# Git Handbuch für Einsteiger

*Der leichte Weg zum Git-Experten*

Paul Fuchs

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Informationen sind im Internet über http://dnb.d-nb.de abrufbar.


www.bmu-verlag.de
info@bmu-verlag.de

Lektor: Matthias Kaiser
Einbandgestaltung: Pro ebookcovers Angie
Druck und Bindung: Wydawnictwo Poligraf sp. zo.o. (Polen)

Taschenbuch-ISBN: 978-3-96645-118-5
Hardcover-ISBN: 978-3-96645-119-2
E-Book-ISBN: 978-3-96645-117-8

# Git Handbuch für Einsteiger

# Inhaltsverzeichnis

Kapitel 1

# Einleitung



Das erste Computerprogramm, das Sie erstellt haben, war wahrscheinlich sehr einfach aufgebaut. Die meisten angehenden Programmierer beginnen damit, ein sogenanntes Hallo-Welt-Programm zu gestalten. Dieses enthält neben den Elementen, die für die Einhaltung der formalen Anforderungen der entsprechenden Programmiersprache erforderlich sind, lediglich einen einzigen Ausgabebefehl. Im Laufe der Zeit kamen dann sicherlich viele weitere Elemente hinzu – von Befehlen für die Ablaufsteuerung über Funktionen bis hin zu Klassen und Objekten. Dennoch ist der Umfang der Programme zu Beginn meistens vergleichsweise gering. Die meisten Projekte umfassen in dieser Phase nur eine einzige Datei mit weniger als hundert Codezeilen.

Bei Programmen mit diesem Umfang ist es leicht, den Überblick über den Code zu behalten. Innerhalb weniger Minuten lässt sich das komplette Programm durchgehen – beispielsweise um nach Fehlern zu suchen oder um Ergänzungen daran vorzunehmen. Daher stellt es kein Problem dar, den Code einfach in einem Texteditor zu erstellen und die Änderungen abzuspeichern.

Im Laufe der Zeit werden die Programme aber immer umfangreicher. Selbst Hobby-Programmierer mit etwas Erfahrung erstellen häufig Programme mit mehreren Hundert Codezeilen, die außerdem mehrere Dateien umfassen. Im Studium oder bei der professionellen Softwareentwicklung ist die Länge meistens nochmals deutlich höher. Deswegen ist es schwierig, den Überblick über das Programm beizubehalten. Das kann bei einer Überarbeitung oder einer Erweiterung zu Fehlern führen. Beispielsweise kommt es häufig vor, dass sich eine Änderung in einem bestimmten Programmteil auch auf weitere Bereiche auswirkt. Solche Nebenwirkungen sind in der Regel allerdings nicht erwünscht. Daher ist es notwendig, die Anpassungen wieder rückgängig zu machen. Bei einfachen Programmen mit wenigen Zeilen stellt diese Aufgabe normalerweise kein Problem dar. Falls es doch einmal nicht möglich sein sollte, die Änderungen wieder zurückzunehmen,

ist es außerdem nur mit einem geringen Aufwand verbunden, den Code neu zu erstellen. Bei umfangreichen Programmen stellt es jedoch oftmals ein enormes Problem dar, die letzten Arbeitsschritte rückgängig zu machen. Häufig erinnern Sie sich nicht mehr genau daran, welche Änderungen Sie genau vorgenommen haben. Das kann zu weiteren Fehlern führen, die manchmal zur Folge haben, dass das komplette Programm nicht mehr richtig funktioniert. Es von Grund auf neu zu erstellen, wäre jedoch mit einem enormen Aufwand verbunden – und in einem professionellen Umfeld mit einem erheblichen finanziellen Verlust.

Aus diesem Grund ist es bei umfangreichen Programmen sehr wichtig, die einzelnen Versionen separat abzuspeichern und nicht zu überschreiben. Sollten bei der weiteren Arbeit an dem Programm Fehler auftreten, die sich nicht korrigieren lassen, stellt es kein Problem dar, zu einer vorherigen Version zurückzukehren. Das kann viel Arbeit ersparen und den Entwicklungsprozess beschleunigen.

Eine Möglichkeit besteht nun darin, die einzelnen Versionen immer wieder manuell unter verschiedenen Namen abzuspeichern. Doch das würde einen erheblichen Zusatzaufwand mit sich bringen. Außerdem ist es auf diese Weise schwierig, den Überblick zu behalten. Daher ist es besser, eine Software für die automatische *Versionsverwaltung* zu verwenden. Diese speichert die einzelnen Versionen automatisch ab. Außerdem verfügt sie über viele Funktionen, die es ganz einfach machen, ältere Versionen wiederherzustellen, verschiedene Entwicklungszweige zu erstellen und diese später wieder zusammenzuführen. Das stellt eine starke Erleichterung bei der Programmentwicklung dar.

Besonders häufig treten die genannten Probleme auf, wenn mehrere Programmierer gemeinsam an einem Projekt arbeiten – so wie dies bei den meisten professionellen Software-Projekten üblich ist. In diesem Fall kommt es oftmals zu Interferenzen zwischen den Änderungen verschiedener Mitarbeiter. Programme für die Versionsverwaltung erlauben es jedoch auch, unterschiedliche Entwicklungszweige zu vereinen und dabei die Änderungen verschiedener Programmierer zu übernehmen. Daher ist eine zuverlässige und effiziente Versionsverwaltungssoftware bei Projekten mit mehreren Entwicklern besonders wichtig.

Es gibt viele verschiedene Angebote für die Versionsverwaltung. Dieses Buch befasst sich mit der Software Git. Dabei handelt es sich nicht nur um das am häufigsten verwendete System in diesem Bereich. Darüber hinaus handelt es sich hierbei um ein Open-Source-Programm, das Sie unentgeltlich nutzen können. Das gestaltet die Verwendung von Git ausgesprochen einfach.



## 1.1 Versionsverwaltung – Was ist das?

Git ist eine Software für die Versionsverwaltung. Um diese besser zu verstehen, ist es wichtig, zunächst der Frage nachzugehen, was sich hinter dem Begriff „Versionsverwaltung" verbirgt. Daher befasst sich dieses Kapitel damit, was eine Versionsverwaltung ist und für welche Aufgaben sie sich einsetzen lässt.

Die Versionsverwaltung ist ein System, das dazu dient, unterschiedliche Versionen von *Dateien* und Dokumenten zu erfassen. Meistens handelt es sich hierbei um Dateien, die den Quellcode von Computerprogrammen enthalten. Daher kommt die Versionsverwaltung meistens für die Softwareentwicklung zum Einsatz. Doch kommen auch andere Verwendungsmöglichkeiten infrage. Beispielsweise ist es möglich, mit der Versionsverwaltung Dokumente in einem Büro zu organisieren. Auch bei Wikis oder Content-Management-Systemen, die für die Gestaltung von Webseiten zum Einsatz kommen, handelt es sich um Softwareprodukte, die die Anforderungen einer Versionsverwaltung erfüllen. Das zeigt, dass die Anwendungsmöglichkeiten hierbei sehr vielfältig sind.

### Die wesentlichen Aufgaben der Versionsverwaltung

Im nächsten Schritt ist es notwendig, sich mit den Aufgaben auseinanderzusetzen, die die Versionsverwaltung erfüllen muss.

Von zentraler Bedeutung ist hierbei die Archivierung der Versionen der einzelnen Dateien. Die Versionsverwaltungssoftware muss alle Versionen archivieren, die im Laufe der Zeit entstehen. Um Speicherplatz zu sparen, registrieren die meisten Systeme für die Versionsverwaltung hierbei jedoch lediglich die Änderungen, die im Vergleich zur vorherigen Version auftreten.

Die Archivierung der unterschiedlichen Versionen macht es möglich, den Entwicklungsprozess nachzuvollziehen. Sie stellt eine grundlegende Voraussetzung für alle übrigen Aufgaben dar, die die Versionsverwaltung erfüllt.

Eine weitere wichtige Aufgabe bei der Versionsverwaltung besteht darin, die Veränderungen zu protokollieren. Bei jeder Version, die das System abspeichert, muss eindeutig vermerkt werden, wer die Veränderungen vorgenommen hat und zu welchem Zeitpunkt diese erfolgt sind. Das ist für die Kontrolle der Arbeiten unverzichtbar. Auf diese Weise können die Verantwortlichen bei einem Projekt erkennen, welcher Mitarbeiter in welchem Umfang an der Entwicklung teilgenommen hat. Außerdem können sie auf dieser Grundlage besser entscheiden, welche Änderungen in die endgültige Ausführung der Software übernommen werden sollen.

Die Versionsverwaltung gibt stets eine aktuelle Version vor. Dabei handelt es sich um das Dokument, das den neuesten Stand der Entwicklung anzeigt und das als Grundlage für die weitere Arbeit zum Einsatz kommt. Sollte es sich jedoch herausstellen, dass die neueste Version nicht brauchbar ist, muss es möglich sein, eine der früheren Ausführungen wiederherzustellen und als aktuellen Stand zu markieren. Auf diese Weise lassen sich fehlerhafte Entwicklungen problemlos rückgängig machen.

