

LAPORAN PRAKTIKUM
MATA KULIAH DASAR PEMROGRAMAN

Dosen Pengampu : Triana Fatmawati, S.T, M.T

PERTEMUAN 2 : SISTEM VERSION CONTROL DAN KANBAR BOARD



Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

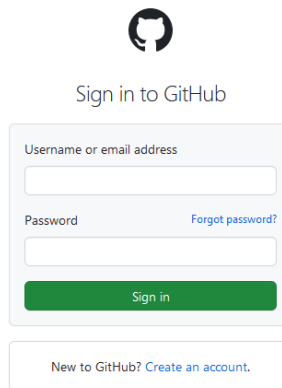
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

2023

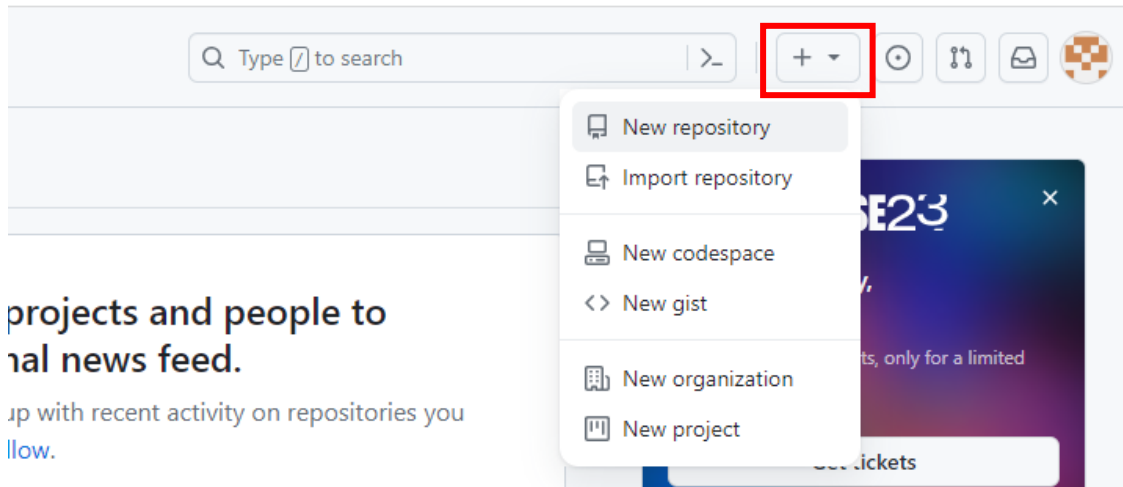
2.1 Percobaan 1 : Menggunakan Github

Berikut adalah hasil dari praktikum percobaan 1 : Menggunakan Github

1. Langkah pertama yaitu membuka situs <https://github.com>. Jika sudah, klik tombol **“Sign up”** untuk yang belum mempunyai akun dan membuat akun Github terlebih dahulu. Namun jika sudah mempunyai akun, langsung saja masukkan email dan password yang digunakan.



2. Setelah masuk ke akun Github masing-masing, klik tombol “+” yang berada di pojok kanan atas kemudian pilih menu “New Repository”.




3. Isi nama repository, deskripsi yang bersifat opsional, dan konfigurasi lainnya. Kemudian, kita dapat memilih untuk repository publik atau pribadi sesuai kebutuhan, jika sudah klik tombol “Create Repository”.

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Required fields are marked with an asterisk (*).


Owner * Repository name *

 yonandamayla /

✓ daspro-jobsheet2 is available.

