Inhalt

Vorwort

Immer, wenn mehrere Personen gemeinsam an einem Softwareprojekt arbeiten, braucht es ein System, um alle durchgeführten Änderungen nachvollziehbar zu speichern. Gleichzeitig gibt ein derartiges Versionsverwaltungssystem allen Entwicklern Zugriff auf das gesamte Projekt. Jeder Programmierer weiß, was die anderen zuletzt gemacht haben, jede Entwicklerin kann den Code der anderen ausprobieren und das Zusammenspiel mit ihren eigenen Änderungen testen.

In der Vergangenheit gab es viele Versionsverwaltungssysteme, z. B. CVS, Subversion (SVN) oder Visual SourceSafe. Im vergangenen Jahrzehnt hat sich aber Git zum De-facto-Standard entwickelt.

Einen wesentlichen Anteil an diesem Erfolg hatte die Webplattform GitHub, die den Einstieg und die Nutzung von Git wesentlich vereinfachte. Unzählige Open-Source-Projekte nutzen das kostenlose Angebot GitHubs zum Projekt-Hosting. Kommerzielle Kunden, die den Quellcode nicht veröffentlichen wollten, zahlen für diesen Service. GitHub ist natürlich nicht die einzige Git-Plattform: Wichtige Konkurrenten sind z. B. GitLab, Azure DevOps und Bitbucket. Dessen ungeachtet kaufte Microsoft 2018 GitHub für unglaubliche 7,5 Milliarden US$. Im Gegensatz zu anderen Übernahmen hat dies bisher der Popularität von GitHub nicht geschadet.

Die Geschichte von Git

Git entstand, weil Linus Torvalds für die Weiterentwicklung des Linux-Kernels ein neues Versionsverwaltungssystem brauchte. Die Entwicklergemeinde hatte zuvor das Programm BitKeeper verwendet. Linus Torvalds war mit dem Programm grundsätzlich zufrieden, eine Lizenzänderung machte aber einen Wechsel erforderlich. Von den damals verfügbaren Open-Source-Programmen genügte keines seinen hohen Ansprüchen.

So stoppte der Linux-Chefentwickler kurzzeitig seine Hauptarbeit und schuf in nur zwei Wochen das Grundgerüst von Git. Der Name *Git* steht sinngemäß für *Blödmann* oder *Depp*, und auch die Hilfeseite `man git` bezeichnet das Programm als *the stupid content tracker*.

Was für ein Understatement das ist, wurde erst nach und nach klar, als Linus Torvalds die Weiterentwicklung von Git längst wieder aus der Hand gegeben hatte: Nicht nur die Kernel-Entwickler stellten ihre Arbeit rasch und problemlos auf Git um, in den folgenden Jahren wechselten immer mehr Softwareprojekte auch außerhalb der Open-Source-Welt zu Git.

Den endgültigen Durchbruch schaffte Git, als sich Webplattformen wie GitHub und GitLab etablierten. Diese Websites vereinfachen das Hosting von Git-Projekten enorm und sind heute aus dem Git-Alltag nicht mehr wegzudenken. (Selbst der Linux-Kernel befindet sich mittlerweile auf GitHub!)

Ein bisschen ist das eine Ironie des Schicksals: Linus Torvalds wichtigstes Ziel beim Design von Git war es, ein dezentrales Versionsverwaltungssystem zu schaffen. Aber erst der zentralistische Ansatz von GitHub und Co. machte Git für Entwickler abseits der Guru-Liga richtig attraktiv.

Es gibt heute Stimmen, die die Bedeutung von Git ebenso hoch einschätzen wie die von Linux. Damit ist es Linus Torvalds gleich zwei Mal gelungen, einen Bereich des Software-Universums vollständig auf den Kopf zu stellen.

Jeder verwendet es, keiner versteht es

Bei aller Begeisterung für Git: Es ist unübersehbar, dass Git von Profis für Profis konzipiert wurde. Wir wollen in diesem Buch gar nicht erst den Eindruck erwecken, Git wäre einfach. Das ist es nicht:

- Häufig führt nicht ein Weg zum Ziel, vielmehr gibt es mehrere Wege. Für die, die Git schon beherrschen, ist das nützlich; aber wenn Sie Git gerade lernen, verwirrt diese Vielfalt.
- Vielen Open-Source-Projekten wird der Vorwurf gemacht, sie seien schlecht dokumentiert. Das kann man bei Git wirklich nicht sagen. Im Gegenteil! Jedes Git-Kommando, jede Anwendungsmöglichkeit wird in `man`-Seiten sowie auf der Webseite *https://git-scm.com/docs* so ausführlich und mit allen erdenklichen Sonderfällen erläutert, dass man sich in den Details geradezu verliert.
- Erschwerend kommt hinzu, dass es ähnliche Begriffe mit unterschiedlichen Bedeutungen gibt, leicht zu verwechselnde Subkommandos, die stark voneinander abweichende Aufgaben erfüllen. Manche Begriffe haben je nach Kontext unterschiedliche Bedeutungen oder werden in der Dokumentation uneinheitlich verwendet.

Wir geben es ganz offen zu: Trotz jahrelanger Git-Praxis haben wir beim Schreiben dieses Buchs noch eine Menge Details dazugelernt!

Über dieses Buch

Natürlich ist es möglich, Git sehr minimalistisch zu verwenden. Allerdings können kleine Abweichungen von der täglichen Routine dann zu überraschenden und oft unverständlichen Nebenwirkungen oder Fehlern führen.

Jeder Git-Einsteiger kennt das Gefühl, wenn ein Git-Kommando eine unverständliche Fehlermeldung liefert: Mit kaltem Schweiß überlegt man, ob man gerade das Repository für alle Entwickler nachhaltig zerstört hat und wen man bitten könnte, Git mit den richtigen Kommandos doch zur Weiterarbeit zu überreden.

