

Einführung

Versionsverwaltung – Was ist denn nun eigentlich genau ein Versionsverwaltungsprogramm? Wodurch zeichnet es sich aus und warum wird es gebraucht? Das sind einige der häufigen ersten Fragen, die zu Beginn aufkommen. Die prinzipielle Bedeutung leitet sich schon aus dem Wort selbst ab: Es handelt sich um die Verwaltung von Versionen. Konkret bedeutet es, dass Sie von Dateien Versionen erzeugen können, die dann sinnvoll verwaltet werden.

Das Wort »Version« klingt zunächst erst einmal nach einer größeren Änderung, doch auch eine kleine Änderung erzeugt eine neue Version einer Datei. Je nach Kontext gibt es ein unterschiedliches Verständnis für den Begriff »Version«. Wenn bei Git von Versionen gesprochen wird, ist damit so gut wie immer die Version einer einzelnen Datei oder einer Sammlung von Dateien gemeint. Im Sinne der Software-Entwicklung werden neue Versionen von Programmen veröffentlicht, also zum Beispiel die Git-Version 2.29.

Aber wofür brauchen Sie nun ein Versionsverwaltungsprogramm wie Git? Viele kennen vermutlich folgendes Problem: Sie gehen einer Tätigkeit nach – sei es das Schreiben an einem Text, das Bearbeiten eines Bildes oder eines Videos – und der aktuelle Stand soll immer mal wieder zwischengespeichert werden. Hauptgrund ist, dass dauernd eine Sicherung der Datei vorhanden sein soll, und ein weiterer Grund ist, dass Sie wieder auf einen älteren Stand zurückspringen können, falls Sie doch einige Schritte rückgängig machen wollen. Die Vorgehensweise zum manuellen Erzeugen solcher Versionen ist unterschiedlich – die einen fügen Zahlen mit Versionsnummern am Ende des Dateinamens an, die anderen erzeugen wiederum Ordner mit dem aktuellen Datum, in denen die Dateien liegen. So passiert es häufiger, dass neben `Bachelorarbeit_v1.odt` und `Bachelorarbeit_v2.odt` noch ein `Bachelorarbeit_v3_final.odt` und `Bachelorarbeit_v3_final_new.odt` liegt. Beide genannten Möglichkeiten funktionieren zwar prinzipiell, sind allerdings weder praktikabel noch wirklich sicher und vor allem fehleranfällig. Das ist besonders dann der Fall, wenn Sie den Dateien keine eindeutigen Namen gegeben haben. Dies trifft insbesondere dann zu, wenn zu viele Versionen einer einzigen Datei rumliegen oder mehrere Dateien gleichzeitig versioniert werden müssen.

Genau bei diesem Problem kommen Versionsverwaltungsprogramme zum Einsatz. Mit diesen werden neben den reinen Veränderungen noch weitere Informationen zu einer Version gespeichert. Darunter fallen in der Regel der Autorenname, die Uhrzeit der Änderung und eine Änderungsnotiz. Diese werden bei jeder neuen

Version gespeichert. Durch die gesammelten Daten können Sie so schnell und einfach eine Änderungshistorie ansehen und verwalten. Falls zwischendurch Fehler in den versionierten Dateien eingeflossen sind, können Sie leicht untersuchen, wann und durch welche Person die Fehler eingeführt wurden, und diese wieder rückgängig machen. Versionsverwaltungsprogramme lassen sich demnach nicht nur von einzelnen Personen nutzen, sondern ermöglichen das Arbeiten im Team mit mehr als einer Person.

