Errmap — Product Spec & Roadmap

2026-09-16 · @Someone

Errmap turns production errors into a self-maintaining runbook: every new error gets an AI-written doc in Notion or Confluence, and every repeat links back to it in Slack. "Errmap" is a working name; check the name and domain before launch.

1. Overview

The problem

Teams fix the same error several times because the fix lives in someone's head, a dead Slack thread, or a closed PR. Error trackers tell you *that* something broke. They don't build up knowledge of *why* it broke and *how it was fixed last time*.

What Errmap does

1. **Catches** errors from Sentry, OpenTelemetry, logs, other tools' webhooks, or its own SDKs.
2. **Groups** repeats of the same error using a fingerprint.
3. **Writes** a runbook for each new error with AI: summary, likely cause, fixes, owners, severity.
4. **Stores** it in the team's own Notion or Confluence (or the built-in knowledge base on Cloud).
5. **Alerts** a Slack or Teams channel with a link to that runbook, and says how often it has been seen.
6. **Learns** over time: links runbooks to the PR that fixed them, finds similar errors, and maps how errors cluster across services.

Positioning

"Turns error alerts into a self-maintaining runbook." Errmap is a companion to Sentry, not a competitor. It does not try to beat Sentry at capture. It owns the **knowledge layer**: runbooks, patterns, fix history and ownership.

Two editions, one codebase

	Errmap OSS (free, self-hosted)	Errmap Cloud (paid, hosted by you)
Error capture	Bring your own: Sentry, OTel, logs, webhooks	Built in: Errmap SDKs, no Sentry needed
Runbooks and error map	Yes	Yes, plus hosted dashboard
Doc stores	Notion, Confluence	Same, plus built-in knowledge base
AI	Bring your own key, or a local model (Ollama)	Included, usage-based
Setup	Docker image	Sign up, add SDK
Paid features	None	SSO, audit logs, IDE/MCP, deploy correlation

Who it is for

- **Primary:** engineering teams of 5 to 100 people that already use Sentry and Notion or Confluence.
- **Cloud:** smaller teams with no error tool who want capture and runbooks in one place.
- **Enterprise (later):** teams needing self-hosting, UK/EU data residency, and SSO.

2. How it works

Every error, from any source, becomes one normalised event that flows through the same pipeline.

Data flow

```
flowchart TD
  A[Sources<br/>Sentry, OTel, logs, SDK] --> B[Normalise + scrub PII]
  B --> C[Fingerprint]
  C --> D{Runbook exists?}
  D -- No --> E[AI drafts runbook]
  E --> F[Write to Notion / Confluence]
  D -- Yes --> G[Append occurrence<br/>update counts]
  F --> H[Alert Slack / Teams<br/>with runbook link]
  G --> H
```

New errors get a fresh runbook. Repeats update the existing one and post "Seen 14×" with the same link.

The building blocks

Part	Job	Examples
Source	Receives errors and turns them into one standard shape	Sentry webhook, OTel, log drains, generic webhook, Errmap SDKs
Scrubber	Removes emails, tokens, card numbers before anything leaves	Regex rules plus custom config
Fingerprint	Decides whether two errors are "the same"	Hash of service + cleaned message + top stack frames
Analyzer	Writes the runbook with AI	Anthropic, OpenAI, Ollama (local)
Doc store	Saves and finds runbooks	Notion, Confluence, built-in KB
Channel	Sends alerts	Slack, Microsoft Teams

Each part is a small plugin behind an interface. That makes it easy for contributors to add new ones, which is good for adoption.

How the Sentry webhook works

It is **push, not pull**. When Sentry sees a new issue, it sends an HTTP POST to your server's URL. Your server just waits for those requests.

```
sequenceDiagram
  participant App
  participant Sentry
  participant Errmap
  participant Notion
  participant Slack
  App->>Sentry: error captured by Sentry SDK
  Sentry->>Errmap: POST issue.created (webhook)
  Errmap-->>Sentry: 200 OK (immediately)
  Errmap->>Sentry: GET latest event (full stack)
  Errmap->>Notion: create runbook
  Errmap->>Slack: alert with runbook link
```

- **issue.created** fires once per new kind of error. This is the main trigger.
- **issue.resolved / unresolved** tells you a fix landed or a regression came back.
- **error.created** fires on every occurrence. It is noisy and may need a higher Sentry plan.
- **Alert rules** (e.g. "50 times in 1 hour") can call the integration for "Seen 50×" updates.

The webhook doesn't carry the full stack trace, so Errmap makes one **pull** call to Sentry's API for the latest event. Pulling is also used once at install to **backfill** existing issues.

Setup in Sentry: Settings → Developer Settings → Custom Integrations → Internal Integration → set the webhook URL, tick `issue`, grant read on Issues & Events, copy the client secret. Check Sentry's current docs, as field names and plan limits change.

Capturing errors without Sentry

Route	Code change?	Context quality	Browser errors?
Errmap SDK	Yes, 3 lines	High (breadcrumbs, user, release)	Yes
OpenTelemetry	Often none (auto-instrumentation)	Medium to high	Partly
Log drains (Vercel, Render, CloudWatch, Loki)	None	Low (message + stack only)	No
Other tools' webhooks (Datadog, Rollbar, Grafana)	None	Medium	Depends on tool

OpenTelemetry is the long-term bet because it is vendor-neutral. Browser errors always need a script in the page.

Cloud architecture

```
flowchart TD
  A[SDKs / OTel / webhooks] --> B[Ingest API in Go<br/>auth, quotas, rate limits]
  B --> C[Queue<br/>Redis Streams or NATS]
  C --> D[Workers<br/>symbolicate, scrub, group]
  D --> E[(ClickHouse<br/>events)]
  D --> F[(Postgres + pgvector<br/>orgs, runbooks)]
  D --> G[AI worker<br/>cached, cost-capped]
  G --> H[Notion / Confluence / Slack]
```

Host in the UK or EU from day one; data residency is a selling point for UK and European buyers.

3. Roadmap

Phases 1 and 2 (about 5 months) are enough to produce Global Talent evidence; Phase 3 onward only happens if the launch gets real traction.

Phase	Months	Theme	Exit criterion
0	0–1	Foundations	Repo, licence, name, IP sign-off done
1	1–3	OSS MVP and launch	Public launch; first 10 external installs
2	3–5	Beyond Sentry	OTel + Confluence live; first external PR merged
3	5–9	Own capture, Cloud beta	5–10 design partners using Cloud
4	9–12	Intelligence (the "error map")	Similar-error hit rate above 30%
5	12–15	Developer workflow	MCP server and VS Code extension published
6	15–24	Teams and enterprise	First paying Business/Enterprise customer

Phase 0: Foundations (month 0–1)

- [ ] Turborepo monorepo with `core`, `server`, `cli` and adapter packages
- [ ] Core types, fingerprinting, PII scrubber, unit tests
- [ ] Licence: Apache 2.0 core plus `/ee` folder under a commercial licence
- [ ] CLA or DCO bot on the repo so relicensing stays possible
- [ ] Name, domain, GitHub org, npm scope
- [ ] Written confirmation from ENODA that the project falls outside your employment IP clause
- [ ] Evidence tracker spreadsheet (stars, downloads, installs, talks, dated monthly)

Phase 1: OSS MVP and launch (months 1–3)

- [ ] Sources: Sentry webhook, generic JSON webhook
- [ ] Store: Notion. Channel: Slack (threaded replies per error)
- [ ] AI runbook draft: summary, cause, fixes, affected code, severity
- [ ] Dedup: "Seen N×" updates the same doc and Slack thread
- [ ] Docker image on GitHub Container Registry, built by a GitHub Action on each tag
- [ ] `npx errmap init` CLI that writes the config and tests each connection
- [ ] Docs site, README quickstart, demo GIF
- [ ] Launch: Show HN, dev.to post, r/devops, Sentry community forum

Phase 2: Beyond Sentry (months 3–5)

- [ ] OpenTelemetry source (OTLP receiver)
- [ ] Log drains: Vercel, Render, Railway, CloudWatch, Loki
- [ ] Confluence store, Microsoft Teams channel
- [ ] **Resolution tracking:** GitHub/GitLab app links each runbook to the PR or commit that fixed it
- [ ] Backfill of existing unresolved Sentry issues on install
- [ ] "Good first issue" labels and an adapter template to attract contributors

