

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 Tunnisa

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. Sariasih 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

1	Praktek Cepat	1
2	Perintah Dasar Bash	13
3	Mengatasi Konfik	21
4	Cara Membaca Pull Request	29

DAFTAR ISI

Daftar Gambar	xi
Daftar Tabel	xiii
Foreword	xvii
Kata Pengantar	xix
Acknowledgments	xxi
Acronyms	xxiii
Glossary	xxv
List of Symbols	xxvii
Introduction	xxix
<i>Rolly Maulana Awangga, S.T., M.T.</i>	
1 Praktek Cepat	1
1.1 Latihan	1
1.1.1 Konfigurasi Key	1
1.1.2 Fork Repositori	4
1.1.3 Navigasi direktori dengan Git Bash	4
	ix

1.1.4	Sinkronisasi dengan Repo Utama	6
1.1.5	Bekerja dengan Git Bash	6
1.1.6	Mengatasi <i>Error</i>	8
2	Perintah Dasar Bash	13
2.1	Perintah Dasar Bash	13
2.1.1	fungsi perintah pada Bash	13
2.1.2	penggunaan VI editor	18
3	Mengatasi Konfik	21
3.1	Pada Repo Lokal	21
3.2	Pada Saat Pull Request di Web	24
3.3	Lupa Compile dan Ingin Menambah Materi Lagi	24
3.4	Bagaimana cara mudah dan cepat untuk memperbaiki error yang tidak terdeteksi	27
4	Cara Membaca Pull Request	29
4.1	Membaca Pull Request Pada Github	29

DAFTAR GAMBAR

1.1	Perintah keygen cukup enter enter saja	2
1.2	Perintah cat untuk membaca key	3
1.3	Setting	3
1.4	New SSH key	3
1.5	Letak Tombol Fork	4
1.6	Hasil Perintah git clone	5
1.7	Hasil Perintah git clone	5
1.8	Hasil Perintah ls, cd dan git status	5
1.9	Direktori kerja git dari Explorer	7
1.10	Permintaan Setting Global	7
1.11	Setelah klik New pull request	8
1.12	Gagal Fetch Upstream	10
1.13	Berbagai macam luaran git status	11

1.14	Permintaan Merge dengan pesan standar yang keluar	11
2.1	Perintah dasar ls pada git bash	14
2.2	Perintah dasar cd pada git bash	14
2.3	Perintah dasar pwd pada git bash	15
2.4	Perintah dasar mv pada git bash	15
2.5	Perintah dasar cp pada git bash	16
2.6	melakukan commit menggunakan VIM editor	16
2.7	mengisi apa yang kita commit	17
2.8	keluar dari halaman VIM editor	17
2.9	membuat file	18
2.10	menambahkan data file	18
2.11	menyimpan file	19
2.12	menyimpan file lalu keluar	19
2.13	keluar tanpa mengubah sesuatu pada file	20
3.1	Muncul editor vi pada saat pull	22
3.2	Konflik Pada Saat <i>git pull</i>	23
3.3	Tanda pembatas antara versi satu dan dua yang konflik	23
3.4	Konflik yang sudah diperbaiki menjadi satu versi baru lagi	24
3.5	Konflik Pada Saat Pull Request	24
3.6	Konflik Pada Saat Pull Request	25
3.7	Hasil Merge Setelah memilih dan menghapus tanda	25
3.8	Memeriksa yang belum ditambahkan	26
3.9	Menambahkan file	26
3.10	Memastikan file yang sudah ditambah	26
3.11	Melakukan proses commit	27
3.12	Memastikan bahwa sudah up to date	27
3.13	Step terakhir	27
4.1	New Pull Request	30
4.2	Kontribusi yang sudah di Merged	30
4.3	Menu Commits	31
4.4	Kontribusi yang telah dilakukan	31
4.5	Pull Request yang sudah dilakukan	32

DAFTAR TABEL

Listings

1.1	Perintah Membuat Key	2
1.2	Perintah Membaca Public Key	2
1.3	Navigasi direktori menuju repositori	4
1.4	Set Repo Asal Sebagai Upstream	6
1.5	Perintah Sinkronisasi dengan repo asal	6
1.6	Perintah Sinkronisasi dengan repo asal	7
1.7	Kesalahan karena bukan pada direktori repositori	9
1.8	Gagal melakukan fetch upstream	9
1.9	Melihat konfigurasi git di repo komputer kita	9
1.10	Hasil perintah <code>git config -l</code>	9
1.11	Perintah menghapus upstream yang salah	9
1.12	Peringatan <i>Permission denied</i>	10

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

- \mathcal{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[?].

$$ABCDEF\alpha\beta\Gamma\Delta\sum_{def}^{abc} \tag{I.1}$$

BAB 1

PRAKTEK CEPAT

1.1 Latihan

Bisa karena biasa, rumit karena tak dicicil. Maka pemakaian git adalah masalah kebiasaan. Oleh karena itu, dalam pembukaan awal buku ini justru kita tidak dulu masuk kepada teori git tapi langsung praktek. setelah berkali-kali praktek 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*.

