The G-Stack Playbook

Set Up a 23-Tool Claude Code Ops Team Like Garry Tan's

What G-Stack actually is

Garry Tan, CEO of Y Combinator, open sourced the exact Claude Code setup he uses to run his own projects. It is called gstack: 23 specialised tools plus a handful of utilities that turn Claude Code from a single assistant into a full operations team, with agents dedicated to product strategy, design, engineering management, QA, and release.

The idea is simple. A solo founder or a small team should be able to operate like a full engineering org: someone questioning the product direction, someone catching bad design, someone protecting the architecture, someone testing in a real browser before release, and someone actually shipping it. gstack gives each of those jobs to a dedicated agent and exposes them as slash commands.

It installs with one command, it is MIT licensed with no premium tier, and it works on Claude Code, Codex, Cursor, and seven other agents. This guide covers how it works, the install for Claude Code and for Codex, the commands I run, and what to do when it breaks.

01

How it actually works

gstack structures work as a sprint: Think, Plan, Build, Review, Test, Ship, Reflect. Each phase is a slash command backed by an agent with its own instructions.

1. You start with a rough idea and run **/office-hours**. The CEO style agent interrogates the product thinking, pushes back if it is thin, and produces a design doc. Then **/plan-ceo-review** and **/plan-eng-review** lock the scope and the architecture (or run **/autoplan** to do CEO, design, and eng review in one go).
2. You approve the plan and Claude builds it. Then **/review** runs a staff engineer code review and auto fixes the obvious issues, **/design-review** audits the UI, and **/qa** opens a real Chromium browser, clicks through the app, finds bugs, fixes them, and writes regression tests.
3. **/ship** syncs, runs the tests, audits coverage, pushes, and opens the PR. **/land-and-deploy** merges, waits for CI, deploys, and checks production health. **/canary** watches for console errors afterwards, and **/retro** gives you a weekly retrospective.

Five roles do most of the heavy lifting, and each one intercepts a different failure point in the normal "just prompt Claude and hope" workflow:

- **CEO agent** catches weak or unclear product direction before work starts. It runs first.
- **Designer agent** catches generic UI, inconsistent spacing, and off brand visuals instead of letting them ship silently.
- **Engineering manager agent** holds the line on architecture, so a quick fix in month three does not quietly undo a decision from month one.
- **QA agent** opens an actual browser, clicks through the app the way a user would, and reports what breaks.
- **Release manager agent** owns the ship step, so deploying is not a separate manual task you have to remember to do carefully.

Once installed the stack works passively too. You do not have to name an agent for every task. Ask for a feature in plain language and it routes through the relevant specialists.

02

The one thing almost everyone gets wrong

The most common mistake is installing gstack and then still telling Claude Code to skip a step "just this once". The value comes from letting every agent run every time, especially QA and the release manager on anything going to real users.

> **WATCH OUT**
>
> The QA agent is the one solo builders skip most, and it is the one that catches the bugs code review and linting never will, because it opens an actual browser and uses the app. Do not let a tight deadline talk you out of it. And the CEO agent runs first, before any code is written; if you find yourself running /office-hours after the build, you have it backwards.

03

The 23 tools, grouped by phase

The full list, so you know what you are installing. Every one of these is a slash command inside Claude Code once setup has run.

> **SOURCE CODE**
>
> The repo, the setup script, and the full docs:
>
> **https://github.com/garrytan/gstack**

PHASE	COMMANDS
Plan	/office-hours, /plan-ceo-review, /plan-eng-review, /plan-design-review, /plan-devex-review, /autoplan
Design and build	/design-consultation, /design-shotgun (4 to 6 mockup variants), /design-html
Review and test	/review, /design-review, /devex-review, /qa, /qa-only, /investigate, /cso (OWASP Top 10 and STRIDE threat model), /codex (independent review from the Codex CLI)
Ship and monitor	/ship, /land-and-deploy, /canary, /benchmark
Document and reflect	/document-release, /document-generate, /retro, /learn
Browser and safety	/browse (real Chromium with screenshots), /open-gstack-browser, /setup-browser-cookies, /careful, /freeze, /guard, /unfreeze
Setup and upkeep	/setup-deploy, /setup-gbrain, /sync-gbrain, /pair-agent, /gstack-upgrade

Because everything is open source, every agent's instructions are editable text. You can tighten what the designer flags, loosen what the engineering manager enforces, or add your own specialist next to the defaults. My advice: run the defaults for two weeks first, then you will know what actually needs changing.

