□□□ उन्नत कार्यप्रणाली और सिद्धांत

यह ब्लॉग पोस्ट □□□□□□□□-4□ की सहायता से तैयार की गई है। □□□□□□□□ □□□□□://□□□□□□□.□□□/□□□□□□□□/□□□□□□□□ पर उपलब्ध है।

पूर्ववत करें

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

- □ फिर से सबमिट करें
- □ स्टेज किए गए फ़ाइलों को वापस लें
- □ फ़ाइलों में किए गए परिवर्तनों को वापस लें

कमांड

- □ `git revert` - यह कमांड किसी पिछले □□□□□□□ को पलटने के लिए उपयोग किया जाता है। यह एक नया □□□□□□□ बनाता है जो पिछले □□□□□□□ के परिवर्तनों को रद्द करता है।
- □ `git cherry-pick` - यह कमांड आपको कई □□□□□□□□ में से किसी एक विशेष □□□□□□□ को चुनकर अपनी वर्तमान □□□□□□□ में लाने की अनुमति देता है।
- □ `git rebase` - यह कमांड आपकी □□□□□□□ के □□□□□□□□ को दूसरी □□□□□□□ के □□□□□□□□ के ऊपर ले जाने के लिए उपयोग किया जाता है। यह आपकी □□□□□□□ □□□□□□□□ को साफ और सरल बनाने में मदद करता है।

इन कमांड्स का उपयोग करके आप अपने □□□ □□□□□□□□□□ में बदलावों को प्रभावी ढंग से प्रबंधित कर सकते हैं।

□□□□□□□-□□□□□□□

- □ हर कमिट से किसी फ़ाइल को हटाना
- □ इसे हटाना चाहते हैं, लेकिन फिर भी इतिहास में रखना चाहते हैं

- सभी कमिट्स से एक फ़ाइल को कैसे हटाएं?

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

खोजें

```
$ 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
```

.□□□ डायरेक्टरी

- □□□□□□□, प्रोजेक्ट कॉन्फ़िगरेशन
- □□□□□, .□□□□□□□□□□
- □□□□□□□□, □□□ डेटाबेस में सभी सामग्री
- □□□□□, शाखाओं के पॉइंटर्स
- □□□□□, वर्तमान शाखा का पॉइंटर
- □□□□□□, स्टेजिंग एरिया की जानकारी

□□□ ऑब्जेक्ट्स

```
$ git init test
   Git       /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` - यह कमांड डेटा को .□□□ डायरेक्टरी में सहेजता है।
- `-w` - ऑब्जेक्ट को लिखें, अन्यथा केवल □□□ वापस करें।
- `--stdin` - स्टैंडर्ड इनपुट से डेटा पढ़ें।
- □670... - 40 अक्षरों का □□□□□□□□□।
- `cat-file` - □□□ ऑब्जेक्ट को देखने के लिए स्विस आर्मी चाकू (सभी काम करने वाला टूल)।

```
$ git cat-file -p d670460b4b4aece5915caf5c68d12f560a9fe3e4


$ 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
```

(यह कोड ब्लॉक है, इसे अनुवादित नहीं किया जाना चाहिए।)

- update-index पेड़ बनाने का कमांड
- --add इंडेक्स बनाएं
- --cacheinfo डेटाबेस से पढ़ें
- 100644 फ़ाइल मोड, सामान्य फ़ाइल के लिए; 100755 एक्ज़ीक्यूटेबल फ़ाइल के लिए; 120000 सिंबलिक लिंक के लिए
- 83, पिछला कंटेंट, वर्जन 1
- \ एक कमांड को दो लाइनों में विभाजित करें

-

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

- write-tree की सामग्री को

