NEWS · DECEMBER 30, 2025

GitHub Is Not a Backup (And Why That Matters)

By Dr. Holger Flick

I came across something this morning that made me pause: a developer celebrating their move from daily ZIP files to GitHub, calling it their "backup solution." While I'm genuinely happy they've discovered version control, this statement reveals a dangerous misconception that's worth addressing.

Let me be clear: GitHub is not a backup. Version control is not backup. Full stop.

What Is GitHub (and Version Control)?

GitHub is a hosting platform for Git repositories - a version control system designed to track *changes* to your code over time. It helps you:

- Understand what changed, when, and why
- Collaborate with other developers without overwriting each other's work
- Create branches to experiment safely
- Review code before merging it into the main codebase
- Revert specific changes when bugs are introduced

GitHub excels at managing the *evolution* of your code. It's about workflow, collaboration, and maintaining a history of decisions. It's an incredible tool—but it's not designed to be your safety net against disaster.

What Is Backup?

Backup is about *disaster recovery and data protection*. A proper backup strategy ensures:

- Your data survives hardware failures
- You can recover from ransomware or malicious deletions
- You have offline or geographically distributed copies
- You can restore to a known-good state after catastrophic events
- Your data persists even if third-party services disappear

Backups protect against the unexpected: server failures, account compromises, service outages, accidental repository deletions, or even GitHub itself having problems.

Why GitHub Cannot Replace Backup

Here's the critical distinction: **GitHub assumes intentional changes; backup protects against unintentional loss.**

Consider these scenarios:

Scenario 1: Accidental force push You accidentally force-push over your main branch, losing weeks of work. GitHub won't save you here—you just overwrote your history. A backup would.

Scenario 2: Account compromise Someone gains access to your GitHub account and deletes all your repositories. GitHub's version history doesn't help when the entire repository is gone. A backup does.

Scenario 3: Corrupted repository Your local `.git` folder becomes corrupted, and you push the corruption to GitHub. Now both your local and GitHub copies are corrupted. Version control is now part of the problem. A backup taken before the corruption saves you.

Scenario 4: GitHub outage or account issues What if GitHub experiences an extended outage during your critical deadline? What if they terminate your account in error (it happens)? What if GitHub's data center has a catastrophic failure? Your backups keep you working.

Scenario 5: Malicious repository deletion GitHub has recovery options for recently deleted repositories—but they're not guaranteed and have time limits. A proper backup has no such limitations.

The Dangerous Statement

When developers treat GitHub as backup, they create a single point of failure. You're trusting:

- GitHub to never lose data
- Your account to never be compromised
- Yourself to never make a destructive mistake
- The service to exist forever with your data intact

That's not redundancy—that's risk.

What You Actually Need

Use GitHub for what it's designed for: tracking changes, enabling collaboration, and managing code evolution. It's excellent at this.

Implement actual backups:

- Automated, regular backups of your repositories to separate storage
- Ideally following the 3-2-1 rule: 3 copies, 2 different media types, 1 offsite
- Periodic verification that your backups actually work
- Consider services like Backblaze, AWS S3 with versioning, or even automated scripts that clone your repos to external drives

Many developers use tools like `git-backup`, automated scripts, or services that sync Git repositories to multiple locations independent of GitHub. Some use their hosting provider's backup features (GitHub repos can be backed up to AWS, for example).

In Conclusion

If you're moving from daily ZIP files to GitHub, that's fantastic progress. GitHub will transform how you work. But please, don't stop there. Set up real backups that exist independently of GitHub. Your future self—the one who just experienced a disaster—will thank you.

GitHub tracks your journey. Backup ensures the map survives.

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —