Designing a Modern Platform Architecture for Content Services

A clear path for moving away from legacy information management systems

rivetlogic

→ nuxeo.com

NUXEO

Designing a Modern Platform Architecture for Content Services

Introduction

Most Enterprise Content Management (ECM) systems were architected during an age when mobile phones were the size of bricks, when Miami Vice pastel suits were all the rage, and Joe Montana was winning Super Bowls. So is it any wonder that these systems are not able to cope with the modern, digital-first environment we find ourselves in today?

This white paper explores some of the modern technologies available today that can be leveraged to get more value from enterprise data and content—technologies that had not even been thought of when most legacy ECM systems were built. These technologies individually provide the potential for huge advances in how we create, manage, process, access, analyze, and secure information within our enterprises—but when used together, they combine to create the most innovative, visionary, flexible, scalable Content Services Platform (CSP) on the market—Nuxeo.

Build or Buy?

Many organizations consider the build vs. buy question, and to help you answer the question, this whitepaper details the various components that a modern organization needs to consider when building, or buying, a solution to manage its information assets.

One of the key questions faced today by CIOs, Chief Architects, and other IT leaders within all industries is whether to build their ideal information management architecture themselves, or to purchase and deploy solutions from outside vendors. The natural inclination for many at first is to build it. However, this takes time and resources, and often fails due to the complexity and the skills required to create a home-grown information environment that addresses the needs of today while having the flexibility to continue to adapt as the business and technologies evolve.

For IT leaders contemplating which path to take, or who are looking to pivot away from the build-it path, this white paper will provide you with knowledge about the key architectural components to consider when evaluating vendor offerings for delivering a modern, future-proof information management environment.

If you choose the build route, then each of these components will need to be configured, integrated to other components, linked to standardized security mechanisms, then tested for scalability, robustness, user-acceptance, and ease of use. Should you wish to consider the "buy" route, each section of this white paper has a "Nuxeo Specifics" sidebar highlighting how the Nuxeo platform uses the component or technology in that section. Whether that is how we use NoSQL databases to provide the ultimate flexibility and scalability, or our cutting-edge work with low code development to allow users to rapidly develop and deploy custom applications, Nuxeo has put its own spin on the most innovative and useful technologies of today.

Over 15 years of software development, 1 million plus lines of open-source code, and thousands of man hours have gone into creating the industry's most modern content services platform. We are a company full of innovators, software engineers, architects, and agents of change. We are people who love to take the latest and greatest technology tools and make good use of them—and that is exactly where Nuxeo has come from.

A Modern Content Services Architecture

A modern Content Services Platform architecture requires many pieces, all needing to work in harmony. It must be scalable and secure, and ultimately enable you to better manage and derive value from your organization's information assets.

There are five key architectural components and concepts that form the core of a modern content services architecture:

- Cloud Native
- NoSQL Database
- Low Code Application Development
- REST API Enabled
- Open Source

Each of these are explained in this white paper, and the Nuxeo Difference sidebars illustrate how they come together to serve as the architectural underpinnings of the Nuxeo Content Services Platform.

5 Key Architectural Features

1. Cloud-native

One of the problems with many legacy software applications today is that they were built 10, 20, even 30 or more years ago and designed as a single—often referred to as monolithic—piece of code. This causes a number of issues:

- Updates, even minor ones, have to be applied to the whole program which is a costly and time-consuming process
- If one part of the software is heavily loaded—then the whole system slows to accommodate it
- These systems try to build everything into one system—so for ECM this means capture, workflow, analytics, reporting and so on. The footprint of the system gets bigger and bigger and bigger—taking longer to install, longer to debug, more time to support and harder to re-engineer to take advantage of new technologies such as cloud and mobile

So, what is the alternative?

Despite being touted as a concept since early 2005, modular software architecture has only really taken off with the adoption of the cloud in the past 7-8 years. The concept of decoupling the specific functional elements of a software application has allowed software design to break free of the shackles of single executables to create flexible, interconnected, collaborative sets of integrated modules.

In the early days these modules were typically created by one vendor. But over time, as open systems and restful APIs have become more prevalent, standardized components have been created for everything from single sign-on authentication, to databases, to visualization.

This approach is often called cloud-native—not because it requires cloud-hosting to be used, but because most cloud services are built using this method.

Cloud native is a lot more than just signing up with a cloud provider and using it to run your existing applications. It affects the design, implementation, deployment, and operation of your application.

