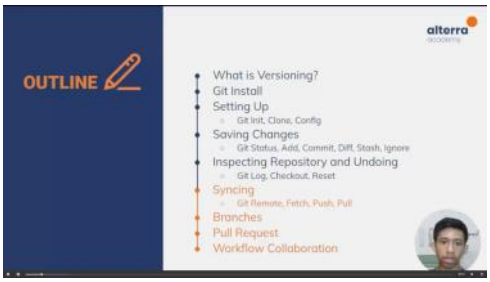
Version Control & Branch Management (Git)

20 January 2022 20:56



INTRO : Apa itu verisoning

Versioning adalah **mengatur versi** dari source **code program** kita

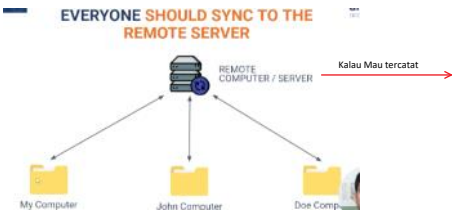
Kita bingung kalo ga ada version control inii kita bakal susah ngebedeain ini mana yang udah bener yah ?

#Tools2 nya

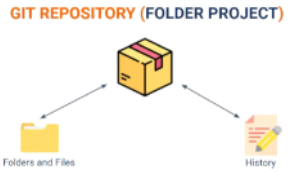
1. Git
2. Visual Code
3. Github sebagai repository nya

#bisa **realworld collaboration**

Pemanfaatan git ini terdistribusi bukan centralisasi sehingga setiap orang
Punya code nya sendiri, sehingga setiap orang masi punya backup an



Kalau Mau tercatat



Git track every file changes.
Your changes, John's changes, everyone!

Commit (kita upload ke server si gituhub)
Saat kita commit, github akan mencatat perubahan nya apa, seperti apa, dan dimana nya yang berubah

Git can undo to some 'points'
We call it as Commit
Commit = the record of changes

Nah kita disini untuk remote computer memanfaatkan github agar saling terhubung

#Sejarah Git hub

Version Control System (VCS)
Source Code Manager (SCM)
Revision Control System (RCS)

Sifatnya localized karena internet tidak berkembang, pekerjaan masi terpusat

Udah local server, mulai terhubung, internet berkembang. Tapi ketika server down gabisa ngapa'in nunggu internet hidup lagi

Git muncul, menggabungkan centralized dengan user2 yang ada
Dari lokal masi bisa edit, sehingga kalo down masi bisa ngerjain di lokal nanti di push ketika server udah mulai kembali lagi

github = git hosting service
Go to github.com and create new repository!

Sebenernya ada tools2 lain tapi kita memanfaatkan tools yang ada namun untuk belajar Version control kita belajar ini dlo aja

- Github layaknya social media bagi programmer
- Kita bisa lihat library orang dan berkontribusi
- Kontribusi memperbagus git hub kita

Git Install

13 February 2022 14:46

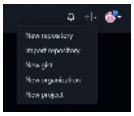
INSTALL GIT WINDOWS

1. Download the latest Git for Windows installer.
2. When you've successfully started the installer, you should see the **Git Setup** wizard screen. Follow the **Next** and **Finish** prompts to complete the installation. The default options are pretty sensible for most users.
3. Open a Command Prompt (or Git Bash if during installation you elected not to use Git from the Windows Command Prompt).

GIT INIT, CLONE, CONFIG

Setelah git config awal2 kasi nama dan email langsung start dengan cara clone #3 dari git repo nya

1. Pertama masuk ke github dlu di www.github.com
2. Buat new respository

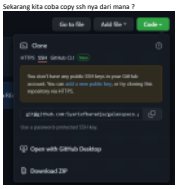


3. Kasih nama respository
4. Terus konfigurasi sisa nya

Jan lupa gitignore < select node
(untuk mencegah beberapa file atau folder tidak teruplod di github nya, karena setiap orang udah punya file dalam folder itu, kita bisa melakukan secara manual)

5. Nah create repository

Sekarang kita coba copy ssh nya dari mana ?



