Datenversionierung mit Git

Advanced Track

RDA Deutschland Tagung 2020

Dr. Carolin Odebrecht, Forschungsdatenmanagement
Sprach- und literaturwissenschaftliche Fakultät | Computer- und Medienservice |
Humboldt-Universität zu Berlin

25.2.2020 – Potsdam

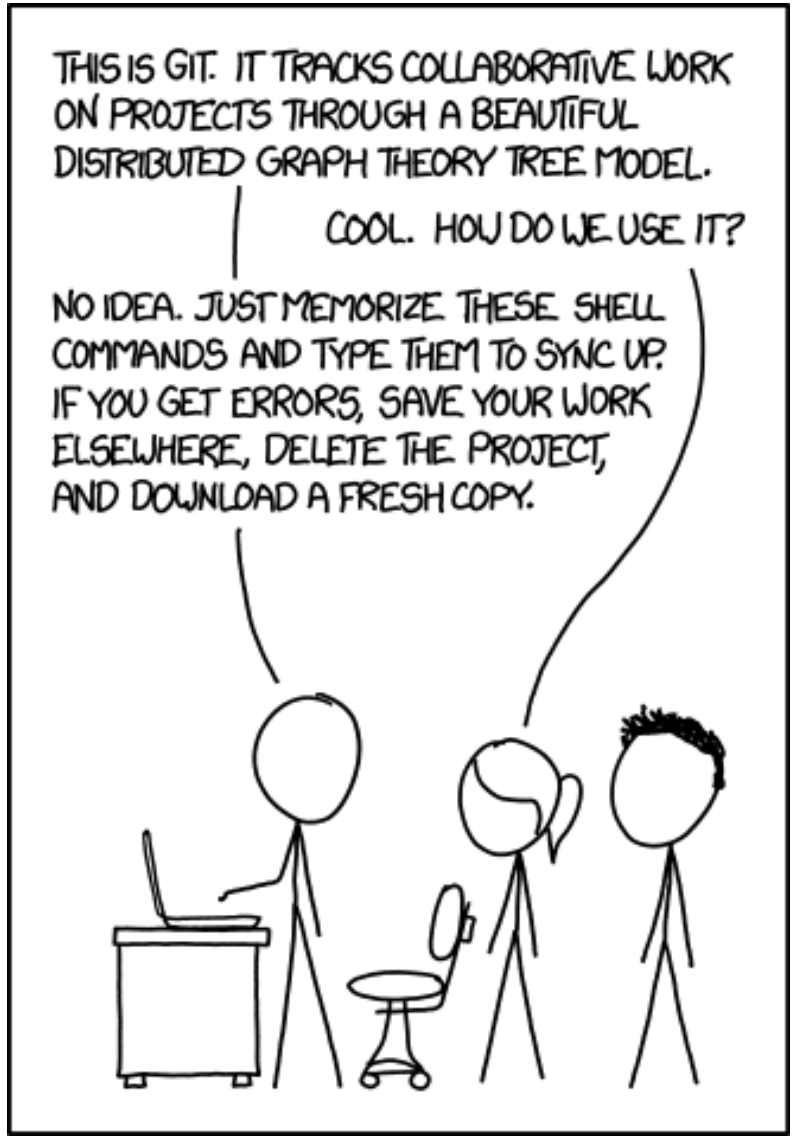


Abbildung: Comic aus xkcd https://xkcd.com/1597/ (besucht 11.01.20202).

Plan

Forschungskontext

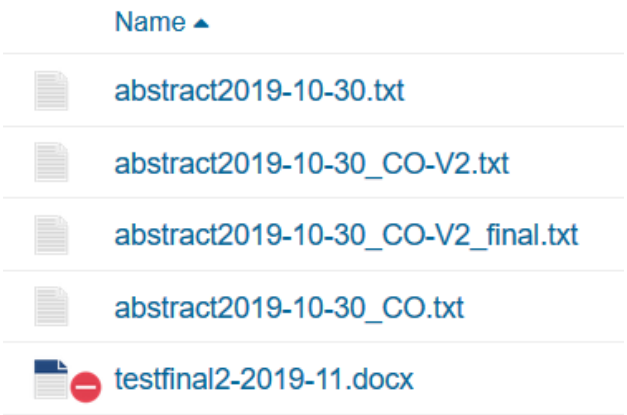
Bisherige Arbeitsweise

Mit welchen Forschungsdaten oder Forschungssoftware haben Sie bereits gearbeitet?
Haben Sie selbst bereits Forschungsdaten oder Forschungssoftware erstellt?
Wie arbeiten Sie? Allein, in einem oder mehreren Teams? Mit spezieller Software?

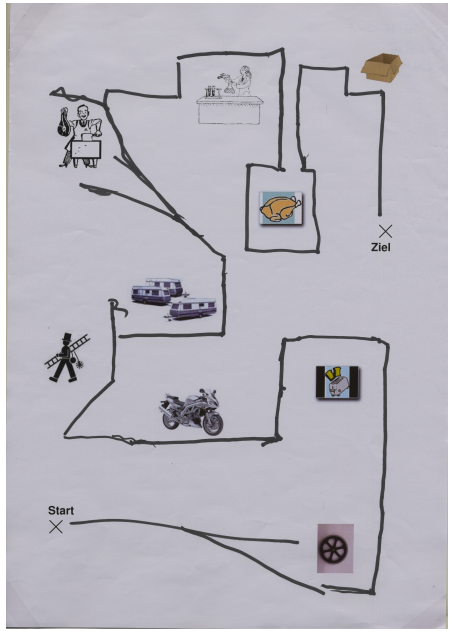
Forschungsdatenmanagement

„Research data management is an explicit process covering the creation and stewardship of research materials to enable their use for as long as they retain value.“
Whyte, A. and Rans, J. this glossary,
http://www.dcc.ac.uk/digital-curation/glossary#R
(besucht 03.06.2019)

Warum Forschungsdatenmanagement?



(a) Ordnung, Struktur und Versionierung



(b) Planung und Dokumentation

(c) Zugang, Referenzierung und Wiederverwendung

(d) Anforderungen von Förderern und Universitäten

Abb (a) Eigener Screenshot CC0 International| Abb.(b) Open Access CC-BY-SA 4.0 International | Abb.(c) BeMaTac Map Karten CC-BY 3.0 International | Abb.(d) DFG Logo (alle Referenzen besucht am 01.11.2019).

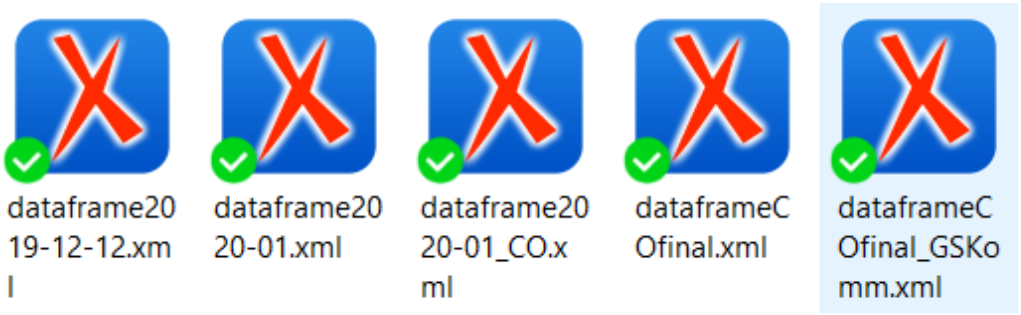
Forschungsdatenmanagement für wen?

- Projekte
- Arbeitsgruppen
- Seminare
- Forschung im Rahmen von Promotionen und anderen Abschlussarbeiten
- Eigene, individuelle Forschung

→ Forschungsdatenmanagement ist erforderlich, sobald mit Daten geforscht wird! Gilt auch für Software.

Plan

Ziel unseres Workshops – Beobachtung



Dateien irgendwo in unseren Systemen, Festplatten oder Clouds . . .

Ziel unseres Workshops – Motivation

„Wo liegt jetzt der wirklich aktuelle und finale Bearbeitungsstand unserer Daten?“

Hintergrund und Ursache

Welche Aspekte des Datenmanagements sind in dieser Frage direkt oder indirekt angesprochen?

Ziel unseres Workshops – Motivation

„Wo liegt jetzt der wirklich aktuelle und finale Bearbeitungsstand unserer Daten?“

Hintergrund und Ursache

Welche Aspekte des Datenmanagements sind in dieser Frage direkt oder indirekt angesprochen?

Häufige Fragen

- Wie können wir gemeinsam an Daten oder Software arbeiten?
- Wie stellen wir sicher, dass keine Änderungen stillschweigend überschrieben wird?
- Ist es möglich, dass jede Änderung nachvollziehbar ist?
- Wie können wir sicher feststellen, welcher Bearbeitungsstand der aktuelle ist?
- Wie können wir parallel zur Bearbeitung eine Dokumentation und Aufgabenorganisation erstellen?

Plan

Version

„Version: a variations of an object with a high degree of similarity. Document versions are never completely equal, but they are similar enough so as to be able to recognize them as the same document. (Martínez-González, 2009)“
(Khosrowpour 2013, S. 943)

Versionskontrolle

„Version Control: a set of mechanisms that support object evolution in computer applications (Martínez-González, 2009)“ (Khosrowpour 2013, S. 943)

Versionierungsystem

„Versioning System: an automated system that allows remote access to source code and enables multiple developers to work on the same version of the source code simultaneously. Examples are CVS, SVN, and BitKeeper. (Kosonen, Henttonen, and Blomqvist, 2008a)“
(Khosrowpour 2013, S. 943)

Software

- Distributed Version Control System (VCS)
  - Git https://git-scm.com/
  - Mercurial https://www.mercurial-scm.org/
  - ...
- Online hosting services
  - GitHub https://github.com/
  - GitLab https://about.gitlab.com/
  - ...
- Clients
  - SmartGit https://www.syntevo.com/smartgit/
  - TortoiseGit https://tortoisegit.org/
  - ...

(alle Links besucht am 12.01.2020)

Git - distributed VCS

- Entwicklung seit 2005 von Linux-Community (Linus Torvald)
- Für die Entwicklung von Software
- Spezialisiert für nicht lineare Entwicklungen
  - Auch für die Entwicklung von Forschungsdaten geeignet
- Frei verfügbar und nutzbar; GNU General Public License, Version 2
- Name *Git* bedeutet so viel wie „Blödmann/Idiot/Depp“

GitLab

- Online-Dienst für Git inklusive u.a.
  - Interface
  - Nutzer- und Rollenverwaltung
  - Issuetracker
  - Wiki-Funktion
- Service des CMS `https://scm.cms.hu-berlin.de`
  - LDAP-Anbindung (HU-Accounts funktionieren)
  - Registrierung von externen Nutzer*innen

Version Control

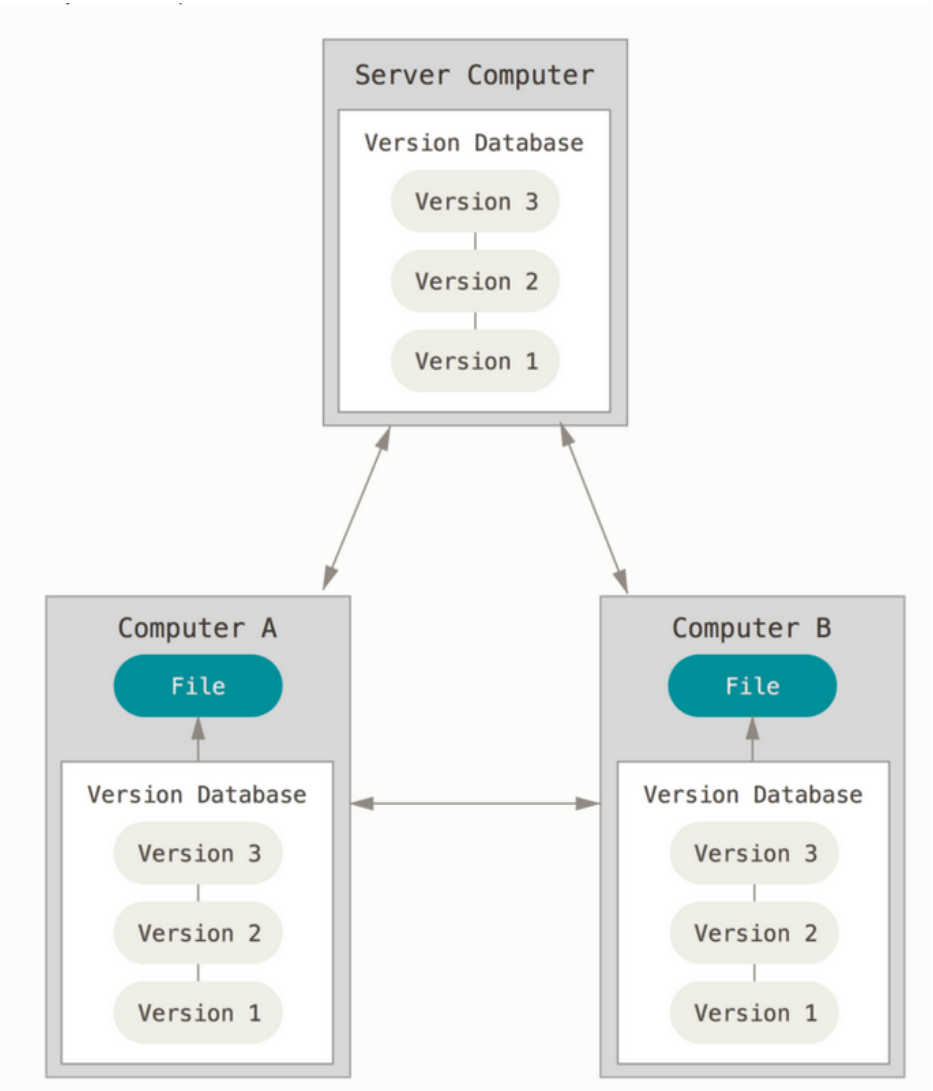


Abbildung: *Remote*: Version der Dateien in einem Repository auf einem Remoteserver (z. B. GitLab). *Local*: Version auf eigenem Rechner. Abbildung 3 aus Git User Guide besucht am 07.01.2020.

Git – Was es tut (ganz grundlegend)

- Daten als Stream von Snapshots
- Arbeitshistorie (über die Erstellung dieser Snapshots)
- Vergleich der verschiedenen Snapshots
- Integrität durch Log aller Veränderungen der Daten
- Verzweigende Entwicklung im Team unterstützt

Konzept

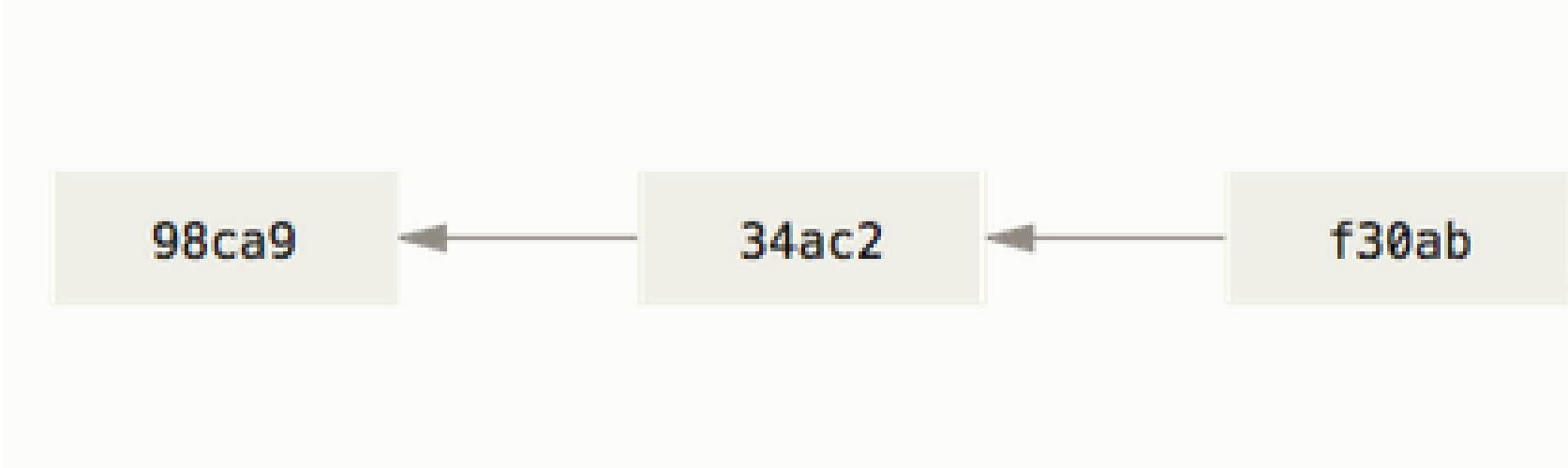


Abbildung: Jeder Snapshot einer Datei wird mit einer Hashnummer versehen und hat einen Vorfahren (außer initiale Erstellung) und einen Nachfahren (außer finale Bearbeitung). *Commits* sind gesichterte Snapshots. Modifizierte Abbildung 17 Git User Guide besucht am 07.01.2020.