Deswegen ist es nicht zielführend, Git zu beschreiben, ohne dabei in die Tiefe zu gehen. Erst ein gutes Verständnis für die Funktionsweise von Git gibt die notwendige Sicherheit, Merge-Konflikte oder andere Probleme sauber beheben zu können.

Gleichzeitig war uns aber klar, dass dieses Buch nur funktionieren kann, wenn wir den wesentlichen Funktionen den Vorrang geben. Trotz 400 Seiten ist dieses Buch *nicht* die allumfassende Anleitung zu Git, die auch den letzten Sonderfall berücksichtigt und jedes noch so exotische Git-Subkommando vorstellt. Wir haben daher in diesem Buch die Spreu vom Weizen getrennt.

Dieses Buch ist in überschaubare Kapitel gegliedert, die Sie wie bei einem Bausteinsystem nach Bedarf lesen können:

- Nach einer kurzen Einführung (»Git in 10 Minuten«) führen wir in den Kapiteln »Learning by Doing«, »Git-Grundlagen« und schließlich »Datenanalyse im Git-Repository« in den Umgang mit Git ein. Dabei konzentrieren wir uns auf die Nutzung von Git auf Kommandoebene und gehen nur am Rande auf Plattformen wie GitHub bzw. auf andere Benutzeroberflächen ein.

  Git-Einsteigern empfehlen wir, mit diesen vier Kapiteln zu starten. Selbst wenn Sie schon etwas Git-Erfahrung haben, sollten Sie sich unbedingt ein paar Stunden Zeit nehmen, um »Git-Grundlagen« zu lesen und einige der dort vorgestellten Techniken (Merging, Rebasing etc.) in einem Test-Repository auszuprobieren.

- Die folgenden drei Kapitel – »GitHub«, »GitLab« sowie »Azure DevOps, Bitbucket, Gitea und Gitolite« – stellen die wichtigsten Git-Plattformen vor. Gerade für komplexe Projekte bieten diese Plattformen nützliche Zusatzfunktionen, z. B. um automatische Tests durchzuführen oder um Continuous Integration zu implementieren.

  Selbstverständlich berücksichtigen wir auch den Fall, dass Sie Ihr Git-Repository selbst hosten möchten. Mit GitLab, Gitea oder Gitolite lässt sich dieser Wunsch relativ leicht realisieren.

- Damit wenden wir uns von den Grundlagen der Praxis zu: Im Kapitel »Workflows« zeigen wir populäre Muster, wie Sie die Arbeit vieler Entwickler mit Git in geordnete Bahnen (*Branches*) leiten.

  Im Kapitel »Arbeitstechniken« stehen fortgeschrittene Git-Funktionen im Vordergrund, z. B. Hooks, Submodule, Subtrees sowie die Zwei-Faktor-Authentifizierung, die alle größeren Git-Plattformen unterstützen.

»Git in der Praxis« zeigt, wie Sie auf Linux-Systemen Konfigurationsdateien (*Dotfiles*) bzw. das ganze `/etc`-Verzeichnis mit Git versionieren, wie Sie ein Projekt von SVN auf Git umstellen oder wie Sie eine simple Website schnell und einfach mit Git und Hugo realisieren.

»Gängige Probleme und ihre Lösungen« helfen Ihnen bei schwer verständlichen Fehlermeldungen aus der Sackgasse. Hier finden Sie auch Anleitungen, wie Sie Sonderwünsche realisieren – z. B. wie Sie große Dateien aus dem Git-Repository entfernen oder wie Sie einen Merge-Vorgang nur für eine ausgewählte Datei durchführen.

- Die »Kommandoreferenz« fasst in aller Kürze die wichtigsten Git-Kommandos und deren Optionen zusammen. Dabei haben wir uns vom Motto »weniger ist mehr« leiten lassen. Unser Ziel war nicht eine vollständige Referenz, sondern eine Art »Essenz von Git«.

Beispiel-Repositorys

Einige Beispiele aus dem Buch stellen wir Ihnen auf GitHub zur Verfügung. Werfen Sie einen Blick auf die Begleitwebsite zum Buch bzw. direkt auf GitHub!

https://gitbuch.info
https://github.com/git-buch

Lieber Leser, liebe Leserin!

Uns ist bewusst, dass Sie vielleicht nicht mit großer Freude die Lektüre dieses Buchs beginnen: Sie wollen oder müssen für ein Projekt Git verwenden. Aber Ihr Ziel ist nicht Git an sich, vielmehr wollen Sie Code produzieren, Ihr Projekt vorantreiben. Sie haben eigentlich weder Zeit noch Lust, sich mit Git zu beschäftigen – Sie wollen gerade so viel wissen, dass Sie Git fehlerfrei anwenden können.

Wir haben dafür Verständnis. Trotzdem empfehlen wir Ihnen dringend, ein paar Stunden mehr als geplant zu investieren, um Git systematisch kennenzulernen.

Wir versprechen Ihnen: Sie gewinnen diese Zeit später zurück! Zu wenig Git-Verständnis bedeutet zwangsläufig, dass Sie immer wieder im Internet nach der Lösung für ein gerade aufgetretenes Problem suchen müssen (oft unter Zeitdruck).

Auch wenn Sie aktuell primär Ihr Projekt im Fokus haben: Git-Kenntnisse sind langfristig eine Kernkompetenz, die Sie als Entwickler(in) in vielen zukünftigen Projekten brauchen werden! In diesem Sinne wünschen wir Ihnen viel Erfolg mit Git!

Michael Kofler (*https://kofler.info*)
Bernd Öggl (*https://webman.at*)