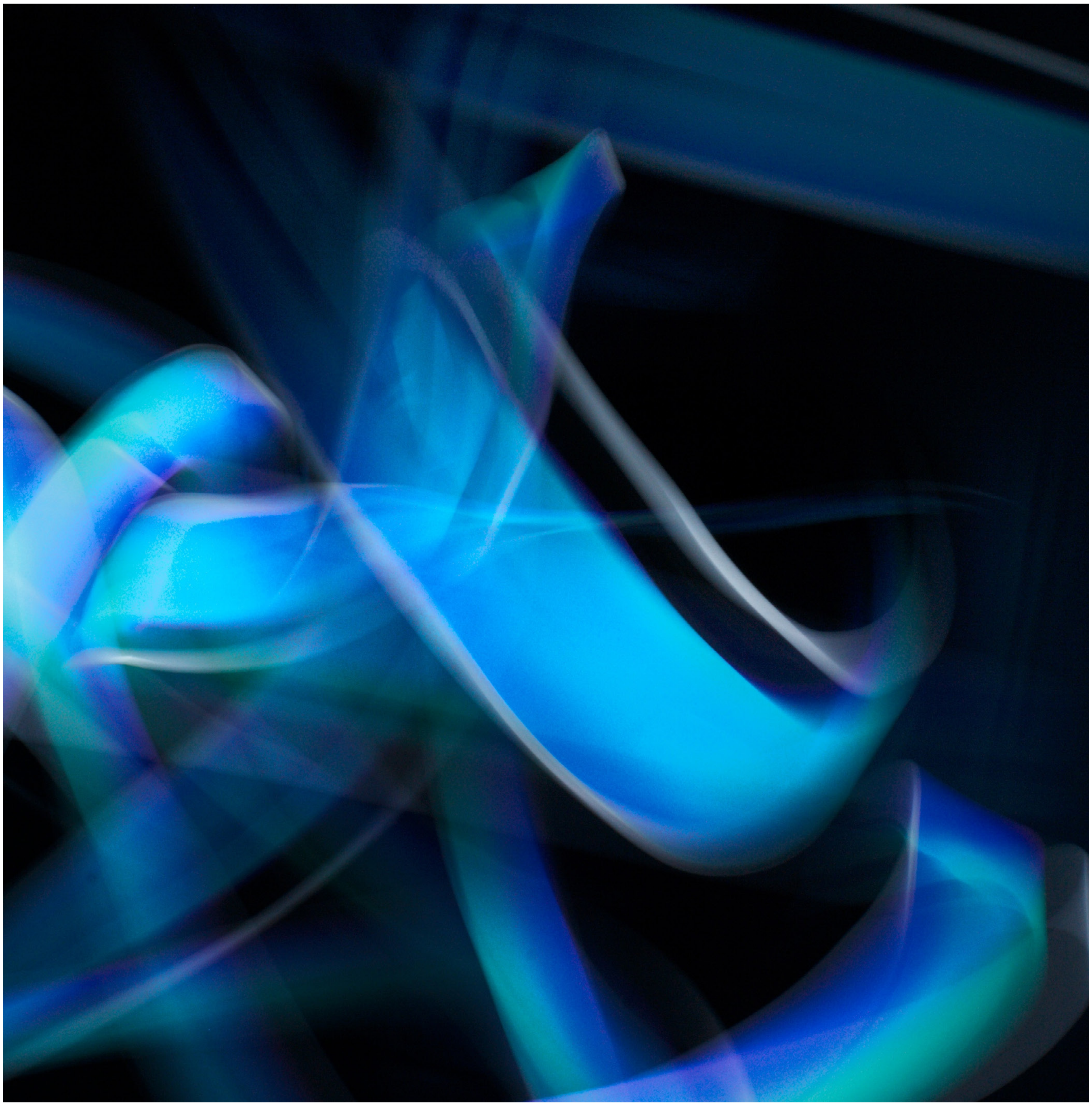


WHITE PAPER

Unlocking Flexibility and Scalability: A Multi-Tenant Content Platform Approach for Information Providers

