storyblok

The Definitive Migration Guide to Storyblok

An end-to-end guide for teams migrating from *any* CMS.

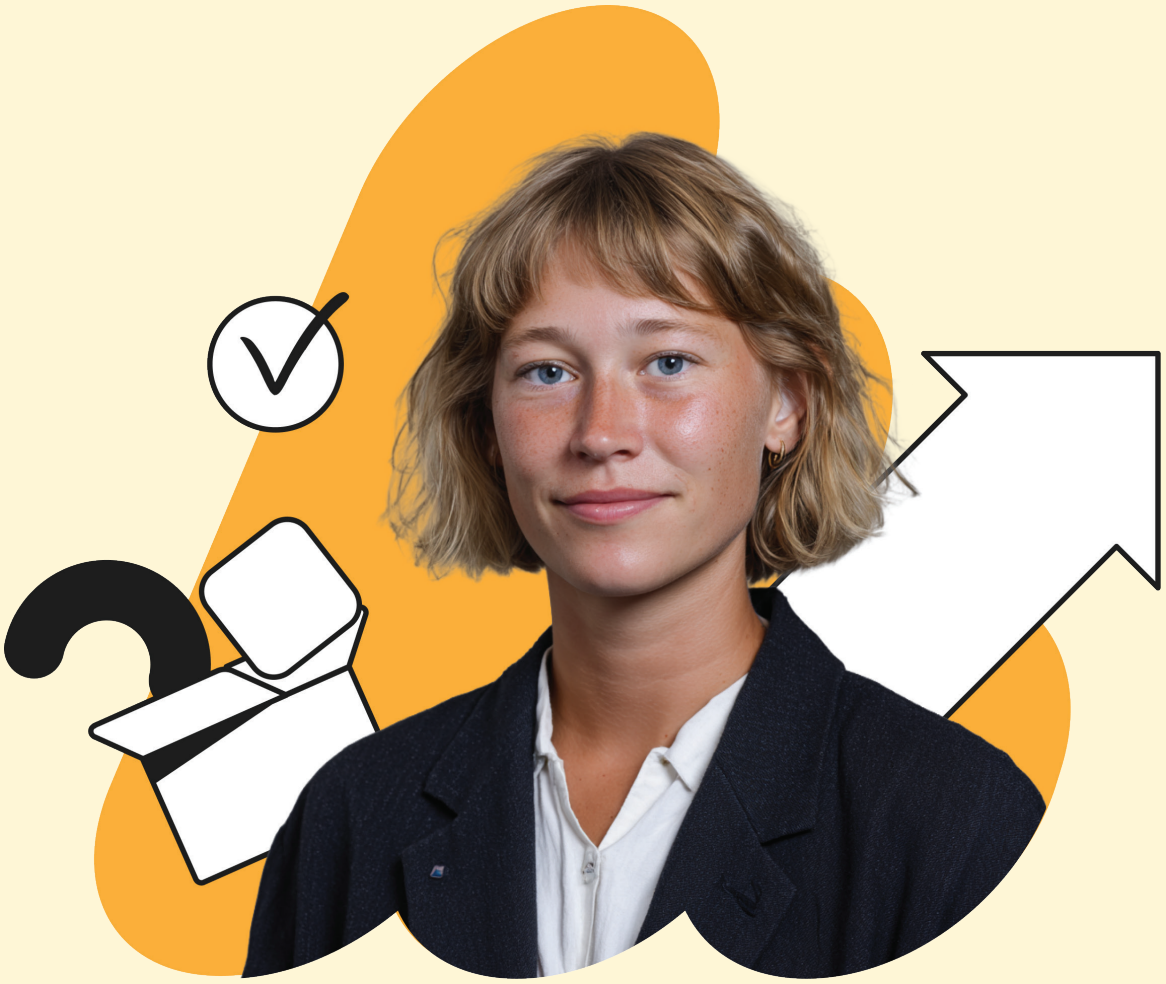
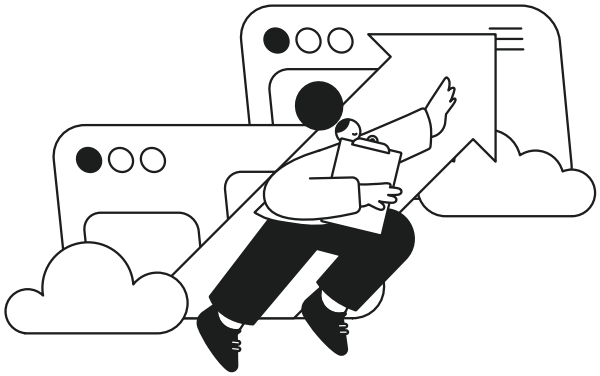


Table of Contents

PART I

The case for migration

Migrating to a headless CMS introduces flexibility and scale that legacy systems simply can't match. With an API-first model, brands can deliver consistent content to any platform or device; developers work in the frameworks they know best; editors can create and publish content without waiting for deployment cycles, and businesses can launch new experiences faster.



The benefits extend across the entire organization:

- **Lower costs** as redundant systems and plugins are retired.
- **Faster delivery** through reusable components and automation.
- **Developer freedom** through a technology-agnostic architecture that removes the constraints of outdated templates and rigid deployment processes.
- **Consistent branding** across every product, site, and region.
- **Simplified localization** from one CMS instance that manages content across multiple markets.
- **Omnichannel reach** where the same content can power websites, apps, or digital displays with platform-specific fields and outputs.

In short, migrating to a headless CMS paves the way for faster delivery, lower total cost of ownership, and the freedom to evolve without starting over.

Even though a CMS migration can seem like a big undertaking, switching to a more functional, scalable, and future-ready system pays off in the long run.

67.5% of senior developers say their CMS holds them back from doing their best work. Almost half describe it as a constant hindrance.

