The Non-Geek's Guide to AI Development Tools

Setting up GitHub and choosing your AI-enabled coding toolset

Tris Hussey | TrisHussey.com ©2026

This guide covers the technical "boring stuff" mentioned in Step 3 of *How to Build Your Own Web App in Five Easy Steps*. This guide will:

1. Set up GitHub to backup and save your work
2. Pick and install an AI-enabled code editor to do the heavy lifting
3. Start building your project

1. Setting up GitHub

GitHub is where your code lives in the cloud. It is your "save-game" point. If the AI hallucinates and breaks your app, you can use GitHub to "revert" back to the version that worked an hour ago.

Step-by-step setup:

1. **Create an Account:** Go to GitHub.com and sign up. Using your Google ID is the fastest way.
2. **Download "GitHub Desktop":** This is the "plain English" version of GitHub. It lets you manage your code without needing to use the command line. You *should* become passingly familiar with using GitHub on the command line —sometimes you'll need to clean something up *or* when your IDE wants to run a git command, you know what it's doing.
    - **Mac Users:** Download the .zip, drag the app to your **Applications** folder.
    - **Windows Users:** Run the .exe installer.
3. **Sign In:** Open the GitHub Desktop app and sign in with the account you just created.
4. **The Commit Habit:** Whenever you finish a small task, open GitHub Desktop, write a short note (e.g., "Added a login button"), and click **Commit to Main and Push to Origin**. This creates your backup and a "check point" for you in case you have an "oops" moment.

Essential GitHub terms

Before diving more, let's de-geek the terms you're going to come across. You will see these terms all the time in chats with your AI agents:

- **Repository (Repo):** This is your project's container. It's simply the folder that holds all your code, assets, PRDs, and the historical "save files" of your project. It should mirror what's on your computer.
- **Commit:** A "save point" in your project's timeline. When you "commit" code, you are telling Git, *"Take a snapshot of exactly how my files look right now and save it."*
- **Branch:** A parallel universe for your code. The baseline code lives on the main branch. If you want the AI to try something risky without breaking your working app, it can create a new branch. Once the feature works, you merge it back into main. This is where I got really confused and thought I needed to do Pull requests (you don't on

your own stuff), merging things back to main, and deploying to Vercel where a lot of my app live. If you get confused—ask your AI.

- **Main (or Master):** The “main” version of your app. This is the master copy that should ideally always work. Once you have an app working and are doing anything other than minor bug fixes (and even then maybe), make a *branch* from main. You can always cut off and abandon a branch with “well that didn’t go as planned” and your working app isn’t a smoldering hunk of code that you have to roll back to the “pre-smoldering” point.
- **Fork:** Copying *someone else’s* entire project repository into your own GitHub account so you can play with it, modify it, and build on top of it without changing their original work.
- **Push:** Sending your local commits (saved on your computer) up to the cloud on GitHub.com so they are securely backed up. I always commit and push. Maybe I’m wrong, but I like to know what I did is backed up on to the cloud.
- **Pull:** Downloading the latest changes from the cloud to your local computer (vital if you collaborate with others or work on multiple devices).
- **Clone:** Downloading a copy of a remote GitHub repository to your local machine for the very first time.

How to create your first repository (using GitHub Desktop)

You don't need to touch code to spin up a repository. Here is how you do it using the Desktop app:

1. Open the **GitHub Desktop** app.
2. Go to the top menu and select **File > New Repository** (or click the **Create a New Repository on your Local Drive** button on the welcome screen).
3. Fill in the details:
    - **Name:** new-app (Remember: lowercase, no spaces, use hyphens!).
    - **Local Path:** Choose your projects folder (e.g., C:/projects or /Users/username/projects).
    - **Git Ignore:** Select **Node** or **Python** depending on your build type (or leave it as None if you're not sure yet). This file tells Git to ignore useless background files. I usually ask my AI IDE to update and edit my .gitingore file as I go.
    - **License:** Leave as None.
4. Click **Create Repository**.
5. Now, look at the top bar and click **Publish Repository**.
    - Make sure **"Keep this code private"** is checked!
    - Click **Publish**. Your project safety net is officially live in the cloud.

GitHub cheat sheet: Basic git commands you'll see your IDE running (and some you can use yourself).

When your AI starts running terminal scripts or when you open up a command line inside your IDE, you will see some basic command line instructions fly by. Here is what they actually mean:

Command	What It Actually Does	Why You See It
git init	Turns a standard local folder into a tracked Git repository.	Creates hidden tracking files to monitor structural changes.
git clone <url>	Downloads an existing project from the cloud to your local machine.	Used when pulling down a template or an existing repository.
git status	Displays modified, added, or deleted files since your last commit.	Checks what has changed before saving progress.
git add .	Prepares all modified files to be saved (the . means "everything").	Bundles changes together for the upcoming snapshot.
git commit -m "msg"	Captures a local timeline snapshot with a descriptive text message.	Finalizes your local "save point".
git push origin main	Ships your local commits up to the main branch in the cloud.	Ensures your work is safely backed up off your machine.
git pull	Grabs the newest updates from the cloud and merges them locally.	Syncs your computer with any external cloud updates.
git checkout -b <name>	Creates a new branch and immediately switches your workspace into it.	Used to test risky features or major overhauls safely.

2. Picking the right IDE for you

An IDE (Integrated Development Environment) is just a fancy word for a text editor that is "smart" enough to talk to AI. Here are the top four choices for 2025:

Cursor

- **Who it's for:** People who want the most powerful AI features right now.
- **The Vibe:** It feels like a standard text editor but has a "Composer" mode that can write entire apps for you across multiple files.
- **Website:** cursor.com
- **Cost:** Free for basic use; ~$20/mo for the "Pro" version (worth it if you're building seriously).

I tried it early on, had no clue what I was doing, so I abandoned it for VS Code, then VS Code for Antigravity. A lot of people love it. I didn't get it at first, so...

Antigravity 2.0 & Antigravity IDE

- **Who it's for:** If you're using the Google ecosystem, especially if you're paying for a Google One AI plan, this is a solid, solid choice. It's easy, pretty fast, and it works. Users tied into the Google ecosystem who want a familiar but highly capable AI workspace.
- **The Vibe:** Google changed a lot about Antigravity recently to make it more like Cursor or Claude, *instead* of a geeky looking coding tool *and* a simple panel to chat with an AI, it's all just the "chat with an AI" part. You can download the geeky coding part separately as Antigravity IDE (I did after Google updated things). Antigravity is optimized for Google's models and integrations, but it can also use Claude's and OpenAI's GPT-OSS models.
- **Website:** antigravity.google
- **Cost:** Competitive with other AI-first IDEs, generous free tier, can be combined with a Google One AI subscription.

Windsurf

- **Who it's for:** Beginners who want a more generous free tier and very fast performance. Again, geekier. And also again, I tried it, got confused because my development environment was a mess, and once I ran out of free use one day; I didn't come back.
- **The Vibe:** Very similar to Cursor. It uses a feature called "Cascade" to act as your AI agent.
- **Website:** windsurf.com
- **Cost:** Often has a lower entry price for Pro features than Cursor.

VS Code & Copilot

- **Who it's for:** People who want the most "stable" and widely used tool. Until really recently, this was what a lot of (most?) developers worked in. It has a huge ecosystem of extensions that Antigravity IDE (and other IDEs based on the open source version of its code) can use. You can't go wrong trying it, it's just more that a little geeky.
- **The Vibe:** This is the base tool that Cursor and Antigravity were built upon. It's powerful but requires a bit more "plugging things in" to get it working like an AI agent.
- **Website:** code.visualstudio.com

3. Putting it all together

You've set up GitHub. You've picked your IDE. Now, let's put this all together so you can *actually* start building stuff instead of just reading all these words.

Starting your first project the right way (so you don't create a mess)

You might have created your first repo (cool!) in the GitHub desktop app or not. Either is okay. Just pick what you did so you don't wind up confusing yourself and all your files. BTW, I went down the confused files path and it took a bit to get things straight (and I'm still learning).

- **Track A: If you already made a repo via GitHub Desktop:** In your IDE, go to File > Open Folder. Select the exact project folder created by GitHub Desktop. If prompted to "Trust the authors," click **Yes**. Your IDE will automatically detect the Git settings and begin tracking your workflow.
- **Track B: If you are starting completely fresh from the IDE:** Create a clean folder on your machine first (e.g., projects/new-app). Open it via File > Open Folder in your IDE. Navigate to the **Source Control** tab (the three-pronged branch icon) and click **"Initialize Repository"**.

For Cursor, Antigravity, or Windsurf:

1. **Download:** Go to the official site for your chosen tool (listed in the links above).
2. **Install:** Follow the standard installer for Mac or Windows.
3. **Onboarding:** When you open the app, it will ask if you want to import settings. If you're brand new, choose **"Start Fresh."**
4. **Login:** Sign in with your email or GitHub ID to enable the AI features. (For Antigravity, use your primary Google account).

In Antigravity 2.0, there's no "set up GitHub" step. If you've set up GitHub on your machine, Antigravity will find it and use all the settings. It's more tied in than I thought at first because it "sees" everything you do in the desktop or command line (like are you on main or a branch in the app).

For VS Code:

1. **Download:** Go to code.visualstudio.com.
2. **Install Extensions:** Click the "Blocks" icon on the left sidebar. Search for and install **"GitHub Copilot"** and **"GitHub Copilot Chat."**

> **Tip**
> Pro Tip for Mac Users
> As mentioned in the workflow guide, don't put your projects folder inside your iCloud folder. I know it seems smart and convenient—it isn't. Put it directly in your "Home" folder or "Documents" (if iCloud is turned off for Documents). This prevents weird syncing errors that drive AIs crazy. I once spent almost an hour trying to get GitHub to sync something because it couldn't stay in sync with iCloud.

4. Now just go build stuff

Yep, that's the end of this guide. I hope you're building something awesome. Don't forget to keep in touch on Substack (GenerallyAI and 30 Plus Days of AI), my website, or LinkedIn.