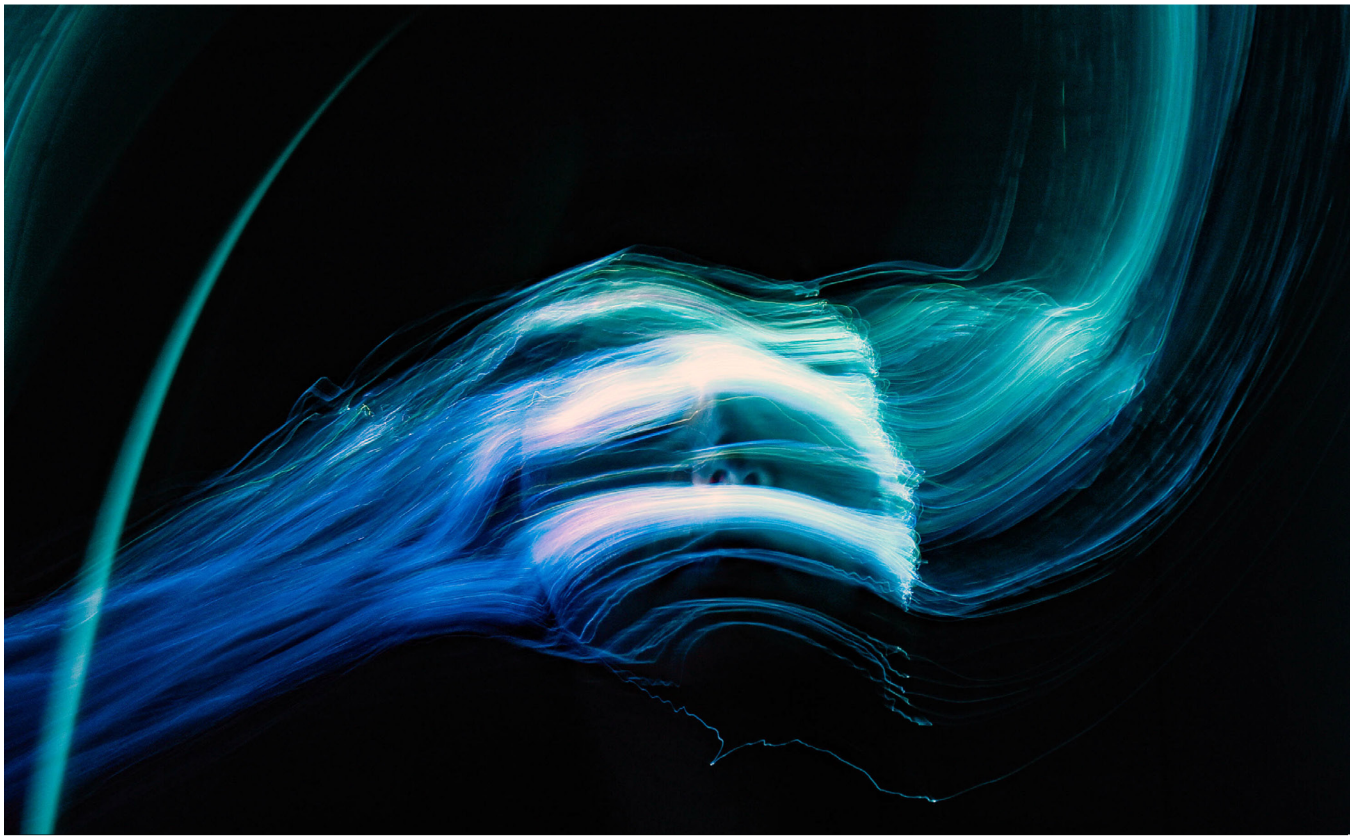
Contents

Introduction

Today, information is a precious commodity, and the enterprises that produce, curate and sell information have a lot of moving parts to manage in order to successfully operate. These organizations are often global, multi-disciplinary and contain numerous business units, many via acquisition. Their content is abundant, complex and dense, and can influence billions of dollars in investment. Typically, all of this content is housed in the enterprise's legacy systems.

It's a complicated ecosystem – a recipe for entropy and barrier to scaling or innovation. For content architects and content platform professionals, this Sisyphean set of problems traditionally had no graceful solution on an enterprise level… until recently. Many "enterprise" platforms were never properly integrated with the rest of the organization, or did get out the door just to fizzle after onboarding a few products or business units. Then the latest tech disrupts the process, and the cycle begins again. Today, we can attempt to use cloud, agile and other contemporary techniques, treating an information provider business unit or product like a startup, but that only adds to the messy web of siloed systems.

The answer may be to ignore the "enterprise" label and borrow a method from SaaS vendors. This paper outlines an approach for large multi-faceted information providers to model their core content platform as a multi-tenant set of assets with a common core. We'll use Alfresco and AWS to make some very concrete points, though the recipe can be baked with many other flavors.