Great repository names are short and memorable. Need inspiration? How about [reimagined-eureka](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

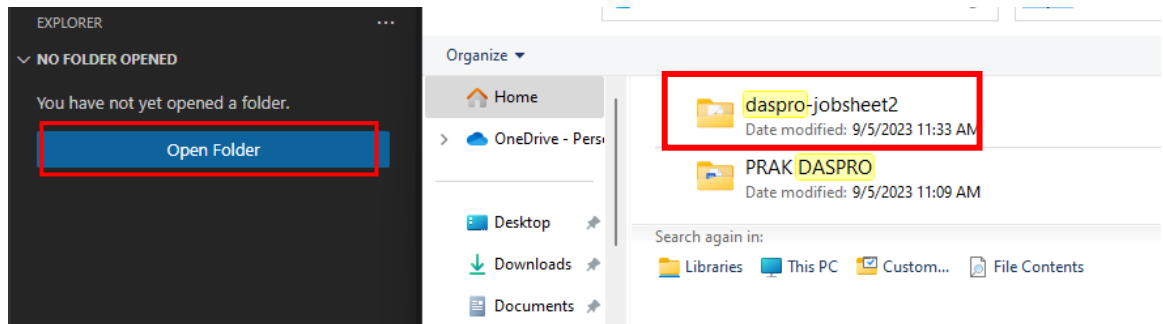
4. Untuk mengelola repository secara lokal, dibutuhkan instalasi Git client terlebih dahulu di <https://git-scm.com/downloads> dahulu untuk mengklonanya ke komputer. Jika sudah selesai, silahkan lakukan proses instalasi.
5. Gunakan perintah Git clone dari terminal CMD (Command Line) untuk mengklonkan repository. Perintah umumnya **git clone** <https://github.com/userame/nama-repository.git>.

```
Asus@LAPTOP-CM15586E MINGW64 /d/DASPRO
$ git clone https://github.com/yonandamayla/daspro-jobsheet2
Cloning into 'daspro-jobsheet2'...
warning: You appear to have cloned an empty repository.
```

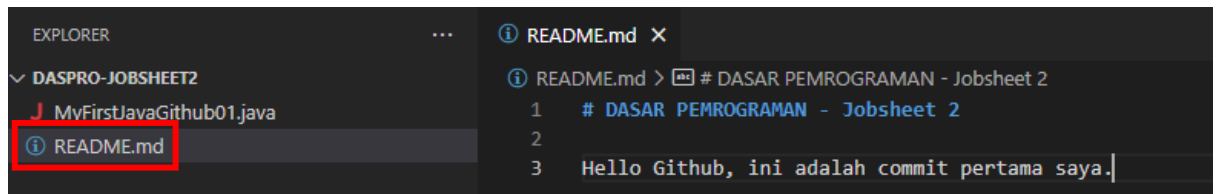
```
Asus@LAPTOP-CM15586E MINGW64 ~
$ dir
-1.14-windows.xml
AppData
Application\ Data
Contacts
```

```
Asus@LAPTOP-CM15586E MINGW64 ~
$ cd daspro-jobsheet2/
```

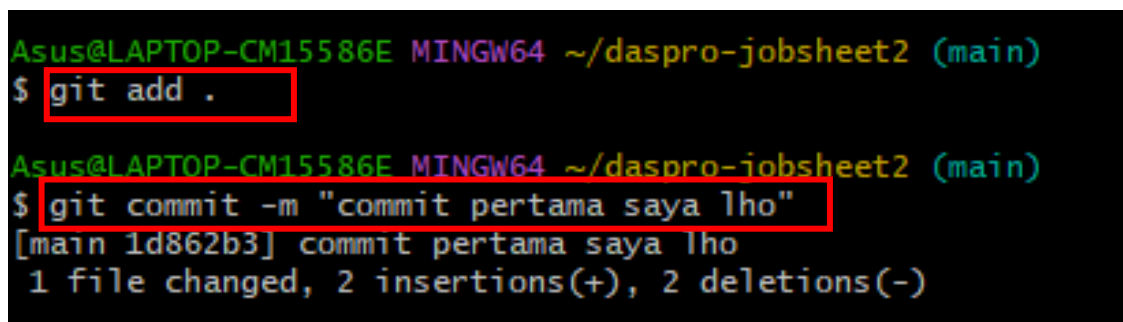
6. Selanjutnya, buka visual studio code untuk membuat atau mengedit berkas-berkas yang ada di dalam repository sesuai kebutuhan. Pilih menu open folder dan cari folder yang diinginkan.



