Arbeiten mit dem Versionsverwaltungssystem

Jannik Kramer

Proseminar Werkzeuge für das wissenschaftliche Arbeiten

14.07.2014

Gliederung

Versionsverwaltung

- Erfassung und Protokollierung aller Änderungen
- Gleichzeitige Entwicklung durch mehrere Personen
- Repository als Datenbank
- Zeitlich geordnete Versionen als Historie

Zentrale Versionsverwaltung

- Client-Server-System
  - Repository liegt auf einem zentralen Server
  - Leicht zu administrieren
- „Single Point of Failure“
- Lock-Modify-Write Arbeitsweise

Dezentrale Versionsverwaltung

- Benutzer besitzt eigenes Repository
  - Ein offizielles Repositor
- Copy-Modify-Merge Arbeitsweise
  - Ermöglicht Verzweigung
  - Erfordert Zusammenführen
- Kein Konflikt bei simultaner Änderung

Git

- 2005 von Linus Torvalds zur Quellcodeverwaltung des Linux Kernel entwickelt
- Fähig große Projekte zu verwalten
- Keine Network Latency Overhead
  - Sehr schnell
- Vollständige Projekthistory in der lokalen Datenbank

Git

- Vier wichtige Transportprotokolle
  - Lokal
  - SSH
  - Git
  - HTTP

- Tools

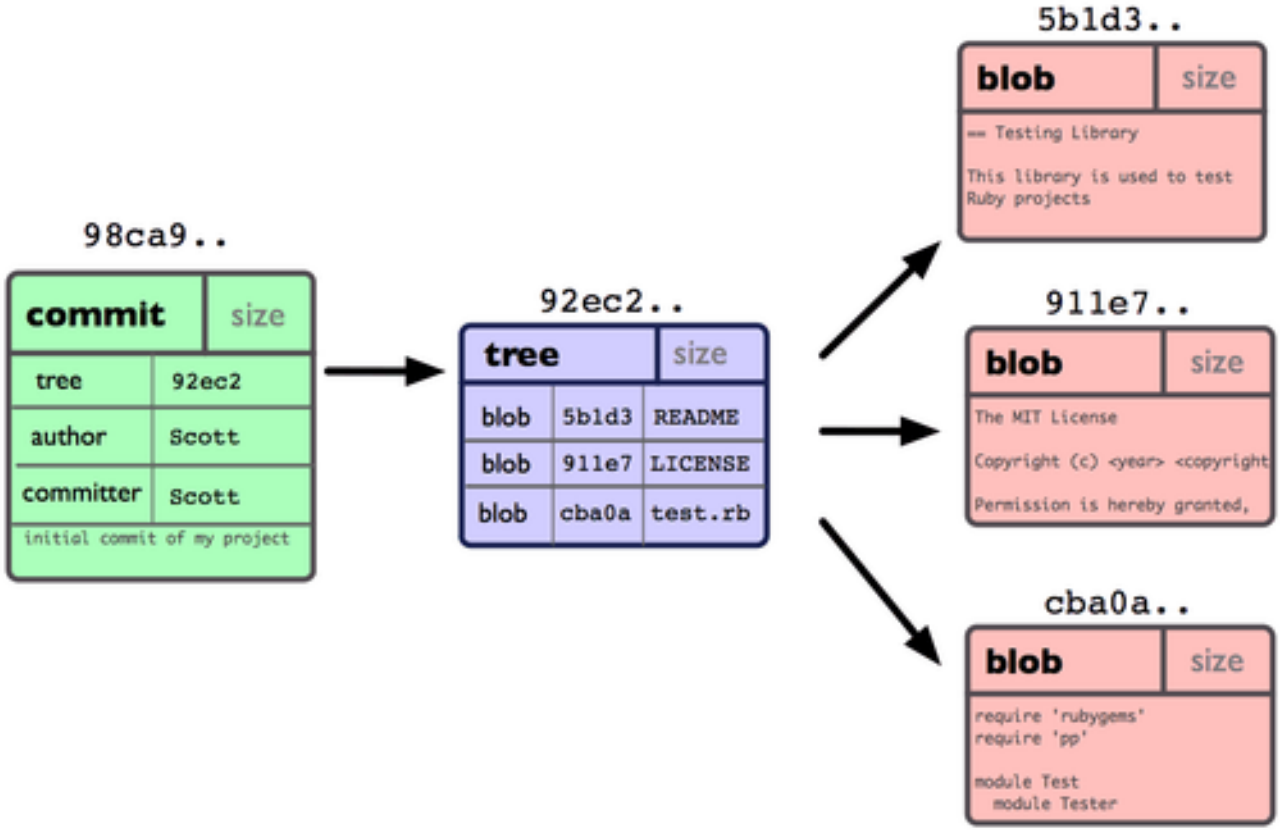
Hauptbestandteile

- Workspace
- Index
- Repository
  - Local repository
  - Remote repository

Git-Objekte

- Commit-Objekt
- Dateien „Blobs“
- Bäume „tree-Objekt“
  - Inhalt des Verzeichnisses
  - Spezifiziert die Zuordnung Dateiname und Blob
- tags

Git-Objekte



http://git-scm.com/figures/18333fig0301-tn.png

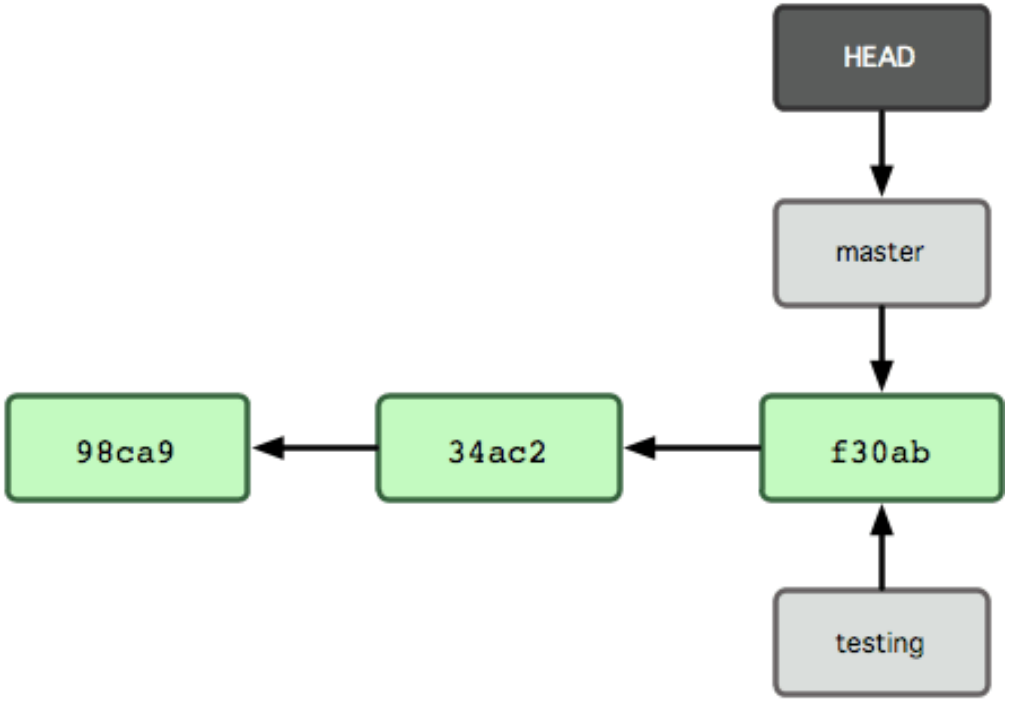
SHA1-Hash

- Änderungen in Prüfsumme umrechnen
- Erstellung beim stagen
- Sicherheit
  - Keine unbemerkten Änderungen
  - Änderungen können nicht verloren gehen