An Incumbent Problem

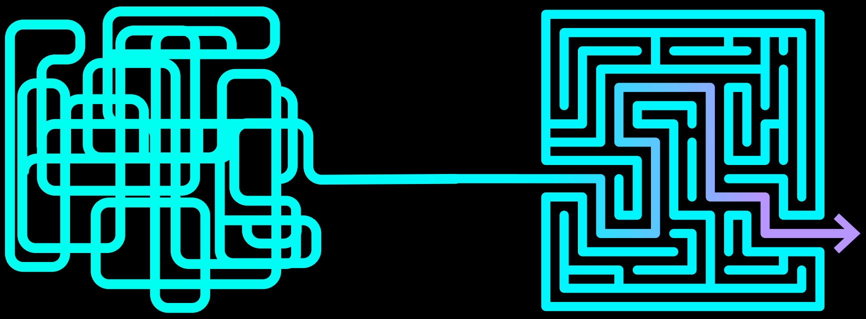
The story is most complex for information providers that have been around the longest. Between all the various systems and formats that have been thrown into the mix through years of acquisitions and building new business units, the resulting technology portfolio is rigid and unforgiving. Even making one small change can turn into a herculean effort, and completely starting over would erase billions in profits.

Additionally, the B2B content stream isn't optimized to flow in one direction like a B2C entertainment media model, for example. B2B information providers that make millions or billions through fees and subscriptions will often have entire books, journals, articles, videos, audio and other unstructured content that is connected to aggregated public or licensed content and the open web. A single screen or page can contain tens of active relationships across billions of content items and petabytes of digital volume.

Gigantic content repositories are not easy to have anywhere, let alone in the cloud. Then, connecting those platforms to hosts of other business systems that haven't yet justified a move to the cloud adds to the complexity. There may even be more than one content repository to consider, or several flows of content in multiple interim repositories, and even many products that cross over many segments of content. These also connect to operations, financials, royalty and other back-office systems. It's quite a web, and not the good kind.

In the following illustration, the left portion characterizes a typical current state for information providers: siloed, heterogeneous, complex, brittle, redundant and arcane, requiring too broad a range of skills, too much undocumented institutional history across the technology team, and too much unproductive work by content teams. Worse, tech dependency makes fluid assignment of staff practically impossible. On the right, we have a more manageable and scaleable target state with shared common elements and differentiation enabled at low cost to support the variety of unique needs across various divisions, business units and products in the enterprise. It also makes new content, products and revenue streams possible faster and at lower cost by leveraging common core, coordinated semantic models and good governance. Innovations spread faster and are more easily shared where applicable elsewhere.

CURRENT STATE TO TARGET STATE: AN ILLUSTRATION



An Incumbent Problem (Cont.)

Content systems like Alfresco, MongoDB, RethinkDB, Couchbase and many similar commercial software packages can provide the core of a content repository, but this challenge also requires deep engineering capabilities, flexible and rapid development methods, and a compatible corporate culture that is matched by your vendor partners. Let's take a closer look at some of the key characteristics of the enterprise content companies described above that could benefit from implementing a next-generation content system:

01

Each division has its own p&l and c-level leader, as well as its own unique content (a mix of any of legal, tax, accounting, real estate, health, insurance, investment research, patents, science research, engineering, risk and more)

02

Each division has overlapping feature needs in mastering content for products

03

Some of the content can be related and linked, providing opportunities for derivative products and new revenue (as long as all divisions are using consistent semantic markup and techniques)

04

Global implications like multi-language, jurisdictional or geographical content dependencies - even regulatory compliance - vary by country, state/province, and supra-national standards and regulations

05

Authoring and editing is often complex and involves tens or hundreds of participants, many of whom are outside the firewall and contracted (not in-house employees)

06

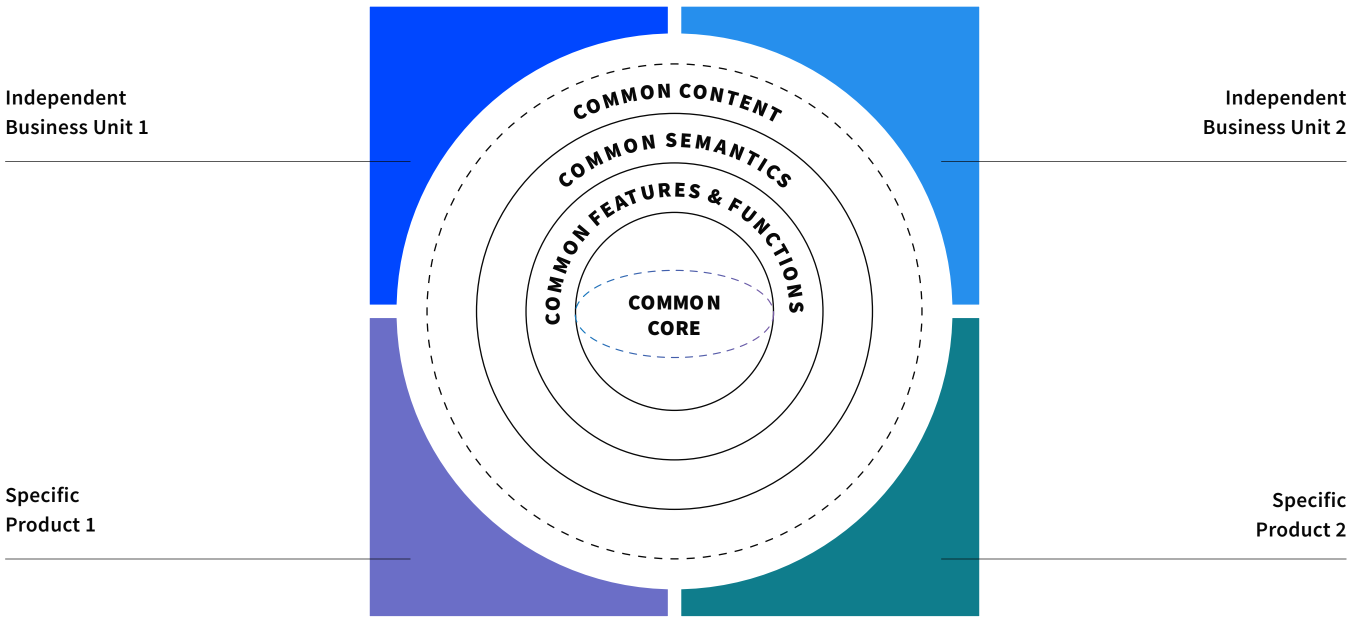
There is a need for (and a potential revenue stream associated with) knowledge bases that underlay content within a domain, or across domains, with a need to maintain variations in knowledge over time, and relative to specific content

