NAME

git-reflog - Manage reflog information

SYNOPSIS

git reflog <subcommand> <options>

DESCRIPTION

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

git reflog expire [--dry-run] [--stale-fix] [--verbose]
[--expire=<time>] [--expire-unreachable=<time>] [--all] <refs>...
*git reflog delete*ref@{specifier}...
git reflog [*show*] [log-options] [<ref>]

Reflog is a mechanism to record when the tip of branches are updated. This command is to manage the information recorded in it.

The subcommand expire is used to prune 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 the end users — instead, see **git-gc(1)**.

The subcommand show (which is also the default, in the absence of any subcommands) will take all the normal log options, and show the log of the reference provided in the command-line (or HEAD, by default). The reflog will cover all recent actions (HEAD reflog records branch switching as well). It is an alias for git log -g --abbrev-commit --pretty=oneline; see **git-log(1)**.

The reflog is 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, and so on. See **gitrevisions(7)** for more details.

To delete single entries from the reflog, use the subcommand delete and specify the *exact* entry (e.g. git reflog delete master@{2}).

OPTIONS

--stale-fix

This revamps the logic — the definition of broken commit becomes: a commit that is not reachable from any of the refs and there is a missing object among the commit, tree, or blob objects reachable from it that is not reachable from any of the refs.

This computation involves traversing all the reachable objects, i.e. it has the same cost as *git prune*. Fortunately, once this is run, we should not have to ever worry about missing objects, because the current prune and pack-objects know about reflogs and protect objects referred by them.

--expire=<time>

Entries older than this time are pruned. Without the option it is taken from configuration 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>

Entries older than this time and not reachable from the current tip of the branch are pruned. Without the option it is taken from configuration 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).

--all

Instead of listing <refs> explicitly, prune all refs.

--updateref

Update the ref with the sha1 of the top reflog entry (i.e. <ref>@{0}) after expiring or deleting.

--rewrite
: While expiring or deleting, adjust each reflog entry to ensure that the old sha1 field points to the new sha1 field of the previous entry.

--verbose
: Print extra information on screen.

GIT

Part of the **git(1)** suite