Git-Branches

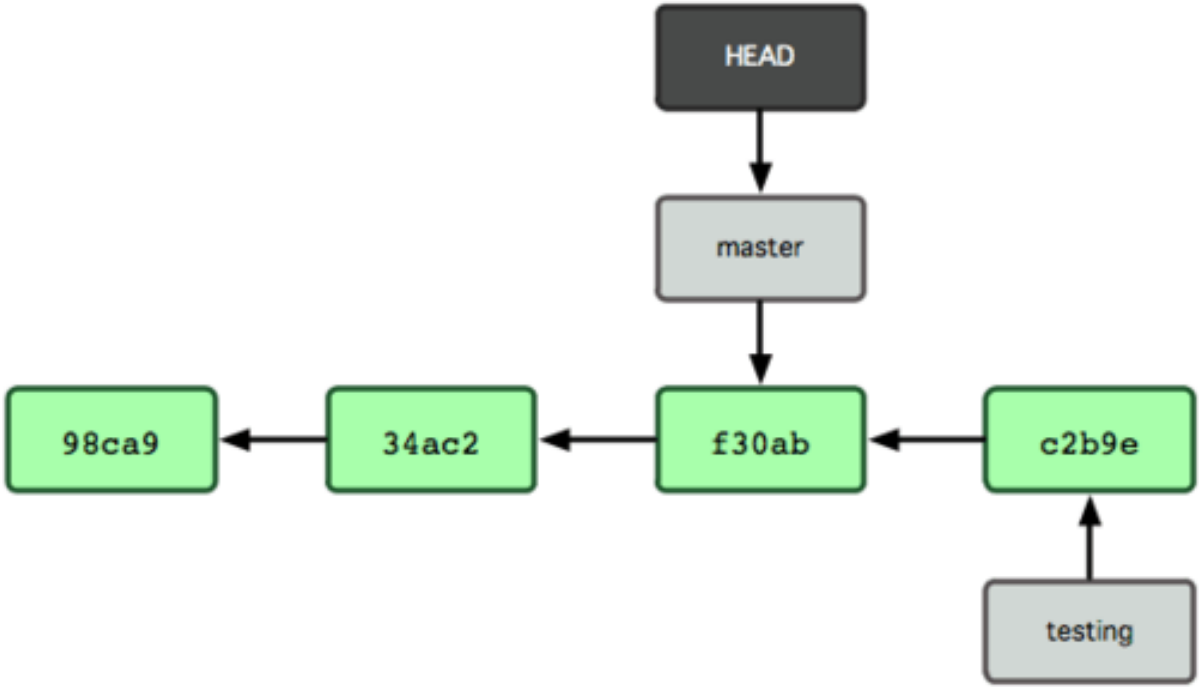
- Branches: Entwicklungszweige
- Textdatei mit einer Commit ID und SHA-1 Prüfsumme
  - schnelle Erstellung neuer Branches
  - gemeinsame Basis dank Ursprungscommit
- Eltern Commits -> Baumstruktur
- HEAD
  - Verweist auf den aktuell benutzten Branch
  - Unterschied Subversion