The Cloud Native Computing Foundation defines cloud-native as:

"... computing [that] uses an open source software stack to be:

01

Containerized. Each part (applications, processes, etc.) is packaged in its own container. This facilitates reproducibility, transparency, and resource isolation.

02

Dynamically orchestrated. Containers are actively scheduled and managed to optimize resource utilization.

03

Microservices-oriented. Applications are segmented into microservices. This significantly increases the overall agility and maintainability of applications."

Let's explore each of these architectural elements in detail.

Containerized

Containers package software with everything needed to execute it into a single package—for example, a Java VM, an application server, and the application itself. This container is then executed in a virtualized environment to isolate the containerized application from its environment.

The main benefits of this approach are that the application becomes independent of the environment and that the container is highly portable. The same container can be executed on a development, test or production system, and if the application supports horizontal scaling, multiple instances of a container can be started or stopped to add or remove instances of the application based on user demand.

Quite simply the container has everything your application needs—you just need to start it.

Orchestration

Deploying an application, with all of its associated library and reference dependencies, in a container is step one—solving the deployment challenges faced by traditional software builds, but to benefit fully from a cloud native platform there are additional requirements.

Running multiple application nodes will require you to:

- Monitor the system;
- Trigger the startup or shutdown of containers;
- Ensure all required configuration parameters are in place;
- Balance the load between active instances;
- Ensure security and collaboration between containers.

Orchestration systems provide the tools to automatically react to any unexpected changes in system load, and to streamline the management of cloud-native systems.

Microservices

Cloud native applications are typically built as a system of microservices.

This architectural style implements a system of multiple, small applications (or microservices), each of which work together to provide the overall functionality of your system. Each microservice delivers one specific piece of functionality, has clearly defined inputs and outputs, is developed and operated by a relatively small team, and importantly, has a published and managed API.

The use of microservices has a number of benefits;

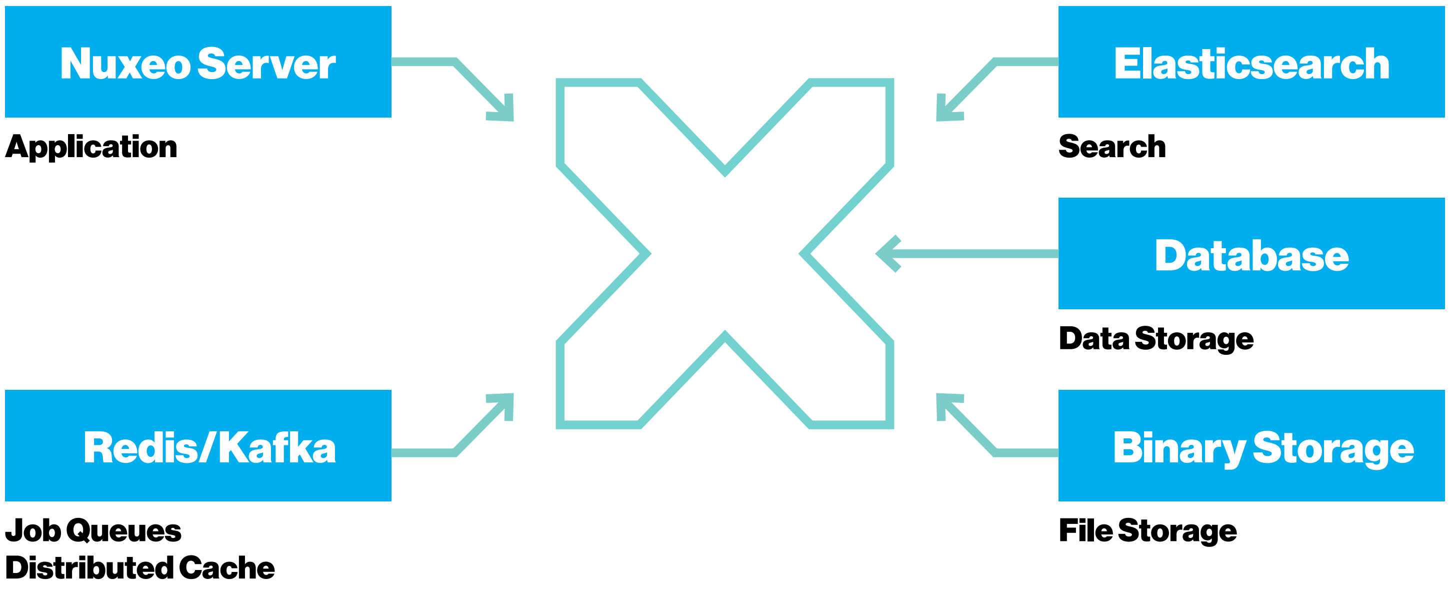
- The delivery and deployment of large, complex applications is broken down into smaller, more manageable pieces each of which;
    - are smaller and faster to test.
    - can be deployed independently.
    - enable you to organize the development effort around multiple teams, each of whom owns, and is responsible for, one or more single service. Each team can develop, deploy, and scale their services independently of the other teams.
- Each microservice is relatively small, making it;
    - easier for a developer to understand, extend, and maintain.
    - more productive for developers using a simpler and faster IDE.
    - faster to start, making developers more productive and speeding up deployments.
- Elimination of long-term commitment to a specific technology stack;
    - When developing a new service, the developer can pick the best technology stack for the microserver.
    - When making major changes to an existing service, it can easily be rewritten using any new technology stack.

