Engineer

GitHub

Microsoft's hosting for git repositories

What it is

GitHub is a hosted service, part of Microsoft, that keeps git repositories on its own servers and builds a place to work around them: a web view of the code and its history, pull requests for proposing and reviewing changes, issues for tracking work, and GitHub Actions for running jobs automatically whenever something happens in a repository.

Underneath sits plain git. A repository cloned from GitHub is an ordinary git repository with its full history, and work moves between a laptop and GitHub with the usual push and fetch. What GitHub adds lives on its side of the connection: review discussions, issues, branch rules, automation settings and the list of who may do what are kept in GitHub's own systems, not in the repository you cloned.

A pull request is GitHub's name for a proposal to merge the changes on one branch into another. It shows the difference line by line, collects comments and approvals, and displays the result of every automated check that ran against the proposed code. A pull request opened as a draft is marked unfinished, and it cannot be merged until someone marks it ready for review.

GitHub Actions is the automation built into the service. A workflow is a YAML file in the repository's `.github/workflows` directory. It names the events that start it, such as a push, a new pull request or a schedule, and lists jobs made of steps, each of which runs a shell command or an action, a packaged piece of automation that can be shared between repositories. Jobs run on runners: virtual machines GitHub starts fresh for each run on Ubuntu Linux, Windows or macOS, or machines of your own registered as self-hosted runners.

Two more services watch what enters a repository. Dependabot reads a `dependabot.yml` file and opens pull requests that bring dependencies up to date, and it separately raises pull requests for dependencies with known vulnerabilities. Push protection inspects incoming pushes for secrets such as access tokens and blocks a push that contains one before it reaches the repository.

Use the concept in a decision

Put “What is GitHub?” to work in a real decision. We can help relate the concept to the system you are building or buying. Tested, released and reversible, so changes reach customers on a Tuesday afternoon rather than late on a Friday night.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Automated GIS SaaS deployment with GitHub Actions — deployment, data and reporting, all off the hands

Automated GIS SaaS deployment with GitHub Actions Get in touch →

Why it matters

For many teams GitHub is where the work happens, not only where it is stored. The code, the discussion of each change, the record of who approved it and the outcome of every check sit side by side, so the question of why a line exists can usually be answered from one page.

The automation matters as much as the hosting. With Actions next to the code, a pull request can carry its own evidence: the tests ran and passed, the build worked, the scanners reported what they saw. Protected branches turn that evidence into a rule. A branch can be set to accept changes only through a pull request, only after a set number of approving reviews, and only once named status checks have passed, so the main line moves through reviewed and checked changes and nothing else.

Because GitHub sits at the point where code enters a project, it is also where supply-chain controls live: which third-party actions a workflow may run and at which exact version, how dependencies are kept current, whether a secret is stopped at the push. These controls are mostly switched on and tuned per repository or per organisation. A large share of a GitHub project’s real security therefore lives in settings, which nobody sees in a code review.

That is the trade-off of a hosted service. You get a complete, maintained environment without running servers, and in return the rules that protect your code are configured in someone else's interface, with defaults chosen for the general case rather than for yours.

What it looks like in practice

Three pages from the company's work show GitHub doing different jobs.

A delivery pipeline built on GitHub Actions splits the work into separate workflows: one for the code-quality gate, one for the tests and one per environment for deploys, with CodeQL, dependency review and a software bill of materials step running alongside them. The path runs from a commit to a service running in Azure, and promotion moves a build through the environments along a defined route instead of by hand. The frontend ships on a distroless image with no shell and no package manager, the Go API on a slim Alpine base.

A continuous security programme uses the controls that surround the code on GitHub. Every third-party GitHub Action is pinned to a commit hash rather than a tag, so nobody can move a tag and change what a workflow runs. Dependabot watches 6 ecosystems, Gitleaks scans the history for credentials, and every build produces a software bill of materials with Anchore and Syft. The page names the real cost too: the volume of findings. A scanner that reports everything trains people to ignore it, so keeping the signal usable takes ongoing triage.

The deployment of a GIS software-as-a-service application moved its test, build and deploy cycle into GitHub Actions workflows, so a release stopped depending on someone remembering the steps. The recurring data processing and reports moved into scheduled Python, Bash and SQL jobs, and the configuration and secrets were standardised so every environment behaved the same way.

Pinning looks like this in a workflow. The long hexadecimal string is the full commit identifier of the release that was reviewed, and the comment keeps the human-readable version beside it:

```
steps:
    - uses: actions/checkout@3d3c42e5aac5ba805825da76410c181273ba90b1 # v7.0.1
```

Where it goes wrong

- **An action pinned to a tag changes without warning.** A tag such as `v4` can be moved or deleted by whoever controls the action's repository, so the code a workflow runs tomorrow may differ from the code reviewed today. GitHub's own guidance is to pin to a full commit SHA, which an attacker could only imitate by producing "a SHA-1 collision for a valid Git object payload".
- **A workflow token holds more power than its job needs.** Every run receives a `GITHUB_TOKEN`, and its permissions decide what a compromised step could change. GitHub recommends making read access to repository contents the default and granting write access only to the jobs that need it.
- **Untrusted text runs as a command.** A workflow that writes a pull request title straight into an inline script lets whoever typed that title choose part of the script. Setting the value as an intermediate environment variable keeps it data. The `pull_request_target` and `workflow_run` triggers start privileged workflows, and GitHub warns that checking out an untrusted pull request inside one exposes the repository to compromise.
- **Administrators walk straight past the branch rules.** By default, a branch protection rule exempts people with admin permissions on the repository from its restrictions. Turning on "Do not allow bypassing the above settings" makes the rules bind them too.
- **A secret slips through push protection.** The form that is on by default protects each user's own pushes to public repositories. The form that guards a repository against every contributor needs GitHub Secret Protection and stays off until someone enables it, and even then anyone with write access can bypass a block by giving a reason, which raises an alert and lands in the audit log. Once a credential is pushed, it lives in every clone, so rotate it instead of trusting a history rewrite.

Further reading

- GitHub Docs — the full documentation for repositories, pull requests, Actions and the security features.
- Understanding GitHub Actions — workflows, events, jobs, steps and runners, and how they fit together.
- Secure use reference — pinning actions, token permissions and script injection, set out by GitHub itself.
- About protected branches — required reviews, required status checks and who may bypass them.
- Dependabot version updates — how a `dependabot.yml` file drives the update pull requests.
- Push protection — the two kinds of push protection, what each blocks, and how a bypass is recorded.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is GitHub?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/github/