Modul 1 Programming
Version Control System: Git dan GitHub

1. Apa itu *Version Control System*?

Version Control System (VCS) adalah suatu sistem yang merekam perubahan yang terjadi pada *source code*. VCS ini memungkinkan penggunanya untuk bekerja berkolaborasi dengan lebih baik. VCS juga memberikan informasi mengenai siapa dan kapan suatu perubahan itu terjadi. Beberapa VCS yang telah ramai digunakan adalah, antara lain, Git, Subversion, Mercurial, dan CVS. Pada modul ini, akan dibahas lebih lanjut mengenai Git dan GitHub.

2. Git

2.1 Overview

Git adalah salah satu *version control system*. Git dibuat oleh Linus Torvalds pada tahun 2005. Git merupakan aplikasi yang gratis dan open-source. Git sangat penting untuk dipelajari karena dalam pekerjaan dalam tim, *source code* akan berganti terus-menerus dan akan sangat menyulitkan jika setiap saat harus dilakukan penyamaan kode secara terus menerus. (INGAT) Git dan GitHub merupakan dua hal yang berbeda, dan GitHub akan dibahas lebih lanjut pada *chapter* berikutnya.

Keyword:

- *repository*/repositori/repo: folder yang menyimpan *history* dari *project* kita.
- *commit*: rekaman/snapshot dari repo.

2.2 Instalasi Git

Untuk cara instalasi Git, silahkan lihat pada link berikut ini https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

2.3 Area dalam Git

Dalam Git, dikenal tiga “area” dalam suatu repo:

- *Working tree*; folder biasa yang diisi dengan pekerjaan kalian.
- *Staging area*; memberi tahu Git bahwa kita melakukan perubahan
- *History*; jika melakukan commit, akan masuk dalam history. Setelah folder diinisiasi sebagai repo, akan tercipta folder “`.git`”. Folder ini juga tempat dari *staging area* dan *history* tersimpan.

2.4 Menggunakan Git

2.4.1 Membuat repositori lokal

1. Buka Terminal
2. Pergi ke folder project. Gunakan command "`cd`"

```
$ cd <nama-lokasi-folder>
```

3. Gunakan "`git init`" untuk menginisiasi pembuatan repo.

```
$ git init
```

4. Repo telah terbuat dan pengerjaan project siap dilaksanakan.

2.4.2 Mencatat perubahan dalam project dalam *staging area* dan mencatat perubahan dalam *history* (*commit*)

1. Buka Terminal
2. Pergi ke folder project

```
$ cd <nama-lokasi-folder>
```

3. Gunakan "`git status`" untuk melihat apakah ada perubahan dalam project

```
$ git status
```

4. Tambahkan file ke dalam *staging area* dengan "`git add <nama-file>`". Jika ingin memasukkan satu folder, ketik "`git add <nama-folder>`". Jika ingin memasukkan semua perubahan, ketik "`git add .`"

```
$ git add .
```

5. Perubahan telah tercatat dalam *staging area*. Selanjutnya, jika semua perubahan yang tercatat sudah cukup …, perubahan dapat dicatat dalam *history* dengan mengetik "`git commit -m "<pesan-commit>""`

```
$ git commit -m ">pesan-commit>"
```

6. Perubahan telah tercatat dalam *history*. Catatan isi *history* dapat dilihat dengan mengetik "`git log`".

```
$ git log
```

2.5 *Branching* dan *Merging*

Branching adalah kegiatan untuk membuat branch/cabang, sedangkan *merging* adalah kegiatan untuk menggabungkan branch kembali ke jalur utama/master. Branch menyebabkan snapshot yang dibuat sebelumnya tidak mengganggu jalur utama.

Branching dilakukan untuk mendukung fitur eksperimental atau percobaan. Jika branch yang dibuat itu sesuai dengan yang diinginkan, branch tersebut kemudian di-*merge* sehingga menyatu Kembali dengan jalur utama. Namun, jika branch yang dibuat tidak sesuai dengan keinginan, branch bisa diabaikan saja atau dihapus tanpa menggangu jalur utama dari project.