Phase 3: Own capture and Cloud beta (months 5–9)

- [ ] SDKs: Node and browser first, then React, Next.js, Go, Python
- [ ] Breadcrumbs: clicks, network calls, console logs before the error

- [ ] Source-map upload via CLI for readable frontend stack traces
- [ ] Hosted ingest API in Go, queue, workers, ClickHouse, Postgres
- [ ] Dashboard: projects, issues, runbooks, team members, API keys
- [ ] Billing (Stripe), free tier, per-org AI spend caps
- [ ] Recruit 5–10 design-partner companies; interview 10 teams before building

Phase 4: Intelligence (months 9–12)

- [ ] **Similar errors:** embedding search suggests "looks like runbook #42, fixed by PR #212"
- [ ] **Deploy correlation:** "started 4 minutes after release v2.3.1"
- [ ] **Ownership routing:** CODEOWNERS and git blame tag the right person
- [ ] **Error map view:** graph of services, clusters and cascading errors
- [ ] **Noise detection:** flags bots, browser extensions and flaky errors
- [ ] **Impact scoring:** users affected, key pages hit, trend direction
- [ ] **Weekly digest:** new errors, regressions, costliest unresolved issues

Phase 5: Developer workflow (months 12–15)

- [ ] **MCP server** so Claude, Cursor and Copilot can query runbooks
- [ ] **VS Code extension:** hover a function to see its linked errors
- [ ] **Slack commands:** `/errmap why <error>`, `/errmap owner <service>`
- [ ] **Draft fix PR:** AI proposes a patch as a draft for human review, never auto-merged
- [ ] **Postmortem generator:** timeline, root cause, fix and follow-ups

Phase 6: Teams and enterprise (months 15–24)

- [ ] SSO/SAML, SCIM, role-based access, audit logs
- [ ] PagerDuty, Opsgenie and incident.io integrations
- [ ] Jira and Linear ticket creation linked to runbooks
- [ ] Onboarding mode: "the 20 errors every new engineer on this service should know"
- [ ] Self-hosted enterprise licence; SOC 2 only once revenue justifies it

4. Code samples

These are starting sketches in TypeScript (server, SDKs) and Go (Cloud ingest), not finished production code.

4.1 Repo layout

```
errmap/
  packages/
    core/            @errmap/core      types, fingerprint, pipeline, scrubber
    analyzer-llm/    @errmap/analyzer  Anthropic / OpenAI / Ollama
    source-sentry/   @errmap/source-sentry
    source-otel/     @errmap/source-otel
    source-logs/     @errmap/source-logs
    store-notion/    @errmap/store-notion
    store-confluence/
    channel-slack/   @errmap/channel-slack
    channel-teams/
    sdk-node/        @errmap/node
    sdk-browser/     @errmap/browser
  apps/
    server/          Docker image (OSS product)
    cli/             npx errmap init
    dashboard/       Next.js (Cloud)
    ingest/          Go ingest API (Cloud)
  ee/                commercial features (SSO, audit logs)
```

4.2 Core types

```
// packages/core/src/types.ts
export interface ErrorEvent {
  message: string;
  stack: string;
  service: string;
  env: string;
  release?: string;
  timestamp: string;
  user?: { id?: string };
  breadcrumbs?: { type: string; message: string; at: string }[];
  meta?: Record<string, unknown>;
}

export interface Runbook {
  fingerprint: string;
  title: string;
  severity: "low" | "medium" | "high" | "critical";
  summary: string;
```

```
  likelyCauses: string[];
  fixes: { step: string; confidence: number }[];
  affectedCode: string[];
  owners: string[];
  tags: string[];
}

export interface DocRef { id: string; url: string }

export interface Source<T = unknown> { parse(input: T): ErrorEvent[] }

export interface DocStore {
  find(fingerprint: string): Promise<DocRef | null>;
  create(doc: Runbook): Promise<DocRef>;
  appendOccurrence(id: string, e: ErrorEvent, count: number): Promise<void>;
  markResolved(id: string, fix?: { pr: string; commit: string }):
Promise<void>;
}

export interface Channel {
  notify(a: { event: ErrorEvent; doc: DocRef; count: number; isNew: boolean;
threadKey: string }): Promise<void>;
}

export interface Analyzer {
  draft(e: ErrorEvent, similar: Runbook[]): Promise<Omit<Runbook,
"fingerprint">>;
}
```

4.3 Fingerprinting

```
// packages/core/src/fingerprint.ts
import { createHash } from "node:crypto";
import type { ErrorEvent } from "./types";

const normFrame = (l: string) =>
  l.trim()
    .replace(/:\d+:\d+/g, "")           // line:col
    .replace(/\?.*$/, "")               // query strings
    .replace(/[0-9a-f]{8,}/gi, "<id>")   // hashes, uuids
    .replace(/\d+/g, "<n>");
```

```
const normMessage = (m: string) =>
  m.replace(/(['"`]).*?\1/g, "<str>")
   .replace(/\b\d+\b/g, "<n>")
   .replace(/https?:\/\/\S+/g, "<url>");

export function fingerprint(e: ErrorEvent): string {
  const frames = e.stack
    .split("\n")
    .filter(l => !/node_modules|internal\//.test(l)) // app frames only
    .slice(0, 5)
    .map(normFrame);
  return createHash("sha1")
    .update([e.service, normMessage(e.message), ...frames].join("|"))
    .digest("hex");
}
```

4.4 PII scrubber

```
// packages/core/src/scrub.ts
const RULES: [RegExp, string][] = [
  [/[\w.+-]+@[\w-]+\.[\w.]+/g, "<email>"],
  [/\b(?:\d[ -]?){13,19}\b/g, "<card>"],
  [/\b(?:sk|pk|ghp|xox[bp])[-_][A-Za-z0-9-_]{10,}\b/g, "<token>"],
  [/Bearer\s+[A-Za-z0-9._-]+/g, "Bearer <token>"],
  [/\b\d{1,3}(?:\.\d{1,3}){3}\b/g, "<ip>"],
];

export function scrub<T>(value: T, extra: RegExp[] = []): T {
  const rules = [...RULES, ...extra.map(r => [r, "<redacted>"] as [RegExp,
string])];
  const clean = (s: string) => rules.reduce((acc, [re, rep]) =>
acc.replace(re, rep), s);
  return JSON.parse(JSON.stringify(value), (_k, v) => (typeof v === "string"
? clean(v) : v));
}
```

4.5 Pipeline

```
// packages/core/src/pipeline.ts
import { fingerprint } from "./fingerprint";
import { scrub } from "./scrub";
import type { Analyzer, Channel, DocStore, ErrorEvent } from "./types";
```

```
export interface Deps {
  store: DocStore;
  channels: Channel[];
  analyzer: Analyzer;
  counter: { incr(fp: string): Promise<number> };  // Redis or SQLite
  lock: { run<T>(key: string, fn: () => Promise<T>): Promise<T> };
}

export async function handle(raw: ErrorEvent, deps: Deps) {
  const event = scrub(raw);
  const fp = fingerprint(event);

  // lock stops two identical errors creating two runbooks at once
  return deps.lock.run(fp, async () => {
    const count = await deps.counter.incr(fp);
    let doc = await deps.store.find(fp);
    const isNew = !doc;

    if (!doc) {
      const draft = await deps.analyzer.draft(event, []);
      doc = await deps.store.create({ ...draft, fingerprint: fp });
    } else {
      await deps.store.appendOccurrence(doc.id, event, count);
    }

    // alert on first sighting, then on 10, 50, 100... to avoid spam
    if (isNew || [10, 50, 100, 500, 1000].includes(count)) {
      await Promise.all(
        deps.channels.map(c => c.notify({ event, doc: doc!, count, isNew,
threadKey: fp })),
      );
    }
    return { fp, doc, count, isNew };
  });
}
```

4.6 AI analyzer

```
// packages/analyzer-llm/src/index.ts
import Anthropic from "@anthropic-ai/sdk";
import { z } from "zod";
import type { Analyzer } from "@errmap/core";

const Schema = z.object({
  title: z.string().max(120),
  severity: z.enum(["low", "medium", "high", "critical"]),
  summary: z.string(),
  likelyCauses: z.array(z.string()).max(5),
  fixes: z.array(z.object({ step: z.string(), confidence:
z.number().min(0).max(1) })).max(5),
  affectedCode: z.array(z.string()),
  owners: z.array(z.string()),
  tags: z.array(z.string()),
});

const SYSTEM = `You are a senior SRE writing a runbook for a production
error.
Return ONLY JSON matching the schema. Be specific to the stack trace.
If unsure, say so and give lower confidence. Never invent file paths.`;

export function anthropicAnalyzer(opts: { model: string; apiKey: string }):
Analyzer {
  const client = new Anthropic({ apiKey: opts.apiKey });
  return {
    async draft(event, similar) {
      const msg = await client.messages.create({
        model: opts.model,
        max_tokens: 1500,
        system: SYSTEM,
        messages: [{
          role: "user",
          content: JSON.stringify({ event, similarRunbooks: similar.slice(0,
3) }),
        }],
      });
      const text = msg.content.map(b => (b.type === "text" ? b.text :
"")).join("");
      return Schema.parse(JSON.parse(text.replace(/```json|```/g,
```

```
"").trim()));
    },
  };
}
```

Set the model from config rather than hard-coding it, since model names change.

4.7 Sentry webhook receiver

```
// packages/source-sentry/src/router.ts
import express from "express";
import crypto from "node:crypto";
import type { ErrorEvent } from "@errmap/core";

export function sentryRouter(opts: {
  clientSecret: string;
  apiToken: string;
  org: string;
  enqueue: (e: ErrorEvent) => Promise<void>;
  onResolved: (issueId: string) => Promise<void>;
}) {
  const r = express.Router();

  r.post("/webhooks/sentry", express.raw({ type: "application/json" }), async
(req, res) => {
    const sig = req.header("sentry-hook-signature") ?? "";
    const expected = crypto.createHmac("sha256",
opts.clientSecret).update(req.body).digest("hex");
    const ok = sig.length === expected.length &&
      crypto.timingSafeEqual(Buffer.from(sig), Buffer.from(expected));
    if (!ok) return res.sendStatus(401);

    res.sendStatus(200); // reply fast, work later

    const resource = req.header("sentry-hook-resource");
    const body = JSON.parse(req.body.toString());
    const issue = body.data?.issue;
    if (resource !== "issue" || !issue) return;

    if (body.action === "created") {
      const latest = await fetch(

`https://sentry.io/api/0/organizations/${opts.org}/issues/${issue.id}/events/
```

```
latest/`,
        { headers: { Authorization: `Bearer ${opts.apiToken}` } },
      ).then(r => r.json());

      const frames = latest.entries?.find((x: any) => x.type === "exception")
        ?.data?.values?.[0]?.stacktrace?.frames ?? [];

      await opts.enqueue({
        message: issue.title,
        stack: frames.reverse().map((f: any) => `at ${f.function}
(${f.filename}:${f.lineNo})`).join("\n"),
        service: issue.project?.slug ?? "unknown",
        env: latest.tags?.find((t: any) => t.key === "environment")?.value ??
"production",
        release: latest.release?.version,
        timestamp: issue.firstSeen,
        meta: { source: "sentry", issueId: issue.id, permalink:
issue.permalink },
      });
    }

    if (body.action === "resolved") await opts.onResolved(issue.id);
  });

  return r;
}
```

4.8 Notion store

```
// packages/store-notion/src/index.ts
import { Client } from "@notionhq/client";
import type { DocStore } from "@errmap/core";

export function notionStore(opts: { token: string; databaseId: string }):
DocStore {
  const notion = new Client({ auth: opts.token });
  const text = (s: string) => [{ type: "text" as const, text: { content:
s.slice(0, 2000) } }];

  return {
    async find(fp) {
      const res = await notion.databases.query({
```

```
        database_id: opts.databaseId,
        filter: { property: "Fingerprint", rich_text: { equals: fp } },
      });
      const page: any = res.results[0];
      return page ? { id: page.id, url: page.url } : null;
    },

    async create(rb) {
      const page: any = await notion.pages.create({
        parent: { database_id: opts.databaseId },
        properties: {
          Name: { title: text(rb.title) },
          Fingerprint: { rich_text: text(rb.fingerprint) },
          Severity: { select: { name: rb.severity } },
          Status: { select: { name: "Open" } },
          Occurrences: { number: 1 },
          Tags: { multi_select: rb.tags.map(name => ({ name })) },
        },
        children: [
          { heading_2: { rich_text: text("Summary") } },
          { paragraph: { rich_text: text(rb.summary) } },
          { heading_2: { rich_text: text("Likely causes") } },
          ...rb.likelyCauses.map(c => ({ bulleted_list_item: { rich_text:
text(c) } })),
          { heading_2: { rich_text: text("How to fix") } },
          ...rb.fixes.map(f => ({ numbered_list_item: { rich_text:
text(`${f.step} (confidence ${Math.round(f.confidence * 100)}%)`) } })),
          { heading_2: { rich_text: text("Occurrences") } },
        ] as any,
      });
      return { id: page.id, url: page.url };
    },

    async appendOccurrence(id, e, count) {
      await notion.pages.update({ page_id: id, properties: { Occurrences: {
number: count }, "Last seen": { date: { start: e.timestamp } } } as any });
    },

    async markResolved(id, fix) {
      await notion.pages.update({ page_id: id, properties: { Status: {
select: { name: "Resolved" } }, "Fixed by": { url: fix?.pr ?? null } } as any
```

```
});
    },
  };
}
```

4.9 Slack channel

```
// packages/channel-slack/src/index.ts
import { WebClient } from "@slack/web-api";
import type { Channel } from "@errmap/core";

export function slackChannel(opts: { token: string; channel: string; threads:
Map<string, string> }): Channel {
  const slack = new WebClient(opts.token);
  return {
    async notify({ event, doc, count, isNew, threadKey }) {
      const text = isNew
        ? `:rotating_light: *New error* in \`${event.service}\`
(${event.env})\n>${event.message}\n:book: <${doc.url}|Runbook created>`
        : `:repeat: Seen *${count}×* — <${doc.url}|open runbook>`;

      const res = await slack.chat.postMessage({
        channel: opts.channel,
        text,
        thread_ts: isNew ? undefined : opts.threads.get(threadKey),
      });
      if (isNew && res.ts) opts.threads.set(threadKey, res.ts);
    },
  };
}
```

In production, store thread IDs in Redis or SQLite rather than an in-memory map.

4.10 Config file

```
// errmap.config.ts
import { defineConfig } from "@errmap/core";

export default defineConfig({
  sources: [
    { type: "sentry", org: "acme", clientSecret: env("SENTRY_SECRET"),
apiToken: env("SENTRY_TOKEN") },
    { type: "otel", port: 4318 },
    { type: "logs", path: "/var/log/app/*.log", service: "billing" },
  ],
  store: { type: "notion", token: env("NOTION_TOKEN"), databaseId:
env("NOTION_DB") },
  channels: [{ type: "slack", token: env("SLACK_TOKEN"), channel: "#errors"
}],
  analyzer: { provider: "anthropic", model: env("LLM_MODEL"), apiKey:
env("ANTHROPIC_API_KEY") },
  // analyzer: { provider: "ollama", model: "llama3.1", url:
"http://ollama:11434" },
  scrub: { extra: [/NIN-\d{11}/] },
  alerts: { thresholds: [10, 50, 100, 500] },
});
```

4.11 Node SDK (for teams without Sentry)

```
// packages/sdk-node/src/index.ts
type Crumb = { type: string; message: string; at: string };

export function init(opts: { dsn: string; service: string; release?: string;
env?: string }) {
  const crumbs: Crumb[] = [];
  const addCrumb = (type: string, message: string) => {
    crumbs.push({ type, message, at: new Date().toISOString() });
    if (crumbs.length > 30) crumbs.shift();
  };

  const capture = (err: unknown, extra: Record<string, unknown> = {}) => {
    const e = err instanceof Error ? err : new Error(String(err));
    return fetch(`${opts.dsn}/ingest`, {
      method: "POST",
      headers: { "content-type": "application/json" },
```

```
      body: JSON.stringify({
        message: `${e.name}: ${e.message}`,
        stack: e.stack ?? "",
        service: opts.service,
        env: opts.env ?? process.env.NODE_ENV ?? "production",
        release: opts.release,
        timestamp: new Date().toISOString(),
        breadcrumbs: [...crumbs],
        meta: extra,
      }),
    }).catch(() => {}); // never crash the host app
  };

  process.on("uncaughtException", err => { capture(err); });
  process.on("unhandledRejection", err => { capture(err); });

  return { capture, addCrumb };
}
```

Usage:

```
import { init } from "@errmap/node";
export const errmap = init({ dsn: "https://ingest.errmap.dev/p/abc123",
service: "billing-api", release: process.env.GIT_SHA });

// Express error middleware
app.use((err, req, res, next) => {
  errmap.capture(err, { route: req.path, method: req.method });
  res.status(500).json({ error: "Internal error" });
});
```

4.12 Browser SDK

```
// packages/sdk-browser/src/index.ts
export function init(opts: { dsn: string; service: string; release?: string
}) {
  const crumbs: { type: string; message: string; at: string }[] = [];
  const push = (type: string, message: string) => {
    crumbs.push({ type, message, at: new Date().toISOString() });
    if (crumbs.length > 30) crumbs.shift();
  };
```

```
  document.addEventListener("click", e => {
    const t = e.target as HTMLElement;
    push("click", `${t.tagName.toLowerCase()}${t.id ? "#" + t.id : ""}`);
  }, true);

  const origFetch = window.fetch;
  window.fetch = async (...args) => {
    const res = await origFetch(...args);
    push("http", `${res.status} ${String(args[0]).slice(0, 100)}`);
    return res;
  };

  const send = (message: string, stack = "") =>
    navigator.sendBeacon(`${opts.dsn}/ingest`, JSON.stringify({
      message, stack, service: opts.service, env: "production",
      release: opts.release, timestamp: new Date().toISOString(),
      breadcrumbs: [...crumbs], meta: { url: location.pathname, ua:
navigator.userAgent },
    }));

  window.addEventListener("error", e => send(e.message, e.error?.stack));
  window.addEventListener("unhandledrejection", e =>
send(String(e.reason?.message ?? e.reason), e.reason?.stack));
}
```

4.13 Log source parser

```
// packages/source-logs/src/parse.ts
import type { ErrorEvent } from "@errmap/core";

const START = /^(\w*(Error|Exception)|panic:|Traceback)/;
const FRAME = /^\s+(at |File |\S+\.go:\d+|goroutine )/;

export function* parseLogs(lines: Iterable<string>, service: string):
Generator<ErrorEvent> {
  let buf: string[] = [];
  const flush = (): ErrorEvent | null => {
    if (!buf.length) return null;
    const [message, ...rest] = buf;
    buf = [];
    return { message, stack: rest.join("\n"), service, env: process.env.ENV
?? "production", timestamp: new Date().toISOString() };
```

```
  };
  for (const line of lines) {
    if (START.test(line)) { const e = flush(); if (e) yield e;
buf.push(line); }
    else if (buf.length && FRAME.test(line)) buf.push(line);
    else { const e = flush(); if (e) yield e; }
  }
  const e = flush();
  if (e) yield e;
}
```

4.14 Cloud ingest API (Go)

```
// apps/ingest/main.go
package main

import (
        "encoding/json"
        "net/http"
        "time"

        "github.com/redis/go-redis/v9"
)

type Event struct {
        Message   string         `json:"message"`
        Stack     string         `json:"stack"`
        Service   string         `json:"service"`
        Env       string         `json:"env"`
        Release   string         `json:"release,omitempty"`
        Timestamp time.Time      `json:"timestamp"`
        Meta      map[string]any `json:"meta,omitempty"`
}

type Server struct {
        rdb    *redis.Client
        keys   KeyStore   // project key -> org
        quotas QuotaStore // monthly event limits per org
}

func (s *Server) ingest(w http.ResponseWriter, r *http.Request) {
        projectKey := r.PathValue("key")
```

```
        org, ok := s.keys.Lookup(r.Context(), projectKey)
        if !ok {
                http.Error(w, "unknown project", http.StatusUnauthorized)
                return
        }
        if !s.quotas.Allow(r.Context(), org) {
                http.Error(w, "quota exceeded", http.StatusTooManyRequests)
                return
        }

        r.Body = http.MaxBytesReader(w, r.Body, 256<<10) // 256 KB
        var ev Event
        if err := json.NewDecoder(r.Body).Decode(&ev); err != nil {
                http.Error(w, "bad payload", http.StatusBadRequest)
                return
        }

        payload, _ := json.Marshal(ev)
        err := s.rdb.XAdd(r.Context(), &redis.XAddArgs{
                Stream: "events",
                MaxLen: 1_000_000,
                Values: map[string]any{"org": org, "project": projectKey,
"event": payload},
        }).Err()
        if err != nil {
                http.Error(w, "try again", http.StatusServiceUnavailable)
                return
        }
        w.WriteHeader(http.StatusAccepted)
}

func main() {
        s := &Server{rdb: redis.NewClient(&redis.Options{Addr: "redis:6379"})
/* keys, quotas */}
        mux := http.NewServeMux()
        mux.HandleFunc("POST /p/{key}/ingest", s.ingest)
        mux.HandleFunc("GET /healthz", func(w http.ResponseWriter, _
*http.Request) { w.WriteHeader(200) })
        http.ListenAndServe(":8080", mux)
}
```

4.15 Similar-error search (pgvector)

```
CREATE EXTENSION IF NOT EXISTS vector;

CREATE TABLE runbooks (
  id           uuid PRIMARY KEY DEFAULT gen_random_uuid(),
  org_id       uuid NOT NULL,
  fingerprint  text NOT NULL,
  title        text NOT NULL,
  status       text NOT NULL DEFAULT 'open',
  fixed_by_pr  text,
  occurrences  int  NOT NULL DEFAULT 1,
  first_seen   timestamptz NOT NULL,
  last_seen    timestamptz NOT NULL,
  embedding    vector(1024),
  UNIQUE (org_id, fingerprint)
);

-- top 3 similar resolved runbooks
SELECT id, title, fixed_by_pr, 1 - (embedding <=> $1) AS similarity
FROM runbooks
WHERE org_id = $2 AND status = 'resolved'
ORDER BY embedding <=> $1
LIMIT 3;
```

4.16 Deploy correlation

```
// packages/core/src/deploys.ts
export function correlate(firstSeen: Date, deploys: { version: string; sha:
string; at: Date }[]) {
  const before = deploys
    .filter(d => d.at <= firstSeen && firstSeen.getTime() - d.at.getTime() <
60 * 60 * 1000)
    .sort((a, b) => b.at.getTime() - a.at.getTime())[0];
  if (!before) return null;
  const mins = Math.round((firstSeen.getTime() - before.at.getTime()) /
60000);
  return { ...before, minutesBefore: mins, note: `Started ${mins} min after
${before.version} (${before.sha.slice(0, 7)})` };
}
```

4.17 MCP server

```
// apps/mcp/src/index.ts
import { McpServer } from "@modelcontextprotocol/sdk/server/mcp.js";
import { StdioServerTransport } from
"@modelcontextprotocol/sdk/server/stdio.js";
import { z } from "zod";
import { runbooks } from "./db";

const server = new McpServer({ name: "errmap", version: "0.1.0" });

server.tool("find_runbook", { query: z.string() }, async ({ query }) => {
  const hits = await runbooks.search(query, { limit: 3 });
  return { content: hits.map(h => ({ type: "text" as const, text:
`${h.title}\n${h.url}\nFix: ${h.topFix}` })) };
});

server.tool("errors_for_file", { path: z.string() }, async ({ path }) => {
  const hits = await runbooks.byFile(path);
  return { content: [{ type: "text" as const, text: JSON.stringify(hits,
null, 2) }] };
});

await server.connect(new StdioServerTransport());
```

4.18 Dockerfile and docker-compose

```
FROM node:20-alpine AS build
WORKDIR /app
COPY package*.json turbo.json ./
COPY packages ./packages
COPY apps/server ./apps/server
RUN npm ci && npx turbo run build --filter=server...

FROM node:20-alpine
WORKDIR /app
ENV NODE_ENV=production
COPY --from=build /app/apps/server/dist ./dist
COPY --from=build /app/node_modules ./node_modules
USER node
EXPOSE 3000
HEALTHCHECK CMD wget -qO- http://localhost:3000/healthz || exit 1
CMD ["node", "dist/server.js"]
```

```
# docker-compose.yml
services:
  errmap:
    image: ghcr.io/errmap/errmap:latest
    ports: ["3000:3000"]
    env_file: .env
    volumes: ["./errmap.config.ts:/app/errmap.config.ts:ro"]
    depends_on: [redis]
  redis:
    image: redis:7-alpine
  # optional local AI
  ollama:
    image: ollama/ollama
    profiles: ["local-ai"]
```

4.19 Release workflow

```
# .github/workflows/release.yml
name: release
on:
  push:
    tags: ["v*"]
jobs:
```

```
  docker:
    runs-on: ubuntu-latest
    permissions: { contents: read, packages: write }
    steps:
      - uses: actions/checkout@v4
      - uses: docker/login-action@v3
        with: { registry: ghcr.io, username: "${{ github.actor }}", password:
"${{ secrets.GITHUB_TOKEN }}" }
      - uses: docker/build-push-action@v6
        with:
          push: true
          tags: ghcr.io/${{ github.repository }}:${{ github.ref_name
}},ghcr.io/${{ github.repository }}:latest
  npm:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with: { node-version: 20, registry-url: "https://registry.npmjs.org"
}
      - run: npm ci && npx turbo run build && npx changeset publish
        env: { NODE_AUTH_TOKEN: "${{ secrets.NPM_TOKEN }}" }
```

5. Runbook sample

This is what Errmap would write into Notion or Confluence for one error. All names, numbers and links are invented for illustration.

> **RB-042 · TypeError: Cannot read properties of undefined (reading 'amount') in calculateInvoiceTotal**
>
> | Field | Value |
> |---|---|
> | Service | billing-api |
> | Environment | production |
> | Severity | High |
> | Status | Resolved (regression watch on) |
> | Occurrences | 214 (last 7 days: 38) |
> | Users affected | 61 |
> | First seen | 3 Sep 2026, 14:12 (4 min after release v2.3.1, commit a1b2c3d) |
> | Last seen | 15 Sep 2026, 09:47 |
> | Owners | @payments-team (CODEOWNERS), last touched by Ada O. |
> | Fixed by | PR #212 "Guard empty line items on draft invoices" |
> | Source | Sentry issue BILLING-7Q |
> | Fingerprint | 9f3c…e21a |

Summary

`calculateInvoiceTotal` crashes when an invoice has no line items. Draft invoices created from the new "Duplicate invoice" button (added in v2.3.1) can have an empty `lineItems` array, and the reducer reads `items[0].amount` to pick the currency.

Impact

- Customers cannot open or send duplicated draft invoices (HTTP 500).
- No data loss; invoices are saved but not viewable.
- Checkout and payments are not affected.

Stack trace (app frames)

```
TypeError: Cannot read properties of undefined (reading 'amount')
    at calculateInvoiceTotal (src/invoices/totals.ts:27)
    at InvoiceService.render (src/invoices/service.ts:88)
    at GET /invoices/:id (src/routes/invoices.ts:41)
```

Breadcrumbs before the error

Time	Type	Event
14:11:58	click	button#duplicate-invoice
14:11:59	http	201 POST /invoices/duplicate
14:12:01	http	500 GET /invoices/inv_<id>

Likely causes (AI, ranked)

1. Empty `lineItems` on duplicated drafts; `items[0]` is undefined. *(confidence 86%)*
2. Line items loaded lazily and not yet present when totals are computed. *(41%)*
3. Migration left legacy invoices with `null` line items. *(18%)*

How to fix

1. Guard the empty case and take currency from the invoice, not the first item:

```
export function calculateInvoiceTotal(inv: Invoice) {
  const items = inv.lineItems ?? [];
  if (items.length === 0) return { amount: 0, currency: inv.currency };
  return {
    amount: items.reduce((sum, i) => sum + i.amount * i.quantity, 0),
    currency: inv.currency,
  };
}
```

2. Add a test for a draft with zero line items.
3. In "Duplicate invoice", copy line items in the same transaction as the invoice.

Quick mitigation (if a fix can't ship yet)

- Turn off the `invoice-duplicate` feature flag.

How to verify

- Occurrences stop after the fix is deployed (Errmap marks the runbook "Resolved" when Sentry resolves the issue).
- `GET /invoices/:id` 5xx rate for drafts returns to baseline.

Similar errors

Runbook	Similarity	Status
RB-017 "reading 'currency' of undefined in quoteTotal"	0.91	Resolved by PR #88
RB-031 "Cannot read 'price' of undefined in cartSummary"	0.78	Resolved by PR #154

Pattern note (error map)

This is the third "first item of an empty list" error in money-calculation code this year. Consider a shared `sumMoney(items, currency)` helper with an empty-list case, and a lint rule against `[0]` on arrays in `src/invoices` and `src/cart`.

Occurrence log

Date	Count	Release	Note
15 Sep 2026	1	v2.3.0 (rollback node)	Old pod still running
5 Sep 2026	0	v2.3.2	Fix deployed (PR #212)
4 Sep 2026	131	v2.3.1	Peak
3 Sep 2026	82	v2.3.1	First seen

Discussion

- Slack thread: #errors, 3 Sep 2026, 14:12
- Human notes: *"Also check the PDF export path, it uses the same helper." (Ada O.)*

AI-generated sections are marked; confidence scores are the model's own estimate and should be checked by an engineer.

6. Slack alert samples

Alerts stay in one thread per error, so a noisy error never floods the channel.

New error (top-level message)

```
🚨 New error · billing-api · production · High
TypeError: Cannot read properties of undefined (reading 'amount')
src/invoices/totals.ts:27 · calculateInvoiceTotal

🕒 Started 4 min after v2.3.1 (a1b2c3d)
👤 Owner: @payments-team
🔎 Looks like RB-017 (91% similar, fixed by PR #88)

📘 Runbook: RB-042  ·  🐞 Sentry: BILLING-7Q
[Acknowledge]  [Assign to me]  [Mute 24h]
```

Repeat (threaded reply)

```
🔁 Seen 50× · 22 users · last 2h
📘 Same runbook: RB-042
```

Resolved (threaded reply)

```
✅ Resolved by PR #212 (merged by Ada O.)
Runbook updated with the fix.
```

Regression (new top-level message, linked back)

```
⚠️ Regression · billing-api · RB-042 is back
Previously fixed by PR #212 on 5 Sep 2026
Returned in v2.3.0 on node pod-7 (rollback?)
📘 Runbook: RB-042 (occurrence log updated)
```

Weekly digest (Monday 09:00)

```
📊 Errmap weekly · 8–14 Sep 2026
New errors: 6   Regressions: 1   Resolved: 9
Top unresolved:
  1. RB-051 Stripe webhook timeout · 312× · checkout-api
  2. RB-048 Redis ECONNRESET · 140× · worker
  3. RB-044 Hydration mismatch /pricing · 97× · web
Noisiest service: checkout-api (41% of events)
```

Slash commands

```
/errmap why ECONNRESET redis
→ RB-048 "Redis ECONNRESET in worker" · fix: raise keepAlive, add retry with
backoff

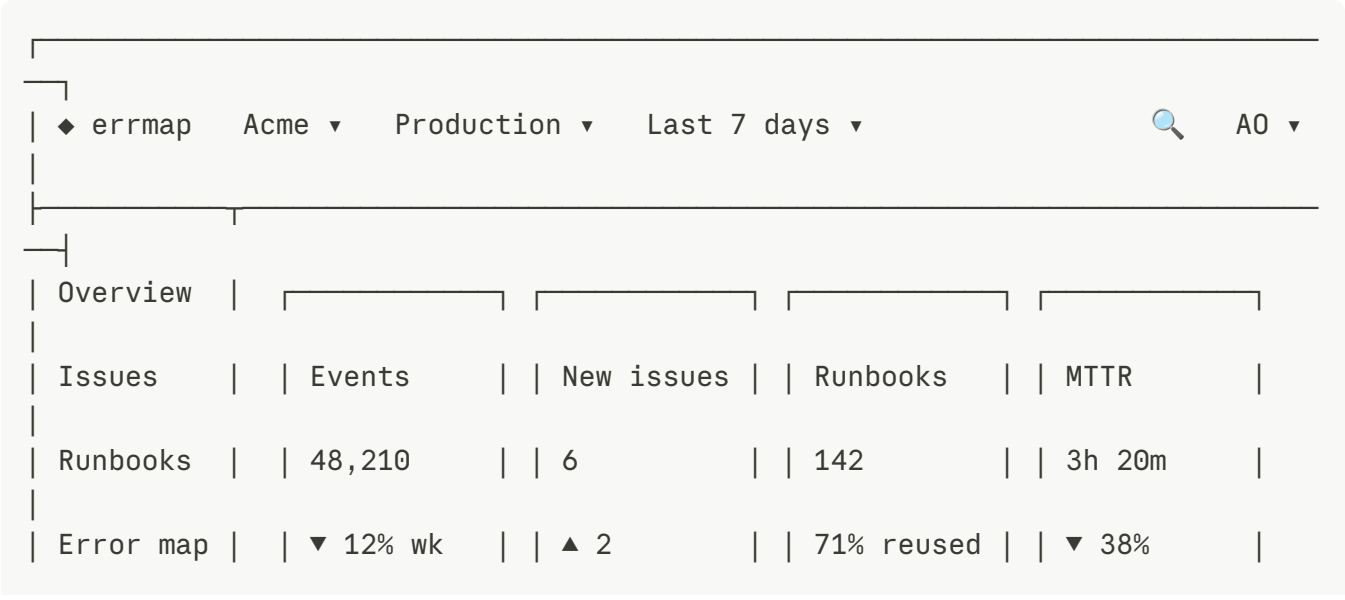
/errmap owner checkout-api
→ @payments-team (CODEOWNERS) · on call: Tunde A.
```

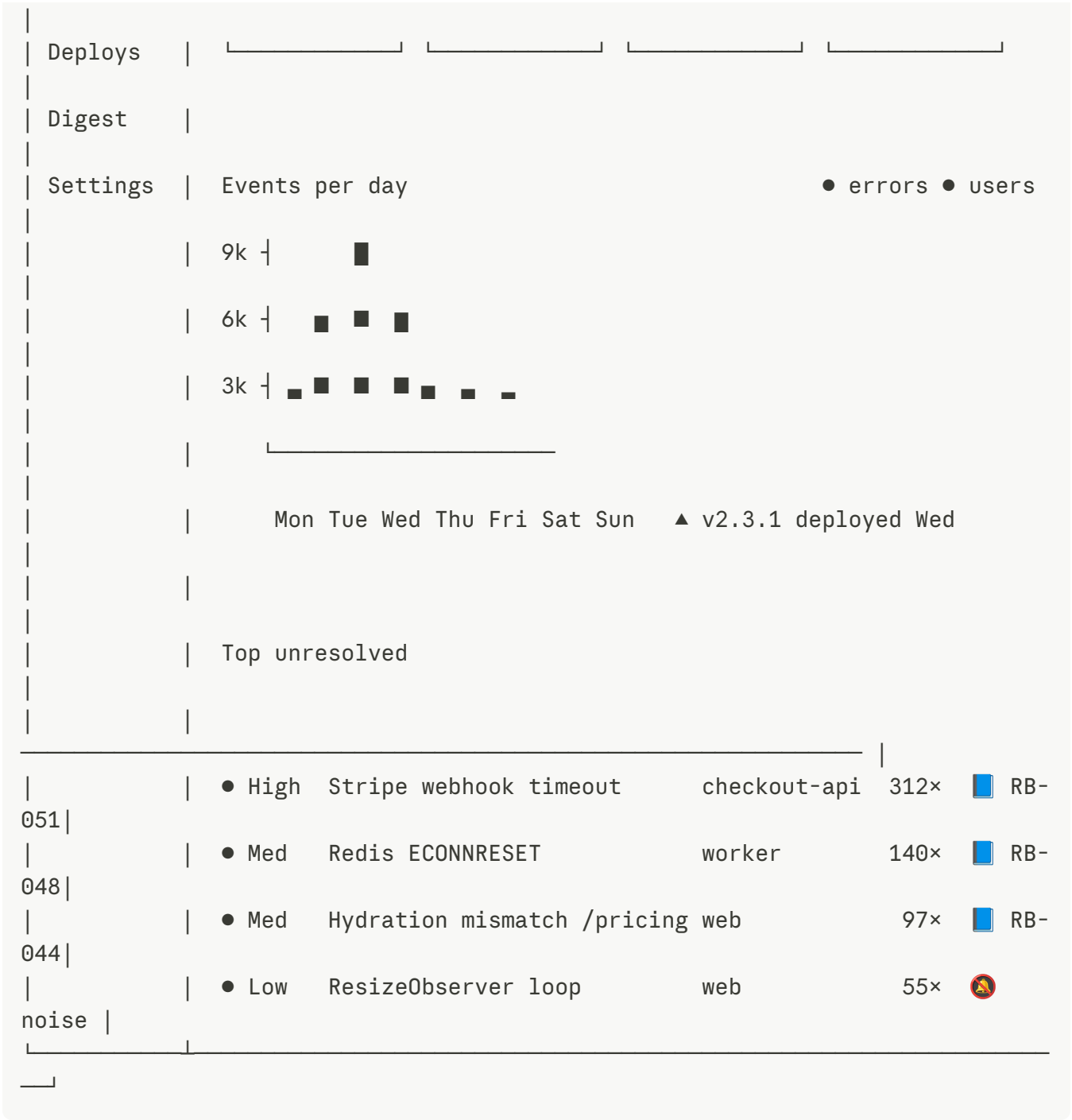
7. Dashboard sample (Cloud)

The dashboard has five screens; the error map and runbook pages are what make it different from a standard error tracker.

Screen	Purpose	Key elements
Overview	Health at a glance	Stat cards, events chart, top issues, recent deploys
Issues	Triage list	Filters, status, owner, occurrences, runbook link
Runbook	One error in depth	Section 5 content, edit, similar errors, occurrence chart
Error map	Patterns across services	Service graph, clusters, cascades
Settings	Setup	Projects, API keys, integrations, AI budget, team, billing

7.1 Overview screen




Deploys
Digest
Settings
Events per day
● errors ● users
9k
6k
3k
Mon Tue Wed Thu Fri Sat Sun
▲ v2.3.1 deployed Wed
Top unresolved
● High Stripe webhook timeout checkout-api 312× RB-051
● Med Redis ECONNRESET worker 140× RB-048
● Med Hydration mismatch /pricing web 97× RB-044
● Low ResizeObserver loop web 55× noise

7.2 Issues screen

```
Issues   [Unresolved ▾] [All services ▾] [Severity ▾] [Owner ▾]  Search…
──────────────────────────────────────────────────────────────────────────
  Sev   Title                                   Service       Count  Users
Owner          Runbook  Trend
  ●H    Stripe webhook timeout                  checkout-api   312     88
@payments      RB-051   ▂▅▇
  ●M    Redis ECONNRESET                        worker         140      0
@platform      RB-048   ▅▇▅
  ●M    Hydration mismatch on /pricing          web             97     64
@web           RB-044   ▂▅▂
  ▲R    TypeError reading 'amount' (regression) billing-api      1      1
@payments      RB-042   ▁
  ●L    ResizeObserver loop limit               web             55     40  —
noise          ▃▃▃
──────────────────────────────────────────────────────────────────────────
[Resolve]  [Assign]  [Merge]  [Mark as noise]  [Regenerate runbook]
```

7.3 Runbook screen

```
← Issues / RB-042                                          [Edit] [Open in
Notion] [Share]
TypeError: Cannot read properties of undefined (reading 'amount')
● High   ✅ Resolved by PR #212   billing-api   214×   61 users   Owner
@payments-team

┌ Occurrences (30 days) ──────────────────────┐  ┌ Deploy correlation
──────────────┐
|      ■                                      |  | First seen 4 min after
v2.3.1   |
|    ▄ ■ ▁                              ▁     |  | Commit a1b2c3d by Ada O.
|
| ──┴─┴─┴──────────────────────────────────┴── |  | Fixed in v2.3.2 (PR #212)
|
└─────────────────────────────────────────────┘
└──────────────────────────────────────┘

Summary · Impact · Stack · Breadcrumbs · Causes · Fix · Verify · Similar ·
History
(sections from the runbook sample, editable; AI sections tagged ✨)

Similar errors                          Human notes
 RB-017  0.91  resolved PR #88           "Also check the PDF export path." —
Ada O.
 RB-031  0.78  resolved PR #154          [Add note]
```

7.4 Error map screen

```
flowchart LR
  W[web<br/>97 events] -->|calls| C[checkout-api<br/>312 events]
  C -->|calls| B[billing-api<br/>214 events]
  C -->|webhook| S[Stripe<br/>timeouts]
  B -->|queue| K[worker<br/>140 events]
  K -->|cache| R[Redis<br/>ECONNRESET]
  R -. cascades .-> C
```

Nodes are services sized by error volume; edges show calls, and dotted edges are cascades detected when one service's errors reliably precede another's within a short window. Clicking a node lists its error clusters (e.g. "empty list access in money code: 3 runbooks").

7.5 Settings: AI budget panel

```
AI usage this month                                  Budget: £40 / month   [Change]
████████████████░░░░░░░░░░  £23.60 used (59%)
Runbooks drafted: 118   Re-drafts: 14   Similar-search queries: 2,310
Provider: Anthropic  ▾      Fallback: Ollama (self-hosted) ▾
When budget is reached:  (•) Use cached / template runbooks  ( ) Stop
drafting
```

7.6 Suggested frontend stack

- Next.js (App Router) and TypeScript, matching your existing monorepo work
- Tailwind and shadcn/ui for components; Recharts or ECharts for charts
- React Flow for the error map graph
- TanStack Query for data fetching; Postgres via Drizzle; ClickHouse for event queries

8. Metrics sample

Track four groups of metrics monthly: product value, OSS adoption, business, and AI cost. All targets below are illustrative starting points, not forecasts.

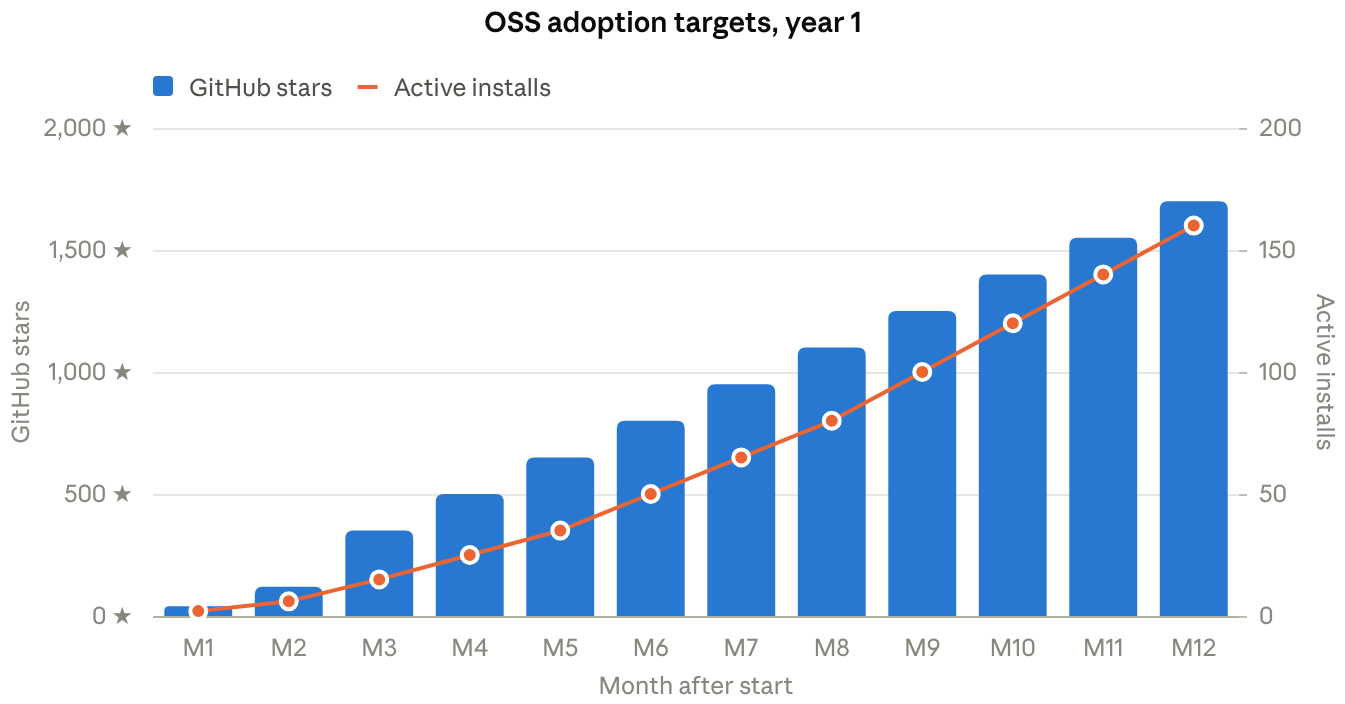
8.1 Product value metrics

These show whether Errmap actually saves engineers time.

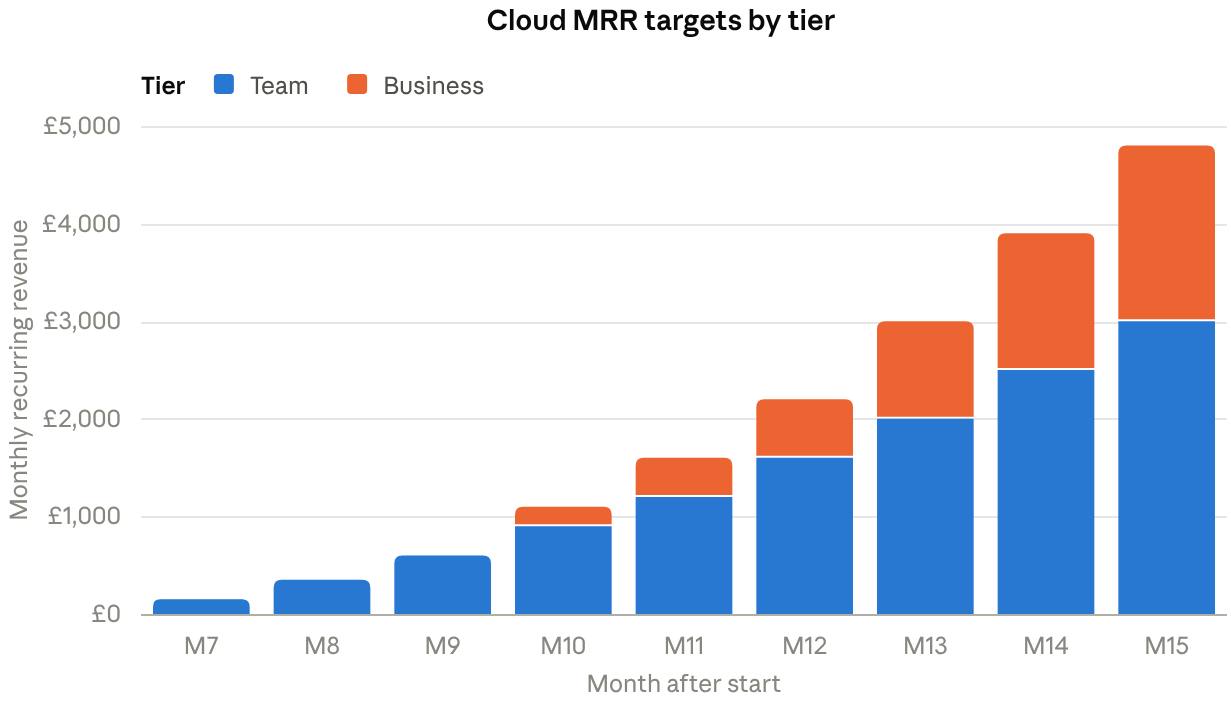
Metric	Definition	Year-1 target
Runbook reuse rate	Share of alerts that link to an existing runbook	Above 60%
Similar-error hit rate	New errors where a suggested similar runbook was clicked	Above 30%
MTTR	Median time from first seen to resolved	Down 30% vs the team's baseline
Runbook accuracy	Engineer rating of AI fix (thumbs up / down)	Above 70% helpful
Resolution link rate	Resolved runbooks linked to a PR or commit	Above 50%

Metric	Definition	Year-1 target
Noise ratio	Events auto-flagged as noise	Tracked, no target
Alert volume per error	Slack messages per unique error	Below 3

8.2 OSS adoption and business targets



Illustrative targets; M3 = launch month. Active installs = servers sending a weekly heartbeat (opt-in).



