

# Git

## Fast Version Control System

Michael Kuhn  
michael.kuhn@informatik.uni-hamburg.de

Arbeitsbereich Wissenschaftliches Rechnen  
Fachbereich Informatik  
Universität Hamburg

2011-09-28

- 1** Einführung
  - Überblick

- 2 Git

- 3 Interna

- 4 Workflows

- 5 Referenzen

- Versionierung/Protokollierung aller Änderungen
  - Erlaubt es auch zu früheren Zuständen zurückzukehren
- Gleichzeitige Entwicklung durch mehrere Personen
  - Koordinierung des gemeinsamen Zugriffs
  - Aber auch eigenständige Projektzweige
- Jeder Benutzer interagiert mit einer Arbeitskopie
  - Änderungen in der Arbeitskopie werden in ein Repository übertragen

- Zentrale Versionsverwaltung
  - Das Repository liegt auf einem zentralen Server
  - Jeder Benutzer hat eine lokale Arbeitskopie
- Dezentrale Versionsverwaltung
  - Jeder Benutzer hat sein eigenes Repository
  - Üblicherweise ein „Master-Repository“

1 Einführung

2 Git

- Einführung
- Unterschiede

3 Interna

4 Workflows

5 Referenzen

- Von Linus Torvalds für die Linux-Verwaltung entwickelt
  - Kann auch mit großen Projekten umgehen
- Verteilt
  - Jeder Benutzer hat ein eigenes vollständiges Repository
    - Komplette Historie etc.
    - Kein Netzwerkzugriff notwendig
- Schnell
  - „I merge 22,000 files several times a day, and I get unhappy if a merge takes more than 5 seconds [...]“
- Speicherplatzeffizient
  - Eine Linux-Arbeitskopie hat 454 MB
  - 5,5 Jahre Historie haben 398 MB
- Die üblichen Funktionen ...
  - Ignorieren von Dateien/Mustern, Hooks etc.

- Drei Hauptbestandteile
  - Arbeitskopie (Projekt)
  - Index (Projekt/.git/index)
  - Repository (Projekt/.git)
- Änderungen in der **Arbeitskopie** werden zuerst in den **Index** übertragen
  - `git add`
- Änderungen im **Index** werden danach in das **Repository** übertragen
  - `git commit`
- D. h. es können auch nur einzelne Änderungen aus der Arbeitskopie in das Repository übernommen werden

- Benutzer arbeiten mit ihrem lokalen Repository
  - Das lokale Repository wird dann in ein Remote-Repository gepusht
- Es existieren verschiedene Transportprotokolle
  - Nativ
    - Via `git-daemon`
    - Größtenteils für anonymen Lesezugriff
  - SSH
    - Um in das eigene Remote-Repository zu pushen
  - HTTP(s)
    - Durch das neue Smart-HTTP-Protokoll inzwischen brauchbar
    - Benötigt keinen separaten Git-Daemon, sondern nur einen Webserver
    - Erlaubt zum Beispiel eine nahtlose Integration in Redmine
  - FTP(s), rsync



- 1 Einführung
- 2 Git
- 3 Interna**
  - Repository-Format
- 4 Workflows
- 5 Referenzen

- Git speichert unveränderliche Objekte
  - Blobs (Dateien)
  - Bäume (Verzeichnisse)
  - Commits
  - Tags
- Objekte werden durch ihre SHA-1-Hashes identifiziert
- Hierarchie
  - Tags referenzieren Commits
  - Commits referenzieren Bäume
  - Bäume referenzieren Blobs und Bäume

- `http://eagain.net/articles/  
git-for-computer-scientists/`

1 Einführung

2 Git

3 Interna

**4 Workflows**

■ Allgemeines

■ Branching & Merging

5 Referenzen

- Öffentliche Commits sollten **niemals** geändert werden
  - Zum Beispiel sollten keine Commits mit `git reset` „gelöscht“ werden
    - Die Commits existieren weiterhin in geklonten Repositories
  - Stattdessen sollte `git revert` benutzt werden
- Kleine Commits erstellen
  - Macht es einfacher Fehler mit `git bisect` zu finden
  - Können vor einem Push jederzeit noch mit `git rebase -i` kombiniert werden
- Korrekte Formatierung der Commit-Messages
  - Die erste Zeile enthält den Betreff
  - Die zweite Zeile ist leer
  - Ab der dritten Zeile folgt die eigentliche Nachricht
  - Manche Werkzeuge setzen dieses Format voraus

- Topic-Branches benutzen
  - Zum Beispiel um neue Funktionalität einzubauen oder Fehler zu beheben
  - Branches sind billig und stören andere nicht
- Entwickler pullen lassen ...
  - Eigene Änderungen in ein Remote-Repository pushen
  - Den Entwicklern Bescheid geben (`git request-pull`)
  - Darauf achten, dass möglichst keine Konflikte entstehen
- ... alternativ Patches per E-Mail schicken (`git format-patch`, `git send-email`)

1 Einführung

2 Git

3 Interna

4 Workflows

**5 Referenzen**

## ■ Git-Webseite und -Dokumentation

- <http://git-scm.com/>
- <http://git-scm.com/documentation>
- <http://www.kernel.org/pub/software/scm/git/docs/gittutorial.html>
- <http://www.kernel.org/pub/software/scm/git/docs/everyday.html>
- <http://git-scm.com/course/svn.html>
- <http://eagain.net/articles/git-for-computer-scientists/>
- <http://zrusin.blogspot.com/2007/09/git-cheat-sheet.html>

## ■ Git-Workflows

- <http://www.kernel.org/pub/software/scm/git/docs/gitworkflows.html>