Kapitel 2: Git installieren und einrichten

Einleitung: Nachdem du nun weißt, was Git ist und warum es so nützlich ist, wird es Zeit, es auf deinem Computer zum Laufen zu bringen. Die Installation selbst ist unkompliziert, aber es gibt ein paar wichtige Einstellungen, die du direkt am Anfang vornehmen solltest – insbesondere deine Identität (Name und E-Mail), die in jedem deiner Commits gespeichert wird. Außerdem musst du sicherstellen, dass PhpStorm weiß, wo Git installiert ist, damit du später alle Versionsverwaltungs-Funktionen direkt aus deiner Entwicklungsumgebung nutzen kannst, ohne ständig zwischen verschiedenen Programmen wechseln zu müssen. Dieses Kapitel führt dich durch alle notwendigen Schritte.

Git installieren und einrichten □□

Die Installation von Git ist auf allen gängigen Betriebssystemen unkompliziert. Hier findest du konkrete Schritte für **Windows**, **macOS** und **Linux** – inklusive der Überprüfung, ob alles geklappt hat.

Installation nach Betriebssystem

Windows

1. **Installer herunterladen**

   Gehe auf die offizielle Git-Website git-scm.com/downloads/win und lade die **64-Bit-Version** herunter (für die meisten modernen Rechner passend).
2. **Installer ausführen**
   Starte die heruntergeladene `.exe`-Datei. Du wirst durch einen Installationsassistenten geführt. Die **Standardeinstellungen** sind für Anfänger vollkommen ausreichend – du kannst also meist einfach auf „Next" klicken. Ein paar Punkte, auf die du achten kannst:
   - **Default Editor:** Hier kannst du z. B. *Visual Studio Code* oder *Notepad++* wählen, falls du Vim nicht magst (Vim ist für Einsteiger oft verwirrend).
   - **PATH-Einstellung:** Wähle „Git from the command line and also from 3rd-party software" – so kannst du Git überall nutzen.
   - **Line Ending Conversion:** Die Standardoption „Checkout Windows-style, commit Unix-style" ist sinnvoll.
3. **Installation abschließen**
   Klicke am Ende auf „Install" und warte, bis der Vorgang abgeschlossen ist.

---

macOS

Auf dem Mac hast du mehrere Möglichkeiten:

1. **Über Xcode Command Line Tools** *(einfachste Methode)*

Öffne das **Terminal** (findest du über Spotlight mit `Cmd + Leertaste` → „Terminal") und gib ein:

```
git --version
```

Falls Git noch nicht installiert ist, erscheint automatisch ein Dialog, der dich fragt, ob du die *Xcode Command Line Tools* installieren möchtest. Bestätige mit „Installieren" – Git ist dann dabei.

2. **Über Homebrew** *(falls du Homebrew bereits nutzt)*

   ```
   brew install git
   ```

   Homebrew ist ein beliebter Paketmanager für macOS. Falls du ihn noch nicht hast, findest du ihn unter brew.sh.

3. **Offizieller Installer**

   Du kannst auch den Installer von git-scm.com/downloads/mac herunterladen und wie unter Windows durchklicken.

Linux

Auf den meisten Linux-Distributionen ist Git entweder schon vorinstalliert oder lässt sich mit einem einzigen Befehl nachinstallieren:

- **Debian/Ubuntu:**

  ```
  sudo apt update
  sudo apt install git
  ```

- **Fedora:**

  ```
  sudo dnf install git
  ```

- **Arch Linux:**

  ```
  sudo pacman -S git
  ```

> **Tipp:** Falls du unsicher bist, welche Distribution du nutzt, kannst du mit `cat /etc/os-release` nachsehen.

Überprüfen, ob die Installation funktioniert hat

Öffne ein **Terminal** (unter Windows: *Git Bash*, *PowerShell* oder *Eingabeaufforderung*; unter macOS/Linux: *Terminal*) und gib folgenden Befehl ein:

```
git --version
```

Wenn alles geklappt hat, siehst du eine Ausgabe wie:

```
git version 2.45.0
```

Die genaue Versionsnummer kann variieren – wichtig ist nur, dass **keine Fehlermeldung** erscheint.

Erste Konfiguration nach der Installation

Bevor du Git produktiv nutzt, solltest du deinen **Namen** und deine **E-Mail-Adresse** hinterlegen. Diese Informationen werden in jedem Commit gespeichert, sodass nachvollziehbar ist, wer welche Änderung gemacht hat.

```
git config --global user.name "Dein Name"
git config --global user.email "deine@email.de"
```

> **Hinweis:** Nutze bei `user.email` am besten dieselbe Adresse, die du später auch bei GitHub verwendest – so werden deine Commits dort korrekt mit deinem Profil verknüpft.

