Git

- Git-Grundlagen: Fork vs. Branch

Git-Grundlagen: Fork vs. Branch

Um die Versionsverwaltung effizient zu nutzen, ist es wichtig, den Unterschied zwischen einem **Branch** (Zweig) und einem **Fork** (Abspaltung) zu verstehen. Beide dienen der Isolation von Code-Änderungen, setzen aber auf unterschiedlichen Ebenen an.

1. Der Branch (Der Zweig)

Ein Branch ist ein Zeiger auf einen bestimmten Entwicklungsstand innerhalb eines **einzelnen Repositorys**. Er ist die kleinste Einheit, um Änderungen getrennt vom Hauptcode (meist `main` oder `master`) zu entwickeln.

- **Ebene:** Repository-intern.
- **Lebenszyklus:** Ein Branch ist oft kurzlebig. Er wird für ein Feature oder einen Bugfix erstellt und nach Abschluss der Arbeit per *Merge* oder *Rebase* wieder in den Hauptstamm integriert.
- **Berechtigung:** Man benötigt Schreibrechte für das Repository, um einen Branch zu pushen.
- **Praxis-Beispiel:** Wenn ich in meinem Dashboard-Projekt eine neue Funktion teste, erstelle ich dafür einen Branch, um den stabilen Code nicht zu gefährden.

2. Der Fork (Die Abspaltung)

Ein Fork ist eine **vollständige Kopie** eines gesamten Repositorys unter einem neuen Besitzer-Account. Technisch gesehen ist es ein neues, eigenständiges Repository, das jedoch die Verbindung zum Original (dem „Upstream“) beibehält.

- **Ebene:** Account- / Server-Ebene.
- **Lebenszyklus:** Ein Fork ist oft langlebiger. Er wird genutzt, um unabhängig am gesamten Projekt zu arbeiten, ohne das Original zu beeinflussen.
- **Berechtigung:** Jeder kann einen Fork von einem öffentlichen Projekt erstellen, ohne Schreibrechte am Original zu besitzen.

- **Praxis-Beispiel:** Bei komplexen Experimenten (z. B. wenn K.I.-Agenten großflächig Code umbauen) nutze ich einen Fork. So kann ich das gesamte Projekt spiegeln und experimentieren, ohne mein Haupt-Repo mit unzähligen Test-Branches zu fluten.

3. Direkter Vergleich

Merkmal	Branch	Fork
Speicherort	Im selben Repository	In einem neuen, eigenen Repository
Abhängigkeit	Fest mit dem Hauptprojekt verbunden	Eigenständig (mit Link zum Original)
Zusammenführung	`git merge` oder `git rebase`	Pull Request (PR) an das Original
Sichtbarkeit	Für alle Projektbeteiligten sichtbar	In meinem eigenen Account-Bereich

4. Kombination im Workflow

In der Praxis werden beide Konzepte oft kombiniert. Ein typischer Workflow sieht so aus:

1. Man erstellt einen **Fork** eines Projekts, um eine eigene Arbeitsumgebung zu haben.
2. Innerhalb dieses Forks arbeitet man mit verschiedenen **Branches**, um einzelne Features sauber zu trennen.
3. Ist ein Feature fertig, wird es im Fork gemerged und bei Bedarf per Pull Request dem Original-Projekt zur Verfügung gestellt.