The Nuxeo Difference

Nuxeo has been designed as a cloud-native solution from day one and has been built to:

- Make the best use of the massively-scalable NoSQL to store and manage metadata;
- Utilize tools such as elasticsearch to enhance the usability and flexibility of finding content;
- Scale independently and horizontally to provide the most cost-effective enterprise-grade performance.

Nuxeo combines these elements seamlessly to deliver the system architecture you wish you had the millions of man hours needed to build yourself.



2. NoSQL Database

A RDBMS (relational database) organizes data into fields, tables, and their associated relationships. However, in the real world, not all data fits naturally into a tabular structure (think of unstructured content, trees, and graph structures). Relational databases work around this limitation by introducing artificial structures such as data in normalized form and parent-child records. These techniques shoe horn non-tabular data into a tabular, record-oriented format.

As a result, the structure of the records in a table (also known as the schema) must be the same. For data that includes optional values, this creates tables with many empty columns and subsequently a lot of wasted space. This is worked around by breaking up records into sub-parts (or sub-tables), which hold the sub-fields and then do cross-table joins when querying to create complete records. This creates additional work every time data is requested—even for relatively simple queries.

A second result is that schema changes are expensive. To amend even one record with a new field, that new field needs to be added to every single record in the table.

By contrast, in a document-oriented database (or a NoSQL database) each document is stored independent of the other, so there is no wastage of space for optional fields and adding a new field to one document doesn't affect any of the other documents. In addition, there are several other benefits of using NoSQL databases over their relational cousins in a modern content services platform.

"As Is" Content

NoSQL databases support storing data "as is." Key value (RDBMS) stores give you the ability to store simple data structures, but document-based NoSQL databases provide you with the ability to handle a wider range of structures including flat or nested entities.

In the microservices world, data needs to be shared and communicated between microservices securely and consistently. This communication happens via message transfer. Typically, the data takes one of these formats:

- A binary object to be passed through a set of layers
- An XML document
- A JSON document

Being able to handle these formats natively in a NoSQL database lessens the amount of code required to convert from the source data format to the format that needs storing. This reduces the complexity of the system and increases the speed of operation.

Un-Structured Text Support

The vast majority of data in enterprise systems is unstructured. NoSQL databases handle indexing of un-structured text either as a native feature or via an integrated set of services. Nuxeo uses the industry standard Elasticsearch to provide this capability.

Being able to manage unstructured text increases the amount of information that can natively be managed and can help organizations make faster and better decisions. For example, advanced use cases include support for multiple languages with faceted search, fuzzy and similarity-based search functionality, and enterprise-grade scalability and speed. Advanced features also include support for dictionaries and thesauri.

Flexibility

Because of the schema-agnostic nature of NoSQL databases, they're very capable of managing schema changes. This means that if your business now requires additional information to be stored against a particular entity, this can be accommodated without issue.

If a document structure changes, these indexes allow organizations to use the information immediately, rather than having to wait for several months before you can test and rewrite systems. Given the increasingly rapid rate at which businesses need to adapt, this flexibility in the underlying information architecture is critical.

Simultaneous and Multiple Data Structures

Many applications need simple object storage, whereas others require highly complex and interrelated structure storage. NoSQL databases provide support for a range of data structures.

- Simple binary values, lists, maps, and strings
- Related information values can be grouped in column families
- Highly complex parent-child hierarchical structures

Reduced Reliance on "Magic" Queries

Structured Query Language (SQL) is the predominant language used to query relational database management systems. Over the years, being able to structure queries so that they perform well has become more of an art than a science—with complex multi-table joins being the norm in order to get anything done. The optimal design of those queries has provided many SQL engineers with well-paid roles for many years.

