

**CERDAS MENGUASAI GIT**



---

# CERDAS MENGUASAI GIT

## Dalam 24 Jam

---

**Rolly M. Awangga**  
Politeknik Pos Indonesia



**Kreatif Industri Nusantara**

***Penulis:***

Rolly Maulana Awangga

ISBN : 978-602-53897-0-2

***Editor:***

M. Yusril Helmi Setyawan

***Penyunting:***

Syafrial Fachrie Pane

Khaera Tunnisia

Diana Asri Wijayanti

***Desain sampul dan Tata letak:***

Deza Martha Akbar

***Penerbit:***

Kreatif Industri Nusantara

***Redaksi:***

Jl. Ligar Nyawang No. 2

Bandung 40191

Tel. 022 2045-8529

Email : awangga@kreatif.co.id

***Distributor:***

Informatics Research Center

Jl. Sariasisih No. 54

Bandung 40151

Email : irc@poltekpos.ac.id

Cetakan Pertama, 2019

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara  
apapun tanpa ijin tertulis dari penerbit

*'Jika Kamu tidak dapat  
menahan lelahnya  
belajar, Maka kamu harus  
sanggup menahan  
perihnya Kebodohan.'*

*Imam Syafi'i*

## CONTRIBUTORS

---

ROLLY MAULANA AWANGGA, Informatics Research Center., Politeknik Pos Indonesia, Bandung, Indonesia



# CONTENTS IN BRIEF

---

<b>1</b>	<b>Belajar Git</b>	<b>1</b>
<b>2</b>	<b>Praktek Cepat</b>	<b>11</b>
<b>3</b>	<b>Perintah Dasar Bash</b>	<b>23</b>
<b>4</b>	<b>Mengatasi Konflik</b>	<b>31</b>
<b>5</b>	<b>Cara Membaca Pull Request</b>	<b>39</b>



# DAFTAR ISI

---

Daftar Gambar	xi
Daftar Tabel	xv
Foreword	xix
Kata Pengantar	xxi
Acknowledgments	xxiii
Acronyms	xxv
Glossary	xxvii
List of Symbols	xxix
Introduction	xxxii
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
<b>1 Belajar Git</b>	<b>1</b>
1.1 Membuat Akun Github	1
1.2 Membuat Akun Gitlab	6
<b>2 Praktek Cepat</b>	<b>11</b>
	ix

2.1	Latihan	11
2.1.1	Konfigurasi Key	11
2.1.2	Fork Repotori	12
2.1.3	Navigasi direktori dengan Git Bash	14
2.1.4	Sinkronisasi dengan Repo Utama	15
2.1.5	Bekerja dengan Git Bash	16
2.1.6	Mengatasi <i>Error</i>	18
<b>3</b>	<b>Perintah Dasar Bash</b>	<b>23</b>
3.1	Perintah Dasar Bash	23
3.1.1	fungsi perintah pada Bash	23
3.1.2	penggunaan VI editor	28
<b>4</b>	<b>Mengatasi Konflik</b>	<b>31</b>
4.1	Pada Repo Lokal	31
4.2	Pada Saat Pull Request di Web	34
4.3	Lupa Compile dan Ingin Menambah Materi Lagi	34
4.4	Bagaimana cara mudah dan cepat untuk memperbaiki error yang tidak terdeteksi	37
<b>5</b>	<b>Cara Membaca Pull Request</b>	<b>39</b>
5.1	Membaca Pull Request Pada Github	39
5.2	Standar Untuk Melakukan Approve	44
Daftar Pustaka		49
Index		51

# DAFTAR GAMBAR

---

1.1	URL Github	2
1.2	Data yang akan didaftarkan	2
1.3	Welcome To Github	3
1.4	Step pada pembuatan Github	4
1.5	Akun Github Berhasil dibuat	4
1.6	Menampilkan Repository	5
1.7	Discover Repositories	5
1.8	Start a Project	6
1.9	URL Gitlab	7
1.10	Sign in with Github	7
1.11	Halaman Gitlab	7
1.12	SSH key	8
1.13	Ambil SSH key menggunakan Git Bash	8

1.14	Copy paste SSH key Bash ke dalam SSH key Gitlab	9
1.15	Membuat Project baru	9
2.1	Perintah keygen cukup enter enter saja	12
2.2	Perintah cat untuk membaca key	13
2.3	Setting	13
2.4	New SSH key	13
2.5	Letak Tombol Fork	14
2.6	Hasil Perintah git clone	14
2.7	Hasil Perintah git clone	15
2.8	Hasil Perintah ls, cd dan git status	15
2.9	Direktori kerja git dari Explorer	16
2.10	Permintaan Setting Global	17
2.11	Setelah klik New pull request	18
2.12	Gagal Fetch Upstream	19
2.13	Berbagai macam luaran git status	21
2.14	Permintaan Merge dengan pesan standar yang keluar	22
3.1	Perintah dasar ls pada git bash	24
3.2	Perintah dasar cd pada git bash	24
3.3	Perintah dasar pwd pada git bash	25
3.4	Perintah dasar mv pada git bash	25
3.5	Perintah dasar cp pada git bash	26
3.6	melakukan commit menggunakan VIM editor	26
3.7	mengisi apa yang kita commit	27
3.8	keluar dari halaman VIM editor	27
3.9	membuat file	28
3.10	menambahkan data file	28
3.11	menyimpan file	29
3.12	menyimpan file lalu keluar	29

3.13	keluar tanpa mengubah sesuatu pada file	30
4.1	Muncul editor vi pada saat pull	32
4.2	Konflik Pada Saat <i>git pull</i>	33
4.3	Tanda pembatas antara versi satu dan dua yang konflik	33
4.4	Konflik yang sudah diperbaiki menjadi satu versi baru lagi	34
4.5	Konflik Pada Saat Pull Request	34
4.6	Konflik Pada Saat Pull Request	35
4.7	Hasil Merge Setelah memilih dan menghapus tanda	35
4.8	Memeriksa yang belum ditambahkan	36
4.9	Menambahkan file	36
4.10	Memastikan file yang sudah ditambah	36
4.11	Melakukan proses commit	37
4.12	Memastikan bahwa sudah up to date	37
4.13	Step terakhir	37
5.1	New Pull Request	40
5.2	Create Pull Request	40
5.3	Mengisi Kontribusi	41
5.4	Berhasil Melakukan Pull Request	41
5.5	Kontribusi yang sudah di Merged	42
5.6	Menu Commits	42
5.7	Kontribusi yang telah dilakukan	43
5.8	Pull Request yang sudah dilakukan	43
5.9	Notifikasi Pada Email	44
5.10	View Pull Request Online	45
5.11	Melihat Files Changed	45
5.12	File Kontribusi	46
5.13	Merge Pull Request	46
5.14	Confirm Merge	47
5.15	Approved Merge Pull Request	47
5.16	Closed and Comment	48



## DAFTAR TABEL

---



# Listings

---

2.1	Perintah Membuat Key	12
2.2	Perintah Membaca Public Key	12
2.3	Navigasi direktori menuju repositori	15
2.4	Set Repo Asal Sebagai Upstream	16
2.5	Perintah Sinkronisasi dengan repo asal	16
2.6	Perintah Sinkronisasi dengan repo asal	17
2.7	Kesalahan karena bukan pada direktori repositori	18
2.8	Gagal melakukan fetch upstream	19
2.9	Melihat konfigurasi git di repo komputer kita	19
2.10	Hasil perintah <i>git config -l</i>	20
2.11	Perintah menghapus upstream yang salah	20
2.12	Access Denied	20
2.13	perintah remote repo utama	20
2.14	Peringatan <i>Permission denied</i>	20



# **FOREWORD**

---

Sepatah kata dari Kaprodi, Kabag Kemahasiswaan dan Mahasiswa



# KATA PENGANTAR

---

Buku ini diciptakan bagi yang awam dengan git sekalipun.

R. M. AWANGGA

*Bandung, Jawa Barat*

*Februari, 2019*



## ACKNOWLEDGMENTS

---

Terima kasih atas semua masukan dari para mahasiswa agar bisa membuat buku ini lebih baik dan lebih mudah dimengerti.

Terima kasih ini juga ditujukan khusus untuk team IRC yang telah fokus untuk belajar dan memahami bagaimana buku ini mendampingi proses Intership.

R. M. A.



## ACRONYMS

---

ACGIH	American Conference of Governmental Industrial Hygienists
AEC	Atomic Energy Commission
OSHA	Occupational Health and Safety Commission
SAMA	Scientific Apparatus Makers Association



## GLOSSARY

---

git	Merupakan manajemen sumber kode yang dibuat oleh linus torvald.
bash	Merupakan bahasa sistem operasi berbasiskan *NIX.
linux	Sistem operasi berbasis sumber kode terbuka yang dibuat oleh Linus Torvald



# SYMBOLS

---

$A$  Amplitude

$\&$  Propositional logic symbol

$a$  Filter Coefficient

$B$  Number of Beats



# INTRODUCTION

---

ROLLY MAULANA AWANGGA, S.T., M.T.

Informatics Research Center  
Bandung, Jawa Barat, Indonesia

Pada era disruptif saat ini. git merupakan sebuah kebutuhan dalam sebuah organisasi pengembangan perangkat lunak. Buku ini diharapkan bisa menjadi penghantar para programmer, analis, IT Operation dan Project Manajer. Dalam melakukan implementasi git pada diri dan organisasinya.

Rumusnya cuman sebagai contoh aja biar keren[1].

$$ABC\mathcal{DEF}\alpha\beta\Gamma\Delta \sum_{def}^{abc} \quad (I.1)$$



# BAB 1

---

## BELAJAR GIT

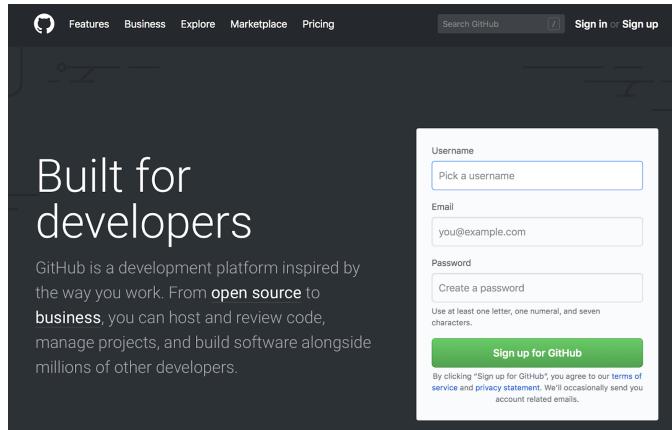
---

### 1.1 Membuat Akun Github

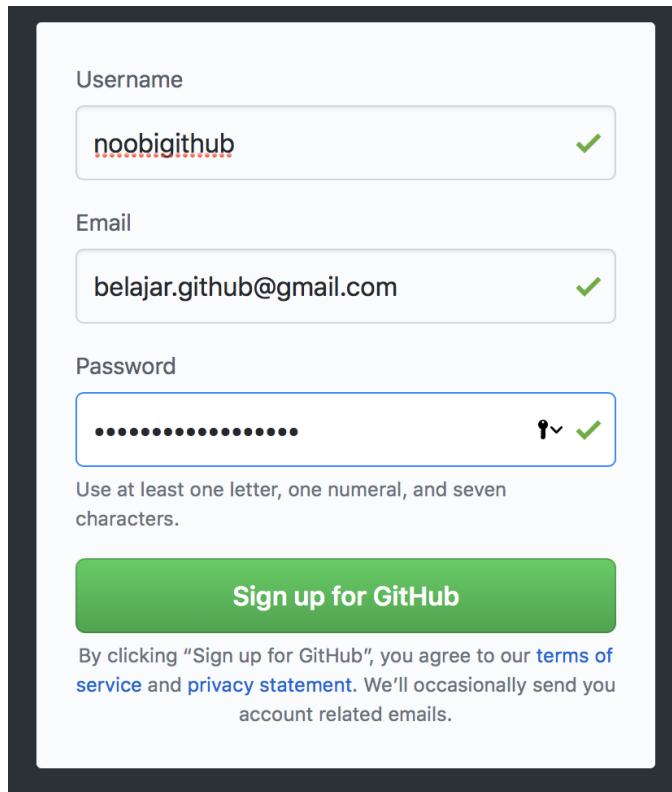
Github merupakan salah satu repository yang populer pada kalangan developer saat ini. Hal ini dikarenakan Github dapat menyimpan project-project open source mereka dimana project tersebut dapat dikembangkan oleh programmer lain baik sebagai team, maupun individual. Disaat kita melakukan perubahan terhadap suatu project, kemudian perubahan tersebut diupload ulang (push) ke repository, maka project yang lama masih tersimpan dan tidak akan hilang. Hal ini memudahkan kita untuk melihat dan menganalisa setiap perubahan yang dilakukan sebelumnya.