Git-Branches



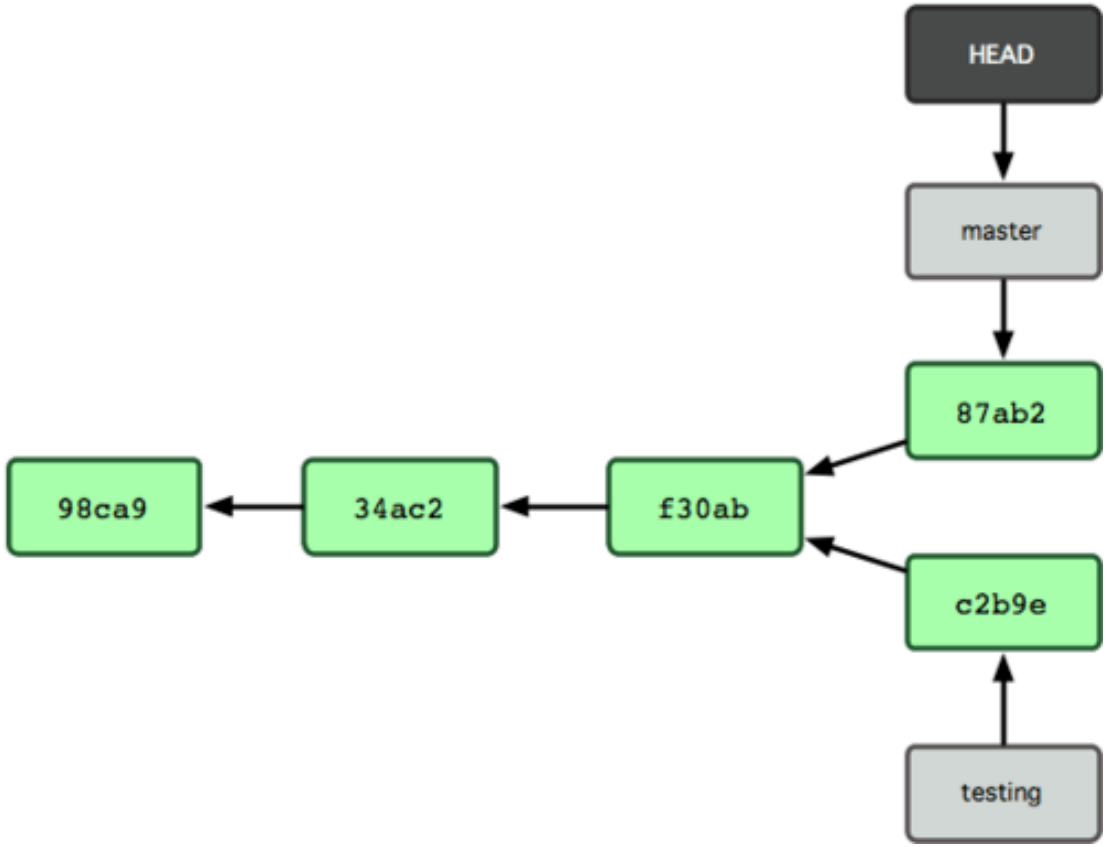
http://git-scm.com/figures/18333fig0305-tn.png

