AWS Glue Cost Analysis and Data Engineering Alternatives for Your Spark Workloads

### June 11, 2025

**Contents**

1. [Executive Summary](#_heading=h.dva9yir6gx0d) 2
2. [Understanding AWS Glue Cost Drivers](#_heading=h.rybius8otbbm) 2
   1. [Key Diagnostic Questions](#_heading=h.gwf9bus4an0a) 2
   2. [Cost and Performance Analysis Approach](#_heading=h.eio9nl71job2) 2
3. [Alternatives to AWS Glue](#_heading=h.q0fvvzi7mwu0) 3
   1. [Amazon EMR on EKS](#_heading=h.wqqq0bsxyp2a) 3
   2. [Amazon EMR on EC2](#_heading=h.rolp77aqnyo) 4
   3. [Amazon EMR Serverless](#_heading=h.gbpn3bdjq88u) 5
4. [Recommendation](#_heading=h.f2yb0roopgxa) 6
   1. [Implementation Steps](#_heading=h.ddb55qad38ly) 6
   2. [When to Stick with Glue](#_heading=h.r7cgryrtn03x) 7
5. [Conclusion](#_heading=h.fi6nxi73bwd3) 7
6. [References](#_heading=h.bhjkok6dkvy6) 7

# Executive Summary

This document analyzes the high costs of AWS Glue. Through diagnostic questions, it identifies cost drivers such as frequent job runs, large datasets, complex Spark pipelines, and limited Spark tuning. Three alternatives—Amazon EMR on EKS, Amazon EMR on EC2, and Amazon EMR Serverless—are evaluated based on performance, scalability, cost-effectiveness, ease of use, flexibility, integration, and operational overhead. AWS benchmarks indicate that EMR on EKS offers up to 61% cost savings and 68% performance improvements for Spark workloads , while EMR Serverless can achieve up to 80% cost savings for intermittent jobs . I recommend **Amazon EMR on EKS** for its optimized performance, containerized scalability, and cost eﬀiciency for your Spark workloads. EMR Serverless is ideal for ad-hoc or intermittent Spark jobs, and EMR on EC2 suits scenarios requiring broader framework support. Practical steps for auditing Glue, optimizing jobs, and piloting alternatives ensure a cost-effective, scalable solution for your architecture.

# Understanding AWS Glue Cost Drivers

High costs in AWS Glue often arise from challenges in performance, scalability, resource utilization, and limited configuration flexibility for Spark workloads. The following diagnostic questions help pinpoint these cost drivers and guide the selection of alternatives for your Spark-based data engineering needs.

## Key Diagnostic Questions

* + 1. Which Glue jobs are actually taking the longest time or using the most DPUs, and why. High DPU (Data Processing Unit) usage may result from complex Spark joins, large datasets, or unoptimized scripts, extending compute time and increasing costs
    2. How frequently do Spark jobs run, and is cost scaling with frequency or data volume. Frequent jobs (e.g., hourly runs lasting 40 minutes) may lead to near-continuous DPU usage, driving up costs. Cost escalation suggests scalability challenges
    3. Are there performance bottlenecks like slow Spark joins or memory issues. Glue job metrics in AWS CloudWatch can reveal high memory usage or slow operations, impacting performance and cost
    4. Is cost driven by data volume or Spark processing challenges. Suboptimal Spark code (e.g., unop- timized Spark SQL) or poorly partitioned data can inflate costs beyond what data volume alone would suggest
    5. Do Spark jobs fail frequently or require retries due to timeouts or memory limits. Failures and retries can double costs, often due to resource contention or Glue’s limitations in handling large- scale Spark jobs
    6. Are Spark pipelines becoming more complex, with multi-step flows or dependencies. Glue is designed for simple ETL tasks but may struggle with complex Spark workflows, leading to resource contention

## Cost and Performance Analysis Approach

To identify cost drivers and performance challenges, I recommend:

* Using AWS Cost Explorer to analyze Glue costs by job name, DPU usage (priced at $0.44 per DPU- hour, billed by the second with a 1-minute minimum ).
* Reviewing CloudWatch metrics for job duration, memory usage, error rates, and resource contention (e.g., CPU or memory bottlenecks).
* Auditing Spark scripts for optimization opportunities.
* Assessing job frequency, data growth, pipeline complexity, and team expertise to evaluate scalability and operational needs.

