

# MODUL 1 – Mengelola versi file dengan git

## 1.1. CAPAIAN PEMBELAJARAN

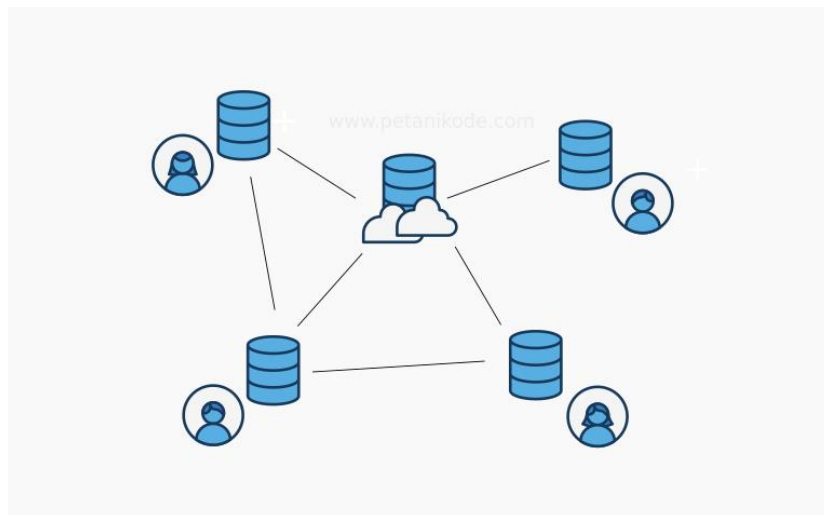
1. Mahasiswa mampu menggunakan git
2. Mahasiswa mampu menerapkan git

## 1.2. ALAT DAN BAHAN

1. Seperangkat komputer lengkap/Laptop dengan koneksi internet
2. Web Browser (Chrome/Firefox/Opera/Edge/Safari/dll)
3. Aplikasi Kantor (Microsoft Office/Libre Office/WPS Office/etc)

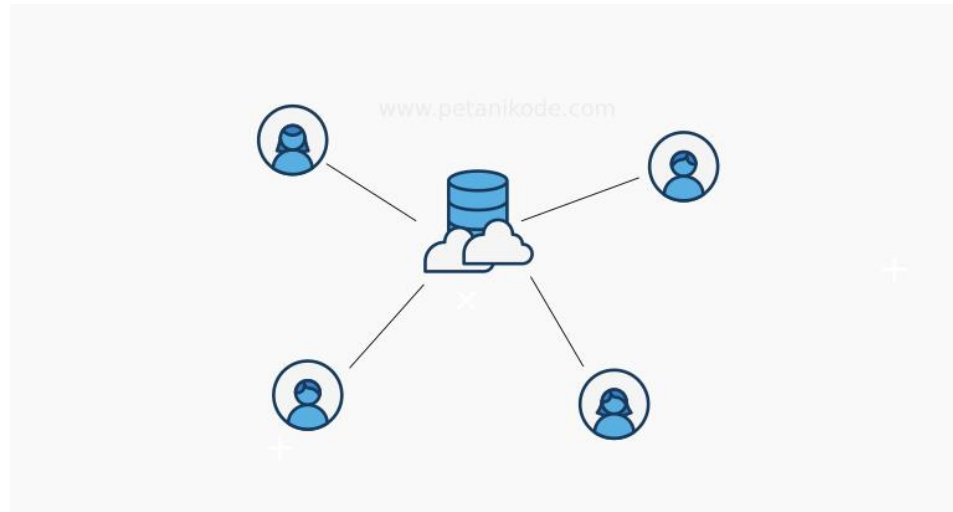
## 1.3. DASAR TEORI

Git adalah salah satu tool yang sering digunakan dalam proyek pengembangan software. Git bahkan menjadi tool yang wajib dipahami oleh programmer, karena banyak digunakan di mana-mana. Git adalah salah satu sistem pengontrol versi (Version Control System) pada proyek perangkat lunak yang diciptakan oleh Linus Torvalds. Pengontrol versi bertugas mencatat setiap perubahan pada file proyek yang dikerjakan oleh banyak orang maupun sendiri. Git dikenal juga dengan distributed revision control (VCS terdistribusi), artinya penyimpanan database Git tidak hanya berada dalam satu tempat saja.



Gambar 1. VCS terdistribusi

Semua orang yang terlibat dalam pengkodean proyek akan menyimpan database Git, sehingga akan memudahkan dalam mengelola proyek baik online maupun offline. Dalam Git terdapat merge untuk menyebut aktifitas penggabungan kode. Sedangkan pada VCS (Version Control System) yang terpusa, database disimpan dalam satu tempat dan setiap perubahan disimpan ke sana.



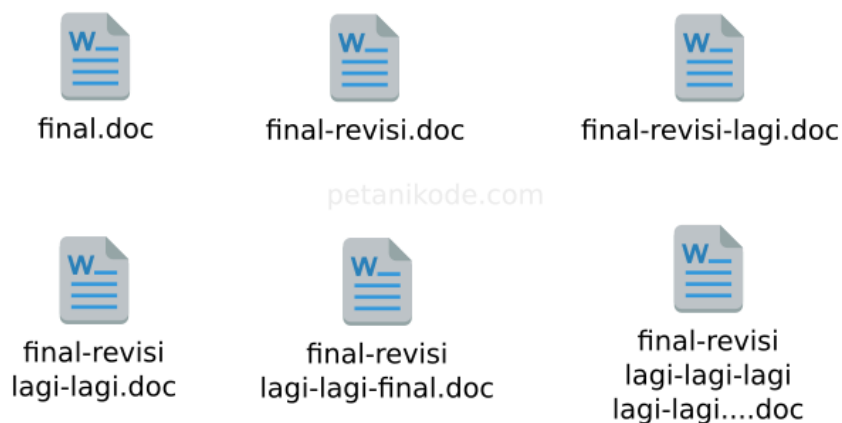
Gambar 2. VCS terpusat

VCS terpusat memiliki beberapa kekurangan:

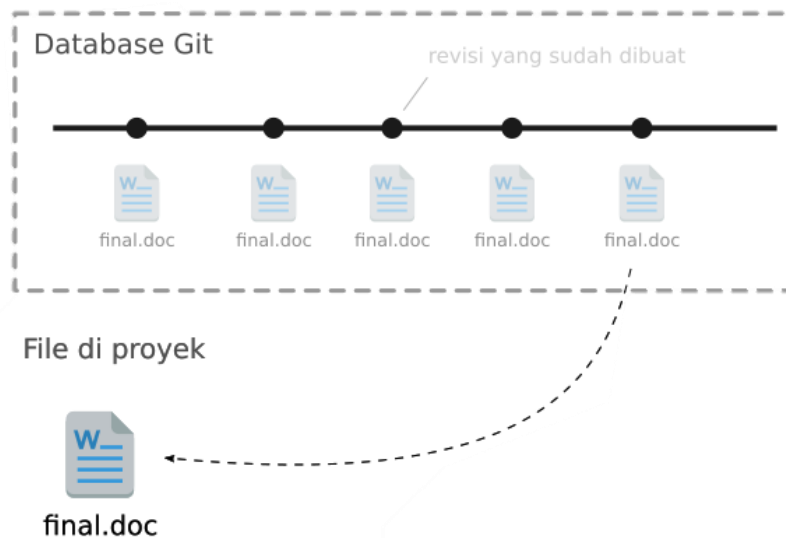
- Semua tim harus terkoneksi ke jaringan untuk mengakses source-code;
- Tersimpan di satu tempat, nanti kalau server bermasalah bagaimana?

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

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



Gambar 3. Kondisi File sebelum dikelola git



Gambar 4. Kondisi file setelah dikelola git

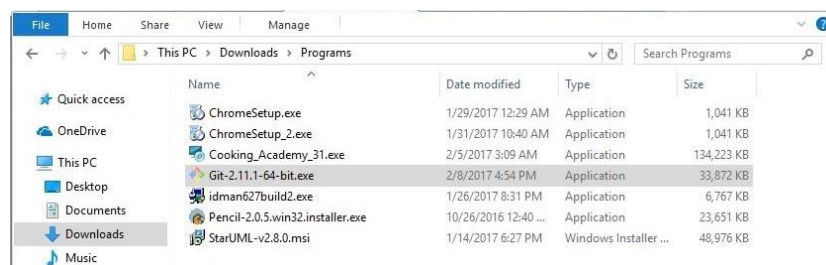
Saat kita ingin menyimpan semua perubahan pada file, biasanya kita membuat file baru dengan “save as”. Lalu, file akan menumpuk dalam direktori proyek seperti pada ilustrasi di atas. Setelah menggunakan git Hanya akan ada satu file dalam proyek dan perubahannya disimpan dalam database. Git hanya akan menyimpan delta perubahannya saja, dia tidak akan menyimpan seluruh isi file yang akan memakan banyak memori. Git memungkinkan kita kembali ke versi revisi yang kita inginkan.

