AUTOMATIZE · AUTHORING & PUBLISHING

Git writer.

Write once. Version everything. Publish everywhere.

Every book, manual or documentation set is a real Git repository — branches, commits, diffs, blame, merges and conflict resolution — driven entirely from a browser, by people who will never install a Git client. Then published to PDF, EPUB, DOCX, HTML and a searchable static site from the same source.

- Real Git, no client needed
- 15 model providers
- Grammar in every language
- 5 export formats
- Self-hosted

13	7	264	2708
BOOK TYPES	GIT HOSTING TARGETS	REST ENDPOINTS	AUTOMATED TESTS

ENGINEERING DOCS · COMPLIANCE · TRAINING · AGENCIES · PUBLISHERS

PRODUCT BROCHURE 2026.1

THE PROBLEM

Docs-as-code works beautifully — for the people who can already use Git.

Engineering wants version control, branches and review. The technical writer, the compliance officer and the trainer want to open a page and type. So the documentation splits: the parts under version control that only engineers touch, and the parts in a wiki that nobody can diff, review or roll back. Both halves go stale, in different ways.

Two systems
The repository and the wiki, disagreeing about what is current.

No review
Wiki pages change with no branch, no approval and no way to see what moved.

Publishing by hand
The PDF for the customer is exported, reformatted and emailed by one person.

Content leaves
Hosted authoring puts unreleased specifications and policies outside your control.

THE SOLUTION

Give everyone the browser. Give the content real Git underneath.

Genuine version control
Not a revision table pretending to be Git. A real repository per book, with branches, annotated tags, merges, three-way conflict resolution, per-file history and blame — driven entirely from the interface.

Review before publish
Branch, comment on a line, suggest a replacement, request changes, approve, merge. The workflow engineering already trusts, made usable by people who are not engineers.

On your own hardware
Including the intelligence: eight of the fifteen supported model runtimes are local, and grammar checking runs inside the application. Nothing has to leave the network.

HOW IT WORKS

1. **Create or import**
   Start a book, or import an existing set from Confluence, Notion, GitBook, Markdown, Word or rich text — the archive format is detected for you.
2. **Write**
   Rich, source or split view with live preview, a drag-and-drop page tree, book-wide find and replace with regular expressions, focus mode, and grammar checking as you go.
3. **Review**
   Work on a branch, invite comments anchored to the text rather than the line number, take suggested edits, approve, and merge — resolving conflicts in the browser if they arise.
4. **Publish**
   Tag a version and generate PDF, EPUB, Word, HTML or a searchable static site, on your own domain, with a changelog feed for readers who follow it.

WRITE & VERSION

The editor, the Git engine and the review workflow.

1 The Git engine

- A genuine, non-bare repository per book, scaffolded with a table of contents, sections and an asset folder
- Commit, **revert a commit**, paged history, per-file history and **blame**
- Branches typed as main, draft, review or archive, with protection and an undeletable default
- Annotated tags; merges with a **dry-run check before you commit to one**
- Diff between any two references, diff of a commit, formatted hunks with rename detection
- **In-browser conflict resolution** — theirs, ours and an editable merged pane per file, with the option to abort
- Per-repository locking, path containment on every write, and branch names validated so no reference expression can be smuggled in
- **Mirroring to seven hosting targets** — GitHub, GitLab, Gitea, Forgejo, Gogs, Codeberg or purely local — with tokens re-resolved per operation and never written into the repository configuration

2 The editor

- Rich, source and split modes with a draggable divider and debounced live preview
- Drag-and-drop page tree with create, rename, move, sub-pages and automatic table-of-contents regeneration
- **Book-wide find and replace** with case, whole-word and regular-expression options — replace-all lands as a commit
- Focus mode, auto-save with auto-commit of drafts, and a full keyboard map
- Image upload by button, paste or drag; assets browsable per book
- **Grammar and spell checking embedded in the application**, in every language the checker ships, with automatic language detection and one-click fixes — no external service, no per-seat grammar licence
- **Readability scoring** — reading ease, grade level and reading time, per file and per book
- Daily word goals, a streak counter and a thirty-day activity sparkline

3 Collaboration & review

- **Live co-editing** with per-user cursors and a presence bar showing who is in the book
- Comments anchored to **the quoted text as well as the line range**, so highlights survive edits above them; threaded, resolvable and assignable
- **Suggested edits** — propose a replacement; accepting writes and commits it
- Review requests from branch to branch with approvals, change requests, threaded review comments and merge
- Mentions with autocomplete, and in-app notifications for mentions, comments, assignments and completed jobs

