

# Pengenalan Git

Detail Materi



Memahami Kegunaan Git

Indikator :  
Dapat menggunakan Git untuk keperluan  
menyimpan proyek aplikasi



Perintah Dasar Git



Menginstall dan Menjalankan Git



Remote Repository Github/Gitlab

## **Modul Sekolah DevOps Cilsy**

Hak Cipta © 2019 **PT. Cilsy Fiolution Indonesia**

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronik maupun mekanis, termasuk mecropy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Adi Saputra, Irfan Herfiandana & Tresna Widiyaman  
Editor: Muhammad Fakhri Abdillah, Rizal Rahman & Tresna Widiyaman  
**Revisi Batch 4**

Penerbit : **PT. Cilsy Fiolution Indonesia**

Web Site : <https://cilsyfiolution.com> , <https://devops.cilsy.id>

### **Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta**

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)

## Daftar Isi

[illegible]

4.4.2. Menggabungkan Cabang.....	28
4.4.3. Mengatasi Adanya Bentrok.....	29
4.4.4. Menghapus Cabang.....	31
4.4.5. Exercise.....	31
4.5. Remote Repository.....	31
4.5.1. Membuat Repositori di GitHub.....	32
4.5.2. Menambah Remote Repository.....	33
4.5.3. Menggunakan SSH di GitHub.....	34
4.5.3.1. Membuat SSH Key.....	34
4.5.3.2. Jalankan SSH Agent dan Load SSH Key.....	35
4.5.3.3. Uji Konektivitas.....	36
4.5.3.4. Mengubah dan Menghapus Remote Repository.....	36
4.5.4. Mengirim Revisi ke Remote Repository.....	36
4.5.5. Mengambil Revisi dan Remote Repository.....	40
4.5.6. Mengambil Revisi dengan Git Fetch.....	41
4.5.7. Mengambil Revisi dengan Git Pull.....	43
4.5.8. Clone Remote Repository.....	45
4.5.9. Exercise.....	46
4.6. Summary.....	46

## 4.

# Pengenalan Git

### Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami Kegunaan Git
2. Memahami Perintah Dasar Git.
3. Mampu Menginstall dan Menjalankan Git.
4. Mampu Menggunakan Remote Repository Github/Gitlab

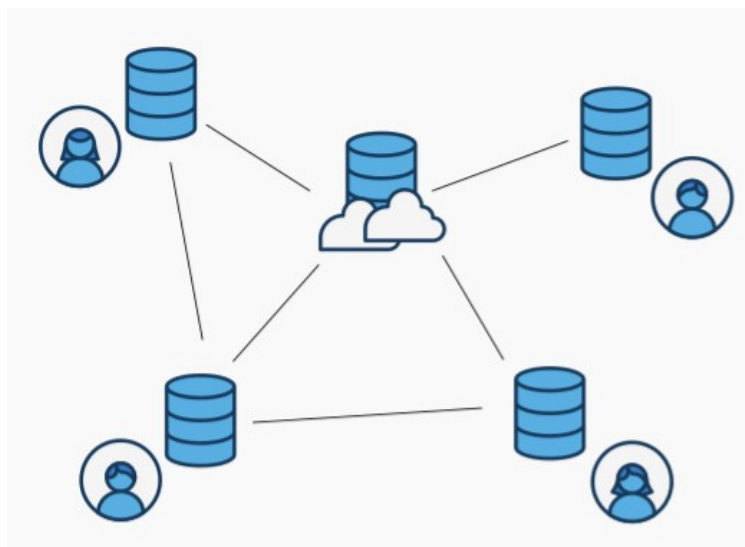
### Outline Materi

1. Pengenalan Git
2. Instalasi dan Konfigurasi Git
3. Membuat Repositori
4. Membuat Revisi
5. Membuat Branch
6. Remote Repositori

## 4.1. Pengenalan Git

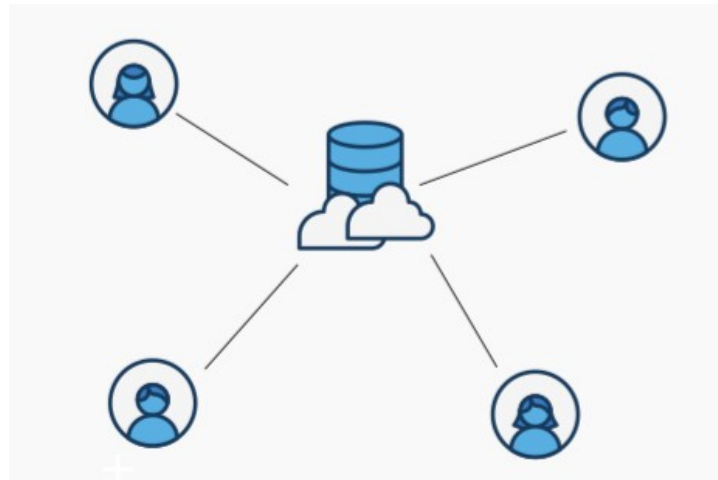
### 4.1.1. Apa itu Git ?

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



*Ilustrasi Distributed Revision Control*

Semua orang yang terlibat dalam coding sebuah proyek akan menyimpan database Git, sehingga akan memudahkan dalam mengelola proyek baik online maupun offline. Dalam Git terdapat *merge*, yaitu aktifitas penggabungan kode. Sedangkan pada VCS yang terpusat, database disimpan dalam satu tempat dan setiap perubahan disimpan ke sana.



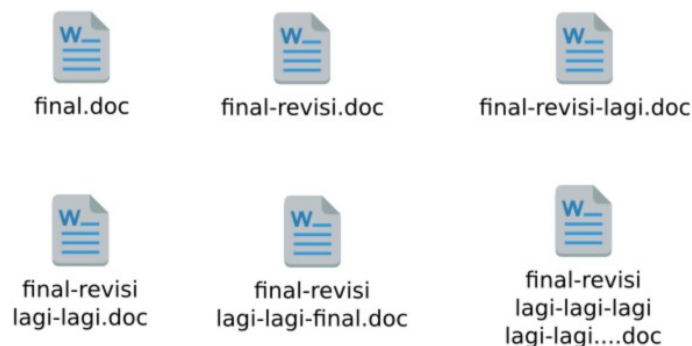
*Ilustrasi Marge*

Meskipun begitu VCS terpusat sendiri memiliki beberapa kekurangan yang diantaranya adalah sebagai berikut.

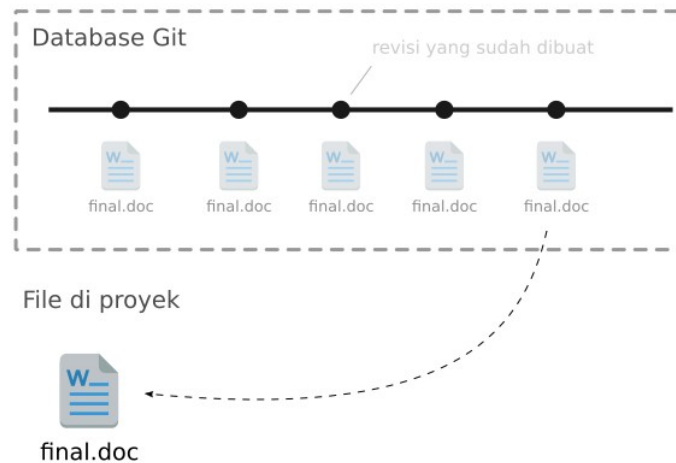
- Semua tim harus terkoneksi ke jaringan untuk mengakses source-code.
- Data tersimpan di satu tempat, akan bermasalah jika server mengalami trouble.

Karena itu, Git hadir untuk menutupi kekurangan yang dimiliki oleh VCS terpusat. Git akan memantau semua perubahan yang terjadi pada file proyek. Lalu menyimpannya ke dalam database baik offline maupun online.

Berikut merupakan perbandingan sebelum dan sesudah menggunakan Git.



*Ilustrasi Sebelum Menggunakan Git*



*Ilustrasi setelah menggunakan Git*

Perbedaan dari kedua tersebut 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 dan Git memungkinkan kita kembali ke versi revisi yang kita inginkan.

Git sangat penting bagi programmer selain untuk mengontrol versi, git juga digunakan untuk kolaborasi. Saat ini Git menjadi salah satu tool terpopuler yang digunakan pengembangan software open source maupun closed source. Beberapa perusahaan raksasa yang menggunakan Git diantaranya adalah Google, Microsoft, Facebook dan berbagai perusahaan raksasa lainnya.

Berikut ini ada beberapa manfaat yang akan Anda rasakan setelah bisa menggunakan Git :

- Bisa menyimpan seluruh versi source code.
- Bisa paham cara kolaborasi dalam proyek.
- Bisa ikut berkontribusi ke proyek open-source.



- Lebih aman digunakan untuk kolaborasi, karena kita bisa tahu apa yang diubah dan siapa yang mengubahnya.
- Bisa memahami cara deploy aplikasi modern.
- dan sebagainya.

