**Rangkuman Belajar Git & GitHub**

**Kerja di lokal Git dan Update ke GitHub**

**git command (local) :**

**git add**

**git add .**

**git branch**

**git branch features**

**git branch –merged**

**git branch -d about**

**git checkout**

**git checkout -b features**

**git commit -am "msg"**

**git commit -m "msg"**

**git config**

**git clone**

**git clone https://github.com/ariefcu/simple-landing-page.git**

**git fetch**

**git fetch sandhikagalih**

**git init**

**git help**

**git log**

**git merge**

**git merge sandhikagalih/master**

**git push -u origin master**

**git push**

**git pull**

**git rebase**

**git remote**

**git remote add origin https://github.co/ariefcu/ariefcu.githbub.io.git**

**git remote add sandhikagalih https://github.com/asandhikagalih/simple-landing-page**

**git remote -v**

**git status**

[**https://medium.com/@gofrendiasgard/git-perjalanan-menuju-dunia-parallel-c4e74364cf84**](https://medium.com/@gofrendiasgard/git-perjalanan-menuju-dunia-parallel-c4e74364cf84)

**git revert <commit-bermasalah>**

**git reflog (mirip dengan : git log –all –oneline –decorate –graph)**

**Git : #1 Apa Itu Git & GitHub**

**Git : #2 Bekerja Dengan GitHub**

**Git : #3 GitHub Branch**

**Git : #4 GitHub Fork**

**Git : #5 Bekerja Dengan Git**

**Git : #6 Git Branch & Merge**

**Git : #7 Git Merge Conflict**

**Git : #8 Git Remote (Ini Paling Penting !!!)**

**Git : #9 GitHub Pages, Web Hosting Gratis dengan GitHub Pages**

**Git : #10 Multiple Remote**

**Git : #11 Remote Branch**

**Git : #12 .gitignore**

**Git : #13 Rebase**

**Git : #14 Git & Web Hosting**

[**https://medium.com/@gofrendiasgard/git-perjalanan-menuju-dunia-parallel-c4e74364cf84**](https://medium.com/@gofrendiasgard/git-perjalanan-menuju-dunia-parallel-c4e74364cf84)

**Git : #1 Apa Itu Git & GitHub**

**Konsep** : Git, Version Control System (VCS) atau Source Code Management (SCM)

**Defisi** : Sistem yang mengelola perubahan dari sebuat dokumen, program kompute, website dan kumpulan informasi lain.

**Kenapa kita harus memakai VCS :**

1. Bisa melacak versi atau history dari perubahan yang terjadi pada software kita
2. Kolaborasi
3. Sharring, membagikan atau memamerkan program yang sudah kita buat

**Git & GitHub merupakan 2 hal berbeda**, bisa bekerja dengan Git tanpa harus bekerja dengan GitHub, dan sebaliknya bisa bekerja dengan GitHub tanpa harus menginstall Git.

**Version Control System :**

* Sebuah sistem yang menyimpan “rekaman/snapshot” perubahan pada source code
* Memungkinkan bekerja berkolaborasi dengan lebih baik
* Mengetahui siapa yang melakukan dan kapan sebuat perubahan terjadi
* Memungkinkan kita untk kembali ke keadaan sebelum perubahan (checkout)

**Definisi Git :** Sebuah VCS terdistribusi untuk mengelola perubahan file di dalam folder

**Repository/Repo :** Folder pada sebuat aplikasi

Riwayat perubahan file disimpan menggunakan serangkaian **commit**

**Branch :** cabang di Git

**Merge** **:** gabung

**GitHub :** Layanan cloud untuk menyimpan & mengelola project/repo git. Instagramnya para programmer. Seperti layaknya Git, tapi ini online.

**Akan lebih terasa jika kita menggabungkan keduanya, Git & GitHub.**

**Kita bisa mengirimkan Source Code kita atau Project kita ke Git Hub (Push) atau sebaliknya kita bisa mengambil Source Code kita atau Project kita dari GitHub (Pull). Yang di Push dan di Pull adalah commit-nya.**

**Syaratnya adalah : GitHub dijadikan Remote. Lalu Clone. Baru bisa lakukan Push dan Pull.**

Akan sangat bermanfaat jika bekerja dengan kolaborasi.

**Git Command/Istilah**

* repo : folder project kita
* commit : rekaman/snapshot dari repo kita
* hash : penanda unik pada sebuah commit
* checkout : berpindah ke sebuah commit
* branch : cabang bebas dari sebuah commit
* merge : menggabungkan branch
* remote : sumber yang memiliki repo
* clone : mengambil repo dari remot
* push : mengirimkan commit ke repo
* pull : mengambil commit dari repo

**Git : #2 Bekerja Dengan GitHub**

Create Account : ariefcu

Email Account : [ariefcu@gmail.com](mailto:ariefcu@gmail.com)

Nama Repo Private : coba-laravel

Nama Repo Public : wpu-resolusi

**Git : #3 GitHub Branch**

Create Account : ariefcu

Email Account : [ariefcu@gmail.com](mailto:ariefcu@gmail.com)

Nama Repo Private : coba-laravel

Nama Repo Public : wpu-resolusi

Checkout untuk pindah cabang

**Pull Request** untuk meminta pemilik dari reponya atau main branchnya untuk menarik data kita

**Merge** untuk menggabungkan 2 cabang

**Merge Conflict** karena 2 baris yang sama dirubah 2 cabang yang berbeda

**Git : #4 GitHub Fork**

**Fork/Forking :**

* Membuat “copy/duplicat” dari repo orang lain (beserta historynya)
* Jembatan antara repo original dan duplikatnya
* Modifikasi terhadap repo original
* Berkontribusi pada repo orang lain
* Fork != Clone
* Karena Clone hanya meduplicat dari GitHub ke komputer lokal kita

