Git
the fast version control system

Alexander Sulfrian

Spline Talks

Institut für Informatik
Freie Universität Berlin

03.05.2010

- Autor: Linus Torvalds

- Autor: Linus Torvalds
- Linux Kernel verwendete das proprietäre BitKeeper

- Autor: Linus Torvalds

- Linux Kernel verwendete das proprietäre BitKeeper
- angebliche Lizenzverletzung durch Reverse Engineering des Protokolls

- Autor: Linus Torvalds

- Linux Kernel verwendete das proprietäre BitKeeper
- angebliche Lizenzverletzung durch Reverse Engineering des Protokolls
- keine kostenlose Nutzung mehr möglich

- Autor: Linus Torvalds

- Linux Kernel verwendete das proprietäre BitKeeper
- angebliche Lizenzverletzung durch Reverse Engineering des Protokolls
- keine kostenlose Nutzung mehr möglich
- keine guten Alternativen

- 3. April 2005: Start der Entwicklung

- 3. April 2005: Start der Entwicklung
- 6. April 2005: Ankündigung auf der Mailingliste

- 3. April 2005: Start der Entwicklung
- 6. April 2005: Ankündigung auf der Mailingliste
- 7. April 2005: „self-hosting“

- 3. April 2005: Start der Entwicklung
- 6. April 2005: Ankündigung auf der Mailingliste
- 7. April 2005: „self-hosting“
- 18. April 2005: erster Merge

- 3. April 2005: Start der Entwicklung
- 6. April 2005: Ankündigung auf der Mailingliste
- 7. April 2005: „self-hosting“
- 18. April 2005: erster Merge
- 29. April 2005: Zielperformance erreicht
  (6.7 Patches pro Sekunde)

- 3. April 2005: Start der Entwicklung
- 6. April 2005: Ankündigung auf der Mailingliste
- 7. April 2005: „self-hosting“
- 18. April 2005: erster Merge
- 29. April 2005: Zielperformance erreicht
  (6.7 Patches pro Sekunde)
- 16. Juni 2005: Veröffentlichung der Version 2.6.12 des Linux Kernels (erste Version, die mit Git verwaltet wurde)

- 3. April 2005: Start der Entwicklung
- 6. April 2005: Ankündigung auf der Mailingliste
- 7. April 2005: „self-hosting“
- 18. April 2005: erster Merge
- 29. April 2005: Zielperformance erreicht
  (6.7 Patches pro Sekunde)
- 16. Juni 2005: Veröffentlichung der Version 2.6.12 des Linux Kernels (erste Version, die mit Git verwaltet wurde)
- 26. Juli 2005: Abgabe der Entwicklung an Junio Hamano

Linus Torvalds

„I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git.“

Linus Torvalds

„I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."

- britischer Jargon für eine dumme oder unfreundliche Person

Linus Torvalds

„I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."

- britischer Jargon für eine dumme oder unfreundliche Person
- mittlerweile aber auch Backronyme vorhanden

Linus Torvalds

„I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."

- britischer Jargon für eine dumme oder unfreundliche Person
- mittlerweile aber auch Backronyme vorhanden
- zum Beispiel: Global information tracker

- alte Versionen von Dateien (unveränderbar) speichern

- alte Versionen von Dateien (unveränderbar) speichern
- möglichst komprimieren

- alte Versionen von Dateien (unveränderbar) speichern
- möglichst komprimieren
- Verwaltung ganzer Verzeichnisse

- alte Versionen von Dateien (unveränderbar) speichern
- möglichst komprimieren
- Verwaltung ganzer Verzeichnisse
- gemeinsamer Zugriff

- alte Versionen von Dateien (unveränderbar) speichern
- möglichst komprimieren
- Verwaltung ganzer Verzeichnisse
- gemeinsamer Zugriff
- Vergleich verschiedener Versionen

- alte Versionen von Dateien (unveränderbar) speichern
- möglichst komprimieren
- Verwaltung ganzer Verzeichnisse
- gemeinsamer Zugriff
- Vergleich verschiedener Versionen
- Zusammenführen verschiedener Änderungen

git - **Verteilte** Versionsverwaltung:

- kein zentraler Server nötig (aber möglich)

git - **Verteilte** Versionsverwaltung:

- kein zentraler Server nötig (aber möglich)
- komplette Geschichte der Dateien lokal verfügbar

git - **Verteilte** Versionsverwaltung:

- kein zentraler Server nötig (aber möglich)
- komplette Geschichte der Dateien lokal verfügbar
- keine Netzwerkverbindung nötig

git - **Verteilte** Versionsverwaltung:

- kein zentraler Server nötig (aber möglich)
- komplette Geschichte der Dateien lokal verfügbar
- keine Netzwerkverbindung nötig
- schnell!

git - **Verteilte** Versionsverwaltung:

- kein zentraler Server nötig (aber möglich)
- komplette Geschichte der Dateien lokal verfügbar
- keine Netzwerkverbindung nötig
- schnell!
- klein!

```
git init
```

```
git init
```

```
git add .
```

```
git init

git add .

git commit -m "initial commit"
```

```
commit 82f5ea346a2e651544956a8653c0f58dc151275c
Author: Alice <alice@example.com>
Date:   Thu Jan 1 00:00:00 1970 +0000

    Initial commit.
```

```
git add foo
git add bar/
```

```
git add foo
git add bar/
```

```
git rm for
git rm -r bar/
```

```
git add foo
git add bar/
```

```
git rm for
git rm -r bar/
```

```
git mv foo foobar
```

