**Git Vs GitHub**

VCS (version Control system)

* Version Control System, disebut juga revision control system atau source code management adalah system yg mengelola perubahan dari sebuah document, program computer, website dan Kumpulan informasi lain.
* Sebuah system yg dapat mengelola perubahan dalam dokumen
* Kenapa? Digunakan

1. Untuk melakukan kolaborasi.
2. Melacak versi atau perubahan.
3. Mempamerkan program kita.

* VCS (sebuah system yg dpt meyimpan rekaman / snapshot perubahan pada source code
* Memungkinkan bekerja berkolaborasi dengan baik
* Mengetahui siapa yg melakukan dan kapan sebuah perubahan terjadi
* Memungkinkan kita untuk Kembali ke keadaan sebelum perubahan (checkout)

GitHub (version Control system) online

* Sebuah website
* Layanan cloud untuk menyimpan & mengelola project / repo git
* Instagram nya programmer
* Website untuk mengelola project Git
* Merge conflict (baris yg sama diubah oleh 2 brach yg berbeda)

Kita atau yg punya branch main yg harus membereskan conflict tsb.

Fork (tak sama dengan clone)

* Proses duplikat / mengambil repo di GitHub orang lain (beserta history-nya)
* Masuk akun yg dituju lalu pilih repositori-nya lalu cari fork(klik)
* Jembatan antara repo original dan duplikat-nya
* Sehingga dapat dengan mudah modifikasi terhadap repo originalnya
* Berkontribusi pada repo orang lain
* Fork bukan sekedar mengambil seperti download
* Untuk mengusulkan perubahan setelah fork, dan melakukan perubahan, klik (New full request)

Git? (version Control system) offline

* Sebuah vcs terdistribusi untuk mengelola perubahan file di dalam folder (repository / repo)
* Riwayat perubahan file disimpan menggunakan serangkaian commit
* Branch (istilah untuk cabang dari commit)
* Merge (proses penggabungan Branch)
* Repo (folder project )
* Commit (rekaman / snapshot dari repo kita pada keadaan tertentu)
* Hash (penanda unik pada sebuah commit)
* Checkout (berpindah ke sebuah commit / gak jadi)
* Remote (sumber yg memiliki repo (GitHub) )
* Clone (mengambil repo dari remote (GitHub) )
* Push (mengirimkan commit ke repo)
* Pull ( mengambil commit dari repo)

Implementasi di Git

* Masuk ke : <https://git-scm.com>
* Di Git ada Git Client (secara GUI)
* E-Book Pro Git di web: <https://git-scm.com/book/id/v2> untuk mempelajari lebih lanjut
* Perintah perintah git

1. $ git init (meng-inisialilasi repo git di computer)
2. $ git add <file(s)> (menambah file kedalam staging area)
3. $ git status (mengetahui status repo kita(ada file baru, berubah, dll) )
4. $ git commit (melakukan commit)
5. $ git config (memasukan konfigurasi kedalam git-nya)
6. $ git branch (membuat branch)
7. $ git help (mengetahui cara kerja sebuah perintah)
8. Dll

* 3 area pada repo git

1. Working tree (folder tempat bekerja)
2. Staging area (memberitahui git bahwa kita sudah lakukan perubahan)
3. History (masuknya file setelah di commit)
4. Staging area & History akan tersimpan pada sebuah file/folder yg Namanya (.git)
5. Pwd untuk cek kita ada di directory mana
6. Ls untuk cek ada apa saja di directory tersebut
7. Cd (…..) untuk merubah directory ingin pindah kemana
8. Cara supaya git mengawasi folder yg kita buat

* Pertama masuk ke folder nya menggunakan perintah di atas di cmd
* Lalu ketik git init
* Lalu setelah di init kita cek status nanti ada file yg belum di track supaya bisa commit masukkan filenya ke staging area dulu
* Gunakan perintah git add (nama file dan .extensinya)
* Untuk mengeluarkan dari staging area masukkan perintah : git rm –cached (nama file)
* Lalu untuk commit masukkan perintah :

git config –global user.email: “email kita”

git config –global user.name: “your name”

Karena git perlu tahu kita siapa

* Perlu commit kita beri perintah terlebih dahulu

git commit -m “tulis massage nya”

* Untuk memindahkan semua file ke staging area kita beri perintah :

git add .(titik/semua file yg diubah)

* Untuk tahu apa saja yg kita lakukan beri perintah: git log, git log -3 (untuk lihat perubahan 3 kali terakhir), git log – (namaFile) (untuk tahu secara spesifik yg terjadi pada suatu file), untuk mengembalikan keadaan karena tak jadi lakukan perubahan beri perintah :

git checkout (5digit pertama commit) -- (nama file secara spesifik)

Melakukan branch & merge pada Git

* Kalo file sudah mode modified bisa langsung beri perintah add dan commit bersamaan : git commit -a -m “massage-nya”, tapi kalo diawal atau bukan mode modified harus di add dulu
* Implementasi Branch’

1. Untuk tahu kita ada di branch yg mana ada namanya head
2. Untuk cek ada branch apa saja perintah : git branch
3. Untuk buat branch baru : git branch <namaBranch>
4. Branch yg aktif warna hijau
5. git log --all --decorate --oneline --graph (menampilkan visualisasi branch dalam bentuk graph)
6. untuk supaya tak perlu Panjang buat shorthand nya dengan cara

alias graph=” git log --all --decorate --oneline --graph”

jadi tinggal panggil graph saja

1. untuk berpindah branch

git checkout <namaBranch>

1. jadi saat berpindah2 branch perubahan yg terjadi masing masing branch berbeda

satu sama lain

1. untuk tahu branch yg aktif lihat ada Head-nya saat di cek dengan perintah graph

* implementasi Merge’

1. 2 jenis merge

Fast forward merge = terjadi Ketika barcnh yg digabungkan berada di jalur langsung,

1. Pindah dulu Head nya ke master
2. Lalu perintah git merge <namaBranch tujuan>
3. Untuk cek branch mana yg sudah di merge : git branch --merge
4. Hapus branch peritah: git branch -d <namaBranch yg ingin dihapus>
5. Untuk menghapus branch tanpa merge: git branch -D <namaBranch>

Three way merge = menggabungkan merge kiri & kanan atau taka da jalurnya

1. Pastikan ada di master dahulu git checkout master
2. Sama saja git merge <namaBranch tujuan>
3. Lalu bakal masuk codeEditor dan diminta lakukan pesan lalu close pesannya otomatis merge selesai
4. Lalu tinggal delete saja branch yg tak diperlukan

Git merge conflict (ada conflik saat merge) harus di resolve dulu

* Saat lakukan merge otomatis akan dibuka file yg conflict-nya
* Lakukan resolve
* Setelah resolve belum beres selanjutnya masukkan si hasil merge (modified) ke staging area
* Ada sebuah keadaan Ketika ingin checkout tapi bukan ke branch, tapi ke sebuah commit atau mundur masuk git log dulu untuk ambil 7 digit commit

1. Saat log tekan (q) untuk keluar
2. Pertama copy (7 digit dari commit-nya)
3. Lalu git checkout (7 digit-nya), lalu kita akan dlm keadaan detached HEAD, atau berada di keadaan commit yg ingin kita tuju(mundur)
4. Jika ingin kembali ke master Caranya tinggal checkout lagi ke master
5. Jika ingin berada Kembali ke keadaan suatu commit(mudur) dan melanjutkan commit tersebut caranya : git checkout (7digitCommit), lalu
6. Buat branch baru (tes)
7. Lalu buat HEAD mengarah ke tes (branch baru), dengan cara checkout ke tes (branch baru)

Git Remote, remote adl duplikat dari repo yg kita punya

* Tak hanya bekerja di local tetapi terhubung juga di GitHub
* 1 repo local bisa terhubung ke beberapa remote sekaligus
* GitHub = sebagai remote
* Git Remote (GitHub) adl
* Cara computer punya cloning dari repo di GitHub