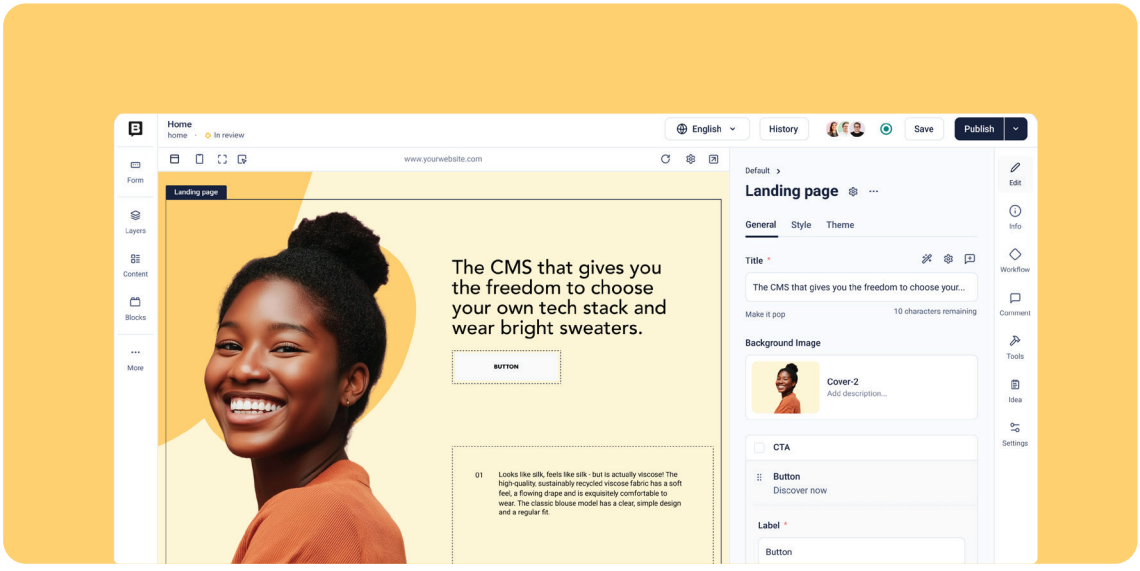
Source: Storyblok survey of 200 senior developers

Why Storyblok

Storyblok combines the technical depth of a **composable**, **API-driven CMS** with the usability marketing teams expect. The **Visual Editor** allows editors to see how content appears in real time across devices, removing the guesswork common in other headless systems. Developers gain flexibility through **modern SDKs**, a **robust Management API**, and a clean **component-based structure** that fits naturally into modern frontend architectures.

On the operational side, Storyblok provides enterprise-grade governance with **role-based permissions**, **audit trails**, **version control**, and **multi-environment workflows**. Structured blocks put your brand where it matters, right from websites, apps, product UIs, and AI search.

Headless architecture also supports **best-of-breed integration**. You can combine Storyblok with your preferred Digital Asset Manager (DAM), commerce engine, translation platform, or analytics stack. Each service does what it does best, and your business can remain adaptable to new technologies as they emerge.



The next evolution of headless is already here. Orchestration, automation and context are reshaping how content is managed and discovered. Storyblok is leading this shift by giving organizations visibility across their content network and the tools to optimize for the new generation of **Generative Search Engines (GEO)** and **AI-powered discovery**.

As search and decision journeys become increasingly AI-driven, **structured and well-connected content is what determines visibility**. Storyblok is the foundation for how your content performs in an AI-powered world. One update becomes the truth everywhere. Facts stay current, voice stays consistent, experiences stay coherent, so you can be discoverable, credible, and most importantly, chosen.

A migration to Storyblok is therefore a strategic step toward intelligent content operations, where teams have full control over both delivery and discoverability in the era of AI.

Guide objectives

Migrating may feel like a significant project, but it's an investment into long-term efficiency. A well-executed migration reduces technical debt, improves collaboration, and lays the foundation for continuous innovation.

This guide combines **strategic insight with practical instruction**, designed to help digital leaders, developers, and content teams understand what a successful migration looks like from start to finish, and how to plan each phase with confidence.

Specifically, this guide will:

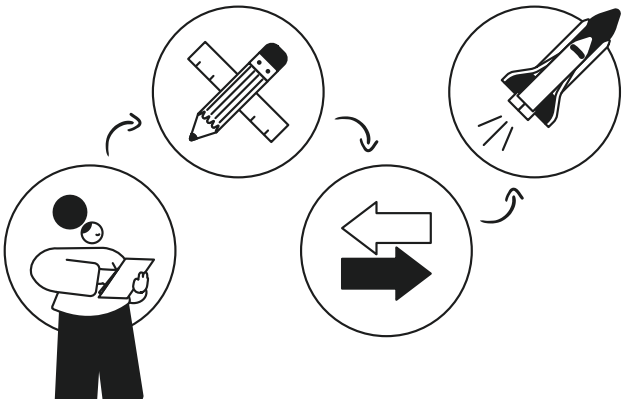
- Outline an **end-to-end migration roadmap** through our four-step Playbook.
- Help teams **assess resources**, define roles, and identify risks early.
- Decipher which route is best for your business: **a self-managed** or **partner-assisted** migration.

Whether your organization is moving from a legacy CMS or another headless platform, this guide serves as the single reference point for planning a smooth, secure, and measurable migration.

PART II

The migration playbook

We recommend a **structured four-step approach**, refined through hundreds of customer projects and partner conversations. This approach scales for any starting point, whether a WordPress monolith, or another headless platform, ensuring every migration moves from discovery to deployment in a controlled, measurable way.



Each phase links to supporting deep-dive sections and training resources in Part III of this guide, allowing you to dive in and scan as needed.

1 Step 1: Assess and plan

As with most projects, a successful migration begins with clarity.

Clarity on goals, scope, and what success really means.

Start by defining the **business drivers** for migration, such as faster content velocity, reduced maintenance costs, improved collaboration, or better scalability. Then translate those into measurable outcomes such as time-to-publish reduction, SEO parity, or infrastructure savings. Align on these with your stakeholders and **obtain the buy-in** needed to ensure heralded success throughout.

Then, conduct a **comprehensive audit** of your current CMS setup. This should include:

- Content structure, templates, and metadata.
- Integrations (e.g., DAM, analytics, or translation tools).
- SEO footprint, AI discoverability and backlink health.

This is also an excellent opportunity to see which pieces of content you should retain, redesign or simply retire; you can mock up a **content mapping matrix** to categorize each piece of content, and ensure you focus on high-value assets while reducing noise from the get-go.

Next, **gauge the effort, skills, and cost**. Some organizations have strong internal developer and engineering capacity, while others benefit from Storyblok-certified partners for complex integrations or large-scale content migrations. Regardless of who leads the work, define the critical roles and responsibilities of your migration team, along with your **timeline and success metrics**.

Example Deliverables: Migration scope document, RACI chart, estimated timeline, success KPI definitions.

See also: "Teams, training and enablement" section below.

