



Blueprint for the modern website: an insider's guide to deploying content infrastructure

Contents

Introduction	3
Inside the Contentful platform	4
Getting started: content modeling sets the foundation	7
What is content modeling?	
Inside Contentful's content modeling toolbox	
Creating your content model in a few easy steps	
Extending your content model in contentful	
Getting to production: integrating with your delivery pipeline	19
Shipping content continuously	
Conclusion	29



Introduction

Websites have long been viewed as large, one-time projects. They are often only revamped every few years with minimal upkeep in the interim, resulting in a recipe for falling behind as technology and customer behavior rapidly shifts and evolves.

Now, product managers and other company stakeholders are applying the same principles to their website as they do to software, taking an agile approach that supports continuous improvement and adapts to ever-changing external forces.

The key to a modern website strategy is highly adaptable content infrastructure—web teams can use flexible APIs and serverless architecture to ship faster, launch timely campaigns, and never get bogged down by the limitations of their CMS.

This white paper will walk you through exactly what a content platform can do for your business:

1

Learn how Contentful's powerful APIs allows developers and content creators to deliver website improvements faster and more effectively.

2

Walk through how to build and manage your own content infrastructure using Contentful.

3

Understand how content modeling can create structured chunks of content that act exactly as you need them to on each device, platform, and browser.

In each section, we provide links to relevant content that will help you research concepts further.



Inside the Contentful platform

At its core, the Contentful platform offers four powerful REST APIs (and soon GraphQL) that are fully decoupled to ensure a more resilient service. These stateless APIs produce compact JSON payloads that give developers full programmatic control over content, assets and translations.



Content Management API – allows developers to create or update content and content models programmatically



Content Delivery API – delivers published content to applications and keeps content synced and up to date; backed by multiple CDNs



Content Preview API – allows content creators to view draft content in staging environments



Images API – retrieves and dynamically manipulates images on the fly, such as cropping, resizing or compression

These APIs also power an extensible web app for content creators. Your teams can extend the UI with your own widgets, or use ready-made features such as localization and roles/permissions to build a custom editorial workflow.



Markdown editor – allows editors to create and update content



Media editor – allows editors to edit images as they embed them



UI extensions – allow teams to integrate third-party services for videos, experimentation, etc., or enhance workflows



Content modeling – a visual UI for managing content models and adding new reference fields or validation

To rapidly publish content across the globe, Contentful uses multiple global content delivery networks (CDNs) such as Cloudfront and Fastly to enable regional access and API payloads in the sub-100 millisecond range.

Read more:

[Contentful Concepts: API Basics](#)

[Contentful Concepts: Webhooks](#)

[Contentful Concepts: UI Extensions](#)

[Contentful Concepts: Content Models](#)

[Contentful Tutorials: Getting Started](#)

Getting started: Content modeling sets the foundation

In today's data-rich applications, data is stored and handled by the application in a structured way, allowing various technologies to access and use the data in real time to provide the best possible user experience. Data models define how data will be structured and allow for data portability, scalability, and change. This approach has become fundamental to the modern web experience.

What is content modeling?

In Contentful's content infrastructure approach, your website content is handled as data, whether it be expressed in text, images, video or another file format. Content modeling gives your content a modular structure, so APIs can easily access and reuse it across many different use cases. Structured content makes it easy for humans to experience your content and for websites, applications and other systems to consume your content.

Read more:

[Hitchhiker's Guide to Structured Data](#)

[Patience and the Right Tools:](#)

[Developer's Guide to Content Modeling](#)

[Contentful Concepts: Contentful](#)

[Data Model](#)



Content models vs. web pages: chunks vs. blobs

In the past, websites were created by building web pages using an approach that treated content like a library of Microsoft Word documents. This content-as-a-page lived in traditional CMS systems that were designed to manage individual documents within fairly rigid page templates.

In today's content-is-everywhere world, the web page approach makes it extremely difficult to manage individual pieces of content that need to be repurposed across many contexts. It also poses a significant hurdle to both content editors and developers, who must take the time to build new page templates for every new instance of content on every supported device. When teams slow down, so does website growth, which puts business value at risk.

"We are in a war between giant, unstructured blobs of content, and clean, well-structured fields of content that have metadata attached. We are in a war of Blobs versus Chunks. You all are on Team Chunk. We cannot let the blobs win."

Karen McGrane, Content Strategist, Author

Put another way, legacy CMSes tended to create "blobs" of content made up of many different "chunks" or individual content elements, such as text, images, or media, and wrapped together with rigid formatting. Moving away from blobs of content living in legacy CMS is the first step towards modernizing your website architecture. The second step is to create a content model based on structured chunks of content, enabling your product team to build a flexible, scalable, and future-proof website.

This chunk-based content model supports the needs of your entire team, from content creators to designers to developers. It centralizes and structures every piece of content, no matter how large or small, making it easy for editors to keep your content fresh across myriad touchpoints.

Moreover, a chunk-based content model is easy to replicate and makes the content itself elastic and portable, so your product team can deliver a growing body of content via API to any digital experience, on any device, using any technology stack.

Contentful makes it easy to create content models that support all of your user journeys through your portfolio of web experiences. Conceptually, your content model is the blueprint of your website; it guides the Contentful APIs to place each block of content where you specify, so that your website delivers a fully realized digital experience at every endpoint.

"[Contentful] supports powerful content modelling primitives as code and content model evolution scripts, which allow treating it as other data store schemas and applying [evolutionary database design](#) practices to CMS development."

ThoughtWorks¹ Technology Radar, May 2018

Inside Contentful's content modeling toolbox

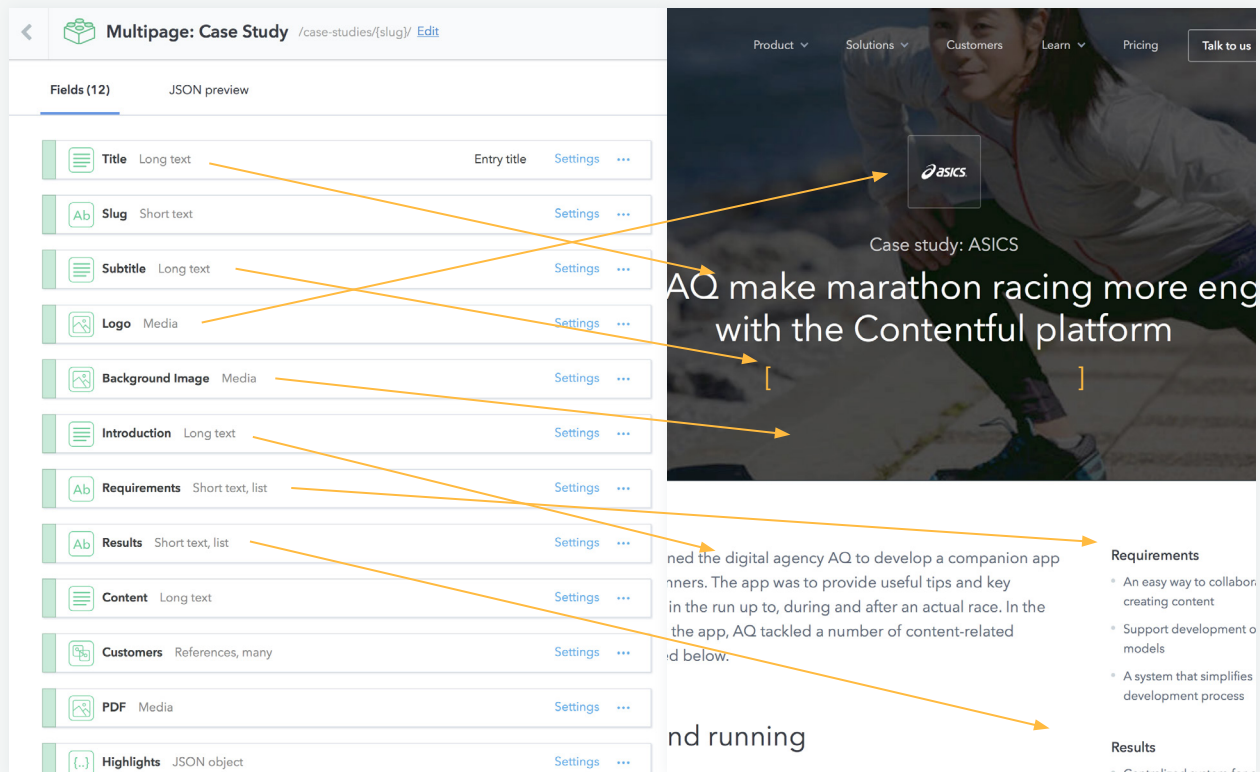
In a traditional CMS, your content must fit pre-defined buckets specified by page templates. If the content doesn't fit, then a new template is needed. Content modeling in Contentful works differently. Your content doesn't have to fit a particular model—you make the model to fit your website's content needs and you can evolve this structure as your website grows. The platform gives your team tools, such as the [Contentful web app](#) or [Content Management API](#), to easily define a unique content model that makes sense for your website.

So, how do you define a content model in Contentful? It is made up of individual content types, each representing a single unit or module of content within your website. For example, a customer story video with a headline and caption is an individual content module that can appear in numerous places on a website.

A content type is made up of a number of fields which denote the category of data that will be included in the content experience.

When creating new content types, it's helpful to think broadly about how it will apply to both existing and future content. Will your content creators always be entering the same kinds of data? Or will the data change depending on certain criteria? Will the content type be reused in other ways? For example, if you are creating a blog and don't want to constantly retype the author's name and their bio, you can create a chunk containing all that information and reuse it across multiple posts.





An example of a content type (left) and the resulting web page (right).

For example, let's say you're building a content type for a news article. The article title would have its own field, the body copy would have another, and the associated image and video would each have their own field. Your content model can also include layout options and metadata describing how this article should be delivered across channels. For example, you could set a field as "featured," which would present the article differently from other articles on your site. Your content model packages the news article into a modular, portable format, making it easy to repurpose for display on a laptop browser, a mobile application, a smart watch, an information kiosk or any other digital device.

Read more:

[The Beginner's Guide to Contentful](#)
[Content Modeling Basics](#)

Mapping content types to the front end

Content types offer three important benefits to the following members of the product team:

1

Content creators

or editors can independently control how their content is displayed, which in turn gives them control over how they deliver their narrative.

2

Developers

can give editors control over the content experience without requiring them to code or use a WYSIWYG editor—everything is accessible through the web app.

3

Designers

can give editors the freedom to lay out their narrative without impacting the core design, ensuring a consistent user experience and preventing multiple variants.

With Contentful, this healthy tension between the three perspectives is balanced by a flexible content model. Editors can pick content types and fields from a shared library, populate them with content, combine them with other content types, and compose a digital experience, one block at a time. This concept of a “component library” or “design system” is another key feature of modern web development. These component libraries map to front-end components, enabling teams to deploy content updates to production faster, with greater efficiency and consistency.

Read more:

[Topics and Assemblies: Your Weapons in the War of Blobs vs. Chunks](#)

[Getting Up to Speed on Composable Entries](#)

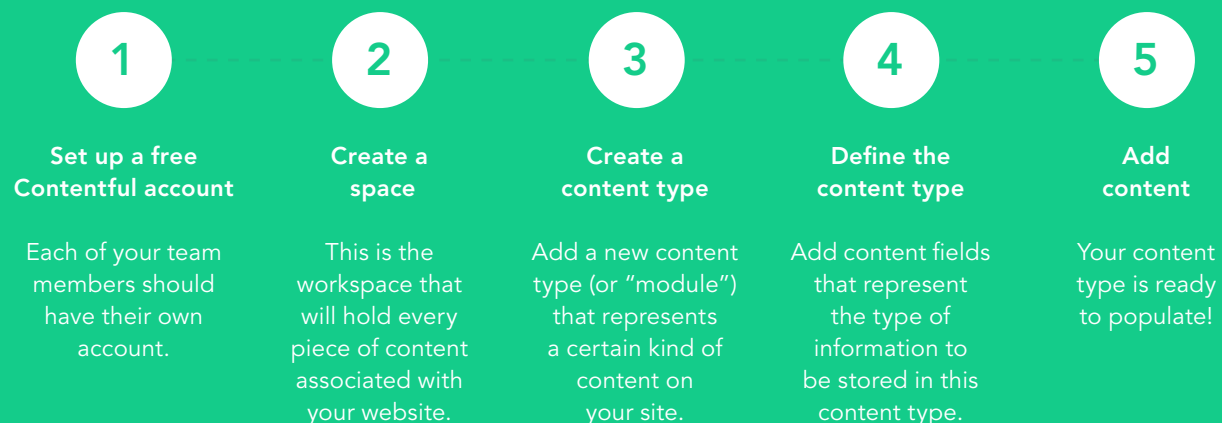
Creating your content model in a few easy steps

Whether you are migrating an existing website to Contentful, or building one from scratch, the basic steps to creating a content model are the same. Start by developing a deep understanding of the purpose and nature of all pieces of content that contribute to your website experience, from large to small. Consider how use of that content will expand and evolve in the future so you can make your model as elastic as possible.

Contentful provides two tools to create and manage your content model. Editors and non-technical users can use the Contentful web app, which provides a visual representation of your content model with easy point-and-click tools. Developers can also use the Contentful Content Management API, which gives them a more programmatic approach. Both tools connect to the same content model and allow the whole product team to participate in the process.

Content modeling at a glance

The basic steps to creating a content model on Contentful are:

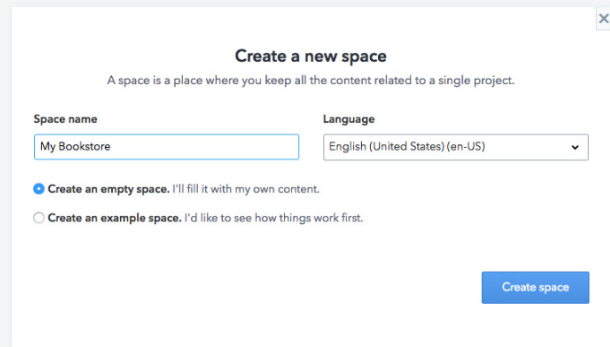


Building your content model is quick and easy. The challenge is to conceptually understand which parts of your site should be represented as content types, so that they can be efficiently repurposed across your website and future portfolio of digital experiences.

Content modeling in practice

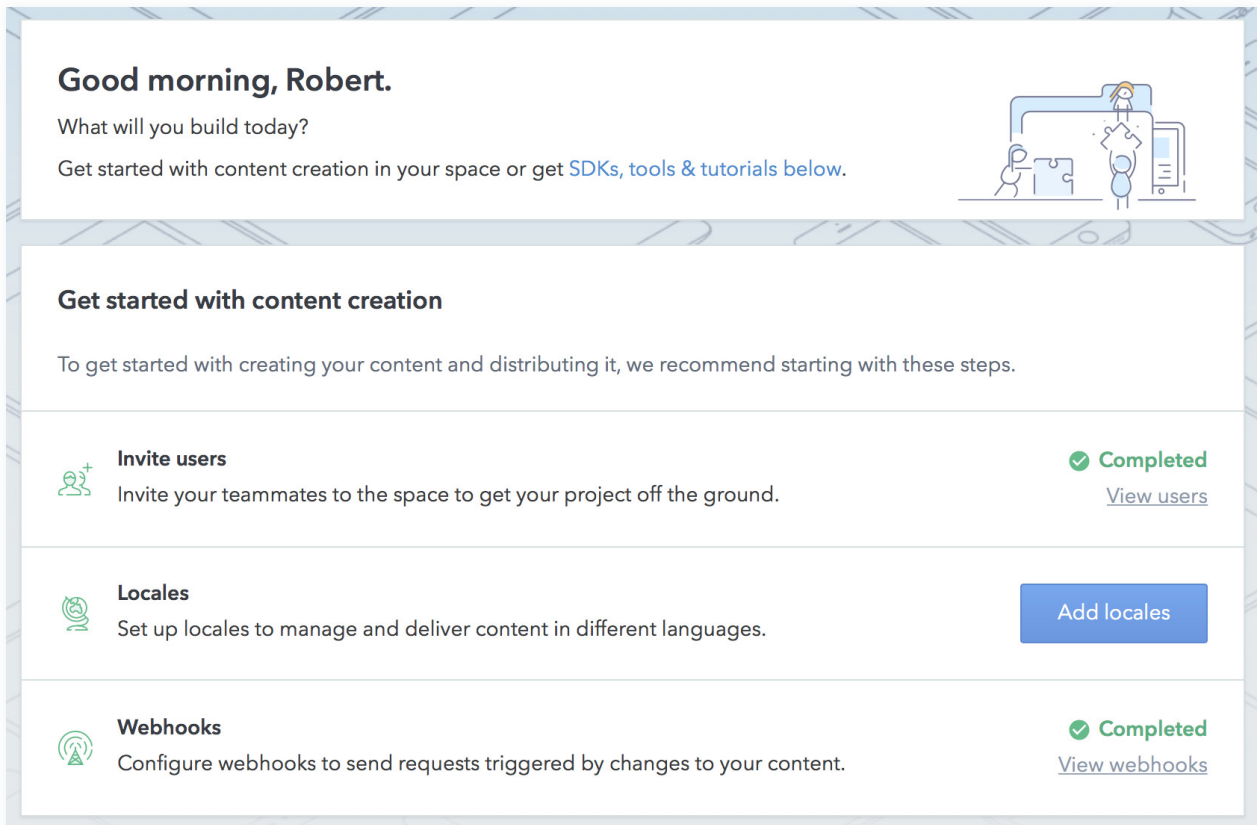
Let's walk through the content process with a sample use case. Let's say you're selling books online and you have a large inventory to showcase on your site. Your content model will store everything that you want visitors to know about your business and each of your books for sale, as well as all of the user interface information and other content that creates a great experience on your site.