Git-Branches



http://git-scm.com/book/de/Git-Branching-Was-ist-ein-Branch%3F

Git-Branches



http://git-scm.com/book/de/Git-Branching-Was-ist-ein-Branch%3F

Git definiert drei Dateizustände

Commited: „gesichert “

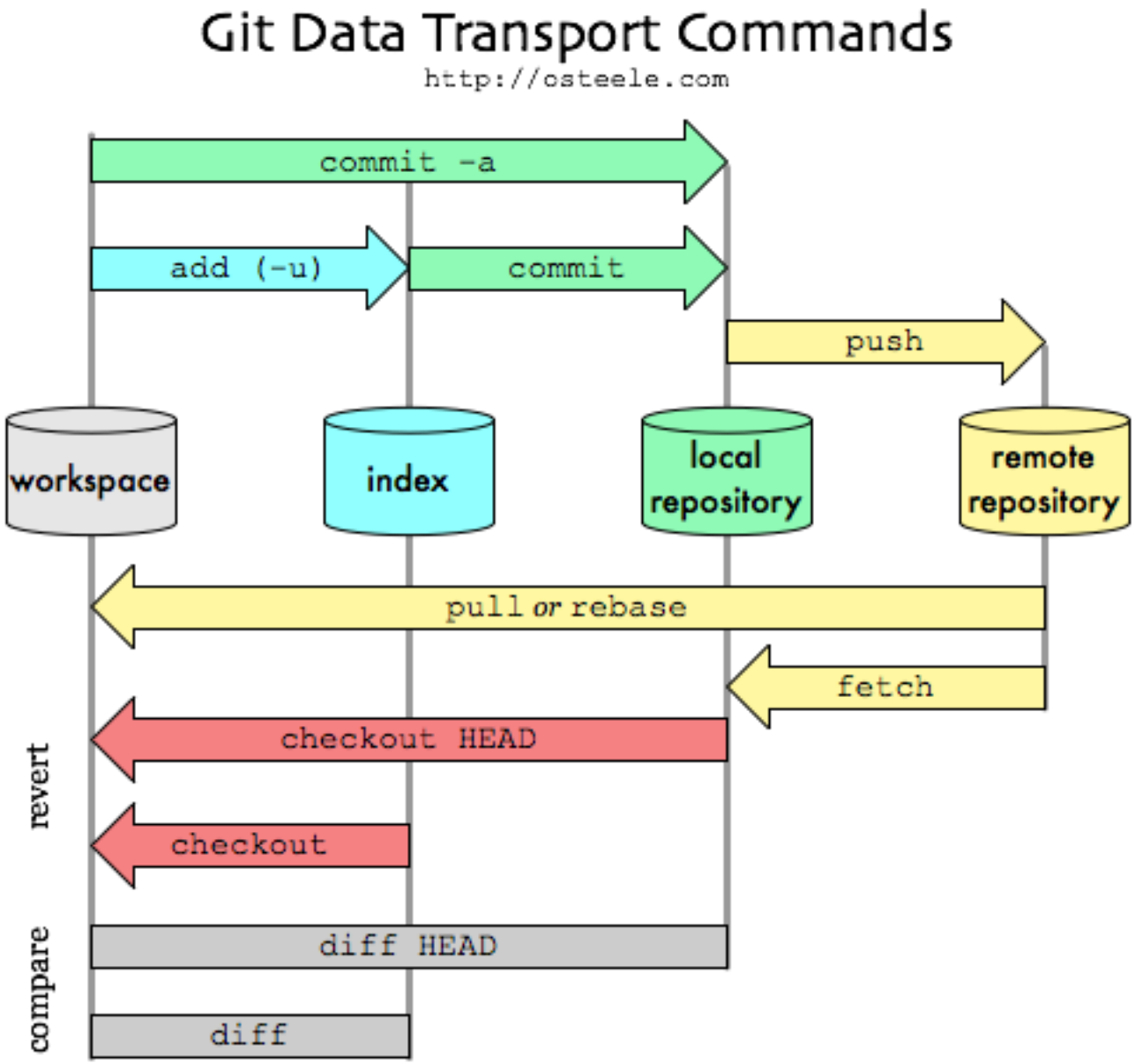
Modified: „verändert“

Staged: „Zum Index hinzugefügt“

Git Arbeitsprozess

- Bearbeitung der Dateien im Arbeitsverzeichnis
- Markierung der Dateien/Hinzufügen zum Index für den Commit
- Anlegung des Commit, Snapshots werden im Git Verzeichnis gespeichert