Schließlich muss es die Versionsverwaltung ermöglichen, die Arbeiten an einem gemeinsamen Projekt zu koordinieren. Die bereits angesprochene Protokollierung stellt hierfür eine wichtige Grundlage dar. So ist es möglich, nachzuvollziehen, welcher Mitarbeiter welche Änderungen vorgenommen hat. Darüber hinaus ist es notwendig, die Rechte zu verwalten. Es muss genau vorgegeben sein, welche Teilnehmer die Dateien bearbeiten dürfen und welche nicht. Außerdem ist es erforderlich, dass klar geregelt ist, wer das letzte Wort bei Konflikten oder bei der Bestimmung der Hauptversion hat.

### Abspaltungen

Von großer Bedeutung sind auch Abspaltungen. Diese erlauben es, unterschiedliche Entwicklungszweige zu erstellen. Die Gründe dafür, dass solche Zweige notwendig sind, können vielfältig sein. Eine Möglichkeit besteht darin, dass experimentelle Entwicklungszweige erstellt werden sollen, bei

denen es nicht sicher ist, dass das gewünschte Ergebnis eintritt. Außerdem ist es möglich, auf diese Weise verschiedene Versionen einer Software zu entwickeln oder ganz neue Projekte zu gestalten. Dabei gibt es zwei unterschiedliche Möglichkeiten, die als *Branch* (Ast) und *Fork* (Gabel) bezeichnet werden.



Branches werden innerhalb des gleichen *Repositorys* erzeugt. Das bedeutet, dass sie nach wie vor Teil des gemeinsamen Projekts bleiben. Die letztendliche Entscheidungsgewalt über die Inhalte liegt daher bei der Person, die für das Gesamtprojekt verantwortlich ist. In einigen Fällen ist es beabsichtigt, dass jeder Zweig zu einem eigenen Ergebnis führt – beispielsweise um unterschiedliche Ausführungen einer Software zu erstellen. Häufig dienen die Zweige jedoch nur der internen Strukturierung des Projekts. Das Ziel besteht darin, abschließend eine einheitliche Version zu erstellen. In diesem Fall ist es notwendig, die einzelnen Zweige zusammenzuführen. Dabei muss es möglich sein, zu entscheiden, welche Änderungen der einzelnen Zweige in die gemeinsame Version übernommen werden und welche nicht. Diese Aufgabe wird als Merge bezeichnet und stellt ebenfalls einen wichtigen Bestandteil eines Systems für die Versionsverwaltung dar.

Eine weitere Möglichkeit besteht darin, Forks zu erstellen. Dabei handelt es sich um eine vollständige Abspaltung. Daher befindet sich der Fork in einem eigenen Repository. Die Versionen im ursprünglichen Repository stellen hierbei lediglich den Ausgangspunkt für die weitere Arbeit dar. Die Entwicklung erfolgt daraufhin vollkommen unabhängig. Da es sich hierbei um ein eigenes Repository handelt, können die Rechte auch vollkommen individuell vergeben werden. Forks kommen insbesondere bei der Entwicklung von Open-Source-Projekten zum Einsatz. Bei kommerzieller Software ist dies weniger verbreitet, da für das Erstellen eines eigenen Forks auch eine Übertragung der Rechte erforderlich wäre.

### Lokale, zentrale und verteilte Versionsverwaltung

Ein weiterer Aspekt, der bei der Versionsverwaltung zu berücksichtigen ist, besteht in der Organisationsform des Repositorys. Die ersten Systeme, die hierfür entwickelt wurden, verfügten über ein lokales Repository. Das bedeutet, dass alle Versionen lediglich auf dem eigenen Rechner gespeichert wurden. Daraus folgt, dass sich diese Systeme nur für einzelne Anwender

eignen – nicht jedoch für die Zusammenarbeit in einem größeren Team. Dennoch gibt es bis heute Programme, die eine solche lokale Umsetzung der Versionsverwaltung verwenden. Insbesondere bei Büro-Software ist dies verbreitet.

Mit der zunehmenden Verbreitung des Internets wurde jedoch schnell klar, dass diese Technik ein enormes Verbesserungspotenzial für die Versionsverwaltung bot. Auf diese Weise entstanden Versionsverwaltungssysteme, die ein zentrales Repository nutzen. Dieses wird auf einem Server abgelegt, der allen Mitarbeiter einen Zugang ermöglicht. Die verschiedenen Anwender können dann ihren Rechten entsprechend darauf zugreifen. Das erleichtert die Zusammenarbeit erheblich. Allerdings fallen dadurch Zusatzkosten für den Betrieb eines Servers an.

Eine weitere Möglichkeit besteht darin, eine geteilte Versionsverwaltung zu verwenden. Das bedeutet, dass jedes Mitglied des Teams über ein eigenes Repository auf seinem Rechner verfügt. Allerdings ist es jederzeit möglich, die eigenen Daten mit den Versionen der übrigen Teammitglieder abzugleichen. Dabei ist es üblich, ein Haupt-Repository zu bestimmen, das den offiziellen Entwicklungsstand widerspiegelt. Diese Form der Versionsverwaltung bietet verschiedene Vorteile. Sie macht es nicht nur möglich, auch ohne Internetanschluss an einem Projekt weiterzuarbeiten. Darüber hinaus ist es auf diese Weise deutlich einfacher, Konflikte zu lösen, die bei einer parallelen Bearbeitung durch verschiedene Mitarbeiter entstehen können. Bei Git handelt es sich um ein System mit verteilten Repositories.

## 1.2 Die Entstehung von Git

Es gibt viele verschiedene Angebote für die Versionsverwaltung. Die ersten Vorläufer entstanden bereits in den 60er-Jahren. Die erste Software, die als vollständige Versionsverwaltung bezeichnet werden kann, kam 1972 auf den Markt: SCCS (Source Code Control System), die von den Bell Laboratories entwickelt wurde. Bei Git handelt es sich hingegen um eine neuere Entwicklung – die erste Version erschien im Jahre 2005. In den folgenden Abschnitten wollen wir einen kurzen Blick auf die Entstehungsgeschichte dieses Systems werfen.

Git entstand im Rahmen der Arbeit am Betriebssystem Linux. Dieses hatte Linus Torvalds zu Beginn der 90er-Jahre entwickelt. Mittlerweile handelt es sich hierbei um eines der am weitesten verbreiteten Betriebssysteme. Insbesondere für den Betrieb von Servern kommt es häufig zum Einsatz. Bei Linux handelt es sich um eine Open-Source-Software. Das bedeutet, dass der Quellcode des Systems offen zugänglich ist und dass für die Benutzung keine Gebühren anfallen. Die Arbeit am Betriebssystemkern – dem sogenannten Linux-Kernel – findet auf freiwilliger Basis statt. Die Entwickler erhalten hierfür kein Gehalt. Zwar gibt es auch einige Firmen, die Angestellte für diese Aufgabe abstellen. Auch viele Mitarbeiter von Universitäten und Forschungseinrichtungen wirken daran mit. Deshalb kommt es hierbei häufig zu einer indirekten Entlohnung. Große Teile des Entwickler-Teams erledigen diese Aufgabe jedoch vollkommen unentgeltlich.

Bei der Entwicklung des Linux-Kernels handelt es sich um ein ausgesprochen umfangreiches Projekt. Deshalb stellt hierbei die Verwendung einer Versionsverwaltung eine Selbstverständlichkeit dar. Das Linux-Entwicklerteam verwendete hierfür zunächst die Software BitKeeper. Hierbei handelt es sich um eine proprietäre Software. Das bedeutet, dass für die Nutzung Lizenzgebühren anfallen. Allerdings stellte damals das Unternehmen BitMover Inc., das BitKeeper vertreibt, die Software für Open-Source-Projekte kostenlos zur Verfügung. Daher fielen für die Verwendung bei der Entwicklung des Linux-Kernels keine Gebühren an.

Dies änderte sich jedoch im Jahre 2005. Damals kündigte das Unternehmen an, dass fortan die Ausnahmeregelung für Open-Source-Projekte aufgehoben werden sollte. Als Folge wären für jeden Entwickler, der am Linux-Kernel mitarbeitete, erhebliche Kosten entstanden. Vor dem Hintergrund, dass es sich hierbei um ein nicht-kommerzielles Projekt handelt und dass die Mitarbeiter diese Aufgabe unentgeltlich erledigen, war diese neue Regelung für das Linux-Team nicht akzeptabel.

Aus diesem Grund beschloss Linus Torvalds, der nach wie vor hauptverantwortlich für die Entwicklung des Linux-Kernels war, ein eigenes System für die Versionsverwaltung zu erstellen.

Zwar gab es zu dieser Zeit bereits einige freie Versionsverwaltungssysteme. Recht bekannt war beispielsweise CVS (Concurrent Version System).

