The American Journal of Applied Sciences
ISSN 2689-0992 Volume 08 - 2026

A Scalable Architecture for Intelligent Document Processing in Multi-Cloud Environments

[1] Suprakash Dutta
[1] Senior Solutions Architect, AMAZON WEB SERVICES Dallas, TX, USA

Received: 28th Nov 2025 | Received Revised Version: 18th Dec 2025 | Accepted: 29th Dec 2025 | Published: 13th Jan 2026

Volume 08 Issue 01 2026 | Crossref DOI: 10.37547/tajas/Volume08Issue01-03

ABSTRACT

This paper reviews an easily expandable plan for smart document handling across multiple cloud systems, aiming to make work easier to manage, more resilient to issues, and improve the total cost of ownership. The importance of this task stems from two factors: first, Intelligent Document Processing (IDP) tools are experiencing growth; second, multi-cloud use is expanding more widely. This increases the primary fight between wanting top-notch help for every step and the dangers of being stuck with one provider, having messy operations, and uneven safety rules. The study aims to create and support a complete design that can hide both setup and software links while offering complete control and standard protection in a mixed environment. The innovation is in the coherent four-layer model, which merges a general control plane atop Kubernetes and Crossplane with portable application runtime Dapr, exposing standard APIs for statelessness, messaging, and service invocation, decomposed IDP microservices, and an overlay layer for management and security. The key findings validate that only the combination of Crossplane at the level of the control plane with GitOps and OPA policies together with Dapr at the level of the application-API can provide real portability, elastic scaling, governed security, while maintaining freedom of choice between cloud services. It proves that workflows crossing provider boundaries can be orchestrated, thus reducing vendor lock-in. The article will be helpful to cloud-platform architects, IT executives, data and MLOps engineers, IDP product teams, and researchers in distributed systems and enterprise AI.

Keywords: intelligent document processing, multi-cloud architecture, application portability, Kubernetes, Crossplane, Dapr, GitOps, data management, security, microservices, LLM.

Cite This Article: Dutta, S. (2026). A Scalable Architecture for Intelligent Document Processing in Multi-Cloud Environments. The American Journal of Applied Sciences, 8(01), 14–25. https://doi.org/10.37547/tajas/Volume08Issue01-03

1. Introduction

Two powerful and interrelated technological movements shape the modern corporate IT landscape. The first is the industrialization of intelligent document processing (IDP). IDP has moved out of the shadows as a niche back-office enabler to take a leading role in digital transformation by automating document-centric processes and mining value from unstructured data. This evolution is being fed by lightning advances in artificial intelligence (AI), machine learning (ML), natural language processing (NLP)—and now, GenAI—driving new frontiers of precision and capability. The second trend is the strategic imperative of multi-cloud computing. The overwhelming majority of companies adopt multi-cloud strategies (Innovation at Work, n.d.). Key drivers include the desire to avoid single-vendor dependence, cost optimization by selecting the best price–performance fit for specific workloads, improved fault tolerance, and access to a broader range of

innovative services from providers such as Amazon Web Services (AWS) (Innovation at Work, n.d.). The convergence of these trends creates a fundamental conflict. Running something as complex, data- and state-heavy as a modern IDP system across many different clouds brings major problems: dealing with other APIs, mismatched security setups, tricky data sync, network delays, and a big skills gap. Just moving things over without changes doesn't work because apps get deeply linked to provider-specific managed tools (like AWS Textract, SQS, Lambda), bringing in a new type of dependency lock-in even when the main app is containerized (Pradhan et al., 2024).