Managing Multi-Tenancy Platform Assets for the Enterprise

First, consider a fundamental architecture that enables organizations to treat each division or product as unique entities, but with shared feature needs and common content semantics. We think of multi-tenancy as a commercial SaaS practice that brings huge benefits to companies like Salesforce, Adobe, Workday and Dropbox. But why not leverage multi-tenancy within the enterprise to provide both global efficiency and local flexibility?

The following illustration shows how a core enterprise set of features, semantics and common content can be extended for unique aspects of many products of divisions. These assets can be managed by a single platform team that complements governance as a way of disciplined unity across the business' P&Ls by product, region or division.

TARGET STATE COMMON CORE WITH BU/PRODUCT VARIATION


Independent Business Unit 1
Independent Business Unit 2
COMMON CONTENT
COMMON SEMANTICS
COMMON FEATURES & FUNCTIONS
COMMON CORE
Specific Product 1
Specific Product 2

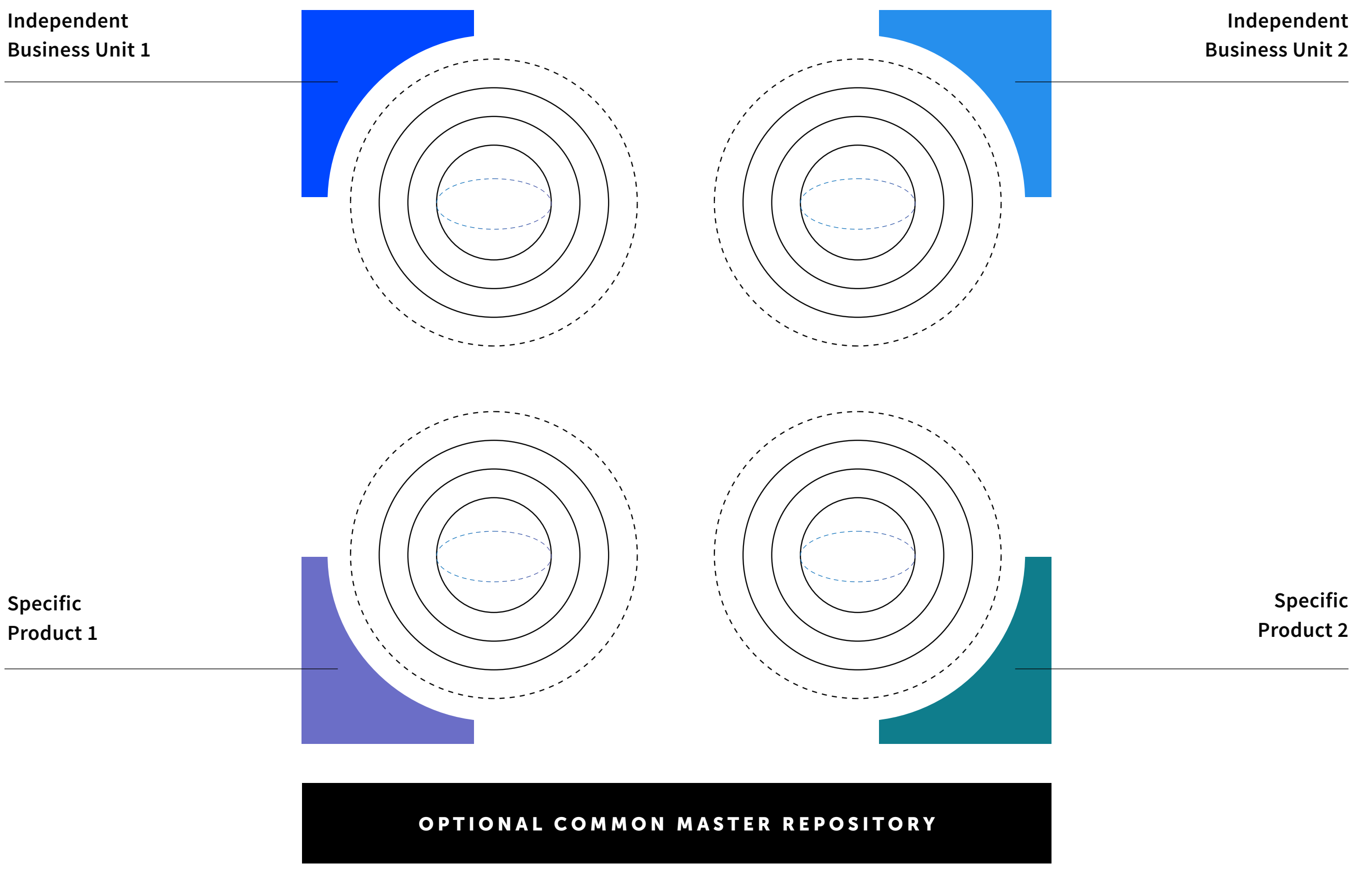
Independent Business Unit: with unique content, features, functions and semantics & shared common core
Specific Product: with unique content, features, functions and semantics & shared common core

