Git □□□□□□□

Dieser Blogbeitrag wurde mit Hilfe von ChatGPT-4o erstellt. Die Keynote ist verfügbar unter https://github.com/lzwjava/Keynotes.

Rückgängig machen

```
$ git commit --amend
$ git add *
$ git status
$ git reset HEAD CONTRIBUTING.md
Unstaged changes after reset:
M CONTRIBUTING.md
$ git status
$ git checkout -- CONTRIBUTING.md
```

(Die Befehle und Ausgaben bleiben auf Englisch, da es sich um technische Anweisungen handelt, die in der Regel nicht übersetzt werden.)

- Änderungen erneut einreichen
- Staging von Dateien rückgängig machen
- Änderungen an Dateien rückgängig machen

Befehle

- `git revert`, Rückgängig machen, erneut commiten, um einen bestimmten vorherigen Commit zu neutralisieren
- `git cherry-pick`, Auswahl bestimmter Commits aus mehreren Commits
- `git rebase`

filter-branch

- Eine bestimmte Datei aus jedem Commit entfernen
- Sie möchten sie löschen, aber sie bleibt in der Historie
- Wie kann man eine Datei aus allen Commits entfernen?

```
$ git filter-branch --tree-filter 'rm -f passwords.txt' HEAD
Rewrite 6b9b3cf04e7c5686a9cb838c3f36a8cb6a0fc2bd (21/21)
Ref 'refs/heads/master' wurde umgeschrieben
```

Suchen

```
$ git grep -n gmtime_r
compat/gmtime.c:3:#undef gmtime_r
compat/gmtime.c:8:return git_gmtime_r(timep, &result);
compat/gmtime.c:11:struct tm *git_gmtime_r(const time_t *timep, struct tm *result)
compat/gmtime.c:16:ret = gmtime_r(timep, result);
compat/mingw.c:606:struct tm *gmtime_r(const time_t *timep, struct tm *result)
compat/mingw.h:162:struct tm *gmtime_r(const time_t *timep, struct tm *result);
date.c:429:if (gmtime_r(&now, &now_tm))
date.c:492:if (gmtime_r(&time, tm)) {
git-compat-util.h:721:struct tm *git_gmtime_r(const time_t *, struct tm *);
git-compat-util.h:723:#define gmtime_r git_gmtime_r
$ git log -SZLIB_BUF_MAX --oneline
e01503b zlib: allow feeding more than 4GB in one go
ef49a7a zlib: zlib can only process 4GB at a time
```

.Git-Verzeichnis

- `config` Projektkonfiguration
- `info` .gitignore
- `objects` alle Inhalte in der Git-Datenbank
- `refs` Zeiger auf die Branches
- `HEAD` Zeiger auf den aktuellen Branch
- `index` Informationen zum Staging-Bereich

Git-Objekte

```
$ git init test
Initialisiertes leeres Git-Repository in /tmp/test/.git/
$ cd test
$ find .git/objects
.git/objects
```

```
.git/objects/info
.git/objects/pack
$ find .git/objects -type f

$ echo 'test content' | git hash-object -w --stdin
d670460b4b4aece5915caf5c68d12f560a9fe3e4
$ find .git/objects -type f
.git/objects/d6/70460b4b4aece5915caf5c68d12f560a9fe3e4
```

- `hash-object`, der Befehl, um Daten im .git-Verzeichnis zu speichern
- `-w`, schreibt das Objekt, andernfalls wird nur der Schlüssel zurückgegeben
- `--stdin`, liest von der Standardeingabe
- d670..., 40-stellige Prüfsumme (Checksum)
- `cat-file`, das Schweizer Taschenmesser zum Anzeigen von Git-Objekten

```
$ git cat-file -p d670460b4b4aece5915caf5c68d12f560a9fe3e4
test content

$ echo 'version 1' > test.txt
$ git hash-object -w test.txt
83baae61804e65cc73a7201a7252750c76066a30
$ echo 'version 2' > test.txt
$ git hash-object -w test.txt
1f7a7a472abf3dd9643fd615f6da379c4acb3e3a
$ find .git/objects -type f
.git/objects/1f/7a7a472abf3dd9643fd615f6da379c4acb3e3a
.git/objects/83/baae61804e65cc73a7201a7252750c76066a30
.git/objects/d6/70460b4b4aece5915caf5c68d12f560a9fe3e4
$ git cat-file -p 83baae61804e65cc73a7201a7252750c76066a30 > test.txt
$ cat test.txt
version 1
$ git cat-file -p 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a > test.txt
$ cat test.txt
version 2
$ git cat-file -t 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a
blob
$ git cat-file -p master^{tree}
```

```
100644 blob a906cb2a4a904a152e80877d4088654daad0c859 README
100644 blob 8f94139338f9404f26296befa88755fc2598c289 Rakefile
040000 tree 99f1a6d12cb4b6f19c8655fca46c3ecf317074e0 lib
$ git cat-file -p 99f1a6d12cb4b6f19c8655fca46c3ecf317074e0
100644 blob 47c6340d6459e05787f644c2447d2595f5d3a54b simplegit.rb
Tree Objects

$ git update-index --add --cacheinfo 100644 \
83baae61804e65cc73a7201a7252750c76066a30 test.txt
update-index
```

(Der obige Codeblock bleibt unverändert, da es sich um einen Git-Befehl handelt, der nicht übersetzt werden sollte.)