Funktionsweise



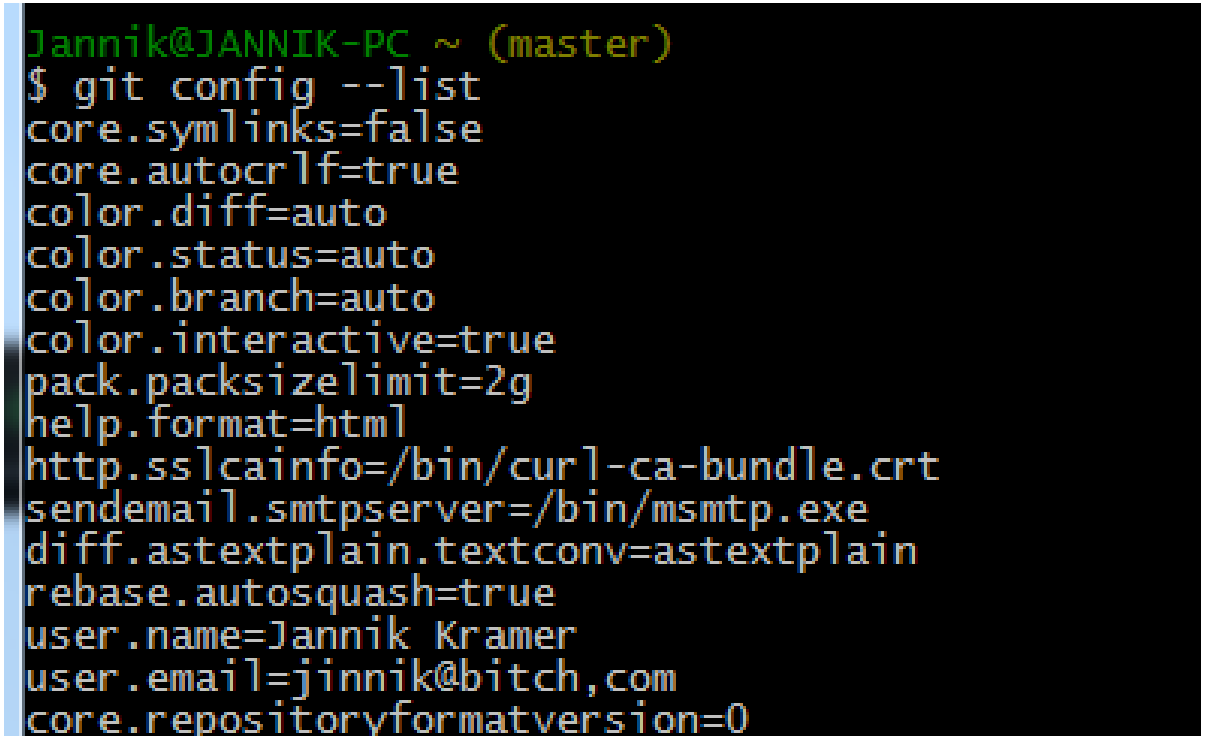
Identität

```
Jannik@JANNIK-PC ~ (master)
$ git config --global user.name "Jannik Kramer"
```

Befehl:

```
$ git config --global user.name "John Doe“
$ git config --global user.email jdoe@example.com
```

Einstellungen anzeigen



Befehl:

```
$ git config --list
```

Repository anlegen

```
Jannik@JANNIK-PC ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/Jannik/.git/
```

Befehl:

```
$ git init
```

Dateien zum Index hinzufügen

```
Jannik@JANNIK-PC ~/gidtest (master)
$ git add datei.txt
warning: LF will be replaced by CRLF in datei.txt.
The file will have its original line endings in your working directory.
```

Befehl:

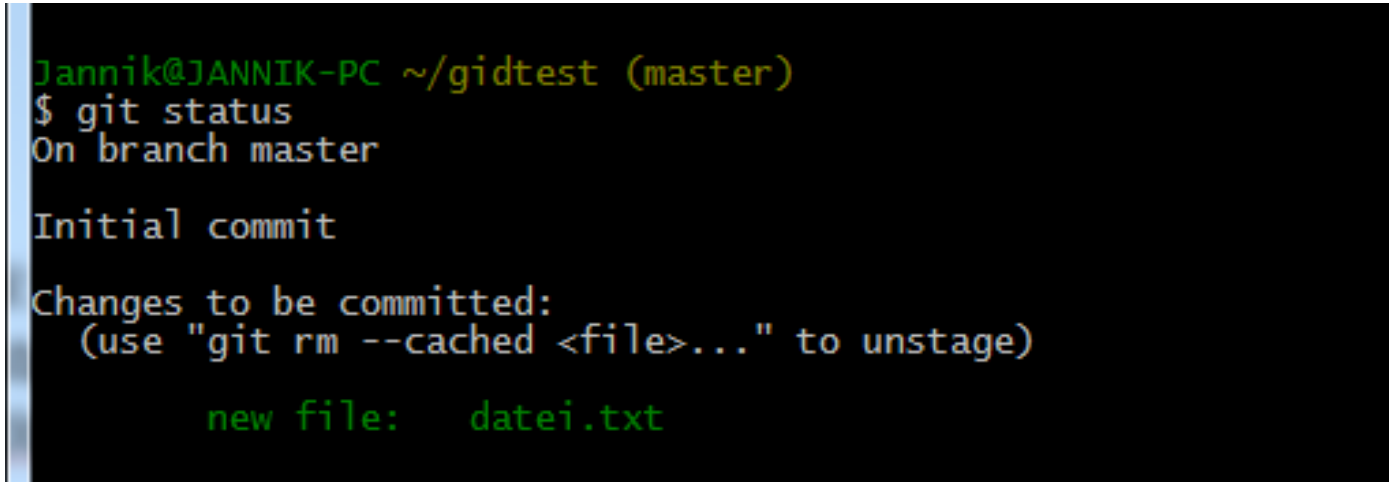
```
$ git add <dateiname>
```

Repository kopieren

Befehl:

```
$ git clone <repository>
$ git clone git://github.com/schacon/grit.git
```

Zustand prüfen



Befehl:

```
$ git status
```

Branching

Einen Neuen Branch erstellen und aktivieren

```
$ git branch <branchname>
```

```
$ git checkout <branchname>
```

Branching

Den Branch wechseln:

```
$ git checkout <branchname>
```

Merging

Branches zusammenführen

```
$ git merge <branchname>
```

Zusammengeführte Branches anzeigen:

```
$ git branch --merged
```

Externe Repository

Anzeigen der konfigurierten Server

```
$ git remote
```

Beispiel:

```
$ git remote -v
name1      git://github.com/name1/grit.git
name2      git://github.com/name2/grit.git
name3      git://github.com/name3/grit.git
```

Externe Repository anlegen

```
$ git remote add <shortname> <url>
```

Beispiel:

```
$ git remote add jo git://github.com/jojo/jannik.git
```

Änderungen herunterladen

```
$ git pull <remotename>
```

Einzelne Branches

```
$ git pull <remonatename> <branchname>
```

Änderungen herunterladen

```
$ git pull origin master
  remote: Counting objects: 7, done.
  remote: Compressing objects: 100% (4/4), done.
  remote: Total 4 (delta 2), reused 0 (delta 0)
  Updating 361303d..f2cd831
  Fast forward
   _layouts/default.html |    1 +
    1 files changed, 1 insertions(+),0 deletions
```