## 4.2. Instalasi dan Konfigurasi Git

### 4.2.1. Instalasi Git di Linux

Berikut merupakan beberapa cara instalasi Git pada GNU/Linux yang terbagi kedalam beberapa distro, distro keluarga Debian dapat menggunakan perintah apt sebagai berikut.

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

Untuk keluarga Fedora dapat menggunakan perintah yum seperti berikut.

```
sudo yum install git
```

Untuk mengecek versi Git kita bisa gunakan perintah dibawah ini.

```
git --version
```

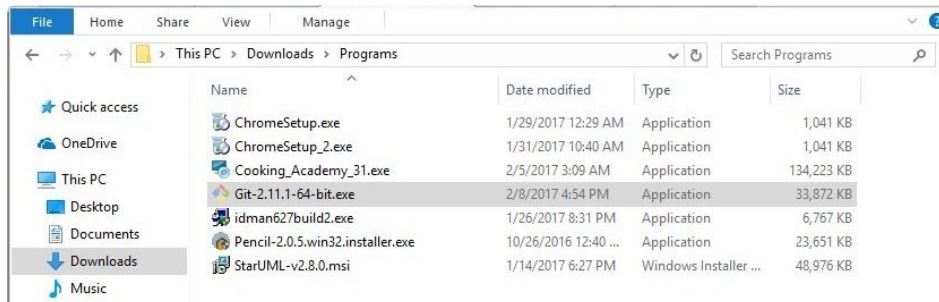
### 4.2.2. Instalasi Git di Windows

Selanjutnya adalah Instalasi Git di Windows, memang tidak seperti di Linux yang hanya perlu mengetikan perintah untuk menginstall. Di windows kita harus men-download terlebih dahulu, kemudian melakukan ritual next > next > finish. Tapi dalam ritual tersebut, ada pilihan yang harus diperhatikan agar perintah git dapat dikenali di CMD.

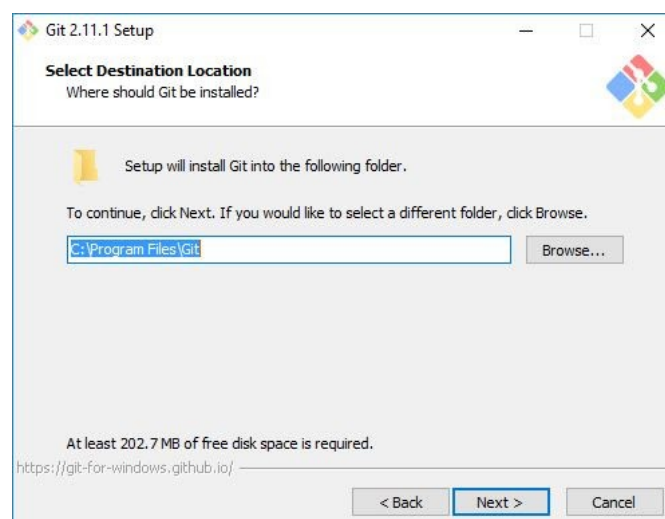
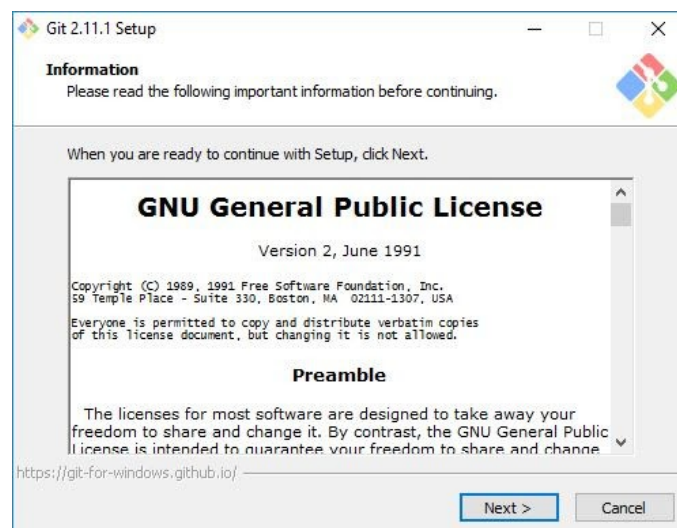
- Untuk Mendownload Git cukup kita buka website resminya Git (git-scm.com). Kemudian unduh Git sesuai dengan arsitektur komputer kita.

Kalau menggunakan 64bit, unduh yang 64bit. Begitu juga kalau menggunakan 32bit.

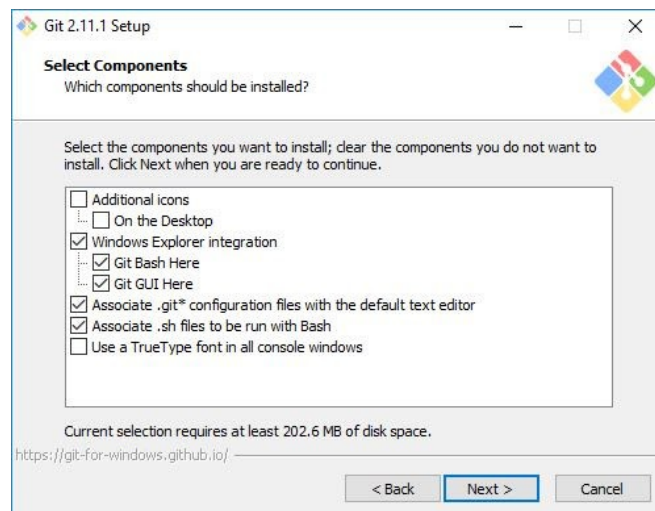
- klik 2x file instaler Git yang sudah didownload.



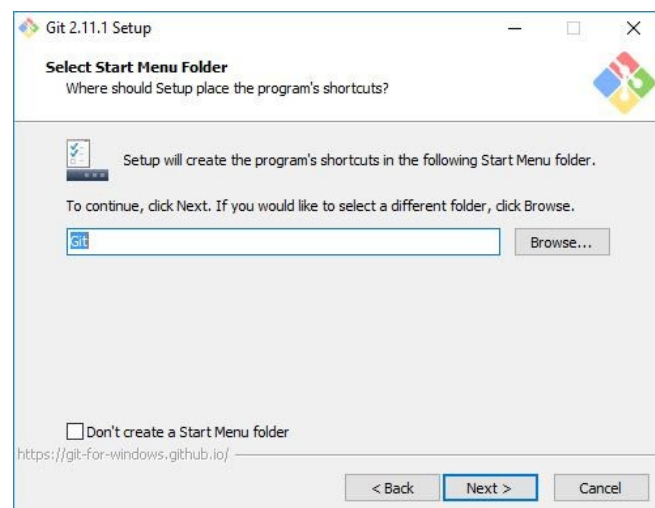
- Setelah itu akan muncul informasi lisensi Git, klik Next > untuk melanjutkan.



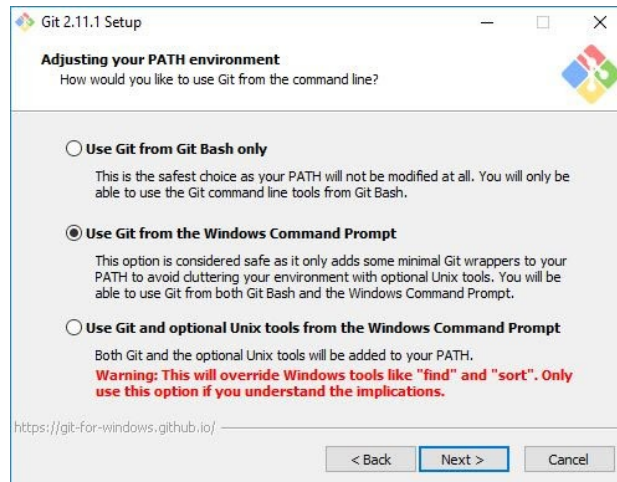
- Selanjutnya menentukan lokasi instalasi. Biarkan saja apa adanya, kemudian klik Next
- Selanjutnya pemilihan komoponen, biarkan saja seperti ini kemudian klik Next.