1. Ke bagian code lalu pilih clone lalu copu HTTPS-nya
2. Masuk ke tujuan ingin disimpan (Desktop, dll) lalu masuk git bash,
3. Lalu perintah: git clone (paste-https-nya)
4. Lalu akan ada otomatis repo-nya di tujuan (Desktop)
5. Tinggal masuk ke reponya di git-bash: cd (nama reponya)
6. Lalu: git remote, akan muncul nama default nya (origin)
7. Lalu: git remote -v = untuk tahu Alamat repo nya
8. Lalu : graph maka akan muncul yg warna merah adalah branch y gada di remote (origin). Artinya sudah connect local dan remote-nya
9. Lalu jikamelakukan perubahan di local tetap harus dilakukan commit dulu, git add . & git commit -m “massagenya”. Lalu cek graph maka commit (perubahannya) di local bakal mendahului dari si remote
10. Nah lalu perlu dikirim commit (perubahannya) ke remote dengan : git push, lalu akan diminta login lagi ke GitHub, tinggal login saja dan beres untuk push.
11. Tinggal cek graph nya
12. (:q) untuk keluar jika terjebak
13. git --list untuk tahu info lengkap
14. git config --global user.name “Namanya/emailnya”

untuk ganti email dan name

* kasus 2 gimana kalua di local terlebih dahulu lalu ke remotenya (GitHub)

1. cd .. untuk keluar directory
2. lalu buat file di folder langsung atau pakai perintah: mkdir (nama reponya)
3. lalu ubah reponya jadi repo git dengan : gir init
4. dan tinggal edit reponya dengan menambah file , dll
5. lalu untuk disimpan ke GitHub
6. pertama buat repo di GitHub seperti biasa
7. hanya saja (Initialize this repository with) jangan di ceklis
8. lalu selanjutnya akan muncul beberapa perintah lihat

perintah: …or push an existing repository from the command line

ikuti saja perintah perintah dibawahnya satu persatu

git remote add origin <https://github.com/muhamadsaepulrizal/belajar2-GitHub.git> (untuk menghubungkan dengan lokal)

lalu untuk memasukkan repo di local ke GitHub:

git push -u origin main

Pokoknya ikuti saja perintah dibawah Tulisan ini: …or push an existing repository from the command line.

1. Jika ada conflict di remote dan di local ada yg mengedit serupa
2. Di remote oleh orang lain ditambahkan style.cs
3. Di local saya tambahkan stle.css juga

Nah saat di git push di local bakal ditolak

Beri perintah git fetch untuk cek commit di remote

1. Lalu kita graph bakal kelihatan di remote dan local beda cabang(branch)
2. Lalu git pull yg ada di remotenya terlebih dahulu
3. Lalu bakala masuk ke code editor disana bakal di suruh memilih mana yg akan dipakai antara punya di local atau di remote hasil orang lain
4. Lalu kita commit file yg dipilihnya
5. Lalu tinggal push saja.

GitHub Pages (membuat GitHub sebagai web Hosting)

* Bisa digunakan sebagai web Hosting hanya untuk web yg statik hanya menggunakan Html, Css, Javascript saja. Secara gratis dengan Batasan-batasan
* Caranya:
* Pertama buat repo di GitHub, lalu untuk penamaannya reponya:

namaAkunGitHub.github.io

* Lalu buat descriptionnya, lalu buat public, jangan centang with a README-nya
* Lalu klik create repo
* Lalu di lokalnya si folder kita buat jadi repo git terlebih dahulu
* Cara jadi repo ada diatas.
* Lalu hubungkan ke GitHub: git remote add origin (Alamat GitHub seperti https…../namaAkunGitHUb/namaRepo yg dibuat.github.io.git).
* Atau untuk link biasanya sudah ada.
* Setelah itu jangan lupa untuk commit repo di local-nya terlebih dahulu.
* Lalu push code di local ke GitHub: git push -u origin main, dilakukan sekali selanjutnya jika hasil edit di local-nya tinggal git push
* Setelah push buat repo jadi halaman web
* Pertama ke settings di GitHub, pastikan branch ke main
* Bagaimana jika suatu saat website ada perubahan?
* Pertama edit saja codingannya
* Lalu tinggal add, commit dan push lagi beres.