NAME

git-add - Add file contents to the index

SYNOPSIS

git add [-n] [-v] [--force | -f] [--interactive | -i] [--patch | -p]
[--edit | -e] [--[no-]all | --[no-]ignore-removal | [--update | -u]]
[--intent-to-add | -N] [--refresh] [--ignore-errors] [--ignore-missing]
[--] [<pathspec>...]

DESCRIPTION

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

The index holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working directory, and before running the commit command, you must use the add command to add any new or modified files to the index.

This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at the time the add command is run; if you want subsequent changes included in the next commit, then you must run git add again to add the new content to the index.

The git status command can be used to obtain a summary of which files have changes that are staged for the next commit.

The git add command will not add ignored files by default. If any ignored files were explicitly specified on the command line, git add will fail with a list of ignored files. Ignored files reached by directory recursion or filename globbing performed by Git (quote your globs before the shell) will be silently ignored. The *git add* command can be used to add ignored files with the -f (force) option.

Please see **git-commit(1)** for alternative ways to add content to a commit.

OPTIONS

<pathspec>...
: Files to add content from. Fileglobs (e.g. *.c) can be given to add all matching files. Also a leading directory name (e.g. dir to add dir/file1 and dir/file2) can be given to update the index to match the current state of the directory as a whole (e.g. specifying dir will record not just a file dir/file1 modified in the working tree, a file dir/file2 added to the working tree, but also a file dir/file3 removed from the working tree. Note that older versions of Git used to ignore removed files; use --no-all option if you want to add modified or new files but ignore removed ones.

-n, --dry-run
: Don't actually add the file(s), just show if they exist and/or will be ignored.

-v, --verbose
: Be verbose.

-f, --force
: Allow adding otherwise ignored files.

-i, --interactive
: Add modified contents in the working tree interactively to the index. Optional path arguments may be supplied to limit operation to a subset of the working tree. See "Interactive mode" for details.

-p, --patch

Interactively choose hunks of patch between the index and the work tree and add them to the index. This gives the user a chance to review the difference before adding modified contents to the index.

This effectively runs add --interactive, but bypasses the initial command menu and directly jumps to the patch subcommand. See “Interactive mode” for details.

-e, --edit
Open the diff vs. the index in an editor and let the user edit it. After the editor was closed, adjust the hunk headers and apply the patch to the index.

The intent of this option is to pick and choose lines of the patch to apply, or even to modify the contents of lines to be staged. This can be quicker and more flexible than using the interactive hunk selector. However, it is easy to confuse oneself and create a patch that does not apply to the index. See EDITING PATCHES below.

-u, --update
Update the index just where it already has an entry matching <pathspec>. This removes as well as modifies index entries to match the working tree, but adds no new files.

If no <pathspec> is given when -u option is used, all tracked files in the entire working tree are updated (old versions of Git used to limit the update to the current directory and its subdirectories).

-A, --all, --no-ignore-removal
Update the index not only where the working tree has a file matching <pathspec> but also where the index already has an entry. This adds, modifies, and removes index entries to match the working tree.

If no <pathspec> is given when -A option is used, all files in the entire working tree are updated (old versions of Git used to limit the update to the current directory and its subdirectories).

--no-all, --ignore-removal
Update the index by adding new files that are unknown to the index and files modified in the working tree, but ignore files that have been removed from the working tree. This option is a no-op when no <pathspec> is used.

This option is primarily to help users who are used to older versions of Git, whose git add <pathspec>... was a synonym for git add --no-all <pathspec>..., i.e. ignored removed files.

-N, --intent-to-add
Record only the fact that the path will be added later. An entry for the path is placed in the index with no content. This is useful for, among other things, showing the unstaged content of such files with git diff and committing them with git commit -a.

--refresh
Don’t add the file(s), but only refresh their stat() information in the index.

--ignore-errors
If some files could not be added because of errors indexing them, do not abort the operation, but continue adding the others. The command shall still exit with non-zero status. The configuration variable add.ignoreErrors can be set to true to make this the default behaviour.

--ignore-missing
This option can only be used together with --dry-run. By using this option the user can check if any of the given files would be ignored, no matter if they are already present in the work tree or not.

--
This option can be used to separate command-line options from the list of files, (useful when filenames might be mistaken for command-line options).

CONFIGURATION

The optional configuration variable core.excludesfile indicates a path to a file containing patterns of file names to exclude from git-add, similar to $GIT_DIR/info/exclude. Patterns in the exclude file are used in addition to those in info/exclude. See **gitignore(5)**.

EXAMPLES

- Adds content from all *.txt files under Documentation directory and its subdirectories:

  $ git add Documentation/*.txt

  Note that the asterisk * is quoted from the shell in this example; this lets the command include the files from subdirectories of Documentation/ directory.
- Considers adding content from all git-*.sh scripts:

  $ git add git-*.sh

  Because this example lets the shell expand the asterisk (i.e. you are listing the files explicitly), it does not consider subdir/git-foo.sh.

INTERACTIVE MODE

When the command enters the interactive mode, it shows the output of the *status* subcommand, and then goes into its interactive command loop.