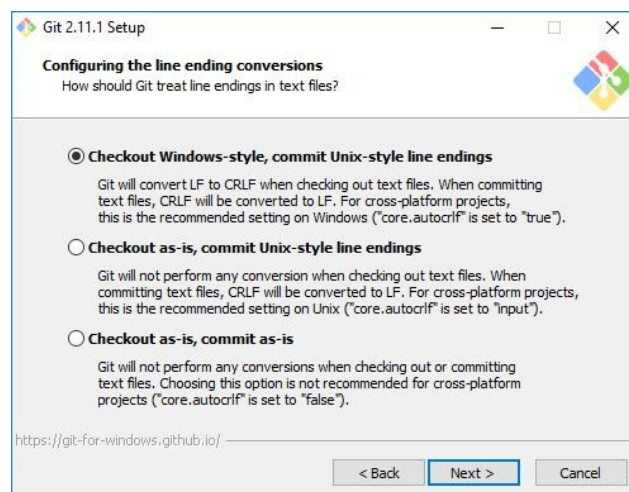
- Selanjutnya pemilihan direktori start menu, klik Next



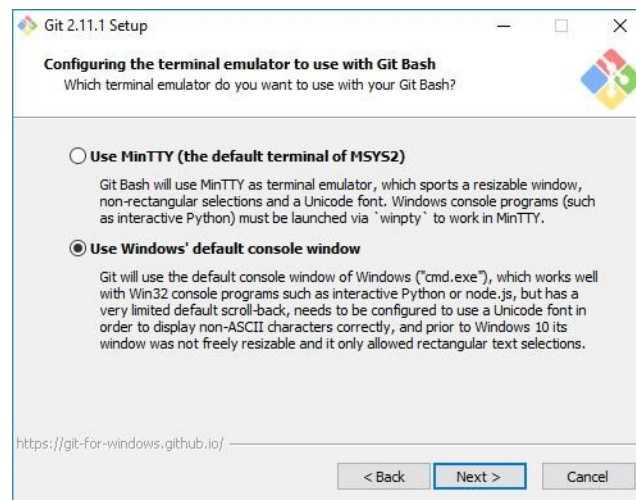
- Selanjutnya pengaturan PATH Environment. Pilih yang tengah agar perintah git dapat di kenali di Command Prompt (CMD). Setelah itu klik Next



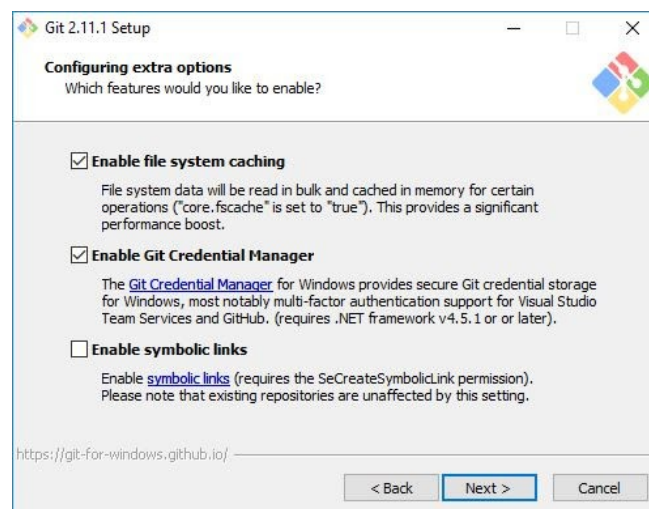
- Selanjutnya konversi line ending. Biarkan saja seperti ini, kemudian klik Next >



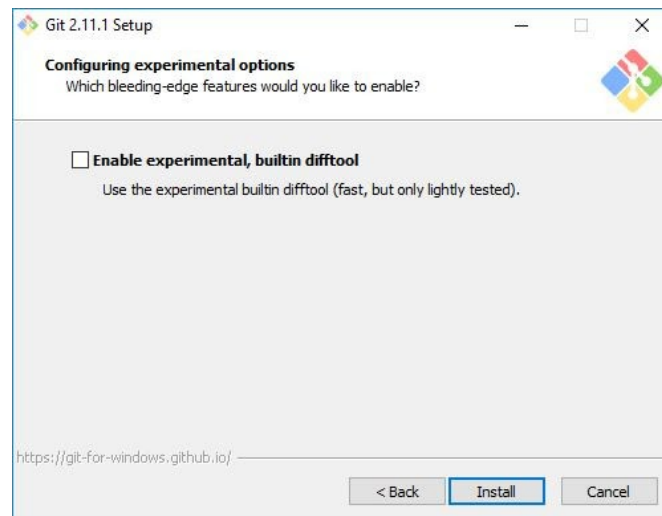
- Pada pemilihan emulator terminal. Pilih saja windows console, kemudian klik Next.



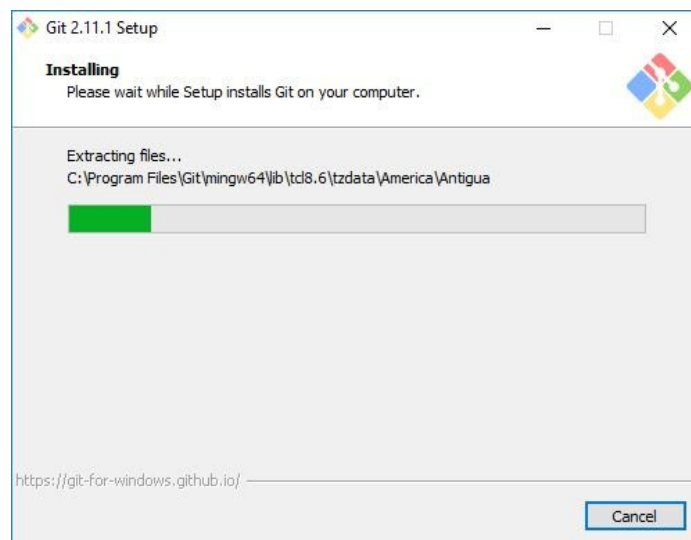
- Selanjutnya pemilihan opsi ekstra. Klik saja Next.



- Selanjutnya pemilihan opsi eksperimental, langsung saja klik Install untuk memulai instalasi.



- Tunggu beberapa saat, instalasi sedang dilakukan.



- Setelah selesai, kita bisa langsung klik Finish.



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

### 4.2.3. Konfigurasi Git

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 "Irfan Herfiandana"
git config --global user.email irfanherfiandana@gmail.com
```

Kemudian periksa konfigurasinya dengan menggunakan perintah berikut.

```
git config --list
```

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

```
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~$ git config --global user.name "Irfan Herfiandana"
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~$ git config --global user.email irfanherfiandana@gmail.com
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~$ git config --list
user.name=Irfan Herfiandana
user.email=irfanherfiandana@gmail.com
```

*Konfigurasi Git*

#### 4.2.4. Perisiapan File Untuk Git

Karena pada git kita akan belajar untuk mengontrol sebuah proyek yang kita miliki, maka dari itu kita harus mempersiapkan proyek yang akan kita kontrol. Disini kita akan coba membuat beberapa file html dan folder yang akan kita gunakan.

Pertama kita buat sebuah folder project kita dengan nama proyekgit. Setelah itu kita buat sebuah file dengan nama **index.html**, isikan script berikut di file tersebut.

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

Setelah itu simpan file, lalu kita buat file lain dengan nama **about.html** dan **contact.html** lalu masukan script yang sama dengan **index.html** dalam kedua file baru tersebut. Buat juga file dengan nama test.php lalu isikan script berikut didalamnya.

```
<?php
echo "Ini adalah test";
?>
```

Setelah itu buat beberapa direktori dengan nama **vendor**, **cache**, dan **upload**. Sehingga jika semua sudah kita buat akan menghasilkan seperti dibawah ini.

```
ubuntu@ip-172-31-12-225:~/belajargit$ ls
about.html  cache  contact.html  index.html  test.php  upload  vendor
ubuntu@ip-172-31-12-225:~/belajargit$
```



## 4.3. Membuat Repositori dan Revisi

### 4.3.1. Membuat Repositori

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

Pembuatan repositori dapat dilakukan dengan perintah : **git init nama-direktori**

```
git init cilsy
```

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

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

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

### 4.3.1.1. Gitignore

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.

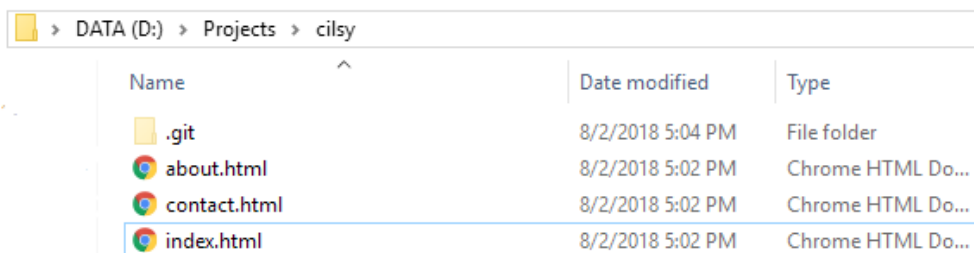
Untuk dapat menggunakan `.gitignore`, buat saja sebuah file bernama `.gitignore` dalam root direktori **proyek/repo**. Misalnya kita isi filenya seperti berikut :

```
/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.

### 4.3.2. Revisi

Sebelumnya kita sudah membuat repositori kosong. Sekarang kita coba tambahkan sebuah file baru. Sebagai contoh, kita akan menambahkan tiga file HTML kosong.



Name	Date modified	Type
.git	8/2/2018 5:04 PM	File folder
about.html	8/2/2018 5:02 PM	Chrome HTML Do...
contact.html	8/2/2018 5:02 PM	Chrome HTML Do...
index.html	8/2/2018 5:02 PM	Chrome HTML Do...

*Pembuatan file html*

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

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilsy (master)
$ 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)
```

*Melihat status Git*

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

Ada Tiga Kelompok Kondisi File dalam Git yang diantaranya sebagai berikut.

### 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*.

Sekarang kita 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
```

Kita dapat melakukan seperti dibawah ini untuk menandai banyak.

