

Git

Was ist Git

- Versionskontrolle
- Verwaltung und Verfolgung von Änderungen
- Anwachsende Sammlung dient auch als
 - Repository (= Lager)
 - Projektgeschichte
 - Kommunikationsmedium
 - Werkzeug zur Team- und Produktverwaltung
- Versionskontrolle
 - zentrale Rolle
 - funktioniert am effektivsten, wenn sie auf Arbeitsgewohnheiten und -ziele des Projektteams zugeschnitten ist
- ist ein verteiltes Versionskontrollsystem (= distributed version control system)
 - jeder Entwickler kann unabhängig voneinander Änderungen vornehmen und aufzeichnen
 - die Änderungen können jederzeit miteinander kombiniert werden – ohne zentrales Repository

Versionskontrollsystem

- Entwicklung und Pflege eines Repositories mit Inhalten
- Ermöglichung des Zugriffs auf historische Daten
- Aufzeichnung aller Änderungen in einem Log

Vorteile von Git

- Leistungsstarke und flexible Versionskontrolle mit geringem Overhead
- Begünstigt gemeinschaftliches Entwickeln
- Unterstützung der verteilten Entwicklung:
 - Parallele und unabhängige Repositories ohne ständige Synchronisation mit zentralem Repository (Entwicklungsengpass)
- Verantwortlichkeit erzwingen
 - Jede Datenänderung wird geloggt
 - Wer hat die Daten verändert und warum
- Förderung der verzweigten Entwicklung
 - Eine Entwicklungslinie (Zweig) in mehrere aufteilen
 - Jeder Zweig hat einen Namen
 - Einfache Wiedervereinigung der Zweige
- GIT = Global Information Tracker

Starten

- Git Bash öffnen
- Git-weite Konfigurationen einstellen:
- `git config --global user.name "Helga Musterfrau"`
- `git config --global user.email "helga.musterfrau@mail.ch"`

[git clone https://github.com/IlkaK/StartAtLBBD.git](https://github.com/IlkaK/StartAtLBBD.git)

Warum klonen?

- Prinzip von Git: Jeder Entwickler zieht sich das komplette Repository mit seinem gesamten Verlauf und allen Branches lokal auf seinen Rechner und arbeitet daran.
 - Verteilte Entwicklung ohne Serverabhängigkeiten möglich
- In Firmen werden Bare- und Entwicklungs-Repositories verwendet
 - Es gibt eine gemeinsame Repository-Struktur für alle
 - Bare-Repository ist der Repository-Klon, der auf einem zentralen Server liegt, welcher die Entscheidungsgewalt hat
 - Entwicklungs-Repository ist ein Repository-Klon auf einem Entwicklungsrechner

Branches in Git

- in einem Repository kann es viele Branches geben, aber immer nur einer ist jeweils im lokalen Repository aktiv (der mit dem * bei git status)
- der aktive Branch bestimmt, welche Dateien ins Arbeitsverzeichnis ausgecheckt werden
- Aktueller Branch = impliziter Operand in Git-Befehlen (z.B. Ziel der Merge-Operation)

Wechseln des Branches

- Anzeigen der Branches:
git branch -a
- Wechseln in einen anderen Branch:
git checkout /feat/someBranch
- Was zeigt jetzt: git branch -a
 - * *feat/someBranch*
 - master
 - remotes/origin/master
 - remotes/origin/*feat/someBranch*

Erstellen eines Branches

- Branch erzeugen:
 - Mit Startcommit => `git branch neuerBranchName startCommit`
 - Ohne Startcommit => `git branch neuerBranchName`
- standardmässig wird als Start-Commit die Revision benutzt, die zuletzt auf dem aktuellen Zweig durch ein Commit bestätigt wurde
- «git branch» führt nur den neuen Branchnamen ein, ändert nicht das Arbeitsverzeichnis und auch sonst nichts
- um dort zu arbeiten, muss erst in den Branch gewechselt werden

Git – Index und Commit im lokalen Repository

- Git trennt mit Index einzelne Entwicklungsschritte, welche mit Commits bestätigt werden
- Index
 - flüchtige Infos, die nur zu einem lokalen Repository gehört
 - beschreibt die Verzeichnisstruktur des gesamten Repositories
 - erfasst eine Version der Gesamtstruktur des Projekts zu einem beliebigen Zeitpunkt (z.B. Zustand, der gerade entwickelt wird)
- Commit
 - zeichnet einen Schnappschuss des Index auf und legt diesen Schnappschuss ab
 - Schnappschuss enthält nur die Änderungen zum vorherigen Schnappschuss
 - ⇒ Liste mit betroffenen Dateien und Verzeichnissen
 - ⇒ neue Blobs für alle Dateien, die sich verändert haben
 - ⇒ neue Bäume für alle Verzeichnisse, die sich verändert haben
 - ⇒ alle Blobs und Baumobjekte, die sich nicht verändert haben, werden wiederverwendet
 - Commit-Schnappschüsse werden miteinander verkettet, wobei jeder Schnappschuss auf seinen Vorgänger zeigt
 - ⇒ Folgen aus Änderungen = Reihe aus Commits

Git – Index und Commit im lokalen Repository

- git add
 - git add meineAlteOderNeueDatei
 - Fügt meineAlterOderNeueDatei zum Index hinzu, beim nächsten Commit wird diese mitgenommen
 - git add *
 - Hinzufügen aller Dateien, die nicht standardmässig ignoriert werden (.gitignore) zum Index, beim nächsten Commit werden alle mitgenommen
- git commit -m «die erste Commit-Nachricht»
 - Comitten der Änderungen, die im Index aufgenommen wurden, der Index ist danach wieder geleert
- Änderungen vom Index und Commit im lokalen Repository können mit git status verfolgt werden

Pull vom Server

- `git fetch`
 - holt Objekte und Metadaten von entferntem Repository
- `git pull`
 - wie `git fetch`, überführt aber auch Änderungen in den Zweigen nach (nach `git fetch` wird sonst `git merge` oder `git rebase` ausgeführt)

Versionskontrollsystem

- Entwicklung und Pflege eines Repositories mit Inhalten
- Ermöglichung des Zugriffs auf historische Daten
- Aufzeichnung aller Änderungen in einem Log

Git

- Leistungsstarke und flexible Versionskontrolle mit geringem Overhead
- Begünstigt gemeinschaftliches Entwickeln
- Unterstützung der verteilten Entwicklung:
 - Parallele und unabhängige Repositories ohne ständige Synchronisation mit zentralem Repository (Entwicklungsengpass)
- Verantwortlichkeit erzwingen
 - Jede Datenänderung wird geloggt
 - Wer hat die Daten verändert und warum
- Förderung der verzweigten Entwicklung
 - Eine Entwicklungslinie (Zweig) in mehrere aufteilen
 - Jeder Zweig hat einen Namen
 - Einfache Wiedervereinigung der Zweige
- GIT = Global Information Tracker