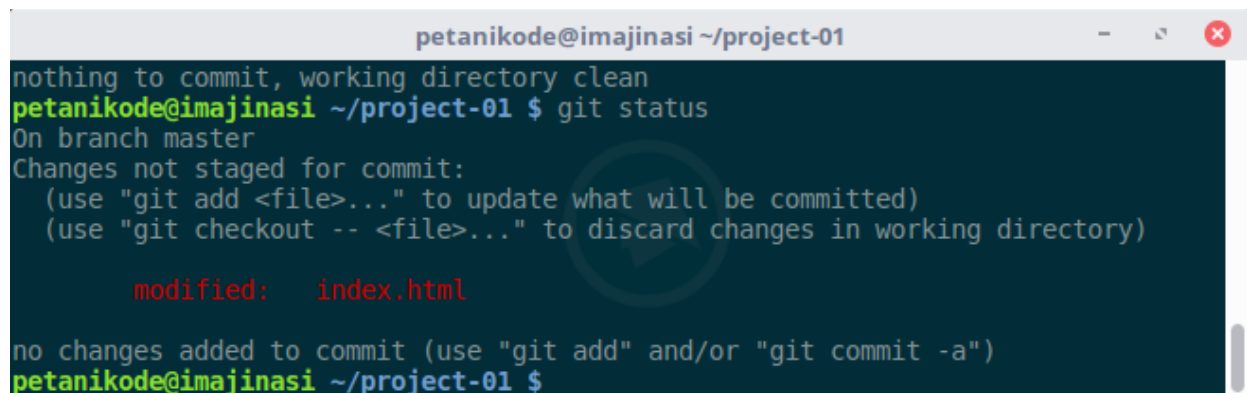
```
<body>
```

```
<p>Hello Semua, Saya sedang belajar Git</p>
```

```
</body>
```

```
</html>
```

Setelah itu ketik lagi perintah `git status`.

A terminal window titled 'petanikode@imajinasi ~/project-01' showing the output of the 'git status' command. The output indicates that the working directory is clean and on the 'master' branch, but there are changes not staged for commit. Specifically, 'index.html' is listed as 'modified'. The terminal also provides instructions on how to stage changes using 'git add' or discard them using 'git checkout --'.

```
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

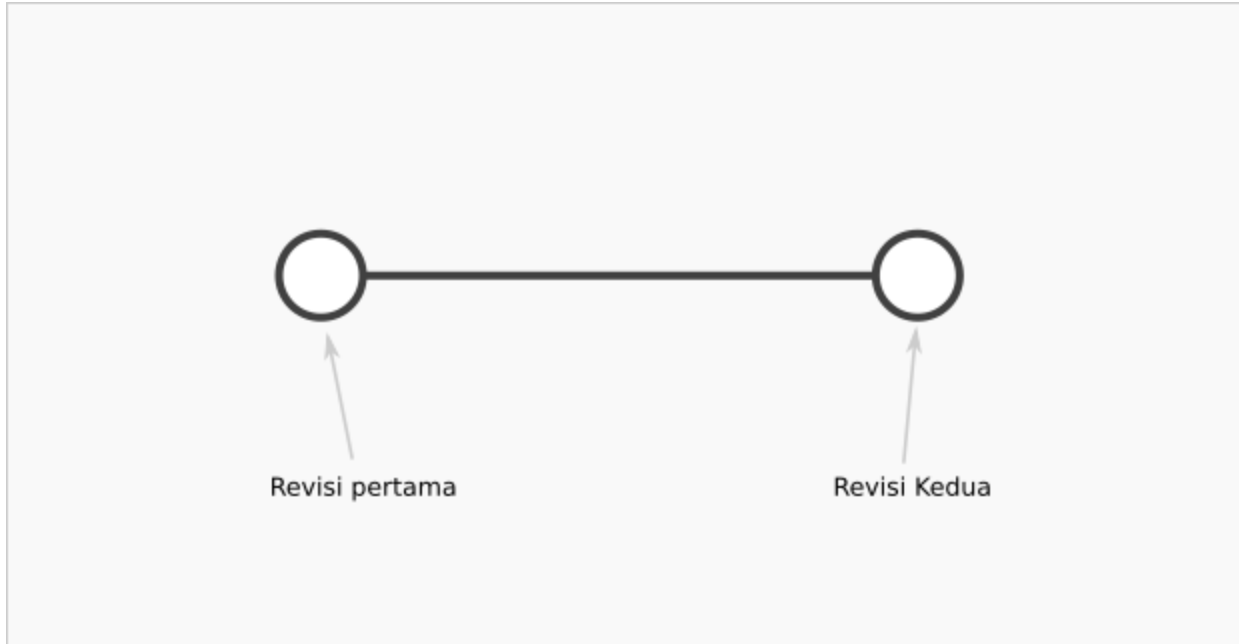
no changes added to commit (use "git add" and/or "git commit -a")
petanikode@imajinasi ~/project-01 $
```

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.


## Tutorial Git #4: Melihat Catatan Log Revisi

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.

A terminal window titled 'petanikode@imajinasi ~/project-01' showing the output of the 'git log' command. The output lists two commits: one with hash '06f735af7724558164c87f6b1ce3ca7778eb1c1b' and message 'ditambahkan isi', and another with hash 'cf08ca0837cf26f1c595be36bb3a6b815e311be1' and message 'commit pertama'. The terminal text is as follows:

```
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 dilakuan.

## 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 memasukan 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'
```

Itulah beberapa cara melihat log revisi pada repositori. Perintah yang digunakan adalah `git log`. Selanjutnya kita akan pelajari perintah `git diff` untuk melihat perbandingan pada revisi.

