.gitignore: The Files That Must Never Reach GitHub

Keep client data, secrets, and local junk out of your repo

Not everything in your project folder belongs on GitHub. Some of it is harmless clutter. Some of it is a career-ending mistake waiting to happen. `.gitignore` is how you draw that line, and for a lawyer it is not optional.

What it is

`.gitignore` is a plain text file you put at the root of your repo. Each line is a pattern for files Git should pretend do not exist: never track them, never stage them, never push them. Git reads it automatically.

```
# .gitignore
.env                  # secrets and API keys
node_modules/         # downloaded dependencies, huge and rebuildable
data/                 # your local database and client files
*.log                 # log files
.DS_Store             # macOS folder junk
subscribers.jsonl     # anything holding personal data
```

The three things it keeps out

Secrets. Your `.env` file holds API keys, database passwords, SMTP credentials. Push one to a public repo and bots scrape it within minutes; people have run up thousands of dollars in charges this way before lunch. `.env` goes in `.gitignore` on day one, every time.

Client data. This is the lawyer-specific one, and it is the one that ends careers. A SQLite database with client records, an intake export, a document with a real name in it: if it lands in a public repo, you have published privileged information to the entire internet, and Git remembers it even after you delete the file. Your duty of confidentiality (ABA Model Rule 1.6) does not pause because you were moving fast. Keep client data out of the repo entirely; `.gitignore` is your backstop.

Bulk and junk. Downloaded dependencies (`node_modules/`), build output, operating-system cruft like `.DS_Store`. None of it belongs in your history; all of it makes the repo bloated and noisy.

Set it up before the first commit

Order matters. `.gitignore` stops Git from tracking files it is *not already* tracking. If you commit `.env` first and add the ignore rule second, the rule does nothing; the file is already in the history. So the sequence is: create the repo, write `.gitignore`, *then* start committing.

If something sensitive already slipped in, ignoring it going forward is not enough. You have to remove it from tracking, and for real secrets you rotate them (change the key or password), because anyone who pulled the repo already has the old one.

```
git rm --cached .env              # stop tracking it (keeps your local copy)
echo ".env" >> .gitignore         # and ignore it from now on
git commit -m "Remove .env from tracking"
```

That gets it out of future commits. It is still back in the old history; for genuine secrets, assume they are burned and rotate them.

A starter you can paste

For most small projects:

```
.env
.env.*
*.log
data/
node_modules/
__pycache__/
.DS_Store
Thumbs.db
```

GitHub also offers ready-made `.gitignore` templates per language when you create a repo. Start from one, then add your own client-data and secrets lines on top.

The playbook

Write `.gitignore` before your first commit. Secrets and client data go in it without exception. If something sensitive already got committed, rotate it and treat the old version as public. The file that never reaches GitHub cannot leak from GitHub.