1.1.1 Konfigurasi Key

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

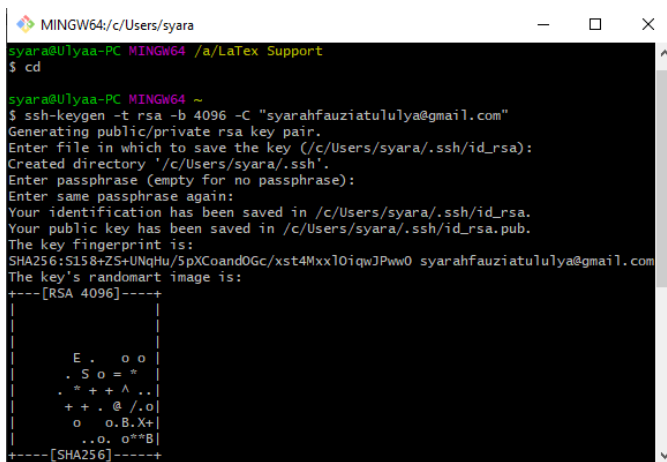
1. Buat akun di www.github.com terlebih dahulu

2. Install **Git Bash** dari alamat 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 1.1.

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

Listing 1.1 Perintah Membuat Key

Hasilnya seperti yang terlihat pada gambar 1.1.



```
MINGW64:/c/Users/syara
syara@Ulyaa-PC MINGW64 /a/LaTeX_Support
$ cd
syara@Ulyaa-PC MINGW64 ~
$ 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:S1S8+ZS+UNqHu/SpXCoandOGc/xst4Mxx10iqwJPww0 syarahfauziatululya@gmail.com
The key's randomart image is:
+----[RSA 4096]-----+
|
|   E .  o o |
|  . S o = * |
|   . * + + ^ . . |
|  + + . @ / . o |
|   o . B . X + |
|  .. o . o * * B |
+----[SHA256]-----+
```

Gambar 1.1 Perintah keygen cukup enter enter saja

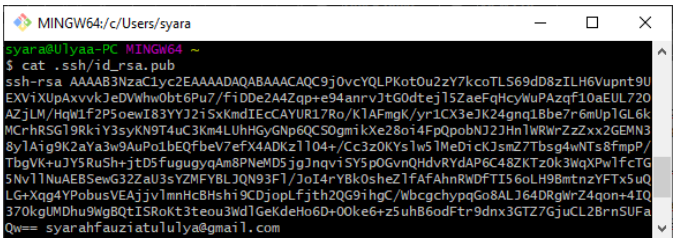
5. Baca *key* yang sudah di *generate* dengan menggunakan perintah 1.2.

```
1 cat .ssh/id_rsa.pub
```

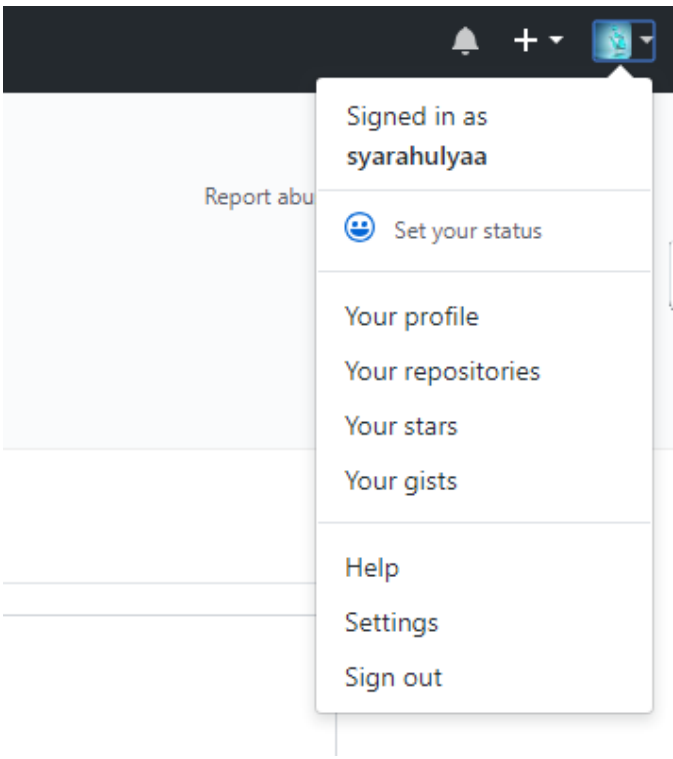
Listing 1.2 Perintah Membaca Public Key

Hasilnya seperti pada gambar 1.2.

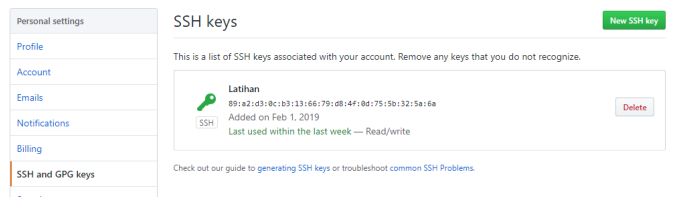
6. Hasil luaran yang dibaca sebelumnya merupakan *key* kita. Kemudian masukkan *key* dengan masuk ke menu **Setting** yang ada dipojok kanan atas, seperti pada gambar 1.3. Lalu pada menu **SSH and GPG Keys** tambahkan **New SSH key** seperti pada gambar 1.4.



Gambar 1.2 Perintah cat untuk membaca key



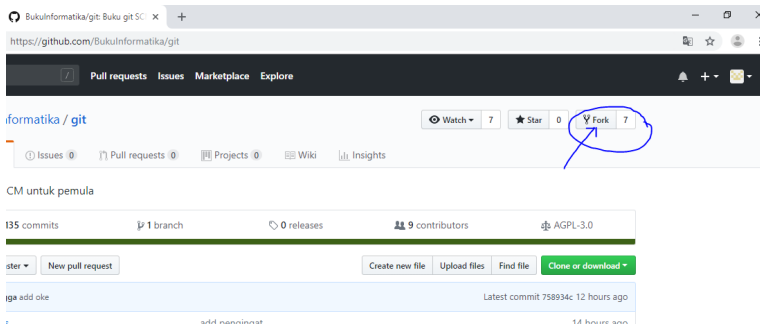
Gambar 1.3 Setting



Gambar 1.4 New SSH key

1.1.2 Fork Repositori

Pertama kita cari repositori utama yang akan kita jadikan tempat berkontribusi dalam repositori tersebut. Jika sudah ketemu, kemudian klik Fork(Tombol kanan atas) seperti pada gambar 1.5 yang dilanjutkan dengan memilih akun kita sebagai tujuan clone



Gambar 1.5 Letak Tombol Fork

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](https://github.com/usernameAnda/Testing), dan kita akan bekerja pada repo hasil fork ini.