Mit Versionsverwaltungsprogrammen lassen sich alle möglichen Dateitypen verwalten. Sie sollten allerdings beachten, dass eine Versionierung nicht für jeden Dateityp praktikabel ist. Besonders hilfreich sind solche Anwendungen vor allem für Arbeiten mit reinen Text-Dateien. Darunter fallen insbesondere Quellcode von Programmen, Konfigurationsdateien oder auch Texte und somit auch Bücher. Der Vorteil bei reinen Textdateien ist, dass Sie die Unterschiede bei Änderungen für jede Zeile nachvollziehen können – das ist bei binären Dateiformaten nicht möglich. Auch für Grafiker kann der Einsatz eines Versionsverwaltungsprogramms sinnvoll sein, denn mit zusätzlichen Tools können auch die Veränderungen zwischen zwei Versionen von Bildern dargestellt werden.

Insgesamt gibt es drei verschiedene Konzepte zur Versionsverwaltung: die lokale, die zentrale und die verteilte Versionsverwaltung.

1.1 Lokale Versionsverwaltung

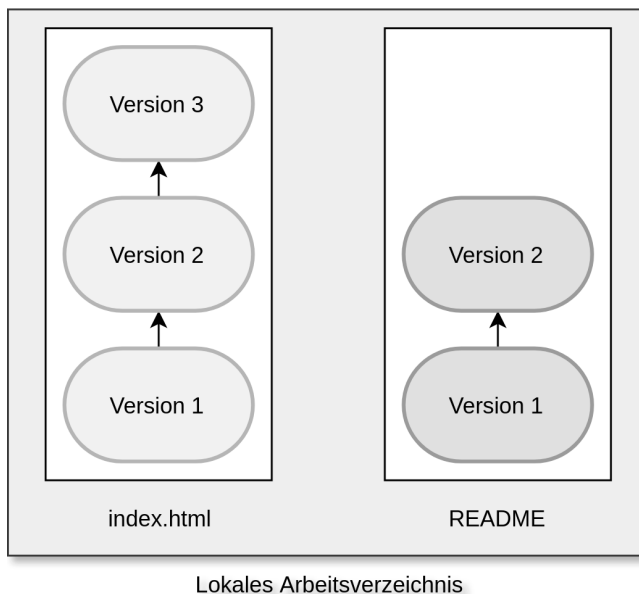


Abb. 1.1: Lokale Versionsverwaltung arbeitet Datei-basiert und lediglich lokal.

Die lokale Versionsverwaltung findet sich eher seltener in produktiven Umgebungen, da sie lediglich lokal arbeitet und häufig nur einzelne Dateien versioniert. Die zuvor erwähnte manuelle Erzeugung von Versionen von Dateien wäre zum Beispiel eine lokale Versionsverwaltung mit einer einzelnen Datei. Sie ist zwar einfach zu nutzen, doch ist es fehleranfällig und wenig flexibel. Echte Versionsverwaltungssoftware, die nur lokal arbeitet, gibt es allerdings auch, darunter »SCSS« und »RCS«. Der größte Nachteil lokaler Versionsverwaltung ist, dass im Normalfall nur eine Person mit den Dateien arbeiten kann, da diese nur lokal auf dem einen Gerät verfügbar sind. Weiterhin besteht keine Datensicherheit, da die Dateien nicht automatisch auf einem anderen Gerät gesichert werden. Der Anwender ist somit allein verantwortlich für ein Backup der Dateien inklusive der Versionshistorie.