2 Step 2: Design and configure

Once you have clarity on the strategy, the next step is to translate it into a scalable architecture that supports long-term content growth and operations.

Before you begin, take this opportunity to review what worked well in your previous CMS setup, and most critically, what didn't, so that your new Storyblok setup can build on proven workflows and address existing gaps.

First, define your localization and internationalization strategy as part of your **content model design**. These decisions influence your component and field structure, media organization, and reference logic. At the same time, plan your technical architecture for staging and testing to ensure safe, isolated deployments and clear review workflows.

Next, translate your audit findings into a content model design that fits a component-based approach. Define how pages, sections, and components will be structured in Storyblok to enable reuse, flexibility and consistency across projects. With your localization and technical framework set, your model can now be confidently finalized and scaled.

Set up your **Storyblok spaces** and **configure permissions** to match your governance model. Align workflows for editorial review, publishing, and version control. Define how translations, scheduling, and approval flows will function across teams.

Finally, align with your development team to ensure frontend architecture and deployment workflows (CI/CD) are prepared to support the content model. This ensures updates roll out smoothly, hosting remains stable and teams can revert changes quickly if needed.

IMPORTANT TO KNOW

Content modeling should be a collaborative process between designers, developers, and editors. For early exploration, teams can reference Storyblok's Blueprints to understand component modeling principles. Refer to the Storyblok Docs for more in-depth and up-to-date guidance on modeling best practices, automation options, and Management API usage.

Example Deliverables: Content component model, workflow and permissions matrix, environment plan, CI/CD setup outline.

See also: "Configuration and setup" section below.

Step 3: Migrate and integrate

This is where planning meets execution.

Choose the migration method that best fits your scale and system, be it an API-based import, a full rebuild, or a hybrid approach. Manual migration is typically used for complex landing pages or highly visual layouts that benefit from a clean start, while for structured content of scale (e.g., product catalogs, blogs, or documentation) automated or bulk migration methods are best. These can use APIs, direct database queries, or even content scraping when the legacy system is inaccessible.

Leverage Storyblok open-source migration scripts or **guides** as a starting point to automate imports via the Management API (these resources show how to map, transform, and validate content at scale). Specifically, validate the imported content by checking for broken links and image references, and ensuring that metadata and SEO settings carry over correctly.

At this stage, **integrate your key systems**, such as DAMs, analytics, translation platforms, and commerce engines, to ensure a cohesive ecosystem from the start. Remember, the goal is not only to move data but also improve and future-proof, by optimizing your structure, media handling, and metadata consistency.

Example Deliverables: Validated content import, integration checklist, redirect and metadata map.

See also: "Content, data and asset migration" and "Integration overview" sections below.

4 Step 4: Verify and launch

Before going live, thorough quality assurance (QA) is critical. QA covers more than functional checks. It includes structure, accessibility, performance and AI discoverability.

Ensure you have a QA checklist, which may include: running full testing across browsers and devices; checking accessibility compliance against standards such as WCAG 2.2 and confirming that all redirects and canonical links are functioning; performing load and performance testing to ensure speed under real-world conditions.

Once acceptance criteria are met, proceed to cutover. Treat launch as the start of a monitoring period; watch analytics and key pages closely for the first 48 hours, then continue optimization. Collect feedback from editors and users, document any issues, and optimize workflows based on real data.

Example Deliverables: Launch checklist, QA and performance test results, redirect validation report, success metrics baseline, optimization plan.

See also: "Testing, QA and go-live" sections below.

ESTIMATING YOUR TIMELINE

A CMS migration can vary greatly in duration depending on the size of your project, internal approval processes, and technical complexity. Use the questions below, organized by the four playbook phases, to surface dependencies, estimate effort, and align expectations across business, design and development. Your answers will help shape a realistic project plan and implementation timeline.

PLAYBOOK PHASE	KEY SCOPING QUESTIONS
Assess and plan	• How many sites, locales, and repositories are in scope? • Can we start with a narrow MVP or pilot (one site, market, or section) to the model and release flow before the full rollout? • Which areas can safely remain on the legacy stack during the transition, and for how long? • What percent of content will be migrated as-is, rewritten, redesigned, or retired? • Who must approve requirements and who is the final decision-maker? • What compliance, security or change management constraints could stretch timing? • Which parts can be handled in-house versus by a partner?

PLAYBOOK PHASE	KEY SCOPING QUESTIONS
Design and configure	• What localization model do you need at launch (languages, fallbacks, translation workflow)? • Which roles, permissions, and editorial workflows must exist on day one? • Have we considered all non-functional requirements, including performance budgets, accessibility standards, SEO targets, and AI discoverability?
Migrate and integrate	• How many content types, entries, and assets are in scope? • Which items will be migrated via API or export, and which must be rebuilt manually? • What transformations are required for rich text, media, and URLs? • Which integrations are launch-critical versus later, and what data mapping and sync cadence is required? • How will redirects, canonical URLs, and SEO metadata be migrated or redefined? • What is your redirect plan for changed URLs and legacy routes?
Verify and launch	• What acceptance criteria define “go-live,” and who signs off? • What rollback and backup procedures do we have in place? • What will we monitor at launch (success metrics)? • How will we train teams, and what is the support plan for the first 30 days?

Key factors influencing timelines:

- Team size and collaboration model across business, content, design, and engineering.
- Internal approvals or change management cadence.
- Scope of content restructuring or redesign.
- Number and complexity of third-party integrations.
- Use of partners versus in-house resources.

GOOD TO KNOW

For multi-brand or multi-market rollouts, we recommend a phased migration that starts with one pilot site or region. This helps validate workflows, components, and integrations before scaling globally.

TEAMS, TRAINING AND ENABLEMENT

A successful migration depends on aligning the right people early.

Define clear roles and ownership across content, development, design, and governance from the start. The following roles are typically involved throughout the migration:

ROLE	PRIMARY FOCUS
Program Manager	Oversees project milestones, budget, and communication between internal teams and partners.
Solution Architect	Defines integration architecture, migration flow, and API connections, ensuring technical scalability.
Front-end Engineer	Implements reusable components, ensures compatibility with the content model, and supports testing.
Back-end / Integration Engineer	Manages data transfer, webhooks, and automation between systems.
Content Engineer / Migration Specialist	Handles content model mapping, localization setup, and data validation.
Quality Assurance (QA) Engineer	Validates data accuracy and content rendering.
Editorial Trainer / Change Lead	Manages editor onboarding, creates sandbox environments, and defines content governance.