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

Atau seperti ini untuk add extention yang dipilih.

```
git add *.html
```

Jika ingin add semuanya, gunakan perintah ini(semua file dan direktori di current directory).

```
git add .
```

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

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ 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
```

*Melihat status Staged*

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

```
git commit -m 'Commit Pertama'
```

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git commit -m 'Commit Pertama'
[master (root-commit) 689c874] Commit Pertama
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 about.html
create mode 100644 contact.html
create mode 100644 index.html

Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git status
On branch master
nothing to commit, working tree clean
```

*Melihat status Commit*

Sekarang kita akan mencoba untuk membuat Revisi kedua. Misalkan skenarionya adalah 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, kita sedang belajar Git</p>
  </body>
</html>
```

Setelah itu ketik lagi perintah *git status*.

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (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:   index.html

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

*Melihat git status*

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

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git add index.html

Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git commit -m 'tambah script'
[master 331f605] tambah script
1 file changed, 10 insertions(+)

Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git status
On branch master
nothing to commit, working tree clean
```

*Melihat status commit*

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



*Ilustrasi 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.

#### **4.3.2.1. Melihat Log Revisi.**

Pada skenario 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
```

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git log
commit 331f6058c6ed132af24d5ca1f17a306d5683f9f1 (HEAD -> master)
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:21:05 2018 +0700

    tambah script

commit 689c87477944198456673de50e225aa7f6272703
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:17:06 2018 +0700

    Commit Pertama
```

*Melihat log revisi*

Pada gambar di atas, terdapat dua revisi perubahan yang telah dilakukan. Untuk menampilkan log yang lebih pendek, kita bisa menambahkan argumen `--oneline`.

```
git log --oneline
```

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git log --oneline
331f605 (HEAD -> master) tambah script
689c874 Commit Pertama
```

*Melihat log lebih pendek*

#### 4.3.2.2. Log pada Nomor Revisi/Commit.

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

```
git log 331f6058c6ed132af24d5ca1f17a306d5683f9f1
```

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git log 331f6058c6ed132af24d5ca1f17a306d5683f9f1
commit 331f6058c6ed132af24d5ca1f17a306d5683f9f1 (HEAD -> master)
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:21:05 2018 +0700

    tambah script

commit 689c87477944198456673de50e225aa7f6272703
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:17:06 2018 +0700

    Commit Pertama
```

*Melihat revisi nomor commit*

#### 4.3.2.3. Log pada File Tertentu.

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

```
git log index.html
```

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git log index.html
commit 331f6058c6ed132af24d5ca1f17a306d5683f9f1 (HEAD -> master)
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:21:05 2018 +0700

    tambah script

commit 689c87477944198456673de50e225aa7f6272703
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:17:06 2018 +0700

    Commit Pertama
```

*Melihat log pada file tertentu*

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ git log --author='Irfan Herfiandana'
commit 331f6058c6ed132af24d5ca1f17a306d5683f9f1 (HEAD -> master)
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:21:05 2018 +0700

    tambah script

commit 689c87477944198456673de50e225aa7f6272703
Author: Irfan Herfiandana <IrfanHerfiandana@gmail.com>
Date: Thu Aug 2 17:17:06 2018 +0700

    Commit Pertama
```

*Melihat log author tertentu*

#### 4.3.2.4. **Melihat Perbandingan Perubahan yang Dilakukan pada Revisi.**

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

```
git diff commit_number
```



```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilsy (master)
$ git diff 689c87477944198456673de50e225aa7f6272703
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>
```

*Melihat perbedaan*

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

Sekarang kita akan mencoba merubah isi dari index.html untuk melihat perbedaannya. Kita coba cek lagi bahwa berikut adalah kode original sebelum diubah:

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

Berikut adalah kode setelah diubah :

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

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

Setelah itu lakukan jalankan perintah git diff lagi.

```
Geekseat@DESKTOP-MH44SII MINGW64 /d/Projects/cilisy (master)
$ 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>
```

*Melihat hasil akhir perbedaan*

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

#### 4.3.2.5. **Melihat Perbandingan pada File.**

Apabila 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.

#### 4.3.2.6. **Melihat Perbandingan antar Revisi/Commit.**

Perintah untuk membandingkan perubahan pada revisi dengan revisi yang lain dapat menggunakan perintah dibawah ini.

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

#### 4.3.2.7. 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>
```

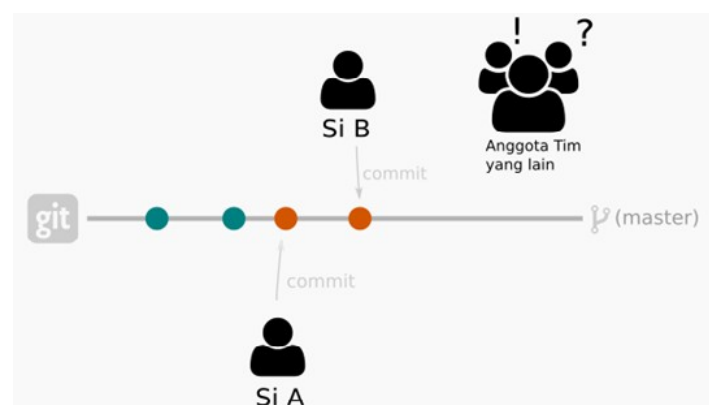
Kita sudah pelajari fungsi dari perintah git diff. Perintah ini untuk melihat perbandingan perubahan apa saja yang telah dilakukan pada repositori.

#### 4.3.3. Exercise

1. Buat sebuah repository Git baru dengan nama Anda !
2. Buat file gitignore pada repository tersebut !
3. Buat beberapa file dan lakukan revisi sebanyak tiga kali !

### 4.4. Branching

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.



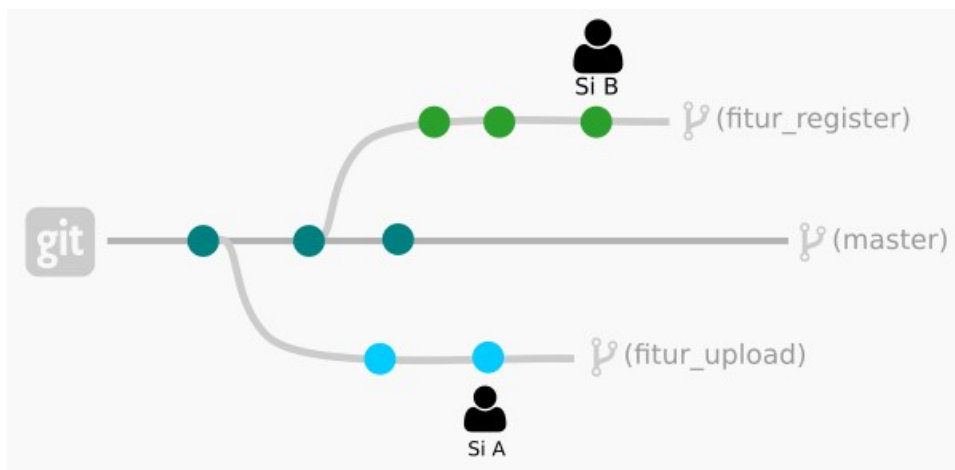
*Ilustrasi kebingungan akibat perbedaan commit*

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).

#### 4.4.1. Membuat Cabang Baru

Perintah untuk membuat cabang adalah `git branch`, kemudian diikuti dengan nama cabangnya. Kita buat sebuah contoh sebagai berikut.

```
git branch fitur_register
```



*Ilustrasi Branch*

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. Untuk memantapkan pemahaman tentang percabangan Git, mari kita coba praktek. 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. Jangan lupa untuk menggunakan perintah git status untuk melihat status repositori. Setelah kita menambahkan file login.html, Selanjutnya kita lakukan commit.

```
git add login.html
```

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

Revisi kita pada cabang halaman\_login sudah disimpan. Sekarang coba kembali ke cabang master.

```
git checkout master
```

Apakah anda menemukan file login.html? Pasti tidak! Sekarang kembali lagi ke cabang halaman\_login.

```
git checkout halaman_login
```

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

```
project-01/
```

```
|— index.html
```

```
|— login.html
```

## 4.4.2. 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.

*Ilustrasi penggabungan cabang*

### 4.4.3. Mengatasi Adanya 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 repostori. Dia harus bertindak adil, kode mana yang harus diambil. Biasanya akan ada proses diskusi dulu dalam mengambil keputusan. Sekarang kita akan coba membuat bentrokan.

Pertama kita pindah dulu ke branch halaman\_login.

