

## MODUL 2 : SDLC DAN GIT

**Waktu : 120 menit**

### Git

Git adalah sistem kontrol versional yang terdistribusi dan berjalan secara lokal pada perangkat pengembangan. Git digunakan untuk *tracking* perubahan pada proses pengembangan perangkat lunak untuk *reproductibility* dan kolaborasi. Implementasi Git pada pengembangan berkelompok maupun individu sangat penting, Git sebagai sebuah sistem untuk *tracking* perubahan memungkinkan pengembangan untuk kembali ke iterasi sebelumnya. *Remote host* dapat digunakan untuk meng-host file dan history pengembangan produk, Git akan digunakan untuk *push* perubahan lokal ke *remote host* dan *pull* perubahan dari kontributor lain dari *remote host*.

### Github Action

Github Action adalah fitur otomatisasi yang terdapat Github. Github Actions dapat digunakan untuk melakukan Testing, Continuous Integration, Continuous Deployment, dan *workflow* lain yang dapat dilakukan otomatisasi. Otomatisasi ini berguna untuk memastikan bahwa aplikasi yang dibuat telah lolos pengujian, sehingga tidak ada kesalahan *build* yang masuk ke *production environment* selain itu mengurangi kemungkinan kesalahan manusia saat melakukan integrasi dan *deployment*. Github Action dapat dikombinasikan dan dikonfigurasi sesuai kebutuhan. Github Action terdiri atas:

- Workflows: adalah proses otomatisasi yang dapat dikonfigurasi. Workflow dikonfigurasi dalam file *.yaml* yang terdapat pada *repository*. Pada satu *repository* dapat terdapat lebih dari satu Workflows
- Events: adalah aktivitas tertentu yang terjadi pada *repository* yang menyebabkan Workflows dieksekusi.
- Jobs: adalah sekumpulan langkah-langkah yang terdapat pada Workflows yang akan dieksekusi oleh Runner di Server.
- Actions: adalah aplikasi custom pada Github Action yang dapat digunakan untuk melakukan perintah kompleks yang sering dilakukan dan berulang.
- Runners: adalah server yang menjalankan Workflows ketika Workflows terpicu oleh suatu Event.

### Github Project

Github Project adalah *project management tools* yang terdapat *built-in* pada Github. Github Project dapat digunakan untuk melakukan perencanaan dan *tracking* pekerjaan. Github Project didasarkan pada "Issues" dan "Pull Requests". Informasi antara Github Project, Issues, dan Pull Requests dapat saling terintegrasi dan sinkronisasi. Github Project dapat dikustomisasi menurut tampilan, filtering, dan pengkelompokan.

### Software Development Life Cycle (SDLC)

SDLC adalah sebuah model manajemen proyek, SDLC menyediakan alur sistematis yang secara garis besar terdiri atas 7 fase untuk membantu proses pengembangan agar menghasilkan produk akhir yang

memiliki kualitas terbaik tetapi tetap efisien dalam konsumsi sumber daya pada proses pengembangannya.

Ketujuh fase dari SDLC adalah:

1. Planning  
Mempersiapkan proyek yang akan dikembangkan dengan memperhatikan ketersediaan sumber daya mulai dari tenaga, waktu, dan biaya. Produk apa yang akan dikembangkan perlu didefinisikan dengan baik pada tahap ini.
2. Requirement analysis  
Menganalisis berbagai aspek yang mempengaruhi produk untuk mengidentifikasi permasalahan dan kebutuhan yang ada. *Feasibility* dari proyek perlu dipastikan dengan melakukan *risk assessment* dan rencana mitigasi dari *risk* yang ditemukan.
3. Software design  
Mengubah spesifikasi produk menjadi spesifikasi desain produk yang dapat digambarkan melalui berbagai bagan dan ilustrasi. Rancangan umum dari detail keseluruhan produk harus dipersiapkan terlebih dahulu pada fase ini agar pengembangan bisa lebih terarah dan efisien.
4. Software development  
Tahapan pengembangan produk sesuai dengan rancangan yang dibuat.
5. Testing  
Menguji produk yang dikembangkan untuk memperbaiki kekurangan dan kesalahan yang masih ada, terdapat berbagai metrik dan *framework* yang dapat digunakan sebagai acuan untuk melakukan pengujian kualitas produk.
6. Deployment  
Men-*deploy* produk agar dapat mulai digunakan
7. Maintenance  
Setelah produk diluncurkan, pengembang harus terus memantau dan memperbaiki produk apabila ditemukan bug atau error.

Beberapa model SDLC yang sering digunakan antara lain adalah *waterfall* dan *agile*.

**Waterfall model** adalah model SDLC tertua dengan pengerjaan fase yang dilakukan secara linear, dimana satu fase akan dikerjakan sampai selesai terlebih dahulu baru memulai pengerjaan fase selanjutnya. Setiap fase umumnya akan memiliki rencanan pengerjaannya sendiri yang menggunakan hasil dari fase sebelumnya. Keunggulan dari model *waterfall* adalah mudah dipahami dan digunakan, pencapaian dan target yang terdefinisi dengan jelas dan *requirement* terdefinisi dengan stabil. Kelemahan dari model ini adalah pengerjaan harus runtut dan detail kecil yang tertinggal dari fase-fase sebelumnya dapat menghambat seluruh fase setelahnya, model *waterfall* juga tidak terlalu fleksibel apabila terdapat perubahan pada *requirement*, dan produk baru dapat dicoba pada tahap paling terakhir.

