LAPORAN ARSITEKTUR KOMPUTER DAN SISTEM OPERASI
PENERAPAN GIT UNTUK KOLABORASI EFEKTIF DALAM PENGEMBANGAN PERANGKAT LUNAK BERBASIS LINUX

Disusun oleh:

Ekky Mulia Lasardi J0403221081

Dosen:

Bayu Widodo, S.T., M.T.

PROGRAM STUDI TEKNOLOGI REKAYASA PERANGKAT LUNAK

SEKOLAH VOKASI

INSTITUT PERTANIAN BOGOR

BOGOR

2024

DAFTAR ISI

Halaman

BAB I
PENDAHULUAN

1.1. Latar Belakang Git

Dalam dunia pengembangan perangkat lunak, sistem kontrol versi adalah alat yang sangat penting untuk mengelola perubahan dalam kode sumber. Git, salah satu sistem kontrol versi terpopuler, memainkan peran krusial dalam memastikan bahwa pengembang dapat bekerja secara efisien dan kolaboratif. Beberapa alasan utama mengapa Git sangat penting adalah:

1. **Manajemen Perubahan**: Git memungkinkan pengembang untuk melacak setiap perubahan yang dibuat dalam kode sumber. Setiap perubahan dicatat dengan detail, termasuk siapa yang membuat perubahan dan kapan perubahan tersebut dilakukan. Hal ini memungkinkan pengembang untuk memahami evolusi proyek dan memudahkan identifikasi serta perbaikan bug.
2. **Kolaborasi**: Git memungkinkan banyak pengembang bekerja pada proyek yang sama secara bersamaan tanpa mengganggu pekerjaan satu sama lain. Dengan fitur *branching* dan *merging*, tim dapat mengembangkan fitur baru atau memperbaiki *bug* dalam cabang yang terpisah sebelum menggabungkannya ke cabang utama.
3. **Keamanan dan Backup**: Dengan Git, setiap salinan repositori adalah salinan lengkap dari seluruh sejarah proyek. Ini berarti bahwa data aman dan ada backup di setiap mesin pengembang, mengurangi risiko kehilangan data.
4. **Pengelolaan Versi**: Git mempermudah pengelolaan berbagai versi perangkat lunak. Pengembang dapat dengan mudah kembali ke versi sebelumnya jika ada masalah, atau membandingkan perbedaan antara versi yang berbeda.

1.2. Relevansi dan Manfaat Penggunaan Git dalam Lingkungan Linux

Menggunakan Git dalam lingkungan Linux sangat relevan dan bermanfaat karena Git dirancang oleh Linus Torvalds, pencipta Linux, dan bekerja sangat baik dalam sistem *Unix-like* seperti Linux. Git biasanya sudah tersedia di banyak distribusi Linux, memudahkan instalasi dan penggunaannya. Selain itu, Git terintegrasi dengan berbagai alat pengembangan di Linux, mendukung otomatisasi melalui *scripting*, dan menawarkan performa tinggi untuk proyek besar. Komunitas *open-source* yang erat dan dukungan luas juga menjadikan Git keterampilan esensial bagi pengembang yang ingin berkontribusi pada proyek-proyek *open-source*.

BAB II
PENJELASAN TEORITIS

2.1. Git

Git adalah sistem kontrol versi terdistribusi yang memungkinkan pengembang untuk melacak dan mengelola perubahan kode sumber dalam proyek perangkat lunak. Konsep dasar Git mencakup repositori, *commit*, *branch*, *merge*, *remote*, serta *pull* dan *push*. Repositori menyimpan semua perubahan yang pernah dilakukan, sementara *commit* mencatat perubahan dengan detail seperti waktu dan pembuat perubahan. *Branch* memungkinkan pengembangan paralel, dan *merge* menggabungkan perubahan dari berbagai *branch*. *Remote repository* memfasilitasi kolaborasi dengan menyinkronkan perubahan antara repositori lokal dan *remote*. *Pull* mengambil perubahan dari *remote*, sedangkan *push* mengirim perubahan ke *remote*. Keuntungan penggunaan Git dalam manajemen kode sumber dan kolaborasi tim pengembang meliputi kolaborasi efektif melalui *branching* dan *merging*, manajemen versi canggih yang memudahkan pelacakan dan pengembalian perubahan, serta distribusi yang aman karena

setiap salinan repositori adalah *backup* lengkap. Git juga terintegrasi dengan berbagai alat pengembangan, meningkatkan efisiensi dan memungkinkan otomatisasi. Skalabilitas dan performanya memastikan pengelolaan proyek besar dengan cepat dan efisien. Dukungan komunitas yang luas menjadikan Git mudah dipelajari dan digunakan, membuatnya menjadi alat berharga dalam meningkatkan produktivitas dan kualitas perangkat lunak.

BAB III
APLIKASI PRAKTIS

3.1. Langkah-Langkah Dasar dalam Menggunakan Git

Adapun langkah-langkah dan contoh konkret dasar untuk membuat dan menggunakan git adalah sebagai berikut:

Inisialisasi Repositori:

- **Langkah**: Mulai dengan membuat direktori proyek dan inisialisasi repositori Git di dalamnya.
- **Perintah**:

```
mkdir nama-proyek
cd nama-proyek
git init
```

Menambahkan File:

- **Langkah**: Tambahkan file atau kode sumber ke dalam direktori proyek.
- **Perintah**:

```
touch file1.txt
echo "print('Hello, World!')" > hello.py
```

Menyimpan Perubahan (Commit):

- **Langkah**: Melacak file baru dan melakukan *commit* untuk menyimpan perubahan.

- **Perintah**:

```
git add .
git commit -m "Initial commit"
```

Membuat Branch:

- **Langkah**: Membuat *branch* baru untuk mengembangkan fitur atau melakukan perubahan spesifik.
- **Perintah**:

```
git checkout -b fitur-baru
```

Menggabungkan Perubahan (Merge):

- **Langkah**: Setelah selesai dengan pengembangan di *branch* fitur, gabungkan perubahan ke *branch* utama.
- **Perintah**:

```
git checkout main
git merge fitur-baru
```

Menghubungkan ke Remote Repository:

- **Langkah**: Hubungkan repositori lokal ke *remote repository* (misalnya di GitHub atau GitLab).
- **Perintah**:

```
git remote add origin https://github.com/username/nama-proyek.git
git push -u origin main
```

Sinkronisasi dengan Remote Repository:

- **Langkah**: Mengambil (*pull*) perubahan terbaru dari *remote repository* dan mendorong (*push*) perubahan lokal.
- **Perintah**:

```
git pull origin main
git push origin main
```

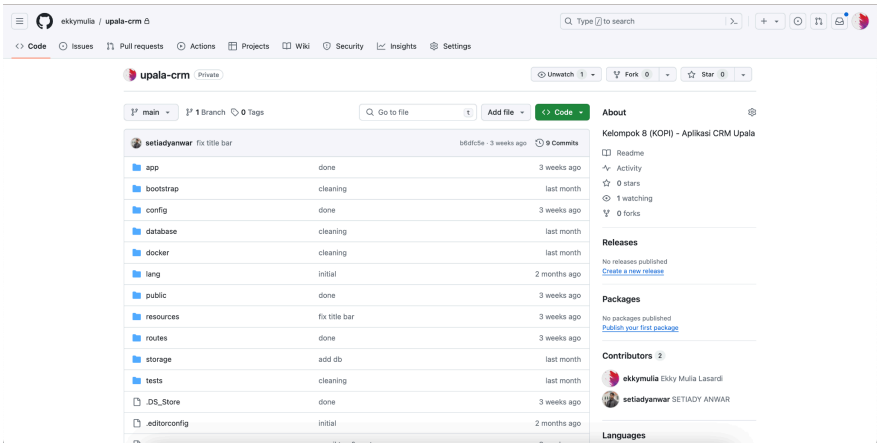
BAB IV
STUDI KASUS

4.1. Studi Kasus dalam Penggunaan Git Sebagai Kontrol Versi

Adapun studi kasus yang dilakukan, pada repository proyek kolaborasi mata kuliah Sistem Informasi dengan nama repositori *upala-crm.* Upala CRM merupakan aplikasi *CRM* dan *Loyalty Management* yang dapat membantu para pemilik usaha dalam meningkatkan hubungan pelanggan, meningkatkan penjualan, dan meningkatkan efisiensi operasional bisnis. Aplikasi ini mudah digunakan dan memiliki fitur yang lengkap, sehingga dapat menjadi solusi yang tepat untuk berbagai jenis usaha.

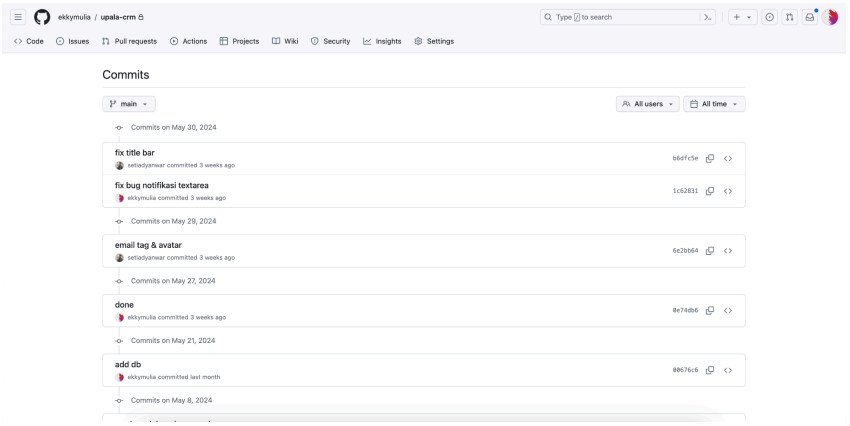
4.2. Penggunaan Git

Dalam proyek pengembangan Upala CRM, Git digunakan sebagai *version control system*, dan Github digunakan sebagai *version control*.



