Engineer

Git

система контролю версій, де зберігається історія сайту

Що це таке

Git — це розподілена система керування версіями: програма, яка записує стан набору файлів як ланцюжок знімків, щоб будь-який попередній стан можна було переглянути, порівняти з поточним або повернути. Проєкт пише назву як Git, а команду як `git`, і обидва написання означають той самий інструмент.

Кожен знімок — це коміт. Коміт зберігає, який вигляд мав кожен відстежуваний файл у цю мить, хто і коли вніс зміну, повідомлення з поясненням причини та посилання на попередній коміт чи коміти. Файл, який git уже бачив, зберігається лише раз: якщо між двома комітами файл не змінився, новий знімок вказує на копію, що вже лежить на диску, тож довга історія займає значно менше місця, ніж повна копія на кожен коміт.

Усе всередині назване за вмістом. Перш ніж зберегти файл, список каталогу чи коміт, git обчислює контрольну суму його байтів, у класичному форматі — хеш SHA-1 із 40 шістнадцяткових символів, і ця сума стає назвою об’єкта. Змініть один байт — і зміниться назва, а разом із нею назва кожного коміту, збудованого зверху. Так git помічає пошкодження даних, і саме тому ідентифікатор коміту через роки вказує на точно той самий код.

Розподілена означає, що кожна копія повна. Клонування стягує всю історію, а не лише найновіші файли, тож перегляд журналу, порівняння двох версій чи створення коміту відбуваються на локальному диску без жодного сервера. Ділитися роботою — окремий, свідомий крок: свої коміти надсилають до іншої копії командою push або забирають чужі командою fetch і зливають. Роль сервера може виконувати будь-яка копія; у більшості команд її виконує одна узгоджена копія, часто на хостинговому сервісі, — за домовленістю, а не тому, що цього вимагає інструмент.

Робота проходить через три місця. Робоче дерево містить файли, які редагують. Область підготовки, яку ще називають індексом, містить те, що увійде до наступного коміту, тож можна закомітити частину змін, а решту лишити на потім. Каталог git, `.git` у корені проєкту, містить базу об’єктів і всі посилання. Гілка — лише рухома назва, що вказує на один коміт, тому створюється миттєво.

Git написали 2005 року Лінус Торвальдс і спільнота ядра Linux після того, як BitKeeper, пропрієтарний інструмент, яким вони користувалися, перестав бути для них безплатним. Цілі були такі: швидкість, простий дизайн, потужна підтримка тисяч паралельних гілок, повна розподіленість і здатність упоратися з проєктом розміру ядра.

Застосуйте поняття до рішення

Застосуйте «Що таке git?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Перевірено, випущено і з можливістю відкату, тож зміни доходять до клієнтів у вівторок удень, а не пізно в п'ятницю.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Автоматизувала розгортання GIS SaaS через CI/CD — розгортання, дані та звітність — усе без ручної роботи

Автоматизувала розгортання GIS SaaS через CI/CD Зв'язатися з нами →

Чому це важливо

Керування версіями відповідає на питання, що виникають, щойно з тими самими файлами працює більше ніж одна людина або одна людина довше за тиждень: що змінилося, коли, хто це зробив і навіщо. Якщо вести облік вручну, відповіді живуть у пам’яті, у назвах файлів на кшталт `final-v3` і в поштових ланцюжках. З git вони лежать поруч із кодом, прив’язані до кожної зміни, і будь-хто може в них шукати.

Розподілена будова змінює щоденну роботу команди. Кожен клон містить усю історію, тож ніхто не чекає на центральний сервер, щоб переглянути журнал чи закомітити, а збій сервера не зупиняє роботу. Гілки дешеві, тож кожна функція, виправлення чи експеримент може мати власну гілку, проходити рев’ю окремо й зливатися, коли готове, а головна лінія тим часом лишається придатною до випуску.

Та сама будова робить git обережним із даними. Майже кожна операція лише додає щось до бази об’єктів, тож закомічену роботу важко втратити: гілку, видалену помилково, rebase, що пішов не так, або reset не на той коміт зазвичай можна скасувати. Страхувальна сітка — reflog, локальний запис того, куди і коли вказувала кожна гілка; усталено записи про досяжні коміти живуть 90 днів, а про недосяжні — 30 днів.

Git — також спільна основа під більшістю інших інструментів у ланцюжку постачання. Форджі, інструменти рев’ю, системи збирання, скрипти розгортання й генератори статичних сайтів читають із нього, і тому історія стає джерелом фактів далеко за межами коду: день, коли сторінку востаннє змінили, коміт, із якого зібрано випуск, зміна, з якою прийшла помилка.

