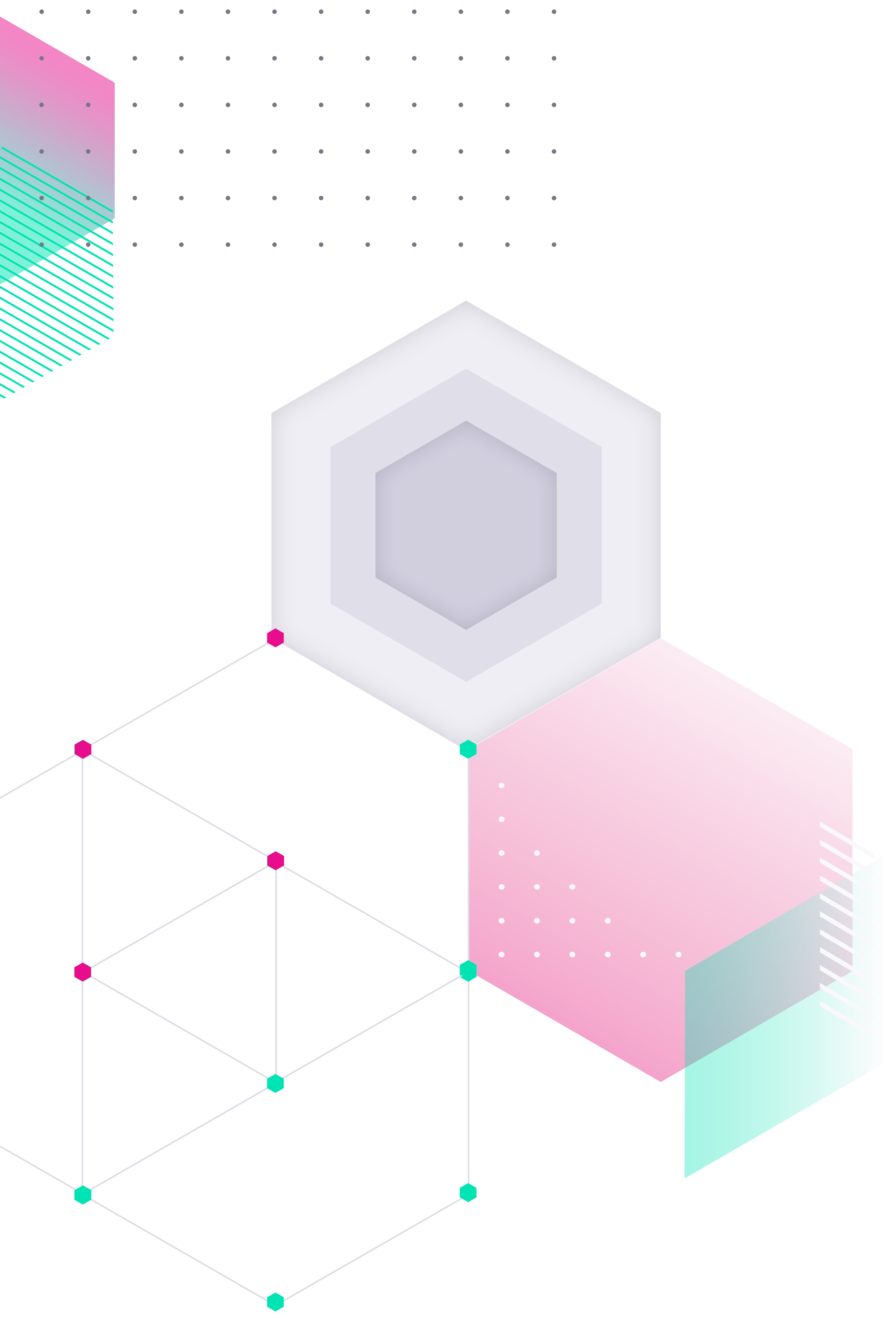
Amplience

2023 HEADLESS CMS BUYER'S GUIDE

Take the Pain Out of Choosing the Right CMS For Your Business



Contents

4 Key Terms and Definitions

6 How to Use This Guide

8 Technical Complexity

12 The Commerce Architecture Journey

20 Business Requirements

24 Commercial Models and Pricing

28 Next Steps

Key Terms and Definitions

We've compiled a few key terms and definitions you should know if you're evaluating a headless CMS. Let's jump in.

(CMS) CONTENT MANAGEMENT SYSTEM
Software that helps users create, manage and modify content for a website or application without the need for specialized technical knowledge.

HEADLESS
Where the 'head' of your CMS – i.e. a frontend presentation layer or user interface like a website, mobile app or smart device – has been separated from the backend functionality. So you can change any of the frontend elements individually without interfering with the backend.

APIS (APPLICATION PROGRAMMING INTERFACES)
A software interface that acts as an intermediary between two applications, allowing them to talk to each other in a predefined way.

CONTENT REPOSITORY
A database of digital content with an associated set of data management, search and access methods allowing application-independent access to the content. The content repository acts as the storage engine for larger applications like your CMS.

DIGITAL EXPERIENCE PLATFORM (DXP)
An emerging category of enterprise software for companies undergoing digital transformation, with the goal of improving the experience customers have online. DXPs can be a single product or a suite of applications working together.

CUSTOMER EXPERIENCE
Your customers' overall feeling about the experience they have with your brand. The sum of every interaction they have with your business, however large or small. It impacts everything from your reputation to your revenue and bottom line.

MACH
MACH stands for Microservices, API-First, Composable and Headless and represents a set of principles that offer a best-of-breed, modular approach to constructing enterprise software stacks.
Learn more: **https://amplience.com/partners/what-is-mach/**

COMPOSABLE
Because MACH architecture operates in the cloud, you have the power of infinite scale and flexibility no matter how you evolve your solution in the future. No more painstaking upgrades. No more sky-high management costs. And significantly less downtime risk.

MICROSERVICES:
The ability to break everything down into separate specialized capabilities and deploy to individual teams based on the problems they're trying to solve. This gives teams access to updates and new features faster, while building the solution your business really needs.

How to Use This Guide

Not sure where to start? Take a look at what's included in this guide below and jump straight to the section you need.

The market is crowded with hundreds of vendors all claiming to provide similar capabilities, functionality and benefits. So how do you cut through all the noise and choose the right CMS for your business? And how do you know if a headless CMS is the right approach for you?

We've guided hundreds of businesses through this buying process, even pointing them to a competitor when we know it's a better fit. And in our experience, there are four key questions that will lead you to make an informed decision:

1. What are your organization's current and future requirements?
2. Which type of CMS architecture is going to work for your business?
3. What are you trying to achieve with your CMS?
4. How do the vendor's pricing, costs and contracts work in practice?

4 Things to Think About Before You Buy a Headless CMS

1. TECHNICAL SOPHISTICATION

a. The Technical Complexity Scale
b. Development Resource Types
c. Developer Experience & Support Requirements

2. CMS ARCHITECTURE

a. MACH vs Traditional Architecture
  i. Headless vs Cyclops
  ii. Microservices vs Monoliths
  iii. Cloud-Native vs Self-Hosted
  iv. API-First vs API Bolt-On
b. Integration Approach & Partner Ecosystem
c. CMS Vendors Caught in the Transition - A Word of Warning

3. BUSINESS REQUIREMENTS

a. Use Cases
b. Jobs to be Done, Personas & Team Roles, Cookie Cutters vs Niche Providers
c. Out of the Box Feature Functionality
d. Do You Need a DAM (Digital Asset Management) System?

4. COMMERCIAL MODEL & PRICING

a. Vendor-Supported vs Open Source
b. Monthly vs Annual Contracts
c. Free Offerings
d. Implementation Costs
e. The Bottom Line

A note for CMS vendors: While we aim to provide the most up-to-date buyer's guide for CMS solutions, we understand that the market can move quickly. If any of the information is out of date or inaccurate, please do let us know.

Technical Complexity

What do we mean by technical complexity?

Essentially , it’s a measure of whether your organization requires a certain type of new technology (in this case a headless CMS). It is all about finding the right fit for you and your current situation. We break technical complexity down into three levels:

1. **Least Complex**
2. **Complex**
3. **Most Complex**

Let’s look at each of those in a little more detail...

Level 1 - Least Complex

You have little-to-no in-house technical resources and/or you have a light amount of content production. An out-of-the-box low or no-code CMS might be the best fit for you.

Level 2- Complex

You have a dedicated team of marketers, content producers and technical team members (either in-house or outsourced). You’ve likely adopted or are considering a platform like WordPress or a hybrid CMS like HubSpot. For small-to-medium retailers and eCommerce brands, you might also view your eCommerce platform as the CMS for your business, perhaps turning to Salesforce Commerce Cloud.

Level 3 - Most Complex

You have adopted, or aspire to adopt, agile best practices and have a well-defined content production workflow. You’re continuously deploying different types of content to a large audience and want to control the customer experience on the frontend at a granular level. You may have started experimenting with personalization and new channels beyond your website. A headless CMS is likely what you need.

Which Type of Development Team Do You Need?

When it comes to deploying and managing a headless CMS, should your development team be internal or external? Do you need one at all? The answer, of course, depends on your business strategy and level or technical complexity.