Your content editor Robert will build your site's content model, and he starts by creating a space for your website in Contentful:



The screenshot shows a modal window titled "Create a new space". Below the title is a subtitle: "A space is a place where you keep all the content related to a single project." There are two input fields: "Space name" with the value "My Bookstore" and "Language" with a dropdown menu showing "English (United States) (en-US)". Below these fields are two radio button options: "Create an empty space. I'll fill it with my own content." (which is selected) and "Create an example space. I'd like to see how things work first." At the bottom right is a blue button labeled "Create space".

Next, Robert begins building the content model by creating his first content type.



The screenshot shows the Contentful dashboard for a new space. At the top, it says "Good morning, Robert." followed by "What will you build today?" and a link to "Get started with content creation in your space or get [SDKs, tools & tutorials](#) below." To the right is an illustration of a person working on a computer. Below this is a section titled "Get started with content creation" with the text "To get started with creating your content and distributing it, we recommend starting with these steps." There are three items in a list: "Invite users" with a plus icon, "Locales" with a globe icon, and "Webhooks" with a signal icon. Each item has a description and a status indicator. "Invite users" and "Webhooks" are marked as "Completed" with a green checkmark and have links to "View users" and "View webhooks" respectively. "Locales" has an "Add locales" button.

Good morning, Robert.
What will you build today?
Get started with content creation in your space or get [SDKs, tools & tutorials](#) below.

Get started with content creation
To get started with creating your content and distributing it, we recommend starting with these steps.

- Invite users** ✓ Completed
Invite your teammates to the space to get your project off the ground. [View users](#)
- Locales**
Set up locales to manage and deliver content in different languages. [Add locales](#)
- Webhooks** ✓ Completed
Configure webhooks to send requests triggered by changes to your content. [View webhooks](#)

Since this is a bookstore site, a great place to start is by building a content type for an individual book.

Create new content type

Name (required)

Book

Api Identifier (required)

book

generated from name

Description

This is the content type for a book


in less than 500 characters

Create


Cancel

Robert defines the content type by adding fields that represent information associated with the book. He chooses "Text" for a book description and "Date and Time" for publication date. He may want to add "Number" for product ID and "Media" for an image of the book cover. He can always add new fields later, for example "Reference," to link to a future blog.


Add new field




Text
Titles, names, paragraphs, list of names




Number
ID, order number, rating, quantity




Date and time
Event date, opening hours




Location
Coordinates: latitude and longitude




Media
Images, videos, PDFs and other files



Boolean
Yes or no, 1 or 0, true or false



JSON object
Data in JSON format



Reference
For example, a blog post can reference its author(s)

Robert then configures the new fields that he's added to his content type.

New Text Field

Name

Title

It will appear in the entry editor

Field ID

title

It is generated automatically based on the name and will appear in the API responses

☒ Short text, exact search


Enables sorting
Maximum 255 characters
Use for titles, names, tags, URLs, e-mail addresses

☐ Long text, full-text search

No sorting
Maximum 50k characters
Use for descriptions, text paragraphs, articles

☐ List

Select this if there is more than one value to store, like several names or a list of ingredients
API response will include a separate block for each field

 These settings cannot be changed later.

Create

Create and configure

Change field type

So far, Robert's content type is taking shape.

The screenshot shows the configuration interface for a content type named 'Book'. At the top, there is a back arrow, a 'Book' icon, the title 'Book', and an 'Edit' link. Below this, there are two tabs: 'Fields (3)' and 'JSON preview'. The 'Fields (3)' tab is active, showing a list of three fields:

- Title** (Short text): Entry title, Settings, ...
- Author** (Short text): Settings, ...
- Publication date** (Date & time): Settings, ...

Once he's completed his content type, he can then start populating it, creating new entries for each book in the store catalog. Robert can easily preview draft content in a secure view—for example, an unpublished book listing—without leaking it or impacting live content.

The screenshot shows the configuration interface for a content entry named 'Steppenwolf'. At the top, there is a back arrow, a 'Steppenwolf' icon, the title 'Steppenwolf', and an 'Info' link. Below this, there are two tabs: 'Actions' and 'Info'. The 'Actions' tab is active, showing a list of fields:

- Title**: Steppenwolf (11 characters, Requires less than 256 characters)
- Author**: Herman Hesse (12 characters, Requires less than 256 characters)
- Publication date**: Sunday, July 10th 1927 (10:00 UTC+02:00)

On the right side, there is a 'STATUS' section with a 'Status: Draft' indicator and a 'Publish' button. Below this, there is a 'PREVIEW' section with an 'Open preview' button. At the bottom, there is a 'LINKS' section with the text 'No other entries link to this entry.'

Robert's fledgling content model, using his first content type, emerges as he adds entries.

5 entries found			
<input type="checkbox"/> Name	Updated ▾	Author	Status
<input type="checkbox"/> A Tale of Two Cities	a few seconds ago	Me	PUBLISHED
<input type="checkbox"/> The Picture of Dorian Gray	a few seconds ago	Me	PUBLISHED
<input type="checkbox"/> All Quiet on the Western Front	a minute ago	Me	PUBLISHED
<input type="checkbox"/> The Moon of Jupiter	a minute ago	Me	PUBLISHED
<input type="checkbox"/> Steppenwolf	2 minutes ago	Me	PUBLISHED

Once Robert has completed a content type for the primary content on the site—products for sale, business information, etc.—he will want to create content types for smaller pieces of content, such as the microcopy and navigation elements that we addressed in the previous chapter. In addition, Robert might want to isolate some pieces of content that will be repurposed in multiple areas. For example, he might want to display an author’s bio associated with all books by that author. In this case, the bio content would be treated as its own content type.