Deploying Instances of Multi-Tenancy Platform Assets for the Enterprise

Having the baseline and catalog of specifics, you then have to deploy instances for each tenant. These tenants may be regions, divisions, or an individual product. However, via governance, it is helpful to avoid proliferation of similar tenancies just as one might with other more common SaaS commercial offerings such as CRM.

That said, the following illustration depicts how to deploy assets as tenancies using Alfresco in AWS.

TENANT DEPLOYMENT


Independent Business Unit 1
Independent Business Unit 2
Specific Product 1
Specific Product 2
OPTIONAL COMMON MASTER REPOSITORY

Deploying Instances of Multi-Tenancy Platform Assets for the Enterprise (Cont.)

AWS containers can be used as a means to multi-tenancy. The Alfresco code is architected as a set of reusable core components and features. Each tenant gets a repository for their unique needs in features and semantics. In addition to good governance, appropriate tooling and discipline is needed to isolate configurations and customizations. Everything shares a core Alfresco baseline and common repository features. Each division or product gets its own Docker container deployed with the common Alfresco stack and it's unique elements.

These techniques can help organizations scale across disparate businesses and products more quickly while reducing maintenance — and the results are significant. When changes to one group's tenancy are isolated from all others, deployments can be isolated to the tenant, thus reducing the time and cost for operations. Loosely coupling the core stack and unique group configurations and customizations also reduces code defects, as well as testing and deployment complexity. In some cases, a total tenant build can be avoided altogether. If there are a few divisions and/or countries, tenants can be scaled independently.

Moving on to operations – support, incident management, necessary outages, breakfix and other areas can be localized to a narrow segment of users. It's also a good idea to loosely couple data model customizations in this scenario. Because of these discreet containers, even Alfresco updates and upgrades become easier.

One caveat is that the content is also stored independently for each tenant. So, to enable multiple cross-disciplinary products that reuse the content, you may want to add an optional master repository into the architecture. Implementing economic tenant outflow and some normalization upstream to a master can make operations like global enrichment and unified ontology, taxonomy and vocabulary much easier. It's a bit more investment, but the complexity, cost, operations, curation, development, testing and deployment that come along with sourcing from many places can be reduced because sources become significantly more consistent before the content heads downstream. This results in significantly faster and cheaper new content product innovation.

Docker brings substantial benefits to development in general, but especially to this sort of multi-tenancy that balances content core consistency with instance variation. Additionally, Docker enables control groups of processes to be bound together in a collection. This unlocks the ability to design and configure features/functions for a tenant in ways that increase control without compromising flexibility. Furthermore, there are runtime benefits and advantages across all supporting activities, including deployments, environment management, recovery, redundancy and testing.

That said, there is a complication with Java applications because of the Java Virtual Machine (JVM). Linux features namespace isolation, which allows control groups to run without awareness of other groups and their resources in Docker. JVM, on the other hand, sees all resources in an Amazon EC2 instance. JVM will grab resources according to how it is configured with a variety of server-level parameters including share of server, thread count, CPUs, etc. It is important that any JVM processes are configured to use less resources than the parent Docker configuration. To do this applications must be compliant with Java Runtime Environment (JRE) version 10 or higher, which can add some extra work when refactoring existing services.

Content, Authoring, Editing and Productizing

For information providers, content is king and their complex content adds a whole new element of intricacy. Because content is so highly valued, investment in it often eclipses investment in technology systems. For example, a system costing millions could house content that costs billions. This content is often stored as JSON, XML or domain-specific flavors like JATS, MOL or XBRL and includes knowledge organization references like OWL, SKOS, taxonomies, vocabularies and more. It's often architected with sections, segments, clauses and standard notations, and references. It can include reusable content items, and even act like computer objects with overloading and overriding.

If you have a picture of a business document costing four or five figures that, in mark up, looks like a 10,000-piece crossword puzzle, you are on the right track. Now, imagine 100+ employees and contractors working on it from authors to editors, reviewers to approvers. From there, it's digitally productized through multiple channels, perhaps even through print production. Add in the edge cases of users working across tenancies - meaning multiple business units, imprints, brands or geographies and/or having multiple sessions open – then slamming the laptop closed and heading for the airport.

First let's knock out the edge case of a user that crosses divisions or products, and has to work in multiple tenants. It's helpful to a single user to live in multiple tenancies with multiple roles. This is where control groups shine, along with a master set of services independent of any one tenancy.

Having implemented complex permissions, we now need controls to gracefully allow our users to work while acknowledging that they have competing activities outside of a system in their workday, work night, work weekend, work travel and maybe even during personal time. It's also common to work offline.

To start, an Elastic Load Balancer (ELB) provides light and extensive ways to monitor user activity in intervals from five to 60 minutes, for example. Both time intervals may be useful. An editorial system for team authoring must account for an odd set of events that occur outside the scope of the system, yet can dramatically impact collaboration, such as a simple lock on something many people need. Some of the authors may be managing directors or key revenue enablers. So, when they dash off to the airport without unlocking a section of content they checked out, the system must be equipped to handle it without losing the work of this author, or holding up the valuable time and work of other users touching this content.

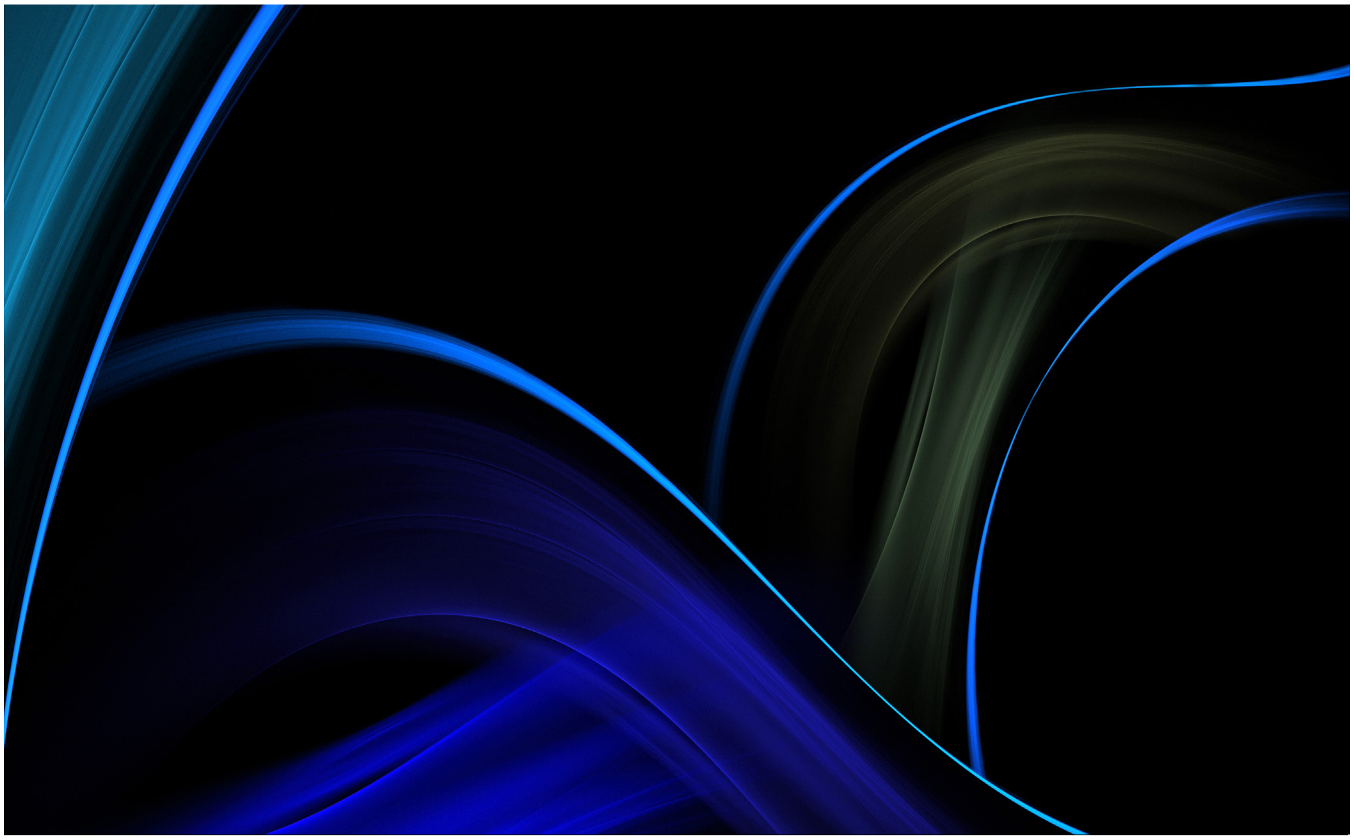
With analytics on top of log processing, an ELB can provide a reasonable impression of the user experience while active. Other functional features like locks, heartbeat services, key activity, auto save and replication combine with the logging to help your system manage real-world states without compromising your stateless architecture. Assuming your content architecture includes segments, when a user explicitly checks out a chunk or implicitly does so by positioning a cursor or beginning to type, you can copy that chunk behind the scenes to a work-inprocess (WiP) store while maintaining sequence.