Як це виглядає на практиці

Власний код компанії показує три боки цього.

Вихідний код раніше жив на сторонньому хостинговому сервісі. Тепер він лежить на git-сервері, який компанія тримає на власному обладнанні: це єдиний двійковий файл під наглядом операційної системи, обраний за те, що він передусім працює через SSH і не має адміністративного вебінтерфейсу. Анонімний доступ і доступ без ключа відхиляються, кожен оголошений репозиторій позначено як приватний, а слухач SSH прив’язаний лише до інтерфейсу зворотної петлі на порту 23231, і ззовні до нього дістаються через проміжний хост. Сторінка чесно називає ціну: власна форджа означає ще й власну автентифікацію, контроль доступу, зберігання даних і резервні копії.

Повні копії історії на кількох машинах легко сплутати з резервною копією. Реалізацію резервного копіювання та тестового відновлення компанії спроєктовано, щоб розширити можливості відновлення за межі знімка провайдера. Її скрипти restic і юніти systemd підтримують дамп бази даних, якщо його налаштовано, надсилання зашифрованих знімків із дедуплікацією до сховища іншого провайдера через SFTP, очищення за політикою зберігання та перевірки цілісності. Окремий щомісячний процес відновлює відомий файл у тимчасове місце й порівнює його байти. Цей тест має справді відбутися, перш ніж він підтвердить можливості відновлення: у записі про реалізацію від 2026-10-27 зазначено, що таймери були вимкнені за

відсутності налаштованого репозиторію; жодного завершеного реального відновлення не зафіксовано. Жодна кількість клонів на ноутбуках не забезпечує доказу, якого бракує.

Історія живить і сам сайт. На сторінці подяк записано, що валідатор стрічок W3C попереджає: більшість записів у стрічках мають однакову дату зміни. Попередження слушне, бо дати беруться з історії git, а більшість цих сторінок справді написано в кількох тих самих комітах. Сайт воліє показати попередження, ніж вигадати дату. Історія каже, що сталося, зокрема й те, що багато чого сталося одночасно.

Одна команда читає такий факт назад. Вона виводить короткий ідентифікатор, дату коміту у форматі ISO 8601 і тему останнього коміту, що торкався одного конкретного файлу:

```
git log -1 --format='%h %cI %s' -- README.md
```

Де виникають проблеми

- **Секрет, закомічений один раз, лишається в історії.** Якщо видалити файл наступним комітом, він зникне з найновішого знімка, але лишиться в попередньому, який уже є в кожному клоні. Облікові дані варто вважати розкритими й замінити; переписування історії згодом звужує поширення, але копії, зроблені тим часом, зберігають старі коміти.
- **Примусове надсилання витирає чужу роботу.** `git push --force` вимикає перевірку того, що віддалена гілка є предком надісланого, тож коміти, які колега надіслав тим часом, випадають із гілки. `--force-with-lease` оновлює віддалену гілку лише доти, доки вона вказує туди, де її бачили востаннє.
- **Клони вважають резервною копією.** Копія на ноутбуці відстає, містить лише ті гілки, які власник отримував, і з неї ніколи навмисно не відновлювалися. Задачі, правила доступу та решта власних даних форджі лежать зовсім поза репозиторієм.
- **Великий двійковий файл назавжди обтяжує кожен клон.** Git зберігає кожну версію кожного файлу, а кожен клон завантажує їх усі, тож відео, яке випадково потрапило в коміт і було видалене наступного дня, усе одно їде на кожну машину, що клонує проєкт згодом.

- **Робота, що лежить в одному місці, зникає разом із ним.** Reflog, який рятує після невдалого reset, живе на одній машині й має строк дії. Коміти, яких ніколи не надсилали, більше ніде не існують, а гілка, яку місяцями не чіпали, відходить так далеко, що злити її назад стає окремим проєктом.

Що почитати далі

- Git documentation — довідковий посібник, книга й відео від самого проєкту.
- Pro Git українською — книга в українському перекладі на сайті проєкту, від першого коміту до бази об’єктів під ним.
- What is Git? — знімки, локальні операції, контрольні суми й три стани в одному розділі.
- A Short History of Git — чому спільнота ядра 2005 року написала власний інструмент і що він мав уміти.
- git-reflog — як знайти, куди вказувала гілка до помилки, і скільки зберігається цей запис.
- git-push — параметри примусового надсилання і що захищає кожен із них.

Словник

Обирайте за фактичними умовами

Опишіть шлях розгортання, експлуатаційні ризики та чергову команду. Ми можемо визначити зміну з перевірками, шляхом відновлення та чіткою відповідальністю.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке git?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/git/