Robert can also use Contentful’s built-in tools to optimize his workflow. For example, he can create a media library to host and manage all media assets in one place, so he can update source files, such as when a book cover changes, without breaking the user experience. Even if the book cover is displayed in multiple locations on the website, such as under “new releases,” “science fiction,” and “award nominees,” changing the cover in his asset library will update it in every location. A built-in markdown editor allows content editors like Robert to write faster and publish across platforms.

Model for usability and functionality, not just content

It's easy to focus your attention on "primary content"—that business-critical product copy, a brand video with viral potential, or an important campaign banner. But what about the smaller bits and pieces of content, such as "microcopy," that also work hard to create a great user experience?

These include form labels, help text, error messages, navigation buttons, and other UI words and phrases that improve usability and support a smooth user journey. Like their more prominent counterparts, these pieces of content also need to be authored, often translated, experimented with, published and managed. With Contentful, your content model gives them structure, too.

Traditional CMS systems are designed to store large, primary content assets and are not well-suited to handling small text strings. To reliably expose these strings to editors within the typical CMS architecture would be costly and complex. Therefore, microcopy is typically stored directly in the front-end source code. To make changes, content editors or UI/UX designers must submit a change request to developers, who then must deploy a new version of the site code. When microcopy is stored as a static asset, updating it requires a lot more work, making it harder to optimize and evolve.

With Contentful, your microcopy is governed by your content model and managed as its own set of content types. It is delivered to your end user experience via API, rather than hard-coded in application code, making it as flexible and dynamic as the rest of your content. Content editors and UI designers can easily take control of these small but mighty contributors to the user experience to revise, test and maximize their effectiveness. By transforming your microcopy from a static asset to dynamic content, product teams are empowered to create a smoother, more effective website journey without the frustrating back-and-forth microcopy editing required by a traditional CMS-based workflow.

Your content model can also include your navigation. Traditionally, navigation has been dictated by a CMS or managed in markup. By including navigation in your content model, you make it more accessible, flexible, and dynamic. The content modeling process allows you to explore and fully understand your content and the complexities of relationships among elements, granting you better control over the user experience.

Read more:

[Dynamic Microcopy: Putting UX Text in the Hands of Your Marketing and Editorial Team](#)

[Modeling Navigation](#)

Extending your content model in Contentful

Innovative product teams are always striving to develop new and different ways to improve their product and maximize its value to the business. As they do so, they'll want their content model to be as flexible as possible so it can easily adapt to new use cases and business requirements without significant new work from the team. Contentful enables product teams to leverage their content model to help them work faster and smarter.

Let's continue with the bookstore example and look at three important ways the platform helps Robert extend his new content model.

Easy localization

Once you build a content model in Contentful, this core structure can be repurposed across multiple digital experiences, including localized websites. In the bookstore example, Robert can expand his content model to include new languages, add translations to his English content entries, and provide international customers with a web experience in their own language and books appropriate to their market.

UI Extensions

Developers can customize the user interface of the Contentful web app to help content editors more easily create, organize and manage content in a consistent way that works best for their team and business. For example, Robert can attach specific instructions to content types and fields that guide his content editors in their modeling or content entry workflows.

Integration with other cloud platforms

Contentful APIs integrate easily with other cloud platforms, so you can work directly with externally hosted content in the Contentful web app. For example, Robert can pull in an author video from YouTube to help promote a book, or pull reviews from a book blogger's website.

Read more:

[The Most Foolproof Way to Get Started with Contentful](#)

[Content Modeling Example: Creating a Digital Lookbook](#)

[Contentful Concepts: UI Extensions](#)

Getting to production: integrating with your delivery pipeline

In the past, software was treated like a boxed product, with new versions released once every year or two. The release process was expensive and time consuming. Product teams would build the entire product, then development would grind to a halt while the team tested and debugged the entire thing until it was ready for release. The rise of the cloud has paved the way for new models of software delivery that can keep pace with today's evolving digital experiences.