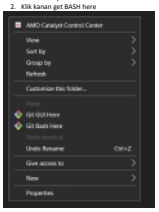
Dari code < ssh < copy link

(https juga sabi)

Clone = Download

Sekarang set up folder clone nya

1. Cari foldernya
2. Klik kanan git BASH here



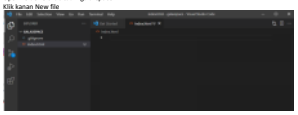
3. Git Clone <link>

4. Nah isi dari folder galaxspace di folder lokal sama dengan yang ada di repository github
5. Step ke 2 yakni di step nya ialah $ cd <nama repository nya>
   - Tulis cd
   - Tab untuk cek
   - Tulis nama repository nya
   - $ cd galaxspace

Nah ini tanda kita udah masuk ke main

Terus kita buka VSCode dimana kita bisa memasukkan file ke repository

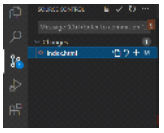
1. Open Folder Github tadi <galaxspace>
2. Klik kanan New file



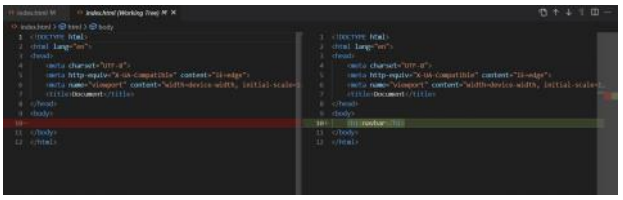
3. Nah kita mau masukkan index tadi ke repo kita yang ada di online
   (kalau temen2 kita pengen tau isi dari program kita apa, itu bisa akses langsung ke repo nya
   - Ga perlu kirim manual
   - Bisa dikerjain banyak orang

Git diff and stash

- **Git Diff** mengetahui bagaimana perubahan terjadi pada file yang kita edit
- **Git Stash** untuk menyimpan perubahan, sehingga dapat dikembalikan di ke perubahan yang diinginkan



Cara melihat perbedaan juga bisa click apa yang changes



Cara melihat perbedaan :

1. Jika ada perubahan
2. Save dulu
3. Git diff
   - Hijau menandakan perbedaan
   - Merah mendakan di line yang sama sebelum nya seperti apa sebelum ada perubahan

Setelah melakukan git diff kita bisa melakukan **git stash apply**

1. Titik 3

2. Lalu klik stash < stash paling atas
3. Perubahan akan ilang dan janlup dinamai
4. Untuk mengambil perubahan tadi klik stash < pop stash
5. Pilih perubahan yang diinginkan

Lalu ada **Git ignore**
Fungsinya mana aja yang **boleh kita masukin** ke dalam repository mana aja **yang gaboleh**
(kita ga perlu masukkin library-library yang bisa diinstall oleh temen2 kita)

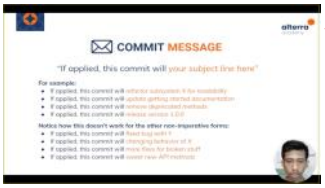
Contoh : node moduls
Soalnya didalam node moduls isinya banyak banget library, nah library2 tersebut gausa dimasukkin repository

- Cukup di download secara lokal

Contohnya buat package jason, nanti package jason bakal nge list apa aja library yg kita pake

Kalo kita masukin ke sono temen kita tinggal melakukan rujukan kepada package.json supaya nanti instalasi sesuai package.json

Ada beberapa staging area dalam GIT



Kalo bisa -m (subjek nya itu harus jelas)

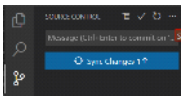
Nah supaya masuk ke staging area **klik tanda plus** yang kita masuk lewat logo branch (source control)
(manual nya "git add .")

