Rebuilding Git With Golang

Bab 1

Apa itu git ?

Git adalah suatu tool yang sering kali digunakan untuk pengembangan software. Fungsinya adalah sebagai sistem pengontrol versi (*Version Control System*) pada proyek perangkat lunak.

Kenapa kita karus menggunakan git?

Bab 2

Perkenalan directory .git

Langkah pertama saat kita menggunakan git biasanya menggunakan comand `git init`, comand tersebut akan membuat folder baru dengan nama `.git`. untuk struktur foldernya akan jadi seperti ini :

```
.git
|- config
|- description
|- HEAD
|- hooks
|  |- pre-commit.sample
|- info
|  |- exclude
|- objects
|  |- info
|  |- pack
|- refs
   |- heads
   |- remotes
   |- tags
```

.git/config

File `.git/config` berisi konfigurasi setting yang hanya berlaku untuk repositorynya sendiri.

```
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
```

- kita menggunakan repository format versi 0
- git haru menyimpan setiap mode dari file (contoh apakah file `a` dapat di eksekusi)
- bukan **bare** repository, artinya repository ini adalah tempat pengguna mengedit pekerjaan, copy file, dan membuat commit.
- reflog true, artinya semua perubahan pada file di .git/refs dicatan di .git/log

File `.git/config` juga menyimpan alamat remote repository

```
[remote "origin"]
        url = git@github.com:needkopi/gt.git
        fetch = +refs/heads/*:refs/remotes/origin/*
```

.git/description

File ini berisi nama repository, ini dipakai oleh **gitweb**. secara default isi file tersebut seperti ini :

```
Unnamed repository; edit this file 'description' to name the repository.
```

.git/HEAD

File ini berisi referensi commit saat ini.

```
ref: refs/heads/master
```

.git/hooks

Folder ini berisi script yang akan dijalankan oleh git sebagai bagian dari perintah inti tertentu. Contohnya ketika kita menjalankan perintah `commit`, git akan mengeksekusi file `.git/hooks/pre-commit` jika file tersebut ada. ecara default git akan membuatnya dengan nama `.git/hooks/pre-commit.sample`, kita dapat mengaktifkan file tersebut dengan cara menghapus `.sample`.

.git/info

Secara default ketika pertama kali menjalankan perintah `git init`, git akan membuat kan file baru bernama `.git/info/exclude`. Mungkin kebanyakan dari kita lebih familiar dengan file `.gitignore`, fungsi dari kedua file tersebut mirip, yaitu untuk mendeteksi atau mengabaikan file-file mana saja yang tidak akan di bagikan ke repository. Perbedaan dari keduanya file `.gitignore` bersifat global, yang artinya semua user yang mengakses repository tersebut akan mengabaikan file yang sama. sedangkan untuk file `.git/info/exclude` ini hanya berlaku untuk user tersebut.

.git/objects

Folder `.git/objects` digunakan untuk menyimpan data atau database bagi git.

.git/refs

Folder `.git/refs` ini digunakan untuk menyimpan berbagai jenis pointer yang disimpan didalam database `.git/object`. pointer ini bisanya hanya file yang berisi id. Folder `.git/refs/heads` berfungsi untuk menyimpan id commit terakhir di lokal. Folder `.git/refs/remotes` berfungsi menyimpan id commit terakhir dari berbagai remote di repository. Folder `.git/refs/tags` berfungsi untuk menyimpan tags.

Bab 3

Perintah `init`

Sebelum membuat beberapa perintah git yang lain, perintah pertama yang harus kita buat adalah `init` dengan tujuan untuk membuat folder `.git`. Pertama buat file `main.go` untuk sekarang file ini hanya berisi switch case perintah yang akan digunakan

```
package main

import (
    "os"
    gtInit "gt/init"
)

func main() {

    command := os.Args[1]

    switch command {
    case "init":
        gtInit.Init()
    }
}
```

Lalu kita buat pacakge init

```
// init/init.go
package init

...

func Init() {

}
```

Yang pertama kita lakukan adalah mengambil lokasi directory kita saat ini menggunakan `os.Getwd()`. lalu kita cek apakan argumen perintah kita mengandung nama folder atau tidak. bila mengandung nama folder yang kita lakukan adalah mengabungkan path directory saat ini dengan nama folder tersebut mengunakan `path.Join()`

> pada perintah git init biasanya menggunakan `git init <nama folder>`

```
...

func Init(){
    pwd, err := os.Getwd()
    if err != nil {
        log.Println(err)
        return
    }
```

```
    foldernames := os.Args
    var foldername string
    if len(foldernames) > 2 {
        foldername = foldernames[2]
    }

    if foldername != "" && foldername != "." && foldername != ".." {
        pwd = path.Join(pwd, foldername)
    }
    ...
}
```

Selanjutnya kita membuat folder `.git`

```
func Init() {

    ...

    git_path := path.Join(pwd, ".gt")
    if err := os.Mkdir(git_path, 0755); err != nil {
        log.Println(err)
        return
    }

    ...
}
```

Untuk saat ini kita hanya akan membuat folder `objects` dan `refs` terlebih dahulu didalam folder `.git`

```
func Init() {

    ...

    var folders = []string{"objects", "refs"}

    for _, folder := range folders {
        if err := os.Mkdir(path.Join(git_path, folder), 0755); err != nil {
            log.Printf("%s: %v\n", git_path, err)
            return
        }
    }

}
```

Untuk full codenya akan jadi seperti ini

```
func Init() {
    pwd, err := os.Getwd()
    if err != nil {
        log.Println(err)
        return
    }

    foldernames := os.Args
    var foldername string
    if len(foldernames) > 2 {
        foldername = foldernames[2]
    }
```

```
    if foldername != "" && foldername != "." && foldername != ".." {
        pwd = path.Join(pwd, foldername)
    }

    git_path := path.Join(pwd, ".gt")
    if err := os.Mkdir(git_path, 0755); err != nil {
        log.Println(err)
        return
    }

    var folders = []string{"objects", "refs"}

    for _, folder := range folders {
        if err := os.Mkdir(path.Join(git_path, folder), 0755); err != nil {
            log.Printf("%s: %v\n", git_path, err)
            return
        }
    }

    log.Printf("Initialize empty gt repository in %s", git_path)
}
```