

SIC Batch 5

Week 2 - Git & Github



Requirement

- Install GIT (<https://git-scm.com>)
- Membuat akun Github (<https://github.com>)
- Download Github Desktop (<https://desktop.github.com>) (Optional)

Pengenalan GIT

**Git adalah Tools untuk
programmer**

Git sebagai Version Control System



Apa itu Version Control System

Tugasnya adalah *mencatat* setiap *perubahan* pada *File* (termasuk code yang kita buat) pada suatu proyek baik dikerjakan secara *individu* maupun *tim*.

Git adalah aplikasi yang dapat melacak setiap perubahan yang terjadi pada suatu folder atau file.

Git biasanya digunakan oleh para programmer sebagai tempat penyimpanan file pemrograman mereka, karena lebih efektif.

—

**File -file yg disimpan
menggunakan git akan
terlacak seluruh
perubahannya, termasuk siapa
yang mengubah.**

**Sepandai apapun
#programmer, tidak akan
pernah bisa bekerja sendirian
selamanya**

—



'WHY' should use GIT and Github?

Dengan menggunakan GIT dan Github, kamu akan bisa bekerja dalam sebuah tim. Tujuan besarnya adalah kamu bisa *berkolaborasi* mengerjakan proyek yang sama tanpa harus repot copy paste folder aplikasi yang terupdate.

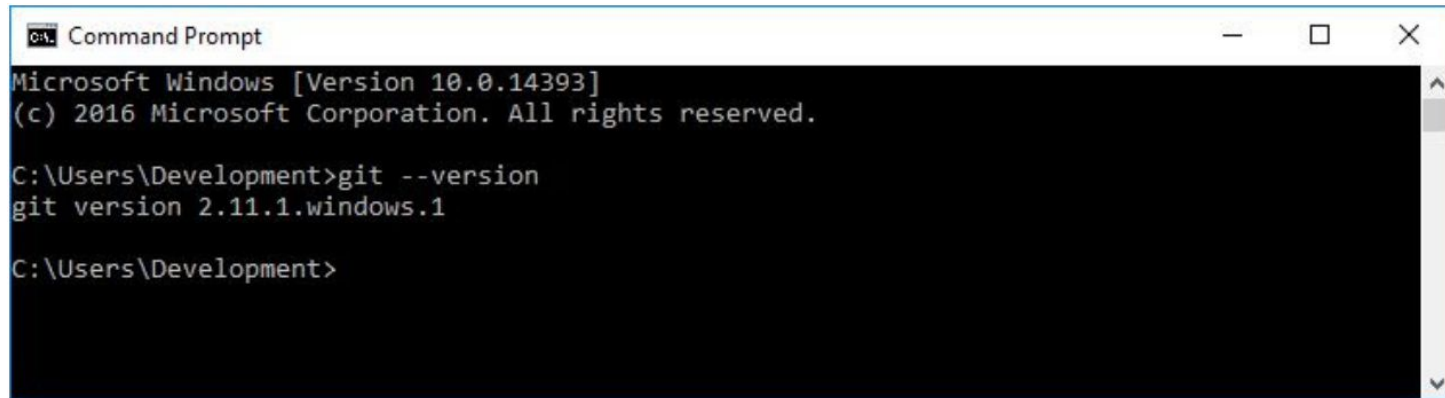
Kamu juga *tidak perlu menunggu* rekan dalam satu tim kamu menyelesaikan suatu program dahulu untuk berkolaborasi. Kamu bisa membuat file didalam proyek yang sama atau membuat code di file yang sama dan menyatukannya saat sudah selesai.