Although NoSQL databases support SQL access, they do so for compatibility with existing applications such as business intelligence (BI) tools. NoSQL databases support their own access languages that can interpret the data being stored, rather than require a relational model within the underlying database.

Application developers don't need to know the inner workings of databases before using them. NoSQL databases empower developers to work on what is required in the applications instead of trying to force relational databases to do what is required.

This developer-centric mentality to the design of databases and their access via application programming interfaces (API) is one of the key reasons why NoSQL databases have become very popular among application developers.

Scalability on Commodity Hardware

NoSQL databases automatically manage the partitioning of a database across several servers. This means that as data storage requirements grow, the infrastructure can be extended by adding (relatively) inexpensive servers. These servers are added to the existing database cluster, making them work as a single data service. This is known as horizontal scaling.

Contrast this to the relational database world where new, more powerful and thus more expensive hardware is required to scale up (aka vertical scaling).

Providing high availability and low-cost scalability via a NoSQL database by using inexpensive hardware and storage is one of NoSQL's major assets.

Metadata Matters

A key aspect of any information management platform is how metadata is handled. By using a NoSQL database to manage metadata, numerous benefits are achieved.

The flexible nature of NoSQL means that schema changes can be readily accommodated, versus the rigid nature of many traditional ECM schemas. No longer does a schema change required major updates to the entire system.

Enterprises today manage a wide variety of content with increasing focus on rich media in addition to text and pdf content. The ability for NoSQL to manage content "as is" delivers simplicity and speed. In addition, the ability for NoSQL to scale massively on commodity hardware provides enterprises with a cost-effective way to address the increasing volume of content and data required to be managed in today's enterprises.

And finally, NoSQL enables a break from traditional document focused systems—meaning that businesses with cases, matters, patients, and more complex content-driven processes and workflows can make use of the power and flexibility of managing their information securely and efficiently.

The Nuxeo Difference

Nuxeo fully embodies the power of NoSQL databases to deliver benefits such as “as-is” content storage, massive scalability, and simultaneous data structures—but the most important from a content services standpoint is that of metadata flexibility.

The ability to dynamically change the underlying schema of a Content Services Platform is both new and extremely significant. In traditional ECM systems, schemas are fixed and require intensive and costly scoping exercises upfront to define and create the schemas necessary for a specific project. If that schema was wrong or needed an amendment, the whole underlying structure of the relational database needed to be changed. The flexibility of NoSQL allows changes to be made to the content schema dynamically, allowing the Nuxeo Content Services Platform to be as agile as the organization it supports. This, combined with the ability to provide multiple views or facets on information and content, creates a key differentiator for platforms like Nuxeo that deploy on NoSQL.

Super Scalable

One Billion Assets Benchmark

1B
Documents stored in 9 hours

30k
Documents imported/second

5x
Faster bulk import than any SQL DB

30x
Faster than the competition

3. Low-code Application Development

User requirements within the business world are more challenging than ever before. Largely fueled by the rise in consumer technologies that provide simple user interfaces with rich cross-app integration (like mobile and tablet apps), business users now expect the same level of simplicity and ease of use within the enterprise. These capabilities cannot be delivered using the traditional methods of software application development—like the waterfall method.

The new model for building applications is based around a low code, rapid design, and prototyping environment—a user-friendly capability to quickly create solutions that integrate content, data, external services, and workflows that actually solve the real challenges faced by the modern knowledge worker.

This style of app development can only be achieved if built on top of a strong foundation—a foundation where business logic can be combined with consistent and reliable data. With that foundation in place, however, low code development can deliver significant benefits.

Stakeholder Engagement and Satisfaction

Low-code platforms allow the rapid creation and update of apps, so developers can share fully functional features with stakeholders in days or even hours. A dynamic and rapid cycle fosters more active engagement and provides a clear, ongoing sense of progress and responsiveness. In addition, something as simple as delivering an application that uses the domain-specific language or terms that an end-user is expecting can reduce the time taken to learn a new system, and reduce resistance to change.

Lower Risk—Higher ROI

With a low-code platform, all the security, cross-platform support, and data integration capabilities have already been built. As a result, developers and others can focus on solving business problems with rapid prototyping and visual delivery.