Du kannst deine Einstellungen jederzeit überprüfen mit:

```
git config --list
```

Zusammenfassung

Schritt	Windows	macOS	Linux
Installieren	Installer von git-scm.com	`git --version` (Xcode Tools) oder Homebrew	`apt`, `dnf` oder `pacman`
Prüfen	`git --version`	`git --version`	`git --version`
Konfigurieren	`git config --global ...`	`git config --global ...`	`git config --global ...`

Damit ist Git einsatzbereit und du kannst mit deinem ersten Repository loslegen! □□

Git konfigurieren: Name und E-Mail-Adresse einrichten ⚙

Bevor du deinen ersten Commit machst, solltest du Git mitteilen, **wer du bist**. Das klingt vielleicht nach einer Formalität, hat aber einen sehr konkreten Grund.

Warum ist diese Konfiguration wichtig?

Jeder Commit, den du in Git erstellst, wird mit einem **Autor** versehen – also mit einem Namen und einer E-Mail-Adresse. Diese Informationen werden **dauerhaft in der Versionshistorie gespeichert** und sind für jeden sichtbar, der Zugriff auf das Repository hat.

Das ist aus mehreren Gründen relevant:

- **Nachvollziehbarkeit:** In einem Projekt (egal ob allein oder im Team) kannst du später genau sehen, *wer* eine bestimmte Änderung gemacht hat. Das hilft beim Debugging und bei der Zusammenarbeit enorm.
- **Zuordnung auf Plattformen wie GitHub:** Wenn deine konfigurierte E-Mail-Adresse mit deinem GitHub-Konto verknüpft ist, werden deine Commits automatisch deinem Profil zugeordnet – inklusive Profilbild und Verlinkung.
- **Professionalität:** Commits mit „Unknown" oder einer kryptischen Kennung wirken unprofessionell und erschweren die Zusammenarbeit.

So richtest du Name und E-Mail ein

Öffne ein Terminal (unter Windows z. B. *Git Bash*, *PowerShell* oder *CMD*; unter macOS/Linux das normale Terminal) und führe diese beiden Befehle aus:

```
git config --global user.name "Dein Name"
git config --global user.email "deine.email@beispiel.de"
```

Ersetze dabei natürlich die Platzhalter durch deine echten Daten. Ein konkretes Beispiel:

```
git config --global user.name "Max Mustermann"
git config --global user.email "max.mustermann@gmail.com"
```

> □□ **Hinweis zur E-Mail:** Verwende am besten dieselbe E-Mail-Adresse, die du auch bei GitHub (oder einer anderen Plattform) nutzt. So werden deine Commits korrekt zugeordnet.

Was bedeutet `--global`?

Die Option `--global` sorgt dafür, dass diese Einstellungen **für alle Git-Repositories** auf deinem Computer gelten. Die Konfiguration wird in einer Datei namens `.gitconfig` in deinem Benutzerverzeichnis gespeichert.

Falls du für ein *einzelnes Projekt* andere Daten verwenden möchtest (z. B. eine Firmen-E-Mail für berufliche Projekte), kannst du die Konfiguration **ohne** `--global` im jeweiligen Repository-Ordner ausführen:

```
git config user.name "Firmenname Max"
git config user.email "max@firma.de"
```

Diese lokale Einstellung überschreibt dann die globale – aber nur für dieses eine Repository.

Überprüfen, ob die Konfiguration funktioniert hat

Mit diesen Befehlen kannst du dir anzeigen lassen, was Git aktuell gespeichert hat:

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

Die Ausgabe sollte dann entsprechend dein Name und deine E-Mail sein. Alternativ kannst du dir *alle* globalen Einstellungen auf einmal anzeigen lassen:

```
git config --global --list
```

Was passiert, wenn du das nicht konfigurierst? □□

Wenn du versuchst, einen Commit zu erstellen, ohne Name und E-Mail konfiguriert zu haben, passiert Folgendes:

1. **Git verweigert den Commit** und zeigt eine Fehlermeldung wie diese:

   ```
   Author identity unknown

   *** Please tell me who you are.

   Run
     git config --global user.email "you@example.com"
     git config --global user.name "Your Name"
   ```

2. **Kein Commit wird erstellt**, bis du die Konfiguration nachholst.

Das ist also kein stilles Problem, das später Ärger macht – Git zwingt dich förmlich dazu, diese Angaben zu machen, bevor du loslegst. Trotzdem ist es besser, das *einmal sauber am Anfang* zu erledigen, als bei jedem neuen Projekt darüber zu stolpern.