Illustrative; Cloud beta opens around M6, paid plans from M7.

Illustrative targets written for this plan · months 1–15

Metric	Where to get it	Why it matters
GitHub stars, forks	GitHub insights	Visible interest; Global Talent evidence
Docker pulls	GHCR / Docker Hub	Real trial usage
npm weekly downloads	npm stats	SDK and adapter usage
Active installs	Opt-in anonymous heartbeat	The honest usage number
External contributors, merged PRs	GitHub	Community evidence
Issues opened by strangers	GitHub	Proof of real users
Cloud signups, activation	Product analytics (PostHog)	Activation = first runbook created within 24 hours
Free-to-paid conversion	Stripe + analytics	Target 3–5%
Net revenue retention	Stripe	Target above 100%

8.3 AI cost metrics

AI calls are the largest variable cost, so meter them from day one.

Metric	Target
AI cost per runbook	Below £0.05
AI cost per org per month (Team tier)	Below 15% of the plan price
Cache hit rate (repeat errors skip AI)	Above 80%
Re-draft rate (engineers ask for a new draft)	Below 15%
Orgs hitting their AI budget cap	Below 5%

8.4 Evidence tracker (Global Talent)

Keep a dated monthly log; assessors value trends over time.

Month	Stars	Docker pulls	npm downloads/wk	External PRs	Talks/posts	Testimonials
Example: M3	350	1,200	180	2	Show HN post, dev.to article	1

8.5 Instrumentation sample

```
// apps/server/src/metrics.ts  (Prometheus)
import client from "prom-client";

export const metrics = {
  events: new client.Counter({ name: "errmap_events_total", help: "Events
received", labelNames: ["source", "service"] }),
  runbooksCreated: new client.Counter({ name:
"errmap_runbooks_created_total", help: "New runbooks" }),
  runbookReused: new client.Counter({ name: "errmap_runbook_reused_total",
help: "Alerts linked to an existing runbook" }),
  aiCost: new client.Counter({ name: "errmap_ai_cost_gbp_total", help: "AI
spend in GBP", labelNames: ["provider"] }),
  aiLatency: new client.Histogram({ name: "errmap_ai_draft_seconds", help:
"AI draft time", buckets: [1, 2, 5, 10, 20] }),
};

// runbook reuse rate = reused / (reused + created)
```

9. Pricing, licensing and go-to-market

The plan is an Apache 2.0 open core with paid Cloud and enterprise features, priced on events plus AI runbooks.

9.1 Pricing (indicative)

Tier	Price	Events / month	AI runbooks	Key features
OSS	Free	Unlimited (self-hosted)	Bring your own key	All sources, stores, channels
Cloud Free	£0	5,000	20	1 project, 7-day history

Tier	Price	Events / month	AI runbooks	Key features
Team	£25–49	100,000	Unlimited (fair use)	Integrations, deploy correlation, 90-day history
Business	£199+	1,000,000	Unlimited	SSO, audit logs, MCP/IDE, error map, priority support
Enterprise	Custom	Custom	Custom	Self-hosted `/ee`, UK/EU residency, SLA

Overage: charge per extra 100k events. Cap AI spend per org and fall back to template runbooks when the cap is hit.

9.2 Licensing

Option	Can others modify?	Protection	Fit for Global Talent
MIT / Apache 2.0	Yes	None against resellers	Best
AGPL	Yes	Must share changes if hosted	Good
Elastic 2.0 / FSL / BSL	Privately	Blocks competing hosted service	Weaker (not OSI open source)
Closed source	No	Full	Weak

Recommended: Apache 2.0 for `packages/` and `apps/server`; commercial licence for `ee/`; switch the core to AGPL only if a large player starts reselling it. You cannot fully prevent modification in open source. Your real protection is the trademark, the hosted service, and being the active maintainer.

Must-dos before launch:

- [ ] CLA or DCO so you can relicense later
- [ ] Trademark search and registration for the name (UK IPO)
- [ ] ENODA written confirmation on IP; build only on personal time and equipment
- [ ] Legal review of licence and terms (not legal advice here)

9.3 Go-to-market

Stage	Channel	Action
Launch	Hacker News, dev.to, r/devops, r/node	Show HN with the demo GIF and a "how it works" post
Launch	Sentry community, integration listing	Position as a Sentry companion
Months 3–6	Edinburgh and London JS / DevOps / AI meetups	Talk: "From alerts to runbooks with LLMs"
Months 3–6	Blog series	Fingerprinting, PII scrubbing, evaluating AI runbooks
Months 5–9	Direct outreach	10 discovery calls, 5–10 design partners
Months 12+	Conferences	CFPs for JSConf, NodeConf EU, SREcon, KubeCon
Ongoing	Marketplace listings	Slack App Directory, Notion and Atlassian marketplaces

9.4 Competitive landscape

Check competitors' current features before launch; this space moves quickly.

Player	What they do	Errmap's angle
Sentry, Rollbar, Bugsnag	Error capture and grouping, some AI debugging	Integrate, don't compete; own the knowledge layer
incident.io, Rootly, PagerDuty	Incident response, AI summaries	Error-level runbooks before anything becomes an incident
Notion AI, Confluence AI	General doc AI	Automatic, error-driven, deduplicated docs
Internal wikis	Manual runbooks	Written and maintained automatically

10. Global Talent, risks and next steps

Adoption is the evidence, not the code: aim for a public launch within 3 months and a dated record of usage after it.

10.1 Evidence mapping

Check the current Tech Nation / endorsing body criteria before collecting evidence, as they change.

Phase	Evidence produced	Likely criterion
1–2	Stars, downloads, external PRs, adapters built by others	Contributions to the field via open source
2–5	Talks, blog posts, conference CFPs	Recognition / thought leadership
3	Design-partner letters, companies using Cloud	Innovation as a founder
3–6	Paying customers, revenue, hires	Commercial impact
All	Letters from maintainers or users who rely on it	Recommendation letters

10.2 Risks

Risk	Likelihood	Mitigation
Sentry or incident tools ship the same feature	Medium	Stay multi-source; own the doc-store and fix-history angle
Low adoption after launch	Medium	Kill criterion below; talk to users before building Cloud
Companies won't send stack traces to AI	High	Self-host, PII scrubbing, Ollama option, UK/EU hosting
AI runbooks are wrong	Medium	Confidence scores, human edit, feedback loop, never auto-fix
AI cost overruns	Medium	Caching, per-org caps, cheaper models for repeats

Risk	Likelihood	Mitigation
Employment IP conflict with ENODA	Low if handled	Written confirmation, own time and equipment
Burnout (ENODA, FRSTDAY, Bubbl, MSc)	High	Ship Phases 1–2 only; Cloud is conditional

10.3 Kill and pivot criteria

- Fewer than 200 stars or 10 active installs within 3 months of launch → rethink positioning before Phase 3.
- Fewer than 3 of 10 interviewed teams would pay for the knowledge layer → stay OSS-only.
- Runbook helpful rate below 50% after 2 months → fix AI quality before adding features.

10.4 Next 30 days

- [ ] Pick the final name; check domain, npm scope and trademark
- [ ] Get ENODA IP confirmation in writing
- [ ] Create the monorepo with `core`, `source-sentry`, `store-notion`, `channel-slack`, `server`
- [ ] Implement fingerprint, scrubber and pipeline with tests
- [ ] Wire a sample Express app to Sentry and a test Notion workspace end to end
- [ ] Write the README quickstart and record the demo GIF
- [ ] Set up the evidence tracker spreadsheet
- [ ] Book 5 discovery calls with engineering leads