To help make the decision easier for you, we've broken down some of the main developer resource types below: what they are, how they work, the benefits, the drawbacks.

INTERNAL DEVELOPMENT TEAM

An internal team grants you more control over your CMS and its implementation, along with the frontend experiences you can create for your customers. The downside is increased costs and headcount, which may begin to distract from your organization's core objectives. Larger organizations can typically afford to build an internal team, but with the advent of MACH solutions it's possible for much smaller internal teams to deploy a high-performing, scalable, integrated CMS.

EXTERNAL DEVELOPMENT TEAM

An external team allows you control costs over time and potentially have better insight into project timelines, which means you can often get to market faster, with more flexibility to choose a specific implementation specialist with deep knowledge of a particular CMS. For a headless CMS, it's important to understand not just a developer's familiarity with a certain CMS but also the broader frontend technology stack that they'll implement – React or Vue.js, for example – and the pros and cons of each framework.

NO DEVELOPMENT TEAM

The advent of no-code and low-code platforms means you may not even need a development team if you're happy with the out-of-the-box functionality. The downside is you'll reach the limit of what's possible with this type of solution relatively quickly, which may hold you back from achieving your ambitions. Your business will eventually need to migrate to a more flexible CMS, which will require an external or internal development team.

Understanding Your Technical Team

Understanding your technical team's knowledge, capabilities and experience is another important factor. All of this will affect your time-to-market for any given solution and the ongoing maintenance or enhancement requirements.

If your technical team is highly proficient in a specific CMS, this may influence your decision. On the other hand, a team with little experience for a selected CMS may require onboarding, training and time with the platform to become efficient and meet deadlines. Your team's experience is important to understand. But it's also important to take a holistic view when selecting a CMS, ensuring that the past doesn't hold your business back from stepping into the future.

If your team is not proficient with a given CMS, it's important to evaluate the quality of the documentation, examples, guides, onboarding customer success and support provided. These elements are critical. A poor developer and user experience can lead to higher implementation and maintenance costs. A CMS that provides self-service resources may be cheaper from a pricing standpoint, but your organization may need additional support and training if problems arise.

ASSESSING YOUR EXISTING TECHNOLOGY

Understanding your existing technology stack can also help you quickly understand your organization's technical complexity. If your technology stack already includes other complimentary headless products that use cloud-native microservices with an API-first approach, you'll soon want to consider a headless CMS.

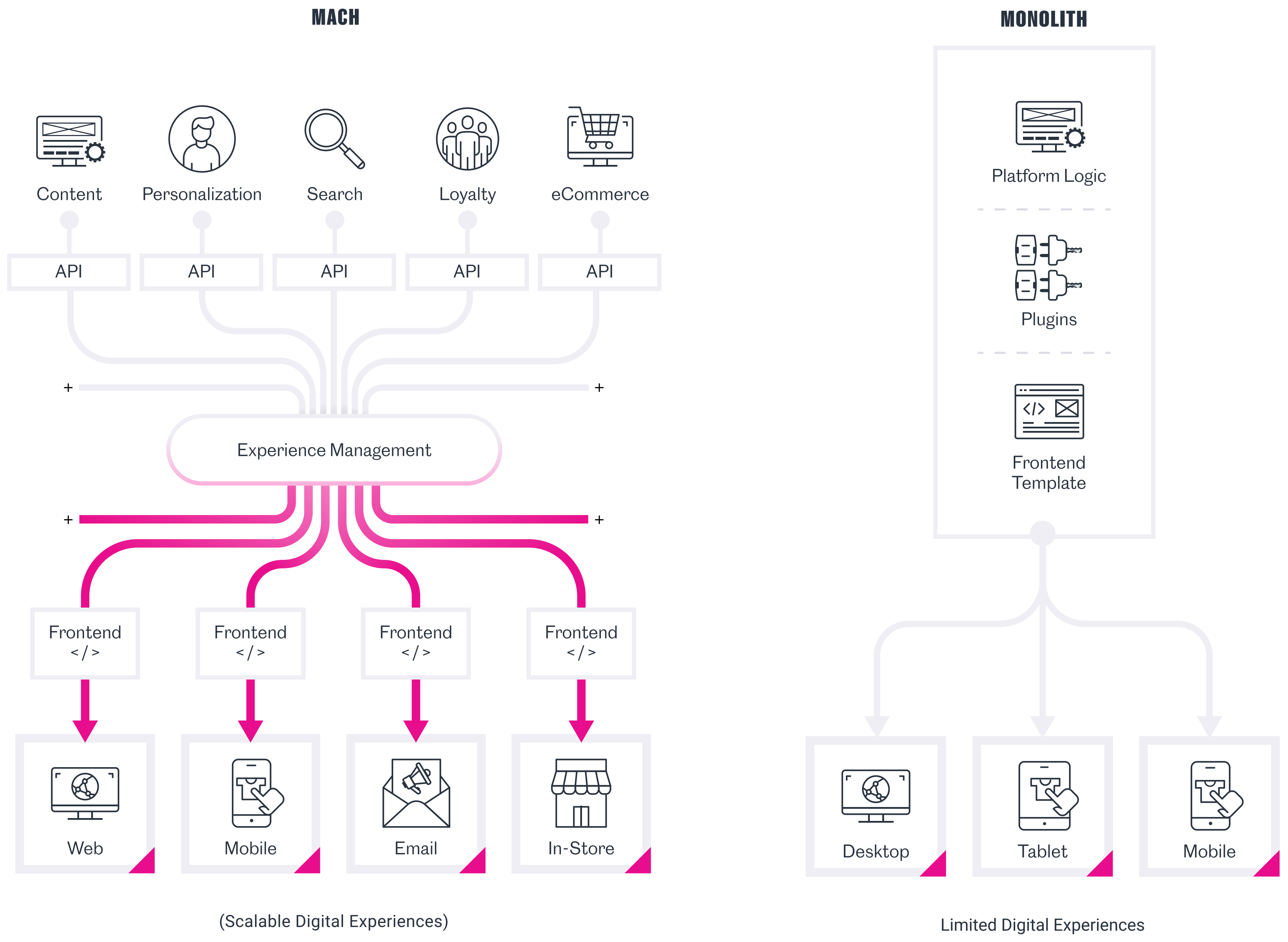
.

CMS Architecture

Mach vs. Traditional Architecture

Once you have considered your technical complexity, it's important to understand the different types of architectural principles that are available to your business. There are essentially two options to choose between, as summarized on this page.

Head to the next page for a much deeper dive into the world of CMS architecture.



MACH

Stands for microservices, API-first, cloud-native, and headless.

Check out the MACH Alliance if you want to learn more.

TRADITIONAL ARCHITECTURE

Typically has two or three of the following traits: 1) it's monolithic (i.e. built as a single unit), 2) APIs are bolted on as an afterthought, 3) it's self-hosted on your own servers.

Headless vs. Cyclops

Headless CMS: A backend-only CMS built from the ground up as a content repository that lets you access content via APIs and display it on any device. The term 'headless' comes from the concept of chopping the 'head' (the frontend, e.g., the website) off the 'body' (the backend, i.e. the content repository).

Cyclops CMS: A CMS where the body (platform logic) and head (frontend) are intrinsically linked to power a singular, rigid, template-driven website.

WHEN A HEADLESS ARCHITECTURE IS THE RIGHT CHOICE:

1. You want to create lightweight, high-performing, customer-first experiences
2. Your customer experience spans across one or multiple channels (your website and beyond)
3. You need to use modern programming languages, frameworks and technologies

WHEN A CYCLOPTIC ARCHITECTURE IS THE RIGHT CHOICE:

1. You're only operating a single website
2. You don't need to create a differentiated experience across multiple channels

Monolith vs. Microservices

Monolithic Architecture: A single-tiered software application where the user interface and data access code are combined into a single program from a single platform. A monolithic application is self-contained and independent from other computing applications. The design philosophy is that the application is not just responsible for one specific task but can perform every task needed to complete a particular function. Monolithic architecture is more rigid by nature, so it will become difficult, time-consuming and costly if you want to go beyond its out-of-the-box capabilities.

Microservice Architecture: A variant of the service-oriented architecture (SOA) structural style – arranges an application as a collection of services that are loosely coupled, independently deployable, organized around business capabilities and are highly maintainable and testable. This architecture is much more customizable. But you will need a development team to pair the CMS of your choice with the right frontend technology, language and framework.

WHEN MICROSERVICE ARCHITECTURE IS THE RIGHT CHOICE:

1. You want to create lightweight, high-performing, customer-first experiences
2. Your customer experience spans across one or multiple channels (your website and beyond)
3. You need to use modern programming languages, frameworks and technologies

WHEN A MONOLITHIC ARCHITECTURE IS THE RIGHT CHOICE:

1. You only have one online or "web" channel (e.g. a website)
2. You don't need much in the way of customized features
3. Your technical complexity level is 1 or 2
4. You just want a standard grid-based website up and running fast using templates

Cloud-Native vs. Self-Hosted

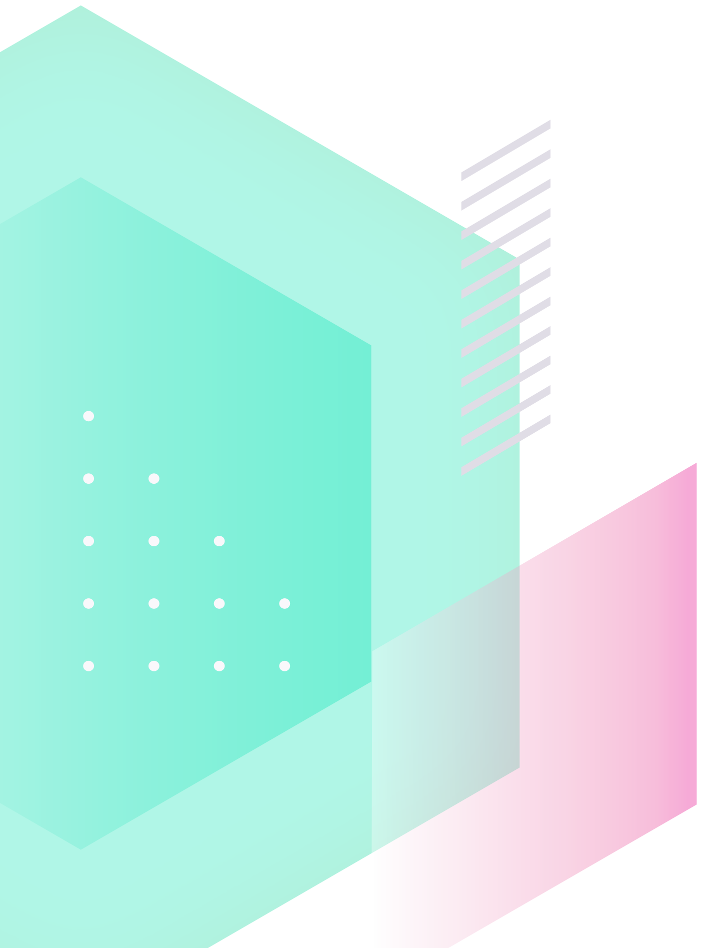
Cloud-native CMS: typically comes with lower maintenance costs and headaches as the software vendor maintains the product for you. Cloud-native providers are specialized in supporting their own products and services and typically you'll pay a monthly or annual subscription for the maintenance of the CMS in the cloud. Cloud-native vendors typically do not grant access to the source code but can offer reassurances for their customers by adding software escrow agreements into enterprise level contracts, which allows you to choose where your data will be hosted geographically.

Self-hosted CMS: typically installed onto your own servers and therefore comes with increased maintenance costs, potentially greater security risks and you'll need to keep the software up to date and operational with the latest features and functionality. Expect to maintain a team of development resources to help you achieve this. Some legal teams at larger corporations will often request ownership over the code or where the CMS is geographically located, which may lean your organization towards a self-hosted version of the CMS where you have access to the codebase.

API-First vs. API Bolt-On

API-first CMS: these are built from the ground up as a collection of API endpoints providing full coverage over the platform's features and functionality. Businesses that use an API-first platform have more flexibility as they can programmatically hook into any part of the CMS they need to request data from or send actions to. This typically makes it easier for development teams to follow best practices when it comes to cleanliness of code, ultimately reducing implementation and ongoing maintenance costs.

API bolt-on CMS: this approach essentially relegates APIs to second class citizen status within your solution, typically resulting in rigid functionality designed for a singular purpose and custom frameworks that require specialized developers. This creates longer implementation times, clunky work-arounds and increased costs, especially for more custom projects that try to use a bolt-on API and then have to fall back on the monolithic approach when the API capabilities are lacking.



The Impact on Integration and Maintenance

A traditional monolithic platform typically adopts a plugin-based approach to extend functionality beyond what is offered out of the box.

These plugins are built under the CMS platform's own framework. Typically there will be a marketplace where you can browse plugins and install them with one click.

This is great if you want to get a piece of functionality deployed quickly, but it can be hard to know who is responsible for maintaining the plugins' underlying code.

Plugins typically interact with and affect multiple areas of a CMS, including the backend logic and the frontend views or templates. Installing multiple plugins over time can lead to compatibility issues and bugs, which become increasingly difficult to isolate and fix.

Not All APIs Are Created Equal.
A legacy CMS will largely follow the bolt-on API approach (SAP, WordPress, Drupal) where as more modern CMS providers will provide an API-first approach.

This is due to plugins being granted large surface areas of access and control over various CMS components, leading to a tangled web that can become difficult and slow to support and maintain.

