

Versionsverwaltung mit git

Christoph Mallon

31. Juli 2012

Konfiguration: Wer bin ich?

```
%git config --global user.name 'Christoph Mallon'  
%git config --global user.email 'mallon@cs.uni-saarland.de'
```

- ▶ Autoren- und Committerangabe in Commits.
- ▶ `--global`: Globale Konfiguration (`~/.gitconfig`).
- ▶ Ohne `--global`: Depot-lokale Konfiguration (`.git/config`).

git init: Depot anlegen

```
%git init hello  
%cd !$
```

- ▶ Neues Depot mit Arbeitskopie in Verzeichnis `hello` anlegen.
- ▶ In aktuelles Verzeichnis, falls kein Pfad angegeben.

git add: Dateien hinzufügen

```
%vi hello.c
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    puts("hello, world");
```

```
    return 0;
```

```
}
```

```
%git add hello.c
```

- ▶ Merkt den *aktuellen Inhalt* von `hello.c` zum Einbuchen vor.
- ▶ Stand wird im *Index* abgelegt (auch *cache* oder *staging area*).
- ▶ `git add -A`: Stand *aller* nicht ignorierten (später) Dateien vormerken.
- ▶ `git add -u`: Stand aller zuvor hinzugefügten Dateien vormerken.
- ▶ `git add -p`: Bei jedem Änderungsblock zum Vormerken nachfragen. (sehr praktisch!)

git commit: Einbuchen

```
%git commit -m 'Implement the famous hello-world program.'  
[master d7ef55b] Implement the famous hello-world program.  
1 file changed, 7 insertions(+)  
create mode 100644 hello.c
```

- ▶ *Vorgemerkten Stand* in Commit verwandeln.
- ▶ Ohne `-m`: Editor zum Eingeben des Logeintrags.
- ▶ `-a`: Automatisch vorher `git add -u` ausführen.
- ▶ Erstes Einbuchen erzeugt automatisch Zweig namens `master`.
- ▶ Hexzahl ist Anfang des internen Bezeichners des Commits.
(später mehr dazu)

git diff: Änderungen anschauen

```
%vi Makefile
```

```
all: hello
```

```
%git add Makefile
```

```
%git diff
```

- ▶ Keine Ausgabe: Kein Unterschied zwischen Arbeitskopie und Index.
- ▶ Vorgemerkte Änderungen anzeigen mittels `git diff --cached:`

```
%git diff --cached
diff --git a/Makefile b/Makefile
new file mode 100644
index 0000000..b8937ca
--- /dev/null
+++ b/Makefile
@@ -0,0 +1 @@
+all: hello
```

git diff: Änderungen anschauen

```
%vi Makefile
```

```
.PHONY: all
```

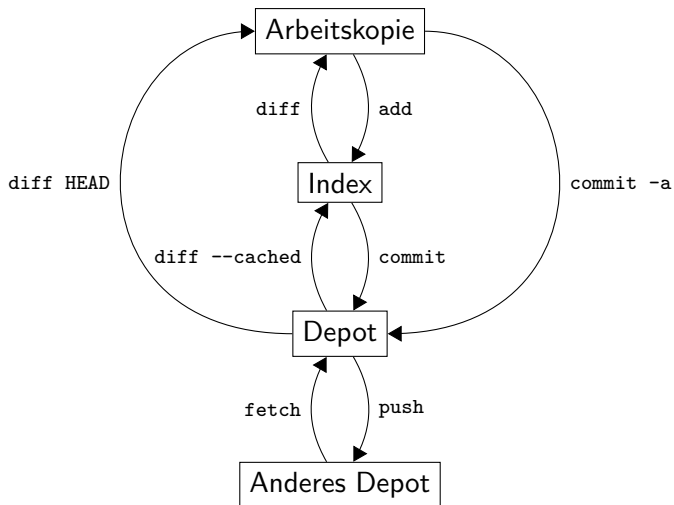
```
all: hello
```

```
%git diff
```

```
diff --git a/Makefile b/Makefile
index b8937ca..55d0233 100644
--- a/Makefile
+++ b/Makefile
@@ -1,3 @@
+.PHONY: all
+
+all: hello
```

- ▶ `git commit` bucht nur vorgemerkte Änderungen ein!
- ▶ `git add`, um neueren Stand vorzumerken.
- ▶ oder `git commit -a`, um alle Änderungen einzubuchen.

Stufendiagramm



git status: Stand von Arbeitskopie und Index

```
%git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
# new file:   Makefile
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
#
# modified:   Makefile
#
#
```

- ▶ Makefile hat Änderungen, die bereits zum Einbuchen vorgemerkt wurden.
- ▶ Es gibt weitere Änderungen, die noch nicht vorgemerkt wurden.

git status -s: Kurzversion

```
%git status -s
AM Makefile
%git commit -a -m 'Add a Makefile.'
[master 5dc8ef4] Add a Makefile.
 1 file changed, 3 insertions(+)
 create mode 100644 Makefile
%git status -s
%
```

- ▶ Kurzversion des Status. (ähnlich zu `svn status`)
- ▶ Erste Spalte: Index relativ zu Depot.
- ▶ Zweite Spalte: Arbeitskopie relativ zu Index.
- ▶ A: Neu.
- ▶ M: Geändert.
- ▶ D: Gelöscht.
- ▶ R: Umbenannt.
- ▶ c: Kopiert.
- ▶ U: Konflikt.
- ▶ ?: Nicht vermerkt.

.gitignore: Dateien ignorieren

```
%make
cc      hello.c    -o hello
%git status -s
?? hello
%echo '/hello' > .gitignore
%git status -s
?? .gitignore
%git add -A
%git status -s
A .gitignore
%git commit -m 'Ignore the executable.'
[master 15e38ef] Ignore the executable.
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
```

- ▶ / am Anfang: Ignorieren relativ zu .gitignore.
- ▶ Kein / am Anfang: Auch in Unterverzeichnissen ignorieren.
- ▶ * und ? üblich verwendbar. (Z.B. *.log bei T_EX)
- ▶ Bereits vermerkte Dateien werden dadurch nicht ignoriert!

Woraus besteht ein Commit?

```
%git cat-file -p HEAD
tree 007b781f12147a9fcc5c980583e6c986fb172088
parent 5dc8ef436eca1b58344b26659f27a1804807803d
author Christoph Mallon <mallon@cs.uni-saarland.de> 1343594322 +0200
committer Christoph Mallon <mallon@cs.uni-saarland.de> 1343594322 +0200
```

Ignore the executable.

- ▶ Verweis auf den Stand des Verzeichnisbaums (tree).
- ▶ 0-N Verweise auf die Eltern-Commits (geordnete Liste).
 - ▶ 0: Erster Commit.
 - ▶ 1: Normaler Commit.
 - ▶ N: Commit zum Zusammenführen mehrerer Entwicklungszweige (üblich 2).
- ▶ Autor mit Zeitstempel.
- ▶ Einbuchender mit Zeitstempel.
- ▶ Logeintrag.
- ▶ Wird über SHA-1 dieser Daten identifiziert.

- ▶ `master`: Zweigname; verweist auf einen Commit.
- ▶ `HEAD`: Symbolische Referenz; verweist auf den Zweignamen des aktuellen Zweigs.

Wie benennt man Commits?

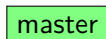
- ▶ SHA-1 oder eindeutiger Präfix davon.
- ▶ Symbolische Referenz.
- ▶ Zweigname, Markierungsname.
- ▶ Suffix $\sim N$: Nter Vorgänger (ohne Angabe $N=1$).
 - ▶ `HEAD \sim 2`: Vorgänger des Vorgängers des Commits, auf den der Zweigname zeigt, auf den `HEAD` zeigt.
- ▶ Suffix $\sim N$: Nter Eltern-Commit (bei Zusammenführungen; ohne Angabe $N=1$).
- ▶ $\sim N$ und $\sim N$ beliebig mehrfach verkettbar (selten benötigt).

Beispiel: Depotentwicklung

Legende:



Commit



Zweig



Markierung



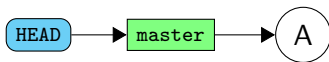
Symbolische Referenz

```
git init
```

Ein neues, leeres Depot.

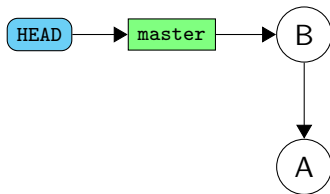
```
git commit -m 'A'
```

Erster Commit legt implizit Zweig `master` an.

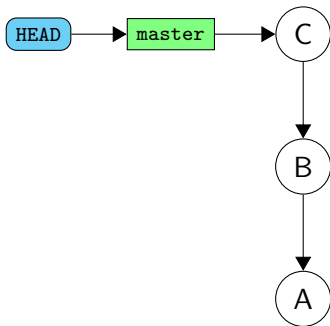



```
git commit -m 'B'
```

Jeder Commit verweist auf seine(n) Eltern-Commit(s).

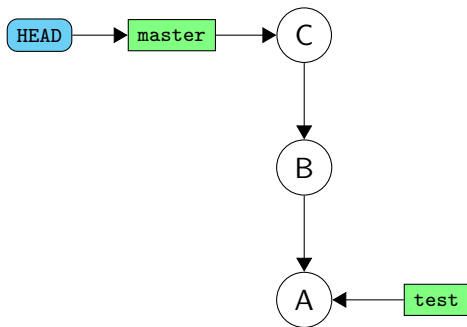


```
git commit -m 'C'
```



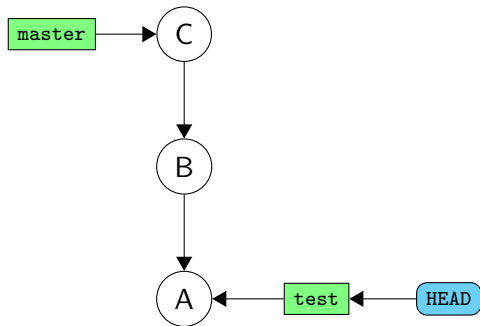
```
git branch test HEAD~2
```

Zweig anlegen: Ein Name, der auf einen Commit verweist.

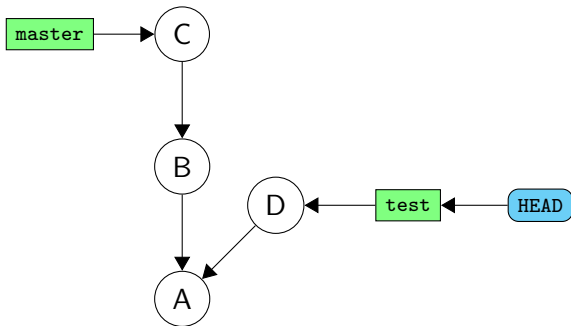


```
git checkout test
```

Zweig wechseln: HEAD weist auf aktuellen Zweig, Arbeitskopie entsprechend angepasst.

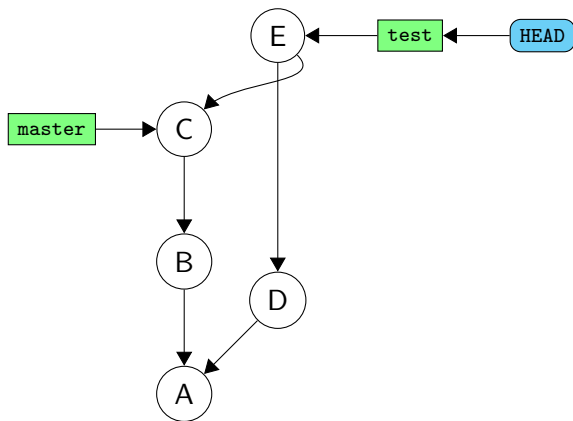


```
git commit -m 'D'
```



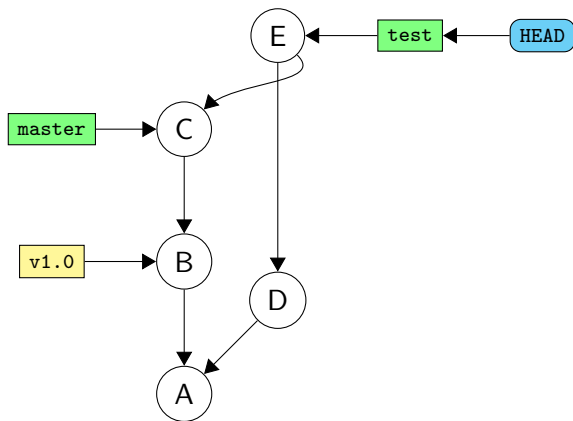
git merge master

Zusammenführen: E hat zwei Eltern.

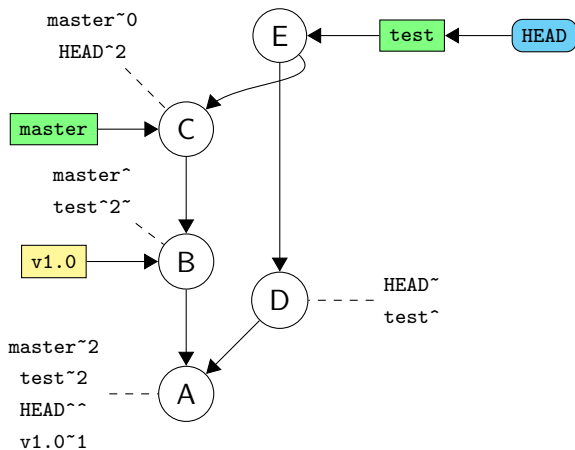


```
git tag v1.0 master~
```

Markierung anlegen: Ein Name, der auf einen Commit verweist.



