BUILD TIMES

A WEB DEVELOPMENT PERIODICAL BY EDUARDO BOUÇAS

YOU SHOULD KN
BEFORE CHOOSIN

MARCH 25TH, 2025

Picking the technology stack for a project is an important and consequential decision. In the enterprise space in particular, it often involves a multi-year commitment with long-lasting implications on the roadmap of the project, the pace of its development, the quality of the deliverables, and even the ability to assemble and maintain a happy team.

The open-source software model is a fundamental answer to this. By using software that is developed in the open, anyone is free to extend it or modify it in whatever way fits their use case. More crucially, the portability of open-source software gives developers and organisations

the freedom to move their infrastructure between different providers without fear of getting locked in to a specific vendor.

This is the expectation with Next.js, an open-source web development framework created and governed by Vercel, a cloud provider that offers managed hosting of Next.js as a service.

There is nothing wrong with a company profiting from an open-source software it created, especially when that helps fund the development of the project. In fact, there are plenty of examples of that model working successfully in our industry.

But I think that can only work sustainably if the boundaries between the company and the open-source project are abundantly clear, with well-defined expectations between the maintainers, the hosting providers and the users about how and where each feature of the framework can be used.

I want to explain why I don't think this transparency exists today.

My goal is not to stop anyone from using Next.js, but to lay out as much information as possible so developers and businesses can make an informed decision about their technology stack.

Declaration of interest

Let me lead with a declaration of interest:

- I work at Netlify and have done so for over four years

- Netlify is a frontend cloud platform that supports Next.js and other web frameworks as part of its product offering
- Netlify and Vercel are direct competitors

It's important for me to establish this for a few reasons.

My job involves building the infrastructure and tooling needed to support the full feature set of Next.js on Netlify, which has exposed me to the internals of the framework in a way that most people won't see. Over the years, I have seen concerning patterns of tight coupling between the open-source framework and the infrastructure of the company that builds it.

My employment is also the reason why I have always been very wary of voicing these concerns in public. As a Netlify employee, I don't really get to voice an objective concern about Next.js without people dismissing my claims as Netlify unleashing one of its minions to spread FUD about a competitor.

I'm not keen on exposing myself and the company to that type of debate, so I have always chosen to work behind the scenes in supporting the developers who decide to deploy their sites on Netlify and shield them from all the complexity that goes into making that possible.

But then something happened.

Last weekend, Vercel disclosed a critical security vulnerability with Next.js. This type of issue is normal, but the way Vercel chose to handle

it was so poor, reckless and disrespectful to the community that it has exacerbated my concerns about the governance of the project.

For me, things change once your decisions put other people at risk, so I felt the urge to speak up.

Openness and governance

I'll come back to this incident later, but before that I want to back up a little and give you a peek behind the curtain. My history of reservations about the openness and governance of Next.js stem from a series of decisions made by Vercel over the years that make it incredibly challenging for other providers to support the full feature set of the framework.

I'll cover these by laying out a series of facts about how Next.js is built. I'll then add some of my own considerations about how those facts live up to the expectations of an open, interoperable, enterprise-grade software product.

Fact #1: No adapters

Most modern web development frameworks use the concept of adapters to configure the output of the framework to a specific deployment target: Remix, Astro, Nuxt, SvelteKit and Gatsby are just a few examples. This pattern allows developers to keep the core of their applications untouched, and simply swap the adapter if they decide to start deploying to a different provider.

These adapters can be maintained by framework authors, by the hosting providers, by the community, or all of the above. Frameworks are typically structured in such a way that it's possible for anyone to build their own adapter in case one isn't available for the provider of their choice.

Next.js does not have the concept of adapters and they have stated in the past that they would not support them. The output of a Next.js build has a proprietary and undocumented format that is used in Vercel deployments to provision the infrastructure needed to power the application.

Vercel's alternative to this was the Build Output API, a documented specification for the output format of frameworks who wish to deploy to Vercel.

This is not an adapter interface for Next.js, and in fact has nothing to do with Next.js. The announcement blog post said that Next.js supports this format, but as of today that isn't true.

In November 2023, the Next.js documentation has been updated to say that Next.js would adopt the Build Output API in the following major version of the framework (which would be version 15):

> *Next.js produces a standard deployment output used by managed and self-hosted Next.js. This ensures all features are supported across both methods of deployment. In the next major*

version, we will be transforming this output into our Build Output API specification.

Next.js 15.0.0 was released in October 2024 without support for the Build Output API.

Vercel have built the Build Output API because they wanted their customers to leverage the rich ecosystem of frameworks in the space, but their own framework doesn't support it to this day.

This means that any hosting providers other than Vercel must build on top of undocumented APIs that can introduce unannounced breaking changes in minor or patch releases. (And they have.)

Late last year, Cloudflare and Netlify have joined OpenNext, a movement of different cloud providers that collaborate on open-source adapters for Next.js. Shortly after, Vercel have engaged with the movement and committed to building support for adapters. They haven't made any timeline commitments, but have recently said they are actively working on it.

It's important to remember that it's been almost three years since the launch of the Build Output API, and to this day the framework still isn't portable. I'm cautiously optimistic about that actually changing this time.

Fact #2: No official serverless support

The official methods for self-hosting Next.js require running the application in a stateful way, as long-running servers. While technically possible, this is very hard to operate in any real-world production environment where a single instance isn’t sufficient.

The setup needs to be able to dynamically scale up very quickly in order to handle sudden bursts of traffic, while at the same time being able to scale down to zero in order to be cost-effective. This last part is essential when working with server components, for example, where the deep tangling between client and server code can break older clients unless every version of the server code ever deployed is available indefinitely.

One obvious answer to these requirements is serverless computing, as attested by official Next.js documentation that confirms the benefits of this model:

> *Serverless allows for distributed points of failure, infinite scalability, and is incredibly affordable with a "pay for what you use" model.*

This clearly advantageous computing paradigm is precisely how Vercel has run Next.js sites in their own infrastructure for years. Given that Next.js is an open framework, it is reasonable to expect that you'd be able to use that same model in any serverless provider of your choice. But it's not that simple.

Next.js once had a serverless mode that you could enable with a configuration property, but it was removed without further explanation in October 2022. No equivalent mode was ever introduced.

The official React documentation, which the Next.js team help maintain, says that Next.js can be deployed to *«any serverless hosting»*, but there is no official documentation whatsoever for this.

This means that any providers who want to offer support for Next.js with the same computing model that the framework itself promotes must reverse-engineer their way to a custom implementation.

Fact #3: Vercel-specific code paths

Next.js has code paths that are only ever executed for sites deployed to Vercel. An example of this is a private flag called *minimal mode*, which allows Vercel to shift work away from the framework and run it on their edge infrastructure.

Here's an example of why that matters. Next 12 introduced middleware, a way to address use cases such as feature flags, A/B tests and advanced routing. What's common in all of these use cases is the need to run logic on the hot path, behind the cache, with very low latency.

The announcement included this:

> *This works out of the box using `next start`, as well as on Edge platforms like Vercel, which use Edge Middleware.*

In practice, this means that you have two options: use `next start` and run middleware alongside the rest of your application in your *origin* server (which is typically running in a single region, after the cache), or use one of the «*Edge platforms like Vercel*» to run middleware at the edge, before the cache, unlocking all the incredible use cases that Vercel boasted in the resources linked in the announcement.

The phrase «*Edge platforms like Vercel*» surely means that there are many alternatives out there because other providers were given the option to also implement middleware at the edge, right? No.

This secret minimal mode is what allowed Vercel to break out middleware from the rest of the application so they could run it at the edge, but only Vercel has access to it.

Netlify does support running middleware at the edge, but we've done it at the expense of having a full team of engineers dedicated to reverse-engineering the framework and building our own edge middleware implementation on top of undocumented APIs. This type of commitment is just impossible for smaller companies that simply do not have the resources to fight this battle, which makes most of them stop trying.

As far as I know, Netlify is the only cloud provider to support the full feature set of Next.js outside of Vercel, which doesn't make sense to me. With Next.js having such a sizeable share of the market, I would expect a lot more hosting options, which would foster competition and innovation across the board, ultimately benefitting users and the web.

So why is there a hidden door in Next.js for which only Vercel holds the key? I think it's expected that the framework maintainers regularly experiment with features before they're launched, but minimal mode isn't that. We're talking about an entirely different operation mode for the framework, which has been in the code base for many years and which unlocks capabilities that are reserved for the for-profit company that owns the framework.

If WordPress had a privileged code path that was only accessible to sites deployed to Automattic properties, would it be trusted as a truly open project and would it have the dominance it has today?

Security posture

Let's go back to the security incident. On Friday, March 21st at 10:17 AM (UTC), Vercel published a CVE for a critical security incident, ranked with a severity of 9.1 out of 10.

In essence, it was possible for anyone to completely bypass Next.js middleware by sending a specific header in the request. This is important because authentication was one of the flagship use cases of

middleware, and this exploit meant that anyone could bypass the authentication layer and gain access to protected resources.

As the incident unravelled, a few things became apparent. First of all, the vulnerability was reported to the Next.js team on February 27th, but it wasn't until March 14th that the team started looking into it. Once they did, they started pushing fixes for Next 14 and Next 15 within a couple of hours.

So by March 14th (at the latest), Vercel knew they had a serious incident on their hands. The responsible thing to do at that point would be to immediately disclose the vulnerability to other providers, so that they could assess the impact to their own customers and take any necessary actions to protect them as quickly as possible. At times like these, our duty to protect users should rise above any competition between companies.

That is not what happened. It took Vercel 8 (eight) days to reach out to Netlify. In that time, they managed to push patches to Next.js, cut two releases, and even write a blog post that framed the incident as something that Vercel's firewall had *«proactively protected»* their customers from (even though their CTO later said that their firewall had nothing to do with it).

I think it's incredibly disingenuous to spin a critical security vulnerability in your open-source project as a strength of your product, with absolutely no consideration for whether users in other providers were also affected and what they should do to mitigate. In fact, they

wouldn't even know this, because they hadn't even reached out to us at this point.

After being called out on social media, Vercel have rewritten the blog post to remove any mention of their firewall and clarify which providers had been affected and whether their customers had to take any action.

Vercel has then released a postmortem where they said — for the first time — that on March 21st they were able to «*verify Netlify and Cloudflare Workers were not impacted*». This is directly contradicted by their staff reaching out to Netlify on March 22nd offering help to «*get a patch up*». If we were not impacted, what was there to patch?

This lack of consideration for any users outside of Vercel has created unnecessary anxiety and confusion for a lot of people, leaving some providers scrambling to find a solution and then having to partially roll it back, others announcing that they were not vulnerable when in reality they were, etc.

As you read this, it's impossible for anyone to know how many sites out there are still vulnerable to this exploit, many of which would've been safe if things were handled differently.

And at the height of all this mess, Vercel's leadership had... a different focus.

But Vercel owns Next.js

They do. And they have every right to make a business out of the framework that they've put so much work, talent, time and energy into building and growing. I'm not disputing that.

But that growth holds them to a high bar of standards that, in my opinion, they have repeatedly failed to meet.

«If Vercel own Next.js, what incentive do they have to open it up to other providers?» is a question I sometimes see and which I find intriguing. What incentives does Redis have for opening up their software when they own Redis Cloud? Why make Grafana open when Grafana Cloud is owned by the same company? Or WordPress, ClickHouse and many others?

The incentive is that they *have* to do those things if they choose to publish their software as open-source and not as a closed, proprietary solution. Their success is intrinsically associated their users having the guarantee that they are free to choose whatever provider offers the service that meets their needs at any given time.

Wrapping up

It's not my business to say which framework you should use. If you like Next.js and you still think it's the best tool for the problem you need to solve, you should absolutely use it. But I hope that this information helps you feel more confident about your decision, whichever way you're leaning.

As for me, I'll keep doing my job to help support the developers who chose to deploy their sites to Netlify, whatever their framework of choice is. And competition aside, I'm genuinely looking forward to help Vercel make Next.js more open and interoperable through the OpenNext movement. ∎

Vercel's response

This is a placeholder where I will share any response I receive from Vercel to this post. I have received none. If you would like to see these issues clarified, please consider soliciting a response from Vercel by sharing this post on social media.

Update (March 26th): Added a note about Vercel's most recent postmortem and a section for Vercel's response.