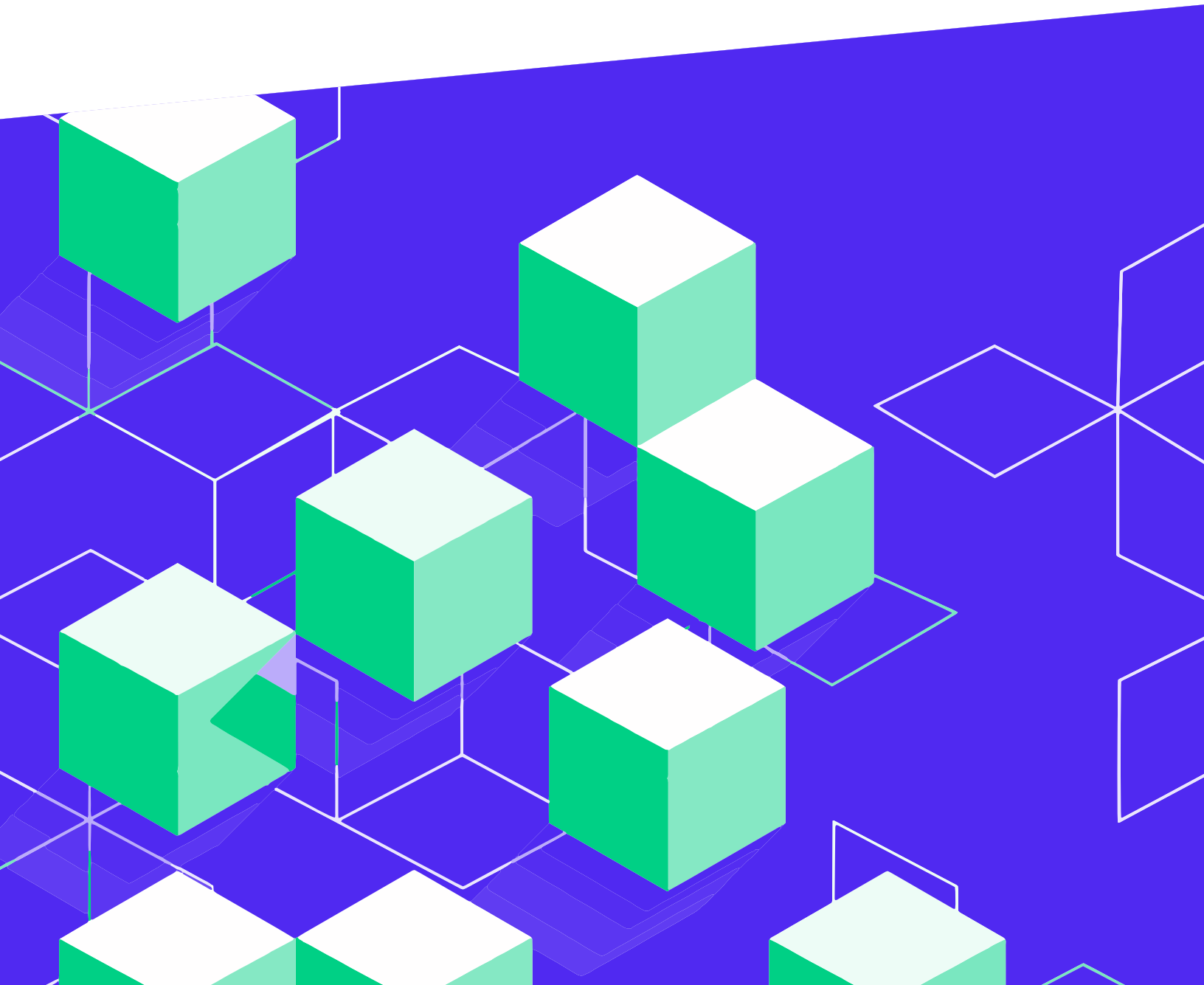
nacelle

Build vs. Buy

Headless Commerce Architecture



nacelle

Contents

nacelle

Executive Summary

This guide leads technical decision makers and eCommerce teams through the pros and cons of building or buying a headless commerce solution. It delves into what a custom build will entail, compared to buying an off-the-shelf solution. It also includes a decision making framework for “build vs. buy” deliberation.

So, You Want to go Headless?

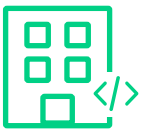
The benefits of a robust headless commerce solution that powers a highly performant frontend are too lucrative to ignore. At a high level, the right architecture can be a conduit for an improved and custom online shopping experience. Lightning fast page load speeds and mobile-first functionality give way to boosted conversion rates, higher AOV, and increased sales, among other KPI improvements.

Beneath the surface, there are even more advantages such as improved developer workflows and simplified backend API and code management. It’s no secret that a performant frontend is only as performant as the APIs that power the backend. In order to tap into speed and performance gains, you need a highly effcient and speedy way to provide the right data to the frontend at the right time.

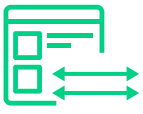
But before tapping into these headless commerce “wins,” you need to implement the right headless commerce architecture for your brand that encompasses technical requirements, internal competencies, and budget.

Enter the “build vs. buy” debate. Your team needs to decide how to implement the best solution for your operation—you might choose to build a custom solution in-house, tap an agency for build support, buy an “end-to-end” solution, or buy a solution you can build on top of in order to get the ideal blend of custom and out-of-the-box functionality.

In-House Solution



Agency Solution



End-to-End Solution

Buy & Build Solution

Your brand will have many stakeholders in this decision, but your engineering team should ultimately drive the answer to whether or not it's best for your brand to "build or buy."

Most eCommerce brands are stacked with top-notch developers. The question usually isn't whether or not it can be done in-house, but if building in-house is the best use of your dev team's time. As a rule of thumb it's best to make sure that anything you build and maintain in-house is going to differentiate your brand, become part of your IP, and set you apart from competitors in a profitable way. It certainly shouldn't leave you crippled by technical debt in the future. If building a solution in-house doesn't check those boxes, it's likely not a good use of time or resources.

It's also important to establish a stance on monolithic vs. microservices and best-of-breed functionality. Our view is that a best-of-breed strategy, also known in the industry as composable commerce, offers the most flexibility and responsiveness to the changing eCommerce landscape while minimizing technical debt.

Many merchants reach a point on their path to enterprise-status (if they're not there already) where they realize the need for microservices, especially as they build custom parts of their tech stack. Having the right headless commerce architecture in place before growing your best-of-breed strategy will make the process significantly easier across the board.

This guide will cover the different "build vs. buy" options: building in-house, building with an agency, buying an "all-in-one" solution, or buying AND building (buying a system with customizable opportunities on top). But first, let's talk about where you are now, and the headless commerce features that might appeal to you most.

Image by DCStudio on Freepik

nacelle

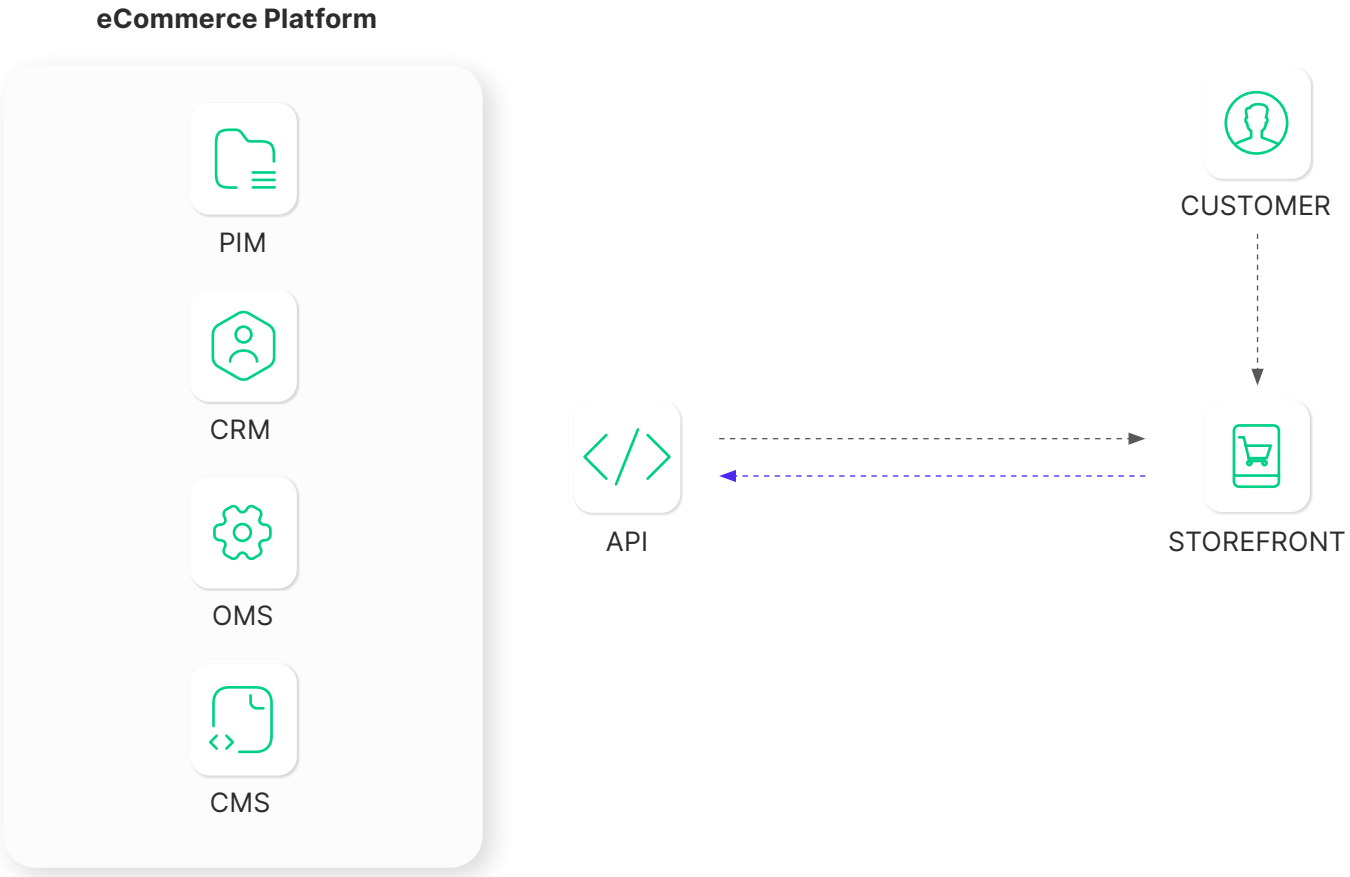
Going Headless With an eCommerce Platform

