NAME

git-notes − Add or inspect object notes

SYNOPSIS

git notes [list [<object>]]
git notes add [−f] [−−allow−empty] [−F <file> | −m <msg> | (−c | −C) <object>] [<object>]
git notes copy [−f] (−−stdin | <from−object> [<to−object>])
git notes append [−−allow−empty] [−F <file> | −m <msg> | (−c | −C) <object>] [<object>]
git notes edit [−−allow−empty] [<object>]
git notes show [<object>]
git notes merge [−v | −q] [−s <strategy>] <notes−ref>
git notes merge −−commit [−v | −q]
git notes merge −−abort [−v | −q]
git notes remove [−−ignore−missing] [−−stdin] [<object>...]
git notes prune [−n] [−v]
git notes get−ref

DESCRIPTION

Adds, removes, or reads notes attached to objects, without touching the objects themselves.

By default, notes are saved to and read from **refs/notes/commits**, but this default can be overridden. See the OPTIONS, CONFIGURATION, and ENVIRONMENT sections below. If this ref does not exist, it will be quietly created when it is first needed to store a note.

A typical use of notes is to supplement a commit message without changing the commit itself. Notes can be shown by *git log* along with the original commit message. To distinguish these notes from the message stored in the commit object, the notes are indented like the message, after an unindented line saying "Notes (<refname>):" (or "Notes:" for **refs/notes/commits**).

Notes can also be added to patches prepared with **git format−patch** by using the **−−notes** option. Such notes are added as a patch commentary after a three dash separator line.

To change which notes are shown by *git log*, see the "notes.displayRef" configuration in **git-log**(1).

See the "notes.rewrite.<command>" configuration for a way to carry notes across commands that rewrite commits.

SUBCOMMANDS

list

List the notes object for a given object. If no object is given, show a list of all note objects and the objects they annotate (in the format "<note object> <annotated object>"). This is the default subcommand if no subcommand is given.

add

Add notes for a given object (defaults to HEAD). Abort if the object already has notes (use **−f** to overwrite existing notes). However, if you're using **add** interactively (using an editor to supply the notes contents), then − instead of aborting − the existing notes will be opened in the editor (like the **edit** subcommand).

copy

Copy the notes for the first object onto the second object (defaults to HEAD). Abort if the second object already has notes, or if the first object has none (use −f to overwrite existing notes to the second object). This subcommand is equivalent to: **git notes add [−f] −C $(git notes list <from−object>) <to−object>**

In **−−stdin** mode, take lines in the format

<from−object> SP <to−object> [SP <rest>] LF

on standard input, and copy the notes from each <from−object> to its corresponding <to−object>. (The optional **<rest>** is ignored so that the command can read the input given to the **post−rewrite** hook.)

append
: Append to the notes of an existing object (defaults to HEAD). Creates a new notes object if needed.

edit
: Edit the notes for a given object (defaults to HEAD).

show
: Show the notes for a given object (defaults to HEAD).

merge
: Merge the given notes ref into the current notes ref. This will try to merge the changes made by the given notes ref (called "remote") since the merge−base (if any) into the current notes ref (called "local").

  If conflicts arise and a strategy for automatically resolving conflicting notes (see the "NOTES MERGE STRATEGIES" section) is not given, the "manual" resolver is used. This resolver checks out the conflicting notes in a special worktree (**.git/NOTES_MERGE_WORKTREE**), and instructs the user to manually resolve the conflicts there. When done, the user can either finalize the merge with *git notes merge −−commit*, or abort the merge with *git notes merge −−abort*.

remove
: Remove the notes for given objects (defaults to HEAD). When giving zero or one object from the command line, this is equivalent to specifying an empty note message to the **edit** subcommand.

prune
: Remove all notes for non−existing/unreachable objects.

get−ref
: Print the current notes ref. This provides an easy way to retrieve the current notes ref (e.g. from scripts).

OPTIONS

−f, −−force
: When adding notes to an object that already has notes, overwrite the existing notes (instead of aborting).

−m <msg>, −−message=<msg>
: Use the given note message (instead of prompting). If multiple **−m** options are given, their values are concatenated as separate paragraphs. Lines starting with **#** and empty lines other than a single line between paragraphs will be stripped out.

