netlify Jamstack

Headless Commerce: How a Headless + Jamstack Architecture Yields High Performance and Maximum Conversions

THE DEFINITIVE GUIDE

What You'll Learn:

- **Why Headless Commerce gives retailers a competitive advantage:**
    - Personalized experiences and custom design create market differentiation
    - Instant page loads results in increased engagement and conversions
    - Achieving better Core Web Vitals and performance scores translates to better SEO, discoverability and increased traffic
    - Developers can ship and iterate 10x faster, accelerating your ability to deliver new content, campaigns and experiments
- **Discover best practices for building highly performant e-commerce sites:**
    - Pre-build critical content for instant page loads and better SEO
    - Fetch dynamic content using Netlify Functions and third party APIs
    - Defer long-tail product pages and less popular content with on-demand builds
    - Use an agnostic platform to integrate the technologies you need in one seamless workflow, including e-commerce backend, SSGs, CMS and more
- **Make your business case with results from popular retailers who have successfully implemented a headless strategy:**
    - Spring (formerly Teespring) scaled to 6 million sites and more than 1 million visitors / day, after migrating from its monolithic Ruby on Rails architecture to the decoupled Jamstack architecture
    - Hyper-growth DTC brand Manscaped entered 10 new international markets, efficiently, while consolidating from 6 Shopify sites to one
    - Victoria Beckham Beauty created a unique brand and luxury retail experience using headless Shopify, and was able to launch an entirely new e-commerce business in 3 months, thanks to the Jamstack
- **Get started by augmenting your e-commerce application or migrating it to a headless architecture on the Jamstack:**
    - Discover templates, videos and resources to start your headless commerce journey
    - Talk to a Netlify expert who can help architect your stack and plan your migration

Table of Contents

PART I

Why Jamstack?

Why Jamstack?

When the coronavirus pandemic inhibited in-person shopping across the globe, e-commerce boomed. Stay-at-home orders encouraged many consumers to finally embrace digital storefronts. In the summer of 2020, IBM's US Retail Index indicated that e-commerce barreled through 5 years worth of growth in a mere 5 months.[1] Data from 2021 indicates that those trends are here to stay.[2]

Online shopping is nothing new. But as e-commerce grows in popularity, **customers' expectations for performance and experience on e-commerce sites is rapidly evolving.** Today's digital consumers demand an exceptional digital experience. They expect extremely rapid loading of pages, services, and images. They're accustomed to speedy and reliable checkout processes, and extensive personalization of the shopping experience. If sites don't meet those expectations, other retailers are just a click away.

Failing to meet shoppers' new, heightened expectations comes at a high cost. The probability of a bounce increases 32% as page load time goes from 1 second to 3 seconds.[3] At the extreme end of the spectrum, Amazon estimates that a slowdown of just one second in their site would cost the company $1.6 billion in sales each year.[4]

But as e-commerce companies race to deliver on these new customer demands, **developers are facing massive architectural challenges.**

Many of the monolithic e-commerce platforms that serviced customer demands five years ago simply aren't cut out to meet these new challenges. Without an army of developers, they can't provide the scale, the customization, and the speed necessary to meet consumer demands.

These architectural challenges are prompting a move to a new way of building for the web: the Jamstack. Jamstack is a modern architecture and set of best practices for building websites and web applications. The UI is compiled, the frontend is decoupled, and data is pulled in as needed. In the commerce space, the Jamstack architecture is often synonymous with the term headless commerce, since the frontend of the application is decoupled from the backend logic and infrastructure. It brings all sorts of benefits, from easier scaling to faster, more reliable workflows. For e-commerce companies, it can mean the difference between deploying a monolithic Shopify site once a week to creating unique, personalized shopping experiences, and shipping regularly with reliable developer workflows.

In this whitepaper, we'll explore some of the main benefits of headless commerce, with real-life examples of companies that are already reaping the rewards. We'll explore how implementing a Jamstack architecture is allowing these companies to meet customer demands, through achieving:

Near instantaneous page loads

With Jamstack, the entire front end is prebuilt into highly optimized static pages and assets during a build process. Not only does this massively improve page load speed and increase conversions, it also boosts your search rankings and discoverability.

Performance at scale

Because Jamstack projects don't rely on server-side code, they can be distributed instead of living on a single server. This means increased resilience and elastic scale that can handle any viral ad campaigns, major project launches, or Black Friday traffic that comes your way.

Personalized shopping experiences

The more of your app you can push to the edge, taking advantage of edge functions and API services, the better the user experience. And with Jamstack's Git-based workflows and a thriving ecosystem, developer workflow improvements mean you can ship more, faster.

Put together, these three pillars help e-commerce companies of any size deliver on the experiences users are becoming accustomed to on major e-commerce platforms. Headless commerce and the Jamstack empowers teams to build best-in-class solutions to meet customer demands in a way that's just not possible on monoliths.

1 - https://techcrunch.com/2020/08/24/covid-19-pandemic-accelerated-shift-to-e-commerce-by-5-years-new-report-says/

2 - https://econsultancy.com/stats-roundup-the-impact-of-covid-19-on-ecommerce/

3 - https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/page-load-time-statistics/

4 - https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales

PART II

The Limitations of Monoliths

The Limitations of Monoliths

Digital storefronts, and the applications that build them, have high expectations. That's because the digital storefront can be a difficult place to convert potential customers to buying customers. For starters, how does a digital storefront create “experiential shopping” the way a brick-and-mortar location can? Where’s the ambiance? The lighting, the smells, the smiling staff, not to mention the opportunity to see, hear, touch, and try on or test the product? **User experience—including unique design and the ability to push fresh content —is a competitive differentiator.**

Secondly, with digital storefronts, pricing is transparent. With a matter of clicks, anyone can comparison shop for better prices from competitors around the world. This drives down margins and puts the emphasis on volume. That, in turn, makes lost conversion opportunities more costly than ever before. Another challenge of digital commerce is capturing and keeping the attention of the shopper. A shopper in a brick-and-mortar store may be content to browse or stand in line for 15 minutes; a digital shopper is much more impatient and easily distracted. **For online shoppers, milliseconds matter.**

Poor performance is expensive

These challenges explain why e-commerce managers are acutely aware of the need to improve website performance. Downtime, slow load times, bounce-rates, etc., all directly and negatively affect the bottom line.

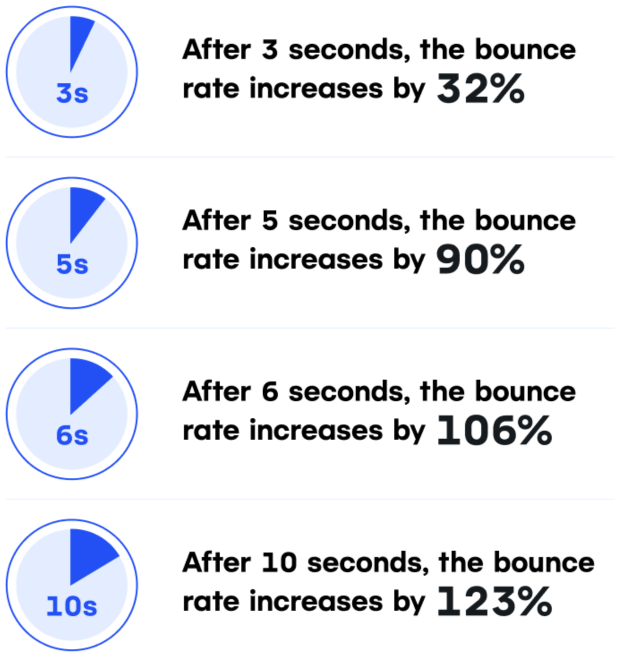
- Improving site load time by a tenth of a second boosts conversion rates by 8%.[5]
- The probability of bounce increases 32% as page load time goes from 1 second to 3 seconds.[6]
- The average time it takes to fully load a mobile landing page is 22 seconds, but 53% of mobile site visitors leave a page that takes longer than three seconds to load.
- Unexpected downtime is expensive. In 2018, J Crew lost an estimated $700,000 in sales when their site couldn’t scale to manage Black Friday traffic loads.[7]

5 - https://www2.deloitte.com/content/dam/Deloitte/ie/Documents/Consulting/Milliseconds_Make_Millions_report.pdf

6 - https://www.thinkwithgoogle.com/marketing-strategies/app-and-mobile/page-load-time-statistics/

7 - https://www.businessinsider.com/jcrew-website-crashes-on-black-friday-2018-11

THE EFFECT OF PAGE LOAD FROM 1 SECOND TO 10 SECONDS



After 3 seconds, the bounce rate increases by **32%**

After 5 seconds, the bounce rate increases by **90%**

After 6 seconds, the bounce rate increases by **106%**

After 10 seconds, the bounce rate increases by **123%**

And unfortunately, the numbers look even worse on mobile. MachMetrics, for instance, describes how mobile phone use impacts e-commerce experiences:

> ***"In 2017, mobile internet usage surpassed desktop as the majority. [Three] years later, this is still the case with 53% of website visits coming from mobile devices. Projections for the next few years expect the mobile market share to increase as mobile devices become more powerful and more prevalent."***

Unfortunately, the data from Backlinko[8] paints a bleak picture when it comes to mobile internet browsing. They found that the average web page takes 87% longer to load on mobile vs. desktop. In addition, the average desktop Time to First Byte (TTFB) speed is 1.28 seconds on desktop and 2.5 seconds on mobile. Finally, they found that the average time it takes to fully load a webpage is 10 seconds on desktop and 27 seconds on mobile.[9]

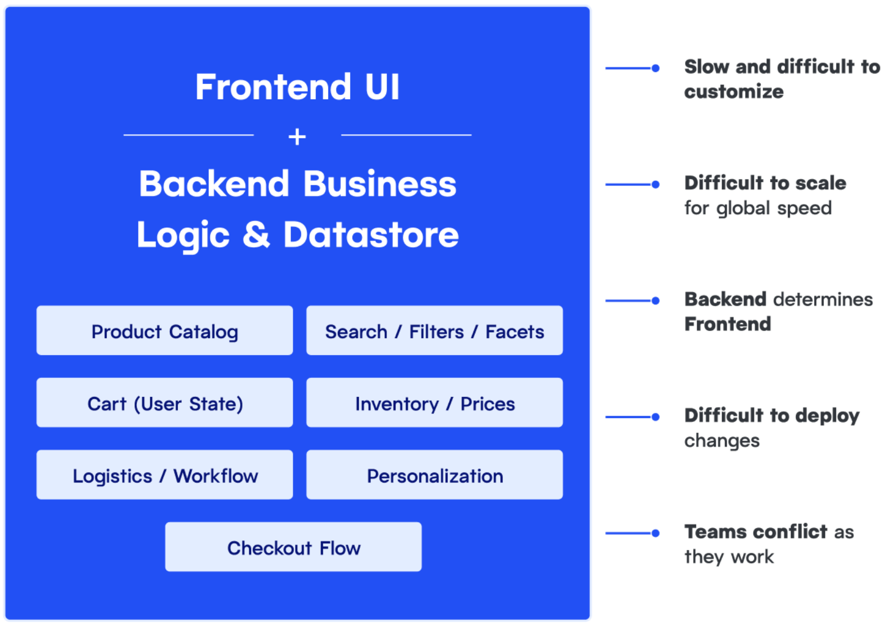
8 - https://backlinko.com/page-speed-stats

9 - https://www.machmetrics.com/speed-blog/average-page-load-times-for-2020/

Why Monolithic Platforms Can't Meet Customer Demands

LEGACY APPROACH

Monolithic


Frontend UI
+
Backend Business Logic & Datastore
Product Catalog
Search / Filters / Facets
Cart (User State)
Inventory / Prices
Logistics / Workflow
Personalization
Checkout Flow
Slow and difficult to customize
Difficult to scale for global speed
Backend determines Frontend
Difficult to deploy changes
Teams conflict as they work

To keep up with market trends, e-commerce companies need to produce a competitive web experience. But that can be a *really* hard thing for dev teams to provide when they're fighting against an older tech stack. It's a constant struggle to keep the monolithic app reliable and performant, like managing security events, traffic spikes, and tech debt.

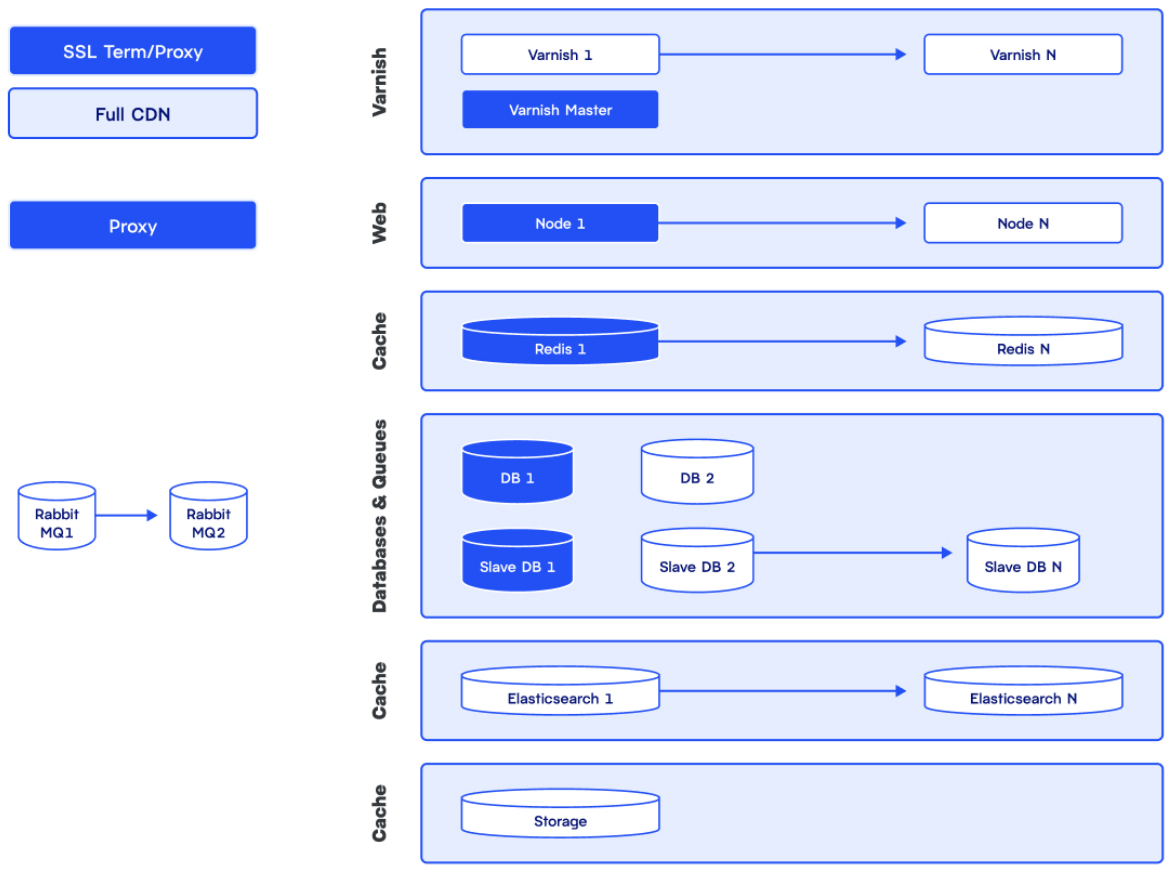
There's another kind of challenge companies face under this architecture: organizational hurdles. With monolithic e-commerce applications, the organization is often built around the tech stack, so there's a huge coordination tax to get anything done. Any change means aligning with everyone involved and managing dependencies across teams. In order to ship bug-free code, teams wind up shipping less.

Technological Limitations of Monoliths

Monolithic applications structurally link the frontend UI to backend services. All of these services are structurally tied to this monolith: the entire product catalog, together with detail pages, the search function, filters, cart with the user state, the checkout workflows and more. With this architecture, you can see three major limitations:

1. **It's time consuming to change anything.** The backend services are tied to and serve the entire frontend as a unit, which makes it hard for developers to customize the site and express the company's unique brand.
2. **It's not performant.** Because every request traverses the stack, any change has to be coordinated across every layer. This creates a significant performance hurdle.
3. **It's brittle.** All requests from all visitors hit a common origin, which make it susceptible to DDOS attacks, and creates a single point of failure.

How to Optimize the monolith



There are ways to optimize a monolith, but they're time-intensive, and often require a fleet of developers to maintain. To illustrate, pictured below is Adobe's recommendation to run Magento, one of the traditional, open source e-commerce platforms.

You can see here the many different components that are required to optimize the performance of this system. These include a Varnish cluster for caching generated HTML, several web nodes, a caching layer built with Redis to lighten the load on the database, a large database cluster with replicates, a dedicated search pod on top of Elasticsearch, and dedicated storage layer for images as well as messaging queues, CDNs, proxying, and so on. All of these complicated components are required to operate at a fundamental level of performance for a modern e-commerce experience.

And even with these optimizations in place, it doesn't get around the other organizational problem we've identified: it's still *really* slow to make changes.

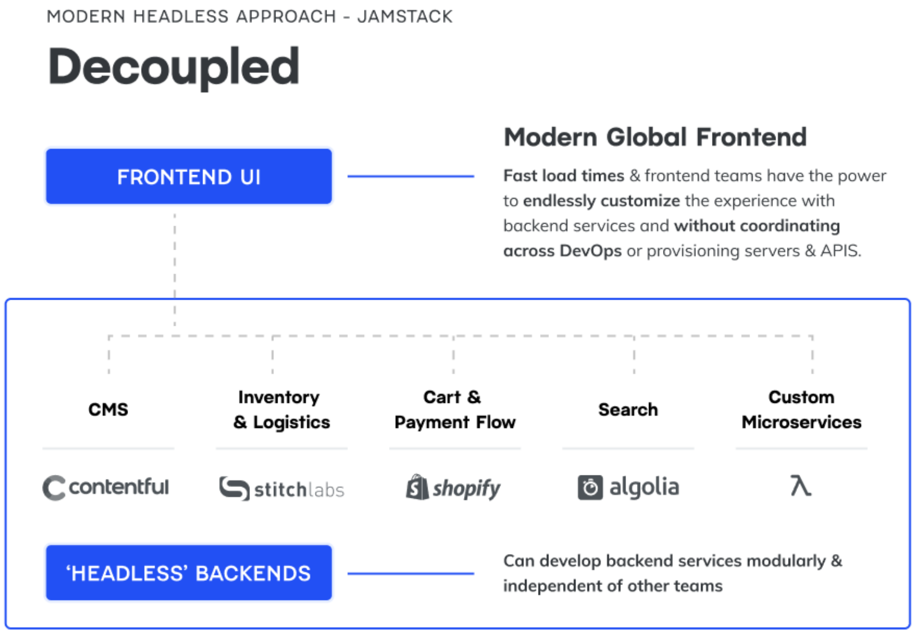
There's a better way.

PART III

How Headless Commerce is Different

How Headless Commerce is Different

The headless approach, by contrast, disaggregates the monolithic stack, decoupling the frontend from the backend, tying them together with APIs. By doing so, headless gives us the unique opportunity to combine best-of-breed services for all the components of e-commerce.


MODERN HEADLESS APPROACH - JAMSTACK
Decoupled
FRONTEND UI
Modern Global Frontend
Fast load times & frontend teams have the power to endlessly customize the experience with backend services and without coordinating across DevOps or provisioning servers & APIS.
CMS
Inventory & Logistics
Cart & Payment Flow
Search
Custom Microservices
contentful
stitchlabs
shopify
algolia
λ
'HEADLESS' BACKENDS
Can develop backend services modularly & independent of other teams

This offers developers endlessly customizable architecture where best-in-class tools perform specific tasks. Serverless functions glue all of these services together along with your own internal services and data, so frontend developers can easily tailor API calls to each part of the experience. This means:

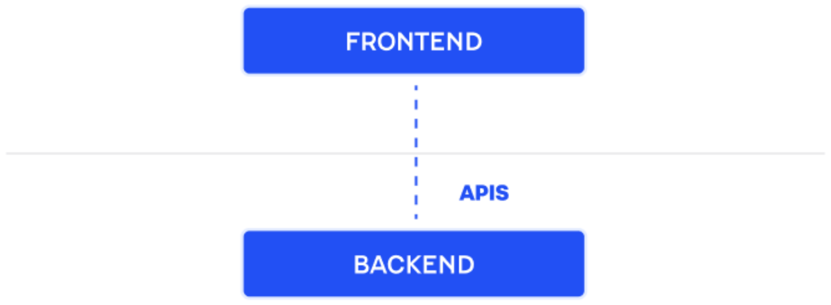
- For e-commerce teams that really want to provide a unique or differentiated experience (eg. luxury brand), customization is much easier.
- For small e-commerce projects that just want to add a shopping cart and checkout workflow it becomes very easy to add that to your site.
- Eliminates the coordination and dependencies between teams.
- You unlock frontend teams to move faster and at their own pace.

So now, frontend developers can write and check-in code without ever having to think about all these complex dependencies on the backend.

Jamstack Overview

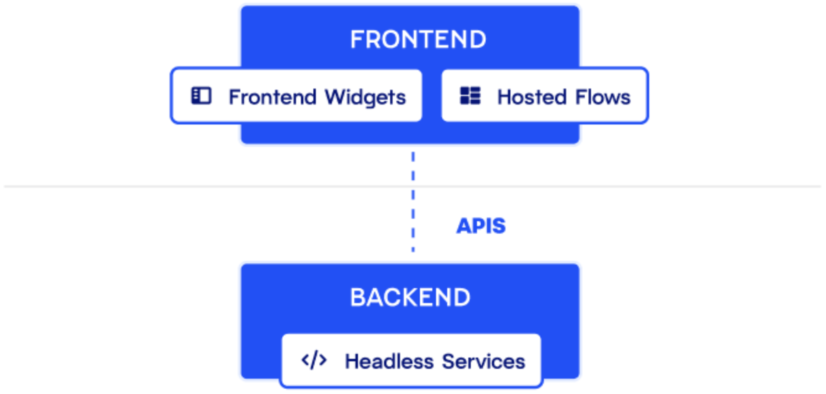
Jamstack is an architecture designed to make the web faster, more secure, and easier to scale. It builds on many of the tools and workflows which developers love—including Git-based workflows, frontend frameworks like React and Vue, and serverless functions—and which bring maximum productivity. The core principles of pre-rendering content, and decoupling the frontend from the backend, enable developers to not only build and iterate e-commerce applications faster, but create a much faster experience for their customers visiting their site.

In the traditional approach, a website request requires that the server do a lot of work, pass the content via network transit to the user's browser, and the user's browser engine also does a bit of work on the content before it is finally made available to the user. Even in the case of single-page applications where very little work is required of the server, the user's browser essentially receives an empty app shell, and the user's browser then does all the work of stitching together the view for the user.



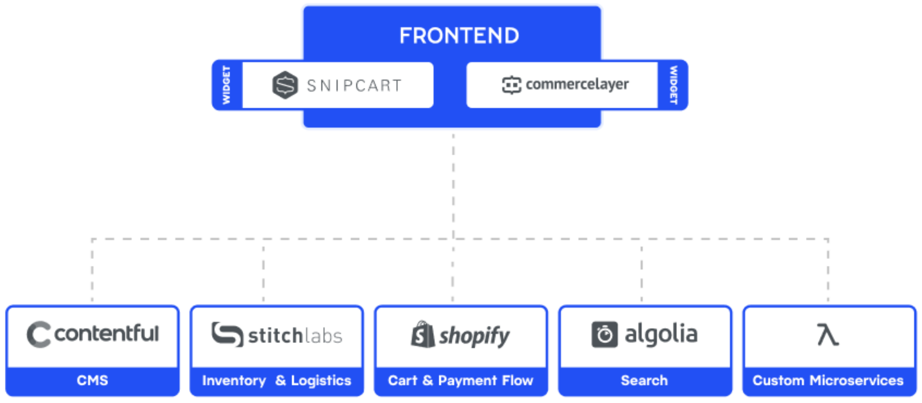
Customizing E-Commerce Frontends

A self-standing frontend enables developers to take many different architectural approaches. Some may give their users a luxury experience by completely customizing how the site looks and performs. Or, in the case of smaller projects, a developer can quickly add features by pulling in widgets to power the shopping cart or the checkout flow. Others may take the approach of hosted flows, where you build a fully customizable product catalog and search, but the actual checkout takes place somewhere else.

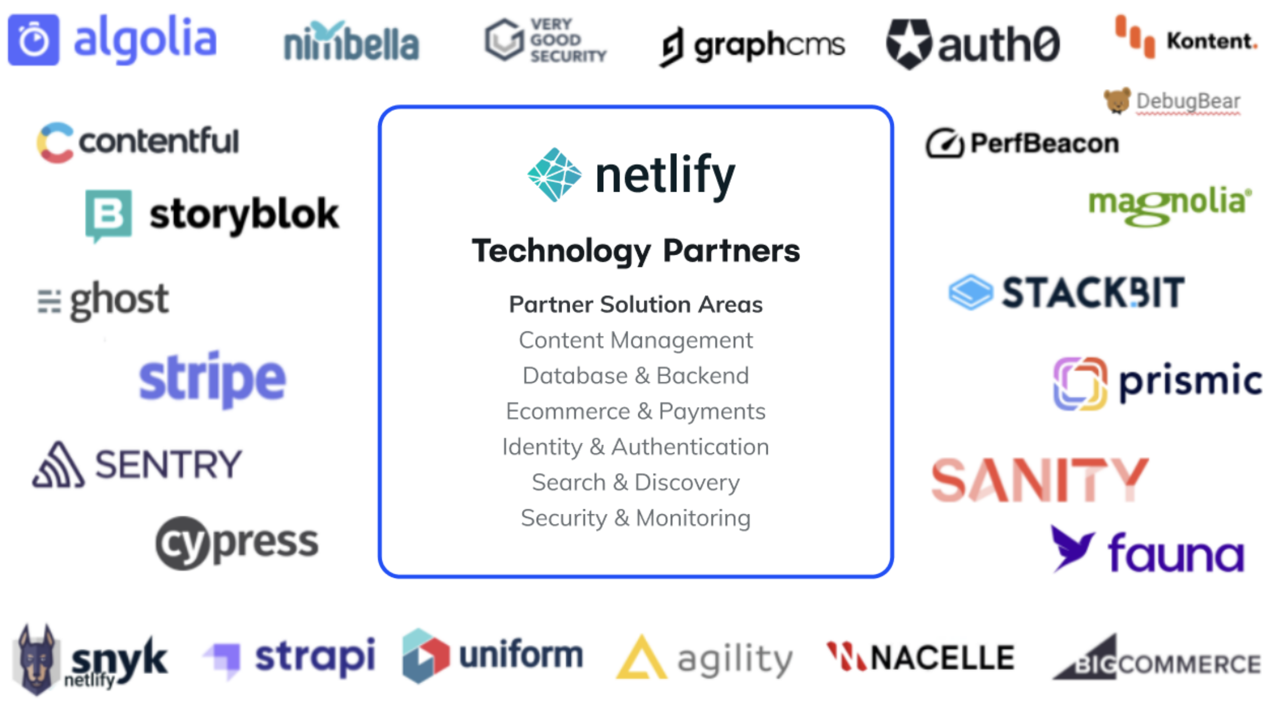


Customizing E-Commerce Backends

Going further, the real power of headless is that there's rarely only one backend. For example, you might use a content management system like Contentful to manage your product catalog. There are a wealth of choices for CMS systems. You could use Stitchlabs to manage inventory and logistics or run payments through Shopify. Algolia is a common option for faceted search and filtering. Serverless functions can be used as a custom layer to glue all of these different services together. With this flexibility, web developers building the frontend can tailor the API calls to a variety of different backend headless services based on what they are building, not what the platform will allow.



The backend headless services—such as the CMS, inventory, logistics, checkout, and payment services—are exposed through a REST API or a GraphQL API as JSON documents. Developers can pick any site-generating framework or technology—for example, Next.js, Eleventy, Gatsby, Gridsome or Nuxt—that provides the right user



experience for their customers. And that's just the beginning of the Jamstack ecosystem, which continues to grow each year:

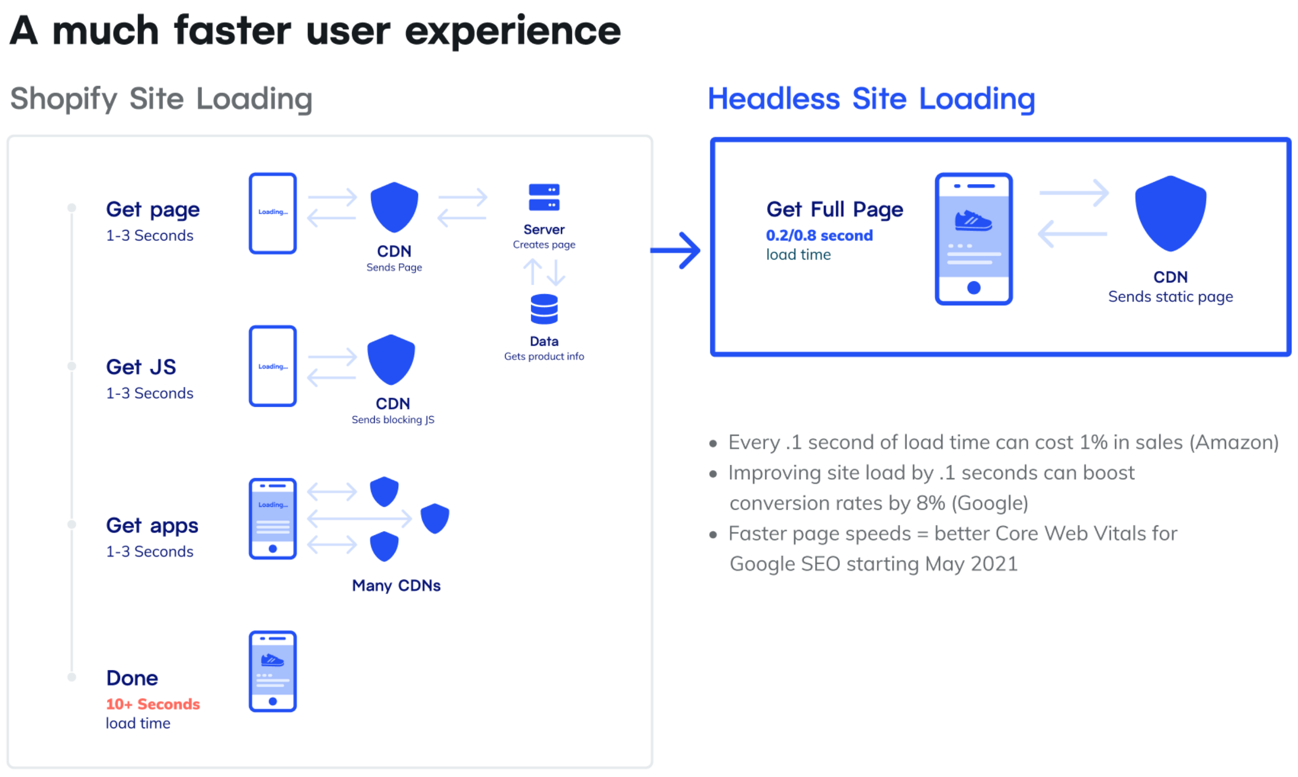
This flexibility truly transforms website development and performance. Enterprises are looking to achieve the page load speed, scalability, and customization that headless architecture can offer.

Let's dive into those three benefits a bit more.

Jamstack Sites Provide Near Instantaneous Page Loads

Jamstack sites are fast. By pre-rendering sites ahead of time, initial content can be served to visitors immediately. Whereas a Shopify site needs to traverse the entire stack in order to get the JavaScript and all other third-party apps in order to paint pixels to the screen, a Jamstack site sends users static pages that have been cached to the edge.

Platforms like Netlify manage caching for you, which means caching greatly improves site load times by enabling access to resources that have either been fetched from a previous



- Every .1 second of load time can cost 1% in sales (Amazon)
- Improving site load by .1 seconds can boost conversion rates by 8% (Google)
- Faster page speeds = better Core Web Vitals for Google SEO starting May 2021

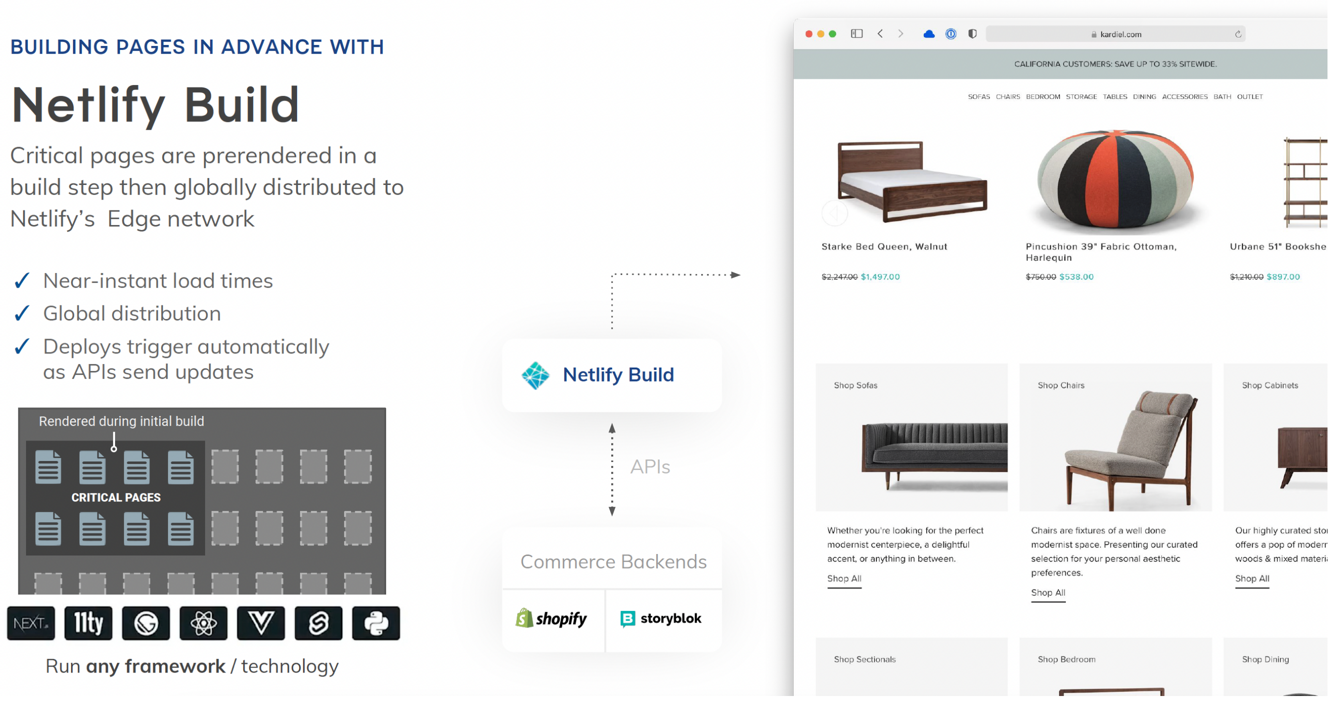
site visit or retrieved in advance.