Sebelum mempelajari lebih dalam mengenai github, disini akan dijelaskan bagaimana cara membuat akun git pada github :

1. Pertama, buka browser dan masukkan alamat github dengan url : <https://github.com> seperti pada gambar 1.1
2. Kemudian masukkan nama user yang akan didaftarkan sebagai akun Github seperti pada contoh 1.2

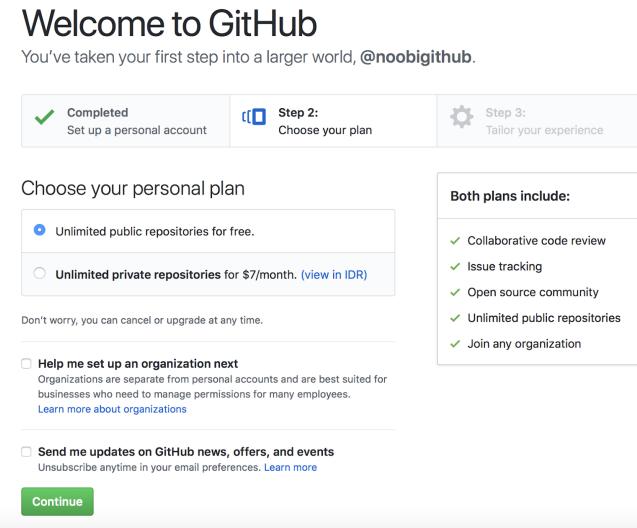


Gambar 1.1 URL Github



Gambar 1.2 Data yang akan didaftarkan

3. Setelah memasukkan username dan alamat email, selanjutnya masukkan password yang akan digunakan untuk login github
4. Setelah itu klik tombol sign up for Github dan kita akan diarahkan pada halaman seperti pada gambar 1.3



**Gambar 1.3** Welcome To Github

5. Halaman tersebut akan meminta kita untuk memilih rencana penggunaan repository pada github. Apakah kita akan menggunakan private repository atau public repository (Berbayar atau Gratis)
6. Private Repository memungkinkan repository yang kita gunakan pada github tidak terlihat oleh pengguna lainnya selain pengguna yang didaftarkan sebagai kolaborator
7. Sedangkan pada Public Repository pengguna lainnya dapat melihat repository yang kita miliki. Pada tutorial ini yang akan perlihatkan adalah Public Repository
8. Setelah memilih Public Repository kemudian klik continue untuk melanjutkan pembuatan akun github tersebut. Dan setelah di klik, maka akan diarahkan ke dalam halaman seperti pada gambar 1.4
9. Pada halaman selanjutnya kita akan diminta untuk memasukkan data profil user yang akan digunakan pada Github
10. Setelah memasukkan data profil kita telah berhasil membuat akun di Github dan akan diarahkan pada halaman seperti pada gambar 1.5

# Welcome to GitHub

You'll find endless opportunities to learn, code, and create, @noobigithub.

 Completed  
Set up a personal account

 Step 2:  
Choose your plan

 Step 3:  
Tailor your experience

How would you describe your level of programming experience?

Totally new to programming     Somewhat experienced     Very experienced

What do you plan to use GitHub for? (check all that apply)

Design     Research     School projects  
 Development     Project Management     Other (please specify)

Which is closest to how you would describe yourself?

I'm a hobbyist     I'm a student     I'm a professional  
 Other (please specify)

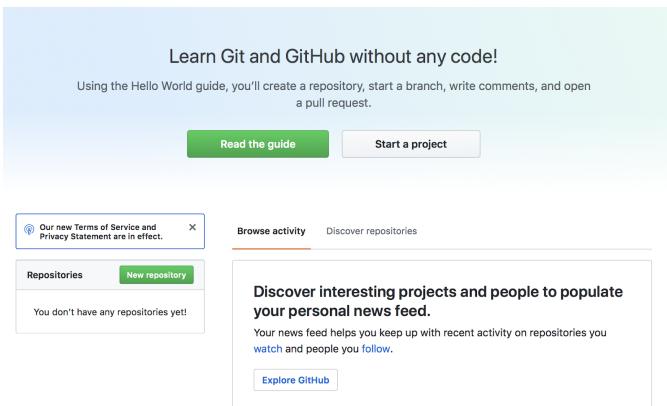
What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source

**Submit**

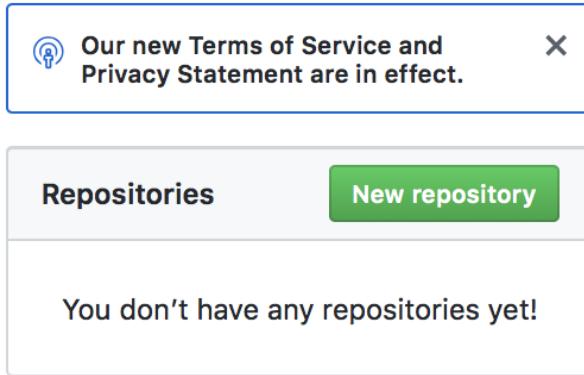
[skip this step](#)

**Gambar 1.4** Step pada pembuatan Github



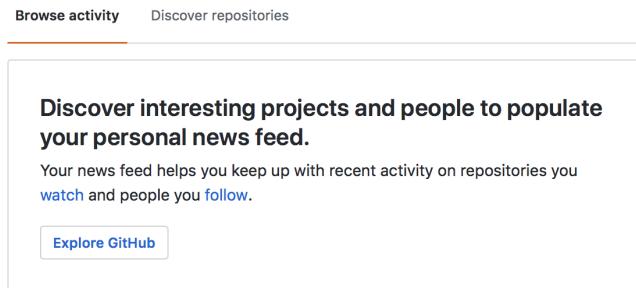
**Gambar 1.5** Akun Github Berhasil dibuat

11. Setelah berhasil membuat akun github, kita sudah siap untuk membuat repository baru didalamnya
12. Gambar 1.6 merupakan menu yang digunakan untuk menampilkan repository yang kita buat atau repository yang mendatarkan kita sebagai kolabolator



**Gambar 1.6** Menampilkan Repository

13. Selanjutnya pada gambar 1.7 merupakan menu yang digunakan untuk mencatat segala bentuk aktivitas kita di dalam repository
14. Melalui menu ini juga kita dapat mencari repository yang dibuka secara public melalui tab **Discover Repositories**



**Gambar 1.7** Discover Repositories

15. Selanjutnya untuk membuat project baru (repository) klik tombol start a project seperti pada gambar 1.8

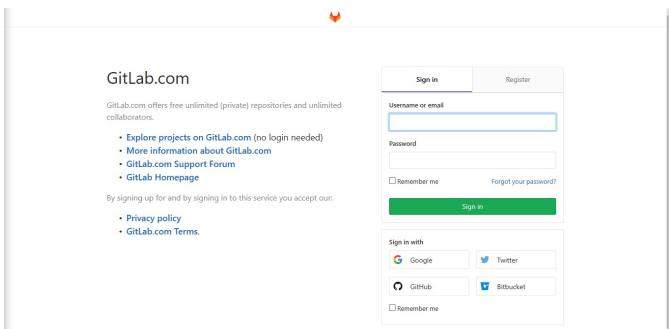


**Gambar 1.8** Start a Project

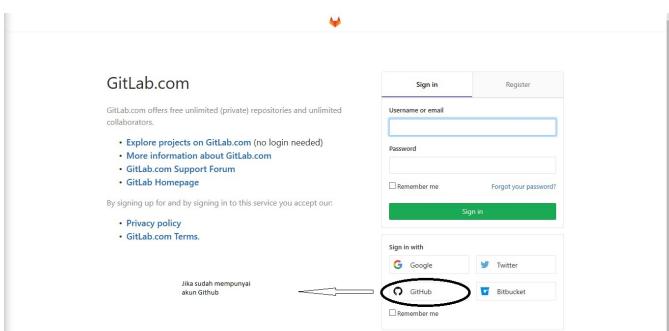
## 1.2 Membuat Akun Gitlab

Gitlab ini sama halnya dengan Github dimana kalangan developer biasanya menggunakan Gitlab sebagai tempat penyimpanan project-project open source sehingga dapat dikembangkan oleh programmer lain baik sebagai team maupun individual. Namun, terdapat perbedaan antara Github dan Gitlab sebagai berikut :

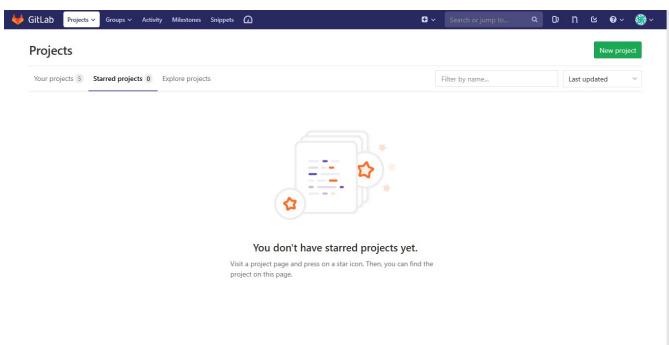
- Menawarkan public project dan private project
  - fitur public dan private pada Gitlab dapat kita akses secara gratis, beda halnya dengan Github yang hanya menyediakan public saja yang secara gratis.
- Snippet support
  - pada Gitlab terdapat fitur ini yaitu memungkinkan pengguna dapat berbagi potongan kecil source code tanpa membagikan keseluruhan code.
- Progress status
  - Pada Gitlab para developer dapat memberikan label dalam project yang sedang dikerjakan dengan label *Work in Progress* sehingga dapat memberikan kejelasan atas project yang sedang dibuat atau dikembangkan. Setelah mengetahui sedikit tentang perbedaan antara Github dan Gitlab, kita akan mencoba membuat akun Gitlab. Sebelum kita membuat ternyata Gitlab terintegrasi dengan Github :
    1. Pertama, buka browser dan masukkan alamat Gitlab dengan url : <https://gitlab.com> seperti pada gambar 1.9
    2. Karena Gitlab ini terintegrasi dengan Github, maka jika sudah mempunyai akun Github kita tinggal sign in menggunakan akun Github seperti pada gambar 1.10
    3. Setelah itu, kita authorized akun Github ke dalam akun Gitlab
    4. Setelah berhasil maka kita diarahkan ke halaman seperti pada gambar 1.11
    5. Selanjutnya kita akan configurasi SSH key pada Gitlab 1.12
    6. kita ambil SSH key yang sudah kita buat pada akun Github, kita buka Git Bash ketikan seperti gamabar 1.13



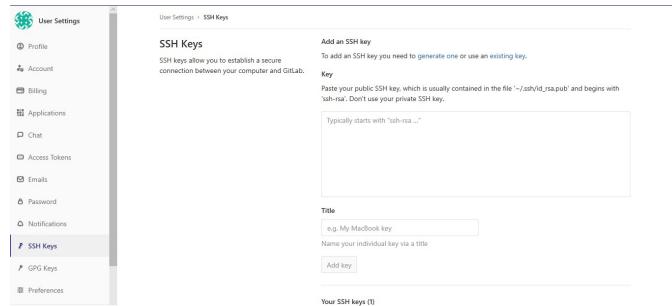
**Gambar 1.9 URL Gitlab**



**Gambar 1.10 Sign in with Github**



**Gambar 1.11 Halaman Gitlab**



**Gambar 1.12**    SSH key

**cd .ssh**, kita masuk ke direktori .ssh

**ls**, untuk melihat list file yang terdapat pada direktori .ssh

**cat id\_rsa.pub**, lalu kita panggil ssh key dalam file id\_rsa.pub

```

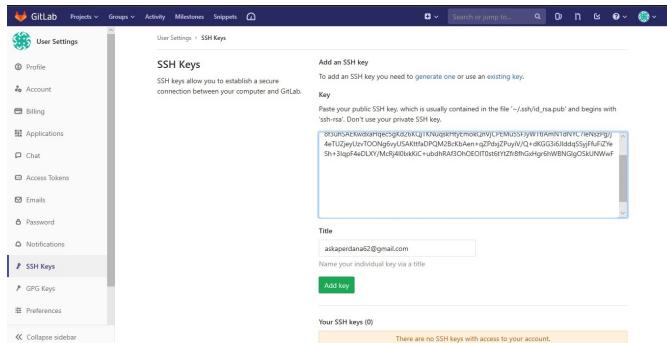
MINGW64:/c/Users/askahar/.ssh
askahar@DESKTOP-0TPK2GF MINGW64 ~
$ cd .ssh
askahar@DESKTOP-0TPK2GF MINGW64 ~/.ssh
$ ls
id_rsa  id_rsa.pub  known_hosts

askahar@DESKTOP-0TPK2GF MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAQABAAQACxPeC4c9ct+jQHIXKkj1X+q4Q3j7uufCubO7vcYqmIN
uh8f3uhSAEKwdxahQec5gKd26KQjTKNuqskskItyEmokNvJcPEMu5SFJyWTt1AmN1dNYC7leNsPjg/j4e
TUZjeyUzvTOONq6vyUSAkttfadPQM2BcKbAen+aZPdxizPuyiV/Q+dKGG316J1ddqSSy1FfuFizYeSh+

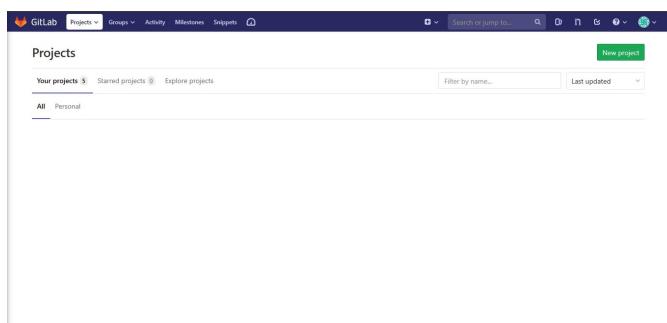
```

**Gambar 1.13**    Ambil SSH key menggunakan Git Bash

7. Setelah kita mendapatkan SSH key kita, lalu kita copy paste ke dalam SSH key Gitlab spserti pada gambar 1.14
8. Selanjutnya jika sudah memasukkan SSH key kita dapat memulai menggunakan Gitlab kita, untuk membuat project baru klik tombol new project seperti pada gambar 1.15



**Gambar 1.14** Copy paste SSH key Bash ke dalam SSH key Gitlab



**Gambar 1.15** Membuat Project baru



## BAB 2

---

# PRAKTEK CEPAT

---

### 2.1 Latihan

Bisa karena biasa, rumit karena tak dicil. Maka pemakaian git adalah masalah kebiasaan. Oleh karena itu, dalam pembukaan awal buku ini justru kita tidak dulu masuk kepada teori git tapi langsung praktik. setelah berkali-kali praktik maka akan timbul beberapa pertanyaan mengenai fungsi dari perintah-perintah git yang bisa dibaca pada bagian selanjutnya.