2.5.1 Membuat cabang

- Gunakan "`git branch <nama-branch>`" untuk membuat branch.

```
$ git branch <nama-branch>
```

- Gunakan "`git branch`" untuk melihat semua branch.

```
$ git branch
```

- Untuk pindah ke branch lain, gunakan "`git checkout <nama-branch-lain>`"

```
$ git checkout <nama-branch-lain>
```

More about Git: https://www.w3schools.com/git/

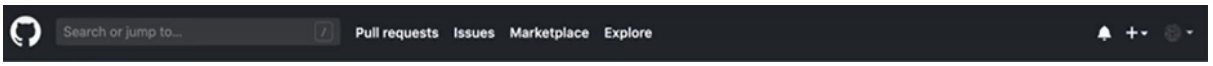
3. GitHub

3.1 Overview

GitHub adalah layanan *cloud* untuk menyimpan dan mengelola *project* / repo git. GitHub memungkinkan sekelompok orang untuk berkolaborasi dengan menggunakan GitHub sebagai *remote*, yaitu sumber dari repo. Para kolaborator bisa melakukan *clone* terlebih dahulu agar repo yang ada pada github juga terpasang pada komputer lokal mereka. Setelah itu, para kolaborator bisa melakukan *pull* untuk mengambil *commit* dari repo GitHub dan juga bisa melakukan *commit* ke repo GitHub.

3.2 Membuat Repository dan File

1. Klik tanda `+` lalu pilih `create new repository`



2. Lengkapi spesifikasi dari repository lalu tekan tombol `create`
3. Klik `Create new file`
4. Isi file, lalu `Commit new file`
5. Jika ingin mengedit file tekan tombol ikon pensil, masukkan perubahan, dan `Commit changes`. Jika dilakukan penambahan, tanda `+` akan muncul dan juga muncul warna hijau pada baris yang ditambah, sedangkan jika dilakukan pengurangan, tanda `-` akan muncul dan juga muncul warna merah pada baris yang dikurangi.
6. Untuk melihat perubahan secara visual bisa dilihat pada bagian `Insight` -> `Network`

3.3 Fork

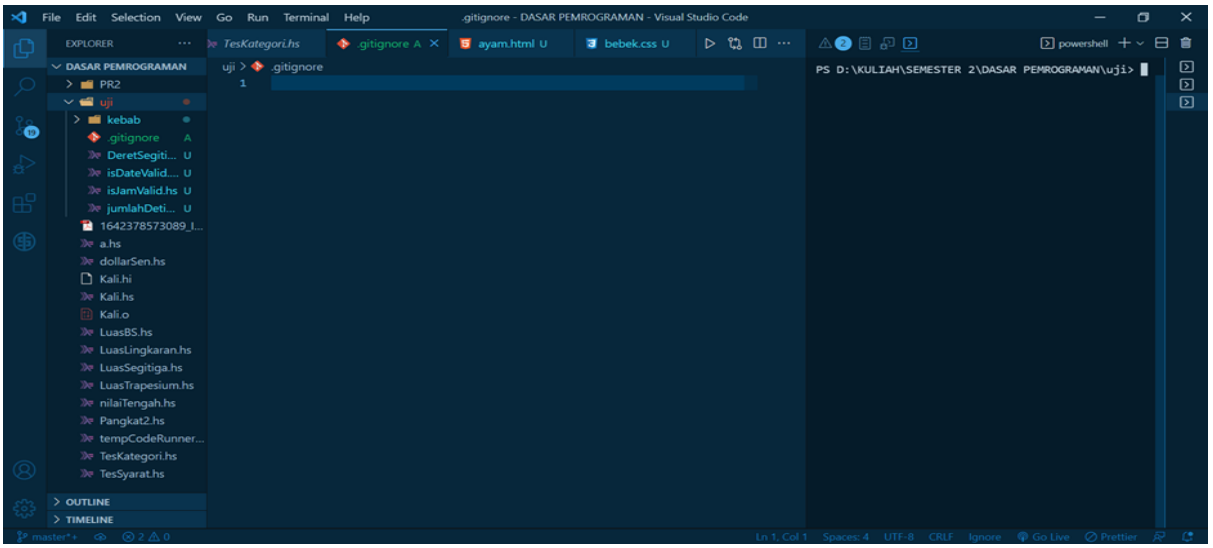
Forking adalah kegiatan membuat *copy* / duplikat dari repo orang lain beserta *history*-nya. *Forking* berfungsi sebagai penghubung antara repo original dan duplikatnya. Dengan melakukan *fork*, kita bisa melakukan modifikasi terhadap repo original agar sesuai dengan yang dibutuhkan tanpa mengganggu repo originalnya. Selain itu, kita juga bisa berkontribusi untuk repo orang lain melalui pull request pada repo yang kita *fork* tadi. Lebih lanjut, *fork* dan *clone* memiliki perbedaan yang cukup signifikan. *Fork* melakukan duplikat dari repo menuju repo pengguna, sedangkan *clone* mengambil repo dari *remote* menuju komputer lokal.