7. Tambahkan file dengan **klik kanan** – **New File**, dan beri nama file tersebut dengan nama **“README.md”**. Isikan file **“README.md”** seperti berikut :



8. Jika sudah selesai, simpan perubahan pada visual studio code dan commit dengan perintah **git commit**. Setelah itu akan muncul commit yang menjelaskan perubahan yang telah dilakukan.



9. Untuk memperbarui repository di Github dengan perubahan yang dilakukan secara lokal, gunakan perintah **git push**, misalnya **git push origin nama-branch** akan mengirimkan perubahan branch di Github.
10. Buka akun Github, klik **akun – Settings – Developer Settings – Tokens (classic) – Generate new token (classic)**. Isikan bagian **Note**, **Expiration**, dan **Select scopes** sesuai gambar di bawah ini. jika sudah, klik tombol **Generate token**.

Note

What's this token for?

Expiration *

Custom... ▾

02/09/2024 📅 ▾

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

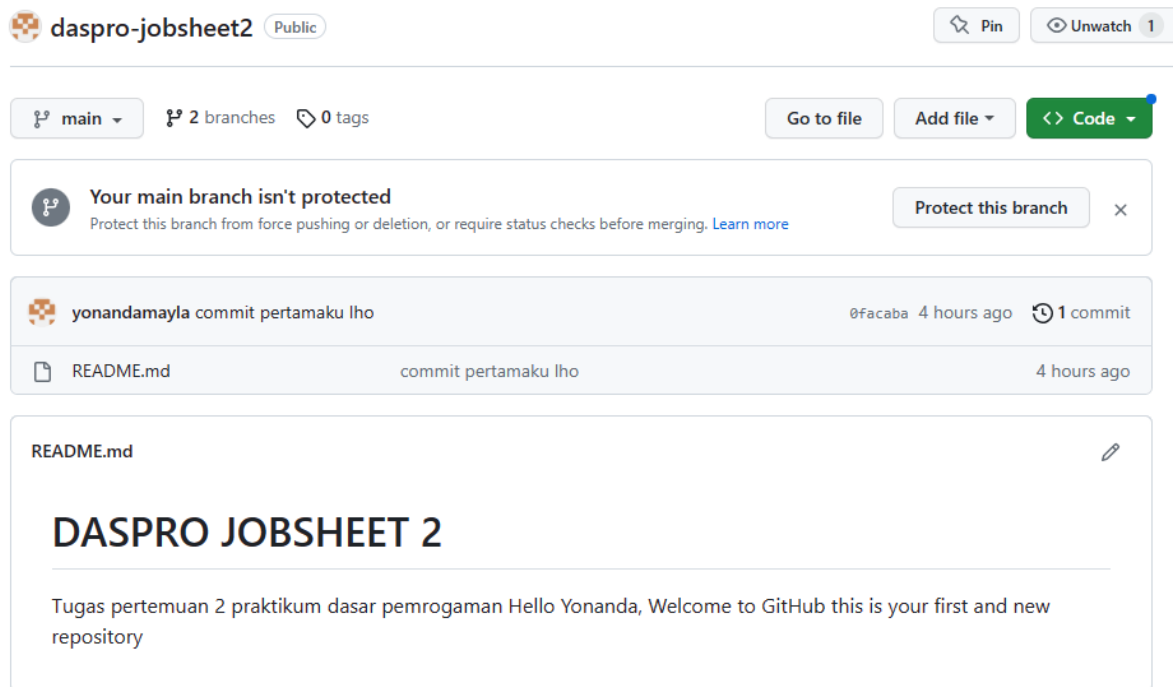
<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status

Simpan token tersebut karena tidak bisa dilihat Kembali untuk digunakan push yang selanjutnya.

11. Jalankan perintah git push.

```
Asus@LAPTOP-CM15586E MINGW64 /d/DASPRO/daspro-jobsheet2 (main)
$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/yonandamayla/daspro-jobsheet2
 * [new branch]      main -> main
```

12. Cek halamannya Github kita dan tampilannya akan seperti ini



The screenshot shows the GitHub interface for a repository named 'daspro-jobsheet2', which is public. At the top, there are buttons for 'Pin' and 'Unwatch' (with a count of 1). Below this, a navigation bar shows the 'main' branch selected, with '2 branches' and '0 tags' indicated. Action buttons include 'Go to file', 'Add file', and 'Code' (with a dropdown arrow). A notification banner states 'Your main branch isn't protected' and provides a 'Protect this branch' button. The commit history shows a single commit by 'yonandamayla' titled 'commit pertamaku lho', made 4 hours ago. Below the commit list, the 'README.md' file is displayed, containing the title 'DASPRO JOBSHEET 2' and a description: 'Tugas pertemuan 2 praktikum dasar pemrograman Hello Yonanda, Welcome to GitHub this is your first and new repository'.

daspro-jobsheet2 Public

Pin Unwatch 1

main 2 branches 0 tags

Go to file Add file Code

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#) Protect this branch

yonandamayla commit pertamaku lho @facaba 4 hours ago 1 commit

README.md commit pertamaku lho 4 hours ago

README.md

DASPRO JOBSHEET 2

Tugas pertemuan 2 praktikum dasar pemrograman Hello Yonanda, Welcome to GitHub this is your first and new repository

Pertanyaan

1. Jelaskan perbedaan perintah git commit dan git push?
2. Apakah bisa alurnya dibalik, membuat folder atau proyek terlebih dahulu kemudian upload (push) ke Github? Jika bisa, buktikan!

Jawaban :

1. Perbedaan perintah antara git commit dengan git push yaitu :

Git commit adalah perintah yang digunakan untuk menyimpan perubahan yang telah dibuat dalam repositori. Hal ini membuat snapshot (commit) baru dari kode kita dengan pesan komit yang menjelaskan perubahan yang dilakukan.

Perintah git commit hanya berlaku di dalam repositori lokal dan tidak akan memengaruhi repositori jarak jauh (remote repository) seperti GitHub. Hal ini merupakan langkah penting dalam proses pengembangan karena memungkinkan kita untuk mengelola sejarah perubahan kode kita di repositori lokal.

Git push adalah perintah yang digunakan untuk mengirimkan (mendorong) perubahan yang telah dikomit di repositori lokal ke repositori jarak jauh, seperti GitHub. Hal ini adalah tindakan yang diperlukan jika kita ingin berbagi atau menggabungkan perubahan kode dengan tim atau mengamankan kode di repositori yang dapat diakses secara online.

Git push hanya berfungsi setelah kita melakukan git commit, sehingga kode yang telah dikomit dapat disinkronkan dengan repositori jarak jauh.

2. Membuat folder atau proyek terlebih dahulu kemudian mengunggahnya (push) ke GitHub adalah memungkinkan. Langkah-langkahnya adalah sebagai berikut :
 - a) Buat folder atau proyek lokal di komputer dan inisialisasi repository Git di dalamnya (jika belum ada). Ketikkan perintah “**mkdir nama-proyek**”, “**cd nama-proyek**”, “**git init**”.


```

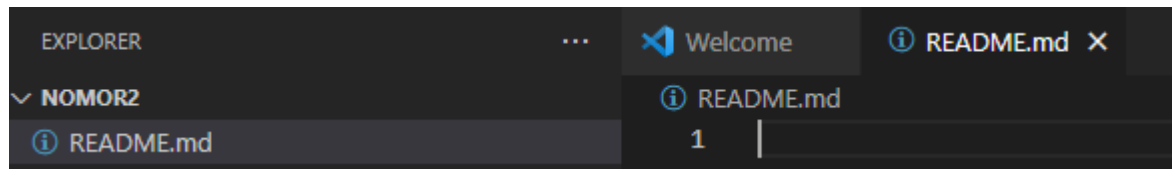
Asus@LAPTOP-CM15586E MINGW64 ~
$ mkdir nomor2

Asus@LAPTOP-CM15586E MINGW64 ~
$ cd nomor2

Asus@LAPTOP-CM15586E MINGW64 ~/nomor2
$ git init
Initialized empty Git repository in C:/Users/Asus/nomor2/.git/

