



**XING** 

# **git** verteilte Versionskontrolle

**Jens Muecke**  
Duesseldorf, 2. Oktober 2009

1. Merkmale
2. Entstehungsgeschichte
3. Aufbau
4. Einfuehrung
5. Erfahrung und Tipps

- Dezentrales Content-Management-Tool
- Entwickelt fuer nicht-lineare Entwicklung
- Fuer kleine und sehr grosse Projekte
- Extrem skalierbar
- Verfuegbar fuer alle aktuellen Betriebssystemen
- Open Source (GPL)
- Unverfaelschliche Versionshistorie



- BitKeeper ist schuld
- linux.conf.au 2005
- ReverseEngineering von Andrew Tridgell

Foto von  
Gopal Vijayaraghavan, Bangalore, India



- Verbindung aufbauen

**telnet thunk.org 5000**

```
Trying 69.25.196.29...  
Connected to thunk.org.  
Escape character is '^]'.  
git>
```

- Hilfesystem benutzen

**help**

```
? - print this help  
abort - abort resolve  
check - check repository  
clone - clone the current repository  
help - print this help  
httpget - http get command  
[...]
```

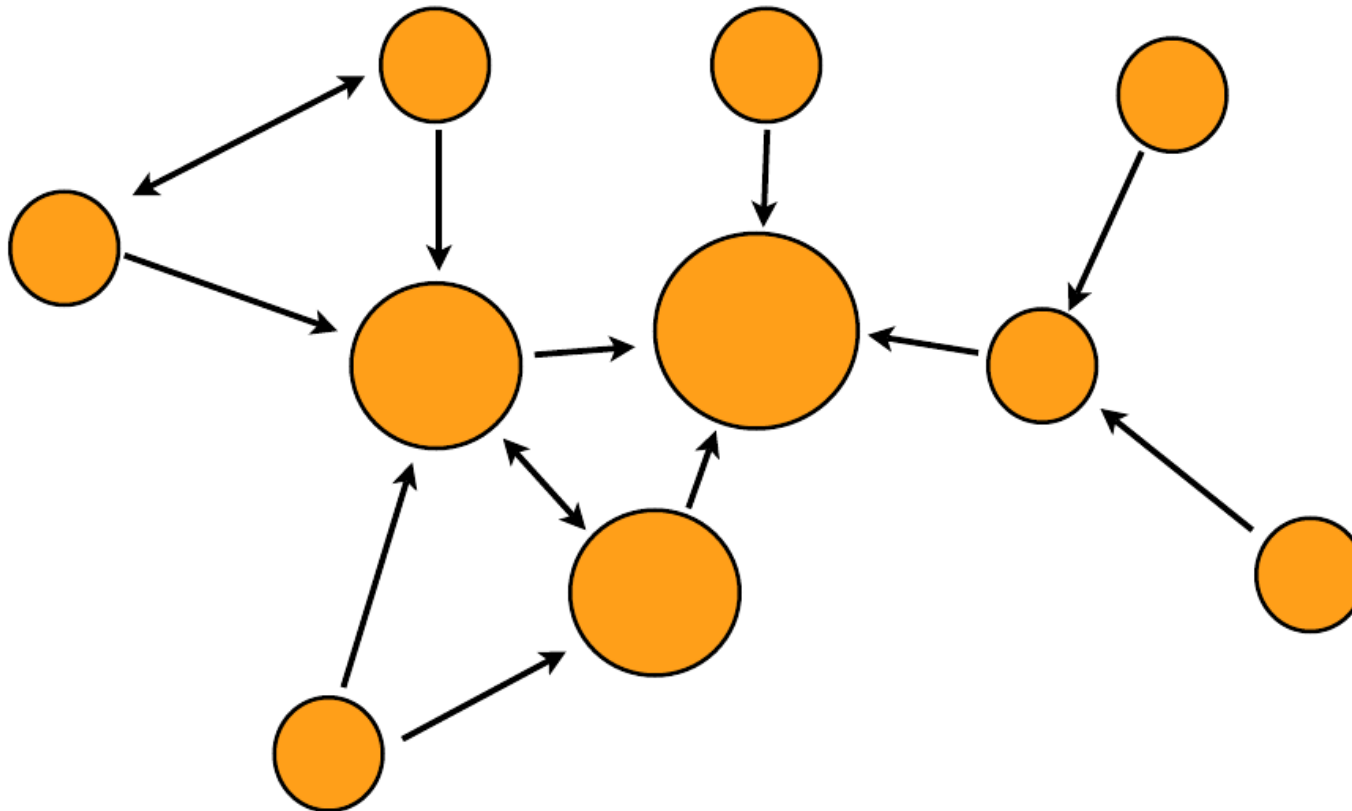
- SourcePuller wurde angekuendigt
- Larry McVoy (CEO BitMover) war sticksauer
- Ein riesen FlameWar um BitKeeper wurde ausgeloesst
- Die freie Entwickler-Lizenz fuer BitKeeper wurde zurueckgezogen

