NAME

git-replace − Create, list, delete refs to replace objects

SYNOPSIS

git replace [−f] <object> <replacement>
git replace [−f] −−edit <object>
git replace [−f] −−graft <commit> [<parent>...]
git replace [−f] −−convert−graft−file
git replace −d <object>...
git replace [−−format=<format>] [−l [<pattern>]]

DESCRIPTION

Adds a *replace* reference in **refs/replace/** namespace.

The name of the *replace* reference is the SHA−1 of the object that is replaced. The content of the *replace* reference is the SHA−1 of the replacement object.

The replaced object and the replacement object must be of the same type. This restriction can be bypassed using **−f**.

Unless **−f** is given, the *replace* reference must not yet exist.

There is no other restriction on the replaced and replacement objects. Merge commits can be replaced by non−merge commits and vice versa.

Replacement references will be used by default by all Git commands except those doing reachability traversal (prune, pack transfer and fsck).

It is possible to disable use of replacement references for any command using the **−−no−replace−objects** option just after *git*.

For example if commit *foo* has been replaced by commit *bar*:

$ git −−no−replace−objects cat−file commit foo

shows information about commit *foo*, while:

$ git cat−file commit foo

shows information about commit *bar*.

The **GIT_NO_REPLACE_OBJECTS** environment variable can be set to achieve the same effect as the **−−no−replace−objects** option.

OPTIONS

−f, −−force
: If an existing replace ref for the same object exists, it will be overwritten (instead of failing).

−d, −−delete
: Delete existing replace refs for the given objects.

−−edit <object>
: Edit an object's content interactively. The existing content for <object> is pretty−printed into a temporary file, an editor is launched on the file, and the result is parsed to create a new object of the

same type as <object>. A replacement ref is then created to replace <object> with the newly created object. See **git-var**(1) for details about how the editor will be chosen.

−−raw
: When editing, provide the raw object contents rather than pretty−printed ones. Currently this only affects trees, which will be shown in their binary form. This is harder to work with, but can help when repairing a tree that is so corrupted it cannot be pretty−printed. Note that you may need to configure your editor to cleanly read and write binary data.

−−graft <commit> [<parent>...]
: Create a graft commit. A new commit is created with the same content as <commit> except that its parents will be [<parent>...] instead of <commit>'s parents. A replacement ref is then created to replace <commit> with the newly created commit. Use **−−convert−graft−file** to convert a **$GIT_DIR/info/grafts** file and use replace refs instead.

−−convert−graft−file
: Creates graft commits for all entries in **$GIT_DIR/info/grafts** and deletes that file upon success. The purpose is to help users with transitioning off of the now−deprecated graft file.

−l <pattern>, −−list <pattern>
: List replace refs for objects that match the given pattern (or all if no pattern is given). Typing "git replace" without arguments, also lists all replace refs.

−−format=<format>
: When listing, use the specified <format>, which can be one of *short*, *medium* and *long*. When omitted, the format defaults to *short*.

FORMATS

The following format are available:

- *short*: <replaced sha1>
- *medium*: <replaced sha1> → <replacement sha1>
- *long*: <replaced sha1> (<replaced type>) → <replacement sha1> (<replacement type>)

CREATING REPLACEMENT OBJECTS

git-hash-object(1), **git-rebase**(1), and **git−filter−repo**[1], among other git commands, can be used to create replacement objects from existing objects. The **−−edit** option can also be used with *git replace* to create a replacement object by editing an existing object.

If you want to replace many blobs, trees or commits that are part of a string of commits, you may just want to create a replacement string of commits and then only replace the commit at the tip of the target string of commits with the commit at the tip of the replacement string of commits.

BUGS

Comparing blobs or trees that have been replaced with those that replace them will not work properly. And using **git reset −−hard** to go back to a replaced commit will move the branch to the replacement commit instead of the replaced commit.

There may be other problems when using *git rev−list* related to pending objects.

SEE ALSO

git-hash-object(1) **git-rebase**(1) **git-tag**(1) **git-branch**(1) **git-commit**(1) **git-var**(1) **git**(1) **git−filter−repo**[1]

GIT

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

NOTES

1. git-filter-repo
   https://github.com/newren/git-filter-repo