Terus buka new terminal dgn cara terminal < new terminal

- Git commit -m "kata" (-m itu adalah pesan atau subjek lah ya)
- Git push origin (origin adalah main / default dari github nya kita bisa ganti)

Jadi basically logikanya

- Di working directory ditambahkan ke source control (git add untuk masuk ke staging area)
- Terus git commit di wrap menjadi satu package
- Terus di git push origin untuk di up ke repository secara online



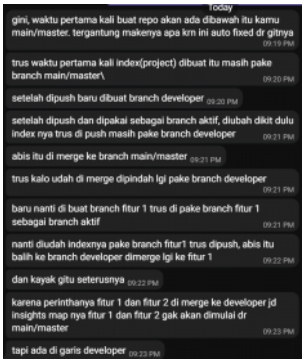
Bisa juga git push langsung klik sync changes

GIT STATUS, ADD, COMMIT

```
$ git status

$ git add <directory>
$ git add hello.py
$ git add .

$ git commit -m "add config file"
```



Inspecting Repository

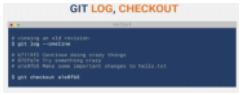
Pindah versi

Nah sekarang kita belajar gimana caranya **inspecting repository**

Bagaimana kita memanajemen versi dari file yang kita buat

- Tadi kita uda commit ke beberapa file
- Uda analisis juga nama file dan perubahannya

Nah disini kita akan kembali gimana **kembalii ke versi sebelumnya**



- Jan lupa git checkout main

Tampilan git log –oneline

Kita bisa memilih sampe versi mana kita bisa masuk

Setelah memilih kita bisa pake git checkout <nomer versi>

Gimana cara kembali lagi :

- Bisa pake : git checkout main

Syncing

Git remote -v

Ini adalah tampilannya

Menampilkan link dari repository yang udh kita access

Kalo kita mau fetching origin, nanti tulisin aja origin

Kalo kita pake versi sebelumnya nih, tapi belum push dan **berbeda dari versi di repository**

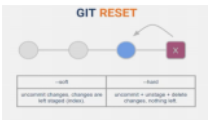
Kalo kita pengen kembali ke versi yang ada di repository bisa pake

$ git pull origin main

Kalo mau push juga

$ git push origin main

Git Reset



Soft : perubahannya ndak terjadi sebanyak yang di hard (si commit an tetep ada)
Hard : bisa untuk penghapusan commit an yg ada di versi yg telah kita masukin

*Rekomendasi
Pake reset yang soft karena hard akan ngerusak dari commit an itu sendiri
(ketika nanti ada perubahan yang kita pen kembali kalo pake hard, tidak bisa akses)

Branch and conflict

15 February 2022 12:42

Branches = percabangan didalam pembuatan github sendiri

Kalau pengen ngembangin project kita dengan **banyak orang**

Lebih baik tarik dari **Main**

Abis itu masukin ke **branch2** yang telah di buat

- Nanti ada rekomendasi seperti apa **susunan branch** yang bagus

GIT BRANCHING

#Create new Branch

Nah untuk awal2 kita buat **branch developer**

Branch developer berfungsi untuk kita nge push sebelum ke main, kita push dulu ke developer

$ git branch developer

Kita cek dulu apakah ada bug2, atau dalam segi keamanan, kalau udah tercipta bagus nanti push semua ke main

Nah pas kita cek belom ada di **repository**, kenapa ?

Karena kita belum melakukan push terhadap branch developer

#Push Branch baru

$ git push -u origin developer

- **Origin** : nama git yang terhubung
- **Developer** : nama branch yang baru dibuat

Akan muncul branch baru di repository

#Gimana nambahin dari branch push ke main

Cara pertama pindah dulu ke **branch developer**

Cara pindah branch dari main :

- **$ git checkout <branch>**

$ git checkout developer

Di contoh kita nambah sesuatu misal di h2 <di branch developer>

Commit & push

Cek di repository

Branch dan main berbeda ternyata

Merge

Nah skrg kita gabungin yg di developer ke arah yg ada di **Main**

*Balik lagi ke branch main

GIT MERGE

Nah skrg kita pake

- **$ git merge new- feature**

$ git merge developer

Terus **Push**

- Intinya nanti kalo ada perubahan apapun tolong di push di developer dulu
- Udah clear, udah testing, clear bug, cek keamanannya, baru up ke **main**

Teknik Pull Request

Dimana untuk kita berkontribusi pada repository yg sudah ada

- Baik itu repository temen, perusahaan, atau kita sendiri atau akun yang lain

1. Kita **Fork** diu repository yang open
   (forking itu kyk kloning tapi pake repo orang lain)
2. Lalu kita buka git bash
3. $ git clone <link https repository>

Sek lanjut besok pagi

#Workflow Collaboration

Dimana kita akan belajar bagaimana si mengelola branch dengan bagus

Objective : memahami **workflow** kolaborasi dari **github** dan **gitlab**

#Gimana cara **optimalkan** workflow dari github atau gitlab ?

Apakah kita bekerja akan selalu seperti ini ?
Jawabannya : **Tidak**

- Biasanya versi diatas digunakan ketika ingin mengembangkan **small product**
- Atau produk pribadi yang gak dikerjakan bersama-sama
- Produk buat latihan bisa gunakan tipe diatas

#yang rekomended untuk projek besar dan tim

- **Master** : adalah segala hal yang terkait dengan produk yang udah bener2 fix no bug dengan fitur2 yg udah **fix**
- Untuk developer : Ini yang akan selalu kita akses
  - Ngebuat branch baru
  - Ngebuat fitur baru
  - Dan ngelakuin merge

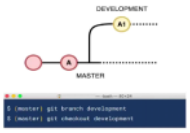
#contoh workflow nya

1. Kita **pull** dari **master** nya
2. Kita namain **developer**
3. Ketika pengen buat suatu fitur baru pull lagi dari **developer** ke **branch fitur**
4. Kalo udah jadi di merge ke **developer**
5. Kalo **developer** udah jadi di **merge** ke **master/main**

#TIPS-TIPS

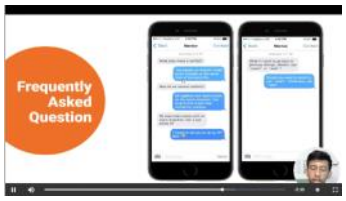
Biarkan branch master tidak terdistribusi

Kerjakan semua di branch developer



Hindari edit langsung di development

#Conflict



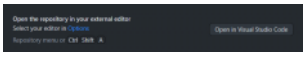
Conflict adalah dimana kita **ngerubah satu line** yang sama pada branch yang berbeda

Example :

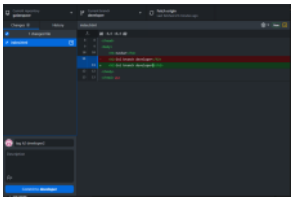
- Kita ngerubah di satu line di main dan developer secara bersamaan
- Jangan lupa commit dan push

Skrg kita kolaborasikan dengan aplikasi **git desktop**

1. Kita open melalui **git dekstop** visual code nya



2. Lalu buat perbedaan di branch developer (commit nya di **git desktop**)

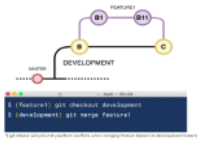


Skrg untuk Main lakukan **perubahan juga** (step nya sama)

1. Open melalui git visual code nya
2. Buat perbedaan
3. Commit dan push

3. Lalu push origin

Kalo masukin fitur yang udah ada kalo mau merge
Masukin ke development aja



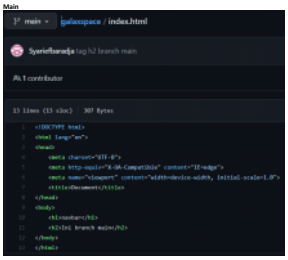
Jangan pernah dari fitur ke main langsung

Yang boleh akses main itu hanya dari development ketika udah selesai

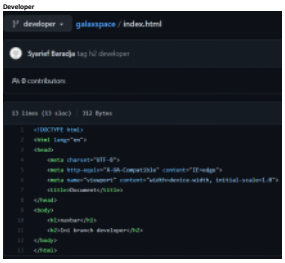
- Dicek oleh **projek manager / manager**

