A FIND THE GAPS AUDIT

The 2026 *Documentation Debt* Report.

An audit of nine widely-adopted open-source projects — and what 6,397 drift findings say about the state of software documentation.

9	2,996	1,261
PROJECTS AUDITED	FEATURES ANALYZED	LARGE (BREAKING) FINDINGS

LARGE	MEDIUM	SMALL
Breaks implementation	Causes confusion or rework	Creates friction

PROJECTS

Airbyte · Elasticsearch · Grafana · Mattermost · MongoDB · n8n · PostHog · Sentry · Terragrunt

Documentation debt is a problem everyone *acknowledges* and nobody measures.

So we measured it. We ran **Find The Gaps** — Doc Holiday's open analysis tool — across 9 widely-adopted open-source projects, looking for the specific places where what the documentation says and what the code actually does have drifted apart.

Airbyte, Elasticsearch, Grafana, Mattermost, MongoDB, n8n, PostHog, Sentry, and Terragrunt. These aren't obscure repos. They're the kind of projects that show up in production stacks everywhere, maintained by teams that genuinely care about developer experience. If documentation drift is happening here, it's happening everywhere.

We analyzed **2,996** features. We found **6,397** drift findings — specific places where the documentation and the code no longer agree.

> *A drift finding isn't a typo. It's a specific place where a developer following the documentation will encounter a reality that doesn't match what they were told.*

Find The Gaps classifies each finding by severity: **Large** (breaks implementation), **Medium** (causes confusion or rework), and **Small** (creates friction). Across our nine projects, **1,261** of the **6,397** findings were Large. **More than one in five drift findings will stop a developer cold.**

Nine projects. One *green* row.

Each row is a project, each column a class of finding. The colors aren't decoration: pink for Large, amber for Medium, muted for Small. The single green row is n8n.

PROJECT	FEATURES	UNDOCUMENTED	TOTAL DRIFT	LARGE	MEDIUM	SMALL
Grafana	234	1	1,843	377	993	473
PostHog	267	38	1,229	280	685	264
Elasticsearch	1,280	9	950	138	535	277
MongoDB	363	7	751	107	426	218
Airbyte	154	0	647	149	316	182
Sentry	240	9	362	72	220	70
Terragrunt	65	0	350	75	200	75
Mattermost	145	1	247	63	142	42
n8n OUTLIER	248	14	18	3	11	4

Read across the row: Grafana documented essentially all of its 234 features — and still generated nearly 8 drift findings per feature. n8n documented 248 features and generated 18 total.

SOURCE: FIND THE GAPS AUDIT MAY 2026

Aggregate numbers are easy to nod at. These four are *harder to ignore*.

FINDING 01 — LARGE

Grafana: *1,843 drift findings across 234 features.*

Nearly 8 findings per feature — in a monitoring and observability platform where developers are often debugging under pressure and leaning hard on the docs to get it right. When the docs are wrong, the cost isn't just time. It's confidence in the system being monitored.

FINDING 02 — MEDIUM

PostHog: *38 completely undocumented user-facing features.*

PostHog is a product analytics platform — a tool designed to help teams understand what their users are doing. The irony of it having 38 features that users can't find documentation for is not lost on us.

FINDING 03 — PATTERN

Airbyte & Terragrunt: *zero undocumented features, hundreds of drift findings.*

Airbyte has docs for every one of its 154 features — and still generated 647 drift findings. Terragrunt: zero undocumented features, 350 drift findings. Documented doesn't mean accurate. You can have documentation for every feature and still have documentation that's wrong.

FINDING 04 — OUTLIER

n8n: *18 total drift findings across 248 features.*

By any measure, exceptional. A workflow automation platform spanning AI agent building, real-time collaborative editing, vector store integrations, and OAuth2 flows — with only 18 drift findings total. But it also creates its own sort of problem, which we'll get to next.

When good documentation becomes a *trap*.

n8n's numbers are genuinely impressive. **Eighteen** drift findings across 248 features is a track record most projects would envy — the work of a team that has made documentation a real priority, not a release afterthought.

The problem is what that track record does to developer trust. When you spend time in a codebase with documentation this reliable, you stop second-guessing it. You assume the docs are right.

And that assumption is exactly what makes n8n's remaining gaps more dangerous than the hundreds of Large findings in the eight other projects we audited combined.

0.07	2.32	~32×
Drift findings per feature	Drift findings per feature	Cleaner than its peers
N8N	THE OTHER EIGHT, AVERAGED	BY THE SAME MEASURE

> *None of these are catastrophic in isolation. Together, in a project where developers have been trained to trust the documentation implicitly, they're landmines.*

THREE CASES, OVERLEAF →

Three small drifts. One *broken assumption*.

Each of the cases below is a single place where n8n's documentation doesn't match what the code does. In a project anyone else would envy, three is plenty.

CASE 01

MEDIUM

The response code that *breaks automated tests.*

The credential-transfer endpoint is documented as returning `200 OK`. The API actually returns `204 No Content`. Any developer writing automated tests against the documented behavior will see successful operations flagged as failures, and spend hours debugging code that isn't broken.

CASE 02

LARGE

The security feature that *doesn't exist.*

The docs describe the ability to configure redirect domains and Content Security Policies in Security Settings to protect against SSRF attacks. The underlying code doesn't expose those controls. A developer hardening their n8n instance follows the documentation, finds no such settings, and either assumes they misconfigured something — or assumes the protection lives elsewhere and moves on.

CASE 03

MEDIUM

The feature with *no instructions.*

Dynamic credential resolution lets n8n fetch credentials from external systems at runtime. The documentation is essentially a landing page with no implementation guidance — no required fields, no length constraints, no examples. Developers discover this after they've already committed to using it.

The problem isn't effort or intention — it's *math*.

The teams maintaining these projects aren't negligent; several of them are among the best engineering organizations in open source.

Modern software moves faster than the people documenting it. Every PR that renames a field, changes a response code, adds a required parameter, or removes a configuration option creates documentation debt that has to be manually identified, manually assigned, and manually fixed.

> *For a project like Elasticsearch — 1,280 features, 950 drift findings — that's not a documentation problem. That's a documentation surface-area problem. Keeping the docs in sync through human effort alone is, practically speaking, impossible.*

The result is what you see in this report: **6,397** places where a developer following the documentation will encounter a reality that doesn't match what they were told.

A NOTE ON METHODOLOGY

Find The Gaps uses LLMs to compare codebases against their documentation and surface areas of drift. LLMs make mistakes — we say so plainly in the tool itself, and we're saying so again here. Individual findings should be validated before acting on them. What the aggregate numbers represent is real: the scale of the gap between what documentation says and what code does, across projects that have every reason to get it right.

Documentation that stays true to the code. *Automatically.*

Doc Holiday connects to your release pipeline and rewrites the documentation when code deploys. When a response code changes, the docs update. When a field is added or removed, the docs reflect it. When a feature is deprecated, the documentation says so — before a developer finds out the hard way.

WITHOUT

A PR ships. A response code changes. Three sprints later, a developer hits the old docs and spends a day debugging code that isn't broken. **Multiply that by 6,397.**

WITH DOC HOLIDAY

A PR ships. Doc Holiday reads the diff, drafts the documentation update, and queues it for review. The docs match what the code does. **Every release.**

RUN FIND THE GAPS ON YOUR PROJECT

doc.holiday/find-the-gaps →

It's *free*. We'd love to know what you find.