3.4 .gitignore

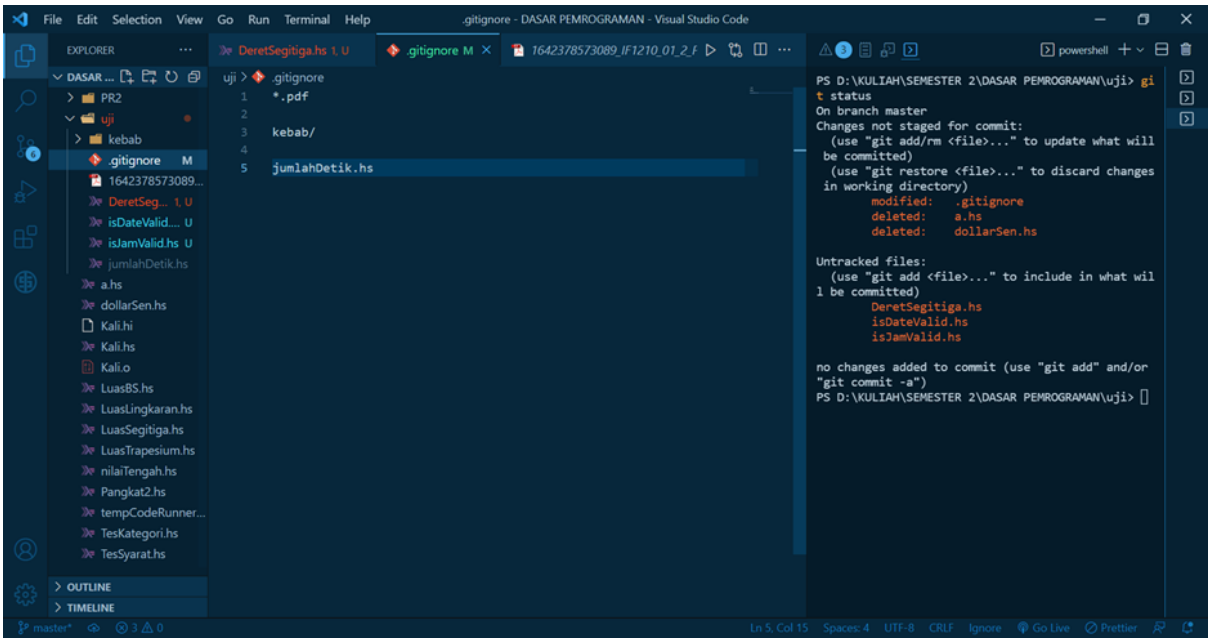
.gitignore adalah suatu file yang berfungsi untuk menyembunyikan file/folder tertentu agar tidak ikut terpanggil saat ada command yang memanggil semua file/folder.

Cara menggunakan gitignore

1. Buat file`.gitignore`



2. Tulis nama file / folder / ekstensi yang ingin diabaikan



File/folder/ekstensi yang ditulis di file `.gitignore` diabaikan

Github Cheat sheet : https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet