

# Tutorial GitHub dan Git

**STEP BY STEP**

YOHANES DWI LISTIO

## Daftar Isi

|   |    |
|---|----|
| Pendahuluan .....   | 1  |
| Apakah itu GitHub? .....  | 1  |
| Fungsi dan Manfaat GitHub .....                                 | 1  |
| Persiapan.....  | 3  |
| Membuat akun GitHub.....  | 3  |
| Membuat Repositori GitHub .....                                 | 3  |
| Menambahkan <i>Collaborator</i> ke dalam Repositori GitHub..... | 5  |
| Instalasi Git.....  | 6  |
| Instalasi Git.....  | 6  |
| Konfigurasi awal Git .....                                      | 10 |
| Tutorial Dasar Git.....   | 12 |
| Membuat Repositori Baru dalam Proyek .....                      | 12 |
| Membuat Revisi dalam Repositori.....                            | 13 |
| Kelompok Kondisi File dalam Git.....                            | 13 |
| Revisi Pertama.....   | 14 |
| Melihat Catatan Log Revisi .....                                | 15 |
| Membandingkan Revisi.....                                       | 16 |
| Membatalkan Revisi.....   | 17 |
| Git Checkout.....   | 17 |
| Git Reset .....   | 18 |
| Membatalkan Perubahan File dalam Kondisi <i>Committed</i> ..... | 18 |
| Kembali ke Beberapa Commit Sebelumnya .....                     | 19 |
| Maju ke Beberapa Commit Sesudahnya.....                         | 19 |
| Membatalkan Semua Perubahan yang Ada (Git Revert).....          | 20 |
| Mengambil Revisi dari Repositori.....                           | 20 |
| Mengambil Revisi dengan Git Fetch.....                          | 21 |
| Mengambil Revisi dengan Git Pull .....                          | 21 |
| Clone Repositori Remote .....                                   | 21 |
| Percabangan untuk Mencegah Konflik .....                        | 22 |
| Membuat Cabang Baru .....                                       | 22 |
| Menggabungkan Cabang .....                                      | 23 |
| Mengatasi Bentrok.....  | 23 |

|                        |    |
|------------------------|----|
| Menghapus Cabang ..... | 24 |
| Apa Selanjutnya? ..... | 25 |
| Daftar Rujukan.....    | 26 |

# Pendahuluan

## Apakah itu GitHub?

Banyak yang mengatakan bahwa GitHub adalah media sosial bagi para *programmer*. Pertanyaannya, seperti apa GitHub ini? Apakah seperti Facebook? Ataukah seperti Twitter di mana kita bisa melakukan *mention*? Atau bahkan seperti Instagram yang bisa mengunggah gambar dan video?

Banyak juga yang mengatakan bahwa GitHub merupakan *tools* untuk revisi kode, mengatur versi dan membenarkan *bug* yang ada pada *source code* Anda.

Tentu saja bagi Anda hal-hal ini sulit dimengerti tanpa penjelasan berupa contoh nyata untuk mempermudah pemahaman. Jangan khawatir, bagian ini akan menjelaskan tentang apa itu GitHub dan fungsinya.

GitHub adalah aplikasi berbasis *website* yang memberikan layanan berupa penyimpanan repositori secara gratis. Repositori sendiri adalah tempat dimana Anda dapat menyimpan file-file berupa *source code*. Jika masih sulit membayangkan, coba ingat, pernahkah Anda membuat *folder* di Google Drive, kemudian Anda isi dengan file-file pekerjaan/kuliah Anda atau mungkin *software*? Nah, repositori pada GitHub itu semacam Anda menyimpan file-file di Google Drive tadi.

GitHub juga merupakan *Version Control System* (VCS) yang paling populer. Perusahaan-perusahaan besar seperti Facebook, Twitter, dan Google saja menggunakan GitHub untuk kolaborasi dengan ratusan *programmer*-nya dalam pengembangan aplikasi mereka.

## Fungsi dan Manfaat GitHub

GitHub berfungsi untuk menyimpan repositori yang telah dijelaskan di atas dan juga sebagai alat kolaborasi dalam pengerjaan suatu proyek. Lebih jelasnya, Anda mendapatkan proyek pembuatan *website company profile*. Untuk mengerjakan proyek itu, Anda mengajak seorang teman untuk mengerjakan bagian *backend*-nya (PHP, MySQL dan sebagainya). Agar pengerjaan dapat selesai dengan mudah, Anda memutuskan untuk menggunakan GitHub. Jadi meskipun tidak mengerjakan di tempat yang sama atau bertemu langsung, proyek Anda tetap dapat selesai dan dapat dikerjakan oleh Anda berdua. Itulah yang dimaksud sebagai alat kolaborasi.

Selain itu, ada beberapa hal yang bisa kita lakukan di GitHub :

1. Kita bisa mengikuti (*follow*) *programmer* lain. Jadi, apa yang dilakukan oleh *programmer* yang Anda ikuti bisa Anda ketahui.
2. Star, fungsinya sama dengan *bookmark*.
3. Watch, mengawasi repositori tertentu. Sehingga ketika ada perubahan, Anda bisa mendapatkan notifikasi.
4. Fork, ini mirip dengan *copy-paste*. Ketika Anda menemukan *source code programmer* lain, Anda bisa melakukan *fork* sehingga nantinya tidak perlu lagi susah-susah mencari *source code* tersebut. Cukup buka di repositori Anda saja.

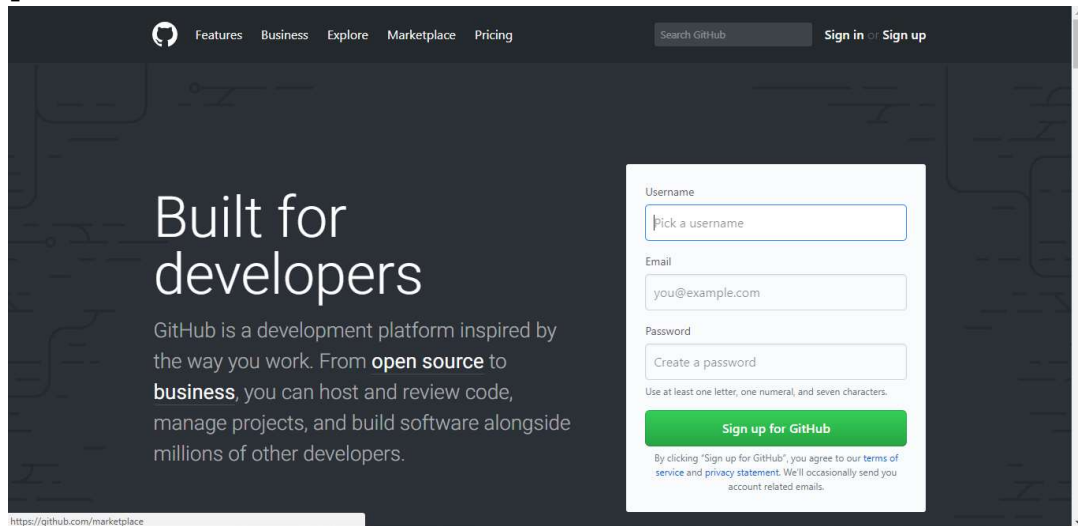
Bagusnya lagi, GitHub bisa menjadi sarana bagi Anda mendapatkan pekerjaan. Ada banyak *programmer* yang mengajak *programmer* lain yang juga menggunakan GitHub untuk bekerjasama dalam pengerjaan suatu proyek. Hal ini bisa terjadi karena GitHub memiliki laman profil yang berisi

data pribadi Anda seperti *e-mail*, foto, jumlah repositori yang dimiliki dan jumlah pengikut (*followers*). Makin banyak pengikutnya, biasanya orang tersebut jago dalam *coding* dan banyak memberikan manfaat bagi programmer lain dan itu juga merupakan salah satu pertimbangan seseorang ketika hendak merekrut *programmer* lain untuk mengerjakan suatu proyek.

## Persiapan

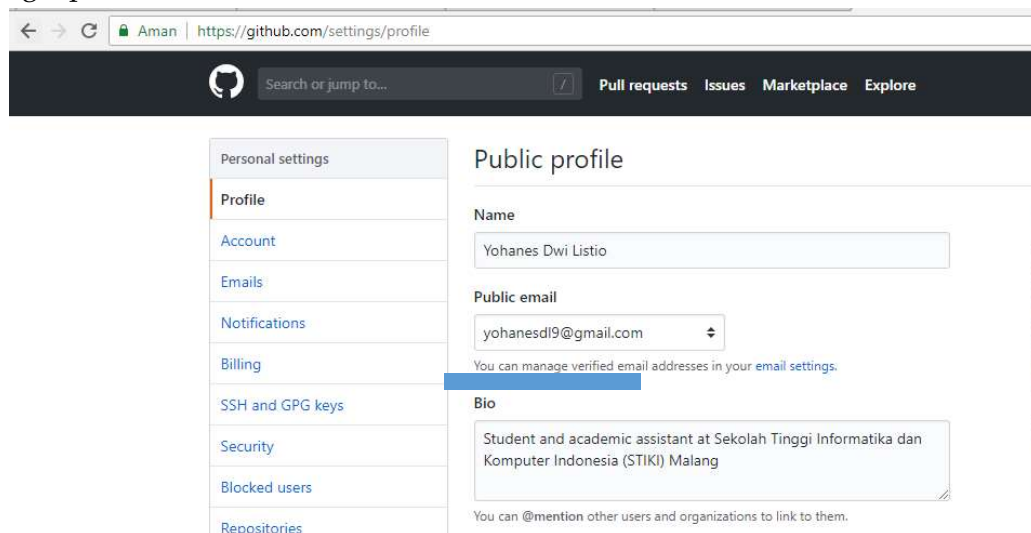
### Membuat akun GitHub

1. Akses [github.com](https://github.com). Masukkan username, email, dan password pada *field* yang disediakan. Klik **Sign up for GitHub**.



*Membuat akun GitHub*

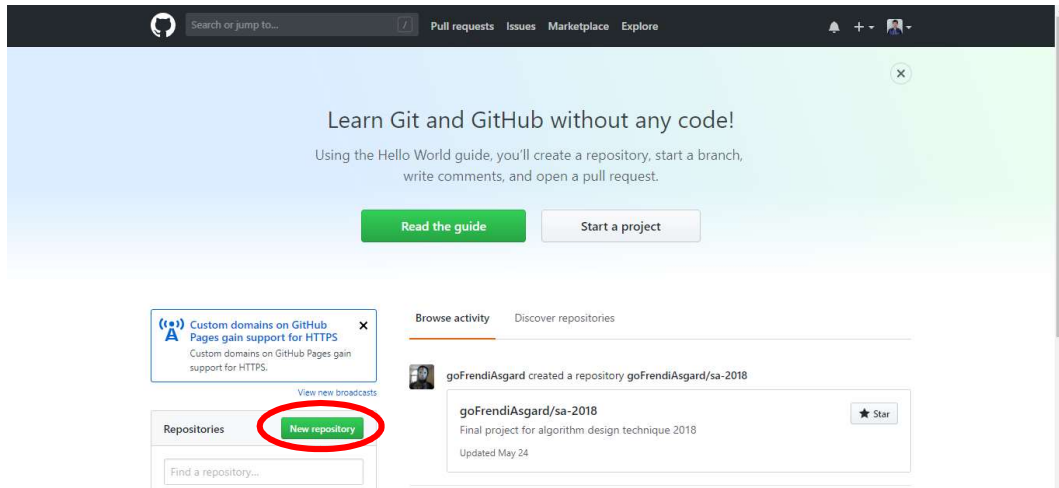
2. Pilih opsi **Unlimited public repositories for free**.
3. Kamu akan menerima *e-mail* verifikasi pada *e-mail* kamu.
4. Klik pada tautan yang dikirim melalui email untuk menyelesaikan proses verifikasi.
5. Masuk ke akun GitHub yang sudah kamu buat. Jangan lupa masuk ke Settings dan masukkan nama lengkapmu.



*Menambahkan nama lengkap pada akun GitHub*

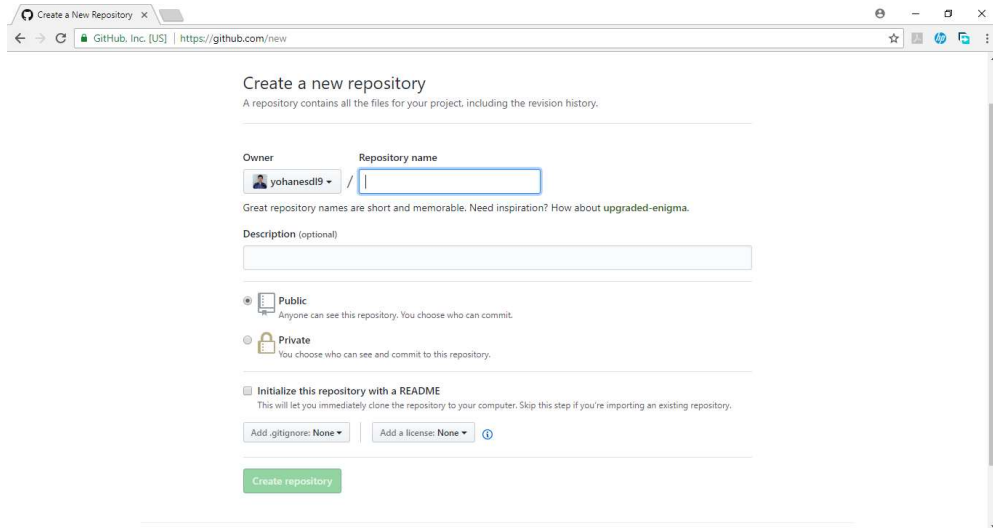
### Membuat Repositori GitHub

1. Pastikan kamu sudah masuk ke GitHub dengan akun yang sudah kamu buat.
2. Klik pada tombol **New repository**



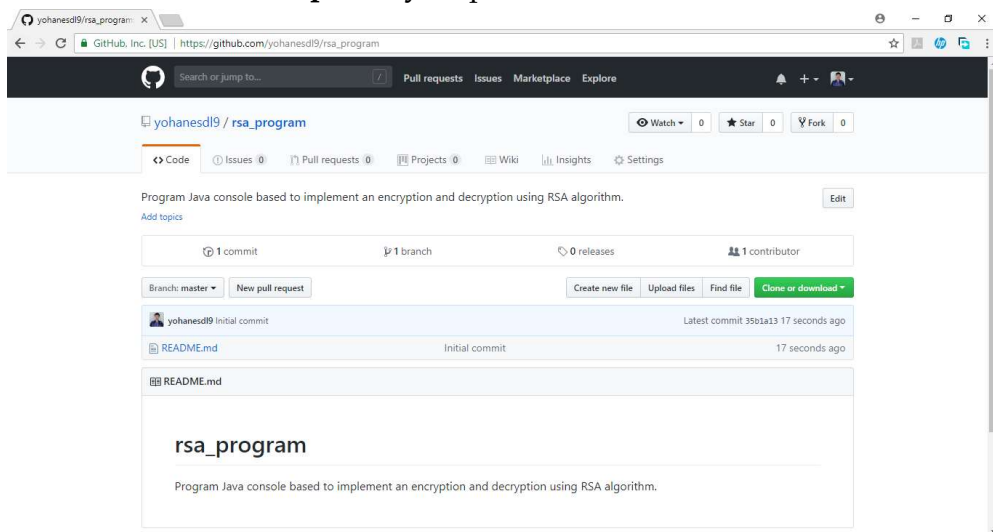
*Tampilan home page GitHub*

3. Masukkan nama repositori sesuai judul project kamu. Kamu dapat menjelaskan project apa yang dibuat pada bagian Description. Jangan lupa centang pada pilihan **Initialize this repository with a README**.



*Membuat repositori baru di GitHub*

4. Jika sudah selesai, klik **Create Repository**. Repositori kamu berhasil dibuat!

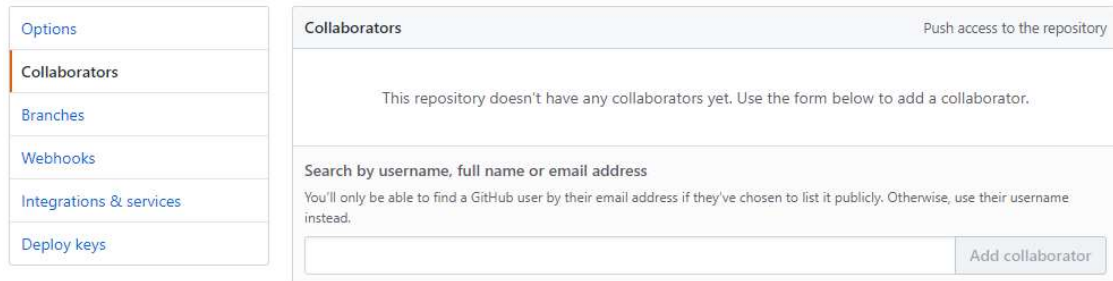


*Tampilan awal repositori baru di GitHub*

**KETERANGAN :** Untuk project secara berkelompok cukup membuat satu repositori di akun salah satu anggota kelompok. Anggota kelompok lainnya akan ditambahkan sebagai *collaborator*, yang akan dibahas sesudah ini.

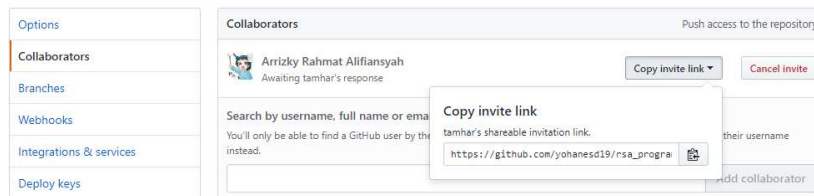
## Menambahkan *Collaborator* ke dalam Repositori GitHub

1. Klik tab **Settings** pada tampilan repositori GitHub kamu, lalu klik pada **Collaborators** di menu sebelah kiri. Akan muncul tampilan seperti berikut.



*Tampilan pengaturan repository GitHub*

2. Tambahkan teman kamu sebagai *collaborator* dengan mengetikkan username atau alamat email dari akun GitHub teman kamu. Klik **Add collaborator**.



*Tampilan Add Collaborators di repositori GitHub*

3. Akan muncul akun teman kamu sebagai *collaborator*. Mintalah temanmu untuk login di akun GitHub-nya dan memasukkan *invite link* tersebut.



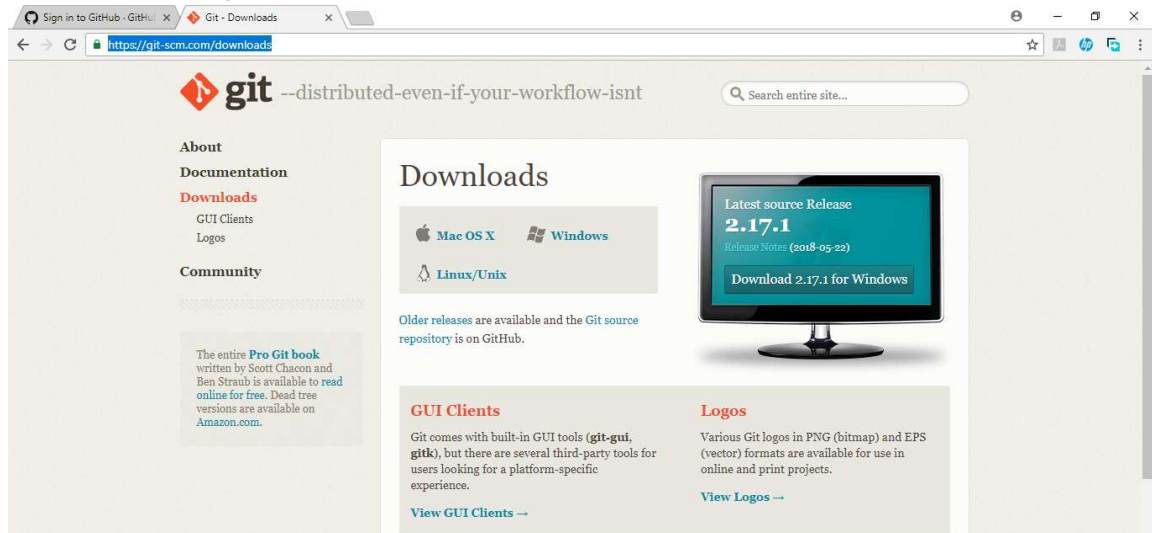
## Instalasi Git

Kita sudah belajar bagaimana membuat akun GitHub, membuat repositori dan menambahkan teman sebagai *collaborator* di repositori GitHub kita. Pada bagian ini kita memerlukan perantara untuk memasukkan *file* ke repositori GitHub kita dan juga mengambil *file* dari repositori GitHub kita, yaitu Git.

Proses instalasi dan konfigurasi awal Git akan dijelaskan sebagai berikut :

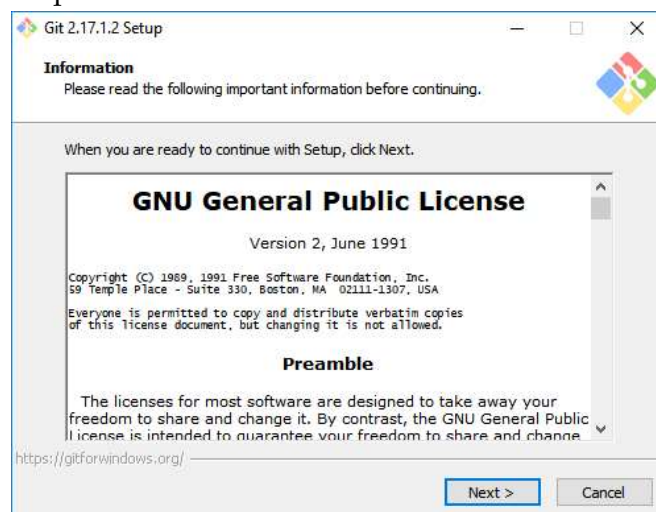
## Instalasi Git

1. Akses situs <https://git-scm.com/downloads> . Pilih **Download 2.17.1 for Windows**.



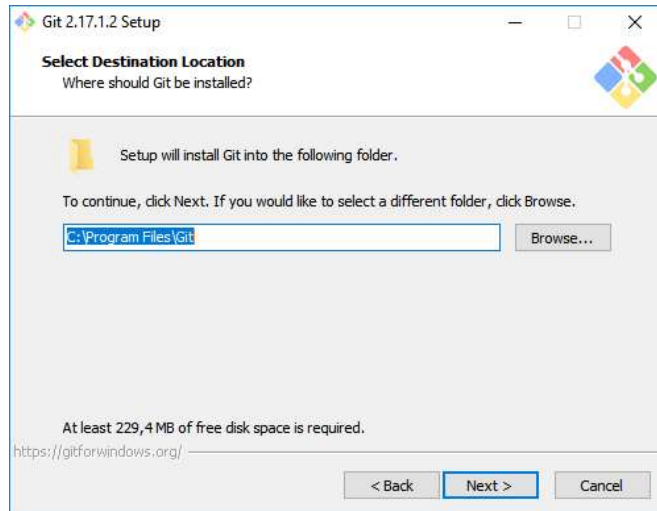
*Situs web resmi Git*

2. Silahkan tunggu installer Git di-*download* di komputer kamu.
3. Jika sudah selesai, buka folder di mana kamu men-*download* installer Git. Klik dua kali. Jika muncul pertanyaan **Do you want to allow this app to make changes to your device?** maka pilih Yes.
4. Akan muncul tampilan seperti berikut. Klik Next.



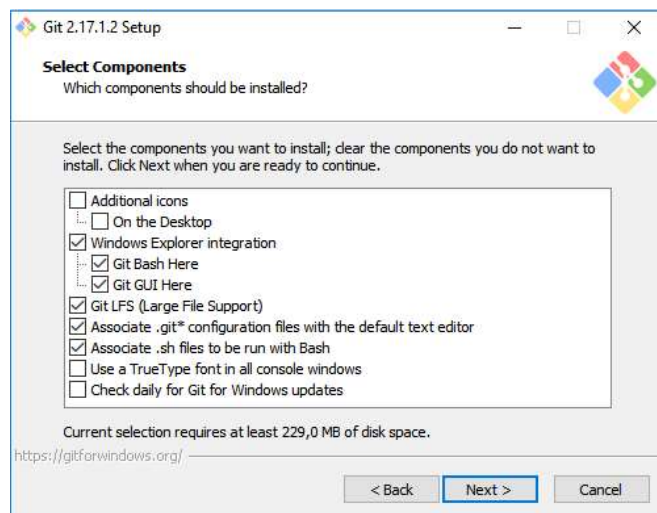
*Tampilan informasi lisensi Git*

5. Klik Next. Bagian ini untuk menentukan lokasi instalasi Git.



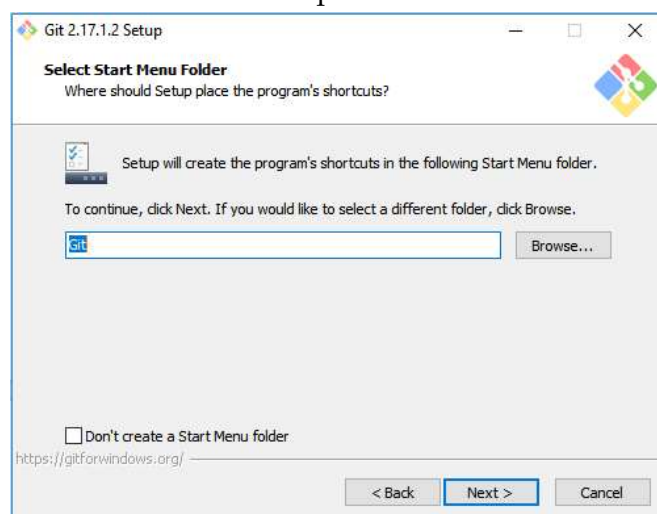
*Menentukan lokasi instalasi Git*

6. Centang pada pilihan Additional icons. Ini akan otomatis menambahkan *icon* Git ke Desktop (opsional). Klik Next.



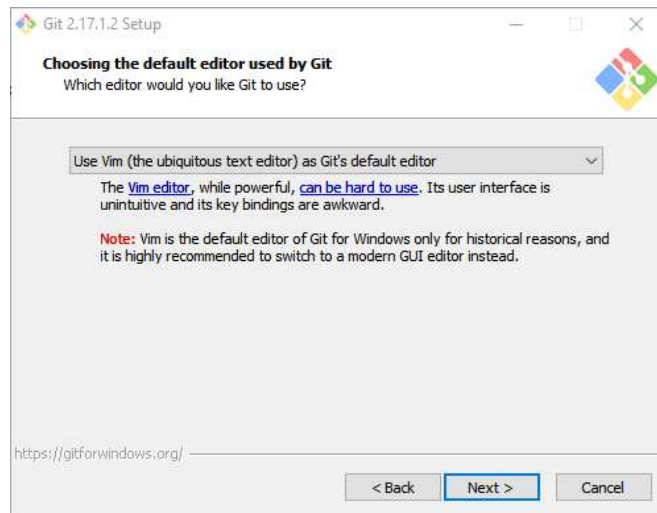
*Memilih komponen Git yang akan diinstal*

7. Klik Next. Bagian ini untuk memilih direktori pada Start Menu



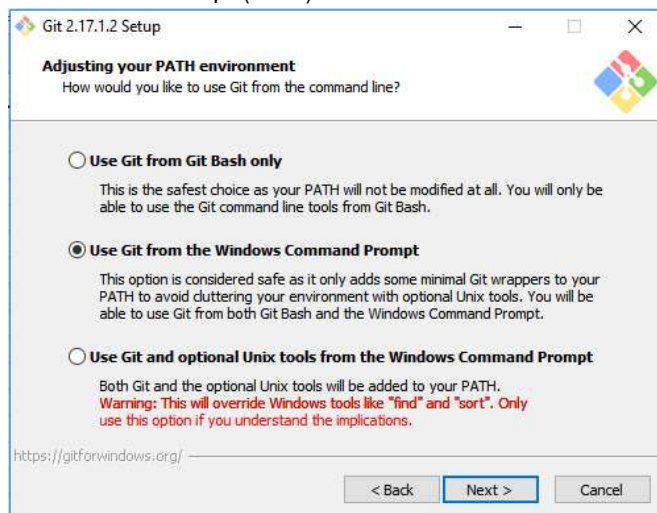
*Memilih komponen Git yang akan diinstal*

8. Klik Next. Bagian ini untuk memilih *editor* default yang akan digunakan oleh Git.



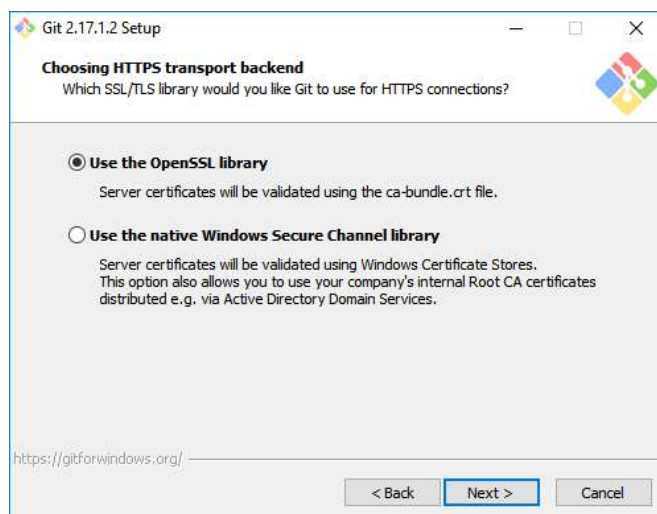
*Memilih editor default yang akan digunakan Git*

9. Selanjutnya pengaturan *PATH Environment*. Pilih *Use Git from the Windows Command Prompt* agar perintah git dapat di kenali di *Command Prompt* (CMD). Klik Next.



