

Github-ing

Varun Chوريا · 2026-07-16 · <https://www.varunchoria.com/github-ing/>

The world is moving toward systems-first work, and I do not think most people understand what that means yet.

Every room has some version of the same conversation.

What skills can we build? Which workflows can we connect? Can one agent hand work to another? Can the whole process run without 6 people chasing each other across Slack, email, and a spreadsheet called `final_v7`?

These questions used to live with engineers, operations people, and the one person in every company who enjoyed Zapier a little too much.

Now they are table stakes.

The macro is the point

Capital is tighter. Customer wallets are tighter. Tools appear and disappear faster than procurement can finish reviewing them.

I do not believe every company is about to become a 10-person unicorn. I do believe every leadership team has now seen what unusually small, AI-native teams can produce. You cannot unsee that.

ARR per employee used to be finance hygiene. Now it is an operating philosophy.

The question inside companies has changed from “how do we make everyone more productive?” to “how much meaningful capacity can this system create per employee?”

Productivity is doing the same work faster.

Capacity is being able to take on more useful work without the organization collapsing under its own coordination cost.

That is a much bigger change.

[I wrote earlier](#) about leading a small team with a mandate to become AI-native. The goal was not to generate more documents. It was to increase the amount of directionally correct work we could move from idea to market.

Speed plus direction. The physical definition of velocity, applied to an organization.

This is also why I think a lot of middle management is in trouble.

Not all of it. Judgment, coaching, conflict resolution, and decision-making remain real work.

But a lot of management is information routing dressed up as leadership. Collect status. Reformat status. Present status. Send status back down with comments.

AI will attack that layer before it replaces good judgment.

Organizations will get flatter because the cost of coordination has become visible, and because machines are getting good at the administrative work that kept many layers busy.

That is the macro. GitHub is one of the consequences.

GitHub is where ideas become accountable

Put bluntly, GitHub is where ideas become accountable.

AI has made producing things dramatically cheaper. A nontechnical person can now build a calculator, draft a workflow, create a landing page, analyze a dataset, document a process, and deploy a working tool.

I still do not understand code beyond basic HTML.

But software has always had 2 different forms of literacy.

One is understanding code.

The other is understanding how software works: inputs, outputs, dependencies, failure states, users, and whether the thing creates an outcome anybody values.

AI has lowered the cost of the first one. It has not removed the need for the second.

The cost of producing code is collapsing. The cost of deciding what should exist is not.

GitHub is the modern Google Drive for people like me, except it does not have to become a graveyard of ideas that could have been.

A repository can show the work changing.

It can show what was added, what broke, what got removed, who decided, and why the current version exists.

That makes GitHub less like storage and more like connective tissue across tools, people, products, and decisions.

GitHub is not your database

There is a reasonable argument against putting everything in GitHub.

[Cursor built ChatGTM](#), an internal system that pulls live information across accounts, product usage, and CRM data, then synthesizes it for teams.

[Hightouch described another model](#), with GitHub acting as a system of record for documentation, logic, rules, and skills.

These can look like competing choices.

They are not.

Live customer and operational data should live in systems designed for live data. Your CRM, warehouse, product database, and analytics layer exist for a reason.

GitHub is better for the durable logic around those systems.

Definitions. Documentation. Agent skills. Workflow rules. Prompts that became infrastructure. Configuration. Decision records. The code connecting one system to another.

The live number belongs in the database.

The explanation of what the number means, who owns it, how it is calculated, and what should happen when it changes belongs somewhere versioned.

That distinction protects accuracy without making the company dependent on whichever AI tool is fashionable this quarter.

Documentation is part of the product

My opinion is that documentation itself is becoming a product.

Not documentation as a pile of meeting notes.

Documentation as the operating logic of the business.

Every organization is broken in its own way. Reporting lines are different. Dotted lines are different. Political constraints are different. Data quality is different. Even the definition of a qualified lead can trigger a minor civil war.

That uniqueness is exactly why one generic AI tool cannot become the operating system for every company.

The tools will change.

The durable layer is the business, technical, and economic logic describing how your company works.

Versioning changes how that logic can be managed.

You can review why a workflow changed. You can compare the old definition with the new one. You can see when a rule created an unintended consequence. You can schedule an agent to find duplicates, contradictions, dead steps, and instructions nobody follows.

More importantly, you can reverse a bad decision without reconstructing history from somebody's memory.

That is not developer hygiene.

That is organizational memory.

GitHub is also a portfolio

Portfolios used to belong to designers and engineers.

That excuse is gone.

The barrier to building a useful lead magnet, internal tool, research workflow, or small product used to be high enough that a marketer could reasonably say, "I had the idea, but I needed engineering."

Now many of these things can be built with a frontier model in an afternoon.

Not perfectly. Not always safely. Not without judgment.

But built.

Would I refuse to hire a marketing or GTM person because they do not have a GitHub profile? No.

Would the burden of proof be higher than it used to be? Absolutely.

A resume tells me what somebody was near.

A repository can tell me what they thought, what they attempted, what failed, and whether they returned to improve it.

That is much closer to proof of work.

[I wrote about this before](#): do the work once, automate it the next time, then document what happened. Over time, that becomes a portfolio without needing permission from a job title.

The old line says ideas are cheap and execution is everything.

Ideas are now cheaper than ever.

The valuable part is the trail between the idea and the outcome.

You do not need to become an engineer

This is not an argument that every person should learn computer science.

It is an argument that every knowledge worker should understand systems well enough to make their work inspectable.

Learn enough GitHub to create a repository.

Learn enough Markdown to document a decision.

Learn enough version control to see what changed.

Learn enough about software to know when the model is confidently wrong.

Then build something useful and leave a trail.

AI gave almost everyone execution leverage.

GitHub gives that execution memory, evidence, and accountability.

That is why it matters.