Git Einführung

Alvar, post@0x21.biz

30.07.2018

`git --table-of-contents`

`git --why-should-i-care?`

Git ist eine sehr verbreitete Versionsverwaltungssoftware. Aber...

- was heißt das?
- wofür brauche *ich* das?
- wieso sollte *mich* das überhaupt interessieren?

`git --vcs-example`

Situation

Bearbeiten eines **Projekts** am Computer - unabhängig davon, ob es sich um Design, Programmcode oder Text handelt.

Ablauf

1. Datei anlegen
2. Datei speichern
3. Datei bearbeiten
4. Datei erneut speichern
5. Datei löschen

git --vcs-abstract

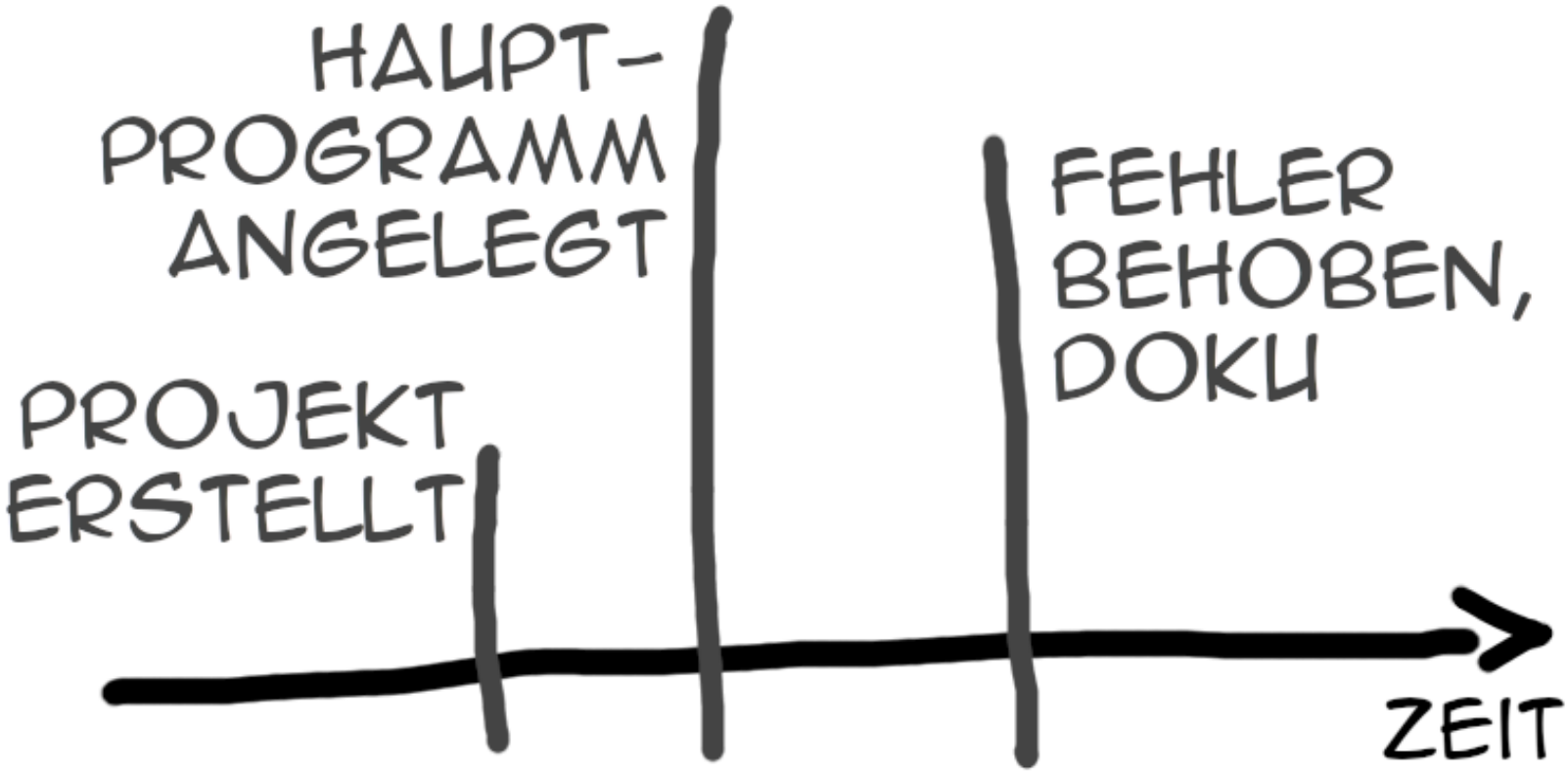
Ablauf, abstrakt

- Wiederkehrende Änderungen am Projekt
- Speichern einer Datei bringt uns zum nächsten Zustand

Zustand?

- T_i: aktueller Zustand (der Dateien) des Projekts
- T_{i-1}: vorherige Zustand, d.h. es wurde etwas geändert um in den jetzigen Zustand zu gelangen
- T_{i+1}: folgender Zustand, durch Änderung am aktuellen Zustand

git --vcs-less-abstract



`git --vcs-bottom-line`

- Zustand eines Meilensteins in der Versionsverwaltung vermerken
- Kommende Änderungen bauen darauf auf
  - *Zurückgehen* zum vorherigem Zustand möglich
  - (Mehrere) experimentelle Änderungen können gefahrlos vorgenommen werden

Wir erhalten somit ein *Journal* für unser Projekt, welches noch viel mächtiger als ein bloßes Backup ist.

git --slides

Foliensatz zu Git aus Lukas Kalbertodts Kurs Programmieren in Rust, alternativ auch auf YouTube.

```
git clone 'https://github.com/LukasKalbertodt
                                /programmieren-in-rust.git'

programmieren-in-rust/slides/
|-- ...
|-- 02-Git-GitHub-Rust-Environment.pdf
+-- ...
```