Instalasi GIT

**Download dan jalankan hasil
download GIT kamu seperti
instal aplikasi pada umumnya**

—

Cek apakah instalasi berhasil



```
Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Development>git --version
git version 2.11.1.windows.1

C:\Users\Development>
```



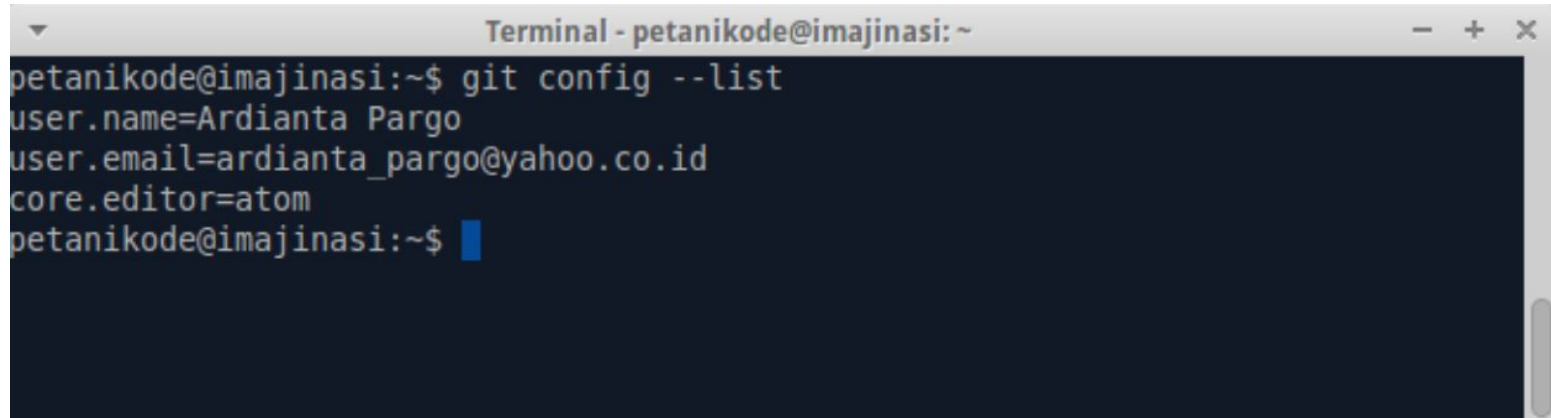
Setup Awal

```
git config --global user.name "Petani Kode"  
git config --global user.email contoh@petanikode.com
```



Cek apakah setup berhasil

```
git config --list
```



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


WARNING:
email yang disetup HARUS
SAMA dengan yang digunakan
pada GITHUB

Repository GIT

**Repository adalah direktori
proyek yang kita buat.**

1 Repo = 1 Proyek = 1 Direktori

—



Membuat Repository

```
git init proyek-01
```

**Command line tadi akan
membuat sebuah direktori
baru**

—

Bagaimana jika folder sudah ada sebelumnya?