-

```
$ 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
```

कमिट ऑब्जेक्ट्स

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

□0□364□0□□74□579 लेखक □□□□□□□□ □□□□□□□□□□□□□□□□.□□□ 1462090215 +0800 कमिटर □□□□□□□□ □□□□□□□□□ □□□□□.□□□ 1462090215 +0800 पहला कमिट

- hash

□ □□□□ 'दूसरा कमिट' | □□□ □□□□□□□-□□□□ 0155□□ -□ □5□□□1□□4 □946□6367□07□□45□□□242□□□7□□002□5□□□72□1
□ □□□□ 'तीसरा कमिट' | □□□ □□□□□□□-□□□□ 3□4□9□ -□ □946□6 09490□□051□34□3693□□□5□7□□63□□□27□295904
□ □□□ □□□ –□□□□ 09490 □□□□□□ 09490□□051□34□3693□□□5□7□□63□□□27□295904 □□□□□□□: □□□□□□□□
□□□□□□□□□□□□□□□.□□□ □□□□: □□□ □□□ 1 16:38:55 2016 +0800 तीसरा कमिट □□□/□□□□.□□□ | 1 + 1 फ़ाइल बदली, 1 इंसर्शन(+) □□□□□□ □946□6367□07□□45□□□242□□□7□□002□5□□□72□1 □□□□□□: □□□□□□□□
□□□□□□□□□□□□□□□.□□□ □□□□: □□□ □□□ 1 16:37:01 2016 +0800 दूसरा कमिट □□□.□□□ | 1 + □□□□.□□□ | 2 +- 2 फ़ाइलें बदली, 2 इंसर्शन(+), 1 डिलीशन(-) □□□□□□ □5□□□1□□4□7□089□□□03□□□□□5□0439□□6□72037 □□□□□□: □□□□□□□□ □□□□□□□□□□□□□□□.□□□ □□□□: □□□ □□□ 1 16:10:15 2016 +0800 पहला कमिट □□□□.□□□ | 1 + 1 फ़ाइल बदली, 1 इंसर्शन(+)

- `commit-tree` blob tree
- `-p`

तीसरा कमिट दूसरा कमिट पहला कमिट ट्री ट्री ट्री “संस्करण 2” “नई फ़ाइल” “संस्करण 1” 09490□ □946□6 □5□□□1 □8329□ 0155□□ 3□4□9□ 83□□□□ □□49□0 1□7□7□ □□□□.□□□ □□□.□□□ □□□□.□□□ □□□.□□□ □□□□.□□□ बैक

** **

□ □□□□ .□□□/□□□□
.□□□/□□□□ .□□□/□□□□/□□□□□ .□□□/□□□□/□□□□ □ □□□□ .□□□/□□□□ -□□□□ □
□ □□□□ “09490□□051□34□3693□□□5□7□□63□□□27□295904” > .□□□/□□□□/□□□□□/□□□□□□ □ □□□ □□□ –□□□□□□□=□□□□□□□□ □□□□□□□ 09490□□051□34□3693□□□5□7□□63□□□27□295904 तीसरा कमिट □946□6367□07□□45□□□242□□□7□□002□5□□□72□1 दूसरा कमिट □5□□□1□□4□7□089□□□03□□□□□5□0439□□6□72037 पहला कमिट □ □□□ □□□□□□□-□□□ □□□□/□□□□□/□□□□□□ 09490□□051□34□3693□□□5□7□□63□□□27□295904 □ □□□ □□□□□□□-□□□ □□□□/□□□□□/□□□□ □946□6367□07□□45□□□242□□□7□□002□5□□□72□1 □ □□□ □□□□□□ * □□□□□□ □□□□

-

- `update-ref`,

तीसरा कमिट दूसरा कमिट पहला कमिट ट्री ट्री ट्री “संस्करण 2” “नई फ़ाइल” “संस्करण 1” 09490□ □946□6 □5□□□1 □8329□ 0155□□ 3□4□9□ 83□□□□ □□49□0 1□7□7□ □□□□.□□□ □□□.□□□ □□□□.□□□ □□□.□□□ □□□□.□□□ बैक
□□□□/□□□□□/□□□□□□ □□□□/□□□□□/□□□□

□ □□□ .□□□/□□□□ □□□: □□□□/□□□□□/□□□□□□ □ □□□ □□□□□□□□-□□□ □□□□ □□□□/□□□□□/□□□□□□ □ □□□ □□□□□□□□-□□□ □□□□ □□□□/□□□□□/□□□□ □ □□□ .□□□/□□□□ □□□: □□□□/□□□□□/□□□□

(,)

symbolic-ref Git Git ( )

, , :

