Git-Unterstützung in PhpStorm aktivieren

Ziel dieser Einrichtung

PhpStorm besitzt eine integrierte Unterstützung für Git. Sie ergänzt die Git-Kommandozeile um grafische Ansichten für Änderungen, Commits, Branches, Konflikte, Historie und Remotes.

Damit PhpStorm Git verwenden kann, müssen zwei Voraussetzungen erfüllt sein:

1. **Git for Windows ist installiert** und im Terminal verfügbar.
2. PhpStorm kennt den Pfad zur Git-Programmdatei und das gewünschte Projekt ist als Git-Repository eingerichtet.

> **Wichtig:** Git ist kein optionales PhpStorm-Plugin, das separat installiert werden muss. Die Versionskontrollintegration gehört standardmäßig zur IDE. PhpStorm nutzt jedoch eine lokale Git-Installation, um Git-Befehle auszuführen.

Voraussetzungen prüfen

Öffne vor der Einrichtung ein Terminal, beispielsweise **Git Bash**, **PowerShell** oder das integrierte Terminal von PhpStorm, und prüfe die Installation:

```
git --version
```

Eine erfolgreiche Ausgabe ähnelt diesem Beispiel:

```
git version 2.47.1.windows.1
```

Erscheint stattdessen eine Fehlermeldung wie `git is not recognized` oder `git: command not found`, ist Git entweder nicht installiert oder nicht in der `PATH`-Variable verfügbar. Richte in diesem Fall

zunächst Git for Windows korrekt ein.

Prüfe außerdem, ob Git bereits eine Identität kennt:

```
git config --global user.name
git config --global user.email
```

Diese Werte sind nicht nötig, um Git in PhpStorm zu *aktivieren*, aber spätestens für den ersten Commit erforderlich.

Ein bestehendes Git-Repository in PhpStorm öffnen

Wenn dein Projekt bereits ein `.git`-Verzeichnis enthält, erkennt PhpStorm das Repository meistens automatisch.

1. Starte PhpStorm.
2. Wähle **File → Open**.
3. Öffne den **Stammordner** des Projekts — also den Ordner, in dem sich das versteckte Verzeichnis `.git` befindet.
4. Bestätige bei Bedarf, dass das Projekt in einem neuen Fenster oder im aktuellen Fenster geöffnet werden soll.

Nach dem Öffnen sollte PhpStorm die Git-Integration aktivieren. Typische Hinweise dafür sind:

- Der Projektname oder ein Branch-Name erscheint in der Statusleiste.
- Das Menü **Git** ist verfügbar.
- Im Projektfenster werden geänderte Dateien farblich markiert.
- Das Werkzeugfenster **Commit** kann über die linke oder untere Werkzeugleiste geöffnet werden.
- Unter **Git → Show Git Log** ist die Historie erreichbar.

Falls diese Elemente nicht erscheinen, prüfe die folgenden Schritte manuell.

Ein neues Projekt unter Versionskontrolle stellen

Ein lokaler Projektordner ohne `.git`-Verzeichnis ist zunächst noch kein Git-Repository. Du kannst Git direkt aus PhpStorm aktivieren.

1. Öffne das gewünschte Projekt in PhpStorm.
2. Wähle im Hauptmenü **VCS → Enable Version Control Integration**.
3. Wähle im Dialog **Git** aus.
4. Bestätige mit **OK**.

PhpStorm führt im Hintergrund im Wesentlichen diesen Befehl aus:

```
git init
```

Dadurch entsteht im Projektstamm ein verborgenes Verzeichnis namens `.git`. Es enthält die komplette lokale Git-Datenbank: Historie, Branch-Referenzen, Konfiguration, Index und weitere Metadaten.

Ergebnis kontrollieren

Öffne das integrierte Terminal über **View → Tool Windows → Terminal** und führe aus:

```
git status
```

Bei einem neuen Repository sieht die Ausgabe ungefähr so aus:

```
On branch main

No commits yet

nothing to commit
```

Der Name des Anfangs-Branches kann je nach Git-Konfiguration auch `master` oder anders lauten. Für neue Projekte ist `main` heute die übliche Konvention.

Git-Programm in PhpStorm konfigurieren

PhpStorm muss wissen, welche Git-Programmdatei es aufrufen soll. Öffne dazu:

File → Settings → Version Control → Git

Unter macOS wäre der entsprechende Pfad **PhpStorm → Settings → Version Control → Git**. In diesem Kurs liegt der Fokus jedoch auf Windows 11.

Im Feld **Path to Git executable** sollte normalerweise automatisch ein gültiger Pfad eingetragen sein, beispielsweise:

```
C:\Program Files\Git\bin\git.exe
```

oder:

```
C:\Program Files\Git\cmd\git.exe
```

Wähle anschließend **Test**.

Bei erfolgreicher Erkennung meldet PhpStorm etwa:

```
Git executed successfully
```

Zusätzlich wird die gefundene Git-Version angezeigt.

Welcher Git-Pfad ist korrekt?