- `update-index`Befehl zum Erstellen des Baums
- `--add`Index erstellen
- `--cacheinfo`Aus der Git-Datenbank lesen
- `100644`Dateimodus, für normale Dateien; 100755 für ausführbare Dateien; 120000 für symbolische Links
- 83baae, ist der vorherige Inhalt, Version 1
- `\`, eine Befehlszeile in zwei Zeilen aufgeteilt

write-tree

```
$ git write-tree
d8329fc1cc938780ffdd9f94e0d364e0ea74f579
$ git cat-file -p d8329fc1cc938780ffdd9f94e0d364e0ea74f579
100644 blob 83baae61804e65cc73a7201a7252750c76066a30 test.txt
$ git cat-file -t d8329fc1cc938780ffdd9f94e0d364e0ea74f579
tree
```

- `write-tree`, schreibt den Inhalt des Staging-Bereichs in ein Tree-Objekt.

read-tree

```
$ echo 'new file' > new.txt
$ git update-index test.txt
$ git update-index --add new.txt
```

```
$ git write-tree
0155eb4229851634a0f03eb265b69f5a2d56f341
$ git cat-file -p 0155eb4229851634a0f03eb265b69f5a2d56f341
100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt
100644 blob 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a test.txt
$ git read-tree --prefix=bak d8329fc1cc938780ffdd9f94e0d364e0ea74f579
$ git write-tree
3c4e9cd789d88d8d89c1073707c3585e41b0e614
$ git cat-file -p 3c4e9cd789d88d8d89c1073707c3585e41b0e614
040000 tree d8329fc1cc938780ffdd9f94e0d364e0ea74f579 bak
100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt
100644 blob 1f7a7a472abf3dd9643fd615f6da379c4acb3e3a test.txt
```

(Der obige Codeblock wurde nicht übersetzt, da er Befehle und Ausgaben enthält, die in der Regel nicht übersetzt werden.)

Commit-Objekte

```
$ echo 'first commit' | git commit-tree d8329f
d5ffe1aa4b7b089eee03dccea5e0439ad6d72037
$ git cat-file -p d5ffe1aa4b7b089eee03dccea5e0439ad6d72037
tree d8329fc1cc938780ffdd9f94
```

e0d364e0ea74f579 author lzwjava lzwjava@gmail.com 1462090215 +0800 committer lzwjava lzwjava@gmail.com 1462090215 +0800 erster Commit

```
- Der Hash-Wert wird unterschiedlich sein, da sowohl das Erstellungsdatum als auch der Autor unterschie
```

$ echo ‘zweiter Commit’ | git commit-tree 0155eb -p d5ffe1aa4 e946d6367f07de45cac242dca7cd002f5eaa72
$ echo ‘dritter Commit’ | git commit-tree 3c4e9c -p e946d6 09490bf051c34b3693dfd5c7fb63dfe27b295904
$ git log –stat 09490 commit 09490bf051c34b3693dfd5c7fb63dfe27b295904 Author: lzwjava lzwjava@gmail.com Date: Sun May 1 16:38:55 2016 +0800 dritter Commit bak/test.txt | 1 + 1 Datei geändert, 1 Einfügung(+) commit e946d6367f07de45cac242dca7cd002f5eaa72b1 Author: lzwjava lzwjava@gmail.com Date: Sun May 1 16:37:01 2016 +0800 zweiter Commit new.txt | 1 + test.txt | 2 +- 2 Dateien geändert, 2 Einfügungen(+), 1 Löschung(-) commit d5ffe1aa4b7b089eee03dccea5e0439ad6d72037 Author: lzwjava lzwjava@gmail.com Date: Sun May 1 16:10:15 2016 +0800 erster Commit test.txt | 1 + 1 Datei geändert, 1 Einfügung(+)

- `commit-tree`, der erste Parameter verweist auf einen Blob oder Tree
- `-p` verweist auf den Eltern-Knoten

dritter Commit zweiter Commit erster Commit Baum Baum Baum „Version 2“ „neue Datei“ „Version 1“ 09490b e946d6 d5ffe1 d8329f 0155eb 3c4e9c 83baae fa49b0 1f7a7a test.txt new.txt test.txt new.txt test.txt bak

Das Prinzip von Branches

$ find .git/refs
.git/refs .git/refs/heads .git/refs/tags $ find .git/refs -type f
$ echo “09490bf051c34b3693dfd5c7fb63dfe27b295904” > .git/refs/heads/master $ git log –pretty=oneline master 09490bf051c34b3693dfd5c7fb63dfe27b295904 dritter Commit e946d6367f07de45cac242dca7cd002f5eaa72b1 zweiter Commit d5ffe1aa4b7b089eee03dccea5e0439ad6d7
erster Commit $ git update-ref refs/heads/master 09490bf051c34b3693dfd5c7fb63dfe27b295904
$ git update-ref refs/heads/test e946d6367f07de45cac242dca7cd002f5eaa72b1 $ git branch
* master test

