Kapitel 10: PhpStorm Git-Funktionen meistern □□

Einleitung: PhpStorm ist nicht nur eine großartige IDE für PHP-Entwicklung, sondern hat auch eine der besten Git-Integrationen auf dem Markt. Viele Entwickler nutzen jedoch nur einen Bruchteil der verfügbaren Funktionen. In diesem abschließenden Kapitel tauchst du tiefer in die Möglichkeiten ein: Du lernst, die Diff-Ansicht effektiv zu nutzen, um Änderungen zu visualisieren, einzelne Code-Zeilen statt ganzer Dateien zu committen (sogenanntes „Partial Commit"), und die Historie deines Projekts detailliert zu analysieren. Mit den richtigen Tastenkombinationen wird Git in PhpStorm zum natürlichen Teil deines Workflows, der dich nicht aufhält, sondern unterstützt.

Git-Funktionen in PhpStorm – Der komplette Überblick □□

PhpStorm hat eine der **umfangreichsten Git-Integrationen** aller IDEs. Fast alles, was du sonst im Terminal machen würdest, kannst du direkt in der grafischen Oberfläche erledigen – oft sogar komfortabler. Hier bekommst du einen strukturierten Überblick über alle wichtigen Git-Funktionen und wo du sie findest.

Die wichtigsten Bereiche der Git-Oberfläche

PhpStorm organisiert seine Git-Funktionen an mehreren Stellen. Sobald du diese kennst, findest du dich schnell zurecht:

```
flowchart TB
    subgraph UI["PhpStorm Git-Oberfläche"]
        MENU["□□Hauptmenü\nGit-Menü oben"]
        TOOLBAR["□□Toolbar\nSchnellzugriff-Icons"]
        STATUSBAR["□□Statusleiste\nBranch-Anzeige unten rechts"]
        TOOLWINDOW["□□Tool-Fenster\nGit-Tab unten"]
        CONTEXT["□□□Kontextmenü\nRechtsklick auf Dateien"]
        GUTTER["□□Editor-Gutter\nLinke Seite im Editor"]
    end

    MENU --> TOOLWINDOW
    TOOLBAR --> TOOLWINDOW
    STATUSBAR --> TOOLWINDOW
```

Das Git-Hauptmenü

Das **Git-Menü** in der oberen Menüleiste ist dein zentraler Anlaufpunkt für alle Git-Operationen. Hier findest du wirklich *alles* – von den Basics bis zu fortgeschrittenen Funktionen.

Menüpunkt	Funktion	Tastenkürzel
Commit	Öffnet das Commit-Fenster	`Strg + K` (Win/Linux) / `Cmd + K` (Mac)
Push	Lokale Commits auf Remote hochladen	`Strg + Shift + K` / `Cmd + Shift + K`
Pull	Änderungen vom Remote holen und mergen	–
Fetch	Nur Remote-Infos holen (ohne Merge)	–
Merge	Branches zusammenführen	–
Rebase	Branch auf anderen Branch umbasieren	–
Branches	Branch-Verwaltung öffnen	–
Show Git Log	Commit-Historie anzeigen	–
Rollback	Änderungen verwerfen	–
Stash Changes	Änderungen temporär zwischenspeichern	–
Unstash Changes	Zwischengespeicherte Änderungen wiederherstellen	–

> **Tipp:** Die meisten Menüpunkte haben Untermenüs mit weiteren Optionen. Es lohnt sich, diese einmal durchzugehen!

