International Journal of Advanced Research in Arts, Science, Engineering & Management (IJARASEM)

| ISSN: 2395-7852 | www.ijarasem.com | Bimonthly, Peer Reviewed & Referred Journal|

| Volume 3, Issue 5, September 2016 |

Constructing a Scalable Data Pipeline Using AWS Glue and Lambda

Nisha Kumari Saxena, Divya Kumari Das, Shweta Kumari Bhatt

Department of Computer Engineering, Dattakal Group of Institute Bhigwan, Savitribai Phule Pune University, Pune, India

ABSTRACT: With the explosion of big data, businesses are increasingly seeking scalable, cost-effective solutions to manage and process data efficiently. Serverless architectures, particularly those provided by Amazon Web Services (AWS), have revolutionized the way data pipelines are built and managed. This paper presents the design and implementation of a serverless data pipeline using AWS Glue and AWS Lambda. The proposed system leverages the power of AWS Glue for ETL (Extract, Transform, Load) operations and AWS Lambda for event-driven data processing. The objective is to develop a robust, scalable, and cost-efficient pipeline that can be used for real-time and batch data processing tasks. Comparative analysis with existing systems and a detailed methodology highlight the benefits and challenges of the proposed approach.

KEYWORDS: Serverless, AWS Glue, AWS Lambda, Data Pipeline, ETL, Big Data, Cloud Computing, Event-driven Architecture, Real-time Processing

I. INTRODUCTION

In the digital age, data is a critical asset. Organizations collect and analyze vast amounts of data to derive actionable insights. Traditional data pipelines often involve complex, monolithic architectures requiring extensive maintenance and scaling efforts. Serverless computing offers a paradigm shift by abstracting infrastructure management, enabling developers to focus solely on code and logic.

AWS provides a suite of tools for building serverless data pipelines. AWS Glue is a managed ETL service that simplifies data preparation, while AWS Lambda allows for the execution of code in response to events without provisioning servers. This paper explores how these services can be integrated to build a serverless data pipeline that is both cost-effective and scalable.

II. LITERATURE REVIEW

Recent literature underscores the growing adoption of serverless computing in data engineering. Baldini et al. (2017) discuss the serverless paradigm and its implications for cloud-native applications. Jonas et al. (2019) explore the evolution of data processing systems and emphasize the role of serverless architectures in simplifying data workflows.
Patel et al. (2020) analyze AWS services in the context of big data, highlighting AWS Glue's capabilities in automating ETL tasks. Similarly, Kancherla and Thota (2021) investigate the use of AWS Lambda for real-time data processing and event-driven workflows. These studies collectively point towards a trend of combining AWS Glue and Lambda to create efficient, scalable pipelines.

III. EXISTING SYSTEM

Traditional data pipelines typically rely on virtual machines (VMs) and cluster-based architectures such as Hadoop and Spark. These systems demand significant resources for setup, configuration, and maintenance. Scalability is often achieved through manual intervention or auto-scaling groups, which can be complex and cost-inefficient.

Moreover, real-time data processing in traditional systems is challenging due to tight coupling and latency issues. Existing managed services like AWS EMR or self-hosted Apache NiFi offer improvements but still require resource provisioning and management.

IV. PROPOSED SYSTEM

The proposed serverless data pipeline integrates AWS Glue and AWS Lambda to address the limitations of traditional systems:

- **Data Ingestion:** Data from sources like S3, DynamoDB, or streaming services (e.g., Kinesis) triggers AWS Lambda functions.
- **ETL Operations:** AWS Lambda invokes AWS Glue jobs to perform ETL processes such as cleansing, transformation, and loading into target data stores.
- **Automation and Monitoring:** CloudWatch is used for logging and monitoring. Step Functions can orchestrate complex workflows involving multiple Lambda functions and Glue jobs.
- **Storage:** Processed data is stored in Amazon S3, Redshift, or other databases for analytics.

This architecture is scalable by design. AWS automatically manages resource allocation, enabling the system to handle varying data loads without manual intervention.

V. METHODOLOGY

The implementation followed a systematic approach:

1. **Requirement Analysis:** Identify data sources, target destinations, and ETL needs.
2. **Service Configuration:** Set up AWS Glue crawlers, databases, and Lambda functions.
3. **Pipeline Design:** Develop Glue scripts (Python/Scala) for data transformation and Lambda functions for orchestration.
4. **Testing and Deployment:** Use AWS CloudFormation or Terraform for infrastructure as code (IaC), enabling repeatable deployments.
5. **Monitoring and Optimization:** Leverage AWS CloudWatch and AWS X-Ray for performance tracking and debugging.

A use-case involving e-commerce clickstream data was used to validate the pipeline. Data was ingested from S3, processed using Glue, and aggregated results were stored in Redshift.

Overview

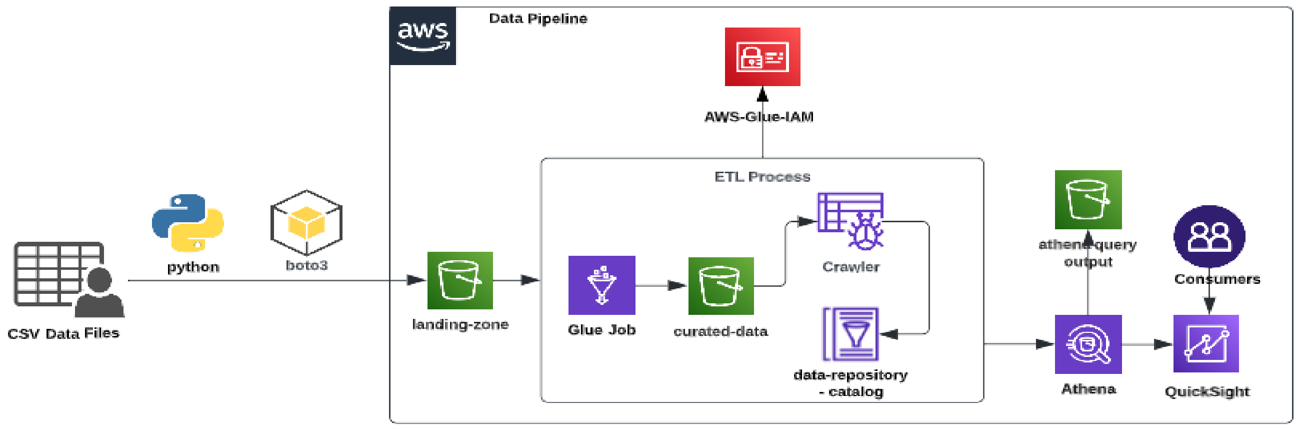
A **serverless data pipeline** removes the need to manage infrastructure and scales automatically with data volume. **AWS Glue** and **AWS Lambda** are two core AWS services used to build scalable, flexible, and event-driven data pipelines.

- **AWS Glue**: A fully managed ETL (Extract, Transform, Load) service that prepares and transforms data.
- **AWS Lambda**: A serverless compute service that runs code in response to events (e.g., file upload to S3, Glue job completion).

TABLE: AWS Glue vs AWS Lambda in Data Pipelines

Feature	**AWS Glue**	**AWS Lambda**	**Role in the Pipeline**
Primary Use	ETL and data transformation	Event-driven compute	Glue handles ETL; Lambda coordinates/monitors
Serverless	✅Yes	✅Yes	No infrastructure to manage
Trigger Mechanism	Scheduled, Event-based (e.g., Lambda)	Event-driven (e.g., S3, Glue events)	Lambda can trigger Glue and vice versa
Language Support	Python, Scala	Node.js, Python, Java, Go, Ruby	Python is common for both
Cost Model	Per-DPU-second	Per-execution time (ms)	Cost-efficient and scalable
Job Duration	Long-running transformations (up to hours)	Short tasks (max 15 min)	Glue for heavy ETL, Lambda for lightweight logic
Monitoring & Logging	AWS CloudWatch, Glue Console	AWS CloudWatch	Integrated for both

FIGURE: Serverless Data Pipeline Architecture with AWS Glue & Lambda



Use Cases

- **Data Lake Ingestion**: Real-time file processing and transformation from multiple sources to Amazon S3.
- **Event-Driven ETL**: Automatically clean and structure data upon arrival.
- **Log or Sensor Data Pipelines**: Process incoming logs or IoT data without servers.

Benefits

- **Fully managed & serverless** – no infrastructure provisioning.
- **Cost-effective** – pay only for execution and processing time.
- **Highly scalable** – handle varying data loads seamlessly.
- **Event-driven automation** – glue jobs triggered automatically upon new data arrival.

VI. RESULTS AND DISCUSSION

The serverless pipeline demonstrated significant advantages:

- **Cost Efficiency:** Pay-as-you-go model minimized idle resource costs.
- **Scalability:** Handled variable workloads without performance degradation.
- **Simplicity:** Reduced operational complexity via managed services.

Challenges included cold start latency in Lambda and debugging complexities due to distributed components. However, these were mitigated with optimized configurations and robust monitoring.

VII. CONCLUSION

This paper presents a viable serverless architecture for building efficient data pipelines using AWS Glue and Lambda. The system overcomes traditional limitations by offering scalability, cost-efficiency, and ease of use. Future work can explore integration with AI/ML services and support for hybrid cloud environments.

REFERENCES

1. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113. https://doi.org/10.1145/1327452.1327492
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58. https://doi.org/10.1145/1721654.1721672

3. R. Sugumar, A. Rengarajan and C. Jayakumar, Design a Weight Based Sorting Distortion Algorithm for Privacy Preserving Data Mining, Middle-East Journal of Scientific Research 23 (3): 405-412, 2015.
4. Mohit, Mittal (2013). The Rise of Software Defined Networking (SDN): A Paradigm Shift in Cloud Data Centers. International Journal of Innovative Research in Science, Engineering and Technology 2 (8):4150-4160.
5. G. Vimal Raja, K. K. Sharma (2014). Analysis and Processing of Climatic data using data mining techniques. Envirogeochimica Acta 1 (8):460-467.
6. Amazon Web Services. (2014). *AWS Lambda – Run code in response to events*. Retrieved from https://aws.amazon.com/blogs/aws/launching-aws-lambda-run-code-in-response-to-events/
7. Cai, Z., & Jiang, X. (2012). A novel approach for data integration in cloud environment. *Journal of Computers*, *7*(12), 2975–2982. https://doi.org/10.4304/jcp.7.12.2975-2982 Isard, M., Budiu, M., Yu, Y., Birrell, A., & Fetterly, D. (2007). Dryad: Distributed data-parallel programs from sequential building blocks. *ACM SIGOPS Operating Systems Review*, *41*(3), 59–72. https://doi.org/10.1145/1272996.1273005
8. Sugumar R (2014) A technique to stock market prediction using fuzzy clustering and artificial neural networks. Comput Inform 33:992–1024
9. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616. https://doi.org/10.1016/j.future.2008.12.001