```bash
git symbolic-ref --short HEAD
```

यह कमांड वर्तमान ब्रांच का नाम प्रदर्शित करेगा।

इसी तरह, यदि आप किसी विशिष्ट रेफरेंस को इंगित करने वाले सिंबोलिक नाम को बदलना चाहते हैं, तो आप `git symbolic-ref` का उपयोग कर सकते हैं। उदाहरण के लिए:

```
git symbolic-ref refs/heads/new-branch refs/heads/old-branch
```

यह कमांड new-branch को old-branch की ओर इंगित करने वाले सिंबोलिक रेफरेंस के रूप में सेट करेगा।

टैग सिद्धांत

```
$ git update-ref refs/tags/v1.0 e946d6367f07de45cac242dca7cd002f5eaa72b1
$ git tag
v1.0
$ git tag -a v1.1 e946d6367f07de45cac242dca7cd002f5eaa72b1 -m '     '
$ 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
```

□□□□□□□□

```
$ git remote add origin git@github.com:schacon/simplegit-progit.git
$ git push origin master
Counting objects: 11, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 716 bytes, done.
Total 7 (delta 2), reused 4 (delta 1)
To git@github.com:schacon/simplegit-progit.git
a11bef0..ca82a6d master -> master
$ cat .git/refs/remotes/origin/master
ca82a6dff817ec66f44342007202690a93763949
```

- refs/remotes डायरेक्टरी में रखा जाता है
- □□□□□□□□ के संदर्भ (□□□□□□□□□□□□) और शाखाओं (□□□□□□□□□□) में अंतर यह है कि वे केवल पढ़ने योग्य (□□□□-□□□□) होते हैं
- आप git checkout कर सकते हैं, लेकिन □□□□□ नहीं बदलता है, इसलिए □□□□□□□ का उपयोग करके □□□□□□□□ के संदर्भ को बदला नहीं जा सकता है

```
$ 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 files changed, 709 insertions(+), 2 deletions(-)
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 033b4468fa6b2a9547a70d88d1bbe8bf3f9ed0d5
22044
$ echo '# testing' >> repo.rb
$ git commit -am 'modified repo a bit'
[master 2431da6] modified repo.rb a bit
1 file changed, 1 insertion(+)
$ git cat-file -p master^{tree}
100644 blob fa49b077972391ad58037050f2a75f74e3671e92 new.txt
100644 blob b042a60ef7dff760008df33cee372b945b6e884e repo.rb
100644 blob e3f094f522629ae358806b17daf78246c27c007b test.txt
$ git cat-file -s b042a60ef7dff760008df33cee372b945b6e884e
22054
$ git gc
Counting objects: 18, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (18/18), done.
Total 18 (delta 3), reused 0 (delta 0)
$ find .git/objects -type f
$ git verify-pack -v .git/objects/pack/pack-978e03944f5c581011e6998cd0e9e30000905586.idx
```

□□□□□□□□

```
$ 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
```

यह कोड ब्लॉक □□□ कमांड्स और कॉन्फ़िगरेशन को दर्शाता है। इसमें `git remote add` कमांड का उपयोग करके एक नया रिमोट रिपॉजिटरी जोड़ा गया है, और फिर `git log` कमांड का उपयोग करके विभिन्न रिमोट ब्रांचों के लॉग देखे गए हैं। इसके अलावा, रिमोट कॉन्फ़िगरेशन में `fetch` और `push` सेटिंग्स को भी परिभाषित किया गया है।

/□□/□□□□□□□

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

□□□□□://□□□□□□□.□□□/□□□□□□□□/□□□□□□□□□□-□□□□□□□ □□□□□://□□□□□□□.□□□/□□□□□□□□/□□□□□□□□□□-□□□□□□□ “‘

संदर्भ सामग्री

- □ □□□ □□□