April 10, 2025

Rspack joins the Next.js ecosystem

One of Rspack's primary goals is to seamlessly integrate with webpack-based projects or frameworks, providing an enhanced development experience with minimal migration costs.

In the JavaScript ecosystem, Next.js has one of the most advanced build systems, with deeply customized webpack configurations and a rich plugin ecosystem. This made it an ideal candidate for testing and proving Rspack's compatibility and robustness. Integrating Rspack into Next.js demonstrates Rspack's applicability in complex projects and provides Next.js users with an alternative solution to improve developer experience.

Rspack comes to Next.js

Today, we're excited to introduce next-rspack, a community-driven plugin bringing direct Rspack support to Next.js. This integration offers a fast, webpack-compatible alternative for teams not yet ready to adopt Turbopack.

> Get started today by visiting our documentation or checking out the official Next.js example.

Before landing this support, we explored integration possibilities by creating rsnext, a fork of Next.js designed to prototype a near drop-in solution. This early fork played a valuable role in validating feasibility and discovering edge cases. Also, it made us realize

that while Rspack's high compatibility with webpack gave us a head start, achieving stable integration still required significant effort and collaboration.

Partnering with Vercel on shared foundations

The launch of next-rspack is just one aspect of our broader collaboration with Vercel. This partnership extends beyond Next.js integration, as both teams focus on improving shared foundational technologies such as SWC and Lightning CSS—widely adopted tools across the JavaScript ecosystem.

By working together to enhance these core components, we're laying a stronger foundation for better developer experience, performance, and reliability. These efforts benefit not only Rspack and Next.js, but also help uplift the broader JavaScript ecosystem. A rising tide lifts all boats.

To ensure ongoing reliability, next-rspack is now integrated into Next.js's continuous integration pipeline, proactively catching regressions and maintaining compatibility. Although still experimental, it currently passes around 96% of integration tests, giving us the confidence to publicly release this integration. You can track the latest status with arewerspackyet and following our twitter for latest progress about next-rspack.

For teams not yet ready to adopt Turbopack, next-rspack offers a solid, fast alternative with excellent compatibility and straightforward setup.

We deeply appreciate Vercel's collaboration and shared commitment to improving the tools that developers rely on every day. We'll continue working together to refine this integration and support the future of modern JavaScript development.

Performance insights

App router users

Currently, the App Router implementation with `next-rspack` is **slower than Turbopack**, and may even be slower than webpack. This is due to some **JavaScript plugins** that cause heavy Rust-JavaScript communication overhead.

We have **experimentally ported the theses plugins to Rust**, which dramatically improved performance—almost on par with Turbopack. And we are exploring how to address the long-term maintenance challenges that come with deep integration.

Page router users

The situation is much more promising:

- **Development Mode**: 2x faster than webpack
- **Production Mode**: 1.5x faster than webpack

Users deeply integrated into the webpack ecosystem will find migration easier.

There're some known bottlenecks which limit further performance improvement(like huge Rust-JavaScript communication overhead, slower output file tracing implementation) which can be solved in the future, with expected improvements, we foresee:

- **5x faster builds and HMR in development**
- **3x faster production builds**

Frequently asked questions

How will it remain supported?

next-rspack is already integrated into Next.js's CI pipeline, helping us catch regressions early and keep compatibility high. Support will evolve alongside both Next.js and Rspack, with ongoing collaboration between both teams and the open source community.

Who maintains it?

next-rspack is a community plugin, but its development and integration rely on close collaboration between the Rspack and Vercel teams to ensure continued support and progress.

Does this have any impact on Turbopack? is Vercel adopting Rspack?

Rspack doesn't replace Turbopack. It's an alternative solution for those with extensive webpack configurations who are not ready to migrate to Turbopack.

What are known issues?

As of now, next-rspack passes around 96% of integration tests, and progress can be monitored on arewerspackyet.

- Some edge cases and advanced features may still require workarounds or additional support. Let us know how it went in the feedback discussion, even when you don't run into problems.
- Due to the current plugin implementation, the performance of the App Router is still suboptimal and there's still plenty of room for performance improvement.
- Since Rspack is not 100% compatible with webpack's API, some of your webpack plugins may not work smoothly on Rspack. Let us know if you have compatibility problems.

How can you help?

Try out next-rspack, report issues, contribute code or docs, and join the community discussions. Your feedback and contribution is valuable.

Future plans

- **Increase Test Coverage**: We aim to raise our test coverage from the current 96% to nearly 100% in the next quarter.
- **Enhance Performance**: We'll explore deeper integration with Next.js through native plugins to improve build performance.
- **Iterate Based on User Feedback**: Continue supporting more community plugins from the Next.js ecosystem.
- **Improve Integration Workflow**: Establish a more robust CI/CD pipeline between Rspack and Next.js to ensure the stability and reliability of next-rspack support.
- **Better RSC Support**: Turbopack's unified module graph unlocks faster and simpler RSC implementation. Rspack will deliver a similar API to bring first-class, high-performance RSC support to the ecosystem.
- **Module Federation Support**: We are discussing with Next.js team about improved support of Module Federation.

Through 2024, stability and artifact integrity were a primary focus for Rspack. In 2025 we are focusing more on speed opportunities and broad ecosystem.

Stay tuned—we're just getting started.

📝 Edit this page on GitHub

Previous Page
Overview

Next Page
Announcing Rspack 1.3