Git

Ein verteiltes Versionskontrollsystem

Julius Roob

Chaos inKL.

18. Oktober 2011

Übersicht

Verteilte Versionskontrolle

- Versionskontrolle
- kein zentraler Server
- gleichberechtigte Repositories
- lokale Repositories
- wirkt kompliziert
- ist einfach

Entstehung

- April 2005: Freie Bitkeeper Lizenz wird zurückgezogen
- Linus Torvalds untersucht Alternativen...
- ... und schreibt Git

Repository

Alles liegt in *.git* im Projektordner.
Anlegen mit

```
~/Projekte/Aurora> git init
```

im Projektordner
Oder Inhalt eines bestehenden Repos ziehen:

```
/tmp> git clone https://github.com/Zottel/Aurora.git
Cloning into Aurora...
(Here be dragon output)
/tmp> ls -a Aurora
.  ..  aurora.lua  config.example.lua  data  doc
.git  .gitignore  irc.lua  modules
```

Commit

- Dateien zum Stage hinzufügen:

```
~/Projekte/Aurora> git add aurora.lua
```

- Commit:

```
~/Projekte/Aurora> git commit -m "Fast nix"
```

Kleinigkeiten

- Welche Dateien sind geändert/staged?

  ~/Projekte/Aurora> **git status**

- Aus dem Stage entfernen

  ~/Projekte/Aurora> **git rm --cached aurora.lua**

- Löschen

  ~/Projekte/Aurora> **git rm aurora.lua**

- Alle Commits des aktuellen Branch

  ~/Projekte/Aurora> **git log**

- Unterschied zwischen Commits

  ~/Projekte/Aurora> **git diff <erster> <zweiter>**

Branch

Schick für:

- Testen von Ideen
- Größere Umbauten
- Zusammenführen von Änderungen

Grundlagen:

- Neuer Branch

  > **git branch <name>**

- Branch wechseln

  > **git checkout <branch>**

- Branches zusammenführen

  > **git merge <branch>**

Remote

Verweise auf andere GIT Repositories

- *origin*: Standard Ursprung eines geklonten Repos
- Remote hinzufügen:

  > **git remote add <name> <url>**

- Lokales Repo aktualisieren (inklusive merge)

  > **git pull <remote name>**

- Entferntes Repo aktualisieren - Pushen

  > **git push <remote name>**

Veröffentlichen

Veröffentlichen mit:

- Git Server
- Github, Gitorious etc...
- SSH
- HTTP

Entwicklungsmodelle

- Zentrales Repo
- „Friendly Dictator“ - Linux Kernel

Objekte

Fünf Basisdatentypen

- Blob: Datei
- Tree: Verzeichnis
- Commit
- Tag
- Ref: Verweis auf speziellen Commit (z.B. Tag)

SHA1

Jedes Objekt wird über SHA1 Hash identifiziert

- Datenintegrität
- Schützt gegen Manipulation

Tags können direkt mittels GPG signiert werden.

Commits

Commits beinhalten:

- Beliebig viele Vorgänger (mehrere bei *Merge*)
- Wurzelverzeichnis

Branches: Mehrere Commits mit einem Vorgänger.

Stash

Änderungen „kurz“ zwischenspeichern: Stash

- Kurz aktuellen Zustand „wegspeichern“

  > **git stash**

- gespeicherten Zustand wiederherstellen

  > **git stash pop [<stash>]**

- Stashes auflisten

  > **git stash list**

.gitignore

```
julius@eve:~/Projekte/JuliOS_CPP$ cat .gitignore
test
*.swp
*.o
kernel
```

Cherry Picking

Einzelne Commits aus anderem Branch übernehmen

> **git cherry-pick d8c12ba1...**

Den Commit noch mittels *git commit* in den aktuellen Branch aufnehmen. Fertig

Remote Fetch

Sicherer, sauberer als „git pull“
Entfernten Entwicklungsstand in einen extra branch ziehen

> **git fetch <remote> <quellbranch>:<zielbranch>**

Beispiel:

> **git fetch z0ttel master:z0ttel**

Stände vergleichen

> **git diff z0ttel**

Merge der Branches

> **git merge z0ttel**

Bisect

Commit aufspüren mit dem ein Fehler eingeführt wurde

- Bisect beginnen

  > **git bisect start**

- Einen als Fehlerhaft bekannten Commit angeben

  > **git bisect bad <commit>**

- (Möglichst) letzten guten Commit angeben

  > **git bisect good <commit>**

Binärsuche

Bisect - automatisch

- Bisect beginnen

  > **git bisect start**

- Einen als Fehlerhaft bekannten Commit angeben

  > **git bisect bad <commit>**

- (Möglichst) letzten guten Commit angeben

  > **git bisect good <commit>**

- Testskript bestimmen

  > **git bisect run <skript>**

- z.B.

  > **git bisect run make test**

SVN

Mit git-svn auch lokales Git mit entferntem SVN Server möglich.

Alternativen

Alternative Dezentrale Versionskontrollsysteme:

- Mercurial mit *hg* als Kommandozeilentool
- Bazaar
- Darcs

Quellen

- http://blog.ianbicking.org/distributed-vs-centralized-scm.html
- http://git-scm.com/about
- http://de.wikipedia.org/wiki/Secure_Hash_Algorithm
- http://de.wikipedia.org/wiki/Git

Fragen?

Fragen, Anmerkungen, Verbesserungen?

Danke

Danke