

Git & GitHub for Developers

Course Duration ~35 Hours · 5 Weeks · Live Online · Scenario-Driven Labs on Realistic Repositories

Curriculum reviewed: July 2026 — the platform ships continuously; this syllabus tracks current capabilities

Mental Model

Recovery & Surgery

Review & Governance

CI/CD Automation

Security

AI-Era Workflow

In short: This is not a course about committing code. It's a course about everything that happens after — reviewing, integrating, automating, securing and recovering. The part that decides whether a team of ten ships weekly or breaks nightly. Three things carry it, all absent from free tutorials. First, **version control in the age of AI-authored code**: when coding agents open pull requests directly, the bottleneck moves from writing code to reviewing and integrating it — branch protection, required checks and repository governance become the primary control surface over what reaches production, and AI-generated code carries the same risks as human code, only faster. Second, **recovery and history surgery**: reflow, bisect, undoing bad merges and rebases, hard conflicts — nobody is senior until they can fix a repository at 11pm without panicking, and that is a teachable skill. Third, **GitHub as a delivery and security platform**: automation pipelines, environments and gates, governance rulesets, secret and code scanning, dependency management and supply-chain hygiene. Five focused weeks, deliberately shorter than its siblings — you're learning a discipline, not a framework — and it multiplies every other course in the catalogue: they all assume you can collaborate on a shared codebase; this is the one that makes that true. You leave with a disaster-recovery lab behind you, a governed repository capstone, and a real open source contribution — the one credential any hiring manager can verify directly.

Duration	5 weeks · ~35 hours · live + hands-on — deliberately shorter than its siblings	Format	Live online · scenario-driven labs on realistic repositories
Prerequisite	You write code in some language; basic command line comfort. No prior Git experience required — we go beyond the basics quickly.	Curriculum	7 modules, from the object model to security, scale and capstone
Portfolio	8 scenario labs plus a governed-repository capstone — and a real open source contribution	Ideal for	Fresh IT graduates, experienced developers who never went past three commands, and teams adopting AI coding tools

Course Highlights

- Every module leads with what you build, ship or recover from — not what you "learn about"
- Git taught as a mental model of the graph — every confusing situation becomes obvious
- Recovery as a core competency: reflow, bisect, history surgery, hard conflicts
- Reviewing AI-generated changes at volume — what to check first, what humans must still own
- Governance as a control surface: protection rules, required checks, code ownership, merge queues
- CI/CD automation with environments, approval gates and release automation
- Security as practice: secret-leak incident response, dependency and code scanning, supply-chain hygiene
- A real open source contribution, start to finish — a credential anyone can verify

Hands-on Projects & Portfolio

- **Disaster Recovery Lab** — a deliberately broken repository: lost commits, a bad rebase, a poisoned history — restored without losing work
- **Bisect Investigation** — find the exact commit that introduced a regression in a long history
- **Branching-Model Exercise** — the same feature through trunk-based and branch-based flows, comparing what each cost
- **Code Review Workshop** — a realistic pull request, including a machine-generated one, with written review feedback assessed
- **Governance Build-Out** — protection rules, required checks, code ownership and a merge queue on a multi-contributor repository
- **CI/CD Pipeline** — matrix builds, caching, environment gates and automated releases
- **Secret-Leak Incident Response** — detect, rotate, purge from history, and prevent recurrence
- **Open Source Contribution** — a real pull request to a real project, start to finish

- **Capstone** — take an unmanaged repository to fully governed, automated and secured

Tools & Stack

Git · GitHub · GitHub Actions · GitHub CLI · pull requests & code review · repository rulesets · CODEOWNERS · merge queues · Dependabot · secret scanning · code scanning · GitHub Projects · pre-commit hooks · semantic versioning · conventional commits · Docker · VS Code

Career Outcomes

An honest framing: this is rarely a job title on its own. It is the skill that makes every other engineering role work — and it is disproportionately visible, because your commit history, pull requests and review comments are the most public artefacts of how you work. As more code is generated rather than typed, review, integration and governance become the scarce skills — exactly what free Git tutorials don't teach. A well-maintained public contribution history is one of the few credentials a hiring manager can verify directly. Where it pays:

- Software Engineer at any level · Senior Engineer and Tech Lead · DevOps and Platform Engineer
- Release Engineer · Engineering Manager · Open source contributor building a visible public track record

Who This Course Is For

- Final-year students and fresh IT graduates who know commit and push — and freeze at anything else
- **Developers with years of experience who have never learned Git beyond three commands** — a very large group, welcomed without condescension
- Engineers moving into senior, lead or review-heavy roles
- Teams adopting AI coding tools who now face far more pull requests than before
- DevOps and platform engineers owning CI/CD and repository governance
- Anyone who wants to contribute to open source but doesn't know how to start

Module 1: How Git Actually Works

5 Hours

YOU BUILD

- A hand-drawn (then tool-verified) map of a real repository's graph — objects, refs, HEAD — that turns every confusing Git situation you've ever hit into something you can picture

TOPICS

- The object model (blobs, trees, commits); refs and branches as pointers; the staging area
- HEAD and detached HEAD; what a merge and a rebase really do to the graph
- Remotes and tracking — a mental model, not a command list

Assessment: Quiz + Lab

Module 2: Daily Practice, Done Properly

5 Hours

YOU SHIP

- A working history that reads as documentation — conventional commits, semantic tags, clean ignore rules — plus the removal of a file that should never have been committed

TOPICS

- Meaningful commits and history as documentation; conventional commit messages; interactive staging; stashing
- Tags and semantic versioning; configuration and aliases; ignore rules and cleaning up wrongly committed files
- Large file handling; working across multiple checkouts

Assessment: Quiz + Lab

Module 3: Branching & Collaboration Models

5 Hours

YOU SHIP

- The branching-model exercise: the same feature run through trunk-based and branch-based flows, with a written comparison of what each cost — and a model recommendation for a given team

TOPICS

- Trunk-based development versus long-lived branches versus release branching; feature flags as an alternative to branch isolation
- Merge versus rebase as a team decision rather than a preference; keeping a readable history
- Forks and contribution workflows; choosing a model for team size and release cadence

Assessment: Quiz + Lab**Module 4: Recovery & History Surgery**

5 Hours

YOU RECOVER

- The disaster lab: a deliberately broken repository — lost commits, a bad rebase, a poisoned history — restored without losing work, plus a bisect investigation pinning the commit that broke a build

TOPICS

- The reflog and recovering "lost" work; undoing commits, merges and rebases safely; interactive rebase for cleaning history; cherry-picking
- Resolving hard conflicts including binary and structural ones; bisecting to the commit that introduced a bug
- Rewriting history safely — and the rules for when you must not; repairing a repository someone else broke

Assessment: Quiz + Lab**Module 5: Code Review & Governance**

5 Hours

YOU SHIP

- The governance build-out — protection rules, required checks, code ownership, a merge queue — plus the review workshop on a realistic PR, including a machine-generated one, with your written feedback assessed

TOPICS

- Pull request practice that catches real problems; reviewing for design rather than style; feedback without friction; keeping changes small and reviewable
- Branch protection and repository rulesets; required checks and approvals; code ownership; merge queues
- Reviewing AI-generated changes at volume — what to check first, what humans must still own, and keeping review meaningful when PR volume rises

Assessment: Quiz + Lab**Module 6: Automation & Delivery**

5 Hours

YOU SHIP

- A CI/CD pipeline with matrix builds, caching, environment gates and automated releases — then debug a deliberately failing pipeline to done

TOPICS

- Continuous integration workflows; matrix builds and caching; reusable and composite workflows
- Secrets and environment management; deployment environments with approval gates; release automation and changelogs; packages and artifacts
- Self-hosted and larger runners; cost control; debugging a failing pipeline; automating routine repository work

Assessment: Quiz + Lab**Module 7: Security, Scale & Capstone**

5 Hours

YOU SHIP

- The secret-leak incident response — detect, rotate, purge, prevent — a real open source pull request, and your capstone: one repository taken from unmanaged to fully governed, automated and secured, defended

TOPICS

- Secret scanning and credential-leak response; dependency vulnerability management; code scanning; signed commits and artifact provenance
- Supply-chain risk in third-party actions and dependencies; permissions and least-privilege tokens; organisation-level governance; monorepo strategies
- Open source contribution practice; capstone build and defense

Assessment: Capstone Project + Review

www.progmatrix.com