- Linux Kernel 2.6.12-rc2 war gerade fertig
- Nach Alternativen fuer BitKeeper wurde gesucht (selbst Tarballs und Patchfiles wurden in Betracht gezogen)
- Entwicklung wurde in der Zeit eingestellt
- Erste lauffaehige Version von git wurde entwickelt (innerhalb von zwei Wochen)

- |                  |                                |
|------------------|--------------------------------|
| · 03. April 2005 | Entwicklungsstart              |
| · 06. April 2005 | Erste Ankuendigung             |
| · 07. April 2005 | Self Hosting                   |
| · 18. April 2005 | erster mehrfacher Branch merge |
| · 16. Juni 2005  | Linux 2.6.12 Release - mit git |

- Dezentrale Entwicklung mit mehr als 7000 Entwicklern
  - Austausch von Patches untereinander
  - Schnell und Effektiv bei einer Quellcodebasis des Kernel
- 
- Von 2.6.12-rc2 bis 2.6.12 - 1859 Commits
  - Aenderungen laut Greg Kroah-Hartman, 5. Juni 2008, google tech talk
    - Durchschnitt 2007 - 2008 pro Tag
    - 4300 Zeilen hinzugefuegt
    - 1800 Zeilen geloescht
    - 1500 Zeilen veraendert

- Es gibt kein “zentrales” Repository (wie SubVersion oder CVS)
- Jeder ist sich sein eigenes Repository
- Jeder kann mit jedem Patches tauschen (in beide Richtungen)
- Repositories werden nicht kopiert, sondern geklont
- Ermöglicht offline arbeiten



- Jede Version (Commit) erzeugt ein 20-Byte-Hash (SHA-1), welcher eindeutig auf die Version verweist
- Der Hash schliesst Vorversionen mit ein (kryptographische Sicherheit – keine Aenderung der Historie moeglich ohne die Gueltigkeit zu verlieren)
- Damit ist auch ein Download aus nicht vertrauenswuerdigen Quellen moeglich – da ueberpruefbar
- MergeTracking (ueber Eltern/Kinder)

- Lokales Dateisystem
- GIT (TCP Port 9418)
- SSH (TCP Port 22)
- HTTP/S (TCP Port 80 & 443)
- FTP/S (TCP Port 20 & 21)
- Email



- Ein ChangeSet ist ein Delta zwischen zwei Versionsstaenden
- Aehnlich wie Patchfiles (diff -git)
- Verzeichnisse werden nicht mitverwaltet
- Beinhaltet auch Dateiattribute (rwxrwxrwx)
- Keine Subsets (wie Subversion) aber SubModules
- Ein ChangeSet kann mehrere Eltern und mehrere Kinder haben
- Aenderungen auf Zeilenbasis
- Auch Binaerdateien moeglich

- Es gibt nur ein Repository Verzeichnis (.git)
- Projekthistorie ist komprimiert und i.d.R kleiner als die lokale Arbeitskopie
- Im “bare” Mode liegen nur die Repository-Informationen vor



