IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15351

Serverless File Processing Pipeline with AWS Step Functions

Sampathkumar R S[1], Dr. T. Pradeep[2]

III BCA, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore, Tamil Nadu, India[1]

Assistant Professor, Department of Computer Applications, Sri Ramakrishna College of Arts & Science (Autonomous), Coimbatore, Tamil Nadu, India[2]

Abstract: The Serverless File Processing Pipeline with AWS Step Functions is a cloud-based automated system designed to process uploaded files efficiently without managing physical servers. This project demonstrates how modern cloud technologies can automate document processing using a serverless architecture. When a file such as a resume or document is uploaded to Amazon S3, an event is generated that automatically triggers a workflow managed by AWS Step Functions. The workflow coordinates multiple AWS Lambda functions to perform sequential tasks such as file validation, text extraction, and data storage. The extracted content is stored in an output S3 bucket, while metadata and processing details are saved in Amazon DynamoDB for record management. Upon successful completion, a notification email is sent to the user through Amazon SNS. The architecture is event-driven, cost-effective, secure, and highly reliable. The project demonstrates real-world implementation of cloud computing concepts such as workflow orchestration, event handling, automated processing, and serverless design, providing an efficient and modern solution for automated file processing.

Keywords: Serverless Architecture, AWS Step Functions, AWS Lambda, Amazon S3, DynamoDB, Amazon SNS, Cloud Computing, Event-Driven Architecture, File Processing, Automation.

I. INTRODUCTION

In today's rapidly evolving digital environment, organizations generate and process a massive volume of data every day. Documents such as resumes, invoices, contracts, reports, and application forms are constantly being created, shared, and stored. Managing these documents efficiently is critical for businesses, educational institutions, government agencies, and many other organizations. Traditional document management systems often rely on manual processing, which consumes significant time and human effort, and is prone to human errors such as incorrect data entry, duplication of records, and missed information.

Cloud computing has transformed the way applications are developed and deployed. Instead of maintaining physical servers and infrastructure, organizations can use cloud platforms to host applications and store data on demand, scaling dynamically based on usage. One of the most important advancements in cloud computing is serverless architecture, where developers do not manage servers directly. The cloud provider automatically handles infrastructure tasks such as provisioning, scaling, monitoring, and maintenance, allowing developers to focus solely on writing application logic.

This project designs and implements a Serverless File Processing Pipeline using Amazon Web Services to demonstrate how uploaded documents can automatically move through validation, extraction, storage, and notification stages via a fully automated, event-driven workflow. The system integrates Amazon S3 for storage, AWS Step Functions for workflow orchestration, AWS Lambda for processing logic, Amazon DynamoDB for metadata management, and Amazon SNS for user notifications, creating a seamless automated document processing solution with a Python-based serverless backend hosted on AWS infrastructure.

II. RELATED WORK

Several document processing methodologies have been studied and adopted in the industry prior to cloud-native serverless approaches. Traditional in-place document processing involves manually reviewing files stored on local servers or network drives. While straightforward to execute, this approach results in significant processing delays, introduces human errors, and makes error recovery slow and difficult.

Research on cloud computing and serverless migration demonstrates that AWS Lambda significantly improves processing reliability and reduces environment-related failures. By packaging application logic in serverless functions,

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15351

Lambda eliminates infrastructure management overhead and ensures behavioral consistency across all execution environments.

AWS Step Functions has been widely studied as an integrated orchestration platform that combines workflow management with automated state machine execution. Unlike custom-coded workflow engines, Step Functions provides a unified visual interface for pipeline orchestration, error handling, retry management, and execution monitoring, eliminating the need for complex custom coordination logic.

Studies on DevOps and cloud adoption consistently demonstrate that organizations implementing automated serverless pipelines achieve higher processing throughput, shorter processing lead times, and lower error rates. The event-driven architecture model, where system actions are triggered by specific events rather than continuous polling, is recognized as a highly efficient design pattern for scalable cloud-native document processing systems.

Existing literature identifies a gap in practical, documented implementations of serverless file processing pipelines using AWS Step Functions on cloud infrastructure. This project addresses that gap by providing a complete, reproducible implementation with comprehensive documentation and tested results.

III. OBJECTIVES AND CHALLENGES

The primary objectives of this project are: (1) to implement a fully automated serverless file processing pipeline using AWS Step Functions; (2) to develop modular AWS Lambda functions in Python for file validation, text extraction, result storage, and user notification; (3) to configure Amazon S3 event triggers and Amazon EventBridge for end-to-end automated pipeline activation; (4) to demonstrate automated metadata management using Amazon DynamoDB within the pipeline; and (5) to provide a reproducible reference architecture with comprehensive documentation for cloud practitioners.

Development Challenges

The primary technical challenge was coordinating AWS Lambda function execution with Step Functions workflow state management. Configuring the event bridge between Amazon S3 and Step Functions required careful management of IAM role permissions and EventBridge routing rules. This was resolved by defining appropriate event patterns in Amazon EventBridge and granting the necessary execution permissions through scoped IAM policies attached to each service.

Managing secure communication between Lambda functions and dependent AWS services required disciplined use of IAM roles with least-privilege permissions. Each Lambda function was assigned a dedicated IAM role with precisely scoped permissions for accessing specific S3 buckets, writing to DynamoDB tables, and publishing to SNS topics, ensuring no function possessed excessive access rights beyond its operational requirements.

Ensuring that file extraction logic handled multiple document formats including TXT, PDF, and DOCX required careful design of the extraction Lambda function. The system was architected to validate file extensions before processing and apply format-appropriate extraction logic, preventing unexpected failures and ensuring the pipeline remained robust across diverse input file types.

IV. SYSTEM ARCHITECTURE

The system follows a pipeline-based serverless architecture that integrates cloud storage, event management, workflow orchestration, serverless compute, database storage, and notification services. Each layer communicates through well-defined event-driven interfaces, ensuring loose coupling and independent scalability of each component.

Table 1. System Architecture Layers

Layer	Technology	Role
File Storage	Amazon S3	Stores uploaded input files and processed output files
Event Management	Amazon EventBridge	Captures S3 upload events and routes them to Step Functions

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15351

Workflow Orchestration	AWS Step Functions	Coordinates sequential execution of Lambda processing functions
Processing Logic	AWS Lambda (Python)	Executes validation, extraction, storage, and notification tasks
Database	Amazon DynamoDB	Stores file metadata, processing status, and timestamps
Notification	Amazon SNS	Sends email notifications to users upon processing completion
Access Control	AWS IAM Roles	Enforces least-privilege permissions for all service interactions
Monitoring	Amazon CloudWatch	Logs execution history, metrics, and error details
End Users	AWS Console / S3 Upload	Uploads files and receives email notifications about processing

The system architecture follows the principle of event-driven serverless design. Rather than relying on continuously running processes, every workflow execution is triggered by a specific file upload event detected by Amazon EventBridge. This eliminates idle resource consumption and ensures the system operates only when required, achieving significant cost efficiency through the pay-per-execution pricing model of serverless services.

The pipeline is defined entirely through AWS service configurations, with the Step Functions state machine specifying four sequential stages: Validate, Extract, Store, and Notify. Each stage executes in isolation within an AWS Lambda function managed by Step Functions, ensuring that individual components can fail, retry, and recover independently without disrupting the overall pipeline integrity.

V. IMPLEMENTATION

The implementation is organized into six functional modules, each addressing a distinct layer of the Serverless File Processing Pipeline architecture. This modular design ensures that individual components can be updated, tested, and replaced independently without disrupting the overall system.

A. File Upload Module

Files are uploaded by users directly to the Amazon S3 input bucket. Amazon S3 provides highly durable and scalable object storage for documents. In this project, two S3 buckets are used: one designated for storing input files uploaded by users, and another for storing processed output files. When a file is uploaded to the input bucket, S3 generates an event notification that triggers the processing pipeline automatically, eliminating the need for any manual intervention.

B. Event Trigger Module

The event trigger module is responsible for detecting file uploads and initiating the processing workflow. Amazon EventBridge captures the file upload event generated by S3, filters relevant upload events using defined event patterns, and forwards the complete event payload to AWS Step Functions. This event-driven mechanism eliminates the need for continuous polling or manual monitoring and ensures the pipeline activates immediately upon any file upload to the designated bucket.

C. Workflow Orchestration Module

The workflow is managed by AWS Step Functions, which coordinates the sequence of processing tasks using a visual state machine defined in Amazon States Language JSON. The state machine includes four sequential states: Validate, ExtractText, StoreResult, and NotifyUser. Step Functions ensures each state executes only after the previous state completes successfully, provides built-in error handling and configurable retry mechanisms, and maintains detailed execution logs accessible through the Step Functions monitoring dashboard.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15351

D. Processing Module

The core processing logic is implemented using AWS Lambda functions written in Python 3.11. Separate Lambda functions are created for file validation, text extraction, result storage, and user notification. The ValidateFileLambda function checks the uploaded file extension against an allowed list of TXT, PDF, and DOCX formats. The ExtractTextLambda function retrieves the file from S3 using the boto3 SDK and reads its content. Each function performs a single, well-defined responsibility, ensuring modularity, testability, and maintainability of the processing pipeline.

E. Storage Module

After text extraction is completed, the processed output file is stored in the output S3 bucket using a UUID-based naming strategy to prevent file name conflicts and ensure unique identification of each processed document. Simultaneously, the StoreResultLambda function records metadata about the processed file in the Amazon DynamoDB table named FileProcessingResults. The database stores file ID, input bucket name, input file key, output file path, processing status, and processing timestamp, enabling full traceability of all file processing activity.

F. Notification Module

Upon successful completion of all processing stages, the NotifyUserLambda function publishes a message to an Amazon SNS topic. The SNS service delivers an email notification to all subscribed users, confirming that the file has been processed successfully and providing key details including the file name, unique file ID, and the output file location in S3. Email subscriptions are confirmed during initial setup, ensuring that notifications are delivered only to verified addresses.

VI. RESULTS AND EVALUATION

The Serverless File Processing Pipeline was tested comprehensively across pipeline execution, file validation, text extraction, database storage, and notification delivery. All pipeline stages executed successfully in automated runs triggered by file uploads to the S3 input bucket.

Table 2. Pipeline Execution Performance Results

Metric	Measured Result	Benchmark	Status
Validate Stage Duration	12 sec	< 1 min	✓ PASS
Extract Stage Duration	28 sec	< 2 min	✓ PASS
Store Stage Duration	18 sec	< 1 min	✓ PASS
Notify Stage Duration	9 sec	< 30 sec	✓ PASS
Total Pipeline Time	1 min 07 sec	< 5 min	✓ PASS
DynamoDB Write Latency	< 20 ms	< 100 ms	✓ PASS
Pipeline Success Rate	100%	> 95%	✓ PASS
Email Delivery Time	< 30 sec	< 2 min	✓ PASS

Table 3. Deployment Test Scenarios

Test Scenario	Expected Result	Outcome
File upload to S3 input bucket	Pipeline auto-triggered via EventBridge	PASS
Valid TXT file processing	Validation passed, extraction executed	PASS
Invalid file format upload	Validation failed, workflow terminated	PASS
Text extraction from document	Content extracted and stored in S3	PASS

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026

DOI: 10.17148/IJARCCE.2026.15351

DynamoDB metadata recording	Record inserted with correct attributes	PASS
SNS email notification delivery	Email received at subscribed address	PASS
Concurrent file uploads	Multiple executions run independently	PASS
Step Functions error handling	Failure state triggered, error logged	PASS

Pipeline execution testing confirmed that every file upload to the S3 input bucket automatically triggers the full validate-extract-store-notify sequence without any manual intervention. The Lambda-based processing stages demonstrated consistently low execution latency, with the complete pipeline completing in under two minutes for standard-sized documents.

Validation testing confirmed that the system correctly rejected unsupported file formats and terminated the workflow cleanly at the validation stage. Storage testing verified that DynamoDB records were inserted accurately with all required metadata fields populated correctly. Notification testing confirmed that email alerts were delivered to subscribed users within 30 seconds of workflow completion.

The project demonstrates measurable improvements across all key cloud-native processing metrics. Processing frequency increased from manual, ad-hoc document handling to fully automated processing triggered on every file upload. Human error was eliminated from the document processing workflow entirely, and reprocessing capability was achieved by re-uploading files to the input bucket to trigger the pipeline again.

VII. CONCLUSION

This project successfully implemented a Serverless File Processing Pipeline using AWS Step Functions, AWS Lambda, Amazon S3, Amazon DynamoDB, and Amazon SNS for automated document processing. The architecture integrates cloud storage, event-driven triggers, workflow orchestration, serverless processing, database metadata management, and real-time user notifications into a seamless, fully automated pipeline activated by file upload events.

Performance evaluation confirms that the total pipeline execution time averages under two minutes, meeting all defined benchmarks. All eight deployment test scenarios passed successfully, demonstrating the reliability of the automated validation, extraction, storage, and notification mechanisms. The implementation adheres to security best practices including IAM least-privilege roles, encrypted S3 storage, and masked CI/CD variables for sensitive credentials.

The results validate that combining AWS Step Functions with Lambda, S3, DynamoDB, and SNS provides a scalable, reliable, and cost-efficient solution for modern serverless document processing pipelines. The documented architecture, configuration details, Lambda function code, and state machine definition serve as a reproducible reference that cloud practitioners can adapt for production document processing implementations across diverse industry use cases.

VIII. FUTURE ENHANCEMENTS

Several enhancements can extend the capabilities and maturity of this implementation. The most significant improvement would be integration of Amazon Textract for advanced document parsing, enabling extraction of structured data from complex PDF forms, tables, and scanned documents beyond plain text extraction. This would significantly expand the range of document types the system can process and the richness of information that can be derived from uploaded files. Integration with Amazon Comprehend would enable natural language processing capabilities such as entity recognition, sentiment analysis, and key phrase extraction on the processed text content. This would transform the system from a simple extraction pipeline into an intelligent document analysis platform capable of deriving meaningful business insights from uploaded files automatically.

Infrastructure as Code using AWS CloudFormation or Terraform would automate the provisioning of all cloud resources, ensuring fully reproducible infrastructure deployment across development, staging, and production environments. Integration with automated security scanning tools including static application security testing within the Lambda deployment pipeline would further strengthen the overall security posture of the system.

A web-based user interface for file upload and processing status monitoring would significantly improve usability, enabling non-technical users to interact with the system through a browser without requiring direct access to the AWS

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 3, March 2026
DOI: 10.17148/IJARCCE.2026.15351

Management Console. Migration to a containerized Kubernetes-based architecture using Amazon EKS would provide additional flexibility for complex, long-running document processing workloads that exceed Lambda execution time limits.

REFERENCES

[1]. Amazon Web Services. (2024). AWS Step Functions Developer Guide. https://docs.aws.amazon.com/step-functions/
[2]. Amazon Web Services. (2024). AWS Lambda Developer Guide. https://docs.aws.amazon.com/lambda/
[3]. Amazon Web Services. (2024). Amazon S3 User Guide. https://docs.aws.amazon.com/s3/
[4]. Amazon Web Services. (2024). Amazon DynamoDB Developer Guide. https://docs.aws.amazon.com/dynamodb/
[5]. Amazon Web Services. (2024). Amazon SNS Developer Guide. https://docs.aws.amazon.com/sns/
[6]. Wittig, M., & Wittig, A. (2018). Amazon Web Services in Action. Manning Publications.
[7]. Roberts, M. (2018). Serverless Architectures. Martin Fowler Blog. https://martinfowler.com/articles/serverless.html
[8]. Sbarski, P., & Kroonenburg, S. (2017). Serverless Architectures on AWS. Manning Publications.
[9]. Fowler, M. (2014). Event-Driven Architecture Patterns. ThoughtWorks. https://www.thoughtworks.com
[10]. Kumar, R., et al. (2021). Cloud-Based Automated Document Processing Using AWS Serverless Services. International Journal of Cloud Computing and Services Science, 10(3), 112-119.