Skenarionya adalah kita akan melakukan kontribusi terhadap sebuah repo *Open Source* di GitHub. Jadi latihan ini adalah latihan bagaimana ikut berkontribusi kepada repo *Open Source* yang ada di Github yang selama ini digunakan oleh para relawan kode untuk bersama membangun kode program berbasis *Open Source*.

#### 2.1.1 Konfigurasi Key

Pertama kita masuk kepada tahapan *setting* atau konfigurasi *key* untuk mengakses semua repo dari *profile* kita dari *git bash*:

1. Buat akun di [www.github.com](http://www.github.com) terlebih dahulu

2. Install **Git Bash** dari alamat [git-scm.com/downloads](http://git-scm.com/downloads) di komputer Anda, kemudian buka *git bash*.
3. Pastikan sudah ada di *home directory* dengan mengetikkan perintah `cd` kemudian *enter*. Untuk mengetahui posisi Anda ada di direktori mana ketikkan perintah `pwd` dan *enter*.
4. **Generate Key** dengan perintah listing 2.1.

```
| ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

**Listing 2.1** Perintah Membuat Key

Hasilnya seperti yang terlihat pada gambar 2.1.

```
MINGW64:/c/Users/syara
$ ssh-keygen -t rsa -b 4096 -C "syarahfauziatululya@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/syara/.ssh/id_rsa):
Created directory '/c/Users/syara/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/syara/.ssh/id_rsa.
Your public key has been saved in /c/Users/syara/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:S158+Z5+UNqHu/SpXCoand0Gc/xst4Mxx10iqwJPww0 syarahfauziatululya@gmail.com
The key's randomart image is:
+---[RSA 4096]----+
| |
| |
| E . o o
| . S o = *
| . * + + ^ ..
| + + . @ / .o
| o o o .B.X+
| .o o o**B|
+---[SHA256]-----+
```

**Gambar 2.1** Perintah keygen cukup enter enter saja

5. Baca *key* yang sudah di *generate* dengan menggunakan perintah 2.2.

```
| cat .ssh/id_rsa.pub
```

**Listing 2.2** Perintah Membaca Public Key

Hasilnya seperti pada gambar 2.2.

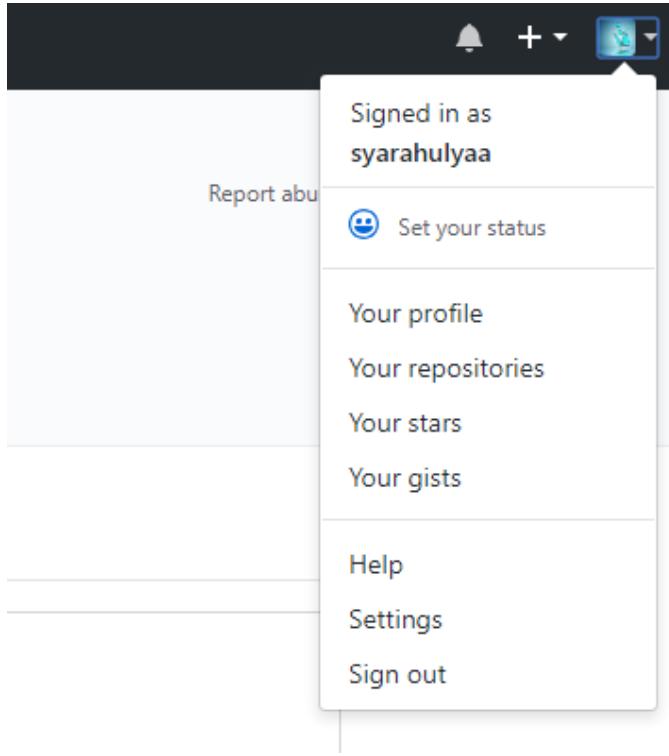
6. Hasil luaran yang dibaca sebelumnya merupakan *key* kita. Kemudian masukkan *key* dengan masuk ke menu **Setting** yang ada dipojok kanan atas, seperti pada gambar 2.3. Lalu pada menu **SSH and GPG Keys** tambahkan **New SSH key** seperti pada gambar 2.4.

### 2.1.2 Fork Repозитори

Pertama kita cari repositori utama yang akan kita jadikan tempat berkontribusi dalam repositori tersebut. Jika sudah ketemu, kemudian klik Fork(Tombol kanan atas)seperti

```
MINGW64:/c/Users/syara
syarahulyaa-PC MINGW64 - 
$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3zaC1yc2EAAQABAAQACQC9j0vcYQLPKot0u2zY7kcoTLS69dD8zILH6Vupnt9U
EXViXUpaxvkJedVwh0bt6P07/f1De2A4Zqp+e94annvJtG0dtej152aeFqjcyWuAzqzf10aEUL720
AZjLM/HqWLf2P5ewI83YYJ2i5XmdIECAYUR17Ro/K1AFmgk/yr1CX3eJK24gnqlBbe7r6mUp1GL6k
McRhRSG19Rk1Y3syKN9T4uC3kN4LUhGyGNp6QC5Q9mxKxe28oi4FpQobNj23Hn1WRlwZzzxx2GENM3
8y1Atg9K2aYa3w9AuPo1b5Qfbey7efx4ADkz1104+/Cc3zOKYs1w5IMeDi ckJsmZ7Tbsq4nWts8fmp/
TbgVK+uJYSRuSh+jtD5fugugyaq#mBNemD5jg3jnqv1S5poGvnQHdvRYdaPGC48ZKTz0k3WqXpw1fcTG
5Nv11NuAEBSewG32zaU3sYZMFYBLQN93f1/JoI4rYbkosheZ1fAhnRwDTT156oLH98mtnzYFTx5uQ
LG+Xgg4YPubusVEAjv1mnHCBshi9CdjopLfjth7Qg91hgC/WbcgchypqGo8ALJ64DRgwz4qon+4IQ
370kgUDhu9wgBqtISRoKt3teou3w1dGeKdeHo6D+00ke6+z5uhB6odFtr9dnx3GTZ7GjuCL2BnrSUfa
Qw== syarahfauziatululyaa@gmail.com
```

**Gambar 2.2** Perintah cat untuk membaca key



**Gambar 2.3** Setting

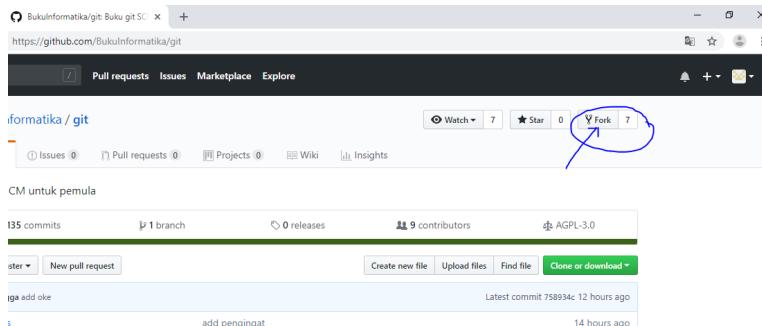
Personal settings	SSH keys	New SSH key
Profile		
Account		
Emails		
Notifications		
Billing		
SSH and GPG keys		

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

Latihan	89:a2:d3:0c:b3:13:66:79:d8:4f:8d:75:90:32:5a:6a
SSH	Added on Feb 1, 2019
	Last used within the last week — Read/write
<a href="#">Delete</a>	

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

**Gambar 2.4** New SSH key

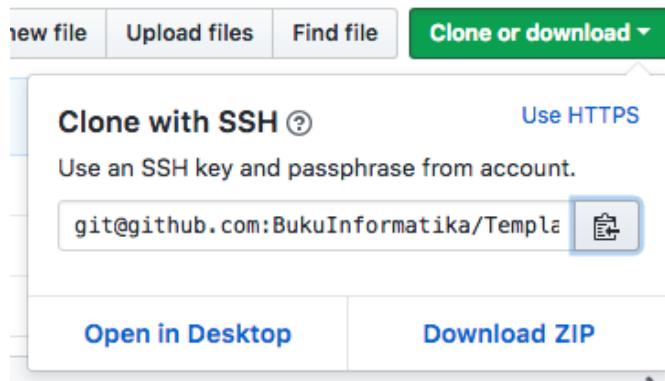


Gambar 2.5 Letak Tombol Fork

pada gambar 2.5 yang dilanjutkan dengan memilih akun kita sebagai tujuan clone fork tersebut. Setelah selesai maka kita akan memiliki repo yang sama dengan repo yang anda fork. Contoh apabila anda melakukan Fork dari <https://github.com/RepoAsal/Testing> maka anda akan memiliki repo <https://github.com/usernameAnda/Testing>, dan kita akan bekerja pada repo hasil fork ini.

### 2.1.3 Navigasi direktori dengan Git Bash

Pertama kita tentukan terlebih dahulu folder tempat kita mengerjakan repo hasil fork kita. Misal di Drive D: folder Ganteng. Maka buka git bash kita dan arahkan menuju folder tersebut dengan perintah `cd /D/Ganteng`. Buka web repo fork kita di github klik tombol hijau **Clone or Download** pilih Clone with SSH(Gambar 2.6) lalu salin kode yang tampak seperti `git@github.com:usernameAnda/Testing.git`.



Gambar 2.6 Hasil Perintah git clone

Pada Git Bash ketik `git clone git@github.com:usernameAnda/Testing.git` hasilnya seperti terlihat pada gambar 2.7.

```

MINGW64:/a/SIG
$ git clone git@github.com:syarahulyaa/SIG.git
Cloning into 'SIG'...
The authenticity of host 'github.com (52.74.223.119)' can't be established.
RSA key fingerprint is SHA256:nThbgkXUpJM17ELIGOspRomTxdCARLviKw6E5Y8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,52.74.223.119' (RSA) to the list of known
hosts.
remote: Enumerating objects: 3611, done.
Receiving objects: 29% (1079/3611), 5.30 MiB | 1.18 MiB/s    s

```

**Gambar 2.7** Hasil Perintah git clone

Setelah selesai, ketik perintah *ls* maka akan muncul direktori baru yaitu folder *Testing* atau sesuai dengan nama repo Anda. masuk ke direktori tersebut dengan perintah *cd Testing*. Kemudian ketik *git status* akan muncul status dari repo git kita, ringkasan perintahnya bisa dilihat pada perintah listing 2.3.sebagai contoh lihat hasilnya pada gambar 2.8.

```

1 cd /D/Ganteng
2 git clone git@github.com:usernameAnda/Testing.git
3 ls
4 cd Testing
5 git status

```

**Listing 2.3** Navigasi direktori menuju repositori

```

$ ls
git/
syara@Ulyaa-PC MINGW64 /A/Git
$ cd git

syara@Ulyaa-PC MINGW64 /A/Git/git (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

syara@Ulyaa-PC MINGW64 /A/Git/git (master)

```

**Gambar 2.8** Hasil Perintah ls, cd dan git status

#### 2.1.4 Sinkronisasi dengan Repo Utama

Karena ini repo hasil Fork maka kita harus selalu di sinkronisasi dari repo aslinya(tidak otomatis tersingkron). Sehingga perlu melakukan setting sumber repo tujuan yang

merupakan repo asal. Pertama pastikan git bash sudah pada direktori repositori. Untuk melakukan sinkronisasi dengan repo asal kita harus mengeset satu kali dengan perintah listing 2.4.

```
1 git remote add upstream https://github.com/RepoAsal/Testing.git
```

#### **Listing 2.4 Set Repo Asal Sebagai Upstream**

Setelah melakukan *setting* sekali di awal, kemudian selanjutnya kita tinggal melakukan sinkronisasi terus menerus sebelum melakukan perubahan dengan perintah listing 2.5.

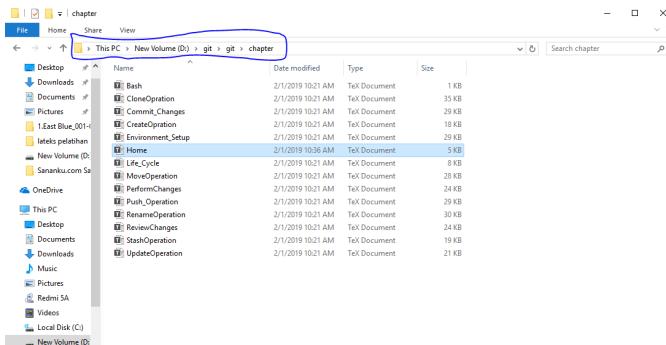
```
1 git pull origin master
2 git fetch upstream
3 git pull upstream master
4 git push origin master
```

#### **Listing 2.5 Perintah Sinkronisasi dengan repo asal**

### **2.1.5 Bekerja dengan Git Bash**

Sekarang kita mulai bekerja pada repo kita. Melakukan penambahan atau perubahan pada *file*. Kemudian perubahan tersebut diminta untuk dimasukkan di repo utama tempat kita *fork* repo kita. Urutan pekerjaan yang kita ulang terus menerus adalah sebagai berikut :

1. Sebelum mulai mengerjakan, sinkronisasi kembali dengan repo asli lagi agar terhindar dari konflik dengan perintah di listing 2.5.
2. Silahkan edit satu file yang akan di ubah atau ditambah lalu di simpan dan di tutup yang berada di direktori yang sudah disetting di navigasi direktori. Seperti terlihat pada gambar 2.9.



**Gambar 2.9** Direktori kerja git dari Explorer

3. Cek status git dengan perintah *git status*, jika terdapat warna merah berarti belum kita add, jika terdapat warna hijau berarti belum kita commit.

4. File yang barusan diubah wajib kita daftarkan pada daftar perubahan yang kita lakukan yaitu dengan perintah *git add namafilenya.tex*
5. Cek status git dengan perintah *git status*, jika terdapat warna merah berarti belum kita add, jika terdapat warna hijau berarti belum kita commit.
6. Setelah file diubah maka kita wajib menambahkan komentar terhadap file yang kita ubah tersebut agar mempermudah kontributor yang lain mengetahui apa saja yang kita perbuat terhadap file yang sudah kita add. Perintah untuk memberi komentar dari file yang sudah di *git add* adalah :*git commit -m 'perubahan apa yang telah kita lakukan di ceritakan di sini secara lengkap'*
7. Akan muncul permintaan konfigurasi global seperti gambar 2.10. Maka kita harus memasukkan konfigurasi global dengan perintah 2.6. Setelah itu kita ulang kembali perintah *git commit*.

```

1 git config --global user.email "awangga@gmail.com"
2 git config --global user.name "awangga"
3 git commit -m "perubahan apa yang telah kita lakukan di ceritakan
di sini secara lengkap"
```

**Listing 2.6** Perintah Sinkronisasi dengan repo asal

```

MINGW64:/A/Git/git
$ git add chapter/Home.tex
warning: LF will be replaced by CRLF in chapter/Home.tex.
The file will have its original line endings in your working directory
syara@Ulyaa-PC MINGW64 /A/Git/git (master)
$ git commit -m 'menambahkan git pull upstream master'
*** Please tell me who you are.
Run
  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"
to set your account's default identity.
Omit --global to set the identity only in this repository.
fatal: unable to auto-detect email address (got 'syara@Ulyaa-PC.(none)')
syara@Ulyaa-PC MINGW64 /A/Git/git (master)
$ git config --global user.email "syarahfauziatuluulya@gmail.com"
syara@Ulyaa-PC MINGW64 /A/Git/git (master)
$
```

**Gambar 2.10** Permintaan Setting Global

Setting global ini hanya sekali saja ketika baru melakukan instalasi git bash, selanjutnya tidak akan muncul kembali permintaan setting global ini.

8. Cek status git dengan perintah *git status*, jika terdapat warna merah berarti belum kita add, jika terdapat warna hijau berarti belum kita commit. Lihat gambar 2.13.
9. Kemudian file yang sudah kita ubah kita upload ke website [github.com](https://github.com) dengan perintah *git push origin master*.

10. Sinkronisasi kembali dengan repo asli lagi sebelum beberapa detik melakukan **New pull request** agar terhindar dari konflik dengan perintah pada listing 2.5. Apabila terdapat konflik jangan panik, itu namanya merge. Yang artinya menyatukan pekerjaan di web repo dengan repo lokal komputer kita. Apabila keluar tiba-tiba text editor dalam git-bash sehingga kita tidak bisa memasukkan perintah(tidak ada tanda dolarnya). Maka simpan saja dengan menekan tombol *esc* kemudian ketik :*wq* yang berfungsi untuk menyimpan dan tekan tombol *enter*.

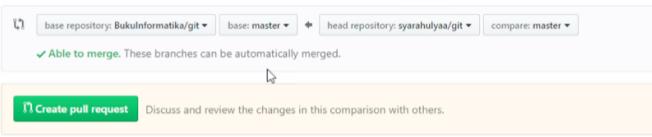
11. Buka web repo kita di website [www.github.com](https://github.com) Contoh disini

<https://github.com/usernameAnda/Testing>

kemudian klik **New pull request**. Pastikan yang sebelah kiri atau base kita set kepada repo utama tempat fork kita yaitu RepoAsal/Testing dan compare pada sebelah kanan adalah repo kita, disini dicontohkan usernameAnda/Testing dan ilustrasi bisa dilihat di gambar 2.11. Klik tombol hijau *Create Pull Request* kemudian teruskan sampai ada kembali tombol hijau yang kita klik lagi.

### Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



**Gambar 2.11** Setelah klik New pull request

12. Beritahukan admin repo utama untuk accept Pull Request Anda. Jika sudah di accept lakukan lagi langkah dari awal.

#### 2.1.6 Mengatasi *Error*

Seorang programmer atau anak if pantang menyebutkan bahwa programnya error dan menyerah begitu saja. Error merupakan sebuah anugerah yang harus kita syukuri. Beruntungnya di git semua error memiliki petunjuk yang jelas. Sehingga apabila kita mendapati error, pastikan membaca dengan baik error nya dan selesaikan errornya dengan tenang dan santai. Karena error git sangat sederhana dan mudah sekali untuk diatasi. Atasi error tersebut dan jangan lari dari error tersebut.

**2.1.6.1 Kesalahan direktori** Apabila bertemu error seperti pada listing 2.7. Pastikan git bash kita berada pada direktori repositori git yang dikerjakan, biasanya ditandai ada nama *branch* di git bash nya.

```
| fatal: not a git repository (or any of the parent directories): .git
```

**Listing 2.7** Kesalahan karena bukan pada direktori repositori

Sebagai contoh pada gambar 2.10 terlihat di ujung sebelum perintah dimasukkan ada tulisan *master* warna biru muda. Itu artinya kita berada pada direktori repositori dengan *branch master*. Di sebelah kiri *master* ada tulisan kuning */A/Git/git*, yang artinya kita berada pada drive A: windows. Pada drive A: tersebut ada folder Git(G besar) yang didalamnya ada repo clone dengan folder git(g kecil). Posisi kita ada di dalam folder git tersebut. Sebagai contoh kedua pada gambar 2.9, maka tulisan pada git bash tempat kita bekerja menjadi */D/git/git* atau */D/git/git/chapter*.

### 2.1.6.2 Gagal Fetch Upstream pada Github

Apabila anda pada saat melakukan *git fetch upstream* atau *git pull upstream master* keluar error seperti pada listing 2.8.

```
1 fatal: 'upstream' does not appear to be a git repository
2 fatal: Could not read from remote repository.
3
4 Please make sure you have the correct access rights
5 and the repository exists.
```

**Listing 2.8** Gagal melakukan fetch upstream

Maka ada dua kemungkinan, kemungkinan pertama anda belum melakukan set upstream seperti pada perintah di listing 2.4. Silahkan lakukan dahulu perintah di listing 2.4. Salah satu contoh repo yang belum melakukan perintah dari listing 2.4 bisa dilihat di gambar 2.12.

```
[awangga:MeanShift_py awangga$ git fetch upstream
fatal: 'upstream' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
awangga:MeanShift_py awangga$ git pull upstream master
fatal: 'upstream' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
awangga:MeanShift_py awangga$ git config -l
credential.helper=osxkeychain
user.name=Rolly Maulana Awangga
user.email=rolly@awang.ga
core.repositoryformatversion=0
core.filenode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precacheunicode=true
remote.origin.url=git@github.com:mattnedrich/MeanShift_py.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
```

**Gambar 2.12** Gagal Fetch Upstream

Apabila anda sudah melakukan set upstream maka untuk melakukannya pengecekan kita lihat dengan perintah di listing 2.9.

```
1 git config -l
```

**Listing 2.9** Melihat konfigurasi git di repo komputer kita

Jika sudah ter set maka akan ada bagian yang berisi *remote.upstream.url* seperti pada listing 2.10.

```
1 remote . upstream . url=https://github.com/RepoAsal/Testing.git
2 remote . upstream . fetch=+refs/heads/*:refs/remotes/upstream/*
```

**Listing 2.10** Hasil perintah *git config -l*

Pastikan upstream berbentuk *https* bukan *ssh*. Apabila anda salah melakukan setting upstream anda bisa menghapusnya dengan perintah yang ada di listing 2.11. Baru setelah itu lakukan lagi set upstream dengan perintah di listin 2.4.

```
1 git remote remove upstream
```

**Listing 2.11** Perintah menghapus upstream yang salah

**2.1.6.3 Gagal Fetch Upstream pada Gitlab** Apabila pada saat melakukan **git fetch upstream** terjadi kesalahan seperti berikut 2.12

```
1 remote: HTTP Basic: Access denied
2 fatal: Authentication failed for 'https://gitlab.com/RepoAsal/Testing
.git'
```

**Listing 2.12** Access Denied

Maka cara yang dilakukan yaitu ganti clone https di repositori utama menjadi clone ssh repositori utama sehingga pada saat remote repositori utama 2.13

```
1 git remote add upstream git@gitlab.com:RepoAsal/Testing.git'
```

**Listing 2.13** perintah remote repo utama

**2.1.6.4 Salah Clone** Buat pemula, kesalahan ini kadang terjadi jika tidak hati-hati membaca langkah-langkah sebelumnya. Apabila bertemu pesan kesalahan seperti pada listing 2.14. Dari listing tersebut terliat *jonluca/Anubis.git* adalah repo utama bukan repo clone kita perhatikan pada baris pertama listing to *awangga* yang merupakan user github kita. Sehingga ini adalah kesalahan dalam clone, seharusnya repo yang di clone adalah *awangga/Anubis.git*. Ulang kembali kepada langkah clone repo, jika clone repo belum ada maka ulangi langkah fork.

```
1 ERROR: Permission to jonluca/Anubis.git denied to awangga.
2 fatal: Could not read from remote repository.
3
4 Please make sure you have the correct access rights
5 and the repository exists.
```

**Listing 2.14** Peringatan *Permission denied*

**2.1.6.5 Lupa langkah** Sering-sering menggunakan git status untuk mengetahui sejauh mana kita melakukan perubahan. git status akan memberitahukan kita langkah apa saja yang harus kita lakukan, jika kita lupa langkah-langkah pengerjaan diatas. Pada gambar 2.13, jika terdapat warna merah berarti belum kita add, jika terdapat warna hijau berarti belum kita commit dan terakhir kita juga diberikan petunjuk untuk segera *git push*.

```
[awangga:git awangga$ git status
On branch master
Your branch is up to date with 'origin/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:   chapter/Home.tex
    modified:   git.pdf

no changes added to commit (use "git add" and/or "git commit -a")
[awangga:git awangga$ git add chapter/Home.tex git.pdf
[awangga:git awangga$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   chapter/Home.tex
    modified:   git.pdf

[awangga:git awangga$ git commit -m "Menambahkan tata cara melakukan pull request di web github"
[master 76e0063] Menambahkan tata cara melakukan pull request di web github
 2 files changed, 8 insertions(+), 5 deletions(-)
[awangga:git awangga$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[awangga:git awangga$ git push origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 429.57 Kib | 1.90 MiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To github.com:BukuInformatika/git.git
  313f9a4..76e0063  master -> master
awangga:git awangga$ ]
```

Gambar 2.13 Berbagai macam luaran git status

**2.1.6.6 Konflik** Ketiga, sebelum melakukan pekerjaan dan sebelum melakukan **New pull request**(dua kali). Pastikan repo lokal di komputer kita sinkron dengan yang ada di github baik itu repo asli maupun repo hasil fork milik kita dengan perintah pada listing 2.5. Apabila terdapat konflik jangan panik, itu namanya merge. Yang artinya menyatukan pekerjaan di web repo dengan repo lokal komputer kita. Apabila keluar tiba-tiba text editor dalam git-bash sehingga kita tidak bisa memasukkan perintah seperti pada gambar 2.14. Maka simpan saja dengan menekan tombol *esc* kemudian ketik :*wq* yang berfungsi untuk menyimpan dan tekan tombol *enter*.

**Gambar 2.14** Permintaan Merge dengan pesan standar yang keluar

# BAB 3

---

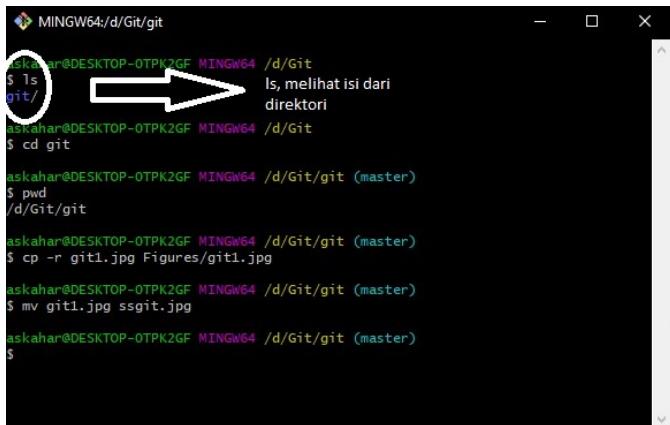
## PERINTAH DASAR BASH

---

### 3.1 Perintah Dasar Bash

#### 3.1.1 fungsi perintah pada Bash

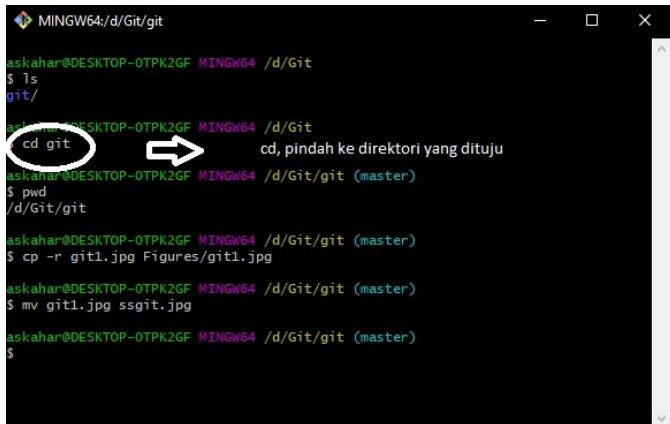
1. ls fungsi perintah **ls** pada bash ini yaitu untuk melihat isi dari suatu direktori. Contoh lihat pada gambar 3.1
2. cd fungsi perintah **cd** (*change directory*) pada bash ini yaitu untuk berpindah ke direktori yang dituju. Contoh lihat pada gambar 3.2
3. pwd fungsi perintah **pwd** pada bash ini yaitu untuk mengetahui path direktori yang sedang aktif. Contoh lihat pada gambar 3.3
4. mv fungsi perintah **mv** pada bash ini yaitu untuk mengubah nama file. Contoh lihat pada gambar 3.4
5. cp fungsi perintah **cp** pada bash ini yaitu untuk mencopy file. Contoh lihat pada gambar 3.5



A screenshot of a terminal window titled "MINGW64/d/Git/git". The terminal shows a user named "askahar" at a "DESKTOP-OTPK2GF" host. The user runs the command "ls git/" which lists the contents of the "git" directory. A large white arrow points from the left towards the "ls" command.

```
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git
$ ls git/
ls, melihat isi dari
direktori
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$
```

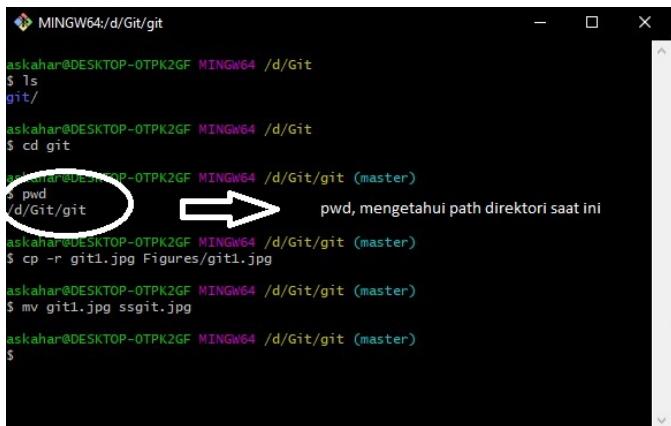
Gambar 3.1 Perintah dasar ls pada git bash



A screenshot of a terminal window titled "MINGW64/d/Git/git". The terminal shows a user named "askahar" at a "DESKTOP-OTPK2GF" host. The user runs the command "cd git" which changes the current working directory to "git". A large white arrow points from the left towards the "cd" command.

```
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ ls git/
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ cd git
cd, pindah ke direktori yang dituju
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$
```

Gambar 3.2 Perintah dasar cd pada git bash



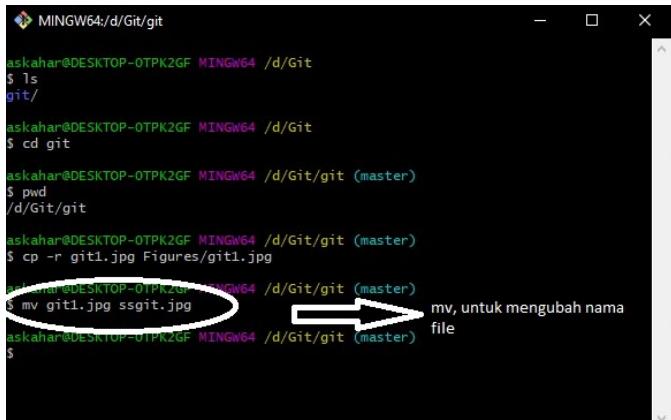
A screenshot of a Git Bash terminal window titled 'MINGW64/d/Git/git'. The terminal shows a user's session with several commands:

```
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
```

An annotation consists of a white circle on the left and a black arrow pointing to the right, with the text 'pwd, mengetahui path direktori saat ini' (pwd, to find out the current directory path) written next to it.

```
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$
```

Gambar 3.3 Perintah dasar pwd pada git bash



A screenshot of a Git Bash terminal window titled 'MINGW64/d/Git/git'. The terminal shows a user's session with several commands:

```
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
```

An annotation consists of a white circle on the left and a black arrow pointing to the right, with the text 'mv, untuk mengubah nama file' (mv, to change file name) written next to it.

```
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
```

Gambar 3.4 Perintah dasar mv pada git bash

```
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-OTPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git

askahar@DESKTOP-OTPK2GF MINGW64 ./d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-OTPK2GF MINGW64 ./d/Git/git (master)
$ mv git1.jpg ssgit1.jpg

askahar@DESKTOP-OTPK2GF MINGW64 ./d/Git/git (master)
$
```

cp, untuk mencopy file

**Gambar 3.5** Perintah dasar cp pada git bash

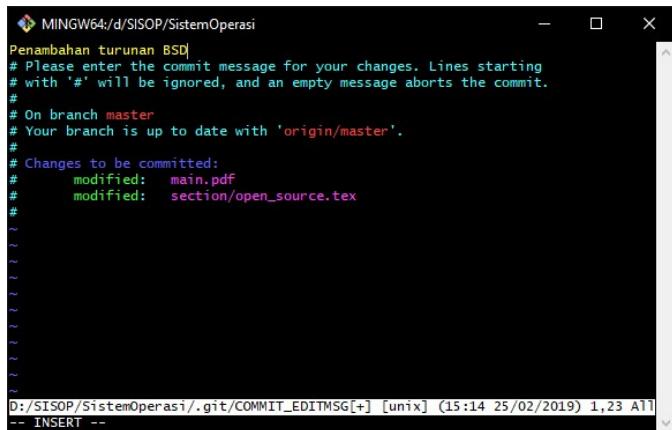
6. :wq fungsi perintah :wq pada bash ini yaitu untuk keluar dari Vim editor ketika kita melakukan perubahan file menggunakan Vim editor. Contoh ketika melakukan commit menggunakan VIM editor 3.6

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up to date with 'origin/master'.
#
# Changes to be committed:
#       modified:   main.pdf
#       modified:   section/open_source.tex
#
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
D:/SISOP/SistemOperasi/.git/COMMIT_EDITMSG [unix] (15:14 25/02/2019) 1,0-1 All
```

**Gambar 3.6** melakukan commit menggunakan VIM editor

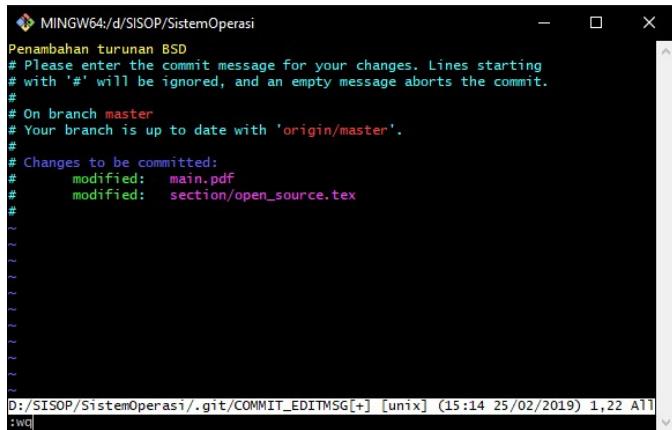
selanjutnya setelah mengetikan commit maka akan masuk ke VIM editor tekan i pada keyboard sehingga kita dapat mengisi atau *insert*, lalu isi apa yang kita commit 3.7

jika, kita telah melakukan commit maka untuk keluar dari halaman VIM editor ini ketikan :wq untuk keluar dari halaman VIM editor 3.8



A screenshot of a terminal window titled "MINGW64:/d/SISOP/SistemOperasi". The window displays a Git commit message editor. The message starts with "Penambahan turunan BSD" and includes instructions for entering a commit message. It lists changes made to files "main.pdf" and "section/open\_source.tex". The status bar at the bottom shows the path "D:/SISOP/SistemOperasi/.git/COMMIT\_EDITMSG[+]" and the date/time "15:14 25/02/2019". The text "1,23 All" is also visible. A cursor is positioned at the end of the message, and the status bar shows "-- INSERT --".

Gambar 3.7 mengisi apa yang kita commit



A screenshot of a terminal window titled "MINGW64:/d/SISOP/SistemOperasi". The window displays a Git commit message editor. The message is identical to the one in Gambar 3.7. The status bar at the bottom shows the path "D:/SISOP/SistemOperasi/.git/COMMIT\_EDITMSG[+]" and the date/time "15:14 25/02/2019". The text "1,22 All" is visible. A cursor is positioned at the end of the message, and the status bar shows ":wq".

Gambar 3.8 keluar dari halaman VIM editor

### 3.1.2 penggunaan VI editor

- menambahkan file menggunakan VI editor di bash

ketikan vi namafile di bash untuk menambahkan file 3.9

```

MINGW64:/d/SISOP/SistemOperasi/section
$ vi Linux
askahar@DESKTOP-0TPK2GF MINGW64 /d/SISOP/SistemOperasi/section (master)
$ ls
1.tex          OShaundrate.tex
deadlock.tex   OsSemaphore.tex
definisi_awal.tex OSTkinter.tex
FirmataKomunikasiArduino.tex OurGrouper.tex
fragmentation.tex pengertian.tex
InsertDBMySQL.tex PostgreTutorial.tex
InsertDBmysql.tex processaddressspace.tex
KeliCaraInstalpip.tex pullrequestdeadlock.tex
KeliFirmataKomArduino.tex PySerial.tex
KeliProsesOS.tex SerialCommLinux.tex
KeliSemaphore.tex SERIALCOMMWINDOW5.tex
kelzmatplotliblib.tex SISTEMOPERASI.log
'kelebihan_penggunaan usb.txt' SISTEMOPERASI.tex
kernel.tex     SQLite.tex
memory_allocation.tex starvation.tex
mongodB.bib    usb_to_serial.tex
MongoDB.tex
open_source.tex vpython.tex

askahar@DESKTOP-0TPK2GF MINGW64 /d/SISOP/SistemOperasi/section (master)
$ vi Linux.tex