2. Materials and Methodology

This study is made up of seven elements selected from academic, commercial, and technological knowledge. Fundamental regulations for forming clouds by Pahl et al. combine with a present review of multi-cloud arrangements from Innovation at Work. New measures in intelligent document management are imparted by Lin et al., in which the TWIX arrangement significantly enhances productivity by means of LLM–visual amalgamation. The market direction and customer needs, as outlined in Gartner reports are also taken into account, considering the trend toward ModelOps and heightened competitive commoditization. Practical application for microservice decomposition together with managed services is demonstrated in the Amazon Textract engineering case, as told by Pradhan et al. The methodology consists of three interrelated stages. First, a systematic review of literature covering 2019–2025 to develop patterns about scalability and portability that might have relevance to IDP. It then sets side by side the proposed approaches towards dealing with infrastructure and application dependencies in Kubernetes + Crossplane and Dapr, along with other control-plane ways. Next, by using a design-science approach, it developed an additional version of the four-layer framework whose components were associated with multi-cloud challenges via problem-solution mapping.

3. Results and Discussion

The technological trajectory of IDP has evolved from simple data capture to content-understanding systems. The earlier phase used optical character recognition (OCR) to convert text. These systems were brittle, template-bound, and limited in scope of application. The big bang for the world of IT was the shift toward ML-based IDP—now the semi-structured documents like invoices and receipts could be processed by training models on layouts and patterns recognized within documents to dramatically increase accuracy and flexibility (Pradhan et al., 2024). The most recent disruption, not previously discussed, is the integration of large language models into human conversation in digital form, which efficiently takes IDP beyond simple extraction into actual understanding (Moravcik et al., 2024). Previously unavailable advanced features include long-document summarization, context-aware answers to questions within a document, and extracted, normalized data placed inside structures such as JSON files; now unattainable for handling fully unstructured documents like legal contracts or scientific papers, where what matters cannot be found through predictable formatting. Progress is measured more by higher bars and deeper forensics. Academic studies, such as TWIX, demonstrate that hybrid pattern guesswork with semantic cleanup via LLM outperforms not only commercial cloud tools like AWS Textract but also pure visual LLMs by over 25 percent in F1-score at significantly faster speeds and lower costs (Lin et al., 2025). This underscores that a one-size-fits-all model is not optimal. This evolution leads to an important conclusion: IDP is not a single task but rather a portfolio of specialized capabilities. Different tools and models perform various tasks: TWIX does well with template-driven documents, LLM with unstructured text, and AWS Textract for signature detection. The best IDP system is not a product but rather an Orchestration workflow that can route a document to the appropriate processing engine (i.e., invoice to Amazon Textract, legal contract to an LLM on Amazon Bedrock, standardized form to Amazon Textract). This means that a multi-cloud architecture is not merely a nice-to-have for resilience but a necessity for peak performance and cost efficiency in IDP. This conclusion strongly motivates the architecture proposed in the present work. Table 1 provides a comparison of IDP.

Table 1. Comparison of Intelligent Document Processing (IDP) paradigms (compiled by author)

Characteristic	Traditional OCR	ML-based IDP	GenAI / LLM-based IDP
Core Technology	Template matching, layout analysis	Convolutional/Recurrent Neural Networks (CNN/RNN)	Transformers, Large Language Models (LLM)
Document Types	Structured only	Structured, semi-structured	Structured, semi-structured, unstructured
Main Capability	Text conversion	Field extraction, classification	Contextual understanding, summarization, and generation
Accuracy Ceiling	Moderate, depends on template quality	High (up to 99% for known formats)	Near human-level, depends on model and prompting
Customization & Training	Manual templates	Large labeled datasets	Zero-shot/Few-shot learning, fine-tuning
Key Limitation	Fragility, inflexibility	Requires large datasets, struggles with new layouts	Risk of "hallucinations", high computational cost