# Tutorial Git #5: Melihat Perbandingan Revisi dengan Git Diff

## 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.

```
petanikode@imajinasi ~/project-01
petanikode@imajinasi ~/project-01 $ git diff
diff --git a/index.html b/index.html
index 481cc02..c5082e6 100644
--- a/index.html
+++ b/index.html
@@ -5,6 +5,6 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Semua, Saya sedang belajar Git</p>
+     <p>Hello Dunia!, Saya sedang belajar Git</p>
   </body>
 </html>
petanikode@imajinasi ~/project-01 $
```

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>
```

Kita sudah pelajari fungsi dari perintah `git diff`. Perintah ini untuk melihat perbandingan perubahan apa saja yang telah dilakukan pada repositori. Selanjutnya, kita akan belajar membatalkan revisi.

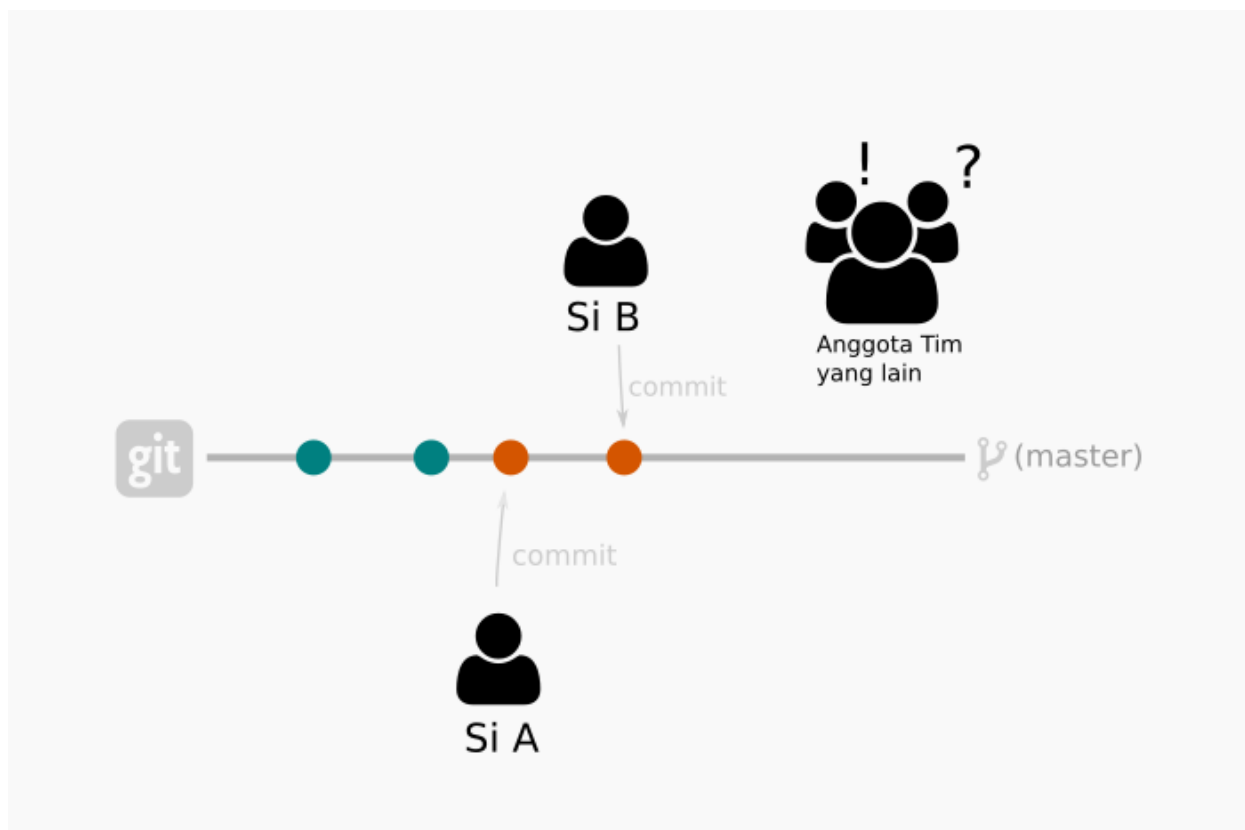
## Tutorial Git #6: Menggunakan Percabangan untuk Mencegah Konflik

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*).

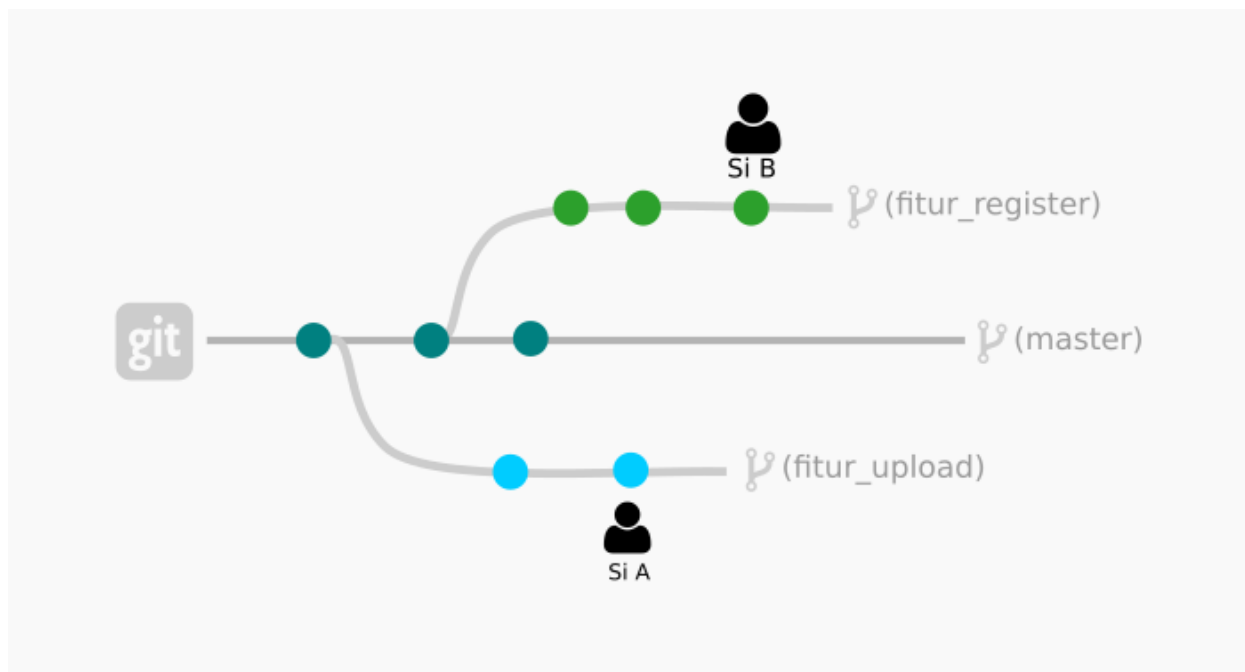
## Cara Membuat Cabang Baru

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"
```

Bagus! 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. Yep! 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).

