

Version Control System (Git)

Tugas ini dikerjakan di dalam kelompok kecil sebanyak 2 orang. Walaupun demikian, setiap commit harus dilakukan secara individu dan semua peserta akan mendapatkan pekerjaan yang sama. Pada tugas ini dilatih untuk memanfaatkan fitur git pada contoh kasus yang sederhana. Tapi nanti harapannya kalian pun akan dapat mengamplifikasikan pemakaian git ini pada pengerjaan project yang lebih besar. PERHATIAN: Praktikum ini membutuhkan koordinasi antar tim. Ikuti instruksi per langkah, dilarang mendahului nomor selanjutnya jika belum diminta.

Untuk pengerjaannya silahkan ikutilah instruksi berikut:

A. Warming up

1. Pada pengerjaan ini digunakan git dengan server dari <https://gitlab.com>. Jika anda sudah memiliki akun silakan login masing-masing, sedangkan jika belum mempunyai akun maka silakan register terlebih dahulu.
2. Anda bekerja berpasangan. Buatlah pasangan, yang nantinya akan direpresentasikan sebagai mahasiswa A dan mahasiswa B. Jika tidak mendapatkan pasangan hubungi dosen untuk membentuk kelompok 3 orang.
3. Buatlah sebuah project baru/repository dengan nama "rpl2024-git". Pilih menu project member yang dapat diakses melalui sidebar lalu pilih menu Manage-> Members.
4. Pada halaman members ini, tambahkan akun dengan role minimal Maintainer sebagai berikut:
 - a. Di repository milik A, tambahkan mahasiswa B sebagai maintainer.
 - b. Di repository milik B, tambahkan mahasiswa A sebagai maintainer. Jika kelompok 3 orang, maka yang ditambahkan sebagai maintainer adalah mahasiswa C.
 - c. Jika kelompok 3 orang, Di repository milik C, tambahkan mahasiswa A sebagai maintainer.
5. Tambahkan pula akun @raymondchandra sebagai maintainer supaya kami dapat memeriksa. Kelalaian melakukan langkah ini dapat berdampak hasil karya Anda tidak dinilai. Buka command prompt. Lalu masukkan perintah: `git clone <url_repo>`. Maka akan muncul folder sesuai yang dibuat (repo milik sendiri). Pada folder tersebut buatlah sebuah file "Calculator.java". File tersebut berisi kode berikut:

```
1 public class Calculator{
2
3 }
```

6. Lakukan staging dengan memasukkan perintah `git add "Calculator.java"` atau `git add .` (titik menandakan semua file yg berubah di direktori saat ini berada (".") dan turunannya).
7. Lakukan commit dengan memasukkan perintah `git commit -m "Create skeleton of Calculator class"`.
8. Lakukan push dengan perintah `git push origin main`.
9. Setelah mahasiswa lain selesai tahap 8, Lakukan clone di folder yang berbeda (repo mahasiswa lain yang sudah add anda sebagai maintainer, lihat instruksi nomor 4).
10. Pada repo mahasiswa lain ini bukanlah file Calculator lalu tambahkan implementasi kode method plus() sebagai berikut:

```
1 public class Calculator{
2     public int plus (int a, int b){
3         return a + b;
4     }
5 }
```

11. Lalu lakukan staging->commit->push di repo mahasiswa lain. Message commit harus merepresentasikan perubahan yang dilakukan.
12. Pindah ke repo sendiri, lakukan pull (`git pull origin main`). Seharusnya pada file Calculator, maka akan muncul hasil perubahan dari mahasiswa lain.
13. Anda juga dapat memeriksa dengan perintah `git log --oneline` untuk melihat semua commit yang sudah dilakukan.

B. Branch

1. Buatlah branch dengan perintah `git branch <nama_branch>` di kedua repo. Nama branch gunakan nama mahasiswa sendiri (khusus di pengerjaan tugas ini). Lalu pindah ke branch tersebut dengan perintah `git checkout <nama_branch>`.
2. Buka file Calculator.java (repo sendiri). Tambahkan kembali implementasi kode sebagai berikut:

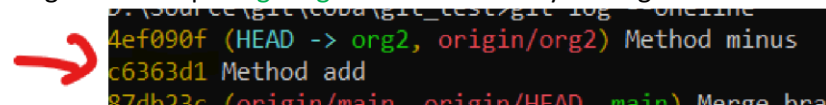
```

1 public class Calculator{
2     public int plus (int a, int b){
3         return a + b;
4     }
5
6     public int minus (int a, int b){
7         return a + b;
8     }
9 }

```

Lalu lakukan commit dan push. (perintah push pastikan di sesuai dengan nama branch nya, bukan origin main.

3. Kode pada nomor 2 ini sengaja salah pada method minus. Untuk memperbaiki kode ini terdapat 2 alternatif yaitu mengganti kode seperti biasa dan lakukan push kembali. Atau kita bisa mencoba untuk melakukan revert (salah satu keunggulan version control system). Pada kesempatan ini kita akan mencoba cara kedua.
4. Pertama-tama kita harus mengetahui titik commit revert terlebih dahulu. Kode commit didapatkan dengan melihat pada `git log --oneline` contohnya sebagai berikut:



```

4ef090f (HEAD -> org2, origin/org2) Method minus
c6363d1 Method add
87db23c (origin/main, origin/HEAD, main) Merge branch

```

5. Lakukan revert dengan perintah `git revert <commit>` (kode commit bisa berbeda-beda tiap repository). Maka anda akan kembali ke posisi sebelum dilakukan commit. Lalu lakukan fungsi push untuk memasitikan revert terjadi juga di server.
6. Sekarang perbaiki method minus dengan kode yang seharusnya. Lalu push kembali.
7. Pindah ke repo mahasiswa lain, (pastikan sudah di branch sendiri di repo lain) lalu buka file Calculator.java. Pada file ini hanya terdapat 1 fungsi plus saja. Tambahkan method multiply sebagai berikut:

```

1 public class Calculator{
2     public int plus (int a, int b){
3         return a + b;
4     }
5
6     public int multiply (int a, int b){
7         return a * b;
8     }
9 }

```

Lalu lakukan commit dan push.

