```
git-clone(1)
============

NAME
----
git-clone - Clone a repository into a new directory


SYNOPSIS
--------
[verse]
'git clone' [--template=<template-directory>]
	  [-l] [-s] [--no-hardlinks] [-q] [-n] [--bare] [--mirror]
	  [-o <name>] [-b <name>] [-u <upload-pack>] [--reference <repository>]
	  [--dissociate] [--separate-git-dir <git-dir>]
	  [--depth <depth>] [--[no-]single-branch] [--no-tags]
	  [--recurse-submodules[=<pathspec>]] [--[no-]shallow-submodules]
	  [--[no-]remote-submodules] [--jobs <n>] [--sparse] [--[no-]reject-shallow]
	  [--filter=<filter>] [--] <repository>
	  [<directory>]

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

Clones a repository into a newly created directory, creates
remote-tracking branches for each branch in the cloned repository
(visible using `git branch --remotes`), and creates and checks out an
initial branch that is forked from the cloned repository's
currently active branch.
```

```

After the clone, a plain `git fetch` without arguments will update
all the remote-tracking branches, and a `git pull` without
arguments will in addition merge the remote master branch into the
current master branch, if any (this is untrue when "--single-branch"
is given; see below).

This default configuration is achieved by creating references to
the remote branch heads under `refs/remotes/origin` and
by initializing `remote.origin.url` and `remote.origin.fetch`
configuration variables.


OPTIONS
-------
-l::
--local::
        When the repository to clone from is on a local machine,
        this flag bypasses the normal "Git aware" transport
        mechanism and clones the repository by making a copy of
        HEAD and everything under objects and refs directories.
        The files under `.git/objects/` directory are hardlinked
        to save space when possible.
+
If the repository is specified as a local path (e.g., `/path/to/repo`),
this is the default, and --local is essentially a no-op.  If the
repository is specified as a URL, then this flag is ignored (and we
never use the local optimizations).  Specifying `--no-local` will
override the default when `/path/to/repo` is given, using the regular
Git transport instead.
+
*NOTE*: this operation can race with concurrent modification to the
source repository, similar to running `cp -r src dst` while modifying
`src`.

--no-hardlinks::
        Force the cloning process from a repository on a local
        filesystem to copy the files under the `.git/objects`
        directory instead of using hardlinks. This may be desirable
        if you are trying to make a back-up of your repository.

-s::
--shared::
        When the repository to clone is on the local machine,
        instead of using hard links, automatically setup
        `.git/objects/info/alternates` to share the objects
        with the source repository.  The resulting repository
        starts out without any object of its own.
+
*NOTE*: this is a possibly dangerous operation; do *not* use
it unless you understand what it does. If you clone your
repository using this option and then delete branches (or use any
other Git command that makes any existing commit unreferenced) in the
source repository, some objects may become unreferenced (or dangling).
These objects may be removed by normal Git operations (such as `git commit`)
which automatically call `git maintenance run --auto`. (See
linkgit:git-maintenance[1].) If these objects are removed and were referenced
by the cloned repository, then the cloned repository will become corrupt.
+
Note that running `git repack` without the `--local` option in a repository
cloned with `--shared` will copy objects from the source repository into a pack
in the cloned repository, removing the disk space savings of `clone --shared`.
It is safe, however, to run `git gc`, which uses the `--local` option by
default.
+
If you want to break the dependency of a repository cloned with `--shared` on
its source repository, you can simply run `git repack -a` to copy all
objects from the source repository into a pack in the cloned repository.

--reference[-if-able] <repository>::
        If the reference repository is on the local machine,
        automatically setup `.git/objects/info/alternates` to
        obtain objects from the reference repository.  Using
        an already existing repository as an alternate will
        require fewer objects to be copied from the repository
        being cloned, reducing network and local storage costs.
        When using the `--reference-if-able`, a non existing
        directory is skipped with a warning instead of aborting
        the clone.
+
*NOTE*: see the NOTE for the `--shared` option, and also the
`--dissociate` option.

--dissociate::
        Borrow the objects from reference repositories specified
        with the `--reference` options only to reduce network
        transfer, and stop borrowing from them after a clone is made
        by making necessary local copies of borrowed objects.  This
        option can also be used when cloning locally from a
        repository that already borrows objects from another
        repository--the new repository will borrow objects from the
        same repository, and this option can be used to stop the
        borrowing.

-q::
--quiet::
        Operate quietly.  Progress is not reported to the standard
        error stream.
```

```

-v::
--verbose::
	Run verbosely. Does not affect the reporting of progress status
	to the standard error stream.

--progress::
	Progress status is reported on the standard error stream
	by default when it is attached to a terminal, unless `--quiet`
	is specified. This flag forces progress status even if the
	standard error stream is not directed to a terminal.

--server-option=<option>::
	Transmit the given string to the server when communicating using
	protocol version 2.  The given string must not contain a NUL or LF
	character.  The server's handling of server options, including
	unknown ones, is server-specific.
	When multiple `--server-option=<option>` are given, they are all
	sent to the other side in the order listed on the command line.

-n::
--no-checkout::
	No checkout of HEAD is performed after the clone is complete.

--[no-]reject-shallow::
	Fail if the source repository is a shallow repository.
	The 'clone.rejectShallow' configuration variable can be used to
	specify the default.

--bare::
	Make a 'bare' Git repository.  That is, instead of
	creating `<directory>` and placing the administrative
	files in `<directory>/.git`, make the `<directory>`
	itself the `$GIT_DIR`. This obviously implies the `--no-checkout`
	because there is nowhere to check out the working tree.
	Also the branch heads at the remote are copied directly
	to corresponding local branch heads, without mapping
	them to `refs/remotes/origin/`.  When this option is
	used, neither remote-tracking branches nor the related
	configuration variables are created.

--sparse::
	Employ a sparse-checkout, with only files in the toplevel
	directory initially being present.  The
	linkgit:git-sparse-checkout[1] command can be used to grow the
	working directory as needed.

--filter=<filter-spec>::
	Use the partial clone feature and request that the server sends
	a subset of reachable objects according to a given object filter.
	When using `--filter`, the supplied `<filter-spec>` is used for
	the partial clone filter. For example, `--filter=blob:none` will
	filter out all blobs (file contents) until needed by Git. Also,
	`--filter=blob:limit=<size>` will filter out all blobs of size
	at least `<size>`. For more details on filter specifications, see
	the `--filter` option in linkgit:git-rev-list[1].

--mirror::
	Set up a mirror of the source repository.  This implies `--bare`.
	Compared to `--bare`, `--mirror` not only maps local branches of the
	source to local branches of the target, it maps all refs (including
	remote-tracking branches, notes etc.) and sets up a refspec configuration such
	that all these refs are overwritten by a `git remote update` in the
	target repository.

-o <name>::
--origin <name>::
	Instead of using the remote name `origin` to keep track of the upstream
	repository, use `<name>`.  Overrides `clone.defaultRemoteName` from the
	config.

-b <name>::
--branch <name>::
	Instead of pointing the newly created HEAD to the branch pointed
	to by the cloned repository's HEAD, point to `<name>` branch
	instead. In a non-bare repository, this is the branch that will
	be checked out.
	`--branch` can also take tags and detaches the HEAD at that commit
	in the resulting repository.

-u <upload-pack>::
--upload-pack <upload-pack>::
	When given, and the repository to clone from is accessed
	via ssh, this specifies a non-default path for the command
	run on the other end.

--template=<template-directory>::
	Specify the directory from which templates will be used;
	(See the "TEMPLATE DIRECTORY" section of linkgit:git-init[1].)

-c <key>=<value>::
--config <key>=<value>::
	Set a configuration variable in the newly-created repository;
	this takes effect immediately after the repository is
	initialized, but before the remote history is fetched or any
	files checked out.  The key is in the same format as expected by
	linkgit:git-config[1] (e.g., `core.eol=true`). If multiple
	values are given for the same key, each value will be written to
	the config file. This makes it safe, for example, to add
	additional fetch refspecs to the origin remote.
+
Due to limitations of the current implementation, some configuration
variables do not take effect until after the initial fetch and checkout.