As mentioned earlier, MACH architecture consists of multiple microservices that communicate to each other, typically through lightweight server-less functions and user interface (UI) extensions. Server-less functions are small, discrete pieces of code that handle the data transformation and actions required between services via webhooks and queue-based systems (such as syncing product catalogue data to or from a CMS).

UI extensions, on the other hand, provide administrators with more powerful content types where the underlying data is stored in another dedicated microservice.

Most headless CMS solutions offer webhooks, server-less function examples and UI extensions to rapidly extend CMS capabilities and integrations with other services.

These extensions do require a developer to implement, support and maintain and are best suited for a level 3 technically complex organization. Although typically the maintenance and hosting costs for these extensions is very low.

A Word of Warning: Check Your CMS Vendor's Claims

For the last few years we've seen multiple traditional, monolithic CMS providers make bold claims that their solutions are in already headless, when in reality they have simply bolted APIs onto their existing products.

Some legacy CMS providers have either begun to acquire smaller headless CMSs or build their own from the ground up, which will take several years to get up to speed with their original offerings.

This has created a great deal of confusion for customers during their selection process. The pitches from these vendors typically contain a mismatch of legacy features and functions that aren't compatible with the newer headless functionality.

The MACH Alliance
A not-for-profit organization whose members are committed to providing pre-built integrations between each member organization, making it far quicker and easier to deploy a full solution that meets all your business needs. **Find out more**

For example, the admin experience and page builder are entirely separate from the headless APIs, giving your marketing team no control over the frontend experiences should you chose to go with the headless implementation from one of these vendors.

Transitioning From Traditional Architecture to MACH

If your organization has reached Level 1 or 2 technical complexity and is advancing to the next level, you'll be faced with two choices:

1. A full re-platform 'rip and replace'
2. A parallel approach, often referred to as 'dehydration' or 'strangulation'

Full 'rip and replace' re-platforming ideally allows your organization to have a clean break from the old platform over to the new platform, potentially cutting down ongoing maintenance costs that come from running multiple solutions. It's possible to keep your existing solutions and architecture in place, however, and retain the system you may have already invested heavily into over the past 5-10 years.

This parallel approach is easier if you're moving to a MACH architecture as you can tackle smaller pieces of the overall solution over time, rather than in one larger, longer project.

Business Requirements

What is Your RFP Process?

Many RFP processes begin by leveraging an RFP template from a favorable CMS vendor or crowd-sourced internally by assembling a list (or oftentimes a wish list) of must-have features for each stakeholder that will interact with or be affected by the choice of CMS implemented. While this is great to understand from a functional perspective, most RFPs end up being answered by vendors in the same way: most mature CMSs have the basic features and functionality available out of the box, but the differences begin to show themselves around more advanced functionality and integrations, specifically the work involved in solving these criteria.

Headless platforms will likely provide example integrations whereas monolithic platforms will provide installable plugins. Each of these come with pros and cons documented above in the section titled “Integration Approach & Partner Ecosystem”.

How Many Channels Are You Working With?

One of the main factors to consider when it comes to your business requirements is the number of channels you are looking to develop, support and manage on top of your selected CMS. Headless providers are a great choice if you’re looking to support multiple digital channels, touchpoints and devices beyond just simple websites – microsites, mobile, in-store or internet of things, for example. It’s far quicker, simpler and cheaper to roll out new channels with a consistent backend architecture.

A monolithic CMS may struggle to support additional devices and channels as it’s not designed from the ground up (API-first) to support these use cases. If your business only needs one website, you could easily achieve that with pretty much any CMS. But the type of experience you’re attempting to create might also influence your decision.

A monolithic, template-driven approach might work well for a simple, cookie-cutter, grid-based website. But if you want to break free from your competitors and create unique, differentiated customer experience, a headless CMS is what you need.

Teams, Roles, and Responsibilities

Mapping internal team structures, roles and responsibilities will play a vital role in working out your business requirements.

In MACH architecture, teams and individuals will likely specialize in specific workflows and processes and may need to navigate between a small number of different tools in addition to your CMS to complete a given task. A monolithic CMS, on the other hand, might enable your organization to work from within a single tool.

One example of specialization is the difference between a merchandiser and a marketer. The merchandiser will likely need a dedicated product information management (PIM) tool to enable them to manage large product catalogs, while a marketer will need tools that enable them to run more efficient marketing campaigns and launches.

You can also pinpoint pain points in your workflow processes and work on solutions to alleviate these bottlenecks. The user experience of your CMS administration dashboard will directly affect the level of effort required to complete a given task.