8. Check kondisi saat ini. Jika sesuai instruksi maka:
 - a. Pada repo sendiri branch sendiri, file Calculator.java berisi method plus dan minus.

- b. Pada repo mahasiswa lain branch sendiri, file Calculator.java berisi method plus dan multiply.
 - c. Pada branch mahasiswa lain isinya adalah kebalikannya.
9. Pada keadaan seperti ini pasti akan terjadi conflict jika kedua branch tersebut digabungkan di branch main.
10. Buka halaman gitlab (repo sendiri). Pada sidebar terdapat menu Merge requests, lalu buat merge request baru. Kemudian anda diminta untuk memasukkan dari branch mana ke branch yang mana. Masukkan dari branch sendiri ke branch main.
11. Lalu lakukan approve dan merge pada requests ini.
12. Selanjutnya buka halaman gitlab (repo mahasiswa lain). Lalu buatlah merge requests yang sama. Tetapi kali ini proses merge dilakukan oleh mahasiswa lain sebagai pemilik repository tersebut.
13. Setelah mahasiswa lain melakukan merge request, bukanlah request tersebut di repo sendiri. Pada kali ini, ketika mau melakukan merge maka akan terjadi conflict. Lakukan resolve conflict agar pada file tersebut terdapat 3 buah method yaitu plus, minus, dan multiply.
14. Kembali ke command prompt, pindahkan ke branch main dengan fungsi `git checkout main`, lalu ambil kode paling baru dengan `git pull origin main`. Seharusnya file sudah terdiri dari 3 method tersebut.

C. Bonus: Git Squash (opsional)

Terkadang kita butuh menggabungkan beberapa commit ke dalam commit tertentu. Biasanya untuk kerapihan management oleh atasan. Misal terdapat 20 commit kecil yang dirasa kurang signifikan ke 1 commit saja. Lakukan instruksi berikut untuk melakukan hal tersebut.

1. Buatlah branch baru dengan nama squash dan checkout ke branch tersebut.
2. Buatlah file baru dengan nama CRUD.txt yang berisi teks "create". Lakukan commit dan push.
3. Edit file dengan menambahkan baris baru yaitu teks "read". Lakukan commit dan push.
4. Edit file dengan menambahkan baris baru yaitu teks "update". Lakukan commit dan push.
5. Edit file dengan menambahkan baris baru yaitu teks "delete". Lakukan commit dan push.
6. Jika sudah maka jika dipanggil git log akan muncul seperti gambar berikut:

```
4c96fc2 (HEAD -> squash, origin/squash) add delete
4c421ca add update
bb171da add read
386d670 add create
```

(Note: Kode commit mungkin berbeda)

7. 4 baris ini merepresentasikan 4 buah fitur versi disederhanakan. Jika dilihat dari log mungkin akan terasa commit nya terlalu banyak. Oleh karena itu kita akan melakukan git squash.
8. Fungsi git squash dilakukan dengan perintah `git rebase -i head~4`. Opsi -i menandakan membuka halaman interaktif, dan head~4 artinya mengambil 4 commit dari head (menyesuaikan kondisi di log).
9. Pada halaman interaktif akan keluar tampilan sebagai berikut:

```

pick 386d670 add create
pick bb171da add read
pick 4c421ca add update
pick 4c96fc2 add delete

# Rebase 6e9c972..4c96fc2 onto 6e9c972 (4 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                               commit's log message, unless -C is used, in which case
#                               keep only this commit's message; -c is same as -C but
#                               opens the editor
# x, exec <command> = run command (the rest of the line) using shell

```

10. Ubahlah perintah “pick” di read, update, delete menjadi “squash”. (yg pertama tetap pick)

```

pick 386d670 add create
squash bb171da add read
squash 4c421ca add update
squash 4c96fc2 add delete

```

11. Setelah selesai, untuk keluar dari menu ini ketikan tombol (esc) -> :wq -> (enter). Setelah keluar akan muncul menu untuk mengubah commit message. Silahkan ubah jika diperlukan (opsional). Contoh: disini dihapus commit message yang lain dan mengubah commit message keseluruhan menjadi “add crud”. Keluar dengan menekan (esc) -> :wq -> (enter).

```

# This is a combination of 4 commits.
# This is the 1st commit message:
add crud
# This is the commit message #2:
# This is the commit message #3:
# This is the commit message #4:
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#

```

12. Kembali ke branch squash dengan git checkout squash.
 13. Simpan perubahan ini dengan git push origin +squash. (+ menandakan force push)
 14. Lihat pada git log, seharusnya 4 commit tersebut sudah digantikan dengan 1 commit saja dengan message “add crud”.

Untuk mempermudah penilaian, kumpulkan ke assignment sebuah txt yang berisi:

1. Identitas diri
2. Link ke repository Anda
3. Link ke repository pasangan Anda (tunjukkan dengan jelas mana repo Anda dan mana repo pasangan Anda)