1.1.3 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 1.6) lalu salin kode yang tampak seperti `git@github.com:usernameAnda/Testing.git`.

Pada Git Bash ketik `git clone git@github.com:usernameAnda/Testing.git` hasilnya seperti terlihat pada gambar 1.7.

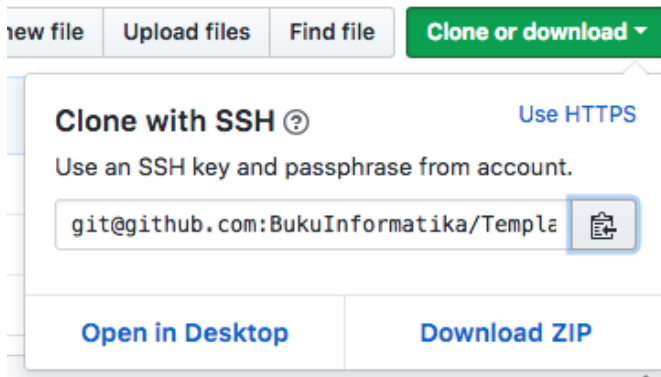
Setelah selesai, ketik perintah `ls` maka akan muncul direktori baru yaitu folder Testing atau sesuai dengan nama repo Anda. masuk ke direktori tersebut dengan perintah `cd Testing`. Kemudian ketik `git status` akan muncul status dari repo git kita, ringkasan perintahnya bisa dilihat pada perintah listing 1.3.sebagai contoh lihat hasilnya pada gambar 1.8.

```

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

```

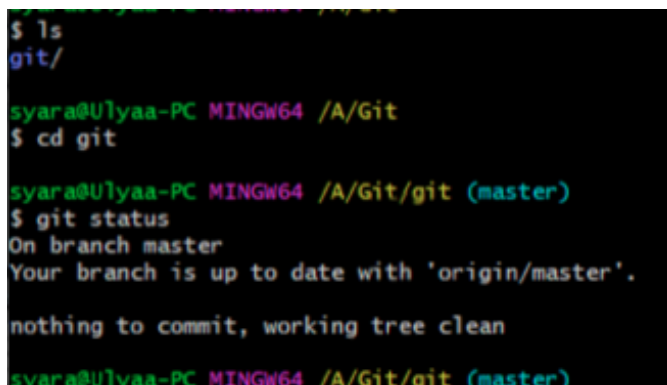
Listing 1.3 Navigasi direktori menuju repositori



Gambar 1.6 Hasil Perintah git clone



Gambar 1.7 Hasil Perintah git clone



Gambar 1.8 Hasil Perintah ls, cd dan git status

1.1.4 Sinkronisasi dengan Repo Utama

Karena ini repo hasil Fork maka kita harus selalu di sinkronisasi dari repo aslinya(tidak otomatis tersinkron). 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 1.4.

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

Listing 1.4 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 1.5.

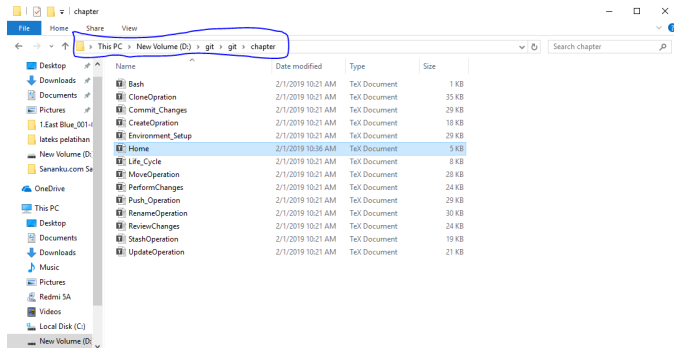
```
1 git pull origin master
2 git fetch upstream
3 git pull upstream master
4 git push origin master
```

Listing 1.5 Perintah Sinkronisasi dengan repo asal

1.1.5 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 1.5.
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 1.9.
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



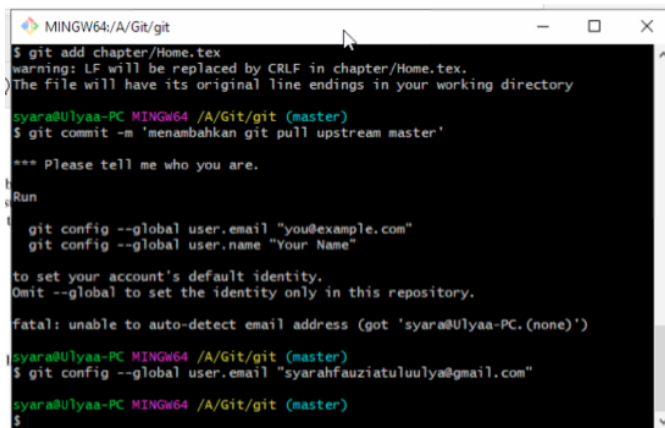
Gambar 1.9 Direktori kerja git dari Explorer

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 1.10. Maka kita harus memasukkan konfigurasi global dengan perintah 1.6. 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"
```

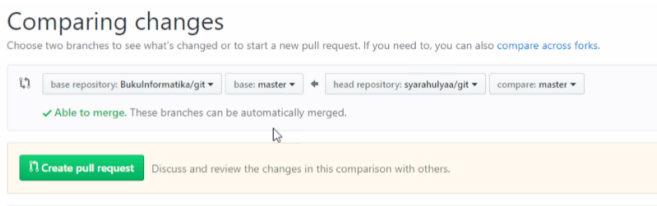
Listing 1.6 Perintah Sinkronisasi dengan repo asal



Gambar 1.10 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 1.13.
9. Kemudian file yang sudah kita ubah kita upload ke website 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 1.5. 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 1.11. Klik tombol hijau *Create Pull Request* kemudian teruskan sampai ada kembali tombol hijau yang kita klik lagi.



Gambar 1.11 Setelah klik New pull request

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

1.1.6 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.

1.1.6.1 Kesalahan direktori Apabila bertemu error seperti pada listing 1.7. Pastikan git bash kita berada pada direktori repositori git yang dikerjakan, biasanya ditandai ada nama *branch* di git bash nya.

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

Listing 1.7 Kesalahan karena bukan pada direktori repositori

Sebagai contoh pada gambar 1.10 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 didalam folder Git ada repo clone dengan folder git(g kecil). Posisi kita ada di dalam folder git tersebut. Sebagai contoh kedua pada gambar 1.9, maka tulisan pada git bash tempat kita bekerja menjadi */D/git/git* atau */D/git/git/chapter*.

1.1.6.2 Gagal Fetch Upstream Apabila anda pada saat melakukan *git fetch upstream* atau *git pull upstream master* keluar error seperti pada listing 1.8.

```
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 1.8 Gagal melakukan fetch upstream

Maka ada dua kemungkinan, kemungkinan pertama anda belum melakukan set upstream seperti pada perintah di listing 1.4. Silahkan lakukan dahulu perintah di listing 1.4. Salah satu contoh repo yang belum melakukan perintah dari listing 1.4 bisa dilihat di gambar 1.12.

Apabila anda sudah melakukan set upstream maka untuk melakukan pengecekan kita lihat dengan perintah di listing 1.9.

```
1 git config -l
```

Listing 1.9 Melihat konfigurasi git di repo komputer kita

Jika sudah ter set maka akan ada bagian yang berisi *remote.upstream.url* seperti pada listing 1.10.

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

Listing 1.10 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 1.11. Baru setelah itu lakukan lagi set upstream dengan perintah di listing 1.4.

```
1 git remote remove upstream
```

Listing 1.11 Perintah menghapus upstream yang salah


```

[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.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=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 1.12 Gagal Fetch Upstream

1.1.6.3 Salah Clone Buat pemula, kesalahan ini kadang terjadi jika tidak hati-hati membaca langkah-langkah sebelumnya. Apabila bertemu pesan kesalahan seperti pada listing 1.12. Dari listing tersebut terlihat *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 1.12 Peringatan *Permission denied*

1.1.6.4 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 1.13, 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*.

1.1.6.5 Konflik Ketiga, sebelum melakukan pekerjaan dan sebelum melakukan **New pull request**(dua kali). Pastikan repo lokal di komputer kita sinkron dengan

Gambar 1.13 Berbagai macam luaran git status

```

Merge branch 'master' of github.com:BukuInformatika/git

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.

:wo

```

Gambar 1.14 Permintaan Merge dengan pesan standar yang keluar

BAB 2

PERINTAH DASAR BASH

2.1 Perintah Dasar Bash

2.1.1 fungsi perintah pada Bash

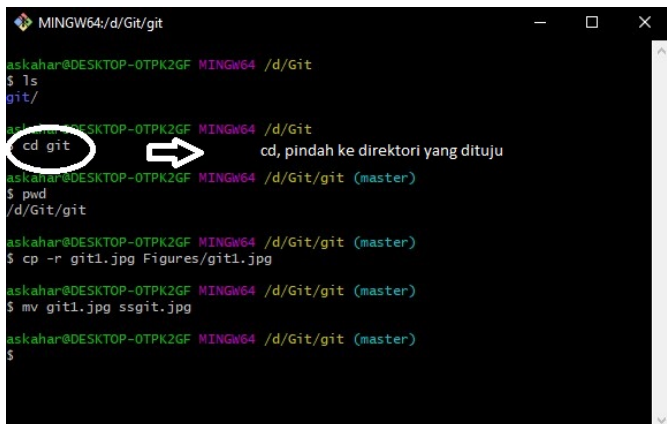
1. `ls` fungsi perintah **ls** pada bash ini yaitu untuk melihat isi dari suatu direktori. Contoh lihat pada gambar 2.1
2. `cd` fungsi perintah **cd** (*change directory*) pada bash ini yaitu untuk berpindah ke direktori yang dituju. Contoh lihat pada gambar 2.2
3. `pwd` fungsi perintah **pwd** pada bash ini yaitu untuk mengetahui path direktori yang sedang aktif. Contoh lihat pada gambar 2.3
4. `mv` fungsi perintah **mv** pada bash ini yaitu untuk mengubah nama file. Contoh lihat pada gambar 2.4
5. `cp` fungsi perintah **cp** pada bash ini yaitu untuk *mencopy file*. Contoh lihat pada gambar 2.5



```
MINGW64:/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$
```

A white circle highlights the `ls` command in the first line of the terminal output. A white arrow points from this circle to the text "ls, melihat isi dari direktori" on the right side of the image.

Gambar 2.1 Perintah dasar ls pada git bash



```
MINGW64:/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$
```

A white circle highlights the `cd git` command in the terminal output. A white arrow points from this circle to the text "cd, pindah ke direktori yang dituju" on the right side of the image.

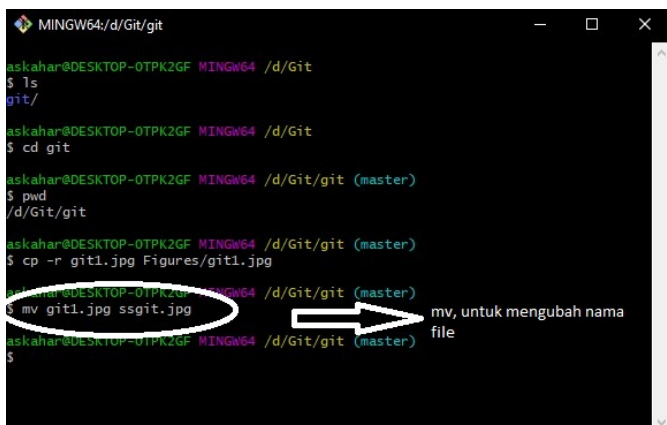
Gambar 2.2 Perintah dasar cd pada git bash



```
MINGW64:/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$
```

A white circle highlights the `$ pwd` command and its output `/d/Git/git`. A white arrow points from this output to the text "pwd, mengetahui path direktori saat ini".

Gambar 2.3 Perintah dasar pwd pada git bash



```
MINGW64:/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ ls
git/
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git
$ cd git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ pwd
/d/Git/git
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ cp -r git1.jpg Figures/git1.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$ mv git1.jpg ssgit.jpg
askahar@DESKTOP-0TPK2GF MINGW64 /d/Git/git (master)
$
```

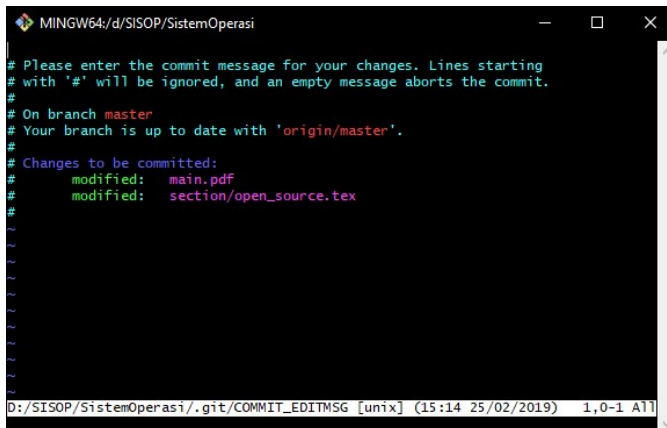
A white circle highlights the `$ mv git1.jpg ssgit.jpg` command. A white arrow points from this command to the text "mv, untuk mengubah nama file".

Gambar 2.4 Perintah dasar mv pada git bash



Gambar 2.5 Perintah dasar cp pada git bash

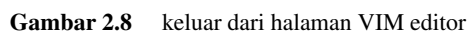
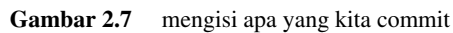
6. :wq fungsi perintah **:wq** pada bash ini yaitu untuk keluar dari Vim editor ketika kita melakukan perubahan file menggunakan Vim editor. Contoh ketika melakukan commit menggunakan VIM editor 2.6



Gambar 2.6 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 2.7

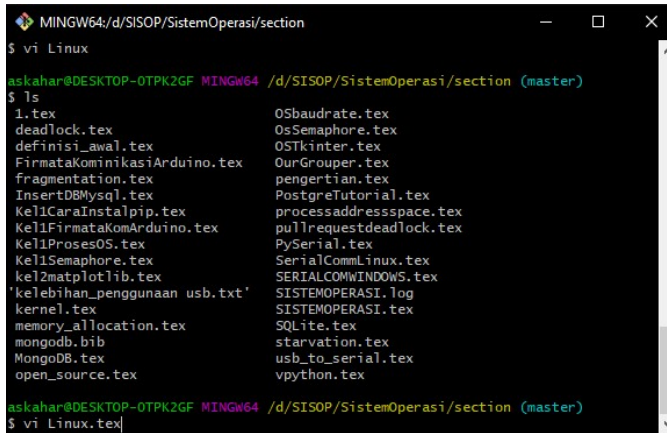
jika, kita telah melakukan commit maka untuk keluar dari halaman VIM editor ini ketikan **:wq** untuk keluar dari halaman VIM editor 2.8



2.1.2 penggunaan VI editor

1. menambahkan file menggunakan VI editor di bash

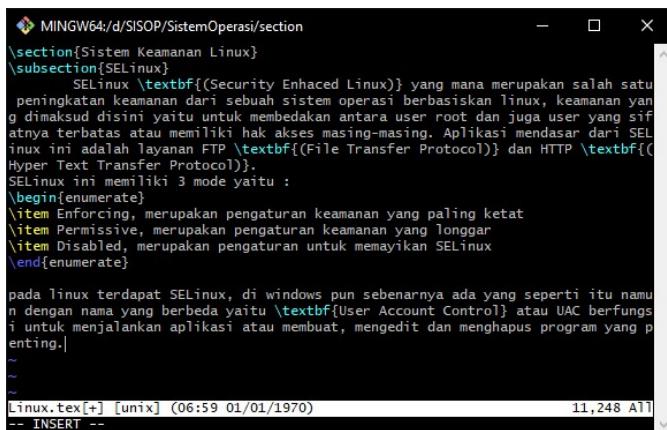
ketikan `vi namafil` di bash untuk menambahkan file 2.9



