



MENU >

# No tricks, just Application Metrics.

Instrument once, slice any way you need, trace any spike to the fix.

GET STARTED WITH SENTRY

TRY APPLICATION METRICS



## Code **breaks,** fix it faster

Application monitoring software considered "not bad" by millions of developers.

GET STARTED

SEE HOW IN SANDBOX



GitHub

Disney

ATLASSIAN

Developer first.

# Always.



```
npm install @sentry/browser
```

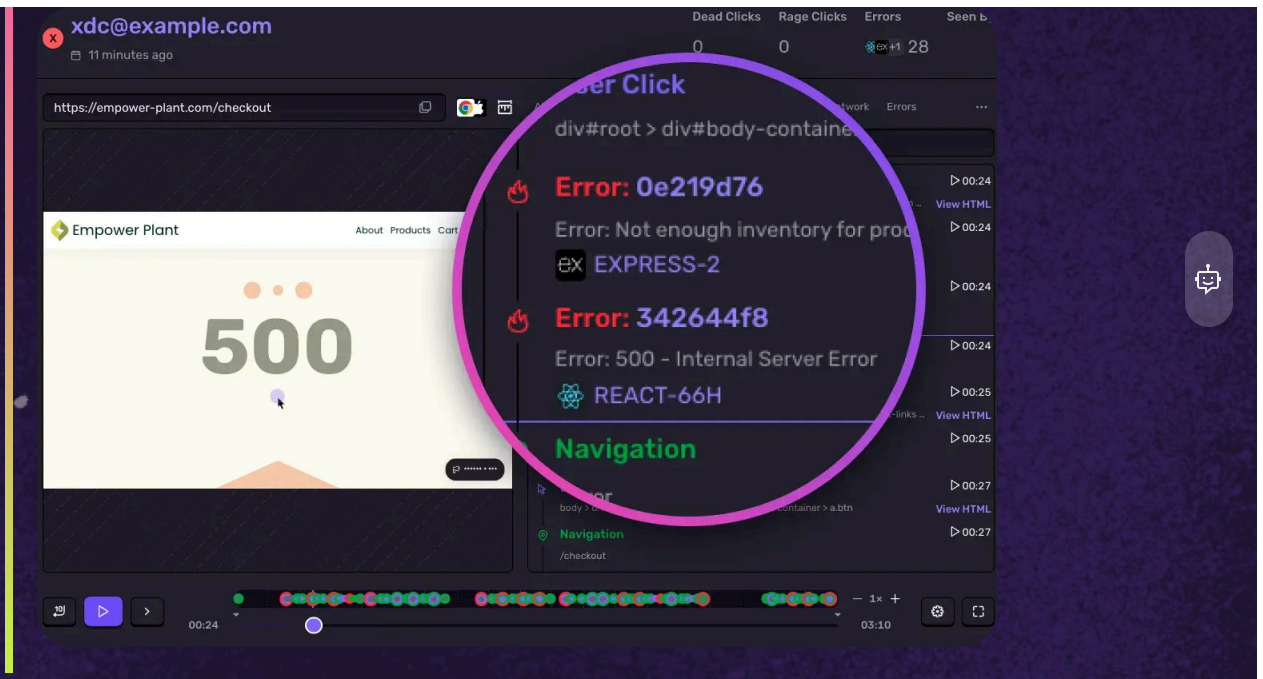
## Monitor in five lines

Drop in the SDK. No agents to install. No performance surprises.

## Everything's connected

Yeah, other tools exist. But errors, logs, replays, spans, profiles, and metrics – all connected by the same trace? That's kind of our thing.

Debug 500's, trace slow requests, replay `fetch()` failures, and fix the broken code that caused it.



Catch slow queries, N+1s, and request timeouts before the 'why is this so slow?' posts fill up your feed.

Map every incident to the release, PR, and owner -- automatically.

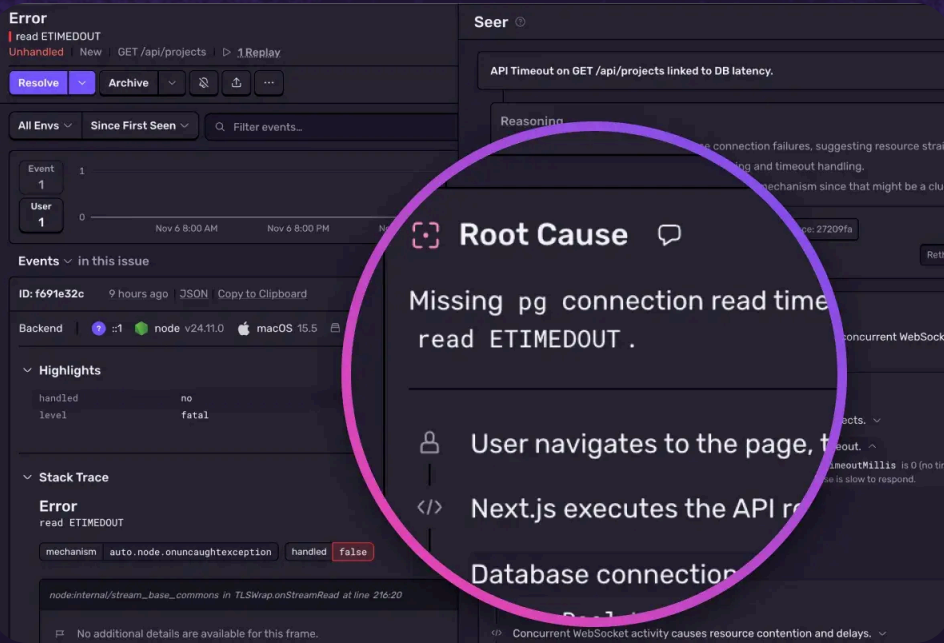
## Debugging needs context— with or without AI

Seer, our AI debugger, uses Sentry context – logs, commits, traces, stack

# trace - so you can stop guessing and *it* can fix issues for you.



Analyzes every signal to explain why your code failed, not just where.



Fixes what's broken while you ship what's next - generating precise, merge-ready patches.

Stops bad code before it starts bad days.  
Correlating PRs against real error and performance history to catch regressions before they ship.

# Loved by developers worldwide



*We wouldn't have scaled without Sentry. Most of our incidents are hardware-related—and we debug them all inside Sentry*



**Nova DasSarma**  
Systems Lead, Anthropic



*Sentry's high-quality tooling helps Disney+ maintain high-quality service to its tens of millions of global subscribers.*



**Andrew Hay**  
Director at Disney Streaming Services, Disney+



*The signal we get from Sentry is the most reliable indicator of software issues and is used throughout Instacart because it can be easily configured for each service regardless of the language or framework.*



**Igor Dobrovitski**  
Infrastructure Software Engineer, Instacart



# Get started in minutes

Five lines of code. That's it. No complex setup,  
no performance hits, no waiting around.

Next.js ▾

See -- it's really just one command.

```
npx @sentry/wizard@latest -i nextjs
```

Big thanks to Sentry for sponsoring this content. [View all sponsors](#)

```
pip install --upgrade sentry-sdk
```

## Configure your DSN:

```
import sentry_sdk

sentry_sdk.init(
    "https://<key>@sentry.io/<project>",

    # Set traces_sample_rate to 1.0 to capture 100%
    # of transactions for Tracing.
    # We recommend adjusting this value in producti
    enable_tracing=True,
    traces_sample_rate=1.0,
)
```



Grab the [Sentry Node SDK](#):

```
npm install @sentry/node
```



Configure your SDK:



```
const Sentry = require('@sentry/node');  
Sentry.init({ dsn: 'https://<key>@sentry.io/<projec
```



Grab the [Sentry React SDK](#):



```
npm install @sentry/react
```

We recommend putting the Sentry initialization code into its own file and including that file as the first import in your application entry point as shown in the example below:



```
import { useEffect } from "react";  
import * as Sentry from "@sentry/react";  
  
Sentry.init({  
  dsn: "https://examplePublicKey@o0.ingest.sentry.i  
  integrations: [  
  ],  
  // Set `tracePropagationTargets` to control for w  
  tracePropagationTargets: [/^\/\//, /^https:\/\/your  
});
```

Include the Sentry initialization file as the first import statement:

```
// Sentry initialization should be imported first!  
import "./instrument";  
import App from "./App";  
import { createRoot } from "react-dom/client";  
  
const container = document.getElementById("app");  
const root = createRoot(container);  
root.render(<App />);
```

Install the **NuGet** package to add the Sentry dependency:

```
dotnet add package Sentry
```

Initialize the SDK as early as possible, like in the Main method in Program.cs/Program.fs:

```
using (SentrySdk.Init(o => {  
    // Tells which project in Sentry to send events to  
    o.Dsn = "https://<key>@sentry.io/<project>";  
    // When configuring for the first time, to see what's going on  
    o.Debug = true;  
    // Set TracesSampleRate to 1.0 to capture 100% of transactions  
    // We recommend adjusting this value in production  
    o.TracesSampleRate = 1.0; }))  
{
```

```
// App code goes here - Disposing will flush even  
}
```

Grab the [Sentry Go SDK](#):



```
go get "github.com/getsentry/sentry-go"
```



Configuration should happen as early as possible in your application's lifecycle:

```
package main  
  
import (  
    "log"  
    "time"  
  
    "github.com/getsentry/sentry-go"  
)  
  
func main() {  
    err := sentry.Init(sentry.ClientOptions{  
        Dsn: "https://<key>@sentry.io/<project>",  
        EnableTracing: true,  
        // Specify a fixed sample rate:  
        // We recommend adjusting this value in pro  
        TracesSampleRate: 1.0,  
    })  
    if err != nil {  
        log.Println("Sentry SDK initialization failed: ", err)  
    }  
}
```



```
// Or provide a custom sample rate:
TracesSampler: sentry.TracesSampler(func(ct
    // As an example, this does not send so
    // transactions to Sentry based on their
    if ctx.Span.Name == "GET /health" {
        return 0.0
    }

    return 1.0
}),
})
if err != nil {
    log.Fatalf("sentry.Init: %s", err)
}
// Flush buffered events before the program term
// Set the timeout to the maximum duration the
defer sentry.Flush(2 * time.Second)
}
```

To integrate Sentry into your Xcode project, specify it in your Podfile, then run `pod install`:

```
platform :ios, '9.0'
use_frameworks! # This is important

target 'YourApp' do
```

```
pod 'Sentry', :git => 'https://github.com/getsent  
end
```

Initialize the SDK as soon as possible in your application lifecycle,  
such as in your AppDelegate

application:didFinishLaunchingWithOptions method:

```
import Sentry // Make sure you import Sentry

func application(_ application: UIApplication,
                 didFinishLaunchingWithOptions launchOptions: [U

    SentrySDK.start { options in
        options.dsn = "https://<key>@sentry.io/<pro
        options.debug = true // Enabled debug when
        // Example uniform sample rate: capture 100
        options.tracesSampleRate = 1.0
    }

    return true
}
```

Add the `sentry-ruby` gem to your Gemfile:

```
gem "sentry-ruby"
```

## Configure your DSN:

```
Sentry.init do |config|
  config.dsn = 'https://<key>@sentry.io/<project>'

  # Set a uniform sample rate between 0.0 and 1.0
  # We recommend adjusting the value in production:
  config.traces_sample_rate = 1.0

  # or control sampling dynamically
  config.traces_sampler = lambda do |sampling_context|
    # sampling_context[:transaction_context] contains the transaction context
    # sampling_context[:parent_sampled] contains the parent's sampling decision
    true # return value can be a boolean or a float
  end
end
```

Install the `sentry/sentry` package with Composer:

```
composer require sentry/sentry
```

To capture all errors, even the one during the startup of your application, you should initialize the Sentry PHP SDK as soon as possible.

```
\Sentry\init(['dsn' => 'https://<key>@sentry.io/<pr  
    // Specify a fixed sample rate:  
    'traces_sample_rate' => 0.2,  
    // Or provide a custom sampler:  
    'traces_sampler' => function (SentryTracingSamp  
        // return a number between 0 and 1  
    }, 1]);
```

Install the `sentry/sentry-laravel` package with Composer:

```
composer require sentry/sentry-laravel
```

Add Sentry reporting to `bootstrap/app.php`:

```
<?php  
  
use Illuminate\Foundation\Application;  
use Illuminate\Foundation\Configuration\Exceptions;  
use Illuminate\Foundation\Configuration\Middleware;  
use Sentry\Laravel\Integration;  
  
return Application::configure(basePath: dirname(__D  
    ->withRouting(  
        web: __DIR__.'/../routes/web.php',  
        commands: __DIR__.'/../routes/console.php',
```

```
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware)
        //
    })
    ->withExceptions(function (Exceptions $exceptions)
        Integration::handles($exceptions);
    )->create();
```

Enable Sentry Tracing in config/sentry.php:

```
// Specify a fixed sample rate:
'traces_sample_rate' => 0.2,
// Or provide a custom sampler:
'traces_sampler' => function (SentryTracingSampling
    // return a number between 0 and 1
),
```

Run this Artisan command to configure the Sentry DSN:

```
php artisan sentry:publish --dsn=<paste-your-DSN-here>
```

Add the Sentry dependency:



```
dotnet add package Sentry.AspNetCore
```

Configure Sentry in `appsettings.json`.



```
"Sentry": {  
  "Dsn": "https://examplePublicKey@o0.ingest.sentry  
  "Debug": true,  
},
```

Then add the SDK by simply calling `UseSentry`:



```
public static IHostBuilder CreateHostBuilder(string  
    Host.CreateDefaultBuilder(args)  
        .ConfigureWebHostDefaults(webBuilder =>  
            {  
                // Add the following line:  
                webBuilder.UseSentry();  
            });
```

Grab the [Sentry Java SDK](#):



```
<dependency>  
  <groupId>io.sentry</groupId>
```

```
<artifactId>sentry-spring-boot-starter</artifactId>
<version><VERSION></version>
</dependency>
```



Configure your DSN in `application.properties`:

```
sentry.dsn=https://<key>@sentry.io/<project>
# Set traces_sample_rate to 1.0 to capture 100%
# of transactions for performance monitoring.
# We recommend adjusting this value in production.
sentry.traces-sample-rate=1.0
```



Grab the [Sentry Vue SDK](#):

```
npm install @sentry/vue
```



Configure your DSN:

```
import { createApp } from "vue";
import * as Sentry from "@sentry/vue";

const app = createApp({
  // ...
});
```



```
Sentry.init({
  app,
  dsn: "https://<key>@sentry.io/<project>",
  // This enables automatic instrumentation (highly
  // but is not necessary for purely manual usage
  // If you only want to use custom instrumentation
  // * Remove the BrowserTracing integration
  // * add Sentry.addTracingExtensions() above your
  integrations: [Sentry.browserTracingIntegration()

  // We recommend adjusting this value in productio
  // for finer control
  tracesSampleRate: 1.0,
  // Set tracePropagationTargets to control for whi
  tracePropagationTargets: ['localhost', /^https://
});

app.mount("#app");
```

To use the SDK, initialize Sentry in your Solid entry point `index.jsx` before you render your Solid app:

```
// index.jsx / index.tsx

import * as Sentry from "@sentry/solid";
import { useBeforeLeave, useLocation } from "@solid
import { render } from "solid-js/web";
```

```
import App from "./app";

// Initialize the Sentry SDK here
Sentry.init({
  dsn: "__DSN__",
  integrations: [Sentry.browserTracingIntegration()

  // Performance Monitoring
  tracesSampleRate: 1.0, // Capture 100% of the tr
  // Set 'tracePropagationTargets' to control for w
  tracePropagationTargets: ["localhost", /^https:\\/

});

const app = document.getElementById("app");

if (!app) throw new Error("No #app element found in

render(() => <App />, app)
```