```

- b) Tambahkan file-file atau direktori-direktori yang ingin di unggah ke dalam proyek. Dengan terlebih dahulu memasukkan folder nama-proyek yang sudah dibuat ke dalam visual studio code dan membuat file baru yang diberi nama dengan “**README.md**”



- c) Buat commit untuk perubahan-perubahan. Dengan memberikan mengetikkan **git add .** dan **git commit -m “Pesan commit kita”**


```

Asus@LAPTOP-CM15586E MINGW64 ~/nomor2 (master)
$ git add .

Asus@LAPTOP-CM15586E MINGW64 ~/nomor2 (master)
$ git commit -m "Pesan commit yonanda"
[master (root-commit) 9066f29] Pesan commit yonanda
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md

```


- d) Buat repository kosong di Github melalui antarmuka web Github.


Owner *  yonandamayla / Repository name *

✓ nomor2 is available.

Great repository names are short and memorable. Need inspiration? How about [probable-octo-carnival](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

- e) Tambahkan repository Github sebagai remote untuk repository lokal: **git remote add origin <https://github.com/username/nama-repositori.git>**. Ganti URL dengan dengan URL repositori Github yang sesungguhnya.

```
Asus@LAPTOP-CM15586E MINGW64 ~/nomor2 (master)
$ git remote add origin https://github.com/yonandamayla/nomor2.git
```

- f) Push perubahan Anda ke repositori GitHub: **git push -u origin master**

```
Asus@LAPTOP-CM15586E MINGW64 ~/nomor2 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 216 bytes | 216.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/yonandamayla/nomor2.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

Asus@LAPTOP-CM15586E MINGW64 ~/nomor2 (master)
$
```

2.2 Percobaan 2 : Dasar Kolaborasi di Github

Berikut adalah hasil dari praktikum percobaan 2 : Dasar Kolaborasi di Github

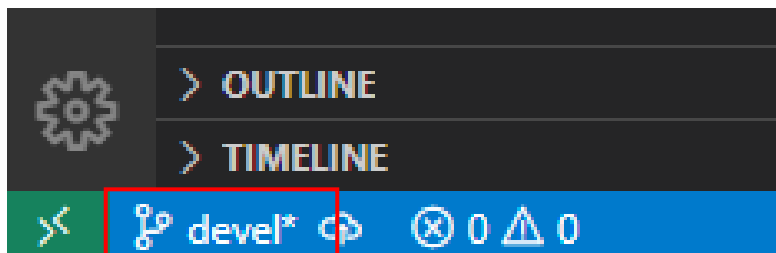
1. Gunakan perintah **git branch nama-branch** untuk membuat branch baru dan **git checkout nama-branch** untuk beralih ke branch tersebut.

```
Asus@LAPTOP-CM15586E MINGW64 /d/DASPRO/daspro-jobsheet2 (main)
$ git branch devel

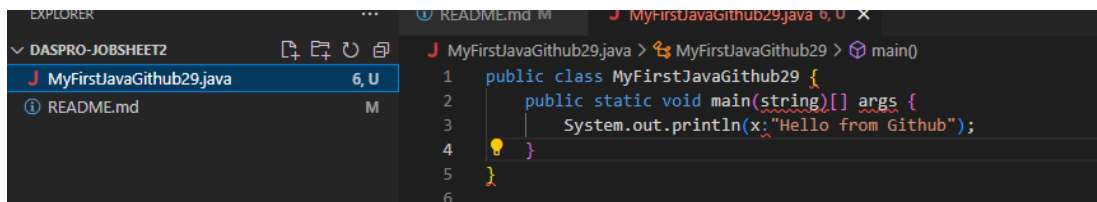
Asus@LAPTOP-CM15586E MINGW64 /d/DASPRO/daspro-jobsheet2 (main)
$ git check out devel
git: 'check' is not a git command. See 'git --help'.

The most similar command is
    checkout
```

