Git & Repository

Pre-Course

Apa itu Git

REFACTORY COURSE

Sistem Kontrol Versi

Git adalah sistem kontrol versi terdistribusi yang dirancang untuk menangani segala sesuatu mulai dari proyek kecil hingga proyek sangat besar dengan kecepatan dan efisiensi.

Dibuat oleh Linus Torvalds pada tahun 2005 untuk pengembangan kernel Linux. Git banyak digunakan untuk manajemen kode sumber (SCM) dalam pengembangan perangkat lunak.

Basic Commands

- *git init*: Menginisialisasi repositori Git baru.
- *git clone*: Mengkloning repositori yang ada.
- *git add*: Menambahkan perubahan pada area pementasan.
- *git commit*: Mencatat perubahan pada repositori.
- *git status*: Menampilkan status perubahan sebagai tidak terlacak, dimodifikasi, atau bertahap.
- *git log*: Menampilkan riwayat penerapan.
- *git branch*: Lists, creates, or deletes branches.
- *git checkout*: Beralih cabang atau memulihkan file.
- *git merge*: Menggabungkan cabang.
- *git pull*: Fetches and integrates changes from a remote repository.
- *git push*: Sends changes to a remote repository.
- *git fetch*: Unduh objek dan referensi dari repositori lain.
- *git reset:* Batalkan perubahan dengan mengatur ulang HEAD saat ini ke keadaan tertentu.
- *git revert*: Buat komit baru yang membatalkan perubahan dari komit yang ditentukan.
- *git rebase*: Terapkan kembali komitmen di atas tip dasar lainnya.
- *git cherry-pick*: Terapkan perubahan dari komitmen tertentu ke cabang saat ini.
- git stash: Simpan sementara perubahan yang belum siap diterapkan.
- *git tag*: Kelola tag untuk menandai komitmen tertentu.

Apa itu Repository

REFACTORY COURSE

Sistem Kontrol Versi Terdistribusi

Repositori (sering disingkat "repo") adalah tempat sentral di mana data disimpan dan dikelola. Dalam konteks Git, repositori adalah tempat Git menyimpan semua file proyek dan seluruh riwayat perubahan. Ada dua jenis repositori: lokal dan jarak jauh.

Types of Repositories:

- **Local** (*Repositori yang disimpan di mesin lokal Anda*)
- **Remote** (*Repositori yang dihosting di server jarak jauh, memungkinkan banyak pengguna untuk berkolaborasi*)

Sistem Kontrol Versi Terdistribusi

Repositori (sering disingkat "repo") adalah tempat sentral di mana data disimpan dan dikelola. Dalam konteks Git, repositori adalah tempat Git menyimpan semua file proyek dan seluruh riwayat perubahan. Ada dua jenis repositori: lokal dan jarak jauh.

https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories

Advantages of Using Git and Repositories

Collaboration

- ***Collaboration***: Beberapa pengembang dapat mengerjakan proyek yang sama secara bersamaan.
- ***Version History:*** Lengkapi riwayat setiap perubahan, memungkinkan rollback ke versi sebelumnya.
- ***Branching and Merging***: Kerjakan fitur atau perbaikan secara mandiri dan gabungkan ke dalam proyek utama.
- ***Distributed Development***: Setiap pengembang memiliki salinan lengkap proyeknya, sehingga tahan terhadap kehilangan data.

https://docs.github.com/en/pull-requests/collaborating-with-pull-requests

Popular Hosting Services for Git Repositories

Semver (Semantic Versioning)

Semantic Versioning (SemVer) *adalah skema pembuatan versi untuk perangkat lunak yang menyampaikan makna tentang perubahan mendasar. Ini menggunakan format nomor versi tiga bagian: MAJOR.MINOR.PATCH, setiap bagian bertambah berdasarkan aturan spesifik tentang sifat perubahan.*

Version Number Format:

- **MAJOR** -> *Incremented for incompatible API changes.* **(1.0.0 → 2.0.0)**
- **MINOR** -> *Incremented for backward-compatible new functionality.* **(1.0.0 → 1.1.0)**
- **PATCH** -> *Incremented for backward-compatible bug fixes.* **(1.0.0 → 1.0.1)**

Rules & Guideline Semver

1. ***Version Format:*** MAJOR.MINOR.PATCH.
2. ***Initial Development:*** Before version 1.0.0, anything may change at any time. The public API should not be considered stable.
3. ***MAJOR Version Zero (0.y.z):*** Initial development phase, unstable.
4. ***Incrementing Versions***
5. ***Pre-release and Build Metadata***
    a. Pre-release versions (e.g., alpha, beta, rc) are for testing and are not considered stable.
    b. Build metadata is for providing additional information and does not affect version precedence.

Benefits of Semantic Versioning

- ***Clarity:*** Memberikan cara yang jelas dan mudah dipahami untuk menyampaikan arti nomor versi.
- ***Predictability:*** Pengguna dapat mengantisipasi dampak pembaruan ke versi baru.
- ***Compatibility:*** Membantu mengelola ketergantungan dan kompatibilitas antar paket perangkat lunak.
- ***Communication:*** Memfasilitasi komunikasi yang lebih baik antara pengembang, pengguna, dan pemangku kepentingan lainnya tentang perubahan dalam perangkat lunak.

Terimakasih