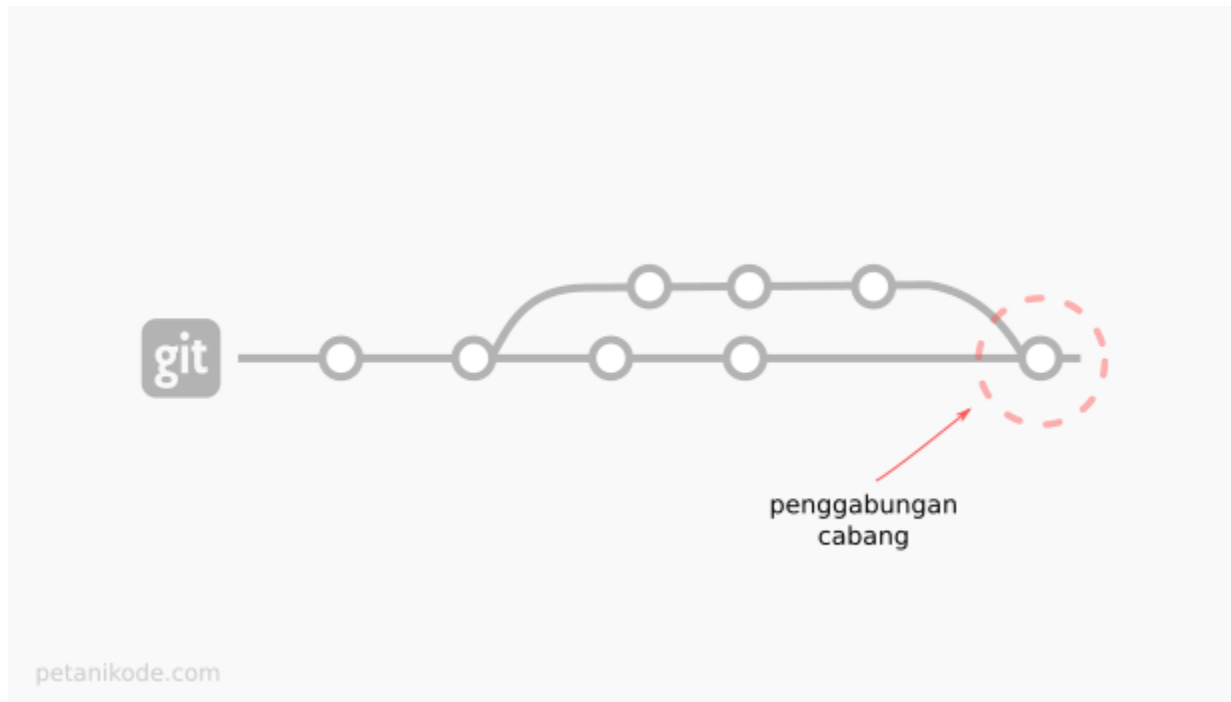
Pertama, kita harus pindah dulu ke cabang `master`.

```
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.

## Mengatasi Bentrok

Bentrok biasanya terjadi jika ada dua orang yang mengedit file yang sama.

Kenapa bisa begitu, 'kan 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.

Baiklah, sekarang kita akan coba membuat bentrokan 😊.

Pindah dulu ke branch `halaman_login...`

```
git checkout halaman_login
```

Setela 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.

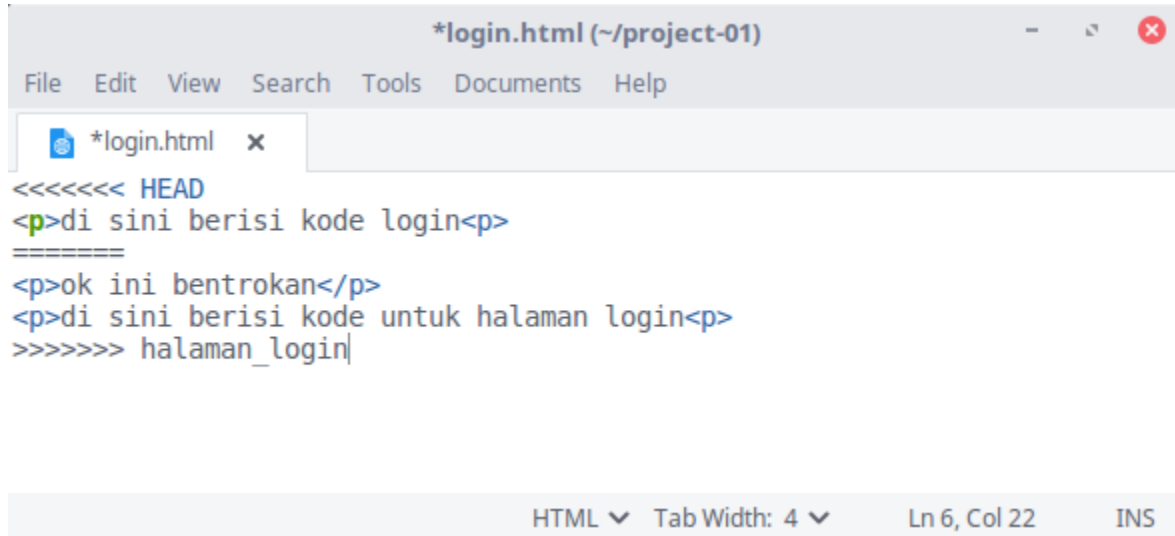
```
$ git merge halaman_login
```

```
Auto-merging login.html
```

```
CONFLICT (content): Merge conflict in login.html
```

```
Automatic merge failed; fix conflicts and then commit the
result.
```

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



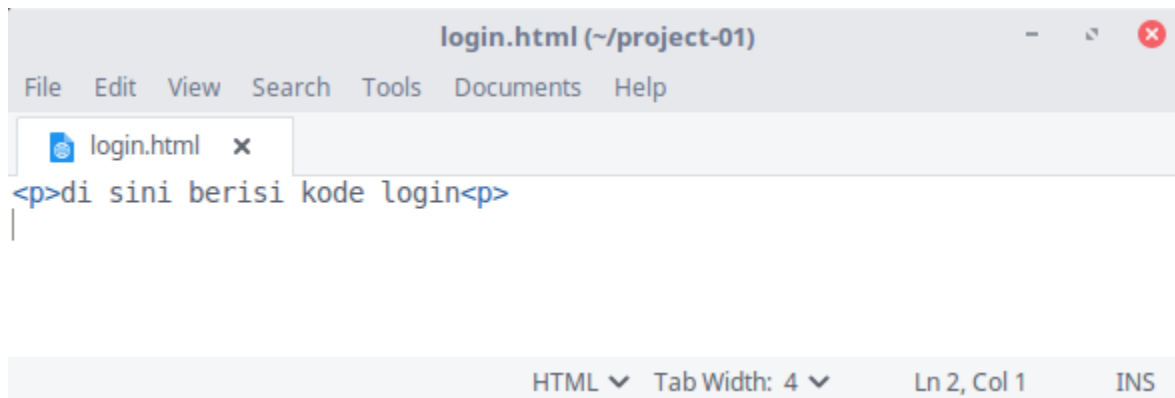
The screenshot shows a text editor window titled `*login.html (~/.project-01)`. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The tab bar shows `*login.html` with a close button. The editor content is as follows:

```
<<<<<<< HEAD
<p>di sini berisi kode login<p>
=====
<p>ok ini bentrokan</p>
<p>di sini berisi kode untuk halaman login<p>
>>>>>> halaman_login|
```

The status bar at the bottom indicates the file type is HTML, tab width is 4, the cursor is at line 6, column 22, and the mode is INS.

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

Silahkan eliminasi salah satu dari kode tersebut.



The screenshot shows the same text editor window after resolving the conflict. The title bar is now `login.html (~/.project-01)`. The tab bar shows `login.html`. The editor content is:

```
<p>di sini berisi kode login<p>
|
```

The status bar at the bottom indicates the file type is HTML, tab width is 4, the cursor is at line 2, column 1, and the mode is 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.

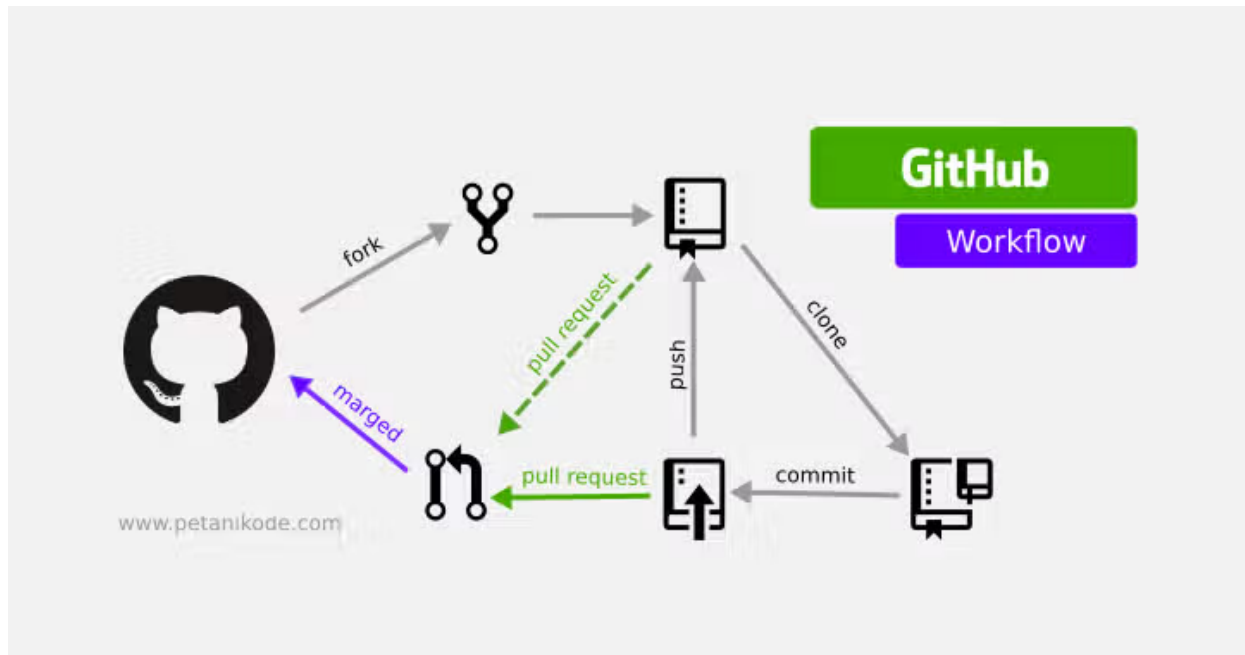
Agar repositori kita bersih dan rapi.

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

Contoh:

```
git branch -d halaman_login
```

## Github Workflow: Cara Berkontribusi di Proyek Open Source



Github saat ini sudah menjadi rumah bagi proyek-proyek open source.

Jutaan programmer dari seluruh dunia menaruh kodenya di sana.

Bahkan proyek open source seperti [Linux](#), [Cinnamon](#), [Laravel](#), [BlankOn](#), dll. menggunakan Github.

Bagaimana proyek-proyek itu digarap bersama?

Itulah yang akan kita pelajari pada artikel ini.

Kita akan belajar tentang Git dan Github *workflow* atau alur kerja Github untuk berkontribusi pada proyek open source.

Kita akan langsung melakukan kontribusi pada proyek “dummy” yang sudah saya siapkan.

Sebelumnya siapkan dulu alat-alat berikut:

1. [Akun Github](#);
2. [Git](#);
3. [Teks Editor](#).

## Langkah-langkah Berkontribusi pada Proyek Open Source di Github

Ada beberapa langkah yang harus dilakukan untuk berkontribusi di Proyek open source:

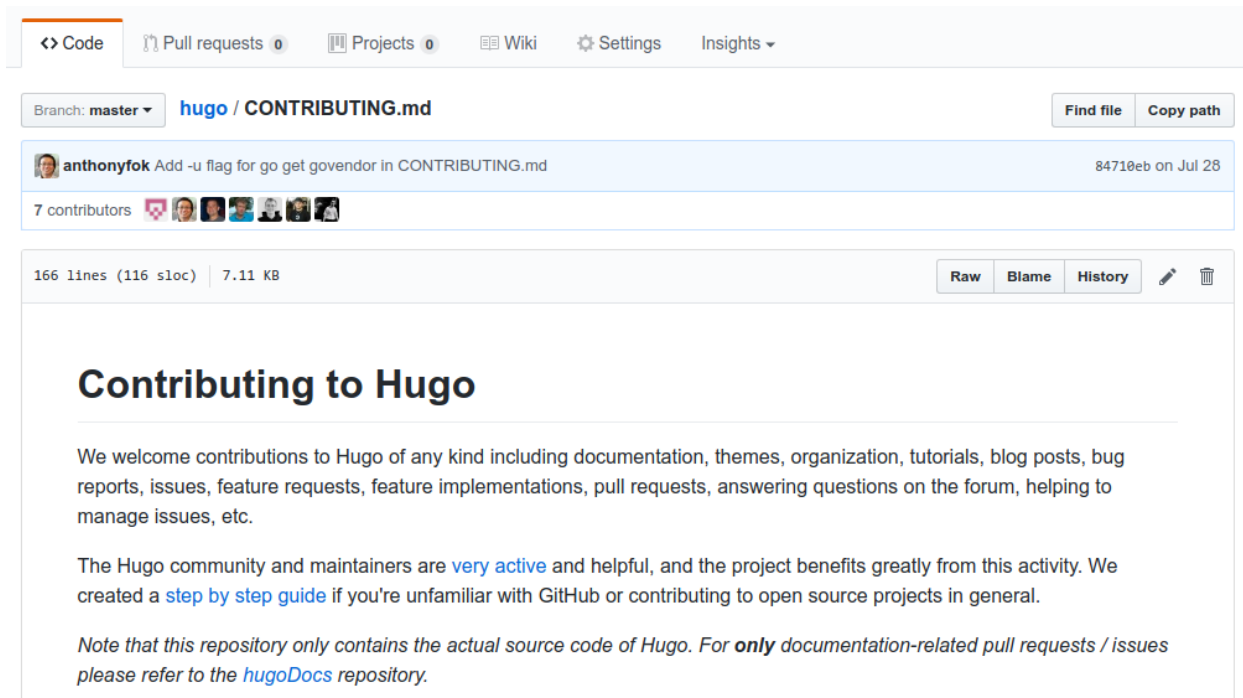
### 1. Baca Aturan Berkontribusi

Jika kamu tertarik berkontribusi pada sebuah proyek di Github, pastikan membaca aturan kontribusi.

Pada aturan kontribusi, biasanya akan ada larangan dan tata cara berkontribusi yang harus dipatuhi bersama.

Aturan berkontribusi biasanya ditulis pada file `CONTRIBUTING.md` dan `README.md`.

Contoh aturan berkontribusi pada [proyek Hugo](#):



Apa yang akan terjadi bila saya tidak mengikuti aturan berkontribusi?

Bisa jadi kontribusimu akan ditolak oleh admin.

Karena itu, bacalah aturan berkontribusi dengan teliti dan seksama.

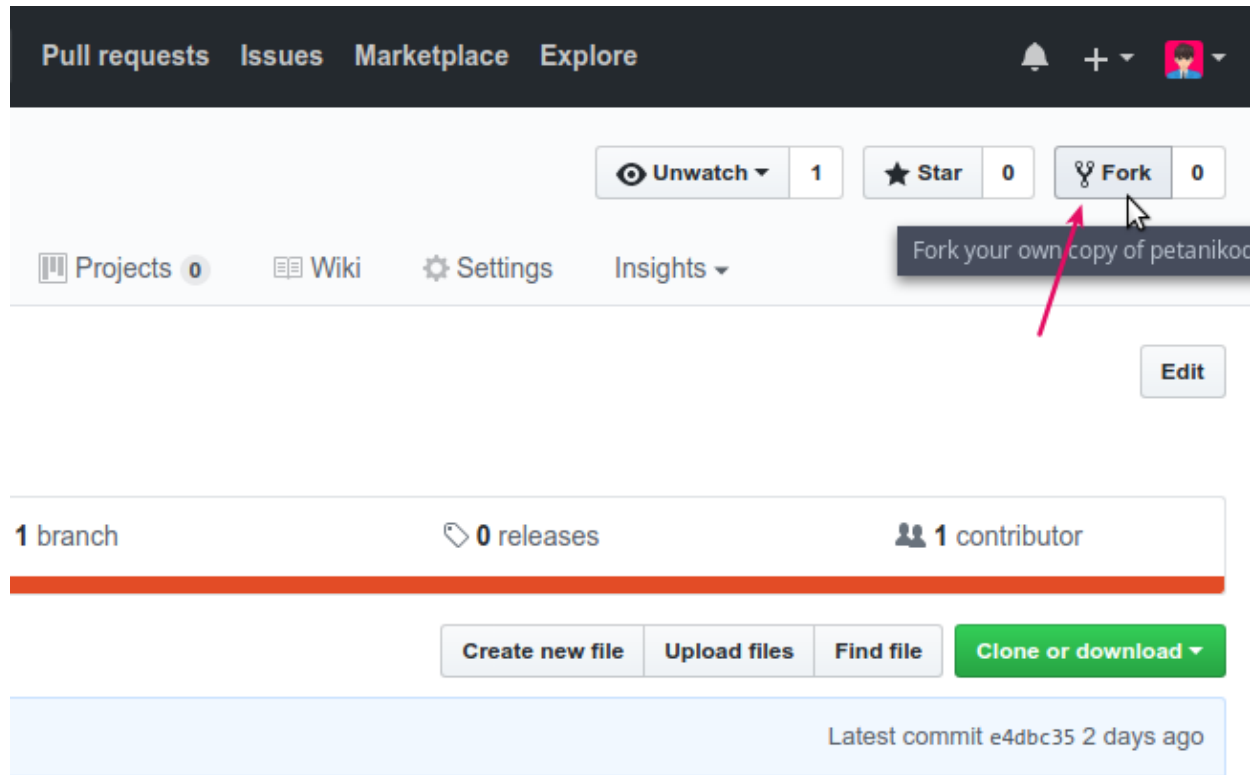
