NEWS · MAY 9, 2026

No Developer Should Be Without Version Control: Set up your own private GitHub with Gitea in 10 minutes

By Dr. Holger Flick

At a recent conference where I had the pleasure of speaking to a room full of talented Delphi developers, I once again — as I do at every event — asked the audience to raise their hand if they used version control in their daily work.

The result was, to put it mildly, **staggering**.

Well below 50% raised their hands. And when I narrowed it further — who knows what Git is? Who has used GitHub or any other Git hosting platform? — the numbers dropped even further. In 2026. At a professional developer conference.

I was not surprised. But I was genuinely motivated to do something about it.

Why So Many Developers Skip It — And Why That's Changing

As a **GitKraken Ambassador**, version control advocacy is close to my heart. And I genuinely believe the hesitation I see is not laziness — it is a perception problem. Many developers, especially those who have worked solo for years, simply have not experienced a catastrophic loss *yet*. Version control feels like insurance: easy to skip until the moment you desperately need it.

Others have a different concern: *privacy*. They hear "Git" and think "GitHub" and immediately worry about their proprietary code ending up in Microsoft's cloud, or worse, accidentally made public. That is a completely legitimate concern — and it is one that Gitea solves entirely.

The encouraging news is that attitudes are shifting. More Delphi projects are appearing on GitHub and GitLab. Younger developers entering the community bring Git habits with them. The community is moving in the right direction. This post is for everyone who has been meaning to make the jump — consider this your nudge.

The Excuses I Hear (And Why They Don't Hold Up)

Over the years I have heard every reason imaginable for not using version control:

- *"My projects are too small for that."* — No project is too small to benefit from a complete history of every decision you ever made in the code.
- *"I work alone, so I don't need it."* — You collaborate with your past self. Future you will thank present you.
- *"Setting up a Git server is too complicated."* — This one, I want to address directly.
- *"I don't want my code on GitHub — privacy concerns."* — Completely valid. Which is why you don't have to use GitHub at all.

That last point is important. A lot of developers seem to think version control means GitHub, and GitHub means your code is public or in the hands of Microsoft. That is simply not true. You can run your own Git server, on your own hardware, under your own control, in about **10 minutes**.

Enter Gitea: Your Own Private GitHub

Gitea is a lightweight, self-hosted Git service. It looks and works almost identically to GitHub — repositories, issues, pull requests, organizations, web-based code browsing, the works. But it runs entirely on your own infrastructure. No subscriptions. No privacy concerns. No vendor lock-in. No internet required.

And with Docker Compose, getting it running is trivially easy.

Setting Up Gitea with Docker Compose

Here is a complete, production-ready Docker Compose setup for Gitea. You can run this on a spare PC, a Raspberry Pi, a NAS, or any Linux machine on your network.

Prerequisites

- Docker and Docker Compose installed
- A machine with at least 1 GB RAM (512 MB will work, but 1 GB is comfortable)
- A few minutes of your time

The Docker Compose File

Create a directory for your Gitea installation, for example `~/gitea`, and inside it create a file named `docker-compose.yml` with the following content:

```
version: "3.8"

services:
  gitea:
    image: gitea/gitea:latest
    container_name: gitea
    environment:
      - USER_UID=1000
      - USER_GID=1000
      - GITEA__database__DB_TYPE=postgres
      - GITEA__database__HOST=db:5432
      - GITEA__database__NAME=gitea
      - GITEA__database__USER=gitea
      - GITEA__database__PASSWD=gitea_secret_password
    restart: always
    volumes:
      - gitea_data:/data
      - /etc/timezone:/etc/timezone:ro
      - /etc/localtime:/etc/localtime:ro
    ports:
      - "3000:3000"   # Web interface
      - "2222:22"     # SSH for Git operations
    depends_on:
      - db
    networks:
      - gitea_network
```

```
  db:
    image: postgres:16-alpine
    container_name: gitea_db
    restart: always
    environment:
      - POSTGRES_USER=gitea
      - POSTGRES_PASSWORD=gitea_secret_password
      - POSTGRES_DB=gitea
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - gitea_network

volumes:
  gitea_data:
  postgres_data:

networks:
  gitea_network:
    driver: bridge
```

Starting Gitea

Open a terminal in your `~/gitea` directory and run:

```
docker compose up -d
```

That's it. After a moment, open your browser and navigate to:

```
http://localhost:3000
```

You will be greeted by the Gitea installation wizard. Walk through it once — set your administrator username and password, confirm the database settings (they are pre-filled from your compose file), and you are done.

Migrating Existing Projects

Once Gitea is running, getting your existing Delphi projects into it is straightforward:

```
# Navigate to your existing project folder
cd /path/to/your/delphi/project

# Initialize a Git repository
git init

# Create a .gitignore for Delphi (see below)
# Add all files
git add .
git commit -m "Initial commit"

# Add your Gitea server as the remote
git remote add origin http://192.168.1.100:3000/yourusername/yourproject.git

# Push
git push -u origin main
```

A Sensible .gitignore for Delphi Projects

Don't commit compiled binaries, DCU files, or IDE-generated clutter. Create a `.gitignore` file in your project root:

```
# Delphi compiled output
*.dcu
*.dcp
*.dcpil
*.dpu
*.bpl
*.bpi
*.lib
*.exe
*.dll
*.map
*.drc

# Local IDE settings
*.local
*.identcache
*.tvsconfig
*.dsk

# Backup files
*.~*
__history/

# Output directories
Win32/
Win64/
Android/
iOS/
OSX/

# Package cache
*.cache
```

GitKraken + Gitea: The Perfect Combination

Having a Gitea server is only half the picture. You also need a great Git client — and this is where GitKraken comes in. As a GitKraken Ambassador, I recommend it without hesitation, and its integration with Gitea is seamless.

GitKraken connects directly to your self-hosted Gitea instance. You add it as a custom hosting provider, authenticate once, and from that point on you can browse your repositories, clone, push, pull, and manage branches with a visual interface that makes Git immediately approachable — even for developers who have never used version control before. The learning curve that puts so many people off Git essentially disappears.

Two features in particular stand out for anyone getting started:

The visual commit graph shows you the full history of your repository as a clear, interactive timeline. Branches, merges, tags — everything is visible at a glance. For developers used to working linearly, this single view is often the moment version control finally *clicks*.

AI-assisted commit messages are a game changer for building good habits. One of the most common mistakes beginners make is writing meaningless commit messages like "fix" or "changes". GitKraken's AI analyzes what you actually changed and suggests a meaningful, descriptive commit message for you. Over time, reading good commit messages trains you to write them naturally — and your repository history becomes genuinely useful rather than a cryptic log.

If you have been hesitant about Git because the command line feels intimidating, GitKraken is the answer. Pair it with your private Gitea server and you have a professional, privacy-respecting version control setup that rivals anything a large enterprise uses — running entirely on your own hardware.

One note though: GitKraken's free tier does not support self-hosted Git servers. You will need at least the Pro plan to connect to Gitea. However, the investment is well worth it for the productivity and ease of use it provides — especially if you are new to version control. Also, GitHub offers the concept of Pull Requests. Gitea calls them "Pull Requests" as well, however, GitKraken does not support Pull Requests for self-hosted Git servers. That is a minor inconvenience, but it does not detract from the overall value of using GitKraken with Gitea for your version control needs.

Your Code Is Your Intellectual Property — Manage It Like One

Let me be clear upfront: version control is **not** a backup solution, and I have written about that distinction before. If you have a solid backup strategy, great — keep it. Version control serves an entirely different and complementary purpose.

What version control gives you is **history with intent**. Not just a copy of your files from last Tuesday, but a complete, annotated record of *why* your code looks the way it does today. Every change recorded. Every decision traceable. Every experiment reversible — without fear.

Think about what that actually means in practice:

You can work fearlessly. Need to refactor a complex unit but worried about breaking something? With version control, you branch off, experiment freely, and either merge it back or discard it entirely. Nothing is ever truly lost. That freedom changes how you write code.

You have a professional audit trail. A client insists the software behaved differently three months ago. With version control, you can show exactly what changed, when, and why — down to the line. That is not just useful, it can be the difference between winning and losing a dispute. In regulated industries — medical, financial, industrial — this traceability is not optional.

Your knowledge survives personnel changes. When a team member leaves, their expertise does not walk out the door with them. Every commit message, every branch, every merge tells the story of decisions made and problems solved. A new developer can read that history and understand the codebase far faster than any handover document.

You can always answer "what changed?" That question comes up constantly — after a bug report, after a client call, after a deployment. Without version control, answering it means relying on memory or diffing ZIP archives. With version control, it is a two-second operation.

Your source code encodes years of accumulated expertise, hard-won domain knowledge, and professional reputation. Version control is how you manage that asset with the seriousness it deserves.

Need Help Getting Started?

If you are a Delphi developer — individual or team — and you want to get started with version control but are not sure where to begin, I am here to help. Whether it is setting up Gitea, migrating existing projects, establishing branching strategies, or integrating Git into your CI/CD workflow, this is exactly the kind of consulting and mentoring work I do at **FlixEngineering LLC**.

In 2026, no professional developer should be without version control. Let's fix that — one repository at a time.

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 —