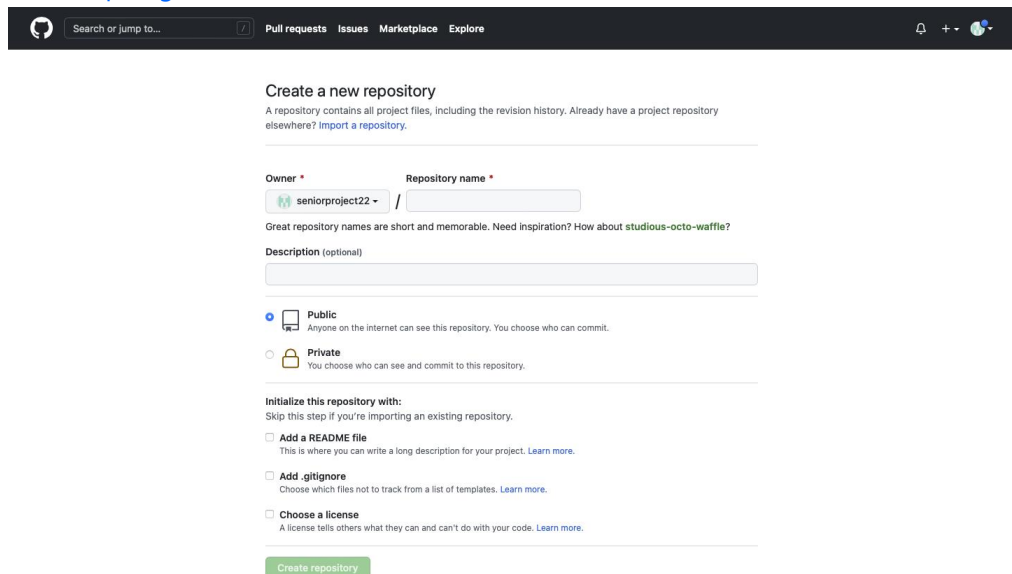
**Agile model** adalah model SDLC yang memisahkan pengembangan produk ke dalam beberapa *cycle* agar dapat dengan segera memberikan produk yang sudah fungsional. Produk hasil dari metodologi agile umumnya akan terdiri atas beberapa kali perilsan, dengan pengujian pada setiap perilsan produk akan digunakan sebagai acuan pengembangan berikutnya. Keunggulan dari model *agile* adalah mudah untuk mengakomodasi perubahan pada *requirement* dan produk fungsional dapat diuji pada tahap-

tahap awal. Model *agile* kurang cocok untuk diimplementasikan pada proyek dengan dependensi yang kompleks.

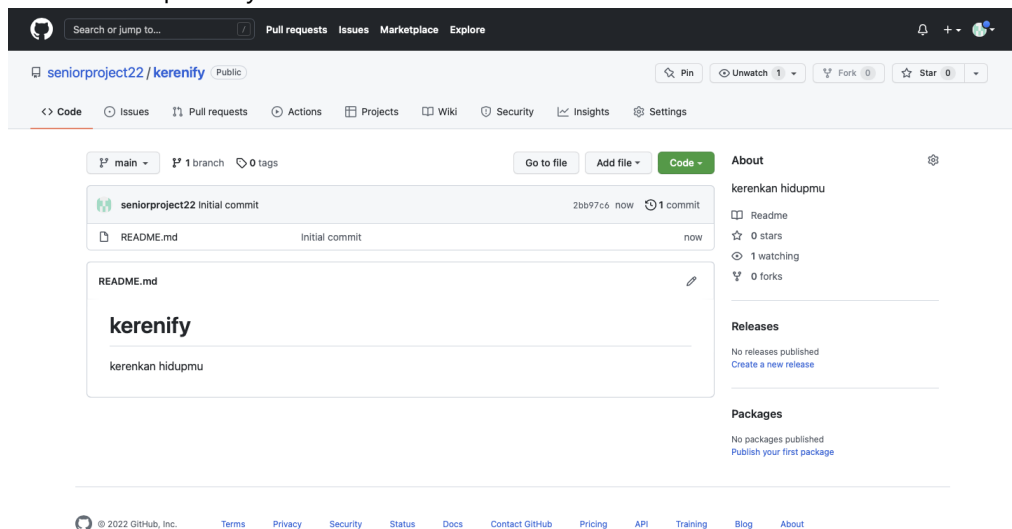
## LAB 2.1: REPOSITORY KELOMPOK

Lab 2.1 dikerjakan secara individu dengan berkoordinasi dalam satu kelompoknya. Setiap individu mengumpulkan laporan praktikum Lab 2.1 pada **Modul 2.1 – GitHub**.