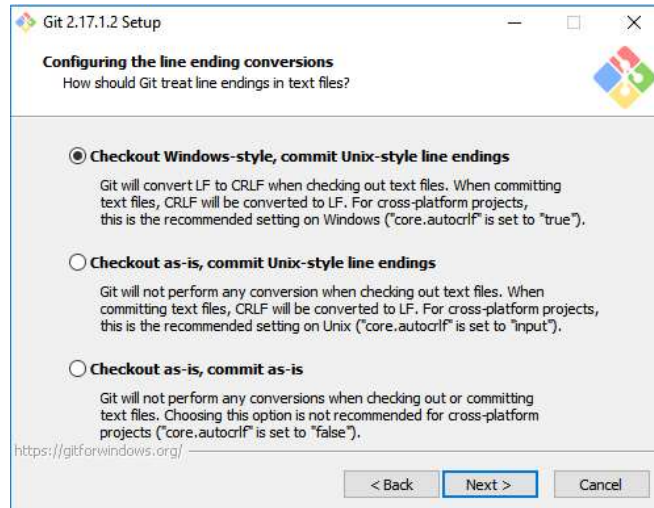
*Memilih PATH Enviroment untuk Git*

10. Klik Next. Bagian ini untuk memilih *library* SSL/TLS yang akan digunakan Git untuk koneksi HTTPS.



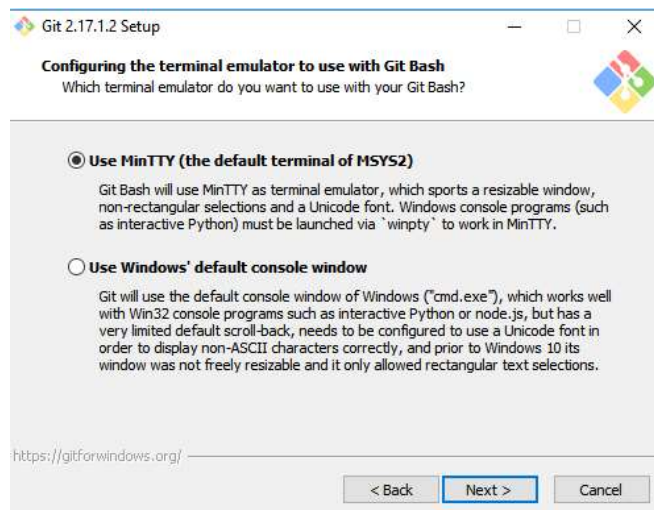
*Memilih library SSL/TLS untuk koneksi HTTPS pada Git*

11. Klik Next. Bagian ini untuk mengkonfigurasi konversi *line ending*.



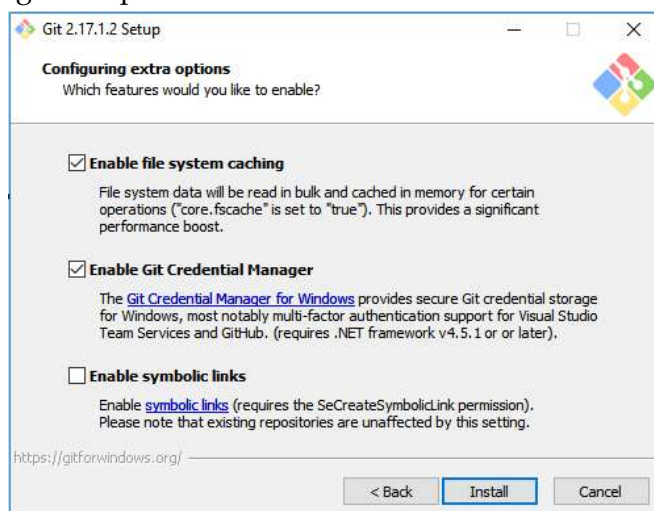
*Mengkonfigurasi konversi line ending*

12. Selanjutnya mengkonfigurasi *emulator* terminal yang akan digunakan dengan Git Bash. Klik Next.



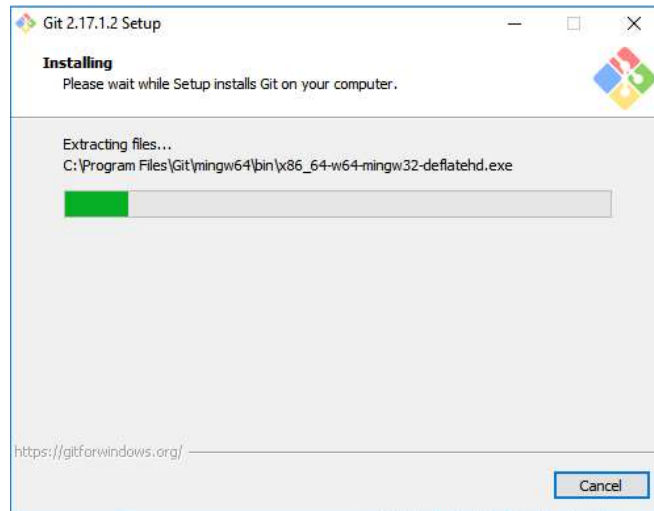
*Mengatur emulator terminal yang digunakan dengan Git Bash*

13. Selanjutnya mengkonfigurasi opsi ekstra. Klik Install.



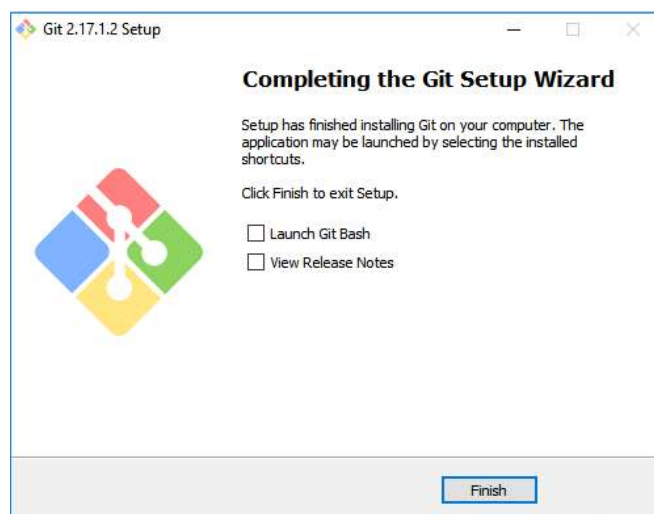
*Mengkonfigurasi opsi ekstra*

14. Tunggu proses instalasi Git sampai selesai.



*Proses instalasi Git*

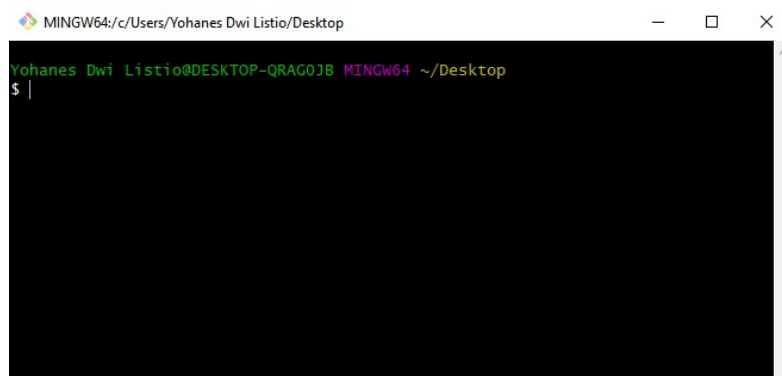
15. Proses instalasi selesai. Hilangkan centang pada **View release notes**. Klik Finish.



*Instalasi Git selesai*

## Konfigurasi awal Git

Konfigurasi awal yang harus dilakukan pada Git adalah mengatur *name* dan *email*. Klik kanan pada desktop Anda lalu pilih Git Bash here. Akan muncul *command prompt* Git seperti berikut ini.



*Tampilan awal Git Bash*

Untuk mengatur *name* dan *email* ketikkan perintah berikut :

```
git config --global user.name = "Nama"
```

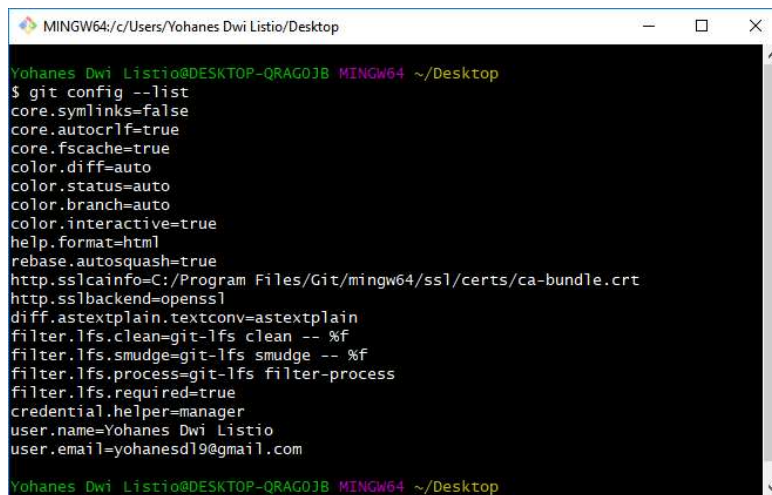
```
git config --global user.email = "example@xxx.com"
```

Nama disini yang dimaksud adalah nama lengkap yang digunakan pada saat membuat akun GitHub. Lihat kembali pada langkah ke-5 bagian *Membuat akun GitHub*.