It is absolutely possible to go headless with your existing tech stack, including your current eCommerce platform. As mentioned above, implementing a headless commerce solution before making tech stack changes - or to better orchestrate and manage your best-of-breed stack - is hugely beneficial in itself.

The idea is to keep the platforms, third-party applications, and tools that are working well for your business, while still reaping the rewards of a performant headless commerce solution.

Some of the top headless commerce features that merchants on an eCommerce platform should look for include the ability to keep the tech stack you have if you want to, but also the option to start integrating new tools when you're ready.

No rip-and-replace data or platform migration will be required with this approach. This offers the flexibility to keep your tech stack agile and flexible while you go to market much faster. It also lets merchants at this stage incorporate important aspects of their stack such as a headless content management system (CMS), which is typically always the first tool added after going headless. Third-party application integrations and backend management simplification - such as one codebase for dev teams to manage - will also be appealing.



nacelle

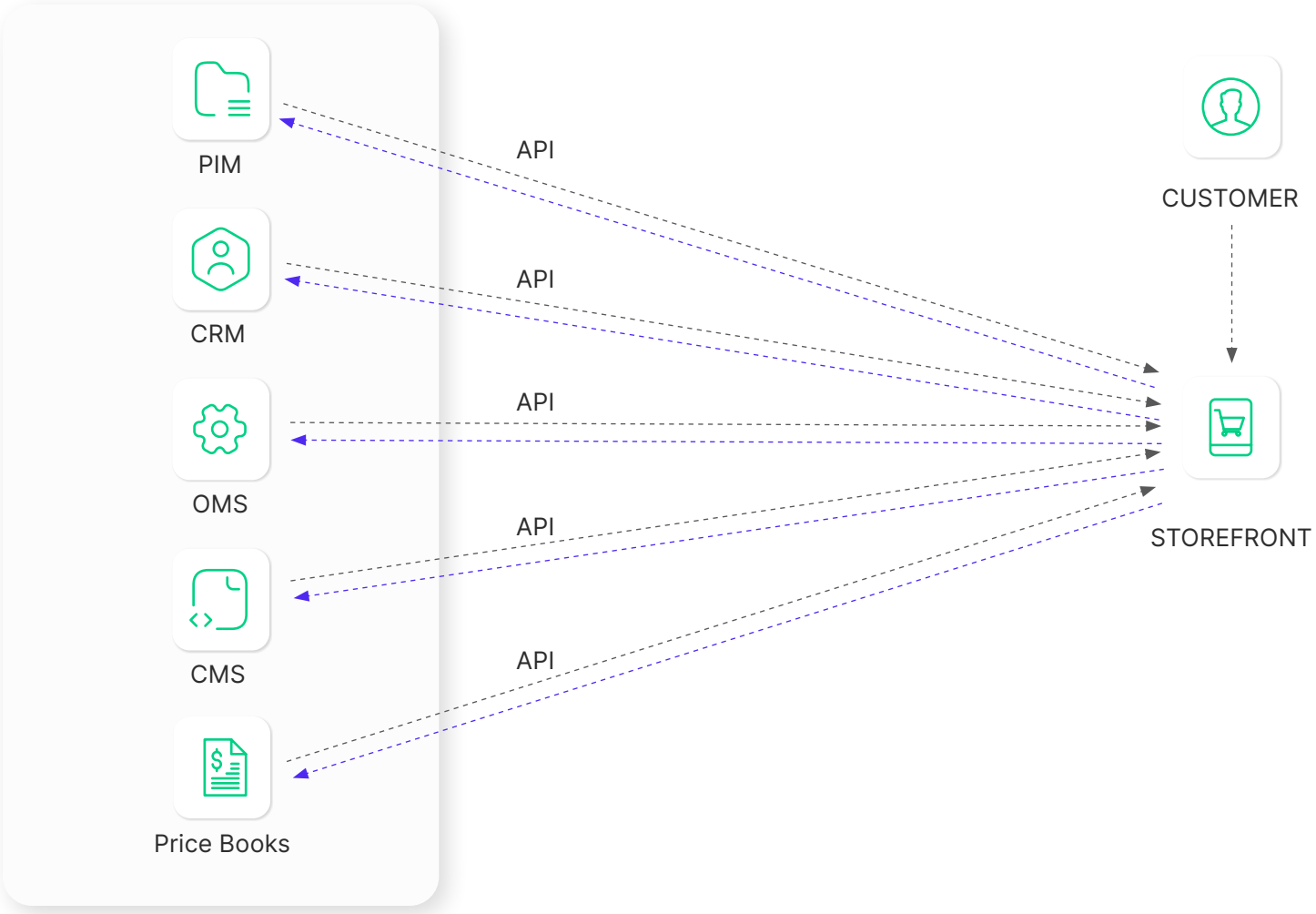
Going Headless With a Custom Stack

This scenario is often applicable to enterprise merchants or merchants on the verge of growing into enterprise-status. At this stage, engineering initiatives are what set your brand apart from competitors and delight customers in unique ways.

Microservices, also known in the industry as distributed systems, are typically already being incorporated here because this strategy allows engineering teams to customize, build, and manage certain aspects of their stack in-house.

Attractive headless commerce characteristics at this stage will include rapid data flow between the frontend and backend, as well as between disparate backend systems. Also, a headless commerce architecture that acts as a layer of abstraction will simplify the “piping and plumbing” on your backend. Look for functionality that will simplify developer workflows (like the ability to use Git Push in the development cycle) and ultimately alleviate pressure on developer bandwidth.

If you choose a “buy and build” hybrid option, ensure your solution has a managed service element that handles updates to backend systems and APIs so your team isn’t bogged down with maintenance that detracts from their differentiating engineering projects. This will also ensure you always have the most up to date version of the tools that fuel your webstore. We’ll dive further into the “buy and build” approach below.



nacelle

Building a Custom Headless Commerce Solution

If you decide to build a custom headless commerce solution in-house or in conjunction with an agency, there are considerations that might not initially come to mind such as maintenance costs, data transferring, and scalability issues that could impact the scope of your project.