Calm down, kamu bisa gunakan ini:

```
git init .
```

GIT STATUS, GIT ADD, GIT COMMIT

**Buat 2-3 file code pada
direktori yang telah dibuat**

—

1. GIT STATUS



Git Status

```
davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) x git status
On branch master

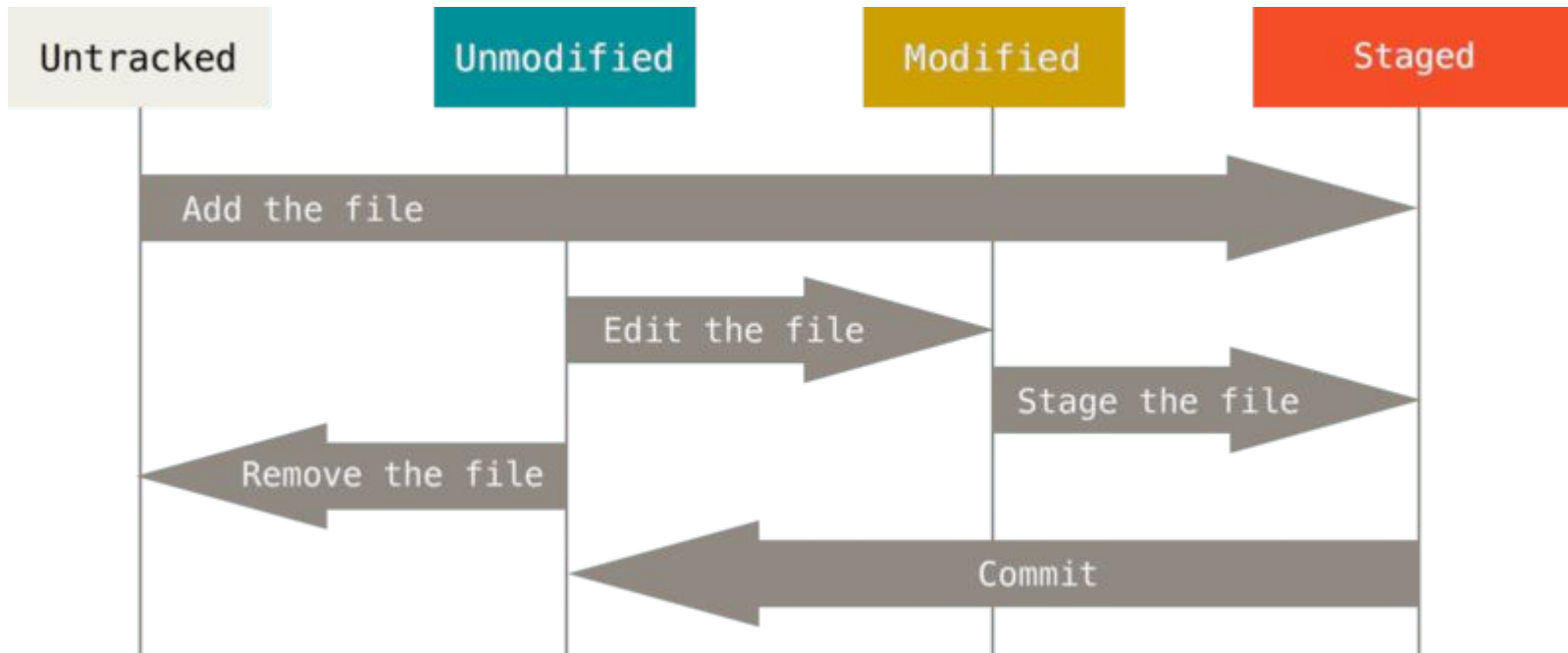
No commits yet

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

        index.html
        produk.html

nothing added to commit but untracked files present (use "git add" to track)
→ proyek-01 git:(master) x
```

3 Kondisi File pada GIT





Modified

Modified adalah kondisi dimana revisi atau **perubahan sudah dilakukan**, tetapi **belum ditandai (untracked)** dan **belum disimpan** dalam version control.



Staged

Staged adalah kondisi dimana **revisi sudah ditandai** (modified) namun **belum disimpan** di version control.



Committed

Commit/committed adalah kondisi dimana **revisi sudah disimpan** pada version control.

2. GIT ADD

Setelah cek status dengan 'git status', selanjutnya kita ubah status 'untracked file' dan 'unmodified' menjadi modified

Gunakan git add

```
git add index.html
```

atau

```
git add .
```

```

davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) ✕ git add .
→ proyek-01 git:(master) ✕ git status
On branch master

No commits yet

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

        new file:   index.html
        new file:   produk.html

→ proyek-01 git:(master) ✕
```

Tapi gimana kalau untracked
file nya dalam jumlah besar?

Calm down, bisa
menggunakan perintah ini:

```
git add .
```

3. GIT COMMIT

Lakukan 'git commit' untuk save perubahan pada version control

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

```

davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) x git commit -m "Commit pertama"
[master (root-commit) 890d53d] Commit pertama
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
 create mode 100644 produk.html
→ proyek-01 git:(master) git status
On branch master
nothing to commit, working tree clean
→ proyek-01 git:(master) 
```

Revisi Kedua pada GIT



davidwinalda@Davids-Air: ~/documents/david/works/proyek-01



```
→ proyek-01 git:(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")
```

```
→ proyek-01 git:(master) ✕
```


Perhatikan!

Kondisi sudah tidak di
'untrackted files' tapi 'modified
files'

GIT LOG



Kedua revisi diatas kita anggap sebagai checkpoint.

Jika nanti ada kesalahan, kita bisa kembali pada checkpoint ini.



GIT LOG

Dari dua revisi yang sudah dilakukan kita dapat melihat catatal log dari revisi - revisi tersebut dengan menggunakan perintah berikut ini:

```
git log
```

```
git log

commit 38cf02067a3838760eeecb46ab2adc4317d1f22 (HEAD -> master)
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 21:10:34 2020 +0700

    Commit kedua

commit 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 20:50:53 2020 +0700

    Commit pertama

(END)
```



TIPS

Untuk git log yang lebih pendek, bisa menggunakan perintah berikut ini:

```
git log --oneline
```

git log --oneline

⌘

38cf020 (HEAD -> master) Commit kedua

890d53d Commit pertama

(END)

Melihat log dari berbagai sisi.

- 1. Melihat log menggunakan nomor version/commit**
- 2. Melihat log file tertentu**
- 3. Melihat log berdasarkan author**

Melihat log dari nomor version

```
davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) git log 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
```

```
git log 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
commit 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 20:50:53 2020 +0700

    Commit pertama

(END)
```

Melihat log dari file tertentu

```
git log index.html
```

```
git log index.html

commit 38cf02067a3838760eeecb46ab2adc4317d1f22 (HEAD -> master)
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 21:10:34 2020 +0700

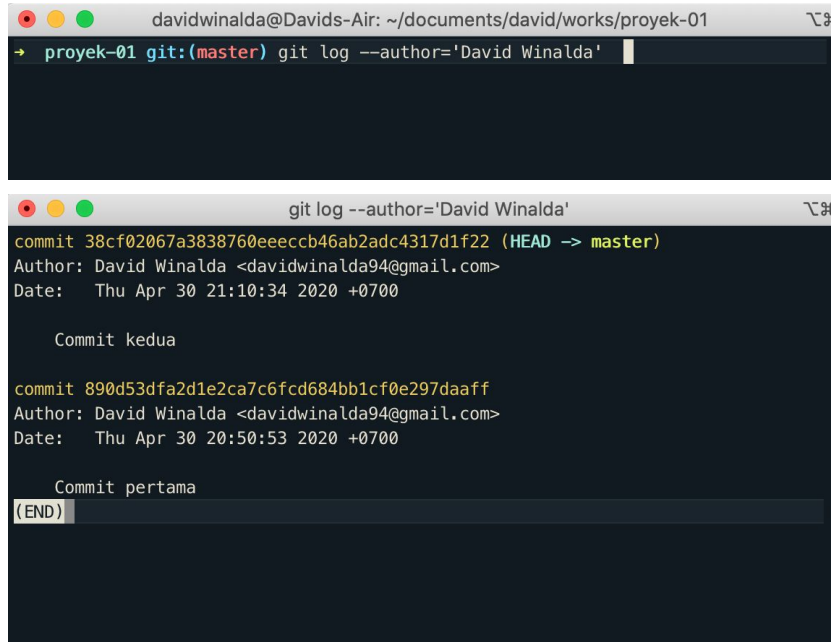
    Commit kedua

commit 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 20:50:53 2020 +0700

    Commit pertama

(END)
```

Melihat log berdasarkan author



```

davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) git log --author='David Winalda'

commit 38cf02067a3838760eeecb46ab2adc4317d1f22 (HEAD -> master)
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 21:10:34 2020 +0700

    Commit kedua

commit 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
Author: David Winalda <davidwinalda94@gmail.com>
Date: Thu Apr 30 20:50:53 2020 +0700

    Commit pertama
(END)
```

GIT CHECKOUT, GIT RESET, GIT REVERT

Jika perubahan yang sedang dilakukan terjadi kesalahan dan kita ingin mengembalikan keadaan seperti sebelumnya maka itu bisa dilakukan :)



Membuat Revisi/Perubahan

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>
```



Membuat Revisi/Perubahan

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>
```


Cek Perubahan

```

davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) x git diff

```

```

git diff
diff --git a/index.html b/index.html
index 300fcf6..19757c1 100644
--- a/index.html
+++ b/index.html
@@ -5,6 +5,7 @@
     <title>Belajar Git - Project 01</title>
   </head>
   <body>
-     <p>Hello Semua, Saya sedang belajar Git</p>
+     <p>Hello Dunia!, Saya sudah belajar Git</p>
+     <p>Belajar git ternyata cukup menyenangkan</p>
   </body>
</html>
\ No newline at end of file
(END)

```



Membatalkan Perubahan - Belum Stagged dan Belum Committed

Jika case nya seperti judul diatas maka bisa dilakukan dengan perintah berikut ini:

```
git checkout index.html
```

Maka akan menjadi:

```
$ git status  
On branch master  
nothing to commit, working directory clean
```

Membatalkan Perubahan - Sudah Stagged namun Belum Committed

Stagged = Sudah di Add

```

davidwinalda@Davids-Air: ~/documents/david/works/proyek-01
→ proyek-01 git:(master) ✕ git add .
→ proyek-01 git:(master) ✕ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html

→ proyek-01 git:(master) ✕
```



Membatalkan Perubahan - Sudah Stagged namun Belum Committed

Jika case nya seperti diatas maka bisa dilakukan dengan perintah berikut ini:

```
git reset index.html
```

Maka akan menjadi:

```
→ proyek-01 git:(master) ✕ git reset index.html
Unstaged changes after reset:
M   index.html
→ proyek-01 git:(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")
```

Kondisi file sudah pada kondisi
'Modified'.

Selanjutnya kita lakukan
proses yg sama sebelumnya.

Menggunakan 'git checkout'



Membatalkan Perubahan - Sudah Stagged namun Belum Committed

Jika case nya seperti judul diatas maka bisa dilakukan dengan perintah berikut ini:

```
git checkout index.html
```

Maka akan menjadi:

```
$ git status  
On branch master  
nothing to commit, working directory clean
```



Membatalkan Perubahan - Sudah Committed

```
commit 1e91872c7bf68a3bdb934c0cf4fa7438461d7f6a (HEAD -> master)
Author: David Winalda <davidwinalda94@gmail.com>
Date:   Fri May 1 00:09:45 2020 +0700
```

Commit Ketiga

```
commit 38cf02067a3838760eeecb46ab2adc4317d1f22
Author: David Winalda <davidwinalda94@gmail.com>
Date:   Thu Apr 30 21:10:34 2020 +0700
```

Commit kedua

```
commit 890d53dfa2d1e2ca7c6fcd684bb1cf0e297daaff
Author: David Winalda <davidwinalda94@gmail.com>
Date:   Thu Apr 30 20:50:53 2020 +0700
```

Commit pertama

(END)

Kita bisa mengembalikan
kondisi ke commit *sebelumnya*
dari commit *terakhir*
menggunakan nomor commit

—

Eits, tapi ada 2 jenis case.

- 1. Kita bisa mengembalikan commit hanya pada file tertentu**
- 2. Kita bisa mengembalikan commit untuk semua file**



Mengembalikan Commit Pada File Tertentu

```
→ proyek-01 git:(master) git checkout 38cf02067a3838760eeecb46ab2adc4317d1f22 index.html
```

```
git reset index.html
```



Mengembalikan Commit Untuk Semua File

Kita hanya perlu menggunakan nomor commit saja. Tidak perlu menambahkan spesifik file.

Berikut penggunaan perintahnya:

```
→ proyek-01 git:(master) git checkout 1e91872c7bf68a3bdb934c0cf4fa7438461d7f6a
```

Jika ingin mengembalikan commit jauh ke bawah. Misal kita ingin kembali pada 3 commit sebelumnya