1. Membuat akun GitHub apabila belum memiliki
2. Salah satu anggota membuat repository Git untuk kelompok, pastikan repository yang dibuat memiliki pengaturan visibilitas untuk publik dan gunakan nama produk untuk nama repository
  - a. Login ke akun GitHub
  - b. Buka <https://github.com/new>



- c. Gunakan nama produk kelompok Anda sebagai nama repository
- d. Tambahkan deskripsi singkat mengenai produk kelompok Anda
- e. Pilih akses untuk Public
- f. Centang Add a README file
- g. Klik Create repository

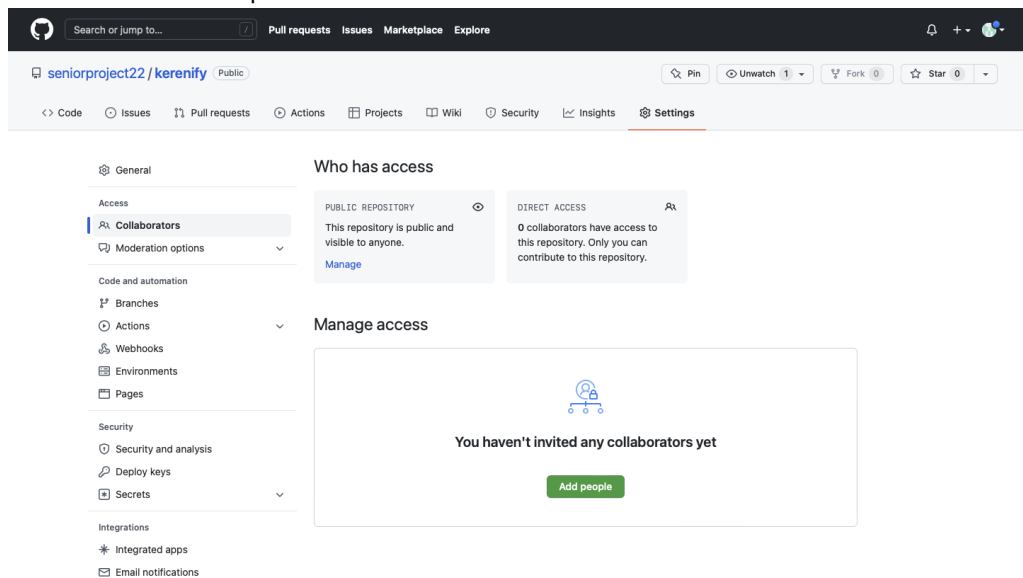


3. Edit isi file README.md menjadi:

```
#KERENIFY
Kerenkan hidupmu

Kelompok Keren
Ketua Kelompok:
Anggota 1:
Anggota 2:
Anggota 3:
Anggota 4:
```

4. Invite semua anggota lainnya ke repository tersebut
- Tab Settings
  - Menu Access > Collaborators
  - Klik tombol Add People



- Invite anggota lain dalam kelompok
  - Setiap anggota Accept invitation
5. Setiap anggota SET UP Git pada perangkat masing-masing

```
#Mengatur kredensial
git config --global user.name <USERNAME>
git config --global user.email <EMAIL>

#Cek kredensial yang diatur
Git config --global user.name
Git config --global user.email

#Membuat working directory
mkdir <NAMA_APLIKASI>
cd < NAMA_APLIKASI >
```

6. Setiap anggota CLONE repository kelompok

```
#Clone repository kelompok
```

```
git clone <URL_REMOTE_REPO> <PATH_KE_LOCAL_DIRECTORY>
```

<URL\_REMOTE\_REPO> adalah link repository kelompok yang telah dibuat

<PATH\_KE\_PROJECT\_DIRECTORY> adalah local directory masing-masing

7. Setiap anggota CREATE NEW BRANCH dan beri nama branch yaitu NIU masing-masing

```
#Membuat branch baru
git checkout -b <NIU>

#Lihat daftar branch
git branch

#Pindah ke branch masing-masing
git checkout <NAMA_BRANCH>
```