Ongoing Delivery

App development doesn't end with a successful launch. Updates, feature improvements, and bug fixes all need to be considered. A low-code platform provides the ability to make complex updates and deliver new features in minutes rather than days—reducing the impact on other projects in the development queue.

Reduction of the IT Skills Gap

The intuitive, visual editors in low-code platforms reduce the dependence on highly-skilled developers in order to realize value. Years of specific programming language knowledge isn't required, which allows more developers to spend more time contributing to business critical projects and innovation. In some cases, even non-technical stakeholders can learn to build their own prototypes. Waiting for IT to deliver solutions is a thing of the past.

Quick Change and Innovation

Custom coding an application is time-consuming and labor-intensive. Low-code platforms enable developers to innovate faster and easily create new functionality or customize features in days or weeks, rather than months or years. This shift allows developers to continually focus on delivering innovation and new value to the business—not bug fixing or maintaining defunct, outdated systems.

Mobile Delivery

The need for mobile apps is driving new and innovative approaches to app development. Low-code platforms provide developers and organizations with toolsets that enable them to develop and deliver applications that delight staff and customers—quickly and consistently.

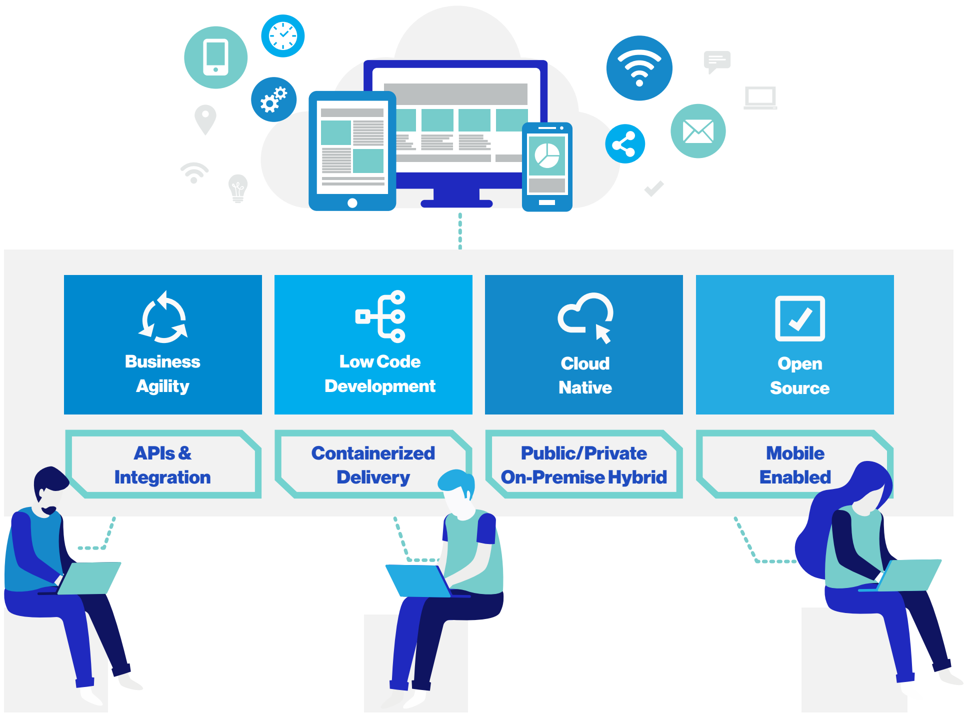
The Nuxeo Difference

One of the downfalls of ECM solutions was their fixed, and often arcane, interfaces. This was so pronounced that a separate industry, known as case management, was created in an attempt to solve the problem. From a user perspective, they really don't care—all they're interested in is the system providing a simple to use interface that combines data, content, and workflows to allow the user to get their job done.

However, the dilemma here is that individual roles within an organization (and even individual users) tend to work in different ways. This degree of personalization was never supported with ECM systems, but the low code development approach provided by modern CSPs like Nuxeo delivers this.

The Nuxeo low code platform enables the delivery of unique personalized interfaces built specifically for departments, teams, or even individuals. Solutions are formed using reusable components, assembled in a visual environment, and connected together to deliver the perfect combination of data, content, and process.

In a recent AIIM1 survey, 76% of respondents said there was a distinct difference in the information management requirements from one business use case to another—meaning that a single, monolithic interface cannot meet the needs of multiple users within the enterprise. However, a low-code development platform that can create several, dedicated solutions built on top of a single information foundation can address this, and many more needs.



