**LAPORAN PEMBUATAN APLIKASI**

**BEKERJA DENGAN GITHUB**

**DI SMK NEGERI 1 KARANG BARU**

**TAHUN 2024**



**DISUSUN OLEH:**

**Nama : Ahmadi Muslim**

**NISN : 0074586930**

**Program Keahlian : Teknik Komputer dan Informatika**

**Kompetensi Keahlian : Rekayasa Perangkat Lunak**

**PEMERINTAH ACEH**

**DINAS PENDIDIKAN**

**SEKOLAH MENENGAH KEJURUAN (SMK) NEGERI 1 KARANG BARU**

**KABUPATEN ACEH TAMIANG**

**TAHUN 2024**

# LEMBAR PENGESAHAN

**LAPORAN PEMBUATAN APLIKASI**

**PENJUALAN BENGKEL KOMPUTER**

**DI SMKN 1 KARANG BARU**

**TAHUN 2024**

**Nama : Ahmadi Muslim**

**NISN : 0074586930**

**Program Keahlian : Teknik Komputer dan Informatika**

**Kompetensi Keahlian : Rekayasa Perangkat Lunak**

Telah disahkan tanggal, ……………………………….

Menyutujui:

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | Pembimbing Sekolah  **Ahmadi Muslim, MP**  NIP. 19860505 201403 1 003 |

# KATA PENGANTAR

Puji syukur saya ucapkan Kehadirat Allah SWT. karena dengan rahmat dan karunia-Nya penulis dapat menyelesaikan laporan praktek kerja industri (Prakerin) yang telah dilaksanakan di SMKN 1 Karang Baru Kabupaten Aceh Tamiang.

Sehubungan dengan terlaksanakannya Prakerin ini tidak terlepas dari bantuan dan dorongan dari semua pihak secara moril maupun materil, oleh karena itu penulis mengucapkan terima kasih sebesar-besarnya kepada:

1. Bapak Fahmi Putra, S.Pd selaku Kepala SMK Negeri 1 Karang Baru
2. Bapak Ahmadi Muslim, MP selaku Kepala Badan Kepegawaian Dan Pengembangan Sumber daya Manusia Kabupaten Aceh Tamiang
3. Bapak Safrizal, S.ST selaku Kepala Bidang Rekayasa Perangkat Lunak SMK Negeri 1 Karang Baru
4. Bapak Ahmadi Muslim, MP selaku Pembimbing di SMK Negeri 1 Karang Baru
5. Bapak Ahmadi Muslim, MP selaku pembimbing di SMKN 1 Karang Baru Aceh Tamiang
6. Bapak/Ibu Guru di SMK Negeri 1 Karang Baru
7. Para pegawai di SMKN 1 Karang Baru

Penulis menyadari bahwa laporan ini masih jauh dari kesempurnaan. Oleh karena itu penulis menghrapkan kritik dan saran yang sifatnya membangun bagi pembaca, penulis berharap semoga laporan ini bermanfaat.

Karang Baru, 20 November 2023

Penyusun,

**Ahmadi Muslim**

# BAB IV BACKUP APLIKASI



## Cara Install Git di Windows

Instalasi Git di Windows memang tidak seperti di Linux yang ketik perintah langsung terinstal.

Kita harus men-download dulu, kemudian melakukan ritual next>next>finish.

Tapi dalam ritual tersebut, ada pilihan yang harus diperhatikan agar perintah git dapat dikenali di CMD.

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

Baiklah, mari kita mulai ritual instalnya. Silahkan klik 2x file instaler Git yang sudah diunduh.



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



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



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



Selanjutnya pemlilihan direktori start menu, klik Next >.



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



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



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



Selanjutnya pemilihan opsi ekstra. Klik saja Next >.



Selanjutnya pemilihan opsi ekspreimental, langsung saja klik Install untuk memaulai instalasi.



Tunggu beberapa saat, instalasi sedang dilakukan.



Setelah selesai, kita bisa langsung klik Finish.

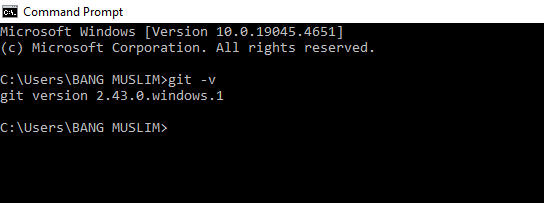


## Setting Dan Konfigurasi Github

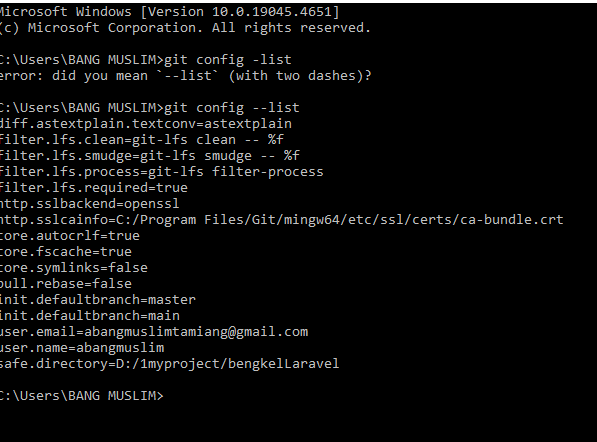
Berikut adalah cara melakukan setingan konfigurasi git local desktop. Setingan ini akan menentukan bagaimana cara git terhubung ke github dan lain sebagainya.

1. **Melihat version dan default branch**

Git –v : untuk melihat versi git, jika belum terinstal maka akan muncul pesan error.

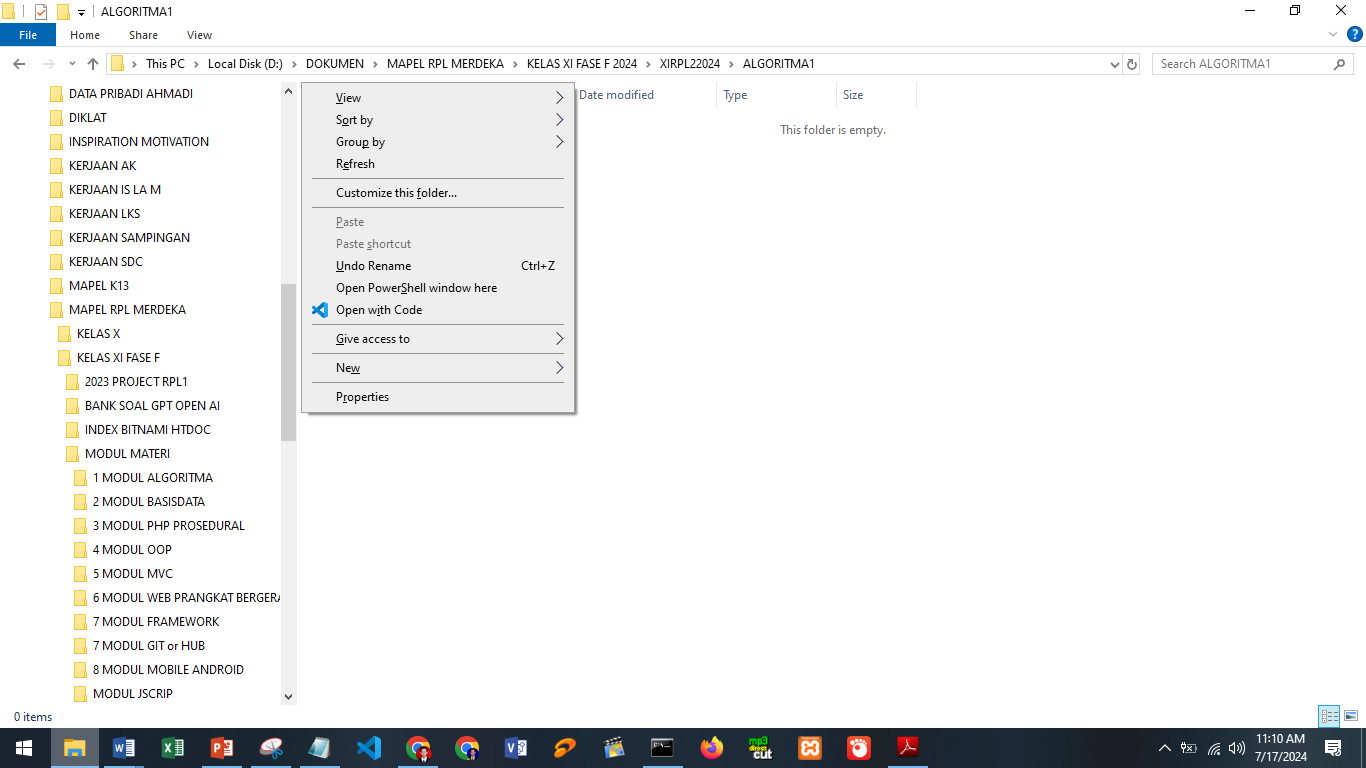


Git config –list : untuk melihat daftar konfigurasi setingan yang sudah dilakukan.

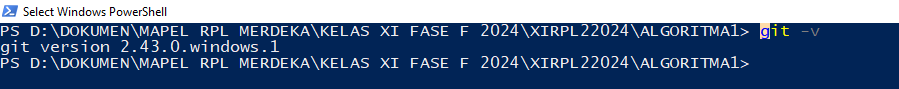


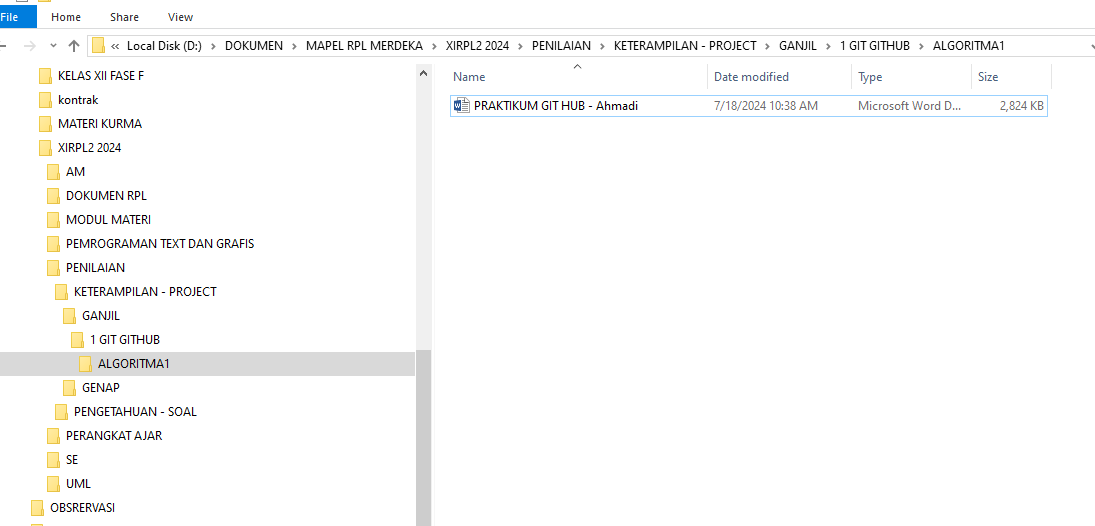
1. **Merubah direktori git**

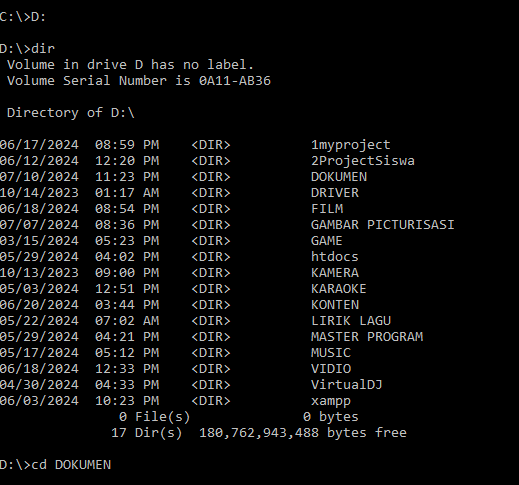
* Tekan windows + Shift + F10

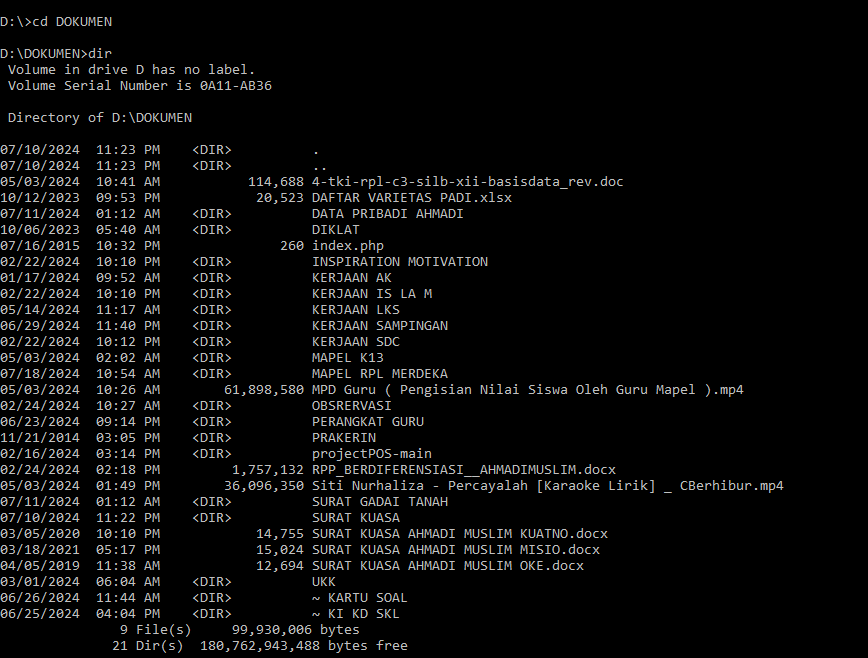


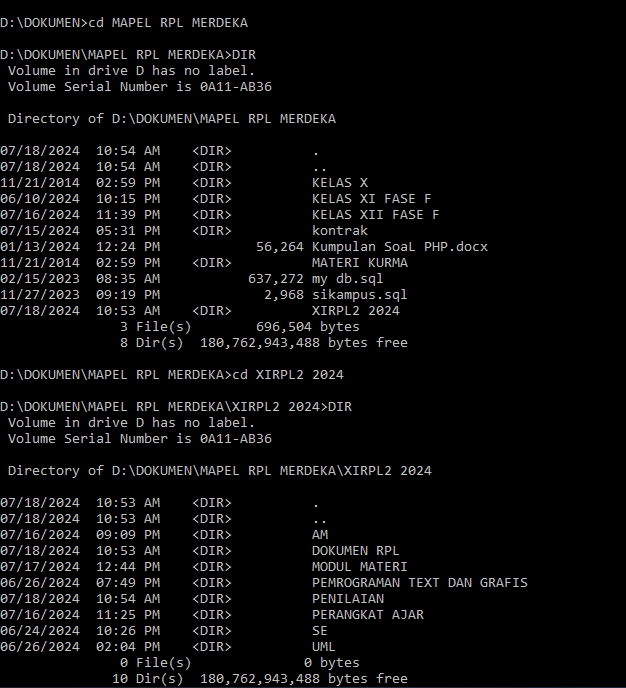
* Klik open powershell

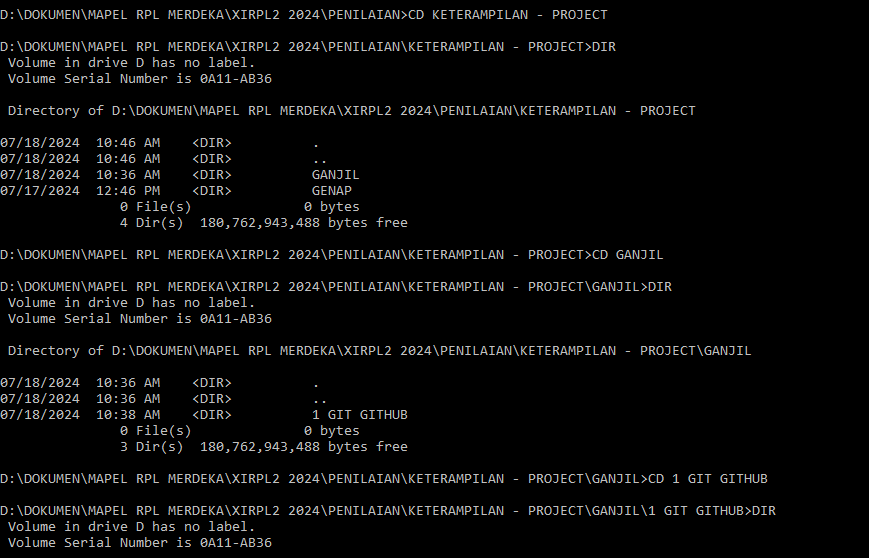


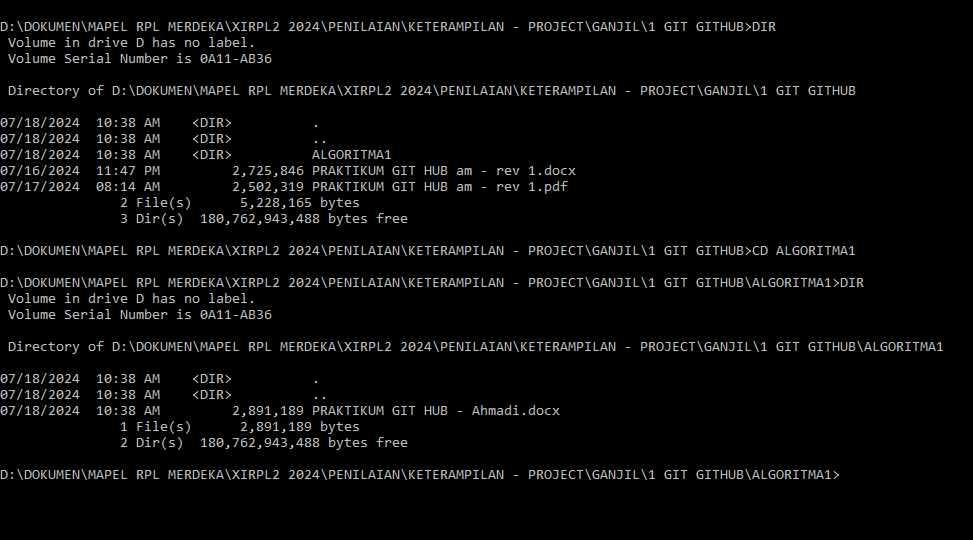




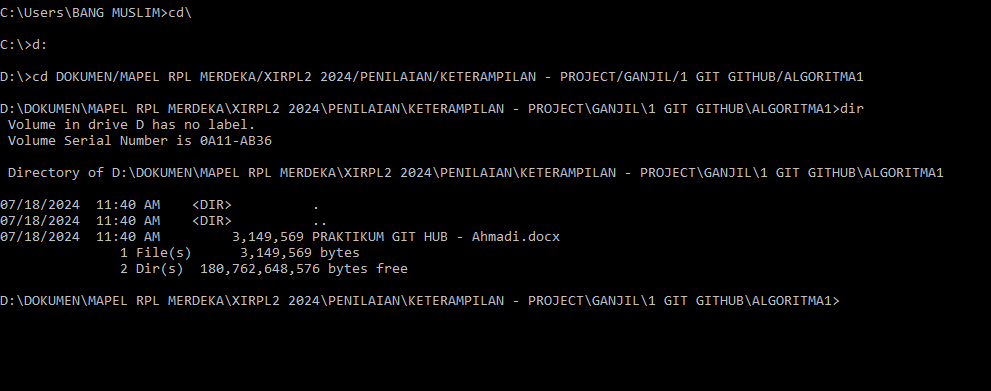








**CARA SIMPLENYA**



1. **Melakukan setting Configurasi Global**

Lakukan setingan configurasi global jika belum ada config ke usesrname di github. Username dan email yang digunakan harus benar benar yang sudah terdaftar.

**Git config --global user.email “email\_user”**:

**Git config --global user.name “nama\_user”**:

**Git config --global user.email “abangmuslimtamiang@gmail.com”**:

**Git config --global user.name “abangmuslim”:**



1. **Melakukan setting Default Branch Main di Configurasi Global**

Lakukan setingan default branch main agar branch yang di update adalah branch main.

git config --global init.defaultBranch main



## Bekerja dengan Git dan Github

Berikut adalah cara bekerja dengan git local desktop dan sekaligus mengupdate/singkronisasi di github.

