

# Vorlesung Unix-Praktikum

## 6. Versionskontrolle mit git

Dirk Frettlöh

Technische Fakultät  
Universität Bielefeld

20. November 2019

# Willkommen zur sechsten Vorlesung

## Was gab es beim letzten Mal?

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

- ▶ Aliasse
- ▶ .bashrc, Umgebungsvariablen
- ▶ Shellskripte

# Willkommen zur sechsten Vorlesung

Was machen wir heute?

- ▶ Versionskontrolle allgemein
- ▶ Versionskontrolle mit git
  - ▶ Prinzipien
  - ▶ Lokale Nutzung

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

## Versionskontrolle (VCS) und git

# Versionskontrolle

## Wozu

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

## Anwendungen:

- ▶ Softwareprojekte
- ▶ Webseiten (z.B. Wikipedia)
- ▶ Allgemein alles mit vielen Texten, die sich fortlaufend ändern

# Versionskontrolle

Wozu

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

## Anwendungen:

- ▶ Softwareprojekte
- ▶ Webseiten (z.B. Wikipedia)
- ▶ Allgemein alles mit vielen Texten, die sich fortlaufend ändern

## Mögliche Probleme bei solchen Anwendungen:

- ▶ Datenverlust
- ▶ Vandalismus, Spam, Sabotage...
- ▶ Ab Version 0.34 tritt ein Fehler auf, den es in 0.33 noch nicht gab
- ▶ Zwei Leute ändern gleichzeitig eine Datei

Lösung: Versionskontrolle (VCS, "version control system")

# Versionskontrolle

## Wozu

1. Datenverlust
2. Vandalismus, Spam, Sabotage...
3. Ab Version 0.34 tritt ein Fehler auf, den es in 0.33 noch nicht gab
4. Zwei Leute ändern gleichzeitig eine Datei

Gegen 1 hilft regelmäßiges Datensichern.

Gegen 2 und 3 sichern mit Versionsnummer:

projekt-10-12-2016

projekt-14-12-2016

projekt-15-12-2016

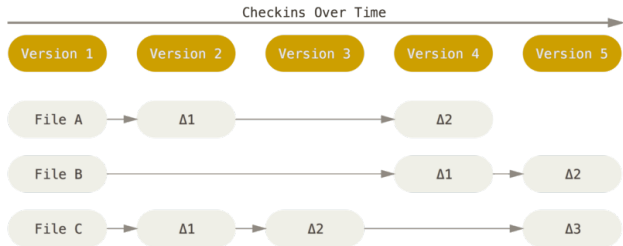
...

Ein Nachteil hier bereits: Viel Speicherplatz nötig.

# Versionskontrolle

## Nur Änderungen Sichern

Um keinen Speicherplatz zu vergeuden, werden in cleveren VCS nur die Änderungen gespeichert (entweder zur letzten Version oder aktuellen)



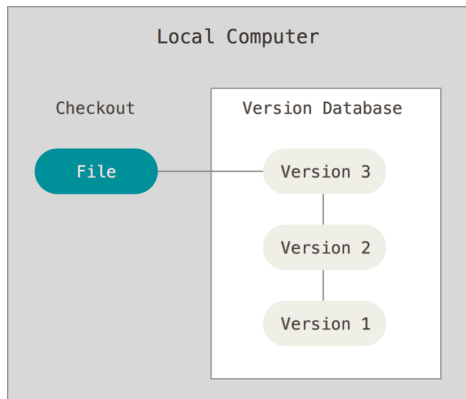
Aus den Änderungen lassen sich die alten Versionen rekonstruieren.



# Versionskontrolle

## Lokales Sichern

Das gibt es entweder lokal:

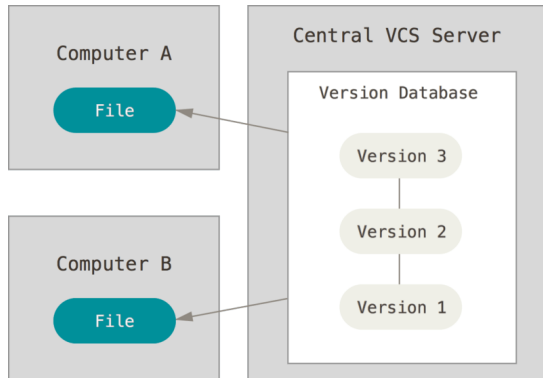


Z.B. RCS (GNU 1982-heute)

