:github-logo-icon-16175-windows.ico?64x64

Github

allgemeine Definitionen

Branch

Ein Branch (deutsch: Zweig) in GitHub ist ein eigenständiger Entwicklungszweig innerhalb eines Repositories. Er ermöglicht es, neue Features zu entwickeln, Fehler zu beheben oder Ideen auszuprobieren, ohne den Hauptzweig (meist „main“ oder „master“) zu beeinflussen. Ein Branch ist dabei wie ein „abgetrennter Arbeitsbereich“ oder eine Art „Paralleluniversum“ für deinen Code

Typischerweise wird ein Branch vom aktuellen Stand eines bestehenden Branches erstellt. Änderungen, die du auf einem Branch vornimmst, sind isoliert von anderen Branches. Erst wenn du zufrieden bist, kannst du die Änderungen über einen sogenannten Merge (Zusammenführung) wieder in den Hauptzweig übernehmen

Repüository

Ein Repository auf GitHub ist das grundlegendste Element der Plattform. Es ist ein Speicherort, an dem du deinen gesamten Code, alle zugehörigen Dateien sowie deren vollständigen Revisionsverlauf ablegen und verwalten kannst. In einem Repository können mehrere Personen gemeinsam arbeiten, Änderungen verfolgen und diskutieren. Repositories können entweder öffentlich (für alle sichtbar) oder privat (nur für ausgewählte Personen zugänglich) sein

Pull

Auf GitHub bedeutet „pull“, dass du Änderungen aus einem entfernten (Remote-)Repository herunterlädst und diese direkt in dein lokales Repository integrierst. Technisch kombiniert der Befehl `git pull` zwei Schritte: Zuerst werden mit `git fetch` die neuesten Änderungen vom Remote-Repository geholt, anschließend werden diese Änderungen mit `git merge` in deinen aktuellen lokalen Branch eingepflegt. So bleibt dein lokaler Stand immer aktuell mit dem, was auf GitHub veröffentlicht wurde.

Mit „pull“ synchronisierst du dein lokales Repository mit den neuesten Änderungen aus dem Remote-Repository auf GitHub

Fetch

Fetch auf GitHub bedeutet, dass du die neuesten Änderungen (wie neue Commits, Branches oder Tags) aus einem entfernten Repository herunterlädst und in dein lokales Repository überträgst –

allerdings ohne diese Änderungen direkt in deinen aktuellen Arbeitsstand (Branch) zu übernehmen.

Mit `fetch` kannst du dir also zuerst anschauen, was sich im Remote-Repository geändert hat, bevor du entscheidest, ob und wie du diese Änderungen in deinen eigenen Code integrierst. Erst durch einen anschließenden Merge oder Rebase werden die heruntergeladenen Änderungen mit deinem aktuellen Stand zusammengeführt

Github.com und Github Dektop für Windows

Um deine Repositories unter Windows sowohl im GitHub Desktop Client als auch auf GitHub im Web abzugleichen und immer synchron zu halten, gehst du wie folgt vor:

a) Abgleichen (Synchronisieren)

Mit GitHub Desktop:

Du kannst dein Repository mit GitHub Desktop synchronisieren, indem du auf den Button „Fetch origin“ oder „Pull origin“ klickst.

- „Fetch origin“ holt die neuesten Änderungen vom Remote-Repository, ohne sie direkt in deinen Branch zu übernehmen.
- „Pull origin“ holt die Änderungen und integriert sie direkt in deinen aktuellen Branch.

Pushen deiner Änderungen:
Wenn du lokal Änderungen vorgenommen hast, kannst du diese mit „Push origin“ wieder auf GitHub hochladen.