1. **Git Init**

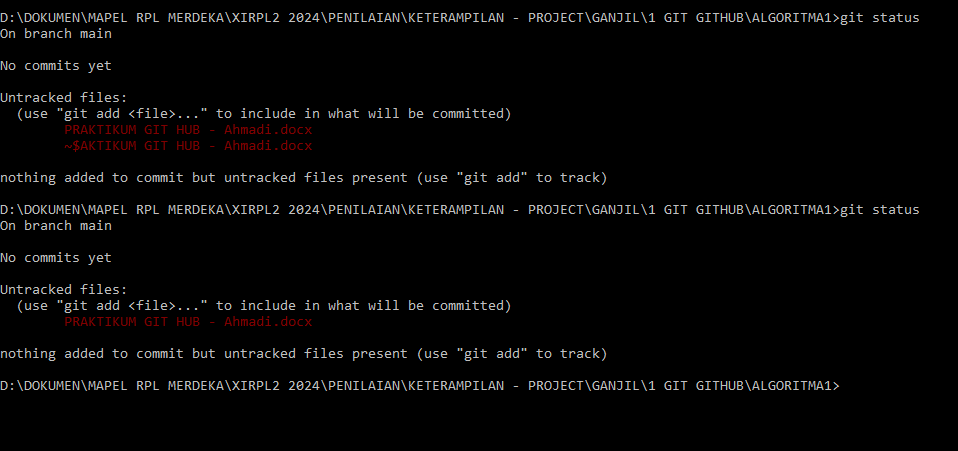
Lakukan git init . untuk menginisialisasi project yang akan di update.

git init.

1. **Git status**

Lakukan git status . untuk melihat status project yang akan di update.

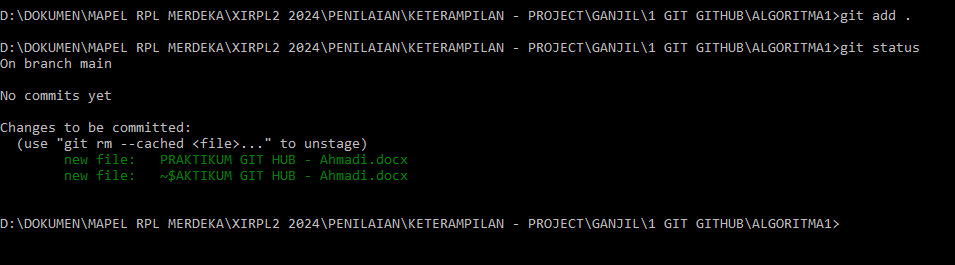
git status.



1. **Git Add**

Melakukan git add . untuk melakukan penambahan project yang akan di update.

git add.

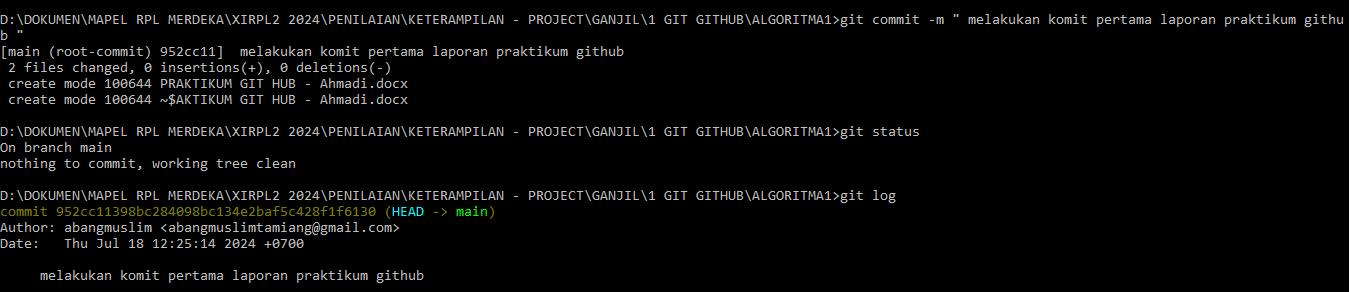


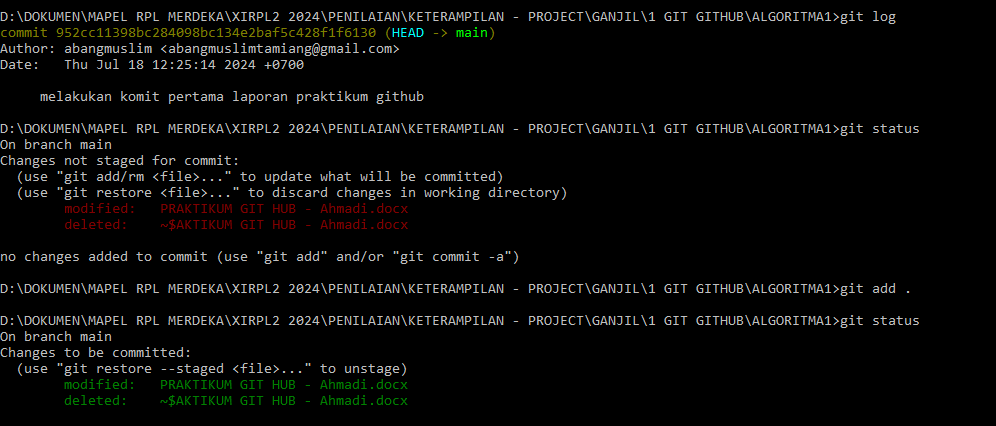
1. **Git Commit**

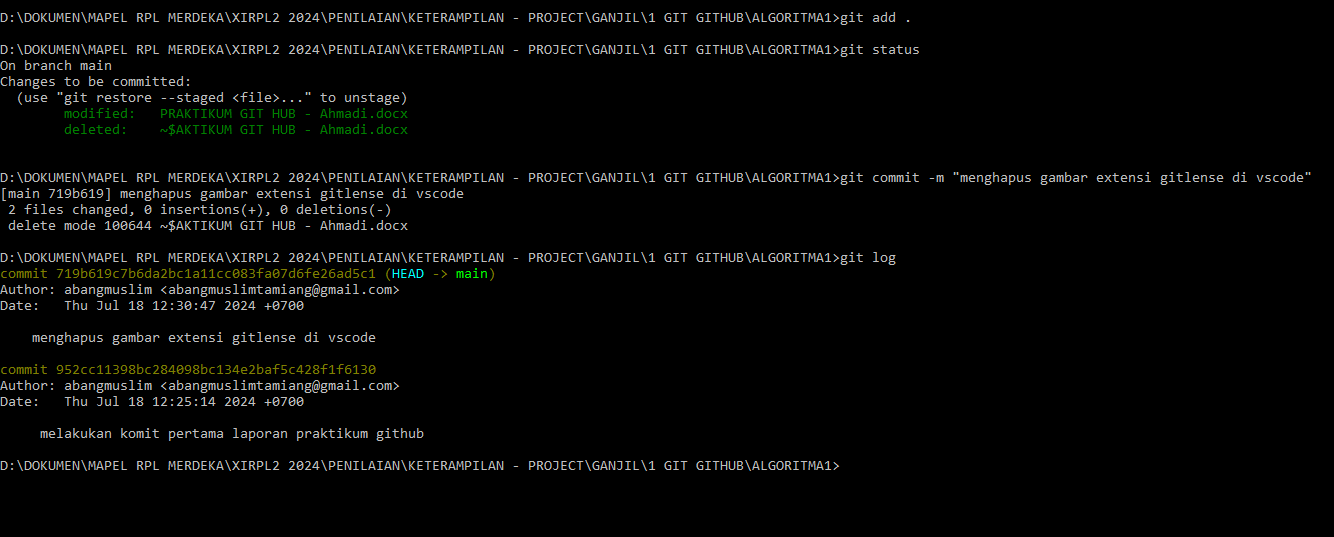
Melakukan git commit . untuk melakukan commit project yang akan di update.

git commit –m “ komentar atau keterangan mengenai update perubahan yang dilakukan di project ”

git commit –m “ melakukan komit pertama laporan praktikum github ”