```

**Gambar 3.9** membuat file

lalu akan masuk ke halaman, tekan **i** untuk menambahkan isi file tersebut  
3.10

```

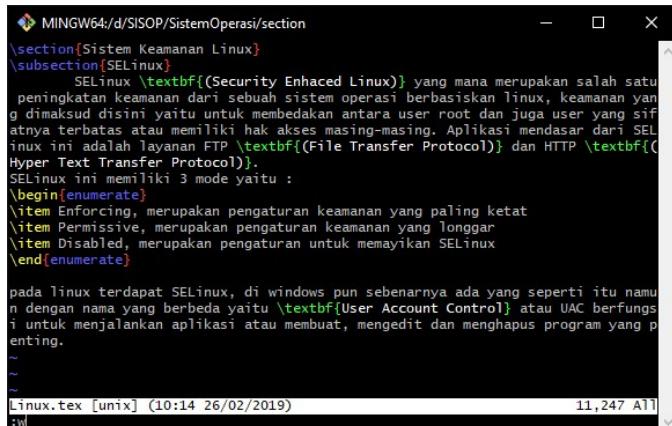
MINGW64:/d/SISOP/SistemOperasi/section
\section{Sistem Keamanan Linux}
\subsection{SELinux}
    SELinux \textbf{(Security Enhanced Linux)} yang mana merupakan salah satu peningkatan keamanan dari sebuah sistem operasi berbasiskan linux, keamanan yang dimaksud disini yaitu untuk membedakan antara user root dan juga user yang siapnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SELinux ini adalah layanan FTP \textbf{(File Transfer Protocol)} dan HTTP \textbf{(Hyper Text Transfer Protocol)}.
    SELinux ini memiliki 3 mode yaitu :
\begin{enumerate}
\item Enforcing, merupakan pengaturan keamanan yang paling ketat
\item Permissive, merupakan pengaturan keamanan yang longgar
\item Disabled, merupakan pengaturan untuk memayikan SELinux
\end{enumerate}
pada linux terdapat SELinux, di windows pun sebenarnya ada yang seperti itu namun dengan nama yang berbeda yaitu \textbf{(User Account Control)} atau UAC berfungsi untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang penting.
-
-
-
Linux.tex[+] [unix] (06:59 01/01/1970)           11,248 A11
-- INSERT --

```

**Gambar 3.10** menambahkan data file

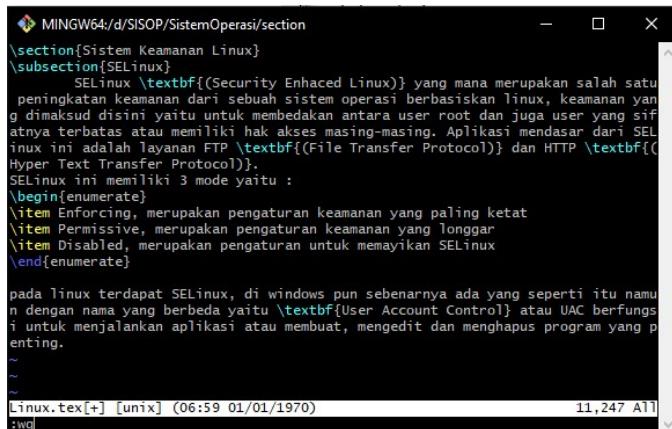
jika sudah mengisi data di dalam file untuk menyimpan file tersebut ketikan **esc** pada keyboard lalu **:w** untuk menyimpan file tanpa keluar dari halaman vi  
3.11

untuk menyimpan file lalu keluar dari halaman vi maka ketikan **:wq** 3.12



```
MINGW64:/d/SISOP/SistemOperasi/section
\section{Sistem Keamanan Linux}
\subsection{SELinux}
    SELinux \textbf{(Security Enhanced Linux)} yang mana merupakan salah satu peningkatan keamanan dari sebuah sistem operasi berbasiskan linux, keamanan yang dimaksud disini yaitu untuk membedakan antara user root dan juga user yang sifatnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SELinux ini adalah layanan FTP \textbf{(File Transfer Protocol)} dan HTTP \textbf{(Hyper Text Transfer Protocol)}.
    SELinux ini memiliki 3 mode yaitu :
\begin{enumerate}
\item Enforcing, merupakan pengaturan keamanan yang paling ketat
\item Permissive, merupakan pengaturan keamanan yang longgar
\item Disabled, merupakan pengaturan untuk mematikan SELinux
\end{enumerate}
pada linux terdapat SELinux, di windows pun sebenarnya ada yang seperti itu namun dengan nama yang berbeda yaitu \textbf{User Account Control} atau UAC berfungsi untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang penting.
-
-
-
Linux.tex [unix] (10:14 26/02/2019) 11,247 All
:w
```

Gambar 3.11 menyimpan file



```
MINGW64:/d/SISOP/SistemOperasi/section
\section{Sistem Keamanan Linux}
\subsection{SELinux}
    SELinux \textbf{(Security Enhanced Linux)} yang mana merupakan salah satu peningkatan keamanan dari sebuah sistem operasi berbasiskan linux, keamanan yang dimaksud disini yaitu untuk membedakan antara user root dan juga user yang sifatnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SELinux ini adalah layanan FTP \textbf{(File Transfer Protocol)} dan HTTP \textbf{(Hyper Text Transfer Protocol)}.
    SELinux ini memiliki 3 mode yaitu :
\begin{enumerate}
\item Enforcing, merupakan pengaturan keamanan yang paling ketat
\item Permissive, merupakan pengaturan keamanan yang longgar
\item Disabled, merupakan pengaturan untuk mematikan SELinux
\end{enumerate}
pada linux terdapat SELinux, di windows pun sebenarnya ada yang seperti itu namun dengan nama yang berbeda yaitu \textbf{User Account Control} atau UAC berfungsi untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang penting.
-
-
-
Linux.tex[+] [unix] (06:59 01/01/1970) 11,247 All
:wq
```

Gambar 3.12 menyimpan file lalu keluar

namun jika tidak ingin menyimpan atau mengubah file tersebut (*discard all changes*) maka ketikan :q! 3.13

```
\section{Sistem Keamanan Linux}
\subsection{SELinux}
    SELinux \textbf{(Security Enhanced Linux)} yang mana merupakan salah satu peningkatan keamanan dari sebuah sistem operasi berbasiskan linux, keamanan yang dimaksud disini yaitu untuk membedakan antara user root dan juga user yang siapnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SELinux ini adalah layanan FTP \textbf{(File Transfer Protocol)} dan HTTP \textbf{(Hyper Text Transfer Protocol)}.
    SELinux ini memiliki 3 mode yaitu :
\begin{enumerate}
\item Enforcing, merupakan pengaturan keamanan yang paling ketat
\item Permissive, merupakan pengaturan keamanan yang longgar
\item Disabled, merupakan pengaturan untuk memayikan SELinux
\end{enumerate}

pada linux terdapat SELinux, di windows pun sebenarnya ada yang seperti itu namun dengan nama yang berbeda yaitu \textbf{(User Account Control)} atau UAC berfungsi untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang penting.
~  

~  

~  

Linux.tex [unix] (10:14 26/02/2019) 11,247 A11
:q!
```

**Gambar 3.13** keluar tanpa mengubah sesuatu pada file

### 1. Fungsi dasar yang digunakan pada vi editor

Perintah	Penjelasan
vi filename	Membuat file baru jika file belum dibuat, atau membuka file yang telah dibuat sebelumnya
vi -R filename	Membuka file yang sudah dibuat sebelumnya, namun hanya dapat dibaca tanpa bisa diubah
view filename	Sama seperti perintah sebelumnya hanya dapat membaca tanpa bisa melakukan edit

### 2. Menggerakan kursor di dalam file

Perintah	Penjelasan
k	Menggerakan cursor ke atas satu baris
j	Menggerakan cursor ke bawah satu baris
h	Menggerakan cursor ke kiri satu karakter
l	Menggerakan cursor ke kanan satu karakter

## BAB 4

---

# MENGATASI KONFIK

---

### 4.1 Pada Repo Lokal

Semakin banyak yang bekerja maka semakin sering terjadi konflik. Sebagai contoh apabila dalam satu waktu atau satu hari ada empat orang melakukan penambahan atau pengurangan kode atau tulisan pada repo yang sama dan file yang sama, maka bisa dipastikan pasti akan terdapat konflik. Sehingga konflik merupakan hal yang biasa dan tidak perlu panik. Hanya kita harus mengetahui bagaimana melakukan solusi merge terhadap konflik tersebut. Karena satu satunya cara agar konflik bisa tersolusikan adalah dengan merge. Beberapa hal yang harus sering diperhatikan agar penyelesaian konflik tidak bikin kita pusing, antara lain :

- sebelum melakukan pekerjaan pastikan sudah melakukan *git pull origin master*, *git fetch upstream*, *git pull upstream master*, *git push origin master*.
- setelah menyelesaikan editing pada satu file, maka lakukan add dan commit hanya untuk satu file, hindari commit untuk beberapa file. jadi git add satufile, git commit apa yang dilakukan pada file tersebut. Jadi apabila ada dua file maka ada dua kali commit.

- sebelum melakukan `pull request` pastikan juga sudah melakukan `git pull origin master`, `git fetch upstream`, `git pull upstream master`, `git push origin master`.

Nah karena konflik adalah sebuah kepastian maka cara mengatasinya pun harus tepat. Pertama biasanya commit datang setelah melakukan `git pull origin master`, `git fetch upstream`, `git pull upstream master`, `git push origin master`. Biasanya akan muncul editor `vi` seperti pada gambar 4.1 pada saat memasukkan perintah pull. Kita tinggal menyimpannya dengan menekan tombol `esc`, tombol `:`, tombol `w`, tombol `q`, tombol enter.

**Gambar 4.1** Muncul editor vi pada saat pull

Baru setelah itu kita buka file yang konflik tersebut, sebagai contoh pada gambar 4.2. Perhatikan di gambar 4.2 ada tulisan *CONFLICT (content)*: *Merge conflict in chapters/1.tex*.

Artinya kita harus membuka file *chapters/1.tex* karena ada konflik di dalamnya. Buka dengan editor kesukaan kita dan cari tanda konflik seperti pada gambar 4.3. Perhatikan tugas kita adalah melakukan *merge*, apa itu *merge*? Merge adalah proses untuk melakukan penggabungan dari file atau baris yang telah di ubah dari beberapa versi yang konflik. Disini kita lihat bahwa pada gambar 4.3 ada tanda sama dengan sebagai pemisah antara versi satu dan versi dua dengan batas atas tanda kurang dari beberapa kali dan batas bawah lebih dari beberapa kali. Artinya kita harus menggabungkan hasil versi satu dan versi dua tersebut menjadi satu kesatuan sesuai dengan permintaan dari pemilik repo. Cara penggabungannya mau tidak mau suka atau tidak suka, kita baca manual dan perhatikan mana yang berbeda dan mana yang harus di sempurnakan kemudian di jadikan satu versi. Sehingga hasil penggabungan bisa dilihat pada gambar 4.4, tentunya batas versi dari konflik sudah dihapus jangan sampai tertinggal karena pasti akan masih dianggap konflik. Dan bisa jadi bukan hanya ada satu konflik dalam satu file tersebut, kita harus mencarilagi batas konflik dua versi tersebut. Setelah selesai maka kita bisa menyimpan dan melakukan git add dan git commit.

```
[awangga:Keleketex awangga$ git pull origin master
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 41 (delta 17), reused 33 (delta 13), pack-reused 0
Unpacking objects: 100% (41/41), done.
From github.com:BukuInformatika/Keleketex
 * branch           master    -> FETCH_HEAD
   c9a7252..38d50ba  master    -> origin/master
Auto-merging chapters/1.tex
CONFLICT (content): Merge conflict in chapters/1.tex
Automatic merge failed; fix conflicts and then commit the result.
[awangga:Keleketex awangga$ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 2 and 6 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Changes to be committed:

  new file:  figures/1.PNG
  modified:  main.pdf
  new file:  src/1/tabel.tex

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:  chapters/1.tex

[awangga:Keleketex awangga$ mate chapters/1.tex
```

**Gambar 4.2** Konflik Pada Saat *git pull*

```
32 <<<<< HEAD
33
34 \textit{Numbering} merupakan perintah yang digunakan
  . biasanya diberikan pada awal baris baru. Sedangkan
  . dengan penomoran berupa symbol atau poin (bulleted
  . penomoran .
35
36 =====
37 \textit{Numbering} merupakan perintah yang digunakan
  . biasanya diberikan pada awal baris baru \ref{lst:P}
38 >>>> 38d50bad90d1a5c5b9442a0dc1b66fa7426b46cd
```

**Gambar 4.3** Tanda pembatas antara versi satu dan dua yang konflik

```

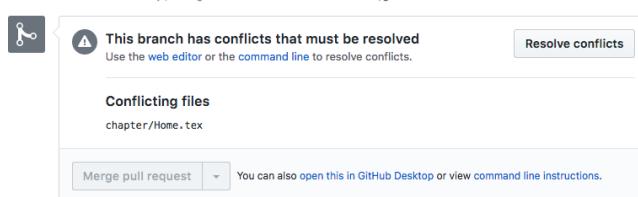
31
32
33
34 \textit{Numbering} merupakan perintah yang diguna
35 biasanya diberikan pada awal baris baru. Sedangkan
36 dengan penomoran berupa symbol atau poin (bullete
penomoran .