Multi-cloud computing has become the dominant paradigm in corporate IT. The main drivers are well known: avoiding vendor dependence to protect pricing and strategic flexibility, reducing costs by matching workloads with the most economical provider, increasing resilience and disaster recovery by distributing risk, and gaining access to best-in-class services while stimulating innovation. Implementing a multi-cloud strategy, however, is fraught with significant difficulties. Working with multiple consoles, APIs, and resource models raises operational overhead and demands specialised skills. Enforcing consistent security policies, managing identities, and meeting regulatory requirements in disjoint environments is a major obstacle. Each cloud exposes its security primitives, creating potential gaps. Seamless connectivity and data exchange between services running in different clouds are hard to achieve. Lack of standardization hinders portability. Cross-cloud traffic suffers higher latency and bandwidth limits than intra-cloud traffic, and transferring large data volumes can be slow and expensive (Innovation at Work, n.d.). Teams need to gain skills on many platforms. It is both complex and costly to acquire and maintain this knowledge. The suggested fix for this rests on fundamental software engineering rules for cloud systems. Big apps break down into small free services, each taking care of one business task. Main rules involve loose tying, high unity, and spread out data control. This setup is key to making systems that can grow and stay strong. Designing for failure is the core philosophy of fault-tolerant architecture. It embraces redundancy by eliminating single points of failure through deployment across multiple availability zones and regions, as well as graceful degradation and fault isolation by applying patterns such as Circuit Breaker so that the failure of a non-critical component does not bring down the entire system. Cloud-native systems must be self-adaptive, responding dynamically to changes in load or environment. This is often achieved through control loops (for example, MAPE-K: Monitor, Analyze, Plan, execute over a Knowledge base) and a pervasive "design for automation" mindset that spans infrastructure, deployment, and recovery (Pahl et al., 2018). An example of such an infrastructure is shown in Figure 1.

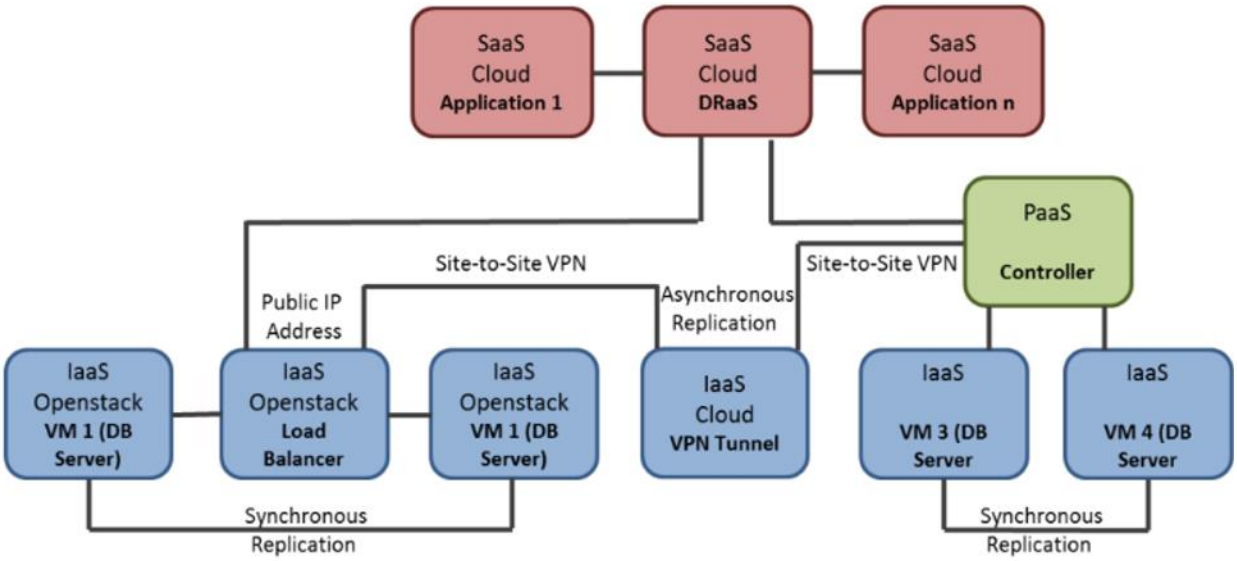

Fig. 1. A Sample Cloud Architecture Use Case (Pahl et al., 2018)