1.2 Zentrale Versionsverwaltung

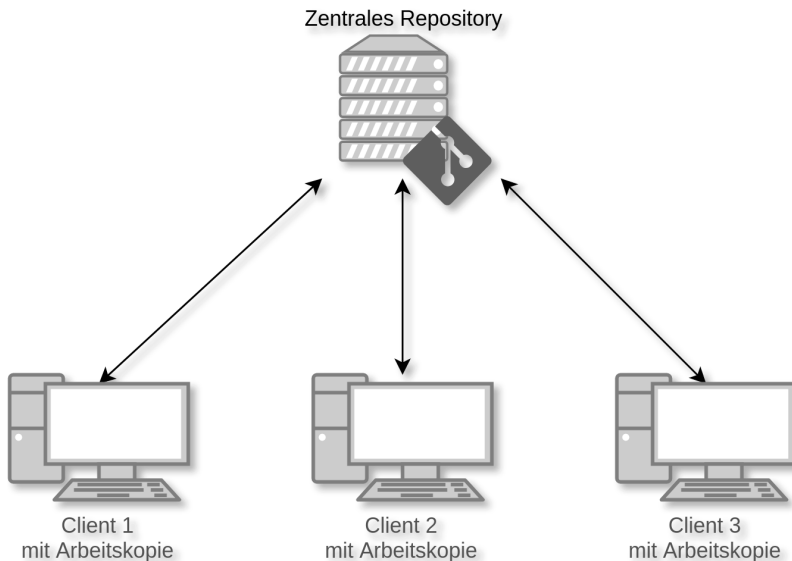


Abb. 1.2: Zentrale Versionsverwaltung arbeitet mit Arbeitskopien auf Clients.

Zentrale Versionsverwaltungen befinden sich heute vergleichsweise noch häufig im Einsatz. Bekannte und verbreitete Vertreter dieser Art sind Subversion und CVS. Das Hauptmerkmal zentraler Versionsverwaltungen ist, dass das Repository lediglich auf einem zentralen Server liegt. Das Wort »Repository« ist Englisch und steht für »Lager«, »Depot« oder auch »Quelle«. Ein Repository ist somit ein Lager, in dem die versionierten Dateien liegen. Autorisierte Nutzer verfügen über eine lokale Arbeitskopie einer Version, auf der sie ihre Arbeiten erledigen.

Die Logik und die Daten der Versionsverwaltung liegen größtenteils auf dem zentralen Server. Beim Wechsel von Revisionen oder beim Vergleichen von Änderungen wird stets mit dem Server kommuniziert. Wenn der Server also offline ist, kann der Nutzer zwar mit der Arbeitskopie ein wenig weiterarbeiten. Allerdings ist die Einsicht älterer Versionen oder das Ansehen anderer Entwicklungslinien nicht möglich, da es sich lediglich um eine Arbeitskopie einer Version und keine Kopie des vollständigen Repositories handelt.

1.3 Verteilte Versionsverwaltung

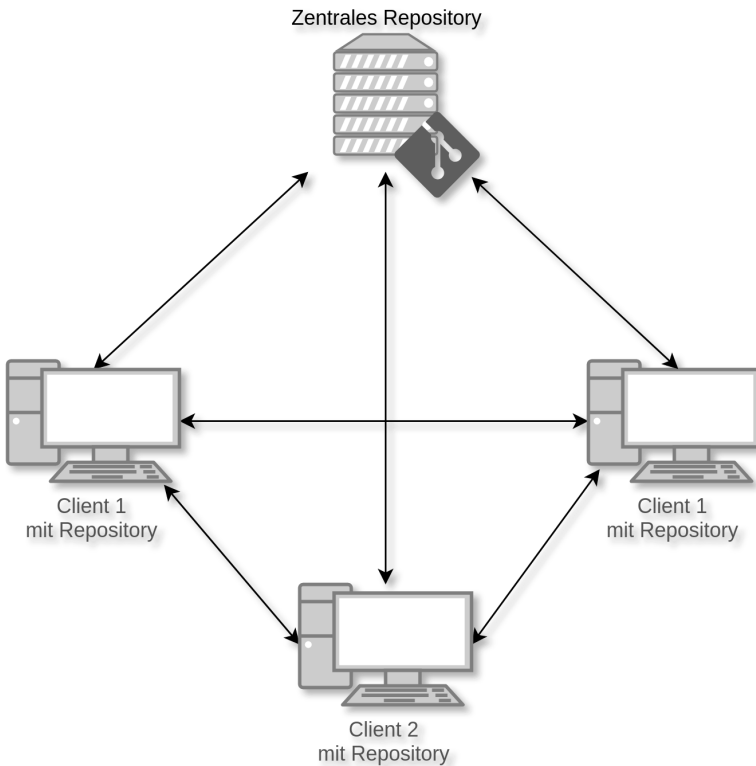


Abb. 1.3: Verteilte Versionsverwaltung arbeitet mit Repositories auf Clients und Servern.