```
git checkout halaman_login
```

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

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

Setelah itu, lakukan commit lagi:

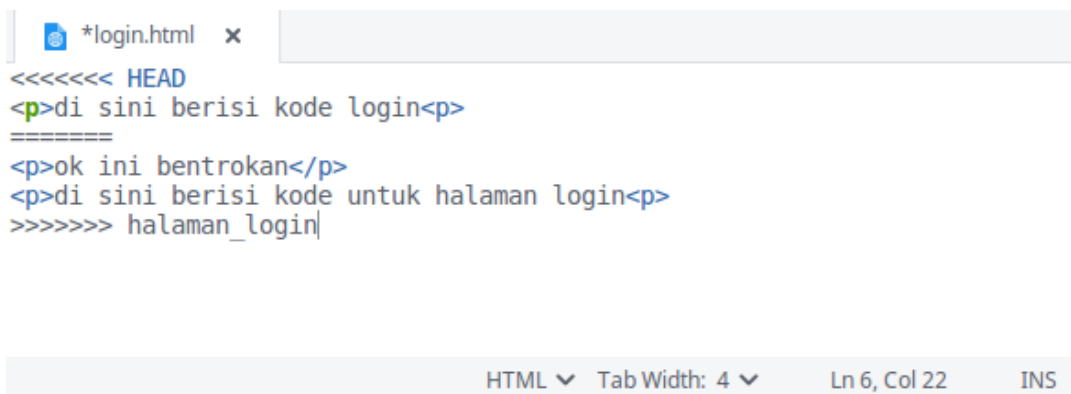
```
git add login.html
git commit -m "ubah isi login.html"
```

Selanjutnya pindah ke cabang master dan lakukan perubahan juga di cabang ini. Ubah file yang sama seperti di cabang halaman\_login, setelah itu, lakukan commit di cabang master.

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

Terakhir, coba gabungkan cabang halaman\_login dengan cabang master, maka akan terjadi bentrok.

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



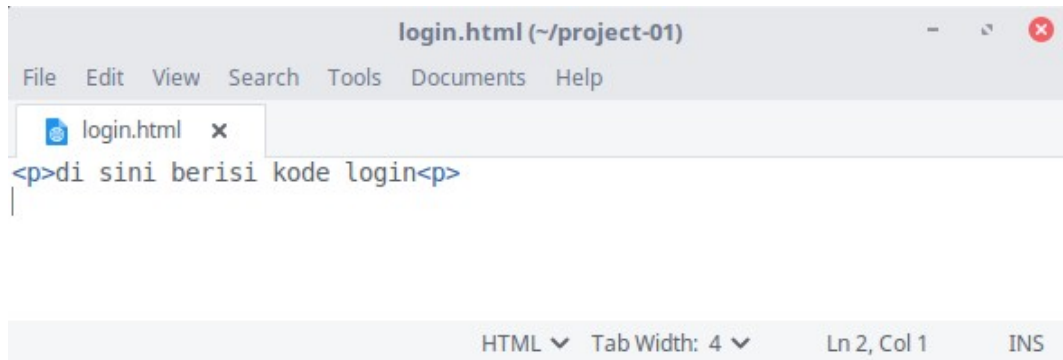
```

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

*Error Merge 1*

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

Kedua kode cabang dipisahkan dengan tanda =====. Sekarang.. tugas kita adalah memperbaikinya. Silahkan eliminasi salah satu dari kode tersebut.



*Error Merge 2*

Setelah itu lakukan commit untuk menyimpan perubahan ini.

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

#### 4.4.4. Menghapus Cabang

Cabang yang sudah mati atau tidak ada pengembangan lagi, sebaiknya dihapus. Agar repository 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
```

#### 4.4.5. Exercise

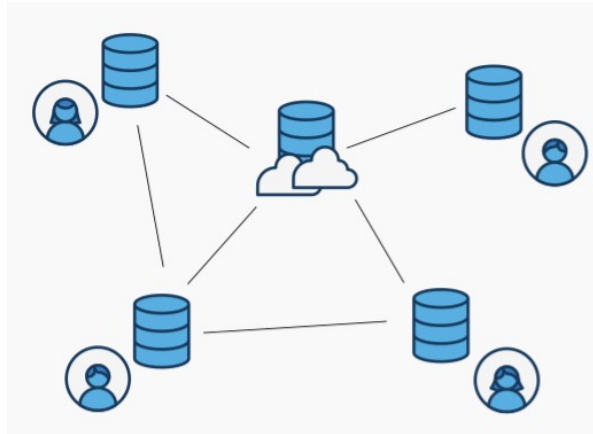
1. Buat dua cabang baru pada repository yang Anda buat.
2. Buat source code pada kedua cabang tersebut dan gabungkan keduanya.

### 4.5. Remote Repository

Pada proyek pengembangan software yang melibatkan banyak orang (tim), kita tidak hanya akan menyimpan sendiri repository proyeknya. Semua tim yang terlibat dalam pengkodean (coding) akan menyimpan repository lokal di komputernya masing-masing. Setelah itu, akan dilakukan penggabungan ke



repository inti atau remote. Biasanya akan ada repository pusat atau untuk menyimpan source code yang sudah digabungkan (merge) dari beberapa orang.



*Ilustrasi remote repository*

Di mana menyimpan repository remote-nya? Bisa di server kantor atau bisa juga menggunakan layanan seperti Github, Gitlab, Bitbucket, dll. Github adalah layanan yang paling populer untuk menyimpan (hosting) repository secara remote. Banyak proyek open source tersimpan di sana. Kita akan menggunakan Github pada tutorial ini, pastikan Anda sudah memiliki akun Github dengan cara mendaftar di **github.com**.

#### **4.5.1. Membuat Repositori di GitHub**

Silahkan buka Github, kemudian buat sebuah repository dengan nama belajar-git seperti berikut ini.

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

Repository name

belajar-git ✓

Great repository names are short and memorable. Need inspiration? How about **solid-lamp**.

Description (optional)

Belajar Git

☒ Public

Anyone can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

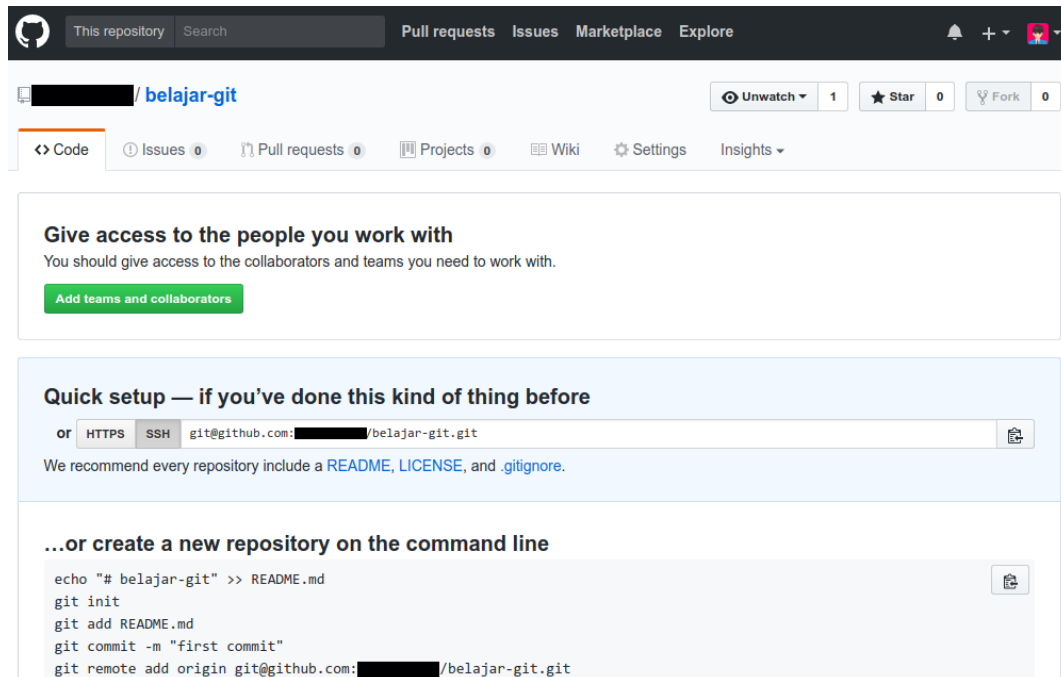
Add .gitignore: None ▾

Add a license: None ▾ ⓘ

Create repository

*Membuat repository 1*

Maka sekarang kita punya repository kosong bernama belajar-git di Github.

*Membuat repository 2*

Silahkan buka kembali repository lokal yang pernah kita buat, yaitu **cilsy**. Berikutnya kita akan coba upload repository lokal ini ke repository remote Github.

### 4.5.2. Menambah Remote Repository

Sebelum kita bisa upload semua revisi yang ada di repository lokal, kita harus menambahkan remote repository-nya terlebih dahulu. Remote repository dapat kita tambahkan dengan perintah seperti ini :

```
git remote add github https://github.com/username/belajar-git.git  
git remote add github git@github.com:username/belajar-git.git
```

Perbedaan https dengan SSH adalah bantu autentikasinya. Untuk https, kita akan diminta user dan password setiap kali melakukan push, sedangkan SSH, kita hanya melakukan 1 kali autentikasi, yaitu dengan mendaftarkan public key kita ke repository.

### 4.5.3. Menggunakan SSH di GitHub

SSH memungkinkan kita untuk melakukan push ke repository github tanpa login. Berbeda dengan cara yang biasa (melalui HTTPS), kita harus memasukkan username dan password setiap kali melakukan push. Tapi dengan SSH kita tidak akan melakukan itu lagi. Berikut adalah langkah-langkahnya.

#### 4.5.3.1. Membuat SSH Key.

Pertama-tama kita akan membuat sebuah SSH Key. SSH Key ini adalah sebuah kombinasi 2 file terenkripsi (publik dan private) yang akan dicocokkan antara di server Remote repository dengan di lokal (pc/laptop). Ketika 2 file ini dinyatakan cocok, maka kita dianggap sebagai orang yang punya autentikasi tanpa perlu memasukkan password dan username lagi. Cara untuk membuat ssh key adalah masuk ke terminal dan ketikkan ssh-keygen lalu enter. Untuk passphrase dikosongkan saja.