Torvalds stellte sich jedoch entschieden gegen die Verwendung von CVS. Einige Jahre später sagte er, dass er CVS als Negativbeispiel für die Entwicklung von Git verwendete. Er sagte, CVS sei ein hervorragendes Beispiel dafür, was man nicht tun sollte. Wenn einmal Zweifel bei der Entwicklung auftraten, war es stets ratsam, genau das Gegenteil von CVS zu tun. Darüber hinaus gab es das Versionsverwaltungssystem Monotone. Dieses entsprach zwar bereits deutlich stärker den Vorstellungen von Torvalds. Dennoch entschied er sich letztendlich gegen eine Weiterentwicklung dieser Software. Ein Grund hierfür war die mangelhafte Effizienz. Ein weiterer Grund bestand laut Torvalds in der grundlegenden Konzeption des Systems, das laut Torvalds die Entwickler dazu verleitete, unordentliche Repositories zu erstellen.

Aus den genannten Gründen beschloss Torvalds, eine vollkommen neue Software zu erstellen. Die Entwicklung der ersten Version dauerte nur wenige Tage. Die grundsätzliche Konzeption orientierte sich dabei an BitKeeper. Bei der neuen Versionsverwaltung sollte es sich ebenfalls um ein verteiltes System handeln. Außerdem sollten die Arbeitsabläufe ähnlich wie bei BitKeeper sein. Eine der wichtigsten Anforderungen bestand außerdem darin, dass das System eine hohe Sicherheit gegen Verfälschungen bietet – sowohl unbeabsichtigter als auch böswilliger Natur.

Der Name Git bedeutet übersetzt etwa Blödmann oder Idiot. Torvalds verkündete die Namensgebung mit den folgenden Worten:

*„I'm an egotistical bastard, and I name all my projects after myself. First 'Linux', now 'Git'."*

Der Hintergrund dieses Witzes besteht darin, dass Torvalds mit der Namensgebung des Linux-Betriebssystems ursprünglich nicht einverstanden war. Er hielt die Anlehnung an seinen eigenen Vornamen als zu egozentrisch. Allerdings setzte sich dieser Name gegen seinen Willen durch. So kam der Witz zustande, dass er auch sein zweites großes Projekt nach sich selbst benennen wollte und ihm deshalb den Namen „Blödmann" gab. Darüber hinaus gab es jedoch noch weitere Vorteile, die für diese Bezeichnung sprachen: Git ist kurz, einfach auszusprechen und schnell auf der Tastatur zu schreiben. Außerdem war dieser Begriff noch nicht in Verwendung.

## 1.3 Was zeichnet Git aus?

Zum Abschluss des Einleitungskapitels wollen wir uns noch mit den wesentlichen Eigenschaften von Git auseinandersetzen. Auf diese Weise erhalten Sie bereits einen ersten Überblick über die Möglichkeiten, die dieses System bietet. Außerdem erfahren Sie, wie es sich gegenüber anderen Alternativen für die Versionsverwaltung abgrenzt und welche Vorteile die Nutzung von Git bietet.



### Open Source Software

Eine der wesentlichen Eigenschaften dieser Software besteht darin, dass sie quellenoffen ist. Das bringt den Vorteil mit sich, dass Sie sie vollkommen unentgeltlich nutzen können. Wenn Sie die Aufgaben und Beispiele in diesem Buch erledigen möchten, können Sie daher Git problemlos installieren, ohne dass hierfür zusätzliche Kosten anfallen.

Open-Source-Software bringt außerdem einen weiteren Vorteil mit sich: Der komplette Quellcode liegt offen. Das macht es möglich, diesen selbst zu betrachten und sich einen Überblick über die Implementierung zu verschaffen. Wenn Sie über die entsprechenden Programmierkenntnisse verfügen, können Sie den Code sogar verändern oder erweitern und auf diese Weise Ihre eigene personalisierte Version erstellen.

### Verteiltes System

Bei der Vorstellung der Versionsverwaltung wurde bereits erwähnt, dass es hierbei lokale, zentrale und verteilte Systeme gibt. Bei Git handelt es sich um ein verteiltes System. Das bedeutet, dass zwar jeder Anwender über seine eigene Kopie auf seinem Rechner verfügt. Allerdings ist es jederzeit möglich, diese mit einem anderen Repository, das sich auf einem über das Internet zugänglichen Server befindet, zu synchronisieren. Dabei ist das Entwicklungsteam nicht auf ein einziges Repository auf einem zentralen Server beschränkt. Es ist auch möglich, mehrere Repositorys auf unterschiedlichen Servern und Online-Plattformen abzulegen. Wenn beispielsweise jeder Mitarbeiter seine Arbeiten online zur Verfügung stellt, können die übrigen Team-Mitglieder auch direkt darauf zugreifen – ohne den Umweg über einen zentralen Server.

Für die meisten Projekte ist es allerdings notwendig, eine offizielle Version zu bestimmen. Diese ist aus technischer Sicht jedoch genau gleich gestaltet wie die Repositories der übrigen Mitarbeiter. Es ist lediglich wichtig, dass alle Team-Mitglieder wissen, unter welcher Adresse dieses offizielle Repository erreichbar ist, damit sie ihre eigenen Daten mit diesem synchronisieren können. Wenn ein Mitarbeiter nun seine Arbeit in dieses offizielle Repository übertragen will, kann er die Änderungsvorschläge dem Projektverantwortlichen zum einen per E-Mail zuschicken. Zum anderen ist es möglich, dass die Mitarbeiter das Recht erhalten, ihre Arbeiten direkt in das offizielle Repository einzufügen. Häufig kommen hierfür separate Zweige zum Einsatz. Der Teamleiter kann die Vorschläge dann begutachten und in den Hauptzweig übernehmen, falls sie die Anforderungen erfüllen.

### Verzweigungen

Die Möglichkeit, Verzweigungen zu erstellen, ist bei den meisten Versionsverwaltungssystemen gegeben. Git legt hierauf jedoch einen besonderen Wert. Diese Software macht es besonders einfach, einzelne Zweige zu erstellen und zusammenzuführen. Git enthält Tools für die Visualisierung der Zweige, die es erleichtern, den Überblick zu behalten. Darüber hinaus erledigt es alle Aufgaben rund um das Erstellen von Verzweigungen mit hoher Effizienz. Das verhindert beispielsweise die langen Wartezeiten, die bei anderen Systemen beim Zusammenführen der Zweige bei umfangreichen Projekten üblich sind.

### Eigenes Protokoll für den Datentransfer

Um die Daten mit einem anderen Repository zu synchronisieren, ist es notwendig, sie über das Internet zu übertragen. Hierfür sind Protokolle notwendig, die die Regeln für den Datenaustausch vorgeben. Nur wenn sowohl der Sender als auch der Empfänger das gleiche Protokoll verwenden, ist sichergestellt, dass die Informationen korrekt ankommen. Bekannte Beispiele für solche Protokolle sind HTTP und HTTPS.

Eine Besonderheit von Git besteht darin, dass hierfür ein eigenes Protokoll entwickelt wurde. Dieses zeichnet sich durch eine deutlich höhere Effizienz als die übrigen Alternativen aus. Das führt dazu, dass es auf diese Weise

möglich ist, selbst umfangreiche Projekte schnell zu laden. Allerdings besteht hierbei die Einschränkung, dass es mit diesem Protokoll nur möglich ist, Daten von einem anderen Repository auf den eigenen Rechner zu übertragen. Der umgekehrte Weg wird nicht unterstützt. Das bedeutet, dass es über das Git-Protokoll nicht möglich ist, eigene Entwürfe auf einem entfernten Repository abzulegen. Allerdings unterstützt Git auch alle weiteren gängigen Protokolle: SSH, HTTPS und FTP. Damit ist es auch möglich, Daten in einem Repository abzulegen. Meistens kommt hierfür SSH zum Einsatz. Es steht Ihnen jedoch auch frei, für alle Aktionen ein beliebiges anderes Protokoll zu verwenden.



### Hohe Sicherheit bei der Versionsgeschichte

Git legt einen hohen Wert darauf, eine Manipulation der früheren Versionen zu verhindern. Zu diesem Zweck kommt ein *Hash-Wert* zum Einsatz. Hierbei handelt es sich um eine Nummer, die nach einer bestimmten Funktion aufgrund des Inhalts der entsprechenden Datei ermittelt wird. Das System ist so gewählt, dass es praktisch unmöglich ist, dass zwei verschiedene Dokumente den gleichen Hash-Wert erzeugen.

Beim Speichern einer Version wird nun automatisch der zugehörige Hash-Wert ermittelt und gemeinsam mit dem Dokument festgehalten. Außerdem enthält jede Version den Hash-Wert der Vorgängerversion.

Dieses einfache System stellt einen zuverlässigen Manipulationsschutz dar. Sollte ein Angreifer den Inhalt einer der vorherigen Versionen abändern, muss er hierfür auch den Hash-Wert anpassen. Tut er dies nicht, kommt es zu einer Unstimmigkeit, die das Programm sofort bemerkt. Die nachfolgende Version enthält jedoch den Hash-Wert des ursprünglichen Dokuments. Wenn diese beiden Werte durch die Manipulation nicht mehr übereinstimmen, bemerkt das Programm dies ebenfalls. Daher wäre es auch notwendig, die Nachfolgeversion zu manipulieren. Diese Änderung würde nun jedoch wiederum deren eigenen Hash-Wert verändern. Um die Konsistenz aufrechtzuerhalten, wäre es bei einer Manipulation daher notwendig, alle nachfolgenden Versionen bis hin zur aktuellen Version ebenfalls anzupassen. Das würde nicht nur einen erheblichen Aufwand bedeuten. Darüber hinaus wäre es sehr wahrscheinlich, dass eine solch umfangreiche Manipulation auffällt.