## 2. Fork & Clone Repository

Setelah kita selesai membaca aturan berkontribusi, langkah selanjutnya adalah melakukan fork repository ke akun kita.

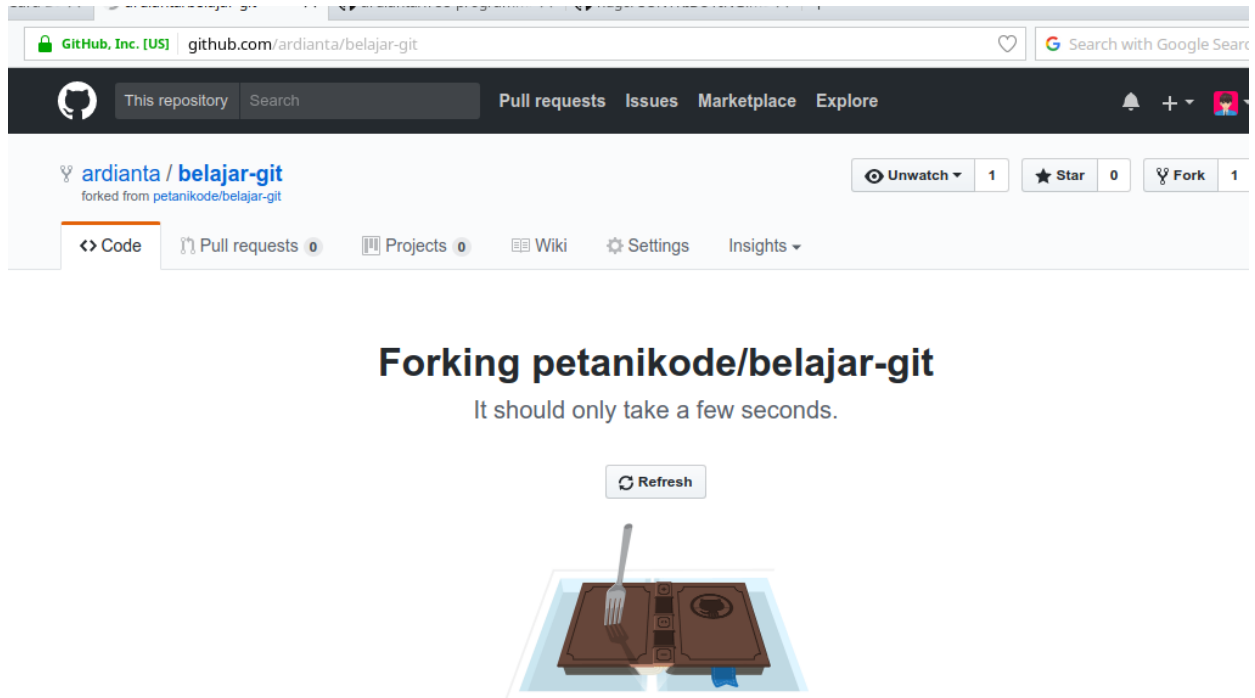
Fork bisa kita artikan sebagai menyalin repository dari akun orang lain atau organisasi ke akun kita sendiri.

Nah untuk memprektek, saya sudah menyiapkan proyek bernama [belajar-git](#).

Silahkan buka proyek tersebut, lalu klik tombol *Fork*.



Tunggu beberapa saat, repository sedang di-fork.



Setelah itu, akan ada repository baru bernama `belajar-git` di akun kita.

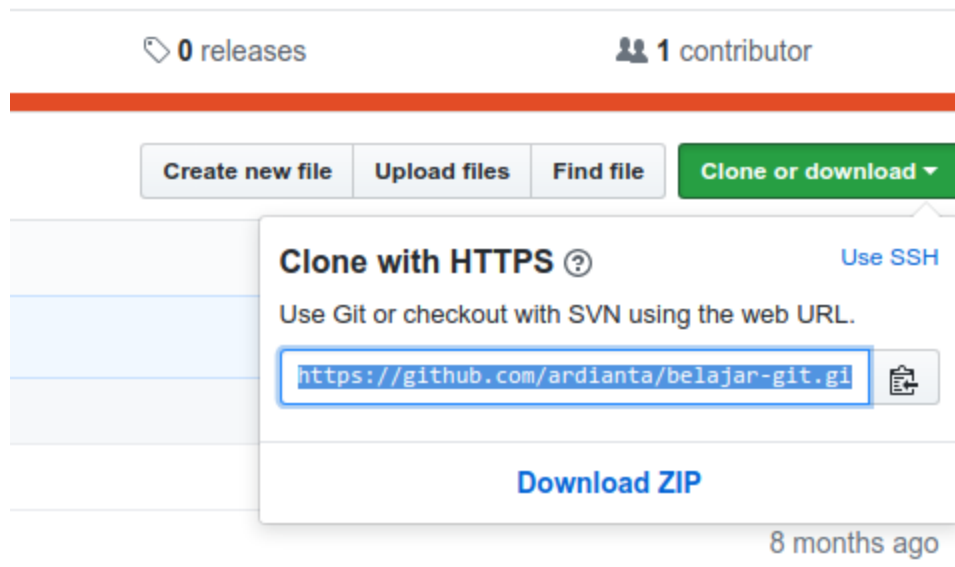


The screenshot shows the GitHub interface for a repository named 'belajar-git' by user 'ardianta'. The repository is a fork of 'petanikode/belajar-git'. The top navigation bar includes links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository header shows '8 commits', '1 branch', '0 releases', and '1 contributor'. Below this, there are buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The main content area displays a list of files and their commit history:

File	Commit Message	Time
README.md	membuat file README.md	2 days ago
about.html	commit pertama	8 months ago
contact.html	commit pertama	8 months ago
index.html	mengubah judul dan teks di body	2 days ago
login.html	ditambahkan login.html	7 months ago
register.html	Membuat file register.html	2 days ago

Sekarang repository `belajar-git` telah menjadi milik kita dan bebas melakukan apapun terhadapnya.

Selanjutnya silahkan *clone* (download) repository tersebut ke komputer lokal.



HTTPS:

```
git clone https://github.com/ardianta/belajar-git.git
```

SSH:

```
git clone git@github.com:ardianta/belajar-git.git
```

Saya biasanya [menggunakan SSH](#), biar tidak memasukkan password saat melakukan *push*.

```
petanikode@imajinasi ~/tmp
petanikode@imajinasi ~ $ cd tmp/
petanikode@imajinasi ~/tmp $ git clone git@github.com:ardianta/belajar-git.git
Cloning into 'belajar-git'...
remote: Counting objects: 22, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 22 (delta 9), reused 22 (delta 9), pack-reused 0
Receiving objects: 100% (22/22), done.
Resolving deltas: 100% (9/9), done.
Checking connectivity... done.
petanikode@imajinasi ~/tmp $
```

### 3. Lakukan Modifikasi

Setelah kita melakukan *clone* ke komputer lokal, selanjutnya silahkan buka dengan teks editor dan lakukan modifikasi.

Sebagai contoh, saya membukanya dengan [VS Code](#).

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Belajar Git: Remote Repository</title>
6   </head>
7   <body>
8     <p>Hello Semua, Saya sedang belajar Git</p>
9
10    <p>Yay! saya sedang belajar tentang remote repository</p>
11  </body>
12 </html>
13
```

Buatlah beberapa perubahan dan simpan perubahan yang kamu lakukan dengan Git.

Lakukan *commit* terhadap apa yang kamu rubah.

```
petanikode@imajinasi ~/tmp/belajar-git
petanikode@imajinasi ~/tmp $ cd belajar-git/
petanikode@imajinasi ~/tmp/belajar-git $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        CONTRIBUTING.md
        kontributor/

nothing added to commit but untracked files present (use "git add" to track)
petanikode@imajinasi ~/tmp/belajar-git $ git add CONTRIBUTING.md
petanikode@imajinasi ~/tmp/belajar-git $ git commit -m "ditambahkan aturan berkontribusi"
[master 7527022] ditambahkan aturan berkontribusi
1 file changed, 12 insertions(+)
create mode 100644 CONTRIBUTING.md
petanikode@imajinasi ~/tmp/belajar-git $ git add kontributor/ardianta.txt
petanikode@imajinasi ~/tmp/belajar-git $ git commit -m "kontribusi ardianta"
[master 652df64] kontribusi ardianta
1 file changed, 5 insertions(+)
create mode 100644 kontributor/ardianta.txt
petanikode@imajinasi ~/tmp/belajar-git $
```

Perlu diperhatikan juga:

- Gunakan pesan *commit* yang informatif dan mewakili apa yang sudah kamu ubah.
- Hindari menggunakan `git add .` untuk melakukan *commit* ke semua file.

## 4. Push Kontribusimu

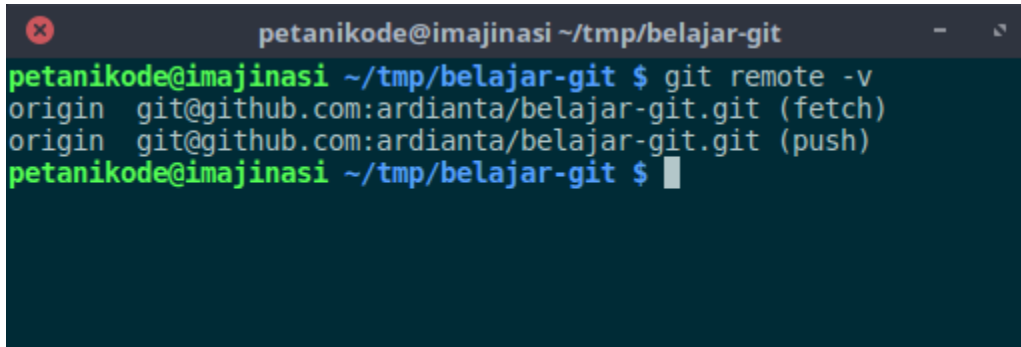
Setelah kita selesai melakukan perubahan dan *commit*, langkah berikutnya adalah melakukan *push*.

Push-nya ke mana?

Ya ke repository hasil fork tadi.

Repository yang kita *clone* dari Github, akan otomatis [membuat remote](#) bernama origin.

Untuk melihatnya, gunakan perintah `git remote -v`.

A terminal window titled 'petanikode@imajinasi ~/tmp/belajar-git' showing the output of the 'git remote -v' command. The output lists the 'origin' remote with its fetch and push URLs. The prompt is at the end of the last line.

