Technical Requirements for Launching a Diamond OA Publication

A Systems-Level Architecture for Metadata-Centric, Automated Scholarly Publishing

zeba.academy

Zeba Academy

First Published by Zeba Academy and Zeba Books.

Publication Year: 2026

Document Series: Zeba Academy Blueprints -- Sovereign Systems Technical Directives

Purpose: This series of blueprint directives is authored to combat the "enshittification" and unnecessary bloat of modern software. Our goal is to reclaim sovereign control over our systems by bridging the gap between deep academic theory and high-stakes industrial implementation. We believe that software should be fast, permanent, and most importantly, understandable to the person who owns and uses it.

Principal Architect: Sufyan bin Uzayr, Google Cloud-Certified Professional DevOps Engineer.

Core Stack: Linux, Rust, Zig, C++, Flutter, and PHP.

Email: hello@zeba.academy

Technical Requirements for Launching a Diamond OA Publication

A Systems-Level Architecture for Metadata-Centric, Automated Scholarly Publishing

System Definition and Operational Constraints

A Diamond OA system must be entirely automated, cost-neutral, and require little to no human intervention. It requires deterministic procedures and well-organized data processing to ensure consistency and scalability. The technology must also integrate nicely with other academic offerings while being cost-effective.

Designing a Deterministic, Zero-Revenue Publishing Engine

An OA publishing system for Diamonds must be built to be free of charge and purely computational infrastructure.[1] That is, all the advantages it has will derive solely from efficient software computations, not from money circulation or human intervention. The OA publishing system operates as an automated computational pipeline and requires no manual involvement, unlike traditional publishing.

It has to be created as a directed acyclic graph of stateful transformations, where each manuscript goes through several processing stages that must be

[1] Diamond Open Access (UNESCO Open Science Framework) - https://www.unesco.org/en/diamond-open-access - Accessed: 09 April 2026

idempotent – that is, repeated execution of any process yields the same results without contradictions.

The following safety measures have to be adhered to:

- **Deterministic execution model**: All transformations must produce consistent outputs for identical inputs, hence preventing non-deterministic behavior.
- **Schema-bound validation**: All data elements, particularly metadata, must conform to predefined schemas, such as XML or JSON.
- **Event-driven orchestration**: People should not initiate workflow transitions; instead, system-generated events should be used.

This transforms the publication process into a controlled execution environment, in which manuscripts function as computer programs rather than editorial artifacts. The end result is a system that can be reproduced, tested, and expanded without increasing operational costs by an equal amount.

Infrastructure Architecture and Deployment Topology

The infrastructure must support scalable, fault-tolerant operations with a clear separation between the storage and application layers. It should allow for automatic deployment and ensure the system's availability at all times.

Building a Cloud-Native, Container-Orchestrated Runtime Environment

The infrastructure must be designed as a cloud-native, containerized, distributed system that can be isolated, portable, and scalable. The architecture should keep stateless computation and stateful persistence layers separate. This allows for self-scaling while keeping flaws distinct.

The main components of the system are:

- **Compute Layer**: Kubernetes manages and hosts stateless services at the compute layer.[2] These services handle intake, transformation, validation, and rendering.
- **Persistence Layer:** A relational database management system like PostgreSQL that stores metadata, stateful workflow data, and audit logs. The failover and replication features must be enabled.
- **Object Storage Layer:** Object storage compliant with the S3 interface for storing immutable objects such as XML files, manuscripts, and output.
- **Ingress Layer:** Reverse proxy solutions such as NGINX and Envoy for terminating TLS connections and routing traffic.

Infrastructure provisioning needs to be declarative, which can be achieved with tools like Terraform. The infrastructure can be version-controlled, cloned, and instantiated on any platform.

[2] Kubernetes Documentation for Container Orchestration - https://kubernetes.io/docs/home/ - Accessed: 09 April 2026

Operational resilience is gained through:

- Health probes, which check for activity and readiness, are used to monitor services.
- Database and storage system failover strategies are automatically generated.
- Backup strategies that employ snapshots and check them frequently.

When it comes to safety, consider the following:

- TLS provides encryption from beginning to end.
- Token-based authentication (JWT and OAuth 2)
- RBAC stands for "role-based access control."

This design ensures that the system can handle partial failures without causing additional problems, hence maintaining availability and consistency.

Editorial Workflow Engine and State-Oriented Processing

To ensure that everything is consistent and traceable, the editorial system must process manuscripts using defined, state-driven processes. It should enable regulated updates, role-based actions, and automated event processing throughout the publication's lifecycle.