# Alternatives to AWS Glue

I have evaluated three AWS services as alternatives to Glue for your Spark workloads: Amazon EMR on EKS, Amazon EMR on EC2, and Amazon EMR Serverless. Each is assessed for performance, scalability, cost-effectiveness, ease of use, flexibility, integration, and operational overhead to address Glue’s shortcomings, such as a maximum of 100 DPUs per job, fixed 16GB memory per core, and limited Spark configuration options.

## Amazon EMR on EKS

**Overview:** Amazon EMR on EKS runs Spark workloads on Kubernetes clusters, leveraging containerized environments for optimized performance and scalability.

#### Performance:

* **Advantages:** Offers up to 68% performance improvement for Spark workloads due to optimized container orchestration, addressing memory or join bottlenecks (Key Diagnostic Question 3) . Full control over Spark configurations optimizes joins, memory usage, and partitioning, surpassing Glue’s 32GB executor memory limit with support for instance types up to 24TB of memory .
* **Limitations:** Performance benefits depend on proper Kubernetes and Spark configuration.

#### Scalability:

* **Advantages:** Kubernetes autoscaling dynamically adjusts resources, handling variable Spark workloads or growing datasets eﬀiciently (Key Diagnostic Question 2). Supports large-scale processing without Glue’s DPU caps .
* **Limitations:** Initial EKS cluster setup is more complex than Glue.

#### Cost-Effectiveness:

* **Advantages:** Provides up to 61% cost savings compared to Glue’s $0.44 per DPU-hour due to eﬀicient resource utilization and spot instances (Key Diagnostic Questions 1, 2) .

#### Ease of Use:

* **Advantages:** Simplifies Spark job execution with managed Kubernetes and EMR integration.
* **Limitations:** Requires Kubernetes expertise, less user-friendly than Glue for teams new to containers (Key Diagnostic Question 7) .

#### Flexibility:

* **Advantages:** Supports advanced Spark configurations and Kubernetes-native tools, ideal for com- plex Spark pipelines with intricate dependencies .
* **Limitations:** Focused on Spark, less versatile for non-Spark frameworks compared to EMR on EC2 .

#### Integration:

* **Advantages:** Integrates with Glue Data Catalog, S3, Athena, and AWS Step Functions for or- chestration .
* **Limitations:** Kubernetes-specific integration adds complexity compared to Glue.

#### Operational Overhead:

* **Advantages:** Managed Kubernetes reduces some administrative tasks.
* **Limitations:** EKS cluster management increases overhead compared to Glue .

**Use Case Fit:** Ideal for high-performance, containerized Spark workloads with complex dependencies, frequent runs, or large datasets, leveraging Kubernetes for scalability and isolation (Key Diagnostic Questions 2, 3, 6).

**Reference:** [https://aws.amazon.com/blogs/big-data/amazon-emr-on-amazon-eks/provides-up-to-61-lower-c](https://aws.amazon.com/blogs/big-data/amazon-emr-on-amazon-eks/provides-up-to-61-lower-costs-and-up-to-68-performance-improvement-for-spark-workloads/)

## Amazon EMR on EC2

**Overview:** Amazon EMR on EC2 runs Spark and other frameworks on managed EC2 clusters, offering flexibility for Spark performance tuning and cost optimization.

#### Performance:

* **Advantages:** Full control over Spark configurations optimizes joins, memory usage, and parti- tioning, addressing bottlenecks (Key Diagnostic Question 3). Supports instance types with up to 24TB of memory, surpassing Glue’s 32GB limit .
* **Limitations:** Performance optimization requires expertise in Spark and cluster configuration, without the containerized optimizations of EKS.

#### Scalability:

* **Advantages:** Autoscaling adjusts cluster size based on workload, supporting large-scale Spark datasets and frequent jobs (Key Diagnostic Question 2). Handles hundreds of nodes, exceeding Glue’s DPU limits .
* **Limitations:** Scaling requires configuring minimum and maximum nodes, less dynamic than EKS autoscaling.

#### Cost-Effectiveness:

* **Advantages:** Spot instances (e.g., $0.011–$0.27 per hour per instance ) and reserved instances reduce costs compared to Glue’s $0.44 per DPU-hour, especially for frequent Spark jobs (Key Diagnostic Questions 1, 2) . Autoscaling minimizes overprovisioning.
* **Limitations:** Costs can escalate during peak loads without proper optimization .

#### Ease of Use:

* **Advantages:** EMR Notebooks simplify interactive Spark development, more familiar for teams experienced with EC2.
* **Limitations:** Cluster setup requires more effort than Glue or EMR Serverless (Key Diagnostic Question 7) .

#### Flexibility:

* **Advantages:** Supports Spark, Hive, HBase, Presto, and TensorFlow, enabling diverse workloads beyond Spark (e.g., machine learning, streaming) .
* **Limitations:** Less optimized for containerized Spark workloads compared to EKS.

#### Integration:

* **Advantages:** Integrates with Glue Data Catalog, Athena, Redshift, and Kinesis for streaming .
* **Limitations:** Integration requires manual configuration compared to Glue’s seamless setup.

#### Operational Overhead:

* **Advantages:** Managed service reduces some administrative tasks.
* **Limitations:** Cluster management and security configuration increase overhead compared to Glue or EKS .

**Use Case Fit:** Suitable for Spark workloads requiring broader framework support (e.g., Hive, Presto) or when Kubernetes expertise is limited, but less optimized for containerized Spark compared to EKS (Key Diagnostic Questions 6, 7).

**Reference:** <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html>

## Amazon EMR Serverless

**Overview:** Amazon EMR Serverless runs Spark and Hive workloads without cluster management, offering a serverless alternative to Glue with fine-grained scaling.

#### Performance:

* **Advantages:** Achieves up to 80% cost savings for intermittent Spark jobs due to faster cold-start times and optimized runtime (Key Diagnostic Question 3) . Pre-initialized capacity enables near-instant job starts for interactive analytics .
* **Limitations:** Dynamic resource allocation may introduce slight job latency for time-sensitive Spark workloads .

#### Scalability:

* **Advantages:** Automatically scales workers based on workload, ideal for intermittent or variable Spark jobs (Key Diagnostic Question 2). Handles petabyte-scale processing without DPU limits .
* **Limitations:** Less control over scaling compared to EKS or EC2 .

#### Cost-Effectiveness:

* **Advantages:** Pay-per-use pricing (e.g., $0.0526 per vCPU-hour, $0.00526 per GB-memory-hour ) avoids overprovisioning, offering savings for short or intermittent Spark jobs (Key Diagnostic Questions 1, 2). Up to 80% cheaper than Glue for ad-hoc workloads .
* **Limitations:** Costs may exceed EKS or EC2 for long-running Spark jobs due to higher per-unit pricing .

#### Ease of Use:

* **Advantages:** No cluster management simplifies setup, approaching Glue’s ease of use (Key Diagnostic Question 7) .
* **Limitations:** Limited Spark configuration options compared to EKS or EC2.

#### Flexibility:

* **Advantages:** Supports Spark and Hive, suitable for batch and streaming Spark workloads .
* **Limitations:** Less flexible than EKS or EC2 for advanced Spark tuning .

#### Integration:

* **Advantages:** Integrates with Glue Data Catalog, S3, and AWS Step Functions for orchestration .
* **Limitations:** Fewer integration options than EKS or EC2 for non-AWS services.

#### Operational Overhead:

* **Advantages:** Serverless model eliminates cluster management, minimizing overhead .
* **Limitations:** Monitoring requires CloudWatch integration, similar to Glue.

**Use Case Fit:** Ideal for ad-hoc, intermittent, or streaming Spark workloads requiring minimal setup and cost eﬀiciency (Key Diagnostic Questions 2, 6, 7).

**Reference:** [https://aws.amazon.com/blogs/big-data/announcing-amazon-emr-serverless-preview-run-big-da](https://aws.amazon.com/blogs/big-data/announcing-amazon-emr-serverless-preview-run-big-data-applications-without-managing-servers/)

# Recommendation

I recommend **Amazon EMR on EKS** as the primary alternative to AWS Glue for your Spark workloads due to its optimized performance, containerized scalability, and cost eﬀiciency. Key reasons include:

* **Performance:** Offers up to 68% performance improvement for Spark workloads, with full control over configurations to optimize joins, memory usage, and partitioning, addressing Glue’s 32GB executor memory limit with support for instance types up to 24TB (Key Diagnostic Question 3).
* **Scalability:** Kubernetes autoscaling dynamically handles variable Spark workloads and large datasets, surpassing Glue’s 100 DPU limit (Key Diagnostic Question 2) .
* **Cost-Effectiveness:** Provides up to 61% cost savings compared to Glue’s $0.44 per DPU-hour through eﬀicient resource utilization and spot instances (Key Diagnostic Questions 1, 2).
* **Flexibility:** Supports advanced Spark configurations and Kubernetes-native tools, ideal for com- plex Spark pipelines with dependencies (Key Diagnostic Question 6) .
* **Isolation:** Containerized environments ensure resource isolation, reducing contention for concur- rent Spark jobs (Key Diagnostic Question 5) .

**Amazon EMR Serverless** is recommended for ad-hoc or intermittent Spark workloads (e.g., daily 10-minute jobs) due to its ease of use, minimal overhead, and up to 80% cost savings compared to Glue (Key Diagnostic Questions 2, 7) . **Amazon EMR on EC2** is suitable for Spark workloads requiring broader framework support (e.g., Hive, Presto) or when Kubernetes expertise is limited, offering similar 24TB memory support but less optimized for containerized Spark (Key Diagnostic Questions 6, 7) .

## When to Stick with Glue

Glue remains suitable for your Spark workloads if:

* Spark jobs are simple, single-step ETL tasks with minimal dependencies and low resource con- tention
* Ease of use case and minimal operational overhead are priorities (Key Diagnostic Question 7)
* : Integration with AWS Glue Data Catalog and services like Athena is critical for

* Performance and scalability needs are met through optimization (e.g., partitioning, caching Spark)

# Conclusion

High AWS Glue costs for Spark workloads often stem from challenges in performance, scalability, resource contention, and limited flexibility, particularly for frequent runs, large datasets, or complex pipelines. I recommend Amazon EMR on EKS for its optimized performance (up to 68% improvement), containerized scalability, and cost eﬀiciency (up to 61% savings) for Spark workloads ]. Amazon EMR Serverless excels for ad-hoc or intermittent Spark jobs, offering ease of use and up to 80% cost savings ], while Amazon EMR on EC2 is suitable for Spark workloads needing broader framework support or simpler setup ]. By auditing Glue usage, optimizing Spark jobs, and piloting EKS, your architecture can achieve optimal performance, scalability, and cost-effectiveness for Spark-based data engineering needs.

# References

* [https://aws.amazon.com/blogs/big-data/amazon-emr-on-amazon-eks-provides-up-to-61-lower-costs-a](https://aws.amazon.com/blogs/big-data/amazon-emr-on-amazon-eks-provides-up-to-61-lower-costs-and-up-to-68-performance-improvement-for-spark-workloads/)
* <https://www.missioncloud.com/blog/amazon-emr-serverless-vs-aws-glue>
* <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html>
* [https://aws.amazon.com/blogs/big-data/optimizing-costs-for-apache-spark-workloads-on-amazon-em](https://aws.amazon.com/blogs/big-data/optimizing-costs-for-apache-spark-workloads-on-amazon-emr/)
* <https://aws.amazon.com/emr/pricing/>
* [https://aws.amazon.com/blogs/big-data/amazon-emr-launch-time-performance-improvements-and-cost-](https://aws.amazon.com/blogs/big-data/amazon-emr-launch-time-performance-improvements-and-cost-reductions-for-price-sensitive-apache-spark-workloads/)
* <https://aws.amazon.com/glue/pricing/>
* [https://aws.amazon.com/blogs/big-data/announcing-amazon-emr-serverless-preview-run-big-data-ap](https://aws.amazon.com/blogs/big-data/announcing-amazon-emr-serverless-preview-run-big-data-applications-without-managing-servers/)
* [https://aws.amazon.com/blogs/big-data/introducing-amazon-emr-serverless-ga-run-big-data-applic](https://aws.amazon.com/blogs/big-data/introducing-amazon-emr-serverless-ga-run-big-data-applications-without-managing-servers/)
* <https://aws.amazon.com/blogs/big-data/integrating-amazon-emr-with-the-aws-glue-data-catalog/>
* [https://aws.amazon.com/blogs/big-data/streamline-your-data-processing-with-amazon-emr-serverles](https://aws.amazon.com/blogs/big-data/streamline-your-data-processing-with-amazon-emr-serverless-and-the-aws-sdk-for-pandas/)