Engineer

Git

Versionskontrolle für die Historie dieser Website

Was es ist

Git ist ein verteiltes Versionskontrollsystem: ein Programm, das den Zustand einer Menge von Dateien als Kette von Momentaufnahmen erfasst. Jeder frühere Zustand lässt sich damit ansehen, mit der Gegenwart vergleichen oder wiederherstellen. Das Projekt schreibt seinen Namen Git und den Befehl `git`. Beides bezeichnet dasselbe Werkzeug.

Jede Momentaufnahme ist ein Commit. Ein Commit speichert, wie jede erfasste Datei zu diesem Zeitpunkt aussah, wer die Änderung wann vorgenommen hat, eine Nachricht mit der Begründung und einen Verweis auf den oder die Vorgänger-Commits. Git speichert eine bereits bekannte Datei nur einmal: Bleibt sie zwischen zwei Commits unverändert, verweist die neue Momentaufnahme auf die bereits auf der Festplatte liegende Kopie, statt sie erneut zu schreiben. Eine lange Historie kostet deshalb deutlich weniger Platz als eine vollständige Kopie je Commit.

Im Inneren wird alles nach seinem Inhalt benannt. Bevor Git eine Datei, eine Verzeichnisauflistung oder einen Commit speichert, berechnet es eine Prüfsumme über dessen Bytes. Im klassischen Format ist das ein SHA-1-Hash mit 40 Hexadezimalzeichen. Diese Prüfsumme wird zum Namen des Objekts. Ändert sich auch nur ein Byte, ändert sich der Name mit, ebenso der Name jedes darauf aufbauenden Commits. So erkennt Git beschädigte Daten. Deshalb verweist ein Commit-Bezeichner auch Jahre später noch auf genau denselben Code.

Verteilt bedeutet, dass jede Kopie vollständig ist. Beim Klonen eines Repositorys kommt die gesamte Historie mit, nicht nur der jüngste Dateistand. Das Lesen des Logs, der Vergleich zweier Versionen und das Erstellen eines Commits erfolgen daher alle auf der lokalen Festplatte, ohne Server. Arbeit zu teilen ist ein eigener, bewusster Schritt: Sie pushen Ihre Commits zu einer anderen Kopie oder holen

mit Fetch Commits aus einer solchen und führen sie zusammen. Jede Kopie kann als Server dienen. In den meisten Teams übernimmt eine vereinbarte Kopie diese Rolle, oft bei einem Hostingdienst. Das ist eine Konvention und keine Vorgabe des Entwurfs.

Die tägliche Arbeit durchläuft drei Orte. Der Arbeitsbaum enthält die Dateien, die Sie bearbeiten. Die Staging-Area, auch Index genannt, enthält den vorgesehenen Inhalt des nächsten Commits. So lässt sich ein Teil der Änderungen committen und der Rest für später aufheben. Das Git-Verzeichnis, `.git` auf der obersten Projektebene, enthält die Objektdatenbank und alle Referenzen. Ein Branch ist lediglich ein verschiebbarer Name, der auf einen Commit zeigt. Deshalb lässt er sich augenblicklich anlegen.

Git wurde 2005 von Linus Torvalds und der Linux-Kernel-Gemeinschaft geschrieben, nachdem BitKeeper, das bis dahin verwendete proprietäre Werkzeug, nicht mehr kostenlos genutzt werden durfte. Die Ziele waren Geschwindigkeit, ein einfacher Entwurf, gute Unterstützung für Tausende paralleler Branches, vollständige Verteilung und die Fähigkeit, ein Projekt von der Größe des Kernels zu verwalten.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Git?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Geprüft, veröffentlicht und rückgängig zu machen: Änderungen erreichen Ihre Kunden am Dienstagnachmittag statt spät am Freitagabend.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. GIS-SaaS-Deployments mit GitHub automatisiert — Bereitstellung, Daten und Berichte ohne Handarbeit

GIS-SaaS-Deployments mit GitHub automatisiert Kontakt aufnehmen →

Warum es wichtig ist

Versionskontrolle beantwortet die Fragen, die entstehen, sobald mehr als eine Person oder eine Person länger als eine Woche dieselben Dateien bearbeitet: Was hat sich wann, durch wen und warum geändert? Von Hand festgehalten, stecken die Antworten im Gedächtnis, in Dateinamen wie `final-v3` und in E-Mail-Verläufen. Mit Git stehen sie neben dem Code, mit jeder Änderung verknüpft und für alle durchsuchbar.