Configuration variables known to not take effect are:
`remote.<name>.mirror` and `remote.<name>.tagOpt`.  Use the
corresponding `--mirror` and `--no-tags` options instead.

--depth <depth>::
	Create a 'shallow' clone with a history truncated to the
	specified number of commits. Implies `--single-branch` unless
	`--no-single-branch` is given to fetch the histories near the
	tips of all branches. If you want to clone submodules shallowly,
	also pass `--shallow-submodules`.

--shallow-since=<date>::
	Create a shallow clone with a history after the specified time.

--shallow-exclude=<revision>::
	Create a shallow clone with a history, excluding commits
	reachable from a specified remote branch or tag.  This option
	can be specified multiple times.

--[no-]single-branch::
	Clone only the history leading to the tip of a single branch,
	either specified by the `--branch` option or the primary
	branch remote's `HEAD` points at.
	Further fetches into the resulting repository will only update the
	remote-tracking branch for the branch this option was used for the
	initial cloning.  If the HEAD at the remote did not point at any
	branch when `--single-branch` clone was made, no remote-tracking
	branch is created.

--no-tags::
	Don't clone any tags, and set
	`remote.<remote>.tagOpt=--no-tags` in the config, ensuring
	that future `git pull` and `git fetch` operations won't follow
	any tags. Subsequent explicit tag fetches will still work,
	(see linkgit:git-fetch[1]).
+
Can be used in conjunction with `--single-branch` to clone and
maintain a branch with no references other than a single cloned
branch. This is useful e.g. to maintain minimal clones of the default
branch of some repository for search indexing.

--recurse-submodules[=<pathspec>]::
	After the clone is created, initialize and clone submodules
	within based on the provided pathspec.  If no pathspec is
	provided, all submodules are initialized and cloned.
	This option can be given multiple times for pathspecs consisting
	of multiple entries.  The resulting clone has `submodule.active` set to
	the provided pathspec, or "." (meaning all submodules) if no
	pathspec is provided.
+
Submodules are initialized and cloned using their default settings. This is
equivalent to running
`git submodule update --init --recursive <pathspec>` immediately after
the clone is finished. This option is ignored if the cloned repository does
not have a worktree/checkout (i.e. if any of `--no-checkout`/`-n`, `--bare`,
or `--mirror` is given)

--[no-]shallow-submodules::
	All submodules which are cloned will be shallow with a depth of 1.

--[no-]remote-submodules::
	All submodules which are cloned will use the status of the submodule's
	remote-tracking branch to update the submodule, rather than the
	superproject's recorded SHA-1. Equivalent to passing `--remote` to
	`git submodule update`.

--separate-git-dir=<git-dir>::
	Instead of placing the cloned repository where it is supposed
	to be, place the cloned repository at the specified directory,
	then make a filesystem-agnostic Git symbolic link to there.
	The result is Git repository can be separated from working
	tree.

-j <n>::
--jobs <n>::
	The number of submodules fetched at the same time.
	Defaults to the `submodule.fetchJobs` option.

<repository>::
	The (possibly remote) repository to clone from.  See the
	<<URLS,GIT URLS>> section below for more information on specifying
	repositories.

<directory>::
	The name of a new directory to clone into.  The "humanish"
	part of the source repository is used if no directory is
	explicitly given (`repo` for `/path/to/repo.git` and `foo`
	for `host.xz:foo/.git`).  Cloning into an existing directory
	is only allowed if the directory is empty.

:git-clone: 1
include::urls.txt[]

```

```
EXAMPLES
--------

* Clone from upstream:
+
------------
$ git clone git://git.kernel.org/pub/scm/.../linux.git my-linux
$ cd my-linux
$ make
------------


* Make a local clone that borrows from the current directory, without checking things out:
+
------------
$ git clone -l -s -n . ../copy
$ cd ../copy
$ git show-branch
------------


* Clone from upstream while borrowing from an existing local directory:
+
------------
$ git clone --reference /git/linux.git \
        git://git.kernel.org/pub/scm/.../linux.git \
        my-linux
$ cd my-linux
------------


* Create a bare repository to publish your changes to the public:
+
------------
$ git clone --bare -l /home/proj/.git /pub/scm/proj.git
------------


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