ScopeTreasure Courses

GIT & GITHUB COURSES · LESSON G.6 · CODE SCOPE SERIES

.gitignore + Git ki Aam Ghaltiyaan

Keep junk and secrets out forever: .gitignore patterns, removing an already-tracked file with git rm --cached, and four classic mistakes fixed WRONG → RIGHT.

Introduction

Every project creates files that should never travel: `node_modules` with its ten thousand files, build output, editor junk — and the dangerous one, `.env` with real passwords and API keys. **`.gitignore`** is the list you give Git on day one: "never track these." This lesson writes that list properly, learns the one command that removes a file Git is ALREADY tracking (`git rm --cached`), and then fixes four classic beginner mistakes the WRONG → RIGHT way. *(Roman Urdu: .gitignore wo list hai jo Git ko kehti hai — ye files kabhi history mein nahi jayengi. Secret wali file to bilkul nahi.)*

Key points

- **`.gitignore` is a plain file at the repo root, itself committed:** lines are patterns — `node_modules/` (a folder), `*.log` (every log file), `.env` (one exact file), `dist/` (build output). Blank lines and `# comments` are allowed.
- **It only stops NEW tracking:** a file Git already tracks keeps being tracked even after you list it. The pattern is a door-lock, not an eviction.

  (Roman Urdu: Jo file pehle se track ho rahi hai, use ignore list rok nahi sakti — pehle use history se nikaalna parta hai.)
- **`git rm --cached <file>` evicts without deleting:** it removes the file from Git's index (the next commit drops it from the repo) while the file itself stays safely on your disk.
- **A pushed secret is a public secret:** deleting it in a later commit does NOT remove it from history — anyone can dig it out. Rotate (change) the password/key itself; that is the only cure.
- **`git status --ignored` shows the invisible:** it lists which files the patterns are catching — your proof the ignore list actually works.

Code examples

Example 1 — A real .gitignore, and status proving it works

```
$ cat .gitignore
# Dependencies — kabhi commit nahi
node_modules/

# Build output
dist/

# Secrets — SAB SE ZAROORI
.env

# Editor aur OS junk
.vscode/
*.log
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

What it does: the project HAS `node_modules/`, a `dist/` folder and a `.env` file on disk — yet `git status` shows only `.gitignore` itself as untracked. Every ignored file is invisible to Git, exactly as intended. Commit `.gitignore` once, and the protection travels with the repo to every clone.

What you will see: before this file existed, status drowned in thousands of `node_modules` lines; now it is one calm line. Run `git status --ignored` and the ignored names appear in their own grey section — listed, but never staged.

Example 2 — The file is ALREADY tracked: git rm --cached

```
$ git rm --cached .env
rm '.env'
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    .env

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .env
$ git commit -m ".env ko tracking se hataya — ab ignore mein hai"
[main e5f6g7h] .env ko tracking se hataya — ab ignore mein hai
 1 file changed, 34 deletions(-)
```

What it does: `.env` was committed BEFORE the ignore rule existed, so Git kept tracking it. `git rm --cached` removes it from the index only: the next commit records it as deleted-from-repo, while the real file stays on disk (it shows as "untracked" — and your `.env` pattern now keeps it untracked forever). Your app keeps running; the repo stops carrying the secret.

What you will see: open the folder — `.env` is still there, untouched, with your keys inside. And the honest warning: if that old commit was ever PUSHED, the secret still lives in the online history — change the password itself. `--cached` fixes the future, not the past.

Example 3 — Mistake: committing before writing the message properly (the four fixes in action)

```
$ git log --oneline
e5f6g7h (HEAD -> main) .env ko tracking se hataya — ab ignore mein hai
f6g7h8i asdf
$ git commit --amend -m "Setup: project files aur pehla page"
[main g7h8i9j] Setup: project files aur pehla page
 Date: Tue Jan 12 10:24:00 2027
 2 files changed, 23 insertions(+)
```

What it does: this is one of the four classic fixes below — a useless message ("asdf") on a commit that has NOT been pushed yet, repaired with `git commit --amend`, which rewrites the last commit with the proper message. (Rewriting is safe only before pushing — after a push, history is shared property; G.7 draws that line precisely.)

What you will see: `git log` now shows the clean message in place of "asdf", with a NEW commit ID — the old one is gone, which is exactly why amend is a local-only tool.

Quick recap

- `.gitignore` at the root, committed once: patterns for folders (`node_modules/`), wildcards (`*.log`) and exact files (`.env`). It blocks new tracking only.
- Already-tracked file? `git rm --cached` removes it from Git, keeps it on disk — then the ignore pattern guards it.
- A pushed secret is compromised forever: rotate the secret itself, then fix the tracking.

Common mistakes

1. **WRONG:** adding `.env` to `.gitignore` and assuming you are safe. **RIGHT:** check if it is already tracked (`git ls-files .env`) — if yes, `git rm --cached .env`, commit, and rotate any key that was ever pushed.
2. **WRONG:** committing `node_modules` "so the project runs anywhere" — thousands of files, giant repo, and it still is not how deployment works. **RIGHT:** ignore it; `package.json` + `npm install` rebuilds it on any machine in a minute.
3. **WRONG:** one message for everything — "update", "fix", "asdf" — a history nobody can read. **RIGHT:** the message says what and why, in the pattern of a story chapter: "Add login form validation" beats "changes".
4. **WRONG:** running `git add .` blindly in a new project, sweeping junk and secrets into the first commit. **RIGHT:** write `.gitignore` BEFORE the first add, and let `git status` show you the exact list that is about to be staged.

Practice

1. In a practice repo, create `node_modules` (a folder with two dummy files), a `.env` with a FAKE password, and an `app.js`. Write the `.gitignore` from Example 1, then prove with plain `git status` that only `.gitignore` and `app.js` are untracked.
2. Commit `app.js` and `.gitignore`. Now deliberately commit the `.env` (add it with `git add -f .env`), then rescue the repo: `git rm --cached .env`, commit, and verify with `git ls-files` that `.env` is gone from tracking while the file still exists on disk.

Practice quiz

1. What do the .gitignore patterns node_modules/, *.log and .env each mean?

- They delete those files
- A folder, every log file, and one exact file, to be ignored
- They are comments only
- They force those files to be committed

Answer: A folder, every log file, and one exact file, to be ignored

2. A file is already tracked by Git, and you now add it to .gitignore. What happens?

- Git stops tracking it at once
- Nothing, it only stops NEW tracking; the tracked file keeps being tracked until you evict it with git rm --cached <file>, which keeps it on your disk
- The file is deleted from your disk
- Git ignores it only on GitHub

Answer: Nothing, it only stops NEW tracking; the tracked file keeps being tracked until you evict it with git rm --cached <file>, which keeps it on your disk

3. A secret was pushed to GitHub. What is the only real fix?

- Delete it in a later commit, history will forget it
- Rotate (change) the password or key itself; deleting it later does not remove it from history, anyone can dig it out
- Rename the file
- Add it to .gitignore afterwards

Answer: Rotate (change) the password or key itself; deleting it later does not remove it from history, anyone can dig it out

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/git-gitignore-mistakes · More free tools and notes at https://scopetreasure.com