git-refs

git-refs(1) ==========

NAME ---- git-refs - Low-level access to refs

SYNOPSIS --------

git refs migrate --ref-storage-format=<format> [--no-reflog] [--dry-run] git refs verify [--strict] [--verbose] git refs optimize [--all] [--no-prune] [--auto] [--include <pattern>] [--exclude <pattern>] git refs list [--count=<count>] [--shell|--perl|--python|--tcl] [(--sort=<key>)…] [--format=<format>] [--include-root-refs] [--points-at=<object>] [--merged[=<object>]] [--no-merged[=<object>]] [--contains[=<object>]] [--no-contains[=<object>]] [(--exclude=<pattern>)…] [--start-after=<marker>] [--stdin | (<pattern>…)] git refs exists <ref> git refs create [--message=<reason>] [--no-deref] [--create-reflog] <ref> <new-value> git refs delete [--message=<reason>] [--no-deref] <ref> [<old-value>] git refs update [--message=<reason>] [--no-deref] [--create-reflog] <ref> <new-value> [<old-value>] git refs rename [--message=<reason>] <old-ref> <new-ref>

DESCRIPTION -----------

This command provides low-level access to refs.

COMMANDS --------

`migrate`
Migrate ref store between different formats.

The ref format migration has several known limitations in its current form:

- It is not possible to migrate repositories that have worktrees.
- There is no way to block concurrent writes to the repository during an ongoing migration. Concurrent writes can lead to an inconsistent migrated state. Users are expected to block writes on a higher level. If your repository is registered for scheduled maintenance, it is recommended to unregister it first with linkgit:git-maintenance[1].

These limitations may eventually be lifted.

`verify`
Verify reference database consistency.

`list`
List references in the repository with support for filtering, formatting, and sorting. This subcommand is an alias for linkgit:git-for-each-ref[1] and offers identical functionality.

`exists`
Check whether the given reference exists. Returns an exit code of 0 if it does, 2 if it is missing, and 1 in case looking up the reference failed with an error other than the reference being missing. This does not verify whether the reference resolves to an actual object.

`optimize`
Optimizes references to improve repository performance and reduce disk usage. This subcommand is an alias for linkgit:git-pack-refs[1] and offers identical functionality.

`create`
Create the given reference, which must not already exist, pointing at `<new-value>`.

`delete`
Delete the given reference. This subcommand mirrors `git` `update-ref` `-d` (see linkgit:git-update-ref[1]). When `<old-value>` is given, the reference is only deleted after verifying that it currently contains `<old-value>`.

`update`
Update the given reference to point at `<new-value>`. If `<old-value>` is given, the reference is only updated after verifying that it currently contains `<old-value>`. As a special case, an all-zeroes `<new-value>` deletes the branch, whereas an all-zeroes `<old-value>` ensures that the branch does not yet exist.

`rename`
Rename the reference `<oldref>` to `<newref>`. The old reference must exist and the new reference must not yet exist, and both must have a well-formed name (see linkgit:git-check-ref-format[1]).

OPTIONS -------

The following options are specific to `git` `refs` `migrate`:

`--ref-storage-format=<format>`
The ref storage format to migrate the ref store to. Can be one of:

`files`;; for loose files with packed-refs. ifndef::with-breaking-changes[] This is the default. endif::with-breaking-changes[] `reftable`;; for the reftable format. ifdef::with-breaking-changes[] This is the default. endif::with-breaking-changes[]

`--ref-format=<format>`
Deprecated alias of `--ref-storage-format=<format>`.

`--dry-run`
Perform the migration, but do not modify the repository. The migrated refs will be written into a

separate directory that can be inspected separately. The name of the directory will be reported on stdout. This can be used to double check that the migration works as expected before performing the actual migration.

`--reflog`

`--no-reflog`
Choose between migrating the reflog data to the new backend, and discarding them. The default is "--reflog", to migrate.

The following options are specific to `git refs verify`:

`--strict`
Enable stricter error checking. This will cause warnings to be reported as errors. See linkgit:git-fsck[1].

`--verbose`
When verifying the reference database consistency, be chatty.

The following options are specific to ’git refs list’:

`<pattern>…`
If one or more <*pattern*> parameters are given, only refs are shown that match against at least one pattern, either using `fnmatch`(3) or literally, in the latter case matching completely or from the beginning up to a slash.

`--stdin`
The list of patterns is read from standard input instead of from the argument list.

`--count=<count>`
Stop after showing <*count*> refs.

`--sort=<key>`
Sort on the field name <*key*>. Prefix `-` to sort in descending order of the value. When unspecified, `refname` is used. You may use the `--sort=<key>` option multiple times, in which case the last key becomes the primary key.

`--format[=<format>]`
A string that interpolates `%(fieldname)` from a ref being shown and the object it points at. In addition, the string literal `%%` renders as `%` and `%xx` - where `xx` are hex digits - renders as the character with hex code `xx`. For example, `%00` interpolates to `\0` (*NUL*), `%09` to `t` (*TAB*), and `%0a` to `n` (*LF*).

When unspecified, <*format*> defaults to `%(objectname)` `SPC` `%(objecttype)` `TAB` `%(refname)`.

`--color[=<when>]`
Respect any colors specified in the `--format` option. The <*when*> field must be one of `always`, `never`, or `auto` (if <*when*> is absent, behave as if `always` was given).

`--shell`

`--perl`

`--python`

`--tcl`
If given, strings that substitute `%(fieldname)` placeholders are quoted as string literals suitable for the specified host language. This is meant to produce a scriptlet that can directly be "eval"ed.

`--points-at=<object>`
Only list refs which points at the given object.

`--merged[=<object>]`
Only list refs whose tips are reachable from the specified commit (`HEAD` if not specified).

`--no-merged[=<object>]`
Only list refs whose tips are not reachable from <*object*>(`HEAD` if not specified).

`--contains[=<object>]`
Only list refs which contain <*object*>(`HEAD` if not specified).

`--no-contains[=<object>]`
Only list refs which don’t contain <*object*> (`HEAD` if not specified).

`--ignore-case`
Sorting and filtering refs are case insensitive.

`--omit-empty`
Do not print a newline after formatted refs where the format expands to the empty string.

`--exclude=<excluded-pattern>`
If one or more `--exclude` options are given, only refs which do not match any <*excluded-pattern*> parameters are shown. Matching is done using the same rules as <*pattern*> above.

`--include-root-refs`
List root refs (`HEAD` and pseudorefs) apart from regular refs.

`--start-after=<marker>`
Allows paginating the output by skipping references up to and including the specified marker. When paging, it should be noted that references may be deleted, modified or added between invocations. Output will only yield those references which follow the marker lexicographically. Output begins from the first reference that would come after the marker alphabetically. Cannot be used with `--sort=<key>` or `--stdin` options, or the <*pattern*> argument(s) to limit the refs.

The following options are specific to ’git refs optimize’:

`--all`

The command by default packs all tags and refs that are already packed, and leaves other refs alone. This is because branches are expected to be actively developed and packing their tips does not help performance. This option causes all refs to be packed as well, with the exception of hidden refs, broken refs, and symbolic refs. Useful for a repository with many branches of historical interests.

`--no-prune`

The command usually removes loose refs under `$GIT_DIR/refs` hierarchy after packing them. This option tells it not to.

`--auto`

Pack refs as needed depending on the current state of the ref database. The behavior depends on the ref format used by the repository and may change in the future.
- "files": Loose references are packed into the `packed-refs` file based on the ratio of loose references to the size of the `packed-refs` file. The bigger the `packed-refs` file, the more loose references need to exist before we repack.
- "reftable": Tables are compacted such that they form a geometric sequence. For two tables N and N+1, where N+1 is newer, this maintains the property that N is at least twice as big as N+1. Only tables that violate this property are compacted.

`--include` `<pattern>`

Pack refs based on a `glob(7)` pattern. Repetitions of this option accumulate inclusion patterns. If a ref is both included in `--include` and `--exclude`, `--exclude` takes precedence. Using `--include` will preclude all tags from being included by default. Symbolic refs and broken refs will never be packed. When used with `--all`, it will be a noop. Use `--no-include` to clear and reset the list of patterns.

`--exclude` `<pattern>`

Do not pack refs matching the given `glob(7)` pattern. Repetitions of this option accumulate exclusion patterns. Use `--no-exclude` to clear and reset the list of patterns. If a ref is already packed, including it with `--exclude` will not unpack it.
When used with `--all`, pack only loose refs which do not match any of the provided `--exclude` patterns.
When used with `--include`, refs provided to `--include`, minus refs that are provided to `--exclude` will be packed.

The following options are specific to commands which write references:

`--create-reflog`
Create a reflog for the reference even if one would not ordinarily be created.

`--message=<reason>`
Use the given <reason> string for the reflog entry associated with the update. An empty message is rejected.

`--no-deref`
Operate on <ref> itself rather than the reference it points to via a symbolic ref.

GIT --- Part of the linkgit:git[1] suite

□□□ #1
□ □□□ □□□ 2026-10-31 20:12:23 UTC
□ □□□ □□□ 2026-10-31 20:12:23 UTC