## Interoperabilität mit anderen Programmen für die Versionsverwaltung

Es wurde bereits gezeigt, dass es viele verschiedene Systeme für die Versionsverwaltung gibt. Für Anwender, die in mehreren Teams mitwirken, die hierfür unterschiedliche Software-Produkte verwenden, kann dies Probleme mit sich bringen. Zum einen wäre es notwendig, alle Systeme auf dem Rechner zu installieren. Zum anderen wäre auch eine umfangreiche Einarbeitung erforderlich.

Git bietet jedoch den Vorteil, dass es eine Interoperabilität mit anderen Systemen ermöglicht. Hierfür stehen verschiedene Hilfsprogramme bereit. Diese können die Arbeit erheblich erleichtern. Auf diese Weise lässt sich Git mit vielen verschiedenen anderen Versionsverwaltungssystemen verbinden – beispielsweise mit Subversion, CVS oder GNU arch.

Alle Programmcodes aus diesem Buch sind
zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:
https://bmu-verlag.de/books/git/


Außerdem erhalten Sie die eBook-Ausgabe zum Buch
kostenlos auf unserer Website:

https://bmu-verlag.de/books/git/
**Downloadcode:** siehe Kapitel 14

Kapitel 2

# Die Vorbereitungsmaßnahmen

Bevor wir mit der praktischen Arbeit mit Git beginnen können, sind einige Vorbereitungsmaßnahmen notwendig. Zunächst werden wir uns mit dem *Kommandozeileninterpreter* befassen. Hierbei handelt es sich um ein Werkzeug, mit dessen Hilfe Sie über schriftliche Befehle die Funktionen Ihres Betriebssystems steuern können. Der Kommandozeileninterpreter ist für die Arbeit mit Git unverzichtbar. Daher geben wir zu Beginn eine kurze Einführung in dessen Verwendung.

Bei Git handelt es sich um eine Software, die Sie vor der Anwendung auf Ihrem Rechner installieren müssen. Daher zeigen wir Ihnen auch, wie der Installationsprozess auf verschiedenen Betriebssystemen abläuft. Schließlich befassen wir uns mit der grundlegenden Konfiguration von Git und mit der integrierten Hilfe, die diese Software anbietet.

## 2.1 Der Kommandozeileninterpreter: wichtiges Hilfsmittel für die Arbeit mit Git

Es bestehen unterschiedliche Möglichkeiten, um Git zu verwenden. Die klassische Form besteht darin, einen Kommandozeileninterpreter zu nutzen. Dieser ist auf jedem Betriebssystem vorinstalliert. Moderne Betriebssysteme wie Windows, macOS und die meisten aktuellen Linux-Distributionen verfügen zwar über grafische Benutzeroberflächen, mit denen Sie alle wesentlichen Funktionen des Computers steuern können. Dennoch stellen sie als Alternative einen Kommandozeileninterpreter bereit. Dieser ermöglicht es, die entsprechenden Aufgaben mithilfe schriftlicher Befehle zu erledigen.

Die Verwendung des Kommandozeileninterpreters stellt allerdings nicht die einzige Möglichkeit dar, um Git zu nutzen. Im Laufe der Zeit entstanden auch verschiedene Angebote, die eine grafische Benutzeroberfläche für Git bereitstellen. Obwohl deren Nutzung auf den ersten Blick einfacher

und intuitiver erscheinen mag, arbeiten wir in diesem Buch vorwiegend mit der Kommandozeile. Die Gründe hierfür sind vielfältig. Beispielsweise ist auf diese Weise sichergestellt, dass Sie alle Funktionen von Git nutzen können. Die grafischen Benutzeroberflächen beschränken sich in der Regel auf die wesentlichen Steuerungsmöglichkeiten. Außerdem gibt es viele unterschiedliche Programme mit grafischen Benutzeroberflächen für Git. Die Steuerung und die Darstellung weisen dabei jedoch große Unterschiede auf. Um einen umfassenden Überblick zu gewährleisten, wäre es daher notwendig, die entsprechenden Aktionen jeweils für verschiedene Anwenderprogramme vorzustellen, wodurch sich das Buch deutlich in die Länge ziehen würde. Schließlich gibt es noch einen weiteren praktischen Grund, mit der Kommandozeile zu arbeiten. Wenn Sie sich später für ein Programm mit grafischen Oberflächen entscheiden, sollte die Umstellung auch ohne große Einarbeitung leicht fallen. Umgekehrt ist dies jedoch nicht der Fall. Wenn Sie ein GUI-Programm für die Arbeit mit Git nutzen und später mit der Kommandozeile arbeiten möchten, ist die Umstellung nicht ganz einfach.

Wenn Sie bereits über Programmierkenntnisse verfügen und Git für die Verwaltung Ihrer Programmdateien verwenden möchten, dann sind Sie wahrscheinlich bereits an den Umgang mit der Kommandozeile gewöhnt. Da für die Programmierung grafischer Benutzeroberflächen fortgeschrittene Kenntnisse notwendig sind, befassen sich Anfänger zunächst fast immer mit sogenannten Konsolenprogrammen, die deutlich einfacher aufgebaut sind. Deren gemeinsame Eigenschaft besteht darin, dass sie im Kommandozeileninterpreter ablaufen. Obwohl die meisten Leser daher wahrscheinlich bereits über einige Vorkenntnisse im Umgang mit diesem Werkzeug verfügen, werden alle Befehle, die für die Steuerung notwendig sind, im Verlaufe des Buchs an der entsprechenden Stelle erklärt. Daher ist es nicht notwendig, sich vor der Lektüre in die Verwendung des Kommandozeileninterpreters einzuarbeiten.

Es gibt viele verschiedene Kommandozeileninterpreter. Somit ist es wichtig, auf diesen Aspekt kurz einzugehen und die Angebote in Abhängigkeit vom verwendeten Betriebssystem kurz vorzustellen.

Windows wird standardmäßig mit dem Kommandozeileninterpreter cmd.exe ausgeliefert. In der deutschsprachigen Version wird dieser als Ein-

gabeaufforderung bezeichnet. Sie erreichen dieses Programm beispielsweise, indem Sie das Start-Menü anklicken und daraufhin den Ordner „Windows-System" auswählen. Alternativ dazu können Sie auch den Begriff „cmd" oder „Eingabeaufforderung" in die Suchleiste eingeben. Abbildung 2.1 zeigt, wie dieser Kommandozeileninterpreter gestaltet ist.


**Abb. 2.1** Die Eingabeaufforderung unter Windows

Alternativ dazu hat Microsoft auch das Programm Windows PowerShell entwickelt. Dieses ist unter neueren Windows-Versionen ebenfalls vorinstalliert. Es zeichnet sich durch etwas umfangreichere Befehle als cmd.exe aus und außerdem verwendet es eine andere Scriptsprache. Beide Aspekte sind jedoch für die Arbeit mit Git ohne Bedeutung. Wenn Sie es wünschen, können Sie daher auch die Windows PowerShell verwenden. Diese erreichen Sie ebenfalls über das Start-Menü (im Ordner Windows PowerShell) oder über die Suchfunktion. Abbildung 2.2. zeigt bereits, dass die Gestaltung hierbei recht ähnlich wie bei cmd.exe ist. Die grundlegenden Funktionen sind ebenfalls die gleichen. Bei der Windows PowerShell handelt es sich seit einiger Zeit außerdem um ein plattformübergreifendes Projekt. Daher können Sie sie auch unter macOS oder Linux verwenden. Hier müssen Sie das Programm jedoch separat installieren.