Cek konfigurasi yang sudah dibuat menggunakan perintah

```
git config --list
```

Jika muncul seperti ini, artinya konfigurasi yang dilakukan berhasil.



```
MINGW64: c:/Users/Yohanes Dwi Listio/Desktop
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Desktop
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
rebase.autosquash=true
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
credential.helper=manager
user.name=Yohanes Dwi Listio
user.email=yohanesd19@gmail.com
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Desktop
```

*Mengecek konfigurasi Git Bash*

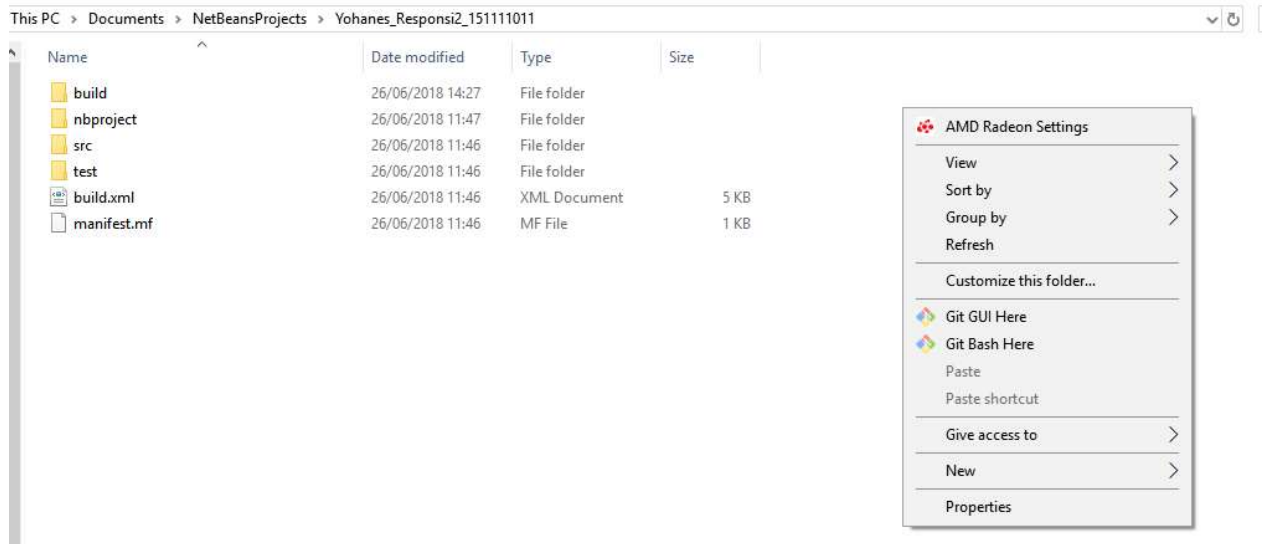


# Tutorial Dasar Git

## Membuat Repositori Baru dalam Projek

Pada langkah-langkah ini kita akan membuat repositori baru dalam direktori projek kita. Bagian ini mengasumsikan bahwa kita sudah mempunyai direktori projek yang di dalamnya sudah ada *file-file coding* di dalamnya.

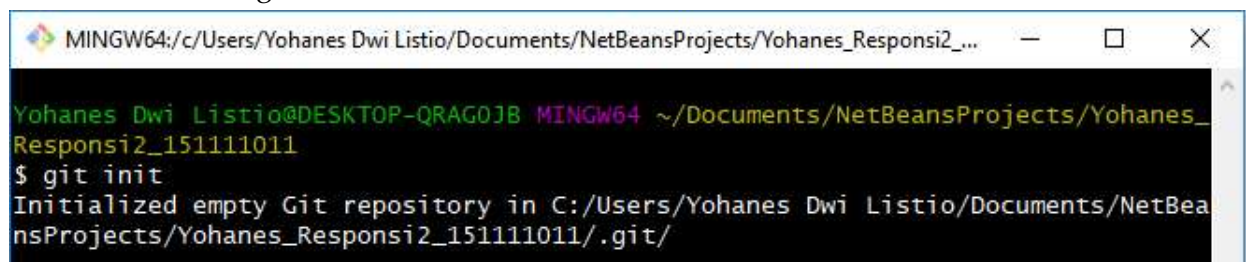
1. Masuk ke dalam direktori projek yang akan dibuat repositori barunya. Klik kanan lalu pilih **Git Bash Here**, sehingga akan muncul *command line* Git.



*Direktori projek Git yang akan dibuat repositori baru*

2. Ketikkan perintah `git init`. Perintah ini akan membuat sebuah direktori bernama `.git` di dalam projek kita. Direktori ini digunakan Git sebagai basis data untuk menyimpan perubahan yang kita lakukan.

**HATI-HATI!** Kalau kita menghapus direktori ini, maka semua rekaman/catatan yang dilakukan oleh Git akan hilang.



*Menginisialisasi repositori Git kosong dalam direktori projek.*

3. Ketikkan perintah berikut untuk menambahkan *remote* ke repositori yang sudah dibuat di GitHub. Contohnya kita sudah membuat repositori di GitHub bernama `responsi-grafis`. Jangan lupa menambahkan `.git` di akhir alamat repositori!

**CATATAN :** semua alamat repositori GitHub menggunakan HTTPS.

`git remote add [nama_remote] [alamat_repositori].git`

Contoh: `git remote add origin https://github.com/yohanesd19/responsi-grafis.git`



*Menambahkan remote ke repositori dalam GitHub*

4. Lakukan *pulling* dari *version control* repositori GitHub dengan perintah :

```
git pull [nama_remote] [nama_branch]
```

Untuk nama branch gunakan **master** karena branch yang digunakan pada repositori yang baru dibuat adalah master.

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_
Responsi2_151111011 (master)
$ git pull origin master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/yohanesd19/responsi-grafis
* branch          master      -> FETCH_HEAD
* [new branch]    master      -> origin/master
```

*Pull dari version control repositori GitHub*

## Membuat Revisi dalam Repositori

Pada contoh direktori proyek di atas, sudah ada *file-file coding* yang siap untuk ditambahkan ke Git. Ketikkan `git status` untuk mengecek status repositori.

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_
Responsi2_151111011 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    build.xml
    build/
    manifest.mf
    nbproject/
    src/

nothing added to commit but untracked files present (use "git add" to track)
```

*Pengecekan status repositori*

Dari keterangan di atas, saat ini kita berada di *branch* master dan ada 2 file dan 3 direktori (dimana sudah terisi file-file) yang belum ditambahkan.

## Kelompok Kondisi File dalam Git

Sebelum membuat revisi untuk pertama kali dalam Git, ada 3 kelompok kondisi file dalam Git yang perlu kita ketahui, di antaranya adalah :

1. **Modified.** Pada kondisi ini revisi/perubahan sudah dilakukan tetapi belum ditandai dan belum disimpan di dalam *version control*. Pada contoh sebelumnya, ada 2 file dan 3 direktori yang dalam kondisi *modified*.
2. **Staged.** Pada kondisi ini revisi sudah ditandai tetapi belum disimpan di dalam *version control*. Untuk mengubah kondisi file dari *modified* ke *staged*, gunakan perintah `git add nama_file`. Atau jika ingin mengubah semua kondisi file ke *staged*, gunakan perintah `git add -A`.  
Contoh: `git add build.xml` (mengubah kondisi file `build.xml` ke *staged*)
3. **Committed.** Pada kondisi ini revisi sudah disimpan di *version control*. Untuk mengubah kondisi file dari *staged* ke *committed*, gunakan perintah `git commit`.



## Revisi Pertama

1. Ubah semua kondisi file menjadi *staged* dengan menggunakan perintah `git add -A`.

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_Responsi2_151111011 (master)
$ git add -A
warning: LF will be replaced by CRLF in src/org/yourorghere/GLRenderer.java.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in src/org/yourorghere/SimpleGLCanvas.form.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in src/org/yourorghere/Yohanes_Responsi2_151111011.java.
The file will have its original line endings in your working directory.
```

Jika muncul pesan warning seperti diatas, biarkan saja.

2. Cek dengan `git status` untuk melihat apakah file sudah dalam kondisi staged.

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_Responsi2_151111011 (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   build.xml
        new file:   build/built-jar.properties
        new file:   build/classes/org/yourorghere/GLRenderer$vector.class
        new file:   build/classes/org/yourorghere/GLRenderer.class
        new file:   build/classes/org/yourorghere/Objek.class
        new file:   build/classes/org/yourorghere/Yohanes_Responsi2_151111011$1.class
        new file:   build/classes/org/yourorghere/Yohanes_Responsi2_151111011$2.class
        new file:   build/classes/org/yourorghere/Yohanes_Responsi2_151111011$3$1.class
        new file:   build/classes/org/yourorghere/Yohanes_Responsi2_151111011$3.class
        new file:   build/classes/org/yourorghere/Yohanes_Responsi2_151111011$4.class
        new file:   build/classes/org/yourorghere/Yohanes_Responsi2_151111011.class
```

3. Ubah kondisi file menjadi *committed* sehingga semua perubahan disimpan oleh Git, dengan perintah `git commit -m "[Komentar mengenai commit ini]"`.

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

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_Responsi2_151111011 (master)
$ git commit -m "Commit pertama"
[master ec7737d] Commit pertama
23 files changed, 2331 insertions(+)
create mode 100644 build.xml
create mode 100644 build/built-jar.properties
create mode 100644 build/classes/org/yourorghere/GLRenderer$vector.class
create mode 100644 build/classes/org/yourorghere/GLRenderer.class
create mode 100644 build/classes/org/yourorghere/Objek.class
```

4. Cek kembali menggunakan `git status`. Revisi pertama sudah dibuat, tetapi langkah terakhir belum kita lakukan, yaitu melakukan *push* ke repositori GitHub kita.

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_R
esponsi2_151111011 (master)
$ git status
On branch master
nothing to commit, working tree clean
```

5. Ketikkan perintah `git push [nama_remote] [nama_branch]` untuk melakukan *push* ke repositori GitHub. Jika Anda diminta untuk memasukkan username dan password akun GitHub Anda, masukkan saja username dan password akun GitHub Anda.

```
Yohanes Dwi Listio@DESKTOP-QRAG0JB MINGW64 ~/Documents/NetBeansProjects/Yohanes_R
esponsi2_151111011 (master)
$ git push origin master
Counting objects: 34, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (30/30), done.
Writing objects: 100% (34/34), 31.25 KiB | 415.00 KiB/s, done.
Total 34 (delta 0), reused 0 (delta 0)
To https://github.com/yohanesd19/responsi-grafis.git
9e89f25..ec7737d master -> master
```

## Melihat Catatan Log Revisi

Sebelumnya, kita sudah melakukan revisi pada repositori yang kita buat. Selanjutnya, kita akan melihat catatan-catatan log dari revisi tersebut.

Git sudah menyediakan perintah `git log` untuk melihat catatan log perubahan pada repositori. Berikut ini contoh penggunaannya.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProje
ct (yohanes)
$ git log
commit b6862834194d3b119ac315d5c1b2dd720296e913 (HEAD -> yohanes, origin/yohanes)
Author: Yohanes Dwi Listio <yohanesd19@gmail.com>
Date: Sun Jul 15 19:05:05 2018 +0700

    Adding darker cone to Earth when moon position is in Sun light's area
```

Untuk menampilkan log yang lebih pendek, tambahkan opsi `--oneline`, sehingga output yang dihasilkan adalah sebagai berikut.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProje
ct (yohanes)
$ git log --oneline
b686283 (HEAD -> yohanes, origin/yohanes) Adding darker cone to Earth when moon p
osition is in Sun light's area
157864d Adding Venus planet by Mr. Go request while test
f4b0fe1 Add max and min zoom
023d4ba Removing checkImageHeight and checkImageWidth
f0b1fc8 Adding control for increase/decrease rotation speed
```

Jika ingin melihat log pada nomor *commit* tertentu, kita bisa memasukkan nomor *commit*-nya.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProjec
t (yohanes)
$ git log b6862834194d3b119ac315d5c1b2dd720296e913
commit b6862834194d3b119ac315d5c1b2dd720296e913 (HEAD -> yohanes, origin/yohanes)
Author: Yohanes Dwi Listio <yohanesd19@gmail.com>
Date: Sun Jul 15 19:05:05 2018 +0700

    Adding darker cone to Earth when moon position is in Sun light's area
```



Untuk melihat log pada file tertentu, kita dapat memasukkan nama filenya.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/eis_ppk/application/models (master)
$ git log mod_akademik.php
commit fa9ccdc58a21380409a54b2e99ebf236180e87f9 (HEAD -> master, origin/master)
Author: Yohanes Dwi Listio <yohanesdl9@gmail.com>
Date: Thu Jul 26 10:17:34 2018 +0700

    Perbaiki tampilan L-AKD24 dan filter status pada L-AKD25
```

Dalam repositori yang dikerjakan oleh banyak orang, kita dapat melihat revisi apa saja yang dilakukan oleh orang tertentu dengan perintah berikut.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProject (yohanes)
$ git log --author='tamhar'
commit f4b0fe18027ae4a09c686cfc57ad4281df8d509
Author: tamhar <arrizky.zen5@gmail.com>
Date: Fri Jul 13 13:57:29 2018 +0700

    Add max and min zoom

commit 6e25b4cf0c1515e0a657681a4af010038d9689a9
Author: tamhar <arrizky.zen5@gmail.com>
Date: Wed Jul 11 20:56:11 2018 +0700

    Add comment about keys binding
```

## Membandingkan Revisi

Untuk melihat perbedaan perubahan yang dilakukan pada revisi, kita akan menggunakan perintah `git diff`.

Misalkan kita ingin melihat perubahan yang dilakukan pada revisi tertentu, gunakan perintah `git diff [nomor_commit]`, seperti contoh berikut ini.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProject (yohanes)
$ git diff 157864d36c5c6654860d914a7db1b698aa30e444
```

```
diff --git a/main.cpp b/main.cpp
index 7f6432b..761236e 100644
--- a/main.cpp
+++ b/main.cpp
@@ -38,15 +38,15 @@ void initGL(){
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
}

-void drawCone(GLfloat height, GLfloat alpha, GLfloat base_d){
+void drawCone(GLfloat height, GLfloat alpha, GLfloat top, GLfloat base_d){
    glDisable(GL_DEPTH_TEST);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glEnable(GL_COLOR_MATERIAL);
    glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
    glColor4f(0.0, 0.0, 0.0, alpha);
-    gluCylinder(q, 1.5, base_d, height, 60, 60);
-    gluDisk(q, 0.0, 1.5f, 60, 60);
+    gluCylinder(q, top, base_d, height, 60, 60);
+    gluDisk(q, 0.0, top, 60, 60);
    glDisable(GL_COLOR_MATERIAL);
}
```

Jika diperhatikan, terdapat simbol (+) dan berwarna hijau yang berarti ada kode yang ditambahkan, serta tanda (-) dan berwarna merah yang menunjukkan kode yang dihapus. Dan perbedaan perubahan yang ditampilkan adalah perubahan dari semua file yang ada pada repositori.

Anda dapat menggunakan perintah `git diff [nama_file]` jika hanya ingin melihat perbedaan perubahan yang dilakukan pada file tertentu saja.

Kita juga dapat menggunakan perintah ini untuk membandingkan revisi antar *commit* maupun antar cabang (*branches*) dengan perintah :

`git diff [nomor_commit] [nomor_commit]` untuk perbandingan revisi antar *commit*, atau

`git diff [nama_cabang] [nama_cabang]` untuk perbandingan revisi antar cabang (*branches*).

## Membatalkan Revisi

Kadang-kadang dalam revisi yang dilakukan kita melakukan kesalahan dan kita ingin mengembalikannya menjadi keadaan semula. Maka kita perlu menyuruh Git melakukan itu. Ada 3 perintah yang biasanya digunakan, yaitu `git checkout`, `git reset` dan `git revert`. Berikut ini cara penggunaannya.

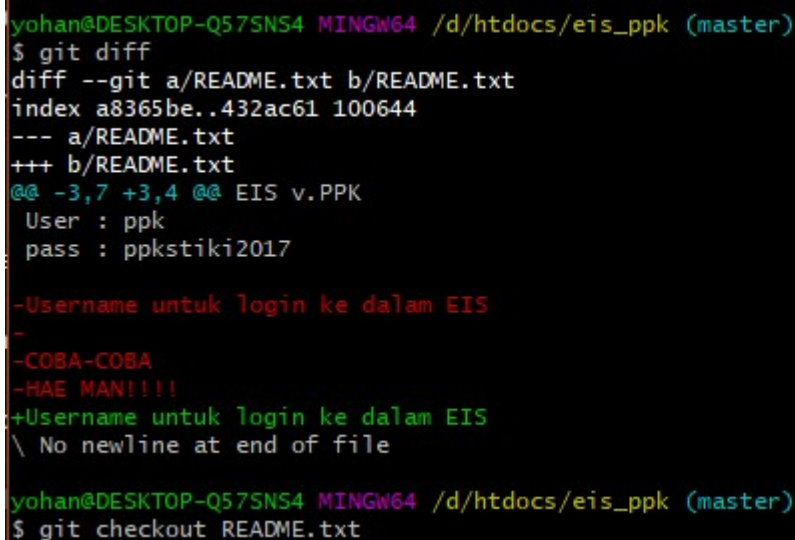
**CATATAN :** Hati-hati dalam menggunakan perintah-perintah ini karena jika kita membuat revisi yang cukup banyak, revisi akan terasa sia-sia setelah perintah ini dijalankan.

## Git Checkout

Perintah ini digunakan bila revisi yang dilakukan belum pada kondisi *staged* maupun *committed*. Untuk melakukannya ketikkan perintah :

`git checkout [nama_file]` jika hanya ingin membatalkan revisi pada file tertentu saja, atau

`git checkout -- .` jika ingin membatalkan revisi pada semua file yang diubah.



```
yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/eis_ppk (master)
$ git diff
diff --git a/README.txt b/README.txt
index a8365be..432ac61 100644
--- a/README.txt
+++ b/README.txt
@@ -3,7 +3,4 @@ EIS v.PPK
 User : ppk
 pass : ppkstiki2017

-Username untuk login ke dalam EIS
-
-COBA-COBA
-HAE MAN!!!!
+Username untuk login ke dalam EIS
\ No newline at end of file

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/eis_ppk (master)
$ git checkout README.txt
```

```

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ 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:   application/controllers/welcome.php
        modified:   application/views/compiler.php

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

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ git checkout -- .

```

## Git Reset

Perintah ini digunakan bila revisi yang dilakukan sudah pada kondisi *staged*, tetapi belum dalam kondisi *committed*. Untuk melakukannya ikuti langkah berikut :

1. Ketikkan perintah `git reset` untuk mengembalikan kondisi file menjadi *modified*.
2. Ketikkan perintah `git checkout -- .` untuk membatalkan perubahannya.

Jika ingin melakukan reset pada file-file tertentu saja, pada perintah `git reset` tambahkan nama file yang ingin di-reset (`git reset [nama_file]`), kemudian ketikkan `git checkout [nama_file]`.

```

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ git reset
Unstaged changes after reset:
M   application/controllers/welcome.php
M   application/views/compiler.php

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ 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:   application/controllers/welcome.php
        modified:   application/views/compiler.php

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

```

```

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ git checkout -- .

yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ git status
On branch master
nothing to commit, working tree clean

```

## Membatalkan Perubahan File dalam Kondisi *Committed*

Perintah ini digunakan bila revisi yang dilakukan sudah sampai pada tahap *committed*. Untuk melakukannya, kita perlu mengetahui nomor *commit*, kemudian mengembalikan revisi seperti pada nomor *commit* tersebut.

Untuk mengetahui nomor *commit* kita dapat menggunakan perintah `git log`.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ git log
commit fe19569e3b4544b6ce824b2300be1fa2415dd199 (HEAD -> master, origin/master)
Merge: 1191aad e4b7441
Author: Yohanes Dwi Listio <yohanesd19@gmail.com>
Date: Sat Jul 7 15:00:12 2018 +0700

    Merge branch 'master' of https://github.com/yohanesd19/JavaOnlineCompiler

commit 1191aad7970c810d1df260f9705082a2ac0a6bba
Author: Yohanes Dwi Listio <yohanesd19@gmail.com>
Date: Sat Jul 7 14:59:06 2018 +0700

    Adding Server Time and Modify Views
```

Untuk mengembalikan kondisi sebuah file seperti pada commit sebelumnya, ketikkan perintah

`git checkout [nomor_commit] [nama_file]`

Kemudian kita bisa mengembalikan file tersebut dalam kondisi *modified* dengan git reset.

Untuk mengembalikan seluruh file dalam *commit*, cukup lakukan *checkout* ke nomor *commit* tanpa diikuti nama file, seperti pada contoh di bawah ini.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler (master)
$ git checkout fe19569e3b4544b6ce824b2300be1fa2415dd199
Note: checking out 'fe19569e3b4544b6ce824b2300be1fa2415dd199'.

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 performing another checkout.

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

    git checkout -b <new-branch-name>

HEAD is now at fe19569 Merge branch 'master' of https://github.com/yohanesd19/JavaOnlineCompiler
yohan@DESKTOP-Q57SNS4 MINGW64 /d/htdocs/java_compiler ((fe19569...))
$ git status
HEAD detached at fe19569
nothing to commit, working tree clean
```

## Kembali ke Beberapa Commit Sebelumnya

Untuk kembali ke beberapa commit sebelumnya, ketikkan perintah berikut :

`git checkout HEAD~[jumlah n-commit kembali]`

Misalkan kita ingin kembali ke 2 commit sebelumnya, maka perintahnya seperti berikut :

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProject (yohanes)
$ git checkout HEAD~2
```

## Maju ke Beberapa Commit Sesudahnya

Untuk maju ke beberapa commit sesudahnya, ketikkan perintah berikut :

`git checkout HEAD~[jumlah n-commit kembali]`

Misalkan kita ingin maju ke 3 commit setelahnya, maka perintahnya seperti berikut :



```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProject ((f4b0fe1...))
$ git checkout HEAD@{3}
Previous HEAD position was f4b0fe1 Add max and min zoom
HEAD is now at b686283 Adding darker cone to Earth when moon position is in Sun light's area
```

Perintah ini hanya bisa dijalankan bila Anda berada pada beberapa commit sebelum commit terakhir Anda.

## Membatalkan Semua Perubahan yang Ada (Git Revert)

Jika kita ingin mengembalikan semua file ke suatu commit, ketikkan perintah berikut :

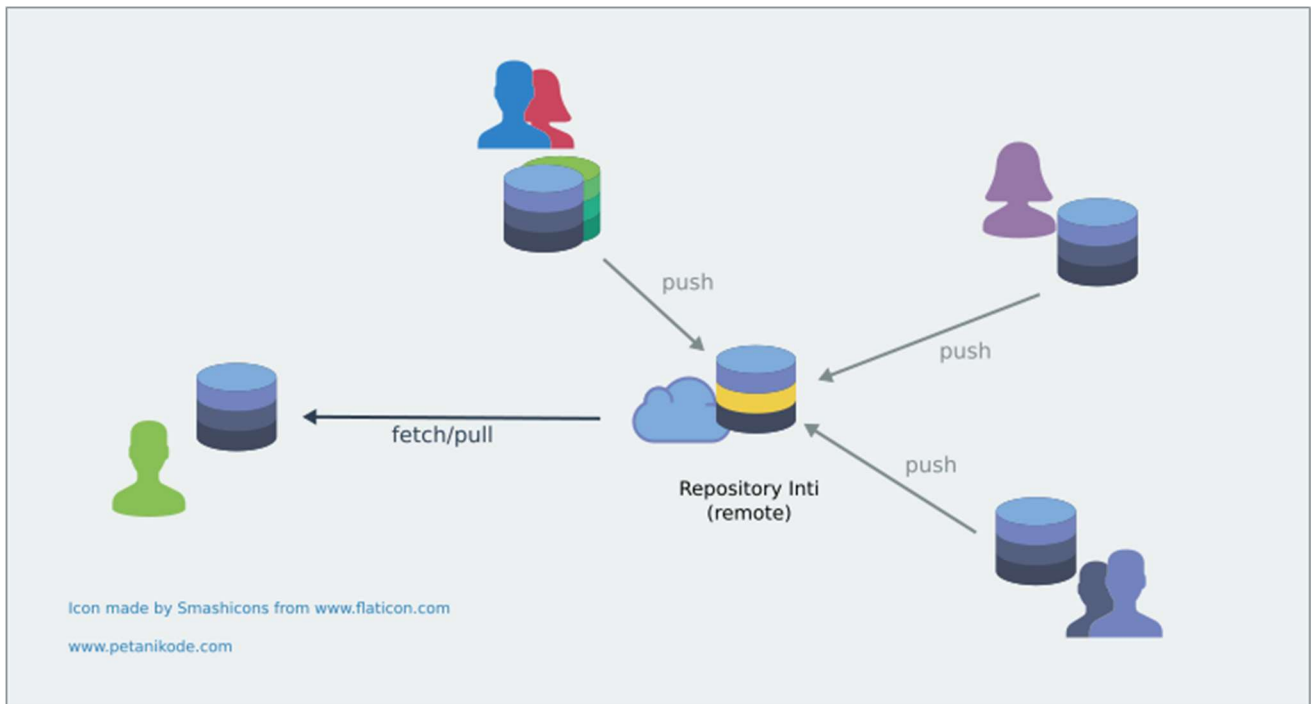
```
git revert -n [nomor_commit]
```

Contoh : `git revert -n 1191aad7970c810d1f260f9705082a2ac0a6bba`

## Mengambil Revisi dari Repositori

Saat kita bekerja dengan repositori yang memiliki banyak kontributor, kita seharusnya mengambil dulu revisi terbaru dari repositori inti agar tidak bentrok.

Misalnya begini, pada repositori remote (yang ada pada GitHub) ada kontributor lain yang sudah menambahkan dan mengubah sesuatu di sana.



Maka perubahan-perubahan tersebut harus diambil dahulu, agar repositori lokal kita tetap *up-to-date* atau sama persis seperti repositori pada remote.

Ada dua perintah untuk mengambil revisi dari repositori remote, yaitu :

1. `git fetch [nama_remote] [nama_cabang]`
2. `git pull [nama_remote] [nama_cabang]`

Perbedaan dari kedua perintah tersebut adalah sebagai berikut :

1. Perintah `git fetch` hanya akan mengambil revisi (*commit*) saja dan tidak langsung melakukan penggabungan (*merge*) terhadap repository lokal. Bila kita sudah membuat perubahan di

repository lokal, maka sebaiknya menggunakan `git fetch` agar perubahan yang kita lakukan tidak hilang.

2. Perintah `git pull` akan mengambil revisi (*commit*) dan langsung melakukan penggabungan (*merge*) terhadap repository lokal. Bila kita tidak pernah melakukan perubahan apapun dan ingin mengambil versi terakhir dari repository remote, maka gunakanlah `git pull`.

Untuk memahami secara lebih jelas penggunaan keduanya, coba lakukan langkah-langkah berikut ini.

## Mengambil Revisi dengan Git Fetch

Misalkan kita ingin mengedit README.md. Setelah mengubah file tersebut, lakukan langkah-langkah berikut :

1. Masukkan perintah `git add -A` sehingga README.md dalam kondisi *staged*.
2. Masukkan perintah `git commit -m "Pesan Commit"` sehingga README.md dalam kondisi *committed*.
3. Masukkan perintah `git fetch [nama_remote] [nama_cabang]` untuk mengambil perubahan terbaru pada repositori remote. Terlihat bahwa revisi sudah diambil tetapi di repositori lokal belum ada yang berubah dari file README.md.
4. Jika ingin menggabungkan *commit* dari repositori remote dengan lokal, gunakan perintah `git merge [nama_cabang] [nama_remote]/[nama_cabang]`.

## Mengambil Revisi dengan Git Pull

Kita akan mencoba contoh yang sama seperti pada kasus `git fetch`, tetapi dengan cara `git pull`.

1. Lakukan langkah 1-2.
2. Selanjutnya, masukkan perintah `git pull [nama_remote] [nama_cabang]`. Semua revisi akan diambil dan langsung digabungkan (*merge*). Hati-hati bahwa hal ini dapat memicu bentrok pada kode (untuk mengatasi bentrok jika ada, baca subbab Mengatasi Bentrok di bab Percabangan untuk Mencegah Konflik).

## Clone Repositori Remote

*Clone* repositori remote sama saja dengan menyalin repositori dari remote ke lokal. Untuk melakukannya, gunakan perintah