Modeling Manuscript Lifecycle as a Finite State Machine (FSM)

The workflow needs to be designed as a finite-state machine, with the manuscript as the state machine and its life cycle defined beforehand.[3] This makes everything easier and ensures that state transitions occur only when needed.

Some frequent states include:

- Submission
- Initial validation
- Review by peers.
- Revision
- Acceptance
- Making
- Publishing

There are regulations for each transition between states:

- **Rules for Access Control:** Ensuring that only approved roles can initiate transitions
- **Validation limits:** Checking whether the metadata is complete and the file is secure.
- **Triggers for Events:** Created by activities performed by users or automated systems

[3] Workflow Management System (WfMS): Concepts and Architecture - https://en.wikipedia.org/wiki/Workflow_management_system - Accessed: 09 April 2026

Some noteworthy architectural features include:

- **Immutable versioning**: Every change is stored as a new object, thus keeping previous versions secure.
- **Events trigger processes:** tools like Kafka and RabbitMQ start background tasks in response to events.
- **Full audit logging:** Every change is recorded with a timestamp and the name of the user who made it.

There are two primary storage systems that organize data:

- Relational databases for metadata and process states.
- Object storage for manuscript files in various versions.

These processes can be made even more efficient with:

- Asynchronous tasks that will be handled at some point.
- Tasks are scheduled for alerts, deadlines, and system maintenance.

This converts the editing workflow into a programmable execution engine, guaranteeing consistency, traceability, and scalability.

XML-First Publishing and Canonical Data Modeling

To ensure that all outputs and integrations are consistent and work together, the system must use organized, machine-readable data.

Establishing JATS XML as the Authoritative Data Layer

When publishing, the entire system should employ an XML-first strategy.

JATS XML should be the standard format for displaying scholarly content.[4] This will avoid redundancy and ensure all formats are consistent.

The ingestion pipeline carries out transformations:

1. Documents in DOCX or LaTeX are parsed
2. Data normalization occurs.
3. Transformation into JATS XML using procedures like Pandoc.
4. Validation of XML with schema definitions

The XML schema contains:

- The structure of the document (sections, figures, and tables)
- Information about contributors (authors, affiliations, ORCID IDs)
- Bibliographic citations with persistent IDs.

The XML document is the only place you can find the truth. All additional forms are created on the fly via transformation pipelines.

There are rendering layers, including:

- XSLT adjustments that produce HTML
- PDF output typesetting engines based on LaTeX or CSS.

This paradigm ensures that data is normalized, can be utilized on various systems, and will last for a long time because XML can be read by machines and is not limited to a single platform.

[4] JATS: Journal Article Tag Suite (NISO Standard Documentation) - https://jats.nlm.nih.gov/publishing/ - Accessed: 09 April 2026

Persistent Identifier Infrastructure and API Integration

To ensure that scholarly systems can connect and collaborate, the system must employ automated API actions to add persistent identities.[5] This layer should enable external services to both provide and receive metadata consistently.

Implementing DOI and Identity Systems as Distributed Services

Persistent identities are critical for interoperability, and they should be configured as API-driven distributed services. Crossref and other outside entities assign a DOI to each article and register it.

The procedure of registering a DOI comprises the following:

- Create deterministic IDs.
- Putting metadata in organized forms like XML or JSON.
- Sending and verifying responses to the API.

To ensure that the system is reliable, it must perform the following:

- Retry options for failed API calls.
- Complete logging of request-response cycles.
- Keep an eye on resolution endpoints.

[5] Crossref REST API and Metadata Deposit Guide - https://www.crossref.org/documentation/retrieve-metadata/rest-api/ - Accessed: 09 April 2026

Additional integrations are:

- ORCID for determining who the author is.
- IDs to identify institutional affiliations

These IDs constitute a distributed metadata graph, which allows:

- Inter-platform linking
- Citation tracking
- Semantic enrichment

The integration layer should handle network interruptions and support asynchronous processing.

Metadata Exposure and Discovery Protocol Engineering

To function effectively with indexing and discovery services, the system must provide structured and standards-compliant metadata. It should ensure that the formatting is consistent, easy to find, and compatible with harvesting techniques.

Enabling Machine-Level Harvesting and Indexing

Standardized metadata exposure standards make content discoverable by allowing external systems to readily collect and index it.

The system should have OAI-PMH endpoints that support:

- Obtaining XML records with unique IDs.
- Incremental harvesting using time stamps

- Strict validation of the schema

To add metadata to the display layer, use the following syntax:

- Schema.org tags improve search engine performance.
- Sitemaps created on the fly.
- Canonical URLs for Finding Resources

Here are some strategies to boost performance:

- API endpoints have low latency.
- Caching ways to reduce the burden on the database
- Quickly conduct queries on large datasets

This ensures that the publication can be found by machines on a broad scale, eliminating the need for manual indexing or interaction.[6]

Production Pipeline and Deterministic Rendering System

The system must have an automated and repeatable production workflow that converts structured content into standard output formats. It should ensure that all rendering processes are consistent, precise, and efficient.