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

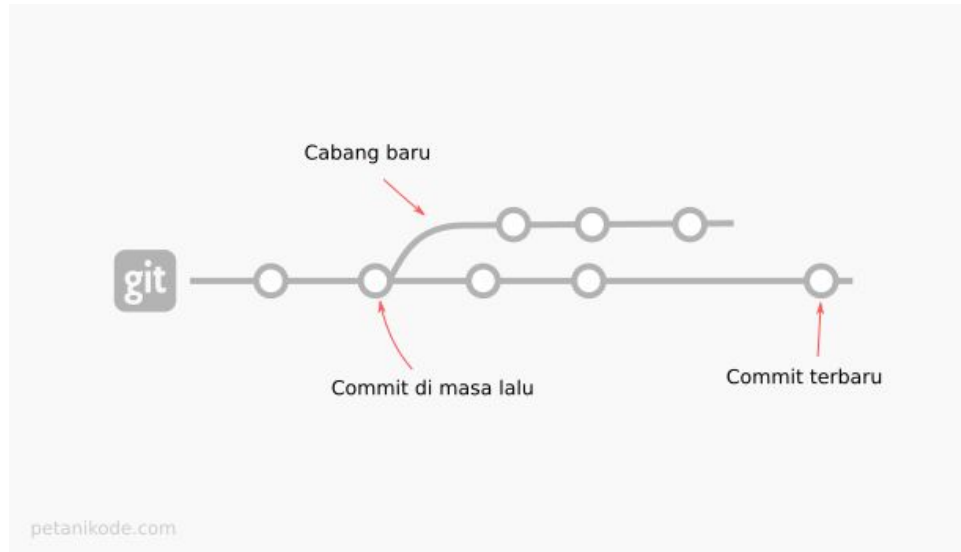


GIT REVERT

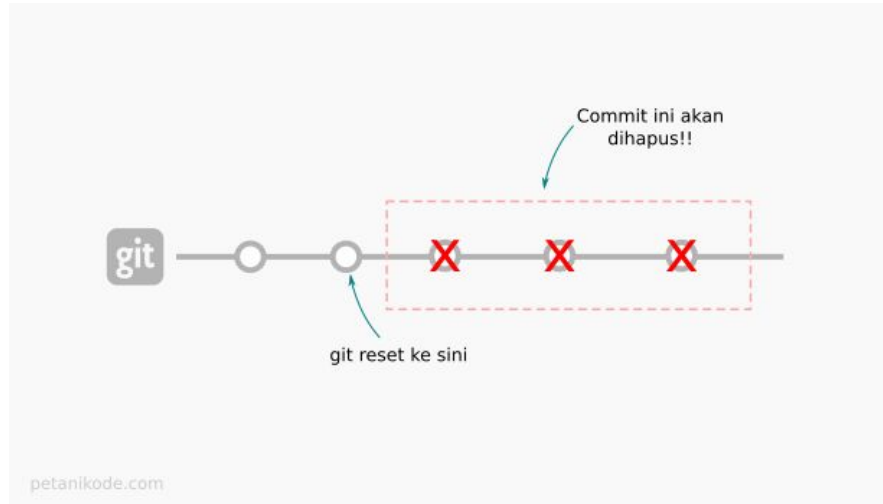
GIT Revert akan membatalkan semua perubahan yang ada tanpa menghapus commit terakhir. Jika menggunakan GIT Reset, commit terakhir akan hilang.