Die Statusleiste – dein ständiger Begleiter

Am **unteren rechten Rand** von PhpStorm findest du die Statusleiste mit der **Branch-Anzeige**. Diese kleine Anzeige ist überraschend mächtig:

- **Branch-Name anzeigen:** Du siehst sofort, auf welchem Branch du gerade arbeitest
- **Branch wechseln:** Ein Klick öffnet das Branch-Popup mit allen lokalen und Remote-Branches
- **Neuen Branch erstellen:** Direkt aus dem Popup heraus

- **Branch-Aktionen:** Merge, Rebase, Rename, Delete – alles per Rechtsklick auf einen Branch

```
flowchart LR
    STATUS["□□Statusleiste\nmain"] -->|Klick| POPUP["Branch-Popup"]
    POPUP --> LOCAL["□□Lokale Branches"]
    POPUP --> REMOTE["☁□ Remote Branches"]
    POPUP --> NEW["□ New Branch"]
    POPUP --> CHECKOUT["□□Checkout"]
```

Das Git-Tool-Fenster

Das **Git-Tool-Fenster** am unteren Rand (Reiter „Git") ist das Herzstück für die Arbeit mit der Versionshistorie. Es hat mehrere Tabs:

Log-Tab □□

Hier siehst du die **gesamte Commit-Historie** deines Projekts in einer grafischen Darstellung:

- **Commit-Graph:** Visualisiert Branches und Merges als Linien
- **Commit-Details:** Autor, Datum, Nachricht auf einen Blick
- **Diff-Ansicht:** Klick auf einen Commit zeigt alle Änderungen
- **Suche und Filter:** Nach Autor, Datum, Nachricht oder Datei filtern
- **Branch-Filter:** Nur bestimmte Branches anzeigen

Local Changes / Changes-Tab □□

Zeigt alle **aktuell geänderten Dateien** – also alles, was seit dem letzten Commit verändert wurde:

- **Unversioned Files:** Neue Dateien, die noch nicht getrackt werden
- **Modified Files:** Geänderte Dateien
- **Staging-Funktion:** Dateien per Checkbox zur Staging Area hinzufügen
- **Diff-Vorschau:** Doppelklick zeigt die Änderungen im Detail

Console-Tab □□

Zeigt die **tatsächlichen Git-Befehle**, die PhpStorm im Hintergrund ausführt. Sehr nützlich, um zu verstehen, was passiert – oder um Fehlermeldungen zu analysieren.

Kontextmenü – Git per Rechtsklick

Wenn du im **Projektbaum** oder im **Editor** mit der rechten Maustaste auf eine Datei klickst, findest du unter **Git** zahlreiche dateispezifische Optionen:

Funktion	Was sie tut
Add	Datei zur Staging Area hinzufügen
Commit File	Nur diese Datei committen
Show History	Zeigt alle Commits, die diese Datei betreffen
Show Diff	Vergleicht aktuelle Version mit letztem Commit
Annotate with Git Blame	Zeigt zeilenweise, wer wann was geändert hat
Rollback	Änderungen an dieser Datei verwerfen
Compare with Branch	Datei mit Version aus anderem Branch vergleichen

---

Der Editor-Gutter – Änderungen auf einen Blick

Am **linken Rand des Editors** (neben den Zeilennummern) zeigt PhpStorm farbige Markierungen für Änderungen:

Farbe	Bedeutung
Grün	Neue Zeilen (hinzugefügt)
Blau	Geänderte Zeilen
Dreieck/Pfeil	Gelöschte Zeilen (an dieser Stelle wurde etwas entfernt)

Klickst du auf diese Markierungen, öffnet sich ein kleines Popup mit mehreren Optionen:

- **Änderung anzeigen:** Zeigt den Diff für diese Stelle
- **Rollback:** Setzt nur diese Änderung zurück (nicht die ganze Datei!)
- **Copy old text:** Kopiert den ursprünglichen Text in die Zwischenablage

> □□ Diese Funktion ist **extrem praktisch**, um einzelne Änderungen chirurgisch rückgängig zu machen, ohne gleich die ganze Datei zurückzusetzen.