Content, Authoring, Editing and Productizing (Cont.)

An immediate replica gives before and after comparison capabilities as well as markup and conflict resolution between multiple users. The beauty of this approach is that it enables concurrent editing of the same segment or chunk. Yes, modern cloud-based office productivity apps like Office and Google Docs provide this functionality. However, the typical use case is a few intermittent users or just two users collaborating in real time, but that's just managing two cursors and nothing to the degree of complexity we've been discussing.

When ordinary work is complete and check-ins begin, this technique enables a smooth merge and, when parallel editing is enabled, conflict resolution occurs at the character level. Now, consider alternative paths. This complex engineering allows for seamless, large-scale collaboration and recovery. By recovery, we don't mean system failure, but recovery from interrupted work in the real world as mentioned above. For example, I leave London for a layover in New York, open my laptop and continue working for a bit before the flight to LA. Meanwhile back in London and Dubai, my colleagues have ignored or unlocked my chunk, made edits and checked back in before their flights to Tokyo. In LA, I promote my WiP. No editing conflicts, no problem. Overlapping edits get routed to a curator for resolution. This sort of resolution is a business requirements option, not a system failure.

That level of overlapping arrows in the workflow becomes a bit easier with a more advanced repository at the infrastructure level. You may want to swap out Amazon S3 storage in your infrastructure for their Elastic File System (EFS) or perhaps SoftNAS network access storage.



Conclusion

Let's take a look at the big picture. Previously, attempting to solve this problem may have entailed a massive enterprise analysis and then a monolithic editorial system implementation or purchasing a twentieth-century document management system to archive finished documents. Now, the underpinnings are ubiquitous, so we can solve for unique niches while ring-fencing commonalities between platform features and content semantics.

By establishing a roadmap that starts with one or two business units and grows, we may never again have to approach this from the top down. Of course, this is predicated on a disciplined, agile and lean SDLC with continuous integration and DevOps (CI/CD) that all support modern business continuous improvement (CI). To be successful, engineering excellence is mandatory, most likely with the help of a vendor that has expertise around the editorial market. However, technology is never as complex as people, and strong governance and discipline must also be implemented along with a culture of collaboration across business units and geographies.

Assuming you're ready, those who know this domain will understand that the features highlighted above will appeal to leaders, authors and editors struggling with ever more complex content, lagging speed to market for new products and distribution across many channels, and avoiding cannibalizing legacy revenue. A disciplined, well-governed approach across siloes is a precursor to increasing ROI at scale. The reusability and interoperability reduce the cost of new revenue from new products, new markets and new customers. True reuse of content with a decreased scale of editorial costs becomes routine, with siloes being the exception. This delivers a ski-sloped ROI on content for large content companies.

Technology has changed radically in the last decade, and many incumbent information providers have yet to step into or fully complete a transformation to contemporary technology. Well-governed information providers have both internal technology pragmatists and key service vendors that bring both industry expertise and technology engineering knowledge from other domains.

