

# Inhaltsverzeichnis

## Erste Schritte

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Grundlegende Konzepte</b> .....                         | <b>1</b>  |
| 1.1      | Dezentrale Versionsverwaltung – alles anders? .....        | 1         |
| 1.2      | Das Repository – die Grundlage dezentralen Arbeitens ..... | 3         |
| 1.3      | Branching und Merging – ganz einfach! .....                | 5         |
| 1.4      | Zusammenfassung .....                                      | 7         |
| <b>2</b> | <b>Erste Schritte mit der Kommandozeile</b> .....          | <b>9</b>  |
| 2.1      | Git einrichten .....                                       | 9         |
| 2.2      | Das erste Projekt mit Git .....                            | 9         |
| 2.3      | Zusammenarbeit mit Git .....                               | 13        |
| 2.4      | Zusammenfassung .....                                      | 18        |
| <b>3</b> | <b>Erste Schritte mit SourceTree</b> .....                 | <b>21</b> |
| 3.1      | SourceTree konfigurieren .....                             | 21        |
| 3.2      | Das erste Projekt mit Git .....                            | 21        |
| 3.3      | Zusammenarbeit mit Git .....                               | 24        |
| 3.4      | Zusammenfassung .....                                      | 28        |

## Arbeiten mit Git

|          |   |           |
|----------|---|-----------|
| <b>4</b> | <b>Was sind Commits?</b> .....                            | <b>29</b> |
| 4.1      | Zugriffsberechtigungen und Zeitstempel .....              | 30        |
| 4.2      | Die Befehle add und commit .....                          | 30        |
| 4.3      | Exkurs: Mehr über Commit-Hashes .....                     | 31        |
| 4.4      | Eine Historie von Commits .....                           | 32        |
| 4.5      | Eine etwas andere Sichtweise auf Commits .....            | 32        |
| 4.6      | Viele unterschiedliche Historien desselben Projekts ..... | 33        |
| 4.7      | Zusammenfassung .....                                     | 36        |

|          |  |           |
|----------|--|-----------|
| <b>5</b> | <b>Commits zusammenstellen</b> .....   | <b>37</b> |
| 5.1      | Der status-Befehl .....  | 37        |
| 5.2      | Der Stage-Bereich speichert Momentaufnahmen .....                                      | 41        |
| 5.3      | Was tun mit Änderungen, die nicht übernommen werden<br>sollen? .....                   | 42        |
| 5.4      | Mit <code>.gitignore</code> Dateien unversioniert lassen .....                         | 43        |
| 5.5      | Stashing: Änderungen zwischenspeichern .....   | 44        |
| 5.6      | Zusammenfassung .....  | 45        |
| <b>6</b> | <b>Das Repository</b> .....  | <b>47</b> |
| 6.1      | Ein einfaches und effizientes Speichersystem .....                                     | 47        |
| 6.2      | Verzeichnisse speichern: Blob und Tree .....   | 48        |
| 6.3      | Gleiche Daten werden nur einmal gespeichert .....                                      | 49        |
| 6.4      | Kompression ähnlicher Inhalte .....  | 49        |
| 6.5      | Ist es schlimm, wenn verschiedene Daten zufällig denselben<br>Hashwert bekommen? ..... | 50        |
| 6.6      | Commits .....  | 50        |
| 6.7      | Wiederverwendung von Objekten in der Commit-Historie ...                               | 51        |
| 6.8      | Umbenennen, verschieben und kopieren .....   | 52        |
| 6.9      | Zusammenfassung .....  | 54        |
| <b>7</b> | <b>Branches verzweigen</b> .....   | <b>57</b> |
| 7.1      | Parallele Entwicklung .....  | 57        |
| 7.2      | Bugfixes in älteren Versionen .....  | 58        |
| 7.3      | Branches .....   | 58        |
| 7.4      | Aktiver Branch .....   | 59        |
| 7.5      | Branch-Zeiger umsetzen .....   | 62        |
| 7.6      | Branch löschen .....   | 62        |
| 7.7      | Und was ist, wenn man die Commit-Objekte wirklich<br>loswerden will? .....             | 63        |
| 7.8      | Zusammenfassung .....  | 64        |
| <b>8</b> | <b>Branches zusammenführen</b> .....   | <b>65</b> |
| 8.1      | Was passiert bei einem Merge? .....  | 66        |
| 8.2      | Konflikte .....  | 67        |
| 8.3      | Fast-Forward-Merges .....  | 72        |
| 8.4      | First-Parent-History .....   | 73        |
| 8.5      | Knifflige Merge-Konflikte .....  | 74        |
| 8.6      | Zusammenfassung .....  | 76        |
| <b>9</b> | <b>Mit Rebasing die Historie glätten</b> .....   | <b>79</b> |
| 9.1      | Das Prinzip: Kopieren von Commits .....  | 79        |
| 9.2      | Und wenn es zu Konflikten kommt? .....   | 81        |
| 9.3      | Branches umpflanzen .....  | 82        |

---

|           |  |            |
|-----------|--|------------|
| 9.4       | Was passiert mit den ursprünglichen Commits nach dem Rebasing? ..... | 84         |
| 9.5       | Cherry-Picking .....   | 85         |
| 9.6       | Zusammenfassung .....  | 85         |
| <b>10</b> | <b>Repositories erstellen, klonen und verwalten .....</b>            | <b>87</b>  |
| 10.1      | Ein Repository erstellen .....                                       | 87         |
| 10.2      | Das Repository-Layout .....  | 87         |
| 10.3      | Bare-Repositories .....  | 88         |
| 10.4      | Vorhandene Dateien übernehmen .....                                  | 88         |
| 10.5      | Ein Repository klonen .....  | 89         |
| 10.6      | Wie sagt man Git, wo das Remote-Repository liegt? .....              | 89         |
| 10.7      | Kurznamen für Repositories: Remotes .....                            | 90         |
| 10.8      | Zusammenfassung .....  | 91         |
| <b>11</b> | <b>Austausch zwischen Repositories .....</b>                         | <b>93</b>  |
| 11.1      | Fetch, Pull und Push .....   | 93         |
| 11.2      | Remote-Tracking-Branches .....                                       | 94         |
| 11.3      | Einen Remote-Branch bearbeiten .....                                 | 95         |
| 11.4      | Ein paar Begriffe, die man kennen sollte .....                       | 96         |
| 11.5      | Fetch: Branches aus einem anderen Repository holen .....             | 97         |
| 11.6      | Fetch: Aufrufvarianten .....   | 97         |
| 11.7      | Erweiterte Möglichkeiten .....                                       | 101        |
| 11.8      | Zusammenfassung .....  | 101        |
| <b>12</b> | <b>Versionen markieren .....</b>                                     | <b>103</b> |
| 12.1      | Arbeiten mit Tags erstellen .....                                    | 103        |
| 12.2      | Welche Tags gibt es? .....   | 104        |
| 12.3      | Die Hashes zu den Tags ausgeben .....                                | 104        |
| 12.4      | Die Log-Ausgaben um Tags anreichern .....                            | 105        |
| 12.5      | In welcher Version ist es »drin«? .....                              | 105        |
| 12.6      | Wie verschiebt man ein Tag? .....                                    | 105        |
| 12.7      | Und wenn ich ein »Floating Tag« brauche? .....                       | 106        |
| 12.8      | Zusammenfassung .....  | 106        |
| <b>13</b> | <b>Abhängigkeiten zwischen Repositories .....</b>                    | <b>107</b> |
| 13.1      | Abhängigkeiten mit Submodulen .....                                  | 107        |
| 13.2      | Abhängigkeiten mit Subtrees .....                                    | 113        |
| 13.3      | Zusammenfassung .....  | 118        |
| <b>14</b> | <b>Tipps und Tricks .....</b>  | <b>121</b> |
| 14.1      | Keine Panik – Es gibt ein Reflog! .....                              | 121        |
| 14.2      | Lokale Änderungen temporär ignorieren .....                          | 122        |
| 14.3      | Änderungen an Textdateien untersuchen .....                          | 123        |

|      |  |     |
|------|--|-----|
| 14.4 | alias – Abkürzungen für Git-Befehle .....              | 124 |
| 14.5 | Branches als temporäre Zeiger auf Commits nutzen ..... | 124 |
| 14.6 | Commits auf einen anderen Branch verschieben .....     | 125 |
| 14.7 | Mehr Kontrolle bei Fetch, Push und Pull .....          | 127 |

## Workflows

|      |  |            |
|------|--|------------|
| 15   | <b>Workflow-Einführung .....</b>           | <b>129</b> |
| 15.1 | Warum Workflows? .....                     | 129        |
| 15.2 | Welche Workflows sind wann sinnvoll? ..... | 130        |
| 15.3 | Aufbau der Workflows .....                 | 131        |

## Workflows: Entwickeln mit Git

|    |   |     |
|----|---|-----|
| 16 | Ein Projekt aufsetzen .....                 | 135 |
| 17 | Gemeinsam auf einem Branch entwickeln ..... | 153 |
| 18 | Mit Feature-Banches entwickeln .....        | 161 |
| 19 | Mit Bisection Fehler suchen .....           | 181 |

## Workflows: Releaseprozess

|    |  |     |
|----|--|-----|
| 20 | Kontinuierlich Releases durchführen .....    | 193 |
| 21 | Periodisch Releases durchführen .....        | 203 |
| 22 | Mit mehreren aktiven Releases arbeiten ..... | 217 |

## Workflows: Repositorys pflegen

|    |                                      |     |
|----|--------------------------------------|-----|
| 23 | Große Projekte aufteilen .....       | 231 |
| 24 | Kleine Projekte zusammenführen ..... | 239 |
| 25 | Lange Historien auslagern .....      | 245 |

## Workflows: Umstieg auf Git

|    |   |     |
|----|---|-----|
| 26 | Das sechszwanzigste Kapitel .....                 | 255 |
| 27 | Andere Versionsverwaltungen parallel nutzen ..... | 257 |
| 28 | Ein Projekt nach Git migrieren .....              | 269 |

## Mehr über Git

|       |  |     |
|-------|--|-----|
| 29    | Integration mit Jenkins .....                              | 285 |
| 29.1  | Vorbereitungen .....                                       | 285 |
| 29.2  | Ein einfaches Git-Projekt einrichten .....                 | 286 |
| 29.3  | Hook als Build-Auslöser .....                              | 287 |
| 29.4  | Ein Tag für jeden erfolgreichen Build .....                | 289 |
| 29.5  | Pull-Requests bauen .....                                  | 291 |
| 29.6  | Automatischer Merge von Branches .....                     | 293 |
| 30    | Was gibt es sonst noch? .....                              | 295 |
| 30.1  | Interaktives Rebasing – Historie verschönern .....         | 295 |
| 30.2  | Umgang mit Patches .....                                   | 296 |
| 30.3  | Patches per Mail versenden .....                           | 297 |
| 30.4  | Bundles – Pull im Offline-Modus .....                      | 297 |
| 30.5  | Archive erstellen .....                                    | 298 |
| 30.6  | Grafische Werkzeuge für Git .....                          | 298 |
| 30.7  | Repository im Webbrowser anschauen .....                   | 299 |
| 30.8  | Zusammenarbeit mit Subversion .....                        | 300 |
| 30.9  | Notizen an Commits .....                                   | 300 |
| 30.10 | Hooks – Git erweitern .....                                | 301 |
| 30.11 | Github – Hosting von Repositories .....                    | 301 |
| 31    | Die Grenzen von Git .....                                  | 303 |
| 31.1  | Hohe Komplexität .....                                     | 303 |
| 31.2  | Komplizierter Umgang mit Submodulen .....                  | 305 |
| 31.3  | Ressourcenverbrauch bei großen binären Dateien .....       | 306 |
| 31.4  | Repositories können nur vollständig verwendet werden ..... | 306 |
| 31.5  | Autorisierung nur auf dem ganzen Repository .....          | 307 |
| 31.6  | Mäßige grafische Werkzeuge für die Historienauswertung ... | 308 |

**Anhang**

|   |     |
|---|-----|
| »Schritt für Schritt«-Anleitungen ..... | 313 |
| Workflow-Verzeichnis .....              | 315 |
| Index .....                             | 321 |