Die Toolbar – Schnellzugriff für häufige Aktionen

In der **oberen Toolbar** findest du Icons für die häufigsten Git-Operationen. Je nach PhpStorm-Version und Konfiguration können diese leicht variieren:

- **Update Project** (blauer Pfeil nach unten): Holt Änderungen vom Remote
- **Commit** (grüner Haken): Öffnet das Commit-Fenster
- **Push** (grüner Pfeil nach oben): Pusht Commits zum Remote
- **History** (Uhr-Symbol): Zeigt die Historie der aktuellen Datei
- **Rollback** (gekrümmter Pfeil): Verwirft Änderungen

Das Commit-Fenster im Detail

Das Commit-Fenster (`Strg + K` / `Cmd + K`) verdient besondere Aufmerksamkeit, weil du es **am häufigsten** nutzen wirst:

```
flowchart TB
    subgraph COMMIT["Commit-Fenster"]
        FILES["□□Dateiliste\nmit Checkboxen"]
        DIFF["□□Diff-Vorschau\nrechts daneben"]
        MESSAGE["✍□ Commit-Nachricht\nTextfeld unten"]
        OPTIONS["⚙□ Optionen\nAmend, Sign-off, etc."]
        BUTTONS["□□Buttons\nCommit / Commit and Push"]
    end

    FILES --> DIFF
    MESSAGE --> BUTTONS
    OPTIONS --> BUTTONS
```

Wichtige Funktionen im Commit-Fenster:

- **Dateien selektiv auswählen:** Nicht alle geänderten Dateien müssen in einen Commit
- **Diff vor dem Commit prüfen:** Doppelklick auf eine Datei zeigt die Änderungen
- **Amend-Checkbox:** Letzten Commit korrigieren statt neuen erstellen
- **Commit and Push:** Beides in einem Schritt erledigen

Die am häufigsten genutzten Funktionen □□

Im Alltag wirst du einen kleinen Teil der Git-Funktionen **sehr oft** nutzen, während andere nur in speziellen Situationen gebraucht werden:

Täglich mehrfach genutzt

1. **Commit** (`Strg + K` / `Cmd + K`)
   Du wirst ständig Commits machen – das ist das Herzstück von Git.
2. **Push** (`Strg + Shift + K` / `Cmd + Shift + K`)
   Nach dem Committen willst du deine Arbeit sichern.
3. **Pull / Update Project**
   Zu Beginn jeder Arbeitssession holst du dir den aktuellen Stand.
4. **Branch wechseln** (Statusleiste unten rechts)
   Wenn du mit Feature-Branches arbeitest, wechselst du regelmäßig.
5. **Diff anzeigen** (Editor-Gutter oder Doppelklick im Commit-Fenster)
   Du willst vor dem Commit sehen, was du geändert hast.

Regelmäßig genutzt

6. **Neuen Branch erstellen** (Statusleiste → New Branch)
   Für jedes neue Feature oder Experiment.
7. **Merge** (Git-Menü oder Branch-Popup)
   Wenn ein Feature fertig ist, mergst du es zurück.
8. **Git Log** (Git-Tool-Fenster)
   Um die Historie zu durchsuchen oder alte Commits zu finden.
9. **Rollback** (Editor-Gutter oder Kontextmenü)
   Um Änderungen zu verwerfen, die du doch nicht willst.
10. **Show History** (Kontextmenü auf einer Datei)
    Um zu sehen, wie sich eine bestimmte Datei entwickelt hat.

Gelegentlich genutzt

11. **Amend Commit** (Checkbox im Commit-Fenster)
    Wenn du den letzten Commit korrigieren musst.
12. **Git Blame / Annotate** (Kontextmenü)
    Um herauszufinden, wer eine bestimmte Zeile geschrieben hat.
13. **Stash / Unstash** (Git-Menü)
    Um Änderungen temporär beiseitezulegen.
14. **Cherry-Pick** (Rechtsklick auf Commit im Log)
    Um einzelne Commits in einen anderen Branch zu übernehmen.
15. **Revert Commit** (Rechtsklick auf Commit im Log)
    Um einen Commit rückgängig zu machen (mit neuem Commit).

Übersicht: Wo finde ich was?