4. REST-APIs

One of the core elements of a cloud-native architecture is the concept of microservices—connected, modular, specific function entities that facilitate dynamic and modern software design. As previously discussed, microservices enable the standardized interconnection of applications and other microservices, both on-premises and in cloud environments. But to do that, these microservices need a structured way to communicate with each other. This is achieved via application programmers' interfaces or APIs.

An API is a defined set of ways in which to communicate with and control an application, microservice, or any software entity. This typically includes ways to control variables and parameters, create new instances, execute functions, and so on.

There are numerous benefits to providing and utilizing REST-APIs including:

Separation of Client and Server

The REST protocol completely separates the user interface from the server and the data storage. This improves the portability of the interface to other platforms, increases the scalability of projects, and allows the different pieces of any component to evolve independently.

Language Independence

A REST API adapts to the type of syntax or platforms being used, which gives considerable freedom when changing or testing new environments within the development. With a REST API, you can easily have PHP, Java, Python, or Node.js servers.

Future Proofing

From a content services perspective, REST APIs serve two purposes. First, they enable future proofing of the platform—allowing integration to the best of breed tools of today, then the ability to swap them for the best of breed tools of tomorrow as they are developed. Second, they allow the platform to become a microservice host of its own—enabling parts of the platform to be exposed to other consuming entities such as websites, applications, mobile apps, or other content microservices.

“We built the Nuxeo platform from the ground up to be as flexible and extensible as possible and we want to provide that same high level of flexibility and extensibility through the API. This means that our API must be dynamic and composable.”

