Git auf Subversion

Git ist ein verteiltes Versionskontrollsystem, das sich in einigen Eigenschaften von traditionellen Versionskontrollsystemen unterscheidet.

== Auf bestehendes SVN aufsetzen ==
Beachten wir den ersten Vorteil, dann braucht man sich nicht wundern, dass das erste Klonen des SVN Repositories etwas länger dauert.

Quellcode

```
1. git svn clone https://svn.easy-coding.de/wcf
```

Um so mehr freuen wir uns, wenn wir die Dateigrößen betrachten. Vergleichen wir ein Projekt mit 2800 Revision. SVN Checkout meldet 45 MB. Bei Git sind es nur 18 ;)

== Gesamte Versionsgeschichte ==
Jeder Benutzer besitzt eine lokale Kopie der gesamten Projektgeschichte. Es ist kein zentraler Server notwendig. Mit SVN kann man lediglich die letzte Revision einer Datei wiederherstellen. Weil Git außerdem mit Kompression arbeitet macht die Dateigröße einer kompletten Historie oft nur einen Bruchteil der Größe eines SVN Checkouts aus.

Quellcode

```
1. git log
```

== Offline Commits ==
Muss man mehrere abhängige Änderungen gemeinsam übertragen bleibt einem bei SVN nichts anderes übrig als alle Änderungen in einer Commit Nachricht zusammenzufassen. Oft kann man gar nicht mehr alle Änderungen nachvollziehen. Git löst dieses Problems. Mit Git kann man alle Commits sammeln und mit einem finalen **dcommit** an den Server senden.
Dort kommen die Lognachrichten wieder einzeln an.

Bei Git muss man jede Datei die commitet werden soll mit add hinzufügen. Alternativ kann man den Parameter mit -a setzen und alle versionierten Dateien werden automatisch commitet.

Quellcode

```
1. #create robots.txt
2. git add robots.txt
3. git commit -m "added the new robots.txt"
4. #do some changes to other files
5. git commit -a -m "did security improvements"
```

== Nicht-lineare Entwicklung ==
Sowohl das Erstellen neuer Entwicklungszweige (branching) sowie das Verschmelzen (merging) sind einfach und effizient. Git enthält Programme, mit deren Hilfe sich die nicht-lineare Geschichte eines Projektes einfach visualisieren lässt, und mit deren Hilfe man in dieser Geschichte navigieren kann.

Welches Branches existieren?

Quellcode

1. git branch

Erstellen eines neuen lokalen Branch:

Quellcode

1. git branch experimente

Zum anderen Branch wechseln:

Quellcode

1. git checkout experimente

Hierbei wird das Dateisystem mit allen Unterschieden zwischen experimente und master umgeschrieben. Änderungen gehen keine verloren - man kann jederzeit wieder zu master wechseln.

== Flexibler Datentransfer ==
Daten können mit einer Reihe verschiedener Protokolle übertragen werden. Git besitzt ein sehr effizientes eigenes Protokoll, das auf TCP-Port 9418 arbeitet. Ebenso kann der Transfer über SSH oder (weniger effizient) über HTTP, HTTPS, FTP oder rsync erfolgen.

== Nützliche Git Tools ==
=== Log ===

Quellcode

1. git log

=== Diff ===

Quellcode

1. git diff

=== Grep ===
Jeder kennt es bei SVN. Man sucht etwas im Quelltext, muss aber erstens die versteckten SVN Ordner ausblenden und zweitens lange darauf warten bis grep die Binären Dateien durchsucht hat. Mit Git kein Problem, denn es bringt eine eigene grep Implementierung mit.

Quellcode

1. git grep 'getInstance('

== Git konfigurieren ==
Einstellungen lassen sich entweder global für ein ein Projekt durchführen.

=== Git in Farbe ===

Quellcode

1. git config --global color.diff auto
2. git config --global color.status auto
3. git config --global color.branch auto

=== Git mit SVN Befehlen ===
Ich halte nicht viel von Aliasen. Befehle heißen nicht ohne Grund, wie sie eben heißen. Um für die Umsteiger doch ein paar Aliase zu definieren, der legt diese wie folgt an:

Quellcode

1. git config --global alias.update 'svn fetch'

== Literatur ==

- git.or.cz/course/svn.html
- linux.yyz.us/git-howto.html