Engineer

Forge

a shared host for code repositories and collaboration

What it is

A forge, also called a software forge, a code forge or a git forge, is the server a team agrees to treat as the home of its repositories, together with the services it runs around them. Every full clone of a Git repository carries the whole history, so in principle no copy outranks another. In practice a team needs one place to push to, one place that decides who may read and write, and one place where the conversation about the code happens. That place is the forge.

Underneath, a forge keeps bare repositories, the history-only form with no files checked out, and speaks one or more of Git's transfer protocols. The Git book lists four: the local protocol for a directory on the same machine, HTTP, SSH, and Git's own protocol, a daemon on port 9418 with no authentication at all. Everything else is built on top, and how much varies widely.

Forgejo's documentation describes the full end of the range: issue tracking, pull requests, an integrated wiki, a package registry for many language ecosystems, Forgejo Actions for running jobs, repository mirrors, migration from other services, branch and tag protection, and webhooks. Soft Serve sits at the other end. It describes itself as a "self-hostable Git server for the command line": a single binary called `soft` that serves repositories over SSH, HTTP or the Git protocol, controls access with SSH public keys, and supports collaborators, private repositories and webhooks. Both are forges; they differ in how much of the team's work they try to hold.

Forges have also started to talk to each other. ForgeFed, a protocol that federates platforms for hosting and collaborating on software projects, built as an extension of ActivityPub, models repositories, ticket trackers, patch trackers, projects and teams as actors that send each other activities. Its aim is a common language across networks of forges and project management tools, so that a

person on one server can open a ticket against a repository on another, and a repository can move between servers while the old copy is archived as read-only and points to its new home.

Use the concept in a decision

Put “What is a forge?” 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

The forge is where policy becomes enforceable. A rule that lives on a developer’s machine applies only while that machine is set up correctly, but a hook on the server runs on every push. Git’s `pre-receive` hook reads each pushed ref as an old id, a new id and a name on standard input, and if it exits with a non-zero status, none of the refs are updated. Branch protection, required reviews and signed-commit checks on larger forges refine the same idea: the shared copy refuses what the team has agreed not to accept.

Choosing a forge is also choosing who holds the code. A hosted service takes on the running costs and the uptime, and in return the code, the access list and every discussion sit on someone else’s hardware under someone else’s terms. Self-hosting reverses the trade. Neither choice is free, and the difference matters most to a business that tells its clients where their data lives.

A forge also holds data that Git does not. Commits and tags travel with every clone, while issues, review threads, access lists and server-side hooks belong to the forge alone. That split decides what a backup has to cover and what a move between forges has to carry.

What it looks like in practice

This company runs its own forge, and three projects on this site show what that took.

The company's source code used to live on a third-party hosting service, and the project to deploy the company's own git forge starts from the observation that running a forge means running a forge: authentication, access control, storage, backups, and a public face for it. The forge is Soft Serve, a single binary supervised by the operating system, chosen for being SSH-first with no administrative web interface. It is private by default: anonymous access is refused, keyless access is refused, and every declared repository is marked private. Its SSH listener binds only to the loopback interface on port 23231 and is reached from outside through a jump host. A multiplexer that would put HTTPS and git over SSH on one public port is written and left switched off, because a listener nobody uses is surface. The result is stated plainly: the company hosts its own code on its own hardware.

This site's own repository lives there: its instructions name the self-hosted forge, Soft Serve over SSH, as the canonical remote, call the older hosted copy a legacy mirror, and say there is no hosted CI: the quality gate runs locally and in git hooks.

A server-side hook needs to be tested in the receiving environment of a real Git push, so the hook was rehearsed against a real bare repository and a real push before being trusted. The rehearsal creates a bare repository, installs the hook, clones, commits, pushes and asserts on both the accept and the reject path. It found two defects, neither visible in the file. The second is the instructive one: Git exports a variable naming the repository directory when it runs a hook, the test suite the hook started inherited it, and nineteen tests went red on correct code until the environment was cleared at the boundary. The page also names its limit: forced updates and branch deletions are pushes the rehearsal does not construct.

All of this shares one single-core host with 512 MB of memory and everything else the company serves in public, from the seven TLS domains to the onion service, a Gemini and a Gopher server and intrusion banning. Backup automation is implemented separately; that is distinct from active backups. A forge with a web panel would have been one more resident application on that budget, which is why web-panel platforms were ruled out at the design stage; on that host, available memory bottomed out at about 310 MB of 464 over a recorded day of samples.

Where it goes wrong

Anonymous visitors get read access by default on some forges. Soft Serve's documentation says that out of the box it gives read-only permission to anonymous connections, governed by two settings, `anon-access` and `allow-keyless`. A private forge sets the first to `no-access` and the second to false, and the result is worth testing from a machine with no key loaded. Git's own daemon is blunter still: the Git book notes that a repository it serves is either available to everyone to clone or not served at all.

A hook that calls Git in another directory acts on the wrong repository. During a push, Git runs the receiving hooks inside the bare repository and exports `GIT_DIR` and related variables so that Git commands in the hook find it. Anything the hook starts inherits them, including a test run in a checkout somewhere else. The githooks manual says to clear these variables before working in a foreign repository. A minimal `pre-receive` hook that tests each pushed commit in a throwaway directory does it in a subshell:

```
while read -r _ newrev _; do
  git cat-file -e "$newrev^{commit}" 2>/dev/null || continue
  tree=$(mktemp -d)
  git archive "$newrev" | tar -x -C "$tree"
  (
    unset $(git rev-parse --local-env-vars)
    cd "$tree" && ./run-tests
  ) || exit 1
done
```

A restored forge comes back with its code and without its rules. Any clone can repopulate a lost server's repositories, but the Git book warns that server-side hooks and similar settings stay behind, and issues, reviews, keys and access lists were never in the clones. Backing up a forge means backing up its data directory and its configuration, then restoring both somewhere and checking that a push the hook should reject is rejected.

Every listener a forge opens is something to defend. Soft Serve's sample configuration has SSH on port 23231, HTTP on 23232, the Git daemon on 9418 and a statistics server on 23233. A team that only pushes over SSH can switch the rest off or bind them to loopback, so the open ports stay the size of what is used.

Moving to another forge carries the code and strands the conversation. A mirror push copies every ref under `refs/`, branches and tags included, and that is all it copies. Issues, review threads, labels and access lists need the new forge's migration tool, or a protocol such as ForgeFed's, which copies commits, tags, tickets, merge requests and comments and leaves the old repository archived and pointing at the new one.

Further reading

- ForgeFed specification — the federation protocol for forges, with its actors, its activities and the move flow that archives an old repository.
- Soft Serve — the project's own documentation for a single-binary, SSH-first Git server, including the `anon-access` and `allow-keyless` settings.
- Forgejo documentation — administrator and user guides for a full-featured self-hosted forge, from installation and backups to Actions and mirrors.
- githooks — every hook Git runs, including `pre-receive` and the environment a push hook inherits.
- The Protocols — the four ways Git moves data between machines and what each costs in setup and security.
- ActivityPub — the W3C recommendation that ForgeFed extends.



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 a forge?” 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/forge/