Einführung in Git

Dirk Deimeke

My own IT

19. August 2013

Inhalt

Allgemeines zu Versionskontrollsystemen (1)

Dateien versionieren

- alte Revisionen sichern und wieder herstellen
- Unterschiede zwischen Versionen anzeigen

Zentralisiertes Versionskontrollsystem (VCS)

- CVS oder Subversion (SVN)
- Daten und Historie liegen auf einem zentralen Server
- Mitarbeiter checkt eine lokale Arbeitskopie aus

Allgemeines zu Versionskontrollsystemen (2)

Verteiltes Versionskontrollsystem (distributed VCS)

- Mercurial (HG), Bazaar (BZR) oder Git
- Alle Veränderungen liegen lokal und nicht nur auf dem Server
- Jede Kopie (Klon) ist somit ein vollumfängliches Backup
- Verschiedene lokale Kopien können synchronisiert werden

Und jetzt Git

Linus Torvalds

- Initiator und Hauptentwickler von Git
- Ist stolz auf Git, mehr als auf Linux

Zitat Linus Torvalds

Linux ist etwas nachgemachtes, Git ist etwas völlig Neues.

Zitat Anwender

Git ist cool :-)

Git-Objekte (1)

Keine Angst, es gibt nur vier. Und mehr Theorie gibt es auch nicht, dafür gibt es wirklich gute Bücher.

Blobs – „Binary Large Objects“

In Blobs liegen alle Dateien, die Dateien werden durch ihre Prüfsumme (SHA1 Hash) referenziert. In Blobs ist nicht der Name enthalten. So stellt Git sicher, dass mehrere Dateien mit gleichem Inhalt nur ein Mal gespeichert werden.

Trees

Ein Baumobjekt steht für einen Verzeichniszweig, darin enthalten sind „Blob identifier“ (Referenzen auf Dateien), Pfadnamen und ein paar Metadaten. Baumobjekte können sich gegenseitig referenzieren. Damit kann man einen kompletten Verzeichnisbaum darstellen.

Git-Objekte (2)

Commits

Ein Commit beinhaltet die Daten für jede Veränderung eines Repositories. Enthalten sind Autor, Committer, Commit Datum und die Lognachricht. Jeder Commit zeigt auf ein einzelnes Verzeichnisobjekt, das den Stand des Repositories zu einem bestimmten Zeitpunkt repräsentiert. Bis auf den allerersten Commit (Init), hat jeder Commit einen Eltern (Parent) Commit.

Tags

Tags sind einfach lesbare Namen für Objekte (meistens für Commits), die durch ihre SHA1-Prüfsumme identifiziert werden.
1e9036eb5764aea857bbd5f0522efbfec3ff4a36 ist zwar eindeutig, aber „Version-1.1“ ist deutlich lesbarer.

Git installieren

Git Homepage

- yum install git
- aptitude install git-core

Erste Konfiguration

Nutzerkonfiguration

```
git config --global user.name "Dirk Deimeke"
git config --global user.email "dirk@deimeke.net"
git config --global color.ui always
```

Erstes Repository

Einfaches Beispiel

```
mkdir repository
cd repository
git init
git add -A .
git commit -m "Init"
ls -la .git
cat .git/config
```

Arbeiten

Und die ersten Dateien

```
git status
vim Erste-Datei.txt
git status
git add Erste-Datei.txt
git status
git commit -m "Erste Datei, jetzt frisch"
git status
```

Differenzen

Unterschiede

```
vim Erste-Datei.txt
git status
git diff Erste-Datei.txt
vim Zweite-Datei.txt
git status
git diff HEAD
git commit -am "Aenderungen an der ersten Datei"
git add Zweite-Datei.txt
git commit -m "Zweite Datei"
```

Mist gebaut?

Datei irrtümlich gelöscht

```
rm Zweite-Datei.txt
git status
git checkout Zweite-Datei.txt
```

Einfache Kommandos

Was man erst einmal braucht

```
git log          # Liest die Historie und zeigt alle
                 # Changes
git log -p       # zeigt ausserdem Differenzen
                 # zwischen Changes
git log --stat   # Summiert Changes auf

git mv           # Datei oder Verzeichnis umbenennen
git rm           # Datei oder Verzeichnis loeschen

git grep         # Durchsucht das Repository nach
                 # einem Muster (nicht binaer)
```

Fortgeschrittene Kommandos

Das braucht man selten

```
git archive # Archiv eines Teilbaums erstellen
git revert  # Nimmt den letzten Commit mit einem
            # neuen Commit zurueck
git blame   # Wer traegt Schuld?
```

Manpages

`man git`

Besonderheit für Git-Kommandos, es gibt Hilfe für jedes Kommando, die Manpage für `git add` findet sich unter `git-add`.
Es gibt viele, **wirklich** viele Manpages.

Protokolle

Auf Git kann über viele verschiedene Protokolle zugegriffen werden

- Disk oder Filesystem
- HTTP[S]
- Git Protokoll über TCP
- Git Protokoll über SSH (bevorzugt)

Klonen

Meistens, Klon über SSH

```
git clone user@host:/some/path/to/repo.git xxx
```

- Macht einen lokalen Klon ins Verzeichnis xxx.
- Checkt den Zweig „HEAD“ als master aus
- Setzt das entfernte Repository als Origin

Hochladen

Lokal am Master arbeiten

```
arbeiten ; git commit -a
```

Eventuell prüfen, was sich verändert hat

```
git fetch origin ; git diff master origin/master
```

Aktualisieren vom Upstream

Mit merge: `git pull origin master`
Mit rebase: `git pull --rebase origin master`

Und Hochladen zum Upstream

```
git push origin master
```

Links (1)

- Git Homepage
- Wikipedia über Git
- Git in 15 Minuten – interaktiver Online Kurs
- Git How To
- CRE130 Verteilte Versionskontrollsysteme (Podcast)

Links (2)

- Pro Git Buch (Gratis!)
- Pro Git auf GitHub
  `git clone https://github.com/progit/progit.git progit.git`

- O'Reilly zu Git
- Open Source Press Git

- GitHub
- GitHub Pages

Subversion und Git

Man kann mit `git svn` auch auf Subversion-Repositories zugreifen, eventuell muss das svn-Feature nachinstalliert werden. Vorteil ist, dass man dabei die Vorteile von Git (alles ist lokal) hat.

Der folgende Link enthält einen Git Crash Course, für die die Subversion bereits kennen: Git - SVN Crash Course

Ich habe einmal aufgeschrieben, wie man seine Subversion-Repositories nach Git migrieren kann: Migration von Subversion zu Git

Ausblick und Lizenz

Ausblick

In der nächsten Folge geht es um die Arbeit mit Remote Repositories am Beispiel von GitHub.

Lizenz