Beispiele für Commit-Benennungen



git log: Das Logbuch

- ▶ `git log`: Logeinträge topologisch sortiert ab `HEAD`.
- ▶ Explizite Angabe von Commit (als Zweig, SHA-1, ...): Log ab diesem Punkt.
- ▶ `--name-status`: Grobe Angaben über Dateiänderungen (A, M, ...; ähnlich `svn log -v`).
- ▶ `--patch`: Änderungen zum jeweiligen Eltern-Commit anzeigen.
- ▶ `--graph`: Explizite Darstellung der Eltern-Relation als Linien.
- ▶ `--decorate`: Zeigt (symbolische) Referenznamen im Log an.
- ▶ `--oneline`: Nur erste Zeile jedes Logeintrags anzeigen.
- ▶ Graphische Version: `gitk`

git branch: Zweige verwalten

```
%git branch
* master
%git branch test HEAD~
%git branch
* master
  test
```

- ▶ Ohne Parameter: Zeigt alle lokalen Zweige.
 - ▶ * vor aktuell ausgebuchtem Zweig.
- ▶ `git branch name commit`: Legt Zweig `name` mit Stand `commit` an.
 - ▶ Ohne Angabe von `commit`: HEAD.
- ▶ `git branch -d name`: Löscht Zweig.
 - ▶ Darf nicht aktuell ausgebuchter Zweig sein.
- ▶ Zweig(namen) sind nur Verweise aus Commits.
 - ▶ Mehrere Zweige mit selbem Stand.
 - ▶ Zweig löschen löscht keine Commits, sondern nur den Zweignamen.

git checkout: Zweig wechseln

```
%git branch
* master
  test
%git log --decorate --oneline
83a7a86 (HEAD, master) Ignore the executable.
5dc8ef4 (test) Add a Makefile.
7d442eb Implement hello world.
%git checkout test
Switched to branch 'test'
%git branch
  master
* test
%git log --oneline
5dc8ef4 (HEAD, test) Add a Makefile.
7d442eb Implement hello world.
%git checkout master
Switched to branch 'master'
%git branch -d test
Deleted branch test (was 5dc8ef4).
```

- ▶ Wechselt zu angegebenen Zweig.
- ▶ Arbeitskopie muss unverändert zu bisherigem Stand sein.

git tag: Markierungen verwalten

```
%git tag
%git tag v1.0 master~
%git tag
v1.0
%git log --decorate --oneline
83a7a86 (HEAD, master) Ignore the executable.
5dc8ef4 (tag: v1.0) Add a Makefile.
7d442eb Implement hello world.
%git log --decorate --oneline v1.0
5dc8ef4 (tag: v1.0) Add a Makefile.
7d442eb Implement hello world.
%git tag -d v1.0
Deleted tag 'v1.0' (was 5dc8ef4)
```

- ▶ Markierungen ähnlich zu Zweigen: Verweis auf Commit.
- ▶ Unterschied: Kann nicht ausgebucht werden.
- ▶ `git tag`: Zeigt alle lokalen Markierungen.
- ▶ `git tag name commit`: Legt Markierung `name` mit Stand `commit` an.
 - ▶ Ohne Angabe von `commit`: HEAD.
- ▶ `git tag -d name`: Löscht Markierung.

git show: Einzelne Commits anschauen

```
% git show
commit 83a7a8699d0ad4170f72ea1914538518c542a46e
Author: Christoph Mallon <christoph.mallon@gmx.de>
Date:   Sun Jul 29 22:38:42 2012 +0200
```

Ignore the executable.

```
diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..e92569d
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1 @@
+/hello
```

- ▶ `git show commit`: Logeintrag und Änderungen von `commit` anzeigen.
 - ▶ Ohne Angabe von `commit`: HEAD.

Mist, Datei vergessen

TODO `git commit -amend`

Mist, Datei vergessen

TODO Bild zu `git commit -amend`

TODO git remote, git clone

TODO git fetch

TODO git merge, git pull

TODO git rebase, git pull -rebase

TODO Bild zu git fetch, git merge, git rebase

TODO git push, non-fast-forward

TODO Alias