



Linux Ubuntu 26.04 Manual Pages on command 'git-reflog.1'

\$ man git-reflog.1

GIT-REFLOG(1)

Git Manual

GIT-REFLOG(1)

NAME

git-reflog - Manage reflog information

SYNOPSIS

git reflog [show] [<log-options>] [<ref>]

git reflog list

git reflog exists <ref>

git reflog write <ref> <old-oid> <new-oid> <message>

git reflog delete [--rewrite] [--updateref]

 [--dry-run | -n] [--verbose] <ref>@{<specifier>}...

git reflog drop [--all [--single-worktree] | <refs>...]

git reflog expire [--expire=<time>] [--expire-unreachable=<time>]

 [--rewrite] [--updateref] [--stale-fix]

 [--dry-run | -n] [--verbose] [--all [--single-worktree] | <refs>...]

DESCRIPTION

This command manages the information recorded in the reflogs.

Reference logs, or "reflogs", record when the tips of branches and other references were updated in the local repository. Reflogs are useful in various Git commands, to specify the old value of a reference. For example, HEAD@{2} means "where HEAD used to be two moves ago", master@{one.week.ago} means "where master used to point to one week ago in this local repository", and so on. See gitrevisions(7) for more details.

The command takes various subcommands, and different options depending on the subcommand:

The "show" subcommand (which is also the default, in the absence of any subcommands) shows

the log of the reference provided in the command-line (or HEAD, by default). The reflog covers all recent actions, and in addition the HEAD reflog records branch switching. `git reflog show` is an alias for `git log -g --abbrev-commit --pretty=oneline`; see `git-log(1)` for more information.

The "list" subcommand lists all refs which have a corresponding reflog.

The "exists" subcommand checks whether a ref has a reflog. It exits with zero status if the reflog exists, and non-zero status if it does not.

The "write" subcommand writes a single entry to the reflog of a given reference. This new entry is appended to the reflog and will thus become the most recent entry. The reference name must be fully qualified. Both the old and new object IDs must not be abbreviated and must point to existing objects. The reflog message gets normalized.

The "delete" subcommand deletes single entries from the reflog, but not the reflog itself.

Its argument must be an exact entry (e.g. "git reflog delete master@{2}"). This subcommand is also typically not used directly by end users.

The "drop" subcommand completely removes the reflog for the specified references. This is in contrast to "expire" and "delete", both of which can be used to delete reflog entries, but not the reflog itself.

The "expire" subcommand prunes older reflog entries. Entries older than expire time, or entries older than expire-unreachable time and not reachable from the current tip, are removed from the reflog. This is typically not used directly by end users ? instead, see `git-gc(1)`.

OPTIONS

Options for show

`git reflog show` accepts any of the options accepted by `git log`.

Options for delete

`git reflog delete` accepts options `--updateref`, `--rewrite`, `-n`, `--dry-run`, and `--verbose`, with the same meanings as when they are used with `expire`.

Options for drop

`--all`

Drop the reflogs of all references from all worktrees.

`--single-worktree`

By default when `--all` is specified, reflogs from all working trees are dropped. This option limits the processing to reflogs from the current working tree only.

Options for expire

--all

Process the reflogs of all references.

--single-worktree

By default when --all is specified, reflogs from all working trees are processed. This option limits the processing to reflogs from the current working tree only.

--expire=<time>

Prune entries older than the specified time. If this option is not specified, the expiration time is taken from the configuration setting gc.reflogExpire, which in turn defaults to 90 days. --expire=all prunes entries regardless of their age; --expire=never turns off pruning of reachable entries (but see --expire-unreachable).

--expire-unreachable=<time>

Prune entries older than <time> that are not reachable from the current tip of the branch. If this option is not specified, the expiration time is taken from the configuration setting gc.reflogExpireUnreachable, which in turn defaults to 30 days. --expire-unreachable=all prunes unreachable entries regardless of their age; --expire-unreachable=never turns off early pruning of unreachable entries (but see --expire).

--updateref

Update the reference to the value of the top reflog entry (i.e. <ref>@{0}) if the previous top entry was pruned. (This option is ignored for symbolic references.)

--rewrite

If a reflog entry's predecessor is pruned, adjust its "old" SHA-1 to be equal to the "new" SHA-1 field of the entry that now precedes it.

--stale-fix

Prune any reflog entries that point to "broken commits". A broken commit is a commit that is not reachable from any of the reference tips and that refers, directly or indirectly, to a missing commit, tree, or blob object.

This computation involves traversing all the reachable objects, i.e. it has the same cost as git prune. It is primarily intended to fix corruption caused by garbage collecting using older versions of Git, which didn't protect objects referred to by reflogs.

-n, --dry-run

Do not actually prune any entries; just show what would have been pruned.

--verbose

Print extra information on screen.

GIT

Part of the git(1) suite

Git 2.53.0

03/02/2026

GIT-REFLOG(1)