Several Nacelle employees, including our CEO Brian Anderson, came from the eCommerce agency world. At his previous startup, Anderson ran a technical eCommerce agency where his team specialized in custom storefronts and headless progressive web application (PWA) builds. This was before "headless commerce" was a widespread industry buzzword.

The outcome of their efforts was overcoming the uncharted obstacles of creating headless builds for clients on a brand-by-brand basis, so we empathize with the inhouse/agency "custom build" pain points—it's ultimately what inspired the birth of Nacelle.

Building In-House

Based on experience, the build cost alone of a custom headless commerce solution is between 3,000 to 5,000 human developer hours. Maintenance costs will inevitably extend well beyond that figure, especially if and when unforeseen challenges arise.

Experience is key to headless build success and not all headless builds are created equal. For example, an engineer's headless build experience with one eCommerce store might not translate well to another if the webstores have important differences such as catalog size or tech stack goals.

There are common "gotchas" with building a headless commerce system that your build team needs to be intimately familiar with. Often it's the "back of the napkin effect" where in theory, everything should run beautifully, but once in production, unforeseen challenges can derail desired results.

First, a strong data layer must be in place because it needs to aggregate everything on your backend including your PIM, CMS, and OMS. Even if you're leveraging an eCommerce platform like Shopify Plus, you'll likely want to pair it with a headless CMS for easier content management, so you'll still face the challenge of joining and merging disparate data sources together into one.

Furthermore, not all APIs are of the same caliber. Rate limits, poorly designed APIs, and slow APIs can hinder your ability to get the right data to your customers on the frontend at the right time. It might seem like calling various APIs for your static builds and PWA builds will work, but it's deceiving because it won't scale the right way. If you can't flow a large amount of data into your frontend very quickly, you're going to have a significant problem on your hands.

Other important considerations include content modeling and the relationship with relevant third-party applications. Different apps have different levels of APIs and JavaScript SDKs, and frankly, some just work better in a headless environment than others.

nacelle

Building With an Agency

If you don't have the in-house engineering resources to build a custom solution, an appealing alternative is to outsource the build to an eCommerce agency for support.

That said, some of the pain points associated with building a headless commerce solution in-house will also be applicable to building with an agency, including that the agency's headless build experience may or may not translate to your headless needs depending on the types of webstores they have experience with.