These principles reveal the so-called portability paradox. At first glance, containerised microservices are considered portable. An organization may believe its IDP application is portable because its components (for example, a classification service and an extraction service) are containerised. Yet a typical cloud architecture for IDP is deeply integrated with provider-managed services: S3 for storage, Textract for extraction, Comprehend for classification, Step Functions for orchestration (Pradhan et al., 2024). Attempting to move this portable application to another cloud would require a complete re-architecture to use that provider's equivalent services. The code that interacts with these services must be rewritten. This shows that true portability depends not only on application code but also on its dependencies. The architecture is "locked" even if the container is not. Therefore, a genuinely portable multi-cloud architecture must abstract not only infrastructure but also these application-level dependencies – the key challenge addressed at Layer 3 of the proposed architecture.

The proposed architecture is a four-layer conceptual model designed to systematically address the challenges of deploying IDP in multi-cloud environments. Each layer provides a specific type of abstraction that together yields a resilient, scalable, and truly portable system. Layer 1: Unified Control Plane – manages all infrastructure resources across clouds through a single declarative API. Layer 2: Decoupled IDP Microservices – the containerised business logic of the IDP process. Layer 3: Abstracted Application Runtime – offers a portable API for common application-level dependencies such as state management, messaging, and service invocation. Layer 4: Cross-Cutting Governance and Security – a logical layer that enforces policies through the control plane and other mechanisms.

Layer 1 directly tackles the core multi-cloud issues of management complexity and heterogeneity. The fundamental principle is to use Kubernetes not only for container orchestration but also as a universal control plane for all cloud resources. To implement this, Crossplane – an open-source project under the CNCF – is proposed. Crossplane extends the Kubernetes API to manage any resource (for example, a database, object storage, or message queue) at any cloud provider using standard Kubernetes YAML manifests. The platform team defines abstract CompositeResourceDefinitions (XRDs) that represent logical application requirements, such as IDP-ObjectStore, IDP-Queue, or IDP-AIService. Developers request these abstract resources without needing to know the underlying cloud provider. Compositions then translate these requests into provider-specific resources (for example, an IDP-ObjectStore request results in an AWS S3 bucket) based on defined policies for cost, performance, or data location. This grants developers self-service capability while centralising control.

All multi-cloud infrastructure state is defined declaratively in a Git repository and governed by a

GitOps controller such as ArgoCD or FluxCD. This arrangement provides a single source of truth, automatic reconciliation, and an auditable history of every change, thereby eliminating configuration drift.

The approach turns the control plane into a locus of governance. Because every allocation of infrastructure passes through the single Crossplane control plane, a powerful enforcement point for policy emerges. Security and compliance rules (for example, 'all storage must be encrypted', 'all workloads must have a network policy', 'EU data must use resources located in the EU') can be expressed as policies with tools such as OPA or Gatekeeper and are applied automatically by the control plane before any resource is created. The result is a shift from reactive manual auditing to proactive automated prevention, embedding the organization's security policy directly in the infrastructure-management workflow.

Layer 2 contains the core business logic, decomposed into independent containerised microservices that correspond to the logical stages of the IDP pipeline. Ingestion-Service accepts documents from email, API, and SFTP. Classification-Service determines the document type, for instance, invoice, contract, or receipt. Extraction-Service makes calls to the underlying AI or ML models to get the data. Enrichment-Service does post-processing, which includes validation of the data, normalization of the data, and, when necessary, makes calls to external APIs for extra information. Validation-Service manages the human-in-the-loop workflow by sending low-confidence results to a specialist. Egress-Service sends the structured data to systems like ERP or CRM. Each service is built and deployed independently, allowing for independent scaling. The Extraction-Service may require substantial GPU resources and can scale separately from the lightweight Ingestion-Service, entirely consistent with microservice principles.

