git-filter-repo Rewrite Git history

START HERE

`git clone <url> <dir>`
Work in a fresh clone

`git filter-repo --analyze`
Report what bloats history

`git filter-repo --path <path>`
Keep only this path

`git filter-repo --path <path> --invert-paths`
Drop this path

`git filter-repo --subdirectory-filter <dir>`
Make a subdir the root

`git filter-repo --replace-text <file>`
Scrub secrets in content

`git filter-repo --strip-blobs-bigger-than 10M`
Remove huge files

`git remote add origin <url>`
Re-add the removed remote

Fresh clone throwaway copy → `filter-repo` → **Rewritten history** new hashes, origin removed → `inspect` → **Check result** log, ls-files → `remote add + push --force` → **New remote** collaborators re-clone

CONCEPTS

Run it in a fresh clone: it rewrites every commit hash and refuses on repos that look used.

Filters combine: path selection, renames and content edits run in one pass over history.

Old hashes in commit messages are rewritten to the new ones unless you pass --preserve-commit-hashes.

ANALYZE

`git filter-repo --analyze`
Write size and path reports
Output goes to .git/filter-repo/analysis; nothing is modified.

`git clone --no-local <src> <dst>`
Fresh clone of a local repo

`git filter-repo --force`
Skip the fresh-clone check
Only when you have a backup; the check protects you.

`git filter-repo --dry-run`
Save before/after streams only

`git filter-repo --quiet`
Pass quiet to git commands

`git filter-repo --debug`
Print extra detail

SELECT PATHS

`--path <path>`
Keep a file or directory

`--path <dir>/ --path <file>`
Keep several paths (union)

`--invert-paths`
Remove the matched paths instead

SELECT PATHS (cont.)

`--path-glob '<glob>'`
Match paths by glob

`--path-regex '<regex>'`
Match paths by regex

`--use-base-name`
Match on file name only

`--paths-from-file <file>`
Read path rules from a file

`--refs <ref>`
Limit rewrite to some refs

RESTRUCTURE

`--subdirectory-filter <dir>`
Extract a subdir as the root

`--to-subdirectory-filter <dir>`
Move the root into a subdir

`--path-rename <old>:<new>`
Rename a path or directory
Does not select paths; pair with --path if you want to filter too.

`--tag-rename <old>:<new>`
Rename tags by prefix

CONTENT AND MESSAGES

`--replace-text <file>`
Replace strings in file contents
*File lines: literal, regex:<re> or glob:<g>, optionally followed by ==>text. Default replacement is ***REMOVED***.*

`--replace-message <file>`
Replace strings in messages

`--strip-blobs-bigger-than <size>`
Drop files over a size

`--strip-blobs-with-ids <file>`
Drop blobs listed by ID

CONTENT AND MESSAGES (cont.)

`--mailmap <file>`
Rewrite authors via mailmap

`--use-mailmap`
Use the .mailmap file

`--sensitive-data-removal`
Clean leaked data end to end
Fetches all refs first and can discard unpushed local changes.

`--preserve-commit-hashes`
Keep old hashes in messages

PRUNING AND CALLBACKS

`--prune-empty always|auto|never`
Handle commits left empty

`--prune-degenerate always|auto|never`
Handle degenerate merges

`--partial`
Mix old and new history
Keeps origin and skips gc and reflog expiry.

`--source <src> --target <dst>`
Write result to another repo

`--filename-callback '<python>'`
Rewrite file names in Python

`--message-callback '<python>'`
Rewrite messages in Python

`--name-callback '<python>'`
Rewrite author names

`--email-callback '<python>'`
Rewrite author emails

`--commit-callback '<python>'`
Edit whole commit objects

PRUNING AND CALLBACKS (cont.)

`--refname-callback '<python>'`
Rename refs, tags included

AFTER THE REWRITE

`git remote add origin <url>`
Restore the removed remote

`git push --force --all`
Publish rewritten branches

`git push --force --tags`
Publish rewritten tags

`git log --oneline --stat`
Check the result

`git ls-files`
Confirm remaining files

RECIPES

Split a subdir into its own repo

```
git clone --no-local mono lib
cd lib
git filter-repo --subdirectory-filter libs/core
```

Purge a file from all history

```
git clone <url> clean && cd clean
git filter-repo --path secrets.env --invert-paths
git remote add origin <url>
git push --force --all
```

Scrub a leaked token

```
# expressions.txt
literal:AKIAEXAMPLE123==>***REMOVED***
regex:password=\w+==>password=***REMOVED***

git filter-repo --replace-text expressions.txt
```

Shrink a bloated repo

```
git filter-repo --analyze
less .git/filter-repo/analysis/blob-shas-and-paths.txt
git filter-repo --strip-blobs-bigger-than 10M
```