```
MINGW64/d/SISOP/SistemOperasi/section
$ vi Linux
askahar@DESKTOP-0TPK2GF MINGW64 /d/SISOP/SistemOperasi/section (master)
$ ls
1.tex                      OSbaudrate.tex
deadlock.tex               OSSemaphore.tex
definisi_awal.tex          OSTkinter.tex
FirmataKominikasiArduino.tex OurGrouper.tex
fragmentation.tex          pengertian.tex
InsertDBMysql.tex          PostgreTutorial.tex
Kel1CaraInstalpip.tex      processaddressspace.tex
Kel1FirmataKomArduino.tex  pullrequestdeadlock.tex
Kel1ProsesOS.tex           PySerial.tex
Kel1Semaphore.tex          SerialComLinux.tex
kel2matplotlib.tex         SERIALCOMWINDOWS.tex
'kelebihan_penggunaan usb.txt' SISTEMOPERASI.log
kernel.tex                 SISTEMOPERASI.tex
memory_allocation.tex      SQLite.tex
mongodb.bib                starvation.tex
MongoDB.tex                usb_to_serial.tex
open_source.tex            vpython.tex
askahar@DESKTOP-0TPK2GF MINGW64 /d/SISOP/SistemOperasi/section (master)
$ vi Linux.tex
```

Gambar 2.9 membuat file

- 2.10 lalu akan masuk ke halaman, tekan `i` untuk menambahkan isi file tersebut

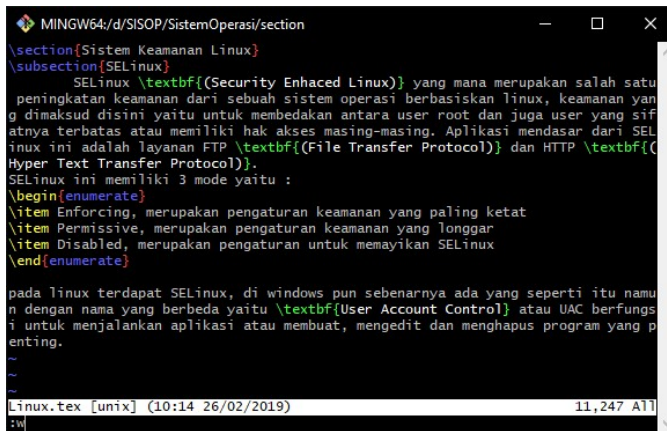


```
MINGW64/d/SISOP/SistemOperasi/section
\section{Sistem Keamanan Linux}
\subsection{SELinux}
SELinux \textbf{(Security Enhaced Linux)} yang mana merupakan salah satu
peningkatan keamanan dari sebuah sistem operasi berbasis linux, keamanan yan
g dimaksud disini yaitu untuk membedakan antara user root dan juga user yang sif
atnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SEL
inux 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 namu
n dengan nama yang berbeda yaitu \textbf{User Account Control} atau UAC berfungs
i untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang p
enting.
~
~
Linux.tex[+] [unix] (06:59 01/01/1970) 11,248 All
-- INSERT --
```

Gambar 2.10 menambahkan data file

- 2.11 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

untuk menyimpan file lalu keluar dari halaman vi maka ketikan `:wq` 2.12

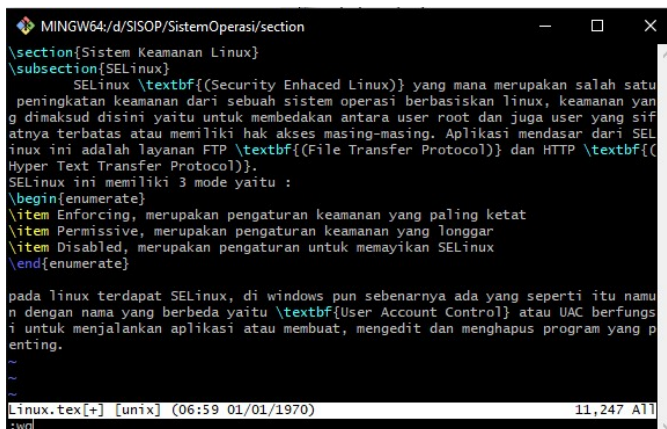


```

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 berbasis linux, keamanan yang
    dimaksud disini yaitu untuk membedakan antara user root dan juga user yang sif
    atnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SEL
    inux ini adalah layanan FTP \textbf{(File Transfer Protocol)} dan HTTP \textbf{(H
    yper 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 namu
    n dengan nama yang berbeda yaitu \textbf{(User Account Control)} atau UAC berfungs
    i untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang p
    enting.
    ~
    ~
    ~
    Linux.tex [unix] (10:14 26/02/2019) 11,247 All
    :w|
  
```

Gambar 2.11 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 berbasis linux, keamanan yang
    dimaksud disini yaitu untuk membedakan antara user root dan juga user yang sif
    atnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SEL
    inux ini adalah layanan FTP \textbf{(File Transfer Protocol)} dan HTTP \textbf{(H
    yper 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 namu
    n dengan nama yang berbeda yaitu \textbf{(User Account Control)} atau UAC berfungs
    i untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang p
    enting.
    ~
    ~
    ~
    Linux.tex[+] [unix] (06:59 01/01/1970) 11,247 All
    :wq|
  