# Versionskontrolle

## Zentrales Sichern

...oder zentral:



Damit können mehrere Leute das Projekt bearbeiten.  
Beispiele: Subversion (Apache 2000-heute), CVS (1990-2008)

# Versionskontrolle

## Zentrales Sichern

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Dazu war der Ablauf:

- ▶ Nutzer A holt die aktuelle Version auf Rechner A
- ▶ Blockiert die Datei, die er bearbeiten will ("checkout")
- ▶ Lädt die geänderte Version auf den Server ("checkin")

Das wollen wir hier nicht vertiefen. Wir wollen auf git hinaus, das ist leichter zu lernen, wenn man die alten Systeme *nicht* kennt.

# Versionskontrolle

git

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

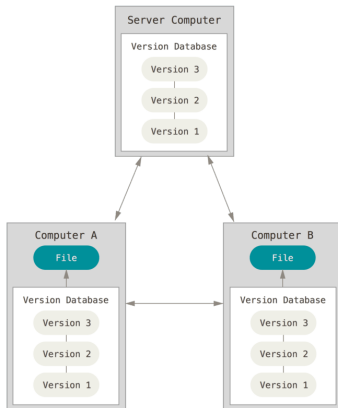
git (Linus Torvalds, Junio Hamano u.v.a., 2005-heute)

- ▶ Heute das meistgenutzte VCS (z.B. für den Linux-Kern, LibreOffice, Android, Gnome, Eclipse, Debian, PHP, ... git selbst; oder das Softwaregruppenprojekt im 3.&4. Semester an der Techfak)
- ▶ Dezentral
- ▶ Snapshots

# Versionskontrolle

## git

git speichert dezentral:

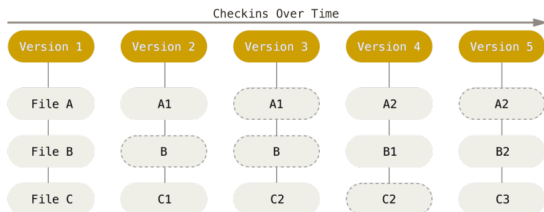


Jeder beteiligte Rechner speichert eine komplette Kopie.

# Versionskontrolle

## git

git speichert nicht nur Änderungen, sondern alles:



Natürlich auch effizient (keine Änderung: Verweis auf letzte Version der jeweiligen Datei)

# Versionskontrolle

## git

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Wichtiger für den Anwender sind folgende Fakten.

Unser Arbeitsverzeichnis mit dem im git gesicherten Projekt sei `~/projekt`.

Es gibt zwei Arten von Dateien in `projekt`:

- ▶ im git gesichert: "tracked"
- ▶ nicht im git gesichert: "untracked"

# Versionskontrolle

git

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Wichtiger für den Anwender sind folgende Fakten.

Unser Arbeitsverzeichnis mit dem im git gesicherten Projekt sei `~/projekt`.

Es gibt zwei Arten von Dateien in `projekt`:

- ▶ im git gesichert: "tracked"
- ▶ nicht im git gesichert: "untracked"

Getrackte Dateien können drei Zustände haben:

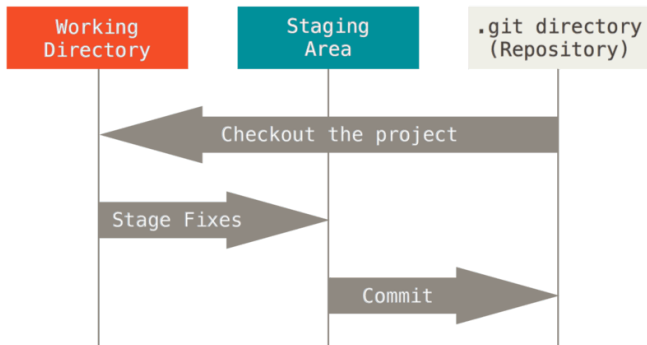
- ▶ unmodified (gegenüber der gesicherten Version unverändert)
- ▶ modified (gegenüber der gesicherten Version verändert)
- ▶ staged (verändert und für die nächste Sicherung bereitgestellt)



# Versionskontrolle

## git

Damit ergibt sich für die getrackten Dateien folgender Arbeitsablauf (bzgl. **wo** sie liegen):



(unmodified, modified  
or untracked)

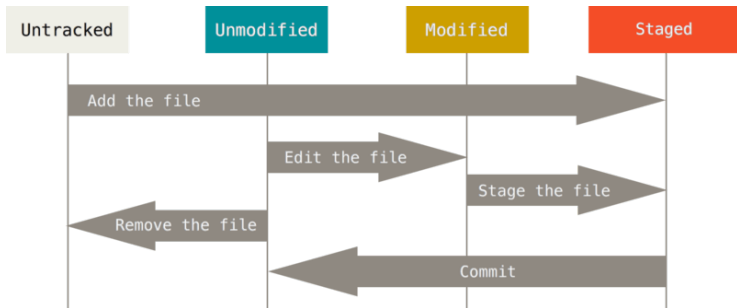
(staged)

(unmodified)

# Versionskontrolle

## git

Und für alle Dateien im Projekt gibt's folgende Möglichkeiten:  
(bzgl ihres **Zustands**)



# git - Lokales Arbeiten

## Bezeichnungen

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Ein einfacher Arbeitsablauf mit lokalem git.

Bezeichnungen:

- ▶ *Index*: der Stage-Bereich
- ▶ *Repo, Repository*: der wirkliche git-Speicher
- ▶ *wd*: kurz für *working directory*

# git - Lokales Arbeiten

## 0. Installieren

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Auf Techfak-Rechnern ist git installiert.

Ansonsten testen, etwa auf Ubuntu-Rechner so:

```
$ git --version
```

The program 'git' is currently not installed. You can install it by typing:

```
sudo apt-get install git
```

*(Zum Installieren also nun...)*

# git - Lokales Arbeiten

## 0. Voreinstellungen

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Nur einmal am Anfang (auf jedem Arbeitsrechner):

```
$ git config --global user.name "Dirk Frettlöh"  
$ git config --global user.email  
"dfrettloeh@techfak.uni-bielefeld.de"  
$ git config --global core.editor emacs
```

Das wird in der Datei `~/.gitconfig` gespeichert.

# git - Lokales Arbeiten

## 1. Anlegen eines git-Repos

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Ein neues Projekt anlegen ist sehr einfach:

Es sei ~/projekt ein Verzeichnis mit den Dateien eins.dat, zwei.dat, drei.dat.

```
$ cd projekt
```

```
$ git init
```

Ab jetzt gibt es ein Unterverzeichnis .git in projekt.  
Und ab jetzt kann git benutzt werden.

# git - Lokales Arbeiten

## 1. Dateien bearbeiten, *stagen* und *commiten*

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Am Anfang sind alle Dateien (eins.dat, zwei.dat, drei.dat) *untracked*.

Alle Dateien *stagen*:

```
$ git add *
```

Alle gestageten Dateien *commiten*:

```
$ git commit
```

# git - Lokales Arbeiten

## 1. Dateien bearbeiten, *stagen* und *commiten*

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Am Anfang sind alle Dateien (eins.dat, zwei.dat, drei.dat) *untracked*.

Alle Dateien *stagen*:

```
$ git add *
```

Alle gestageten Dateien *commiten*:

```
$ git commit
```

Nach git commit öffnet sich ein Editor, dort Kommentar eintragen (Z.B. 'Neues Projekt xyz'), Speichern, beenden.

Alternativ Kommentar mit

```
$ git commit -m "Neues Projekt xyz"
```



# git - Lokales Arbeiten

## 1. Dateien bearbeiten, *stagen* und *commiten*

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Aktuellen Status anzeigen:

```
$ git status
```

Nun evtl eine Datei ändern, z.B. drei.dat. (status angucken!)

Diese *stagen* und *commiten*:

```
$ git add drei.dat
```

```
$ git commit -m "drei.dat korrigiert"
```

# git - Lokales Arbeiten

## 1. Dateien bearbeiten, *stagen* und *commiten*

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Aktuellen Status anzeigen:

```
$ git status
```

Nun evtl eine Datei ändern, z.B. drei.dat. (status angucken!)

Diese *stagen* und *commiten*:

```
$ git add drei.dat
```

```
$ git commit -m "drei.dat korrigiert"
```

Oder die alte Version wiederherstellen

```
$ git checkout HEAD drei.dat
```

Auch möglich: komplettes wd aus dem repo wiederherstellen:

```
$ git reset --hard
```

# git - Lokales Arbeiten

## 1. Dateien bearbeiten, *stagen* und *commiten*

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Status gibt es auch kurz und knapp:

```
$ git status -s
```

Eine irrtümlich *gestagete* Datei wieder *unstagen*:

```
$ git reset -- drei.dat~
```

Eine irrtümlich zugefügte Datei wieder aus dem repo entfernen:

```
$ git rm drei.dat~ (löscht aus dem repo und dem wd)
```

```
$ git commit (sonst noch gestaged)
```

# git - Lokales Arbeiten

## Übersicht

Unix-  
Praktikum

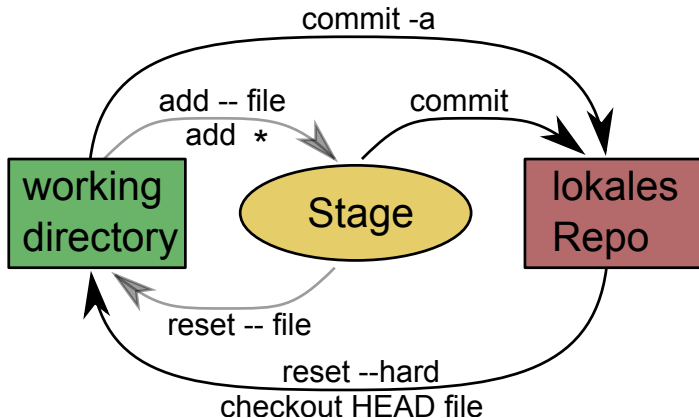
Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten



# git - Lokales Arbeiten

## Übersicht

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Weil git so riesig ist, gibt es keinen perfekten schnellen Zugang.  
Es gibt auch viele schlechte Infos im Netz.

Geeignet:

- ▶ progit (eBook mit 574 Seiten, insbes. Kap. 1+2)
- ▶ man giteveryday
- ▶ Spezielle Frage in Suchmaschine eingeben, auf den stackexchange-Treffer klicken
- ▶ `git cheatsheet` googeln? Selber machen?

# git - Lokales Arbeiten

## Übersicht

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

\$ man giteveryday (zeigen) Insbesondere:

- ▶ git-init to create a new repository. ✓
- ▶ git-log to see what happened.
- ▶ git-checkout and git-branch to switch branches.
- ▶ git-add to manage the index file. ✓
- ▶ git-diff and git-status to see what you are doing. ✓
- ▶ git-commit to advance the current branch. ✓
- ▶ git-reset, git-checkout (with path) undo changes. ✓
- ▶ git-merge to merge between local branches.
- ▶ git-rebase to maintain topic branches.
- ▶ git-tag to mark a known point.

# Git

## Lokales Arbeiten

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

**Recall:** `git status` zeigt aktuelle Zustände der Dateien  
(modified, staged,...)

`git log` zeigt die Versionsgeschichte (rückwärts).

```
$ git log
```

```
commit dd54eeed5aa6f7e37265e9d2f47a2e31886fc185
```

```
Author: Dirk Frettlöh <dfrettloeh@techfak.uni-bielefeld.de>
```

```
Date:   Wed Jan 4 16:09:20 2019 +0100
```

```
    drei.dat~ gelöscht
```

```
commit f7129820169d4ec0ae14e9dccbdb807ff4dd2e4c
```

```
Author: Dirk Frettlöh <dfrettloeh@techfak.uni-bielefeld.de>
```

```
Date:   Wed Jan 4 15:47:09 2019 +0100
```

```
    drei repariert
```

```
.....
```

# Git - Lokales Arbeiten

Nebenbei - weitere grep-Optionen

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

`git log` hat — wie die meisten git-Befehle — extrem viele Optionen (zeigen: `man git-log`)

`git log` lässt sich aber auch mit `grep` kombinieren. Dazu interessant:

- ▶ `grep -v xyz test.txt` findet alle Zeilen in `test.txt`, die `xyz` *nicht* enthalten



# Git - Lokales Arbeiten

Nebenbei - weitere grep-Optionen

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

`git log` hat — wie die meisten git-Befehle — extrem viele Optionen (zeigen: `man git-log`)

`git log` lässt sich aber auch mit `grep` kombinieren. Dazu interessant:

- ▶ `grep -v xyz test.txt` findet alle Zeilen in `test.txt`, die `xyz` *nicht* enthalten
- ▶ `grep -A5 xyz test.txt` zeigt alle Zeilen in `test.txt`, die `xyz` enthalten, jeweils *zusammen* mit den 5 Zeilen danach
- ▶ `grep -B3 xyz test.txt` zeigt alle Zeilen, die `xyz` enthalten, jeweils *zusammen* mit den 3 Zeilen davor
- ▶ `grep -A5 -B3 xyz test.txt` zeigt alle Zeilen, die `xyz` enthalten, jeweils *zusammen* mit den 3 Zeilen davor und den 5 Zeilen danach

# Git - Lokales Arbeiten

Recall: lokale git-Befehle

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

`$ man giteveryday`

- ▶ `git-init` to create a new repository. ✓
- ▶ `git-log` to see what happened. ✓
- ▶ `git-checkout` and `git-branch` to switch branches.
- ▶ `git-add` to manage the index file. ✓
- ▶ `git-diff` and `git-status` to see what you are doing. ✓
- ▶ `git-commit` to advance the current branch. ✓
- ▶ `git-reset`, `git-checkout` (with path) undo changes. ✓
- ▶ `git-merge` to merge between local branches.
- ▶ `git-rebase` to maintain topic branches.
- ▶ `git-tag` to mark a known point.

# Git - Lokales Arbeiten

## Branches

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT  
MESSAGES GET LESS AND LESS INFORMATIVE.

xkcd.com

Zu Branches (=mehrere parallele Versionen eines Projekts)  
später. Bisher immer nur ein branch: `master`

# Git - Lokales Arbeiten

git ignore

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Praktisch für emacs-Benutzer: `.gitignore`

Der Editor emacs legt beim Bearbeiten von `file.txt` automatisch eine Sicherungskopie `file.txt~` an.

Man kann git sagen, diese zu ignorieren (nie im repo zu sichern):

Eine Datei `.gitignore` im wd anlegen, dort reinschreiben etwa:

```
*.tgz
```

```
*~
```

Ab jetzt ignoriert git alles, was mit `.tgz` oder mit `~` endet.

# Git - Lokales Arbeiten

Eine Datei aus dem repo entfernen

Unix-  
Praktikum

Dirk Frettlöh

Version

control

Allgemein

git

Lokales Arbeiten

**Recall:** (s.o.) Wenn es zu spät ist (oder das ignore nicht funktioniert)

Wurde `file1.txt` irrtümlich hinzugefügt

```
git rm file1.txt
git commit -m "remove file1.txt"
```

# Überblick

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

- ▶ git-init to create a new repository. ✓
- ▶ git-log to see what happened. ✓
- ▶ git-add to manage the index file. ✓
- ▶ git-diff and git-status to see what you are doing. ✓
- ▶ git-commit to advance the current branch. ✓
- ▶ git-reset, git-checkout (with path) undo changes. ✓

...sowie grep-Optionen:

- ▶ -v xyz: alles ohne xyz anzeigen
- ▶ -A 3: auch die 3 Zeilen nach (*after*) dem Treffer anzeigen
- ▶ -B 5: auch die 5 Zeilen vor (*before*) dem Treffer anzeigen
- ▶ "^abc": gibt nur die Zeilen aus, die mit abc *anfangen*
- ▶ "abc\$": gibt nur die Zeilen aus, die mit abc *aufhören*

# Ausblick

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

Nächste Woche:

- ▶ Branches (Parallele Versionen eines Projekts)
- ▶ Remote (Verteiltes Arbeiten)
- ▶ Auf alte Version zurücksetzen

# Ende der heutigen Vorlesung

Unix-  
Praktikum

Dirk Frettlöh

Version  
control

Allgemein

git

Lokales Arbeiten

**Vielen Dank fürs Zuhören!**

**Bis nächste Woche!**