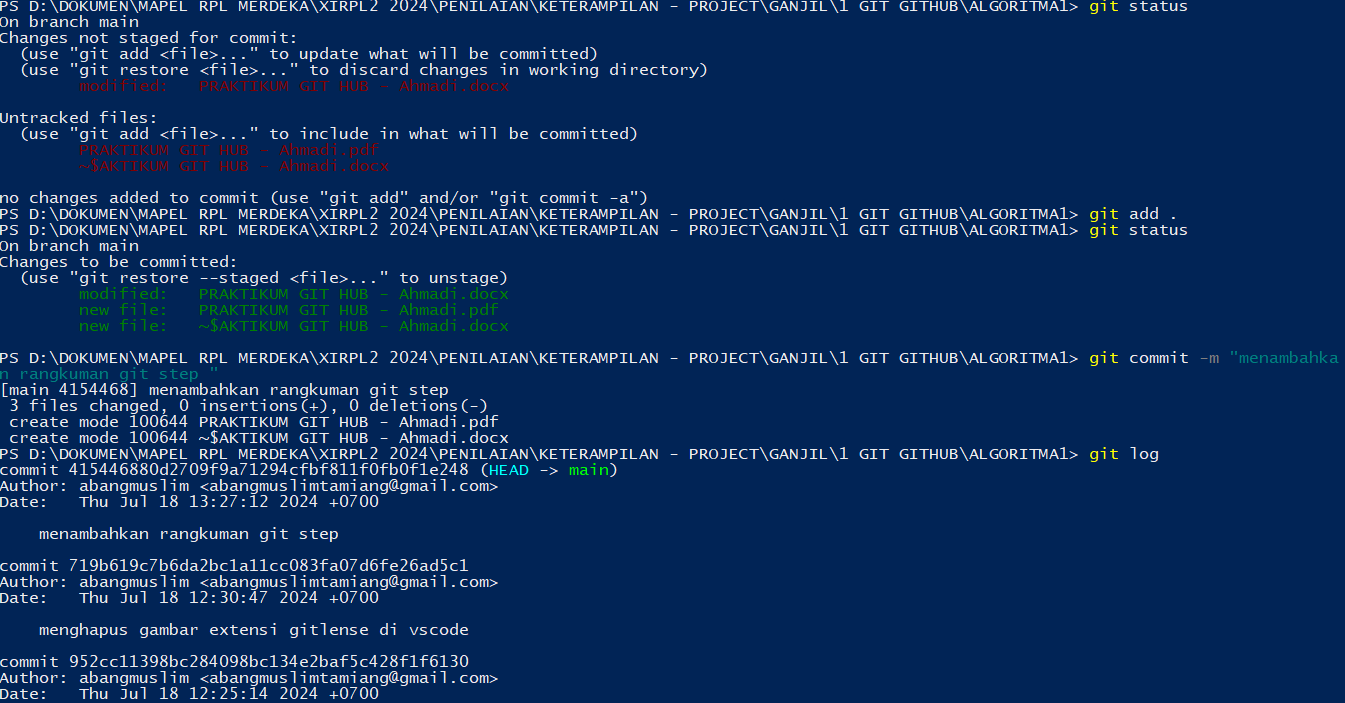


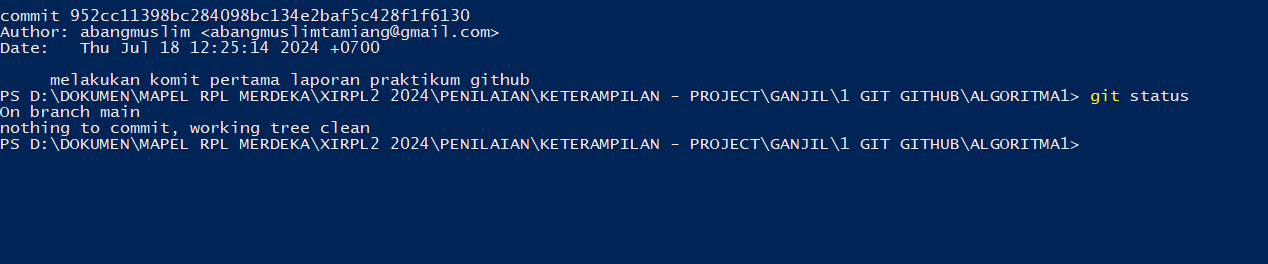




Agar lebih efektif dan effisien kita dapat menggunakan git lense. Dan Untuk melakukan commit Jangan lupa memberikan nama pada commit yang kita buat agar perubahan (check point) menjadi lebih dapat dimengerti semua orang, terutama kita sendiri.

**Melakukan commit berulang (minimal 3)**





1. **Git push**

Melakukan git push . untuk melakukan update project kita ke github.

git push.

Agar lebih efektif dan effisien kita dapat menggunakan extensi git lens yang sudah terinstal di vs code.

1. **Git log**

Melakukan git log . untuk melihat/check update file project kita ke github.

git log.

### Agar lebih efektif dan effisien kita dapat menggunakan extensi git lens yang sudah terinstal di vs code.

### Namun kita bisa juga langsung mengecek ke repository kita di github. Yaitu dengan mengklik bagian commit.

Dengan menggunakan git log di comand prompt dapat pula ita lihat riwayat commit yang telah kita lakukan



### Di menu commit pada github kita dapat juga melihat riwayat dan jumlah commit yang pernah kita buat, lengkap dengan tanggal dan keterangan dari penamaan commit tersebut.



RANGKUMAN:

1. Git init .
2. Git status
3. Git add .
4. Git status
5. Git commit –m “keterangannya”
6. Git log
7. Ulang lagi dari git status..
8. **Menghapus Konfigurasi global remote**

### Menghapus configurasi username

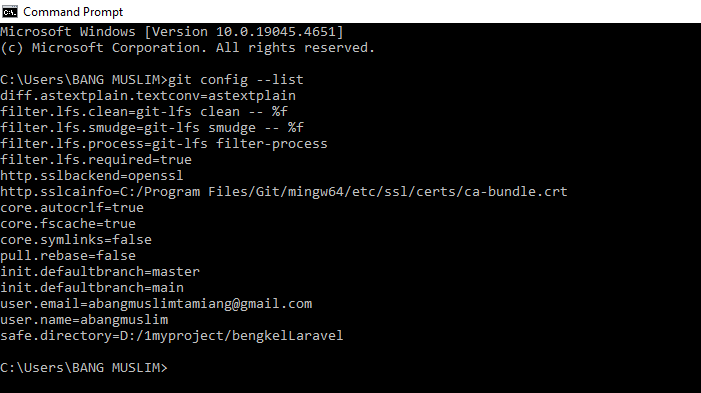
### git config --global --replace-all user.name "FName LName"

git config --global --replace-all user.name "abangmuslim"

### Menghapus configurasi email

### git config --global --replace-all user.email [email-id@domain.com](mailto:email-id@domain.com)

git config --global --replace-all user.email "abangmuslimtamiang@gmail.com"



### Menghapus configurasi remote origin

git remote remove origin

git remote set-url origin NEW\_URL

git remote set-url origin <https://github.com/abangmuslim/bengkelLaravel.git>

### paksa singkron ke master origin

git push origin master --force

1. **Merubah Repository dari Local Remote Global Config**

### Membuat repository baru, kemudian mengikuti comand/prompt yang tersedia



Hapus dulu git remote origin dengan:

git remote remove origin

### …or create a new repository on the command line

echo "# bengkelLaravel" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/abangmuslim/bengkelLaravel.git

git push -u origin main

### …or push an existing repository from the command line

git remote add origin https://github.com/abangmuslim/bengkelLaravel.git

git branch -M main

git push -u origin main

### Melihat remote origin main branch pada setingan project



GIT REMOTE REMOVE ORIGIN

### Melakukan singkronisasi pada menu **Repository** di bagian **Source Control** project

### Perubahan repository project dapat dilihat pada github



## Git Ignore

Git ignore adalah file yang ditinggalkan saat mengupdate/singkronisasi di github. Artinya file ini tidak akan ikut terupdate ke github.

1. **Menambahkan File di Git Ignore**

### Membuat repository baru, kemudian mengikuti comand/prompt yang tersedia



1. **Menghapus File di Git Ignore**

### Membuat repository baru, kemudian mengikuti comand/prompt yang tersedia



1. **Clonning file dari github**

### Membuat repository baru, kemudian mengikuti comand/prompt yang tersedia

### Cara Mengkloning Repositori Github

### Dari Repositori Github, klik Klon

### Salin URL klon

### Di Terminal (Mac) atau baris perintah (Windows git bash), pindah ke direktori lokal

### Gunakan perintah git clone bersama dengan URL yang disalin



git clone https://github.com/abangmuslim/bengkelLaravel.git







# Git Fork dan Git Clone: ​​Perbedaan dan mengapa itu penting!

[#Bahasa Indonesia: github](https://dev.to/t/github)[#pemula](https://dev.to/t/beginners)[#Mengetik](https://dev.to/t/git)[#sumber terbuka](https://dev.to/t/opensource)

" **Fork** " dan " **Clone** " adalah dua istilah yang umum digunakan dalam sistem kontrol versi seperti Git untuk membuat salinan repositori. Sayangnya, setiap hari, saya melihat orang-orang mencampuradukkan keduanya.

Meskipun keduanya memiliki beberapa kesamaan, keduanya bukanlah hal yang sama dan memiliki tujuan yang berbeda. Tujuan artikel ini adalah untuk memperjelas perbedaan di antara keduanya karena hal ini penting untuk kolaborasi dan manajemen Repositori yang efektif.

## Git Fork: Apa fungsinya?

Saat Anda melakukan forking pada Repo, Anda membuat salinan independennya pada platform hosting seperti GitHub. Repo yang di-forking dapat dimodifikasi secara independen dari aslinya.

Inilah sebabnya mengapa Forking umum dilakukan dalam Pengembangan Open-Source. Forking memungkinkan kontributor yang berminat membuat perubahan pada proyek tanpa memengaruhi basis kode asli secara langsung.

Akan tetapi, perlu diingat bahwa repositori bercabang tetap memelihara koneksi ke basis kode asli untuk menarik pembaruan, tetapi ini opsional.

### Kapan Harus Mem-fork Proyek:

* **Berkontribusi pada Proyek Sumber Terbuka:**

Jika Anda ingin berkontribusi pada proyek sumber terbuka yang dihosting di GitHub, forking memungkinkan Anda membuat salinan proyek Anda sendiri. Anda dapat membuat perubahan, bereksperimen, dan mengusulkan perubahan tersebut kembali ke proyek asli dengan mengirimkan Pull [Request.](https://docs.github.com/en/pull-requests)

* **Mempertahankan Versi Pribadi:**

Fork repositori untuk membuat versi Anda sendiri dari proyek tertentu yang ingin Anda modifikasi secara ekstensif atau sesuaikan dengan kebutuhan spesifik Anda. Ini memungkinkan Anda memiliki basis kode independen yang dapat Anda kelola secara terpisah dari proyek asli.

* **Membangun Proyek Turunan:**

Forking berguna saat Anda ingin membangun proyek baru berdasarkan proyek yang sudah ada. Dengan forking, Anda dapat memulai dengan basis kode yang sudah ada dan memodifikasinya untuk membuat proyek baru sambil tetap mempertahankan koneksi ke proyek asli.

## Git Clone: ​​Apa fungsinya?

Kloning adalah proses pembuatan salinan lokal dari repositori di komputer Anda. Saat Anda mengkloning repositori, Anda membuat replika persis dari seluruh repositori, termasuk seluruh riwayat, cabang, dan komitmennya.

Anda dapat membuat perubahan, membuat cabang baru, dan mengirim perubahan Anda kembali ke repositori jarak jauh

Tidak seperti Repositori Bercabang, repositori kloning bersifat independen dari repositori asli, dan perubahan yang dibuat pada kloning tidak akan memengaruhi repositori asli kecuali Anda secara eksplisit menerapkan perubahan tersebut.

### Kapan Anda harus mengkloning:

* **Pengembangan Pribadi:**

Jika Anda ingin bekerja pada repositori secara lokal di komputer Anda tanpa berkontribusi kembali ke proyek asli, Anda dapat mengkloning repositori tersebut.

Hal ini memungkinkan Anda memiliki salinan lengkap riwayat proyek dan mengerjakannya secara mandiri.

* **Kolaborasi dengan Tim:**

Pengklonan biasanya digunakan saat beberapa anggota tim perlu mengerjakan proyek yang sama secara bersamaan. Setiap anggota tim dapat mengkloning repositori ke komputer lokal mereka, membuat perubahan, dan mengirim perubahan tersebut kembali ke repositori jarak jauh agar dapat dilihat dan diterapkan oleh orang lain.

* **Pekerjaan Offline:**

Mengkloning repositori memungkinkan Anda memiliki salinan lokal basis kode, yang memungkinkan Anda mengerjakan proyek bahkan saat tidak ada koneksi internet. Anda dapat melakukan perubahan secara lokal dan mengirimkannya ke repositori jarak jauh setelah Anda mendapatkan kembali akses internet.

## Forking dan Cloning memiliki tujuan yang berbeda

Forking biasanya digunakan untuk pengembangan kolaboratif, yang memungkinkan Anda membuat salinan repositori yang independen untuk mengusulkan perubahan kembali ke proyek asli.

Di sisi lain, kloning digunakan untuk membuat salinan lokal repositori di komputer Anda, sehingga Anda dapat bekerja secara mandiri dan membuat perubahan tanpa memengaruhi repositori asli atau berkolaborasi dengan orang lain secara langsung.

1. **Git Update / Git Instal**

### Setelah repository terdownload/tercloning ke pc maka perlu dilakukan beberapa langkah

### Instal dependensi composer

Composer instal

### Instal dependensi nodejs

npm instal

### Update dependensi composer

Composer update

### Membuat key generate

Php artisan key:generate

### Menghadirkan file configurasi .env

Cp .env.example .env



## How To Change The Last Commit Message

To change the most recent commit message that is still on your local Git repository, you can run the Git command below:

git commit --amend -m "Add commit message here"

This changes the latest commit message. The **-m** option lets you write a new commit message without Git opening your default text editor. To see the change, you can run the Git command below:

git log

You can type **:q** to exit.

Furthermore, if the commit you amended is already in a remote repository, then you will need to forcefully push it to the remote. Assuming you are currently on the ***develop*** branch, run the Git command below to do this:

git push origin develop --force

From above, the remote repository (i.e. **origin**) will be updated to reflect the updated commit message.

Git makes the operation of amending the latest commit that already exists in the Git repository possible, but it is not recommended. Please check the FAQ section to understand why amending an already pushed commit is not recommended.

Let’s now see how to change the messages for multiple commits in Git.

## How To Change Multiple Commits Messages

You may also want to change multiple or previous Git commit messages. You can achieve this using **git rebase** interactively.

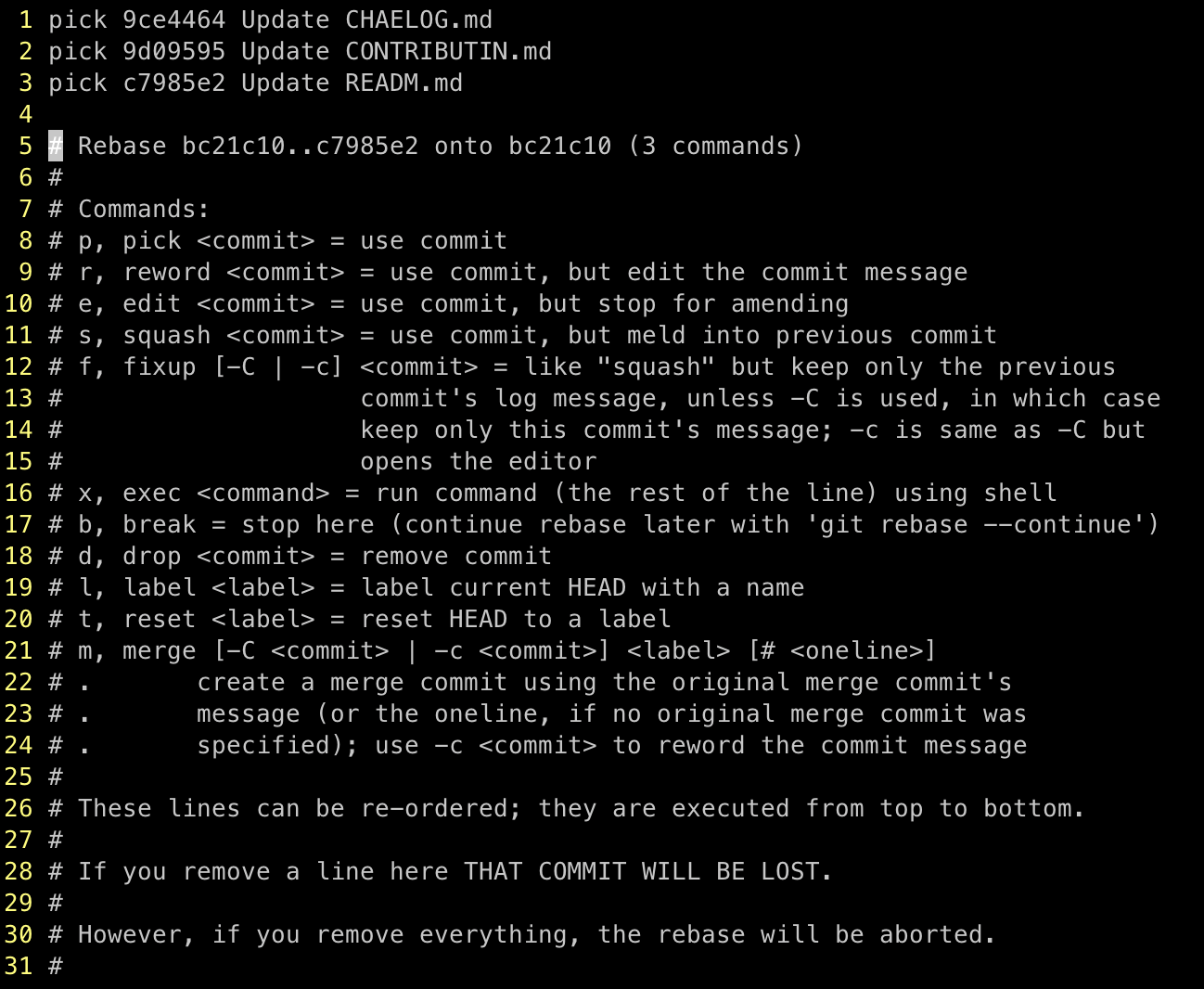
The **git rebase** command can be run in an interactive mode, which gives you the opportunity to change commits. It’s a powerful Git tool for rewriting history.

To change the commit messages of the last 3 commits, for example, run the following command:

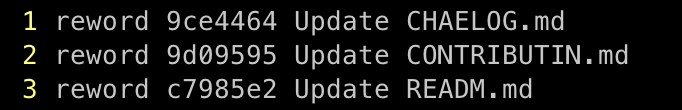
git rebase -i HEAD~3

From above, the option **-i** means interactive. **HEAD~3** is a relative commit reference, which means the last 3 commits from the current branch HEAD.

The **git** **rebase** command above will open your default text editor (Vim for example) in an interactive mode showing the last 3 commits like below:

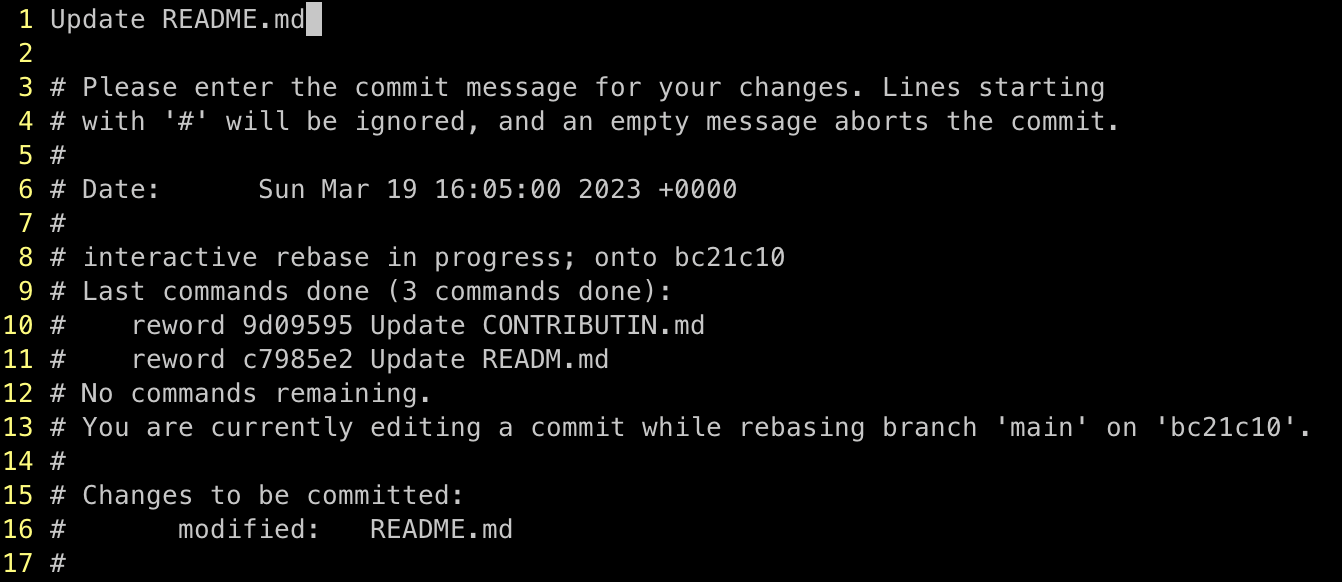
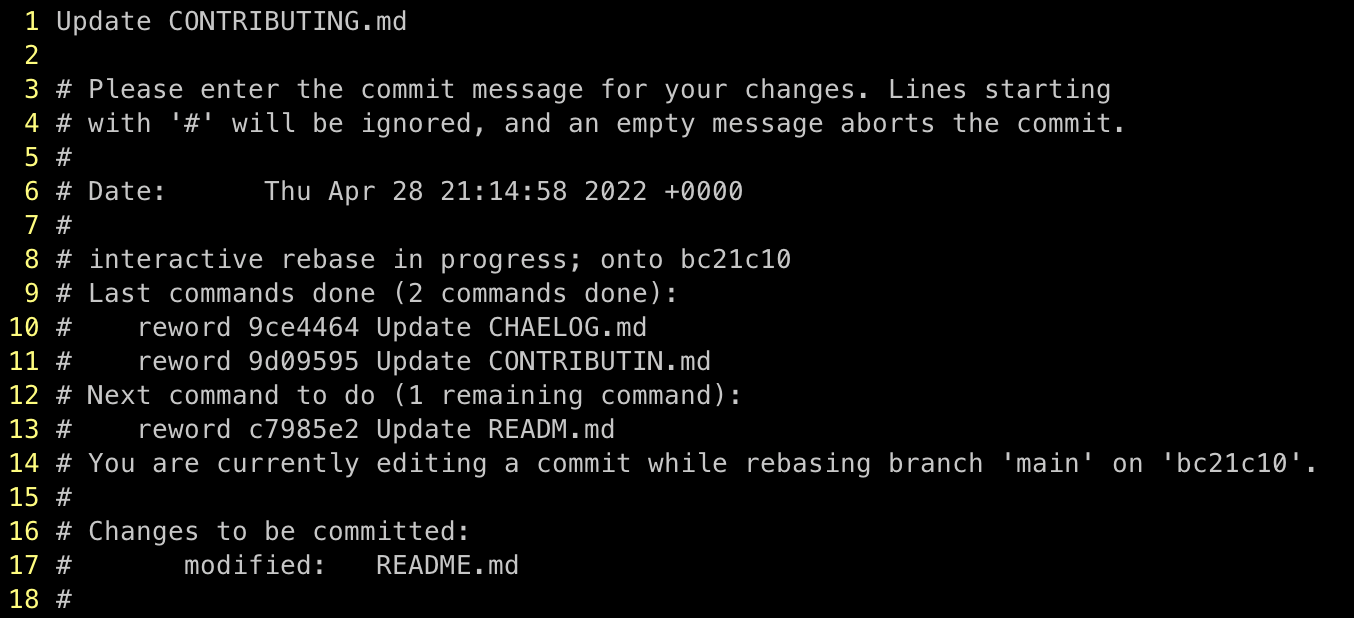
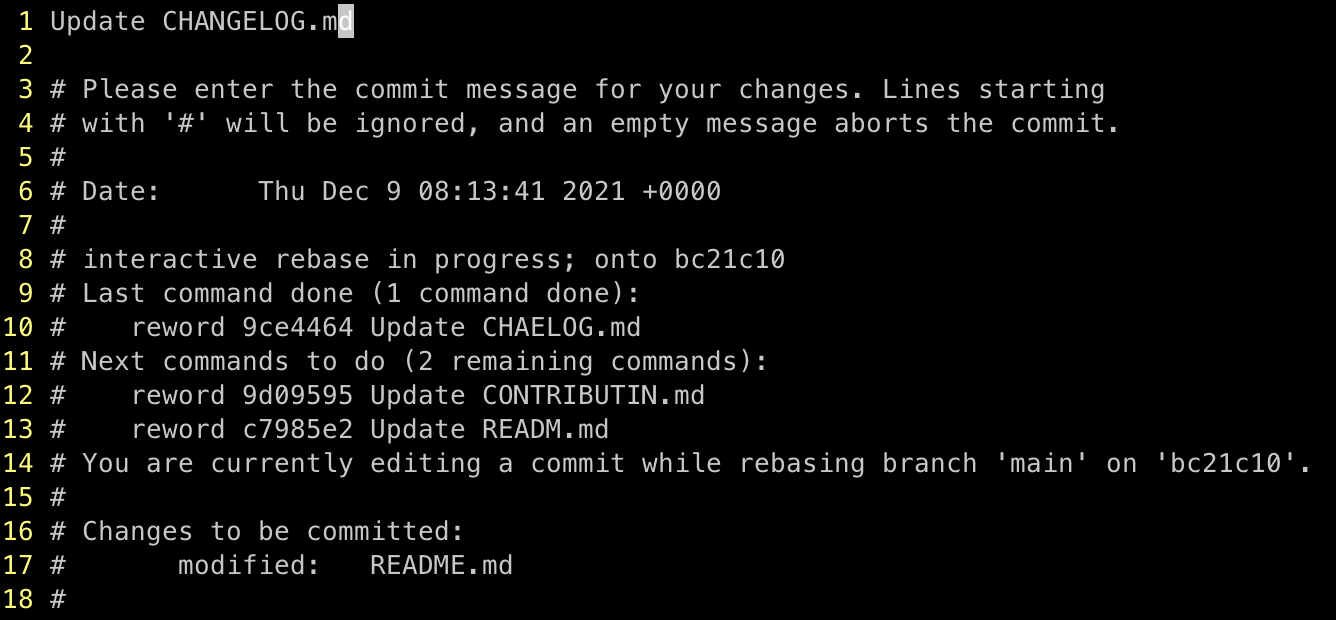


From above, you can see the several commands you can use during a Git rebase session. However, our focus is on the **reword** command highlighted on line 9, which is used to edit commit messages during a rebase operation. Lines 1 to 3 show the last 3 commits, which have some typos in their commit messages. For the commit messages you want to edit, change the word **pick** to **reword** by pressing **i** to go into the insert mode as seen below:



Then, press **esc** to leave the insert mode, followed by **:wq** to save and quit.

Next, Git opens up your text editor again for each commit for you to update the commit message:



You can see the updated commit messages when you run the **git log** command.

As seen in the previous section, if you change the message of a commit that’s already in the remote repository, you will need to forcefully push it to the remote using **git push origin <current\_ branch>** **--force**

## FAQ

Below is a frequently asked question about editing commit messages in Git.

### **Is it recommended to edit the message of a commit that is already pushed to a remote repository?**

It is highly discouraged because it can cause code conflicts for other team members. Anytime you change a commit message, the commit hash is replaced, which makes these technically new commits. Hence, changing commits that are already in the remote repository is not recommended.

**Menghapus commit**

Menghapus commit dalam Git adalah tindakan yang mungkin perlu dilakukan dalam pengembangan perangkat lunak. Dengan menggunakan perintah-perintah khusus seperti git revert, git reset, dan parameter-parameter tertentu, Kamu dapat membersihkan history repository Git dengan mudah. Sooo.... Mari kita bahas cara menghapus commit dengan praktis dan panduan langkah demi langkah. Simak dengan seksama ya!

### Dengan Perintah Git Log

Sebelum kita masuk ke proses menghapus commit, penting untuk memahami history commit di dalam repositor Kamu. Dengan perintah git log, Kamu dapat melihat daftar commit secara kronologis dengan informasi yang relevan seperti ID commit, penulis, dan pesan commit.

Jika Kalian periksa git log dan menemukan commit yang ingin dihapus, langkah pertama adalah mencatat ID commit tersebut. Hal ini memastikan bahwa Kamu tahu commit mana yang akan dihapus.

**Baca Juga:** [Full stack Developer Roadmap, Sekarang memiliki Alur Belajar Baru! Buruan, jangan sampai ketinggalan!](https://www.codepolitan.com/blog/full-stack-developer-roadmap-alur-belajar-baru/)

#### Git Revert: Membatalkan Commit Tanpa Menghapus History

Salah satu cara untuk menghapus commit adalah dengan menggunakan perintah git revert. Perintah ini membuat commit baru yang membatalkan perubahan yang terjadi pada commit sebelumnya. Ini merupakan pilihan yang lebih aman karena tidak menghapus history commit, melainkan menambahkan commit baru yang membatalkan perubahan.

Pertama, kita lihat dulu history commit dengan perintah git log/ git log --oneline.

$ git log --oneline

8caas8c (HEAD -> main) add javascript

9df889s add paragraph

op09821 adding h1

7dcdd7e adding css

4cc653w layout dasar

Misalnya kamu ingin membatalkan modifikasi "add paragraph" pada commit id 9df889s. Maka ketikan pada terminal:

$git revert 9df889s

Setelah perintah telah di eksekusi oleh git, akan muncul COMMIT\_EDITMSG pada [*text editor*](https://en.wikipedia.org/wiki/Text_editor) default yang di setting pada waktu instalasi.

output yang dihasilkan akan seperti ini, kamu perlu melakukan git log/ git log --oneline:

$ git log --oneline

5da5ass (HEAD -> main) Revert "add paragraph"

8caas8c add javascript

9df889s add paragraph

op09821 adding h1

7dcdd7e adding css

4cc653w layout dasar

Bisa dilihat, commit baru dibuat (commit 5da5ass) untuk membatalkan add paragraph pada commit id 3cc339d. Perubahan dengan menggunakan git revert, tidak akan mempengaruhi commit id lainnya.

#### Git Reset: Membersihkan History dengan Hati-hati

Jika Kamu ingin membersihkan history commit secara lebih drastis, git reset bisa menjadi pilihan. Perintah ini memungkinkan Kamu untuk mengatur posisi HEAD dan branch ke commit tertentu. Tersedia beberapa parameter reset, yaitu --soft, --mixed, dan --hard. Namun perlu kehati-hatian dalam melakukan f=git reset. Karena bisa menghapus commit secara permanent.

Untuk keperluan pembelajaran, kita bisa menambahkan file index.html dan app.js

$ git add index.html

$ git commit -m "add menu bar"

[main 781ae23] add menu bar

1 file changed, 7 insertions(+)

git add app.js

PS F:\Project\gitWeb> git commit -m "add cont b"

[main aafb485] add cont b

1 file changed, 2 insertions(+), 1 deletion(-)

Kemudian, bisa kita cek commit history

$ git log --oneline

aafb485 (HEAD -> main) add cont b

781ae23 add menu bar

5da5ass Revert "add paragraph"

8caas8c add javascript

9df889s add paragraph

op09821 adding h1

7dcdd7e adding css

4cc653w layout dasar

###### parameter --soft: Kembali ke ID Commit dengan Perubahan Tersimpan

Jika kamu ingin kembali ke commit tertentu tanpa menghapus perubahan di working directory, gunakan parameter --soft.

Pertama kita log dahulu history commit

$ git log --oneline

5da5ass (HEAD -> main) Revert "add paragraph"

8caas8c add javascript

9df889s add paragraph

op09821 adding h1

7dcdd7e adding css

4cc653w layout dasar

Contoh kita akan hapus 5da5ass dan 8caas8c, maka gunakan perintah berikut

Kemudian lakukan perintah seperti di bawah ini:

$ git reset --soft 9df889s

Jika kita cek log history:

$ git log --oneline

9df889s (HEAD -> main) add paragraph

op09821 adding h1

7dcdd7e adding css

4cc653w layout dasar

Commit sudah dihapus dari history dan perubahan tidak dilakukan pada file. Jika kita periksa status:

$ git status

On branch main

Changes to be committed:

(use "git restore --staged ..." to unstage)

new file: app.js

modified: index.html

File bersangkutan masih dalam fase staged. (pada parameter mixed file dipindahkan ke fase unstaged).

Jika kita ingin menghapus secara permanen, maka langkah selanjutnya adalah menjalan perintah git reset dan git checkout.

$ git reset .

$ git checkout .

**Baca Juga:** [13+ Rahasia Dibalik Skills Frontend Developer di Buru Perusahaan](https://www.codepolitan.com/blog/3-rahasia-dibalik-skills-frontend-developer-di-buru-perusahaan/)

###### parameter --mixed: Kembali ke ID Commit dengan Perubahan Unstaged

Jika kamu ingin menghapus perubahan dari staging area dan kembali ke commit tertentu, gunakan parameter --mixed.

Contoh, kita akan menghapus commit penambahan const b (id aafb485) dan menu bar (id 781ae23), maka perintahnya adalah:

$git reset --mixed 5da5ass

Unstaged changes after reset:

M app.js

M index.html

**Note:** id yang digunakan adalah id terakhir dimana commit akan TETAP disimpan. Pada contoh id 3d53c8b (revert “add paragraph”).

Coba check file yang dimaksud (index.html dan app.js), perhatikan element navigasi dan deklarasi const tidak dihapus. Jika anda cek dengan git status.

$ git status

On branch main

Changes not staged for commit:

(use "git add ..." to update what will be committed)

(use "git restore ..." to discard changes in working directory)

modified: app.js

modified: index.html

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

Untuk menghapus secara permanent, gunakan perintah git checkout.

$ git checkout \*

Updated 2 paths from the index

Jika Anda periksa, maka perubahan sudah dihapus dari file yang bersangkutan.

###### parameter --hard: Kembali ke ID Commit dan Hapus Perubahan

Jika kamu siap untuk menghapus perubahan di working directory dan staging area, gunakan parameter --hard.

Perhatian: harap ekstra hati-hati menggunakan perintah git reset –hard.

$ git log --oneline

9df889s (HEAD -> main) add paragraph

op09821 adding h1

7dcdd7e adding css

4cc653w layout dasar

Contoh, kita ingin menghapus snapshot kembali ke id 4cc653w (layout dasar).

$ git reset --hard 4cc653w

history commit snapshot akan dihapus, semua perubahan pada file akan otomatis dihapus juga.

Git Restore: Mengembalikan File Secara Otomatis

Sama seperti parameter --soft dan --mixed pada git reset, git restore juga dapat membantu Anda mengelola perubahan di working directory. Dengan menggunakan git restore, Anda dapat mengembalikan file ke keadaan sebelumnya tanpa mengubah commit.

git restore layout dasar

###### Unstaged Changes: Perubahan yang Belum Disimpan

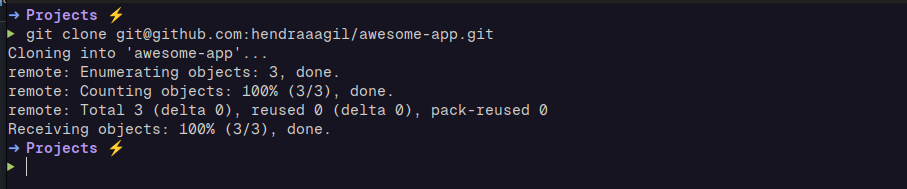
Perhatikan bahwa perintah-perintah di atas akan mengubah status commit, namun perubahan yang belum disimpan (unstaged changes) masih ada di working directory. Pastikan untuk melakukan git add dan git commit lagi jika sudah siap untuk menyimpan perubahan.

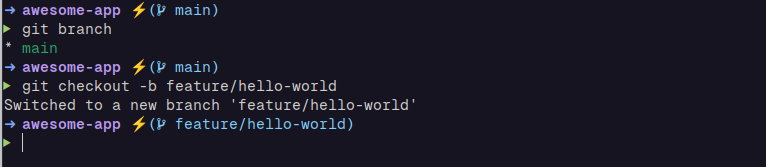
##### Kesimpulan

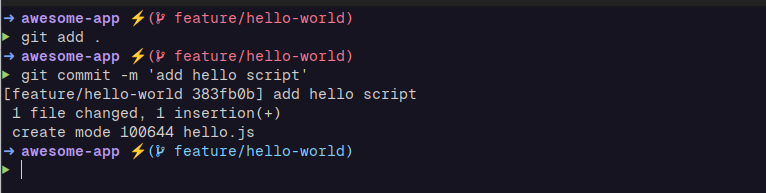
Dalam proses pengembangan perangkat lunak, terkadang kita perlu membersihkan sejarah commit untuk berbagai alasan. Melalui penggunaan perintah git revert, git reset dengan berbagai parameter seperti --soft, --mixed, dan --hard, serta git restore, Anda dapat mengelola commit dengan lebih baik.

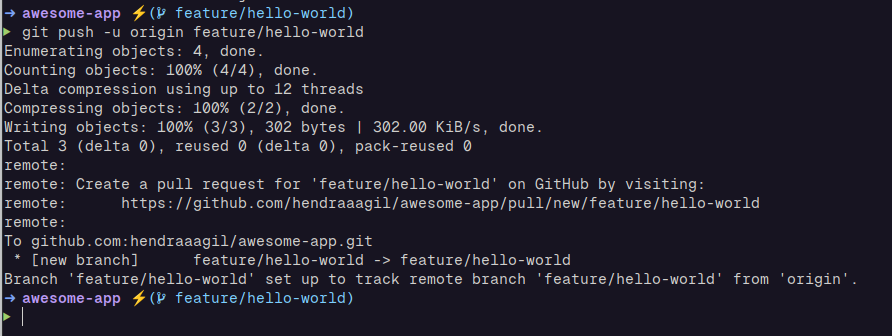
Penting untuk selalu berhati-hati saat menggunakan perintah-perintah ini, karena penggunaan yang tidak benar dapat menyebabkan kehilangan data. Dengan pemahaman yang baik tentang setiap perintah, Anda dapat menghapus commit dengan aman dan efisien. Jangan lupa untuk selalu membuat [*backup*](https://kbbi.web.id/bekap) sebelum melakukan perubahan besar pada repository Anda. Happy coding!

## Berkolaborasi di GitHub

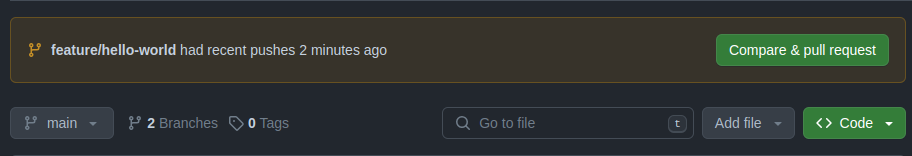
Agar bisa berkolaborasi dengan pengembang lain, tentu yang pertama kita perlukan adalah sebuah repositori di GitHub. Jika sudah membuatnya, kita bisa *cloning*repositori tersebut ke komputer kita.

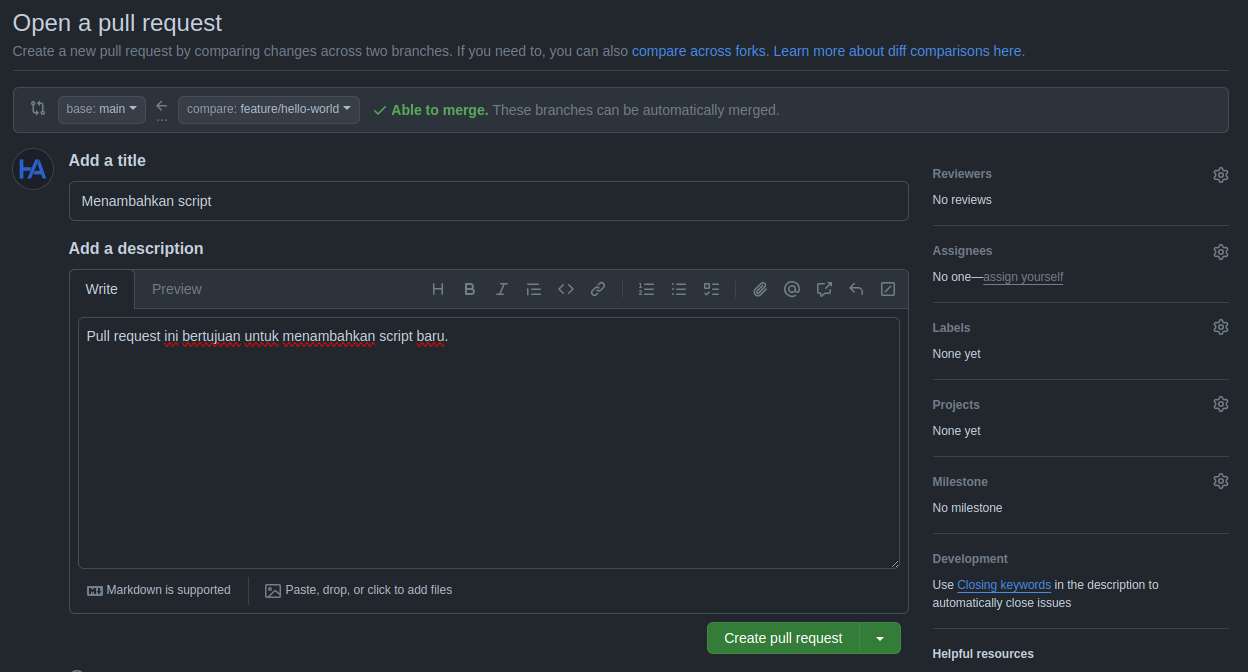
Secara *default*, Ia sudah membuatkan *branch* utama, yaitu branch main. Untuk memudahkan kolaborasi dengan pengembang lain, kita bisa membuat *branch*baru terlebih dahulu.