Ablauf im Überblick:

1. Änderungen aus dem Web oder von anderen synchronisieren:
   - Klicke auf „Fetch origin“ (holt nur Änderungen) oder „Pull origin“ (holt und integriert Änderungen).
2. Eigene Änderungen hochladen:
   - Klicke auf „Push origin“

b) Automatisches Synchronisieren bei Änderungen

Eine komplett automatische Synchronisierung bei jeder Änderung ist mit GitHub Desktop allein nicht vorgesehen. Die Synchronisierung (Pull/Push) musst du manuell anstoßen. Es gibt aber Workarounds:

- **Regelmäßig manuell synchronisieren:**
  Gewöhne dir an, vor und nach deiner Arbeit auf „Fetch origin“/„Pull origin“ und „Push origin“ zu klicken.
- **Automatisierung mit Skripten:**
  Fortgeschrittene Nutzer können mit einem Bash- oder PowerShell-Skript regelmäßig `git pull` und `git push` ausführen, um Änderungen automatisch zu synchronisieren[5]. Das ist aber außerhalb von GitHub Desktop und setzt Git-Kenntnisse voraus.

Andere Tools:
In Entwicklungsumgebungen wie Visual Studio kannst du ebenfalls Synchronisierungsfunktionen nutzen, die Pull und Push in einem Schritt ausführen können.

Zusammengefasst

- **Abgleichen:**
  Nutze „Fetch origin“, „Pull origin“ und „Push origin“ in GitHub Desktop, um lokale und Remote-Repositories abzugleichen27.
- **Synchronisieren:**
  Synchronisiere regelmäßig manuell oder automatisiere den Vorgang mit Git-Skripten, wenn du fortgeschritten bist5. Eine vollautomatische Synchronisation bei jeder Änderung ist mit GitHub Desktop allein nicht möglich, aber mit etwas Disziplin oder eigenen Skripten kannst du deine Repositories sehr einfach aktuell halten.

1. Ein bestehendes GitHub-Repository klonen

Wenn das Repository schon auf GitHub existiert:

1. GitHub Desktop öffnen
   Klicke auf File → Clone repository…
2. Wähle das gewünschte Repository aus und klicke auf Clone.
3. Wähle einen lokalen Ordner, in dem das Repository gespeichert werden soll.

Ergebnis:
Das lokale Repository ist jetzt mit dem GitHub-Repository verbunden und GitHub Desktop weiß, wie es heißt und wo es liegt.

2. Ein neues lokales Repository mit GitHub verbinden

Wenn du ein neues Repository lokal erstellt hast und es auf GitHub veröffentlichen willst:

1. **Repository lokal in GitHub Desktop öffnen**
2. Klicke oben auf **Publish repository**.
3. Gib den gewünschten Namen für das Repository auf GitHub ein (das kann derselbe oder ein anderer Name sein).
4. Wähle, ob das Repository öffentlich oder privat sein soll.
5. Klicke auf **Publish Repository**.

Ergebnis:
Das Repository wird auf GitHub erstellt und mit deinem lokalen Repository verknüpft.

3. Ein bestehendes lokales Repository mit einem

bestehenden GitHub-Repository verbinden

Wenn du ein Repository lokal hast und es mit einem bereits existierenden GitHub-Repository verbinden willst (z.B. weil du das Repository umbenannt hast oder die Verbindung verloren ging):

1. Öffne das lokale Repository in GitHub Desktop.
2. Klicke auf **Repository → Repository Settings…**
3. Unter **Remote** kannst du die URL des GitHub-Repositories eintragen oder ändern.
   Die URL findest du auf GitHub im Web unter „Code“ (z.B. `https://github.com/dein-benutzername/dein-repository.git`)
4. Speichere die Änderungen.

Ergebnis:
Dein lokales Repository ist jetzt mit dem gewünschten GitHub-Repository verbunden.

Zusammengefasst

- Beim **Klonen** oder **Veröffentlichen** wird der Name automatisch übernommen.
- Bei Bedarf kannst du die Remote-URL in den Repository-Einstellungen anpassen.

Tipp:
Du kannst den Namen und die Verbindung jederzeit in den Repository-Einstellungen von GitHub Desktop überprüfen und ändern.