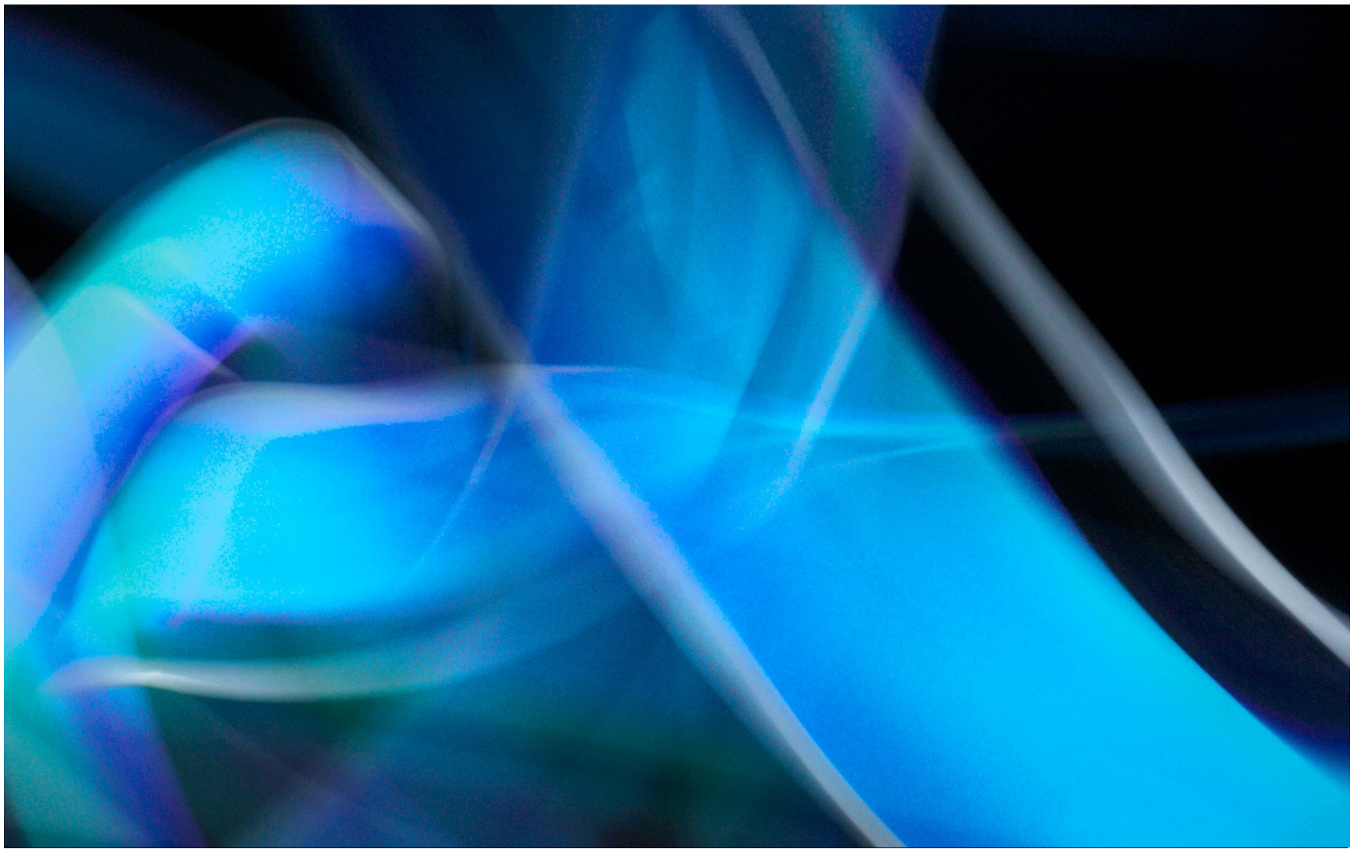
About EPAM Systems

Since 1993, EPAM Systems, Inc. (NYSE: EPAM) has leveraged its advanced software engineering heritage to become the foremost global digital transformation services provider – leading the industry in digital and physical product development and digital platform engineering services.

Through its innovative strategy; integrated advisory, consulting, and design capabilities; and unique 'Engineering DNA,' EPAM's globally deployed hybrid teams help make the future real for clients and communities around the world by powering better enterprise, education and health platforms that connect people, optimize experiences, and improve people's lives. In 2021, EPAM was added to the S&P 500 and included among the list of Forbes Global 2000 companies.

Selected by Newsweek as a 2021, 2022 and 2023 Most Loved Workplace, EPAM's global multidisciplinary teams serve customers in more than 50 countries across six continents. As a recognized leader, EPAM is listed among the top 15 companies in Information Technology Services on the Fortune 1000 and ranked four times as the top IT services company on Fortune's 100 Fastest Growing Companies list. EPAM is also listed among Ad Age's top 25 World's Largest Agency Companies for three consecutive years, and Consulting Magazine named EPAM Continuum a top 20 Fastest Growing Firm.

Learn more at **www.epam.com** and follow EPAM on **Twitter** and **LinkedIn**.



EPAM Systems, Inc.

41 University Drive, Suite 202
Newtown, PA 18940, USA

P: +1-267-759-9000
F: +1-267-759-8989