Many monolithic CMSs provide page builders like Hubspot, Salesforce Experience Builder, Webflow, Wix or Adobe Experience Manager, enabling you to create website pages under strict guidelines without the need for a developer.

Headless CMS providers are maturing in this regard and can also provide similar levels of control over the layout and structure of page content with inline or new tab preview capabilities.

Digital Asset Management (DAM)

WHAT IS A DAM?

A DAM gives your organization the control, portability, flexibility, access and reporting of digital assets between multiple parties and systems, whether that's inside your organization or with external organizations, customers and partners. DAM systems are focused on delivering the right content to the right people across all channels and devices, with the ability to track, monitor and measure engagement of these assets. A digital asset is more than just the media file. It also includes extra metadata and information about that asset that adds additional value to the file itself. Examples of metadata could be name, author, creation data, fees, etc.

Digital assets become infinitely more reusable when metadata is linked to them. A DAM provides a secure repository that facilitates the creation, organization, management, production, distribution, and potentially monetization of your company's digital assets. And you typically get advanced functionality like the ability to manipulate, transform, secure and process rich media files and their metadata.

WHY WOULD YOU NEED A DAM?

As your organization grows in size, the need for a DAM will become increasingly obvious, with tens of thousands or even millions of assets being created, stored and used across multiple teams, channels and systems.

While many smaller organizations can simply use their CMS to act as a simple DAM, over time this approach will become problematic as more people, teams and systems get involved and need to access to the same digital assets.
Using a CMS or file sharing service as a rudimentary DAM can become a bottleneck and silo inside of your organization, making it time-consuming and difficult for your teams to access the digital assets they need.

A proper DAM acts as a centralized hub that all teams have access to store and manage a growing digital asset library. This alleviates bottlenecks and the multiple silo problem, saving your organization time and money.

In essence, a DAM enables your business to increase the quality of your customer experience across all digital channels, tying together in-store and online shopping experiences with increasing levels of sophistication and personalization.

IS A DAM RIGHT FOR YOU?

If your organization is relatively small with only a handful of contributors, a file sharing service like Dropbox or Google Drive may suffice. You may also look to use your CMS if your organization is focused on a single website-driven channel.

You should keep in mind, however, that these assets will likely need to be migrated to a DAM if your company grows at some point in the future.

For larger organizations with multiple teams or those with a large quantity of digital assets, a DAM will be a necessity.

Commercial Model and Pricing

- **Vendor-supported model:** when software is developed, maintained and supported by a commercial entity, for a licensing fee, in which the source code is usually not accessible or modifiable.
- **Open-source model:** when software is developed through a collaborative effort, in which individuals can contribute to the final product, it's self-maintained and the software is freely available.

Which Commercial Model Is Right For You?

WHEN A VENDOR - SUPPORTED MODEL IS THE RIGHT CHOICE:

- You want complete support of your backend systems, scaling and maintenance to be handled by a third party
- You don't want your source code to be easily changed or be susceptible to security vulnerabilities over time

Examples of vendors that use this type of model include both MACH and monolithic providers such as: Amplience, Contentful, Contentstack, WebFlow, Hubspot, and SquareSpace. It's important to note that this vendor license supported model may cost more upfront for headless implementations, and you won't be able to fork out and customize the backend source code.

WHEN AN OPEN-SOURCE MODEL IS THE RIGHT CHOICE:

- You have a digital complexity of at least 2 and ideally 3.
- You want complete control over your source code for editing and collaboration.
- You want a community of developers constantly updating and helping with bug fixes.

Examples of open-source vendors include Wordpress and Drupal. It's important to note that although it is free to get started, ongoing support and maintenance costs can get expensive. Teams also need to take on the responsibility of supporting your open-source CMS implementation and integration with third-party applications. Robust integration projects can be quite challenging at the enterprise scale of the spectrum and often require a team of expensive certified developers to assemble all the resources required for a functioning digital experience and adhering to organizational security.

Free Trials & Tiers

Many vendor-supported models also provide a free offering in the form of either a free trial or free tier enabling you to try before you buy or build a proof of concept for your decision-making process. These plans are usually severely restricted in the functionality provided or the amount of usage or throughput that can be placed onto the CMS. Free tiers are typically aimed towards developers and hobby level projects. Amplience and Contentstack provide free trials that enable you to test the CMS and build a proof of concept over a given length of time. Contentful provides both a free tier for developers as well as trials of their enterprise version.

Monthly vs. Annual Contracts

Enterprises will typically sign multi-year annual contracts with most technology vendors, especially those that will become critical parts of their business operations, such as their CMS and eCommerce platform of choice. License-based vendors are largely incentivized to offer annual subscriptions to their customers as it binds both parties into a long-term relationship and shared interest in the success of the project.