## 1.4. PRAKTIKUM

### 1.4.1. Instalasi Git

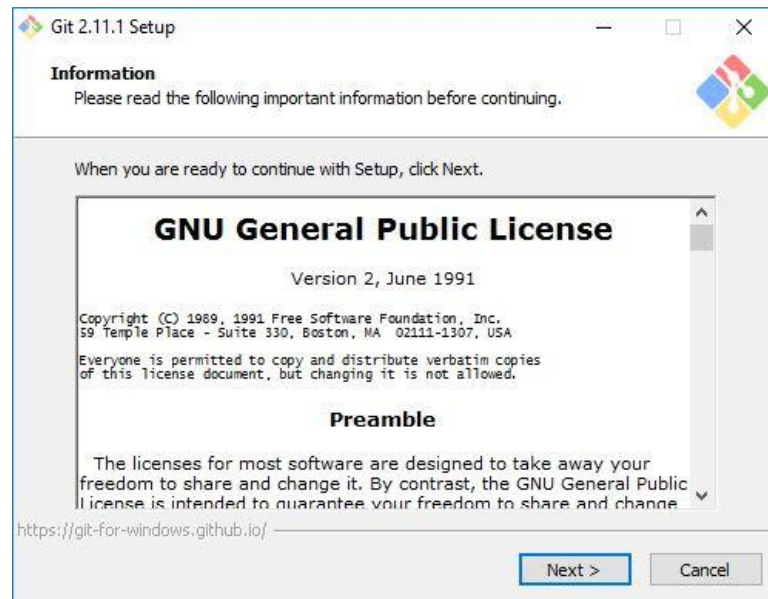
Silakan buka website resminya Git ( [git-scm.com](https://git-scm.com)). Kemudian unduh Git sesuai dengan arsitektur komputer kita. Kalau menggunakan 64bit, unduh yang 64bit. Begitu juga kalau menggunakan 32bit.

Silakan klik 2x file instaler Git yang sudah diunduh.



Gambar 5. File Installer Git

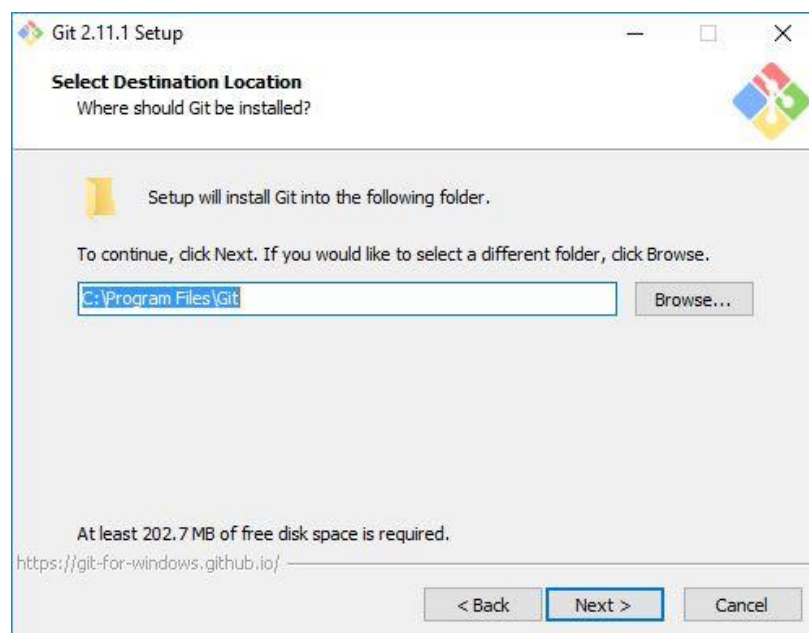
Maka akan muncul informasi lisensi Git, klik Next > untuk melanjutkan.



*Gambar 6. Lisensi*

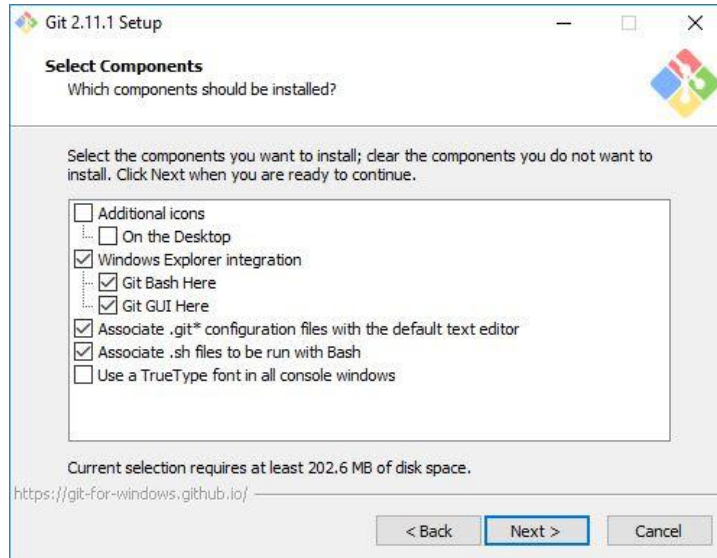
Selanjutnya menentukan lokasi instalasi. Biarkan saja apa adanya, kemudian klik Next

>.



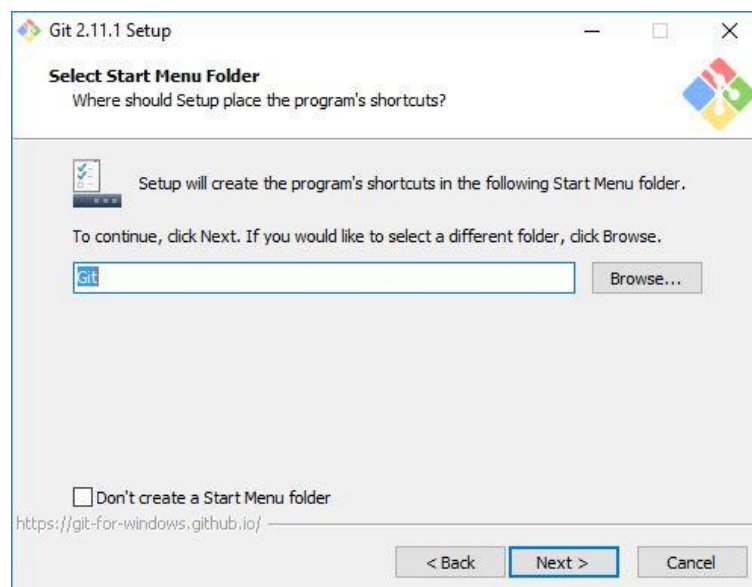
*Gambar 7. Penentuan Lokasi Instalasi*

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



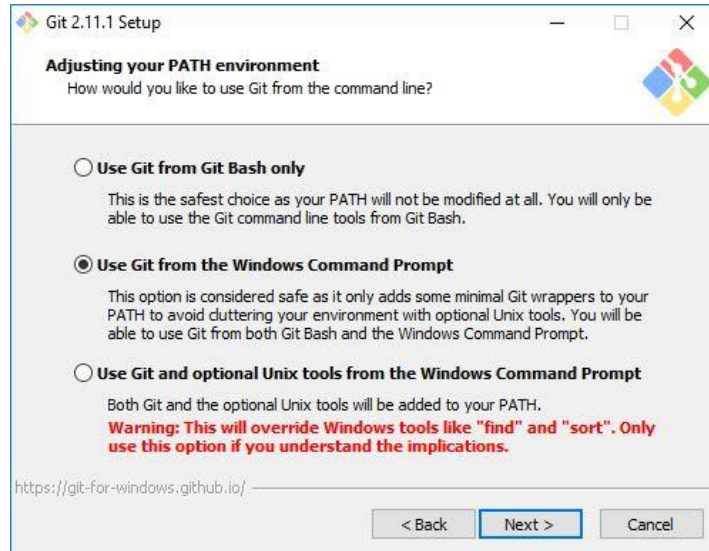
*Gambar 8. Pilihan Komponen Isntallasi*

Selanjutnya pemilihan direktori start menu, klik Next >.



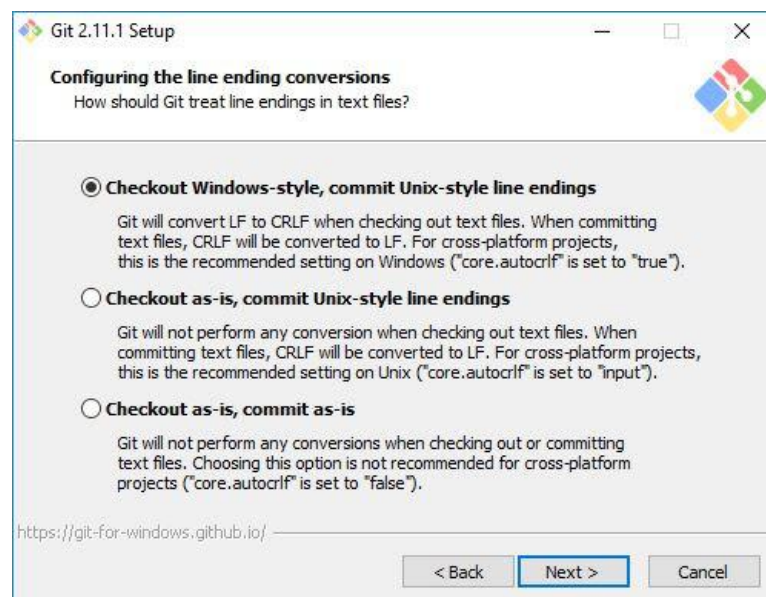
*Gambar 9. Pilihan direktori*

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



Gambar 10. Pengaturan PATH

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



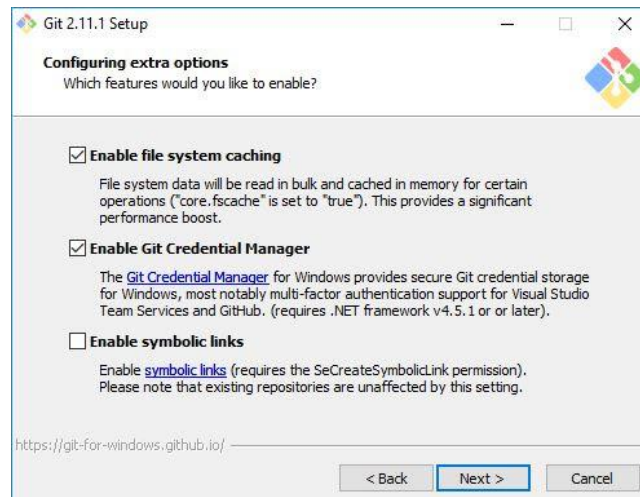
Gambar 11. Konversi Line Ending

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



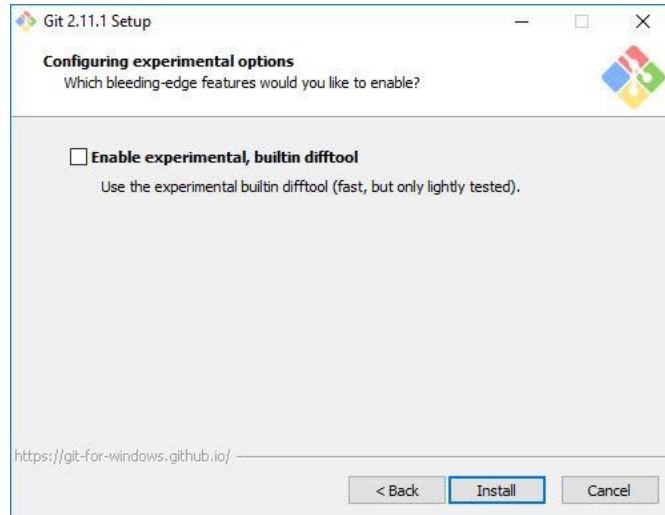
*Gambar 12. Pemilihan emulator terminal*

Selanjutnya pemilihan opsi ekstra. Klik saja Next >.



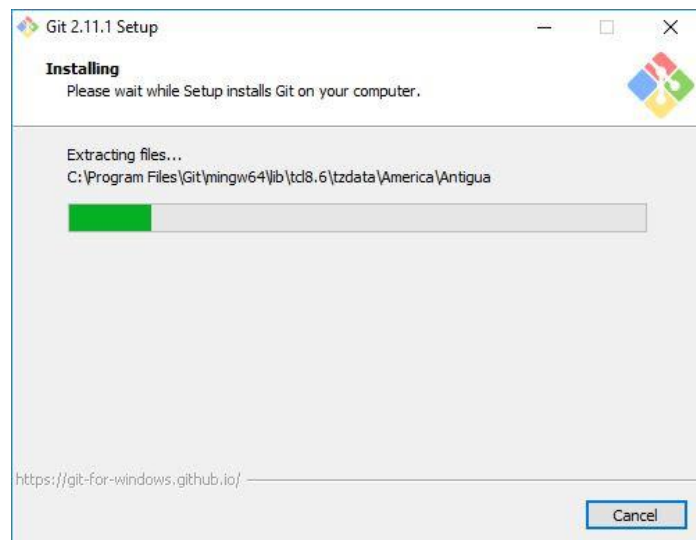
*Gambar 13. Pemilihan opsi extra*

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



*Gambar 14. Pilihan eksperimental*

Tunggu beberapa saat, instalasi sedang dilakukan.



*Gambar 15. Proses Instalasi*

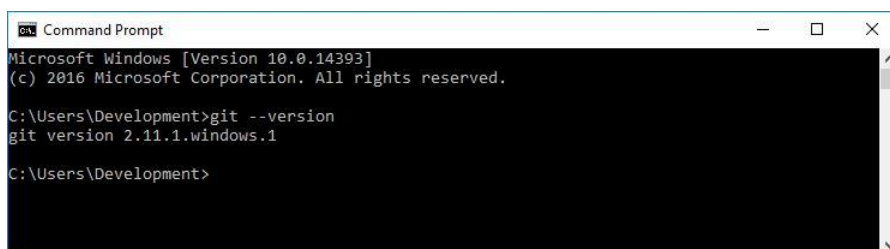
Setelah selesai, kita bisa langsung klik Finish.





*Gambar 16. Proses Instalasi Berhasil*

Untuk mencobanya, silakan buka CMD atau PowerShell, kemudian ketik perintah git --version.



*Gambar 17. Tampilan CLI git*

#### **1.4.2. Konfigurasi Awal**

Ada beberapa konfigurasi yang harus dilakukan sebelum mulai menggunakan Git, seperti menentukan name dan email.

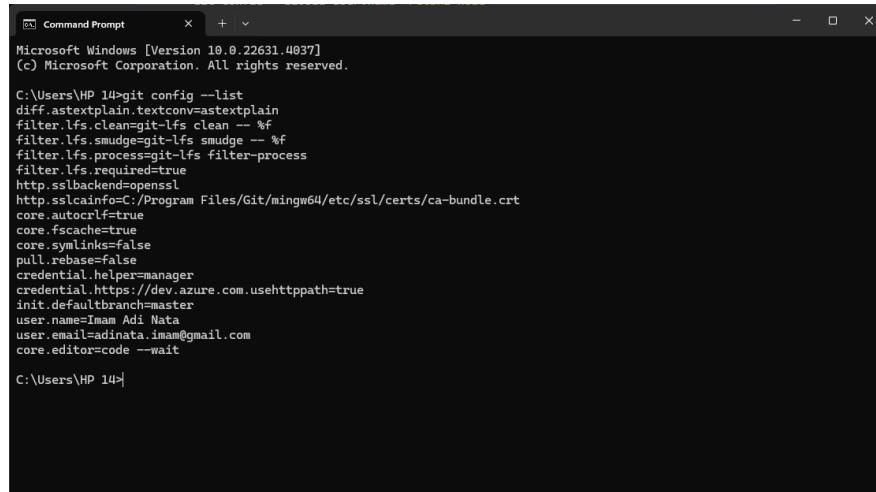
Silakan lakukan konfigurasi dengan perintah berikut ini.

```
git config --global user.name "Imam Adi Nata"
git config --global user.email imam@contoh.com
```

Kemudian periksa konfigurasinya dengan perintah:

```
git config --list
```

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



```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP 14>git config --list
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
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Imam Adi Nata
user.email=adinata.imam@gmail.com
core.editor=code --wait

C:\Users\HP 14>
```

*Gambar 18. Tampilan konfigurasi git*

Konfigurasi core.editor bersifat opsional. Sedangkan name dan email wajib. Jika kamu memiliki akun Github, Gitlab, Bitbucket atau yang lainnya maka username dan email harus mengikuti akun tersebut agar mudah diintegrasikan. Selain konfigurasi awal ini, kamu juga bisa konfigurasi SSH key untuk Github, Gitlab, dan Bitbucket.

Konfigurasi SSH key yang akan kita gunakan adalah dengan github. Sebelum melakukan konfigurasi ini, silahkan membuat akun github secara mandiri pada web github.

Kita bisa melakukan push ke Github melalui dua metode yakni, melalui protokol HTTPS dan SSH. Jika kita menggunakan HTTPS, maka kita akan dimintai password di setiap kali melakukan push. Sedangkan kalau pakai SSH, ini tidak perlu karena password-nya akan diwakili oleh SSH Key. Pada modul ini, kita akan belajar gimana cara setup SSH Key untuk Github. Sehingga nanti bisa melakukan push tanpa harus ngetik password terus menerus.

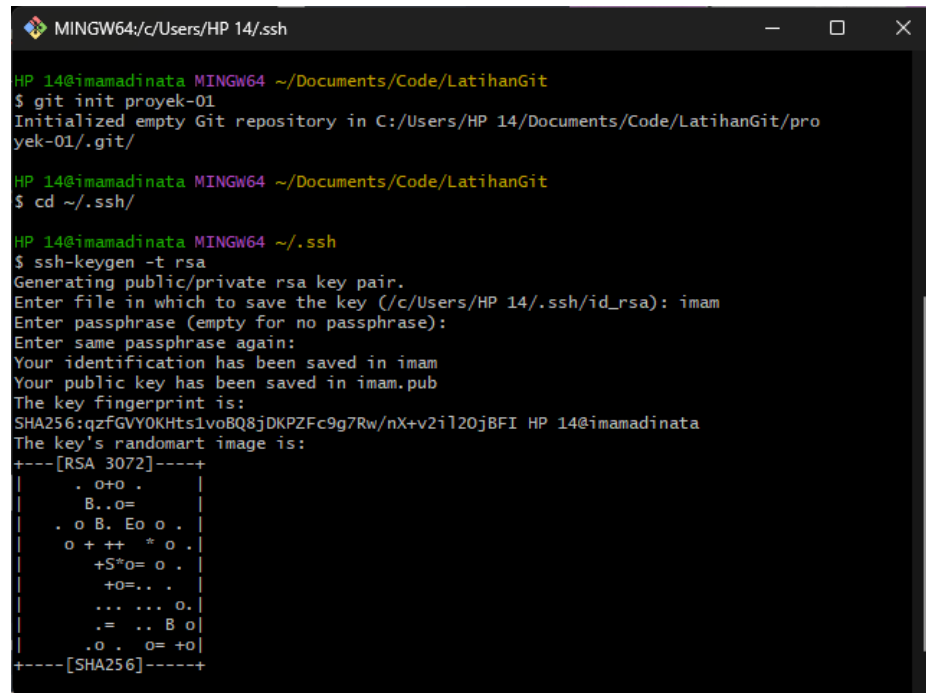
SSH Key adalah Key yang dipakai untuk berkomunikasi dengan server melalui protokol SSH. SSH Key terdiri dari publik key dan private key. Publik key akan kita taruh ke server dalam hal ini adalah Github, kemudian private key akan kita simpan di lokal. Private key sebenarnya mewakili password, sehingga kamu tidak boleh sembarangan membagikannya. Intinya, private key dan publik key akan dicocokkan secara otomatis saat kita terhubung dengan SSH.

Kita bisa membuat SSH Key dengan perintah ssh-keygen, ini bisa kamu ketik di Terminal dan Git Bash.

Silakan buka Terminal, kemudian ketik perintah berikut untuk membuat

```
SSH Key:
cd ~/.ssh/ #pindah ke direktori ssh
# membuat private key dan public key
ssh-keygen -t rsa
```


Pada perintah ini, kita masuk dulu ke folder .ssh yang ada di Home, kemudian kita generate SSH Key dengan ssh-keygen. Berikutnya, kita akan diminta mengisi id dan passphrase.



```
MINGW64:/c/Users/HP 14/.ssh
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit
$ git init proyek-01
Initialized empty Git repository in C:/Users/HP 14/Documents/Code/LatihanGit/proyek-01/.git/
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit
$ cd ~/.ssh/
HP 14@imamadinata MINGW64 ~/.ssh
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/HP 14/.ssh/id_rsa): imam
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in imam
Your public key has been saved in imam.pub
The key fingerprint is:
SHA256:qzfGVYOKHts1voBQ8jDKPZFc9g7Rw/nX+v2il20jBFI HP 14@imamadinata
The key's randomart image is:
+---[RSA 3072]-----+
|  . o+o .           |
|  B..o=            |
|  . o B. Eo o .    |
|  o + ++ * o .     |
|  +S*o= o .        |
|  +o=. . .         |
|  ... .. o .       |
|  . = .. B o       |
|  . o . o= +o      |
+---[SHA256]-----+
```

Gambar 19. Generate SSH key

Pada contoh di atas, saya mengisi id dengan imam dan passphrase tidak diisi. Langsung saja tekan Enter saat pengisian passphrase. Jika kita mengisi passphrase, maka akan diminta mengisinya lagi saat melakukan push melalui SSH. Setelah selesai akan ada dua file baru di dalam direktori ~/.ssh/.



```
HP 14@imamadinata MINGW64 ~/.ssh
$ ls
imam  imam.pub  known_hosts
HP 14@imamadinata MINGW64 ~/.ssh
$
```

Private Key berisi kunci rahasia yang tidak boleh diketahui siapapun dan Public Key berisi kunci publik yang akan kita taruh di Bitbucket.

Setelah membuat private key kita akan menambahkan SSH tersebut ke dalam github. Sebelumnya ambil dulu kunci publik yang sudah anda buat, gunakan perintah cat.

```
cat ~/.ssh/id_anda.pub
```

Copy semua teks yang ditampilkan.

```
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (master)
$ cd ~/.ssh
bash: cd: ~/.ssh: No such file or directory

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (master)
$ cd ~/.ssh

HP 14@imamadinata MINGW64 ~/.ssh
$ ls
imam  imam.pub  known_hosts

HP 14@imamadinata MINGW64 ~/.ssh
$ cat imam.pub
ssh-rsa /... sgDFpi
JbSQb14ji hb8Jzu
T/r7RnHD: LfCDP/
hY1svdNw navsPo
zTsLJ4NS: YqXnXV
/ExFoZ3e: WYxJjN
hu4DtSsi 14@i
imamadinat
```

Lalu masuk ke Settings>SSH and GPG Keys dengan alamat <https://github.com/settings/keys>, tambahkan SSH Key baru dengan mengklik New SSH Key. Setelah itu masukkan atau paste public key yang telah kita copy.

## Add new SSH Key

Title

LaptopHPW11

Key type

Authentication Key

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQGDQUvJgtqotuHfV58sq7HkobAh/chEwOjUpKORsgDFpilbSQbl4jq5oUFNaNBklyonr
ELqUzhk5Tmzz0wYE7s5Pk1Oidtve/IX+CK5rg+Sao4/ZVsrMQ2Whb8JzuT/r7RnHDSZYAnzDm7/A9HR0TJf7eFSSBNv+CkKYv
1/Kk4Tc/zcsTP6gMdPxf1EJSRaQ4ElsGm1fCDP/hY1svdNWQN9dcT8pj1gL+OWUImjspXZvFrO5gra69KclqYMC3N+HX8fPY
LOHfh6cAFCOYw1zonavsPozTsLJ4NSgmTlwu/AxWxBnPD2gQ0wZa5DNXCxnNt+amzd+va2fYFXos/JF4RgAVdRPs+jC2iBxR
YqXnXV/ExFoZ3ejrVVtKfsGHlHdyofPvuRRKidNtV5JDIUKQ85ncTMgb4xt0D8CMc9LHHV14kq/4xVQNYxjNhu4DtSsiC+be/g
774eBnXppPuU2UGHAtzBl0p6dmY+LY0Z4AcMtGh7peZ0HryUoXuBYvLaU= HP_14@imamadinata
```

Add SSH key

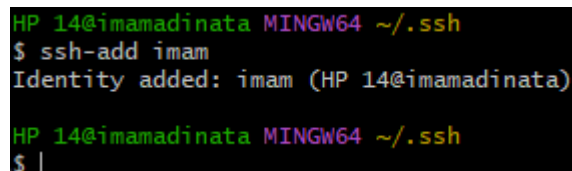
*Gambar 20. Input SSH Key ke github*

Jangan lupa untuk memberi judul dengan nama komputer yang akan digunakan untuk mempermudah penggunaan.

Setelah berhasil menyimpan SSH key lalu kita akan menguji SSH key tersebut. Karena kita baru pertama kali buat SSH Key, kita harus tambahkan dulu identitas kita. Caranya ketik perintah berikut:

```
ssh-add imam
```

Silakan ganti imam dengan id dari SSH key yang kamu buat tadi. Jika berhasil, maka akan muncul seperti ini:



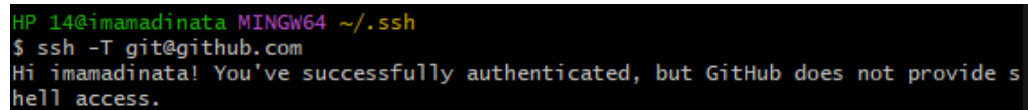
```
HP 14@imamadinata MINGW64 ~/.ssh
$ ssh-add imam
Identity added: imam (HP 14@imamadinata)
HP 14@imamadinata MINGW64 ~/.ssh
$ |
```

*Gambar 21. tambah identitas berhasil*

Setelah itu, baru kita bisa tes konek ke Github dengan SSH. Caranya, ketik perintah berikut:

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

jika berhasil maka akan tampil pesan berikut :



```
HP 14@imamadinata MINGW64 ~/.ssh
$ ssh -T git@github.com
Hi imamadinata! You've successfully authenticated, but GitHub does not provide shell access.
```

*Gambar 22. Koneksi ke github berhasil*

Setelah komputer di-restart, SSH Key harus ditambahkan lagi agar bisa digunakan. Nah, biar tidak mengetik perintah ssh-add berulang-ulang, kita bisa buat konfigurasi agar SSH Key otomatis ditambahkan.

Buat file baru di dalam direktori ~/.ssh/ bernama config.

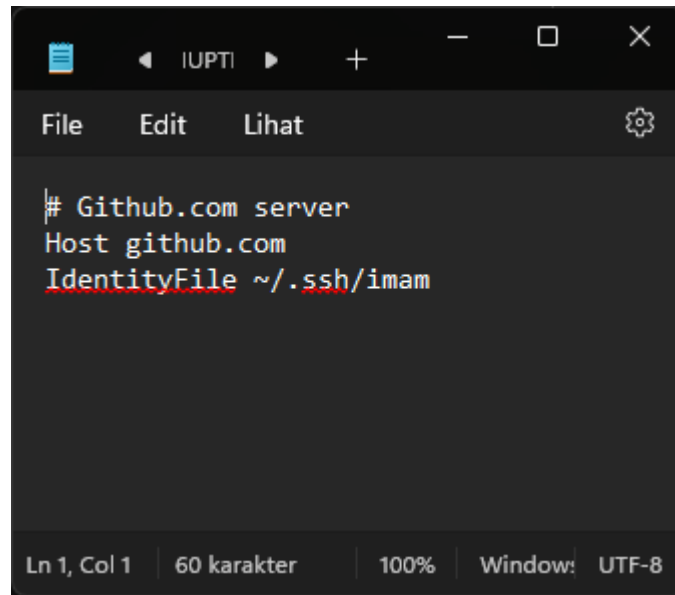


```
HP 14@imamadinata MINGW64 ~/.ssh
$ ls
config  imam  imam.pub  known_hosts
```

Isi file config dengan konfigurasi seperti berikut :

```
# Github.com server
Host github.com
IdentityFile ~/.ssh/imam
```

Ganti imam dengan id SSH Key yang kamu buat.



*Gambar 23. Isian file config*

kita sudah berhasil setup SSH Key untuk menggunakan Github melalui SSH. Selanjutnya, Cobalah untuk clone repository dengan SSH dan lakukan push dan pull.

Secara default, repository Git akan menggunakan nama branch master ketika kita baru pertama membuat repository. Nama ini sebenarnya mulai ditinggalkan dan disarankan pakai nama main. Soalnya di Github.. default branch atau branch utama yang digunakan adalah main. Saat kita mau upload repo ke Github, nantinya kita akan diminta untuk mengubah master menjadi main. Cara agar Git otomatis menggunakan main secara default dengan menambahkan konfigurasi berikut :

```
git config --global init.defaultBranch main
```

Dengan demikian, Git akan otomatis menggunakan nama main sebagai branch utama.

#### **1.4.3. Membuat Repository Git**

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

Pembuatan repository dapat dilakukan dengan perintah `git init nama-dir`. Contoh:

```
git init proyek-01
```

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

akan membuat sebuah direktori bernama `.git` di dalam proyek kita. Direktori ini digunakan Git sebagai database untuk menyimpan perubahan yang kita lakukan.

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

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.

```
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git init .
Reinitialized existing Git repository in C:/Users/HP 14/Documents/Code/LatihanGit/proyek-01/.git/
```

*Gambar 24. pembuatan repositori*

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

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

#### a. `.gitignore`

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

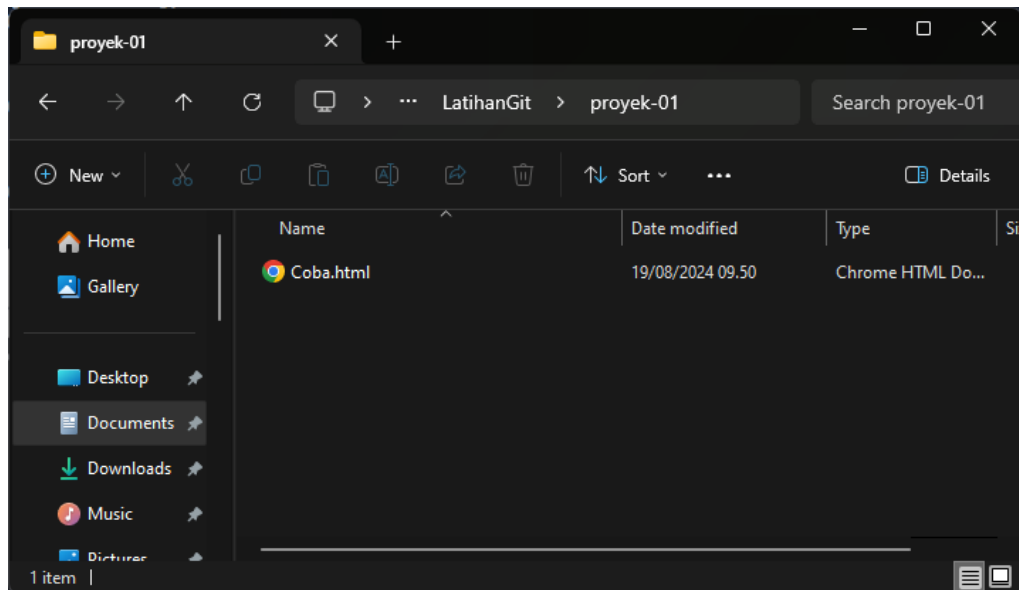
```
/vendor/
/upload/
/cache
test.php
```

Pada contoh file `.gitignore` di atas, kita memasukkan 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.

#### **1.4.4.** Membuat Revisi

Sekarang coba tambahkan sebuah file baru. Sebagai contoh, saya akan menambahkan tiga file HTML kosong.



Gambar 25. Tambah file ke dalam proyek

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

```

MINGW64/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
$ git branch -m master main
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git init .
Reinitialized existing Git repository in C:/Users/HP 14/Documents/Code/LatihanGit/proyek-01/.git/
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ ls
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git status
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
  
```

Gambar 26. tampilan git status

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

Tiga Kelompok Kondisi File dalam Git :

a. Modified

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



### b. Staged

Staged adalah kondisi di mana 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 Coba.html
```

### c. Committed

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

#### 1) Membuat Revisi Pertama

Sekarang kita akan sudah tahu kondisi-kondisi file dalam Git. Selanjutnya, silakan ubah kondisi tiga file HTML tadi menjadi staged dengan perintah `git add`.

```
git add Coba.html
```

Atau kita bisa melakukannya seperti ini jika file lebih dari 1

```
Git add file1.html file2.html .....
```

Atau seperti berikut

```
Git add *.html
```

Atau seperti ini (untuk semua file dan direktori):

```
Git add .
```

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

```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
No commits yet

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

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

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git add Coba.html

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git status
On branch main

No commits yet

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

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 27. file staged

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

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

Setelah itu, coba cek dengan perintah git status lagi.