```

**Gambar 4.4** Konflik yang sudah diperbaiki menjadi satu versi baru lagi

## 4.2 Pada Saat Pull Request di Web

Konflik yang terjadi pada web GitHub bagian ini biasanya terjadi pada saat melakukan pull request. Tertulis pada website ada konflik seperti gambar 4.5.



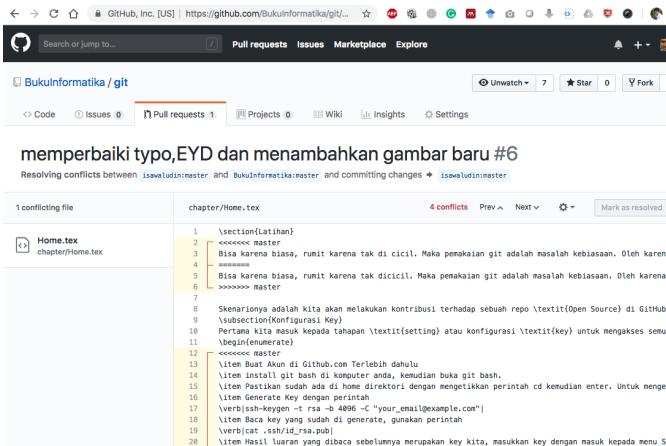
**Gambar 4.5** Konflik Pada Saat Pull Request

Tetap tenang, ini sangatlah mudah untuk dilakukan solusinya yaitu dengan merge. Pertama kita klik **Resolve conflicts** yang terlihat pada gambar 4.5. Merge adalah proses memilih salah satu atau menggabungkan bagian yang ditandai oleh git. Sebagai contoh misalnya di gambar 4.6, tertulis ada 4 konflik.

Konflik yang pertama pada tulisan *Bisa karena bisa.....* Maka kita tinggal melakukan merging dengan cara memilih kata pada baris ketiga atau kelima, atau bisa juga menggabungkan keduanya, jika memang isinya berbeda atau memodifikasi lagi. Setelah memilih jangan lupa menghapus tanda konflik seperti yang terlihat pada baris 2,4 dan 6. Baris 2 artinya itu tanda mulai, baris 6 artinya itu tanda akhir, dan baris 4 itu tanda untuk memilih apakah yang atas atau yang bawah. Sehingga definisi merge pada konflik ini terlihat pada gambar 4.7.

## 4.3 Lupa Compile dan Ingin Menambah Materi Lagi

Bagaimana jika anda lupa meng-*compile* padahal anda sudah melakukan tahap terakhir yaitu **git push origin master**. Ada solusi jitu untuk menanganinya tolong simak baik-baik ya.



**Gambar 4.6** Konflik Pada Saat Pull Request

2  
 3      **Bisa karena biasa, rumit karena tak dici**  
 4  
 5

**Gambar 4.7** Hasil Merge Setelah memilih dan menghapus tanda

1. *Compile* file **main.tex**, jika sudah selesai dan tanpa error anda kembali lagi ke gitbash.
2. Inputkan perintah **git status** untuk memeriksa kembali apa saja yang belum kita tambahkan seperti pada gambar 4.8.

```
$ git status
On branch master
Your branch is up to date with 'origin/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:   main.pdf
        modified:   section/chapter1.tex

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

**Gambar 4.8** Memeriksa yang belum ditambahkan

3. Tambahkan file yang belum ditambahkan, seperti pada gambar 4.9.

```
Udin@DESKTOP-QVM9RVJ MINGW64 /d/internship2TA/Laporan2019 (master)
$ git add main.pdf

Udin@DESKTOP-QVM9RVJ MINGW64 /d/internship2TA/Laporan2019 (master)
$ git add section/chapter1.tex
```

**Gambar 4.9** Menambahkan file

4. Melakukan proses pemeriksaan kembali untuk memastikan file yang ditambahkan sudah sesuai, seperti pada gambar 4.10.

```
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   main.pdf
        modified:   section/chapter1.tex
```

**Gambar 4.10** Memastikan file yang sudah ditambah

5. Melakukan proses *Commit* seperti pada gambar 4.11.
6. Nah ini adalah **Point Penting** yang selalu lupa untuk dilakukan yaitu melakukan perintah *git pull upstream master*, mengapa harus melakukan hal tersebut? **Alasannya untuk menghindari terjadinya konflik dan memastikan bahwa repositori kita sudah up to date** seperti pada gambar 4.12.

```
Udin@DESKTOP-QVM9RVJ MINGW64 /d/internship2TA/Laporan2019 (master)
$ git commit -m 'updated'
[master 962e21a] updated
 2 files changed, 2 insertions(+), 2 deletions(-)
```

**Gambar 4.11** Melakukan proses commit

```
Udin@DESKTOP-QVM9RVJ MINGW64 /d/internship2TA/Laporan2019 (master)
$ git pull upstream master
From https://github.com/D4TI/Laporan2019
 * branch            master      -> FETCH_HEAD
Already up to date.
```

**Gambar 4.12** Memastikan bahwa sudah up to date

7. Step terakhir sudah pasti memasukkan perintah **git push origin master** seperti pada gambar 4.13.

```
Udin@DESKTOP-QVM9RVJ MINGW64 /d/internship2TA/Laporan2019 (master)
$ git push origin master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 32.10 KiB | 1.53 MiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To github.com:isawaludin/Laporan2019.git
  689d47f..962e21a master -> master
```

**Gambar 4.13** Step terakhir

#### 4.4 Bagaimana cara mudah dan cepat untuk memperbaiki error yang tidak terdeteksi

Semakin banyak yang ikut berkontribusi di dalam sebuah repositori maka akan semakin sering menemui konflik, error merge dan permasalahan lainnya. Bagaimana solusinya jika tidak mengetahui error nya dimana. **Perhatikan dengan seksama ya.** Ini merupakan langkah yang tidak disarankan namun ampuh untuk menangani solusi.

1. Hapus repo lama di direktori masing-masing,
2. Buka git bash dan melakukan proses clone ulang dengan perintah **git clone LinkSSHdariRepo**,
3. Lalu masukkan perintah **cd RepoygSudahdiDownload**,
4. Remote kembali repo kita sesuaikan dengan repo asal dengan perintah **git remote add upstream LinkHttps dari RepoAsal**,

5. Lalu melakukan Fetch dengan perintah **git fetch upstream**,
6. Melakukan Pull dengan perintah **git pull origin master**,
7. Lakukan kembali perintah **git fetch upstream**,
8. Melakukan Pull dengan perintah **git pull upstream master**,
9. Maka akan muncul pemberitahuan errornya dimana, dan periksa di chapter/-file yang sudah diberitahukan. Dan tinggal diperbaiki yang mana baris yang tidak selaras dan yang menimbulkan error merge dan error compile. **Selamat mencoba.**

## BAB 5

---

# CARA MEMBACA PULL REQUEST

---

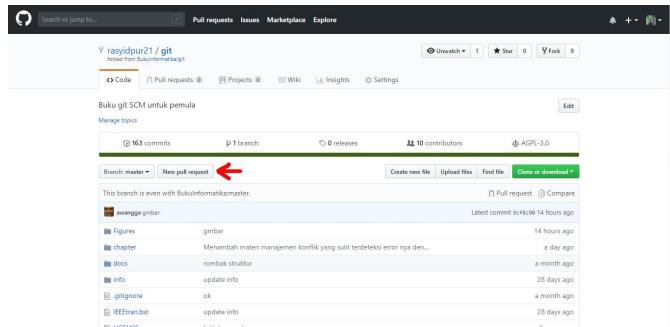
### 5.1 Membaca Pull Request Pada Github

Pada section ini kita akan mempelajari bagaimana membaca Pull Request pada Github. Untuk membaca pull request hal yang harus kita lakukan pertama adalah melakukan kontribusi terhadap project yang sedang kita kerjakan. Contohnya kita sedang membuat project tentang laporan skripsi dengan format latex, pastikan terlebih dahulu bahwa kita telah melakukan kontribusi pada project tersebut dengan memodifikasi salah satu chapter atau bagian manapun yang terdapat dalam laporan tersebut. Setelah itu barulah kita dapat membuat pull request pada repository yang kita kerjakan.

*Pull Request* sendiri dapat dikatakan sebuah istilah yang bisa kita artikan sebagai permintaan untuk menggabungkan kode. Jika kita sudah melakukan perubahan pada repository yang kita fork sebelumnya, kita dapat melakukan pull request untuk menggabungkan kode yang sudah kita modifikasi dengan repository inti atau sumber. Untuk lebih jelasnya berikut adalah hal-hal yang perlu kita lakukan untuk membuat pull request :

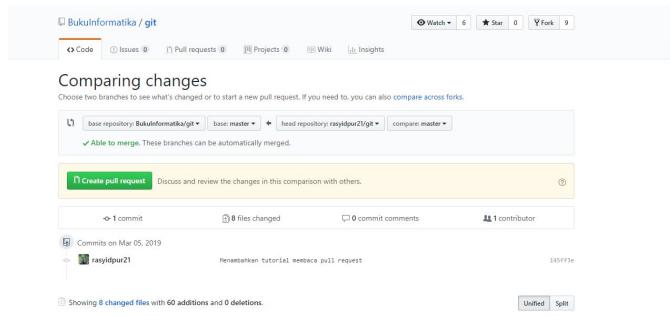
1. Buka repository yang sedang kalian kerjakan pada akun github yang kalian miliki kemudian pilih New Pull Request seperti pada gambar

## 5.1



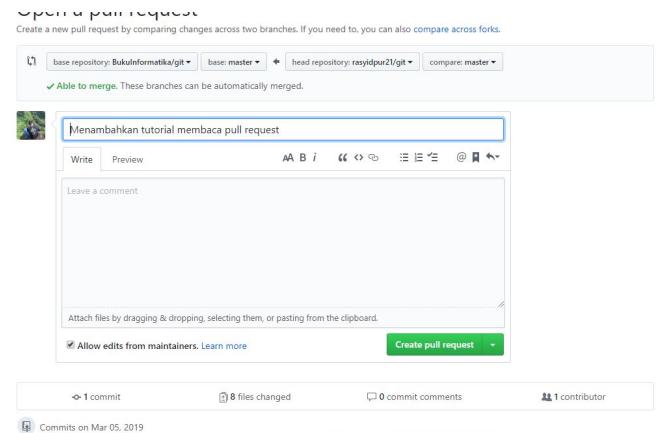
**Gambar 5.1** New Pull Request

2. Setelah itu Github akan melakukan komparasi, apakah kode yang telah dimodifikasi bentrok atau tidak.
3. Jika tidak terjadi bentrok atau konflik biasanya akan muncul notifikasi "*Able to Merge*"
4. Selanjutnya pilih button **Create Pull Request** seperti pada gambar 5.2

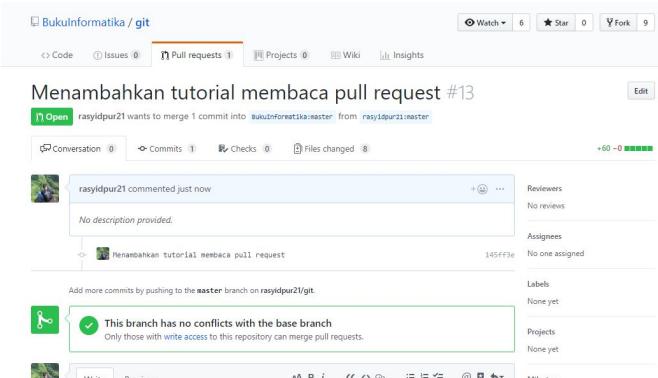


**Gambar 5.2** Create Pull Request

5. Setelah memilih button **Create Pull Request** biasanya kita diharuskan mengisi kontribusi apa yang sudah kita lakukan terlebih dahulu, seperti pada gambar 5.3
6. Selamat!! kita telah berhasil melakukan pull request seperti pada gambar 5.4



**Gambar 5.3** Mengisi Kontribusi



**Gambar 5.4** Berhasil Melakukan Pull Request

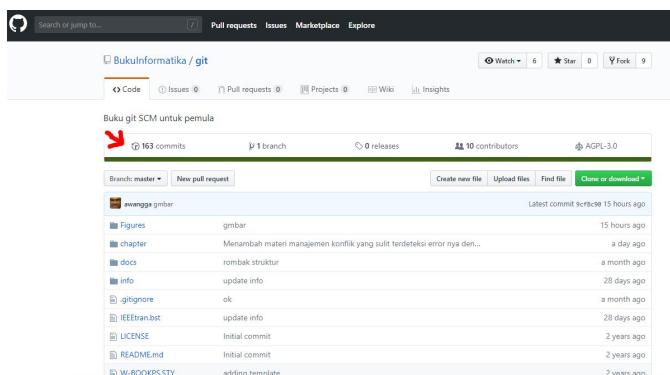
- Jika telah melakukan pull request admin atau owner akan melakukan review pada kontribusimu apakah akan diterima atau ditolak.
- Jika kontribusi yang kita lakukan telah diterima, maka akan ada notifikasi yang bertuliskan "*Merged*" yang berwarna ungu seperti pada gambar 5.5



**Gambar 5.5** Kontribusi yang sudah di Merged

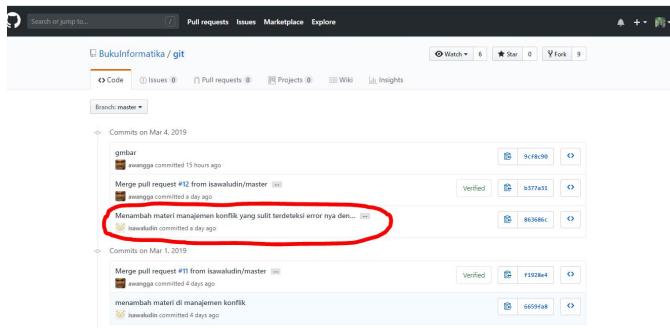
Kemudian setelah kita melakukan pull request kita dapat mengetahui kode atau perintah apa saja yang sudah kita perbaharui. Berikut adalah cara membaca pull request yang sudah kita modifikasi :

- Buka menu commits pada repository yang kalian kerjakan seperti pada gambar 5.6



**Gambar 5.6** Menu Commits

- Kemudian klik kontribusi yang telah kalian lakukan seperti pada gambar 5.7



**Gambar 5.7** Kontribusi yang telah dilakukan

- Setelah membuka kontribusi yang sudah kita lakukan, kita dapat melihat hal apa saja yang sudah kita perbaharui dalam project yang kita kerjakan. Contohnya seperti pada gambar 5.8

```

Showing 2 changed files with 8 additions and 5 deletions.

13  chapter/Home.tex
    @@ -73,13 +73,14 @@
    \subsection{Sinkronisasi dengan Repo Utama}
    73  \begin{lstlisting}[language=bash, caption=Perintah Sinkronisasi dengan repo asal,breaklines]
    74  git fetch upstream
    75  git pull upstream master
    76 + git push origin master
    77 \end{lstlisting}
    78
    79 \subsection{Bekerja dengan Git Bash}
    80 Menggunakan Git Bash bekerja pada repositori. Melakukan penambahan atau perubahan pada \textit{file}. Kemudian perubahan tersebut
    81 akan otomatis diambil oleh sistem dan repositori utama tempat kita \textit{fork} repositori kita. Urutan pekerjaan yang kita ulang terus menerus
    82 adalah sebagai berikut :
    83 \begin{itemize}
    84     \item \textbf{Sinkronisasi} dengan repositori asli \textbf{Fork} kita awal melakukan klik tombol \textbf{fork} dengan perintah :
    85     \item \textbf{Sinkronisasi} kembali dengan repositori asli \textbf{Push} kita upload file ke website \textit{github.com} dengan perintah \textbf{git push origin master}.
    86     \item \textbf{Sinkronisasi} kembali dengan repositori asli \textbf{Pull} kita \textit{pull} repositori asli ke repositori kita dengan perintah \textbf{git pull upstream master}.
    87 \end{itemize}
    88 @@ -113,12 +114,14 @@
    89 \subsection{Bekerja dengan Git Bash}
    90 Kemudian file yang sudah kita ubah kita upload ke website \textit{github.com} dengan perintah \textbf{git push origin master}.
    91 Buka web repositori kita di website \textit{www.github.com}
    92 Contoh diini:
    93
    94
    95
    96
    97
    98
    99
    100
    101
    102
    103
    104
    105
    106

```

**Gambar 5.8** Pull Request yang sudah dilakukan

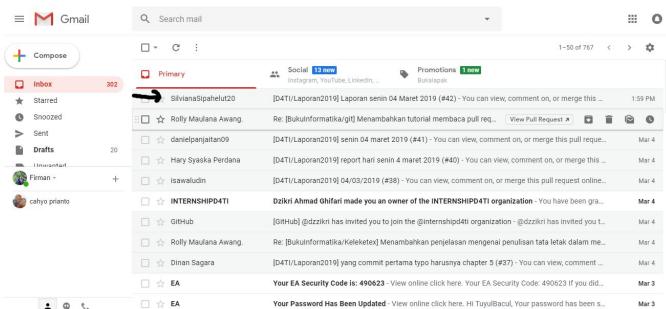
- Pada gambar 5.8 di bagian yang di lingkari garis merah terdapat notifikasi "Showing 2 changes files with 8 additions and 5 deletions"
- Arti dari notifikasi tersebut adalah terdapat 2 files yang berubah dengan menambahkan 8 dan menghapus 5 line dalam project yang kita kerjakan
- Format file yang kita tambahkan, hapus ataupun tetap (tidak berubah) ditandai dengan warna **hijau**, **merah** dan **putih** seperti pada bagian yang dilingkari dengan gari biru pada gambar 5.8

7. Line yang berubah dapat kita lihat pada bagian yang ditandai oleh garis hitam
8. Jika line berwarna hijau, artinya ada format yang ditambahkan pada file tersebut
9. Jika line berwarna merah, artinya ada format yang dihapus atau dirubah pada file tersebut
10. Jika line berwarna putih, artinya tidak ada format yang berubah pada file tersebut

## 5.2 Standar Untuk Melakukan Approve

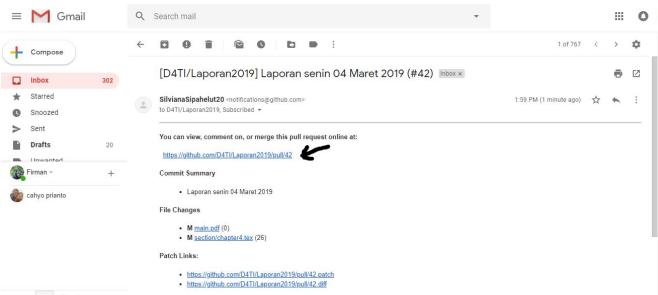
Dalam github orang yang dapat melakukan **"Approve Merge"** hanyalah **Admin** atau **Owner**. Biasanya admin akan melakukan approve jika kontribusi yang dilakukan oleh seseorang dalam suatu project telah sesuai dengan apa yang dikerjakan dan dengan apa yang dilaporkan. Pada section ini akan dijelaskan langkah-langkah bagaimana seorang **Admin** atau **Owner** menentukan standar untuk melakukan **Approve**

1. Pastikan peserta telah melakukan pull request dalam repository yang telah admin buat
2. Biasanya jika peserta telah melakukan pull request maka admin akan mendapatkan notifikasi di alamat emailnya seperti pada gambar 5.9

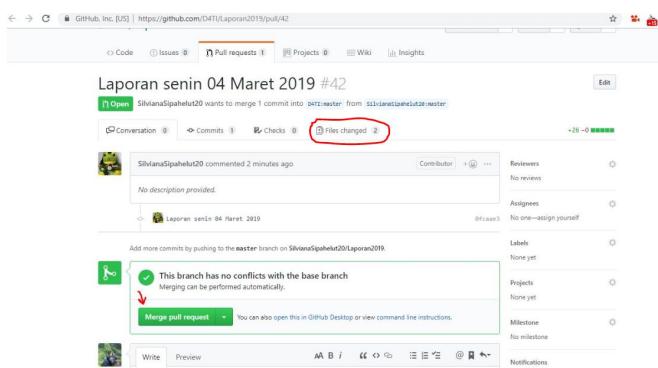


Gambar 5.9 Notifikasi Pada Email

3. Kemudian buka notifikasi email tersebut dan klik link untuk melihat pull request yang dilakukan peserta seperti pada gambar 5.10
4. setelah terhubung dengan link github diatas, pilih menu **File Changed** untuk melihat perubahan apa saja yang telah dilakukan oleh kontributor seperti pada gambar 5.11:

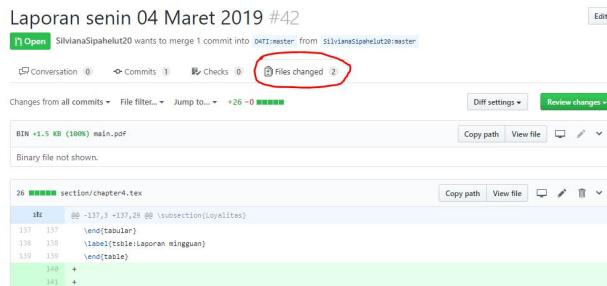


**Gambar 5.10** View Pull Request Online



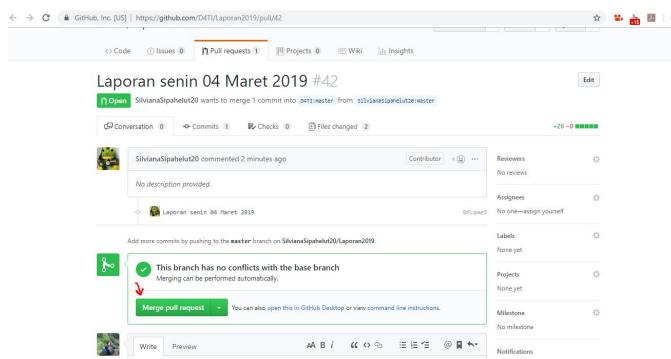
**Gambar 5.11** Melihat Files Changed

- Setelah itu kita akan melihat apa saja yang telah dimodifikasi oleh kontributor seperti pada gambar 5.12:



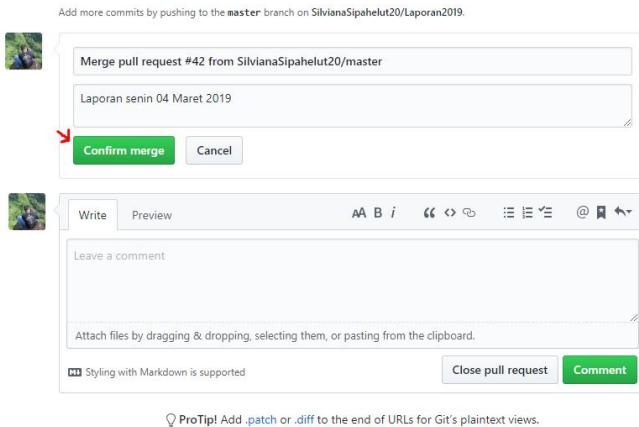
**Gambar 5.12** File Kontribusi

- Dari perubahan yang telah dilakukan, pastikan bahwa isi dari apa yang dilaporkan kontributor sesuai dengan modifikasi atau perubahan yang telah dia lakukan pada project tersebut
- Setelah itu admin dapat kembali ke halaman sebelumnya untuk melakukan merging
- Jika file yang di rubah sesuai maka admin dipersilahkan memilih button *Merge Pull Request* seperti pada gambar 5.13



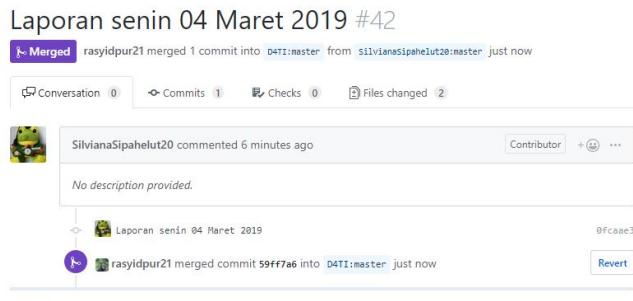
**Gambar 5.13** Merge Pull Request

- Setelah itu langkah selanjutnya adalah mengklik button **Confirm Merge** untuk melakukan konfirmasi seperti pada gambar 5.14



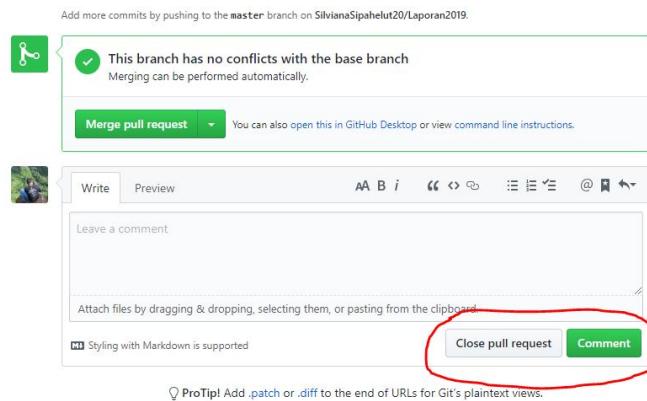
**Gambar 5.14** Confirm Merge

- Setelah melakukan langkah-langkah tersebut, admin telah berhasil melakukan Approve dengan Merge Pull Request seperti pada gambar 5.15



**Gambar 5.15** Approved Merge Pull Request

- Catatan : Jika isi kontribusi tidak sesuai admin dapat memilih button **Close Pull Request** dan memberikan komentar mengenai hal yang harus diperbaiki pada button **Comment** seperti pada gambar 5.16:



**Gambar 5.16** Closed and Comment

## DAFTAR PUSTAKA

---

1. R. Awangga, “Sampeu: Servicing web map tile service over web map service to increase computation performance,” in *IOP Conference Series: Earth and Environmental Science*, vol. 145, no. 1. IOP Publishing, 2018, p. 012057.



# Index

---

disruptif, xxxi  
modern, xxxi