Remove the upstream information for <branchname>. If no branch is specified it defaults to the current branch.

--edit-description
Open an editor and edit the text to explain what the branch is for, to be used by various other commands (e.g. format-patch, request-pull, and merge (if enabled)). Multi-line explanations may be used.

--contains [<commit>]
Only list branches which contain the specified commit (HEAD if not specified). Implies --list.

--merged [<commit>]
Only list branches whose tips are reachable from the specified commit (HEAD if not specified). Implies --list.

--no-merged [<commit>]
Only list branches whose tips are not reachable from the specified commit (HEAD if not specified). Implies --list.

<branchname>
The name of the branch to create or delete. The new branch name must pass all checks defined by git-check-ref-format(1). Some of these checks may restrict the characters allowed in a branch name.

<start-point>
The new branch head will point to this commit. It may be given as a branch name, a commit-id, or a tag. If this option is omitted, the current HEAD will be used instead.

<oldbranch>
The name of an existing branch to rename.

<newbranch>
The new name for an existing branch. The same restrictions as for <branchname> apply.

--sort=<key>
Sort based on the key given. Prefix - to sort in descending order of the value. You may use the --sort=<key> option multiple times, in which case the last key becomes the primary key. The keys supported are the same as those in git for-each-ref. Sort order defaults to sorting based on the full refname (including refs/... prefix). This lists detached HEAD (if present) first, then local branches and finally remote-tracking branches.

--points-at <object>
Only list branches of the given object.
Examples
Start development from a known tag
$ git clone git://git.kernel.org/pub/scm/.../linux-2.6 my2.6
$ cd my2.6
$ git branch my2.6.14 v2.6.14   (1)
$ git checkout my2.6.14
This step and the next one could be combined into a single step with "checkout -b my2.6.14 v2.6.14".

Delete an unneeded branch
$ git clone git://git.kernel.org/.../git.git my.git
$ cd my.git
$ git branch -d -r origin/todo origin/html origin/man   (1)
$ git branch -D test   (2)
Delete the remote-tracking branches "todo", "html" and "man". The next fetch or pull will create them again unless you configure them not to. See git-fetch(1).

Delete the "test" branch even if the "master" branch (or whichever branch is currently checked out) does not have all commits from the test branch.

Notes
If you are creating a branch that you want to checkout immediately, it is easier to use the git checkout command with its -b option to create a branch and check it out with a single command.

The options --contains, --merged and --no-merged serve three related but different purposes:

--contains <commit> is used to find all branches which will need special attention if <commit> were to be rebased or amended, since those branches contain the specified <commit>.

--merged is used to find all branches which can be safely deleted, since those branches are fully contained by HEAD.

--no-merged is used to find branches which are candidates for merging into HEAD, since those branches are not fully contained by HEAD.

SEE ALSO
git-check-ref-format(1), git-fetch(1), git-remote(1), “Understanding history: What is a branch?” in the Git User’s Manual.

GIT
Part of the git(1) suite