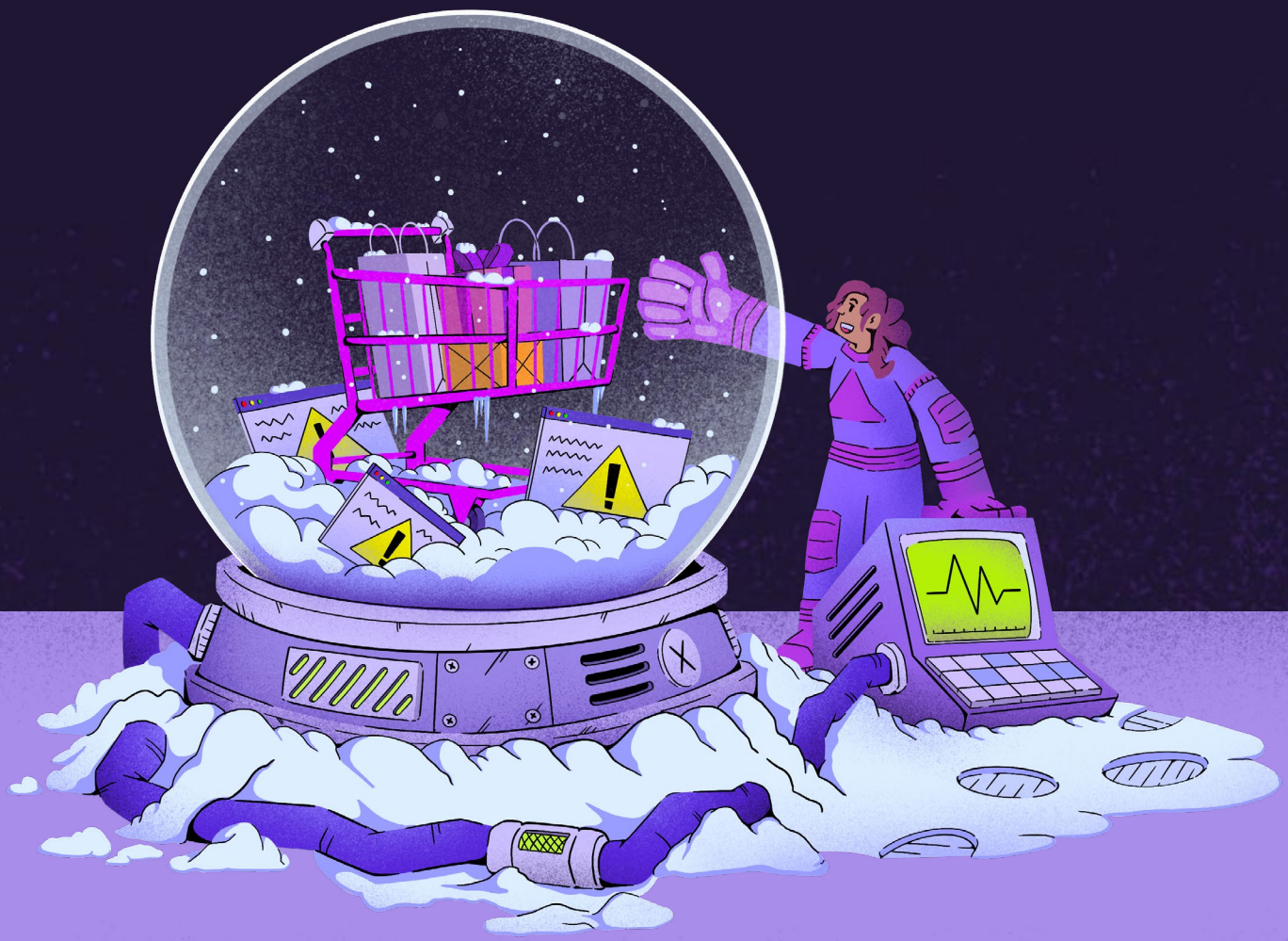
SENTRY

Monitoring Critical E-commerce Experiences: Developer's Checklist



Nothing tests your nerves like a checkout error during peak traffic (looking at you, Black Friday). When errors or performance degradations show up — and let's be honest, they will — it's critical you get answers about what's breaking and how to fix it.

E-Commerce Application Monitoring Readiness

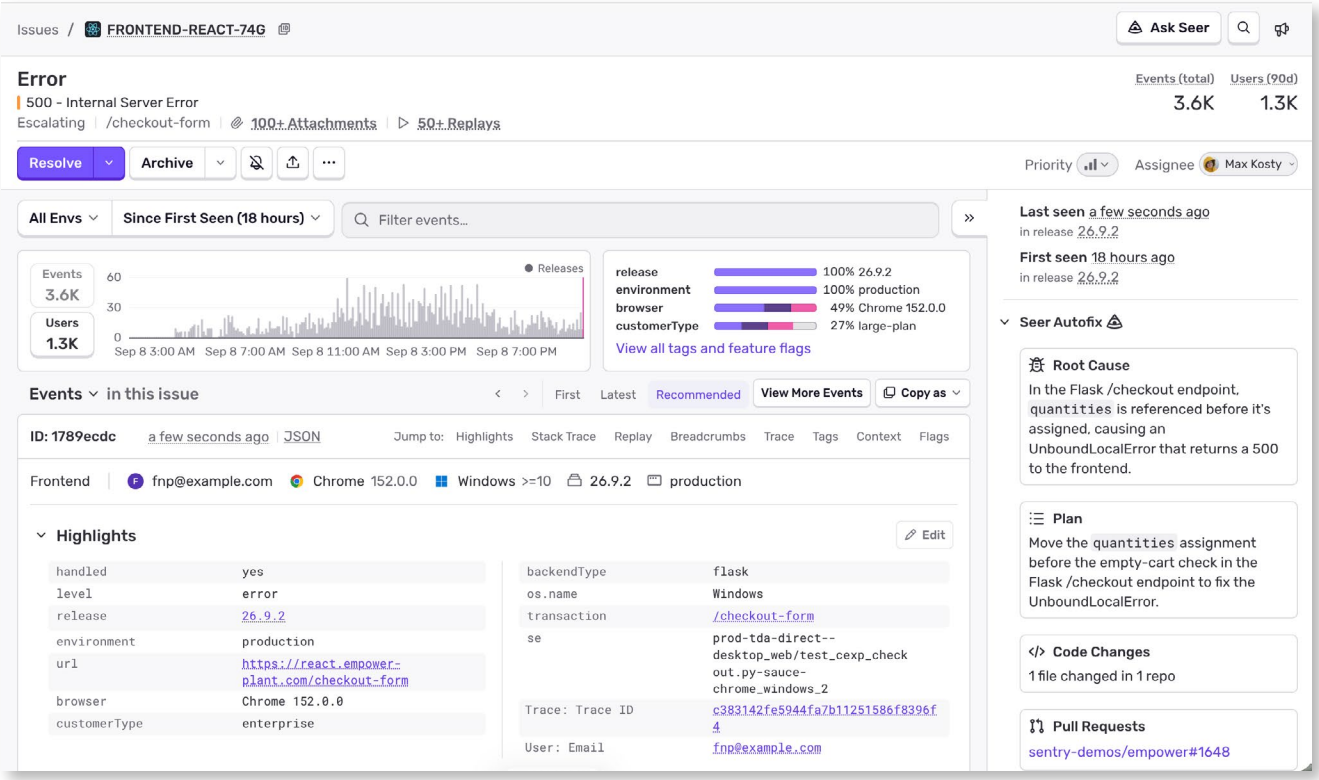
When something breaks during peak traffic, you're not looking for "more data." You're looking for answers: What broke? How many people hit it? Where did the time go? What did the user see? Comprehensive monitoring means each of those questions already has an answer waiting, proven out during normal traffic, so peak season isn't the first time you're testing it.

Here's your e-commerce monitoring checklist, mapped to the question each piece answers:

- **"What just broke?"** Implement error monitoring to catch every exception in checkout and payment flows, with the stack trace and context to fix it fast.
- **"Is this worth waking someone up for?"** Prioritize fixes with smart error alerting so noise gets filtered and revenue-threatening issues get routed to the right person immediately.
- **"What did the customer actually experience?"** Understand the user experience with session replays and user feedback to see the exact steps that led to a complaint, not just the error behind it.
- **"Where did the time go, and how many people were affected?"** Optimize site performance using distributed tracing to follow slow requests and logs to see the state behind a decision, before either costs you a conversion.
- **"Can someone else fix the bugs?"** Let Sentry fix issues for you so straightforward issues get resolved on their own, and fewer problems are waiting for you in the morning.

Implement error monitoring

Every error in your critical experiences like sign-in and checkout is potential revenue walking out the door. Readiness starts with making sure you're capturing errors in prod so you don't have to wait for customer complaints to tell you what's wrong. Use Sentry's error monitoring to automatically track exceptions across your checkout and payment systems, complete with the context you need to fix issues quickly.



- **Prioritize what matters:** Issues default to a "recommended" sort—surfacing spikes, persistent problems, and new issues first. Grouped issues keep related failures together, so you see high-impact errors like payment failures, not noise.
- **Understand the full picture:** Get the connected context you need to understand what went wrong and why, with readable stack traces, tags, and breadcrumbs of user actions so you can see what happened leading up to an error.
- **Detect, diagnose and fix issues faster:** Connect projects directly to your codebase to let Seer, Sentry's AI debugging and remediation tool, automatically diagnose and fix problems in your code. Autofix lets Seer generate pull requests directly in your connected repos or using the coding agent of your choice.

Prioritize fixes with smart error alerting

If you can't separate noise from critical issues, you risk missing the problems that actually cost you money. Error capture is automatic: every exception in checkout and payment flows becomes an issue immediately. The next step is routing issues to the right person and deciding what else beyond errors deserves tracking.

First, set up Alerts that route what matters, including your existing error issues:

- **Filter before you notify:** Use Sentry's WHEN/IF/THEN alert builder to narrow which issues actually page someone, for example only fatal events, or new issues rather than recurring ones. Environment filters keep staging noise out of production channels.
- **Route by severity and channel:** Critical payment errors page on-call via PagerDuty or Opsgenie; medium-severity issues go to Slack. One Alert can cover every project and Monitor, so routing logic isn't rebuilt each time.
- **Throttle and test:** Set how often an Alert can re-fire for the same issue so a flapping error doesn't page someone every few minutes, and use Send Test Notification to confirm routing works before you need it to.

Then, layer on Monitors for what automatic error capture won't catch.
Monitors watch other signals: spans, logs, releases, metrics, uptime, and scheduled jobs, and turn a breach into an issue just like an error, with the same triage and assignment. A few worth having before peak traffic:

- **Uptime Monitor on checkout and payment endpoints:** Get an issue the moment your payment gateway or checkout API goes down, times out, or returns an unexpected response, often before customers complain.
- **Metric Monitor on crash-free rate or failed-payment volume:** Set a fixed threshold for known bad conditions (like crash-free session rate dropping below 99%), and a dynamic threshold for metrics that swing with traffic, so a spike gets flagged against your own baseline, not a number you picked once and forgot about.
- **Track the signals errors won't catch:** Not every problem throws an exception. Use Sentry Metrics to send counters, gauges, and distributions for things like checkout. failed, cart.abandoned, or payment.retry directly from your code. Because every

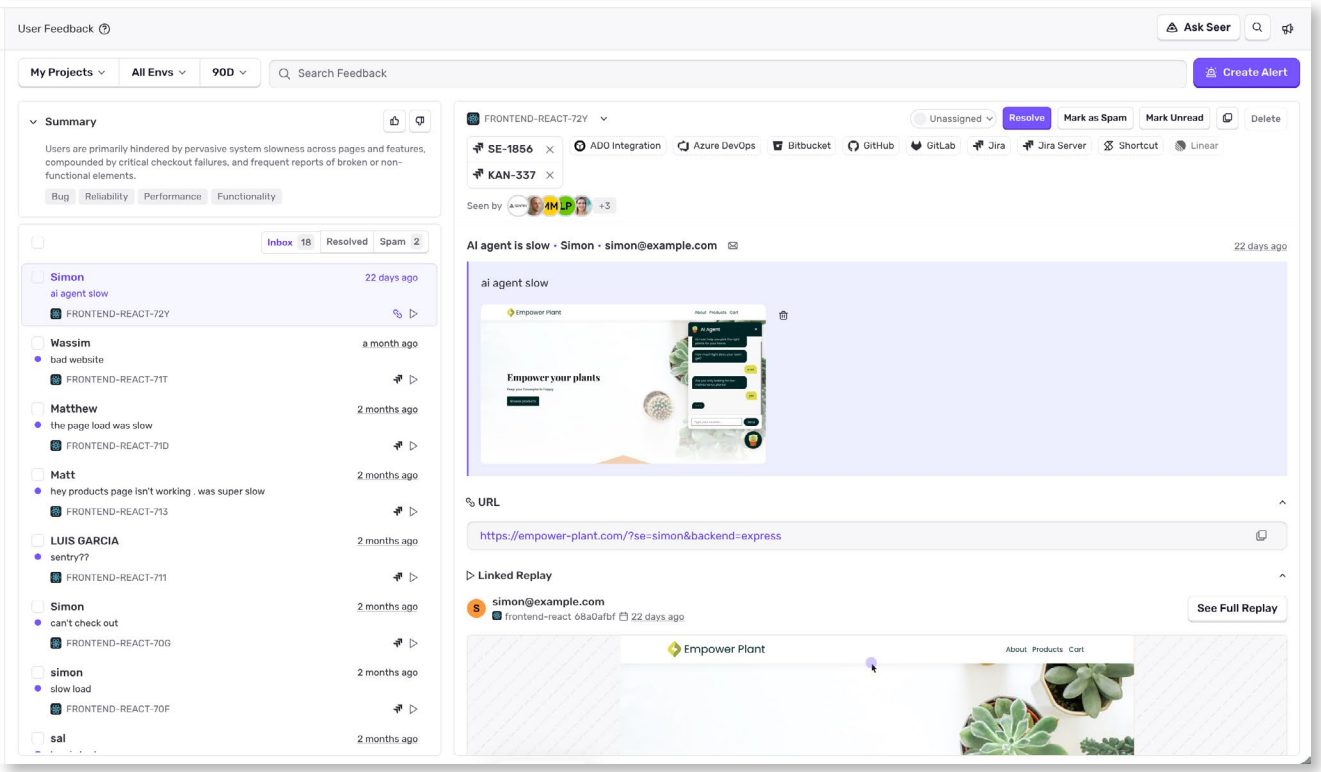
metric event is trace-connected, a spike in failed checkouts isn't just a number on a chart. Click into it and you're looking at the exact trace, spans, and logs behind it.

- **Cron Monitor on scheduled jobs:** Inventory sync, abandoned-cart emails, and order-fulfillment batch jobs all tend to run unattended. A missed window, long run, or failure becomes an issue instead of a stalled order you find out about later.

Alerts get the right fires to the right people. Monitors make sure you're watching more than exceptions, so nothing burning revenue slips through.

Understand the user experience with session replays and user feedback

Most customers will just leave your site before telling you when something breaks. Sentry's Session Replay shows you a video-like playback of user interactions on your site, in relation to network requests, DOM events, and console messages. By combining Session Replay with the User Feedback Widget, which lets you collect feedback directly from end users, you can understand critical details about errors and the UX of your site.

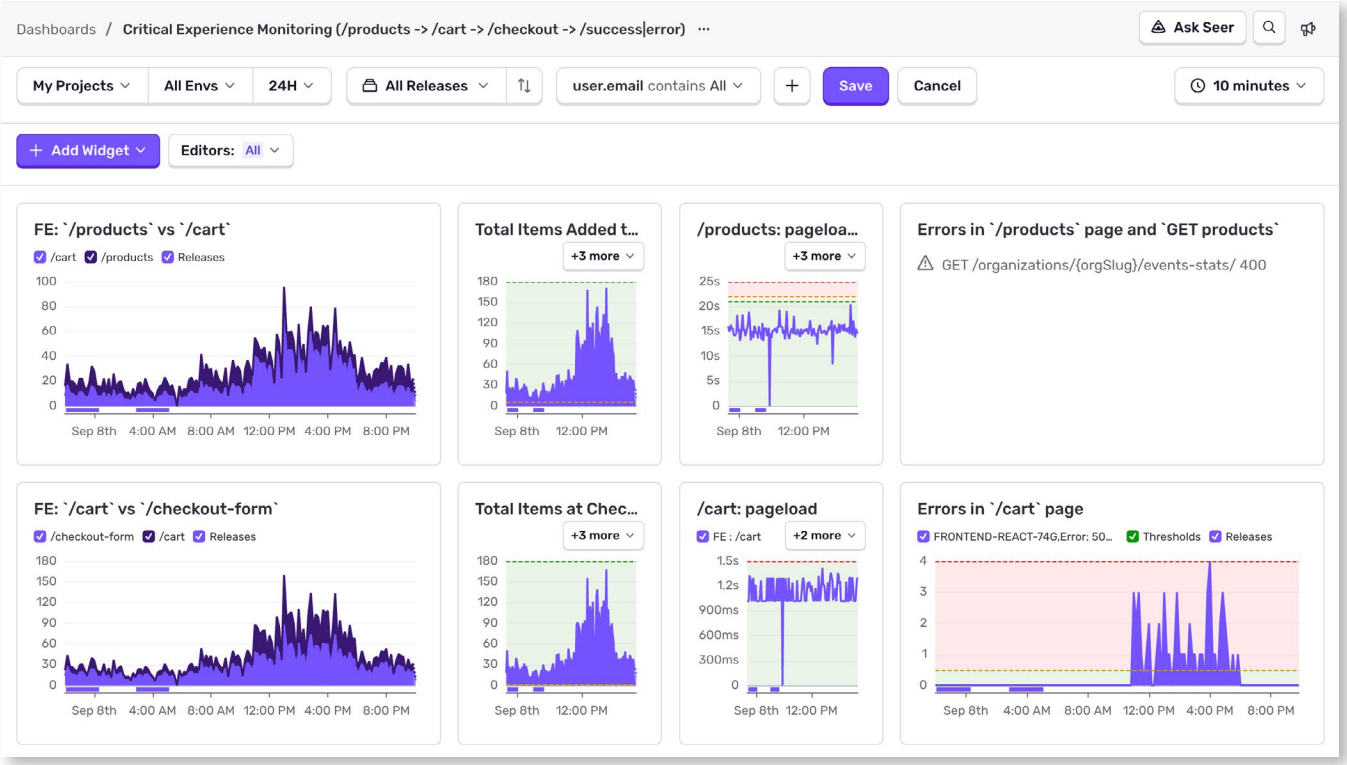


- **Use visual context to debug issues:** Replays show the exact steps that led to an issue, helping you reproduce bugs without guesswork or endless back-and-forth.
- **Connect feedback to real behavior:** User feedback submissions can be paired with replays to see what the customer experienced leading up to the submission.
- **Route feedback efficiently:** Critical problems like payment failures should go straight to engineers, while general design suggestions can wait for the next product review cycle.

Once you've fixed issues, keep feedback loops open. Replays and reports together give you a continuous view of user experience, so you can spot and resolve new frustrations before they cost you more customers.

Optimize site performance

A 100-millisecond delay can reduce conversions by 7%. During holiday traffic spikes, slow pages don't just frustrate shoppers – they push them directly to competitors. Real User Monitoring (RUM) data shows how performance issues like poor LCP and FCP directly impact abandonment rates when customers are trying to check out their cart full of limited-time product deals.



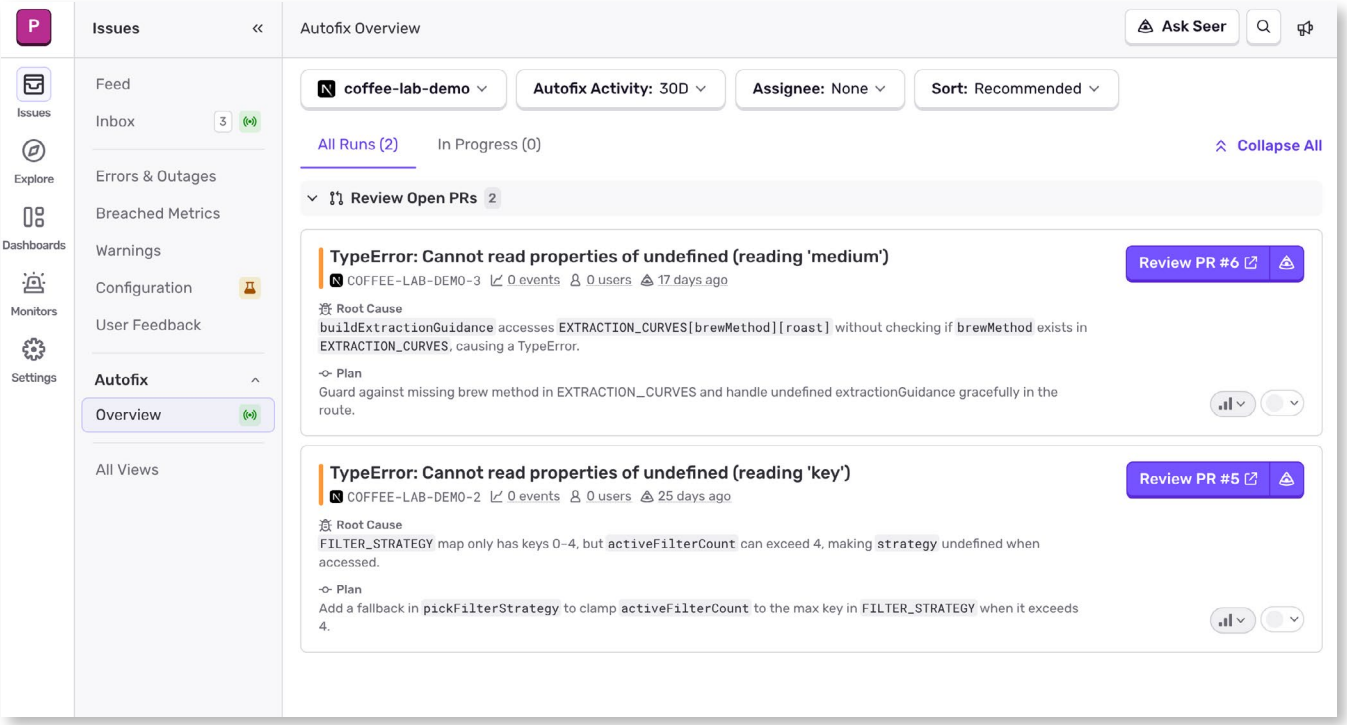
Start by capturing real-world performance data. Use **Sentry's Real User Monitoring** to track how actual shoppers experience your site during peak conditions, and head to **Insights -> Frontend** in Sentry to surface any metrics that fall outside recommended thresholds.