Thierry Delprat, CTO, Nuxeo

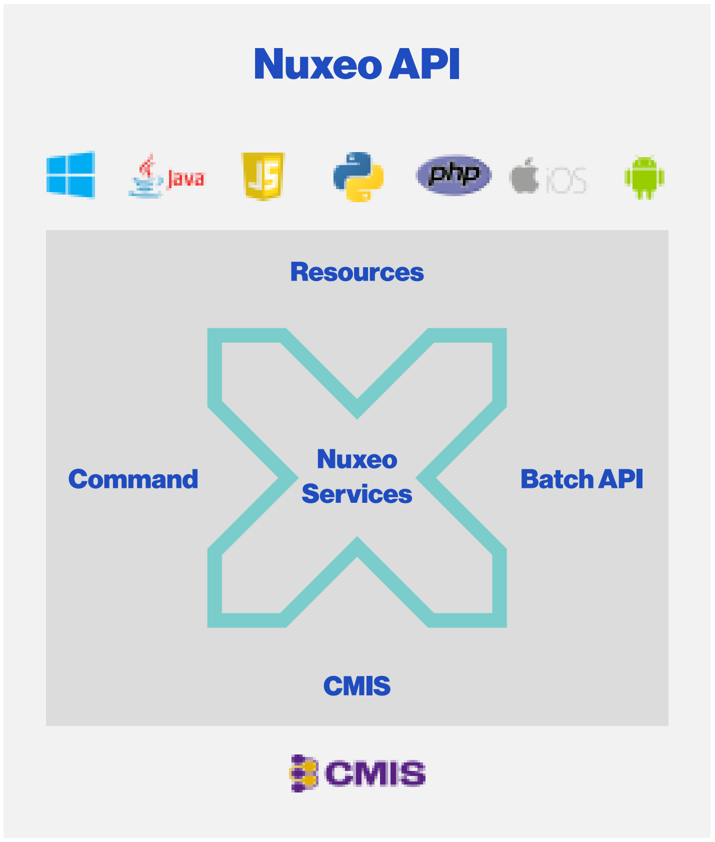
The Nuxeo Difference

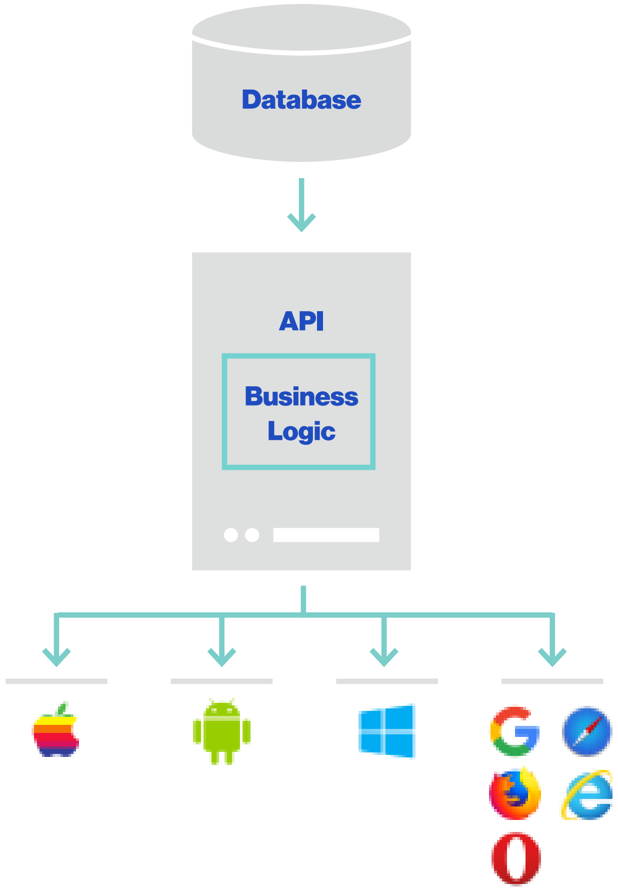
Every new piece of functionality in Nuxeo is created “API-first” which means functionality is built from the ground up to be connected, documented, and available throughout all aspects of the platform. This approach fully embodies the open nature of the Nuxeo platform.

Nuxeo has been an advocate of open APIs for many years and this extends to the use of REST-APIs for both consumption and delivery of services.

For consumption, Nuxeo can make use of any REST-API enabled application, service, or tool. This method is used to integrate to numerous pieces of the overall Nuxeo solution including AI, digital signature and records management capabilities.

For delivery, any content, data, or workflow from within the Nuxeo Platform can be made available via a REST-API. This includes services from external services that have been consumed by Nuxeo, providing developers with the ability to “roll their own” APIs within the platform and expose, promote, and potentially monetize them.





5. Open Source

Once upon a time, open source was thought of as free, indie, unsecure software. Those days are over. Open source is now a recognized business model for the delivery of software that adheres to the highest coding standards. Open source now means developer enabled software that opens it's source to the world for inspection.

Enterprises looking to make smart use of open source software will find plenty of reasons to do so, such as:

Community

Enterprise open source solutions often have thriving global communities around them. They are bound by a common drive to support and improve a solution that both the enterprise and the community benefit from (and believe in). United around improving these solutions, they introduce new concepts and capabilities faster, better, and more effectively than internal teams working on proprietary solutions.

Transparency

Open source code means you get full visibility into the code base, as well as all the discussions about how the community develops features and addresses bugs. In contrast, proprietary code is largely produced in private by a few people and may come with unforeseen limitations and other unwelcome surprises. With open source, you're protected against lock-in risks and can see exactly what you're getting.

Reliability

The reliability of open source code tends to be superior because there are more eyes on it. The output tends to be extremely robust, tried, and tested code. In fact, open source code now powers about 90% of the internet and is being rapidly adopted across major enterprises for this reason.

Better Security

Open source software code is often more secure than proprietary code because it is much more thoroughly reviewed and vetted by the community (and any issues that do arise tend to be patched more diligently). Despite often being a point of hesitation for enterprise's looking to adopt open source software, concerns about security are largely unfounded.

Everyone's Doing It

Many large enterprises are now implementing open source solutions, and often making corporate policies promoting the use of open source, bringing the strength of their resources to the communities that support open source solutions.

In addition to these advantages, open source software has the long-term viability to outlast proprietary developers that come and go. And, thanks to supportive communities that are energized to continually introduce innovations, open source software remains at the forefront of advancing technology as a whole and meeting enterprises' needs as they evolve going forward.