**Mencoba fork dari webprogrammingunpas dan lakukan pull request.**

**Test kolaborasi.**

**Git : #5 Bekerja Dengan Git**

Install dari[**https://git-scm.com/**](https://git-scm.com/)

**Console**

**Git Client** (GUI – Graphical User Interface)

**Pro Git eBook :**

htpps://git-scm.com/book/en/v2 for english

htpps://git-scm.com/book/id/v2 for bahasa

Banyak yang pakai Git, antara lain : Google, facebook, Microsoft, Twitter dll

Git Command (local) :

* git init : inisialisasi repo git di komputer kita
* git add <file(s)> : buat nambahin file di staging area
* git status : penting banget, status repo kita
* git commit : untuk melakukan commit
* git config : untuk melakukan configurasi ke dalam gitnya
* git branch : untuk bikin branch
* git help : untuk mau tau perintah

3 area pada repo Git

* **Working Tree** : folder lokal tempat kita bekerja, tempat file-file project
* **Staging Area** : ngasih tau ke Git kita melakukan perubahan
* **History**  : setelah di commit masuk ke sini

Staging Area dan History akan masuk ke folder **.git**

**git add** untukmemindahkan dari Working Area ke Staging Area

Atau pakai **git add .** untuk add semua file

**git commit** untuk memindahkan dari Staging Area ke History

**git status** untuk mengecek kondisi sebelum **commit**

**git config** –global user.name “Ariefcu”

**git config** –global user.email [ariefcu@gmail.com](mailto:ariefcu@gmail.com)

**git commit -m** “commit pertama kali ke repo”

-m diatas untuk memberi message

**git log** untuk melihat commit2 yang sudah pernah dilakukan

**git log -3** untuk commit2 3 terakhir

**git log – style.css** untuk melihat perubahan yang terkair file style.css

**git checkout 5digitawaldarihash – style.css** untuk kembali ke posisi tertentu terkait file style.css

kalau ada file yang di staging berarti harus di commit dahulu biar masuk ke history

**Git : #6 Git Branch & Merge**

**pwd : print working directory,** untuk melihat kita lagi di directory mana

**Tarik folder ke git bash** biar pindah direktorinya ke situ, atau

Buka folder yang kita mau repo lalu **klik kanan dengan buka Git Bash disini**

**git init**, maka foldernya akan berubah menjadi reposity

**git commit -am “menambahkan repo”** -a adalah untuk add jika filenya sudah pernah ditambahan

**head** menandakan branch sedang aktif di branch tersebut

**git branch** untuk melihat branch ada apa aja

**cabang** yang warna hijau adalah yang aktif

**git branch about** untuk membuat cabang about

**git branch post** untuk membuat cabang post

**git branch** maka muncul ada 3, about, master dan post, master tulisan hijau dan ada bintangnya

**git log –all –decorate –oneline –graph** untuk melihat visualisasi graphnya

**alias g=”git log –all –decorate –oneline –graph”** untuk membuat shortcut, hanya berlaku 1 sesi

**git checkout <nama\_branch>** untuk pindah headnya ke nama\_branch

**git checkout about** maka head akan pindah ke about

**merge ada 2 jenis :**

* **Fast Forward Merge**, yang direct path
* **Three-way Merge**

1. Menggabungkan master dengan about

**git checkout master**

**git branch**, memastikan pindah ke master

**git merge about**, Fast-forward

**git branch –merged**, untuk mengetahui branch mana yang sudah di merged

**git branch -d about**, untuk mendelete branch about

**git branch -d post**, error: **The Branch ‘post’ is not fully merged**, if you are sure you want to delete it, run **‘git branch -D post**’

1. Menggabungkan master dengan post

merge antara master dan post tidak bisa langsung di merge, karena **tidak direct path**

**git branch** untuk memastikan di master

**git merge post**, merge made by the ‘ort’ strategy

**Git : #7 Git Merge Conflict**

**Karena 2 branch mengerjakan baris yang sama dalam satu repo.**

**Harus manual untuk solusinya.**

**Sama seperti saat merge, hanya saja ada yang harus diselesaikan secara manual untu solusinya, tidak bisa menggunakan auto merge**

**Git : #8 Git Remote (Paling Penting !!!)**

**Kasus #1 Di Komputer Belum Ada, Mau Mengambil Dari GitHub**

**Git Remote bisa ke GitHub, GitLab atau Bucket**

**Remote intinya adalah duplikat dari punya kita**

Kali ini kita akan pakai **GitHub sebagai remote kita**

Buat repository baru di GitHub, misal test-remote-2

Lalu copy alamat HTTPSnya

Masuk ke folder d:\applications

Lalu **git clone** [**https://github.com/ariefcu/test-remote-2.git**](https://github.com/ariefcu/test-remote-2.git)

**Maka sudah terhubung secara remote**

Cd test-remote-2

**git remote**, hasilnya origin

**git remote -v**,

git remote -v

origin https://github.com/ariefcu/test-remote-2.git (fetch)

origin https://github.com/ariefcu/test-remote-2.git (push)

**Git Status**, on branch master, **your branch is up to date with ‘origin/master’**

**alias g=”git log –all –decorate –oneline –graph”** untuk membuat shortcut, hanya berlalu 1 sesi

**graph**

di vscode buat file baru index.html, lalu save

kemudian di git bash ketik **git bash**

lalu **git add .**

lalu **git commit -m “menambah file index.html”**

**git status**,

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

**graph**,

\* 90197dd (HEAD -> main) menambahkan file index.html

\* 12b9e23 (origin/main, origin/HEAD) Initial commit

**git push**

**Masukkan password jika diminta**

Enumerating objects: 4, done.

Counting objects: 100% (4/4), done.

