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

### Prepared for Your Architecture Team June 15, 2025

**Contents**

1. [Executive Summary](#_bookmark0) 2
2. [Understanding AWS Glue Limitations](#_bookmark1) 2
   1. [Key Diagnostic Questions](#_bookmark2) 2
   2. [Performance Analysis Approach](#_bookmark3) 2
3. [Alternatives to AWS Glue](#_bookmark4) 3
   1. [Amazon EMR on EKS](#_bookmark5) 3
   2. [Amazon EMR on EC2](#_bookmark6) 3
   3. [Amazon EMR Serverless](#_bookmark7) 4
4. [Cost Comparison for Single DPU Compared Compute (1 Day, 24 Hours)](#_bookmark8) 5
   1. [Service and Pricing Breakdown](#_bookmark9) 5
   2. [Summary Comparison Table](#_bookmark10) 6
5. [Recommendation](#_bookmark11) 6
   1. [Implementation Steps](#_bookmark12) 7
   2. [When to Stick with Glue](#_bookmark13) 7
6. [Conclusion](#_bookmark14) 8
7. [References](#_bookmark15) 8

# Executive Summary

This document analyzes the limitations of AWS Glue, a serverless ETL service, focusing on challenges in performance, scalability, resource contention, and configuration flexibility for Spark workloads requiring 1 DPU (G.1X: 4 vCPUs, 16 GB memory, 64 GB disk storage) over 24 hours daily in the US East (Ohio) region. Through diagnostic questions, it identifies performance bottlenecks, scalability constraints, and complexity issues. Three alternatives—Amazon EMR on EC2, Amazon EMR on EKS, and Amazon EMR Serverless—are evaluated based on performance, scalability, ease of use, flexibility, integration, and operational overhead, standardized to 1 DPU for an apples-to-apples comparison [**?**, **?**, **?**, **?**, **?**]. AWS benchmarks indicate that EMR on EKS offers up to 68% performance improvements for Spark workloads [**?**], while EMR Serverless simplifies intermittent jobs [**?**]. I recommend **Amazon EMR on EKS** for its optimized performance and containerized scalability for your Spark workloads. EMR Serverless is ideal for ad-hoc or intermittent jobs, and EMR on EC2 suits scenarios requiring broader framework support. Practical steps for auditing Glue, optimizing jobs, and piloting alternatives ensure a scalable solution for your architecture.

# Understanding AWS Glue Limitations

Limitations in AWS Glue often arise from challenges in performance, scalability, resource utilization, and configuration flexibility for Spark workloads using 1 DPU (G.1X: 4 vCPUs, 16 GB memory, 64 GB disk storage). The following diagnostic questions help pinpoint these issues and guide the selection of alternatives for your Spark-based data engineering needs.

## Key Diagnostic Questions

* + 1. Which Glue jobs consume significant resources, and why? High resource usage may result from complex Spark joins, large datasets, or unoptimized scripts, extending compute time.
    2. How frequently do Spark jobs run, and is performance scaling with frequency or data volume? Continuous 24-hour daily jobs may expose scalability limitations.
    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.
    4. Is performance limited by data volume or Spark processing challenges? Suboptimal Spark code (e.g., unoptimized Spark SQL) or poorly partitioned data can degrade performance.
    5. Do Spark jobs fail frequently or require retries due to timeouts or memory limits? Failures and retries can disrupt workflows, often due to resource contention or Glue’s limitations.
    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.
    7. Does your team prioritize ease of use or require advanced Spark customization? Glue’s serverless model simplifies setup but limits flexibility compared to alternatives.

