PLAIN-ENGLISH REFERENCE

GitHub, the 5 things that matter

Everything that happened with your PR #1, explained without the jargon.

Built for Anthony · ties every term back to your real `Discord-and-Slack-Agents` repo

TL;DR

- **A repo** is the project; **main** is the official, live copy of it that everyone shares.
- You never edit **main** directly. You make a **branch** (your own copy), **commit** changes to it, then open a **pull request** asking to merge them back.
- The whole loop is: **branch → commit → pull request → review → merge**, then repeat.
- The one habit: keep each PR **small** and **merge it fast**. That is the entire lesson from PR #1.
- Your branch sat too long and drifted from **main**, so it "conflicted" and Maddie had to fix it by hand. Small + frequent PRs never hit that.

The loop, start to finish

1 **Branch** Make your own copy of the code to work in, safely away from `main`.

↓

2 **Commit** Save a snapshot of your change, with a short note on what you did.

↓

3 **Pull Request** Propose folding your branch into `main`. Opens it up for others to see.

↓

4 **Review** A teammate reads it, comments, and approves (this is where Maddie came in).

5 **Merge** Your change is folded into `main`. It is now part of the official code.

Then you start over for the next change. **Small loop, run often** beats one giant branch that sits for weeks.

The 5 words, defined

Repository

AKA "REPO"

The whole project: every file plus its complete history of changes.

Yours: `emazingb/Discord-and-Slack-Agents---Claude`

Commit

A SAVED SNAPSHOT

One saved change with a note attached. Like a checkpoint you can always return to.

Maddie's landed as `c678786`

Branch

YOUR WORKSPACE

A separate copy of the code to experiment in without touching the live version. `main` is the official branch everyone trusts.

Pull Request

AKA "PR"

A request that says "please pull my branch into `main`." It is where review and discussion happen before anything lands.

Yours was PR #1

Merge

THE FINISH LINE

Accepting a pull request: your branch's changes get combined into `main` and become official.

main

THE SOURCE OF TRUTH

The one branch that holds the real, shipped code. Brian's point: everyone should work off the same `main` so copies don't drift apart.

The one habit to build

IF YOU REMEMBER NOTHING ELSE

Branch → one small change → open a PR fast → merge → repeat.

Do not let a branch sit for weeks. The longer it sits, the more `main` moves on without it, and the harder it is to fit your work back in. Small PRs review fast, merge clean, and never pile up into a conflict someone else has to untangle.

What Maddie's message actually meant

`conflicting`	Your branch was built off an **old** version of `main`. While it sat, `main` changed, and your edits collided with the newer edits. Git can't auto-pick a winner, so a human has to.
`rebase`	One fix for that: **replay your whole branch on top of the new `main`**, change by change. With ~2,800 lines, that's a lot of collisions to resolve by hand. Painful, which is why she chose the other path.
`cherry-pick`	The path she took: **hand-pick just the useful commits** from your branch and drop them straight onto `main`, skipping the messy parts. Your good work landed without the conflict mess.
`closed the PR`	**Not a rejection.** Your changes were already on `main` (via that cherry-pick at `c678786`), so there was nothing left to merge. The PR's job was done.

Bottom line: nothing went wrong with your *code*. The only issue was timing. The branch sat long enough to drift. Smaller, faster PRs are the cure, and that's exactly what Brian was nudging you toward.

Generated by Claude · 2026-06-25