- Review **Time to First Byte (TTFB)**: Drill into transactions causing slow initial responses by drilling into high-TTFB spans. Surface specific requests like database queries or third-party calls that delay the start of page rendering, so you can prioritize optimizations that actually improve user-perceived performance.
- Pinpoint bottlenecks using **distributed tracing**: Look at API performance, slow database queries, or problematic third-party scripts that are likely to experience degraded performance with higher traffic.
- Use **trace-connected logs** for debugging and optimization: Application and client and server logs are your ground truth during traffic spikes. Stream structured logs in real time (live-tailing) to spot anomalies, set up alerts on specific log patterns (e.g. repeated payment.failed events with payment.failure.reason_code=card_declined), and visualize trends over time with custom **dashboards**. See our blog, **How to structure a log**, for some of our top tips
- Alert on log patterns, not just errors: Query your logs for a pattern that signals trouble, like repeated payment.status:failed events, or a spike in inventory.sync.error – and turn that query into a **Metric Monitor** with a fixed or anomaly-based threshold. A breach becomes an issue automatically, so you catch a payment processor degrading before support tickets pile up.
- Optimize asset loading: Cache static assets (images, JS, CSS) for faster repeat visits, and utilize lazy loading to reduce initial load times.

Once fixes are in place, validate by setting up **metric alerts** to compare before and after state.

Let Sentry fix issues for you

Holidays are stressful enough without your app adding to it. Traffic spikes turn small bugs into full-blown outages fast, right when you can least afford the downtime. Seer catches what's breaking and tells you why, so you're not the one debugging checkout at 2am on Black Friday.



- Use Seer agent to investigate issues: Prioritize issues based on severity, identify the person most closely aligned with the broken code, or even ask the agent to take a pass at a fix for you. Talk in Sentry's UI or on Slack to easily share status updates and proposed fixes with your team.
- Create a plan with Seer Root Cause Analysis: Seer uses Sentry errors, metrics, logs, and session replays as references against your code to diagnose exactly what's going wrong, where, and why. It identifies the issue, provides steps to reproduce it, and can generate a plan on how to fix it.
- Generate PRs with Autofix and agent handoff: Instead of devising a fix yourself, just merge a pull request. Use Seer to create a PR for you directly or through a third-party coding agent. From the planning stages through PR generation, you control how involved you and the team are.

With Seer, you've created a self-healing workflow that ensures your applications stay up and running no matter how heavy the holiday traffic.

Ready before the rush

The tools and practices in this guide give you the visibility you need when everything's on fire and customers are trying to spend money on your site. Set this stuff up during normal traffic when you have time to test and fix issues. Trying to implement monitoring during Black Friday or a half-off sale is like trying to install smoke detectors while your house is burning down.

When problems inevitably happen during peak shopping, you'll know exactly what's broken and how to fix it before your revenue disappears. And if it's too late for this season… there's always next year.