PRADITA University

Kuliah Umum

Lokakarya GIT

Jumat, 27 September 2024

Alfa Yohannis

Contents (1)

PRADITA University

Contents (2)

PRADITA University

Apa itu Sistem Version Control?

PRADITA University

- Alat untuk mengelola perubahan pada kode sumber atau dokumen dari waktu ke waktu.
- Memungkinkan beberapa pengembang untuk berkolaborasi.
- Melacak perubahan dan mengembalikan ke versi sebelumnya jika diperlukan.
- Mencatat setiap perubahan yang dilakukan pada file.
- Memudahkan pelacakan evolusi proyek.

Manfaat Menggunakan VCS

PRADITA University

- **Kolaborasi**:
  - Beberapa anggota tim dapat bekerja secara bersamaan tanpa saling menimpa pekerjaan.
- **Riwayat**:
  - Setiap perubahan dicatat, memberikan riwayat lengkap modifikasi.
- **Kemampuan Mengembalikan**:
  - Mengembalikan ke versi sebelumnya jika terjadi kesalahan.
- **Branching dan Merging**:
  - Fitur atau perbaikan dapat dikembangkan secara terpisah.
  - Dapat digabungkan kembali ke proyek utama setelah selesai.

Local Version Control System (LVCS)

PRADITA University

- Menyimpan semua versi file secara lokal di komputer pengguna.
- Salinan file dibuat secara manual di direktori berbeda.
- Metode ini rentan terhadap kesalahan.
- **Contoh produk**:
  - File system sederhana untuk membuat salinan file secara manual.

Centralized Version Control System (CVCS)

- Menggunakan server pusat untuk menyimpan semua versi file.
- Pengguna harus terhubung ke server untuk mendapatkan atau memperbarui versi file.
- Mempermudah kolaborasi.
- Kelemahan: risiko kegagalan server pusat.
- **Contoh produk**:
  - **Subversion (SVN)**: Sistem version control terpusat yang populer.
  - **Perforce**: Sistem VCS yang digunakan oleh perusahaan besar.
  - **CVS (Concurrent Versions System)**: Sistem version control yang lebih tua.

Distributed Version Control System (DVCS)

- Setiap pengguna memiliki salinan lengkap dari seluruh riwayat proyek.
- Pengguna dapat bekerja secara offline.
- Perubahan dapat digabungkan dengan server pusat atau pengguna lain.
- **Contoh produk**:
  - **Git**: Sistem version control terdistribusi yang paling populer.
  - **Mercurial**: Alternatif untuk Git.
  - **Bazaar**: Sistem VCS yang digunakan di beberapa komunitas open-source.

Latar Belakang Git

PRADITA University

Git adalah sistem version control terdistribusi, yang awalnya dikembangkan oleh Linus Torvalds pada tahun 2005 untuk mengelola pengembangan kernel Linux.

Tujuan desain utama Git:

- **Kecepatan**: Penanganan tugas-tugas version control dengan cepat.
- **Desain Sederhana**: Git dirancang agar mudah digunakan namun kuat.
- **Dukungan Kuat untuk Pengembangan Non-linear**: Kemampuan branching dan merging yang kuat.
- **Pengembangan Terdistribusi**: Setiap pengembang memiliki salinan penuh dari sejarah proyek secara lokal.

Menginstal Git di Linux

PRADITA University

- Untuk Debian/Ubuntu:

```
sudo apt-get install git
```

- Untuk Red Hat:

```
sudo yum install git
```

- Verifikasi instalasi:

```
git --version
```

Membuat Akun GitHub

PRADITA University

GitHub adalah platform populer untuk menghosting repository Git di cloud.

1. Buka `https://github.com/` dan klik tombol `Sign up`.
2. Isi detail yang diperlukan.
3. Verifikasi email dan selesaikan proses pembuatan akun.
4. Setelah masuk, mulai membuat repository dan berkolaborasi.

Git Init: Menginisialisasi Repository

PRADITA University

- Perintah `git init` digunakan untuk menginisialisasi repository Git baru.
- Membuat direktori `.git` di dalam folder proyek.
- Direktori `.git` menyimpan semua metadata dan riwayat version control.
- Perintah:

```
1 git init
```

- Setelah dijalankan, folder proyek menjadi repository Git dan siap untuk melacak perubahan.

Git Add: Menambahkan Perubahan ke Staging

- Perintah `git add` digunakan untuk menambahkan perubahan pada file ke area staging.
- Area staging adalah tahap sebelum perubahan dikomit ke repository.
- File individu atau semua file yang diubah bisa ditambahkan sekaligus.
- Perintah:

```
git add <nama_file>
git add .
```

- Menggunakan `git add .` menambahkan semua file yang berubah ke staging.