It's also important to keep the big picture in mind and note that you might not want an agency to have all the knowledge about maintaining your solution after it's built. This scenario will tie you to an agency long-term and put you at risk of "tribal knowledge" pitfalls should you sever ties with the agency or key people leave.

The reason that Anderson ultimately left the agency world and founded Nacelle is because he realized that building custom headless solutions for clients individually was not sustainable or profitable because he had to rebuild and maintain the same headless infrastructure and APIs every time. He understood it needed to be reimagined as a SaaS business model instead. Custom builds with an agency can be astronomically priced, and that's before factoring in the costs of maintenance or upgrades.

Image by rawpixel.com on Freepik

nacelle

Buying a Headless Commerce Solution

The phrase “build vs. buy” is perhaps overly simplistic when discussing headless commerce architecture. Just as there are different ways to build a solution, there are different ways to buy a solution.

Some headless commerce solutions market themselves as “all-in-one” or “end-to-end” solutions. It may sound turnkey and painless on the surface, but solutions under this umbrella are often fundamentally at odds with the decoupling spirit of headless commerce.

Other solutions, including Nacelle, are better coined as “buy and build” solutions. Inhouse developers will still be involved, and though there’s plenty of out-of-the-box functionality, the solution is not prescriptive and allows for customization.

Buying an End-to-End Headless Commerce Solution

“End-to-end” solutions can be especially appealing to smaller merchants, or merchants who are newer to the eCommerce space, because it allows them to jump right in with a lean development team while being competitive with larger brands.

However, as merchants grow and decide a microservice architecture is a better fit, backtracking out of these “all-in-one” solutions could be a costly, risky endeavor. It will require a risky data migration up front, all at once, and merchants will have to forgo the tech stack they know entirely.

There are a lot of companies out there that claim to be headless, but have an opinion on what CMS or PIM you need to use. Or worse, they force you to use their backend components to get “headless” functionality.

That’s not headless. If there’s a hard dependency between your backend and your frontend, it’s not decoupled, which is the very definition of headless commerce. It’s essentially another monolithic system dressed up as headless.

With monolithic systems, or heavily interdependent “headless” systems, you will experience vendor lock-in if you ever want to change an aspect of your tech stack. This could lead to risky migrations or technical debt in the long run.

nacelle

Buying and Building a Headless Commerce Solution

Buying a customizable headless commerce solution is often the best choice for merchants who want to focus in-house engineering resources on projects that will differentiate their brand, but still need a customizable headless solution connecting their best-of-breed tech stack.

This option allots for the unique requirements of your brand, while eliminating many of the pain points associated with building a headless commerce solution in its entirety.

From data requirements and application integrations to maintenance and management, a solution in this category should inherently address those challenges. These systems take the ideology behind headless commerce a step further than just a backend-frontend separation, because it can decouple every aspect of the backend, as desired.

For instance, it's difficult to ensure real-time interoperability between disparate systems, so it's critical to have a data layer that acts as the "connective tissue" between your tools.

Various systems' data will feed into the data layer, rather than directly into one another and directly into your frontend. This makes your tech stack truly interoperable and flexible to change.

You can add, change, and subtract tools with relative ease, because you only need to change code in one place—the data layer.

The right headless commerce solution will also account for the issues of inconsistent APIs. Nacelle's headless commerce platform, for example, gives merchants one performant, federated API to work with, while managing the APIs from your various systems for you. The managed service component to our offering means our headless commerce experts oversee and manage the "piping and plumbing" of your backend, freeing up your development resources to focus on solving problems that will help differentiate your brand.

Additionally, data velocity is one of the most important aspects of a headless commerce solution. Be sure that if you buy, the architecture you choose enhances—rather than clashes—with the functionality of top frontend frameworks like Nuxt, Next, and Gatsby and JavaScript libraries like Vue and React.

The checklist for headless commerce functionality is quite lengthy, and the above examples are just the start. We'll elaborate and add some of the features you should account for in your headless solution below, regardless of whether or not you build or buy.

nacelle

Why D2C Retailer Enso Rings Decided to Buy and Build

Nathan Call, Director of Technology at Enso Rings, joined the Enso team with headless build experience and prowess of his own. He knew right away that the D2C silicone ring brand needed a headless commerce solution as they moved from a monolithic system to microservice infrastructure. It was just a matter of how and when.

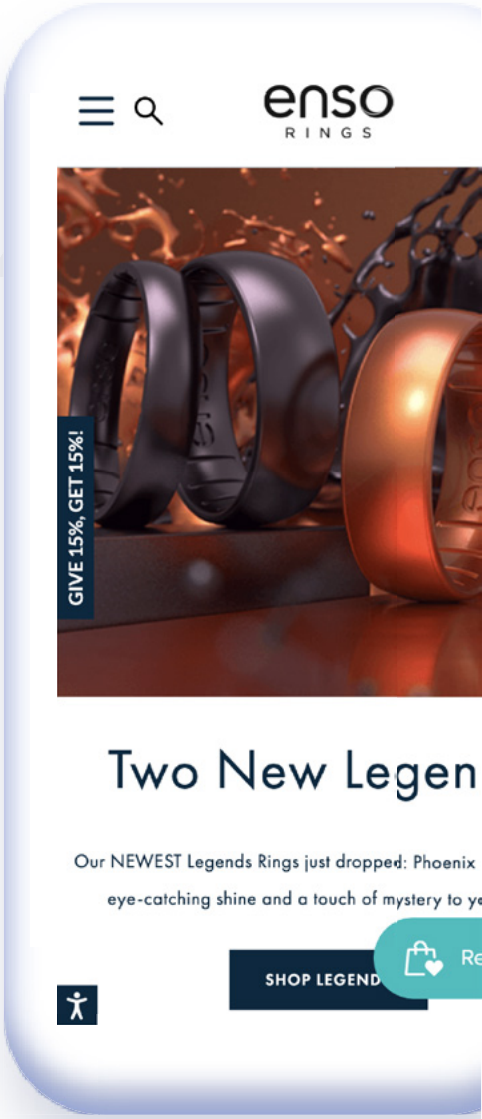
Call conducted extensive build vs. buy analysis, ultimately taking the “buy and build” path with Nacelle. His research started with the open source community and continued with exploring various headless commerce vendors. He tested virtually all of the viable options available for a headless build on top of Shopify Plus.

He found many of the options included great features but realized most would require substantial modifications, data migrations, or extensions to support even the most basic functionality. So, in the true spirit of development, he built a successful proof of concept himself.

“Could it get us where we wanted to go? Absolutely. Could it get us there in the timeframe that we had? No, it couldn’t,” says Call in a video interview on the topic. “It was going to take too much time to get there and it was going to be too hard to maintain with our development team long-term given the other project calendars that we had coming up. Time and money are not unlimited resources and we couldn’t realistically do both.”

The Enso Rings engineering team is full of talented, capable developers, but the decision to build a performant frontend on top of a powerful off-the-shelf headless commerce platform ultimately came down to time and resources.

The decision paid off. Call says the team was able to go-live 5x faster than if they had built the solution from scratch in-house. They also saw a 27% increase in conversions and 13% increase in average order value.



nacelle

What Makes a Great Headless Commerce Solution?

We often encourage merchants to think of a headless commerce solution like an iceberg. Above the water is a small part of the iceberg that represents what immediately comes to mind, like building a performant frontend and application integrations.

Underwater, where the iceberg shows its might, are hidden considerations that are often overlooked when debating build vs. buy. These are important points about what really powers performance on the frontend.

What You See

- PWA Frontend
- APP Integrations

The Rest

- APIs
- Data Velocity & Eventual Consistency
- Layer of Abstraction
- Best-of-Breed Tech Stack Communication
- Unit Test Coverage
- Scalable Performance
- JavaScript SDKs
- CMS & Content Modeling

Miscalculations here can cost money and precious time as you push to take your system live. Here are some top build vs. buy topics that need to be addressed.

APIs:

As mentioned earlier, rate limits, underperforming APIs, and slow APIs can derail architecture success that, in theory, should work—especially when using multiple APIs from each backend system that were not created for this type of functionality. Even if everything looks like it will operate without a hitch on the whiteboard, as you go into production you'll realize you need an extremely robust API to get the right data to your customers on the frontend at the right time. This is why Nacelle makes it a priority to deliver one highlyperformant API to merchants that doesn't have any rate limits. Without it, necessary data flow and site speed benefits from initiatives like static site generation will not be possible at scale because data will be bottlenecked.

Data Velocity and Eventual Consistency:

You need data to flow between your frontend and backend—and between your composable commerce systems—like a firehose, not a playground water fountain with a mind of its own.

High data velocity is necessary no matter which frontend framework you use. Popular frameworks (Vue/Nuxt, React/Next, React/Gatsby) will have issues if you don't accurately account for scalability.

With monolithic systems, you'll likely never have an issue with data communication across different facets of your solution because everything shares the same vendor, language, and database. Monolithic systems are ACID compliant. ACID (atomicity, consistency, isolation, durability) functionality isn't necessarily guaranteed in microservice architecture. For example, with microservices, you're putting unparalleled performance above flawless data consistency, which introduces "eventual consistency" to your tech stack.

Eventual consistency refers to the lag time between systems. It means that communicating data from one system to another will not be instantaneous, but eventually, the lagging system will catch up to the other and accurately reflect any updates.