Unter Windows existieren häufig mehrere ausführbare Git-Dateien. Für PhpStorm ist in der Regel einer dieser Pfade geeignet:

```
C:\Program Files\Git\bin\git.exe
```

```
C:\Program Files\Git\cmd\git.exe
```

Wenn PhpStorm Git automatisch gefunden hat und der Test erfolgreich ist, solltest du den Pfad nicht unnötig ändern.

> **Empfehlung:** Verwende dieselbe Git-for-Windows-Installation in PhpStorm, Git Bash und PowerShell. Unterschiedliche Git-Installationen können abweichende Konfigurationen, Credential Helper oder SSH-Einstellungen verwenden und später schwer nachvollziehbare Unterschiede verursachen.

Versionskontrollzuordnung des Projektordners prüfen

PhpStorm verwaltet, welche Ordner eines Projekts welchem Versionskontrollsystem zugeordnet sind. Das ist besonders relevant bei komplexen Projekten, Monorepos oder verschachtelten Repositories.

Öffne:

File → Settings → Version Control → Directory Mappings

Dort sollte eine Zuordnung sichtbar sein:

Verzeichnis	VCS
Projektstamm, beispielsweise `C:\Projekte\mein-projekt`	Git

Fehlt die Zuordnung:

1. Klicke auf **+**.
2. Wähle den Projektstamm oder den Ordner mit dem `.git`-Verzeichnis.
3. Wähle als VCS **Git**.
4. Übernimm die Einstellungen mit **Apply** und **OK**.

Nicht versehentlich einen Unterordner zuordnen

Die Zuordnung sollte normalerweise auf den Git-Projektstamm zeigen:

```
mein-projekt/
├── .git/
├── src/
├── tests/
├── composer.json
└── README.md
```

Würdest du nur `src/` zuordnen, obwohl `.git/` im übergeordneten Projektordner liegt, können Historie, Änderungen und Branches unvollständig oder gar nicht angezeigt werden.

Die Git-Integration im Alltag erkennen

Nach erfolgreicher Aktivierung arbeitet PhpStorm kontinuierlich mit dem lokalen Git-Repository. Du musst Git nicht vor jeder Änderung erneut einschalten.

Dateistatus im Projektfenster

PhpStorm markiert Dateien abhängig von ihrem Git-Zustand. Die genaue Farbgebung hängt vom Farbschema ab, typische Bedeutungen sind jedoch:

- **neu oder unversioniert:** Die Datei wurde noch nicht zu Git hinzugefügt.
- **geändert:** Eine bereits verfolgte Datei wurde verändert.
- **hinzugefügt:** Die Datei ist für den nächsten Commit vorgemerkt.
- **gelöscht:** Eine verfolgte Datei wurde entfernt.
- **ignoriert:** Eine Regel aus `.gitignore` schließt die Datei aus.

Die Farben sind nur eine visuelle Hilfe. Die zuverlässige Quelle bleibt immer der Git-Zustand, den du auch mit folgendem Befehl prüfen kannst:

```
git status
```

Branch-Anzeige

In der Statusleiste zeigt PhpStorm üblicherweise den aktuell ausgecheckten Branch an, etwa:

```
main
```

Ein Klick auf den Branch-Namen öffnet das Branch-Menü. Dort kannst du später Branches erstellen, wechseln, mergen, rebasen oder Remote-Branches ansehen.

Commit-Werkzeugfenster

Das Werkzeugfenster **Commit** sammelt lokale Änderungen und erlaubt unter anderem:

- Dateien für einen Commit auszuwählen,
- Diffs vor dem Commit zu prüfen,
- Commit-Nachrichten zu schreiben,

- Prüfungen auszuführen,
- Änderungen direkt zu committen oder zu committen und zu pushen.

Die IDE ersetzt dabei nicht das Git-Modell. Sie stellt lediglich eine Oberfläche für dieselben grundlegenden Git-Operationen bereit.

Aktivierung mit der Kommandozeile vergleichen

Die zentrale Aktion lässt sich sowohl in PhpStorm als auch im Terminal ausführen:

Aufgabe	PhpStorm	Kommandozeile
Neues Git-Repository erstellen	**VCS → Enable Version Control Integration → Git**	`git init`
Repository-Zustand prüfen	Commit-Fenster oder Git-Ansichten	`git status`
Branch anzeigen	Statusleiste oder Branch-Menü	`git branch --show-current`
Änderungen ansehen	Commit-Fenster oder Diff-Ansicht	`git diff`
Git-Historie öffnen	**Git → Show Git Log**	`git log`

Diese Zuordnung ist wichtig: Wenn du verstehst, welche Git-Operation PhpStorm ausführt, kannst du bei Problemen gezielt prüfen und bist nicht von der Benutzeroberfläche abhängig.

Aktivierung testen

Lege zum Test eine Datei im Projektstamm an, beispielsweise `README.md`, mit folgendem Inhalt:

```
# Mein Git-Übungsprojekt
```

Speichere die Datei. PhpStorm sollte sie nun als **unversionierte Datei** erkennen.