Where Netlify fits in

Use the commerce backend of your choice and connect the API to Netlify. Critical pages are then prerendered in a build step then globally distributed to Netlify's Edge network, which is distributed just like a CDN, but has advanced functionality for publishing entire sites and applications. Dynamic content can be fetched at runtime using APIs and Netlify Functions, and long-tail product catalogues and less popular content can be deferred for optimal performance using On-Demand Builders. Netlify also provides a breadth of features within one platform and streamlined workflow, including collaborative deploy previews, instant rollbacks, forms, and even split testing.

Netlify provides an agnostic platform to integrate the technologies you need to build a Headless Commerce applications and marketing websites on the Jamstack. This includes many different established and emerging Headless Commerce backends, including

BigCommerce, Shopify, Stripe, CommerceLayer and more; site generators including Next.js, Gatsby, or Nuxt; headless CMS providers including Contentful, Prismic, and Sanity.



BUILDING PAGES IN ADVANCE WITH

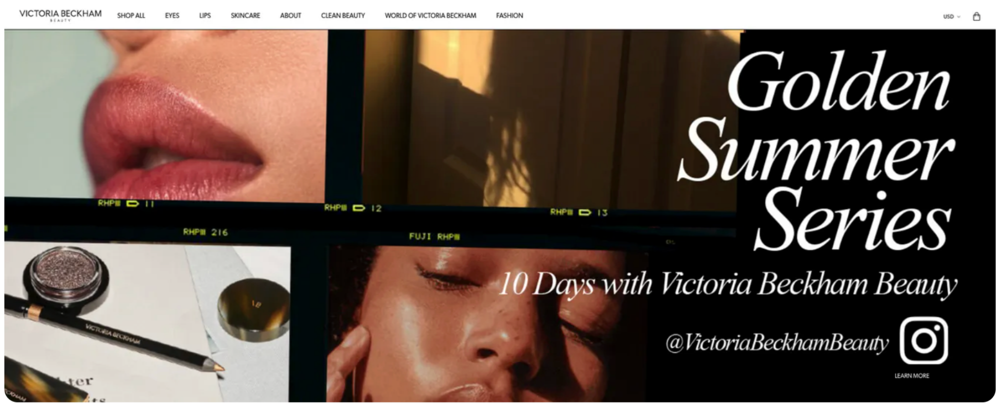
Netlify Build

Critical pages are prerendered in a build step then globally distributed to Netlify's Edge network

- ✓ Near-instant load times
- ✓ Global distribution
- ✓ Deploys trigger automatically as APIs send updates

Adopting Jamstack on Netlify and implementing best practices lays a strong foundation for performance, as well as high Core Web Vitals scores. Much of these performance benefits are enabled by default on Netlify. For example, assets delivered by Netlify Edge are automatically compressed with Brotli encoding for even smaller sizes and decompression efficiency. Netlify Edge is distributed just like a CDN, but with advanced functionality for publishing entire sites and applications. Its global distribution with automated prerendering makes for blazing fast site delivery. Even higher scores are achieved with Netlify's HP Edge, available on Enterprise plans, a network with 30% faster speeds than our standard edge.

Case Study: Fashion brands are asking what makes Victoria Beckham Beauty's site so fast



Victoria Beckham Beauty (VBB) launched a new site for its cosmetics line using a headless architecture approach, and the site has set a new performance bar for the fashion industry. Ryan Foster, head of VBB's web agency Fostr, said "Other fashion brands benchmarking against VBB want to understand how it's so much faster."[10] Indeed, the site has impressive performance, but raw speed is far from the only benefit VBB has achieved by "going headless." In fact, VBB's primary objective in adopting a headless architecture was not speed but flexibility. Rather, Foster and the VBB team wanted fine-grained control over the brand experience—from visual design to user journeys, even down to the exact text of each URL. Development velocity actually *increased* because the headless approach eliminated the design and development constraints of the former, traditional architecture.

VBB adopted a Jamstack using Gridsome and Vue as the layer that allowed their developers to build exactly the experience they wanted for their users. VBB used Contentful for the product catalog, Shopify for the actual checkout flows, and Stitch Labs for inventory and logistics.

No longer does the monolithic platform dictate what developers can or cannot do. VBB has also experienced a boost in developer productivity and velocity, because the ease of working with the tooling of hand-picked and refined applications far exceeds the cumbersome experience of working with PHP-based templates inside a big monolithic system.

Read the case study ➜

10 - https://www.netlify.com/customers/victoria-beckham-beauty/

Jamstack Sites are Performant at Scale

Jamstack sites are not only faster, they're extremely reliable and secure. Because they're distributed to a CDN, Jamstack sites can achieve elastic scale for viral ad campaigns, product launches, or Black Friday traffic.

Case Study: Spring's approach to migrating off a monolith, and how they scaled to 6 million storefronts on the Jamstack

SPRING for creators

Spring is a large e-commerce marketplace where individual designers and content creators can launch their own turnkey storefronts for custom-branded apparel and merchandise. The Spring community is huge, hosting over 6 million storefronts. As the company was experiencing massive growth, Director of Engineering Rick Takes realized their monolithic Ruby on Rails web architecture could not scale to support their expanding web team and initiatives. They needed to reduce build times, deploy faster, and not worry about downtime. He set out to prototype a Jamstack approach using Netlify, breaking its monolithic e-commerce app into microservices.

The result? Spring's deploy time of over 90 minutes has been reduced down to a single minute. Takes' migration from a monolith to microservices has also improved the developer experience and productivity for his team. It enabled them to launch an ambitious new initiative called Branded Stores. The early results from the new Branded Stores on the Jamstack have been

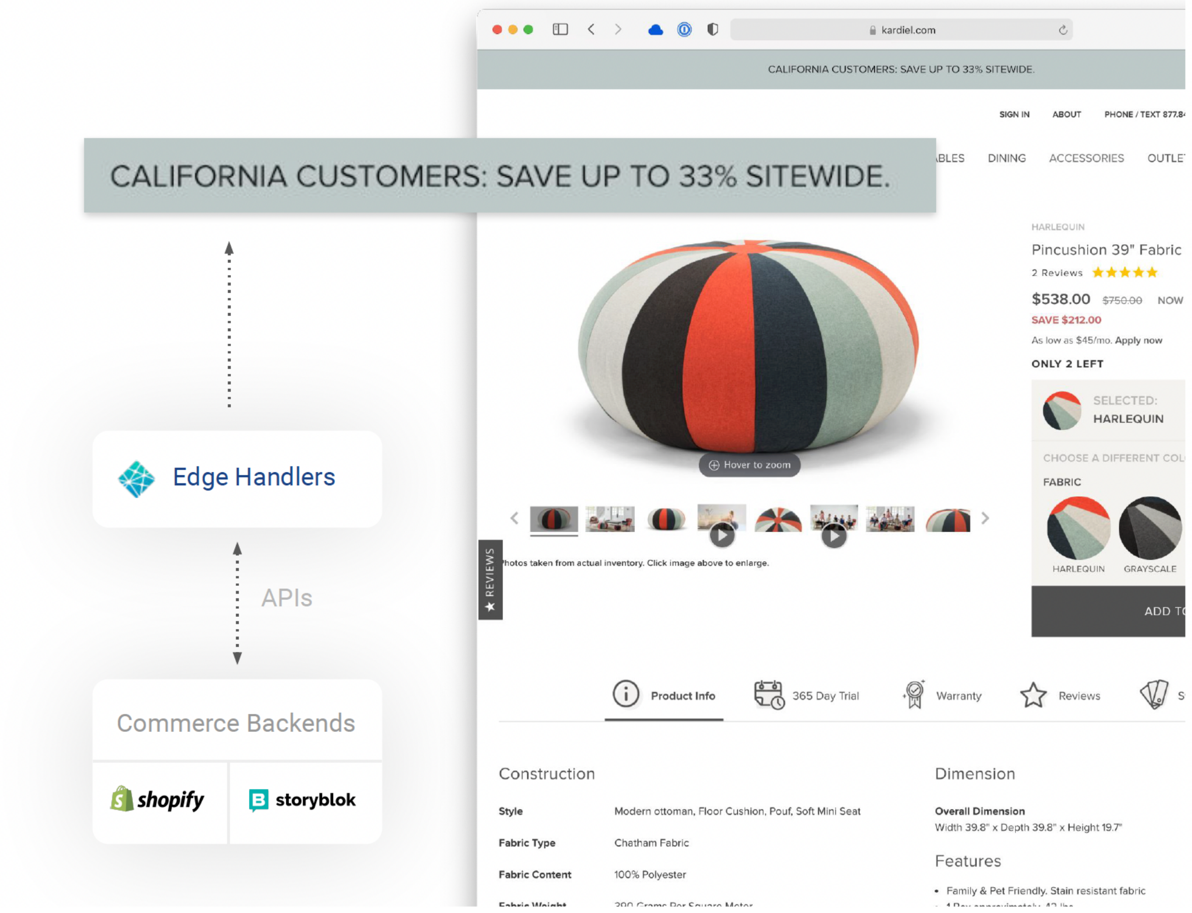
Read the case study ➜

Jamstack Sites are Easy to Personalize

With Jamstack sites, content is delivered instantly from the edge, making it faster to send personalized content to your customers. As pre-rendered pages are served from the edge, they can be customized.

This makes it that much easier to personalize products for each individual visitor, inject custom ads and promotions, and even localize content.

Netlify’s Edge Handlers API can fetch, cache, and manipulate content at the edge, so you can deliver a fast, personalized experience with minimal development work. Use Netlify's git-based workflow and the tooling that your team already knows to build, preview, and deploy Edge Handlers right alongside your web apps and sites. During each build, Netlify will automatically discover your handlers, bundle any required dependencies, and deploy them to the edge.



Case Study: From 6 Shopify sites to 1: Manscaped's multilingual journey into 10 global markets

Until 2019, D2C company Manscaped was fully built on the Shopify stack. But as they grew into different markets and internationalized their website, they realized this wouldn't scale. Each time they entered a new market with new languages, they needed to create a new version of their site. Copying and pasting their theme compounded any existing errors, and made it challenging to make even minor updates—a must-have in D2C. They decided to migrate to the Jamstack.

Not only did their migration to the Jamstack improve site performance, it's allowing them to scale into new markets with relative ease. Instead of copying and pasting their sites, they can now handle multiple sites from a single codebase, and by using Gatsby's internationalization plugin, they're able to be multilingual within the same repo.

Watch Video ➜

Conclusion

E-commerce managers are acutely aware of how website performance impacts business objectives: downtime, slow load times, high bounce-rates, and other performance challenges all directly and negatively affect the bottom line.

Headless commerce with Jamstack offers:

- **Site performance**
- **Flexibility of microservices**
- **Content integration**
- **Developer productivity**
- **Speed of experimentation**
- **Security & PCI**
- **Personalization**

That is why headless commerce with Jamstack is an architectural approach that's right for a new era in which e-commerce system performance has emerged as a critical component in the success of retailers everywhere.

Contact Sales

netlify