



Linux Ubuntu 26.04 Manual Pages on command 'git-init.1'

\$ man git-init.1

GIT-INIT(1) Git Manual GIT-INIT(1)

NAME

git-init - Create an empty Git repository or reinitialize an existing one

SYNOPSIS

```
git init [-q | --quiet] [--bare] [--template=<template-directory>]
        [--separate-git-dir <git-dir>] [--object-format=<format>]
        [--ref-format=<format>]
        [-b <branch-name> | --initial-branch=<branch-name>]
        [--shared[=<permissions>]] [<directory>]
```

DESCRIPTION

This command creates an empty Git repository - basically a .git directory with subdirectories for objects, refs/heads, refs/tags, and template files. An initial branch without any commits will be created (see the --initial-branch option below for its name). If the GIT_DIR environment variable is set then it specifies a path to use instead of ./git for the base of the repository.

If the object storage directory is specified via the GIT_OBJECT_DIRECTORY environment variable then the sha1 directories are created underneath; otherwise, the default \$GIT_DIR/objects directory is used.

Running git init in an existing repository is safe. It will not overwrite things that are already there. The primary reason for rerunning git init is to pick up newly added templates (or to move the repository to another place if --separate-git-dir is given).

OPTIONS

-q, --quiet

Only print error and warning messages; all other output will be suppressed.

`--bare`

Create a bare repository. If `GIT_DIR` environment is not set, it is set to the current working directory.

`--object-format=<format>`

Specify the given object `<format>` (hash algorithm) for the repository. The valid values are sha1 and (if enabled) sha256. sha1 is the default.

Note: At present, there is no interoperability between SHA-256 repositories and SHA-1 repositories.

Historically, we warned that SHA-256 repositories may later need backward incompatible changes when we introduce such interoperability features. Today, we only expect compatible changes. Furthermore, if such changes prove to be necessary, it can be expected that SHA-256 repositories created with today's Git will be usable by future versions of Git without data loss.

`--ref-format=<format>`

Specify the given ref storage `<format>` for the repository. The valid values are:

? files for loose files with packed-refs. This is the default.

? reftable for the reftable format. This format is experimental and its internals are subject to change.

`--template=<template-directory>`

Specify the directory from which templates will be used. (See the "TEMPLATE DIRECTORY" section below.)

`--separate-git-dir=<git-dir>`

Instead of initializing the repository as a directory to either `$GIT_DIR` or `./git/`, create a text file there containing the path to the actual repository. This file acts as a filesystem-agnostic Git symbolic link to the repository.

If this is a reinitialization, the repository will be moved to the specified path.

`-b <branch-name>`, `--initial-branch=<branch-name>`

Use `<branch-name>` for the initial branch in the newly created repository. If not specified, fall back to the default name (currently master, but this will change to main when Git 3.0 is released). The default name can be customized via the `init.defaultBranch` configuration variable.

`--shared[=(false|true|umask|group|all|world|everybody|<perm>)]`

Specify that the Git repository is to be shared amongst several users. This allows users belonging to the same group to push into that repository. When specified, the config variable `core.sharedRepository` is set so that files and directories under `$GIT_DIR` are created with the requested permissions. When not specified, Git will use permissions reported by `umask(2)`.

The option can have the following values, defaulting to `group` if no value is given:

`umask`, `false`

Use permissions reported by `umask(2)`. The default, when `--shared` is not specified.

`group`, `true`

Make the repository group-writable, (and `g+sx`, since the git group may not be the primary group of all users). This is used to loosen the permissions of an otherwise safe `umask(2)` value. Note that the `umask` still applies to the other permission bits (e.g. if `umask` is `0022`, using `group` will not remove read privileges from other (non-group) users). See `0xxx` for how to exactly specify the repository permissions.

`all`, `world`, `everybody`

Same as `group`, but make the repository readable by all users.

`<perm>`

`<perm>` is a 3-digit octal number prefixed with `?0`` and each file will have mode `<perm>`. `<perm>` will override users' `umask(2)` value (and not only loosen permissions as `group` and `all` do). `0640` will create a repository which is group-readable, but not group-writable or accessible to others. `0660` will create a repo that is readable and writable to the current user and group, but inaccessible to others (directories and executable files get their `x` bit from the `r` bit for corresponding classes of users).

By default, the configuration flag `receive.denyNonFastForwards` is enabled in shared repositories, so that you cannot force a non fast-forwarding push into it.

If you provide a `<directory>`, the command is run inside it. If this directory does not exist, it will be created.

TEMPLATE DIRECTORY

Files and directories in the template directory whose name do not start with a dot will be copied to the `$GIT_DIR` after it is created.

The template directory will be one of the following (in order):

? the argument given with the `--template` option;

- ? the contents of the \$GIT_TEMPLATE_DIR environment variable;
- ? the init.templateDir configuration variable; or
- ? the default template directory: /usr/share/git-core/templates.

The default template directory includes some directory structure, suggested "exclude patterns" (see gitignore(5)), and sample hook files.

The sample hooks are all disabled by default. To enable one of the sample hooks rename it by removing its .sample suffix.

See githooks(5) for more general info on hook execution.

EXAMPLES

Start a new Git repository for an existing code base

```
$ cd /path/to/my/codebase
```

```
$ git init    (1)
```

```
$ git add .   (2)
```

```
$ git commit  (3)
```

1. Create a /path/to/my/codebase/.git directory.
2. Add all existing files to the index.
3. Record the pristine state as the first commit in the history.

CONFIGURATION

Everything below this line in this section is selectively included from the git-config(1)

documentation. The content is the same as what's found there:

init.templateDir

Specify the directory from which templates will be copied.

init.defaultBranch

Allows overriding the default branch name e.g. when initializing a new repository.

init.defaultObjectFormat

Allows overriding the default object format for new repositories. See --object-format= in git-init(1). Both the command line option and the GIT_DEFAULT_HASH environment variable take precedence over this config.

init.defaultRefFormat

Allows overriding the default ref storage format for new repositories. See --ref-format= in git-init(1). Both the command line option and the GIT_DEFAULT_REF_FORMAT environment variable take precedence over this config.

Part of the git(1) suite

Git 2.53.0

03/02/2026

GIT-INIT(1)