.gitignore Cheatsheet

By Dejan Panovski • Updated on Nov 13, 2026

.gitignore pattern syntax at a glance: wildcards, directory rules, negation, anchoring, global ignore files, untracking committed files, and git check-ignore.

A `.gitignore` file lists patterns for untracked files that Git should leave out of `git status` and `git add`. Each line holds one pattern. Git matches it relative to the directory that contains the file. The rules below were tested with Git 2.53 on Ubuntu 26.04.

Pattern Basics

A slash at the start or in the middle anchors the pattern to the directory of the `.gitignore` file.

`# comment`	Comment; the # must be the first character on the line
`debug.log`	File or directory named `debug.log` at any depth
`logs/`	Directory named `logs` at any depth, never a file
`/TODO`	`TODO` next to the `.gitignore` file only
`/dist/`	Root `dist` directory, not `app/dist/`
`doc/frotz`	Matches `doc/frotz`, not `a/doc/frotz`
`**/doc/frotz`	`doc/frotz` at any depth
`\#notes`	Literal name `#notes` at any depth
`\!draft`	Literal name `!draft` at any depth
`name\`	Keep a trailing space; unescaped trailing spaces are dropped

Wildcards

`*`, `?`, and `[...]` never match a slash. Use `**` to cross directories.

`*.log`	Every `.log` file at any depth
`access?.log`	One character: `access1.log`, not `access10.log`
`access.[0-2].log`	`access.0.log`, `access.1.log`, or `access.2.log`
`*.[oa]`	Files that end in `.o` or `.a`
`*.[!oa]`	One-character extension other than o or a
`**/build`	`build` at any depth, same as `build`
`logs/**`	Everything inside the root `logs` directory
`a/**/b`	`a/b`, `a/x/b`, `a/x/y/b`
`src/*/temp/`	`temp` exactly one level below `src`
`src/**/temp/`	`temp` at any depth below `src`, including `src/temp`
`docs/**/*.pdf`	PDF files anywhere under `docs`

Negation

A pattern that starts with `!` re-includes a path. The last matching line wins.

`!error.log`	After `*.log`, keep every `error.log`
`!.env.example`	After `.env*`, keep the example file
`!logs/.gitkeep`	After `logs/*`, keep a placeholder in the empty directory
`!/src/`	After `/*`, keep only the root `src` directory
`!*/`	After `*`, re-include directories so later `!` rules can reach their files
`!*.md`	After `*` and `!*/`, keep only Markdown files
`!/.gitignore`	Keep the `.gitignore` file itself when `/*` or `*` ignores everything

Rule Files

A `.gitignore` rule wins over `.git/info/exclude`, which wins over the global file.

`.gitignore`	Shared rules; commit it with the project
`app/.gitignore`	Rules relative to `app/`; they override rules from parent directories
`.git/info/exclude`	Rules for one clone only; never committed
`~/.config/git/ignore`	Global rules for every repository (default path)
`$XDG_CONFIG_HOME/git/ignore`	Global default path when `XDG_CONFIG_HOME` is set
`git config --global core.excludesFile ~/.gitignore_global`	Use a custom global ignore file
`git config --get core.excludesFile`	Show the configured global ignore file

Untracking Files

Ignore rules apply only to untracked files. When others pull the commit, Git deletes the file from their working tree.

`git rm --cached file`	Stop tracking a file and keep it on disk
`git rm -r --cached dir/`	Stop tracking a directory
`git rm -r --cached -n dir/`	Dry run: list the files that would be untracked
`git ls-files -ci --exclude-standard`	Tracked files that match an ignore rule
`git add -f file`	Add a file that an ignore rule matches

Checking Rules

`git check-ignore` reads the same rule files as `git add`.

`git check-ignore -v path`	File, line number, and pattern that match `path`
`git check-ignore -v -n path`	Also print paths that match no rule (shown with `::`)
`git check-ignore -v --no-index path`	Include tracked files, which are skipped by default
`git check-ignore -q path`	Exit status only: 0 ignored, 1 not ignored
`git status --ignored`	List ignored files and directories (`!!` in `--short` output)
`git status --ignored --untracked-files=all`	List each file inside ignored directories
`git ls-files --others --ignored --exclude-standard`	Every ignored file, one per line

Cleaning Ignored Files

Run the `-n` preview first. `git clean -f` deletes files without a confirmation prompt.

`git clean -ndX`	Preview: only ignored files and directories
`git clean -fdX`	Delete ignored files, such as build output
`git clean -ndx`	Preview: ignored and untracked files
`git clean -fdx`	Delete ignored and untracked files
`git stash -a`	Stash changes plus untracked and ignored files instead of deleting them

Common Patterns

Add only the lines that match your stack.

`node_modules/`	npm, pnpm, and Yarn dependencies
`dist/`	Build output from JavaScript bundlers
`target/`	Rust and Maven build output
`*.o`	C and C++ object files
`__pycache__/`	Python bytecode cache
`.venv/`	Python virtual environment
`vendor/`	PHP Composer dependencies
`.env*`	Local environment files; add `!.env.example` to keep the template
`*.log`	Log files
`coverage/`	Test coverage reports
`.idea/`	JetBrains IDE settings
`.vscode/*`	VS Code settings; add `!.vscode/extensions.json` to share extension recommendations
`*.swp`	Vim swap files
`*~`	Backup files from Emacs, gedit, and other editors
`.DS_Store`	macOS directory metadata
`Thumbs.db`	Windows thumbnail cache

Troubleshooting

Quick checks for rules that do not behave as expected.

`File still appears in git status`	Check `git ls-files --error-unmatch -- file`; if tracked, use `git rm --cached file`. Otherwise, check the rule with `git check-ignore -v file`
`! rule has no effect`	The parent directory is ignored; use `logs/*` instead of `logs/`
`! rule above the pattern it should override`	Move it below; the last matching line wins
`check-ignore -v prints a ! pattern`	A negation matched, so the path is not ignored (the exit status is still 0)
`check-ignore prints nothing`	The path is tracked or matches no rule; add `--no-index` or `-n`
`doc/frotz misses a/doc/frotz`	A middle slash anchors the pattern; use `**/doc/frotz`
`*.log # logs matches nothing`	Comments work only on their own line
`Empty directory never shows up`	Git tracks files, not directories; add a placeholder such as `.gitkeep`
`The following paths are ignored by one of your .gitignore files`	`git add` skipped an ignored path; use `git add -f` to add it anyway
`Pattern works on macOS but not on Linux`	Git normally sets `core.ignoreCase=true` when it detects a case-insensitive filesystem; match the file name case for portable rules

Related Guides

Use these guides for full Git workflows.

`.gitignore Guide`	Ignore files and directories, with examples
`Git Cheatsheet`	Everyday Git commands
`Remove Untracked Files`	`git clean` options and dry runs
`git stash`	Save uncommitted, untracked, and ignored files
`Undo the Last Commit`	Remove a file you committed by mistake