ES MUSS NICHT IMMER GIT SEIN.

Fossil ist eine Alternative!

von **Holger Jakobs**

ABER WARUM?

Gibt's Probleme mit Git?

Nicht direkt, aber das Bessere ist der Feind des Guten.

Git ist halt **nur** eine Quelltext-Versionsverwaltung.

Git ist auf die Linux-Kernel-Entwicklung zugeschnitten.

Die meisten Projekte sind jedoch völlig anders organisiert.

Es wird viel zusätzliche Software (Bugtracker, Wiki, ...) benötigt.

WAS BRAUCHT'S FÜR EIN PROJEKT?

Klar, eine Quelltext-

Versionsverwaltung

Einen Bugtracker,

beispielsweise Bugzilla, Redmine, Gitlab ...

Ein Wiki, beispielsweise DokuWiki, MediaWiki, ...

Diese Tools wollen erst ein mal installiert und dann miteinander verheiratet werden.

INSTALLATIONSVORAUSSETZUNGEN

Git benötigt

- `libcurl3-gnutls`
- `libexpat1`
- `libpcre3`
- `zlib1g`
- `perl`
- `liberror-perl`

Darum ist *Git for Windows* ein über 40 MB großer Download.

INSTALLATIONSVORAUSSETZUNGEN

Bugzilla benötigt

- `apache2 mysql-server libappconfig-perl libdate-calc-perl libtemplate-perl libmime-perl build-essential libdatetime-timezone-perl libdatetime-perl libemail-sender-perl libemail-mime-perl libemail-mime-modifier-perl libdbi-perl libdbd-mysql-perl libcgi-pm-perl libmath-random-isaac-perl libmath-random-isaac-xs-perl apache2-mpm-prefork libapache2-mod-perl2 libapache2-mod-perl2-dev libchart-perl libxml-perl libxml-twig-perl perlmagick libgd-graph-perl libtemplate-plugin-gd-perl libsoap-lite-perl libhtml-scrubber-perl libjson-rpc-perl libdaemon-generic-perl libtheschwartz-perl libtest-taint-perl libauthen-radius-perl libfile-slurp-perl libencode-detect-perl libmodule-build-perl libnet-ldap-perl libauthen-sasl-perl libtemplate-perl-doc libfile-mimeinfo-perl libhtml-formattext-withlinks-perl libgd-dev libmysqlclient-dev lynx-cur graphviz python-sphinx`
- einen Datenbankserver (diverse sind geeignet)
- einen Webserver

INSTALLATIONSVORAUSSETZUNGEN

DokuWiki benötigt bei einer Beispielinstallation

- `libapache2-mod-php7.2 libjs-jquery-cookie libjs-jquery-ui libphp-simplepie php php-common php-curl php-geshi php-intl php-ldap php-seclib php-xml php7.2 php7.2-cli php7.2-common php7.2-curl php7.2-intl php7.2-json php7.2-ldap php7.2-opcache php7.2-readline php7.2-xml`
- einen Webserver

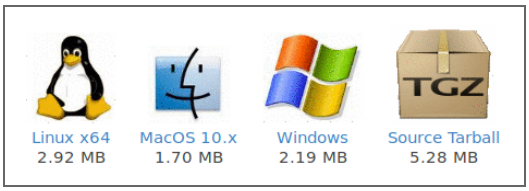
Ja, das alles muss unter Linux kein Problem darstellen, ist aber Arbeit.

INSTALLATIONSVORAUSSETZUNGEN FOSSIL

Fossil benötigt gar nichts!

Fossil ist ein single-file executable.

To install Fossil → download the stand-alone executable and put it on your $PATH.



Get it **now!**
http://fossil-scm.org

INSTALLATION VON FOSSIL

3 Schritte:

1. Download der für das Zielsystem passenden Version.
2. Auspacken des Archivs (`.zip` oder `.tar.gz`)
3. Verschieben der ausführbaren Datei in ein Verzeichnis des Suchpfads.

Danach einfach mal `fossil` eingeben und schauen, ob die Kurzhilfe (`Usage: ...`) erscheint.

VORTEILE VON FOSSIL

VIELES IST BEREITS EINGEBAUT:

- Bugtracker
- Wiki
- Forum
- Technotes

Fossil kann vieles.

VORTEILE VON FOSSIL

- Web-Interface
- Keine Installation nötig! (self-contained)
- Keine Spezial-Protokolle! (SSH, HTTPS, Proxy-fähig)
- Auto-Sync (auf Wunsch)
- Robustes Dateiformat (SQLite3)
- Sehr liberale Lizenz (Open Source)

Fossil ist einfach und sicher.

VORTEILE VON FOSSIL

- Inhalte von Repos sind unveränderlich.
- Die Geschichte kann nicht umgeschrieben werden (kein Rebasing, kein Squashing).
- Es gibt keine „staging area“.
- Beim Sync wird alles synchronisiert (außer privaten Branches und lokalen Einstellungen).
- Alle synchronisierten Repos enthalten alle Branches.
- Repos können auch mit `cp` bzw. `copy` kopiert werden.
- Repo + Fossil-Programm für div. Plattformen auf USB-Stick: immer dabei, immer arbeitsbereit!

Fossil macht Spaß!

VORTEILE VON FOSSIL

Arbeiten im Team

Zum Kooperieren kopiert man einfach das Repo oder klont es.

- Fossil synchronisiert immer das gesamte Repo.
- Das schließt auch Tickets, Wiki-Seiten, Forumsbeiträge, Technotes ein.
- Nur als „privat“ gekennzeichnete Branches sind davon ausgeschlossen.
- Jeder Entwickler hat in seinem Repo immer das Gesamtprojekt.
- Es gehen also keine Entwicklungen verloren, wenn mal ein Rechner untergeht.
- Am einfachsten und sicher geht die Kooperation über SSH.