Delta compression using up to 8 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 477 bytes | 477.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/ariefcu/test-remote-2.git

12b9e23..90197dd main -> main

**git status**

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

**graph**

\* 90197dd (HEAD -> main, origin/main, origin/HEAD) menambahkan file index.html

\* 12b9e23 Initial commit

**git config –global user.name** “Ariefcu”

**git config –global user.email** “[ariefcu@gmail.com](mailto:ariefcu@gmail.com)”

**Menit 18:00 Kasus #2 di Lokal sudah ada, di GitHub belum ada, kirim dari Lokal ke GitHub**

Keluar dari repository dengan **CD ..**

**mkdir** test-remote-3

**cd** test-remote-3

**git init**

Initialized empty Git repository in D:/applications/test-remote-3/.git/

Di **vscode** buat file index.html

**ls** untuk melihat file yang ada

**git status**

**git add .**

**git commit -m “menambahkan file index.html”**

**graph**

**clear**

Di **vscode** update file index.html

**git status**

**git commit -am “mengupdate file index.html”**

**graph**

\* 0b3bbea (HEAD -> master) mengupdate file index.html

\* 58e2474 menambahkan file index.html

**Ceritanya mau manambahkan repository ini ke account GitHub kita**

Di **GitHub create repository baru** dengan **nama yang sama test-remote-3**

**Initialize this repository with a README jangan dicentang**

**git remote**, masih kosong

**Copy dari lokal ke GitHub :**

**git remote add origin https://github.com/ariefcu/test-remote-3.git**

**git remote**

origin

**git remote -v**

origin https://github.com/ariefcu/test-remote-3.git (fetch)

origin https://github.com/ariefcu/test-remote-3.git (push)

**git push -u origin master**

Enumerating objects: 6, done.

Counting objects: 100% (6/6), done.

Delta compression using up to 8 threads

Compressing objects: 100% (4/4), done.

Writing objects: 100% (6/6), 671 bytes | 671.00 KiB/s, done.

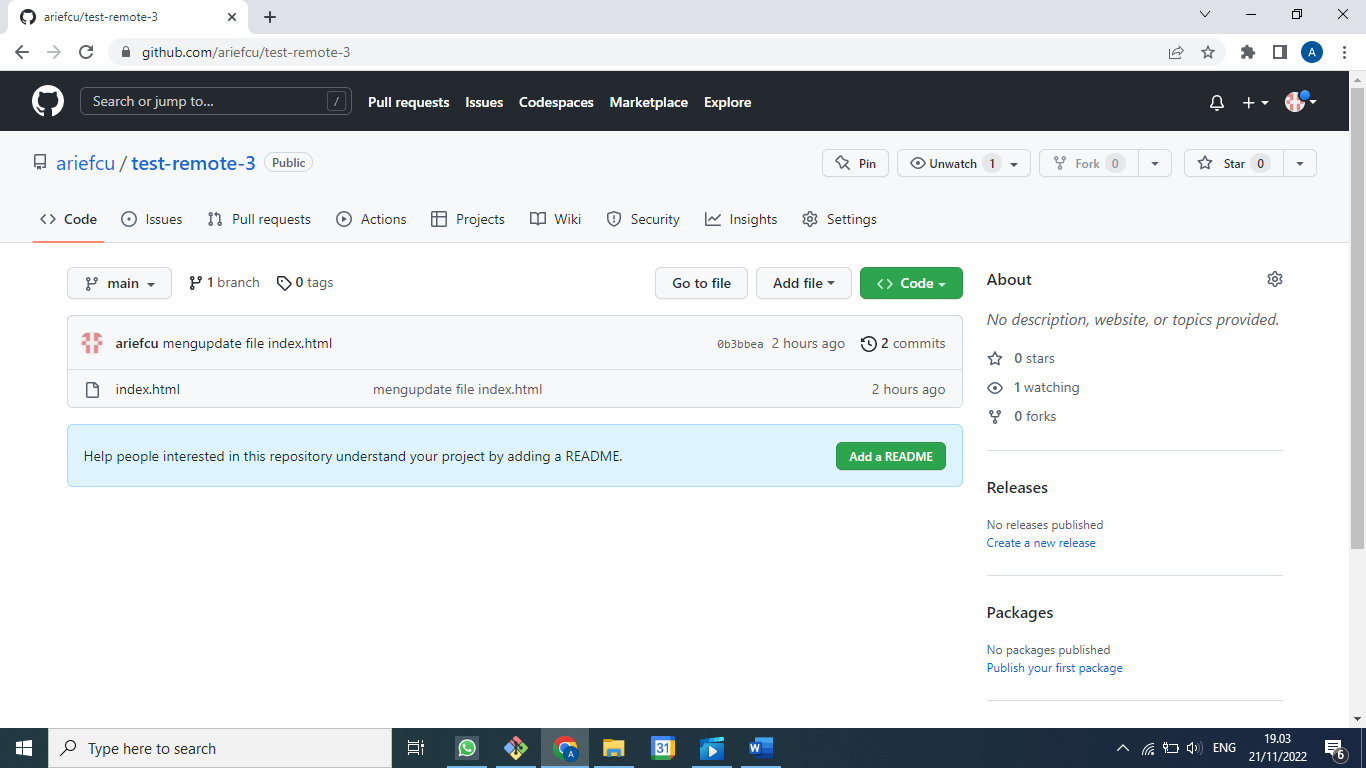
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (1/1), done.

To https://github.com/ariefcu/test-remote-3.git

\* [new branch] main -> main

branch 'main' set up to track 'origin/main'.