Erstellen eines Repositories

`git init`

Klonen eines Repositories

`git clone <quelle>`

Hinzufuegen einer Datei in die Versionskontrolle

`git add <dateiname>`

Loeschen einer Datei aus der Versionskontrolle

`git rm <dateiname>`

Verschieben einer Datei innerhalb des Repositories/umbenennen

`git mv <alt> <neu>`

Aktueller Status ueber die Dateien (untracked, hinzugefuegt, geloescht, unmerged, veraendert, umbenannt)

`git status`

Versionshistorie

`git log`

Details eines ChangeSets

`git show <version>`

Festschreiben eines Versionsstandes	<code>git commit &lt;dateien&gt;</code>
Springen zu einem Versionsstand oder Branch	<code>git checkout &lt;version&gt;</code>
Verschmelzen von zwei Versionen (erzeugt neuen Commit oder via FastForward)	<code>git merge &lt;version&gt;</code>
ChangeSets aus einer anderen Quelle laden und in den eigenen Tree mergen	<code>git pull &lt;quelle&gt;</code>
ChangeSets zu einer anderen Quellen senden	<code>git push &lt;ziel&gt;</code>
Differenz anzeigen	<code>git diff &lt;version1&gt;..&lt;version2&gt;</code>

Alle Tags auflisten

```
git tag
```

Tag erstellen

```
git tag <name>
```

Tag loeschen

```
git tag -d <name>
```

Tag signieren

```
git tag -s <name>
```

Tags verweisen auf eine Version und koennen Wandern. Tags koennen mitgeklont werden, aber jeder kann sich auch seine eignen Tags erstellen. Sie erleichtern den Umgang (statt Hashes).

Lokale Branches auflisten

`git branch`

Alle Branches auflisten

`git branch -a`

Branch erstellen

`git branch <name>`  
`git checkout -b <name>`

Branch loeschen

`git branch -D <name>`

Remote Branch loeschen

`git push <remote> :heads/<name>`

Um auf einen Remote-Branch aufzusetzen und den Anschluss nicht zu verlieren werden Branches getracked.

`git checkout -b mybranch origin/remotebranch --track`

Alle Aenderungen eines Branches koennen mit den `-squash` Parameter zu einen ChangeSet zusammengefuegt werden.

Standardname fuer den default branch ist master.

Alle Quellen auflisten

```
git remote
```

Neue Quelle hinzufuegen

```
git remote add <name> <url>
```

Quelle loeschen

```
git remote rm <name>
```

Quelle umbenennen

```
git remote rename <alt> <neu>
```

Quelle ziehen

```
git fetch <name>
```

Standartname bei "git clone" ist "origin".

- Hookskripte koennen Aktionen starten
- Gelten pro Repository und werden nicht mitgeklont
- Fuer Pruefungen, Logs usw.
  
- Es gibt eine globale (\$HOME/.gitconfig) und eine Repository Konfiguration (.git/config)
- Hier werden Name und Email-Adresse fuer die Commits definiert
- Einstellungen fuer die verschiedenen Plugins

- Git hat ein Gabage Collection “git -gc”
- Branches fuer Releases (CherryPick fuer BugFixes)
- Versuche Features, Bugfixes in der Entwicklung in Branches verlagern
- Strukturen in Personen statt Rechteverteilung (reviews)  
(Ausnutzen von Senden und Holen von Patches)
- Brauchbar zum Tracken von Dateiaenderungen/Konfigurationen
- MergeKonflikte von Entwicklern loesen lassen,  
die den Konflikt ausgeloesst haben
- “git help <command>” hilft immer weiter
- “git svn” zum kopieren von SubVersion Repositories

- Gitx als graphische Oberflaeche (Metroplan)
- gitweb/cgit fuer Weboberflaechen
- Github fuer Hosting (es gibt auch andere)
- Unterstuetzung in allen gaenigen IDEs
- Libraries in allen gaenigen Programmiersprachen
- Als Backend in einigen CMS/Wikis (ikiwiki)

**Thank you**  
**for your kind attention!**

**POWERING RELATIONSHIPS**  
**WWW.XING.COM**