Branches: Change Things Without Breaking What Works

The one idea that turns 'I'm scared to touch it' into confidence

Here is the fear that keeps people from using Git well: "If I change this and it breaks, I've wrecked the working version." Branches are the answer to exactly that fear.

A branch is a parallel copy of your project where you can change anything, break anything, experiment freely, and none of it touches the version that works until you say so.

The mental model

Picture your project's history as a line of commits. The main line is called `main`. When you create a branch, you split off a second line that starts from the same point. You do your work over on the branch. `main` sits untouched, exactly as it was, still deployable, still safe.

When the work on the branch is good, you *merge* it back into `main`. If the work turns out to be a bad idea, you throw the branch away and `main` never knew it happened.

The moves

```
git switch -c intake-redesign      # create a branch and move onto it
# ...make your changes, commit them on the branch...
git switch main                    # hop back to the safe version anytime
git switch intake-redesign         # and back to your work
```

You can jump between `main` and your branch whenever you want. Client calls with an emergency and you need the working version right now? `git switch main`, you are back on solid ground, your half-done experiment waiting untouched on its branch.

Merging the work back

Once the branch is solid, fold it into `main`:

```
git switch main
git merge intake-redesign          # bring the branch's work into main
```

On GitHub itself, this usually happens through a Pull Request, which is a merge with a review step attached. That is its own topic; the point here is that merging is how good work on a branch becomes part of the real thing.

Why this matters for a lawyer who codes

You are not a full-time developer with a QA team behind you. You are shipping tools you also rely on in a live practice. Branches let you try the risky change, the new integration, the "what if I restructure this" idea, without gambling the version that is currently filing your documents. Break things on a branch; keep `main` boring and working.

Delete branches when you are done

A merged branch has served its purpose. Clean it up so your repo does not fill with stale lines.

```
git branch -d intake-redesign      # delete a merged branch
```

No terminal? Same idea

GitHub Desktop has a "Current Branch" menu at the top: click it to create a new branch, switch between branches, or merge one into another. The concept is identical; you are clicking instead of typing.

The playbook

```
git switch -c NAME     # branch off and start experimenting
git switch main        # jump back to the safe version anytime
git merge NAME         # fold good work back in
git branch -d NAME     # delete the branch when done
```

Keep `main` working. Do the scary stuff on a branch. That is the whole habit.