Git Commit: Mengkomit Perubahan

PRADITA University

- Perintah `git commit` digunakan untuk mengunci perubahan ke dalam repository.
- Setiap `commit` memerlukan pesan yang menjelaskan perubahan.
- Perintah:

```
git commit -m "Deskripsi perubahan"
```

- Pesan commit harus jelas untuk memudahkan pelacakan riwayat perubahan.

Git Reset: Menghapus Perubahan dari Staging

- Gunakan perintah `git reset` untuk membatalkan perubahan yang telah ditambahkan ke staging.
- Tanpa mengubah file yang sebenarnya.
- Perintah:

```
git reset <nama_file>
```

- Menghapus file dari staging, tetapi perubahan tetap ada.
- Untuk membatalkan perubahan pada file, gunakan `git checkout`.

Penggunaan File .gitignore

PRADITA University

- File `.gitignore` menentukan file atau direktori yang tidak ingin dilacak.
- Pola atau nama file yang tidak akan ditambahkan ke staging meskipun diubah.
- Contoh file `.gitignore`:

```
# Contoh file .gitignore
node_modules/
*.log
*.tmp
```

- Dengan menambahkan pola, Git akan mengabaikan file-file tersebut.

Membuat Branch

PRADITA University

- Git memungkinkan pekerjaan pada berbagai fitur atau perbaikan menggunakan *branch*.
- Perintah `git branch` untuk membuat cabang baru.
- Perintah `git checkout` untuk berpindah ke cabang tersebut.
- Perintah:

```
1  git branch <nama_branch>
2  git checkout <nama_branch>
```

Menggabungkan Branch

PRADITA University

- Setelah perubahan selesai, cabang bisa digabungkan ke cabang utama.
- Perintah untuk menggabungkan:

```
git checkout main
git merge <nama_branch>
```

- Menggabungkan perubahan dari cabang lain ke dalam cabang utama.

Membuat Repository di GitHub

PRADITA University

- GitHub memungkinkan pembuatan repository private dan public untuk mengelola proyek.
- Langkah-langkah untuk membuat repository:
  1. Masuk ke akun GitHub dan klik tombol `New Repository`.
  2. Berikan nama repository dan deskripsi opsional.
  3. Pilih sifat repository:
     - `Public`: dapat diakses oleh siapa saja.
     - `Private`: hanya bisa diakses oleh pemilik dan kolaborator yang diizinkan.
  4. Pilih opsi tambahan: `README.md`, `.gitignore`, atau lisensi.
  5. Klik `Create repository`.

Menambahkan Remote

PRADITA University

- Setelah membuat repository di GitHub, langkah selanjutnya adalah menambahkan repository tersebut sebagai *remote* di proyek Git lokal.
- Untuk menambahkan remote:
  - Gunakan perintah:

```
1    git remote add origin https://github.com/username/nama-
         repo.git
```

  - `origin` adalah nama default untuk remote repository utama.
  - Nama lain dapat digunakan jika diinginkan.

Push Perubahan ke Remote

PRADITA University

- Setelah remote ditambahkan, kirim perubahan dari repository lokal ke GitHub.
- Gunakan perintah:

```
1  git push -u origin main
```

- Perintah ini mengirim semua commit di cabang `main` ke remote `origin`.
- Opsi `-u` mengatur `main` sebagai cabang default untuk operasi `push` berikutnya.

Sinkronisasi Perubahan

PRADITA University

- Untuk menarik (pull) perubahan dari repository remote ke lokal, gunakan perintah:

```
1  git pull origin main
```

- Perintah ini memastikan repository lokal tetap sinkron dengan repository GitHub.

Memfork Repository

PRADITA University

- Memfork repository memungkinkan pengguna membuat salinan pribadi dari proyek orang lain.
- Memungkinkan eksperimen dan modifikasi tanpa mempengaruhi proyek asli.
- Langkah-langkah untuk memfork repository:
  1. Arahkan ke repository GitHub yang ingin difork.
  2. Klik tombol `Fork` di bagian kanan atas halaman.
  3. Pilih akun sebagai tujuan untuk repository yang difork.
- Setelah memfork, salinan baru dibuat di bawah akun GitHub.

Mengkloning Repository

PRADITA University

- Mengkloning repository memungkinkan pengguna mengunduh salinan repository ke mesin lokal.
- Gunakan perintah berikut di terminal:

```
1       git clone <repository-url>
```

- Gantilah `<repository-url>` dengan URL dari repository yang difork atau asli.
- Perintah ini membuat salinan lokal untuk bekerja secara offline.

