git-remote

git-remote(1) ==========

NAME ---- git-remote - Manage set of tracked repositories

SYNOPSIS --------

git remote [-v | --verbose] git remote add [-t <branch>] [-m <master>] [-f] [--[no-]tags] [--mirror=(fetch|push)] <name> <URL> git remote rename [--[no-]progress] <old> <new> git remote remove <name> git remote set-head <name> (-a | --auto | -d | --delete | <branch>) git remote set-branches [--add] <name> <branch>… git remote get-url [--push] [--all] <name> git remote set-url [--push] <name> <newurl> [<oldurl>] git remote set-url --add [--push] <name> <newurl> git remote set-url --delete [--push] <name> <URL> git remote [-v | --verbose] show [-n] <name>… git remote prune [-n | --dry-run] <name>… git remote [-v | --verbose] update [-p | --prune] [(<group> | <remote>)…]

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

Manage the set of repositories ("remotes") whose branches you track.

`git` `remote` changes only the local repository, i.e. its configuration and its refs, and never modifies a remote repository. Some subcommands, such as `show`, `prune`, `update` and `set-head` `--auto`, contact a remote repository to read from it.

OPTIONS -------

`-v`

`--verbose`
Be a little more verbose and show remote url after name. For promisor remotes, also show which filters (`blob:none` etc.) are configured. NOTE: This must be placed between `remote` and subcommand.

COMMANDS --------

With no arguments, show a list of existing remotes. Several subcommands are available to perform operations on the remotes.

`add`

Add a remote named *<name>* for the repository at *<URL>*. The command `git` `fetch` `<name>` can then be used to create and update remote-tracking branches `<name>/<branch>`.
With `-f` option, `git` `fetch` `<name>` is run immediately after the remote information is set up.

With `--tags` option, `git` `fetch` `<name>` imports every tag from the remote repository.
With `--no-tags` option, `git` `fetch` `<name>` does not import tags from the remote repository.
By default, only tags on fetched branches are imported (see linkgit:git-fetch[1]).
With `-t` `<branch>` option, instead of the default glob refspec for the remote to track all branches under the `refs/remotes/<name>/` namespace, a refspec to track only *<branch>* is created. You can give more than one `-t` `<branch>` to track multiple branches without grabbing all branches.
With `-m` `<master>` option, a symbolic-ref `refs/remotes/<name>/HEAD` is set up to point at remote's *<master>* branch. See also the set-head command.
When a fetch mirror is created with `--mirror=fetch`, the refs will not be stored in the `refs/remotes/` namespace, but rather everything in `refs/` on the remote will be directly mirrored into `refs/` in the local repository. This option only makes sense in bare repositories, because a fetch would overwrite any local commits.
When a push mirror is created with `--mirror=push`, then `git` `push` will always behave as if `--mirror` was passed.

`rename`

Rename the remote named *<old>* to *<new>*. All remote-tracking branches and configuration settings for the remote are updated.
In case *<old>* and *<new>* are the same, and *<old>* is a file under `$GIT_DIR/remotes` or `$GIT_DIR/branches`, the remote is converted to the configuration file format.

`remove`

`rm`

Remove the remote named *<name>*. All remote-tracking branches and configuration settings for the remote are removed.

`set-head`

Set or delete the default branch (i.e. the target of the symbolic-ref `refs/remotes/<name>/HEAD`) for the named remote. Having a default branch for a remote is not required, but allows the name of the remote to be specified in lieu of a specific branch. For example, if the default branch for `origin` is set to `master`, then `origin` may be specified wherever you would normally specify `origin/master`.
With `-d` or `--delete`, the symbolic ref `refs/remotes/<name>/HEAD` is deleted.
With `-a` or `--auto`, the remote is queried to determine its `HEAD`, then the symbolic-ref `refs/remotes/<name>/HEAD` is set to the same branch. e.g., if the remote `HEAD` is pointed at `next`, `git` `remote` `set-head` `origin` `-a` will set the symbolic-ref `refs/remotes/origin/HEAD` to `refs/remotes/origin/next`. This will only work if `refs/remotes/origin/next` already exists; if not it must be fetched first.
Use *<branch>* to set the symbolic-ref `refs/remotes/<name>/HEAD` explicitly. e.g., `git` `remote` `set-head` `origin` `master` will set the symbolic-ref `refs/remotes/origin/HEAD` to `refs/remotes/origin/master`. This will only work if `refs/remotes/origin/master` already exists; if not it must be fetched first. +

`set-branches`

Change the list of branches tracked by the named remote. This can be used to track a subset of the available remote branches after the initial setup for a remote.
The named branches will be interpreted as if specified with the `-t` option on the `git` `remote` `add` command line.
With `--add`, instead of replacing the list of currently tracked branches, adds to that list.

`get-url`

Retrieves the URLs for a remote. Configurations for `insteadOf` and `pushInsteadOf` are expanded here. By default, only the first URL is listed.
With `--push`, push URLs are queried rather than fetch URLs.
With `--all`, all URLs for the remote will be listed.

`set-url`

Change URLs for the remote. Sets first URL for remote <*name*> that matches regex <*oldurl*> (first URL if no <*oldurl*> is given) to <*newurl*>. If <*oldurl*> doesn't match any URL, an error occurs and nothing is changed.
With `--push`, push URLs are manipulated instead of fetch URLs.
With `--add`, instead of changing existing URLs, new URL is added.
With `--delete`, instead of changing existing URLs, all URLs matching regex <*URL*> are deleted for remote <*name*>. Trying to delete all non-push URLs is an error.
Note that the push URL and the fetch URL, even though they can be set differently, must still refer to the same place. What you pushed to the push URL should be what you would see if you immediately fetched from the fetch URL. If you are trying to fetch from one place (e.g. your upstream) and push to another (e.g. your publishing repository), use two separate remotes.

`show`

Give some information about the remote <*name*>.
With `-n` option, the remote heads are not queried first with `git` `ls-remote` `<name>`; cached information is used instead.

`prune`

Delete stale references associated with <*name*>. By default, stale remote-tracking branches under <*name*> are deleted, but depending on global configuration and the configuration of the remote we might even prune local tags that haven't been pushed there. Equivalent to `git` `fetch` `--prune` `<name>`, except that no new references will be fetched.
See the PRUNING section of linkgit:git-fetch[1] for what it'll prune depending on various configuration.
With `--dry-run` option, report what branches would be pruned, but do not actually prune them.

`update`

Fetch updates for remotes or remote groups in the repository as defined by `remotes.<group>`. If neither group nor remote is specified on the command line, the configuration parameter `remotes.default` will be used; if `remotes.default` is not defined, all remotes which do not have the configuration parameter `remote.<name>.skipDefaultUpdate` set to `true` will be updated. (See linkgit:git-config[1]).
With `--prune` option, run pruning against all the remotes that are updated.

DISCUSSION ----------

The remote configuration is achieved using the `remote.origin.url` and `remote.origin.fetch` configuration variables. (See linkgit:git-config[1]).

EXIT STATUS -----------

On success, the exit status is `0`.

When subcommands such as `add`, `rename`, and `remove` can’t find the remote in question, the exit status is `2`. When the remote already exists, the exit status is `3`.

On any other error, the exit status may be any other non-zero value.

EXAMPLES --------

- Add a new remote, fetch, and check out a branch from it

```
$ git remote
origin
$ git branch -r
  origin/HEAD -> origin/master
  origin/master
$ git remote add staging git://git.kernel.org/.../gregkh/staging.git
$ git remote
origin
staging
$ git fetch staging
...
From git://git.kernel.org/pub/scm/linux/kernel/git/gregkh/staging
 * [new branch]      master     -> staging/master
 * [new branch]      staging-linus -> staging/staging-linus
 * [new branch]      staging-next -> staging/staging-next
$ git branch -r
  origin/HEAD -> origin/master
  origin/master
```

```
  staging/master
  staging/staging-linus
  staging/staging-next
$ git switch -c staging staging/master
...
```

- Imitate `git` `clone` but track only selected branches

```
$ mkdir project.git
$ cd project.git
$ git init
$ git remote add -f -t master -m master origin git://example.com/git.git/
$ git merge origin
```

SEE ALSO -------- linkgit:git-fetch[1] linkgit:git-branch[1] linkgit:git-config[1]

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

□□□ #1
□ □□□ □□□ 2026-10-31 20:12:27 UTC
□ □□□ □□□ 2026-10-31 20:12:27 UTC