```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git commit -m "Commit pertama"
[main (root-commit) 4e8d098] Commit pertama
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Coba.html

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git status
On branch main
nothing to commit, working tree clean

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 28. Setelah commit berhasil

Revisi pertama sudah kita buat. Selanjutnya cobalah untuk membuat revisi kedua.

## 2) Revisi Kedua

Ceritanya ada perubahan yang akan kita lakukan pada file Coba.html. Silakan modifikasi isi file index.html. Sebagai contoh saya mengisinya seperti ini.

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

```

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

```

Setelah itu ketik lagi perintah git status.

```

MINGW64/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Coba.html

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git commit -m "Commit pertama"
[main (root-commit) 4e8d098] Commit pertama
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Coba.html

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git status
On branch main
nothing to commit, working tree clean

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   Coba.html

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

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$

```

*Gambar 29. status git modified*

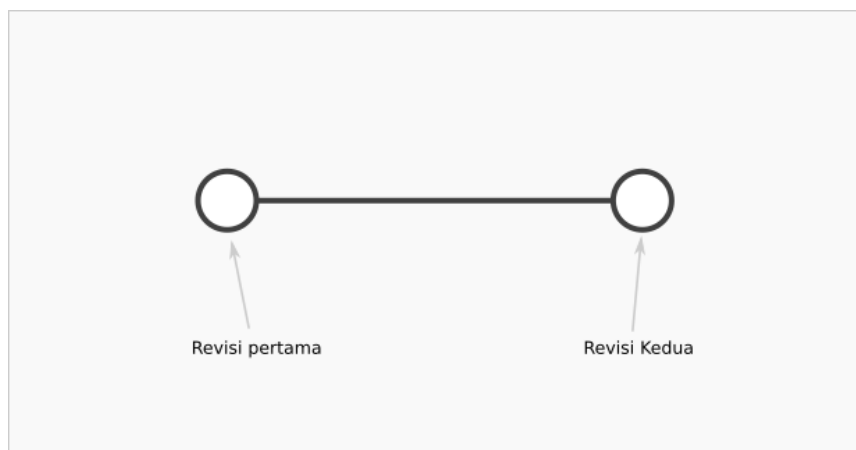
Terlihat di sana, file Coba.html sudah dimodifikasi. Kondisinya sekarang berada dalam modified. Lakukan commit lagi seperti revisi pertama.

```

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

```

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



*Gambar 30. Kondisi revisi git*

Sekarang Git sudah mencatat dua revisi yang sudah kita lakukan. Kita bisa ibaratkan

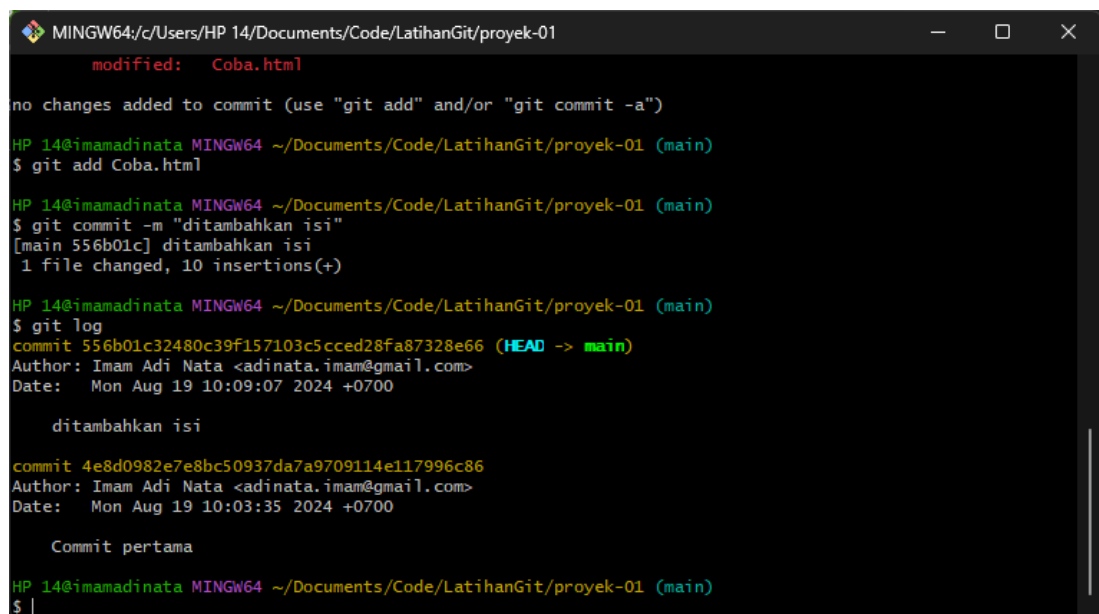
revisi-revisi ini sebagai checkpoint pada Game. Apabila nanti ada kesalahan, kita bisa kembali ke checkpoint ini.

#### 1.4.5. Melihat Log Revisi

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

```
git log
```

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



```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
modified:  Coba.html

no changes added to commit (use "git add" and/or "git commit -a")
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git add Coba.html
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git commit -m "ditambahkan isi"
[main 556b01c] ditambahkan isi
1 file changed, 10 insertions(+)
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git log
commit 556b01c32480c39f157103c5cced28fa87328e66 (HEAD -> main)
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:09:07 2024 +0700

    ditambahkan isi

commit 4e8d0982e7e8bc50937da7a9709114e117996c86
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:03:35 2024 +0700

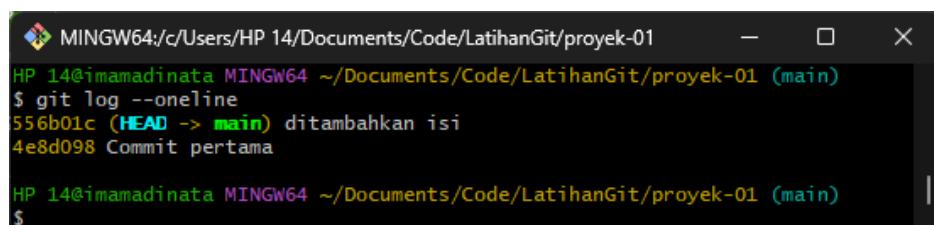
    Commit pertama
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ |
```

Gambar 31. Tampilan 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
```

Maka akan menghasilkan output :



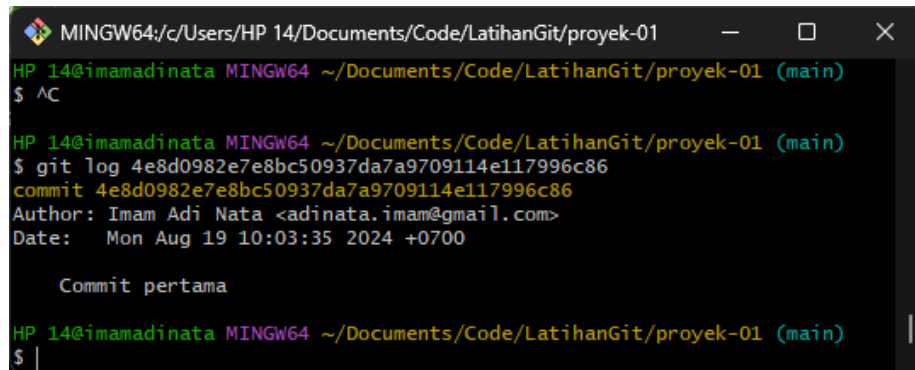
```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git log --oneline
556b01c (HEAD -> main) ditambahkan isi
4e8d098 Commit pertama
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 32. log one line

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

```
git log 4e8d0982e7e8bc50937da7a9709114e117996c86
```