If you're just starting up or you are a SMB where it's difficult to predict cash flow, you will likely be more comfortable with a monthly subscription-based pricing model or opensource model where you simply pay for the hosting after the initial implementation. Many of the smaller CMS providers offer this tier of pricing and typically start from around $10-50 a month on the low end.

IMPLEMENTATION COSTS

Outside of the platform, vendor, product or service costs, you will also need to factor in the implementation costs of your chosen platform. This may vary greatly depending on the size of your project, the size of your team, the location of your team and their experience level. Using an existing internal team or hiring one may be an attractive approach to take as you'll be directly in control of the process. If you're having to build a team, this may take some time to assemble, and you'll need not just developers but also project managers, experienced solution architects and UX experts on hand to see your solution through to completion.

Alternatively, you could have an external solution integrator (SI) or agency implement your solution, but this may cost a premium with additional rates for not only the developers, project managers and other members of staff on top of a percentage premium. This premium rate typically comes with the peace of mind that the project will be managed and implemented by an experienced team, lowering risk.

When working with an SI, it's important to ensure close alignment with estimated project costs, implementation time frames, scope, and expectations. If you take the external implementation approach, it's worthwhile putting your project out to tender and having a handful of solution implementers bid on your project so you can benchmark each offering. The cheapest implementor may not always be the best choice and it's important to weigh the pros and cons of each implementer and look at their past clients/experiences. A blended approach of a small internal team complemented by the support of an external SI is another popular choice.

The great thing here is that you'll have a larger team supporting the initial implementation, speeding up time to market, while also training up a smaller internal team to maintain your implementation post live. Need recommendations on experienced headless solution integrators? We'll happily assist you by providing a short-list of our top picks – please do get in touch.

RESOURCES AND TIME APPROACH

This method is ideal for projects that are difficult to measure due to the extensive size, or where milestones may change along the way. It factors time estimates and the number of resources used into account.

While this approach is typically the most flexible for either party, there can be risks for both sides with feature creep, escalating costs and timescales. At this point, it's likely best to adopt a retainer-based approach for ongoing iterations.

FIXED PRICE WITH A BLENDED RATE APPROACH

An estimated project cost with a fixed price and blended rate is another popular way for SIs to charge clients. This method usually incorporates the average cost of the whole team (combining individual rates and dividing them by the number of team members).

A fixed price works well when there is a straightforward project with fewer milestones. A small micro-site, for example, or the creation of a specific landing page.

BUDGET DEVELOPMENT APPROACH

A budget development approach takes varying options into account that can be added or dismissed during the development stage depending on the client's budget for each part of the process. This method works well for large-scale projects which involve many aspects and moving components.

Since headless CMS implementations can be extensive highly iterative projects, using this approach allows both the agency and client to consider a baseline budget in addition to increases or decreases for additional components over time.

The Bottom Line - Your Budget

Ultimately, price and the wider technology and marketing budget will play a large factor in your decision-making process when it comes to selecting the right CMS for your business.

There is no right or wrong answer when it comes to budget for a new CMS, as every business is different. Knowing even a ballpark figure will help direct you to different ends of the CMS market.

Next Steps

One of the first and most important considerations is to understand your company's technical complexity or whether your business strategy requires you to mature along the scale outlined below from where you are today.

We hope you found our CMS Buyer's Guide insightful. Now that you're equipped with the top considerations for evaluating any CMS, we want to make sure you have all the information you need to understand if Amplience is the right solution for your business.

Please feel free to book a meeting with one of our experts if you're interested in learning more about:

- Specific details on how we compare against other vendors you may be evaluating, such as Contentful and ContentStack
- How to start a trial or POC
- Our pricing options
- Headless solution architecture advice
- Solution implementer recommendations

We'd be happy to jump on a call and answer all of your questions.

Amplience is a headless content and experience platform for forwarding thinking brands and retailers. We empower merchants, marketers and developers around the globe to create incredible multi-channel experiences faster from the most visual CMS and DAM platform on the market today. Over 400 of the world's leading brands including Ulta Beauty, Crate and Barrel and Gap rely on Amplience to create differentiated experiences that drive deeper, more valuable customer relationships.

Amplience supports the industry's transition to Microservice, API-first, Cloud and Headless (MACH) technologies, is MACH certified and an executive member of the MACH Alliance.

Amplience

Amplience UK
6th Floor Tower House
10 Southampton Street
London, WC2E7HA, UK
+44 20 742 6990

Amplience USA
234 Fifth Avenue
New York, NY
10001, USA
+1 917 410 7189

contact@amplience.com
www.amplience.com