- “Bühne“ zwischen dem Arbeitsbereich und dem Repository

- “Bühne“ zwischen dem Arbeitsbereich und dem Repository
- git commit überträgt den Index

- “Bühne“ zwischen dem Arbeitsbereich und dem Repository
- git commit überträgt den Index
- git add, git rm –cached arbeiten nur im Index

- “Bühne“ zwischen dem Arbeitsbereich und dem Repository
- git commit überträgt den Index
- git add, git rm –cached arbeiten nur im Index
- einzelne Änderungen (Dateien, Bereiche, Zeilen) commiten

- “Bühne“ zwischen dem Arbeitsbereich und dem Repository
- git commit überträgt den Index
- git add, git rm –cached arbeiten nur im Index
- einzelne Änderungen (Dateien, Bereiche, Zeilen) commiten
- git commit -a führt automatisch git add/git rm aus und commited dann den Index

```
git log
```

git log

```
commit 766f9881690d240ba334153047649b8b8f11c664
Author: Bob <bob@example.com>
Date:   Tue Jan 1 13:37:00 1970 +0000

    Replace something.

commit 82f5ea346a2e651544956a8653c0f58dc151275c
Author: Alice <alice@example.com>
Date:   Thu Jan 1 00:00:00 1970 +0000

    Initial commit.
```

```
git reset [--mixed] [HEAD]
git reset --soft HEAD^
git reset --hard
```

```
git reset [--mixed] [HEAD]
git reset --soft HEAD^
git reset --hard
```

```
git checkout 82f5
git checkout master
```

```
git chechout -b new-branch [HEAD]
```

```
git chechout -b new-branch [HEAD]
```

```
git branch
```

```
git chechout master
git merge new-branch
```

```
git chechout master
git merge new-branch
```

```
git mergetool
```

- Austausch von Revisionen nötig (Zusammenarbeit)

- Austausch von Revisionen nötig (Zusammenarbeit)
- git clone *URL*
  kopiert ein ganzes Repository inklusive kompletter Geschichte

- Austausch von Revisionen nötig (Zusammenarbeit)
- git clone *URL*
  kopiert ein ganzes Repository inklusive kompletter Geschichte
- git fetch *URL* git pull *URL*
  aktualisiert die lokalen Revisionen des Repositories (pull merged diese zusätzliche mit den lokalen Branches)

- Austausch von Revisionen nötig (Zusammenarbeit)
- git clone *URL*
  kopiert ein ganzes Repository inklusive kompletter Geschichte
- git fetch *URL* git pull *URL*
  aktualisiert die lokalen Revisionen des Repositories (pull merged diese zusätzliche mit den lokalen Branches)
- git push *URL*
  aktualisiert das entfernte Repository, dabei muss das entfernte Repository ein Vorfahre des lokalen sein (*fast-forward*)

- Remote Repositories (URL bei clone wird zu origin)

- Remote Repositories (URL bei clone wird zu origin)
- tracking branches (Verknüpfung von Branches verschiedener Repositories)

- Remote Repositories (URL bei clone wird zu origin)
- tracking branches (Verknüpfung von Branches verschiedener Repositories)
- Git hat keine Authentifizierung

- Remote Repositories (URL bei clone wird zu origin)
- tracking branches (Verknüpfung von Branches verschiedener Repositories)
- Git hat keine Authentifizierung
- git:// Protokoll nur lesender Zugriff

- Remote Repositories (URL bei clone wird zu origin)
- tracking branches (Verknüpfung von Branches verschiedener Repositories)
- Git hat keine Authentifizierung
- git:// Protokoll nur lesender Zugriff
- für schreibenden Zugriff ssh:// (oder auch WebDAV)

- Remote Repositories (URL bei clone wird zu origin)
- tracking branches (Verknüpfung von Branches verschiedener Repositories)
- Git hat keine Authentifizierung
- git:// Protokoll nur lesender Zugriff
- für schreibenden Zugriff ssh:// (oder auch WebDAV)
- Koordination oft über Patches (git format-patch, git send-email, git am)

- Submodule
  Unterordner von eine Repository können auf andere Repositories zeigen.

- Submodule
  Unterordner von eine Repository können auf andere Repositories zeigen.
- Reflog
  Revisionen die nicht in der aktuellen Geschichte sind, werde im reflog aufgezeichnet.

- Submodule
  Unterordner von eine Repository können auf andere Repositories zeigen.
- Reflog
  Revisionen die nicht in der aktuellen Geschichte sind, werde im reflog aufgezeichnet.
- Bisect
  Binäre Suche nach der Revision die einen bestimmten Bug auslöst.

- Submodule
  Unterordner von eine Repository können auf andere Repositories zeigen.
- Reflog
  Revisionen die nicht in der aktuellen Geschichte sind, werde im reflog aufgezeichnet.
- Bisect
  Binäre Suche nach der Revision die einen bestimmten Bug auslöst.
- Stash
  Index und lokale Änderungen werden gesichert und auf für spätere Verwendung gespeichert. (Danach sind keine lokal Änderungen mehr im Verzeichnis)