Note: level of involvement may vary based on project scope, scale, and team structure.

Any training should be practical and role-specific, with editors being onboarded via sandbox environments and walkthroughs of Storyblok's Visual Editor, publishing flow, and localization setup. Well-structured onboarding accelerates adoption, reduces post-launch errors, and builds confidence across distributed content teams.

Establish clear approval timelines for content reviews and deployment cycles, and schedule follow-up training to ensure editor confidence post-launch.

Common pitfalls in CMS migration

Even well-planned migrations can go off track when small oversights compound. Based on lessons from hundreds of Storyblok projects, these are the issues that most often delay launches or create long-term technical debt:

	COMMON PITFALLS...	AVOID BY...
1	Jumping straight into development without aligning on a shared content model, which results in inconsistencies across sites, spaces, and languages.	Involving editors, designers, and developers early, and using Storyblok Blueprints to agree on schema structure before any component is built.
2	Remaining in a single Storyblok space past the proof-of-concept (POC) stage, leading to staging and production unintentionally sharing data.	Planning environment separation early, and establishing distinct spaces for staging, testing, and production before scaling content or user access. Then, use Storyblok's advanced tools to streamline synchronization between these environments.
3	Duplicating or re-uploading assets instead of mapping or linking existing ones, leading to bloated storage and broken SEO links.	Reusing existing assets during migration. If your organization already uses a DAM such as Cloudinary or Bynder, keep that integration active. If you plan to centralize assets in Storyblok, migrate them once and map all URLs to the new location. This approach avoids duplication, preserves SEO equity, and keeps asset delivery consistent.
4	Overlooking governance (permissions, workflows, and version control can sometimes be afterthoughts... until something breaks).	Defining who can edit, publish, and approve content before migration begins. Storyblok's workflow management and version history help maintain traceability and protect data integrity from day one.

Treat a migration like any major release: version control of your content structure (schema), whether using Storyblok's in-app history or by exporting it in code via the Storyblok CLI or API for larger setups. This allows teams to track changes, automate validation, and ensure consistency across environments.

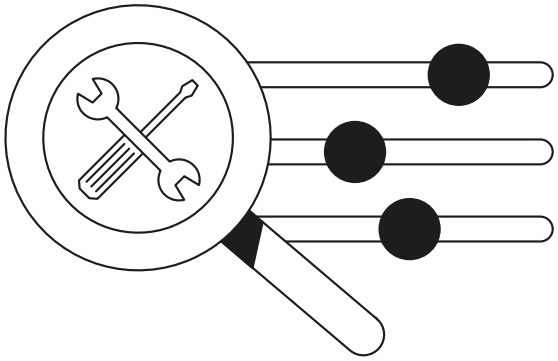
Strong structure and discipline prevent post-launch pain.

PART III

Deep-dive reference sections

Storyblok configuration and setup

Before content is imported or rebuilt, teams should establish their Storyblok environment; this means defining how spaces, roles, workflows, and deployment processes will operate.



AUDIT AND PREPARATION

Success is only viable when you have a complete understanding of the existing ecosystem. Begin by performing a **comprehensive content audit** to determine what needs to be migrated, redesigned, or retired.

Use site-crawling tools to generate a full inventory of URLs and assets, including:

- Subdomains and microsites that may not appear in a standard crawl.
- No-index or campaign-specific pages that may still hold value (e.g., PPC landing pages).
- Legacy RSS feeds and XML sitemaps that should be preserved.
- Custom-built functionalities such as calculators, widgets, or interactive tools.
- Media-heavy or animated content requiring reconfiguration in Storyblok.

At this stage, document technical dependencies and identify where APIs, scripts, or embeds connect to external services. This will support integration planning later in the process.

ENVIRONMENT SETUP

Begin by creating a well-defined environment structure. Most use **multiple Storyblok spaces**, say, one for staging, one for production, one for testing or sandbox experiments, and so forth. As you set these up, consider the following:

- **Roles and permissions**: Assign roles for administrators, editors, reviewers, and developers. Leverage Storyblok's capabilities here to control access by space, folder, or component level.
- **Workflows:** Define approval paths and publishing workflows early. Multi-environment setups benefit from distinct workflows for translation, QA, and release readiness.
- **Internationalization and localization:** Decide how to manage multilingual content at the field level (recommended for most setups), through folder-based structures, or across separate spaces for distributed teams. The choice depends on how your organization manages regional autonomy and governance.
- **CI/CD pipeline:** Establish a continuous integration and deployment flow to automate builds and updates. This minimizes downtime and reduces manual intervention throughout the migration and beyond.

GOOD TO KNOW

Align your environment structure with your content model design to avoid rework. For example, if your architecture uses shared components across brands, create a unified library that can be versioned and deployed consistently.

CONFIGURE AND CONNECT

Once you've set up your Storyblok spaces and environments, the next step is to **connect your development tools to the CMS**.

Storyblok provides a **CLI (Command Line Interface)** and a **Management API** that allow teams to automate setup and migration tasks. Developers can authenticate securely using access tokens that identify either an individual user or a specific project space.

These tools are particularly useful during migrations, as they make it possible to import or update large volumes of content automatically, sync components and configuration between environments, and reduce manual errors by validating entries before publishing.

Before full execution, **perform a test import** using a small subset of your content to validate field mapping accuracy, both internal and external link behavior, as well as the image and media asset display within the **Visual Editor**. Successful imports do not always guarantee that content will display correctly, since component mappings (mapping Storyblok components to frontend components), field types, or field bindings (linking Storyblok content fields to the frontend) may need adjusting.

Once rendering is verified, ensure your editors can preview and adjust entries independently (without developer support).

GOOD TO KNOW

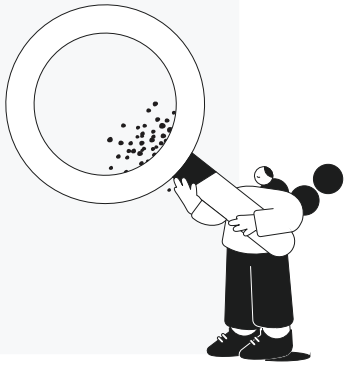
Integrate this testing phase into your build pipeline so automated checks run whenever new schema changes or components are deployed. Validation scripts can flag missing fields, mismatched schemas, or broken links before they reach production, saving significant time during QA!

SCHEDULING AND COORDINATION

Timing is often underestimated during CMS transitions. Plan migration activities during **low-traffic periods** to minimize disruption to live environments. For any larger programs, adopt a **phased rollout**; for instance, starting with a single brand, region, or content type before scaling globally.

TOP TIPS

- **Communicate the timeline early** to all stakeholders, including editors, SEO leads, and developers. This will ensure no parallel activities (e.g., campaigns or site releases) interfere with the migration window.
- **Plan low-traffic deployment windows** to minimize user impact.
- **Run incremental imports** before switching domain settings (DNS) or launch environments.
- **Maintain version control and rollback plans** so you can revert quickly if needed.



STORYBLOK'S TECHNICAL FOUNDATION

Storyblok's architecture is built to simplify complex migrations while maintaining high performance and governance. Its **composable content model** stores every piece of content in a structured format (JSON), which makes it predictable, easy to automate, and simple to map when moving from other systems. This ensures a clean, repeatable process.

Editors can preview, adjust, and validate migrated content directly within Storyblok's **Visual Editor**, which shows how pages will appear on any device. This removes much of the trial and error that slows traditional CMS setups and keeps developers and editors working in parallel without bottlenecks.

Behind the scenes, Storyblok's **global Content Delivery Network (CDN)** distributes content through servers located around the world. This ensures that JSON content responses, images, and other assets are delivered from the location closest to each user, reducing load times

and maintaining a consistent experience across markets. The built-in **Image Service** automatically optimizes assets right in the presentation layer, saving bandwidth and improving performance without extra configuration.

Finally, **role-based permissions, versioning, and multi-environment workflows** give enterprises full control during migration and beyond. Teams can manage who edits what, test safely in staging environments, and roll back changes instantly when needed.

Together, these flexible capabilities form a technical foundation designed for migration at scale and one that can evolve with your business.

Content, data, and asset migration

This stage focuses on moving structured content, media, and metadata from your prior CMS into Storyblok efficiently, safely, and with data integrity preserved. For developer-level instructions, sample scripts, and detailed workflows, refer to Storyblok's CMS Migration concept and the accompanying assets migration documentation.

CONTENT MIGRATION

Content migration refers to the process of transferring digital assets and structured content and related data from one system to another, typically during a platform upgrade, website redesign, or replatforming initiative.

A well-planned migration ensures consistent and optimized content structures, improved editorial workflows as well as localization capabilities. On top of this, it ensures clean data models that are easier to maintain and preserve analytical continuity for further acceleration and monitoring.

Storyblok's modular content architecture allows you to treat migration as an opportunity to simplify. Instead of importing redundant templates, you can rebuild components that support multiple use cases, reducing technical debt in the long run. Therefore, before any content is moved, define your **content mapping matrix**. It will serve as a bridge between your current CMS setup and Storyblok, helping you identify what to retain, redesign or retire:

Retain: Keep core, evergreen content that only needs structural adjustments.

Redesign: Rebuild sections that could benefit from a more modular setup.

Retire: Remove outdated or redundant content that no longer supports business goals.

In this map, you must also make a distinction between pages that you will move by hand, and those that you will import with a dedicated tool or a custom script (your chosen migration

method). Typically, you would want to differentiate between the following types of entries:

Manual migration: used for complex landing pages or highly visual layouts. These often rely on nested blocks, varied content types, or non-standard fields, best rebuilt using Storyblok's Visual Editor.

Bulk migration: Ideal for structured, repetitive content such as blogs, product catalogs, documentation, or news articles. Bulk migrations use automation tools (e.g., Storyblok CLI or Management API) to import entries at scale. Namely, there are three common options to bulk migrate entries:

1. **API access (best case):** If the previous platform has APIs for extracting content, you can build a script that pulls the data, transforms it into the required Storyblok structure, and writes it into the new CMS.
2. **Database (DB) access:** Query the legacy CMS database directly if no APIs are available, aggregating the data for import. This can be a problem in some cases, especially if the structure of the tables is very complex, because you will have to take time to understand it and then perform data aggregation queries.
3. **Content scraping (last resort):** If the previous platform is inaccessible or unreliable, you can retrieve content through a script that scrapes published pages. When the HTML structure is consistent, you can map the extracted elements to the appropriate fields in Storyblok.

GOOD TO KNOW

Before migration begins, implement a temporary content freeze in your source CMS. This prevents new content updates or edits during the export window, ensuring the dataset remains consistent. Once the migration is validated in Storyblok, content updates and publishing can safely resume. Remember, that such content freezes should be limited and well-communicated: editors should know when changes in the old CMS will no longer carry over to the new platform.

DATA TRANSFORMATION

Once your content mapping is set, prepare your data so it fits Storyblok's model. A few light conversions ensure imported content displays correctly, is reusable across channels, and follows consistent rules.

- **HTML to structured format:** Convert rich text from the old CMS into Storyblok's Richtext format so developers control presentation and old inline styling does not leak through.

- **Text to structured objects:** Turn plain file paths or links into structured assets and links with helpful details like alt text, IDs, and focus points. That makes media searchable and easier to optimize.
- **Flat to modular:** Break legacy page templates into reusable Storyblok components so teams can compose ayouts and experiences without rebuilding templates later.

For large imports, teams can automate these conversions with Storyblok's CLI or migration scripts.

ASSET MIGRATION

While content migration handles structured entries and content models, **asset migration** focuses on media files, i.e., the images, videos, and documents that bring your content to life. These files are often stored in legacy CMS directories without metadata or optimization, which can complicate re-use and performance.

In Storyblok, every asset file becomes an entry in the Digital Asset Manager (DAM), which includes transformation, privacy, and CDN delivery by default.

Recommended process:

1. **Export assets** from your previous CMS or DAM, preserving metadata such as filenames, alt text, and tags.
2. **Upload assets** into the Storyblok native DAM or your connected external DAM, ensuring you remove any duplicates.
3. **Generate a mapping log** of old versus new URLs to maintain integrity.
4. **Update internal references** via API scripts or find-and-replace processes.
5. **Validate redirects** for any high-ranking media URLs to preserve SEO equity.

Storyblok's DAM automatically handles image transformations (resize, format conversion, focal point control) and works seamlessly with external services for advanced setups.

SEO, GEO AND VALIDATION

A successful CMS migration must protect your **search visibility** while preparing for the new era of Generative Search (GEO) and AI Overviews. Losing rankings during migration is entirely avoidable with early planning and technical continuity.