Was will ich tun?	Wo finde ich es?
Committen	`Strg + K` oder Git-Menü → Commit
Pushen	`Strg + Shift + K` oder Git-Menü → Push
Pullen	Git-Menü → Pull oder Toolbar-Icon
Branch wechseln	Statusleiste unten rechts
Neuen Branch erstellen	Statusleiste → New Branch
Branches mergen	Git-Menü → Merge oder Branch-Popup → Merge into Current
Historie ansehen	Git-Tool-Fenster (unten) → Log-Tab
Datei-Historie ansehen	Rechtsklick auf Datei → Git → Show History
Änderungen vergleichen	Editor-Gutter oder Rechtsklick → Git → Show Diff
Änderungen verwerfen	Editor-Gutter-Klick → Rollback oder Rechtsklick → Git → Rollback
Merge-Konflikte lösen	Automatisches Popup oder Git-Menü → Resolve Conflicts
Letzten Commit ändern	Im Commit-Fenster: Checkbox „Amend commit"

Mein Tipp für den Einstieg □□

Konzentriere dich am Anfang auf diese **fünf Kernfunktionen**:

1. **Commit** (`Strg + K`)
2. **Push** (`Strg + Shift + K`)
3. **Pull** (Git-Menü oder Toolbar)
4. **Branch wechseln** (Statusleiste)
5. **Git Log ansehen** (Tool-Fenster unten)

Wenn du diese fünf beherrschst, deckst du **90% deines Git-Alltags** ab. Die restlichen Funktionen lernst du dann nach und nach kennen, wenn du sie brauchst – und du weißt jetzt, wo du sie findest! □□

Die Diff-Ansicht in PhpStorm – Änderungen verstehen und gezielt committen □□

Wenn du an deinem Code arbeitest, möchtest du oft wissen: *Was genau habe ich eigentlich seit dem letzten Commit geändert?* Und manchmal möchtest du nicht alle Änderungen auf einmal committen, sondern nur bestimmte Teile. PhpStorm bietet dir dafür eine **leistungsstarke Diff-Ansicht** und die Möglichkeit, Änderungen auf Zeilen-Ebene auszuwählen – sogenannte **partielle Commits** oder **Chunk-basiertes Staging**.

Änderungen seit dem letzten Commit anzeigen

Es gibt mehrere Wege, um zu sehen, was sich in deinen Dateien geändert hat:

Weg 1: Über das Commit-Fenster

1. Öffne das **Commit-Fenster** mit `Strg + K` (Windows/Linux) oder `Cmd + K` (macOS) – alternativ über **Git → Commit**.
2. Im linken Bereich siehst du alle Dateien, die seit dem letzten Commit verändert wurden. Sie sind farblich markiert:
   - **Blau:** Geänderte Dateien (modified)
   - **Grün:** Neue Dateien (untracked/added)
   - **Rot/Grau:** Gelöschte Dateien
3. **Doppelklicke auf eine Datei**, um die Diff-Ansicht zu öffnen.

Weg 2: Direkt im Editor

Wenn du eine Datei geöffnet hast, zeigt PhpStorm dir Änderungen direkt im **Randbereich** (Gutter) an:

- **Blaue Markierung:** Diese Zeilen wurden geändert
- **Grüne Markierung:** Diese Zeilen sind neu hinzugekommen
- **Kleines Dreieck:** Hier wurden Zeilen gelöscht

Klickst du auf eine dieser Markierungen, öffnet sich ein kleines Popup, das dir den **vorherigen Zustand** zeigt und dir erlaubt, die Änderung direkt rückgängig zu machen.

Weg 3: Über das Kontextmenü

1. Rechtsklicke auf eine Datei im **Projektbaum** oder im **Commit-Fenster**.
2. Wähle **Git → Show Diff** (oder einfach **Compare with… → Last Commit**).

Die Diff-Ansicht verstehen

Wenn du die Diff-Ansicht öffnest, siehst du standardmäßig eine **Side-by-Side-Darstellung**:

```
┌──────────────────────────────┬──────────────────────────────┐
│     Alter Zustand            │      Neuer Zustand           │
│     (letzter Commit)         │      (deine Änderungen)      │
├──────────────────────────────┼──────────────────────────────┤
│  function login() {          │  function login() {          │
│      // alte Logik           │      // verbesserte Logik    │
│      return false;           │      $user = checkUser();    │
│  }                           │      return $user !== null;  │
│                              │  }                           │
└──────────────────────────────┴──────────────────────────────┘
```

Die wichtigsten Elemente:

- **Farbige Hervorhebung:** Geänderte, hinzugefügte und entfernte Zeilen sind farblich markiert
- **Pfeile zwischen den Spalten:** Ermöglichen das Übernehmen von Änderungen in die eine oder andere Richtung
- **Navigations-Pfeile oben:** Springen zur nächsten/vorherigen Änderung (`F7` / `Shift + F7`)

> □□ **Tipp:** Du kannst die Darstellung umschalten – über das Zahnrad-Icon in der Diff-Ansicht kannst du zwischen **Side-by-Side** und **Unified View** (alles

> untereinander) wechseln.

Einzelne Änderungen zum Commit auswählen – partielle Commits

Jetzt kommt das wirklich Mächtige: Du musst **nicht die gesamte Datei** committen. PhpStorm erlaubt dir, einzelne **Chunks** (zusammenhängende Änderungsblöcke) oder sogar einzelne **Zeilen** auszuwählen.

Warum ist das nützlich?

Stell dir vor, du hast in einer Datei zwei verschiedene Dinge gemacht:

- Einen Bug im Login-System gefixt
- Nebenbei einen Kommentar verbessert

Das sind logisch **zwei verschiedene Änderungen**, die eigentlich in **zwei verschiedene Commits** gehören (mit jeweils aussagekräftiger Commit-Nachricht). Mit partiellen Commits kannst du genau das tun.

So funktioniert es in PhpStorm

1. **Öffne das Commit-Fenster** (`Strg + K` / `Cmd + K`).
2. **Klicke auf das Zahnrad-Icon** (⚙) im Commit-Fenster und aktiviere die Option **„Include into commit"** oder ähnlich, falls nicht schon aktiv.
3. **Doppelklicke auf die Datei**, um die Diff-Ansicht zu öffnen.
4. In der Diff-Ansicht siehst du nun **Checkboxen** neben jedem Änderungsblock (Chunk):

   ```
   ☑ Chunk 1: Login-Logik geändert (Zeile 15-22)
   ☐ Chunk 2: Kommentar aktualisiert (Zeile 45-46)
   ```

5. **Aktiviere oder deaktiviere die Checkboxen**, um festzulegen, welche Chunks in diesen Commit aufgenommen werden sollen.
6. Für noch feinere Kontrolle: Manche Versionen von PhpStorm erlauben es, per **Rechtsklick auf einen Chunk** einzelne Zeilen auszuwählen.
7. **Schließe die Diff-Ansicht** und schreibe deine Commit-Nachricht nur für die ausgewählten Änderungen.

8. Nach dem Commit sind die **nicht ausgewählten Änderungen** immer noch da – du kannst sie im nächsten Commit mit einer anderen Nachricht speichern.

Visuelles Schema des Prozesses

```
flowchart TD
    A["□□Datei mit mehreren Aenderungen"] --> B["Diff-Ansicht oeffnen"]
    B --> C{"Welche Chunks\ngehoeren zusammen?"}
    C --> D["Chunk 1 auswaehlen:\nLogin-Bugfix"]
    C --> E["Chunk 2 abwaehlen:\nKommentar-Update"]
    D --> F["Commit 1 erstellen:\nFix: Login-Validierung korrigiert"]
    F --> G["Chunk 2 ist noch da\nund wartet"]
    G --> H["Commit 2 erstellen:\nDocs: Kommentar aktualisiert"]
```

Praktisches Beispiel

Du hast folgende Änderungen in `UserController.php`:

Zeile	Änderung	Gehört zu
23-28	Passwort-Validierung hinzugefügt	Feature: Sicherheit
45	Tippfehler in Variable korrigiert	Fix: Typo
67-70	Neue Methode `resetPassword()`	Feature: Sicherheit

Sinnvolle Vorgehensweise:

1. **Erster Commit:** Zeilen 23-28 und 67-70 auswählen
   Commit-Nachricht: `Feat: Passwort-Validierung und Reset-Funktion hinzugefügt`
2. **Zweiter Commit:** Zeile 45 auswählen
   Commit-Nachricht: `Fix: Tippfehler in Variablenname korrigiert`

So bleibt deine Git-Historie **sauber und nachvollziehbar**.

Nützliche Tastenkürzel in der Diff-Ansicht

Aktion	Windows/Linux	macOS
Diff-Ansicht öffnen	Doppelklick auf Datei im Commit-Fenster	Doppelklick auf Datei im Commit-Fenster
Zur nächsten Änderung springen	`F7`	`F7`
Zur vorherigen Änderung springen	`Shift + F7`	`Shift + F7`
Änderung übernehmen (von links nach rechts)	`Strg + Shift + →`	`Cmd + Shift + →`
Änderung rückgängig machen	Klick auf den Rückgängig-Pfeil	Klick auf den Rückgängig-Pfeil
Diff schließen	`Esc`	`Esc`

Zusammenfassung □

- Die **Diff-Ansicht** zeigt dir präzise, was sich seit dem letzten Commit geändert hat – entweder per Doppelklick im Commit-Fenster oder über die farbigen Markierungen direkt im Editor.
- Du kannst **einzelne Chunks** (Änderungsblöcke) auswählen und damit **logisch zusammengehörige Änderungen** separat committen.
- Das führt zu einer **sauberen, nachvollziehbaren Git-Historie**, in der jeder Commit genau eine Sache macht.
- PhpStorm macht diesen Prozess besonders komfortabel durch die visuelle Darstellung mit Checkboxen neben jedem Chunk.

> □□ **Profi-Tipp:** Gewöhne dir an, vor jedem Commit kurz die Diff-Ansicht zu öffnen und zu prüfen, ob wirklich alle Änderungen zusammengehören. Das dauert nur Sekunden, spart dir aber später viel Zeit beim Debugging oder wenn du eine bestimmte Änderung suchen musst.

Git-Tastenkürzel in PhpStorm – schneller arbeiten mit Shortcuts ⌨️

Sobald du die grundlegenden Git-Funktionen in PhpStorm beherrschst, kannst du deinen Workflow erheblich beschleunigen, indem du die wichtigsten Aktionen per Tastenkürzel auslöst. Statt durch Menüs zu navigieren, erledigst du Commits, Branch-Wechsel und vieles mehr in Sekundenbruchteilen. Hier findest du die nützlichsten Shortcuts – organisiert nach Häufigkeit und Anwendungsbereich.

Die wichtigsten Shortcuts für den täglichen Git-Workflow

Diese Tastenkürzel wirst du vermutlich am häufigsten verwenden. Es lohnt sich, sie als Erstes zu verinnerlichen:

Aktion	Windows/Linux	macOS
Commit-Dialog öffnen	`Strg + K`	`Cmd + K`
Push durchführen	`Strg + Shift + K`	`Cmd + Shift + K`
Pull/Update durchführen	`Strg + T`	`Cmd + T`
VCS-Operationen-Popup	`Alt + ` ` (Backtick)	`Ctrl + V`
Git-Log anzeigen	`Alt + 9`	`Cmd + 9`

> 🖱️ **Tipp:** Das **VCS-Operationen-Popup** (`Alt + `` bzw. `Ctrl + V`` auf macOS) ist besonders praktisch – es zeigt dir ein kontextbezogenes Menü mit allen relevanten Git-Aktionen für die aktuelle Datei oder das Projekt.