git --xkcd

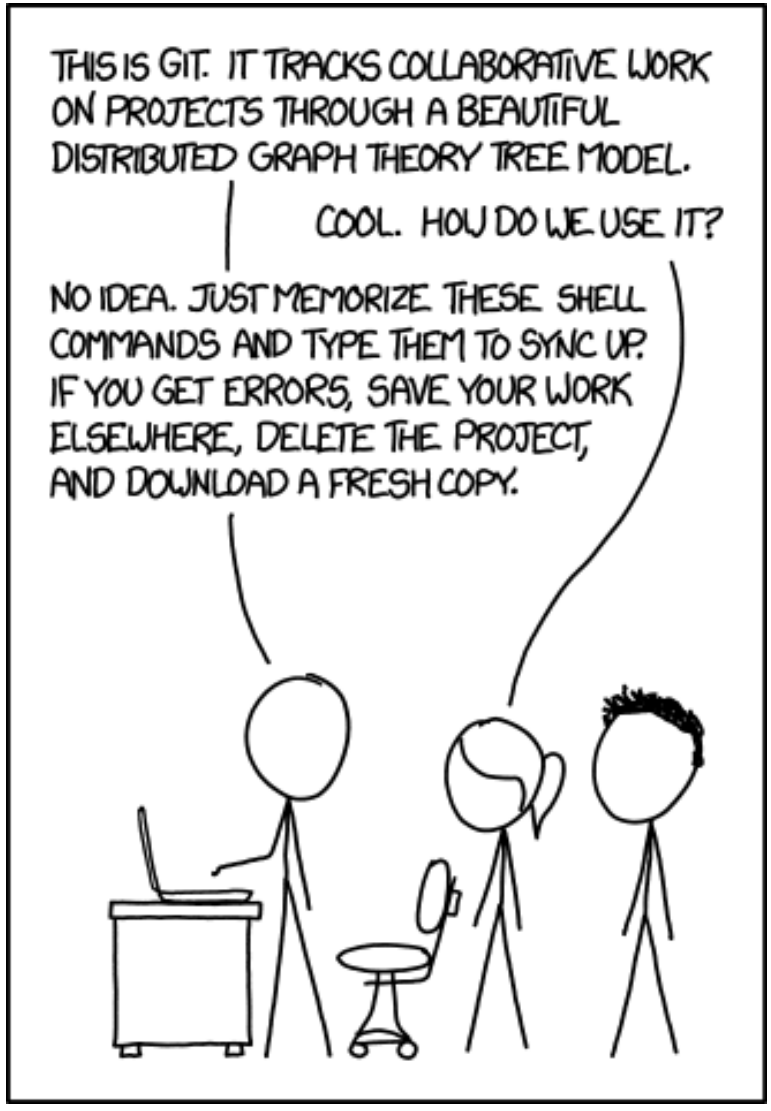


Figure 1: xkcd 1597, CC BY-NC 2.5

`git --hands-on`

1. Erstellen eines neuen Git-Repositories in `git-test`
2. Anlegen von `foo.txt`
3. Commit verfassen, welcher `foo.txt` enthält
4. `foo.txt` abändern und neu committen
5. Neue Datei `bar.txt` erzeugen, `foo.txt` abändern
6. Commit anfertigen, welcher nur `bar.txt` enthält

git --hands-on

```
# 1. Erstellen eines neuen Git-Repositories in ‘git-test‘
git init git-test
cd git-test

# 2. Anlegen von ‘foo.txt‘
touch foo.txt

# 3. Commit verfassen, welcher ‘foo.txt‘ enthält
git add foo.txt
git commit -m "Created foo.txt"
```

git --hands-on

```
# 4. ‘foo.txt‘ abändern und neu committen
echo "asdf" > foo.txt
git commit -a -m "Important change in foo.txt"

# 5. Neue Datei ‘bar.txt‘ erzeugen, ‘foo.txt‘ abändern
touch bar.txt
echo "qwerty" > foo.txt

# 6. Commit anfertigen, welcher nur ‘bar.txt‘ enthält
git add bar.txt
git commit -m "Added bar.txt"
```

git --gimme-all-ur-ssh-pubkeys

Als nächstes verwenden wir Git als verteilte Versionsverwaltungssoftware mit einem *zentralen* Repository auf meinem Server. Für Schreibrechte benötige ich von jedem einen SSH-Public Key und einen zugehörigen (Nick-) Namen.

Optional: Neues SSH-Schlüsselpaar anlegen

```
ssh-keygen -t ed25519 [-f ~/.ssh/git-intro]
```

URL

Bitte fügt dies dazu wie vorgegeben in folgendes Pad ein:
`https://pads.reis.asia/p/git-intro-2018`

git --playground

```
# Bei Keyauswahl durch export festlegen
export GIT_SSH_COMMAND=\
  "ssh -o IdentitiesOnly=yes -i ~/.ssh/git-intro"

git clone \
  'ssh://git@sadachbia.lurk.space:4223/playground'
```

`git --playground`

1. Datei mit dem gewählten (Nick-)Namen erstellen
2. Diese Datei mit einer sinnvollen Commit-Nachricht committen
3. Aktuellen Zustand des Repositories beziehen: `git pull`
   - Eventuell auftretende Merges nun behandeln, keine Panik
4. Commit zum Server pushen: `git push`

`git --playground-v2`

Je nach Gruppengröße bearbeiten wir nun verschiedene Dateien in Kleingruppen.

Dazu bearbeiten dann mehrere Personen jeweils gemeinsam die selbe Datei und versuchen versuchen beim `pull` dann die auftretenden Merge-Konflikte zu beheben.

git --conclusion

Danke für's Mitmachen und keine Angst vor Gits Lernkurve.

Folien

```
git clone \
  'ssh://git@sadachbia.lurk.space:4223/git-introduction'
```