ONE-PAGE SETUP PATH

Luke's Beginner Guide to Codex, Git, and GitHub

The goal is simple: do the setup once, connect your GitHub account, then mostly work through the Codex and GitHub screens instead of typing commands.

Download the PDF version

What You Are Setting Up

Codex

The AI coding agent. Use it when you want to build, edit, test, or troubleshoot real files on your machine.

Git + GitHub

Git tracks versions on your computer. **GitHub** stores those versions online so you can recover, share, and publish them.

Local Folder

This is where the project lives while you work. Do not use Dropbox as the main working folder. GitHub is the backup/sync point.

Permissions

Codex can ask before changing things. Start cautious. Give broader permissions only when you trust the task and the folder.

Step 1: Accounts You Need

[ ] A GitHub account you can log into: github.com

[ ] A ChatGPT account with Codex access, or an OpenAI API key if you want usage-based billing.

[ ] A password manager or at least a reliable way to keep your GitHub, ChatGPT, and API passwords/tokens safe.

> **Best beginner choice:** sign into Codex with ChatGPT. Use an API key later for automation or advanced workflows.

Step 2: Install the Basic Tools

Use the normal installers wherever possible. The point is not to become a terminal person; the point is to give Codex the tools it needs behind the scenes.

1. Install **Codex** from the Microsoft Store.
2. Install **Git for Windows** from git-scm.com/download/win.
3. Install **VS Code** from code.visualstudio.com.
4. Optional for later: install **GitHub Desktop** from desktop.github.com if you want a visual way to commit, push, and manage branches.

If an installer asks for choices, accept the defaults. After installing, restart Codex so it can detect the tools.

Step 3: Sign In and Connect GitHub

1. Open the Codex app.
2. Sign in with your ChatGPT account. This is the easiest path for normal use.
3. In Codex, open settings or connections and connect your GitHub account if the app prompts for it.
4. In GitHub, approve the connection for the repositories you want Codex to use.

If Codex cannot see GitHub later, then use the command fallback at the bottom. But try the app connection first.

Step 4: Make or Open a Project

Use a normal local folder. This keeps Codex, Git, and GitHub working together cleanly.

1. Create a folder in File Explorer, for example `Documents\GitHub\rent-everything`.
2. Open Codex and choose **Add project** or **Open folder**.
3. Select that folder.
4. When Codex asks about permissions, start with the default setting where it asks before risky actions.

If the project already exists on GitHub, use Codex or GitHub Desktop to clone/open it locally instead of starting from an empty folder.

Step 5: Let Codex Prove It Works

In Codex, start with a small test prompt:

```
Create a README.md for a simple app called Rent Everything.
Create or use a dev branch first.
Do not push until I approve.
Show me what changed when you are done.
```

Codex should show changed files in the review/diff screen. If you can see the new `README.md`, the basic loop is working.

Step 6: Put It on GitHub Without Living in Commands

1. Go to GitHub and create a new repository, for example `rent-everything`.
2. Use Codex, GitHub Desktop, or VS Code Source Control to publish the local folder to that repository.
3. Use a `dev` branch for work in progress.
4. When the change looks good, open a pull request on GitHub and merge it into `main`.

The command line can do all of this, but Luke's normal workflow should be Codex for changes and GitHub/GitHub Desktop for review, branches, commits, and pushes.

The Workflow to Remember

Daily GUI Build Loop

1. Open the project folder in Codex.
2. Tell Codex the exact goal.
3. Let it change files and run checks.
4. Review the changed files in Codex.
5. Commit/push from Codex, GitHub Desktop, or VS Code.

Good First Prompt

```
I want to build [thing].
Use the current folder.
Work on a dev branch.
Explain before installing packages.
Run a basic test or smoke check.
Stop before pushing.
```

> **Do not start with full access.** Use approval prompts until you understand what Codex is changing. Full access is powerful, but a bad prompt or wrong folder can make a mess quickly.

Terms That Were Confusing in the Demo

- **Terminal / PowerShell:** the backup place where commands run when the GUI cannot finish something.
- **VS Code:** a good editor for reading files, Markdown, and code.
- **Markdown:** plain text files ending in `.md`; readable by people and easy for AI to use.
- **Branch:** a working copy of the project. Use `dev` for changes before they go live.
- **Pull request:** the GitHub review step before merging branch changes into `main`.
- **Supabase:** a database service. Codex can help write SQL and scripts, but protect API keys and start with manual approval.

When Something Gets Stuck

1. Check whether Codex is waiting for approval.
2. Look at the changed files in Codex, VS Code, or GitHub Desktop.
3. Make sure Codex opened the correct project folder.
4. If GitHub fails, reconnect GitHub in Codex or GitHub Desktop.
5. If the task got too big, start a new Codex thread with a smaller request.

What Success Looks Like

- [ ] Codex opens and is signed in.
- [ ] Codex can open the project folder.
- [ ] GitHub is connected to Codex or GitHub Desktop.
- [ ] You have a local project folder under `Documents\GitHub`.
- [ ] Codex can create or edit a file in that folder.
- [ ] You can commit and push that file to GitHub.

Only If the GUI Gets Stuck

These are backup commands, not the normal workflow. Use them when Codex, GitHub Desktop, or VS Code cannot tell what is happening.

```
codex --version
git --version
git status
gh auth status
```

Built from Drew and Luke's transcript notes plus current OpenAI Codex documentation. Official references: Codex quickstart, Codex authentication, Codex Windows setup, Codex CLI, and AGENTS.md guidance.