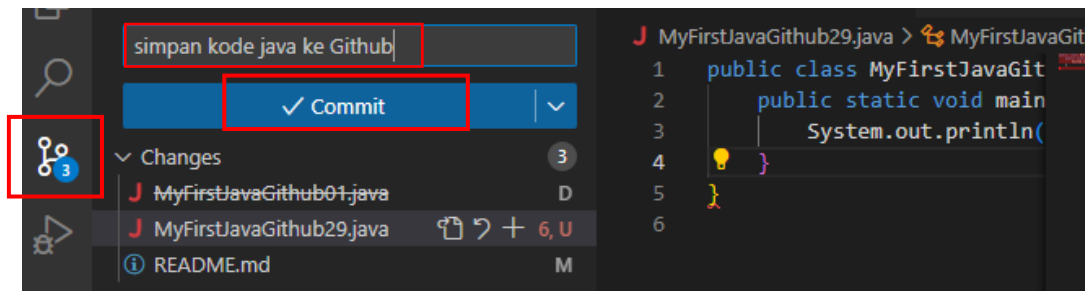
Kemudian pada visual studio code seharusnya akan berganti menjadi branch “**devel**”, jika belum silahkan klik kemudian pilih branch “**devel**” .



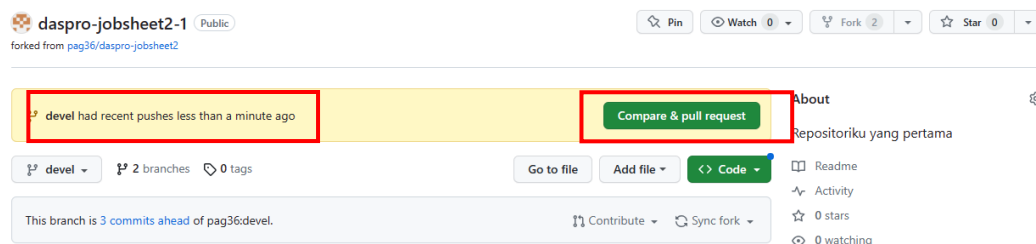
2. Dalam folder **daspro-jobsheet2**, buat file baru yang diberi nama dengan **MyFirstJavaGithub29.java** (sesuai dengan nomor absen). Dan jalankan kode program menggunakan Langkah-langkah pada jobsheet 1.



3. Simpan perubahan tersebut di local dengan cara **commit** kemudian **push** ke Github menggunakan visual studio code. Jangan lupa memberikan pesan Ketika akan melakukan **commit** . caranya adalah dengan meng **klik icon ranting – isikan pesan commit – klik tombol Commit – klik tombol Publish Branch.**



4. Buka halaman Github, seharusnya akan muncul **branch devel** yang bebrapa waktu telah di publish.



5. Selanjutnya, kita bisa membedakan antara **branch main** dan **branch devel**.

The screenshot shows the GitHub interface for the repository 'daspro-jobsheet2'. At the top, the repository name and 'Public' status are visible. Below this, the current branch is 'devel', which is 1 commit ahead of 'main'. A list of recent commits is shown, with the most recent one, 'MyFirstJavaGithub29.java', highlighted by a red box. The commit message is 'Simpan Java ke Github'.

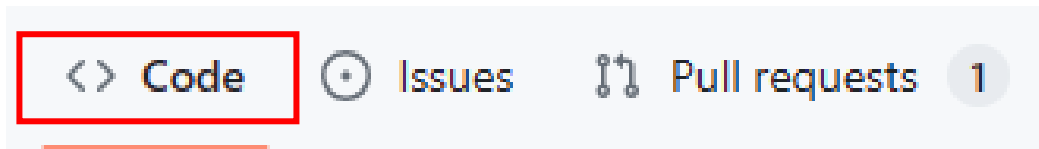
6. Klik tombol **Compare & pull request**, Anda dapat memilih branch mana yang akan digabungkan (devel ke master). Isikan pesan dan klik tombol **Create pull request**, tunggu beberapa saat kemudian klik tombol **Merge pull request**. Terakhir, klik tombol **Confirm merge**.