```

Gambar 2.12 menyimpan file lalu keluar

namun jika tidak ingin menyimpan atau mengubah file tersebut (*discard all changes*) maka ketikkan :**q!** 2.13

```

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 berbasis linux, keamanan yan
g dimaksud disini yaitu untuk membedakan antara user root dan juga user yang sif
atnya terbatas atau memiliki hak akses masing-masing. Aplikasi mendasar dari SEL
inux 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 namu
n dengan nama yang berbeda yaitu \textbf{User Account Control} atau UAC berfungs
i untuk menjalankan aplikasi atau membuat, mengedit dan menghapus program yang p
enting.

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

```

Gambar 2.13 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 sebelum
vi -R filename	Membuka file yang sudah dibuat sebelumnya, namun hanya dapat dibaca tanpa bisa d
view filename	Sama seperti perintah sebelumnya hanya dapat membaca tanpa bisa melakukan ed

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 3

MENGATASI KONFIK

3.1 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 3.1 pada saat memasukkan perintah pull. Kita tinggal menyimpannya dengan menekan tombol *esc*, tombol *:*, tombol *w*, tombol *q*, tombol *enter*.

[illegible]

Gambar 3.1 Muncul editor vi pada saat pull

Baru setelah itu kita buka file yang konflik tersebut, sebagai contoh pada gambar 3.2. Perhatikan di gambar 3.2 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 3.3. 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 3.3 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 3.4, 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

```

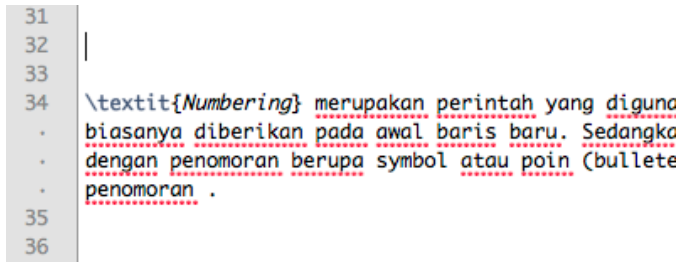
Gambar 3.2 Konflik Pada Saat *git pull*

```

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

```

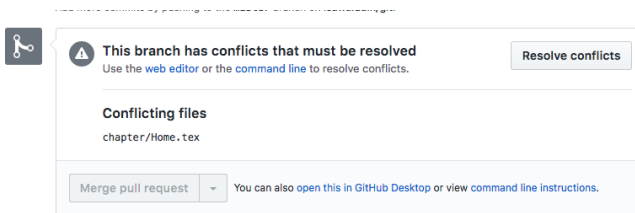
Gambar 3.3 Tanda pembatas antara versi satu dan dua yang konflik



Gambar 3.4 Konflik yang sudah diperbaiki menjadi satu versi baru lagi

3.2 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 3.5.



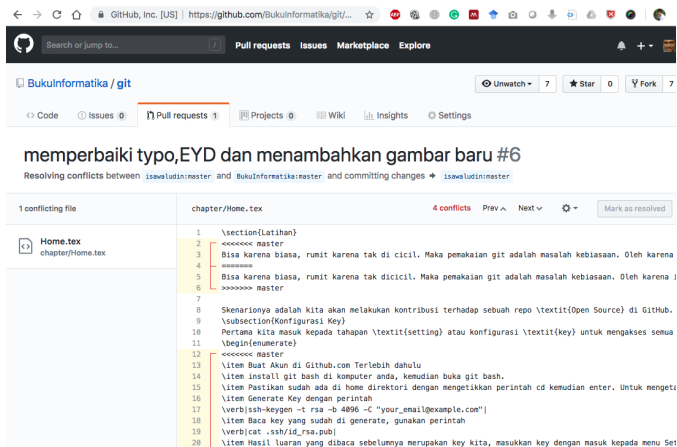
Gambar 3.5 Konflik Pada Saat Pull Request

Tetap tenang, ini sangatlah mudah untuk dilakukan solusinya yaitu dengan merge. Pertama kita klik Resolve conflicts yang terlihat pada gambar 3.5. Merge adalah proses memilih salah satu atau menggabungkan bagian yang ditandai oleh git. Sebagai contoh misalnya di gambar 3.6, tertulis ada 4 konflik.

Konflik yang pertama pada tulisan *Bisa karena biasa.....* 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 3.7.

3.3 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 3.6 Konflik Pada Saat Pull Request

```

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

```

Gambar 3.7 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 3.8.

```
$ 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 3.8 Memeriksa yang belum ditambahkan

3. Tambahkan file yang belum ditambahkan, seperti pada gambar 3.9.

```
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 3.9 Menambahkan file

4. Melakukan proses pemeriksaan kembali untuk memastikan file yang ditambahkan sudah sesuai, seperti pada gambar 3.10.

```
$ 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 3.10 Memastikan file yang sudah ditambah

5. Melakukan proses *Commit* seperti pada gambar 3.11.
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 3.12.

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

Gambar 3.11 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 3.12 Memastikan bahwa sudah up to date

7. Step terakhir sudah pasti memasukkan perintah **git push origin master** seperti pada gambar 3.13.

```
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 3.13 Step terakhir

3.4 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 direktory 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 LinkHttpsdariRepoAsal,**

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 4

CARA MEMBACA PULL REQUEST

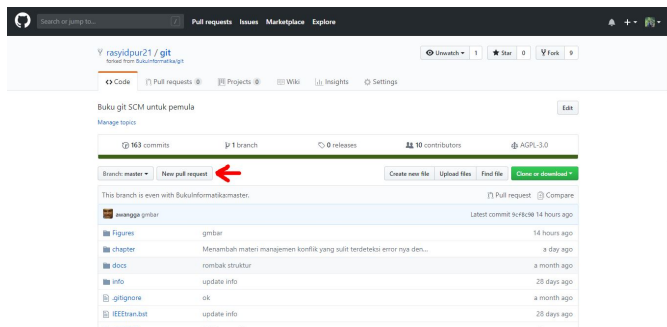
4.1 Membaca Pull Request Pada Github

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 :

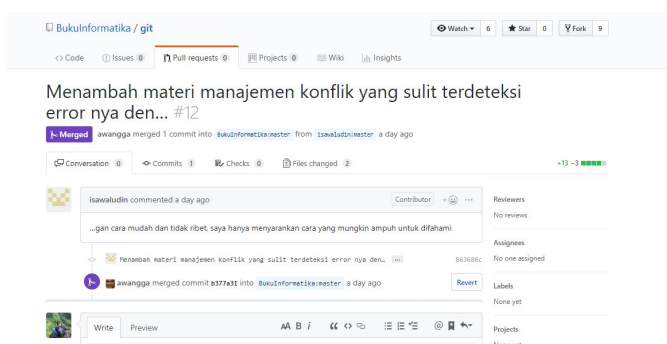
1. Buka repository yang sedang kalian kerjakan pada akun github yang kalian miliki kemudian pilih New Pull Request seperti pada gambar

4.1.



Gambar 4.1 New Pull Request

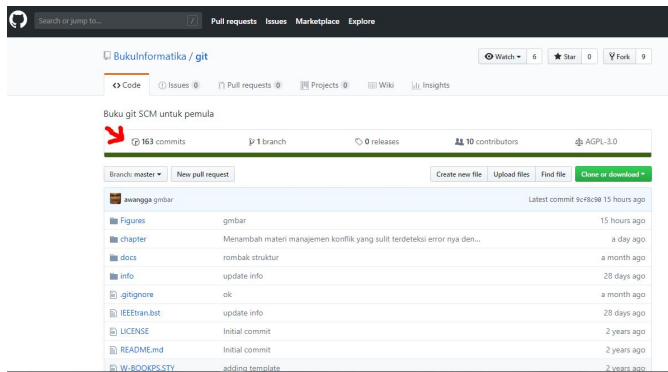
- Setelah itu Github akan melakukan komparasi, apakah kode yang telah dimodifikasi bentrok atau tidak.
- Jika tidak terjadi bentrok atau konflik biasanya akan muncul notifikasi "*Able to Merge*"
- Selanjutnya pilih button **Create Pull Request** seperti pada gambar :
- Jika telah melakukan pull request 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 4.2:



Gambar 4.2 Kontribusi yang sudah di Merged

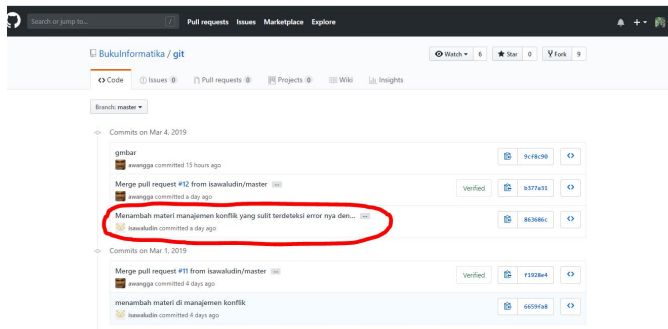
Kemudian setelah kita melakukan pull request kita dapat mengetahui kode atau perintah apa saja yang sudah kita perbaharui. Berikut adalah cara membaca pull request yang sudah kita modifikasi :

1. Buka menu commits pada repository yang kalian kerjakan seperti pada gambar 4.3.



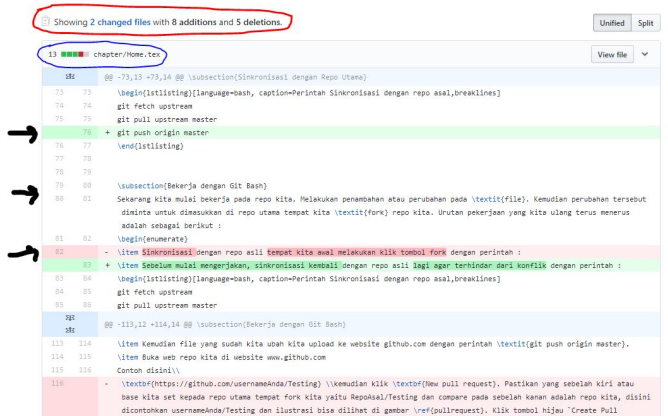
Gambar 4.3 Menu Commits

2. Kemudian klik kontribusi yang telah kalian lakukan seperti pada gambar 4.4 :



Gambar 4.4 Kontribusi yang telah dilakukan

3. 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 4.5 :
4. Pada gambar 4.5 di bagian yang di lingkari garis merah terdapat notifikasi "Showing 2 changes files with 8 additions and 5 deletions"
5. Arti dari notifikasi tersebut adalah terdapat 2 files yang berubah dengan menambahkan 8 dan menghapus 5 line dalam project yang kita kerjakan



Gambar 4.5 Pull Request yang sudah dilakukan

6. Format file yang kita tambahkan, hapus ataupun tetap (tidak berubah) ditandai dengan warna **hijau**, **merah** dan **putih** seperti pada bagian yang dilingkari dengan garis biru pada gambar 4.5
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 dihapuskan atau dirubah pada file tersebut
10. Jika line berwarna putih, artinya tidak ada format yang berubah pada file tersebut

4.2 Standar Untuk Melakukan Approve