To use the SDK, initialize Sentry in your Svelte entry point main.js before you bootstrap your Svelte app:

```
// main.js / main.ts

import App from "./App.svelte";

import * as Sentry from "@sentry/svelte";
import { BrowserTracing } from "@sentry/tracing";
```

```
// Initialize the Sentry SDK here
Sentry.init({
  dsn: "__DSN__",
  release: "my-project-name@2.3.12",
  integrations: [new BrowserTracing()],

  // This enables automatic instrumentation (highly
  // but is not necessary for purely manual usage
  // If you only want to use custom instrumentation
  // * Remove the BrowserTracing integration
  // * add Sentry.addTracingExtensions() above your
  integrations: [Sentry.browserTracingIntegration()

  // We recommend adjusting this value in productio
  // for finer control
  tracesSampleRate: 1.0,

  // Set tracePropagationTargets to control for whi
  tracePropagationTargets: ['localhost', /^https://
});

// Then bootstrap your Svelte app
const app = new App({
  target: document.getElementById("app"),
});

export default app;
```



Just run this command to install and register Sentry's Astro integration.

```
npx astro add @sentry/astro
```

And add your DSN and project config to your `astro.config.mjs` file:

```
import { defineConfig } from "astro/config";
import sentry from "@sentry/astro";

export default defineConfig({
  integrations: [
    sentry({
      dsn: "__DSN__",
      sourceMapsUploadOptions: {
        project: "your-project-slug",
        authToken: process.env.SENTRY_AUTH_TOKEN,
      },
      tracesSampleRate: 1.0,
    }),
  ],
});
```

Grab the [Sentry JavaScript SDK](#):



```
<script src="https://browser.sentry-cdn.com/<VERSIO
```

Configure your DSN:



```
Sentry.init({ dsn: 'https://<key>@sentry.io/<projec
// This enables automatic instrumentation (highly
// but is not necessary for purely manual usage
// If you only want to use custom instrumentation
// * Remove the BrowserTracing integration
// * add Sentry.addTracingExtensions() above your
integrations: [Sentry.browserTracingIntegration()

// We recommend adjusting this value in productio
// for finer control
tracesSampleRate: 1.0,

// Set tracePropagationTargets to control for whi
tracePropagationTargets: ['localhost', /^https://
});
```



# Built to be secure, Designed to not get in your way

Security and compliance aren't just checkboxes—they're built into how we run Sentry. We use industry-standard tech and practices to keep your data safe, and we stay out of your way while doing it.



**SOC 2  
TYPE II  
CERTIFIED**



[CHECK OUT OUR PRIVACY POLICY](#)

[CONTACT US](#)

## Get monthly product updates

# from our newsletter

YOUR EMAIL:

- I want to receive the monthly newsletter and other updates from Sentry. You may unsubscribe at any time.

*By filling out this form, you agree to our [privacy policy](#). This form is protected by reCAPTCHA and Google's [Privacy Policy](#) and [Terms of Service](#) apply.*

SIGN UP



## Fix It

Get started with the only application monitoring platform that empowers developers to fix application problems without compromising on velocity.

TRY SENTRY FOR FREE

GET A DEMO

## Company

ABOUT

BLOG

CAREERS

CONTACT US

TRUST

## Platform

ERROR MONITORING

LOGS

SEER

SESSION REPLAY

METRICS

TRACING

UPTIME MONITORING

PROFILING

CRON MONITORING

INTEGRATIONS

## Solutions

WEB / FULL STACK DEVELOPMENT

MOBILE CRASH REPORTING



GAME CRASH REPORTING

AI OBSERVABILITY

APPLICATION PERFORMANCE MONITORING

APPLICATION OBSERVABILITY

REAL USER MONITORING

ECOMMERCE

ENTERPRISE

STARTUPS

**Get Help**

DOCS

HELP CENTER

STATUS

DEV RESOURCES



[TERMS](#)   [SECURITY & COMPLIANCE](#)   [PRIVACY](#)



© 2026 • Sentry is a registered Trademark of Functional Software, Inc.