git

jenseits von push und pull

Dr. iur. Michael Stehmann

rechtsanwalt-stehmann.de

Zur Person

Rechtsanwalt Dr. Michael Stehmann

- Studium der Rechtswissenschaft an der Universität zu Köln, danach Referendariat im Rheinland, Zweite juristische Staatsprüfung, Zulassung als Rechtsanwalt 1987
- Promotion an der Rechtswissenschaftlichen Fakultät der Universität zu Köln
- Seit März 1999 als selbstständiger Rechtsanwalt in Langenfeld / Rheinland tätig
- Mitglied der Legal Networks der FSFE, Mitglied des Project Management Committees des Apache OpenOffice-Projektes, Vorsitzender des Vereins Freie Software Freunde e.V.

Intention

- Viele Menschen nutzen git,
- kennen aber nützliche Funktionen nicht.

Übersicht

- Was ist git?
- Funktionen, die (vielleicht) nicht jeder kennt
- Eigener git-Webserver

1. Teil - Was ist git?

git ist eine freie Software zur verteilten Versions-verwaltung von Dateien.

Eine Versionsverwaltung ist ein System, das zur Erfassung von Änderungen an Dateien verwendet wird. Versionsverwaltungssysteme werden in der Softwareentwicklung eingesetzt, um Quelltexte zu verwalten.

Man sie aber auch anderweitig nutzen.

Was ist git?

Vorzüge (einige):

- Nicht-lineare Entwicklung
- Kein zentraler Server
- Datentransfer zwischen Repositorien
- Kryptographische Sicherheit der Projektgeschichte

2.Teil

Nützliche Funktionen

Was kommt nicht vor?

- git init
- git commit
- git tag
- git reset
- git update-ref

- git pull
- git fetch
- git push
- git merge
- git rebase
- ...

Übersicht

man git

git clone

- git clone --origin <sprechenderName> …

  kann mit git remote -v überprüft werden

- git clone --bare <Repo> <repo>.git

  legt ein Repository ohne Arbeitsverzeichnis an

git add

- git add -p
- git add -i
- git add -v

git rm und git mv

- git mv <AlterName> <NeuerName>
- git rm <(Pfad/)Datei>
- git rm -r <Verzeichnis>

Wiederherstellen einer Datei

- git show '<commit>:<file>' # Anzeigen
- git reset <commit> -- <file> # Im Index
- git checkout <commit> -- <file> # Im Working Tree

Wiederherstellen einer Datei

- git show '<commit>:<file>' # Anzeigen
- git reset <commit> -- <file> # Im Index
- git checkout <commit> -- <file> # Im Working Tree

git archive

- Erzeugt ein Archiv aus einem benannten „tree“ (Verzeichnis)
- git archive -l zeigt die zur Verfügung stehenden Archivformate
- git archive –format=<fmt> legt den Typ des Archivs fest – Default ist .tar
- git archive --remote=<url> <branch> <(Unter-)Verzeichnis>

Unterordner klonen:

- git archive --remote=<url> <branch> <Unterverz> | tar xvf -

git archive

- Erzeugt ein Archiv aus einem benannten „tree“ (Verzeichnis)
- git archive -l zeigt die zur Verfügung stehenden Archivformate
- git archive –format=<fmt> legt den Typ des Archivs fest – Default ist .tar
- git archive --remote=<url> <branch> <(Unter-)Verzeichnis>

Unterordner klonen:

- git archive --remote=<url> <branch> <Unterverz> | tar xvf -

git cherry-pick

Mit git cherry-pick <commit> kann ein einzelner Commit aus einem anderen Branch “herausgepickt” und in den aktuellen Branch übernommen werden. Git extrahiert dazu einen Patch für die Änderungen des Commits und erstellt einen neuen Commit im aktuellen Branch.

git cherry-pick -x <commit>

Die Information, woher der Commit stammt, kann mit -x in der Historie dokumentiert werden.

git worktree

Mehrere Arbeitsverzeichnisse

- git worktree add <Pfad> <Branch>
- git worktree add -b <Neuer Branch> <Pfad> <bisheriger Branch>
- git worktree list
- git worktree move <bisheriger Ort> <neuer Pfad>
- git worktree remove <Arbeitsverzeichnis>

alternativ:

- rm -rf <Verzeichnis>; git worktree prune

git lfs

- LFS = „Large File Storage“
- Paket git-lfs
- Erweiterung zu git für (große) Binärdateien
- Beispiele: Bilder, Töne, Videos
- Funktion: Statt Binärdatei wird Pointer auf diese abgespeichert, der LFS-Server zeigt
- man git lfs
- git lfs install und git lfs env

git diff - 1

- Unterschiede zwischen dem letzten und dem vorletzten Commit:

  git diff HEAD~1 HEAD

- Besser:

  git diff <alte Commit-ID> <neue Commit-ID>

- Unterschiede letzter Commit und Arbeitsverzeichnis:

  git diff HEAD

git diff - 1

- Unterschiede zwischen dem letzten und dem vorletzten Commit:

  git diff HEAD~1 HEAD

- Besser:

  git diff <alte Commit-ID> <neue Commit-ID>

- Unterschiede letzter Commit und Arbeitsverzeichnis:

  git diff HEAD

git diff - 2

- Unterschiede letzter Commit und Staging-Area:

  git diff --cached
- Unterschiede Staging-Area und Arbeitsverzeichnis:

  git diff
- Statistik:

  git diff --stat

git diff - 3

Konfiguration, welches Diff-Programm git verwendet (im Beispiel melt):

- systemweit (/etc/gitconfig):

  # git config --system merge.tool meld

- nutzerspezifisch (~/.gitconfig)

  $ git config --global merge.tool meld

- für das Repository (.git/config)

  $ git config merge.tool meld

git grep

In allen Dateien nach bestimmtem Text suchen:

- Im Working Tree:

  $ git grep –untracked <text> # tracked + untracked
- Im aktuellen Commit von master:

  $ git grep <text> master
- $ git grep -i <text> # Ignore case
- $ git grep <text> -- '*.c' # Nur in *.c-Dateien

Text kann auch regulärer Ausdruck sein.

Recherchen im Repository

- git log --stat # Änderungsstatistik je Datei
- git log --name-status # Geänderte Dateien auflisten
- git log -p # Mit Patch
- Commits mit einer bestimmten Commit-Message:

  git log --grep=<message>
- git log <Datei> # Commits, die <Datei> verändern
- git log <Verzeichnis>/ # Commits, die Dateien unter <Verzeichnis>/ verändern

Recherchen im Repository

- git log --stat # Änderungsstatistik je Datei
- git log --name-status # Geänderte Dateien auflisten
- git log -p # Mit Patch
- Commits mit einer bestimmten Commit-Message:

  git log --grep=<message>
- git log <Datei> # Commits, die <Datei> verändern
- git log <Verzeichnis>/ # Commits, die Dateien unter <Verzeichnis>/ verändern

Recherchen im Repository

- Commits, die einen bestimmten Text einer Datei hinzufügten oder entfernten

  git log -S'<text>' # In der Historie von HEAD

  git log --all -S'<text>' # In allen Commits

  Mit -G können reguläre Ausdrücke verwendet werden.

- git log --name-status # Geänderte Dateien auflisten
- welcher Commit und Autor hat welche Code-Zeile zuletzt geändert:

  git blame <Datei>

Prompt verschönern

In .bash_profile einfügen:

```
parse_git_branch()
    {     git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \(.*\)/ (\1)/'
}
export PS1="\u@\h \[\033[32m\]\w\[\033[33m\]\$
(parse_git_branch)\[\033[00m\] $ "
```

GUIs

- gitk (im git-Paket enthalten)
- git gui (Paket git-gui)

3. Teil

Webserver

cgit

Anpassungen in /etc/cgitrc

summary-tags=10
enable-index-links=1
enable-commit-graph=1
enable-log-filecount=1
enable-log-linecount=1
enable-git-config=0

scan-path=/srv/git/

Quelle:

https://wiki.archlinux.org/index.php/Cgit#Adding_repositories

cgit

/etc/apache2/conf-available/cgit.conf

```
ScriptAlias /cgit/ "/usr/lib/cgit/cgit.cgi/"
RedirectMatch ^/cgit$ /cgit/
Alias /cgit-css "/usr/share/cgit/"
<Directory "/usr/lib/cgit/">
   AllowOverride All
   Options ExecCGI FollowSymlinks
   Require all granted
</Directory>
```

gitweb

Änderungen in /etc/gitweb.conf

$projectroot = "/srv/git";

gitweb

/etc/apache2/conf-available/gitweb.conf – 1. Teil

```
<IfModule mod_alias.c>
  <IfModule mod_mime.c>
    <IfModule mod_cgi.c>
      Define ENABLE_GITWEB
    </IfModule>
    <IfModule mod_cgid.c>
      Define ENABLE_GITWEB
    </IfModule>
  </IfModule>
</IfModule>
```

gitweb

/etc/apache2/conf-available/gitweb.conf – 2. Teil

```
<IfDefine ENABLE_GITWEB>
  Alias /gitweb /usr/share/gitweb

  <Directory /usr/share/gitweb>
    Options +FollowSymLinks +ExecCGI
    # eingefuegt durch michael 2018-08-05 wg. htaccess
    AllowOverride All
    AddHandler cgi-script .cgi
  </Directory>
</IfDefine>
```

Zum Schluss

Vielen Dank für Ihre Aufmerksamkeit.

Noch Fragen?

Kontakt:

Dr. Michael Stehmann:
info@rechtsanwalt-stehmann.de