Set it up in Claude Code or Codex

Garry calls it a 30 second install. Budget five minutes the first time. You need Claude Code (or Codex), git, and Bun 1.0 or newer. On Windows you also need Node.js.

1. Check the prerequisites in a terminal: `git --version` and `bun --version`. If Bun is missing, install it from bun.sh (one curl command), then open a new terminal.
2. **Claude Code install.** Open Claude Code in any project and paste this as a message. Claude runs it for you:
   `git clone --single-branch --depth 1 https://github.com/garrytan/gstack.git ~/.claude/skills/gstack && cd ~/.claude/skills/gstack && ./setup`
3. The setup script registers all 23 tools at once. There is no per tool config to write. When it finishes, add a short gstack section to your project's `CLAUDE.md` that lists the available skills (setup prints the text to paste).
4. **Codex install.** Clone once to a shared folder, then point setup at Codex:
   `git clone --single-branch --depth 1 https://github.com/garrytan/gstack.git ~/gstack && cd ~/gstack && ./setup --host codex`
   Setup auto detects installed agents; the `--host` flag forces one. Other values: `cursor`, `opencode`, `factory`, `kiro`, `hermes`.
5. Restart the session. Skills are read at session start, so close Claude Code or Codex and open it again inside your project. Type `/` and you should see the gstack commands in the list.
6. Optional but recommended for a team: from your repo run
   `(cd ~/.claude/skills/gstack && ./setup --team) && ~/.claude/skills/gstack/bin/gstack-team-init required && git add .claude/ CLAUDE.md && git commit -m "require gstack for AI-assisted work"`
   That commits the config so everyone on the repo gets the same agents.
7. Optional memory: run `/setup-gbrain` for a persistent knowledge base. Pick local PGLite if you want zero accounts (about 30 seconds). Then `/sync-gbrain` indexes your code.
8. First real run: open a project, describe a small feature, and run **/office-hours**. If the agent starts asking you forcing questions about the product instead of writing code, the install worked.

> 💡 **QUICK TIPS**
>
> - Prefer short commands like /qa over /gstack-qa? Run ./setup --no-prefix. Want them namespaced? ./setup --prefix.
> - To remove it later: ~/.claude/skills/gstack/bin/gstack-uninstall, then delete the gstack section from CLAUDE.md.

---

04 The commands I actually run

A full loop on a real feature looks like this. Paste each one in order and swap the bracketed part.

> **1. START WITH THE IDEA, NOT THE CODE**
>
> ```
> I want to build [ONE SENTENCE: WHAT AND FOR WHOM].
> /office-hours
> ```

> **2. LOCK SCOPE AND ARCHITECTURE**
>
> ```
> /plan-ceo-review
> /plan-eng-review
>
> Or in one step:
> /autoplan
> ```

> **3. REVIEW WHAT GOT BUILT**
>
> ```
> /review
>
> Fix everything you flagged as obvious. List anything that needs my approval before you
> change it.
> ```

> **4. TEST IT IN A REAL BROWSER**
>
> ```
> /qa [STAGING URL]
>
> Click through the [FEATURE NAME] flow as a new user. Fix what breaks and add a
> regression test for each fix.
> ```

> **5. SHIP IT**
>
> ```
> /ship
>
> Then, once the PR is green:
> /land-and-deploy
> /canary
> ```

When it breaks

Every failure from the repo's own troubleshooting list plus the ones I hit, with the fix.

ERROR	FIX
Skills do not appear in the / list	Re-run setup: cd ~/.claude/skills/gstack && ./setup. Then open a new session; an old session will not pick them up.
/browse or /qa fails to open a browser	cd ~/.claude/skills/gstack && bun install && bun run build. The browser tooling needs Bun 1.0 or newer.
bun: command not found	Install Bun from bun.sh, then open a new terminal so the PATH change is picked up.
Codex says it cannot load the skill	cd "${CODEX_HOME:-$HOME/.codex}/skills/gstack" && git pull && ./setup --host codex
Stale install, commands behave oddly after an update	Run /gstack-upgrade, or set auto_upgrade: true in ~/.gstack/config.yaml so it keeps itself current.
Windows: skills stop working after git pull	Windows gets frozen copies, so re-run cd ~/.claude/skills/gstack && ./setup after every pull.
Claude skipped straight to writing code	Run /office-hours first and say so explicitly: "Do not write code until the plan reviews are done." The agents only guard the steps you let them run.