



# git

**Version Control System**  
Rev-0

**Finsa Nurpandi**  
**2018**

## Daftar Isi

<b>Daftar Gambar .....</b>	<b>2</b>
<b>1. Intro to Git.....</b>	<b>3</b>
<b>2. Git Install .....</b>	<b>4</b>
<b>3. Working Locally .....</b>	<b>4</b>
3.1 git init .....	5
3.2 git status .....	5
3.3 git add .....	6
3.4 git commit .....	7
3.5 git log.....	7
3.6 git branch.....	7
3.7 git checkout .....	8
3.8 git merge .....	9
3.9 Basic Merge Conflict.....	9
<b>4. Working Remotely .....</b>	<b>10</b>
4.1 git remote .....	12
4.2 git push.....	13
<b>5. How to contribute?.....</b>	<b>14</b>
5.1 Contributor .....	14
5.2 Project Owner.....	19
5.3 Sync Local Repo with Remote Repo.....	20
5.4 Sync a Fork Repo with Original Repo .....	21
5.5 Merge Branch with Master .....	21
<b>Referensi.....</b>	<b>21</b>

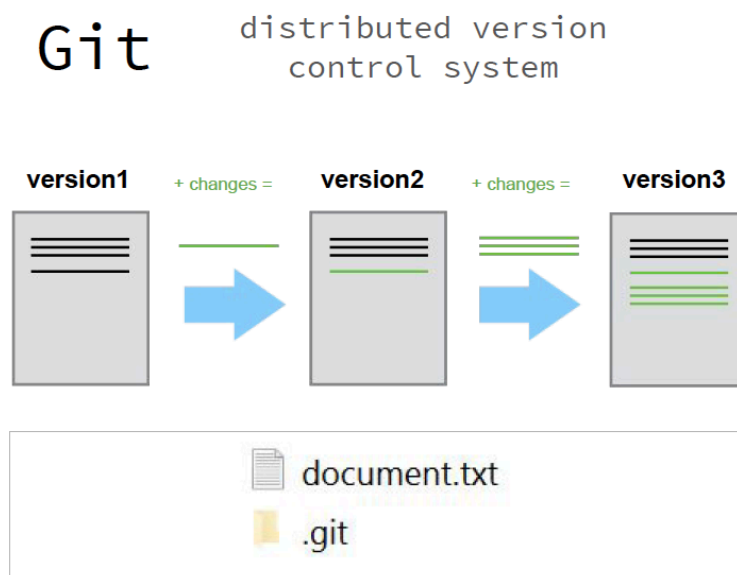
## Daftar Gambar

Gambar 1 Git Version Control .....	3
Gambar 2 Branch System .....	3
Gambar 3 Git on Terminal .....	4
Gambar 4 Memilih Folder Project .....	5
Gambar 5 git init .....	5
Gambar 6 git status .....	5
Gambar 7 git status .....	6
Gambar 8 Git add .....	6
Gambar 9 git status .....	6
Gambar 10 git commit .....	7
Gambar 11 git log .....	7
Gambar 12 git branch .....	8
Gambar 13 git merge conflict .....	9
Gambar 14 file conflict .....	10
Gambar 15 fix conflict .....	10
Gambar 16 Create New Repository .....	11
Gambar 17 Setup Repository .....	12
Gambar 18 git remote add .....	12
Gambar 19 git remote -v .....	12
Gambar 20 git push origin master .....	13
Gambar 21 Repository GitHub .....	13
Gambar 22 Fork Project .....	14
Gambar 23 Forking Process .....	15
Gambar 24 new repo from fork .....	15
Gambar 25 clone repository .....	16
Gambar 26 Compare & pull request .....	17
Gambar 27 Open a pull request .....	18
Gambar 28 Pull request succeeded .....	18
Gambar 29 Pull Request .....	19
Gambar 30 merge pull request .....	19
Gambar 31 Confirm pull request .....	20
Gambar 32 Merged .....	20

## 1. Intro to Git

Git adalah *version control system* yang digunakan bersama-sama oleh para developer untuk mengembangkan perangkat lunak. Seperti pada Gambar 1, Fungsi utama dari Git adalah untuk mengatur versi dari source code perangkat lunak yang sedang dibangun dengan memberi tanda file mana yang dimodifikasi dan baris mana yang ditambah bahkan dihilangkan.

Git memudahkan developer untuk melihat perubahan yang terjadi pada source code, dibandingkan harus membuat file baru seperti `home.js`, `home_1.js`, `home_2.js`, dan `home_fix.js`. Dengan menggunakan Git, developer tidak perlu khawatir akan terjadi bentrok antar source code yang dikerjakannya. Karena setiap developer dapat memiliki branch masing-masing sebagai workspace-nya seperti digambarkan pada Gambar 2.



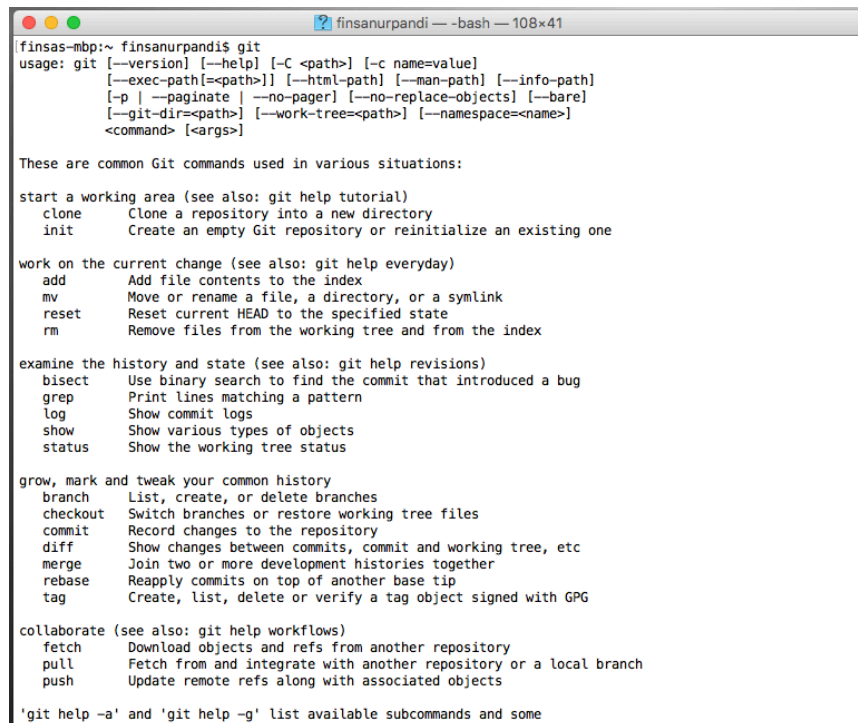
Gambar 1 Git Version Control



Gambar 2 Branch System

## 2. Git Install

Untuk dapat menggunakan Git, pertama yang harus dilakukan adalah download file installer git di <https://git-scm.com/downloads>. Installer tersedia untuk Windows, MacOS, dan Linux. Setelah selesai install git, untuk mengetahui apakah git sudah benar ter-install atau tidak, bisa melakukan pengecekan melalui terminal/cmd dengan mengetikan kata git. Jika hasil keluaran seperti pada Gambar 3, maka instalasi berhasil.



```
finsanurpandi@finsanurpandi$ git
usage: git [--version] [--help] [-C <path>] [--name=value]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  reset      Reset current HEAD to the specified state
  rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect     Use binary search to find the commit that introduced a bug
  grep       Print lines matching a pattern
  log        Show commit logs
  show       Show various types of objects
  status     Show the working tree status

grow, mark and tweak your common history
  branch     List, create, or delete branches
  checkout   Switch branches or restore working tree files
  commit     Record changes to the repository
  diff       Show changes between commits, commit and working tree, etc
  merge      Join two or more development histories together
  rebase     Reapply commits on top of another base tip
  tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
  fetch      Download objects and refs from another repository
  pull       Fetch from and integrate with another repository or a local branch
  push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
```

Gambar 3 Git on Terminal

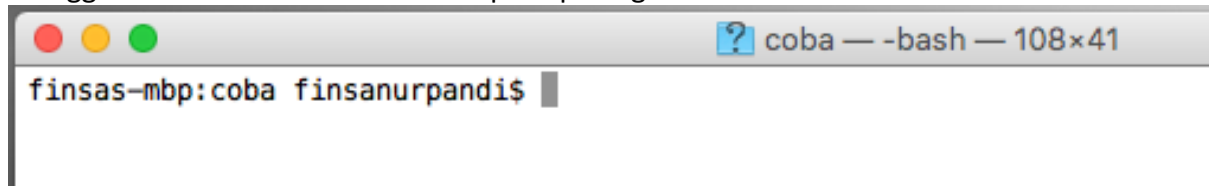
Dalam penggunaan Git ada dua acara, pertama dengan menggunakan Terminal/Command Prompt dan kedua menggunakan Git Desktop. Pada artikel ini, setiap contoh penggunaan git dengan menggunakan Terminal/Command Prompt.

## 3. Working Locally

Sebelum mulai menggunakan Git, ada beberapa perintah dasar yang harus diketahui seperti di bawah ini:

- Git init: Untuk membuat repository pada folder lokal
- Git status: Untuk mengetahui status dari repository Lokal, apakah ada file yang ditambahkan/dihilangkan dan dimodifikasi.
- Git diff: untuk mengetahui list code yang dilakukan perubahan.
- Git add: untuk menambahkan file ke repository local
- Git commit: untuk menyimpan perubahan yang dilakukan
- Git push: untuk mengirimkan perubahan dari repository local setelah melakukan commit ke remote repository
- Git branch: untuk melihat seluruh branch yang ada pada repository
- Git checkout: untuk menukar branch yang aktif dengan branch yang lainnya
- Git merge: untuk menggabungkan branch yang aktif dengan branch yang dipilih
- Git clone: untuk membuat Salinan repository

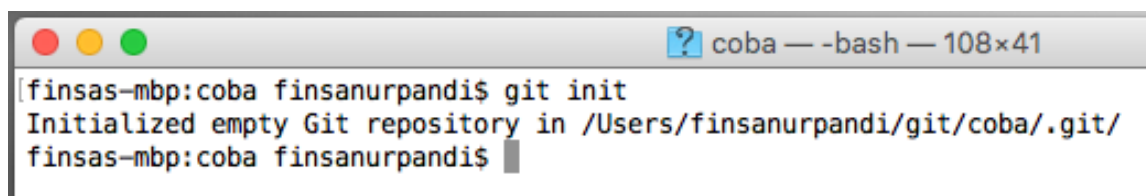
Langkah selanjutnya, buat sebuah folder repository project. Pada contoh ini, penulis menggunakan nama folder “coba” seperti pada gambar di bawah ini.



*Gambar 4 Memilih Folder Project*

Langkah selanjutnya adalah sebagai berikut :

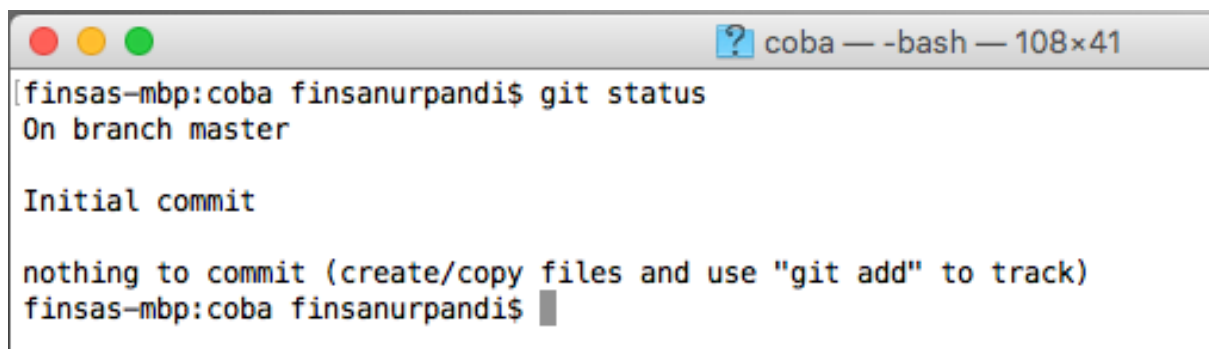
### 3.1 git init



*Gambar 5 git init*

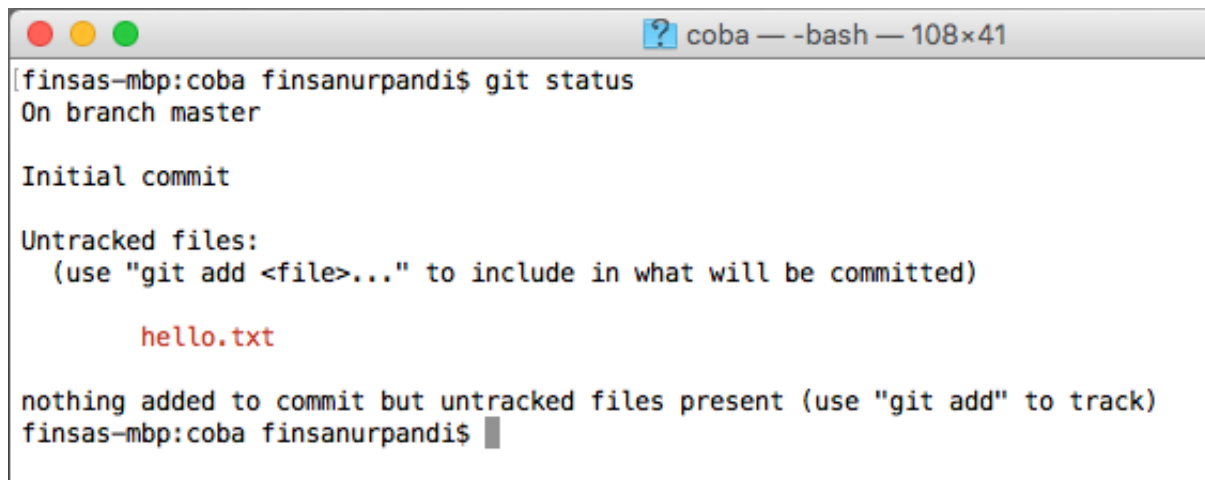
Git init akan menghasilkan folder .git sebagai tanda jika repository local sudah dibuat. Folder .git ini bersifat hidden.

### 3.2 git status



*Gambar 6 git status*

Pada Gambar 6, menunjukan belum ada file yang ditambahkan. Selanjutnya, tambahkan file \*.txt pada folder coba. Lalu lakukan git status kembali, hasilnya akan seperti pada gambar di bawah ini.



```

[?] coba — -bash — 108x41
[finsas-mbp:coba finsanurpandi$ git status
On branch master

Initial commit

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

        hello.txt

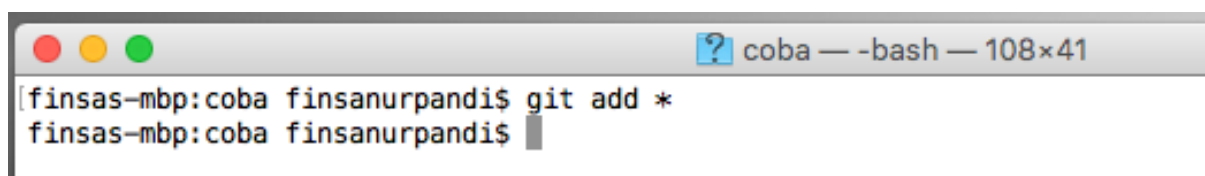
nothing added to commit but untracked files present (use "git add" to track)
finsas-mbp:coba finsanurpandi$

```

Gambar 7 git status

Hasil dari git status terlihat jika ada file baru yang belum ditambahkan ke dalam repository local. File tersebut dengan nama hello.txt. seperti yang sudah dijelaskan sebelumnya, git status akan menampilkan file baru atau file yang dimodifikasi dengan warna merah. File tersebut harus ditambahkan ke dalam repository local dengan menggunakan perintah git add seperti pada langkah selanjutnya.

### 3.3 git add



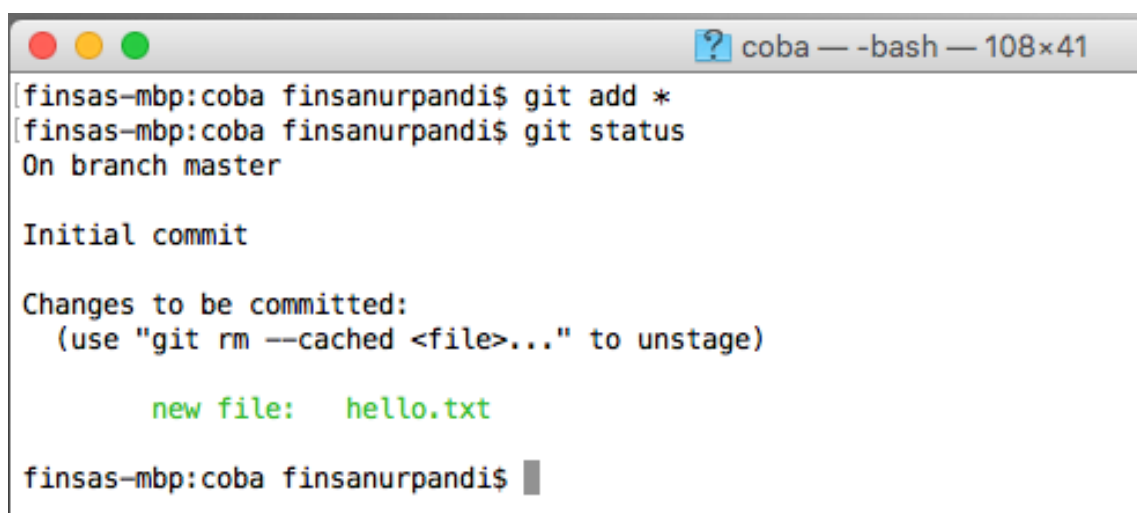
```

[?] coba — -bash — 108x41
[finsas-mbp:coba finsanurpandi$ git add *
finsas-mbp:coba finsanurpandi$

```

Gambar 8 git add

Tanda \* pada git add adalah untuk menambahkan semua file ke dalam repository. Bisa saja hanya menuliskan nama file nya saja, seperti git add hello.txt. setelah melakukan git add, lakukan pengecekan dengan melakukan perintah git status. Hasil dari git status, nama file yang berhasil ditambahkan akan berwarna hijau seperti pada gambar di bawah ini.



```

[?] coba — -bash — 108x41
[finsas-mbp:coba finsanurpandi$ git add *
[finsas-mbp:coba finsanurpandi$ git status
On branch master

Initial commit

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

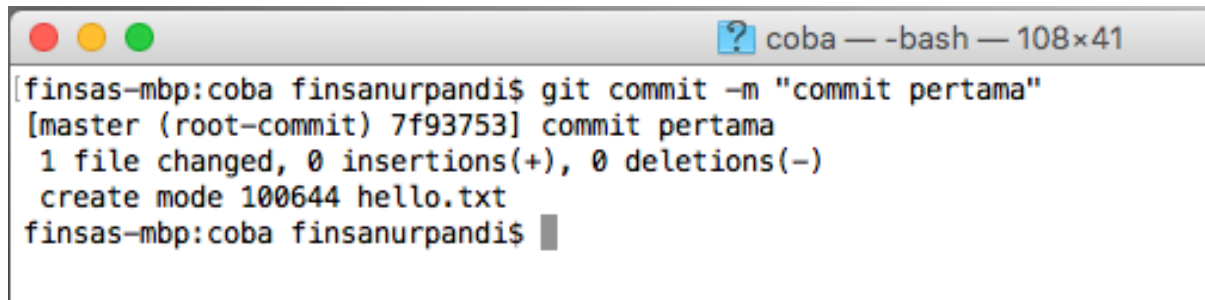
        new file:   hello.txt

finsas-mbp:coba finsanurpandi$

```

Gambar 9 git status

### 3.4 git commit



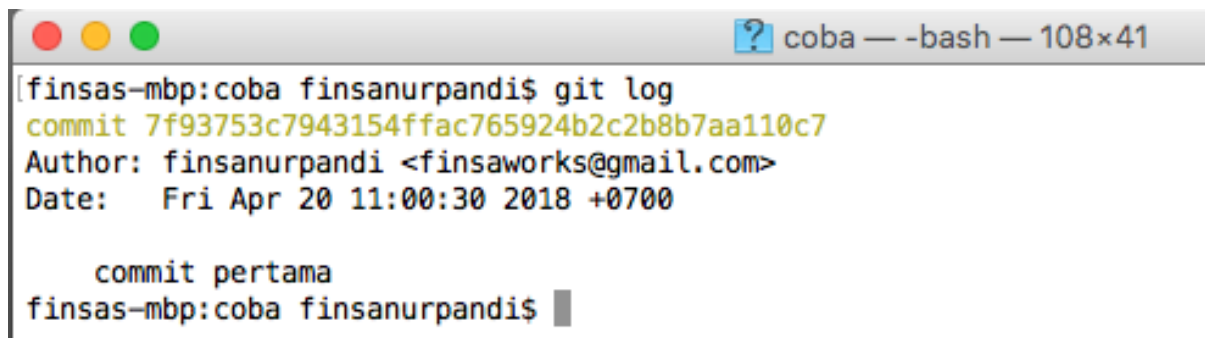
```
[finsas-mbp:coba finsanurpandi$ git commit -m "commit pertama"
[master (root-commit) 7f93753] commit pertama
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
finsas-mbp:coba finsanurpandi$
```

Gambar 10 git commit

Git commit untuk menyimpan setiap perubahan yang terjadi. Contoh pada gambar di atas, perubahan tersebut diberi tanda dengan pesan “commit pertama”. Biasanya, jika melakukan pertama kali menggunakan git dan melakukan commit, kita akan diarahkan untuk mengisi konfigurasi author yang berisi nama dan email dengan menggunakan perintah seperti di bawah ini.

```
$ git config --global user.name "finsanurpandi"
$ git config --global user.email "finsaworks@gmail.com"
```

### 3.5 git log



```
[finsas-mbp:coba finsanurpandi$ git log
commit 7f93753c7943154ffac765924b2c2b8b7aa110c7
Author: finsanurpandi <finsaworks@gmail.com>
Date:   Fri Apr 20 11:00:30 2018 +0700

    commit pertama
finsas-mbp:coba finsanurpandi$
```

Gambar 11 git log

Git log berfungsi untuk menampilkan history dari commit yang telah dilakukan. Semua commit yang telah dilakukan akan ditampilkan seperti pada Gambar 11.

Jika terdapat perubahan kembali pada folder “coba” seperti menambahkan file baru atau melakukan modifikasi pada file “hello.txt”, maka untuk menambahkan pada repository local lakukan perintah di mulai dari git status hingga git commit.

### 3.6 git branch

Pencabangan pada git berguna untuk mencegah konflik ketika bekerja dengan tim pada suatu repository. Konflik dapat terjadi karena kode yang ditulis berbeda dengan yang lainnya. Hal ini dapat terjadi jika anggota A menulis kode dengan suatu algoritma yang dia ketahui, sedangkan anggota B menulis dengan algoritma berbeda. Keduanya melakukan commit dan source code menjadi berantakan karena terjadi konflik. Untuk menghindari hal tersebut, setiap anggota tim mempunyai cabang tersendiri. Anggota A akan mengerjakan fitur X, maka dia harus membuat cabang sendiri. Dengan begitu anggota A dapat bebas melakukan apapun di cabang nya tanpa mengganggu cabang utama (Master).

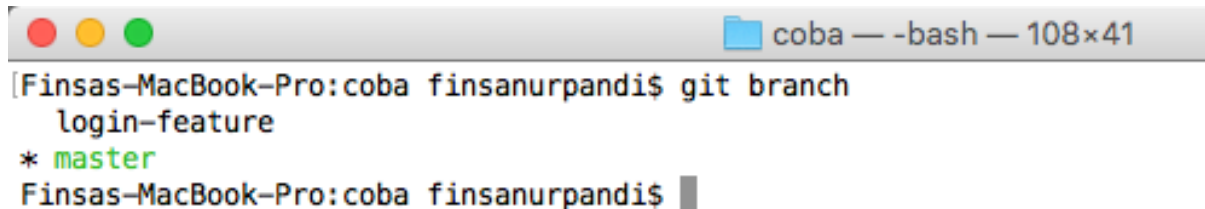


Untuk membuat cabang baru, dengan menggunakan perintah **git branch**, kemudian diikuti dengan nama cabangnya.

```
$ git branch login-feature
```

Untuk melihat cabang yang ada pada suatu repository, dengan menggunakan perintah **git branch**. Hasil yang akan muncul adalah keseluruhan cabang seperti pada Gambar 12.

```
$ git branch
```



*Gambar 12 git branch*

Untuk menghapus cabang, dapat menggunakan perintah **git branch -d nama\_cabang**.

```
$ git branch -d login-feature
```

### 3.7 git checkout

Perintah **git checkout** dapat mengembalikan kondisi file seperti waktu yang dituju. Dengan menggunakan perintah **git checkout nomor\_commit**, dapat mengembalikan semua file seperti pada keadaan commit tersebut. Akan tetapi pengembalian file tersebut bersifat sementara, karena tidak tersimpan dalam database git. Jadi, perintah **git checkout** seperti untuk mengecek file di setiap commit.

```
$ git checkout 57676dbf3aebc141f06de08c0be25a4df6c7980a
```

Selain itu juga, perintah ini dapat digunakan untuk berpindah cabang dan membuat cabang baru. Untuk dapat berpindah ke suatu cabang, dapat menggunakan perintah **git checkout nama\_cabang**.

```
$ git checkout new-feature
```

Untuk membuat cabang baru dan berpindah ke cabang baru tersebut dapat menggunakan perintah **git checkout -b nama\_cabang**.

```
$ git checkout -b login-feature
```

Perintah tersebut sebenarnya merupakan dua perintah yang dilakukan dalam satu baris perintah. Urutan untuk perintah tersebut adalah seperti di bawah ini.

```
$ git branch login-feature  
$ git checkout login-feature
```

### 3.8 git merge

Misalkan Anggota A telah selesai membuat fitur login yang disimpan dalam cabang **login-feature**, maka cabang tersebut siap untuk digabungkan dengan cabang utama (Master). Untuk dapat melakukan penggabungan, kita harus *checkout* dari cabang **login-feature** sebelum melakukan merge.

```
$ git checkout master
```

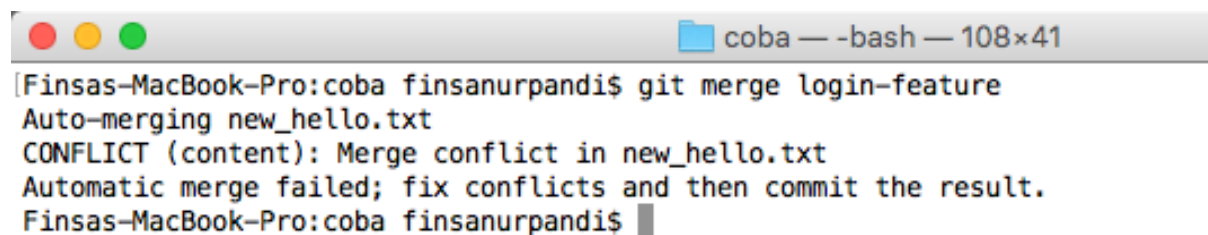
Setelah menuju cabang utama, barulah kita dapat menggabungkan dengan menggunakan perintah **git merge nama\_cabang**.

```
$ git merge login-feature
```

Jika tidak terjadi konflik, file yang terdapat di Master sudah ter-update sesuai dengan yang di cabang login-feature. Untuk penjelasan mengenai konflik yang dapat terjadi ketika melakukan merge, akan dijelaskan pada pembahasan selanjutnya.

### 3.9 Basic Merge Conflict

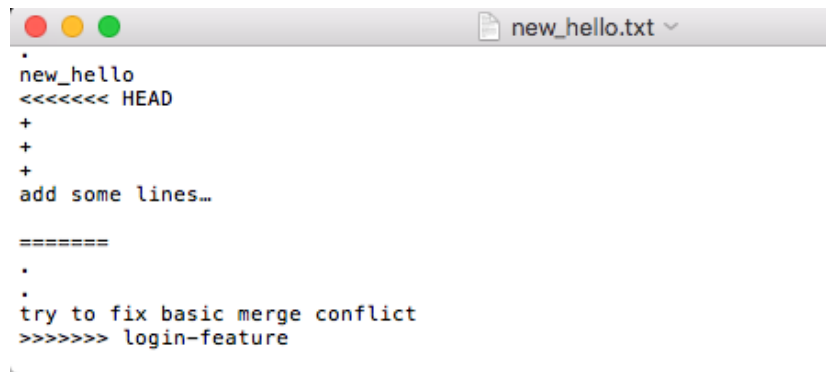
Proses menggabungkan suatu cabang dengan cabang utama tidak selamanya berjalan mulus. Jika kita mengubah isi suatu bagian dari file yang sama secara berbeda pada kedua cabang yang akan digabungkan, akan terjadi konflik. Pada contoh di bawah ini, diasumsikan telah memiliki cabang lain dengan nama login-feature dengan file yang berada di dalamnya adalah new\_hello.txt. File tersebut yang berada di cabang Master dan login-feature masing-masing telah diubah secara berbeda satu sama lainnya dan selanjutnya dilakukan merge. Maka merge tersebut akan terjadi konflik dengan pesan konflik tersebut seperti pada gambar di bawah ini.

A screenshot of a terminal window on a Mac. The window title is 'coba — -bash — 108x41'. The terminal shows the command 'git merge login-feature' being executed. The output indicates an auto-merging attempt for 'new\_hello.txt' which failed due to a content conflict. The message says: 'CONFLICT (content): Merge conflict in new\_hello.txt. Automatic merge failed; fix conflicts and then commit the result.' The prompt returns to 'Finsas-MacBook-Pro:coba finsanurpandi\$' with a cursor.

```
[Finsas-MacBook-Pro:coba finsanurpandi$ git merge login-feature
Auto-merging new_hello.txt
CONFLICT (content): Merge conflict in new_hello.txt
Automatic merge failed; fix conflicts and then commit the result.
Finsas-MacBook-Pro:coba finsanurpandi$ ]
```

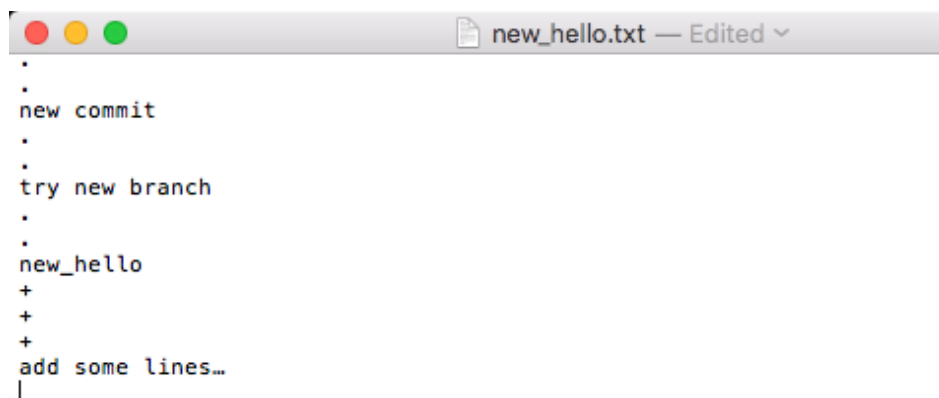
*Gambar 13 git merge conflict*

Pesan tersebut, mengharuskan untuk memperbaiki kode dalam suatu file. Jika pada contoh di atas, file yang terjadi konflik adalah file new\_hello.txt. Selanjutnya buka file yang dimaksud. Konten file tersebut akan seperti di bawah ini.



Gambar 14 file conflict

Tanda <<<<<< HEAD merupakan tanda dimulai perbedaan antara file yang berada di Master dengan cabang yang akan dilakukan merge. Kedua kode cabang dipisahkan dengan tanda =====. Selanjutnya adalah tinggal memperbaiki kode dengan mengeliminasi salah satu dari kode tersebut.



Gambar 15 fix conflict

Setelah selesai diperbaiki, lakukan `git add` dan `git commit` untuk menyimpan perubahan tersebut.

## 4. Working Remotely

Selain secara lokal, git dapat digunakan secara remote. Tetapi, jika secara remote membutuhkan software version control seperti GitHub, GitLab, BitBucket, dll. Pada contoh selanjutnya, akan menggunakan version control system GitHub.

Untuk mengakses GitHub, tinggal mengakses <https://github.com/>. Lakukan register untuk anggota baru dan verifikasi email untuk mengaktifkan akun. Pada contoh kasus kali ini, masih menggunakan repository lokal pada pembahasan sebelumnya. Yang akan dicoba, repository lokal tersebut di “push” ke remote repository GitHub.

Setelah membuat akun GitHub, lalu buat repository baru dengan memilih menu “+” yang berada di sebelah kanan atas. Muncul *dropdown*, lalu pilih *New Repository*. Akan muncul tampilan seperti pada Gambar 12, isi *Repository Name* dengan coba. Sifat repository nya public.


## Create a new repository

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


---

Owner

Repository name

 finsanurpandi ▾


 / 

coba 


Great repository names are short and memorable. Need inspiration? How about **probable-funicular**.

Description (optional)

---

☒  **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** ▾

 | 

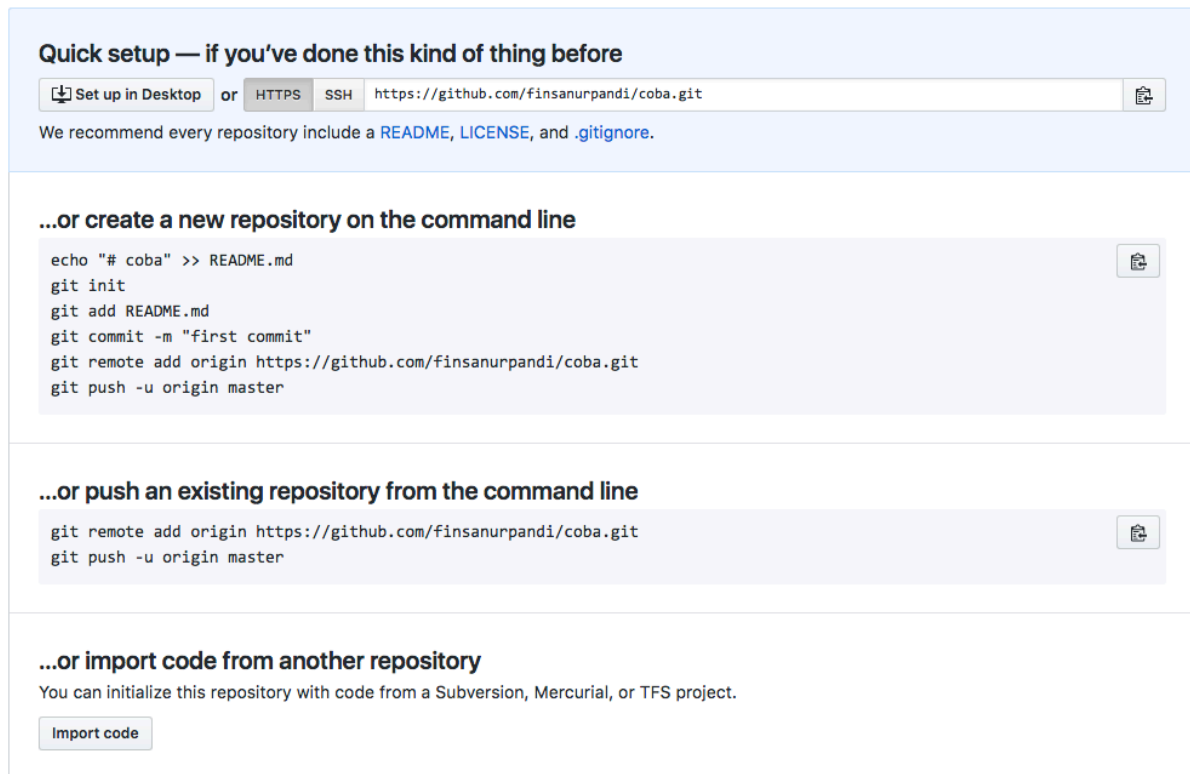
Add a license: **None** ▾ 

---

Create repository

Gambar 16 Create New Repository

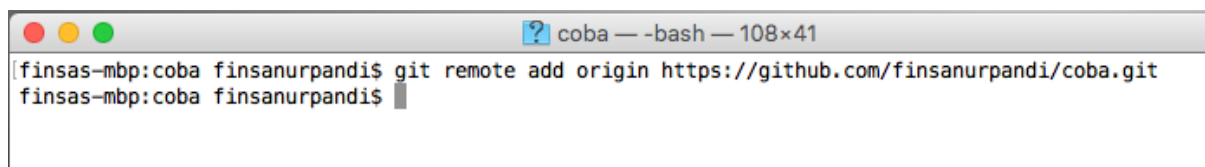
Setelah berhasil membuat repository, akan muncul tampilan cara untuk *push* ke *remote repository* seperti pada Gambar 13. Yang harus diperhatikan adalah pada bagian “**...or push an existing repository from the command line**”, karena dua baris perintah tersebut yang akan digunakan untuk mengirimkan repository lokal yang sudah dibuat sebelumnya ke remote repository GitHub.



Gambar 17 Setup Repository

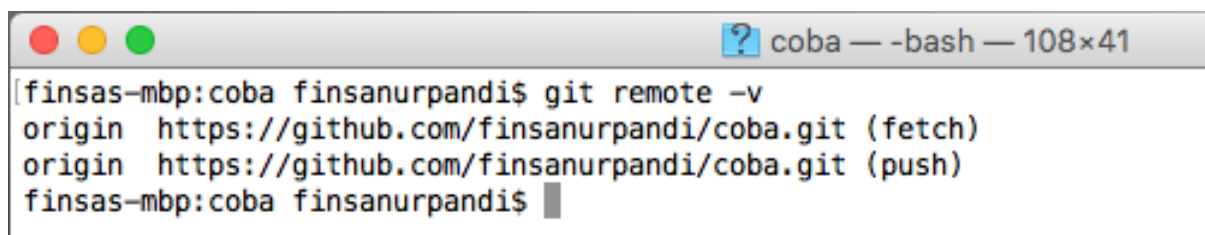
#### 4.1 git remote

Kembali ke command line. Sebelum melakukan push, kita harus melakukan konfigurasi setup ke remote repository. Setup konfigurasi remote ini hanya sekali dilakukan. Selanjutnya, hanya tinggal melakukan push saja. Untuk melakukan setup tersebut, tinggal copy baris code yang ada di GitHub sebelumnya seperti pada Gambar 14.



Gambar 18 git remote add

Jika keluarannya seperti Gambar di atas, maka setup remote repository nya berhasil. Untuk melakukan pengecekan apakah source remote repository nya sudah tersimpan atau tidak, bisa menggunakan perintah **git remote -v** seperti pada gambar di bawah ini.

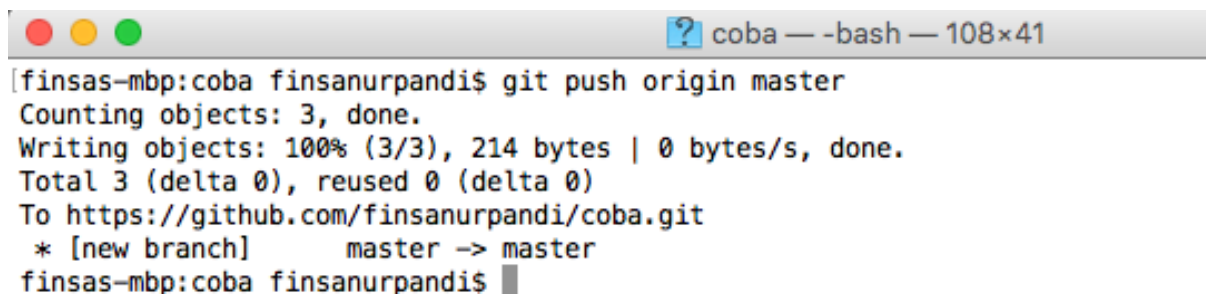


Gambar 19 git remote -v

Langkah selanjutnya adalah melakukan push ke remote repository dengan menggunakan git push.

## 4.2 git push

Setelah melakukan setup remote repository, lakukan push ke remote repository git hub. Ketikkan perintah **git push origin master** seperti pada Gambar 16. **Origin** merupakan nama default dari git remote repository yang dituju. Maka, pada contoh di atas **origin** sama dengan <https://github.com/finsanurpandi/coba.git>. **Master** merupakan branch dari remote repository. Jika menggunakan branch yang lain, maka **master** dapat diganti dengan nama branch tersebut.



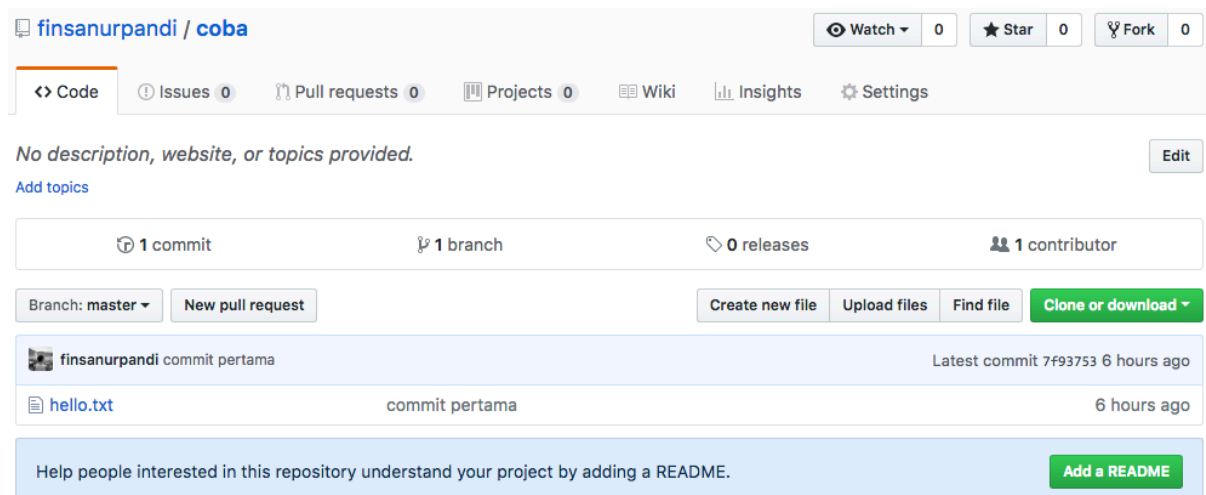
```

coba — -bash — 108x41
[finsas-mbp:coba finsanurpandi$ git push origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 214 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/finsanurpandi/coba.git
 * [new branch]      master -> master
finsas-mbp:coba finsanurpandi$

```

Gambar 20 git push origin master

Setelah proses tadi selesai, kembali ke repository di GitHub, lalu refresh. Maka akan muncul yang sebelumnya berada di repository lokal di github seperti pada gambar di bawah ini.



Gambar 21 Repository GitHub

Untuk melakukan penambahan file atau modifikasi pada file yang ada, maka lakukan secara local terlebih dahulu. Lakukan perintah dari **git status**, **git add \***, **git commit**, dan terakhir lakukan **git push origin master** untuk melakukan update pada remote repository GitHub.

## 5. How to contribute?

Selain menyimpan suatu project ke dalam remote repository GitHub, kita juga dapat berkontribusi pada project milik orang lain. Karena dalam setiap project pada GitHub bersifat public, jadi siapapun dapat menjadi contributor pada project lain. Tetapi dengan beberapa persyaratan yang ditentukan dari kriteria project yang dikerjakan oleh pemilik repository project tersebut.

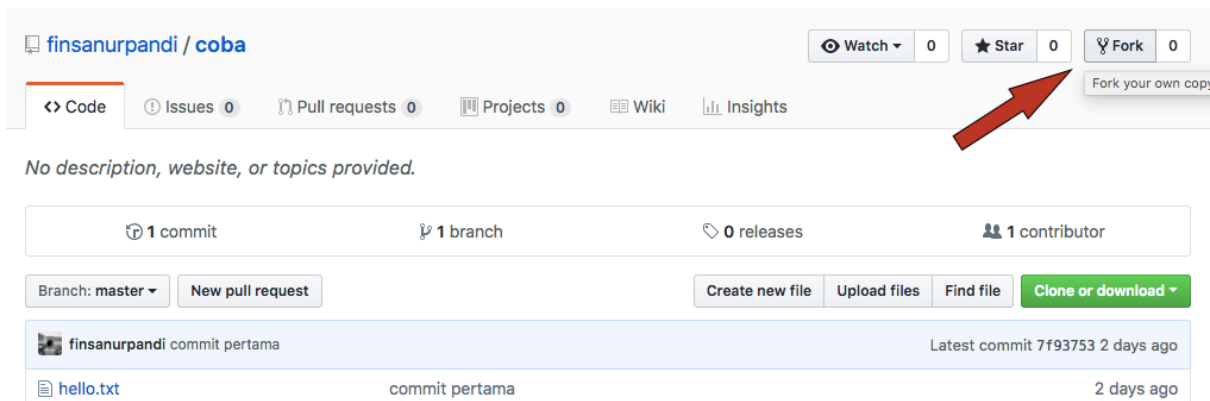
Pada contoh kali ini, akan dijelaskan beberapa langkah penting yang harus dilakukan oleh seorang contributor dan pemilik project. Yang akan dijelaskan pertama adalah bagaimana cara untuk menjadi contributor dalam GitHub.

### 5.1 Contributor

Selain dapat menyimpan file project kita di github, kita juga dapat ikut berkontribusi pada project milik orang lain. Ada beberapa cara yang dapat dilakukan agar dapat ikut berkontribusi. Sebaiknya, etika dalam berkontribusi pada project lain adalah menghubungi pemilik project mengenai ketertarikan untuk ikut mengembangkan project tersebut. Dan juga harus membaca terlebih dahulu, arah pengembangan dari project tersebut. Setelah disetujui, berikut langkah-langkah yang harus dilakukan untuk mulai berkontribusi.

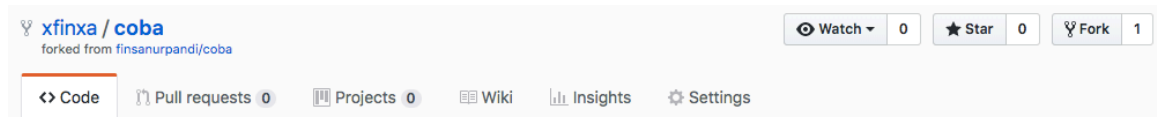
#### a. Fork project

Diasumsikan kita telah mencari suatu project, dan sedang berada di dalam halaman repository project tersebut. Langkah pertama yang harus dilakukan adalah fork project tersebut dengan cara meng-klik tombol Fork yang berada di sebelah kanan atas seperti pada Gambar 22.



Gambar 22 Fork Project

Fork ini berfungsi untuk meng-copy keseluruhan file project dari repository pemilik project ke repository milik kita. Tunggulah beberapa saat, hingga proses forking selesai.



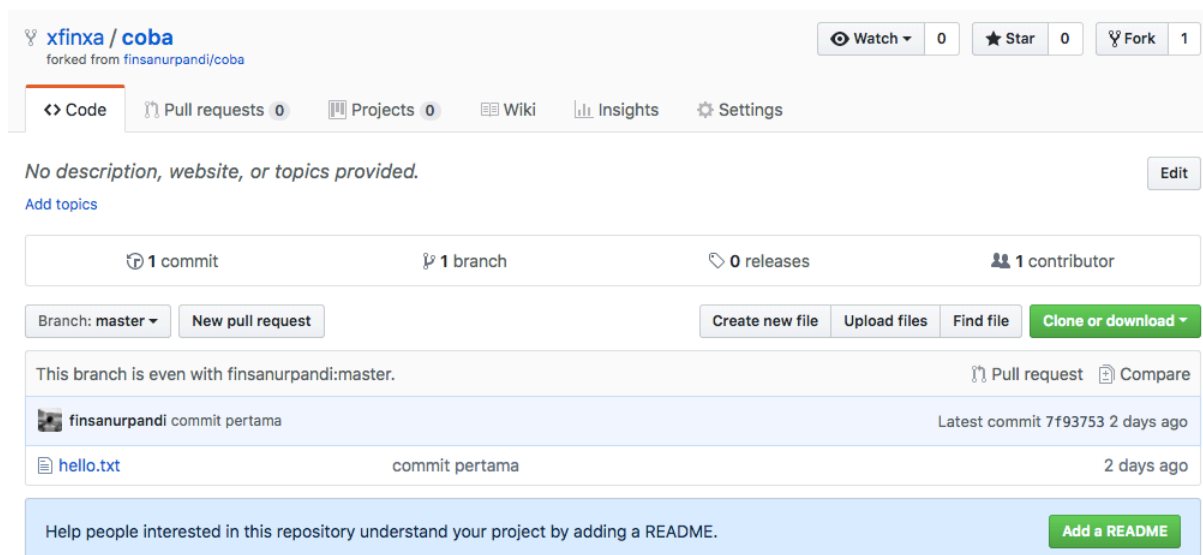
## Forking finsanurpandi/coba

It should only take a few seconds.



Gambar 23 Forking Process

Jika forking selesai dan berhasil, maka project yang kita fork tadi akan muncul di repository milik kita seperti pada gambar di bawah ini.



Gambar 24 new repo from fork

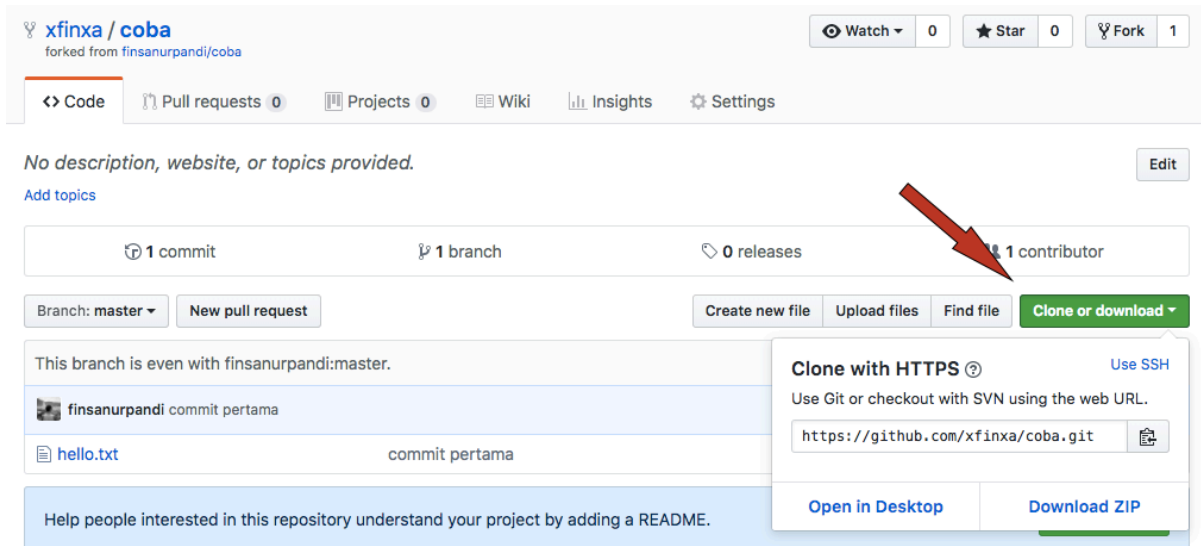
### b. Clone project

Setelah berada di repo milik kita, langkah selanjutnya adalah menduplikasi project tersebut ke repo local. Sebelumnya, copy link <https://github.com/xfinxa/coba> yang berada di sebelah kanan dengan menekan tombol clone or download seperti pada Gambar 25. Setelah itu, melalui command line arahkan ke folder mana repo tersebut akan disimpan. Jika sudah, lakukan perintah **git clone alamat\_repo**.

```
$ git clone https://github.com/xfinxa/coba.git
```

Jika berhasil, repo tersebut sudah berada di folder lokal.





Gambar 25 clone repository

## c. Add upstream

```
$ git remote add upstream https://github.com/finsanurpandi/coba.git
```

Untuk melakukan pengecekan apakah upstream sudah berhasil ditambahkan atau tidak, lakukan perintah **git remote -v**. Hasil dari perintah tersebut seperti pada gambar di bawah ini.

## d. New Branch

Sebelum mulai berkontribusi, sebaiknya buat branch baru sebagai tempat kita melakukan pengembangan. Salah satu alasan pembuatan branch baru adalah untuk menghindari konflik ketika melakukan merge. Gunakan perintah di bawah ini untuk melakukannya.

```
$ git checkout -b new-branch
```

Setelah itu, baru kita sudah siap untuk melakukan pengembangan dan mulai berkontribusi pada project tersebut.

## e. Start Coding

Mulailah melakukan pengembangan pada project tersebut. Setelah melakukan perubahan, lakukan **git add .** dan **git commit**.

## f. Before Pull request

Sebelum melakukan pull request, ada beberapa langkah yang harus dilakukan. Pertama, pindah ke cabang Master dengan perintah berikut.

```
$ git checkout master
```

Selanjutnya, kita akan mengambil kode dari original repo. Hal ini dilakukan untuk mencegah terjadi konflik pada kontribusi kode. Konflik dapat terjadi jika dua atau lebih contributor melakukan perubahan pada satu berkas, terutama perubahan dilakukan pada baris yang sama. Lakukan langkah-langkah berikut untuk melakukannya.

```
$ git fetch upstream
$ git merge upstream/master
```

Setelah memastikan tidak ada konflik dengan original repo, pindah ke branch local development yang sebelumnya digunakan, yaitu new-branch.

```
$ git checkout new-branch
```

Setelah itu, gabungkan cabang tersebut dengan cabang utama, sehingga kontribusi dapat dikirim kembali ke original repo dengan menggunakan perintah di bawah ini.

```
$ git rebase master
```

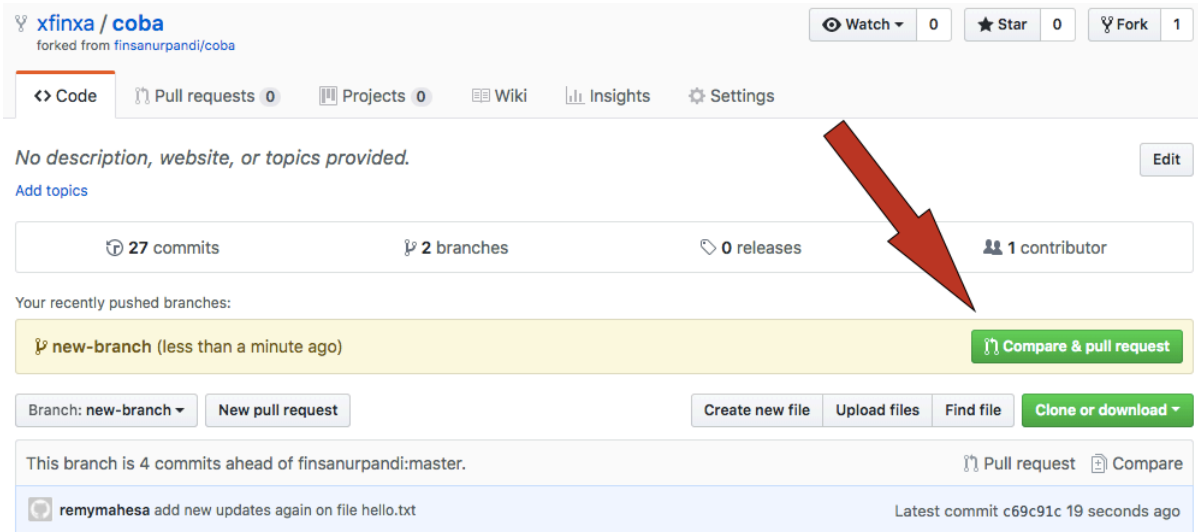
Sebelum push ke repo original pusat, lakukan push terlebih dahulu ke repository hasil fork tadi.

```
$ git push origin new-branch
```

Setelah selesai, maka hasil perubahan yang telah dilakukan siap untuk dilakukan *pull request*.

g. Pull Request

Setelah melakukan push, selanjutnya adalah melakukan pull request dan membandingkan perubahan yang telah dilakukan terhadap original repo. Untuk melakukannya, pada tampilan repository hasil fork di github akan muncul tombol **Compare & pull request** berwarna hijau sebelah kanan seperti pada Gambar 26.

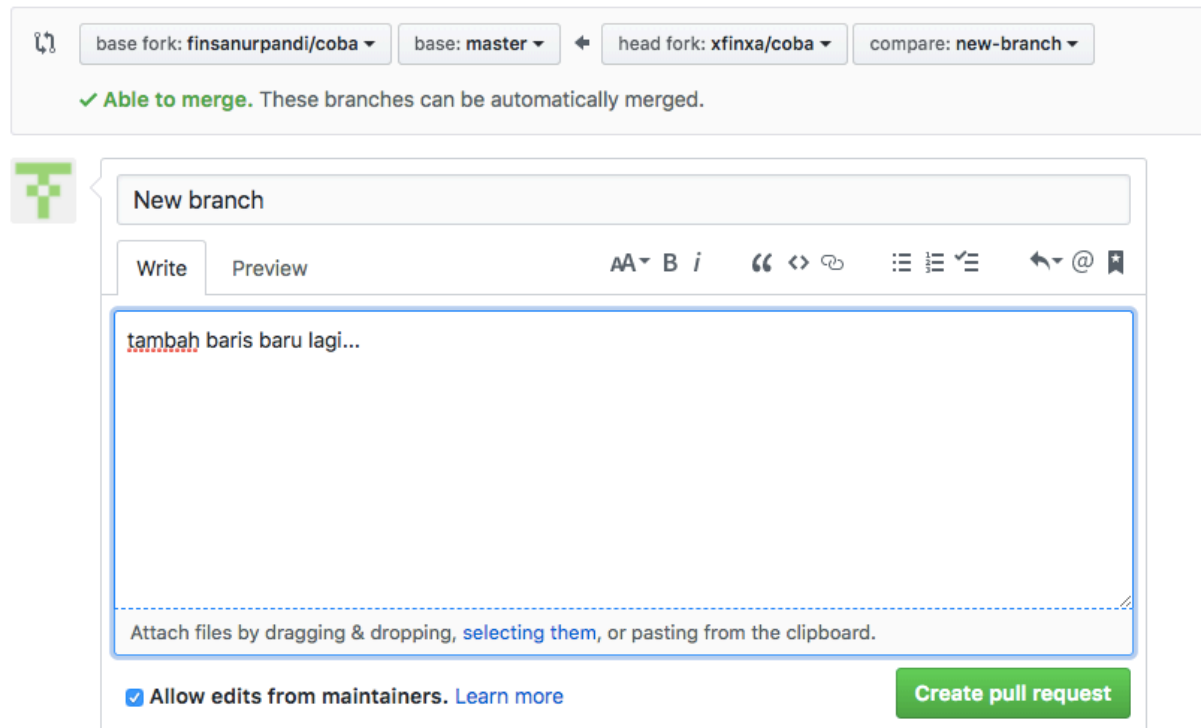


Gambar 26 Compare & pull request

Klik tombol tersebut, maka akan muncul tampilan **Open a pull request** seperti pada Gambar 27. Tambahkan keterangan mengenai perubahan apa yang telah dilakukan, agar pemilik repo dapat mengetahui tujuan dari pull request tersebut.

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

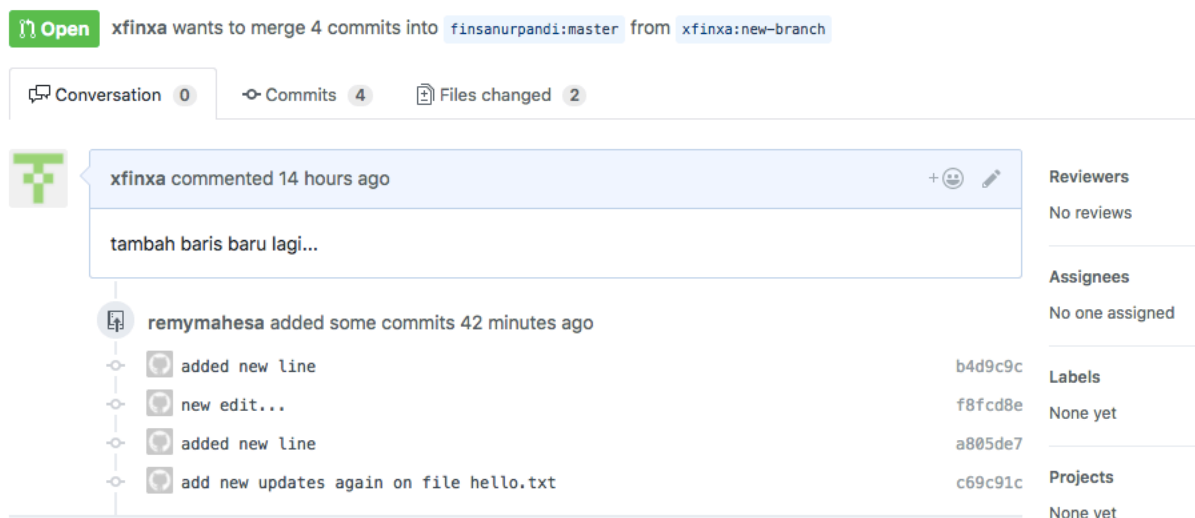


The screenshot shows the GitHub interface for creating a pull request. At the top, there are dropdown menus for 'base fork: finsanurpandi/coba', 'base: master', 'head fork: xfinxa/coba', and 'compare: new-branch'. Below these, a green checkmark indicates 'Able to merge. These branches can be automatically merged.' The main form is titled 'New branch' and has two tabs: 'Write' and 'Preview'. The 'Write' tab is active, showing a text area with the placeholder text 'tambah baris baru lagi...'. Below the text area, there is a note: 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' At the bottom of the form, there is a checkbox labeled 'Allow edits from maintainers. Learn more' which is checked, and a green button labeled 'Create pull request'.

Gambar 27 Open a pull request

Jika telah berhasil, akan muncul tampilan seperti pada Gambar 28, dan tinggal menunggu tanggapan dari pemilik original repo tersebut.

## New branch #4

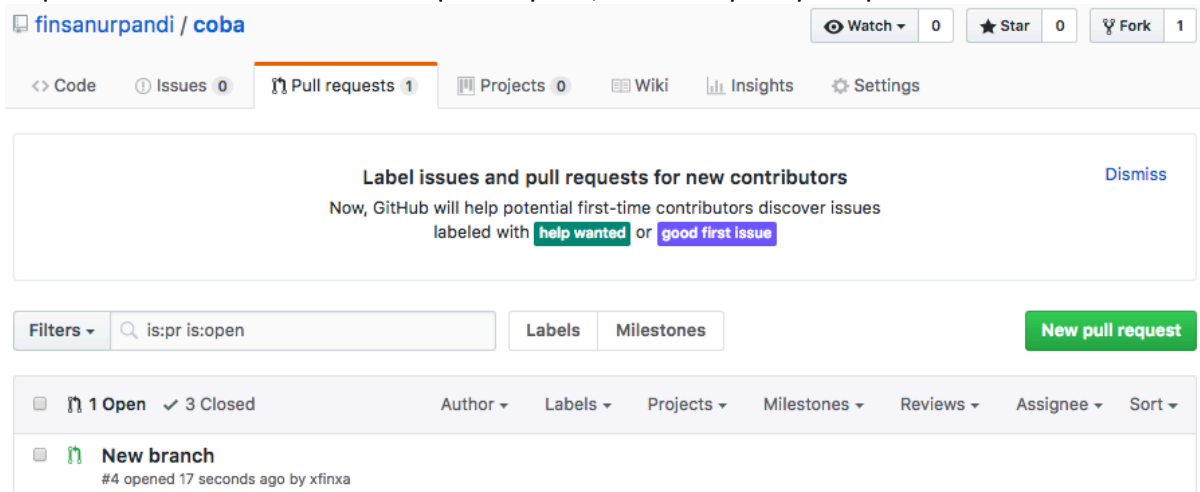


The screenshot shows the GitHub pull request page for 'New branch #4'. At the top, there is a green button labeled 'Open' and a status bar indicating 'xfinxa wants to merge 4 commits into finsanurpandi:master from xfinxa:new-branch'. Below this, there are tabs for 'Conversation' (0), 'Commits' (4), and 'Files changed' (2). The 'Conversation' tab is active, showing a comment from 'xfinxa' 14 hours ago with the text 'tambah baris baru lagi...'. Below the comment, there is a commit from 'remymahesa' 42 minutes ago with the message 'added some commits'. The commit details show four changes: 'added new line', 'new edit...', 'added new line', and 'add new updates again on file hello.txt'. On the right side, there are sections for 'Reviewers' (No reviews), 'Assignees' (No one assigned), 'Labels' (None yet), and 'Projects' (None yet).

Gambar 28 Pull request succeeded

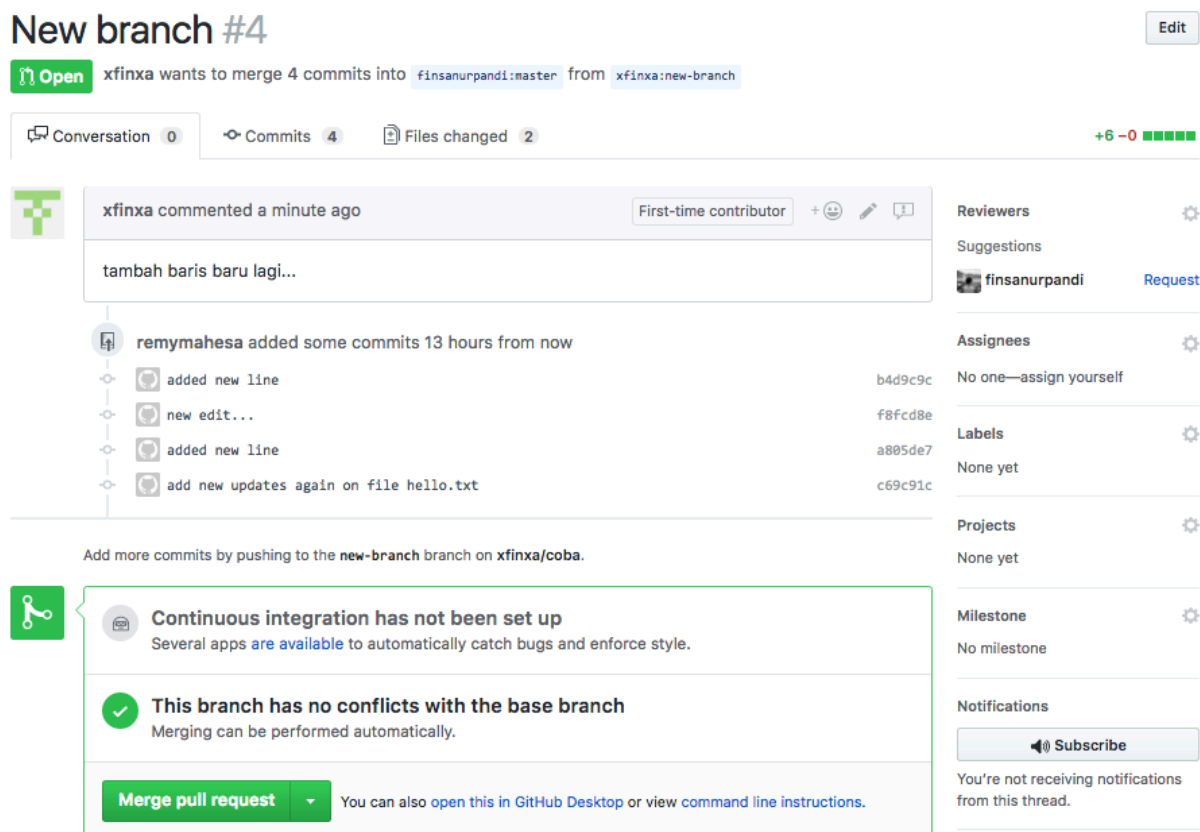
## 5.2 Project Owner

Pemilik project dapat melihat setiap pull request dari contributor di tab menu Pull request. Setelah berada di menu pull request, akan tampil seperti pada Gambar 29.



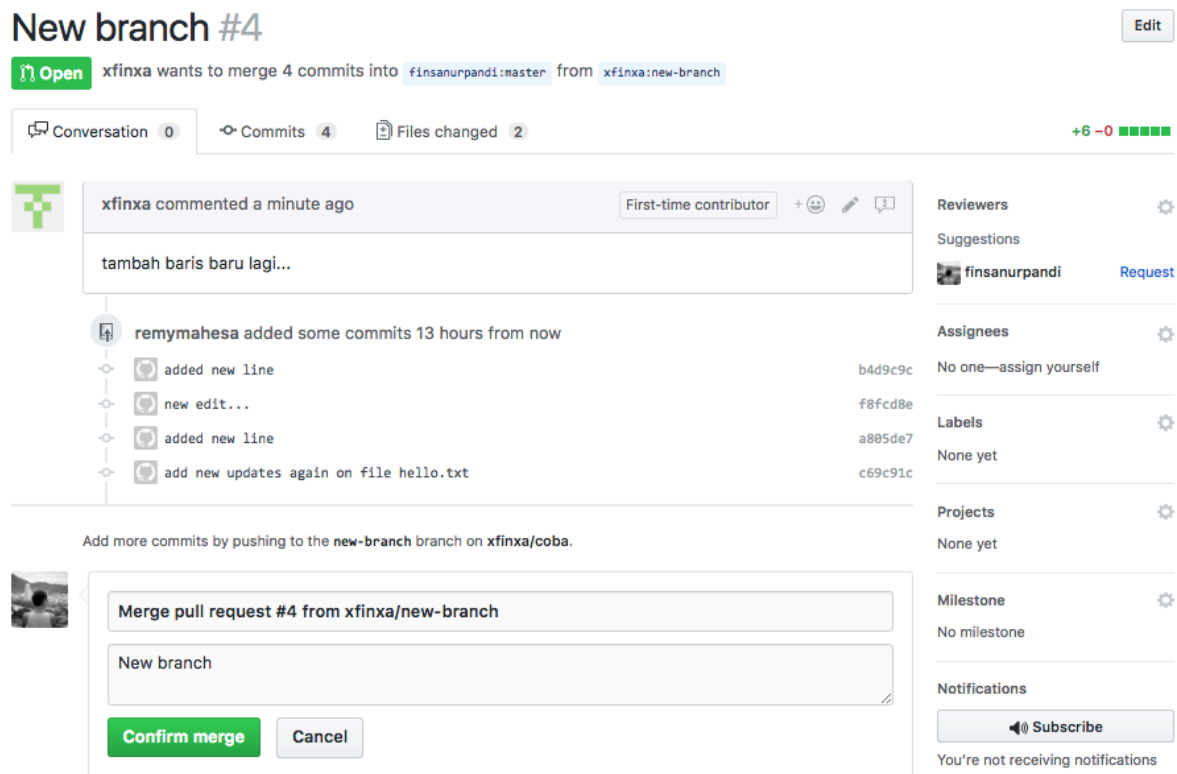
Gambar 29 Pull Request

Pada menu tersebut, akan muncul semua pull request yang dilakukan oleh setiap contributor. Untuk melihat setiap perubahan yang dilakukan, klik pada pull request tersebut, dan akan menampilkan konten seperti pada Gambar 30.



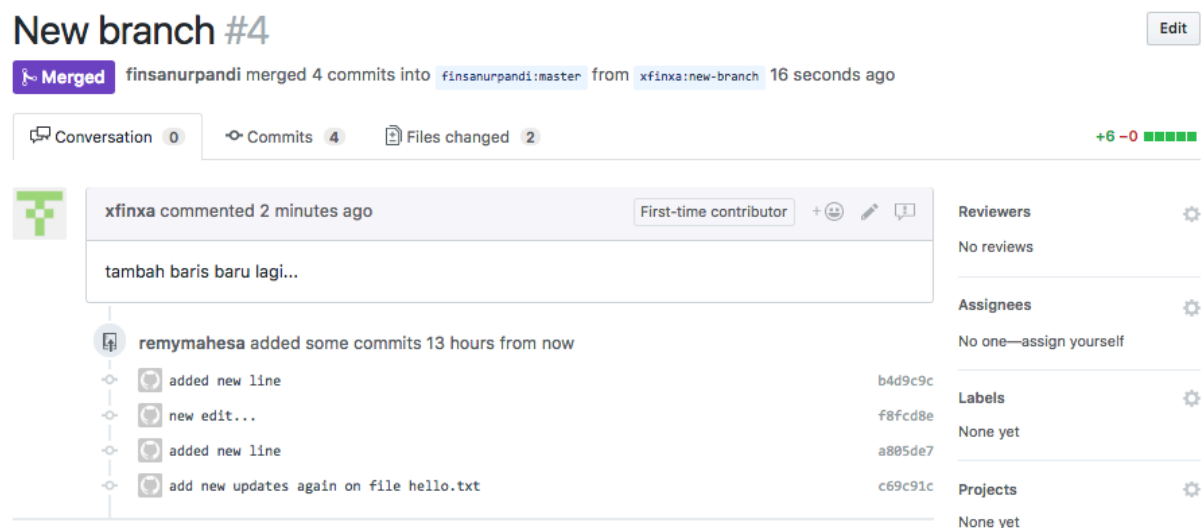
Gambar 30 Merge pull request

Jika tidak terdapat konflik, maka pull request tersebut siap untuk digabungkan. Klik tombol Merge pull request. Dan selanjutnya lakukan konfirmasi penggabungan dengan mengklik tombol Confirm merge seperti pada Gambar 31.



Gambar 31 Confirm pull request

Jika telah berhasil digabungkan, maka tanda kotak hijau dengan tulisan Open akan berubah menjadi warna Ungu dengan tulisan Merged seperti pada Gambar di bawah ini.



Gambar 32 Merged

### 5.3 Sync Local Repo with Remote Repo

Sebagai pemilik original repo, setelah melakukan merge pull request, repo yang ter-update hanya baru di remote repo saja. Local repo belum ter-update. Untuk memperbaharui local repo sesuai dengan yang ada di remote repo, lakukan langkah-langkah berikut.

```
$ git fetch origin
```

Perintah di atas, untuk mengambil remote repo dan memperbaharui local repo sesuai dengan remote repo. Lakukan git reset –hard, hal ini dilakukan untuk menyamakan posisi commit terakhir seperti pada remote repo.

```
$ git reset -hard origin/master
```

Selanjutnya lakukan perintah di bawah ini.

```
$ git clean -f -d
```

Git clean berfungsi untuk membersihkan working tree dengan menghapus file secara rekursif yang sudah tidak digunakan lagi.

#### 5.4 Sync a Fork Repo with Original Repo

Jika terdapat update pada original repo, update tidak dapat langsung terjadi secara otomatis pada fork repo master kita. Untuk dapat memperbaharui fork repo master dengan original repo, ada beberapa tahapan yang harus dilakukan seperti di bawah ini.

```
$ git checkout master  
$ git fetch upstream  
$ git pull upstream master
```

Jika perintah di atas telah berhasil, local repo milik contributor seharusnya sudah sesuai dengan original repo. Setelah sesuai, lakukan push untuk memperbaharui isi dari fork repo milik contributor.

```
$ git push origin master
```

#### 5.5 Merge Branch with Master

Setelah melakukan tahapan sebelumnya, jika dicek terdapat ketidaksamaan antara isi dari branch dengan master, lakukan langkah-langkah berikut ini agar branch dapat sama dengan master branch nya.

```
$ git checkout new-branch  
$ git merge origin/master  
$ git push origin new-branch
```

## Referensi

1. <https://git-scm.com/doc>
2. Ferdinando Santacroce, “Git Essentials”, Packt Publishing
3. Evan Peter Williamson, “Intro to Git and Github”, University of Idaho Library 2016
4. <https://github.com/>