Branch-Management per Tastatur

Das Wechseln und Verwalten von Branches geht mit diesen Shortcuts deutlich schneller:

Aktion	Windows/Linux	macOS
Branches-Popup öffnen	`Strg + Shift + ` `	`Cmd + Shift + ` `
Neuen Branch erstellen	Über Branches-Popup → `New Branch`	Über Branches-Popup → `New Branch`

Das **Branches-Popup** ist dein zentraler Anlaufpunkt für alles rund um Branches: Hier siehst du alle lokalen und Remote-Branches, kannst zwischen ihnen wechseln, neue erstellen, mergen oder löschen – alles ohne die Maus.

Änderungen vergleichen und navigieren

Wenn du verstehen möchtest, was sich geändert hat, helfen diese Shortcuts:

Aktion	Windows/Linux	macOS
Diff zur letzten Version anzeigen	`Strg + D` (im Commit-Fenster)	`Cmd + D`
Zur nächsten Änderung springen	`F7` (im Diff-Viewer)	`F7`
Zur vorherigen Änderung springen	`Shift + F7`	`Shift + F7`
Lokale Historie anzeigen	`Alt + Shift + H`	`Cmd + Shift + H`
Annotate (Blame) anzeigen	Rechtsklick → *Annotate with Git Blame*	–

Änderungen rückgängig machen

Für den Fall, dass du schnell einen Schritt zurück musst:

Aktion	Windows/Linux	macOS
Rollback (Änderungen verwerfen)	`Strg + Alt + Z`	`Cmd + Option + Z`

Aktion	Windows/Linux	macOS
Letzte Aktion rückgängig	`Strg + Z`	`Cmd + Z`

Der **Rollback-Shortcut** ist besonders mächtig: Er setzt ausgewählte Dateien auf den letzten Commit-Stand zurück – praktisch, wenn du dich experimentell verrannt hast.

Nützliche allgemeine Shortcuts mit Git-Bezug

Diese Tastenkürzel sind nicht Git-spezifisch, helfen aber im Git-Workflow enorm:

Aktion	Windows/Linux	macOS
Suche überall (auch Branches, Commits)	`Shift + Shift` (doppelt)	`Shift + Shift`
Aktionen suchen	`Strg + Shift + A`	`Cmd + Shift + A`
Tool-Fenster wechseln	`Alt + [Zahl]`	`Cmd + [Zahl]`

Mit **„Aktionen suchen"** (`Strg + Shift + A`) kannst du jeden Git-Befehl finden, auch wenn du den Shortcut nicht kennst – tippe einfach „Commit", „Push", „Branch" usw. ein.

Shortcuts anpassen und eigene erstellen

Falls dir ein Shortcut nicht zusagt oder du einen fehlenden hinzufügen möchtest:

1. Gehe zu **File → Settings** (bzw. **PhpStorm → Settings** auf macOS)
2. Navigiere zu **Keymap**
3. Suche nach der gewünschten Aktion (z. B. „Git Push")
4. Rechtsklick → **Add Keyboard Shortcut**
5. Drücke deine gewünschte Tastenkombination und bestätige

Du kannst auch eine komplett andere Keymap wählen (z. B. *Eclipse* oder *Visual Studio*), falls du von einer anderen IDE kommst.

Mein Empfohlenes „Starter-Set" □□

Wenn du nur fünf Shortcuts lernen möchtest, nimm diese:

1. `Strg + K` – Commit-Dialog öffnen
2. `Strg + Shift + K` – Push durchführen
3. `Strg + T` – Pull/Update holen
4. `Strg + Shift + ` ` – Branches-Popup öffnen
5. `Strg + Alt + Z` – Änderungen verwerfen (Rollback)

Mit diesen fünf Shortcuts deckst du etwa 80 % deiner täglichen Git-Interaktionen ab – und sparst dabei jede Menge Zeit. Die restlichen Shortcuts kommen dann nach und nach dazu, je nachdem welche Funktionen du häufiger nutzt.