4 Import & migrate

- Confluence, Notion and GitBook archives, plus Markdown, Word and rich text — **format detected automatically**
- GitBook imports follow the existing table of contents in order, then append anything unreferenced
- Bounded and safe: page and file ceilings, archive-entry limits, and traversal defences on every path
- Existing pages are never overwritten — slugs de-duplicate instead
- Unsupported files are skipped with a warning rather than failing the whole run

> **Why the anchor text matters.** Comments pinned to a line number drift the moment somebody inserts a paragraph above them, and a review full of misplaced highlights stops being read. Gitwriter stores the quoted text alongside the range, so a comment stays attached to the sentence it was actually about.

ASSIST, PUBLISH & MEASURE

The model layer, the output and the platform.

5 The model layer

- **Fifteen providers, eight of them local** — so drafting, translation, search and review can run with no external call at all
- Configuration resolves organisation, then tenant, then platform; keys masked everywhere; administrator-supplied addresses guarded against internal-network probing
- Selection actions: check, rewrite, improve with an instruction, shorten, lengthen — streamed token by token
- Book-level work: **continuity checking** across chapters with typed findings, style linting against your own guide, summary and glossary generation
- Eight insight tools including pacing, fact-check, link suggestion, alt text, tone and a character bible
- **Whole-book translation onto its own branch**, page by page, with live progress
- **Semantic search with no vector database to install** — embeddings stored inline with in-process similarity, degrading to keyword scoring when unavailable

6 The agentic writer

- Plans, drafts, reviews, refines and finalises — **on its own branch**, committing each page as it goes, so nothing it does is unreviewable
- Six tools it may call, each path-validated: list, read, edit, create, update the contents, finish
- **Durable, replayable progress** — events carry a sequence number, so a browser that drops resumes exactly where it left off rather than starting again
- Cooperative cancellation, bounded page and iteration counts, and a reaper for abandoned runs

7 Publishing

- **Five formats from one source**: PDF, EPUB, Word, HTML and a static site
- The PDF engine is auto-detected across the usual installations rather than configured by hand
- **Export presets inherited across three levels**: page size, margins, base size, fonts, contents depth and section numbering
- A cover designer with image, colour, subtitle and author overrides, plus front and back matter
- **The static site**: sidebar contents, on-this-page outline, previous and next navigation, a working light and dark toggle, copy buttons on code blocks and **client-side full-text search**
- Versions bound to annotated tags with release notes and per-version downloadable artefacts
- A public reader for published books, an RSS changelog, and a “was this helpful” widget
- **Custom domains** resolved from the incoming host, optionally pinned to a branch
- Per-book backup and restore, hardened against archive-traversal and hook execution

8 Platform & insight

- Platform, tenant and organisation tiers, each with its own administration surface
- **White-label branding with field-level inheritance** — name, logo, favicon, colours and typeface, applied for anonymous readers too
- Reader analytics: views, sessions, searches, helpfulness, per-page figures, top search terms and **drop-off pages**
- Content health: broken internal links and orphan pages, recomputed on every build
- Contribution statistics derived from blame; writing statistics per book
- Log viewer, migration runner, memory diagnostics and a guided first-run setup
- A **63-page help centre inside the product**, searchable — no external docs site required

Worth knowing before a pilot. Mathematics and diagrams render in the editor preview and the in-app reader, but not yet in the exported files or the generated site. Outbound e-mail, password self-reset and multi-factor authentication are not in this release, and provider credentials are not yet encrypted at rest. Deploy on a trusted network and ask us for the roadmap.

WHO BUYS IT

Long-form content that has to be reviewed, versioned and published.

Software & product teams
Product documentation, API references, specifications, runbooks and internal wikis — the docs-as-code workflow, made usable by the people who are not engineers.

Regulated & air-gapped
The sharpest differentiator: the entire stack, including the model and the grammar checker, runs on-premise with no outbound call.

Compliance & quality
Policies, procedures and specifications where every change needs a branch, an approver and a permanent history you can produce on request.

Documentation agencies
Tenant and organisation separation with per-scope branding and custom domains — many client brands from one instance.

Managed service providers
Client runbooks and handover documentation, versioned and published under the client's own domain.

Training & education
Courses and workbooks published as PDF and EPUB, with reader analytics showing which page people abandon.

Manufacturing & engineering
Manuals and service documentation issued as tagged versions, so the field always knows which revision it has.

Publishers & authors
Continuity checking, character bibles, pacing analysis, writing goals and a cover designer — a novel and an API reference in one product.

Leaving a hosted wiki
Importers for Confluence, Notion and GitBook with automatic detection, so the migration is an afternoon rather than a project.

WHO USES IT

Platform superadmin · **Tenant administrator** · **Organisation administrator** · **Editor** · **Author** · **Reviewer** · **Book owner** · **Anonymous reader**

Two permission dimensions: a system role, and a per-book role from reader through owner. A manager can only grant a role at or below their own — and a book in another tenant answers *not found* rather than *forbidden*, so identifiers cannot be probed.

AGAINST THE ALTERNATIVES

OPTION	WHY IT FAILS HERE
Hosted docs platform	Content leaves; priced per seat; their brand on your reader
A wiki	No branches, no review, no diff, no versioned release
Raw docs-as-code	Excludes everyone who does not use Git — which is most of the authors
Word documents on a share	No single source, no publishing pipeline, no history worth the name

TECHNICAL SPECIFICATION

One JAR, one database, and a Git engine inside it.

Runtime	Java 17 on Spring Boot 4; native-image build supported
Database	PostgreSQL, 30 tables, schema installed by the setup wizard
Git	An embedded Java Git implementation — no Git binary, no external service
Grammar	A full grammar and spell checker running in-process, every shipped language
Export	A document converter plus a TeX engine, auto-detected across common installations
Front end	Server-rendered with vanilla ES6 — no build step, no framework, everything vendored
API	264 endpoints across 39 controllers, 240 with published descriptions
Realtime	Sockets for co-editing; server-sent events for presence, job progress and token streaming
Scale	~58 600 lines of application code, 30 entities, 135 services
Quality	**2 708 tests** — a suite roughly the size of the production code

SECURITY & GOVERNANCE

- Two-dimensional access control — a system role and a per-book role — evaluated in one place
- **Privilege-escalation guard**: nobody can grant a role above their own
- **Restricted books** that cut off implicit organisation-wide access and vanish from listings
- Tenant isolation with cross-tenant identifiers answering **not found rather than forbidden**
- bcrypt at strength twelve; brute-force protection with exponential lockout on login, registration and refresh
- Refresh-token rotation with a server-side revocation store
- **Single sign-on** through Google and Microsoft, configurable at all three levels
- Content security policy, frame denial, strict transport security and referrer policy; specification browser denied by default in production
- Path-traversal defence on every content, tool and branch path; outbound-address guard on administrator-supplied endpoints
- Restore hardened against archive traversal and repository hooks

Not in this release: outbound e-mail and self-service password reset, multi-factor authentication, directory or SAML sign-on, encryption at rest for stored provider credentials, and comprehensive audit logging. The deployment is single-node.

COMMERCIAL

Licensed per installation. Not per author.

Documentation platforms are priced per seat, which is why the reviewer, the compliance officer and the occasional contributor never get an account — and why the content fragments. One licence, unlimited authors, removes the reason the wiki existed.

REVENUE	PER YEAR
Up to €500k	€900
€1M	€1,500
€5M	€2,500
€25M	€5,500

REVENUE	PER YEAR
€100M	€13,000
€500M	€37,000
€1B	€52,000

Annual licence on your own turnover, ex VAT. The schedule is marginal like income tax — each slice of turnover is charged at its own rate, so there are no cliffs — with a €900 minimum and any yearly increase capped at 25%. Every tranche carries unlimited users, projects and organisations inside one tenant, self-hosting and support that rises with the tranche. Hosting and hardware are yours, AI runs on your own provider keys and is billed by them, and consultancy, setup, migration and training are quoted separately. Illustrative: a seat-priced platform at €12 per author a month bills two hundred writers €28,800 a year; at €5M turnover this is €2,500.

NEXT STEP

Export your wiki.

Give us the Confluence, Notion or GitBook archive. We will import it, publish it, and hand you the site and the PDF.

Sales	tim@automatize.eu
Partners	luc@automatize.eu
Web	automatize.eu

Git, and the hosting services Gitwriter can mirror to, are independent of this product. Specifications may change.