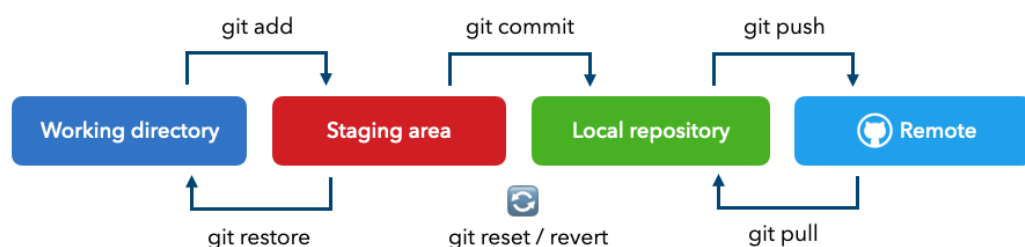
8. Setiap anggota mengedit file README.md pada branch masing-masing dengan menambahkan format: <NAMA LENGKAP–NIM LENGKAP> pada posisi anggota yang sesuai (urutkan berdasarkan NIM anggota dalam kelompok), perhatikan baris yang akan diubah oleh setiap anggota agar tidak muncul conflict saat PULL REQUEST.
9. Setelah melakukan perubahan, simpan file.
10. Jalan kan perintah berikut:

```
#Menambahkan seluruh perubahan ke staging Github
git add .

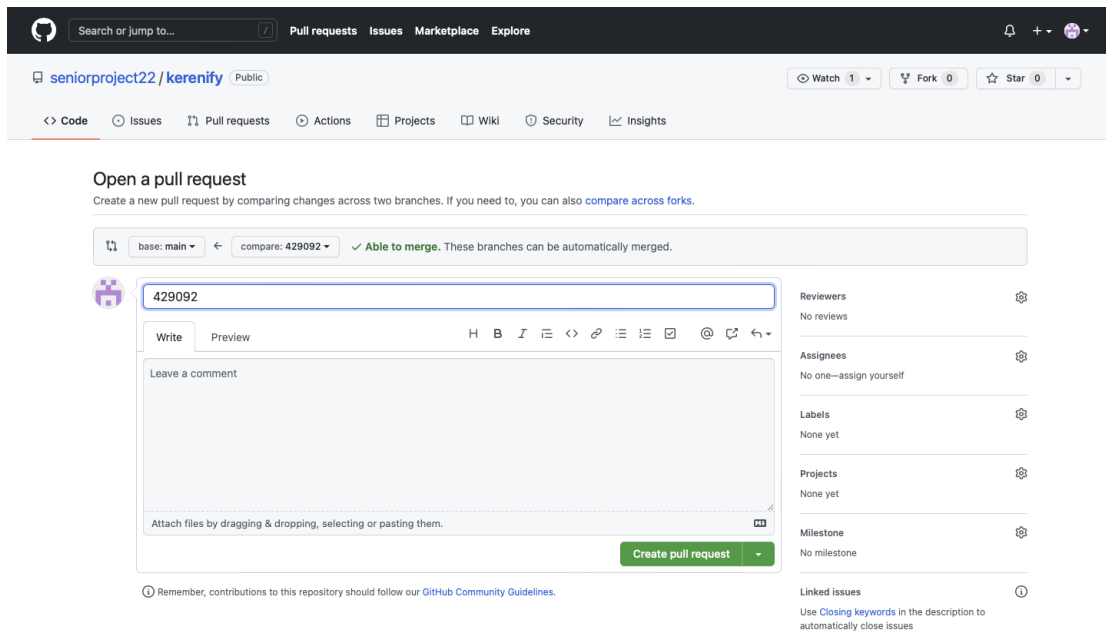
#Membuat commit dan pesan commit
git commit -m "<<isi pesan commit>>"
```

11. PUSH branch masing-masing ke repository GitHub kelompok

```
#Melakukan push dari lokal ke repository Github
git push
```



12. Buka Repository Github pada Browser. Masuk ke tab menu Pull Request. Buat Pull Request dari branch setiap anggota ke branch utama,



13. MERGE PULL REQUEST (jangan delete branch setelah merge)

14. Hasil akhir README setiap kelompok akan tampak seperti:

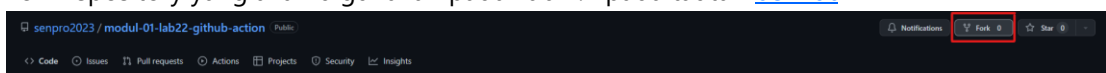
```
#KERENIFY
Kerenkan hidupmu

Kelompok Keren
Ketua Kelompok: ANDI - 19/123456/TK/12345
Anggota 1: ANDI - 19/000001/TK/00001
Anggota 2: ANDI - 19/000002/TK/00002
Anggota 3: ANDI - 19/000003/TK/00003
Anggota 4: ANDI - 19/000004/TK/00004
```

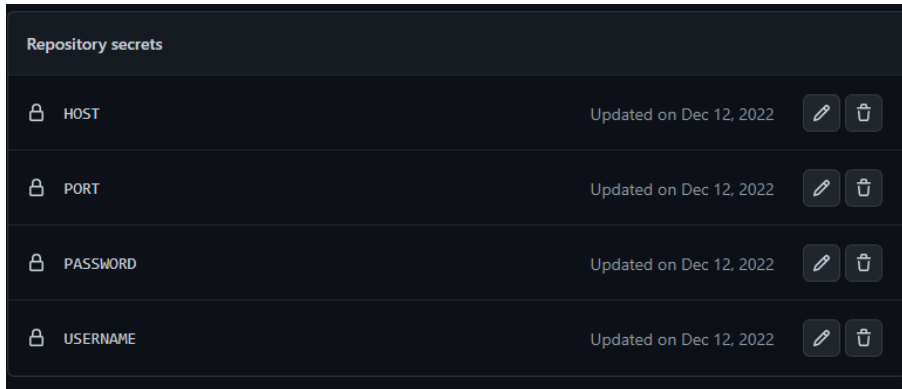
### LAB 2.2: GITHUB ACTION

Lab 2.2 dikerjakan secara individu. Pastikan Git sudah terinstal pada PC. Jika belum lakukan instalasi dengan mengunduh Git for Windows pada tautan [berikut](#). Pelaksanaan Lab 2.2 akan dibantu dengan menggunakan repository yang telah disediakan, berupa web berbasis NextJS. Tidak perlu khawatir jika belum pernah menggunakan NextJS, hanya pemahaman minimum mengenai HTML dan kemampuan membaca yang diperlukan.