Zusammenfassung □□

Befehl	Wirkung
`git config --global user.name "Name"`	Setzt deinen Namen für alle Repos
`git config --global user.email "email"`	Setzt deine E-Mail für alle Repos
`git config --global --list`	Zeigt alle globalen Einstellungen
`git config user.name` (ohne `--global`)	Setzt den Namen nur für das aktuelle Repo

Nach dieser einmaligen Konfiguration bist du bereit für deinen ersten Commit! □□

Git in PhpStorm einrichten und nutzen □□

PhpStorm hat eine **hervorragende Git-Integration**, die dir erlaubt, fast alle Git-Operationen direkt in der IDE durchzuführen – ohne ständig ins Terminal wechseln zu müssen. Die Einrichtung ist in der Regel unkompliziert, da PhpStorm Git oft automatisch erkennt.

Automatische Erkennung prüfen

In den meisten Fällen findet PhpStorm deine Git-Installation **von selbst**, sobald Git korrekt installiert ist. Um das zu überprüfen:

1. Öffne PhpStorm und gehe zu **File → Settings** (auf macOS: **PhpStorm → Settings**).
2. Navigiere im linken Menü zu **Version Control → Git**.
3. Im Feld **Path to Git executable** siehst du den Pfad, den PhpStorm verwendet. Typische Werte sind:
   - **Windows:** `C:\Program Files\Git\bin\git.exe`
   - **macOS:** `/usr/bin/git` oder `/usr/local/bin/git`
   - **Linux:** `/usr/bin/git`
4. Klicke auf den Button **Test** neben dem Pfad. Wenn alles funktioniert, erscheint eine Meldung wie:

   > *„Git version 2.x.x"* □

Falls PhpStorm Git nicht automatisch findet, musst du den Pfad manuell angeben.

Git-Pfad manuell konfigurieren

Sollte der Test fehlschlagen oder das Feld leer sein, kannst du den Pfad selbst eintragen:

1. **Git-Pfad herausfinden** – Öffne ein Terminal (oder die Eingabeaufforderung unter Windows) und gib ein:

```
# Windows (Git Bash oder CMD)
where git

# macOS / Linux
which git
```

Die Ausgabe zeigt dir den vollständigen Pfad zur Git-Executable.

2. **Pfad in PhpStorm eintragen** – Kopiere den Pfad und füge ihn in das Feld **Path to Git executable** ein. Unter Windows achte darauf, dass du auf `git.exe` im `bin`-Ordner verweist (nicht auf `git.cmd`).
3. **Erneut testen** – Klicke wieder auf **Test**, um sicherzustellen, dass die Verbindung funktioniert.

Erste Schritte mit Git in PhpStorm

Sobald Git erkannt wird, stehen dir viele praktische Funktionen zur Verfügung:

- **Neues Repository initialisieren:** Öffne ein Projekt ohne Git und wähle **VCS → Create Git Repository**. PhpStorm erstellt dann `.git` im Projektordner.
- **Bestehendes Repository öffnen:** Wenn du ein Projekt öffnest, das bereits ein `.git`-Verzeichnis enthält, aktiviert PhpStorm die Git-Integration automatisch.
- **Änderungen sehen:** Im Tab **Commit** (links oder unten, je nach Layout) siehst du alle geänderten Dateien. Hier kannst du Dateien stagen und committen.
- **Terminal nutzen:** Falls du doch mal einen Git-Befehl manuell eingeben möchtest, öffne das integrierte Terminal mit **Alt + F12** (Windows/Linux) bzw. **⌥ + F12** (macOS).

Häufige Probleme und Lösungen

Problem	Mögliche Ursache	Lösung
„Git is not installed"	Git nicht installiert oder Pfad falsch	Git installieren und Pfad manuell setzen
Test zeigt Fehler trotz korrektem Pfad	Falsches Binary gewählt (z. B. `git.cmd` statt `git.exe`)	Pfad auf `.../bin/git.exe` korrigieren
Git-Funktionen ausgegraut	Projekt ist kein Git-Repository	Repository initialisieren oder klonen
Commit-Button fehlt	Tool-Window nicht sichtbar	**View → Tool Windows → Commit** aktivieren

> **Tipp:** Nach einer Neuinstallation von Git solltest du PhpStorm einmal neu starten, damit die Änderungen sicher erkannt werden.

Zusammenfassung

Die Git-Integration in PhpStorm einzurichten ist meistens eine Sache von wenigen Klicks: **Settings → Version Control → Git → Test**. Sobald der Test erfolgreich ist, kannst du Commits, Branches, Merges und vieles mehr direkt in der IDE erledigen – bequem und ohne Kontextwechsel.