## Performance Analysis Approach

To identify performance and scalability challenges, I recommend:

* Using AWS Cost Explorer to analyze Glue resource usage by job name and DPU allocation (1 DPU for G.1X, billed by the second with a 1-minute minimum [**?**]).
* Reviewing CloudWatch metrics for job duration, memory usage (limited to 16*.*00 GB per DPU with G.1X [**?**]), error rates, and resource contention.
* Auditing Spark scripts for optimization opportunities (e.g., partitioning, caching, or reducing shuf- fles).
* 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 requiring 1 DPU (G.1X: 4 vCPUs, 16 GB memory, 64 GB disk storage) over 24 hours daily: Amazon EMR on EC2, Amazon EMR on EKS, and Amazon EMR Serverless. Each is assessed for performance, scalability, ease of use, flexibility, integration, and operational overhead to address Glue’s shortcomings, such as limited Spark configuration options and a maximum of 100 DPUs per job [**?**, **?**]. These alternatives are configured to match 1 DPU’s compute resources for an apples-to-apples comparison.

## Amazon EMR on EKS

**Overview:** Amazon EMR on EKS runs Spark workloads on Kubernetes clusters, leveraging container- ized 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 and partitioning, surpassing Glue’s limitations [**?**].
* **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) [**?**].
* **Limitations:** Initial EKS cluster setup is more complex than Glue.

#### 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 con- tainers (Key Diagnostic Question 7) [**?**].

#### Flexibility:

* **Advantages:** Supports advanced Spark configurations and Kubernetes-native tools, ideal for com- plex Spark pipelines with 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 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.

#### Performance:

* **Advantages:** Full control over Spark configurations optimizes joins, memory usage, and parti- tioning, addressing bottlenecks (Key Diagnostic Question 3). Supports instance types with high memory, surpassing Glue’s 16 GB limit [**?**].
* **Limitations:** Performance optimization requires expertise in Spark and cluster configuration.

#### Scalability:

* **Advantages:** Autoscaling adjusts cluster size based on workload, supporting large-scale Spark datasets (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.

#### 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 [**?**].
* **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 [**?**].

**Use Case Fit:** Suitable for Spark workloads requiring broader framework support or when Kubernetes expertise is limited (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:** Optimized runtime supports intermittent Spark jobs with fast cold-start times (Key Diagnostic Question 3) [**?**]. Pre-initialized capacity enables near-instant job starts [**?**].
* **Limitations:** Dynamic resource allocation may introduce slight job latency for time-sensitive workloads [**?**].

#### Scalability:

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

#### Ease of Use:

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

#### Flexibility:

* **Advantages:** Supports Spark and Hive, suitable for batch and streaming 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 (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/)

# Cost Comparison for Single DPU Compared Compute (1 Day, 24 Hours)

To compare the costs of AWS Glue, Amazon EMR on EC2, Amazon EMR on EKS, and Amazon EMR Serverless for Spark workloads, I standardized compute power to 1 AWS Glue DPU (G.1X: 4 vCPUs, 16 GB memory, 64 GB disk storage) [**?**]. Costs are calculated for a 24-hour Spark job (24 hours total) in the US East (Ohio) region, based on pricing documentation as of June 2025 [**?**, **?**, **?**, **?**, **?**]. All configurations are aligned for an apples-to-apples comparison, with EMR on EC2 and EMR on EKS using m5.xlarge instances (4 vCPUs, 16 GB memory) and EMR Serverless matching 4 vCPUs and 16 GB memory with 20 GB free storage.

## Service and Pricing Breakdown

#### AWS Glue (G.1X worker) Specification per DPU:

* 4 vCPUs
* 16 GB Memory
* 64 GB Disk Storage (included)