- Einen Zeiger-Referenz speichern
- `update-ref`, um Referenzen zu ändern oder hinzuzufügen

dritter Commit zweiter Commit erster Commit Baum Baum Baum „Version 2“ „neue Datei“ „Version 1“ 09490b e946d6 d5ffe1 d8329f 0155eb 3c4e9c 83baae fa49b0 1f7a7a test.txt new.txt test.txt new.txt test.txt bak refs/heads/master refs/heads/test

$ cat .git/HEAD ref: refs/heads/master $ git symbolic-ref HEAD refs/heads/master $ git symbolic-ref HEAD refs/heads/test $ cat .git/HEAD ref: refs/heads/test

symbolic-ref

Tag-Prinzip

$ git update-ref refs/tags/v1.0 e946d6367f07de45cac242dca7cd002f5eaa72b1 $ git tag v1.0 $ git tag -a v1.1 e946d6367f07de45cac242dca7cd002f5eaa72b1 -m ‘test tag’ $ cat .git/refs/tags/v1.1 2766532f03289bc5e158629a8b3faffa5f80b8b6 $ git cat-file -p 276653 object e946d6367f07de45cac242dca7cd002f5eaa72b1 type commit tag v1.1 tagger lzwjava lzwjava@gmail.com 1462103203 +0800 test tag

remotes

$ git remote add origin git@github.com:schacon/simplegit-progit.git $ git push origin master Zähle Objekte: 11, fertig. Komprimiere Objekte: 100% (5/5), fertig. Schreibe Objekte: 100% (7/7), 716 Bytes, fertig. Gesamt 7 (Delta 2), Wiederverwendet 4 (Delta 1) An git@github.com:schacon/simplegit-progit.git a11bef0..ca82a6d master -> master $ cat .git/refs/remotes/origin/master ca82a6dff817ec66f44342007202690a93763949

- Im Verzeichnis `refs/remotes` gespeichert
- Der Unterschied zwischen Remote-Referenzen und Branches besteht darin, dass sie schreibgeschützt sind
- Man kann `git checkout` verwenden, aber der HEAD bleibt unverändert, daher können Remote-Referenzen ni

$ curl https://raw.githubusercontent.com/mojombo/grit/master/lib/grit/repo.rb > repo.rb $ git checkout master $ git add repo.rb $ git commit -m ‘added repo.rb’ [master 484a592] added repo.rb 3 Dateien geändert, 709 Einfügungen(+), 2 Löschungen(-) delete mode 100644 bak/test.txt create mode 100644 repo.rb rewrite test.txt (100%) $ git cat-file -p master^{tree} 100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt 100644 blob 033b4468fa6b2a9547a70d88d1bbe8bf3f9ed0d5 repo.rb 100644 blob e3f094f522629ae358806b17daf78246c27c007b test.txt $ git cat-file -s 033b4468fa6b2a9547a70d88d1bbe8 22044 $ echo ‘# testing’ » repo.rb $ git commit -am ‘modified repo a bit’ [master 2431da6] modified repo.rb a bit 1 Datei geändert, 1 Einfügung(+) $ git cat-file -p master^{tree} 100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt 100644 blob b042a60ef7dff760008df33cee372b945b6e884e repo.rb 100644 blob e3f094f522629ae358806b17daf78 test.txt $ git cat-file -s b042a60ef7dff760008df33cee372b945b6e884e 22054 $ git gc Zähle Objekte: 18, fertig. Delta-Kompression mit bis zu 8 Threads. Komprimiere Objekte: 100% (14/14), fertig. Schreibe Objekte: 100% (18/18), fertig. Gesamt 18 (delta 3), wiederverwendet 0 (delta 0) $ find .git/objects -type f $ git verify-pack -v .git/objects/pack/pack-978e03944f5c581011e6998cd0e9e30000905586.idx

Refspec

$ git remote add origin https://github.com/schacon/simplegit-progit [remote “origin”] url = https://github.com/schacon/simplegit-progit fetch = +refs/heads/*:refs/remotes/origin/* $ git log origin/master $ git log remotes/origin/master $ git log refs/remotes/origin/master fetch = +refs/heads/master:refs/remotes/origin/master [remote “origin”] url = https://github.com/schacon/simplegit-progit fetch = +refs/heads/master:refs/remotes/origin/master fetch = +refs/heads/qa/*:refs/remotes/origin/qa/* push = refs/heads/master:refs/heads

`/qa/master`

- `<src>:<dst>`
- `refs/remotes` Lokaler Speicherort

`https://github.com/schacon/simplegit-progit`

`https://github.com/schacon/simplegit-progit`

Referenzen

- Pro Git