−F <file>, −−file=<file>
: Take the note message from the given file. Use − to read the note message from the standard input. Lines starting with **#** and empty lines other than a single line between paragraphs will be stripped out.

−C <object>, −−reuse−message=<object>
: Take the given blob object (for example, another note) as the note message. (Use **git notes copy <object>** instead to copy notes between objects.)

−c <object>, −−reedit−message=<object>
: Like *−C*, but with **−c** the editor is invoked, so that the user can further edit the note message.

−−allow−empty
: Allow an empty note object to be stored. The default behavior is to automatically remove empty notes.

−−ref <ref>
: Manipulate the notes tree in <ref>. This overrides **GIT_NOTES_REF** and the "core.notesRef"

configuration. The ref specifies the full refname when it begins with **refs/notes/**; when it begins with **notes/**, **refs/** and otherwise **refs/notes/** is prefixed to form a full name of the ref.

−−ignore−missing
Do not consider it an error to request removing notes from an object that does not have notes attached to it.

−−stdin
Also read the object names to remove notes from the standard input (there is no reason you cannot combine this with object names from the command line).

−n, −−dry−run
Do not remove anything; just report the object names whose notes would be removed.

−s <strategy>, −−strategy=<strategy>
When merging notes, resolve notes conflicts using the given strategy. The following strategies are recognized: "manual" (default), "ours", "theirs", "union" and "cat_sort_uniq". This option overrides the "notes.mergeStrategy" configuration setting. See the "NOTES MERGE STRATEGIES" section below for more information on each notes merge strategy.

−−commit
Finalize an in−progress *git notes merge*. Use this option when you have resolved the conflicts that *git notes merge* stored in .git/NOTES_MERGE_WORKTREE. This amends the partial merge commit created by *git notes merge* (stored in .git/NOTES_MERGE_PARTIAL) by adding the notes in .git/NOTES_MERGE_WORKTREE. The notes ref stored in the .git/NOTES_MERGE_REF symref is updated to the resulting commit.

−−abort
Abort/reset an in−progress *git notes merge*, i.e. a notes merge with conflicts. This simply removes all files related to the notes merge.

−q, −−quiet
When merging notes, operate quietly.

−v, −−verbose
When merging notes, be more verbose. When pruning notes, report all object names whose notes are removed.

DISCUSSION

Commit notes are blobs containing extra information about an object (usually information to supplement a commit's message). These blobs are taken from notes refs. A notes ref is usually a branch which contains "files" whose paths are the object names for the objects they describe, with some directory separators included for performance reasons [1].

Every notes change creates a new commit at the specified notes ref. You can therefore inspect the history of the notes by invoking, e.g., **git log −p notes/commits**. Currently the commit message only records which operation triggered the update, and the commit authorship is determined according to the usual rules (see **git-commit**(1)). These details may change in the future.

It is also permitted for a notes ref to point directly to a tree object, in which case the history of the notes can be read with **git log −p −g <refname>**.

NOTES MERGE STRATEGIES

The default notes merge strategy is "manual", which checks out conflicting notes in a special work tree for resolving notes conflicts (**.git/NOTES_MERGE_WORKTREE**), and instructs the user to resolve the conflicts in that work tree. When done, the user can either finalize the merge with *git notes merge −−commit*, or abort the merge with *git notes merge −−abort*.

Users may select an automated merge strategy from among the following using either −s/−−strategy option or configuring notes.mergeStrategy accordingly:

"ours" automatically resolves conflicting notes in favor of the local version (i.e. the current notes ref).

"theirs" automatically resolves notes conflicts in favor of the remote version (i.e. the given notes ref being merged into the current notes ref).

"union" automatically resolves notes conflicts by concatenating the local and remote versions.

"cat_sort_uniq" is similar to "union", but in addition to concatenating the local and remote versions, this strategy also sorts the resulting lines, and removes duplicate lines from the result. This is equivalent to applying the "cat | sort | uniq" shell pipeline to the local and remote versions. This strategy is useful if the notes follow a line−based format where one wants to avoid duplicated lines in the merge result. Note that if either the local or remote version contain duplicate lines prior to the merge, these will also be removed by this notes merge strategy.