Git gehört zu den verteilt arbeitenden Versionsverwaltungsprogrammen. Neben Git gibt es auch andere verteilte Versionskontrollprogramme, wie Bazaar oder Mercurial. Im Gegensatz zur zentralen Versionsverwaltung besitzt jeder Nutzer des Repositories nicht nur eine Arbeitskopie, sondern das komplette Repository. Wenn Sie also zwischen verschiedenen Revisionen wechseln oder sich die Historie einzelner Dateien anschauen möchte, dann geschieht das Ganze auf dem lokalen

Rechner. Zuvor muss nur das Repository »geklont« werden. Alle Funktionen stehen dann auch offline zur Verfügung. Ein wesentlicher Vorteil davon ist, dass nicht nur unnötiger Datenverkehr vermieden wird, sondern auch die Geschwindigkeit deutlich höher ist, was durch die fehlende Netzwerklatenz bedingt ist.

Zusätzlich besitzen verteilte Versionsverwaltungssysteme eine höhere Datenausfallsicherheit, da die Kopien der Daten des Repositories in der Regel auf verschiedenen Rechnern liegen. Bei einem Ausfall des Git-Servers ist es daher möglich, weiterzuarbeiten. Nichtsdestotrotz sollten Sie von wichtigen Daten natürlich immer Backups anfertigen, ganz egal ob es sich um lokale, zentrale oder verteilte Versionsverwaltung handelt.

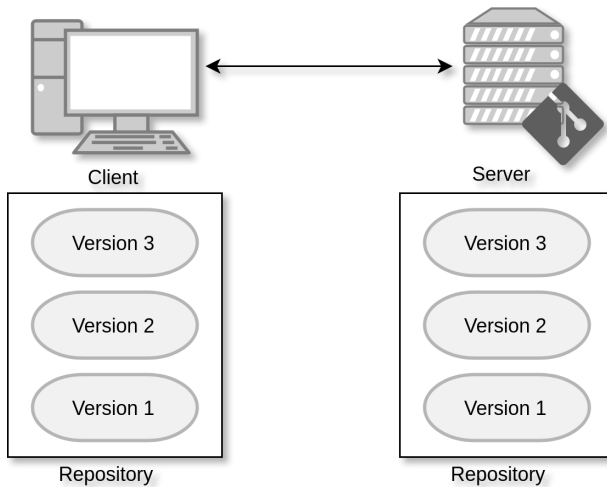


Abb. 1.4: Die Versionshistorie liegt sowohl lokal auf dem Client als auch auf dem Server.

Um den Unterschied zwischen zentralen und verteilten Versionsverwaltungsprogrammen klarer zu machen, kann folgendes Beispiel helfen. Stellen Sie sich vor, dass das Repository ein dicker Aktenordner ist. Darin enthalten sind alle aktuellen Dateien, ältere Versionen der Dateien sowie die Änderungshistorie mitsamt den Kommentaren zu den Änderungen. Sie müssen mit diesen Dateien arbeiten. Wenn es sich um ein zentrales System handelt, dann befindet sich der Aktenordner an einer zentral zugänglichen Stelle, die hier nun Archiv genannt wird. Für Sie heißt es, dass Sie zum Archiv und zu dem Ordner gehen müssen. Dort wird dann eine Arbeitskopie der benötigten Dateien erzeugt und anschließend laufen Sie wieder zurück zum Arbeitsplatz. Wenn Sie die Änderungshistorie von einer oder mehreren Dateien ansehen möchten, müssen Sie immer wieder zum Archiv laufen und den Aktenordner durchblättern, um sich diese anzusehen. Da es sowohl Zeit als auch Energie kostet, immer zum zentralen Aktenordner zu laufen, bietet

es sich an, eine Kopie des ganzen Ordners zu erstellen und mit an Ihren Arbeitsplatz zu nehmen.