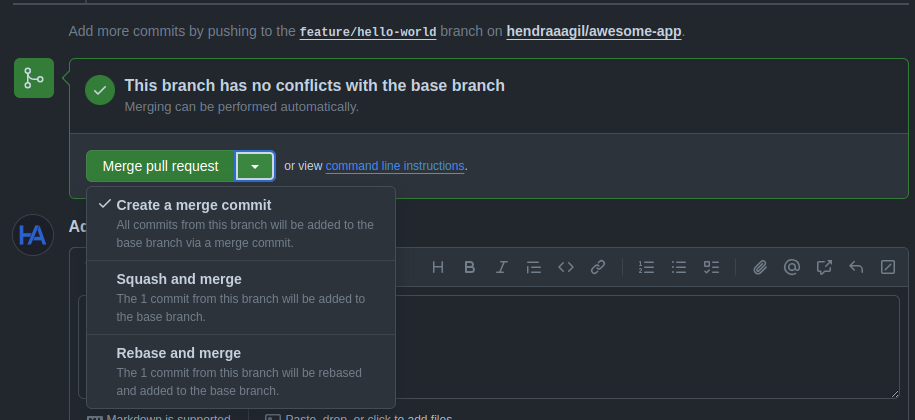
Jika sudah, kita bisa menambahkan kode yang diinginkan dalam *branch* tersebut. Lalu, kita bisa lakukan *commit* untuk *file*yang sudah dibuat.

Jika sudah ter-*commit*, kita bisa *push branch* yang sudah dibuat ke repositori tadi.

## Melakukan *Merge* di GitHub

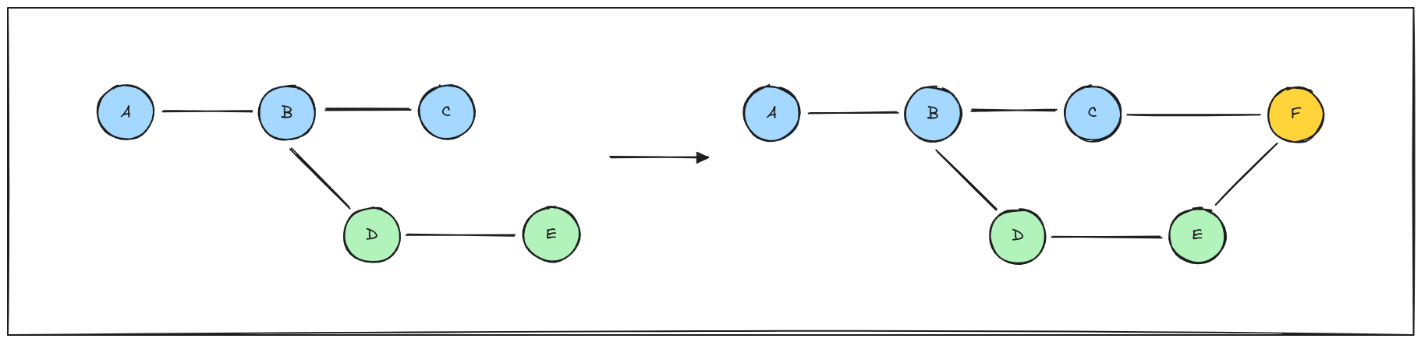
Saat berhasil melakukan *push branch* yang sudah dibuat tadi, pada repositori GitHub kita seharusnya muncul pesan seperti berikut.

Kita bisa langsung mengklik tombol “Compare & pull request” untuk membuat sebuah *pull request* ke *branch* utama, yaitu main. Sebelum membuatnya, kita bisa mengisi judul dan deskripsi yang relevan dengan hal yang kita tambah atau ubah pada kodenya. Jika sudah, kita bisa mengklik tombol “Create pull request”.

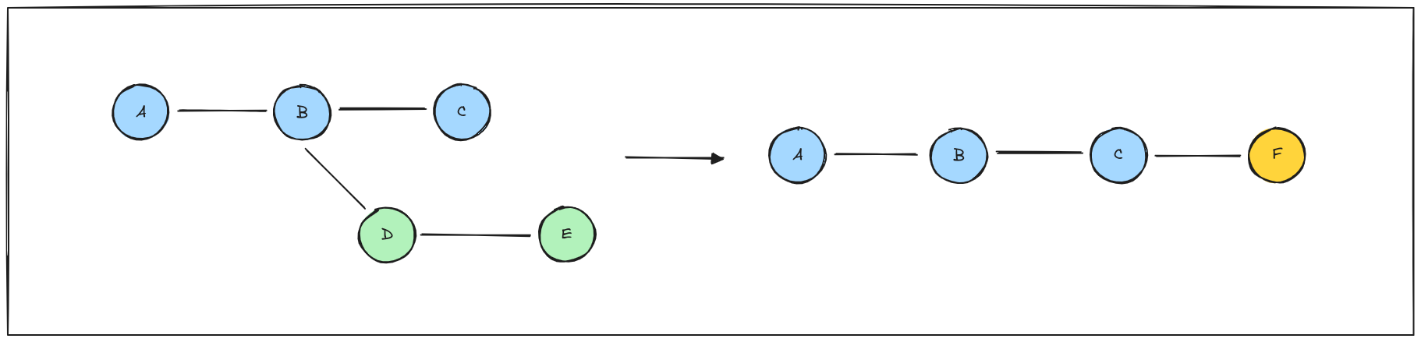
GitHub memberikan beberapa opsi *merge* pada *pull request* yang sudah kita buat.

Di situ, bisa kita lihat ada tiga opsi sebagai berikut.

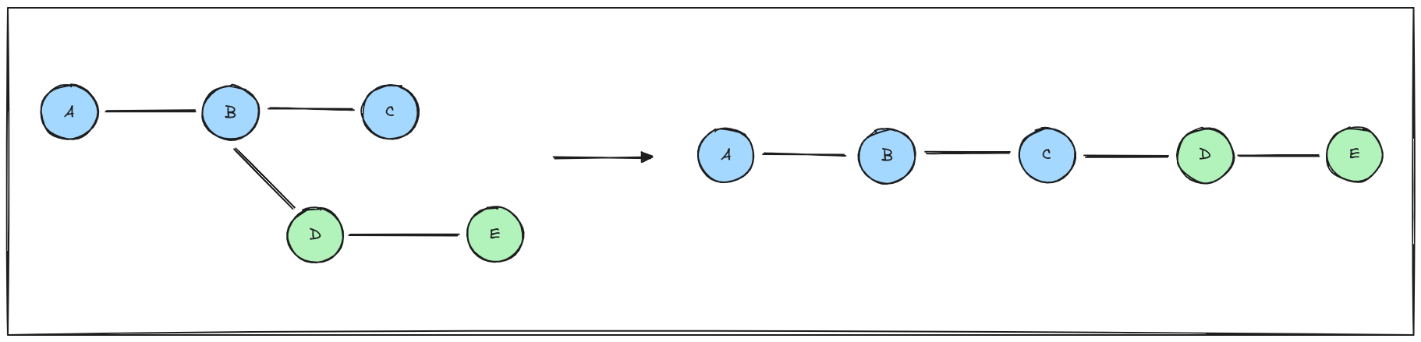
* **Create a merge commit.**

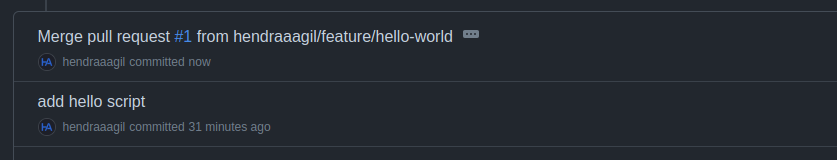
Ini artinya, GitHub akan membuatkan sebuah *merge commit* berisi semua *commit* pada *branch* yang kita berikan ke dalam *branch* utama.

* **Squash and merge.**

Dalam hal ini, GitHub akan menggabungkan semua *commit* pada *branch* yang kita berikan tadi ke dalam satu *commit*. Setelah itu, *commit* tersebut digabungkan ke dalam *branch* utama.

* **Rebase and merge.**

Jika menggunakan ini, GitHub akan menambahkan satu per satu *commit* yang sudah kita buat ke dalam *branch* utama. Jadi tidak ada *commit* tambahan yang dibuat oleh GitHub.

Ketika kita sudah melakukan *merge*, *commit* tadi akan masuk pada *branch* utama.

Jadi, kita bisa berkolaborasi dengan pengembang lain menggunakan fitur *pull request* yang disediakan oleh GitHub. Dalam *pull request* tadi kita juga bisa melakukan review dan memberi komentar ke kode yang sudah dibuat oleh pengembang lainnya. Dengan hal tersebut, kita bisa saling berdiskusi dan memberi saran sebelum menambahkan suatu fitur atau perbaikan pada proyek.

## Tips Mengelola *Merge* di GitHub

Jika skala suatu proyek sudah cukup besar dan banyak pengembang yang terlibat, kita perlu mengelola proses *merge*pada repositori dengan baik. Ada beberapa cara yang bisa kita lakukan sebagai berikut.

* Memberikan *prefix* pada penamaan *branch*.

Ketika ingin membuat suatu *branch* baru, ada baiknya kita juga membuat suatu aturan untuk penamaan *branch* tersebut. Contohnya untuk fitur baru, kita bisa menggunakan *prefix* feature/…, pada perbaikan suatu fitur bisa menggunakan *prefix* bugfix/…, dan sebagainya. Ada baiknya kita diskusikan dengan pengembang lain terkait penamaan *branch* sebelum diimplementasikan pada repositori.

* Menggunakan automasi.

GitHub menyediakan automasi yang bisa kita implementasikan dalam repositori, salah satunya automasi untuk pengecekan pada *pull request* yang dibuat. Kita bisa menggunakan [GitHub Actions](https://github.com/features/actions) untuk melakukan pengecekan otomatis pada skenario tertentu. Misalnya, pengecekan pada format kode, *testing*, dan sebagainya.

## Kesimpulan

Menguasai proses *merge* di GitHub adalah sebuah keterampilan yang cukup penting bagi setiap pengembang yang ingin berkolaborasi secara efektif dalam proyek perangkat lunak.

Dengan memahami langkah-langkah pembuatan *branch*, *commit*, *push*, dan *pull request*, kita dapat memastikan bahwa perubahan kode diintegrasikan dengan lancar pada *branch*utama. Implementasi praktik yang baik, seperti penamaan *branch* yang terstruktur dan penggunaan automasi, dapat membantu menjaga kualitas kode dan efisiensi kerja dalam tim.