EXAMPLES

You can use notes to add annotations with information that was not available at the time a commit was written.

```
$ git notes add −m 'Tested−by: Johannes Sixt <j6t@kdbg.org>' 72a144e2
$ git show −s 72a144e
[...]
   Signed−off−by: Junio C Hamano <gitster@pobox.com>

Notes:
   Tested−by: Johannes Sixt <j6t@kdbg.org>
```

In principle, a note is a regular Git blob, and any kind of (non−)format is accepted. You can binary−safely create notes from arbitrary files using *git hash−object*:

```
$ cc *.c
$ blob=$(git hash−object −w a.out)
$ git notes −−ref=built add −−allow−empty −C "$blob" HEAD
```

(You cannot simply use **git notes −−ref=built add −F a.out HEAD** because that is not binary−safe.) Of course, it doesn't make much sense to display non−text−format notes with *git log*, so if you use such notes, you'll probably need to write some special−purpose tools to do something useful with them.

CONFIGURATION

core.notesRef
: Notes ref to read and manipulate instead of **refs/notes/commits**. Must be an unabbreviated ref name. This setting can be overridden through the environment and command line.

notes.mergeStrategy
: Which merge strategy to choose by default when resolving notes conflicts. Must be one of **manual**, **ours**, **theirs**, **union**, or **cat_sort_uniq**. Defaults to **manual**. See "NOTES MERGE STRATEGIES" section above for more information on each strategy.

  This setting can be overridden by passing the **−−strategy** option.

notes.<name>.mergeStrategy
: Which merge strategy to choose when doing a notes merge into refs/notes/<name>. This overrides the more general "notes.mergeStrategy". See the "NOTES MERGE STRATEGIES" section above for more information on each available strategy.

notes.displayRef

Which ref (or refs, if a glob or specified more than once), in addition to the default set by **core.notesRef** or **GIT_NOTES_REF**, to read notes from when showing commit messages with the *git log* family of commands. This setting can be overridden on the command line or by the **GIT_NOTES_DISPLAY_REF** environment variable. See **git-log**(1).

notes.rewrite.<command>
When rewriting commits with <command> (currently **amend** or **rebase**), if this variable is **false**, git will not copy notes from the original to the rewritten commit. Defaults to **true**. See also "**notes.rewriteRef**" below.

This setting can be overridden by the **GIT_NOTES_REWRITE_REF** environment variable.

notes.rewriteMode
When copying notes during a rewrite, what to do if the target commit already has a note. Must be one of **overwrite**, **concatenate**, **cat_sort_uniq**, or **ignore**. Defaults to **concatenate**.

This setting can be overridden with the **GIT_NOTES_REWRITE_MODE** environment variable.

notes.rewriteRef
When copying notes during a rewrite, specifies the (fully qualified) ref whose notes should be copied. May be a glob, in which case notes in all matching refs will be copied. You may also specify this configuration several times.

Does not have a default value; you must configure this variable to enable note rewriting.

Can be overridden with the **GIT_NOTES_REWRITE_REF** environment variable.

ENVIRONMENT

GIT_NOTES_REF
Which ref to manipulate notes from, instead of **refs/notes/commits**. This overrides the **core.notesRef** setting.

GIT_NOTES_DISPLAY_REF
Colon−delimited list of refs or globs indicating which refs, in addition to the default from **core.notesRef** or **GIT_NOTES_REF**, to read notes from when showing commit messages. This overrides the **notes.displayRef** setting.

A warning will be issued for refs that do not exist, but a glob that does not match any refs is silently ignored.

GIT_NOTES_REWRITE_MODE
When copying notes during a rewrite, what to do if the target commit already has a note. Must be one of **overwrite**, **concatenate**, **cat_sort_uniq**, or **ignore**. This overrides the **core.rewriteMode** setting.

GIT_NOTES_REWRITE_REF
When rewriting commits, which notes to copy from the original to the rewritten commit. Must be a colon−delimited list of refs or globs.

If not set in the environment, the list of notes to copy depends on the **notes.rewrite.<command>** and **notes.rewriteRef** settings.

GIT

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

NOTES

1. Permitted pathnames have the form *ab/cd/ef/.../abcdef...*: a sequence of directory names of two hexadecimal digits each followed by a filename with the rest of the object ID.