WHITE PAPER

The World of Headless CMS

Everything You Need to Know About Headless Content Management

CRAFTER SOFTWARE

Table of Contents

Introduction

Today, almost all markets are continuously changing—not only because of innovation from within each industry, but also due to the innovation going on outside of them. For example, **OC&C Strategy Consultants** forecasts that 55 percent of US households will own a smart device, a sharp rise from the 13 percent observed in 2017. Furthermore, the IoT market is expected to be worth **$520 billion by 2021.**

New development technologies and processes are enabling a range of innovative digital applications, such as augmented/virtual reality (AR/VR), single page applications (SPAs), AI-enabled video experiences, digital signage, cross-platform mobile apps and more. Meanwhile, new devices are rapidly changing consumer behavior and their needs. Consumers are no longer getting their content just from websites and mobile apps anymore, they prefer to engage with various devices such as Amazon Echo, Apple Watch, and the Samsung smart refrigerator.

As consumers adopt these new connected devices, they are increasingly wanting to have unique and personalized experiences on them that are aware of who they are and other interactions they have had, including on other channels.

For brands to be able to deliver to these channels, they have to adapt their web content in order for it to reach out to more touchpoints while avoiding content duplication, high technology costs, or investing in an agency for each different department that has a customer facing channel.

Thankfully there is one predominant solution that is enabling brands to successfully deliver content to a plethora of digital apps, devices and channels—and that is a headless CMS.

A Look Back at Legacy CMS

When it comes to the humble web content management system (CMS), we immediately think of the traditional CMS platforms: WordPress and Drupal (for PHP/MySQL developers), Adobe and Oracle (for Java/JCR developers); Sitecore and Sharepoint (for Microsoft .NET/MSSQL developers). These platforms were huge in the early 2000s thanks to a user-friendly interface that came out-of-the-box, which allowed non-technical users to build and publish websites on their own, which was, back in that time, a huge step in empowering content authors.

But despite some of these traditional CMS platforms powering over 30 percent of the internet, organizations have noticed that these platforms are not suitable for today's dynamic technological landscape that's continuously disrupting consumer behavior. These systems are built on monolithic architectures that assume specific use cases and specific technologies. Developers have to fight the CMS at every step, compromising with respect to their implementations, tools and development process when trying to address modern use cases. Further, these monolithic architectures are difficult to maintain and scale from an operational perspective. This is a huge competitive disadvantage, especially when users expect a page to load in less than 2 seconds. More importantly, these CMS platforms are page-based and simply do not have the support to quickly deliver content to other channels. They could only push their content to page-oriented, web-based applications, and that's it.

Additionally, traditional CMS platforms often struggle to deliver personalized content. Many of them are only able to produce static content and the more dynamic systems are often so slow and difficult to scale that they require Content Distribution Networks (CDN) technology. CDNs are costly and inherently limit the amount of dynamic content and behavior you can achieve. In a time where 81 percent of consumers want brands to make the effort to know them better so they can receive personalized recommendations as well as contextually relevant content, users of traditional CMS platforms are certainly falling behind.

Moreover, if you're operating multiple sites, the content in a traditional CMS will be siloed, resulting in content duplication and inconsistencies as you add more channels.

How a Headless CMS Is Different

To address the issues presented by traditional CMS platforms, many organizations have turned their attention to investing in a headless CMS platform. The main difference between a headless CMS and a traditional CMS is that headless comes as a backend-only solution. There's no built-in front-end delivery layer. Instead, the content is decoupled from any template or front-end content delivery layer and all the content is stored in a single repository.

THE ADVANTAGES OF A HEADLESS CMS

To explain how a headless CMS architecture (see Figure 1) can address the problems of a traditional CMS, we highlight the key advantages:

Deliver Content to Any Device

One of the fundamental advantages of a headless CMS is that it can supply content to literally any device, channel, touchpoint or digital experience app. Developers leverage APIs to retrieve content from a central repository and

and distribute it to not only websites and mobile apps, but also to devices like smart speakers, smartwatches, AR/VR headsets, digital kiosks, video players, and Alexa-enabled connected cars.

Furthermore, a headless CMS is a future-proof solution, meaning you can deliver your content to devices that are yet to enter the market. When the first wave of IoT devices entered the market, many brands were not prepared. Plus, with new channels like VR headsets and smart vehicles entering the market, it is difficult to predict what device will come next. But with a headless CMS, you can distribute content to any channel, even those that don't already exist yet.

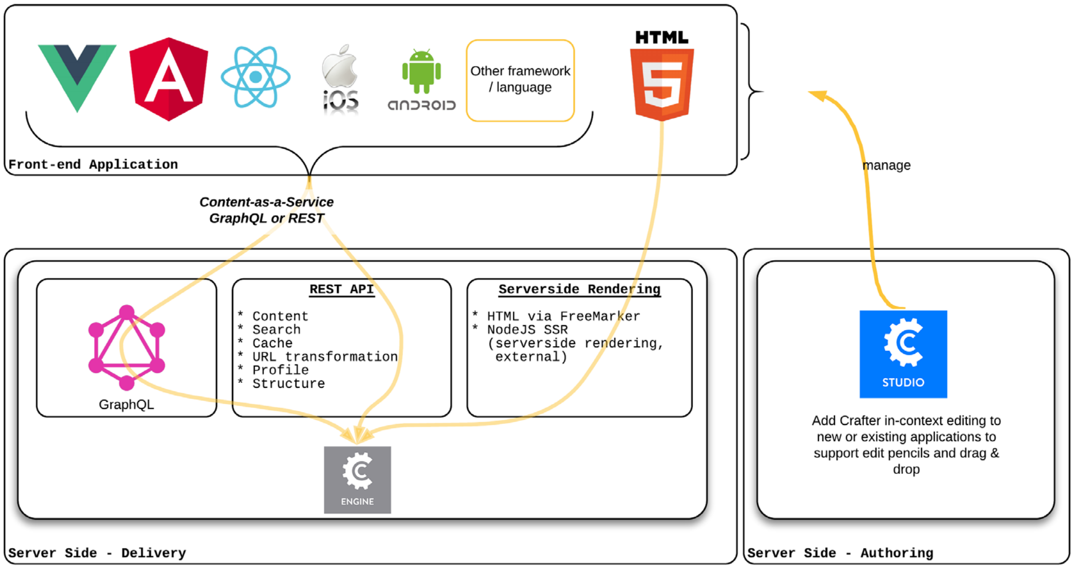

Figure 1. Traditional vs. headless architecture.

Reuse The Same Content

With all the content stored in one central location, a headless CMS is able to store all your content assets in a modular fashion, making it readily available to be delivered to any channel. With the headless model, content authors are able to create their content once and reuse it anywhere. And they're also able to update a piece of content in a single location, and it will update all the channels displaying that particular piece of content.

This prevents content authors from having to duplicate content and allows them to achieve a consistent brand message since the channels are retrieving content from a single source of truth.

More Scope to Innovate

Headless CMS don't come with a built-in front-end content delivery layer, and this provides developers with the freedom and flexibility to create their own front-end applications using the framework (i.e. Angular, React, Vue) and programming language (i.e. JavaScript, HTML, Java, C++, Go, Python, PHP) of their choice and then integrate it into the headless CMS via API. It also allows them to create SPAs and progressive web apps, which can provide a better user experience thanks to its faster load time.

No longer does it matter whether your organization is a "PHP shop", or "Java shop", or ".NET shop"; headless CMS platforms allow you to use whatever programming technology best fits your needs. Besides, with most modern front-end development frameworks (e.g., Angular, React, Vue) based on Javascript, the previous turf wars between PHP/JAVA/.NET are now mostly irrelevant.

And with emerging use cases such as Alexa Skills, smart wearables, AR/VR and mixed reality, digital signage, among many more, a headless CMS enables brands to reach out to these markets with new and exciting innovative applications, giving them the competitive edge.

THE LIMITATIONS OF A HEADLESS CMS

However, while a headless CMS has its advantages, it does have its limitations which can leave non-technical users such as marketers and other content authors in the dark.

Lack of Content Authoring Tools and Marketing Features

While the absence of the front-end delivery layer in the CMS may be a benefit for developers, it is the opposite for marketers. From a marketer perspective, a typical headless CMS only allows marketers to create and manage content through simple content entry forms—it doesn't allow them to fully create experiences on their own.

Coming back to the traditional CMS, these platforms became immensely popular since it provided the features and authoring tools to create web-based experiences with little or no technical assistance. In a way, a headless CMS, for content authors, feels like a huge step back, taking away the empowerment they had once enjoyed with a traditional CMS (see Figure 2).

For any content that content authors produce, they won't be able to preview the content or even tweak its layout without first publishing the content so that it can be consumed and tested on the external app. This is a far cry from the in-context editing, preview and drag-and-drop functionality they enjoyed with traditional CMS platforms.

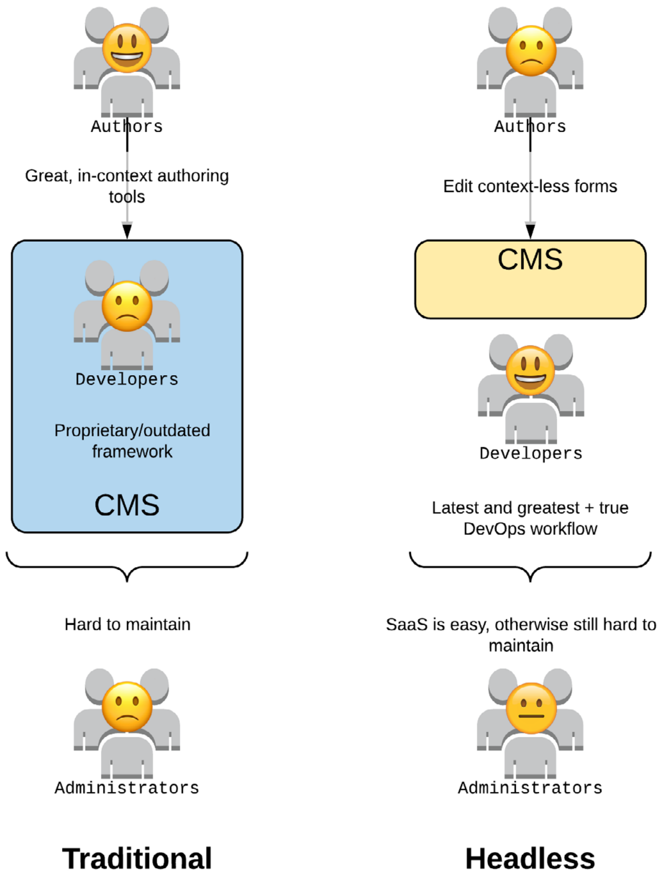
Authors
Great, in-context authoring tools
Developers
Proprietary/outdated framework
CMS
Hard to maintain
Administrators
Traditional
Authors
Edit context-less forms
CMS
Developers
Latest and greatest + true DevOps workflow
SaaS is easy, otherwise still hard to maintain
Administrators
Headless

Figure 2. Headless CMS still has some limitations.

Further, most headless systems are too simple and don't allow content authors to assemble new experiences built with pre-existing functionality. Instead, for the sake of creating experiences, content authors have to heavily rely on their IT team to develop every aspect of their experiences from scratch, leading us on to the next major limitation.

High Total Cost of Ownership

By overly relying on IT teams, the pace of innovation can actually slow. Since developers must create every aspect of the front-end delivery layer from scratch, they can get caught up in purely design and other presentation-level efforts which takes away from dedicated time to work on more fundamental solutions that truly push the business forward, slowing time-to-market.

> Consumers increasingly want to have unique and personalized experiences on devices that are aware of who they are and other interactions that they have had on them, including on other channels.

Headless+: Old School and New School Unite

Based on the limitations of a pure headless CMS, it appears there is a bit of a conundrum in the headless content management space.

On one hand, there is the traditional CMS that provides tools which empower non-technical users but frustrates developers, and limits the types of channels that can be targeted with personalized content.

And on the other hand, we have the headless CMS which developers love since they can use their preferred tools and frameworks and allows brands to deliver content to any device or touchpoint. However, a headless CMS does not provide the authoring interface that marketers need to create experiences on their own.

The solution to this conundrum is headless+, which combines the perks of headless CMS and traditional CMS together (see Figure 3).

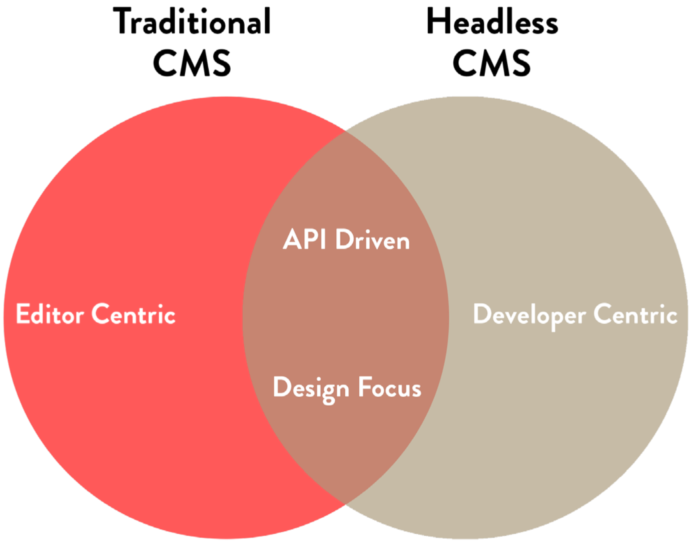

Figure 3. Headless+ combines the best of both.

A headless+ CMS essentially integrates a headless CMS with the developer supplied front-end content delivery layer in a way that supports an authoring experience that content authors love (see Figure 4). Unlike a traditional CMS, this approach is not tightly bound to a specific front-end technology.

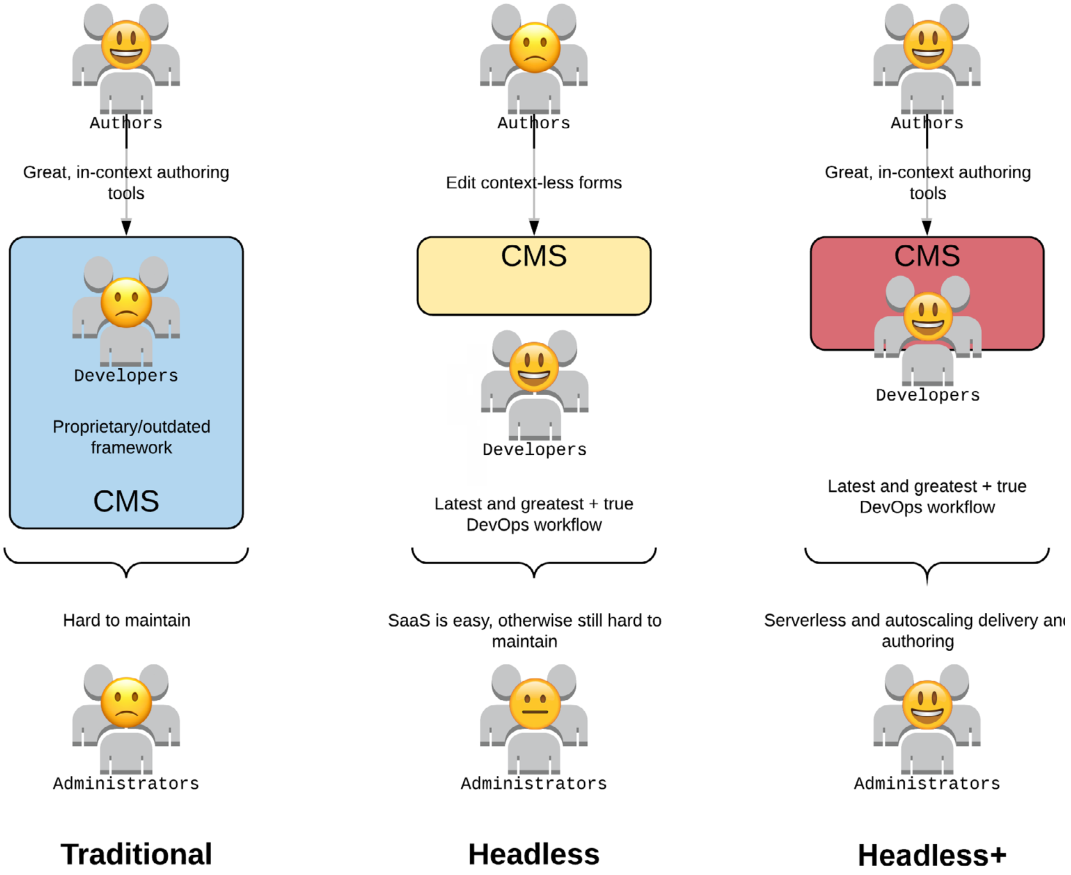
Authors
Great, in-context authoring tools
Developers
Proprietary/outdated framework
CMS
Hard to maintain
Administrators
Traditional
Authors
Edit context-less forms
CMS
Developers
Latest and greatest + true DevOps workflow
SaaS is easy, otherwise still hard to maintain
Administrators
Headless
Authors
Great, in-context authoring tools
CMS
Developers
Latest and greatest + true DevOps workflow
Serverless and autoscaling delivery an authoring
Administrators
Headless+

Figure 4. Headless+ satisfies both developers and content authors.

This model allows content authors to have access to a front-end delivery layer to support seamless preview and in-context editing within the headless environment, enabling them to more easily create experiences like SPAs, websites, landing pages, and microsites on their own, with little to no technical experience. And the API-driven environment of the headless+ allows developers to extract content from the content repository and push it to any device or touchpoint via a number of RESTful

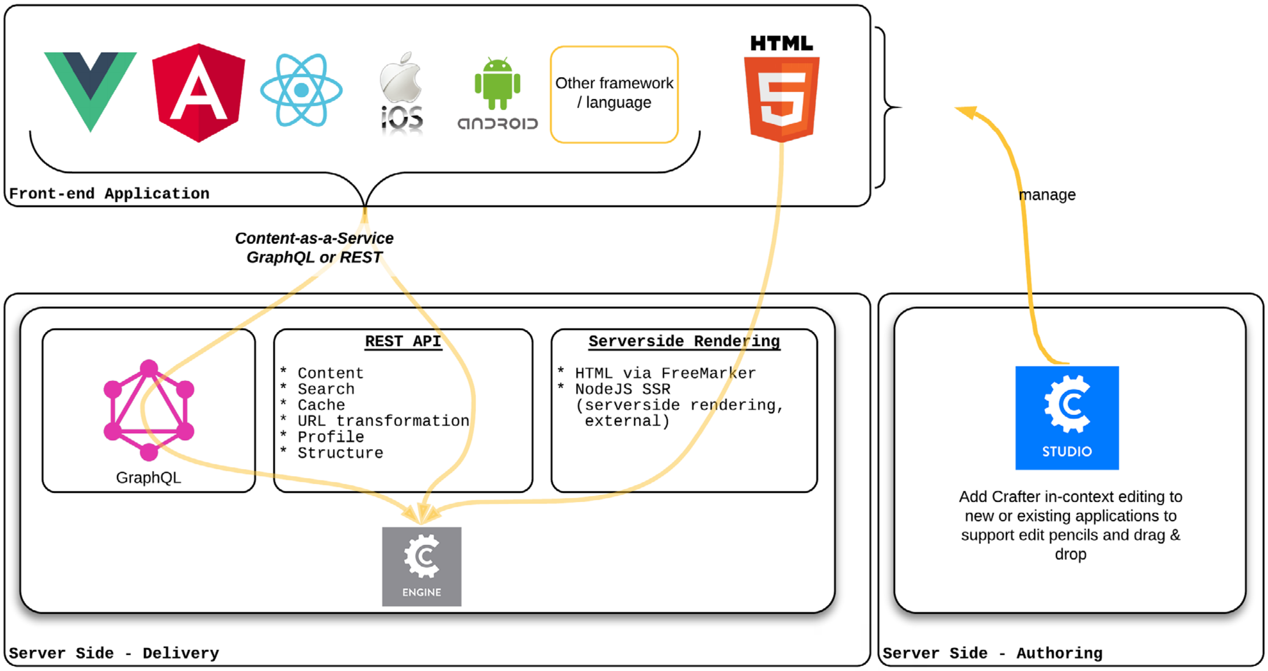

Figure 5. Crafter CMS provides complete developer flexibility, combined with full featured content authoring.

options that should include native GraphQL server support, a native REST API, and ideally the ability for developers to script their own RESTful APIs (see Figure 5).

A headless+ CMS, such as Crafter CMS, provides a digital experience platform that satisfies both content authors and software developers. Additionally, since headless+ solutions empower the marketer and other content authors, it allows them to deliver their content at a much faster time-to-market, while reducing the pressure on IT and reducing the overall total cost of ownership.

As an example of a real-world headless+ CMS use case, Marriott International switched from Microsoft to a **headless+ platform powered by Crafter CMS**. In the space of 12 months, they saw their TCO reduced and were able to easily manage their content in the central repository that now powers their global intranet, AR/VR experiences on hotel properties, and employee-facing mobile apps and digital signage.

Go Beyond Headless to Digitally Innovate with DevOps

In a live presentation given at DeveloperWeek in San Francisco, Russ Danner, Vice President of Products at Crafter Software, spoke about the exponential growth and power of technology, and the demand for digital innovation in user experience and the importance of IT teams to embrace DevOps practices. Nowhere is this more needed than in the digital experience space due to the rapid pace at which new channels are introduced. Because headless CMS platforms separate the front-end from the CMS, they enable a nearly infinite amount of flexibility in terms of development tools and DevOps practices. It sounds like a match made in heaven.

But there is a catch. DevOps is all about redefining roles and leveraging modern tools and process to create harmony and efficiency between developers and operations in traditional software applications. With digital experience-based innovation there's a third constituent with their own work product, namely, content authors and the content itself which are not covered in the DevOps equation. This creates friction that manifests itself in the form of code and content freezes and long difficult deployments that impact the efficiency of both content authors and software developers alike.

DevContentOps™ aligns the workflow, tools and process of developers, content authors and IT operations in a way that allows each to do their jobs collaboratively without friction. Developers and content authors do not have to deal with code and content freezes and bottlenecks.

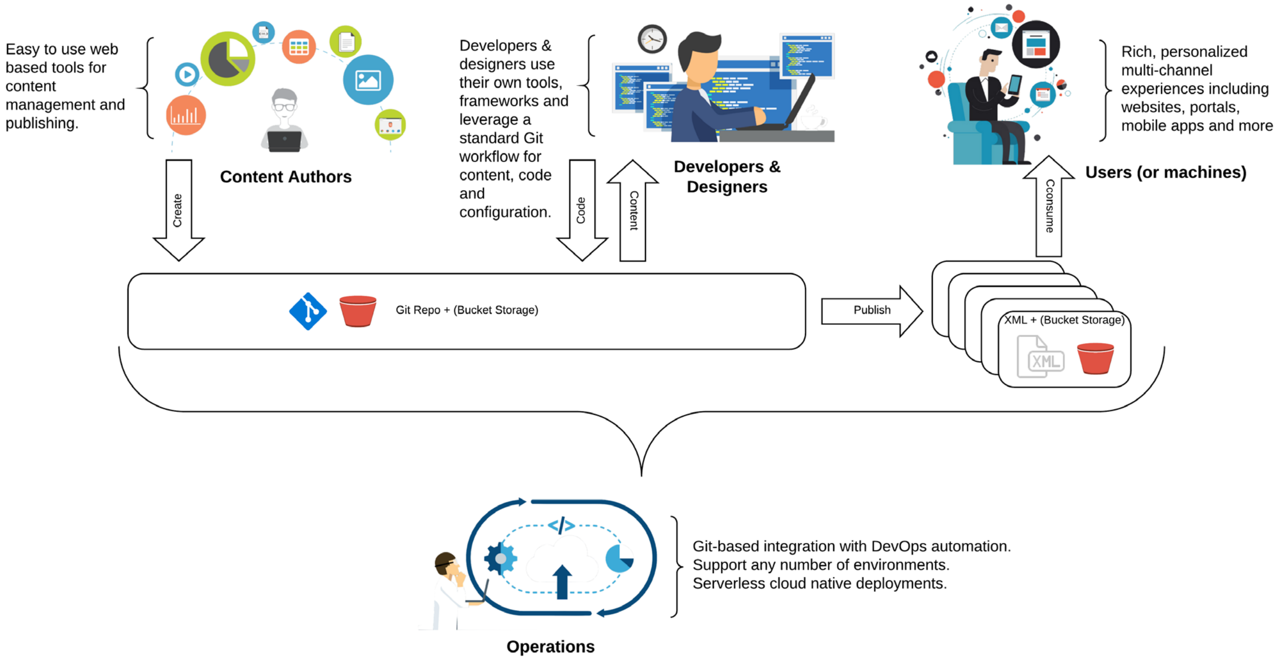
Easy to use web based tools for content management and publishing.
Content Authors
Create
Developers & designers use their own tools, frameworks and leverage a standard Git workflow for content, code and configuration.
Code
Content
Developers & Designers
Rich, personalized multi-channel experiences including websites, portals, mobile apps and more
Users (or machines)
Cconsume
Git Repo + (Bucket Storage)
Publish
XML + (Bucket Storage)
XML
Git-based integration with DevOps automation.
Support any number of environments.
Serverless cloud native deployments.
Operations

Figure 6. Crafter CMS enables DevContentOps™

When developing applications and sites that consume content it should be obvious that in most cases, the content is just as important as the code. Further, there is a kind of relationship or contract between content and code. To think that software developers can work completely independent of the content authors is naive. Developers have multiple environments to support their process that needs to be updated with content from authors. Moreover, content authors are often involved in various stages of development to provide feedback and user acceptance sign off. Content authors need to be part of the developers plus operations mix to see the full agile benefits of DevOps applied to digital experience applications. This is where Crafter's unique support for a DevContentOps™ process extends the basic benefits of a headless+ CMS into the realm of true enterprise innovation (see Figure 6).

This is where both traditional CMS and even the newest headless CMS solutions fall short: they don't integrate content into the DevOps process in a seamless fashion. In other words, they don't support a DevContentOps™ process.

The DevContentOps™ process aligns the workflow, tools and practices of developers, content authors, and IT operations in a way that allows each to do their jobs collaboratively without friction. Content may move backwards from production

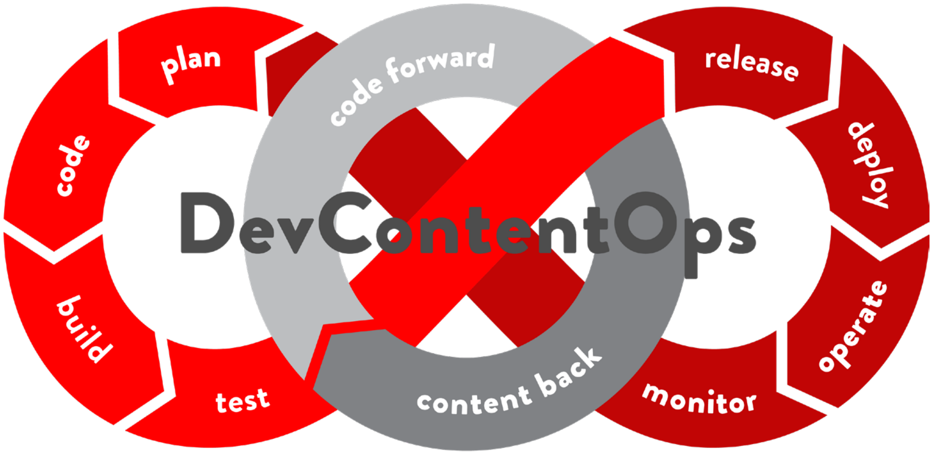

Figure 7. DevContentOps™ drives true enterprise innovation.

