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.

1.1.1 Lokale Versionsverwaltung

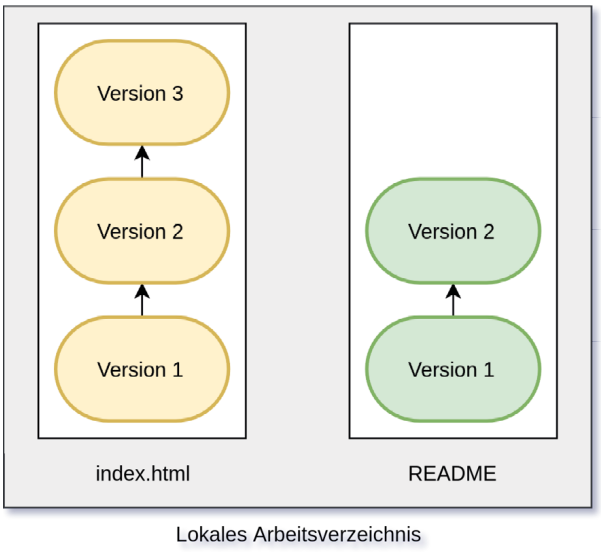

Abb. 1.2: Lokale Versionsverwaltung arbeitet dateibasiert und lediglich lokal.

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.

1.1.2 Zentrale Versionsverwaltung

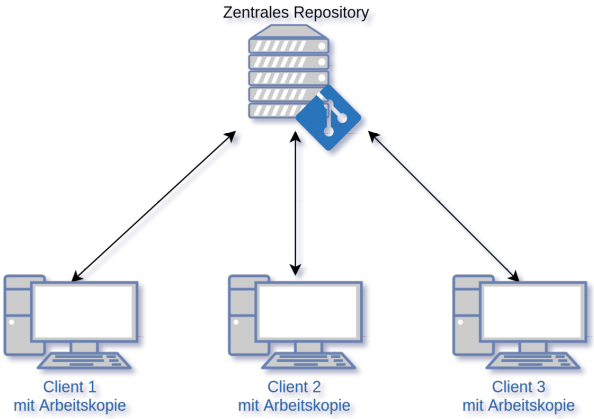

Abb. 1.3: Zentrale Versionsverwaltung arbeitet mit Arbeitskopien auf Clients.

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 Repositorys handelt.

1.1.3 Verteilte Versionsverwaltung

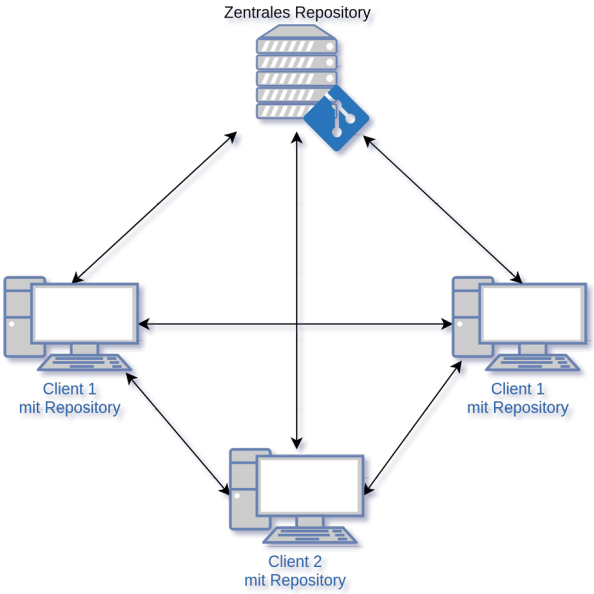

Abb. 1.4: Verteilte Versionsverwaltung arbeitet mit Repositorys auf Clients und Servern.

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 Repositorys 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 Repositorys 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.

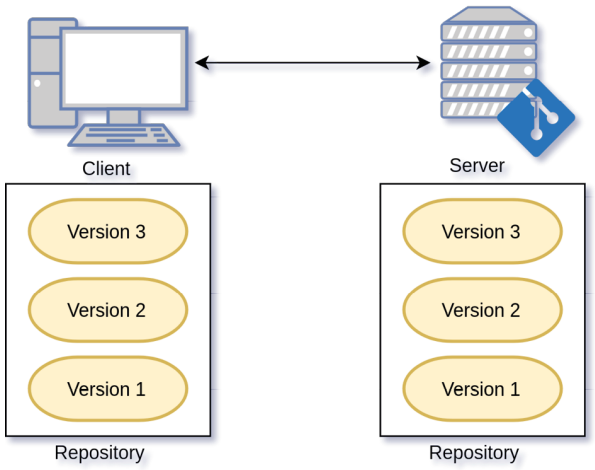

Abb. 1.5: Die Versionshistorie liegt sowohl lokal auf dem Client als auch auf dem Server.

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.

1.2 Git installieren

Bevor Sie losgelegen, müssen Sie Git installieren. Git gibt es nicht nur für die gängigen Betriebssysteme Windows, macOS und Linux, sondern unter anderem auch für FreeBSD, Solaris und sogar Haiku. Die gängigen Linux-Distributionen stellen Git unter dem Paketnamen »git« in der Paketverwaltung zur Verfügung. Nutzer von Windows und macOS können sich Git von der Projektwebsite *https://git-scm.com/downloads* herunterladen.

Während der Arbeit an diesem Buch im Januar 2022 ist die Git-Version 2.34 die neueste Version. Große Unterschiede zu den vorherigen Versionen seit 2.0 existieren hingegen nicht, es sind vielmehr zahlreiche Kleinigkeiten, die über die Zeit eingeflossen sind. Bei Bedarf werden neue Funktionen aus vergleichsweise neuen Versionen hervorgehoben. Gleiches gilt für möglicherweise ältere Versionen von Git, die sich noch in den Paketverwaltungen älterer Linux-Distributionen finden. Obwohl die Version 2.0 von Git schon über acht Jahre alt ist, werden an den ein oder anderen Stellen im Buch noch Unterschiede zu den mittlerweile sehr alten Versionen hervorgehoben. Als Neuankömmling sehen Sie so, was sich getan hat, und wenn Sie nach etlichen Jahren Pausen dann doch wieder Git anfassen, dann geht Ihnen auch nichts verloren.

Die Installation unter Windows ist größtenteils selbsterklärend. So können Sie die Standardeinstellungen beibehalten und bei der Installation entsprechend durchklicken. Etwaig aufploppende Abfragen zu Standardeinstellungen können und werden später im Buch behandelt wie etwa der standardmäßige Branchname beim Anlegen eines Repositorys.

Bei der Nutzung von Git präferiere ich die Git-Bash, da sie im Defaultzustand einige zusätzliche Funktionen bietet, wie die Anzeige des Namens des aktuellen Branches. Außerdem können die gängigen Unix-Kommandos verwendet werden. Alle Befehle in diesem Buch lassen sich problemlos in einer Shell unter macOS und Linux bzw. der Git-Bash unter Windows ausführen. Die Windows-Cmd kann zwar auch verwendet werden, allerdings nenne ich Windows-Cmd-Kommandos nicht noch einmal explizit. Dies ist unter anderem dann relevant, wenn etwa Ordner angelegt oder Dateien verschoben werden sollen.