GIT-GRUNDLAGEN

Git-Grundlagen zeigen viele Vorteile von Git auf:

- Snapshots, nicht Diffs
- Fast jede Operation ist lokal
- Git stellt Integrität sicher
- Git verwaltet fast ausschließlich Daten
- Die drei Zustände (committed, modified, staged)

In Kapitel 2 wirst Du mehr über diese Zustände lernen und darüber, wie Du sie sinnvoll einsetzen und wie Du den Zwischenschritt der Staging Area auch einfach überspringen kannst.

Ach! Bis auf die drei Zustände alles wie bei Fossil!

GIT MUSS KONFIGURIERT WERDEN

Beispiel

```
$ git config --global merge.tool vimdiff
```

Git kann von Hause aus mit den folgenden Diff Programmen arbeiten: kdiff3, tkdiff, meld, xxdiff, emerge, vimdiff, gvimdiff, ecmerge, and opendiff.

Fossil hat ein eingebautes `diff`, keine Konfiguration nötig. Aber es kann ein beliebiges externes Diff-Programm verwendet werden.

WIE KOMMT MAN AN HILFE?

```
$ git help verb
$ fossil help verb
```

Wo ist der Unterschied?

JETZT ABER LOS!

Wir starten mit dem Fossil-Projekt.

Mit `fossil help new` finden wir heraus, wie das geht.

```
$ fossil new ~/FOSSIL/projekt1.fossil
```

Ja, die Repos kommen alle gemeinsam in einen Ordner.

```
$ mkdir ~/halloWelt
$ cd ~/halloWelt
$ fossil open ~/FOSSIL/projekt1.fossil
```

Jetzt mal schauen, was alles da ist!

CODE ERZEUGEN!

Wir schreiben die Datei `halloWelt`.

```
#!/bin/sh
echo "Hallo Welt"
```

Speichern und ausführbar machen.

Ausführen: `$ ./halloWelt`

Sollte `Hallo Welt` ausgeben. Tolle Wurst!

WAS MEINT FOSSIL?

Mal fragen: `$ fossil status`

```
fossil status
repository:   /home/uta/FOSSIL/projekt1.fossil
local-root:   /home/uta/halloWelt/
config-db:    /home/uta/.fossil
checkout:     41abe30...56f 2018-10-24 19:52:11 UTC
tags:         trunk
comment:      initial empty check-in (user: uta)
```

Es ist also noch nichts im Repo.

WAS TUN?

Fragen, was neu ist:

```
$ fossil extra
halloWelt
```

Einfach alles Neue ins Repo aufnehmen (und ggf. fehlendes rauswerfen):

```
$ fossil addremove
ADDED  halloWelt
added 1 files, deleted 0 files
```

UND JETZT?

```
$ fossil status
repository:   /home/uta/FOSSIL/projekt1.fossil
local-root:   /home/uta/halloWelt/
config-db:    /home/uta/.fossil
checkout:     41abe30...56f 2018-10-24 19:52:11 UTC
tags:         trunk
comment:      initial empty check-in (user: uta)
ADDED      halloWelt
```

Noch immer ist nichts im Repo.

ALSO COMMITTEN!

```
$ fossil commit -m "erster Versuch von halloWelt"
New_Version: d4c9447...dc7
```

Jetzt ist es im Repo.

```
$ fossil status
repository:   /home/uta/FOSSIL/projekt1.fossil
local-root:   /home/uta/halloWelt/
config-db:    /home/uta/.fossil
checkout:     d4c9447...dc7 2018-10-24 20:01:41 UTC
parent:       41abe30...56f 2018-10-24 19:52:11 UTC
tags:         trunk
comment:      erster Versuch von halloWelt (user: uta)
```

ABER WELCHE DATEIEN SIND DRIN?

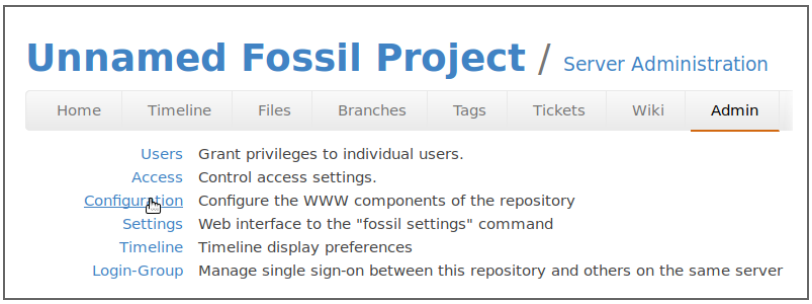
```
$ fossil ls -v --age
 UNCHANGED  2018-10-24 20:01:41  halloWelt
```

Das Kommando heißt als `ls`, wie in der Shell.

)AS AUCH IN BUNT?

Na klar! `$ fossil ui` startet gleich den Browser mit und zeigt die Timeline des Projekts.

Gleich in den Admin-Bereich gehen und das Projekt benennen!



ALLES WEITERE IN DER PRAXIS

Wir starten `fossil server`, damit auch andere Rechner aufs Repo zugreifen können.

- Wir konfigurieren im Browser weiter.
- Wir schauen die Timeline an.
- Wir ändern Dateien, fügen neue hinzu, löschen und benennen um.
- Alles wird in Fossil aufgzeichnet.
- Wir synchronisieren zwischen Repos.
- Wir schreiben Tickets und Wiki-Pages.
- Wir schreiben Forumsbeiträge.

WEITERHIN VIEL SPASS UND ERFOLG MIT FOSSIL!

made with **reveal.js**

holger@jakobs.com
http://plausibolo.de