

Auf einen Blick

Über den Autor	13
Einleitung	27
Teil I: Grundlagen	33
Kapitel 1: Was ist Git?	35
Kapitel 2: Der Einstieg auf Windows	41
Kapitel 3: Der Einstieg auf macOS und Linux	63
Kapitel 4: Hinzufügen, ändern, branchen und mergen	77
Kapitel 5: Weitere Werkzeuge	99
Teil II: Zusammenarbeit	127
Kapitel 6: Git-Dienste	129
Kapitel 7: Guidelines bei der Einführung von Git	157
Kapitel 8: Git-Workflows	167
Kapitel 9: Teamwork mit Git	183
Kapitel 10: Weniger Komplexität durch Feature-Flags	211
Kapitel 11: Kontinuierlich bauen und ausliefern	223
Kapitel 12: Open-Source-Projekte	255
Teil III: Vertiefung	269
Kapitel 13: Unter der Haube	271
Kapitel 14: Zeitreisen mit Git	287
Kapitel 15: Mehr zum Branching und Merging	313
Kapitel 16: Commits und Tags signieren	329
Kapitel 17: Git-Submodule	341
Kapitel 18: Große Dateien und große Repositories	349
Kapitel 19: Git an Ihre Bedürfnisse anpassen	355
Teil IV: Der Top-Ten-Teil	365
Kapitel 20: Zehn Tipps zum Einstieg in Git	367
Kapitel 21: Zehn Tipps zur Erhöhung der Effizienz	371
Kapitel 22: Zehn Dinge, die Sie nicht tun sollten	387
Kapitel 23: Zehn Git-Befehle, die Sie kennen sollten	391
Kapitel 24: Zehn Git-Befehle, die Ihnen bei Problemen helfen	405
Stichwortverzeichnis	413

Inhaltsverzeichnis

Über den Autor	13
Einleitung	27
Über dieses Buch	27
Konventionen in diesem Buch	27
Fachbegriffe	28
Warum Kommandozeile?	28
Der Name für den Haupt-Branch	28
Was Sie nicht lesen müssen	29
Törichte Annahmen über die Leser	29
Wie dieses Buch aufgebaut ist	30
Teil I: Grundlagen	30
Teil II: Zusammenarbeit	30
Teil III: Vertiefung	30
Teil IV: Der Top-Ten-Teil	30
Symbole, die in diesem Buch verwendet werden	30
Wie es weitergeht	31
TEIL I	
GRUNDLAGEN	33
Kapitel 1	
Was ist Git?	35
Versionsverwaltung – zentral oder verteilt?	36
Die Geschichte von Git	37
Was bedeutet der Name »Git«?	38
Lizenz und Betriebssysteme	39
Ausblick	39
Kurz und knackig	39
Kapitel 2	
Der Einstieg auf Windows	41
Die Installation von Git auf Windows	41
Die richtige Kommandozeile	51
Terminal: Cmder	52
Cmd vs. PowerShell	52
Windows-Subsystem für Linux (WSL)	54
Das Windows-Terminal	55
Der richtige Editor	56
Die Konfiguration von Git	58
Authentifizierung	59
Kurz und knackig	62

Kapitel 3	
Der Einstieg auf macOS und Linux	63
Der Einstieg auf macOS	63
Der Einstieg auf Linux	64
Konfiguration	64
Authentifizierung	66
Arbeiten mit dem Terminal	69
Richtig mit dem Terminal umgehen	70
Die richtige Konsole finden	71
Aufgehübscht: Oh-My-Zsh und Powerlevel10k	71
Kurz und knackig	76
Kapitel 4	
Hinzufügen, ändern, branchen und mergen	77
Das lokale Repository	78
Das Remote-Repository	79
Änderungen dem Repository hinzufügen	80
Die lokale Entwicklungsumgebung aktualisieren	81
Schritt für Schritt: Änderungen der Versionsverwaltung hinzufügen	81
Neue Dateien hinzufügen	82
Änderungen durchführen	85
Arbeiten mit Verzweigungen (Branches)	87
Einen Branch erstellen	88
Auf einen anderen Branch wechseln	89
Arbeiten mit Upstream-Banches	90
Änderungen zusammenführen	92
Der Fast-Forward-Merge	92
Der Merge-Commit	94
Konflikte lösen	95
Schritt für Schritt: Branchen und Mergen	96
Kurz und knackig	97
Kapitel 5	
Weitere Werkzeuge	99
Visual Studio Code	99
Unterstützung im Working Directory	99
Arbeiten mit Branches und Tags	100
Unterstützung für Remote-Repositories	100
Statusbar und Editor	100
Visual-Studio-Code-Erweiterungen	101
Benutzeroberflächen für Git	106
Sourcetree	106
GitKraken	108
GitHub Desktop	109
Git-GUI	111
GitHub für unterwegs	115

Visual Studio, Eclipse, IntelliJ und Co.	117
Tools mit Windows-Explorer-Integration	118
TortoiseGit.	118
Git-Extensions.	121
Diff- und Merge-Tools.	122
Kurz und knackig	125

TEIL II
ZUSAMMENARBEIT. 127

Kapitel 6
Git-Dienste 129

GitHub	130
Einstieg.	130
GitHub-Organisationen	134
Ein Repository anlegen	136
Preise	140
Hosting.	140
Bewertung.	140
Azure Repos	141
Einstieg.	141
Azure-DevOps-Organisationen	142
Ein Projekt erstellen.	143
Ein Repository erstellen.	144
Preise	146
Hosting.	146
Bewertung.	146
GitLab	147
Einstieg.	147
Hosting und Preise.	150
Bewertung.	151
Bitbucket.	151
Einstieg.	151
Hosting und Preise.	155
Bewertung.	155
Kurz und knackig	156

Kapitel 7
Guidelines bei der Einführung von Git 157

Das richtige Maß an Governance	157
Eine minimale Governance-Richtlinie.	158
Die Wahl des passenden Git-Systems.	158
Der minimale Git-Workflow	159
Namenskonventionen.	159
Minimale Review-Guidelines	159
Weitere Ergänzungen für Pull-Requests	160

20 Inhaltsverzeichnis

Empfehlungen für Teams	160
Anzahl und Struktur der Repositories	161
Review-Guidelines	162
Release-Branching	163
Umgang mit komplexen Features	163
Commit-Messages und Pull-Requests	164
Merge-Strategien	164
Training	165
Kurz und knackig	165

Kapitel 8 Git-Workflows 167

Was sind Git-Workflows?	167
Trunk-Based-Development	168
GitHub-Flow	170
Release-Flow	173
Git-Flow	175
Die Haupt-Branches in Git-Flow	175
Feature-Branches im Git-Flow	176
Release-Branches	177
Hotfix-Branches	178
Zusammenfassung	179
Den richtigen Workflow finden	180
Kurz und knackig	181

Kapitel 9 Teamwork mit Git 183

Watch, Stars und Forks	183
Was ist ein Fork?	185
Code-Reviews mit Pull-Requests	192
Branch-Protection	199
Automatisierung	202
Code-Owners	204
Der Umgang mit Commits und Messages	205
Der Pull-Request-Lebenszyklus	208
Kurz und knackig	209

Kapitel 10 Weniger Komplexität durch Feature-Flags 211

Was sind Feature-Flags?	211
Der Feature-Lebenszyklus	213
Weitere Einsatzmöglichkeiten von Feature-Flags	215
Wo fängt man an?	216
Frameworks	217
LaunchDarkly	218
Feature-Flags und technische Schulden	219
Kurz und knackig	221

Kapitel 11	
Kontinuierlich bauen und ausliefern	223
Was ist CI und CD?	223
Build-Infrastruktur	225
Kontinuierliche Qualität	225
Tests und Code-Coverage	226
Code-Analyse und Quality-Gates	228
Automatisierte Deployments	229
Infrastruktur als Code	229
App-Stores, Registries und Paketmanager	230
Die Bedeutung von Containern	232
Azure Pipelines	234
Ihre erste Pipeline	234
Technische Schulden meistern	242
GitHub Actions	249
Ihr erster Workflow	249
Ihre erste GitHub-Action	251
Weitere CI/CD-Plattformen	254
Kurz und knackig	254

Kapitel 12	
Open-Source-Projekte	255
Geschichte	256
Open Source versus Open Development	256
Open Source und Sicherheit	257
Ihr erstes Open-Source-Projekt	261
Wann ist der richtige Zeitpunkt?	262
Ist es die richtige Lösung?	262
Name und Branding	262
Open-Source-Checkliste	263
Standardisierung und Automatisierung	264
Eine Community aufbauen	264
Geld verdienen mit Open Source	265
Kurz und knackig	266

TEIL III	
VERTIEFUNG	269

Kapitel 13	
Unter der Haube	271
Warum ist Git so schwierig?	271
Der gerichtete azyklische Graph	272
Der Hash oder SHA-1	273
Die Anatomie eines Commits	274
Was ist denn nun eigentlich Git?	284
Warum ist Git so einfach?	285
Kurz und knackig	285

Kapitel 14

Zeitreisen mit Git	287
Geschichte ist Geschichte.	287
Freie Wahl der Arbeitsweise.	289
Änderungen rückgängig machen	289
Änderungen mit Revert transparent zurücknehmen	289
Commits nachbessern mit Amend	291
Zeitreisen mit Reset.	291
Navigieren in der Zeit	296
Zeitlinien manipulieren.	300
Zurück in die Zukunft mit Rebase	300
Änderungen bündeln mit Squash	302
Rosinen herauspicken.	304
Die Vergangenheit interaktiv manipulieren.	304
Die Vergangenheit mit Filter-Branch manipulieren	308
Manipulierte Zeitlinien mit Force-Push übertragen	310
Fragen Sie nicht, wozu Git fähig ist	310
Kurz und knackig	311

Kapitel 15

Mehr zum Branching und Merging	313
Branches und Tags	313
Vorspulen mit Fast-Forward.	316
Ein echter Merge	317
Ein Octopus-Merge	317
Konflikte automatisch lösen.	319
Konflikte manuell lösen	320
Merge-Strategien	323
Recursive	323
Resolve.	324
Octopus	325
Ours	325
Subtree	325
Daten vom Remote laden mit »git pull«.	325
Kurz und knackig	326

Kapitel 16

Commits und Tags signieren	329
Warum sollte man Commits signieren?	330
Was ist GPG?	333
Installation und Konfiguration.	333
Installation auf dem Mac.	333
Installation auf Windows.	334

Installation auf Linux	334
Ein Schlüsselpaar generieren	334
Weitere E-Mail-Adressen konfigurieren	337
Die Konfiguration von Git	337
Konfiguration von GitHub	338
Visual Studio Code konfigurieren	338
Der erste signierte Commit	339
Kurz und knackig	340

Kapitel 17	
Git-Submodule	341
Submodule in Git-Repository einbinden	341
Submodule klonen	343
Submodule aktualisieren	344
In Submodulen arbeiten	345
Kurz und knackig	347

Kapitel 18	
Große Dateien und große Repositories	349
Git-LFS	349
Wann sollen Sie Git-LFS verwenden?	349
Installation von Git-LFS	350
Git-LFS verwenden	350
Große Dateien sperren	351
Nachteile	351
Große Repositories	352
VFS for Git	352
Scalar	352
Kurz und knackig	354

Kapitel 19	
Git an Ihre Bedürfnisse anpassen	355
Dateien ignorieren	355
Arbeiten mit unterschiedlichen Dateitypen	357
Binärdateien	357
Bestimmte Dateitypen versionieren	357
Inhalte vor dem Ein- oder Auschecken filtern oder bearbeiten	358
Formatierung von Dateien	361
Der richtige Umgang mit Zeilenenden	361
Encoding	362
Leerzeichen, Tabs und Spaces	362
Kurz und knackig	364

**TEIL IV
DER TOP-TEN-TEIL..... 365**

**Kapitel 20
Zehn Tipps zum Einstieg in Git..... 367**

- Lernen und verwenden Sie Git auf der Konsole 367
- Der richtige Git-Prompt 367
- Richten Sie sich eine Wohlfühlkonsole ein. 368
- Lernen Sie die Shortcuts für Ihr Terminal 368
- Lernen Sie einen Editor. 368
- Lernen Sie die Shortcuts für den Editor 368
- Üben Sie in der Git-School 368
- Erstellen Sie ein Konto auf GitHub 368
- Verwenden Sie einen schlanken Workflow 369
- Arbeiten Sie mit Pull-Requests. 369
- Kurz und knackig 369

**Kapitel 21
Zehn Tipps zur Erhöhung der Effizienz 371**

- Arbeiten mit Aliasen in Git 371
- Besser Vergleichen mit »diff-so-fancy«. 374
- Automatisieren mit Git-Hooks 376
 - Clientseitige Hooks 376
 - Serverseitige Hooks 377
- Bessere Commit-Messages mit Commit-Templates 378
- Einzelne Commits testen 379
- Besser Versionieren mit Git-Version. 379
- Merges aufzeichnen mit rerere 381
- Autokorrektur von Kommandos 383
- Arbeit zwischenspeichern mit Stash. 384
- Kurz und knackig 385

**Kapitel 22
Zehn Dinge, die Sie nicht tun sollten 387**

- Direkt auf dem Haupt-Branch arbeiten 387
- Mit langlebigen Branches arbeiten. 387
- Passwörter und andere Secrets in Git speichern 388
- Nicht oft genug committen 388
- Git nicht als Versionsverwaltung nutzen 388
- Große Dateien speichern 388
- Einen Pull-Request für viele Änderungen verwenden 389
- Einen Reset durchführen, ohne vorher zu speichern. 389
- Die veröffentlichte Historie modifizieren. 389
- Einen Force-Push durchführen 389
- Kurz und knackig 390

Kapitel 23**Zehn Git-Befehle, die Sie kennen sollten 391**

Bugs finden mit git bisect	391
Dateien umbenennen mit git mv und git rm	393
In Git suchen mit git grep und git log	394
Den Schuldigen finden mit git blame	397
Aufräumen mit git clean	398
Einen Branch wechseln mit git switch	398
Arbeiten mit Patches: git add --patch und git apply	399
Kurz und knackig	403

Kapitel 24**Zehn Git-Befehle, die Ihnen bei Problemen helfen 405**

Magische Zeitmaschine	405
Vergessen, Änderungen hinzuzufügen	406
Commit-Message nachträglich ändern	406
Aus Versehen auf den Haupt-Branch committed	406
Commit auf falschen Branch	407
Diff zeigt nichts an	407
Älteren Commit rückgängig machen	407
Änderungen an einer Datei rückgängig machen	408
Von Neuem beginnen (a)	408
Von Neuem beginnen (b)	409
Kurz und knackig	409

Stichwortverzeichnis 413