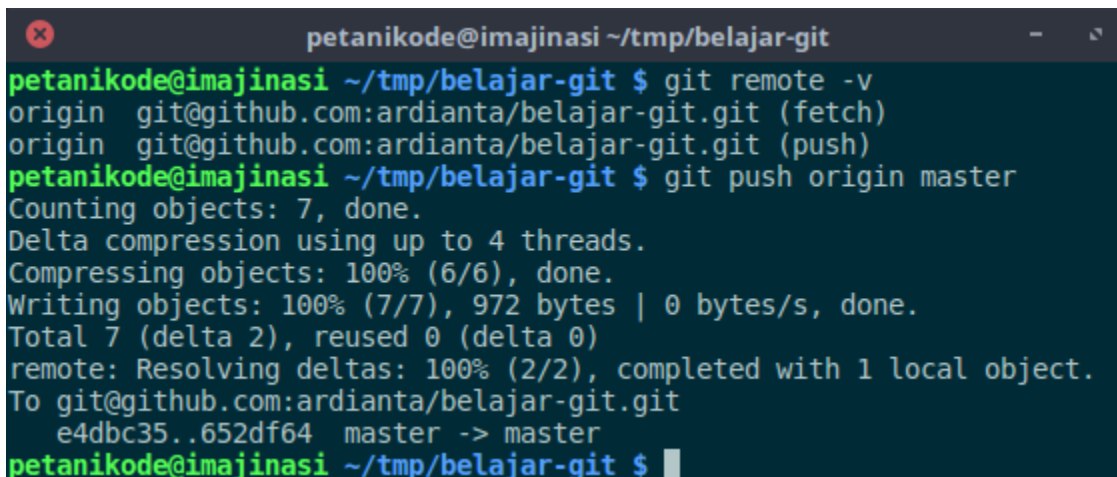
```
petanikode@imajinasi ~/tmp/belajar-git $ git remote -v
origin  git@github.com:ardianta/belajar-git.git (fetch)
origin  git@github.com:ardianta/belajar-git.git (push)
petanikode@imajinasi ~/tmp/belajar-git $
```

Alamat tujuan *push* dan *fetch* mengarah ke alamat repository di akun kita.

Silahkan melakukan *push* dengan perintah berikut.

```
git push origin master
```

Tunggulah beberapa saat...

A terminal window titled 'petanikode@imajinasi ~/tmp/belajar-git' showing the output of the 'git push origin master' command. The output shows the progress of pushing the master branch to the origin repository, including object counting, compression, and writing details. The prompt is at the end of the last line.

```
petanikode@imajinasi ~/tmp/belajar-git $ git remote -v
origin  git@github.com:ardianta/belajar-git.git (fetch)
origin  git@github.com:ardianta/belajar-git.git (push)
petanikode@imajinasi ~/tmp/belajar-git $ git push origin master
Counting objects: 7, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 972 bytes | 0 bytes/s, done.
Total 7 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To git@github.com:ardianta/belajar-git
   e4dbc35..652df64  master -> master
petanikode@imajinasi ~/tmp/belajar-git $
```

Setelah selesai, coba lihat repository `belajar-git` di akun Github-mu.

Apakah berhasil di-push atau tidak?

Kalau berhasil, silahkan lanjutkan ke langkah berikutnya.

## 5. Membuat Pull Request

*Pull Request* adalah istilah yang bisa kita artikan sebagai permintaan untuk menggabungkan kode.

Kita sudah membuat perubahan di repository hasil fork, lalu ingin menggabungkan dengan repository sumber.

Maka kita harus membuat *Pull Request*.

Silahkan klik tombol *New Pull Request* pada repository `belajar-git`.

Belajar Git Edit

[Add topics](#)

10 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

This branch is 2 commits ahead of petanikode:master. Pull request Compare

ardianta kontribusi ardianta		Latest commit 652df64 12 minutes ago
kontributor	kontribusi ardianta	11 minutes ago
CONTRIBUTING.md	ditambahkan aturan berkontribusi	12 minutes ago
README.md	membuat file README.md	2 days ago
about.html	commit pertama	8 months ago
contact.html	commit pertama	8 months ago
index.html	mengubah judul dan teks di body	2 days ago
login.html	ditambahkan login.html	7 months ago
register.html	Membuat file register.html	2 days ago

Setelah itu, Github akan melakukan komparasi.

Apakah ada kode yang bentrok atau tidak?

Kalau tidak ada yang bentrok biasanya akan muncul tulisan hijau “*Able to merge*”.

Selanjutnya silahkan klik tombol *Create Pull Request*.



petanikode / belajar-git

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

base fork: petanikode/belajar-git base: master ... head fork: ardianta/belajar-git compare: master ✓ Able to merge. These branches can be merged.

Create pull request Discuss and review the changes in this comparison with others.

2 commits 2 files changed 0 commit comments

Commits on Oct 03, 2017

- ardianta ditambahkan aturan berkontribusi
- ardianta kontribusi ardianta

Showing 2 changed files with 17 additions and 0 deletions.

Silahkan isi judul *Pull Request* dan pesan yang ingin disampaikan ke komunitas.

base fork: petanikode/belajar-git base: master ... head fork: ardianta/belajar-git compare: master ✓ Able to merge. These branches can be merged.

Kontribusi Belajar Git

Write Preview

Menambahkan aturan kontribusi di file `CONTRIBUTING.md` dan mencoba membuat sebuah kontribusi.

Attach files by dragging & dropping or selecting them.

☒ Allow edits from maintainers. [Learn more](#)

Create pull request

Reviewers  
No reviews—request one

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Setelah itu admin atau owner akan melakukan review kontribusimu.

Biasanya akan terjadi diskusi untuk membahas *pull request* yang telah kita buat.


Apakah akan ditolak atau diterima?

Kalau diterima, biasanya akan ada tulisan “*Marged*” berwarna ungu.

## Added 'Dasar Pemrograman Golang' and Go section #2552


**Merged** eshellman merged 1 commit into EbookFoundation:master from ardianta:master 22 hours ago

Conversation 1 Commits 1 Files changed 1




ardianta commented a day ago

Added *Dasar Pemrograman Golang* written by Noval Agung Prayogo and made Go section




Added 'Dasar Pemrograman Golang' and Go section

1e06c1a



eshellman commented 22 hours ago

Thank you @ardianta ! I'm accepting this PR as-is, but I'd encourage you to add author names where appropriate in another PR.



eshellman merged commit 350d211 into EbookFoundation:master

22 hours ago  
1 check passed

[View details](#) [Revert](#)

**Reviewers**  
No reviews

**Assignees**  
No one assigned

**Labels**  
None yet

**Projects**  
None yet

**Milestone**  
No milestone

Source : [Petani Kode](#)