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the 'Open a pull request' form. At the top, the 'base repository' is 'pag36/daspro-jobsheet2' and the 'base' branch is 'devel'. The 'head repository' is 'yonandamayla/daspro-jobsheet2' and the 'compare' branch is 'devel'. A green checkmark indicates 'Able to merge'. Below this, the 'Devel' branch is selected. The 'Write' tab is active, and the commit message 'Menggabungkan branch devel dan branch master' is entered in the text area.

7. Pindah ke **tab Code**, kemudian amati hasil antara **branch main** dan **branch devel**.



Pertanyaan!

1. Jelaskan fungsi dari Pull requests!
2. Mengapa kita perlu membuat sebuah branch, manfaatnya apa?

Jawaban :

1. Fungsi dari **Pull Requests (PR)** adalah untuk memfasilitasi kolaborasi dalam pengembangan perangkat lunak, terutama dalam lingkungan pengembangan perangkat lunak berbasis Git, seperti GitHub atau GitLab. **PR** memungkinkan seorang pengembang untuk mengusulkan perubahan kode kepada proyek yang dikelola oleh orang lain atau tim. Berikut beberapa fungsi utama dari PR:
 - A) **Memungkinkan Review Kode.** PR memungkinkan orang lain untuk melihat, mengevaluasi, dan memberikan umpan balik terhadap perubahan kode yang diajukan. Ini membantu untuk meningkatkan kualitas kode dan menghindari kesalahan atau bug yang tidak terdeteksi.
 - B) **Memisahkan Pengembangan.** PR memungkinkan pengembang membuat cabang (branch) khusus untuk pekerjaan mereka. Ini memungkinkan mereka untuk bekerja secara terisolasi tanpa mengganggu kode utama dan mencegah konflik yang tidak diinginkan.
 - C) **Pemantauan Proses.** PR dapat digunakan untuk melacak status dan progres perubahan kode. Ini mencakup diskusi, penambahan atau perubahan kode, serta tindak lanjut terkait perubahan tersebut.

D) **Penggabungan Kode.** Setelah perubahan kode dianggap selesai dan telah melewati proses review, PR dapat digabungkan (merged) ke dalam cabang utama (biasanya disebut sebagai "master" atau "main"). Ini memperbarui kode utama dengan perubahan yang diajukan.

2. Membuat branch adalah praktik umum dalam pengembangan perangkat lunak berbasis Git, dan ini memiliki beberapa manfaat utama, seperti :

A) **Solasi Pekerjaan.** Branch memungkinkan pengembang untuk bekerja pada fitur, perbaikan bug, atau perubahan lainnya dalam lingkungan yang terisolasi. Ini berarti perubahan yang dilakukan di branch tidak akan langsung memengaruhi kode di cabang utama (biasanya disebut sebagai "master" atau "main"). Ini membantu mencegah gangguan dan konflik dengan perubahan yang sedang dilakukan oleh orang lain.

B) **Pemantauan Perubahan.** Setiap branch memiliki riwayat perubahan yang terpisah. Ini memungkinkan pengembang untuk melacak dan memahami perubahan yang telah mereka buat, serta untuk memahami sejarah perubahan pada proyek.

C) **Kolaborasi yang Lebih Baik.** Dengan branch, beberapa pengembang dapat bekerja secara bersamaan pada berbagai fitur atau perbaikan. Mereka dapat membuat branch mereka sendiri, mengerjakan tugas mereka, dan kemudian menggabungkan perubahan mereka ke dalam cabang utama melalui PR, yang memudahkan kolaborasi dalam tim.

D) **Pengujian Terpisah.** Pengembang dapat menguji perubahan di branch mereka sendiri sebelum menggabungkannya ke dalam cabang utama. Ini memungkinkan untuk mengidentifikasi masalah atau bug sebelum perubahan tersebut memengaruhi kode utama.

- E) **Rollback Mudah.** Jika ada masalah dengan perubahan yang diajukan di branch, pengembang dapat membatalkan atau menghapus branch tersebut tanpa memengaruhi kode utama.

Dengan demikian, membuat branch adalah praktik yang penting dalam pengembangan perangkat lunak yang membantu mengelola kode sumber secara efisien dan memfasilitasi kolaborasi tim.