Git

Ulrich Hoffmann
Hinnerk Haardt

Inhalt

- Einführung
- Fernzugriff
- Grundlagen
- destruktive Befehle
- branches & tags
- Wartung

Einführung

- Versionsverwaltungssystems 3. Generation
- Entwickler Linus Torvalds (Linux, y'know?)
- verteiltes System

Hilfe!

- git help
- git help commit
- git help -a # >150 Befehle

Vorbereitung

- Name und E-Mail-Adresse setzen:

```
$ git config --global user.name \
"Hinnerk Haardt"

$ git config --global user.email \
haardt@information-control.de
```

Fernzugriff

- Repositories auf Server (Github etc.)
- Bundles auf USB-Stick
- Patches per E-Mail

Repository kopieren

- git clone https://github.com/joyent/node.git

remote repository

- wird von clone automatisch angelegt
- git remote -v
- git remote add demo2 git://10.1.1.1/test

Änderungen holen

- git fetch origin master;
  git merge origin/master
- git pull origin master

Änderungen hochladen

- git push origin master
- git push --tags origin master

Inhalt: Grundlagen

- Daten ein- und auschecken
- Dateien organisieren
- Zustand und Historie

Daten einchecken I

- git status
- git add datei2.txt
- git commit -m "Kommentar"

Daten einchecken II

- selektives Einchecken:
  - git add --interactive
  - git add -p

Commits

- je nach Projektkultur
- entweder logische Einheiten
- oder Einheiten im Arbeitsablauf

Dateien organisieren

- git rm file3.txt
- git mv datei2.txt datei3.txt

Zustand und Historie

- git status
- git diff
- git show
- git log

git help gitrevisions

oder einfach…

```
git gui
```

Destruktive Befehle

- VORSICHT
- Befehle sind nicht immer intuitiv
- revert unterscheidet sich von Subversion
- ggf. vorsichtshalber Repository kopieren

Datei zurücksetzen

- git checkout <branch> <Datei>
- git checkout master file4.txt

git help gitrevisions

Alles zurücksetzen

- git reset --hard

git help gitrevisions

Commit rückgängig machen

- git revert <commit>

git help gitrevisions

Geschichtsklittung

- git commit --amend

Inhalt: branches

- Übersicht
- Standard-Operationen: auflisten, wechseln, anlegen, löschen
- merge
- rebase

branches

- Parallelentwicklung im Sourcecode
- für Features, Bugfixes, …
- viele: rails hat 19, node hat 53 branches
- http://scottchacon.com/2011/08/31/github-flow.html

branch auflisten

- git branch
- git branch -r
- git branch -a

branch anlegen

- git branch cheese;
  git checkout cheese
- git checkout -b cheese

branch wechseln

- git checkout master
- git checkout cheese

branch umbennen

- git branch -m cheese bacon

merge (I)

- Zusammenführen von zwei branches
- git checkout master; git merge bacon

merge (2)

- Konflikte finden:
  - git status
- nach gelöstem Konflikt:
  - git add <filename>
- wenn es zu schlimm aussieht:
  - git merge --abort

rebase

- Alternative zu merge
- erzeugt flache Historie, branches "verschwinden"
- kann zu Problemen führen

branch löschen

- git branch -d bacon

remote branch löschen

- git push origin :cheese
- git push --delete origin cheese

tags

- Markierung eines Objektes, z.B. eines commits
- für Versionen, Releases etc.
- können signiert (PGP) sein
  - Signatur gilt für tree bis zum tag

tags anzeigen

- git tag
- git tag -l

tag anlegen

- git tag -a v2.0 # annotated tag
- git tag -m "Version 2.0" v2.0
- git tag -s v2.0

Wartung

- Git ist ein Dateisystem
- sollte gelegentlich gewartet werden

Garbage Collector

```
$ du -hs .git
282M .git

$ git gc
Counting objects: 14586, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (5407/5407), done.
Writing objects: 100% (14586/14586), done.
Total 14586 (delta 9199), reused 12395 (delta 7599)
Removing duplicate objects: 100% (256/256), done.

$ du -hs .git
273M .git
```

fsck

```
$ git fsck --full
Checking object directories: 100% (256/256), done.
Checking objects: 100% (14586/14586), done.
```

Fragen?

gleich geht es mit

Github weiter…