# Dateien umbenennen und verschieben

# Dateien umbenennen und verschieben

Anders als viele Entwickler zunächst erwarten, speichert Git *keine* expliziten Umbenennungsinformationen. Es gibt in der internen Datenstruktur kein Attribut „diese Datei wurde umbenannt“. Git arbeitet stattdessen mit Snapshots ganzer Verzeichnisbäume und **erkennt** Umbenennungen nachträglich, indem es Dateiinhalte zwischen zwei Commits auf Ähnlichkeit vergleicht. Dieses Verständnis ist wichtig, um das Verhalten von `git status`, `git diff` und `git log` bei verschobenen oder umbenannten Dateien richtig einzuordnen.

## Der Komfortbefehl `git mv`

Für den Alltag stellt Git den Befehl `git mv` bereit, der zwei Schritte in einem einzigen Kommando zusammenfasst: das physische Umbenennen im Dateisystem und das Vormerken der Änderung im Index.

```
git mv alt.txt neu.txt
```

Dieser Aufruf entspricht exakt der folgenden Sequenz:

```
mv alt.txt neu.txt
git add alt.txt neu.txt
```

`git mv` ist also reine Bequemlichkeit – kein eigenständiger Git-Mechanismus. Ein Blick auf `git status` nach einem `git mv` zeigt den erwarteten Zustand:

```
renamed:    alt.txt -> neu.txt
```

# Manuelles Umbenennen

Wer eine Datei über den Windows-Explorer, PhpStorm oder ein beliebiges anderes Werkzeug umbenennt, erzielt am Ende dasselbe Ergebnis. Git registriert zunächst zwei getrennte Änderungen:

```
deleted:    alt.txt
new file:   neu.txt
```

Erst beim Erstellen des Commits – beziehungsweise beim Anzeigen von `git status` mit Ähnlichkeitserkennung – interpretiert Git diese Kombination als Umbenennung. Sofern der Inhalt der Datei weitgehend unverändert bleibt, ist es **funktional gleichwertig**, ob `git mv` verwendet wird oder Löschen und Neuanlegen manuell erfolgen. Für die Übersichtlichkeit im Arbeitsalltag ist `git mv` jedoch vorzuziehen, da es Missverständnisse vermeidet und die Absicht sofort klar kommuniziert.

# Wie Git Umbenennungen erkennt

Die Erkennung basiert auf einem Ähnlichkeitsvergleich, keiner festen Zuordnung. Git betrachtet eine gelöschte und eine neu hinzugekommene Datei als „umbenannt“, wenn ihr Inhalt zu einem konfigurierbaren Prozentsatz übereinstimmt. Der Standardwert liegt bei **50 %**.

```
git status -M          # zeigt Umbenennungen mit Standardschwelle
git diff -M50%          # explizite Schwelle angeben
git log --follow neu.txt
```

Der Parameter `--follow` ist besonders wertvoll: Er sorgt dafür, dass `git log` die Historie einer Datei über Umbenennungen hinweg verfolgt, statt an dem Punkt zu stoppen, an dem der alte Dateiname verschwindet.

> **Hinweis:** Wird eine Datei beim Umbenennen zusätzlich stark inhaltlich verändert, kann Git die Beziehung nicht mehr erkennen. In diesem Fall erscheinen zwei unabhängige Einträge – eine Löschung und eine Neuanlage – und die Historie „reißt“ an dieser Stelle ab.

# Dateien in andere Verzeichnisse verschieben

Das Prinzip gilt unverändert für Verschiebungen zwischen Verzeichnissen:

```
git mv src/Helper.php src/Utils/Helper.php
```

Auch hier merkt Git die neue Position im Index vor. Fehlt das Zielverzeichnis, muss es zuvor angelegt werden, da Git – im Gegensatz zu vielen anderen Werkzeugen – keine leeren Verzeichnisse verwaltet, sondern ausschließlich Dateien.

# Umbenennungen als eigenständige Commits

Für eine nachvollziehbare Projektgeschichte empfiehlt es sich, Umbenennungen **getrennt** von inhaltlichen Änderungen zu committen:

```
git mv OldClassName.php NewClassName.php
git commit -m "refactor: Klasse in NewClassName umbenennen"
```

Werden Umbenennung und Codeänderung im selben Commit vermischt, sinkt die Wahrscheinlichkeit einer korrekten automatischen Erkennung erheblich, und spätere Reviews sowie `git blame`-Analysen werden unnötig erschwert.

# Kontrolle nach dem Umbenennen

Zur Absicherung sollte der Zustand vor dem Commit stets überprüft werden:

```
git status
git diff --staged
```

`git status` bestätigt die erkannte Umbenennung, während `git diff --staged` zeigt, ob neben dem neuen Namen unbeabsichtigt auch Inhalte verändert wurden.

# Ausblick

Der hier beschriebene Kommandozeilen-Workflow bildet die Grundlage. Wie sich Umbenennungen und Verschiebungen komfortabel und mit visueller Rückmeldung direkt in PhpStorm durchführen lassen, zeigt der folgende Abschnitt *„Derselbe Workflow in PhpStorm“* □□ – inklusive automatischer Referenzanpassung im PHP-Code, die die IDE zusätzlich übernimmt.

---

Revision #1
Created 2026-07-26 17:13:21 UTC by art10m
Updated 2026-07-26 17:13:32 UTC by art10m