Proyek dimulai dengan menciptakan repositori Git baru di GitHub. Repositori ini berfungsi sebagai tempat untuk menyimpan kode sumber dan berkolaborasi. Langkah pertama adalah membuat repositori tersebut di github dan kemudian menggunakan perintah pada *shell* komputer lokal *'git clone* https://github.com/ekkymulia/upala-crm' untuk menduplikat

repository yang ada pada github, untuk menautkan repository lokal dengan repositori jarak jauh di GitHub.



Untuk menyimpan perubahan pada kode, menggunakan perintah '*git add .*' dan *'git commit -m "<pesan>"*' kemudian '*git push'*. perubahan tersebut disebut *commit* dan dapat dilihat pada Github.

4.2. Manfaat dan Tantangan

Manfaat dari pembuatan repositori menggunakan Git meliputi kemampuan untuk kolaborasi tim secara efisien dengan melacak dan mengelola perubahan kode, menyediakan riwayat perubahan yang lengkap untuk pemulihan cepat jika diperlukan, serta memberikan platform untuk pemantauan kemajuan proyek secara keseluruhan. Namun, tantangan dalam pengelolaan repositori meliputi penyelesaian konflik pemrograman yang mungkin terjadi, mengelola ukuran repositori yang besar untuk menjaga performa Git tetap optimal, dan mengatasi kurva pembelajaran yang diperlukan untuk memahami konsep-konsep seperti *branch* dan operasi Git lainnya. Dengan strategi manajemen yang baik, tim dapat memaksimalkan manfaat dari sistem kontrol versi Git sambil mengatasi tantangan yang mungkin timbul.

BAB V
ANALISIS DAN REFLEKSI

4.1. Analisis

Penggunaan Git memberikan transformasi besar dalam pengembangan perangkat lunak dibandingkan dengan pendekatan tradisional yang sering kali lebih terbatas. Git sebagai sistem kontrol versi menghadirkan kemampuan untuk kolaborasi tim yang efisien, manajemen riwayat perubahan yang jelas, dan kemudahan dalam manajemen versi. Hal ini sangat mengurangi risiko kehilangan perubahan dan memungkinkan pengembang untuk eksperimen dengan lebih aman.

4.1. Refleksi

Pengalaman dengan Git dalam pengembangan perangkat lunak di Linux telah mengubah cara bekerja secara signifikan. Penggunaan Git telah memfasilitasi kolaborasi tim yang lebih baik dan meningkatkan transparansi dalam pengelolaan kode, dan memungkinkan untuk lebih fokus pada peningkatan fitur dan perbaikan *bug* daripada menangani masalah pengelolaan kode.

BAB V
KESIMPULAN

Git memberikan manfaat yang signifikan dalam pengembangan perangkat lunak di sistem operasi Linux dengan menyediakan sistem kontrol versi yang kuat dan fleksibel. Manfaat utamanya meliputi kemampuan untuk melacak perubahan kode secara detail melalui *commit*, memfasilitasi kolaborasi tim yang efisien dengan manajemen branch dan pull request, serta memungkinkan pengembang untuk kembali ke versi sebelumnya dengan mudah jika diperlukan. Git juga mempromosikan praktik pengembangan yang terstruktur dan terorganisir dengan

baik, yang esensial untuk memastikan pengembangan perangkat lunak yang stabil dan *scalable* di lingkungan Linux.

Rekomendasi untuk menggunakan Git dalam proyek pengembangan perangkat lunak adalah untuk menerapkannya sejak awal proyek. Mulailah dengan menginisialisasi repositori Git untuk mengelola kode sumber dan dokumentasi proyek. Gunakan branch untuk mengembangkan fitur baru atau memperbaiki *bug* tanpa mengganggu versi stabil di branch utama. Selain itu, manfaatkan fitur pull request untuk meninjau dan mendiskusikan perubahan sebelum diintegrasikan ke dalam kode utama. Penerapan praktik ini tidak hanya meningkatkan transparansi dan kualitas kode, tetapi juga memfasilitasi kolaborasi tim yang lebih efisien. Jika proyek melibatkan beberapa tim atau pengembang yang bekerja secara bersamaan, Git akan menjadi alat yang sangat berharga untuk mengelola kompleksitas dan memastikan konsistensi dalam pengembangan perangkat lunak di lingkungan Linux.

DAFTAR PUSTAKA

Liberty, Jesse, and Jon Galloway. 2021. *Git for Programmers: Master Git for Effective Implementation of Version Control for Your Programming Projects*. Packt Publishing Ltd.

Miller, Curtis G. 2022. “Introduction to Git.”

Tsitoara, Mariot. 2020. *Beginning Git and GitHub*. Springer.

N. Deepa, B. Prabadevi, L. B. Krithika, and B. Deepa, “An analysis on Version Control Systems,” *Int. Conf. Emerg. Trends Inf. Technol. Eng. ic ETITE 2020*, no. August, 2020, doi: 10.1109/ic-ETITE47903.2020.39.

J. D. Blischak, E. R. Davenport, and G. Wilson, “A Quick Introduction to Version Control with Git and GitHub,” *PLoS Comput. Biol.*, vol. 12, no. 1, pp. 1–18, 2016, doi: 10.1371/journal.pcbi.1004668.