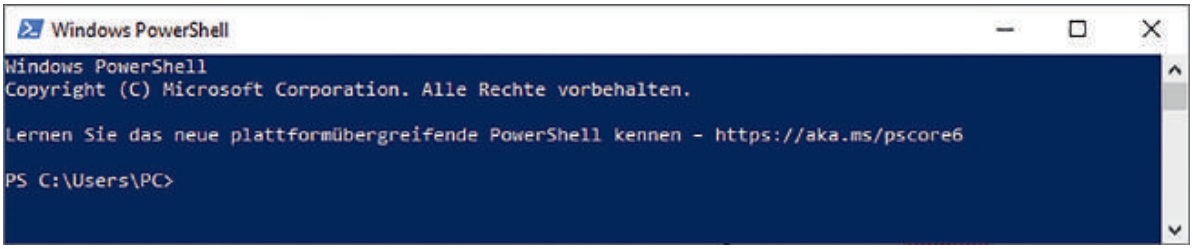


**Abb. 2.2** Die Windows Power Shell

Auch wenn Sie macOS oder Linux verwenden, ist bereits ein passender Kommandozeileninterpreter vorinstalliert. Unter beiden Systemen trägt dieser die Bezeichnung „Terminal". Auf welche Weise Sie diesen erreichen,

hängt vom verwendeten Betriebssystem ab. In jedem Fall sollten Sie das Programm jedoch problemlos finden, wenn Sie den Begriff „Terminal" in die Suchfunktion eingeben. Abbildung 2.3 zeigt das Terminal der Linux-Distribution Ubuntu:



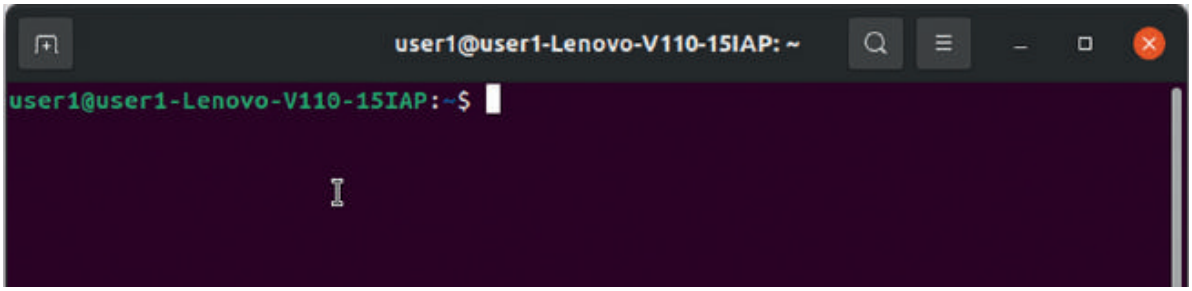


**Abb. 2.3** Das Terminal unter Linux

Bevor Sie mit der Arbeit mit Git beginnen, müssen Sie sich für einen Kommandozeileninterpreter entscheiden. Alle hier vorgestellten Beispiele eignen sich bestens zu diesem Zweck. Bei der Arbeit mit dem Programm kommt es auch zu keinerlei Unterschieden – die Befehle sind hierbei identisch. Lediglich der Installationsprozess läuft dabei auf verschiedene Weise ab. Daher wird dieser im folgenden Kapitel detailliert für die verschiedenen Betriebssysteme vorgestellt.

## 2.2 Git auf verschiedenen Systemen installieren

Bevor wir mit der Arbeit mit Git beginnen können, ist es notwendig, das Programm auf dem Computer zu installieren. Wie Sie dabei vorgehen, hängt vom Betriebssystem ab, das Sie verwenden. Doch selbst bei gleichen Betriebssystemen bestehen unterschiedliche Alternativen. Aus diesem Grund befassen sich die folgenden Abschnitte damit, wie Sie Git auf verschiedenen Systemen installieren können.

### Windows

Git wurde ursprünglich für die Entwicklung des Betriebssystems Linux verwendet. Daher liegt es auf der Hand, dass das Programm für die Nutzung auf einem Linux-Rechner vorgesehen war. Da es hierbei erhebliche Unterschiede zu Windows gibt, war die ursprüngliche Git-Version unter Windows nicht nutzbar.

Da es sich bei Windows um das am weitesten verbreitete Betriebssystem im PC-Bereich handelt, entstanden jedoch bereits nach kurzer Zeit auch hierfür geeignete Versionen. Diese sind allerdings vom eigentlichen Git-Projekt unabhängig. Dennoch sind sie zur ursprünglichen Git-Version kompatibel und auch die Befehlssätze sind dabei identisch.

Auf diese Weise entstanden jedoch mehrere unabhängige Projekte, die eine Portierung für Git auf Windows zum Ziel hatten. Deswegen stehen gerade für dieses Betriebssystem verschiedene Alternativen zur Auswahl. Besonders beliebt ist die Software Git for Windows. Obwohl das Projekt eigentlich unabhängig ist, können Sie die Software auf der offiziellen Git-Homepage herunterladen:

http://git-scm.com/download/win

Bei diesem Download handelt es sich um einen Installer, der den kompletten Installationsprozess organisiert. Dabei können Sie stets die Standard-Einstellungen übernehmen.

Nachdem das Programm installiert ist, können Sie es über die verschiedenen Kommandozeileninterpreter nutzen. Sie können hierfür nicht nur die Eingabeaufforderung oder die Windows PowerShell verwenden. Darüber hinaus wird dieses Programm mit zwei eigenen Kommandozeileninterpretern ausgeliefert. Diese erreichen Sie über das Startmenü im Ordner „Git". Es handelt sich hierbei um Git Bash und Git CMD. Dabei ist insbesondere die Git Bash interessant. Dieses Programm weist die gleichen Eigenschaften wie das Linux-Terminal auf. Das bringt in manchen Situationen Vorteile mit sich. Allerdings können Sie für fast alle Aufgaben auch die vorinstallierten Windows-Kommandozeileninterpreter verwenden. Welche der vorgestellten Alternativen Sie verwenden, bleibt letztendlich Ihnen überlassen.

Um zu überprüfen, ob der Installationsprozess erfolgreich verlaufen ist, können Sie nun einen der genannten Kommandozeileninterpreter öffnen. Geben Sie daraufhin den folgenden Befehl ein:

```
git --version
```

Wenn das Programm korrekt funktioniert, sollte daraufhin angezeigt werden, welche Git-Version Sie soeben installiert haben. Abbildung 2.4 zeigt dies am Beispiel der Eingabeaufforderung cmd.exe:

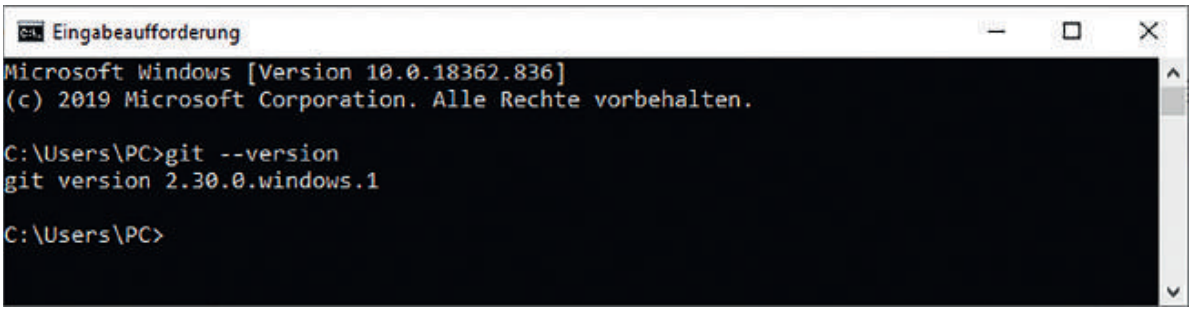


**Abb. 2.4** Die Abfrage der Git-Version



Wie bereits erwähnt, bestehen noch weitere Möglichkeiten, um Git unter Windows zu nutzen. Beispielsweise können Sie Softwarepakete wie Cygwin oder Chocolatey verwenden, bei denen eine Git-Version integriert ist. Eine weitere Möglichkeit stellt GitHub for Windows dar. Wenn Sie diese Möglichkeiten vorziehen, können Sie diese ebenfalls verwenden. Über eine kurze Internetrecherche sollte es möglich sein, die hierfür notwendigen Download-Angebote zu finden. Im weiteren Verlauf dieses Buchs arbeiten wir jedoch mit der soeben vorgestellten Software Git for Windows.

### Linux

Wenn Sie das Betriebssystem Linux verwenden, kann es sein, dass Git bereits vorinstalliert ist. Das hängt von der verwendeten Distribution und von der Version ab. Daher ist es sinnvoll, zunächst einen kleinen Test durchzuführen. Dazu verwenden wir wieder die bereits vorgestellte Versionsabfrage (git --version). Sollte hier eine Versionsnummer angezeigt werden, ist Git bereits verfügbar. Kommt es hingegen zu einer Fehlermeldung, müssen Sie Git separat installieren.

In diesem Fall müssen Sie Git mit dem Paketmanager, den Ihre jeweilige Distribution verwendet, installieren. Unter Debian und allen Derivaten (z.B. Ubuntu und Linux Mint) kommt hierfür der folgende Befehl zum Einsatz:

```
sudo apt install git-all
```

Wenn Sie eine Distribution mit dnf-Paketmanager (z.B. Fedora oder RHEL) nutzen, können Sie diesen Befehl verwenden:

```
sudo dnf install git-all
```

### macOS

Auch für macOS steht eine passende Git-Version zur Verfügung. Diese muss ebenfalls manuell installiert werden. Allerdings gibt es auch einige Programme, die automatisch Git auf dem System installieren. Ein Beispiel hierfür ist die beliebte Entwicklungsumgebung Xcode. Wenn sie diese verwenden, sollte Git bereits verfügbar sein. Daher empfiehlt es sich auch in diesem Fall, die bereits vorgestellte Versionsabfrage durchzuführen. Sollte sich dabei herausstellen, dass Git bereits verfügbar ist, sind keine weiteren Maßnahmen notwendig.

Falls Git noch nicht auf dem System installiert ist, bestehen hierfür ebenfalls verschiedene Möglichkeiten. Die wahrscheinlich einfachste Alternative besteht darin, Git auf der offiziellen Website herunterzuladen und als Stand-Alone-Version zu installieren. Dabei übernimmt ein Installationsassistent den gesamten Prozess. Dieser läuft ähnlich wie bei der Windows-Installation ab. Der Download ist unter folgendem Link möglich:

http://git-scm.com/download/mac

Da es sich bei macOS genau wie bei Linux um ein unix-ähnliches Betriebssystem handelt, können Sie hier übrigens die offizielle Git-Version verwenden, die unter dem genannten Download-Link zur Verfügung steht. Bei der Version, die gemeinsam mit Xcode installiert wird, handelt es sich hingegen um eine leicht abgewandelte Ausführung, die die Zusammenarbeit mit der Entwicklungsumgebung erleichtern soll. Für die Aufgaben in diesem Buch sollten diese geringfügigen Unterschiede jedoch keine Rolle spielen.

## 2.3 Die grundlegende Konfiguration

Git bietet zahlreiche Konfigurationsmöglichkeiten, mit denen Sie die Software an Ihre persönlichen Anforderungen anpassen können. Die meisten Konfigurationen sind nicht zwingend erforderlich. Sie können sie jedoch durchführen, wenn Sie der Meinung sind, dass Sie dadurch Ihre Arbeitsabläufe verbessern können. Allerdings gibt es auch einige Einstellungen, die Sie auf jeden Fall vornehmen sollten, um sinnvoll mit Git arbeiten zu können. Hierbei ist insbesondere die Eingabe Ihres Namens und Ihrer E-Mail-Adresse zu nennen. Dieses Kapitel stellt neben diesen notwendigen Konfigurationen jedoch auch noch einige optionale Konfigurationsmöglichkeiten vor.

## Verschiedene Möglichkeiten für die Konfiguration

Um die Konfiguration von Git vorzunehmen, bestehen verschiedene Möglichkeiten. Die Einstellungen speichert das Programm in zwei verschiedenen Dateien ab. Eine davon befindet sich im Installationsverzeichnis von Git. Sollten Sie die vorgeschlagenen Werte bei der Installation nicht verändert haben, erreichen Sie diese Datei im Windows-Explorer über **Lokaler Datenträger (C:) > Programme > Git** oder über den Kommandozeileninterpreter mit dem Pfad **C:\Programs\Git**. Darin befindet sich der Ordner **etc**, der bei der Installation automatisch erzeugt wurde. Darin ist dann die Konfigurationsdatei **gitconfig** enthalten.

Die zweite Konfigurationsdatei befindet sich in Ihrem Home-Verzeichnis. Dabei handelt es sich um den Ordner, der Ihrem persönlichen Benutzernamen entspricht und der sich wiederum im Verzeichnis **Benutzer** (der im Kommandozeileninterpreter über die englische Bezeichnung **Users** erreichbar ist) befindet. Hier wurde die Datei **.gitconfig** installiert. Der vorangestellte Punkt sagt aus, dass es sich hierbei um eine versteckte Datei handelt. Sollten Sie für den Zugriff den Windows Explorer verwenden, müssen Sie daher beachten, dass die entsprechende Datei normalerweise nicht angezeigt wird. Somit müssen Sie zunächst über die Systemsteuerung vorgeben, dass auch versteckte Ordner angezeigt werden (Darstellung und Anpassung > Versteckte Dateien und Ordner ausblenden).

**Hinweis:** Der hier dargestellte Speicherort der Konfigurationsdateien bezieht sich auf das Betriebssystem Windows. Unter Linux und macOS kommt es dabei zu Abweichungen. Über die Suchfunktion sollten die entsprechenden Dateien jedoch auch unter diesen Betriebssystemen leicht zu finden sein.

Nun wäre es möglich, diese Dateien mit einem Texteditor zu öffnen und die Konfiguration manuell vorzunehmen. Das ist jedoch recht kompliziert. Außerdem besteht dabei ein hohes Fehlerrisiko. Deshalb wählen wir eine andere Alternative für die Konfiguration. Durch entsprechende Befehle können wir die Vorgaben direkt über die Kommandozeile machen. Git speichert die gewünschten Werte dann automatisch in den Konfigurationsdateien.

## Ihre Identität vorgeben

Von großer Bedeutung ist es, Ihre Identität vorzugeben. Lediglich wenn Sie nur alleine an Ihren Projekten arbeiten, können Sie diese Anpassungen vernachlässigen. Wenn Sie jedoch gemeinsam mit anderen Personen an einem Projekt mitwirken, sind diese Angaben von großer Bedeutung. Wenn Sie Änderungen an einer Datei in die Versionsverwaltung übertragen, werden hierbei nicht nur die neuen Inhalte und der Zeitpunkt der entsprechenden Aktion festgehalten. Darüber hinaus vermerkt das Programm, wer die Änderung durchgeführt hat. Das ist für die Kontrolle der Abläufe von großer Bedeutung. Zu diesem Zweck kommen die Angaben zum Einsatz, die Sie in der Konfiguration vorgegeben haben.

Zum einen müssen Sie Ihren Namen eingeben. Öffnen Sie hierfür einen Kommandozeileninterpreter und geben Sie daraufhin den folgenden Befehl ein:

```
git config --global user.name "Mein Name"
```

Ersetzen Sie hierbei den Ausdruck „Mein Name" durch Ihren eigenen Namen. Achten Sie dabei darauf, dass dieser in Anführungszeichen stehen muss. Zum anderen ist für Ihre Identifizierung Ihre E-Mail-Adresse erforderlich. Hierfür können Sie den folgenden Befehl verwenden und dabei ebenfalls die Beispieladresse durch Ihre eigene E-Mail-Adresse ersetzen:

```
git config --global user.email meinemail@beispiel.de
```

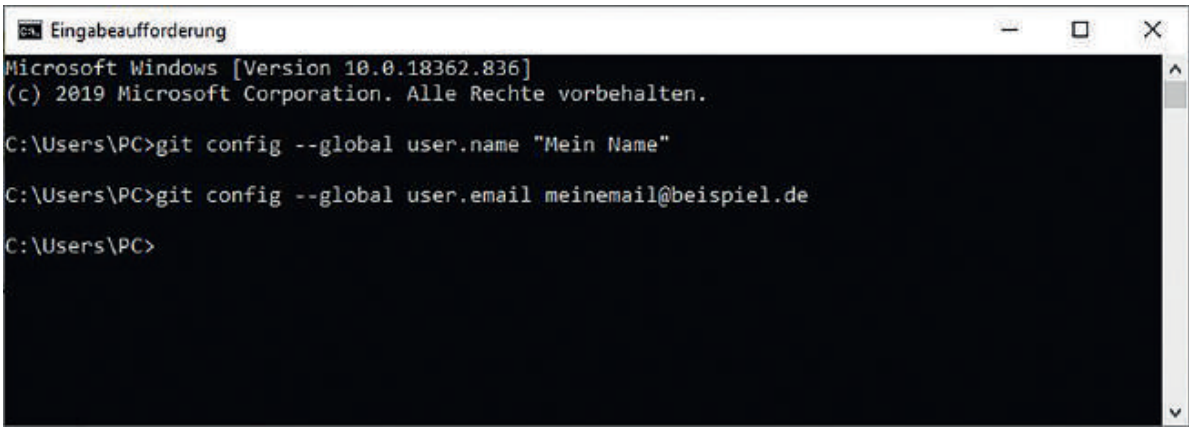


**Abb. 2.5** Die Einstellungen für die Identität

**Hinweis:** Wenn Sie die Einstellungen auf diese Weise vorgeben, haben sie eine globale Gültigkeit. Das bedeutet, dass sie für jedes neue Projekt automatisch übernommen werden. Da die meisten Anwender stets die gleiche Identität verwenden, ist dies sehr praktisch. Es besteht jedoch auch die Möglichkeit, eine Identität nur für ein einzelnes Projekt zu verändern. Dazu müssen Sie dieses zunächst erstellen und öffnen. Wie Sie dabei vorgehen, erfahren Sie im Verlauf der folgenden Kapitel. Wenn Sie sich innerhalb dieses Projekts befinden, können Sie die genannten Befehle ohne den Zusatz `--global` verwenden. In diesem Fall geben Sie die Identität nur für das aktuelle Projekt vor.

### Weitere Konfigurationsmöglichkeiten

Git bietet noch zahlreiche weitere Konfigurationsmöglichkeiten. Im Gegensatz zur Festlegung der Identität sind diese jedoch nicht zwingend notwendig. Dennoch ist es auf diese Weise möglich, die Abläufe des Programms an Ihre Erfordernisse anzupassen.

