

Inhaltsverzeichnis

Erste Schritte

1	Grundlegende Konzepte	1
1.1	Dezentrale Versionsverwaltung – alles anders?	1
1.2	Das Repository – die Grundlage dezentralen Arbeitens	4
1.3	Branching und Merging – ganz einfach!	6
1.4	Zusammenfassung	8
2	Erste Schritte mit der Kommandozeile	9
2.1	Git einrichten	9
2.2	Ein paar Hinweise für Windows-User	9
2.3	Git einrichten	11
2.4	Das erste Projekt mit Git	12
2.5	Zusammenarbeit mit Git	15
2.6	Zusammenfassung	21
3	Erste Schritte mit SourceTree	23
3.1	SourceTree konfigurieren	23
3.2	Das erste Projekt mit Git	24
3.3	Zusammenarbeit mit Git	26
3.4	Zusammenfassung	30

Arbeiten mit Git

4	Was sind Commits?	31
4.1	Zugriffsberechtigungen und Zeitstempel	32
4.2	Die Befehle <code>add</code> und <code>commit</code>	32
4.3	Exkurs: Mehr über Commit-Hashes	33
4.4	Eine Historie von Commits	34
4.5	Das HEAD-Commit	35
4.6	Eine etwas andere Sichtweise auf Commits	35
4.7	Commits untersuchen	36
4.8	Viele unterschiedliche Historien desselben Projekts	37

4.9	Schreibweisen für Commits	40
4.10	Zusammenfassung	41
5	Commits zusammenstellen	43
5.1	Der status-Befehl	43
5.2	Der Stage-Bereich umfasst alle Projektdateien	47
5.3	Was tun mit Änderungen, die nicht übernommen werden sollen?	49
5.4	Mit <code>.gitignore</code> Dateien unversioniert lassen	50
5.5	Stashing: Änderungen zwischenspeichern	51
5.6	Zusammenfassung	52
6	Das Repository	53
6.1	Ein einfaches und effizientes Speichersystem	53
6.2	Verzeichnisse speichern: Blob und Tree	54
6.3	Gleiche Daten werden nur einmal gespeichert	55
6.4	Kompression ähnlicher Inhalte	55
6.5	Ist es schlimm, wenn verschiedene Daten zufällig denselben Hashwert bekommen?	56
6.6	Commits	56
6.7	Wiederverwendung von Objekten in der Commit-Historie ...	57
6.8	Umbenennen, verschieben und kopieren	58
6.9	Zusammenfassung	61
7	Branches verzweigen	63
7.1	Parallele Entwicklung	63
7.2	Bugfixes in älteren Versionen	64
7.3	Branches	64
7.4	Aktiver Branch	65
7.5	Der <i>Master-Branch</i>	68
7.6	Branch-Zeiger umsetzen	68
7.7	Branch löschen	69
7.8	Und was ist, wenn man die Commit-Objekte wirklich loswerden will?	70
7.9	Zusammenfassung	71
8	Branches zusammenführen	73
8.1	Was passiert bei einem Merge?	74
8.2	Konflikte	75
8.3	Fast-Forward-Merges	80
8.4	First-Parent-History	81
8.5	Knifflige Merge-Konflikte	82
8.6	Zusammenfassung	84

9	Mit Rebasing die Historie glätten	87
9.1	Das Prinzip: Kopieren von Commits	87
9.2	Und wenn es zu Konflikten kommt?	89
9.3	Was passiert mit den ursprünglichen Commits nach dem Rebasing?	90
9.4	Empfehlungen zum Rebasing	91
9.5	Cherry-Picking	94
9.6	Zusammenfassung	94
10	Repositorys erstellen, klonen und verwalten	95
10.1	Ein Repository erstellen	95
10.2	Das Repository-Layout	95
10.3	Bare-Repositorys	96
10.4	Vorhandene Dateien übernehmen	96
10.5	Ein Repository klonen	97
10.6	Wie sagt man Git, wo das Remote-Repository liegt?	97
10.7	Kurznamen für Repositorys: Remotes	98
10.8	Das <i>Origin-Repository</i>	99
10.9	Zusammenfassung	99
11	Austausch zwischen Repositorys	101
11.1	Fetch, Pull und Push	101
11.2	Remote-Tracking-Branches	102
11.3	Einen Remote-Branch bearbeiten	103
11.4	Ein paar Begriffe, die man kennen sollte	104
11.5	Fetch: Branches aus einem anderen Repository holen	105
11.6	Fetch: Aufrufvarianten	105
11.7	Push mit <code>--force</code>	110
11.8	Erweiterte Möglichkeiten	110
11.9	Zusammenfassung	111
12	Versionen markieren	113
12.1	Arbeiten mit Tags erstellen	113
12.2	Welche Tags gibt es?	114
12.3	Die Hashes zu den Tags ausgeben	114
12.4	Die Log-Ausgaben um Tags anreichern	115
12.5	In welcher Version ist es »drin«?	115
12.6	Wie verschiebt man ein Tag?	115
12.7	Und wenn ich ein »Floating Tag« brauche?	116
12.8	Zusammenfassung	116

13	Tipps und Tricks	117
13.1	Keine Panik – es gibt ein Reflog!	117
13.2	Lokale Änderungen temporär ignorieren	118
13.3	Änderungen an Textdateien untersuchen	119
13.4	alias – Abkürzungen für Git-Befehle	120
13.5	Branches als temporäre Zeiger auf Commits nutzen	121
13.6	Commits auf einen anderen Branch verschieben	122
13.7	Mehr Kontrolle bei Fetch, Push und Pull	123
13.8	Git-Version auf Ubuntu Linux aktualisieren	125

Workflows

14	Workflow-Einführung	127
14.1	Warum Workflows?	127
14.2	Welche Workflows sind wann sinnvoll?	128
14.3	Aufbau der Workflows	130

Workflows: Entwickeln mit Git

15	Ein Projekt aufsetzen	133
16	Gemeinsam auf einem Branch entwickeln	145
17	Mit Feature-Banches entwickeln	153
18	Mit Forks entwickeln	173

Workflows: Release-Prozess

19	Kontinuierlich Releases durchführen	185
20	Periodisch Releases durchführen	195
21	Mit mehreren aktiven Releases arbeiten	209

Workflows: Repositorys pflegen

22	Ein Projekt mit großen binären Dateien versionieren	223
23	Große Projekte aufteilen	231

24	Kleine Projekte zusammenführen	239
25	Lange Historien auslagern	245
26	http://kapitel26.github.io	255
27	Ein Projekt nach Git migrieren	257

Mehr über Git

28	Integration mit Jenkins	273
28.1	Vorbereitungen	273
28.2	Ein einfaches Git-Projekt einrichten	274
28.3	Hook als Build-Auslöser	275
28.4	Ein Tag für jeden erfolgreichen Build	277
28.5	Pull-Requests bauen	279
28.6	Automatischer Merge von Branches	280
28.7	Mit Jenkins Pipelines arbeiten	281
29	Große Repositorys	285
29.1	Repositorys mit sehr vielen Dateien	285
29.2	Sparse Checkout	286
29.3	Mit Watchman Dateiänderungen schneller erkennen	289
29.4	Repositorys mit großem Speicherbedarf	289
29.5	Shallow Clone	290
29.6	Zusammenfassung	291
30	Abhängigkeiten zwischen Repositorys	293
30.1	Abhängigkeiten mit Submodulen	293
30.2	Abhängigkeiten mit Subtrees	299
30.3	Zusammenfassung	306
31	Was gibt es sonst noch?	307
31.1	Worktrees – mehrere Workspaces mit einem Repository	307
31.2	Interaktives Rebasing – Historie verschönern	308
31.3	Umgang mit Patches	309
31.4	Archive erstellen	309
31.5	Grafische Werkzeuge für Git	310
31.6	Repository im Webbrowser anschauen	311
31.7	Zusammenarbeit mit Subversion	312
31.8	Hooks – Git erweitern	312
31.9	Mit Bisection Fehler suchen	312

32	Die Grenzen von Git	315
32.1	Hohe Komplexität	315
32.2	Komplizierter Umgang mit Submodulen	317
32.3	Ressourcenverbrauch bei großen binären Dateien	318
32.4	Repositorys können nur vollständig verwendet werden	318
32.5	Autorisierung nur auf dem ganzen Repository	319
32.6	Mäßige grafische Werkzeuge für die Historienauswertung ...	320

Anhang

Schritt-für-Schritt-Anleitungen	323
Workflow-Verzeichnis	325
Index	331