```
git clone [URL repositori remote].git [nama direktori]
```

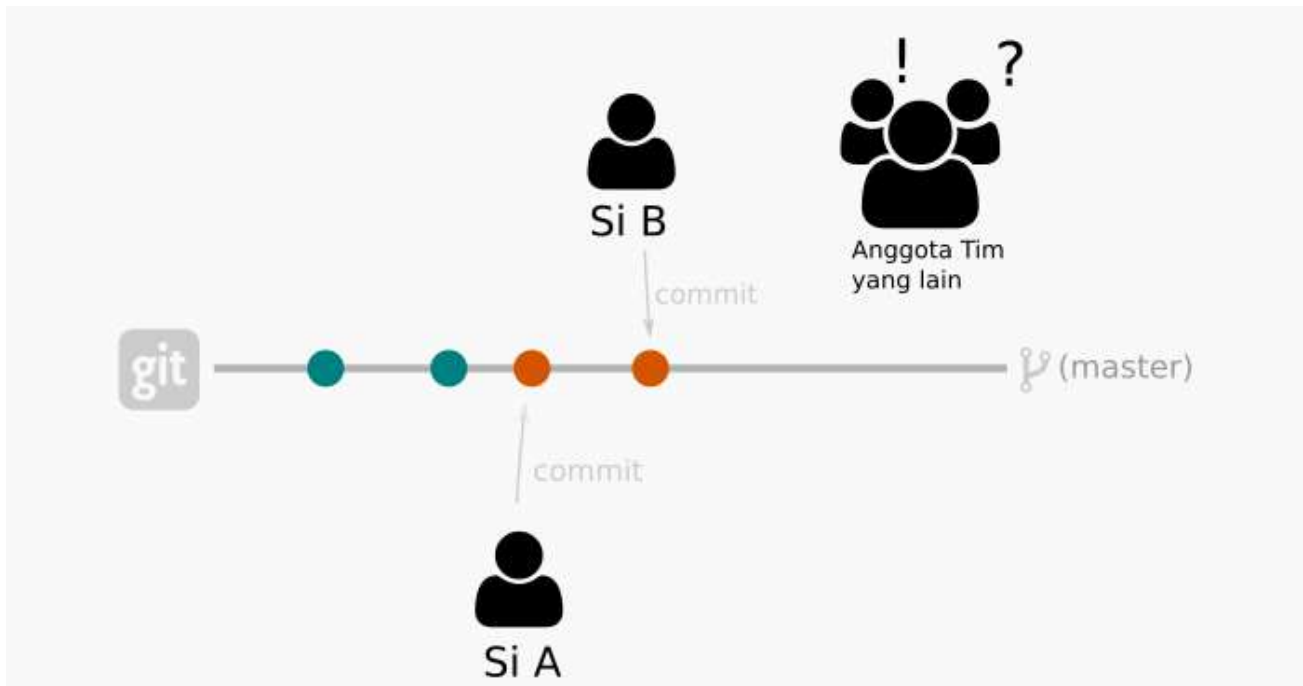
Jika [nama direktori] di atas tidak diberikan, secara otomatis menggunakan nama repositori. Saat Anda melakukan *clone* sebuah repositori dari GitHub, nama remote *origin* diberikan secara otomatis.



## Percabangan untuk Mencegah Konflik

Bayangkan kamu sedang bekerja dengan tim pada suatu repositori Git. Repositori ini dikerjakan secara bersama-sama. Kadang-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 *source code* 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).

## Membuat Cabang Baru

Untuk membuat cabang baru, gunakan perintah `git branch [nama_cabang_baru]`.

Contoh: `git branch cabang_baru`

Setelah itu, kita berpindah ke cabang yang baru saja kita buat, dengan perintah `git checkout [nama_cabang]`

Contoh: `git checkout cabang_baru`

Selanjutnya kita bisa mencoba menambahkan file baru atau mengubah file yang sudah ada, lalu melakukan *commit* dan *push* ke cabang baru tersebut.

Silahkan cek pada cabang master. Jika langkah-langkah di atas dilakukan dengan benar, perubahan pada cabang baru (dalam hal ini cabang\_baru) tidak akan berpengaruh pada cabang master.

## Menggabungkan Cabang

Misalkan kita mempunyai 2 buah cabang, yaitu master (utama) dan cabang\_baru. Kita ingin menggabungkan cabang ini ke cabang master. Ikuti langkah-langkah berikut :

1. Pindah dahulu ke cabang master.
2. Gabungkan dengan perintah `git merge [nama_cabang]`. Namun, Anda harus berhati-hati karena sering terjadi bentrok ketika menggabungkan cabang. Cara mengatasi bentrok akan dibahas pada bagian sesudah ini.

```
yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProje
ct (yohanes)
$ git checkout master
Switched to branch 'master'

yohan@DESKTOP-Q57SNS4 MINGW64 /e/Kuliah/Semester 6/Pemrograman Grafis/FinalProje
ct (master)
$ git merge yohanes
Updating fe92434..b686283
Fast-forward
 FinalProject.depend      | 7 ++-
 FinalProject.layout      | 2 +-
 bin/Debug/FinalProject.exe | Bin 47321 -> 49678 bytes
 main.cpp                 | 111 ++++++-----
 obj/Debug/main.o          | Bin 11990 -> 14409 bytes
 5 files changed, 86 insertions(+), 34 deletions(-)
```

## Mengatasi Bentrok

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

Bisa jadi, di cabang yang mereka kerjakan, ada file yang sama dengan cabang lain. Sehingga, terjadi bentrok pada saat digabungkan. Pemilik/pengelola repositori bertugas dalam menangani konflik ini. Dia harus bertindak adil, kode mana yang harus diambil.

Biasanya, akan ada diskusi terlebih dahulu sebelum memutuskan.

1. Misalkan kita mempunyai 2 buah cabang, yaitu master dan yohanes pada contoh ini, dan kita sekarang berada pada cabang yohanes. Diasumsikan bahwa sudah ada perubahan baik pada kedua cabang ini, dan sekarang kita sudah berada di cabang master.
2. Kita coba gabungkan kedua cabang ini dengan perintah `git merge`. Terjadi error.

```
$ git merge yohanes
Auto-merging login.html
CONFLICT (content): Merge conflict in login.html
Automatic merge failed; fix conflicts and then commit the result.
```

3. Di sini kita diminta memperbaiki kode yang bentrok. Kita akan membuka file dimana terdapat kode yang bentrok (dalam contoh ini `login.html`) dengan *text 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 bentrok</p>
<p>di sini berisi kode untuk halaman login<p>
>>>>>>> halaman_login
```

Kedua kode cabang dipisahkan dengan tanda =====. Untuk memperbaikinya, silahkan eliminasi salah satu dari kode cabang tersebut. Setelah itu lakukan *commit* untuk menyimpan perubahan tersebut. Hal ini juga bisa dilakukan bila terdapat bentrok saat melakukan perintah *git pull* atau perintah lain yang melibatkan merge.

## Menghapus Cabang

Cabang yang sudah mati (tidak ada pengembangan lagi), sebaiknya dihapus agar repositori kita bersih dan rapi.

Untuk menghapus cabang, gunakan perintah *git branch -d* diikuti dengan nama cabang yang ingin dihapus.

Contoh: *git branch -d cabang\_baru*

## Apa Selanjutnya?

Kita sudah belajar beberapa perintah untuk bekerja pada repositori remote, seperti `git remote`, `git push`, `git fetch`, `git pull`, `git clone`, dan lain sebagainya. Semua perintah itu kita perlukan saat berkolaborasi dengan tim di proyek *open source* maupun *closed source*.

Apa selanjutnya?

Perbanyak latihan dan sering-sering menggunakan Git tiap hari agar terbiasa. Penguasaan akan Git *version control* sangat diperlukan *developer* masa kini untuk memudahkan kolaborasi *coding* antar *developer*.

Selain itu, Anda harus membiasakan diri dengan *trial* dan *error*, karena Anda tentu akan menemui *error* ataupun *fatal* yang sering terjadi pada saat bekerja pada repositori remote, baik karena kesalahan Anda ataupun rekan *developer* yang bekerjasama dengan Anda.

## Daftar Rujukan

- Ardianta. (2017, February 8). *Tutorial Git #1: Cara Install Git dan Konfigurasi Awal yang Harus Dilakukan*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-install/>
- Ardianta. (2017, February 9). *Tutorial Git #2: Cara Membuat Repositori Baru dalam Proyek*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-init/>
- Ardianta. (2017, February 13). *Tutorial Git #3: Simpan Perubahan Revisi dengan Git Commit*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-commit/>
- Ardianta. (2017, February 14). *Tutorial Git #4: Melihat Catatan Log Revisi*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-log/>
- Ardianta. (2017, February 16). *Tutorial Git #5: Melihat Perbandingan Revisi dengan Git Diff*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-diff/>
- Ardianta. (2017, February 21). *Tutorial Git #6: Perintah untuk Membatalkan Revisi*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-revert/>
- Ardianta. (2017, February 27). *Tutorial Git #7: Menggunakan Percabangan untuk Mencegah Konflik*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-branch/>
- Ardianta. (2017, September 29). *Tutorial Git #9: Bekerja dengan Remote Repositori*. Diambil kembali dari Petanikode: <https://www.petanikode.com/git-remote/>