
GIT REAL
2

PURGING HISTORY

CHAPTER 3

PURGING HISTORY

Bob has committed a file he shouldn't have

```
$ git log --patch
commit 5a794156e9e1ef9b9f4db07363f299258b1a5a93
Author: Bob <bob@example.com>
Date:   Tue Mar 19 00:03:32 2013 -0700

    Fix security breach.

deleted file mode 100644
--- a/passwords.txt
+++ /dev/null
@@ -1 +0,0 @@
-The server password is: 'foobar'.
...
```

Even if he deletes it now, its contents will still be visible in history

BEFORE WE CONTINUE...

- Reasons *not* to rewrite history:
  - Why bother? Your data is already compromised.
  - Everyone must update their work to reflect your revised commits.
- When you should do it anyway:
  - Committed files violate someone's copyright.
  - Large binary files are making your repo too big.
  - You're rewriting commits that haven't been made public.

BEFORE WE CONTINUE...

```
$ git clone petshop petshop-filter
Cloning into 'petshop-filter'...
done.

$ cd petshop-filter
```

Bob backs up
his entire repo

You **can** lose work when
you're rewriting history!

TREE FILTER

```
$ git filter-branch --tree-filter <command> ...
```

Git will check each commit out into working directory, run your command, and re-commit

Examples:

```
--tree-filter 'rm -f passwords.txt'
```

Remove "passwords.txt" from project root

```
--tree-filter 'find . -name "*.mp4" -exec rm {} \;'
```

Remove video files from any directory

TREE FILTER

```
$ git filter-branch --tree-filter 'rm -f passwords.txt' -- --all
Rewrite f5d1142dd3e830940e9f1361e8c500df8834fd20 (4/4)
Ref 'refs/heads/treats' was rewritten
...
```

```
$ git log --patch
commit 5a794156e9e1ef9b9f4db07363f299258b1a5a93
Author: Bob <bob@example.com>
Date:   Tue Mar 19 00:03:32 2013 -0700

    Fix security breach.
```

"passwords.txt" contents are gone!

Goes through all branches, and removes "passwords.txt" from each commit

`--all` filter all commits in all branches

`HEAD` filter only current branch

COMMAND FAILS, FILTER FAILS

fails if file isn't present

```
$ git filter-branch --tree-filter 'rm passwords.txt' -- --all
Rewrite 539180de10622fd3d6786b2ad22f65bf70a476d9 (1/4)
rm: passwords.txt: No such file or directory
tree filter failed: rm passwords.txt
```

File won't be there in some commits

doesn't fail if file isn't present

```
$ git filter-branch --tree-filter 'rm -f passwords.txt' -- --all
Rewrite f5d1142dd3e830940e9f1361e8c500df8834fd20 (4/4)
Ref 'refs/heads/treats' was rewritten
```

INDEX FILTER

```
$ git filter-branch --index-filter <command> ...
```

command MUST operate on staging area

Git will run your command against each commit, but without checking it out first (so it's faster)

operates on working directory

```
--index-filter 'rm -f passwords.txt'
```

No effect!

operates on staging area

```
--index-filter 'git rm --cached --ignore-unmatch passwords.txt'
```

INDEX FILTER

As with "--tree-filter", if the command fails the filter will stop

fails if file isn't present

```
--index-filter 'git rm --cached passwords.txt'
```

succeeds even if file isn't present

```
--index-filter 'git rm --cached --ignore-unmatch passwords.txt'
```

FORCE

After you run filter-branch, Git leaves a backup of your tree in the ".git" directory

```
$ git filter-branch --tree-filter 'rm -f passwords.txt'
Cannot create a new backup.
A previous backup already exists in refs/original/
Force overwriting the backup with -f
```

By default, you can't run filter-branch again because it won't overwrite the backup

```
$ git filter-branch -f --tree-filter 'rm -f passwords.txt'
Rewrite 68880af108a5eaa19eb543f3455a1d0dcd6ff632 (4/4)
```

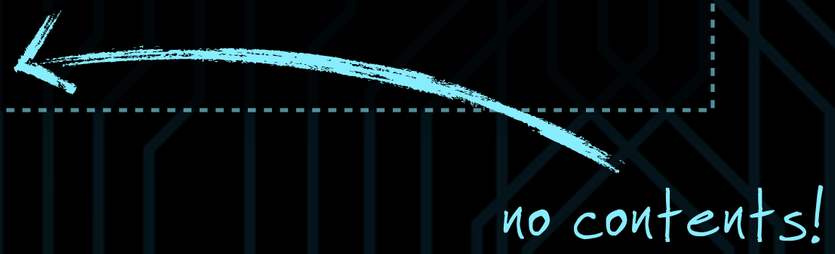
You can force it with the -f option

PRUNE EMPTY COMMITS

Our filters are resulting in some empty commits

```
$ git log --patch
commit 5a794156e9e1ef9b9f4db07363f299258b1a5a93
Author: Bob <bob@example.com>
Date:   Tue Mar 19 00:03:32 2013 -0700

    Fix security breach.
```



PRUNE EMPTY COMMITS

```
$ git log --oneline
68880af Add treats page.
5a79415 Fix security breach.
016b982 Update index.
539180d Initial commit.
```

Commit has no contents

```
$ git filter-branch -f --prune-empty -- --all
```

"--prune-empty" option drops commits that don't alter any files

```
$ git log --oneline
6d9a429 Add treats page.
016b982 Update index.
539180d Initial commit.
```

Empty commit is gone

can prune during filtering, too

```
git filter-branch --tree-filter 'rm -f passwords.txt' --prune-empty -- --all
```