The command loop shows the list of subcommands available, and gives a prompt What now> . In general, when the prompt ends with a single >, you can pick only one of the choices given and type return, like this:

*** Commands ***
1: status 2: update 3: revert 4: add untracked
5: patch 6: diff 7: quit 8: help
What now> 1

You also could say s or sta or status above as long as the choice is unique.

The main command loop has 6 subcommands (plus help and quit).

status
: This shows the change between HEAD and index (i.e. what will be committed if you say git commit), and between index and working tree files (i.e. what you could stage further before git commit using git add) for each path. A sample output looks like this:

  staged unstaged path
  1: binary nothing foo.png
  2: +403/-35 +1/-1 git-add--interactive.perl

  It shows that foo.png has differences from HEAD (but that is binary so line count cannot be shown) and there is no difference between indexed copy and the working tree version (if the working tree version were also different, *binary* would have been shown in place of *nothing*). The other file, git-add--interactive.perl, has 403 lines added and 35 lines deleted if you commit what is in the index, but working tree file has further modifications (one addition and one deletion).

update
: This shows the status information and issues an Update>> prompt. When the prompt ends with double >>, you can make more than one selection, concatenated with whitespace or comma. Also you can say ranges. E.g. 2-5 7,9 to choose 2,3,4,5,7,9 from the list. If the second number in a range is omitted, all remaining patches are taken. E.g. 7- to choose 7,8,9 from the list. You can say * to choose everything.

  What you chose are then highlighted with *, like this:

  staged unstaged path
  1: binary nothing foo.png

* 2: +403/-35 +1/-1 git-add--interactive.perl

To remove selection, prefix the input with - like this:

Update>> -2

After making the selection, answer with an empty line to stage the contents of working tree files for selected paths in the index.

revert
: This has a very similar UI to *update*, and the staged information for selected paths are reverted to that of the HEAD version. Reverting new paths makes them untracked.

add untracked
: This has a very similar UI to *update* and *revert*, and lets you add untracked paths to the index.

patch
: This lets you choose one path out of a *status* like selection. After choosing the path, it presents the diff between the index and the working tree file and asks you if you want to stage the change of each hunk. You can select one of the following options and type return:

y - stage this hunk
n - do not stage this hunk
q - quit; do not stage this hunk or any of the remaining ones
a - stage this hunk and all later hunks in the file
d - do not stage this hunk or any of the later hunks in the file
g - select a hunk to go to
/ - search for a hunk matching the given regex
j - leave this hunk undecided, see next undecided hunk
J - leave this hunk undecided, see next hunk
k - leave this hunk undecided, see previous undecided hunk
K - leave this hunk undecided, see previous hunk
s - split the current hunk into smaller hunks
e - manually edit the current hunk
? - print help

After deciding the fate for all hunks, if there is any hunk that was chosen, the index is updated with the selected hunks.

You can omit having to type return here, by setting the configuration variable interactive.singlekey to true.

diff
: This lets you review what will be committed (i.e. between HEAD and index).

EDITING PATCHES

Invoking git add -e or selecting e from the interactive hunk selector will open a patch in your editor; after the editor exits, the result is applied to the index. You are free to make arbitrary changes to the patch, but note that some changes may have confusing results, or even result in a patch that cannot be applied. If you want to abort the operation entirely (i.e., stage nothing new in the index), simply delete all lines of the patch. The list below describes some common things you may see in a patch, and which editing operations make sense on them.

added content
: Added content is represented by lines beginning with +. You can prevent staging any addition lines by deleting them.

removed content
: Removed content is represented by lines beginning with -. You can prevent staging their removal by converting the - to a (space).

modified content
: Modified content is represented by - lines (removing the old content) followed by + lines (adding the replacement content). You can prevent staging the modification by converting - lines to , and removing + lines. Beware that modifying only half of the pair is likely to introduce confusing changes to the index.

There are also more complex operations that can be performed. But beware that because the patch is applied only to the index and not the working tree, the working tree will appear to undo the change in the index. For example, introducing a new line into the index that is in neither the HEAD nor the working tree will stage the new line for commit, but the line will appear to be reverted in the working tree.

Avoid using these constructs, or do so with extreme caution.

removing untouched content
: Content which does not differ between the index and working tree may be shown on context lines, beginning with a (space). You can stage context lines for removal by converting the space to a -. The resulting working tree file will appear to re-add the content.

modifying existing content
: One can also modify context lines by staging them for removal (by converting to -) and adding a + line with the new content. Similarly, one can modify + lines for existing additions or modifications. In all cases, the new modification will appear reverted in the working tree.

new content
: You may also add new content that does not exist in the patch; simply add new lines, each starting with +. The addition will appear reverted in the working tree.

There are also several operations which should be avoided entirely, as they will make the patch impossible to apply:

- adding context ( ) or removal (-) lines
- deleting context or removal lines
- modifying the contents of context or removal lines

SEE ALSO

git-status(1) git-rm(1) git-reset(1) git-mv(1) git-commit(1) git-update-index(1)

GIT

Part of the **git(1)** suite