1. Fork repository yang akan digunakan pada Lab 2.2 pada tautan [berikut](#)



2. Setelah berhasil melakukan fork. Masuk ke bagian Setting, kemudian klik Secrets and Variables. Pilih menu Actions.
3. Buat Secret baru dengan mengklik "New repository secret". Terdapat 4 secret yang perlu dibuat, yaitu HOST, PORT, USERNAME, dan PASSWORD. Dapatkan nilai untuk masing-masing secret dari Asisten Praktikum. Setelah keempat secret berhasil dibuat, **screenshot** dan **letakkan pada worksheet**.



4. Clone **repository hasil fork** pada komputer lab. Pastikan telah login ke Github.

**Jika Clone Gagal:**

- Buka Git Bash
- Jalankan perintah berikut

```
cd ~
ssh-keygen -t ed25519 -f ~/.ssh/ssh-<<NIU>>
# akan muncul prompt untuk memasukkan password. Isi sesuai
kebutuhan dapat dikosongi

cat ~/.ssh/ssh-<<NIU>>.pub
```

- Copy keluaran dari perintah terakhir. Buka Github. Login ke akun yang digunakan untuk fork repository. Masuk ke Setting → SSH and GPG Keys. Klik "New SSH". Isi "Title". Pada bagian 'Key', paste hasil copy key tadi.
- Jalankan perintah berikut

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/ssh-<<NIU>>
```

- Clone ulang **repository hasil fork** menggunakan pilihan SSH.
- Pada root folder hasil clone, buat folder baru ".github". Perhatikan titik diawal nama folder.
  - Masuk ke folder ".github", buat folder "workflows". Perhatikan penulisan "workflows" menggunakan huruf "s"
  - Masuk ke folder "workflows", buat file baru dengan nama "main.yml"

**Case 1: Jobs untuk Test Build**

- Pada file "main.yml", tuliskan kode berikut. Perhatikan indentasi dan tanda yang digunakan saat menulis script "main.yml". **Screenshot** file "main.yml" dan **letakkan pada Worksheet**. Perhatikan indentasi saat penulisan file.

```
name: Test, Build, and Deploy | Modul 01 - Lab2.2 Senior Project

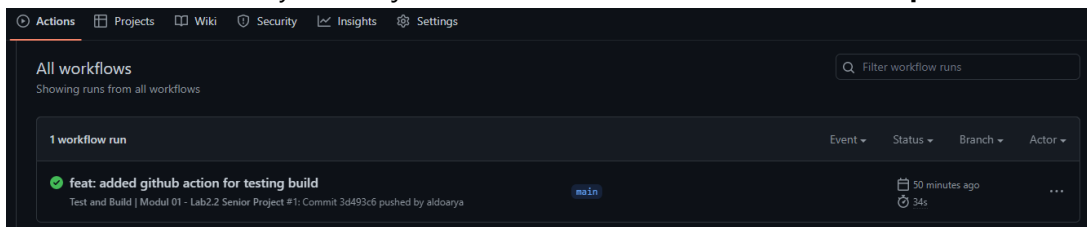
on:
  push:
    branches: [main]

jobs:
  test-build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [16.x]

    steps:
      - uses: actions/checkout@v2
      - name: Testing Build pre-Deploy
        uses: actions/setup-node@v2
        with:
          node-version: ${ matrix.node-version }
          cache: "npm"
      - run: npm i
      - run: npm run build
```

9. Save file "main.yml". Commit dan Push perubahan ke repository hasil fork.
10. Buka laman repository hasil fork pada Github, kemudian masuk ke Tab "Action". Hasil eksekusi Github Action seharusnya menunjukkan "Success". **Screenshot dan letakkan pada worksheet.**



### Case 2: Jobs untuk Terhubung ke VM

11. Buka Github. Login ke akun yang digunakan untuk fork repository. Masuk ke Setting → SSH and GPG Keys. Klik "New SSH". Pada Title dan Key isikan nilai **yang diberikan oleh Asisten Praktikum**.
12. Kembali ke Visual Studio Code. Lengkapi file "main.yml" menjadi:

```
name: Test, Build, and Deploy | Modul 01 - Lab2.2 Senior Project

on:
  push:
    branches: [main]

jobs:
  test-build:
    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [16.x]

    steps:
      - uses: actions/checkout@v2
      - name: Testing Build pre-Deploy
        uses: actions/setup-node@v2
        with:
          node-version: ${ matrix.node-version }
          cache: "npm"
      - run: npm i
      - run: npm run build
```



```
deploy:
  needs: test-build
  runs-on: ubuntu-latest

strategy:
  matrix:
    node-version: [16.x]

steps:
  - name: Build app on VM
    uses: appleboy/ssh-action@master
    with:
      host: ${ secrets.HOST }}
      username: ${ secrets.USERNAME }}
      password: ${ secrets.PASSWORD }}
      port: ${ secrets.PORT }}
      script: |
        eval "$(ssh-agent -s)"
        ssh-add ~/.ssh/⟨NAMA_KEY_DARI_ASPRAK⟩
        echo "Cek folder project";
        [ ! -d "${HOME}/senpro/⟨NIU⟩/modul02/senpro-github-action/" ] &&
        {
          echo "Repository belum di-clone. Cloning...";
          mkdir -p ~/senpro/⟨NIU⟩/modul02;
          cd ~/senpro/⟨NIU⟩/modul02;
          git clone <<LINK_REPOSITORY_FORK>>;
          cd ~/senpro/⟨NIU⟩/modul02/senpro-github-action;
          echo "Install Package dan Build Project";
          npm install;
          npm run build;
        } ||
        {
          echo "Repository sudah ada. Building...";
          cd ~/senpro/⟨NIU⟩/modul02/senpro-github-action;
          git restore .;
          git pull origin main;
          echo "Install Package dan Build Project";
          npm install;
          npm run build;
        }
```

13. **Screenshot** dan **letakkan pada worksheet**
14. Simpan perubahan, commit, dan push perubahan ke repository hasil fork.
15. Pastikan Github Action berjalan dan sukses. **Screenshot** dan **letakkan pada worksheet**
16. Buka Git Bash. Jalankan perintah dibawah

```
ssh <<USERNAME>>@<<HOST>>
cd ~/senpro/⟨NIU⟩/modul02/⟨Nama repository hasil fork>>
cat src/pages/index.js
```

17. **Screenshot** keluaran dan **letakkan pada worksheet**

#### **Case 3: Test Build Gagal**

18. Buka folder repository hasil fork yang telah di clone dengan text editor pilihan (Visual Studio Code)
19. Lakukan perubahan pada "src/pages/index.js".
20. Comment line 1 hingga 4
21. Ubah Nama dan NIM dengan nilai yang sesuai
22. Simpan, commit, dan push perubahan
23. Tunggu hingga Github Action selesai berjalan. **Screenshot** hasil Github Action setelah dilakukan perbaikan. **Letakkan pada Worksheet.**
24. Ulangi langkah 16 dan 17.

#### **Case 4: Perbaikan Build Error**

25. Uncomment line 1 hingga 4
26. Simpan, commit, dan push perubahan

27. Tunggu hingga Github Action selesai berjalan. **Screenshot** hasil Github Action setelah dilakukan perbaikan. **Letakkan pada Worksheet.**
28. Ulangi langkah 16 dan 17.
29. Jawab pertanyaan mengenai Lab 2.2 pada worksheet yang disediakan.

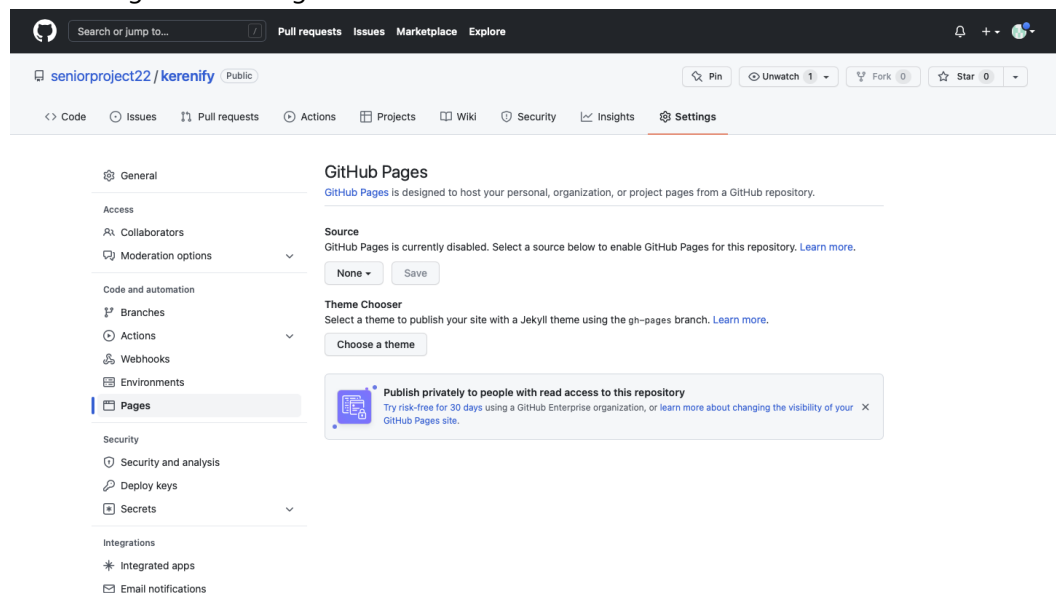
**Note:** Implementasikan Github Action pada proyek yang dikembangkan!

## LAB 2.3: GITHUB PAGE KELOMPOK

GitHub Page dapat digunakan untuk menampilkan repository GitHub menjadi sebuah website statis. Pada Lab 2.3, setiap kelompok membuat sebuah GitHub Page untuk menampilkan hasil pengerjaan proyek.

1. Membuat GitHub Page dari repository kelompok

- a. Tab Settings > Menu Pages



- b. Pilih source yaitu main (branch utama) dan folder /docs
- c. Klik "Choose a theme" dan pilih salah satu dari tema yang tersedia, klik "Select Theme"
2. Gunakan nama produk sebagai judul halaman
- a. Buka \_config.yml dan edit isinya menjadi