Verzweigung von Entwicklungen

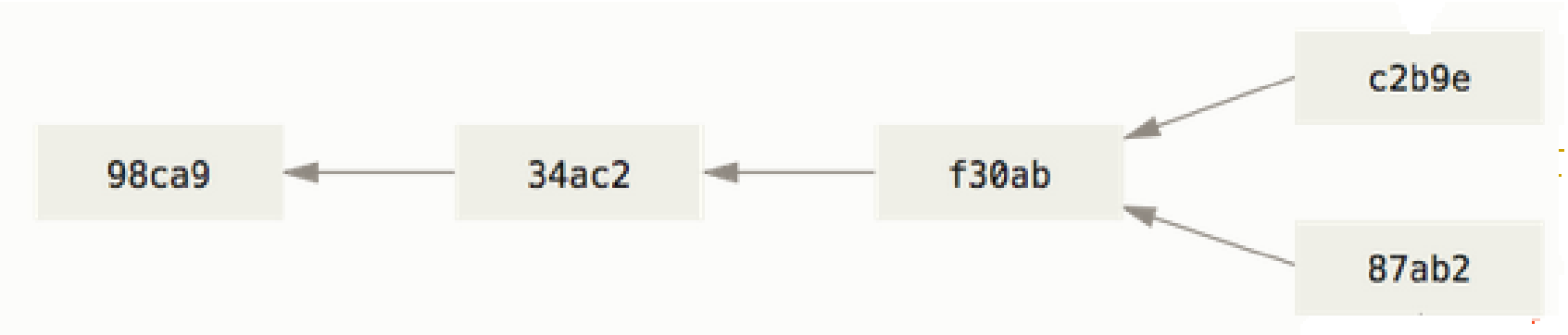


Abbildung: Bearbeitung kann verzweigt erfolgen und ein Snapshot hat dann zwei Nachfahren. Damit erstellt man einen *Branch*, also eine parallele Version von Dateien. Modifizierte Abbildung 17 Git User Guide besucht am 07.01.2020.

Verzweigung von Entwicklungen

Abbildung: Zweige werden benannt. *Head* gibt an, auf welchem Zweig man gerade ist. Abbildung 17 Git User Guide besucht am 07.01.2020.

Merging von Entwicklungszweigen

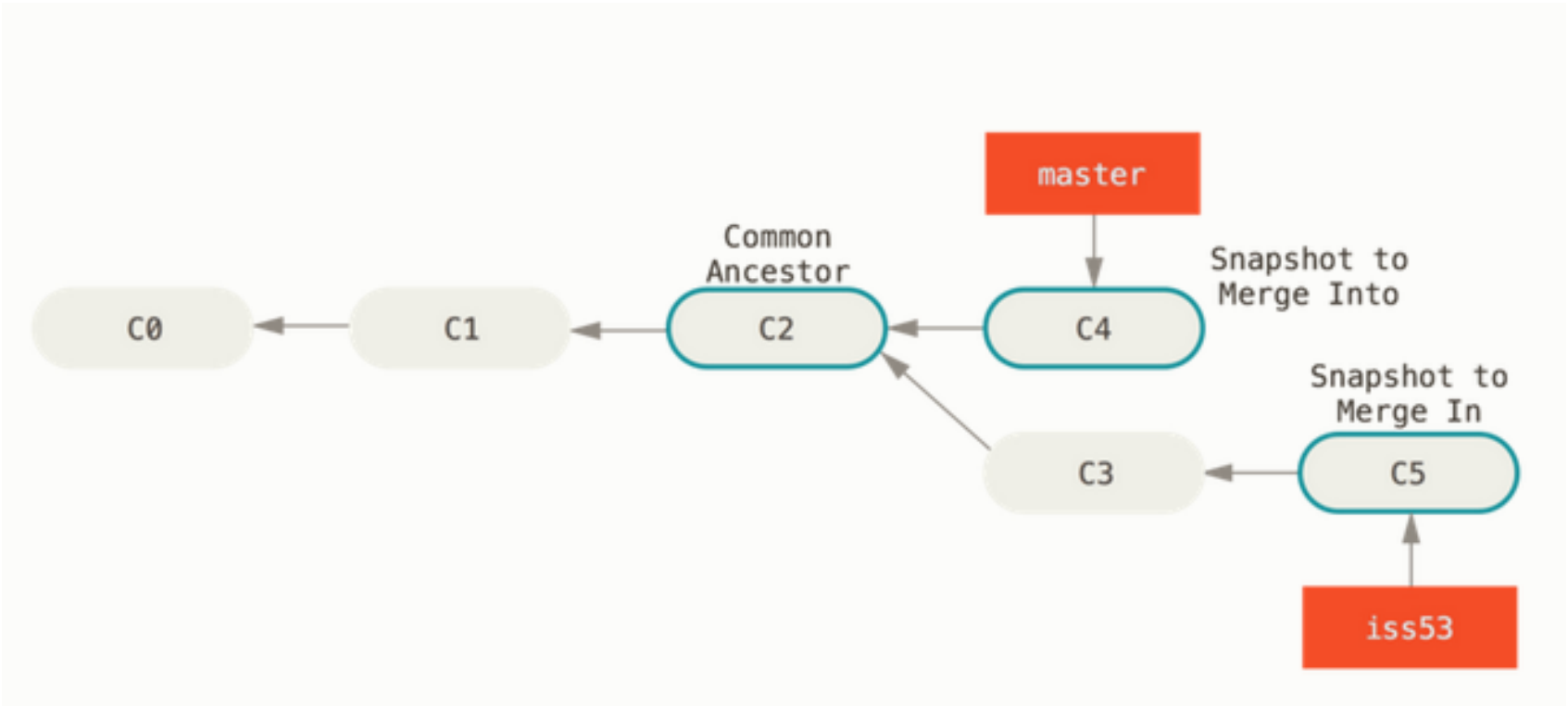


Abbildung: *Merging*: Zusammenführen von zwei Zweigen zu einem Zweig. Entscheidung, welcher Snapshot in welchen anderen Snapshot integriert wird. Abbildung 24 Git User Guide besucht am 07.01.2020.

Merging von Entwicklungszweigen

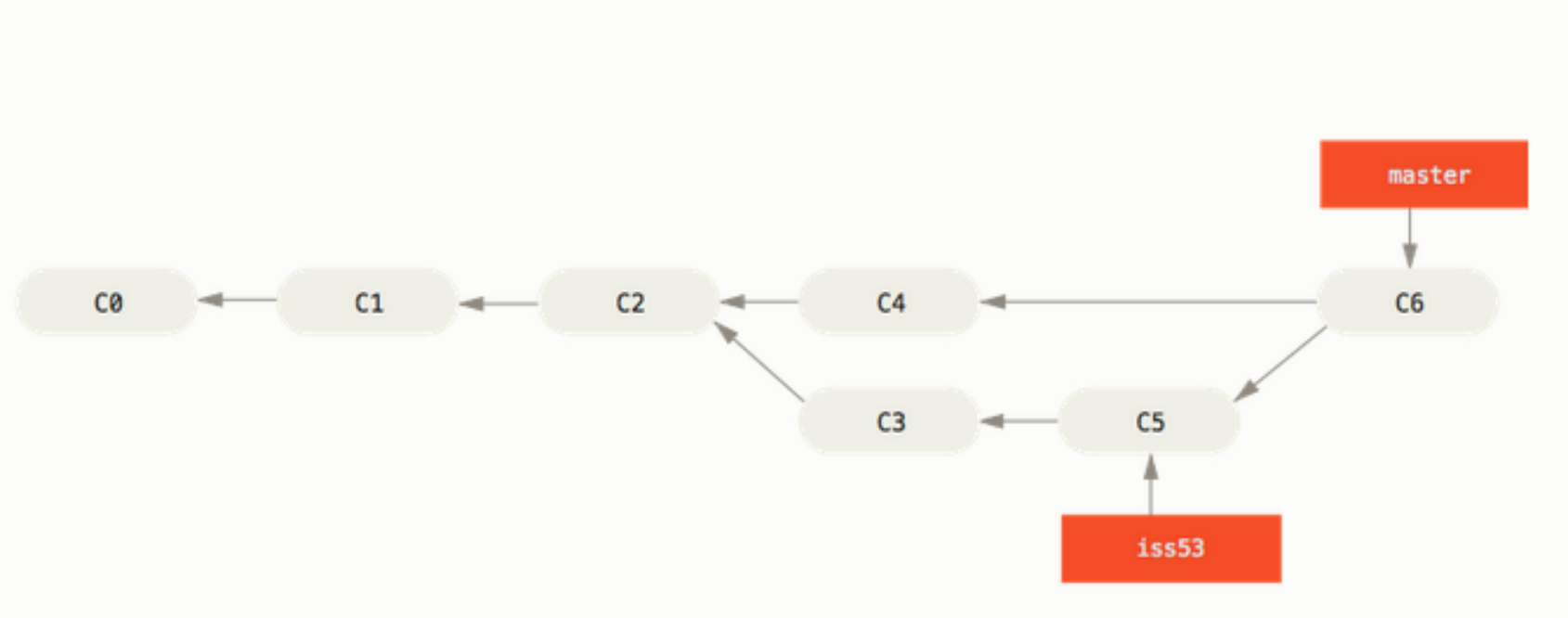


Abbildung: Abbildung 25 Git User Guide besucht am 07.01.2020.

Conflicts

- Git deckt Konflikte auf bevor man Datenstände zusammenführt/überschreibt
  - Dateistände können unterschiedliche, sich widersprechende Änderungen an derselben Stelle
  - Status- und Differenzsicht zum Lösen der Konflikte

Tagging

- Besondere Markierung eines bestimmten Punktes in der Historie (Commit)
- Häufig Markierung einer Veröffentlichung (V1, V2 usw.)
- Git tagging: lightweight und annotated
  - lightweight: Pointer zu einem spezifischen Commit
  - annotated: mit Beschreibung, BearbeiterIn, Bearbeitungsmitteilung
- Getagged werden kann jeder Commit in der Historie!

Plan

Aufgabenkontext

Webdienst GitLab an der HU

Wie arbeiten ausschließlich webbasiert.

https://scm.cms.hu-berlin.de/users/sign_in

Set up

Projekt

Sie sind in ein Projekt in GitLab eingeladen.
Explorieren Sie, was Sie sehen!

Inhalte

Arbeiten im Team

Wir können die folgenden Dateien bearbeiten: *postcard.xml*, *slides.tex* und *searchReplace.py*. Pro Datei ein Team.

Verändern

Bearbeitung

Suchen Sie sich eine Datei aus und verändern Sie diese in irgendeiner Weise mit einer direkte Bearbeitung im GitLab-Editor. Schreiben Sie eine Commit-Nachricht.

Verändern

Bearbeitung

Wiederholen Sie die letzten beiden Schritte noch einmal mit derselben Datei.

Richtlinien

Lizenz

Erstellen Sie eine LICENCE Datei in jedem Ordner. Wählen Sie dabei eine konkrete Lizenz aus.

Historie

Log

Schauen Sie sich die Übersicht in Ihren Commits und im Graph an.

Branch

Verzweigen

Erstellen Sie einen Branch (inkl. Benennung).
Wechseln Sie zu diesem Branch und verändern wieder eine Datei (inkl. Commit).

Merge

Mergerequest

Erstellen Sie einen Mergerequest.

Historie II

Log

Schauen Sie sich die Übersicht in Ihren Commits und im Graph an.

Plan

Team Building

Wer darf was? – Nutzerrollen (grob zusammengefasst)

- Gast: Anschauen
- Reporter: Commitstatus, Issue-Verwaltung
- Developer: Branches, pushes, merge-Anfragen, tagging
- Maintainer: Switch visibility level, manage projects and issues
- Owner bei Gruppen: Gruppenmanagement
  vgl. Übersicht GitLab-Nutzerrollen

Issuetracker

Mögliche Verwendung:

- Formulieren von Fragen und Aufgaben
- Zuweisung an Teammitglieder
- Arbeitsaufwand, Beschreibungen und zusätzliche Materialien (Übersichten, Screenshots etc.)
- Feedback und Nutzerumfragen
- Terminierung und Planung in Boards

Boards

Mögliche Verwendung:

- Planung von Sprints (z. B. für ein Release – Tagging!)
- Welche Aufgabenpakte (Auswahl von Issues) sollen bis wann erledigt sein?
- Wer macht was?

Wiki

- Dokumentation von Dateien und Workflows
- Für User und Entwicker*innen
- In Markdown geschrieben

Plan

Ordnen und Strukturierung

Grundlegendes für (Meta-)Daten und Software:

- Dateibenennung: ohne Sonderzeichen und Leerzeichen, sinnvolle Benennung
- Dateinamen „nicht einfach immer wieder“ umbenennen - Metadaten (Historie) können verloren gehen
- Ordnerstrukturen: angepasst an Workflows und Akteure
- README sinnvoll bis notwendig
- LICENCE notwendig

Eigener Workflow und Git

- Schritte bei der Erstellung, Bearbeitung, Analyse und Visualisierung der Forschungsdaten oder Software
- Zusammenhang zwischen Forschungsdaten, Software und Akteuren
- Verschiedene Abhängigkeiten und Reihenfolgen

FDM Abstrakt: Lebenszyklus von Forschungsdaten (vgl. z. B. Digital Curation Centre 2010)

→ Je nach Kontext unterschiedliche Repositorienstruktur, Zyklen von Commits, Branches, Tags notwendig/sinnvoll.

→ GitLab nimmt diese Organisationsaufgaben nicht ab, unterstützt diese aber enorm!

To git or not to git?!

- Git ideal für textbasierte Dateien wie code/scripts oder LaTeX-Dateien (Diffs möglich, nur Änderungen werden gespeichert)
- Alles andere wird auch versioniert (ggf. speicherintensiv)
- Für eigenes Arbeiten oder die Arbeit im Team (lokal oder verteilt)
- Integration von Kommunikations- und Organisationsfunktionen extrem wertvoll für Projektarbeiten
- Unterstützung für abzweigende Entwicklungen
- Für gemeinsame Dokumenterstellung zu klassischen Publikationen z. B. Dienste wie HU-Box und Only Office ideal (https://box.hu-berlin.de/accounts/login/?next=/)

Workshop

- Motivation und Zielsetzungen für verteiltes Arbeiten
- Einführung in die ersten Grundlagen
- Overhead: Einarbeitung und Problembehandlung wie bei jeder anderen Anwendung auch
- Wesentlich: Team- und Workflowmanagement zusätzlich immer notwendig
- Noch vieles nicht angesprochen, u.a.:
  - Arbeit mit Client: Staging, tracking
  - Branchmanagement
  - Analysetools
  - Clients
  - Wiki, Boards, Issuetracker nur am Rande

Langfristige Verfügbarkeit und Datensicherung

Version Control System ungleich Repositorien im Sinne der Archivierung und Publikation:

- Sicherstellung der Aufrechterhaltung muss personenunabhängig erfolgen
- Häufig institutionell verankert (Rechenzentren, Bibliotheken und Forschungsdatenzentren)
- Regelungen für den Zugriff, Sichtbarkeit/Publikation, persistente Referenzierung

$\rightarrow$ Aufgaben von Repositorien

Dokumentationen

- Git https://git-scm.com/book/en/v2/
- Gitlab HU https://pages.cms.hu-berlin.de/cms-webtech/gitlab-documentation/docs/home/
- Dokumentationen bei anderen Anbietern wie GitHub ebenfalls hilfreich
  - z. B. auch Videotutorial wie https://www.youtube.com/channel/UCP7RrmoueENv9TZts3HXXtw
- Foren wie https://stackoverflow.com sehr nützlich

Feedback und offene Diskussion

Ich freue mich über weitere Fragen, Diskussionen und Feedback!

Herzlichen Dank!

Dr. Carolin Odebrecht
Humboldt-Universität zu Berlin
Sprach- und literaturwissenschaftliche Fakultät
Computer- und Medienservice
Forschungsdatenmanagement
carolin.odebrecht@hu-berlin.de

Referenzen I

Digital Curation Centre (2010). *DCC Curation Lifecycle Model*. URL: http://www.dcc.ac.uk/resources/curation-lifecycle-model (besucht am 29. 02. 2016).

Khosrowpour, Mehdi (2013). *Dictionary of information science and technology*. Hershey, Pa: IGI Global (701 E. Chocolate Avenue Hershey Pennsylvania 17033 USA). DOI: 10.4018/978-1-4666-2624-9. URL: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-2624-9.