# Detect Errors in your Feature Rollouts with Sentry and Split

Sentry and Split bring your feature and application error data together.

Sentry monitors errors from front-end and back-end environments. Split manages your features with flags so you can gradually roll out new code, and quickly turn it off if things so south.

Split and Sentry integrate to detect errors caused by feature releases. Split will alert you to elevated error rates and give you the ability to instantly kill the feature.
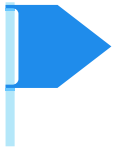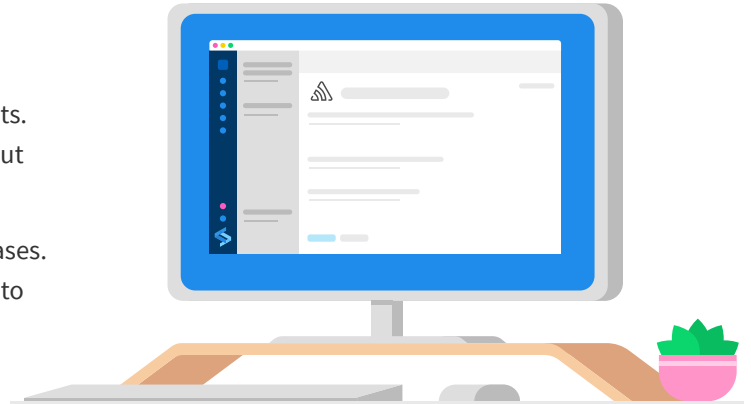
Separate Deploy from Release with Feature Flags

Split's Feature Management is a powerful platform to adopt feature flags across teams. Feature flags make it possible to deploy rapidly, while releasing features later when they can be rolled out gradually to your customers.

Detect Issues Fast with Sentry Exception Data

Split's Feature Monitoring ingests exceptions from your Projects in Sentry with a quick-to-setup integration. Split analyzes Sentry exception data in combination with your feature flags to detect and alert you when a new feature goes bad.

Accelerate Continuous Delivery across Teams

Speed up your development cadence with feature flags and detailed exception data from Sentry. Deploy as fast as you can develop, reduce the risk of each release, and make on-call shifts as easy to handle as the flip of a kill switch.

Make your Releases a Non-Event
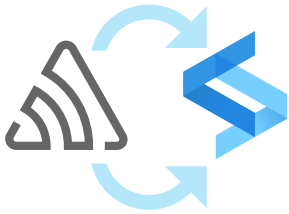
Capture exceptions from Sentry

Route frontend and backend errors from Sentry into Split with an out-of-the-box integration. Easily create metrics - for example, total exceptions or exception rate per user - to match your service level objectives (SLOs) or other engineering quality goals.

Monitor errors caused by new features

Split statistically analyzes every feature you roll out with all of your Sentry exception data to automatically detect issues. Split can identify issues even when you release many features at once or when you roll out new functionality to a subset of users.

Alert the correct team

Every feature flag is associated to the team members who are responsible. That means that errors caused by a new feature can be correctly routed to the correct team, no matter where they happen across your stack.