Membuat Pull Request

PRADITA University

- Setelah melakukan perubahan pada repository yang dikloning, langkah berikutnya adalah mengirimkan perubahan untuk ditinjau.
- Langkah-langkah untuk membuat pull request:
  1. Dorong perubahan ke repository yang difork:

```
git push origin <branch-name>
```

  2. Buka repository asli di GitHub.
  3. Klik pada tab `Pull requests`.
  4. Klik tombol `New pull request`.
  5. Pilih cabang dan berikan deskripsi tentang perubahan.
  6. Klik `Create pull request`.
- Pull request memfasilitasi diskusi dan tinjauan sebelum digabungkan.

Mengambil dan Mengirim Perubahan

RADITA niversity

- Untuk menyinkronkan perubahan antara repository lokal dan jarak jauh:
  - Mengambil perubahan terbaru dari repository jarak jauh:

```
1        git pull origin <branch-name>
```

  - Mengirim perubahan lokal ke repository jarak jauh:

```
1        git push origin <branch-name>
```

- Perintah ini memastikan semua kontributor bekerja dengan versi kode terbaru.

Menyelesaikan Konflik Penggabungan (1)

- Konflik penggabungan dapat terjadi jika beberapa kontributor melakukan perubahan pada bagian yang sama.
- Langkah-langkah untuk menyelesaikan konflik:
  1. Ambil perubahan terbaru dari repository jarak jauh.
  2. Jika terjadi konflik, Git akan menandai file yang konflik.
  3. Buka file yang konflik dan cari penanda konflik (misalnya, `<<<<<<< HEAD`).
  4. Edit file secara manual untuk menyelesaikan konflik.

Menyelesaikan Konflik Penggabungan (2)

- Setelah menyelesaikan konflik, siapkan perubahan:

```
git add <resolved-file>
```

- Komit perubahan yang telah diselesaikan:

```
git commit -m "Resolved merge conflict"
```

- Langkah-langkah ini membantu mengelola dan menyelesaikan konflik penggabungan.

Hindari Mengunggah File Binary Target

- File binary target, seperti file hasil kompilasi atau executable, sebaiknya tidak diunggah ke repository.
- File tersebut dapat cepat membesar dan tidak perlu dilacak di dalam version control.
- Simpan kode sumbernya dan biarkan setiap pengguna melakukan kompilasi di mesin masing-masing.

Hindari File Binary Berukuran Besar

PRADITA University

- File binary besar, seperti gambar, video, atau file media lainnya, sebaiknya dihindari.
- Gunakan platform penyimpanan eksternal seperti Git LFS atau layanan penyimpanan awan.
- Ini membantu menjaga ukuran repository tetap kecil dan mudah dikelola.

Hindari File Temporary

PRADITA University

- File sementara yang dihasilkan selama pengembangan, seperti file log, cache, atau konfigurasi lokal, sebaiknya tidak diunggah.
- Gunakan file `.gitignore` untuk mengabaikan file-file ini agar tidak termasuk dalam version control.

```
# Contoh isi .gitignore
*.log
*.tmp
*.cache
```

Tulis Pesan *Commit* yang Jelas

PRADITA University

- Pesan commit yang jelas dan deskriptif memudahkan pengembang lain memahami perubahan.
- Gunakan kalimat yang menjelaskan apa yang diubah dan mengapa.

```
1    git commit -m "Memperbaiki bug pada fungsi login dan
        menambahkan validasi input"
```

Gunakan Branch untuk Fitur Baru dan Perbaikan

- Selalu buat branch baru untuk mengembangkan fitur baru atau melakukan perbaikan.
- Ini menjaga cabang utama tetap stabil dan menghindari konflik yang tidak perlu.

```
1   git checkout -b nama_fitur_baru
```

Jaga *Commit* yang Kecil dan Sering

PRADITA University

- Commit yang kecil dan sering lebih mudah dikelola dan ditelusuri.
- Memudahkan pengembalian perubahan jika terjadi kesalahan.

Perbarui `README.md` Secara Berkala

PRADITA University

- File `README.md` adalah dokumen penting yang menjelaskan proyek.
- Pastikan informasi di dalamnya selalu diperbarui, termasuk petunjuk instalasi, penggunaan, dan dokumentasi lainnya.

Gunakan Pull Request untuk Kolaborasi

- Gunakan fitur pull request untuk mengusulkan perubahan saat bekerja dalam tim.
- Memfasilitasi diskusi dan tinjauan sebelum perubahan diterapkan ke cabang utama.

Bersihkan Repository Secara Berkala

RADITA University

- Lakukan pemeriksaan rutin pada repository untuk menghapus file yang tidak diperlukan.
- Termasuk menghapus branch yang sudah tidak digunakan dan memperbarui file `.gitignore`.

Patuhi Konvensi Penamaan

PRADITA University

- Gunakan konvensi penamaan yang konsisten untuk branch, commit, dan file.
- Memudahkan pengembang lain memahami struktur dan tujuan dari setiap elemen di dalam repository.

Q & A