BEFORE MIGRATION	DURING MIGRATION
• Map and export all legacy URLs, including subdomains and tracking parameters.	• Maintain metadata consistency across titles, descriptions, alt text, and Open Graph tags.

• Create 301 redirects from old to new URLs to preserve existing search authority. • Validate canonical tags to ensure search engines index the correct page versions. • Benchmark Core Web Vitals and Lighthouse scores to establish performance baselines.	• Generate and resubmit XML sitemaps once the new site is live. • Check internal links and content hierarchy for completeness. • Verify that assets and alt text transfer correctly.

Use **Screaming Frog** or **Sitebulb** to confirm redirects, metadata accuracy, and sitemap completeness at scale. Automated parity scripts can further validate that every URL, asset, and tag is properly represented in the new environment.

COLLABORATION THROUGHOUT

Migration success depends on **close coordination** between developers, editors, and SEO teams. Editors are best positioned to validate visual and contextual accuracy, while developers focus on structural integrity and automation reliability.

Establish a shared **UAT (User Acceptance Testing) process** where editors and developers jointly validate imported content directly. If the corresponding frontend components already exist, editors can verify directly inside Storyblok's **Visual Editor**. For new or unlinked schemas, developers should first confirm structural accuracy and field mappings before visual validation. This hybrid verification ensures both the data and the experience match expectations before go-live.

GOOD TO KNOW

Use this phase to train editors in the new system. Reviewing migrated content is an excellent hands-on onboarding opportunity.

Integration overview

This phase focuses on connecting Storyblok to the systems that drive personalization, commerce, translation, and analytics, creating a unified and composable digital stack. Storyblok's headless architecture makes it inherently integration-ready. Connections are API-based, ensuring flexibility, scalability, and long-term maintainability across your technology landscape.

COMMON INTEGRATION LAYERS AND PATTERNS

SYSTEM	PURPOSE	EXAMPLE INTEGRATION
Digital Asset Management (DAM)	Centralize, version and optimize media assets across teams.	**Bynder, Cloudinary, Frontify**
Content Delivery Networks (CDNs)	Deliver content globally with faster load times and edge caching.	**AWS CloudFront, Netlify, Vercel Edge**
eCommerce platforms	Surface product data and connect to real-time inventory from your eCommerce engine.	**Shopify Admin API, BigCommerce, commercetools**
Translation and localization	Enable multi-language workflows and automation.	**Phrase, Lokalise, Smartling**
Analytics and tag management	Connect behavioral and performance insights.	**Google Analytics 4, Segment, Adobe Analytics**
CRM and personalization	Deliver data-driven, tailored experiences.	**Salesforce, HubSpot**

Remember, there is **no one-size-fits-all pattern** for connecting systems: the right approach depends on your data flow needs, infrastructure, and governance model.

Storyblok supports multiple integration patterns depending on your architecture and scale:

- **Direct API Calls:** Use Storyblok's **Management API** and **Content Delivery API** to fetch or push content between Storyblok and other applications. This approach is ideal for websites, apps, or SaaS tools that need live content updates.
- **Webhooks:** Automate actions when content is published or changed. For example, trigger a build in **Netlify**, send analytics updates, or notify a **Slack** channel.
- **Middleware / Integration Platforms:** Larger organizations often use tools like **MuleSoft** or **Azure Logic Apps** to manage data flows between multiple systems. These orchestrate and secure complex integrations at scale.

For teams that want to extend Storyblok's capabilities, the **App** and **Plugin framework** allows developers to build or install custom apps within the **Visual Editor**. This framework enables field plugins that enhance content inputs (for example, linking a product catalog or asset library), sidebar applications that surface external data or analytics dashboards, and custom integrations that streamline editorial workflows. Plugins can communicate with the main Storyblok application via the window.postMessage() method and leverage Storyblok's API as required for the desired functionality.

Refer to the **Storyblok Docs** for the latest guidance on developing and securing custom extensions.

SECURITY AND GOVERNANCE

By removing the need for self-hosted infrastructure, Storyblok eliminates many of the risks that come with traditional CMS setups: security risks, maintenance costs, downtime, and data and compliance risks. Every customer instance benefits from the same monitored, enterprise-grade cloud environment and continuous protection.

Governance best practices:

- [ ] Store API keys and OAuth tokens in a **secure vault** such as AWS Secrets Manager or HashiCorp Vault.
- [ ] Assign **clear ownership** for each integration endpoint and limit permissions to what's needed.
- [ ] Review **webhook configurations** regularly to avoid duplicates or stale triggers.
- [ ] Set up alerts for failed API requests or unusual traffic patterns.

These measures allow migrations and integrations to scale safely, without introducing new points of failure or compliance risk.

Storyblok's managed infrastructure combines platform-level security with customer-side governance, giving teams the confidence to scale without complexity. Some key safeguards:

- **ISO 27001 certified, and SOC 2 Type 1 certified.**
- **Hosted on Amazon Web Services (AWS)**, using the same infrastructure standards trusted by global enterprises.
- **Continuous system monitoring** enhanced by automated and even AI-based threat detection.
- **Web Application Firewalls (WAFs)** and strict change-control processes to prevent unauthorized access.
- **OAuth 2.0 authentication** for all API-based integrations.
- **Regular penetration testing and third-party security audits.**

Testing, QA, and go-live

Manual QA can't keep up with large volumes of content. Integrate **broken-link checkers, accessibility audits,** and **spellcheckers** into your CI/CD pipeline to catch issues early.

In general, QA should cover:

- **Functional testing:** navigation, interactive modules, forms, and API responses.
- **Content verification:** layout accuracy, media rendering, and correct component use.
- **SEO continuity:** metadata, redirects, schema markup, and crawlability.
- **AI preview:** test how content surfaces in AI Overviews or assistants.

- **Accessibility (a11y):** compliance with WCAG 2.2 AA and applicable regional requirements such as the European Accessibility Act.
- **Performance:** Core Web Vitals, Lighthouse, and CDN optimization.

EXAMPLE QA AND VALIDATION TOOLS

Integrate quality checks into your CI/CD pipeline so errors are caught before publishing.

AREA	COMMONLY USED TOOLS
Visual and functional QA	Cypress.io, Percy, Playwright
Performance	WebPageTest, GTmetrix, Pingdom, Lighthouse
SEO and Links	Screaming Frog, Ahrefs
AI discoverability	OtterlyAI, MarketMuse, SurferSEO
Accessibility	axe-core, Storybook, Deque
Content integrity	Diffy, ContentKing

Use **previews** to validate layouts across devices before you hit publish. Automation ensures consistent checks while freeing your QA team to focus on edge cases and user experience validation.