**Di vscode rubah index.html**

**Save**

**Lalu di Git Bash**

**graph**

**git Add .**

**git Commit -am “mengupdate index.html lagi”**

**git push**

**git graph**

**Kasus #3 Di Komputer Sudah Ada, Di GitHub Sudah Ada, Tapi Ada Perubahan Di Masing2 File Index.html Di Baris Yang Sama**

**Ubah file index.html di komputer di baris yang sama**

**Ubah file index.html di GitHub di baris yang sama lalu di commit**

**Di Git Bash**

**clear**

**git add .**

**git commit -m “mengubah file index.html”**

**git push**

To https://github.com/ariefcu/test-remote-3.git

! [rejected] main -> main (fetch first)

error: failed to push some refs to 'https://github.com/ariefcu/test-remote-3.git'

hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.

**clear**

**git status**

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

**clear**

**git fetch**, dia akan ngecek repo yang ada di remote sudah sampai mana commitnya, beda ga sama yang kita punya

**git status**

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

**graph**

\* 28cff3a (HEAD -> main) mengubah file index.html

| \* 2a4bdcb (origin/main) Update index.html

|/

\* 4bdd337 mengupdate index.html lagi

\* 0b3bbea mengupdate file index.html

\* 58e2474 menambahkan file index.html

**git pull**

Auto-merging index.html

CONFLICT (content): Merge conflict in index.html

Automatic merge failed; fix conflicts and then commit the result.

**Di vscode :**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Test Git Remote</title>

</head>

<body>

<<<<<<< HEAD

    <h1>Tes Remote Merge Conflict</h1>

=======

    <h1>Ayo Semangat Pelajari Remote Merge Conflict</h1>

>>>>>>> 2a4bdcb33a59f495fd815b8d86350f9e7ec822d4

    <p>Paragraph X</p>

</body>

</html>

Solusinya jadi seperti ini :

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Test Git Remote</title>

</head>

<body>

    <h1>Tes Remote Merge Conflict Ayo Semangat Pelajari Remote Merge Conflict</h1>

    <p>Paragraph X</p>

</body>

</html>

**git status**

**clear**

**git add .**

**git commit -m “tahapan merge conflict”**

**git status**

**git push**

**git status**

\* 9dd0d1e (HEAD -> main, origin/main) tahapan merge conflic

|\

| \* 2a4bdcb Update index.html

\* | 28cff3a mengubah file index.html

|/

\* 4bdd337 mengupdate index.html lagi

\* 0b3bbea mengupdate file index.html

\* 58e2474 menambahkan file index.html

**Git : #9 GitHub Pages, Web Hosting Gratis dengan GitHub Pages**

Membuat salah satu repo menjadi GitHub Pages

Harus punya repo website statis

Github.com/sandhikagalih, website 3 awal bisa digunakan

1. Bostrap v3
2. Boostrap v4, ngobar 4
3. Company Profile, Materilize

Yang pertama disiapkan adalah websitenya dulu

Bikin dulu repo dulu

New repo, namanya ga boleh sembarang :

ariefcu.github.io

di explorer masuk ke folder company profile, git bash here

**git init**

**git remote add origin** [**https://github.co/ariefcu/ariefcu.githbub.io.git**](https://github.co/ariefcu/ariefcu.githbub.io.git)

**git remote**

**clear**

**git add .**

**git commit -m “inisialisasi commit”**

**git status**

**git push -u origin master**

**git status**

di github sudah terupload

di browser : ariefcu.github.io

kalau mau update tinggal di lokal lalu di push

kalau mau .com tinggal arahkan di githubnya

**Kasus #2 Bikin halaman website untuk masing2 repo**

Alamat urlnya tetap di bawah alamat ariefcu.github.io

**cd ..**

**ls**

**cd wpu-landing**

**git init**

**git status**

**git add .**

**git commit -m “inisial”**

**git status**

Di github repo new namanya bebas wpu-landing

Di github

**git remore add origin copyan dari github**

**git push -u origin master**

**git status**

Di github sudah ada website siap tampil, pastikan index.html harus ada

Scroll sampai github pages, di source, branch yang mana yang mau dijadikan default

Pilih yang master branch

Setelah loading balik lagi ke github pages

Your site is readu to be published at <https://ariefcu.github.io/wapu-landing/>

Kalau diklik otomatis website kalian sudah tampil

Jadi website gratisan, bisa diakses oleh siapapun

Kalau untuk ganti ke .com bisa custome domain

Kalau untuk PHP pakai theme

**Git : #10 Multiple Remote**

Kita bisa terhubung ke beberapa remote sekaligus

Kenapa? Mungkin aja kita ingin singkron dengan beberapa remote sekaligus

Fork punya repo orang lain, misal github/sandhikagalih/simple-landing-page

Lalu copy ke lokal

Tentukan dulu di mana mau copynya, d:\applications

**git clone** [**https://github.com/ariefcu/simple-landing-page.git**](https://github.com/ariefcu/simple-landing-page.git)

**ls**

**cd simple-landing-page**

**git remote**

**git remote -v**

**git remote add sandhikagalih** [**https://github.com/asandhikagalih/simple-landing-page**](https://github.com/asandhikagalih/simple-landing-page.)