Der verteilte Entwurf verändert den Arbeitsalltag eines Teams. Jeder Klon enthält die gesamte Historie. Niemand wartet daher auf einen zentralen Server, um das Log zu lesen oder zu committen, und bei einem Serverausfall können alle weiterarbeiten. Branches sind günstig. Jede Funktion, Korrektur oder jedes Experiment kann deshalb einen eigenen bekommen, einzeln geprüft und im fertigen Zustand zusammengeführt werden, während die Hauptlinie weiter auslieferbar bleibt.

Derselbe Entwurf lässt Git vorsichtig mit Daten umgehen. Fast jeder Vorgang ergänzt nur die Objektdatenbank. Bereits committete Arbeit geht daher nicht leicht verloren: Ein versehentlich gelöschter Branch, ein misslungener Rebase oder ein Reset auf den falschen Commit lassen sich meist rückgängig machen. Das Sicherheitsnetz ist das Reflog, eine lokale Aufzeichnung darüber, wann jeder Branch wohin zeigte. Es wird standardmäßig 90 Tage für noch erreichbare Commits und 30 Tage für nicht mehr erreichbare aufbewahrt.

Git bildet außerdem die gemeinsame Grundlage der meisten anderen Werkzeuge in einer Auslieferungskette. Forges, Review-Werkzeuge, Buildsysteme, Bereitstellungsskripte und Generatoren statischer Websites lesen daraus. Die Historie wird damit weit über den Code hinaus zur Faktenquelle: wann eine Seite zuletzt geändert wurde, aus welchem Commit ein Release gebaut wurde und mit welcher Änderung ein Fehler einzog.

Wie es in der Praxis aussieht

Der eigene Code dieses Unternehmens zeigt drei Seiten davon.

Der Quellcode lag früher bei einem fremden Hostingdienst. Heute liegt er auf einem Git-Server, den das Unternehmen auf eigener Hardware betreibt: einer einzelnen Binärdatei unter Aufsicht des Betriebssystems, ausgewählt wegen des vorrangigen SSH-Zugangs ohne administrative Weboberfläche. Klone und Pushes reisen über SSH zu Port 23231. Dieser lauscht nur auf Loopback; ein entfernter Client außerhalb des Hosts erreicht ihn daher über einen Sprungserver. Anonyme Sitzungen und solche ohne Schlüssel werden abgewiesen, und jedes deklarierte Repository ist als privat markiert. Die Seite benennt den Preis deutlich: Wer eine eigene Forge betreibt, verantwortet auch Authentifizierung, Zugriffskontrolle, Speicher und Sicherungen.

Vollständige Kopien der Historie auf mehreren Rechnern lassen sich leicht mit einer Sicherung verwechseln. Die Implementierung für Sicherungen und Wiederherstellungstests des Unternehmens sollte die Wiederherstellung über einen Snapshot beim Anbieter hinaus ermöglichen. Ihre restic-Skripte und systemd-Units unterstützen einen Datenbankdump, sofern dieser konfiguriert ist, sowie verschlüsselte und deduplizierte Snapshots über SFTP bei einem anderen Speicheranbieter, Bereinigung nach Aufbewahrungsregeln und Integritätsprüfungen. Ein gesonderter monatlicher Ablauf stellt eine bekannte Datei an einem temporären Ort wieder her und vergleicht ihre Bytes. Dieser Test muss tatsächlich laufen, bevor er eine Wiederherstellungsfähigkeit belegt: Der Implementierungsstand vom 2026-10-27 sagt, dass die Timer ohne konfiguriertes Repository deaktiviert waren, und verzeichnet keine abgeschlossene Wiederherstellung im realen Betrieb. Beliebig viele Klone auf Laptops liefern diesen fehlenden Nachweis nicht.