GO/NO-GO FRAMEWORK

Before any production deployment, define **clear acceptance criteria** and a rollback plan.

Go/No-Go recommended checklist:

- [ ] All critical bugs resolved and regression tested.
- [ ] Redirects mapped and validated.
- [ ] All WCAG 2.2 AA checks passed.
- [ ] Lighthouse performance score ≥ 85 %.
- [ ] Backup and rollback environment ready.
- [ ] Business owner sign-off secured.

If a blocker appears during launch, pause deployment and use the **rollback procedure**. For content issues, use Storyblok's **History feature** to restore a previous content version instantly. For deployment-level issues (e.g., broken builds or failed integrations), revert to the last validated release in your hosting provider, such as Vercel or Netlify, document the findings, and redeploy once resolved after. Storyblok's version control ensures content rollback is near-instant.

POST-LAUNCH SEO, GEO, AND PERFORMANCE OPTIMIZATION

Maintaining and improving visibility after migration requires active monitoring and precision. Storyblok's architecture supports both traditional SEO and emerging GEO (Generative Engine Optimization), ensuring your content performs for humans, search engines, and AI systems alike.

Best practice to keep in mind:

SEO	GEO AND AI OPTIMIZATION
• Configure meta tags (title, description, Open Graph) for every content type. • Preserve URL structures where possible; implement **301 redirects** for legacy paths. • Use **canonical tags** to define preferred versions of pages and avoid duplicates. • Apply **structured data** (schema.org) for rich search results in SERPs. • Generate and resubmit updated **XML sitemaps**. • Optimize images in the DAM with descriptive filenames and **alt text**. • Use Storyblok's **global CDN** for fast asset delivery, and apply the **Image Service** to optimize image formats, sizes, and performance.	• Ensure the frontend outputs clean, semantic HTML so search engines and AI systems can easily parse your content structure. • Enrich entries with contextual metadata and consistent naming for entities like people, products, and locations. • Use clear, natural phrasing in titles and summaries (AI systems prefer concise, human-style writing). • Link related entries using relationship fields to build semantic connections across sites and locales. • Follow **E-E-A-T** principles: show experience, expertise, authoritativeness, and trustworthiness. • Test how your content surfaces in **AI Overviews**, and generative search tools.

After go-live, run **Lighthouse**, **Google Search Console**, and **Core Web Vitals** audits to track progress, and use **Netlify** or **Cloudflare Insights** for ongoing page-speed and uptime monitoring. Review these metrics quarterly and adjust as needed.

ACCESSIBILITY AND COMPLIANCE

Accessibility ensures every user can interact with your content without barriers. It's both a design principle and a continuous quality process, and increasingly, a legal requirement. Prioritizing accessibility early creates inclusive experiences, safeguards compliance, and reduces the need for future rework.

When building or migrating your site, **incorporate accessibility into everyday workflows rather than treating it as a final check**. Automated testing tools such as **axe**, **Pa11y**, or **Lighthouse** can help identify common issues quickly, while manual reviews confirm real-world usability. This dual approach gives you the coverage needed to maintain compliance as standards evolve.

At a minimum, your migration plan should include a short accessibility statement confirming alignment with recognized international standards such as WCAG 2.2 AA or newer equivalents, and a commitment to continuous monitoring and improvement.

Recommended checklist:

- [ ] Text alternatives present for images, video, icons, and charts.
- [ ] Headings follow a logical order (H1 - H6) and describe section purpose.
- [ ] Keyboard navigation works for all interactive elements. Focus states are visible.
- [ ] Color contrast meets WCAG 2.2 AA.
- [ ] Avoid conveying meaning by color alone.
- [ ] Forms include labels, instructions, and error messages that can be read by assistive tech.
- [ ] Dynamic content announces changes properly to assistive tech.
- [ ] Language attributes, page titles, and landmarks are set correctly.
- [ ] PDFs and downloadable assets meet accessible document guidelines or have HTML equivalents.

Accessibility should remain an ongoing discipline. Schedule periodic audits, train editors on best practices, and review new templates or components before deployment to ensure consistency across every channel.

RISK AND MITIGATIONS

Every migration carries inherent risk. **Proactive mitigation prevents rework and hiccups.** The table highlights the most common risks and recommended strategies to address them:

RISK	IMPACT	MIGRATION STRATEGY
Data loss during import/format issues	High	Use incremental imports to minimize batch errors, run automated content parity scripts to verify data consistency after each import cycle, and create backups throughout.
SEO and GEO / AI search ranking drop	Medium	Validate redirects and canonical mappings before launch, and perform metadata parity checks to preserve SEO authority and AI discoverability.
Integration failures	High	Conduct contract testing across APIs, and use integration stubs to simulate dependencies during pre-production validation.
User adoption gap	Medium	Provide hands-on training and sandbox play sessions to build confidence, supported by clear SLAs for post-live support.
Performance degradation	Medium	Leverage CDN caching for faster delivery and monitor Core Web Vitals to catch any performance regressions early.

To track readiness, convert this table into your own personal **traffic-light dashboard**, marking each risk Red / Amber / Green based on readiness before go-live. Such a visual summary can assist with any future executive sign-offs.

CONTINUOUS MONITORING AND SUCCESS METRICS

A migration doesn't end when the new site goes live.

The post-launch phase is where the real insights surface.

Monitor platform performance through tools like **Netlify**, **Cloudflare**, or **New Relic** to track uptime and performance, and **Core Web Vitals** for front-end health. Use analytics and search console data to verify that SEO performance meet or exceed the pre-migration baselines.

Finally, measure just how quickly editors can go from draft to live publication compared to the previous CMS setup, to track improvements in efficiency and agility across workflows.

Suggested success metrics:

METRIC	WHY IT MATTERS
Time to first live page	Measures how quickly the new setup delivers tangible results after migration.
Deployment frequency	Indicates how often teams can safely release new content or updates compared to the legacy system.
% automated migration	Reflects process maturity and reliability of data transfer.
Editor time-to-publish delta	Indicates workflow efficiency and content team agility.
Lighthouse score improvement	Reflects front-end performance and asset optimization.
Total Cost of Ownership (TCO) reduction	Tracks system efficiency, plugin reduction, and maintenance savings.
Incidents in first 30 days	Confirms rollout stability and environment reliability.

Note: benchmarks will vary by organization, site complexity, and content scale. Use internal baselines from your previous setup as reference points.

Schedule **30** and **90-day post-launch reviews** to analyse progress and identify optimization opportunities. Treat this as a continuous improvement cycle rather than a project closeout. Storyblok's modular setup will let you iterate quickly without disrupting live content.

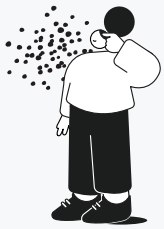
PART IV

Conclusion and next steps

The shift to Storyblok isn't just a technical migration, it is a chance for a creative reset, as your teams gain the freedom to build, adapt, and scale without friction.

If you've reached this point, you'll already have a clear sense of what a migration involves. Here is how to move forward:

1. **Define Scope and Targets:** Identify the initial site or project to migrate. Our team can help you assess feasibility, timeline, and content readiness in line with our scoping questions in this guide.
2. **Book a Discovery Session:** Work with our Solution Engineers to review your existing setup and map your path forward.
3. **Run a 2-Week Pilot:** Validate your migration plan. Assess data mappings, run your first automated import, and confirm team roles before scaling to production.
4. **Expand and Optimize:** Once your pilot succeeds, replicate the framework for future sites, brands, or markets.



READY TO PLAN YOUR MIGRATION?

CONTACT US

Power your migration with expert support

If your team needs implementation support or migration expertise, Storyblok's certified partners can help. Every migration is unique, and our partner network provides flexible packages covering planning, integration, localization, and training.

Explore partners at **storyblok.com/partners** or contact your **Customer Success Manager** to find the right fit.

PART V

Appendix

Tools and resources

Storyblok provides a comprehensive ecosystem of documentation, templates, and starter kits to accelerate migration and onboarding. This includes **Storyblok Blueprints**, our pre-built selection of CMS setups that include content models, folder structures, and components, as well as a plethora of further technical resources right within **Storyblok Docs**, our documentation hub. This hub has an extensive array of User Manuals, and provides essential information for all Storyblok users (not just devs!). Specifically, you can find,

- **CMS Migration:** more on Storyblok migration, platform-specific tutorials, recommended extraction and import workflows, and example scripts for content and asset migration.
- **CLI References:** commands for schema sync, migrations, and bulk operations.

Don't forget that you can take advantage of our **off-the-shelf technology ecosystem** to build a best of breed tech stack, and then take the next step with our extensive **App Directory** for any field extensions and productivity apps.

Procurement via AWS Marketplace

For organizations already using AWS, Storyblok can be purchased directly through **AWS Marketplace**. Note, credit eligibility and terms depend on your organization's AWS program.

Benefits include:

- Simplified procurement and consolidated AWS billing
- Ability to apply existing **AWS credits or committed spend** toward Storyblok
- Centralized vendor management and security compliance via AWS

How it works:

1. Visit Storyblok on AWS Marketplace.
2. Choose your preferred plan and subscribe.
3. Contact your AWS account team or Storyblok representative to enable private-offer pricing or credit application; **alliances@storyblok.com** can offer current offers and guidance.

Note: Storyblok is a verified AWS Marketplace SaaS product. Terms, pricing, and promotions may change at any time.

Glossary

TERM	DEFINITION
Application Programming Interface (API)	An application programming interface (API) is an interface that defines interactions between multiple software applications. These interactions or “calls” follow a specific set of rules and can be extended by the user to add functionalities. APIs enable applications to communicate with the server and different channels.
Digital Asset Management (DAM)	A digital asset is any media file such as an image, video, or document. DAM is the process of storing and rendering rich media for both administrative and operational tasks. DAM allows users to search for every piece of information and establish permissions to protect their information from cyber attacks.
Block / Component	Modular content units in Storyblok used to structure and deliver reusable content across pages, apps, and digital experiences.
Continuous Integration / Continuous Deployment (CI/CD)	Automated testing and deployment workflows that validate and publish updates safely and quickly.
Composable Architecture	A modular tech stack built by combining independent best-of-breed services through APIs.
Content Delivery Network (CDN)	A geographically distributed network of proxy servers and data centers. A CDN enables a quick transfer of assets needed for loading content.
Command Line Interface (CLI)	A terminal-based tool used by developers to run Storyblok management commands, automate imports, and perform migrations.
Canonical Tag	An HTML element that identifies the primary version of a page, preventing SEO duplication issues.
Deployment Environment	Distinct instances (staging, testing, production) that isolate work for safe release management.
Edge Caching	Storing content on CDN edge servers for ultra-fast delivery and minimal latency.
Frontend Framework	JavaScript frameworks such as Next.js, Nuxt, or Astro used to render content pulled from Storyblok.
GraphQL	A query language for APIs that allows precise structured data requests. Storyblok supports GraphQL delivery queries.

Governance Model	Defines permissions, workflows, and approval rules for teams managing content.
Headless CMS	A CMS that decouples backend content management from frontend presentation, enabling omnichannel delivery.
Metadata	Descriptive information about content (e.g., title, tags, descriptions) that supports SEO and internal organization.
Redirect Map	A list mapping old URLs to new ones to preserve SEO and user navigation during migration.
Rich Text Field (RTF)	Storyblok's JSON-based format for storing structured rich text, replacing legacy HTML fields.
Schema	The structure defining how content types and their fields relate within a CMS.
Search Engine Optimization (SEO)	The practice of improving content visibility in search engines while maintaining existing equity during migration.
Generative Engine Optimization (GEO)	An emerging discipline focused on optimizing content for visibility within AI-driven search experiences (e.g., ChatGPT, Gemini, Claude, Google's AI previews and such AI search platforms).
Space	A workspace in Storyblok that contains a project's content, components, and settings.
Static Site Generation (SSG)	Pre-rendering site pages at build time for improved speed, stability, and SEO performance.
Total Cost of Ownership (TCO)	The full lifetime cost of a platform, including licensing, development, hosting, and maintenance.
User Acceptance Testing (UAT)	The final testing phase where stakeholders confirm that the system meets business and technical requirements.
Webhook	An automated message triggered by specific events (e.g., "publish") to sync Storyblok with other platforms.

Demystify more terms with this article.

Get Joyful.

See how Storyblok can revolutionize your content at storyblok.com.

storyblok