**clear**

**graph**

**git fetch sandhikagalih**

**graph**

**ganti index.html di sandhikagalih**

**git status**

**git add .**

**git commit -m “mengubah warna tombol”**

**git push -u origin master**

**git status**

Kalau buka github reponya sudah tambah commit

Kembali ke ariefcu

**git status**

Katanya up to date aja, jadi harus feth

**git fetch sandhikagalih**

**git status**

**graph**

**git merge sandhikagalih/master**

**graph**

Lokal kita sudah sama dengan github sandhikagalih tapi github kita belum sama

Maka untuk menyamakan github kita dengan lokal dan sandhikagalih, caranya :

**git push -u origin master**

**clear**

**graph**

Sekarang ketiganya sudah sama lagi

Sudah menskronkan 3 repo kita

Lokal dan 2 remote yang berbeda

**Git : #11 Remote Branch**

Yaitu bagaimana saat kita membuat branch di repo kita, lalu branch tersebut kita usulkan perubahannya ke repo sumbernya atau repo aslinya, di dalamnya kita akan belajar pull request, bagaimana perubahan kita agar disetujui diterima oleh pemilik repo aslinya

Di lokal bikin branch dulu, agar master selalu sama dengan repo aslinya

**git branch**, Cuma ada master

**git branch features** setelah itu kalian checkout untuk pindah ke branchnya

Atau bisa pakai ini

**git checkout -b features**

Kalau ini kalau branchnya belum ada dibikn dulu dan langsung pindah

Kalau mau satu2 :

**it branch features**

**git branch**

Sudah ada,

Agar pindah ke features

**git checkout features**

Kalau kalian lihat lagi di vscodenya paling kiri bawah sudah berada di branch di fatures

Kembali rubah sedikit di index.html

**git status**

**git add .**

**git commit -m “rubah dikit index.html”**

**gti status**

**graph**

Branch features kita maju 1 commit lainnya masih di belakang

**git push origin features**

**clear**

**graph**

Sudah sama dengan github kita

Sekarang kita usulkan pull request di github kita karena pull request Cuma ada di github

Klik compare & pull request

Klik create pull request

Sampai sini tugas kita sudah selesai, tinggal menuggu sandhika galih untuk menyetujui dan menerima request kita

Di github sandhikagalih, lihat pull request dari kita, lihat detilnya files changed

Kalau sudah ok, bisa review changes untuk diskusi atau komentar terhadap pull request

Merge pull requetsnya, confirm merge

Di master sekarang commitnya nambah 2, merge dan perubahan yang dilakukan oleh orang yang pull request

Di github ariefcu kalau sudah disetujui

Berarti tinggal menyakan lagi yang punya sandhikagalih dengan punya kita

**git fetch sandhikagalih**

**graph**

**git branch**

**git checkout master**

**git merge sandhikagalih/master**

**graph**

Lokal kita sudah sama dengan sandhikagalih/master

Yang belum sama adalah di github punya kita

**git push origin master**

**graph**

Sudah sama semua

Tinggal kalau kita sudah tidak butuh branch features maka bisa kita delete, karena tugasnya adalah untuk pull request saja

**git branch -d features**

**git branch**

Sudah hilang

**graph**

Ternyata di github kita masih ada branch features, kalau mau dihilangkan juga

**git push origin ---d features**

Branchnya sudah terhapus

**Graph**

Sudah tidak ada branch featuresnya

Kalau memastikan bisa dilihat di github kita

**Git : #12 .gitignore**

Sebuah file yang bisa kita simpan di dalam repository git kita agar kedepannya pada saat kita melakukan add dan commit ada file yang tidak ikut terbawa ke dalam addnya

Code . untuk buka vscode

Bisa menyimpan di folder .gitignore :

1. Nama file, contoh : config.txt
2. Folder, contoh : data/
3. Pola, contoh : \*.exe

Sebagai contoh lain adalah config untuk lokal, ga usah dibawa commit

Git push, yang kedua tidak perlu pakai -u karena upstreamnya sudah dianggap seperti yang kita set sebelumnya

Untuk detilnya bisa dilihat : github/gitignore

Atau bisa mudah dengan website gitignore.io dibuatkan rekomendarinya tinggal dimsukkan ke file .ignore, sebagai contoh parameternya adalah windows +, visualcodestudio + laravel, hasilnya adalah sbb :

# Created by https://www.toptal.com/developers/gitignore/api/windows,visualstudiocode,laravel

# Edit at https://www.toptal.com/developers/gitignore?templates=windows,visualstudiocode,laravel

### Laravel ###

/vendor/

node\_modules/

npm-debug.log

yarn-error.log

# Laravel 4 specific

bootstrap/compiled.php

app/storage/

# Laravel 5 & Lumen specific

public/storage

public/hot

# Laravel 5 & Lumen specific with changed public path

public\_html/storage

public\_html/hot

storage/\*.key

.env

Homestead.yaml

Homestead.json

/.vagrant

.phpunit.result.cache

### VisualStudioCode ###

.vscode/\*

!.vscode/settings.json

!.vscode/tasks.json

!.vscode/launch.json

!.vscode/extensions.json

!.vscode/\*.code-snippets

# Local History for Visual Studio Code

.history/

# Built Visual Studio Code Extensions

\*.vsix

### VisualStudioCode Patch ###

# Ignore all local history of files

.history

.ionide

### Windows ###

# Windows thumbnail cache files

Thumbs.db

Thumbs.db:encryptable

ehthumbs.db

ehthumbs\_vista.db

# Dump file

\*.stackdump

# Folder config file

[Dd]esktop.ini

# Recycle Bin used on file shares

$RECYCLE.BIN/

# Windows Installer files

\*.cab

\*.msi

\*.msix

\*.msm

\*.msp

# Windows shortcuts

\*.lnk

# End of https://www.toptal.com/developers/gitignore/api/windows,visualstudiocode,laravel

**Git : #13 Rebase**

Rebase adalah salah satu cara untuk memilih git workflow, sebelumnya ktia selalu pakai merging.