```
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~/.ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa): github
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in github.
Your public key has been saved in github.pub.
The key fingerprint is:
SHA256:QNsRHqJKQnM5a/7DHZr3J7PsM+3p486AjDS9HxF5/s8 ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01
The key's randomart image is:
+---[RSA 2048]---+
| o .. o +. |
| . oo o = o . |
| . . + o o o . |
| o + .. + |
| + =So = o |
| . o * = * . |
| o o + =. . |
| + . oo+o... |
| o. oBB. E |
+---[SHA256]-----+
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~/.ssh$ |
```

#### Membuat SSH-Key

Maka di dalam directory .ssh akan tercipta file baru yaitu id\_rsa sebagai private key dan id\_rsa.pub sebagai public key.

```
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~/.ssh$ ls
authorized_keys  github  github.pub  id_rsa  id_rsa.pub
ubuntu@ubuntu-s-1vcpu-2gb-sgp1-01:~/.ssh$ |
```

*Public key dan private key*

#### 4.5.3.2. Jalankan SSH Agent dan Load SSH Key

Untuk memastikan apakah SSH Agent sudah berjalan atau tidak, gunakan perintah ini:

```
ps -e | grep [s]sh-agent
```

Kalau belum berjalan, gunakan perintah berikut ini untuk menjalankan SSH agent:

```
ssh-agent /bin/bash
```

Berikutnya kita Load SSH Key. Gunakan perintah:

```
ssh-add ~/.ssh/id_rsa
```

Kemudian untuk mengecek, gunakan perintah:

```
ssh-add -l
```

Tambahkan SSH Key ke Github. Sebelumnya ambil dulu publik key yang sudah anda buat, gunakan perintah cat.

```
cat ~/.ssh/id_rsa.pub
```

Copy isi teks yang ditampilkan. Lalu kembali ke Github, masuk ke menu **Settings> SSH and GPG Keys**, buat key baru dengan mengklik **New SSH Key**. Lalu masukkan key yang sudah dicopy.

Personal settings
Profile
Account
Emails
Notifications
Billing
SSH and GPG keys
Security
Blocked users
Repositories
Organizations
Saved replies
Authorized applications
Installed integrations

## SSH keys

New SSH key

There are no SSH keys with access to your account.

Title

Key

ssh-rsa  
AAAAB3NzaC1yc2EAAAADAQABAAQCUaL5/+rjOLgDZ2ZWH88WNIFQYm3Yt7eYJcoQ/XCWp3SXuq/t  
mterPYymEPWkpOaoCUI/NYxkJKaSYrYIREvZy9nwBK8e6xd8mFa7ySfOn16NJ01cXxn5BMP/14geM9vDv0HI  
N8GRAC95NJicur043WSFK05NCE5YbCOBp/ZKaj3NtMdum5SUW17XN6p+/HvFLNJ0UN4oXECI46mkMZt0  
UeIMxv+Q5Ara7V6ano0h4rHu3EpCsg6vGh5/5oh52rYVWVNu5+4bAnb8CuNSpmmsfypsY20FobUGvvhavXz+C  
xq2UQ9i0HmbIPj0rD+4zQuVStrsE+16pot6beW13BG4+x ardianta

Add SSH key

Check out our guide to [generating SSH keys](#) or [troubleshoot common SSH Problems](#).

### Meng-input SSH Key di Github

Sekarang seharusnya laptop Anda sudah terhubung dengan remote repository dengan metode SSH.

#### 4.5.3.3. Uji Konektivitas

Ketik perintah berikut untuk menguji konektivitas SSH ke Github :

```
ssh -T git@github.com
```

Pastikan tidak ada pesan error yang muncul untuk memastikan bahwa konektivitas Github dengan SSH sudah berhasil.

#### 4.5.3.4. Mengubah dan Menghapus Remote Repository

Silahkan ketik perintah `git remote -v` untuk melihat remote apa saja yang sudah ditambahkan.

```

i ~/project-01 $ git remote
i ~/project-01 $ git remote add github git@github.com: username /belajar-git.git
i ~/project-01 $ git remote -v
github git@github.com: username /belajar-git.git (fetch)
github git@github.com: username /belajar-git.git (push)
i ~/project-01 $ git remote
github

```

### Mengubah dan menghapus remote repository

Sekarang kita sudah menambahkan remote di dalam repository lokal. Selanjutnya kita bisa melakukan push atau mengirim revisi ke repository

remote (Github). Nah untuk menghapus dan mengubah nama remote dapat dilakukan dengan perintah berikut.

Ubah nama remote:

```
git remote rename github kantor
```

*Keterangan : github adalah nama remote yang lama, kantor adalah nama remote yang baru.*

Hapus remote:

```
git remove github
```

#### 4.5.4. Mengirim Revisi ke Remote Repository

Perintah yang kita gunakan untuk mengirim revisi ke repository remote adalah git push.

```
git push github master
```

Keterangan: github adalah nama remote, master adalah nama branch tujuan.

Mari kita coba, pastikan repository lokal kita sudah memiliki remote.

```
i ~/project-01 $ git remote
i ~/project-01 $ git remote add github git@github.com: username /belajar-git.git
i ~/project-01 $ git remote -v
github git@github.com: username /belajar-git.git (fetch)
github git@github.com: username /belajar-git.git (push)
i ~/project-01 $ git remote
github
```

*Mengirim revisi*

Setelah itu lakukan beberapa revisi atau commit.

```
git add .
```

```
git commit -m "menambahkan beberapa revisi"
```

Sebagai contoh, disini ada 5 catatan revisi.

```
i ~/project-01 $ git log --oneline
865e50c commit hasil revert
023b078 ditambahkan login.html
6cdfdbb ada perubahan
06f735a ditambahkan isi
cf08ca0 commit pertama
```

*Menampilkan catatan revisi*

Maka tinggal kita kirim saja dengan perintah git push github master. Jika muncul seperti ini, artinya push sukses dilakukan.

```
i ~/project-01 $ git push github master
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 1.16 KiB | 0 bytes/s, done.
Total 13 (delta 5), reused 0 (delta 0)
remote: Resolving deltas: 100% (5/5), done.
To git@github.com:petanikode/belajar-git
* [new branch]      master -> master
```

*Melakukan push*

Sekarang lihat ke Github, pasti semuanya sudah ter-upload ke sana.

The screenshot shows the GitHub interface for a repository named 'belajar-git' by a user 'username'. The repository has 5 commits, 1 branch, 0 releases, and 1 contributor. The 'Code' tab is selected, showing a list of files: 'about.html', 'contact.html', 'index.html', and 'login.html'. The 'index.html' file is highlighted, showing its commit history: 'commit pertama' (8 months ago), 'commit hasil revert' (7 months ago), and 'ditambahkan login.html' (7 months ago). There is a button 'Add a README' at the bottom.

*Memeriksa hasil push*

Coba buat revisi lagi di file index.html. Misalnya perubahannya seperti ini:



```
i ~/project-01 $ git diff index.html
diff --git a/index.html b/index.html
index 481cc02..2bca54f 100644
--- a/index.html
+++ b/index.html
@@ -2,9 +2,11 @@
<html>
  <head>
    <meta charset="utf-8">
-    <title>Belajar Git - Project 01</title>
+    <title>Belajar Git: Remote Repository</title>
  </head>
  <body>
    <p>Hello Semua, Saya sedang belajar Git</p>
+
+    <p>Yay! saya sedang belajar tentang remote repository</p>
  </body>
</html>
```

*Melakukan perubahan file index.html*

Lalu lakukan commit dan push.

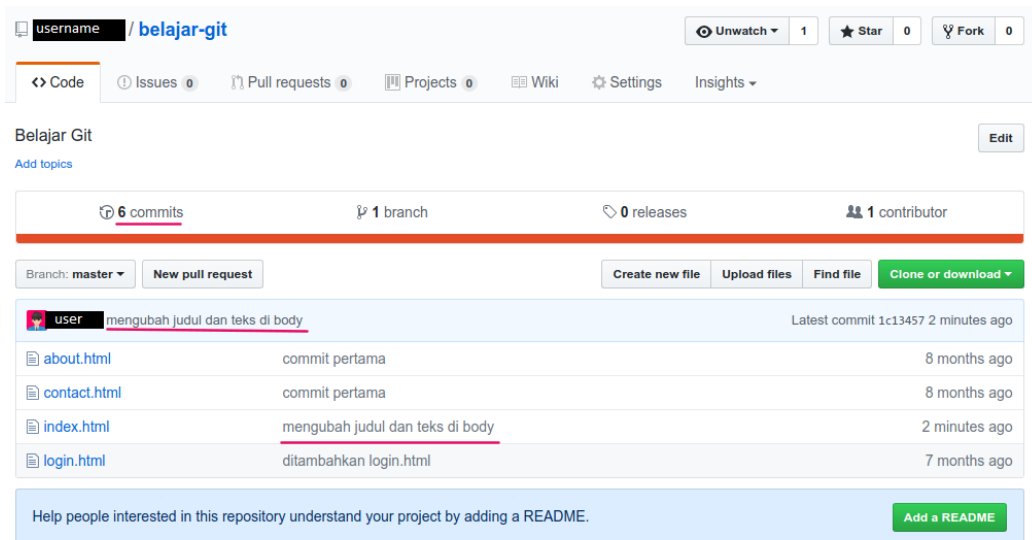
```
git add index.html
git commit -m "mengubah judul dan teks di body"
git push github master
```

Jika berhasil, maka akan tampil seperti ini:

```
i ~/project-01 $
i ~/project-01 $ git add index.html
i ~/project-01 $ git commit -m "mengubah judul dan teks di body"
[master 1c13457] mengubah judul dan teks di body
1 file changed, 3 insertions(+), 1 deletion(-)
i ~/project-01 $ git push github master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 380 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To git@github.com:petanikode/belajar-git.git
865e50c..1c13457 master -> master
```

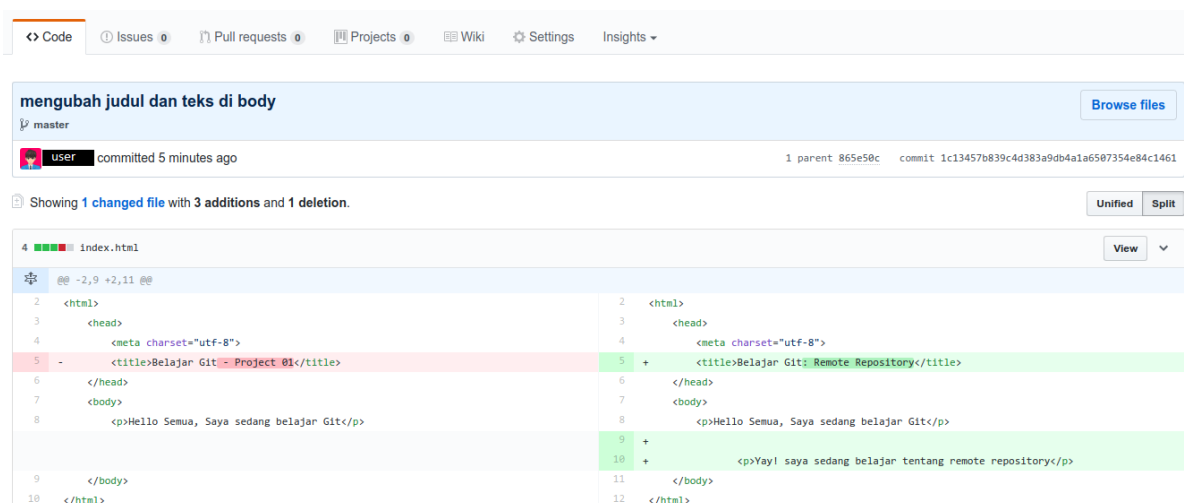
*Hasil push perubahan*

Periksa kembali repository di Github dan perhatikanlah perubahannya.



*Melihat hasil di Github*

Jika kita klik commit terakhir, maka kita akan dibawa ke *git diff*-nya Github. Di sana kita bisa melihat perubahan apa kita yang dilakukan pada commit tersebut.

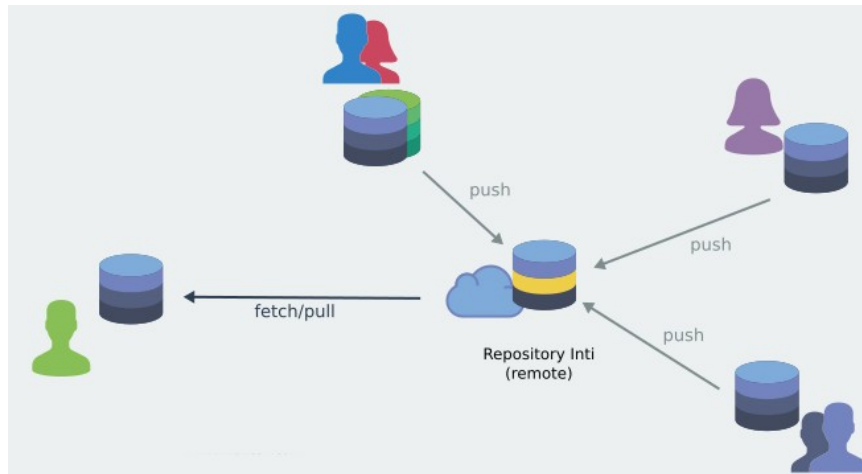


*Melihat perbedaan di Github*

### 4.5.5. Mengambil Revisi dan Remote Repository

Saat kita bekerja dengan repository yang memiliki banyak kontributor, kita seharusnya mengambil dulu revisi terbaru dari repository inti agar tidak bentrok. Misalnya begini, pada repository remote ada kontributor lain yang

sudah menambahkan dan merubah sesuatu di sana. Maka kita harus mengambil perubahan tersebut, agar repository lokal kita tetap ter-update atau sama persis seperti repository remote.



*Mengambil revisi*

Ada dua perintah untuk mengambil revisi dari repository remote:

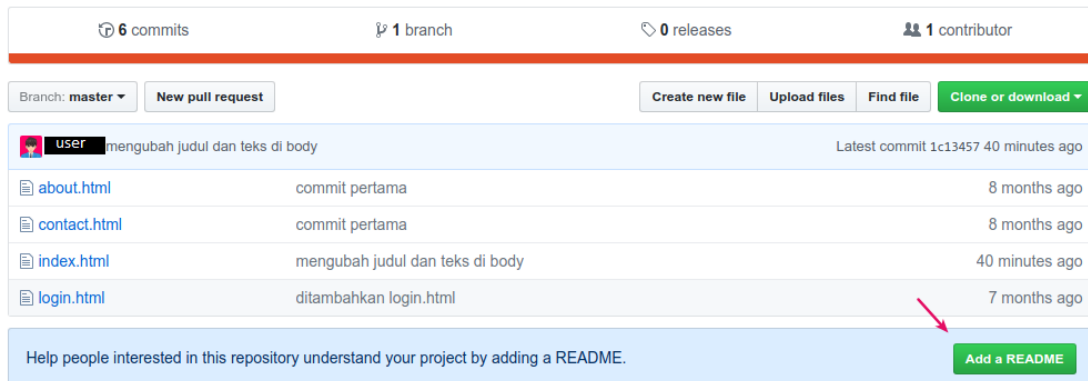
```
git fetch [nama remote] [nama cabang]
git pull [nama remote] [nama cabang]
```

Perbedaan pada keduanya adalah perintah git fetch hanya akan mengambil revisi (commit) saja dan tidak langsung melakukan penggabungan (merge) terhadap repository lokal. Sedangkan git pull akan mengambil revisi (commit) dan langsung melakukan penggabungan (merge) terhadap repository lokal.

Pemakaian salah satu keduanya tergantung dari situasi dan kondisi. Bila kita sudah membuat perubahan di repository lokal, maka sebaiknya menggunakan git fetch agar perubahan yang kita lakukan tidak hilang. Namun, bila kita tidak pernah melakukan perubahan apapun dan ingin mengambil versi terakhir dari repository remote, maka gunakanlah git pull.

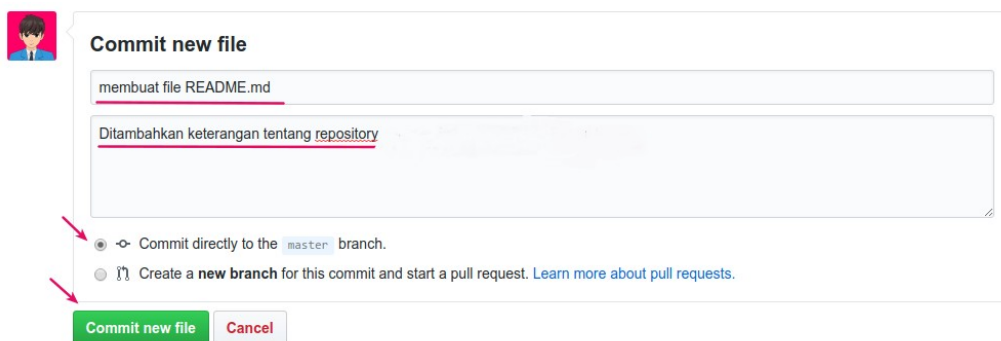
#### 4.5.6. Mengambil Revisi dengan Git Fetch

Sekarang mari kita coba praktekan. Silahkan buka github, dan tambahkan file README.md melalui Github. Klik tombol add README.



### Mengambil revisi dengan git fetch

Setelah itu, isilah file README.md dengan apapun yang Anda inginkan. Setelah selesai, simpan perubahan dengan melakukan commit langsung dari Github.



### Membuat file readme

Pesan commit bersifat opsional, boleh di isi boleh tidak. Karena Github akan membuatnya secara otomatis. Sekarang ada perubahan baru di repository remote dan kita akan mengambil perubahan tersebut. Mari kita lakukan dengan perintah *git fetch*.

```
i ~/project-01 $ git fetch github master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:petanikode/belajar-git
* branch          master       -> FETCH HEAD
  1c13457..3951f15 master       -> github/master
i ~/project-01 $ ls
about.html  contact.html  index.html  login.html
```

### Melakukan git fetch

Revisi sudah diambil, tapi belum ada file README.md di dalam repository lokal. Kenapa bisa begitu? Ya, balik lagi dari pengertian *git fetch*. Dia hanya bertugas mengambil revisi saja dan tidak langsung menggabungkannya dengan repository lokal. Coba kita cek dengan git log.

```
i ~/project-01 $ git log --oneline
1c13457 mengubah judul dan teks di body
865e50c commit hasil revert
023b078 ditambahkan login.html
6cdfdbb ada perubahan
06f735a ditambahkan isi
cf08ca0 commit pertama

i ~/project-01 $ git log github/master --oneline
3951f15 membuat file README.md
1c13457 mengubah judul dan teks di body
865e50c commit hasil revert
023b078 ditambahkan login.html
6cdfdbb ada perubahan
06f735a ditambahkan isi
cf08ca0 commit pertama
```

*Pengecekan dengan git log*

Pada gambar di atas terlihat perbedaan log antara repository lokal dengan repository remote. Bila ingin mengecek apa saja perbedaannya, coba gunakan perintah git diff.

```
git diff master github/master
```

Keterangan: master adalah cabang master di repository lokal, github/master adalah cabang master di repository remote. Hasil outputnya kira-kira akan seperti ini:

```
diff --git a/README.md b/README.md
new file mode 100644
index 0000000..1174eb2
--- /dev/null
+++ b/README.md
@@ -0,0 +1,18 @@
+# belajar-git
+
+Repository ini adalah repository untuk belajar Git.
```

Lalu sekarang bagaimana cara kita menggabungkan commit dari repository remote dengan lokal? Gunakan perintah git merge.

```
git merge master github/master
```

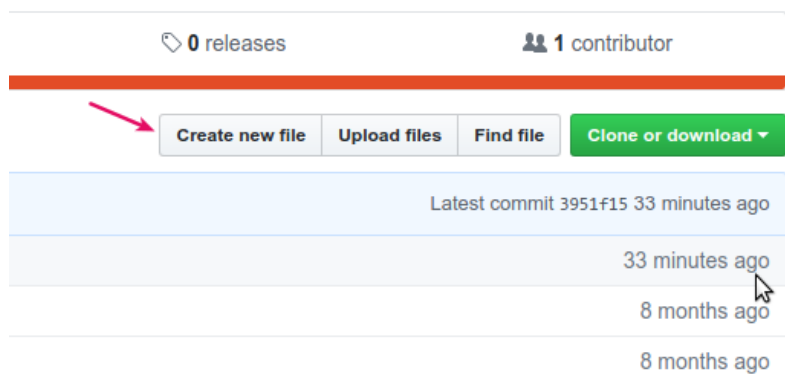
Setelah itu coba ketik ls dan git log lagi, maka kita sudah berhasil menggabungkan revisi dari remote dan lokal.

```
i ~/project-01 $ git merge master github/master
Updating 1c13457..3951f15
Fast-forward
 README.md | 18 ++++++
 1 file changed, 18 insertions(+)
 create mode 100644 README.md
i ~/project-01 $ ls
about.html contact.html index.html login.html README.md
i ~/project-01 $ git log --oneline
3951f15 membuat file README.md
1c13457 mengubah judul dan teks di body
865e50c commit hasil revert
023b078 ditambahkan login.html
6cdfdbb ada perubahan
06f735a ditambahkan isi
cf08ca0 commit pertama
```

*Melihat hasil gabungan revisi remote dan lokal*

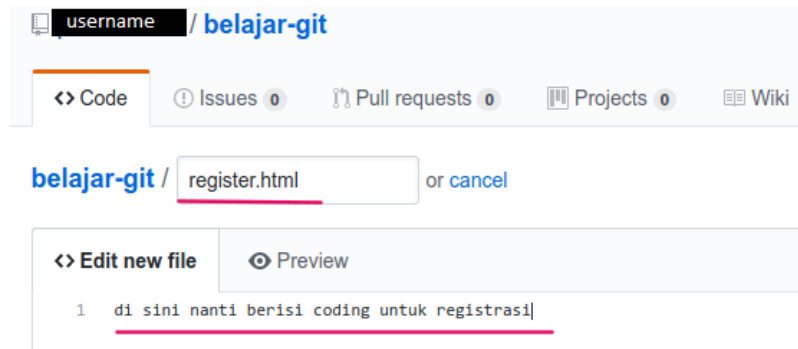
#### 4.5.7. Mengambil Revisi dengan Git Pull

Lakukan hal yang sama seperti tadi. Kali ini kita akan membuat file baru bernama register.html melalui Github.



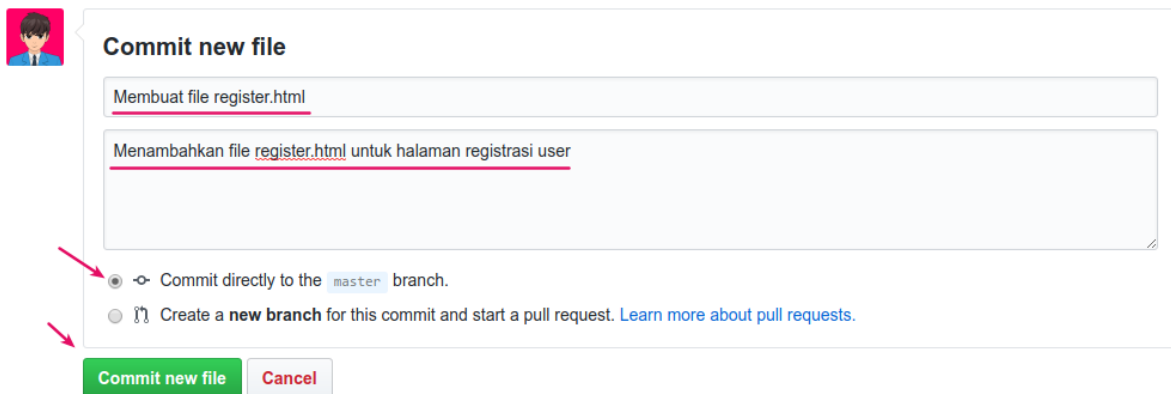
*Membuat file baru*

Berikan nama file dengan register.html dan isi dengan apa saja.



*Mengisi file html*

Simpan revisi dan tambahkan commit seperti ini.



*Melakukan commit file baru*

Sekarang ada perubahan baru di repository remote dan kita akan mengambilnya dengan perintah git pull. Silahkan buka repository lokal dan ketik perintah berikut:

```
git pull github master
```

Maka semua revisi akan diambil dan langsung digabungkan (merge).

```
i ~/project-01 $ git pull github master
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:petanikode/belajar-git
* branch          master      -> FETCH_HEAD
   3951f15..e4dbc35 master      -> github/master
Updating 3951f15..e4dbc35
Fast-forward
   register.html | 1 +
   1 file changed, 1 insertion(+)
   create mode 100644 register.html
i ~/project-01 $ ls
about.html  contact.html  index.html  login.html  README.md  register.html
i ~/project-01 $ git log --oneline
e4dbc35 Membuat file register.html ←
3951f15 membuat file README.md
1c13457 mengubah judul dan teks di body
865e50c commit hasil revert
023b078 ditambahkan login.html
6cdfdbb ada perubahan
06f735a ditambahkan isi
cf08ca0 commit pertama
```

*Pengecekan hasil git pull*

### 4.5.8. Clone Remote Repository

Clone repository bisa kita bilang seperti copy repository dari remote ke lokal. Perintah untuk melakukan clone adalah git clone.

```
git clone https://github.com/username/belajar-git.git [nama dir]
```

Keterangan: https://... adalah URL repository remote, kita juga bisa menggunakan SSH. [nama dir] (opsional) adalah nama direktori yang akan dibuat. Jika kita tidak berikan nama direktori, maka akan otomatis menggunakan nama repository. Mari kita coba. Sekarang kita akan pindah ke direktori Desktop.

```
Cd ~/Desktop
```

Lalu clone remote repo:

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

Maka akan ada direktori baru di sana.

FYI: Saat Anda clone sebuah repository dari Github, nama remote origin akan diberikan secara otomatis.



### 4.5.9. Exercise

1. Buat sebuah akun GitHub.
2. Push repository local ke github
3. Buat clone dari repository yang sudah di push ke GitHub atau GitLab

## 4.6. Summary

Pada bab 4 ini Anda seharusnya sudah memiliki skill dan pengetahuan yang cukup terkait Git dan terbiasa menggunakannya. Dimana Git digunakan sebagai tools pengontrol versi yang pasti digunakan oleh para programmer di perusahaan Anda nanti bekerja. Mereka akan saling menyimpan, menyalin, mengambil kode-kode program mereka menggunakan Git ini.