Die Historie speist auch die Website selbst. Die Credits-Seite hält fest, dass der W3C Feed Validator davor warnt, dass die meisten Feed-Einträge dasselbe Änderungsdatum tragen, und dass die Warnung stimmt: Die Datumsangaben stammen aus der Git-Historie, und die meisten dieser Seiten wurden tatsächlich in einer Handvoll Commits geschrieben. Die Website zeigt lieber die Warnung, als ein Datum zu erfinden. Die Historie sagt, was geschah, auch dass vieles davon gleichzeitig geschah.

Ein Befehl liest eine solche Information wieder aus. Er gibt den kurzen Bezeichner, das Commit-Datum im Format ISO 8601 und den Betreff des letzten Commits aus, der eine Datei berührt hat:

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

Wo es schiefgeht

- **Ein einmal committetes Geheimnis bleibt in der Historie.** Wird die Datei im nächsten Commit gelöscht, fehlt sie in der jüngsten Momentaufnahme, bleibt aber in der vorigen erhalten, die jeder Klon bereits enthält. Behandeln Sie die Zugangsdaten als offengelegt und tauschen Sie sie aus. Ein nachträgliches Umschreiben der Historie begrenzt die Verbreitung, aber zwischenzeitlich erstellte Kopien behalten die alten Commits.

- **Ein Force-Push ersetzt die Arbeit anderer.** `git push --force` schaltet die Prüfung ab, ob der entfernte Branch ein Vorfahr des gepushten Stands ist. Commits, die ein Kollege zwischenzeitlich gepusht hat, fallen dadurch vom Branch. `--force-with-lease` aktualisiert den entfernten Branch nur, solange er noch auf den zuletzt von Ihnen gesehenen Stand zeigt.
- **Klone gelten als Sicherung.** Eine Laptopkopie hinkt hinterher, enthält nur die Branches, die ihr Besitzer zufällig abgerufen hat, und wurde nie gezielt für eine Wiederherstellung verwendet. Issues, Zugriffsregeln und die übrigen eigenen Daten einer Forge liegen vollständig außerhalb des Repositorys.
- **Eine große Binärdatei bläht jeden Klon dauerhaft auf.** Git behält jede Version jeder Datei, und jeder Klon lädt alle herunter. Ein versehentlich committetes und am nächsten Tag gelöschtes Video reist deshalb weiterhin auf jeden Rechner, der das Projekt später klont.
- **Arbeit an nur einem Ort verschwindet mit diesem Ort.** Das Reflog, das einen falschen Reset rettet, lebt auf einem Rechner und läuft ab. Nie gepushte Commits existieren nirgendwo sonst. Ein monatelang unberührter Branch entfernt sich so weit, dass das Zusammenführen zu einem eigenen Projekt wird.

Weiterführende Lektüre

- Git-Dokumentation — Referenzhandbuch, Buch und Videos vom Projekt selbst.
- Pro Git — das gesamte Buch online, vom ersten Commit bis zur darunterliegenden Objektdatenbank.
- Was ist Git? — Momentaufnahmen, lokale Vorgänge, Prüfsummen und die drei Zustände in einem Kapitel.
- Eine kurze Geschichte von Git — warum die Kernel-Gemeinschaft 2005 ein eigenes Werkzeug schrieb und was sie erreichen wollte.
- git-reflog — wie sich der frühere Zielstand eines Branches vor einem Fehler finden lässt und wie lange diese Aufzeichnung bleibt.
- git-push — die Force-Optionen und was jede von ihnen schützt.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Git?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/git/