While a headless commerce solution can't eliminate eventual consistency issues completely, it can make them more tolerable. For instance, Nacelle's platform brings lag time down from hours or even days to minutes and seconds.

Layer of Abstraction:

As previously mentioned, without a data layer, you won't be able to achieve true independence and flexibility in your best-of-breed stack. Without a layer of abstraction, everything will be wired together and changes will require rewriting code across your backend instead of in one place.

Best-of-Breed Tech Stack Communication:

You may have the best-of-the-best tools on your backend, but they can't perform optimally if they can't communicate with your technology ecosystem at large. Your tools need to "speak the same language" and be able to seamlessly share data in order to positively impact your operation. For example, Nacelle acts as the data layer and unifies syntax across your tools in order to achieve desired composable commerce performance.

Unit Test Coverage:

Your headless commerce solution should improve developer workflow while alleviating common engineering pain points. Solutions that allow developers to work in Git Push benefit from improved testing environments and being able to have multiple people work on code without overriding colleagues' work. It will also act as a failsafe of sorts to ensure code isn't being pushed live prematurely.

Scalable Performance:

How will you handle traffic spikes without sacrificing webstore performance or crashing your site? One example we like to use is Twitter. Twitter seems like it's just a form with 160 characters—you could build that in an hour. Of course, it would never scale the right way.

Data flow is paramount and you need to get in the "headless-first" mindset from day one of your build to get this right. We use elastic serverless technology that can scale horizontally in milliseconds. Solutions that were not built to be headless-first from day one will have a very difficult time accounting for this infrastructure-level necessity.

An example of scalability in action is D2C fashion brand, Something Navy. The brand crashed the Nordstrom website four different times while working in collaboration before building its own webstore to handle these spikes in traffic. On the first day of launching their Nacellepowered headless PWA, Something Navy sold $1 million worth of clothing in 30 minutes without a single glitch in site performance. Since then, that time has decreased to about 7 minutes!

JavaScript SDKs:

JavaScript SDKs enable frontend functionality such as fetching data, processing checkouts, and triggering events. Like APIs, SDKs need to be highly performant. Nacelle's SDKs can integrate with the frontend framework of your choice or be used without a framework in a vanilla-js environment.

CMS & Content Modeling:

When going headless with a microservices architecture, adopting a headless CMS is almost always the first tech stack addition, even when merchants keep their eCommerce platform. Implementation details and planning around content modeling are vital.

Once you have a powerful content solution in your stack such as Contentful, Strapi, Sanity, or another great headless CMS on the market, your merchandising team and marketing team need to sit down with engineering to flesh out how it will be used, what frontend components will need to be built, and more. This must be done up front to maximize the success of your headless build.

nacelle

Build vs. Buy Decision Making Framework

“Build vs. buy” may sound polarizing, but as discussed above, there are happy mediums available such as going the “buy and build” route with a customizable headless commerce solution.

Your team ultimately needs to decide what makes the most sense given your unique needs and engineering bandwidth. Strategically maximizing your engineering resources will allow you to differentiate from competitors and delight customers. Deciding where to channel those finite in-house development resources is a top priority for eCommerce success.

As you make the build vs. buy headless commerce selection, consider these pillars of technical decision making.

Technical Debt:

It might take months or even years to feel the pain points of a poor tech stack decision, but failing to hypothesize the cost of choosing a system that doesn’t have longevity could be financially devastating to amend.

First, consider the repercussions of investing in a monolithic system or an opinionated “headless” solution with deep dependencies. Down the line, if your team decides it needs to adopt a composable commerce strategy, or you grow to a place where your engineering team wants to start building custom elements of your stack, you’ll need to escape the vendor-lock situation. This likely means risky (and costly) rip-and-replace migrations and potential data loss.

Implementing the right headless commerce solution first, before going best-of-breed, can make “migrations” safer, and you don’t have to change every system all at once. If you’re already on board with a best-of-breed microservices strategy, think about how the right headless commerce solution can support the composable commerce flexibility you’re trying to achieve.

The right architecture can promote agility for making changes at will. With everything wired together, not only is the process of keeping your tech stack lean going to be more diffcult, you’re going to pay for tools that aren’t serving you well for longer than you need to because you can’t replace them quickly or easily.