Eine Möglichkeit besteht beispielsweise darin, einen Texteditor für die Bearbeitung der Dateien zu verwenden. Wenn Sie hier keine Vorgaben machen, verwendet Git den Standard-Texteditor. Dieser wird direkt in der Kommandozeile ausgeführt. Für die Bedienung ist es notwendig, die genauen Befehle für die Steuerung zu kennen. Das erschwert die Bearbeitung. Aus diesem Grund ist es sinnvoll, zunächst einen zusätzlichen Texteditor zu installieren, falls dieser nicht bereits vorhanden sein sollte. Daraufhin können Sie diesen für die Arbeit mit Git vorgeben. Der folgende Befehl ermöglicht es beispielsweise, den Texteditor Geany als Standard für Git vorzugeben:

```
git config --global core.editor geany
```

Sie können aber auch einen beliebigen anderen Texteditor vorgeben. Dazu müssen Sie lediglich den Namen austauschen. Unabhängig davon, für welchen Texteditor Sie sich entscheiden, müssen Sie darauf achten, diesen zu den Pfadvariablen hinzuzufügen. Sonst findet das System ihn nicht und kann ihn nicht verwenden. Zu diesem Zweck müssen Sie die „Erweiterten Systemeinstellungen" aufrufen und anschließend „Umgebungsvariablen" anklicken. Wählen Sie im unteren Feld nun den Eintrag mit der Bezeichnung „Path" aus und klicken Sie auf „Bearbeiten". Nun müssen Sie

auf „Neu" klicken und hier das Verzeichnis vorgeben, in dem sich der entsprechende Texteditor befindet. Sollten dabei Probleme auftreten, können Sie aber auch einfach weiterhin den Standard-Texteditor verwenden und auf den entsprechenden Konfigurations-Eintrag verzichten.

Eine weitere Konfigurationsmöglichkeit besteht darin, den Namen des Hauptzweigs vorzugeben. Jeder Zweig, den Sie im Rahmen eines Projekts erzeugen, benötigt einen Namen. Dabei ist insbesondere der Hauptzweig von Bedeutung. Dieser erhält beim Erstellen eines neuen Projekts automatisch einen Namen. Gemäß den Standardeinstellungen lautet die Bezeichnung hierfür `master`. Allerdings können Sie hierfür auch andere Vorgaben machen. Um den Namen des Hauptzweigs zu ändern, kommt der folgende Befehl zum Einsatz:

```
git config --global init.defaultBranch hauptzweig
```

Um die ursprüngliche Benennung beizubehalten, können Sie nach diesem Test aber auch gerne den Namen wieder zu `master` zurücksetzen. Im weiteren Verlauf des Buchs werden wir mit der Bezeichnung `master` arbeiten, sodass es sinnvoll ist, diese auch auf Ihrem System vorzugeben.

Auch auf die optische Gestaltung können Sie Einfluss nehmen. Beispielsweise gestaltet Git normalerweise die Ausgaben des Programms farbig. Wenn Sie eine schlichte Darstellung ohne Farben vorziehen, erreichen Sie dies mit dem folgenden Befehl:

```
git config --global color.ui false
```

Hierbei handelt es sich lediglich um eine kleine Auswahl der Konfigurationsmöglichkeiten für Git. Für Anfänger ist es jedoch normalerweise nicht notwendig, hierbei weitere Veränderungen vorzunehmen. Lediglich die Einstellung der Identität und eventuell die des verwendeten Texteditors sind wichtig. Wenn Sie im Laufe der Arbeit mit Git jedoch noch weitere Details entdecken, die Sie konfigurieren möchten, ist dies jederzeit möglich. Im weiteren Verlauf des Buchs werden wir noch einige weitere Möglichkeiten vorstellen. Einen vollständigen Überblick über die Konfigurationsmöglichkeiten entdecken Sie unter dem folgenden Link:

https://git-scm.com/docs/git-config

### Die Einstellungen überprüfen

Wenn Sie die Konfiguration nach Ihren Wünschen verändert haben und später weitere Anpassungen vornehmen möchten, ist es in der Regel sinnvoll, zunächst die aktuellen Einstellungen zu überprüfen. So erkennen Sie, wie die Konfiguration im Moment aussieht, und können entscheiden, welche weiteren Änderungen Sie daran vornehmen möchten. Abbildung 2.6 zeigt, wie diese Angaben aussehen können. Der erforderliche Befehl sieht wie folgt aus:

```
git config --list
```

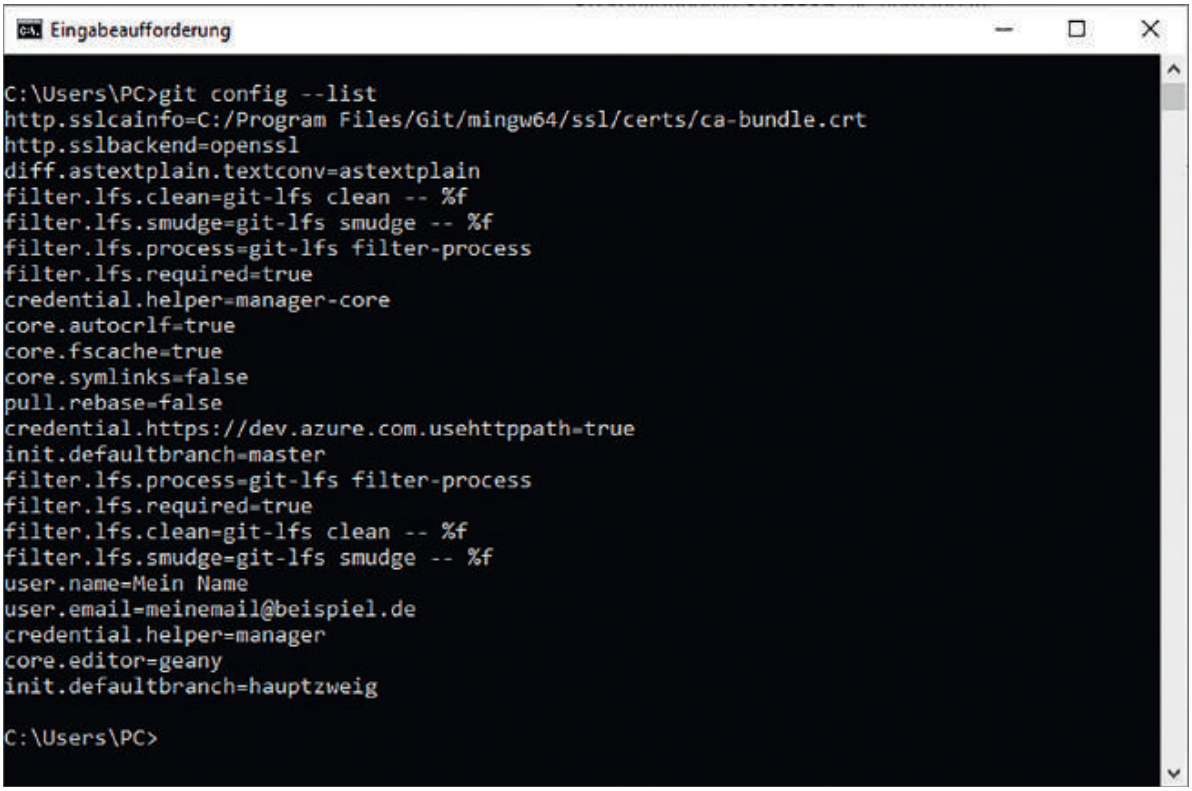


**Abb. 2.6** Die Auflistung der Einstellungen

## 2.4 Wenn es einmal nicht weitergeht: die Git-Hilfe

Bei der Arbeit mit Git treten gelegentlich Probleme auf, bei denen Sie als Anfänger nicht wissen, wie Sie damit umgehen können. Viele Anwendungsmöglichkeiten werden in diesem Buch vorgestellt. Doch sind die Funktionen von Git so umfangreich, dass es unmöglich ist, alle Details zu erklären. Aus diesem Grund ist es wichtig zu wissen, wo Sie Hilfe finden können, wenn es einmal nicht weitergeht.

## Die integrierte Git-Hilfe

Eine häufig genutzte Möglichkeit besteht darin, die integrierte Git-Hilfe zu verwenden. Diese enthält zu jedem Befehl eine Beschreibung. Das ist aus mehreren Gründen sehr praktisch. Zum einen ist diese Beschreibung sehr umfassend und enthält alle Funktionen und Optionen. Zum anderen ist sie ohne lange Suche verfügbar. Außerdem benötigen Sie hierfür keinen Internetanschluss. Zwar ist die Git-Hilfe in Form von HTML-Seiten gespeichert und wird daher im Browser geöffnet. Alle Hilfs-Dateien sind jedoch lokal auf Ihrem Rechner gespeichert, sodass hierfür keine Internetverbindung notwendig ist.

Allerdings gibt es hierbei auch eine Einschränkung: Um die Git-Hilfe zu verwenden, müssen Sie bereits wissen, welchen Befehl Sie nutzen wollen. Ist dieser bekannt, erhalten Sie alle notwendigen Details zur Anwendung. Wenn Sie jedoch erst auf der Suche nach einem passenden Befehl für die Lösung eines Problems sind, kommen Sie damit nicht ans Ziel. Das liegt daran, dass Sie für den Aufruf der Git-Hilfe stets angeben müssen, zu welchem Befehl Sie Informationen erhalten möchten.

Um dies an einem Beispiel zu verdeutlichen, verwenden wir den einzigen Git-Befehl, den wir in diesem Buch bereits verwendet haben: `config`. Wenn Sie Informationen zu einem anderen Befehl benötigen, müssen Sie im Folgenden einfach den Ausdruck `config` durch den entsprechenden Begriff ersetzen. Um die Hilfe dazu aufzurufen, bestehen mehrere Möglichkeiten:

```
git help config
```

- `git config --help`
- `man git-config`
- `git config -h`

Hierbei müssen Sie beachten, dass die dritte Option mit dem Befehl `man` unter Windows nicht funktioniert. Wenn Sie dieses Betriebssystem verwenden, sind Sie auf die übrigen drei Alternativen beschränkt.

Die ersten drei vorgestellten Alternativen ziehen genau das gleiche Ergebnis nach sich: Sie öffnen die umfangreiche Manpage-Hilfe im Webbrowser.

Dabei handelt es sich um sehr umfangreiche Dokumente mit zahlreichen Informationen. Der letzte Befehl führt hingegen zu einer stark verkürzten Darstellung. Diese wird außerdem direkt im Kommandozeileninterpreter geöffnet und nicht im Webbrowser. Diese Alternative ist in Abbildung 2.7 dargestellt und bietet sich insbesondere dann an, wenn Sie einen schnellen Überblick über die Anwendungsmöglichkeiten wünschen.

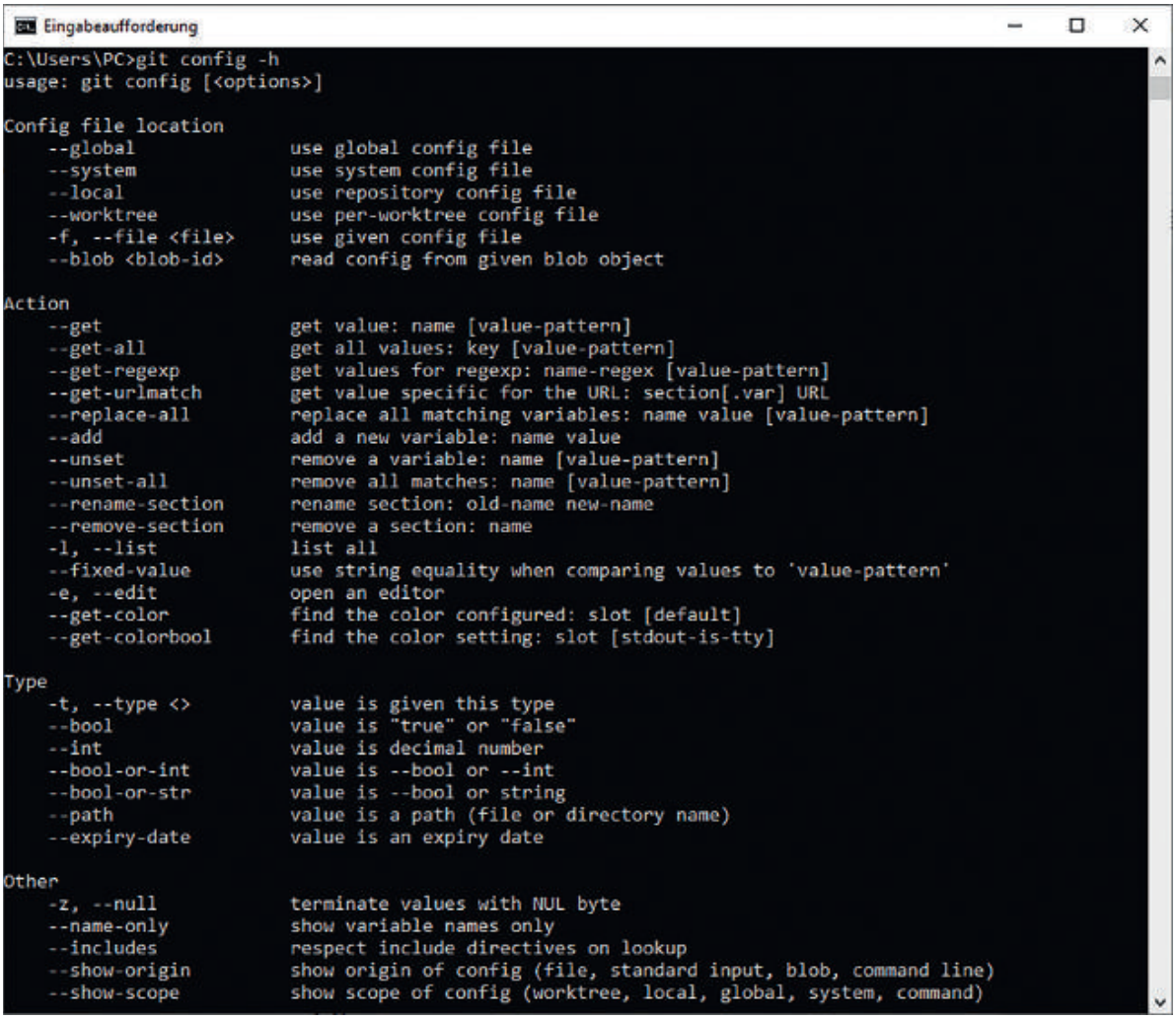


**Abb. 2.7** Die verkürzte Form der Git-Hilfe im Kommandozeileninterpreter

## Informationen aus dem Internet verwenden

Wenn die integrierte Hilfe nicht ausreicht, bietet es sich an, im Internet nach Informationen zu suchen. Das ist insbesondere dann sinnvoll, wenn Sie noch nicht wissen, welchen Befehl Sie verwenden möchten. Es gibt zahlreiche Seiten, die Erklärungen zu Git anbieten. An erster Stelle ist hierbei die offizielle Homepage der Software zu nennen: https://git-scm.com/.

Über eine Internetrecherche lassen sich jedoch auch viele weitere Seiten finden, die die entsprechenden Informationen bereithalten. Sollten Sie mit einer deutschsprachigen Suchanfrage nicht die gewünschten Ergebnisse finden, ist es empfehlenswert, einen englischen Begriff zu verwenden. Hierbei ist das Angebot deutlich umfangreicher, sodass es wesentlich wahrscheinlicher ist, dass Sie die gewünschten Informationen finden.

Schließlich gibt es auch jede Menge Blogs und Foren, die sich mit Git befassen. Die Community ist hier sehr aktiv. Das gibt Ihnen die Möglichkeit, auch selbst Fragen zu stellen. Wenn Sie bei einem konkreten Problem nicht weiterkommen, findet sich meistens ein erfahrener Anwender, der Ihnen weiterhilft.

## 2.5 Übungsaufgaben

Am Ende der meisten Kapitel entdecken Sie einige Übungsaufgaben. Diese dienen dazu, die vorgestellten Inhalte zu vertiefen und zu erweitern. Um die Aufgaben zu lösen, können Sie häufig die im entsprechenden Kapitel vorgestellten Befehle verwenden und sie auf andere Anwendungsfälle übertragen. Häufig handelt es sich jedoch auch um neue Aspekte. In diesen Fällen müssen Sie selbstständig nach einer Lösung suchen – beispielsweise auf den integrierten Hilfs-Seiten, auf der offiziellen Git-Homepage oder auf anderen Internetseiten.

Anschließend stellen wir stets eine Musterlösung vor. Beachten Sie hierbei, dass es häufig verschiedene Möglichkeiten gibt, um ans Ziel zu kommen. Wenn Ihre Lösung alle Anforderungen erfüllt, ist die Aufgabe auch dann korrekt bearbeitet, auch wenn der gewählte Lösungsweg von der dargestellten Musterlösung abweicht. Für dieses Kapitel sollen Sie die folgenden Aufgaben lösen:

1. Git bietet neben den vorgestellten Konfigurationsmöglichkeiten noch viele weitere Optionen. Suchen Sie selbstständig nach einer Möglichkeit, um mit dem `config`-Befehl die Autokorrektur zu aktivieren. Diese führt dazu, dass auch bei einem Tippfehler der korrekte Befehl ausgeführt wird.
2. Einer der wichtigsten Git-Befehle ist `init`. Diesen werden wir im nächsten Kapitel vorstellen. Rufen Sie jedoch bereits jetzt einmal die Hilfe zu

diesem Befehl auf und bringen Sie in Erfahrung, welche Aufgabe Sie damit erledigen können.

Lösungen:



1. `git config --global help.autocorrect 1`
2. Die Hilfe erreichen Sie über einen der folgenden Befehle: `git help init`, `git init --help`, `man git-init` (nicht unter Windows) oder `git init -h` (in verkürzter Form).

Der `init`-Befehl dient dazu, ein neues Repository anzulegen.

Alle Programmcodes aus diesem Buch sind
zum Download verfügbar. Dadurch müssen Sie sie nicht abtippen:
https://bmu-verlag.de/books/git/


Außerdem erhalten Sie die eBook-Ausgabe zum Buch
kostenlos auf unserer Website:

https://bmu-verlag.de/books/git/
**Downloadcode:** siehe Kapitel 14