#### Cost Calculation:

* Cost per DPU-hour: $0*.*44
* 24-hour cost: 24*.*00 hours × $0*.*44 = $10*.*56

#### Resource Cost Breakdown (Included in DPU cost)

vCPU (4 cores) Included Memory (16 GB) Included Storage (64 GB) Included

**Total Glue Cost:** $10*.*56 (flat rate)

#### Amazon EMR on EC2

**Equivalent Instance:** m5.xlarge (4 vCPUs, 16 GB memory)

**EC2 Pricing:** On-demand, US East (Ohio)

* Instance hourly cost: approximately $0*.*19 (typical cost for m5.xlarge)
* EMR hourly uplift: typical uplift for 4 vCPU instance $0*.*06/h (est. typical)

|  |  |  |
| --- | --- | --- |
| **Resource** | **Cost per Hour** | **Total (24 hours)** |
| EC2 Instance (4 vCPU, 16 GB) | $0*.*19 | $4*.*61 |
| EMR service uplift | $0*.*06 | $1*.*44 |
| Storage (EBS, ephemeral) | Included in EC2 cost | Included |

#### Total EMR on EC2 Cost (24h): $6*.*05

#### Amazon EMR on EKS

#### Pricing dimensions provided explicitly (Ohio Region):

* vCPU: $0*.*01/hour
* Memory: $0*.*00/GB/hour

#### Resource Rate (per hour) 24

**Cost Calculation:** (4 vCPUs, 16 GB RAM)

vCPU (4) 4*.*00 × $0*.*01 = $0*.*04

Memory (16 GB) 16*.*00 × $0*.*00 = $0*.*02

EC2 Instance underlying (m5.xlarge) $0*.*19 (similar to EC2 above)

**Total EMR on EKS Cost (24h):** $0*.*97 + $0*.*43 + $4*.*61 = $6*.*01

#### Amazon EMR Serverless

#### Pricing details explicitly provided (Ohio Region):

* vCPU: $0*.*05/hour
* Memory: $0*.*01/GB/hour
* Storage: First 20 GB free; additional storage priced separately.

#### Resource Cost per ho

**Cost Calculation:** (4 vCPUs, 16 GB RAM, default 20 GB storage included free) vCPU (4 cores) 4*.*00 × $0*.*05 =

Memory (16 GB) 16*.*00 × $0*.*01 =

Storage (20 GB) Included free (first

**Total EMR Serverless Cost (24h):** $5*.*05 + $2*.*22 = $7*.*27

## Summary Comparison Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Service** | **Total Cost (24h)** | **vCPU cost** | **Memory cost** | **Storage cost** |
| AWS Glue | $10*.*56 | Included | Included | Included |
| EMR on EC2 | $6*.*05 | $4*.*61 *∗* | Included\* | Included |
| EMR on EKS | $6*.*01 | $0*.*97 | $0*.*43 | $4*.*61 (EC2 instance) |
| EMR Serverless | $7*.*27 | $5*.*05 | $2*.*22 | Included (20GB free) |

\*Included within EC2 instance pricing.

# Recommendation

I recommend **Amazon EMR on EKS** as the primary alternative to AWS Glue for your Spark workloads requiring 1 DPU (G.1X: 4 vCPUs, 16 GB memory, 64 GB disk storage) over 24 hours daily, due to its optimized performance and containerized scalability. Key reasons include:

* **Performance:** Offers up to 68% performance improvement for Spark workloads, with full control over configurations to optimize joins and partitioning, addressing Glue’s limitations (Key Diagnos- tic Question 3) [**?**, **?**].
* **Scalability:** Kubernetes autoscaling dynamically handles variable Spark workloads and large datasets, surpassing Glue’s 1 DPU limit (Key Diagnostic Question 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 on EC2** is recommended for stable Spark workloads requiring broader framework support (e.g., Hive, Presto) or when Kubernetes expertise is limited (Key Diagnostic Questions 6, 7) [**?**, **?**]. **Amazon EMR Serverless** is ideal for ad-hoc or intermittent Spark workloads, offering ease of use and minimal overhead (Key Diagnostic Questions 2, 7) [**?**, **?**].

## Implementation Steps

#### Audit Glue Usage:

* + - * Use AWS Cost Explorer and CloudWatch to identify high-resource Spark jobs, failure rates, and bottlenecks (Key Diagnostic Questions 1, 3, 4, 5).
      * Assess job frequency, data volume, pipeline complexity, and team expertise (Key Diagnostic Questions 2, 6, 7).