#saat di cek ada perbedaan

Main

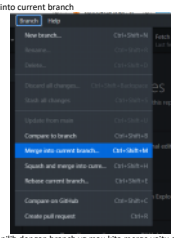


Developer

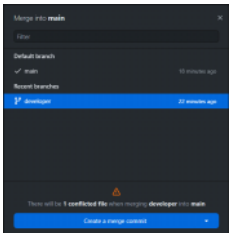


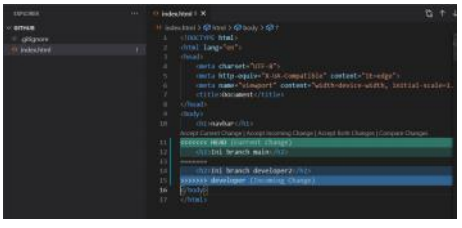
Nah saat mau **dimerge** kedua2nya akan terjadi **conflict (harus ada sesuatu yg dilakuin)**

1. Pertama di **git desktop** ubah ke **main**
2. Branch < merge into current branch



3. Selanjutnya kita pilih dengan branch yg mau kita merge yaitu developer





Apply dari development ke master kalo bener2 udah fix tidak ada bug bener2 fuixxx

#Tugas 1

Tugas

1. Buat sebuah repository di Github
2. Implementasikan penggunaan branching yang terdiri dari master, development, featureA, dan featureB
3. Implementasikan intruksi git untuk push, pull, stash dan merge
4. Implementasikan sebuah penanganan **conflict** di branch developement ketika setelah merge dari branch featureA lalu merge dari branch featureB (Conflict bisa terjadi jika kedua branch mengerjakan di file dan line code yang sama)
5. Gunakan merge no fast forward
6. Kirimkan alamat repository github di :

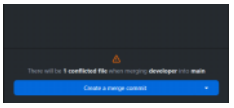
#Tugas 2

Tugas : github

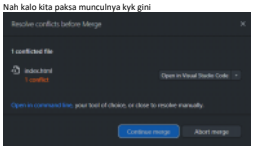
1. Buat lah repository baru di github
2. Masukkan project (terserah kalian) dan buat project itu sebagai master
3. Buatlah brach baru yang berfungsi sebagai develop
4. Buatlah brach ketiga sebagai penambahan fitur, pada brach ini lakukan penambahan fitur. Lakukan merge pada berach ketiga kedalam brach develop.
5. Buatlah brach ke empat sebagai penambah fitur, pada brach ini lakukan penambahan fitur. Lakukan merge pada berach ke empat kedalam brach develop.
6. Buktikan dengan screenshot dari insights=>network. Kumpulkan SS dan juga link dari repo kalian pada alta.id

Tugas Miniproject :

1. Buatlah desain figma untuk kalian gunakan pada mini project
2. Tema pada miniproject bebas, ide kalian akan kita diskusikan jumat. Maka siapkan figma dengan sebaik mungkin.



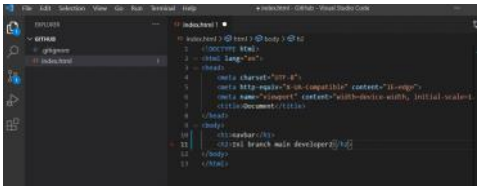
4. Nah kalo kita paksa munculnya kyk gini



Disini kita harus **open in visual studio code**

Jadinya kyk gini

5. Tahap selanjutnya adalah kita harus mendudukkan temen kita sedeveloper kita bicarakan dengan teman kita tadi
6. Kalo gabisa dihubungi ya pahami kode nya, selesaikan konflik nya

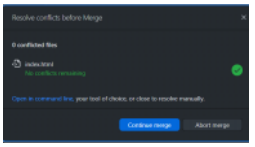


Dihilangin aja antara bisa milih salah satu yg main atau developer

Atau

Bisa digabung jadi 2 2 nya jadi satu.

JANGAN LUPA DI SAVE !



Pas balek ke branch merge tadi udah hijau dan bisa **merge**

7. Setelah itu **push origin** dan cek di repo **main**