Einführung in Git

Praxislabor Digitale Geisteswissenschaften

Dr. David Krassnig

Universitätsbibliothek Frankfurt

14. Dezember 2023

Übersicht

Vorstellungsrunde

- Zu meiner Person:
  - **Name:** David Krassnig
  - **Beruf:** Bibliotheksreferendar
  - **Fachrichtung:** Linguistik / Digital Humanities / Library Science
  - **Git-Erfahrung:** Dissertation, Masterarbeit, Skripte, Webseite etc.
- Zu eurer Person:
  - Fachlicher Hintergrund?
  - Betriebssystem?
  - Git bereits erfolgreich installiert?
  - Erfahrungen mit Bash/Shell?
  - Erfahrungen mit Git?

Hintergrund - Was ist Git?

- Software zur Versionsverwaltung von Dateien
- Hauptziel: Versionskontrolle von Code
- Erfunden von Linus Torvalds
- Open Source
- Betriebssystemübergreifend
- Erlaubt kollaboratives Arbeiten

Hintergrund - Was ist GitHub/GitLab?

- Git: Versionskontrollsoftware
- GitHub/GitLab: Server, auf denen Git-Projekte gehostet werden können
  - Lokales Git-Projekt wird mit remotem Git-Projekt verknüpft
  - Synchronisierung zwischen lokalem und remotem Repositorium
  - Ermöglicht kollaboratives Arbeiten auf Distanz
  - Bietet einige Zusatzfunktionen (z.B. Bug-Reports / Issues)
- GitHub: Microsoft Server
- GitLab: Open Source Software für eigene Server

Hintergrund - Warum Git?

- Arbeit ohne Git
  - Versionierung durch multiple Dateien
  - Organisation via Ordner
  - Unklare Versionierung
  - Unklare Quelle bei Kollaboration
  - Aufwendiger BackUp bei Versionierung
- Arbeit mit Git
  - Versionierung ohne multiple Dateien
  - Eindeutige und bedeutungsvolle Versionierung
  - Eindeutige Zuordnung von Beiträgen
  - Robuster BackUp via Remote-Repositorium
  - Unterschiedsanzeige zwischen zwei Versionen

Hintergrund - Warum Git?

- Einige Nachteile
  - Steile Anfangslernkurve
  - Verwirrung durch mehr vorhandene Funktionen als Nutzer benötigt
  - (Sinnvolle) Versionskontrolle eingeschränkt auf Klartext-Dateien
  - Goethe-Universität betreibt keinen eigenen Git-Server[1]
- Nachteile neutralisierbar
  - Git-Workshops/-Lernkurse
  - Selbst-Einschränkung auf eine Reihe an Standard-Befehlen
  - Nutzung passender Dateiformate
  - Kostenlose GitLab/GitHub Accounts[2]

[1] Zumindest keinen Git-Server, der allen Studierenden für alle studentischen Zwecke offen steht.

[2] Für Studierende gibt es auch *GitHub Pro* kostenlos: https://education.github.com/

Anwendungsfelder - Möglichkeiten

- Organisation eigener Projekte
  - Haus-/Abschlussarbeiten
  - Verwaltung tabellarischer Daten (z.B. Literaturlisten)
  - Referate/Präsentationen
  - Codepflege (z.B. XML, Skripte etc.)
- Kollaboratives Arbeiten (Sammelbände, Lexika, Publikationen etc.)
- Große Datenbestände verwalten
- Akademische Webseiten via GitHub Pages

Anwendungsfelder - Umsetzung

- Digitale Notizbücher auf Markdown-Basis (z.B. Joplin)
- Verwendung von Klartext-Formaten
  - Entspricht den Standards guter wissenschaftlicher Praxis
  - Langfristig nachnutzbar
  - Ermöglicht Inhaltsvergleich via Git
  - LaTeX, Markdown, Plaintext, CSV-Tabellen, SVG-Dateien etc.
- Dokumentation für sich selbst und andere
- Nutzung von Repositoria für kollaboratives Arbeiten
- Sinnvolle sowie nachvollziehbare Datei- und Ordnernamen

Anwendungsfelder - Umsetzung

- Inhaltsvergleich zwischen Versionen möglich
  - Textdateien (LaTeX, Markdown, XML/TEI, Plaintext etc.)
  - Daten (CSV, TSV etc.)
  - Quellcode (Python, HTML etc.)
  - Einfacher Test: Lässt sich das Dokument via Texteditor anzeigen?
- Kein Inhaltsvergleich zwischen Versionen möglich
  - Proprietäre Formate (DOCX, XSLX etc.)
  - Gepackte Dateien (ZIP, RAR, 7Z, GZ etc.)
  - Kompilierter Code / Binärdateien (EXE-Dateien etc.)
  - Nicht-Vektorbasierte Medien (MP4, JPG, MP3 etc.)

Erste Schritte mit Git - Konfiguration

- Nach Installation: Festlegung eurer Identität
  - Benutzername:
    ```
    $ git config --global user.name "EuerName"
    ```
  - E-Mail-Adresse:
    ```
    $ git config --global user.email "eure@email.de"
    ```
  - Standard Branch-Name:
    ```
    $ git config --global init.defaultBranch main
    ```

Erste Schritte mit Git - Funktionsweise

- Alle Git-Projekte basieren auf den folgenden Schritten:
  - Dateien erstellen / bearbeiten
  - Änderungen betrachten / kontrollieren
  - Änderungen für die nächste Abgabe freigeben / zuordnen (*Staging*)
  - Änderungen abgeben (*Commit*)
- Zusätzliche Anfangs-/Endschritte bei Nutzung eines Remote-Repositoriums
  - Änderungen aus dem Remote-Repositorium holen (*Pull*)
  - Änderungen in das Remote-Repositorium übertragen (*Push*)

Erste Schritte mit Git - Funktionsweise

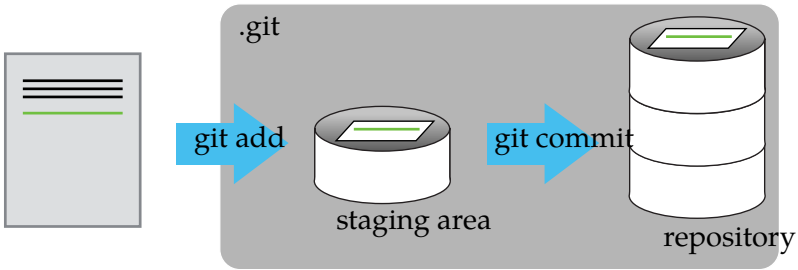


Abbildung: Git-Abgabeprozess

Erste Schritte mit Git - Initialisierung

- Erstellen wir unser erstes Projekt
  - Neuen Ordner erstellen:
    ```
    $ mkdir "erstesProjekt"
    ```
  - In den neuen Ordner wechseln:
    ```
    $ cd "erstesProjekt"
    ```
  - Git-Projekt initialisieren:
    ```
    $ git init
    ```
  - Dateien anzeigen lassen:
    ```
    $ ls -la
    ```
  - Git-Status abfragen:
    ```
    $ git status
    ```

Erste Schritte mit Git - Initialisierung

- Erstellen wir unser erstes Projekt
  - Neue Datei erstellen:
    ```
    $ touch README.md
    ```
  - Neue Datei modifizieren:
    ```
    $ nano README.md
    ```
  - Änderungen zum nächsten Commit zuordnen:
    ```
    $ git add README.md
    ```
  - Commit erzeugen:
    ```
    $ git commit -m "Projektbeschreibung hinzugefügt"
    ```
  - Git-Protokoll abfragen:
    ```
    $ git log
    ```

Erste Schritte mit Git - Dateien zurücksetzen

- Wir können jederzeit Änderungen rückgängig machen:
  - Datei modifizieren:
    ```
    $ nano README.md
    ```
  - Änderungen zum nächsten Commit zuordnen:
    ```
    $ git add README.md
    ```
  - Es sich anders überlegen:
    ```
    $ git reset README.md
    ```
  - Datei zurücksetzen:
    ```
    $ git restore README.md
    ```

Erste Schritte mit Git - Dateien zurücksetzen

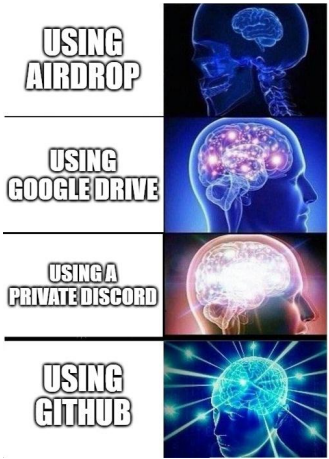
- Oder auch Dateien zu frühere Versionen zurücksetzen:
  - Datei modifizieren:
    ```
    $ nano README.md
    ```
  - Änderungen zum nächsten Commit zuordnen:
    ```
    $ git add README.md
    ```
  - Commit erzeugen:
    ```
    $ git commit -m "Projektbeschreibung erweitert"
    ```
  - Git-Protokoll abfragen und gewünschte ID kopieren:
    ```
    $ git log
    ```
  - Datei zurücksetzen:
    ```
    $ git checkout <ID> README.md
    ```

Rück-/Vorschau

- Was haben wir gemacht?
  - Besprochen was Git ist
  - Beleuchtet wofür Git in den Geisteswissenschaften nutzbar ist
  - Rudimentäre Funktionsweise von Git angeschaut
  - Erstes eigenes Projekt gestartet
  - Dateien geändert, gestaged, committet und zurückgesetzt
- Was machen wir nächstes Mal?
  - Remote Repositorium hinzufügen (GitHub)
  - SSH-Authentifizierung und -verifizierung
  - Branches einführen
  - Kollaborativ arbeiten

Bonus Fun Fact - Wofür steht Git?

- Linus Torvald hat 4 offizielle Erklärungen, je nach Laune des Anwenders:
  - **G**lobal **I**nformation **T**racker
  - Eine zufällige, unverbrauchte und aussprechbare 3-Buchstaben-Kombination
  - **Git** (engl. für *Schwachkopf*)
  - **G**odd*mn **I*****tic **T**ruckload of S**t



Vielen Dank für Eure Aufmerksamkeit!