Maka akan menghasilkan output:



```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ AC

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git log 4e8d0982e7e8bc50937da7a9709114e117996c86
commit 4e8d0982e7e8bc50937da7a9709114e117996c86
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:03:35 2024 +0700

    Commit pertama

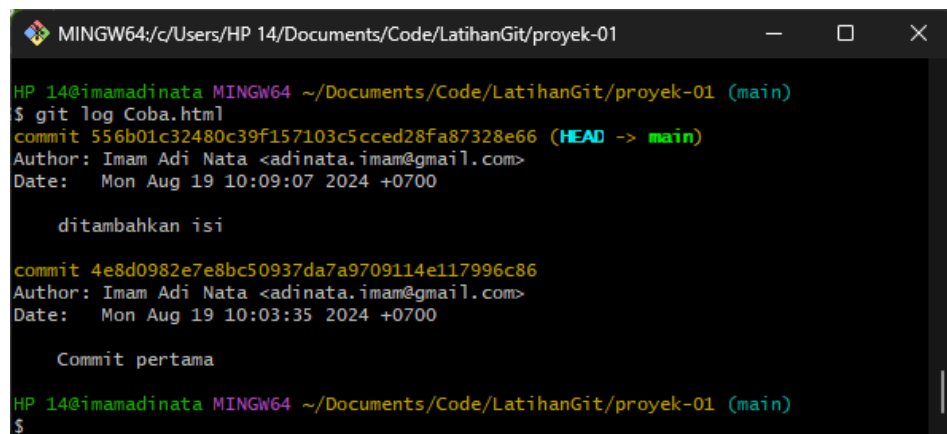
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ |
```

Gambar 33. log revisi berdasarkan nomor

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

```
git log Coba.html
```

Maka akan menghasilkan output :



```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git log Coba.html
commit 556b01c32480c39f157103c5cced28fa87328e66 (HEAD -> main)
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:09:07 2024 +0700

    ditambahkan isi

commit 4e8d0982e7e8bc50937da7a9709114e117996c86
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:03:35 2024 +0700

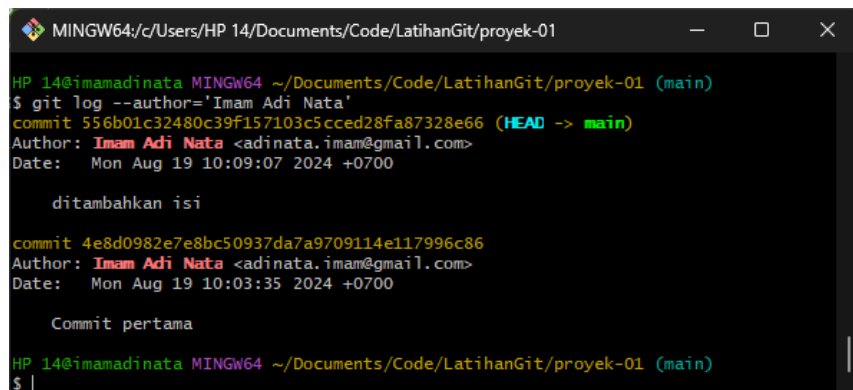
    Commit pertama

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 34. git log per file

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

```
git log --author='Imam Adi Nata'
```



```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git log --author='Imam Adi Nata'
commit 556b01c32480c39f157103c5cced28fa87328e66 (HEAD -> main)
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:09:07 2024 +0700

    ditambahkan isi

commit 4e8d0982e7e8bc50937da7a9709114e117996c86
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:03:35 2024 +0700

    Commit pertama

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ |
```

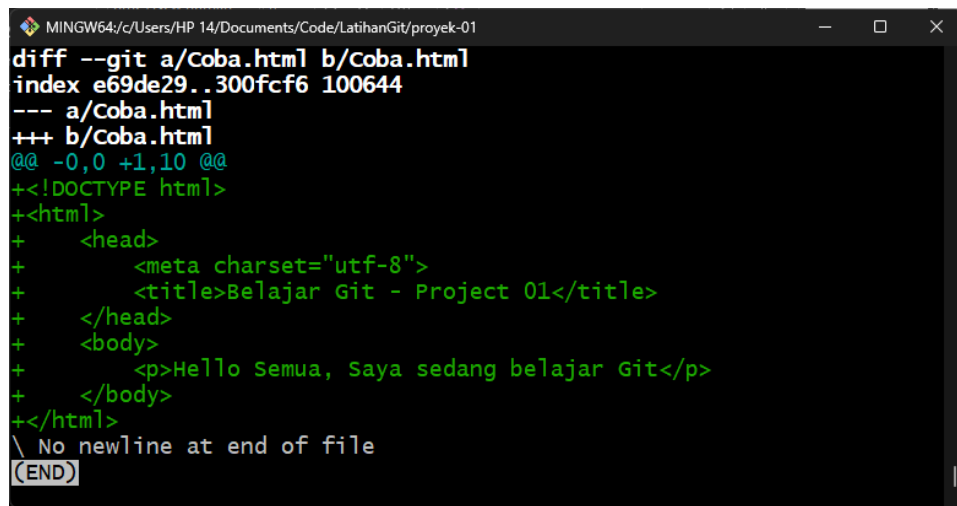
Gambar 35. Log revisi berdasarkan author

#### 1.4.6. Membandingkan Revisi

Perintah git diff berfungsi untuk melihat perbedaan perubahan di revisi. Gunakan perintah berikut ini untuk melihat perubahan yang dilakukan pada revisi tertentu.

```
git diff 4e8d0982e7e8bc50937da7a9709114e117996c86
```

4e8d0982e7e8bc50937da7a9709114e117996c86 adalah nomer revisi yang ingin dilihat.



```
MINGW64:/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
diff --git a/Coba.html b/Coba.html
index e69de29..300fcf6 100644
--- a/Coba.html
+++ b/Coba.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>
\ No newline at end of file
(END)
```

Gambar 36. Perbandingan revisi

Lihatlah hasil di atas, simbol plus (+) artinya kode yang ditambahkan. Sedangkan kalau ada kode yang dihapus simbolnya akan menggunakan minus (-). Perintah git diff akan membandingkan perubahan yang baru saja dilakukan dengan revisi/commit terakhir.

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

```
git diff Coba.html
```

Perintah di atas akan melihat perbedaan perubahan pada file Coba.html saja. Perintah untuk membandingkan perubahan pada revisi dengan revisi yang lain adalah sebagai berikut.

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

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>
```

#### 1.4.7. Membatalkan Revisi

##### 1) Membatalkan Perubahan Sebelum staged

Terkadang pada perubahan yang kita lakukan terjadi kesalahan dan kita ingin mengembalikannya seperti keadaan sebelumnya. Maka kita perlu menyuruh git untuk mengembalikannya. Ada beberapa perintah yang digunakan di antaranya: git checkout, git reset, dan git revert. Jika revisi kita belum staged ataupun mengetahui, kita bisa mengembalikannya menggunakan perintah git checkout nama\_file.html.

Contoh: Misalkan kita akan mengubah isi dari file Coba.html pada repositori project-01.

Sebelum diubah:

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

Setelah diubah

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

Hasil git diff:

```
MINGW64/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
(use "git restore <file>..." to discard changes in working directory)
modified:   Coba.html

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

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git diff
diff --git a/Coba.html b/Coba.html
index 300fcf6..db108ca 100644
--- a/Coba.html
+++ b/Coba.html
@@ -5,6 +5,7 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Semua, Saya sedang belajar Git</p>
+     <p>Hello Semua, Saya sudah belajar Git</p>
+     <p>Belajar git ternyata cukup menyenangkan</p>
   </body>
 </html>
\ No newline at end of file

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 37. git diff pada file Coba.html

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

```
git checkout Coba.html
```

Perubahan yang baru saja kita lakukan akan dibatalkan. Kalau tidak percaya, coba saja periksa file yang sudah diubah tadi atau cek dengan perintah git status.

```
MINGW64/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
--- a/Coba.html
+++ b/Coba.html
@@ -5,6 +5,7 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Semua, Saya sedang belajar Git</p>
+     <p>Hello Semua, Saya sudah belajar Git</p>
+     <p>Belajar git ternyata cukup menyenangkan</p>
   </body>
 </html>
\ No newline at end of file

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git checkout Coba.html
Updated 1 path from the index

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git status
On branch main
nothing to commit, working tree clean

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 38. hasil git checkout

Hati-hati! Terkadang perintah ini sangat berbahaya, karena akan menghapus perubahan yang baru saja dilakukan. Bila kita sudah mengubah banyak hal, maka itu akan sia-sia setelah menjalankan perintah ini.

## 2) Membatalkan Perubahan File yang Sudah dalam Kondisi *staged*

Kondisi staged merupakan kondisi file yang sudah di add (git add), namun belum



disimpan (git commit) ke dalam Git.

Sebagai contoh, kita lakukan perubahan lagi di file Coba.html seperti pada contoh sebelumnya.