Prüfe anschließend im Terminal:

```
git status
```

Die Ausgabe sollte einen ähnlichen Abschnitt enthalten:

```
Untracked files:
  README.md
```

Füge die Datei testweise über PhpStorm zu Git hinzu:

1. Öffne das Werkzeugfenster **Commit**.
2. Aktiviere das Kontrollkästchen neben `README.md`.
3. Gib eine Commit-Nachricht ein, beispielsweise:

```
docs: README hinzufügen
```

4. Wähle **Commit**.

Alternativ erledigst du exakt denselben Ablauf im Terminal:

```
git add README.md
git commit -m "docs: README hinzufügen"
```

Zeige danach die Historie an:

```
git log --oneline
```

Wenn der Commit erscheint, ist Git sowohl lokal als auch in PhpStorm korrekt aktiv.

Häufige Probleme und ihre Ursachen

PhpStorm findet Git nicht

Symptom: Der Test im Bereich **Settings → Version Control → Git** schlägt fehl.

Typische Ursachen:

- Git for Windows ist nicht installiert.
- Der hinterlegte Pfad zeigt auf eine nicht vorhandene Datei.
- Eine alte Git-Installation wurde entfernt, aber PhpStorm verwendet noch den alten Pfad.
- Die Datei `git.exe` ist durch Sicherheitssoftware blockiert.

Lösung: Suche `git.exe` im Installationsordner von Git for Windows, trage einen gültigen Pfad ein und führe erneut **Test** aus.

Das Menü „Git“ fehlt oder zeigt keine Aktionen

Symptom: Das Projekt wird nicht als Git-Repository behandelt.

Typische Ursachen:

- Das Projekt enthält noch kein `.git`-Verzeichnis.
- Der falsche Ordner wurde geöffnet.
- Unter **Directory Mappings** ist keine Git-Zuordnung vorhanden.
- Das Repository liegt in einem übergeordneten Ordner, der nicht Teil des geöffneten Projekts ist.

Lösung: Öffne den tatsächlichen Repository-Stamm oder richte über **VCS → Enable Version Control Integration** ein neues Repository ein. Kontrolliere danach die Directory Mappings.

Änderungen werden nicht angezeigt

Symptom: Eine gespeicherte Datei erscheint nicht im Commit-Fenster.

Prüfe zuerst im Terminal:

```
git status
```

Mögliche Ursachen sind:

- Die Datei wird von `.gitignore` ausgeschlossen.
- Die Datei liegt außerhalb des Repository-Stamms.
- Die Datei wurde noch nicht gespeichert.
- PhpStorm ordnet das Verzeichnis nicht Git zu.
- Die Datei ist Teil einer anderen Changelist oder ein Filter blendet sie aus.

Ob eine Datei durch eine Ignore-Regel ausgeschlossen wird, kannst du präzise prüfen:

```
git check-ignore -v pfad\zur\datei
```

Git zeigt dann die konkrete Ignore-Datei und Regel an, die den Ausschluss verursacht.

Das Projekt enthält mehrere Git-Repositories

Ein Projekt kann mehrere unabhängige Repositories enthalten, etwa bei Modulen, Submodulen oder nebeneinanderliegenden Anwendungen:

```
arbeitsbereich/
├── backend/
│   └── .git/
└── frontend/
    └── .git/
```

In diesem Fall kann PhpStorm mehrere Directory Mappings verwalten. Jedes Repository erhält eine eigene Git-Zuordnung. Prüfe besonders sorgfältig, in welchem Repository du gerade commitest, Branches wechselst oder Remotes konfigurierst.

> **Achtung:** Ein verschachteltes `.git`-Verzeichnis ist nicht automatisch ein Submodule. Ein Submodule ist ein bewusst in Git eingetragener Verweis auf ein anderes Repository. Mehrere zufällig verschachtelte Repositories führen dagegen oft zu Verwirrung.

Sichere Ausgangskonfiguration

Nach diesem Abschnitt sollte dein Projekt diese Eigenschaften haben:

- Git for Windows ist installiert und über `git --version` erreichbar.
- PhpStorm kennt einen funktionierenden Pfad zu `git.exe`.
- Der Projektstamm ist Git zugeordnet.
- Ein `.git`-Verzeichnis existiert im Repository-Stamm.
- PhpStorm erkennt neue, geänderte und vorgemerkte Dateien.
- Der aktuelle Branch wird in der IDE angezeigt.
- `git status` im Terminal und die Anzeige in PhpStorm beschreiben denselben Zustand.

Damit ist die technische Grundlage gelegt. Als Nächstes prüfst du das Git-Programm und seine Version in PhpStorm genauer und richtest anschließend die Commit- und Historienwerkzeuge für den täglichen Workflow ein.

Revision #1
Created 2026-07-25 11:13:48 UTC by art10m
Updated 2026-07-25 11:13:48 UTC by art10m