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

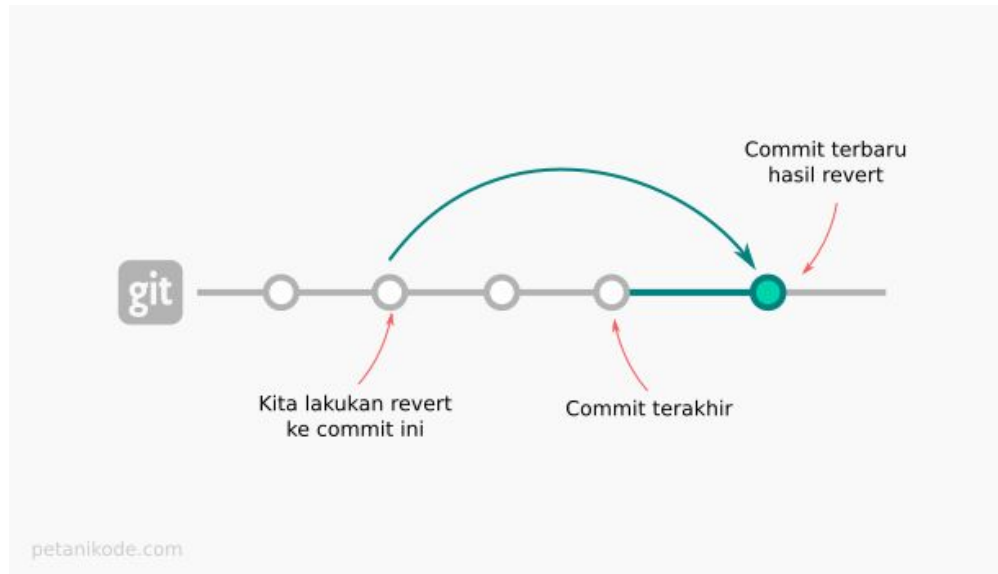
GIT CHECKOUT



GIT RESET



GIT REVERT



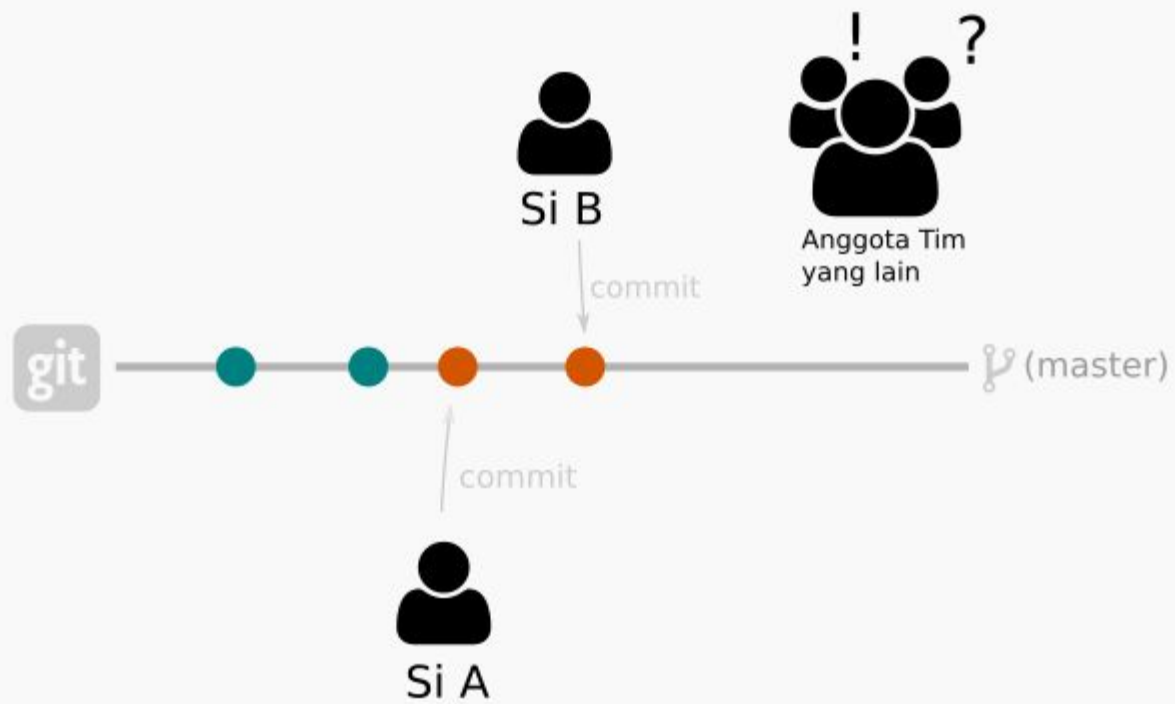
GIT BRANCH

**Fitur yang WAJIB digunakan
jika berkolaborasi dengan
developer atau dalam tim**

—

**Untuk menghindari conflict
code yang dikembangkan.**

**Kita tidak boleh berkolaborasi
dalam project di satu branch
yang sama!**



Misalnya Bowo akan mengerjakan fitur A dan Gigih mengerjakan fitur B.

Masing-masing fitur harus dibuat branch masing-masing.