The Nuxeo Difference

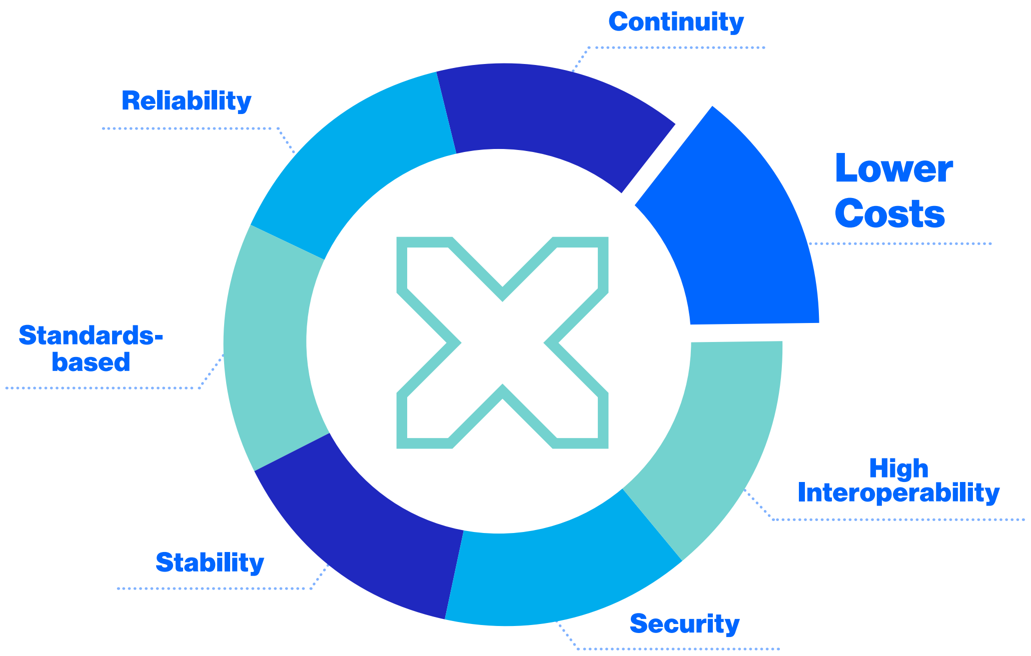
There are a number of open source solutions available within the ECM and Content Services Platform market. However, many of these use open source as a marketing tool more than as a fundamental underpinning. Nuxeo is different for these reasons:

01

One code base. Nuxeo delivers one version of the Nuxeo platform—there is no "community" version and a separate commercial version. One code base for all. This means that absolutely anyone can download the platform and start making use of it—for free. Nuxeo makes its money by providing a cloud-based configuration tool called Nuxeo Studio that enables developers to fast track configuration and deployment, as opposed to working directly with the platform via the API.

02

Open Kitchen. For Nuxeo, open source means being held to the highest standards by the developer community. To that extent, we operate an open kitchen approach, providing full access to all of our source code—to anybody. In the same way that chefs allow diners to look into an open kitchen to see exactly how their food is being prepared, Nuxeo's open kitchen software development facility delivers that very same experience.



Nuxeo Content Services Platform Architecture

Nuxeo brings the strongest elements of a modern content services architecture together into one platform. The platform (developed by the Nuxeo engineering team who has crafted over 5.7 million lines of code across over 200 man years of development) contains, what industry experts agree are, the key architectural components of a modern solution.

The Nuxeo Platform:

- Is cloud-native with a state-of-the-art modular component architecture;
- Has a database-agnostic architecture and can be deployed on NoSQL or traditional SQL databases;
- Leverages innovative and best-of-breed technologies such as ElasticSearch, containerization, and artificial intelligence (AI);
- Provides a low code development capability to drive rapid prototyping and delivery of customized, user-driven applications;
- Provides REST API capabilities, for both consuming and providing microservices;
- Is a true open source solution;
- Features multi-tiered security controls and capabilities.

These architectural underpinnings combine to deliver the most modern Content Services Platform on the market that delivers a full set of features and capabilities, including:

- System Security and Authentication;
- User management;
- Multiple, customizable user interfaces (including mobile);
- Native content viewers;
- Flexible and scalable metadata schema;
- Indexing and search;
- Versioning;
- Audit logs;
- Workflow engine;
- Connectivity to numerous repositories;
- Records and retention management;

and much more.

Conclusion

This whitepaper has highlighted some of the significant architectural elements available to software developers and architects today. It has also explored how Nuxeo has taken these separate elements and combined them to create a uniquely modern, scalable, and flexible Content Services Platform.

The Nuxeo platform;

- combines the latest technology components to deliver a full set of content services such as workflow management, integration, document management, records management, case management, mobile and tablet access, knowledge management, digital asset management, and more.
- provides comprehensive extensibility via APIs and microservices, for both consumption and delivery of content services.
- integrates with existing systems to enable access to information residing in native solutions via content federation and/or migration.
- provides security and audit tracking for all content and data.
- includes cloud-native capabilities to deliver scalability, ease of deployment, and flexibility—all on commodity hardware.
- enables organizational agility by being as dynamic and flexible as your business.

Nuxeo has been developed by technologists and system architects—for technologists and system architects. **Designed to save you time, money, and headaches—this really is the platform you would have built yourself.**

Learn More

To learn more about the Nuxeo platform and how we can help you modernize your legacy systems, please contact us or visit www.nuxeo.com.