Änderungen hochladen

```
$ git push <remotename> <branchname>
```

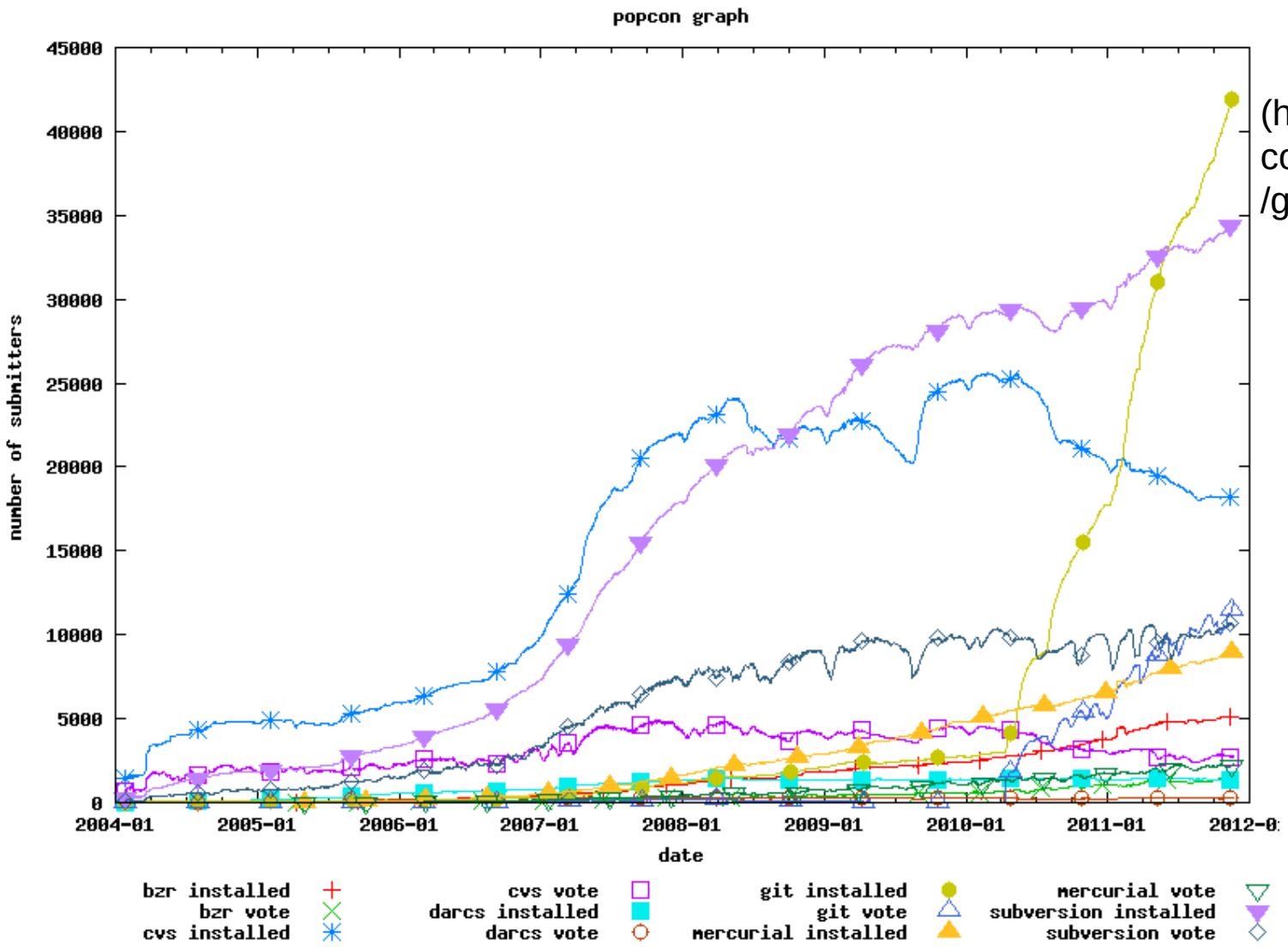
Beispiel

```
$ git push origin master
  Counting objects: 7, done.
  Compressing objects: 100% (4/4), done.
  Writing objects: 100% (4/4), 407 bytes, done.
  Total 4 (delta 2), reused 0 (delta 0)
  To git@github.com:qrush/gitready.git
     361303d..f2cd831  master -> master
```

Ausblick

- Open Source Projekte benutzen DVCS
  - Android, Linux-Kernel, VLC Media Player
- Es existieren Hosting-Dienste
  - GitHub
  - BitBucket
- Steigende Userzahlen

Ausblick



(http://aniszczyk.org/wp-content/uploads/2011/11/gitpopcondebian1.png)

Zusammenfassung

- Einfache, effiziente Arbeitsweise
  - Große Projekte
- Kein zentraler Server benötigt
  - Entwickler laden das komplette Projekt in die lokale Umgebung
- Unterstützung vieler Übertragungsprotokolle
- Gute Unterstützung von nicht-linearer Entwicklung

Quellen

http://aniszczyk.org

http://git-scm.com/

http://de.gitready.com/

http://www.online-tutorials.net/programmierung/git/tutorials-t-3-263.html

http://wr.informatik.uni-hamburg.de/_media/teaching/wintersemester_2011_2012/git-workshop-2011.pdf