#### Optimize Glue as an Interim Step:

* + - * Optimize Spark scripts (e.g., partitioning, caching) to reduce resource usage (Key Diagnostic Question 4).
      * Adjust job frequency to align with 24-hour runs (Key Diagnostic Question 2).

#### Pilot EMR on EKS:

* + - * Migrate a Glue Spark job to EMR on EKS, using optimized Spark settings ([https://docs.](https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-Getting-Started/emr-eks-getting-started.html) [aws.amazon.com/emr/latest/EMR-on-EKS-Getting-Started/emr-eks-getting-started.](https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-Getting-Started/emr-eks-getting-started.html) [html](https://docs.aws.amazon.com/emr/latest/EMR-on-EKS-Getting-Started/emr-eks-getting-started.html)).
      * Monitor performance using CloudWatch.

#### Evaluate EMR Serverless and EMR on EC2:

* + - * Test EMR Serverless for ad-hoc Spark jobs ([https://aws.amazon.com/blogs/big-data/](https://aws.amazon.com/blogs/big-data/announcing-amazon-emr-serverless-preview-run-big-data-applications-without-managing-servers/)

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

* + - * Test EMR on EC2 for workloads needing non-Spark frameworks ([https://docs.aws.amazon.](https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html) [com/emr/latest/ManagementGuide/emr-overview.html](https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html)).
      * Compare performance and ease of use with EMR on EKS.

#### Scale Migration:

* + - * Expand migration to other Spark jobs if performance is validated.
      * Use AWS Step Functions for complex Spark pipelines (Key Diagnostic Question 6).

## 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 and minimal operational overhead are priorities (Key Diagnostic Question 7).
* Integration with AWS Glue Data Catalog and services like Athena is critical.
* Performance and scalability needs are met through optimization (e.g., partitioning, caching) for 1 DPU workloads [**?**].

# Conclusion

AWS Glue’s limitations for Spark workloads requiring 1 DPU (G.1X: 4 vCPUs, 16 GB memory, 64 GB disk storage) over 24 hours daily in the US East (Ohio) region often stem from challenges in performance, scalability, resource contention, and flexibility, particularly for complex pipelines or large datasets [**?**]. I recommend Amazon EMR on EKS for its optimized performance (up to 68% improvement) and containerized scalability for Spark workloads [**?**, **?**, **?**, **?**, **?**]. Amazon EMR on EC2 is ideal for stable workloads or broader framework support, while Amazon EMR Serverless excels for ad-hoc or intermittent jobs [**?**, **?**, **?**]. By auditing Glue usage, optimizing Spark jobs, and piloting EKS, your architecture can achieve optimal performance and scalability for Spark-based data engineering needs.

# References

* [https://aws.amazon.com/blogs/big-data/amazon-emr-on-amazon-eks/provides-up-to-61-lower-cos](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-amazo](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-](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/blogs/big-data/announcing-amazon-emr-serverless-preview-run-big-dat](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-ap](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-catalo](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-serv](https://aws.amazon.com/blogs/big-data/streamline-your-data-processing-with-amazon-emr-serverless-and-the-aws-sdk-for-pandas/)

[**?**]

* <https://aws.amazon.com/ec2/pricing/on-demand/> [**?**]
* <https://aws.amazon.com/emr/features/eks/> [**?**]
* <https://aws.amazon.com/emr/serverless/pricing/> [**?**]
* [https://docs.aws.amazon.com/pdfs/whitepapers/latest/cost-modeling-data-lakes/cost-modeling-](https://docs.aws.amazon.com/pdfs/whitepapers/latest/cost-modeling-data-lakes/cost-modeling-data-lakes.pdf) [pdf](https://docs.aws.amazon.com/pdfs/whitepapers/latest/cost-modeling-data-lakes/cost-modeling-data-lakes.pdf) [**?**]