Konfigurationsdateien mit Git verwalten

Chemnitzer Linuxtage 2011

Julius Plenz

19. März 2011

http://chemnitzer.linux-tage.de/2011/vortraege/782

Das Problem

- Mehrere Rechner, auf denen ähnliche Aufgaben ausgeführt werden
  - Laptop zu Hause, Laptop auf der Arbeit
  - Workstation
  - Server
- Wie synchronisiere ich Konfigurationsdateien?
  - Dateien sollen auf den Rechnern *fast* gleich sein
  - Je nach Rechner kleine Anpassungen
    - `$HOME` anders
    - andere `mutt`-Version $\longrightarrow$ neue Optionen
    - Spezielle Shell-Aliase

`rsync`, you're *out*!

- Leider bieten viele Programme keinen `include`-Befehl
  - Problem könnte man sonst umgehen durch Kommando `include .`*`program`*`.local`
- Ein simples Synchronisieren via `rsync` oder `scp` scheidet aus
  - Mögliche lokale Änderungen gehen verloren
  - Keine Möglichkeit, die lokalen Anpassungen sinnvoll zu kennzeichnen und beizubehalten

Die Idee

- Stammkonfiguration vorhalten
- Kleine Anpassungen als "Patch-Stack" gegen die generische Version verwalten
  - `/home/plenz` $\rightarrow$ `/home/feh`
  - Spezielle Aliase hinzufügen
  - ...

Lösung: Git!

- Git bietet die wesentlichen Techniken
  - Einfaches Branching
  - Übertragung von Änderungen (dezentral)
  - `rebase`: Patch-Stack neu aufbauen
- Bonus: Versionsverwaltung der Dateien *frei Haus*!

Auf geht's!

- Home-Verzeichnis als Git-Repository initialisieren
- Nicht das Verzeichnis `.git` verwenden!

Repository initialisieren

```
GIT_DIR=~/.configs.git GIT_WORK_TREE=$HOME git init
```

- Weil wir nicht `.git` verwenden, muss ein Shell-Alias her

Shell-Alias

```
alias conf="GIT_DIR=~/.configs.git git"
```

Configs importieren

- Zunächst die vorhandenen Configs importieren

Import

```
conf add .vimrc .muttrc .zshrc ...
conf commit -m "Import der Configs"
```

- Stammkonfiguration auf `master`
- Lokale Anpassungen auf `local`

Branch `local` erstellen

```
conf branch local
conf checkout local
```

Stammkonfiguration lagern

- Stammkonfiguration am besten auf eigenem Server lagern
  - Nur SSH-Zugriff nötig
- Alternative: Github, `repo.or.cz`

Auf dem Server

```
git init --bare ~/configs.git
```

Auf dem Client

```
conf remote add origin server:configs.git
conf push -u origin master
```

- `master` wird das erste Mal hochgeladen

Neue Rechner hinzufügen

- Diese Kommandos müssen *einmalig* ausgeführt werden

Configs-Repository klonen nach ~/.configs.git

```
conf clone --bare server:configs.git .configs.git
conf config core.bare false
conf config core.worktree $HOME
conf remote rm origin
conf remote add origin server:configs.git
conf reset master
```

Lokal vorliegende Änderungen übernehmen

```
conf checkout -b local
conf diff
conf add/commit/...
```

Änderung an der Stammkonfiguration

- *Alle* Änderungen werden zunächst im Branch `local` gemacht
  - `conf add`, `conf commit`
- Änderungen per Cherry-Pick in die Stammkonfiguration übernehmen
  - `conf checkout master`
  - `conf cherry-pick ...`
- Dann `local` auf `master` neu aufbauen
  - `conf rebase master local`
  - Rebase ignoriert die doppelten Commits
- `master` hochladen
  - `conf push origin master`
- Struktur verwirrend? $\longrightarrow$ `gitk --all`

Stammkonfiguration-Updates

- Nach Updates suchen
  - `conf remote update`
- Updates integrieren
  - `conf checkout master`
  - `conf merge --ff-only origin/master`
- `local` neu aufbauen
  - `conf rebase master local`

- Rebase kann fehlschlagen. $\rightarrow$ Konflikt lösen

Achtung, Achtung!

- Vorsicht bei folgenden Kommandos
  - `conf checkout -f`
  - `conf clean`
    - Kann mit falschen Optionen komplett `$HOME` löschen!
- Branchwechsel können nicht durchgeführt werden, wenn
  - lokale Änderungen vorliegen, die nicht eingecheckt sind
  - der Checkout eine ungetrackte Datei überschreiben würde
- Git nimmt nativ keine Unix-Rechte auf
  - Vorsicht bei der Verwaltung von `/etc`, zum Beispiel
    - `/etc/sudoers` mit falschen Permissions ist *schwierig* zu reparieren, wenn man das Root-Passwort verlegt hat...

Pros & Cons

- Pro
  - Homogene Umgebung auf allen Rechnern
  - Synchronisation der Dateien bei Bedarf
  - Flexibilität von Git
  - Versionsgeschichte der Konfigurationsdateien
- Contra
  - Disziplin vonnöten
  - Mehr Zeitaufwand als simples quick-'n'-dirty-Editieren
  - Pull-Prinzip, keine automatische Verteilung

Alternativen

- etckeeper – http://kitenet.net/~joey/code/etckeeper/
- Dewi – https://github.com/ft/dewi
- IsiSetup – http://sf.net/projects/isisetup/
- Puppet

Danke!

Fragen und Kommentare gerne an julius@plenz.com.

Configs unter https://github.com/Feh/configs