Layer 3 resolves the previously identified portability paradox. It supplies standard portable APIs for everyday application-level needs, separating the Layer 2 microservices from the cloud-specific managed services provided at Layer 1. A framework such as Dapr (Distributed Application Runtime), another CNCF project, implements this layer. Dapr exposes building-block APIs that microservices invoke through simple HTTP or gRPC calls. State Management enables a service to persist data using the Dapr state API. At the same time, a YAML configuration can specify Amazon ElastiCache (Redis-compatible) in one region or Amazon DynamoDB in another, without requiring changes to the application code. Pub/Sub allows for asynchronous data exchange using the Dapr pub/sub API with backends such as Amazon SQS or Amazon SNS. The service does not know their differences. Secure, reliable discovery and interaction among services are made possible by Service Invocation. By developing against Dapr instead of provider SDKs, Layer 2 microservices attain real portability; the decision of the provider for storage or messaging is left until runtime configuration, which is controlled by the Layer 1 control plane, thus achieving workload portability and reducing vendor dependence.

Layer 4 is a logical layer that integrates with the other three to provide end-to-end governance. Centralised data governance maintains a single catalogue with AWS Glue Data Catalog, discovering and profiling data assets across clouds, tracking lineage, and enforcing quality and classification rules, including PII detection. Unified security posture management enhances the control-plane policy by integrating Cloud Security Posture Management tools that continuously monitor and provide a consistent view across clouds, identifying misconfigurations and threats. Federated identity makes use of such providers as AWS IAM Identity Cente for the central management of user and service accounts across all platforms with consistent authentication and authorization. An architecture sample data flow is as follows: an invoice comes in through Ingestion-Service, Classification-Service routes it to the right Extraction-Service, which might call a foundation model via Amazon Bedrock. Results are saved through the Dapr state API with a database on AWS, and a notification is sent via the pub/sub Dapr API using Amazon SQS/SNS. Hence, the workflow crosses the cloud boundaries.

Scale happens at two layers. At the app layer, Kubernetes scales the pods of microservices horizontally by load, and then the control plane provisions any infra (for example, new cluster nodes) to support that scale. Therefore, short-term traffic surges and long-term document volume growth can be accommodated by scaling. This is a highly available and fault-tolerant system comprising many disparate pieces: work spreads across more than one availability zone or even region within a single cloud; control planes orchestrate failover to another provider for anything from a small regional outage up to total provider outage—high business continuity. Separation of microservices and patterns, such as Circuit Breaker, stops cascades within the application itself.

It adds portability and cost optimization. Crossplane abstracts infrastructure, while Dapr abstracts application dependencies, making both highly portable in terms of infrastructure definitions and application code. Workloads can be moved between clouds by organizations to achieve better pricing, access new features, or potentially avoid a punitive price increase, thereby breaking the lock-in and enabling continuous cost optimization.

Achieved performance is the main trade-off. Possible network latency that may be accrued in a workflow spanning several clouds, and possible data-egress charges are addressed by co-locating data-intensive services and by applying compression, as well as by designing workflows that minimize cross-cloud interactions. Another challenge is in the control plane's complexity; while it does indeed simplify development, it increases the complexity of tasks at the platform level. Building and maintaining Crossplane compositions and Dapr configurations requires a highly skilled platform-engineering team; hence, abstraction is not free and comes with a demand for expertise. Tool maturity must also be considered. While core components such as Kubernetes, Crossplane, and Dapr are mature CNCF projects, the broader ecosystem is still evolving, which means an adopting organization must be ready to work with leading-edge technology and, if necessary, contribute to the community. Examples of how the proposed architecture addresses specific multi-cloud challenges are presented in Table 2 (Pahl et al., 2018; Innovation at Work, n.d.).

Table 2. Multi-cloud challenges and their architectural solutions (compiled by author)

Multi-Cloud Environment Challenge	Mitigating Architectural Element
Vendor lock-in (infrastructure and dependencies)	Level 1 (Crossplane): Abstracts provider APIs. Level 3 (Dapr): Abstracts service SDKs and APIs (databases, queues). Enables backend changes without modifying application code.
Management complexity and operational overhead	Level 1 (Crossplane + GitOps): Unified declarative API for the entire infrastructure. Automated Git-based management eliminates configuration drift.
Inconsistent security and governance	Level 1 (Control Point): Enforces security policies (OPA/Gatekeeper) during resource creation. Level 4: Centralized identity and security posture management (CSPM).
Low fault tolerance/disaster recovery capabilities	Level 1 (Crossplane): Orchestrates deployments across multiple regions and clouds. Level 2 (Microservices): Failure isolation via patterns such as Circuit Breaker.
Skills shortage/cognitive load on developers	Level 1 (XRDs): Developers request simple abstract resources (IDP-Database). Level 3 (Dapr): Developers use simple state and messaging APIs without learning each cloud's SDK.

Thus, the proposed architecture demonstrates that successful intelligent document processing in a multi-cloud environment requires not only containerising and distributing workloads but also deeply abstracting both infrastructure-level and application-level dependencies. The merge of Crossplane and Dapr into a unified multi-level architecture delivers actual portability, organized management, and growth while maintaining the freedom to choose top-quality cloud offerings. It creates a strong base for developing IDP platforms in fast-evolving markets where quick adjustment is crucial and using assets from different sources efficiently is an essential edge over competitors.

4. Conclusion

This paper addresses the challenges of deploying intelligent document processing systems in multi-cloud environments and the architectural dead-ends that such systems encounter due to vendor lock-in and operational complexity when following traditional approaches. It, therefore, proposes a new four-layer architectural model based on systematic abstraction principles. The paper presents the use of Kubernetes, Crossplane, and Dapr as a strong, principled approach to building the next generation of enterprise AI systems for a multi-cloud world, enabling organizations to achieve real portability, fault tolerance, and cost optimization.

The prototype architecture forms several promising directions for future research. Quantitative performance analysis: build a prototype and benchmark rigorously the performances as well as costs for various IDP processes over different cloud combinations. This would result in an objective assessment of how efficient the solution is in finding the optimal configurations for a particular scenario.

Another approach is to implement self-driving control loops. Stretching the setup with more innovative AI-powered feedback loops could enable automatic workload-placement choices based on up-to-the-minute cost, delay, and performance info, giving more nimble and adjustable multi-cloud handling.

A further prospect is employing a distributed ledger for auditing. Using blockchain or other distributed-ledger technologies could create an immutable, auditable trail of all document-processing actions across clouds, enhancing transparency and simplifying regulatory compliance, especially when handling sensitive data.

Finally, significant potential lies in developing domain-specific compositions. Building an open-source library of Crossplane compositions and Dapr components tailored to common IDP scenarios in finance, healthcare, and insurance would accelerate adoption and increase practical value for particular industries.

References

1. Innovation at Work. (2019). *The Multi-Cloud: Challenges and Solutions*. IEEE. Retrieved July 14, 2025, from https://innovationatwork.ieee.org/the-multi-cloud-challenges-and-solutions/
2. Lin, Y., Hasan, M., Kosalge, R., Cheung, A., & Parameswaran, A. G. (2025). *TWIX: Automatically Reconstructing Structured Data from Templatized Documents*. Arxiv. https://arxiv.org/abs/2501.06659
3. Moravcik, M., Segec, P., Kontsek, M., & Zidekova, L. (2024). Model-Driven Approach to Cloud-Portability Issue. *Applied Sciences*, *14*(20), 9298. https://doi.org/10.3390/app14209298
4. Pahl, C., Jamshidi, P., & Zimmermann, O. (2018). Architectural Principles for Cloud Software. *ACM Transactions on Internet Technology*, *18*(2), 1–23. https://doi.org/10.1145/3104028
5. Pradhan, S., Chandrasekaran, M., Michaelraj, S., & Dutta, S. (2024, March 26). *Build a receipt and invoice processing pipeline with Amazon Textract*. Amazon Web Services. https://aws.amazon.com/ru/blogs/machine-learning/build-a-receipt-and-invoice-processing-pipeline-with-amazon-textract/