If you’re thinking about building a custom solution in-house or with an agency, also keep the cost of losing tribal knowledge in mind. If key people leave, how costly will it be to replace them and recover their knowledge?

nacelle

Build Costs:

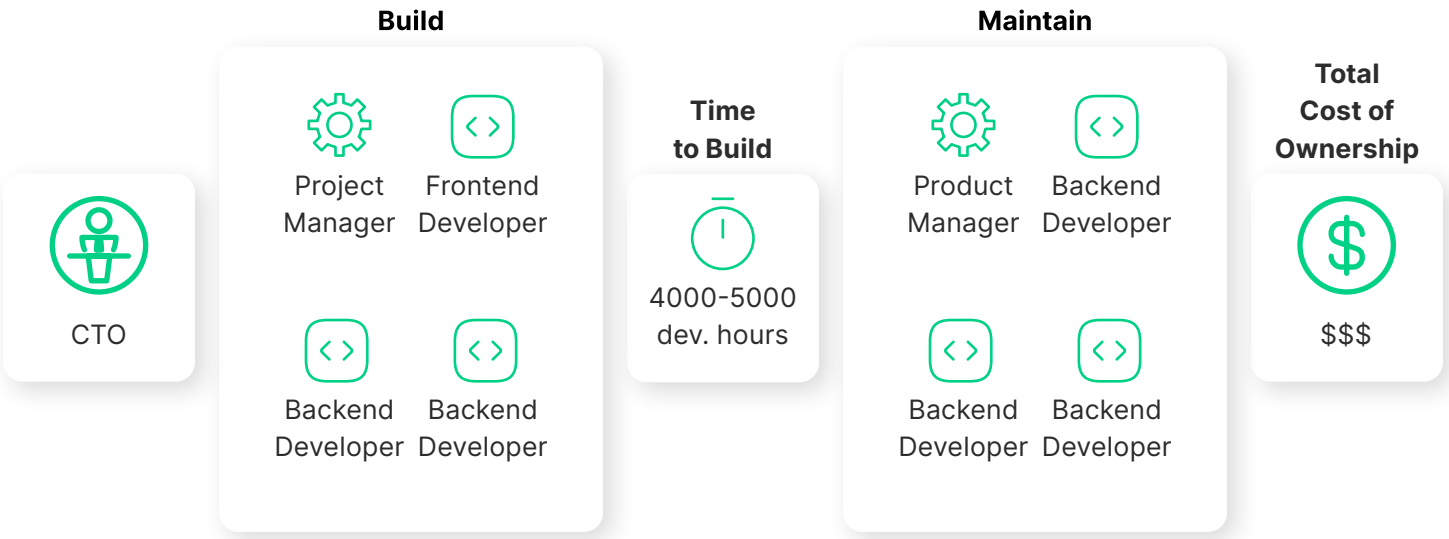
We estimate that the build alone of a custom headless commerce architecture costs 3,000 to 5,000 developer hours. That's a starting point. It does not account for scope creep and overcoming unforeseen challenges not anticipated by your build team, which is possible if they have little to no relevant headless build experience.

Meanwhile, there will be other demands on the development team because the show must go on for your current eCommerce store while you build your headless solution. This could impact dedicated focus to your headless project if it's not accounted for.

Ongoing Maintenance Costs:

Outside of the build itself, there will be ongoing maintenance costs. We estimate in-house teams will need at least one, possibly as many as three, dedicated developers plus DevOp resources for maintenance.

If you're buying, or buying and building, make sure the solution you choose has a managed service aspect so your in-house team doesn't get bogged down by maintenance.



Time to Market:

It's common for merchants to opt to buy a solution for the sake of time. Building in-house or with an agency typically takes 12 to 18 months to go live. Buying a headless commerce solution that you can build on top of takes 3 to 6 months on average.

nacelle

Conclusion

Whether your stack is mostly custom already, or you're preparing to go best-of-breed from a monolithic system, you need to consider all the facets of what it means to build vs. buy a headless commerce solution.

Nacelle embodies "buy and build." It's a "best of both worlds" scenario, because it has out-of-the-box functionality that your team can build on top of to customize your headless experience. It was also designed specifically for headless commerce from day one, so it's built to withstand the future of eCommerce with flexibility, scalability, and optionality.

Everything above and below the headless commerce "iceberg," referenced previously, has been accounted for at Nacelle. Our platform features a headless-first architecture including elastic serverless technology to achieve the benefits associated with headless commerce. Nacelle acts as the layer of abstraction to promote data velocity and best-of-breed tool independence. We take the ethos of decoupling a step further than just backend-frontend separation.

We have a team of headless commerce experts on staff and a managed service component to our offering, we also offer quick start guides to get you up and running quickly. We're not prescriptive. In fact our buy and build methodology encourages engineers to build something unique on top of our platform, without the pain points of an exclusive custom build endeavor.

About Us

Nacelle is a composable commerce platform provider that allows brands and retailers to syndicate commerce and content data to multiple heads, endpoints and channels by transforming, storing and reindexing data in real-time. With Nacelle, companies can future-proof their business by composing the commerce stack they want — giving them the agility needed to build unique and dynamic shopping experiences, while optimizing business operations for growth. Nacelle is a venture-backed company with over $70 million raised from institutional investors including Tiger Global, Index Ventures and iNovia. For more information, go to nacelle.com

Published 2023/01