**Tidak boleh mengganggu
branch 'master' yang sudah
terupdate**

—



GIT BRANCH

Untuk membuat branch, gunakan perintah berikut ini

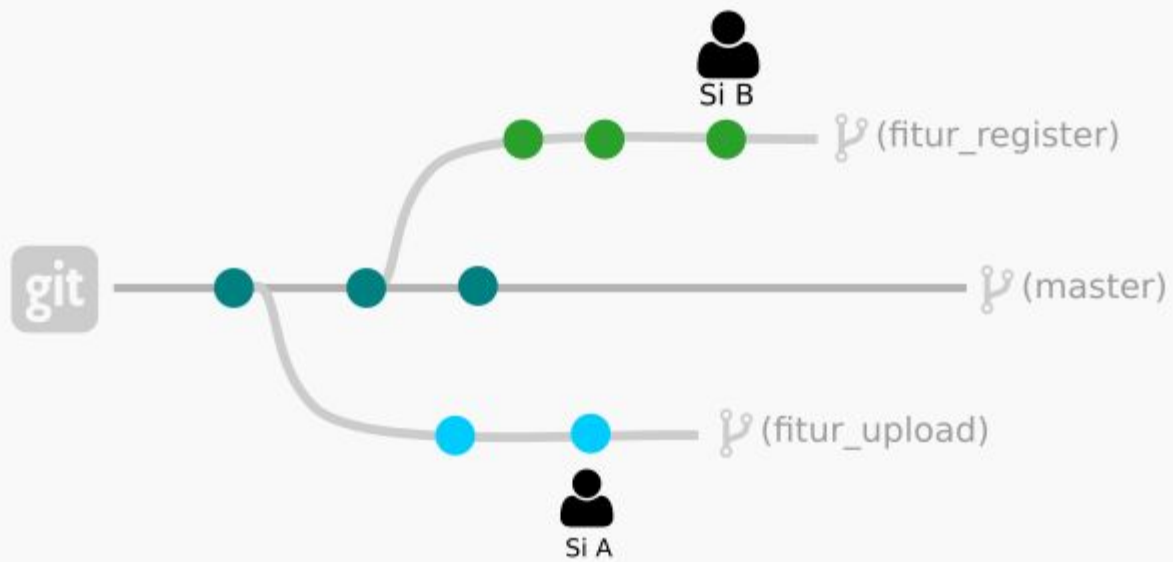
```
git branch <branch>
```



GIT BRANCH

Misalkan kita ingin membuat fitur register. Jadi kita akan membuat branch baru.

```
git branch fitur_register
```



Melihat List Branch

```
→ proyek-01 git:(1e91872) git branch
```

```
* (HEAD detached at 1e91872)
  fitur_register
  fitur_upload
  master
```

```
(END)
```



Pindah ke branch tertentu

Untuk menuju kedalam suatu branch tertentu. Gunakan perintah seperti berikut ini:

```
→ proyek-01 git:(master) git checkout fitur_register  
Switched to branch 'fitur_register'
```



Delete Branch

Untuk menghapus sebuah branch, gunakan perintah seperti berikut ini:

```
git branch -d <branch>
```

```
git branch -d halaman_login
```

GIT MERGE

**Setelah membuat branch baru,
lalu lakukan commit.**

**Saatnya kita menyatukan
pekerjaan ke master file/branch
utama yaitu branch MASTER**



GIT MERGE

Untuk menyatukan branch cabang fitur yang telah kita kembangkan. Gunakan perintah seperti berikut ini:

1. Kita harus checkout dahulu ke branch master

```
git checkout master
```

1. Lalu lakukan merge

```
git merge halaman_login
```

```
# Start a new feature
git checkout -b new-feature master
# Edit some files
git add <file>
git commit -m "Start a feature"
# Edit some files
git add <file>
git commit -m "Finish a feature"
# Merge in the new-feature branch
git checkout master
git merge new-feature
git branch -d new-feature
```




Additional Info

1. Remote repository, tempat untuk menyimpan git repo di internet
2. ``git remote`` untuk melihat daftar remote di repository
3. ``git remote add`` untuk menambahkan remote
4. ``git push`` untuk mengirim perubahan ke remote repository
5. ``git fetch`` untuk mengambil metadata dari remote
6. ``git pull`` untuk mengambil perubahan dari remote ke local
7. “Pull Request”, proses review merge yang dibuat

Exercise



GIT & GITHUB ON PROJECT

- Clone github repository dari mentor kalian/ buat repo baru
- Buat branch baru dengan format feature/NAMA_KALIAN
- Tambahkan file/code python, misalnya membuat sebuah code python untuk melakukan print hello world.
- Lakukan Add dan Commit
- Push ke branch yang sudah dibuat
- Buka website GITHUB dan lakukan Pull Request ke branch master
- Mentor kalian akan mereview code teman-teman dan melakukan merge



Make your Repository

- Ini waktunya publish code Python for Data Science ke GITHUB repository kalian!



Question - Answer in Markdown

- Sebutkan step-step dan command line saat menginisiasi project untuk menggunakan git hingga sampai push ke github
- Apa perbedaan git reset dan git revert?
- Apa bedanya GIT dengan GITHUB?