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! □□

Revision #1
Created 2026-06-17 04:50:18 UTC by art10m
Updated 2026-06-17 04:52:14 UTC by art10m