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

```
git add Coba.html
```

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

```
git reset index.html
```

Cek statusnya lagi:

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

### 3) Membatalkan Perubahan File yang Sudah dalam Kondisi Committed

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

Misalkan, kita ubah kembali file Coba.html.

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

Kemudian kita melakukan commit.

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

Sekarang kita akan melihat nomer commit dengan perintah git log.

```
MINGW64:~/Documents/Code/LatihanGit/proyek-01
1 file changed, 2 insertions(+), 1 deletion(-)

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git log
commit 521e8d87d88784a3c4729e6c0afac1839aa8815d (HEAD -> main)
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 11:33:48 2024 +0700

    belajar git asyique

commit 556b01c32480c39f157103c5cced28fa87328e66
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:09:07 2024 +0700

    ditambahkan isi

commit 4e8d0982e7e8bc50937da7a9709114e117996c86
Author: Imam Adi Nata <adinata.imam@gmail.com>
Date: Mon Aug 19 10:03:35 2024 +0700

    Commit pertama

HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ |
```

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

```
git checkout 4e8d0982e7e8bc50937da7a9709114e117996c86 Coba.html
```

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

#### 4) Kembali ke 3 Commit sebelumnya

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

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

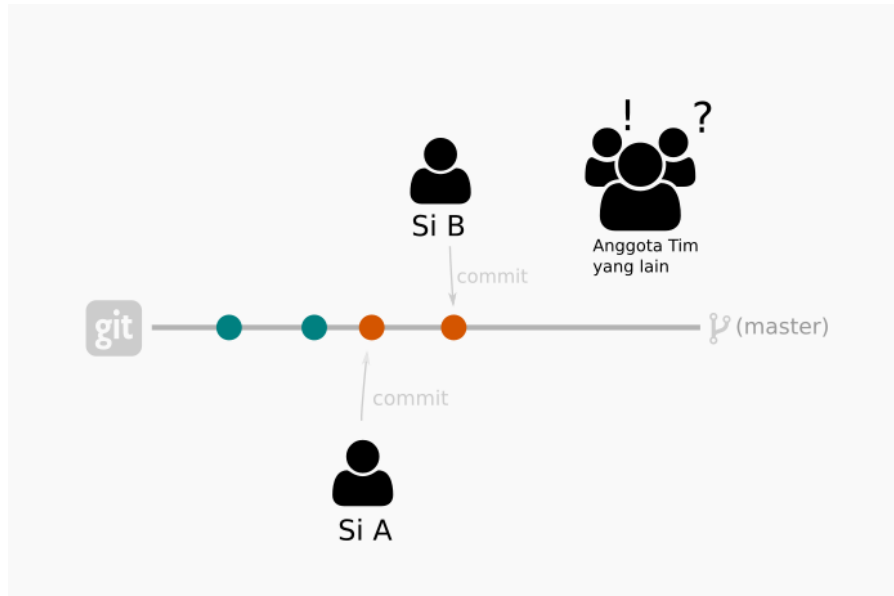
#### 5) Membatalkan Semua Perubahan yang ada

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

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

### 1.4.8. Menggunakan Percabangan

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 akan mengalami kebingungan karena hal tersebut.



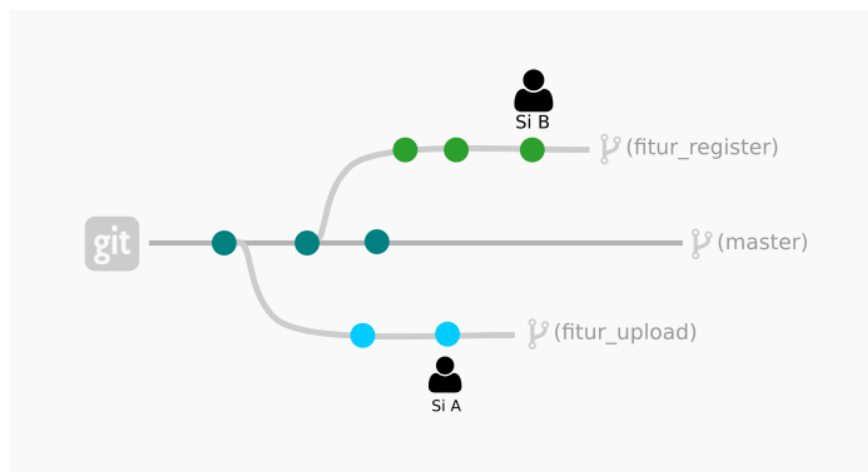
*Gambar 39. Kondisi yang mungkin terjadi*

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 (main).

Perintah untuk membuat cabang adalah `git branch`, kemudian diikuti dengan nama cabangnya.

```
git branch fitur_register
```

Maka Git akan membuat cabang bernama `fitur_register`.



*Gambar 40. Cabang-cabang dalam git*

Sekarang setiap orang memiliki cabangnya masing-masing. Mereka bebas bereksperimen. Untuk melihat cabang apa saja yang ada di repositori, gunakan perintah `git branch`.

```
MINGW64/c/Users/HP 14/Documents/Code/LatihanGit/proyek-01
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$ git branch
* main
HP 14@imamadinata MINGW64 ~/Documents/Code/LatihanGit/proyek-01 (main)
$
```

Gambar 41. melihat cabang dalam git

Tanda bintang (\*) artinya cabang yang sedang aktif atau Kita sedang berada di sana.

Latihan

Untuk memantapkan pemahaman tentang percabangan Git, mari kita coba praktik.

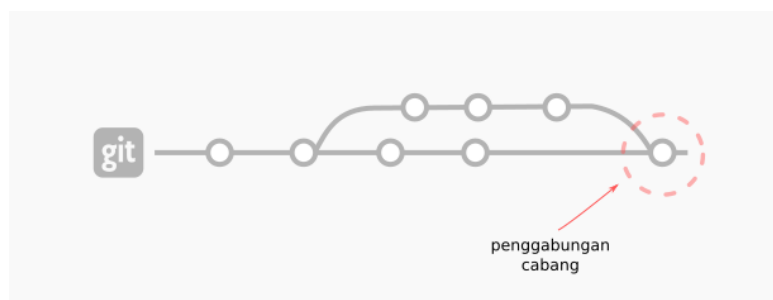
- Pada repositori, buatlah sebuah cabang baru dengan nama halaman\_login
- Setelah itu, pindah ke cabang yang baru saja kita buat dengan perintah: git checkout halaman\_login
- Lalu tambahkan file login.html, isinya terserah anda.
- Kita sudah menambahkan file login.html. Selanjutnya kita lakukan commit.
- revisi kita pada cabang halaman\_login sudah disimpan. Sekarang coba kembali ke cabang main dengan perintah git checkout main.
- Apakah anda menemukan file login.html?
- Sekarang kembali lagi ke cabang halaman\_login.
- Cek lagi, apakah sekarang file login.html sudah ada?

Anggaplah kita sudah selesai membuat fitur login di cabang halaman\_login. Sekarang kita ingin Menggabungkannya dengan cabang main (utama).

Pertama, kita harus pindah dulu ke cabang main. Setelah itu, barulah kita bisa menggabungkan dengan perintah git merge.

```
git merge halaman_login
```

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



Gambar 42. Penggabungan Cabang

kadang sering terjadi bentrok ketika menggabungkan cabang. 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 repositori. Dia harus bertindak adil, kode mana yang harus diambil. Biasanya akan ada proses diskusi dulu dalam mengambil keputusan.

Cabang yang sudah mati atau tidak ada pengembangan lagi, sebaiknya dihapus. Agar repositori kita bersih dan rapi. Cara menghapus cabang, gunakan perintah `git branch` dengan argumen `-d` dan diikuti dengan nama cabangnya.

```
git branch -d halaman_login
```

## 1.5. TUGAS MODUL 1

### 1.5.1. Soal

- Buatlah sebuah repository untuk membangun suatu aplikasi berorientasi objek.
- Isi repository tersebut dengan file bebas
- Lakukan dari inisiasi git sampai dengan push ke github.
- Tuliskan laporan lalu kumpulkan.

### 1.5.2. Petunjuk Pengerjaan

#### a) Laporan:

- Buatlah laporan akhir sesuai dengan praktikum yang dilakukan
- Tuliskan laporan ke dalam akun masing-masing web medium
- Publikasikan laporan tersebut
- Kirimkan tautan laporan ke elita.
- **Batas Pengumpulan:** Sebelum Pertemuan Praktik Ke 2.