Di github kita,

pilih setting,

pilih collaborators,

masukkan siapa yang mau dimasukkan

accept dulu oleh orang yang dimasukkan

di gituhub yang kita undang, accept

jadi bisa rubah program tanpa harus pull request lagi

lalu di git bash orang yang kita undang

**git clone** [**https://github.com/ariefcu/coba-laravel.git**](https://github.com/ariefcu/coba-laravel.git)

cd coba-laravel

**clear**

**git status**

sudah up to date dengan masternya

biasakan untuk merubah sesuatu buat branch dulu

**git brach fitur\_keren**

**lalu git checkout fitur\_keren**

atau bisa langsung

**git checkout -b fitur\_keren**

langsung switch ke branch baru

**ubah index.html**

**git status**

**git commit -am “mengubah index.html lagi dan lagi hehe”**

**clear**

**graph**

ubah index.html

**git status**

**git commit -am “mengubah index.html lagi dan lagi”**

kembali ke ariefcu

ubah index.html

**git commit -am “mengubah index.html lagi dan lagi”**

**push -u origin master**

balik lagi ke yang diundang collaborats

sebelum melakukan merging masternya berubah ga

**git checkout master**

**clear**

**graph**

kalau kita merging maka akan merge commit, merging dengan menambahkan 1 commit

tapi kita tidak akan melakukan itu

**git bracnh**

**git checkout fitur\_keren**

**git rebase master**

**graph**

**git checkout master**

**graph**

**git rebase fitur\_keren**

**graph**

**git push -u origin master**

**graph**

**Git : #14 Git & Web Hosting**

Melakukan sinkronisasi antara lokal, github dan web hosting

[**https://medium.com/@gofrendiasgard/git-perjalanan-menuju-dunia-parallel-c4e74364cf84**](https://medium.com/@gofrendiasgard/git-perjalanan-menuju-dunia-parallel-c4e74364cf84)

**Git Repository: Alam Semesta dalam Genggaman Kita**

Sebelum mengenal git, mungkin teman-teman sudah terlebih dahulu mengenal github, gitlab, atau bitbucket. Ketiganya adalah penyedia layanan untuk menyimpan repository git yang kita miliki. Pada kenyataannya, kita bisa membuat git-server sendiri, atau bahkan menggunakan git tanpa git-server.

Tiga baris berikut adalah perintah sederhana untuk menciptakan repository git folder git-workshop:

* mkdir git-workshop  
  Buat directory git-workshop
* cd git-workshop  
  Masuk ke directory tersebut
* git init  
  Jadikan directory tersebut sebuah git repository.

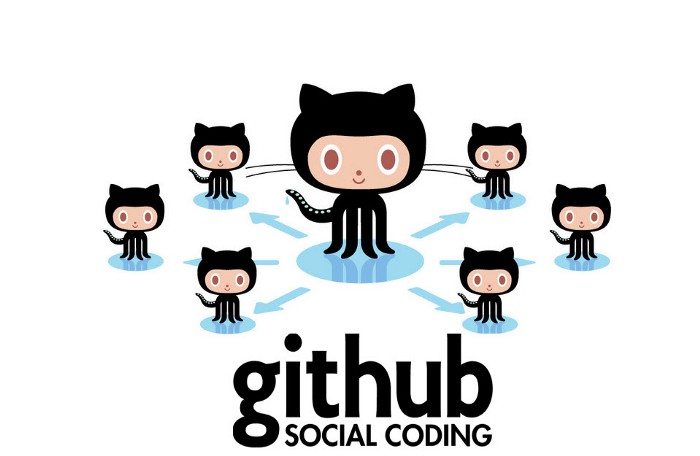
**Menambahkan File ke Git Repository**

Setelah kita memiliki git repository, kita bisa menambahkan sebuah file ke dalam git repository tersebut. Usai menambahkan file, kita perlu membawa file-file tersebut ke stage, sebelum akhirnya kita commit.

* echo "awal yang bagus" > catatan-sejarah.md  
  Di sini kita membuat sebuah file dengan nama catatan-sejarah.md dan mengisinya dengan sebaris kalimat awal yang bagus
* git add . -A  
  Lakukan staging pada semua file secara recursive (parameter -A) pada directory ini (tanda .)
* git commit -m "sebuah awal"  
  Commit semua file yang sudah di-stage dengan pesan sebuah awal.
* git status  
  Melihat status perubahan yang terjadi (ter-stage atau belum)
* git log  
  Melihat semua commit yang sudah terjadi sampai di titik ini

**Publikasi Git Repository ke Github**

Walaupun tidak wajib, namun ada baiknya kita simpan pekerjaan kita di github/bitbucket/gitlab. Selain memudahkan kolaborasi, ini juga dipakai untuk backup seandainya terjadi hal-hal yang tidak diinginkan dengan komputer/laptop kita.



Github: Social media yang minim bacotan netizen+62

Untuk melakukan itu, pertama-tama kalian harus punya account di github dan membuat repository di sana. Sebaiknya gunakan nama yang sama dengan repository lokal supaya tidak membingungkan.

Setelah itu, kalian dapat menjalankan perintah berikut:

* git remote add origin git@github.com:<user>/git-workshop.git  
  Dengan perintah ini, maka repository lokal kita akan mengenal repository github dengan alias origin.
* git push -u origin master  
  Sinkronisasi semua perubahan yang sudah kita lakukan di branch master ke github (dengan alias origin)
* git pull origin master  
  Sinkronisasi semua perubahan yang ada di github ke branch master di repository lokal.

Dengan mempublikasikan repository kita di github, maka kita membuka peluang kolaborasi dengan programmer lain di seluruh dunia.

Orang lain bisa bebas melakukan clone atau fork terhadap repository kita tersebut, melakukan perubahan seperlunya di branch mereka sendiri, dan memberikan pull-request.

Jika kalian masih belum ingin mempublikasikan repository kalian, maka kalian bisa mengatur setting repository menjadi private.

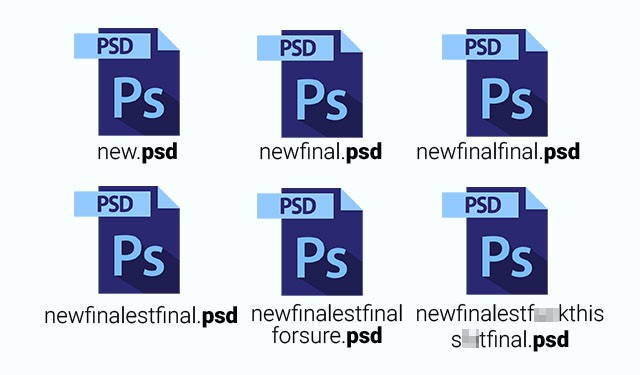
**Mengarungi Sang Waktu dengan Git: Unlimited Undo**

Kalian bisa melakukan beberapa commit yang lain, Lalu mencoba perintah berikut:

* git log  
  Melihat semua commit yang sudah terjadi sampai di titik ini.
* git rev-list --all  
  Melihat semua commit yang sudah terjadi, baik sebelum maupun setelah titik ini.
* git reflog  
  Melihat semua commit, termasuk checkout, yang sudah terjadi sampai di titik ini, sekaligus memberikan “referensi” seperti HEAD~0, HEAD~1, dan seterusnya.
* git checkout HEAD~1 atau git checkout <commit-hash>  
  Berpindah “satu langkah” ke belakang atau berpindah ke commit-hash tertentu
* git checkout master  
  Kembali ke commit terakhir di branch master.

Kalian lihat apa yang terjadi? Semua commit yang kalian lakukan tersimpan dengan baik, dan kalian bisa kembali ke titik manapun sesuka kalian !!!

Dengan adanya “unlimited undo” ini kalian tidak perlu lagi membuat file-file dengan nama konyol semacam ini:



Pekerjaan normies, tidak pakai git.

**Branch: Saat Kau Tak Harus Memilih**

Ada saatnya di mana kita harus berkolaborasi dengan orang lain untuk membuat fitur-fitur berbeda. Dalam kasus seperti ini akan lebih mudah seandainya kita bisa membuat “branch”.

Jadi ada versi source-code yang dikerjakan Si Anton, dan ada versi yang dikerjakan oleh Si Budi. Selama meng-coding, Anton dan Budi bisa bekerja secara terpisah. Lalu selanjutnya pekerjaan mereka bisa disatukan ke branch master.

Ini sangat menarik. Seolah-olah seperti kita menciptakan time-line baru di sebuah dunia parallel !!!

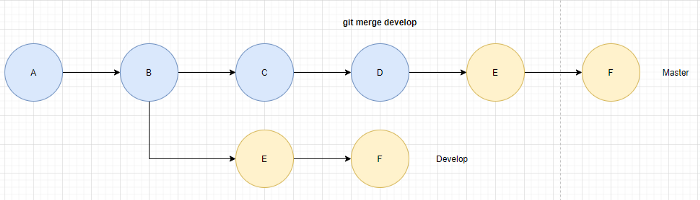


Yang bercabang dan menyatu: Rel kereta api, branch git, dan hubungan kita.

Untuk membuat branch baru kita bisa menggunakan perintah git checkout -b <nama-branch>

Berikut adalah perintah-perintah yang biasa dipakai untuk memanipulasi branch:

* git checkout -b <nama-branch>  
  Membuat branch baru dan berpindah ke branch tersebut
* git checkout <nama-branch>  
  Berpindah ke branch lain
* git merge <nama-branch>  
  Menggabungkan seluruh perubahan yang terjadi di branch nama-branch dengan perubahan yang kita lakukan di branch sekarang.



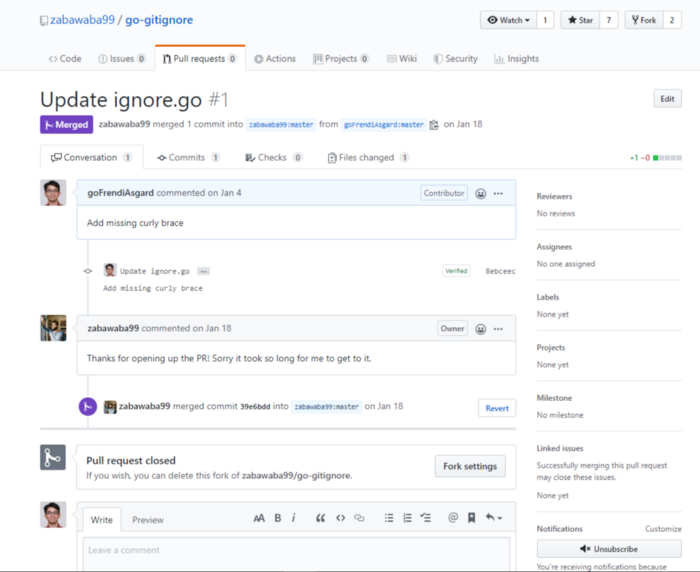
git merge

***PERINGATAN:****Saat bekerja bersama programmer lain, jangan pernah merge dan push langsung ke master. Pastikan semua perubahan yang kalian lakukan sudah melalui pull-request dan proses review.*

**Membuat Pull Request**

Setelah melakukan perubahan di branch kita, maka kita bisa meminta sang pemilik repository untuk menyatukan perubahan yang sudah kita buat ke branch master.

Github, bitbucket, dan gitlab sama-sama memiliki fitur pull-request, dan ini bisa dijadikan ajang untuk saling berkenalan serta mempererat tali silaturahmi sesama programmer.



Contoh Pull request (walaupun cuma nambahin kurung kurawal tutup)

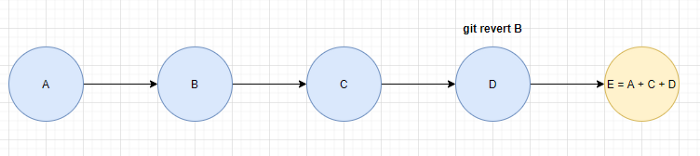
Kecuali kalian bekerja sendiri, maka ada baiknya kalian selalu melakukan pull request untuk setiap perubahan yang kalian lakukan. Di satu sisi ini membuat pekerjaan kalian “tercatat”. Di sisi lain, ini memungkinkan terjadinya code-review oleh rekan-rekan kalian.

**Mengubah Masa Lalu: Git revert**

Tak ada yang abadi. Fitur yang saat ini kita anggap bagus, mungkin beberapa bulan ke depan dianggap sampah. Sesuatu yang terlanjur berada di branch master pun, terkadang harus disesali.

Tak ada yang abadi. Bahkan band sekelas “Peterpan” pun tak abadi

Jika hanya ada sedikit commit yang bermasalah, maka cara terbaik untuk “memperbaiki masa lalu adalah dengan melakukan “revert”:



Git revert

git revert <commit-bermasalah>

Perintah revert memungkinkan kita membuat satu commit baru yang berisi semua commit sebelumnya kecuali commit yang di-revert. Untuk lebih jelasnya, coba lakukan serangkaian perintah berikut:

cd workshop-git-revert  
git init

echo 'alice' > alpha.html  
git add . -A && git commit -m "1st git commit: 1 file"

echo 'becky' > beta.html  
git add . -A && git commit -m "2nd git commit: 2 files"

echo 'callie' > charlie.html  
git add . -A && git commit -m "3rd git commit: 3 files"

echo 'diana' > delta.html  
git add . -A && git commit -m "4th git commit: 4 files"

echo 'ellen' > edison.html  
git add . -A && git commit -m "5th git commit: 5 files"

Sampai di sini, HEAD mengarah pada commit ke lima. Kita bisa me-refer commit-commit sebelumnya dengan “commit-hash” yang bisa didapatkan dengan perintah git log atau git rev-list --all. Namun supaya lebih mudah, kita bisa me-refer commit-commit sebelumnya dengan HEAD~1, HEAD~2 dan seterusnya.

ls  
git reflog

Semisal kita ingin membatalkan penambahan file beta.html (3 commit di belakang HEAD), maka kita bisa menjalankan perintah berikut:

git revert HEAD~3  
ls  
git reflog

Sukses !!! File beta.html sudah menghilang dari peredaran, seolah-olah tidak pernah ada.

Tapi tunggu dulu, pada kenyataannya, git revert bukan meniadakan sejarah, namun menambahkan satu commit baru yang berisi pembatalan satu commit.

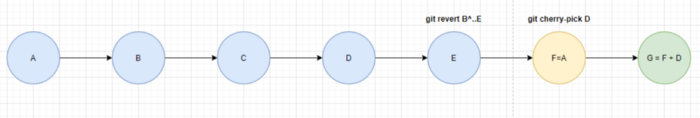
**Memilih commit terbaik: Git cherry-pick**

Kita sudah belajar bagaimana “membatalkan suatu commit”. Sekarang bagaimana jika kita hanya ingin membatalkan beberapa commit dan hanya ingin mengambil sebagian kecil commit saja?



cherry-pick

Kita bisa memanfaatkan fasilitas cherry-pick:



Git cherry-pick

cd workshop-git-cherry-pick  
git init

echo 'alice' > alpha.html  
git add . -A && git commit -m "1st git commit: 1 file"

echo 'becky' > beta.html  
git add . -A && git commit -m "2nd git commit: 2 files"

echo 'callie' > charlie.html  
git add . -A && git commit -m "3rd git commit: 3 files"

echo 'diana' > delta.html  
git add . -A && git commit -m "4th git commit: 4 files"

echo 'ellen' > edison.html  
git add . -A && git commit -m "5th git commit: 5 files"

ls  
git reflog

Sekarang katakan, kita ingin meniadakan perubahan kedua, ke empat, dan ke lima (harapannya, nanti hanya ada file alpha.html dan delta.html saja). Dalam hal ini kita bisa melakukan perintah berikut:

git revert 84e72d2^..733776 (dengan asumsi 84e72d2 adalah hash commit perubahan beta.html dan 733776 adalah hash commit perubahan edison.html)  
git cherry-pick f0d522e (dengan asumsi f0d522e adalah hash commit perubahan delta.html)

Oke, sekarang repository kita hanya berisi alpha.html dan delta.html.

ls  
git reflog

**With Great Power Comes Great Responsibility**

Dalam penggunaan sehari-hari, umumnya kita hanya menggunakan perintah-perintah umum seperti checkout, add, commit, dan push.

Perintah revert dan cherry-pick pada dasarnya tidak menghilangkan commit-commit sebelumnya, melainkan membuat sebuah commit baru. Segala perubahan yang ada masih tetap tersimpan sebagai “sejarah” dan dapat diambil sewaktu-waktu.

Di balik segala keunggulannya, git menuntut penggunanya untuk bertanggung jawab atas setiap tindakan yang mereka lakukan. Menghapus “sejarah” sebenarnya dimungkinkan, namun memiliki resiko yang sangat tinggi, terutama jika semua perubahan kita telah ter-sinkronisasi ke github/bitbucket/gitlab. Oleh karena itu, para developer selalu melakukan hal berikut:

* Tidak meng-upload data-data sensitif (user/password, API key, dsb)
* Tidak meng-upload file-file berukuran besar dan rentan berubah (node\_modules, grafik atau apapun yang di-generate ulang oleh program, file database sqlite)
* Cuci tangan dengan sabun, tidur secukupnya, makan makanan bergizi. (Stay sane and healthy guys, kesehatanmu yang paling utama… he he he…)