Genau das ist dann eine verteilte Versionsverwaltung, da nun zwei vollständige Kopien des Aktenordners existieren – einmal an zentraler Stelle im Archiv und einmal am eigenen Arbeitsplatz. Der Vorteil ist, dass nach der ersten Kopie nur noch die Veränderungen hin- und hergetragen werden müssen. Alles andere kann bequem vom Arbeitsplatz aus gemacht werden, ohne ständig aufzustehen und herumlaufen zu müssen. Konkret bedeutet das, dass Sie an Ihrem Arbeitsplatz sitzen und Ihre Aufgaben erledigen. Sobald die Arbeit abgeschlossen ist, tragen Sie nur die neuen Dateien zum Archiv, wo Sie eine Kopie anfertigen und diese im zentralen Aktenordner abheften. Großer Vorteil ist, dass Sie auch weiterhin arbeiten können, wenn der Weg zum Aktenordner unzugänglich ist, etwa genau dann, wenn Sie unterwegs sind.

Zusammenfassung

- Die lokale Versionsverwaltung funktioniert lediglich auf einem einzelnen Rechner.
- Bei der zentralen Versionsverwaltung liegt das »Gehirn« auf einem zentralen Server, von dem sich alle Mitarbeiter eine Arbeitskopie ziehen können.
- Bei der verteilten Versionsverwaltung liegt das vollständige Repository sowohl auf mindestens einem Server sowie auf allen Clients, wo mit Klonen gearbeitet wird.

1.4 Geschichtliches

Seinen Ursprung hatte Git bei der Entwicklung des Linux-Kernels. Letzterer wurde lange Zeit mit BitKeeper verwaltet, das damals ein proprietäres Programm war. Nachdem die Hersteller von BitKeeper die Lizenz geändert hatten, konnten die Linux-Kernel-Entwickler um Linus Torvalds BitKeeper nicht mehr kostenfrei verwenden, weswegen Linus Torvalds mit der Entwicklung von Git begann. Erst im Mai 2016 wurde BitKeeper unter einer Open-Source-Lizenz veröffentlicht.

Die Entwicklung von Git begann im Jahr 2005 und es gehört somit zu den jüngeren Versionsverwaltungssystemen und das, obwohl es mittlerweile mehr als 15 Jahre alt ist. Linus Torvalds fand es wichtig, dass das zukünftig eingesetzte Programm zur Entwicklung des Linux-Kernels drei spezielle Eigenschaften besitzt. Das sind zum Ersten Arbeitsabläufe, die an BitKeeper angelehnt sind, zum Zweiten die Sicherheit gegen böswillige und unbeabsichtigte Verfälschung des Repositories sowie zum Dritten eine hohe Effizienz. Das Projekt »Monotone« wäre nahezu perfekt für diese Aufgabe gewesen. Das einzige Problem war nur, dass es

nicht sonderlich effizient arbeitete. Letztendlich entschied sich Linus Torvalds für die Entwicklung eines komplett neuen Programms, was er dann Git nannte.

Interessant ist auch die Namensgebung von Git. Das Wort »Git« ist das englische Wort für »Blödmann«. Linus Torvalds selbst sagte spaßeshalber: »I'm an egoistical bastard, and I name all my projects after myself. First ›Linux‹, now ›Git‹.« (Deutsch: »Ich bin ein egoistisches Arschloch und ich benenne alle meine Projekte nach mir selbst. Erst ›Linux‹ und jetzt eben ›Git‹.«). Natürlich gab es auch echte Gründe, das Projekt »Git« zu taufen. Zum einen enthält das Wort lediglich drei Buchstaben, was das regelmäßige Tippen auf der Tastatur erleichtert, zum anderen gab es kein bestehendes UNIX-Kommando, mit dem es kollidieren würde.