to lower environments "at the push of a button" without interrupting content authoring. New code functionality created by software developers may move up through the testing and certification environments with ease. Deployments are dramatically simplified. Developers and content authors do not have to deal with code and content freezes and bottlenecks.

Most systems, whether traditional, headless or headless+, do not (and cannot) support DevContentOps™. Why? Because nearly all CMS platforms are constructed atop either SQL or NoSQL databases, or JCR repositories that have at best, simplistic versioning systems and no ability to sync. Updating environments with new content or code requires time consuming and cumbersome export/import processes that impact the business and developers with content freezes and outages. The underlying storage mechanism and fundamental architecture is to blame.

What's the solution? Technical innovation is required to support a tighter workflow of content and code within the development process. Rather than rely on a traditional approach like a database, Crafter CMS incorporates technology that already has had a huge impact on DevOps.

Crafter uses Git as its repository, providing a truly distributed approach to code and content management. Moreover, Crafter CMS's implementation of its Git-based repository uniquely supports DevContentOps™. It has a time-machine like versioning system that enables sync that makes moving code forward from development and testing processes out to production simple. Content can easily be moved back. There are no content freezes or development work interruptions required to update an environment. Software developers, IT operations and content authors are finally able to collaborate with one another without stepping on each others toes. Furthermore the capability relies on simple, well known, well understood Git mechanics. It’s extremely powerful but what’s more is that it natively plugs in to your automation and monitoring frameworks and toolchain.

Start Innovating with Headless+ and DevContentOps

Customer experience leads to loyalty, and customer loyalty leads to healthy and profitable growth. Today, technology is driving customer expectation. The demand for new, easier ways to connect and engage with products, services and other devices in a seamless, targeted fashion is only going to grow. Astute, customer-focused companies are leveraging innovation as a key competitive advantage. Those who innovate the fastest and in the most scalable fashion will win.

Traditional CMS platforms cannot support all the new digital apps and channels, and are quickly becoming obsolete. Headless CMS architecture solves the multi-channel issue and offers significant developer advantages, but leaves the content authors without proper tooling. Headless+ CMS platforms blend the authoring tools that content editors had with traditional CMS with the flexibility and distribution of headless.

Crafter’s unique DevContentOps™ process is the next step. Many of the benefits of DevOps are wasted if there is friction between content authoring and software development personnel and process. DevContentOps™ support brings these teams together through modern processes and repository technology so that they can collaborate together without friction while maintaining optimal tooling and efficiency for their role-specific tasks.

Getting Started

LOOKING TO LEARN MORE?

- Video: **Introducing Crafter CMS 3.1**
- Download Crafter CMS: **Download**
- Launch an AWS Instance of Crafter CMS: **Launch AMI**
- Request a Trial: **Contact us**

ABOUT CRAFTER SOFTWARE

Crafter Software is on a mission to replace the broken paradigm of traditional content management, and to usher in a new era of fast, agile and easier development of innovative digital experiences. Our flagship product, Crafter CMS, is amazing for developers, easy for content authors, and fantastic for DevOps. We build our software solutions on the foundation of open source, transparency, robust architecture, high performance, superior quality and outstanding customer support. Available on premise in the enterprise or SaaS in the cloud.

Learn more at www.craftersoftware.com and wwww.craftercms.org.

LEARN MORE

CRAFTER SOFTWARE

www.craftersoftware.com

✉ info@craftersoftware.com

☎ +1.703.234.7744