[6] Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) Specification - https://www.openarchives.org/OAI/openarchivesprotocol.html - Accessed: 09 April 2026

Automating Multi-Format Transformation from Canonical XML

The production pipeline functions as a deterministic rendering system. When XML is accepted, it is automatically transformed into publication-ready formats.

The series of changes includes:

- Editing and Normalizing Content
- Validation and augmentation of XML
- rendering in PDF and HTML formats.

The technologies employed were:

- Pandoc for changing formats.
- LaTeX for creating high-quality text.
- XSLT for modifying things in an organized manner

CI/CD pipelines automate procedures, ensuring that:

- Run transformation jobs automatically.
- Outputs governed by version
- Updates for deployments without downtime.

Considerations for performance include:

- loading assets rapidly
- Strategies for caching output.
- File compression for sharing.

The system guarantees deterministic rendering, meaning that identical XML inputs will always produce the same results.[7] This makes it easier to replicate the results.

Preservation, Observability, and Lifecycle Engineering

Long-term sustainability demands resilient preservation mechanisms and continuous observability infrastructure.

Here are some ways to keep things safe:

- Replication among storage nodes that are distributed
- Using long-lasting formats like XML and PDF/A.
- Validating checksums to ensure integrity

The observability stack must have:

- Collecting metrics (similar to Prometheus).
- Logging in one location (such as the ELK stack)
- Systems that send alarms when something goes wrong.

Analytics tools that adhere to privacy standards, such as Matomo, can show you how users interact with your site without jeopardizing personal data.[8]

All maintenance tasks must be completed automatically:

- Backup and restoration testing on a scheduled basis
- Update dependencies and security patches.

---

[7] Pandoc User Guide: Universal Document Conversion and Workflow Automation - https://pandoc.org/MANUAL.html - Accessed: 09 April 2026

[8] Providing Observability and Transparency in Distributed Digital Preservation Systems - https://www.digipres.org/publications/ipres/ipres-2024/papers/providing-observability-and-transparency-in-a-distributed-digita - Accessed: 09 April 2026

- Indexing and optimizing databases.

It is critical that the infrastructure can be duplicated. Declarative designs must be able to rebuild the entire system, enabling rapid recovery from significant failures.

Final Technical Perspective

The Diamond OA Publishing Model is an engineering challenge in distributed computing that requires precise timing among the choreography of processes, metadata modeling, and the automation of infrastructure.

In order to enforce:

- XML-first canonical data models
- Containerized and cloud-native infrastructure.
- API-driven identification integration.
- Deterministic production pipelines

Production pipelines become deterministic, enabling a publishing platform to operate independently and for an extended period.

This strategy ensures that everything is automated, compatible with other systems worldwide, and adheres to academic communication standards. Finally, computerized procedures must replace editorial processes, making academic publishing more efficient.

Zeba Academy is a specialized technical research and training initiative dedicated to the principles of Sovereign Systems Engineering. Founded by Sufyan bin Uzayr - an author and university instructor as well as Google Cloud-Certified DevOps Engineer - Zeba Academy serves as a bridge between deep academic theory and high-stakes industrial implementation.

We reject the "enshittification" of modern software. Our core mission is the promotion of Anti-Bloat Architecture through the mastery of:

- **Systems Languages:** Using Rust, Zig, and C++ to build high-performance foundations that prioritize memory safety and deterministic execution.
- **SRE & DevOps:** Professional-grade automation via Google Cloud, Terraform, and Immutable Infrastructure to eliminate manual "toil" and operational fragility.
- **High-Performance Interfaces:** Utilizing Flutter for cross-platform development to deliver near-native mobile experiences without the lag of standard web-based wrappers.
- **Lean Web Publishing:** Reclaiming WordPress and PHP by stripping away the "slop", using Redis object caching and Unix sockets to transform standard platforms into high-speed, GEO-stable engines for modern publishing.
- **Legacy Modernization:** Applying memory-safe paradigms and modern build systems to century-old computational problems and aging C codebases.

Zeba Academy doesn't just teach code; we architect reliability. By merging the analytical rigor of Historical Research with the precision of Google-Certified Cloud Engineering, we provide our "Operatives" with the directives necessary to build systems that are safe, fast, and permanent.

Website:- https://zeba.academy

زیبا

Zeba Academy

zeba.academy