Shipping content continuously

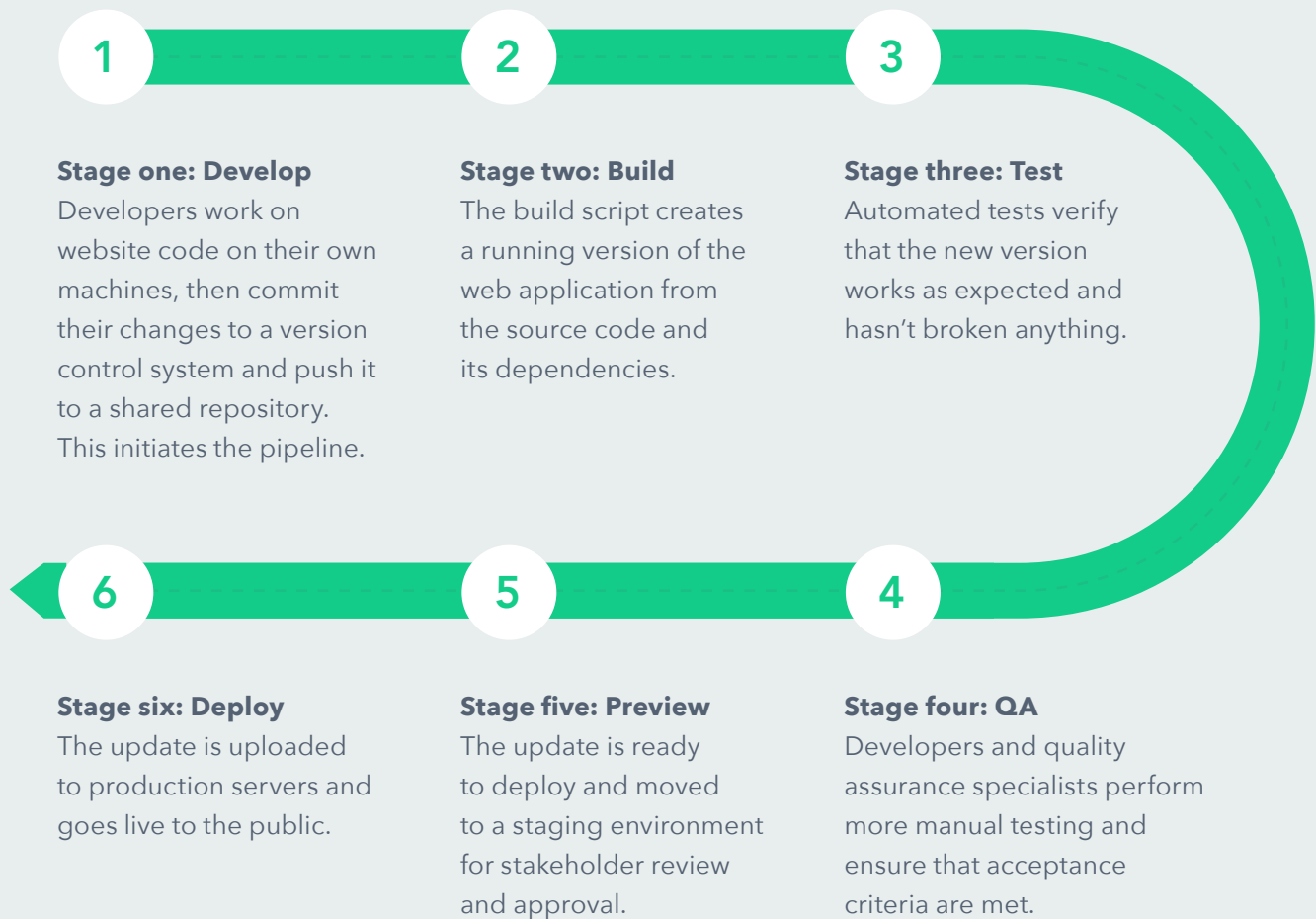
Amongst teams practicing Agile development, the preferred approach to shipping code is a two-part practice known as continuous integration and continuous delivery (often referred to as "CI/CD"). With continuous integration, each developer works on a branch of a website's code, tests it, and then merges it back into a shared master code base—often several times per day. Once code changes have been merged, the continuous delivery processes take the code through various testing stages before it finally goes to production. Releases can happen several times per week, or even per day.

CI/CD practices enable short build/test/release cycles and help your product teams reliably ship smaller sets of changes at a rapid cadence, therefore bypassing the complexities of monolithic annual releases of the past. Your product team benefits from reduced complexity, greater automation, and improved efficiencies. Your product quality improves due to test-driven development, and risk is reduced with small, focused deployments. Finally, your business benefits from faster time to market and more control over release timeframes.



Inside the deployment pipeline

Teams that practice CI/CD use a deployment pipeline—a set of validations through which website updates must pass on their way to production. Your pipeline is composed of a linear set of stages, and each stage has its own environment for running and testing code. A typical pipeline consists of the following:



Contentful's content infrastructure is a cloud-based service that integrates into your website via APIs. The content model is treated as code, and can, for example, be backed up, automatically updated, and replicated like code. It therefore integrates seamlessly into your deployment pipeline at every stage. This means that your developers can extend and update your structure content in the same way that they apply changes to code—using the same fluid, CI/CD approach.

Read more:

[CMS as Code: Contentful's Content Infrastructure](#)

[Contentful Concepts: Integrating Migrations in Your Content](#)

[Delivery Pipeline](#)

Seamless collaboration in Contentful

One of the challenges of content management is that there are two deployment cycles happening at the same time, both requiring authoring, verification and release. One is the software deployment pipeline, in which developers are writing and releasing new versions of the website, including changes to the content infrastructure. In a separate cycle, content creators are authoring and releasing new content and updates. How can the two coexist?

Contentful decouples content infrastructure development from content authoring and management, which allows teams to move fast without breaking things. The platform's ability to support multiple space environments allows members of your product team to build and iterate even the most complex digital experiences in parallel, each working in the environment that is appropriate to their workflow.

Contentful space environments

Using spaces, Contentful's tool that includes your content model and repository, your development team can configure individual space environments for development, QA and production. Your space environment setup maps to your pipeline flow.

1

Developers use their **sandbox environment** for making changes to your content infrastructure.

2

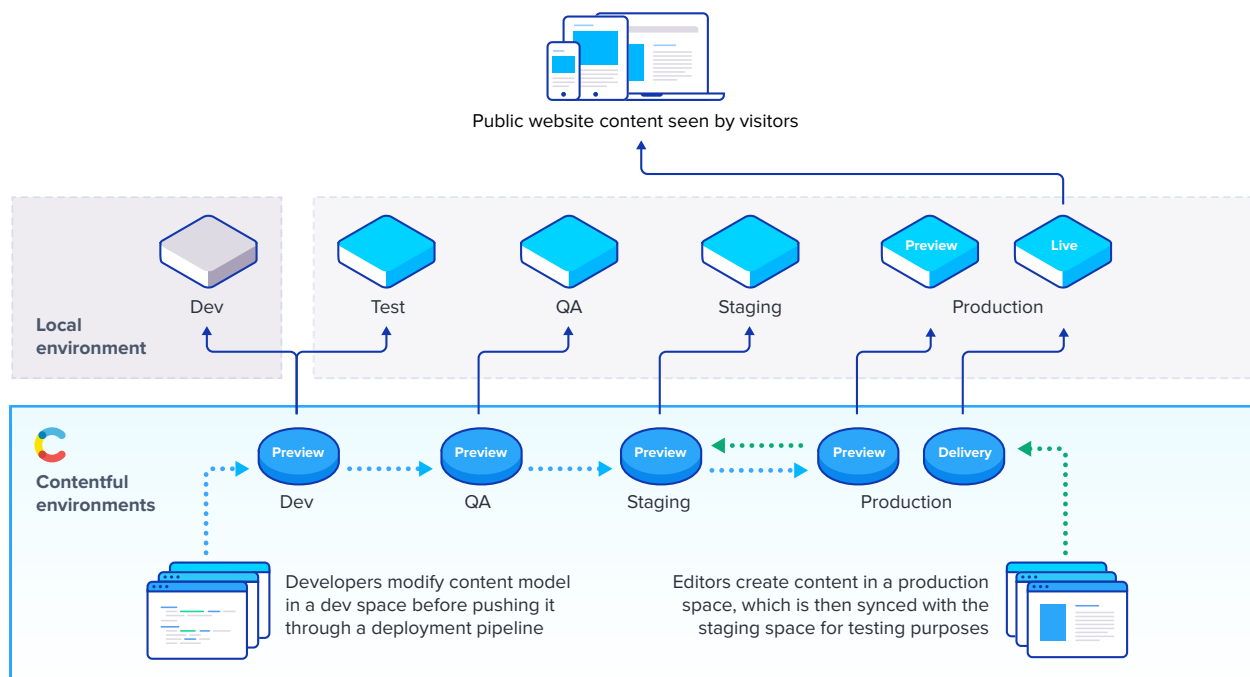
They then migrate content infrastructure update to the **QA environment** to validate changes—without impacting the live site or preventing content editors from updating live content.

3

Once approved, the content infrastructure updates are pushed to the **master environment** that holds your live content.

4

The **master environment** allows your content editors to author and manage content, and your website and other applications to query and utilize content.



The collaboration workflow in practice

Going back to our bookstore example, let's say that the business has expanded to selling music. Using the sandbox environment, the product team adds several new content types to the content model, which updates the content infrastructure. The designers and developers also make any necessary changes to the website UI and add code to accommodate the new content types.

The changes are then migrated to a QA environment and website staging environment for testing and verification. Any bugs or issues get resolved, and the new music section is ready to push to production.

Once it gets to the master environment, content editors populate the new content types with the available music catalog. Using Contentful's Preview API, everyone can see how the new music section looks on the website before it's open for business. If all looks OK, then a simple click on the "publish" button will push it live to customers.

Ultimately, Contentful's flexibility helps a product team reduce complexity and build new website experiences faster, while maintaining the build/test/release discipline of their CI/CD practice to produce high-quality websites. In addition, the platform enables multiple teams to collaborate asynchronously and remotely, helping to scale collaboration across a growing product organization and a growing portfolio of digital experiences.

Read more:

[Why Space Environments are Great for Managing and Maintaining Content Structure for Contentful Projects](#)

[Contentful Concepts: Managing Multiple Environments](#)

Implementing new architectures with Contentful

Today, developers are building the front-end components to their websites using a variety of architectural approaches. Many considerations factor into this strategic decision, including the types of content they have to work with and the types of user experiences they need to deliver.

For example, your team might be working with very large files, such as an interactive timeline, and therefore page load time is critical. Alternately, your team might be building an ecommerce checkout experience where everything needs to happen on the same page to prevent shopping cart abandonment.

Development considerations also come into play. Some teams want to build their products in a single codebase (say Java) and then port them into specific languages (Swift, Windows, Javascript) for distribution on different platforms. Some care about the speed of development and focus on fast release processes, while others care about efficiency and use build systems that allow them to work with design systems and reuse the same components throughout their projects. Hosting costs and security are other factors that guide technical strategy.

Some of the most popular architectural patterns include:



Static – Simple, cheap, easy to deploy and maintain, with fast load times



Single Page Application – Load a single HTML page and dynamically update that page as the user interacts with it, producing a fluid, responsive web experience without constant page reloads



Backend for Frontend (BFF) – Combines data from multiple sources, hides sensitive information, and reduces the number of requests.



Hybrid – Pages are static when rendered, but become interactive once loaded client-side

Contentful supports all of these patterns with tools, plugins, and guidance to help developers integrate content infrastructure into their architecture of choice. For example, the platform supports several static site generators that help turn your static pages into dynamic, interactive experiences.

For teams that experiment or migrate their approach over time from one architectural pattern to another, they might cycle through a number of front-end technologies, going from pure Javascript to Angular to React components, and experiment with a variety of build tools. However, Contentful is one platform that remains constant as development teams evolve their build process or app architecture, or add new delivery channels to their digital portfolio. Teams are free to explore different options before committing to a specific path and can rest assured that their content authoring tools and hosting environments will remain the same for developers and editors.

Read more:

[The New Era of Static Sites:
How Javascript Powers Everything](#)

[Dynamic Static Sites: Implementing
an Oxymoron](#)

[Contentful Tools: Static Site Generators](#)



Getting to scale: managing complexity

Over time, product teams experience growing pains, such as when the team is scaling up the number of internal contributors and external partners. At the same time, they might also need to scale their website portfolio to serve new product lines or global markets. When it comes to content, Contentful provides a range of tools to scale effectively and painlessly.

Managing people: roles, permissions, and workflows

In Contentful, space administrators are tasked with configuring who can do what within the space. Using the Contentful web app, administrators define roles in detail, and set fine-grained controls over access and editing permission. This feature is particularly useful for content creation teams with numerous members, both internal and external, each with distinct tasks in the creation workflow. For example:



Writer

Can access and edit content, create new content, but can neither publish nor delete it.



Translator

Can access content and input a translation, but can't modify the source.



Proofreader

Can only view content or suggest edits.



Photo Editor

Can only access, write, or edit content associated with images.

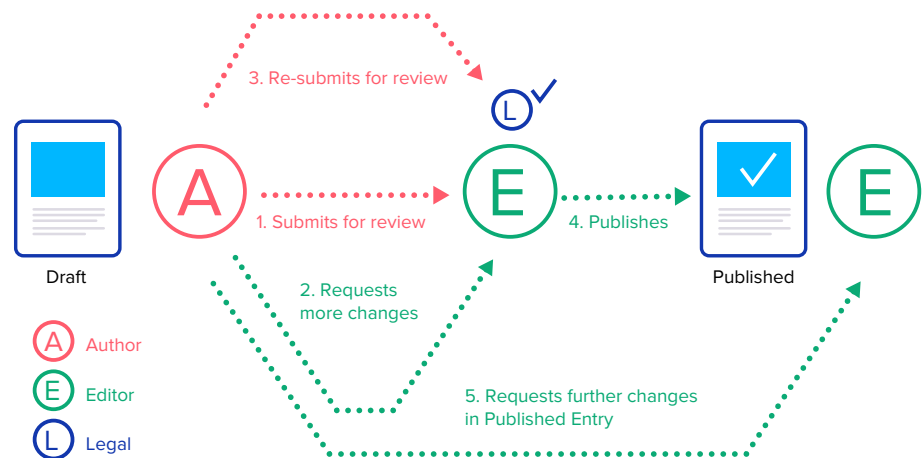


Editor

Can create or edit content, but is the only person who can publish content.

In this example, your space administrator would set different permissions in Contentful for each role on the content creation team to ensure that everyone is working together in harmony.

Contentful enables you to establish even more structure for your team by setting up content approval workflows on the platform. Workflows can be as flexible or as prescribed as needed. In many cases, the entire workflow—content creation, editing, proofreading, translating, review and approval—happens directly in the Contentful web app. If busy, non-technical content approvers might not have the time or inclination to use the web app. Even so, these key non-technical stakeholders are still able to approve content, either directly from Contentful’s preview environment or via email, without ever having to touch the web app.



Setting up content workflows in Contentful requires some configuration by developers, including creating webhooks to automate notifications via email, Slack or any other channel. However, the end result is powerful. For each piece of content, you can check its status at each stage of the workflow to identify bottlenecks and better manage the process.

Step	Entry Status	Review Status	Approved By Legal	Shows In Saved Views
Author submits draft entry for review	Draft	Needs review	-	Needs review
Editor rejects draft and requests more changes	Draft	Needs changes	-	Needs changes
Author re-submits entry for review	Draft	Needs review	-	Needs review
Editor approves	Draft	Ready to publish	-	Needs review

Read more:

[Contentful Guide: Managing Roles & Permissions](#)

[A New Way to Set Up Content Approval Workflows](#)

Managing replication: localization, repurposing, and version control

As content scales, it can easily mushroom out of control. Content repositories are often bloated with duplicate content and teams struggle with accessing and managing multiple versions of a particular content piece. When multiple applications require multiple CMSes, content management becomes even more chaotic.

Contentful's content infrastructure has built-in elasticity along with structural tools that help you prepare to scale from the beginning. So when it comes time to replicate a piece of content for additional use cases, such as a new app or a portfolio of websites, you can do so in a few simple steps.

Using Contentful Locales

Depending on your business, your company might wish to provide a localized experience for customers and partners in other countries. Contentful natively supports localization and provides a built-in feature called "Locales," enabling your product team to spin up localized versions of your website faster.

The platform gives you fine-grained control over which content types and fields can be localized, and everything happens in the same content entry. All translations are entered and stored in the same record, so the team doesn't have to access separate records for each language. In addition, Contentful's roles and permissions feature can limit translator access to a particular language to help prevent mistakes.

Locales offers several other controls that enable you to easily manage localized content. You can disable editing for a particular locale but keep its content visible on the website, or conversely, hide locale content on the website but still make it editable in Contentful. Content fields can be made optional or mandatory by locale, and you can publish different locale content at different times.

You can even set a default fallback language for locale fields that are missing content (e.g. for Swiss visitors, French text appears when German text is unavailable). If your business produces large amounts of content that is translated by an external agency, you might prefer to set up a direct integration with the translation platform used by the partner agency. Contentful integrates with all modern translation platforms, such as Smartling, Transifex, PhraseApp and OneSky.

Read more:

[Using Contentful for Multi-Language Publishing](#)

[Contentful Concepts: Locales](#)

One image, multiple uses

Most websites have a subset of images that are repurposed across numerous parts of the site. These images usually need to be resized or manipulated to fit each context. For example, a logo might appear in one size in your navigation menu, in another size on your product page, and in yet another size on a press release.

With legacy CMSes, editors have to manually produce all these assets for different layouts, which is a painful, error-prone and time-consuming process. With Contentful, you don't have to store multiple asset files of the same image. The [Contentful Images API](#) leverages programmatic transformations that allow you to manipulate images on the fly by appending relevant attributes and input values to the image URL. Editors upload an image once, and then developers use variables to cut and resize the image for specific layouts, eliminating effort and delivering better results. For some demanding use cases, such as fashion or lifestyle websites, there may be a third step for editors to check programmatically re-formatted images and manually correct them for maximum effect.

Read more:

[Contentful Concepts: Images](#)

Content versioning and roll backs

Contentful helps your product team keep track of past versions of content and compare them to the current, live version. The platform produces a snapshot of a content entry each time it is published or republished. Content editors then use the web app to view previously published versions of an entry, find out who published an entry and when, and compare previous snapshots to the current version.

If a mistake has been made, it's easy to restore an earlier version by rolling back the contents of an individual field or the entire entry. Contentful also enables teams to version content types via the API and compare a version in a local development space against the live version on Contentful servers.

Read more:

[Contentful Versioning FAQ](#)

Conclusion

Understanding Contentful and how it fits into your development workflow empowers your team to treat your website like a product that's always evolving. Contentful's flexible content modeling future-proofs your site by making it easy to design tailored content types that are easily updated across all channels, while its API-driven architecture lets teams develop and update content any way they want.

When content can be delivered in many forms to a global, multi-device audience, websites become a more powerful tool for driving your business. Rather than get bogged down with costly one-time website projects, everyone can contribute to creating a better user experience using best practices that scale as the company grows.

Contentful's content infrastructure empowers your entire website team to deliver website experiences faster and keep them fresh with an ongoing pipeline of new content.

To learn more about how Contentful can help your business, please [contact us](#) to talk with an expert or visit contentful.com.



Endnotes

[1] ThoughtWorks, [Technology Radar](#), May 2018