```
theme: <sesuai dengan tema yang sudah dipilih>
title: <NAMA APLIKASI>
description: <DESKRIPSI APLIKASI>
```

3. Tampilkan konten berikut pada GitHub Page kelompok dengan mengedit kontennya pada file index.md
- a. Nama Kelompok
- b. Anggota dan NIM Kelompok
- c. "Project Senior Project TI"
- d. Instansi (Departemen Teknologi Elektro dan Teknologi Informasi, Fakultas Teknik, Universitas Gadjah Mada)

- e. Jawaban dari Modul 1 (Nama produk, jenis produk, latar belakang & permasalahan, ide solusi, dan analisis kompetitor)

## LAB 2.4: MERANCANG SDLC PENGEMBANGAN PRODUK

Lab 2.4 dikerjakan secara berkelompok.

1. Menentukan metodologi SDLC yang akan diimplementasikan oleh kelompok beserta alasan pemilihannya
2. Melakukan tahap 1-3 dari SDLC dengan merancang:
  - a. Tujuan dari produk
  - b. Pengguna potensial dari produk dan kebutuhan para pengguna tersebut
  - c. Use case diagram
  - d. Functional requirements untuk use case yang telah dirancang
  - e. Entity relationship diagram
  - f. Low-fidelity Wireframe
  - g. Gantt-Chart pengerjaan proyek dalam kurun waktu 1 semester
3. Tampilkan hasil pengerjaan Lab 2.3 pada GitHub Page milik kelompok
4. Kumpulkan link GitHub Repository dan GitHub Page pada *google form* yang disediakan

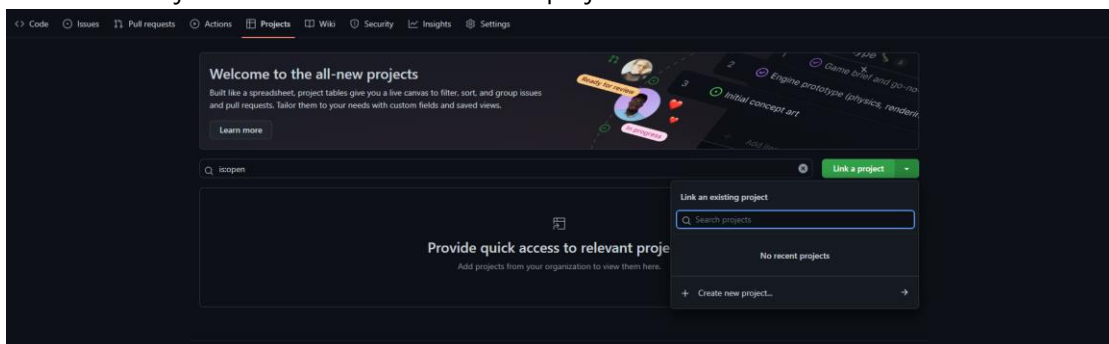
## LAB 2.5: PEMANFAATAN PROJECT MANAGEMENT TOOLS

Lab 2.5 dikerjakan secara berkelompok.

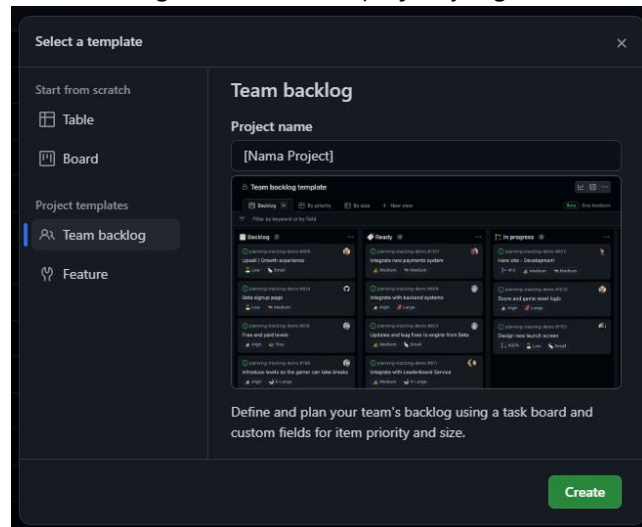
1. Breakdown project yang telah direncanakan untuk dikembangkan sebagai Proyek Senior. Pastikan breakdown yang dilakukan mampu mengubah proyek besar menjadi subproyek atau fase yang lebih kecil dan lebih feasible untuk diselesaikan.
2. Buat detail langkah-langkah atau hal yang harus dilakukan untuk mampu menyelesaikan masing-masing subproyek.
3. Tuliskan breakdown yang telah dilakukan pada worksheet kelompok.
4. Buka kembali repository yang telah dibuat sebelumnya
5. Klik pada Tab 'Projects'



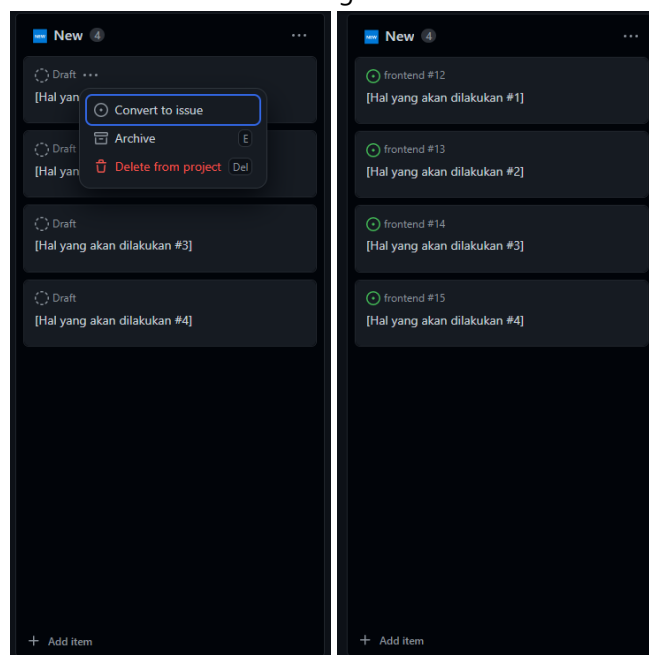
6. Klik 'Link a Project' kemudian klik 'Create new project'



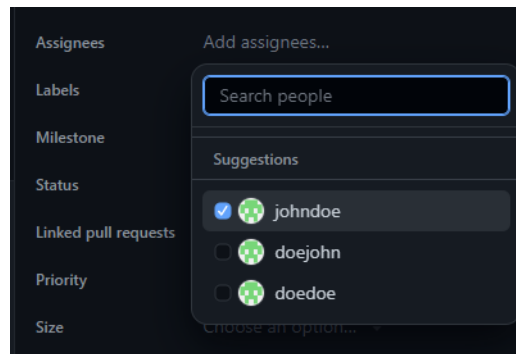
- Gunakan template 'Team Backlog' dan beri nama project yang sesuai. Klik Create.



- Pada kolom 'New'. Klik "+ Add item" kemudian isikan hal yang akan dilakukan untuk menyelesaikan project. Lakukan langkah ini beberapa kali **untuk seluruh langkah-langkah** yang didefinisikan saat breakdown proyek.
- Ubah seluruh item (card) draft yang baru saja dibuat menjadi "Issue" dengan cara hover pada card tersebut, arahkan pada titik tiga (...) yang muncul, kemudian klik "Convert to Issue" dan pilih repository dimana issue tersebut terhubung.



- Klik salah satu card yang telah dibuat, maka akan muncul 'side panel' pada sebelah kanan. Klik pada field "Assignee" dan tugaskan card/issue tersebut kepada orang yang sesuai di tim. Ulangi langkah ini hingga seluruh card yang dibuat telah memiliki assignee yang sesuai.



11. Screenshot kondisi tampilan Github Project pasca melakukan assignment untuk seluruh card. Letakan screenshot pada worksheet pada posisi yang sesuai.
12. Lakukan pengembangan proyek sesuai dengan card/issue yang telah dibuat. Card dapat digeser ke kolom lain (Ready, In Progress, In Review, Done, Backlog) sesuai dengan fase atau tahap penyelesaian card/issue yang sesuai. Kolom dapat ditambahkan atau dihapus (kecuali Backlog, In Progress, Done) sesuai dengan kebutuhan tim. Lakukan juga kustomisasi lain yang dirasa diperlukan dalam tim.
13. "Close Issue" ketika card/issue yang dikerjakan telah selesai.
14. Screenshot kondisi tampilan Github Project setelah satu minggu pengembangan. Letakan screenshot pada worksheet pada posisi yang sesuai.
15. Buat Github Action pada repository kelompok. Github Action yang dibuat disesuaikan dengan kebutuhan kelompok, dapat berupa otomasi test build, otomasi linting, atau otomasi lain yang dirasa perlu. Jelaskan kegunaan Github Action yang dibuat pada kolom yang disediakan di Worksheet Kelompok.

**Note:** Github Project dan Github Action digunakan selama masa pengembangan produk dari Senior Project. Aktivitas pada Github dan Github Project akan menjadi komponen penting untuk penilaian. Ulangi langkah-langkah breakdown dan pembuatan issue saat pengembangan produk, sesuaikan dengan kebutuhan. Github Action juga dapat dikembangkan sesuai kebutuhan dan tahap pengembangan proyek.