AWS Prescriptive Guidance

Getting started with serverless ETL on AWS Glue

AWS Prescriptive Guidance: Getting started with serverless ETL on AWS Glue

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Introduction 1
AWS Glue ETL 2
Authoring in AWS Glue ETL 2
DPUs and worker types 2
Using Python shell 2
Python or Scala on Apache Spark 2
Data Catalog 3
Databases and tables 3
Crawlers and classifiers 3
Connections 4
AWS Glue Schema Registry 4
Features and concepts 6
Logging and monitoring 6
Monitoring options 6
Automation 7
Triggers 8
Workflows 8
Integration 8
Bookmarks 8
DataBrew 9
Best practices 10
Run locally first 10
Use a development endpoint 10
Use partitioning 10
Memory management 10
Scaling 11
FAQ 12
When should I use AWS Glue Python shell instead of AWS Glue with Spark? 12
What is the difference between AWS Glue version 1.0 and AWS Glue version 2.0? 12
Next steps 13
AWS Glue transformations 13
Authoring your first ETL job 13
Pricing 13
Additional resources 14
References 14
Patterns 14
Document history 15

Getting started with serverless ETL on AWS Glue

Adnan Alvee, Data Architect, Amazon Web Services

January 2021

On the Amazon Web Services (AWS) Cloud, AWS Glue provides a fully managed serverless environment where you can extract, transform, and load (ETL) your data. AWS Glue makes it cost-effective to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams.

AWS Glue is serverless, so you don't need to set up any cluster or infrastructure, and you can scale up and down as you need. AWS Glue charges only for the amount of compute time it needs to finish a specific workload. Instead of juggling between various instance types to match a workload, AWS Glue comes with the capability of choosing between three worker types recommended for various workloads:

- Standard
- G.1X (for memory-intensive workloads)
- G.2X (for workloads with ML transforms)

AWS Glue consists of the following components:

- **AWS Glue ETL** – AWS Glue ETL gives you batch and streaming options to author code and move, transform, and aggregate data from one source to another.
- **AWS Glue Data Catalog** – Data Catalog provides a view of the metadata of all your data along with options to crawl specific data sources.
- **AWS Glue DataBrew** – DataBrew is a visual data preparation tool that data analysts and data scientists can use to clean and normalize data. You can choose from more than 250 prebuilt transformations to automate data preparation tasks, all without the need to write any code.

This guide gives you a brief introduction to each of the AWS Glue components and the key concepts and features, such as logging and monitoring, and automation, that you should know before authoring ETL jobs. The Next steps (p. 13) section can get you up to speed with writing code in AWS Glue. If you already have used AWS Glue to some extent, you can use the Best practices (p. 10) section to brush up on any gaps.

AWS Glue ETL

You can use AWS Glue ETL to move, clean, and transform data from one source to another by using its own functions, Apache Spark, and Python. For more information about supported sources, see Connection types and options for ETL in AWS Glue.

Authoring in AWS Glue ETL

AWS Glue ETL code can be authored in the following ways:

- **Python shell** – When the size of the data is very small, you can use Python shell to write Python scripts to manipulate data.
- **Spark** – You can use Scala and Python to author ETL jobs with AWS Glue and Apache Spark.
- **Spark Streaming** – To enrich, aggregate, and combine streaming data, you can use streaming ETL jobs. In AWS Glue, streaming ETL jobs run on the Apache Spark Structured Streaming engine.
- **AWS Glue Studio** – If you are new to Apache Spark programming or are accustomed to ETL tools with boxes-and-arrows interfaces, get started by using AWS Glue Studio.

DPUs and worker types

AWS Glue uses data processing units (DPUs), which are also known as worker types. DPUs can be allocated based on the data and velocity needs of a specific use case.

Using Python shell

You can use either 1 DPU to use 16 GB of memory or 0.0625 DPU to use 1 GB of memory. Note that a Python shell job does not use the Apache Spark environment to run Python, so it is not shown in the following table. Python shell is for jobs where the size of data is very small.

AWS Glue Python or Scala on Apache Spark

The following table shows the different AWS Glue worker types for the Apache Spark run environment for batch, streaming, and AWS Glue Studio workloads. Note that with AWS Glue Studio, you can use only G.1X and G.2X worker types.

	Standard	**G.1X**	**G.2X**
vCPU	4	4	8
Memory	16 GB	16 GB	32 GB
Disk space	50	64	128
Executor per worker	2	1	1

AWS Glue Data Catalog

The AWS Glue Data Catalog consists of the following components:

- Databases and tables
- Crawlers and classifiers
- Connections
- AWS Glue Schema Registry

AWS Glue databases and tables

The Data Catalog consists of database and tables. A table can be in only one database. Your database can contain tables from many different sources that AWS Glue supports.

The following image shows a sample view of a Data Catalog database and corresponding tables.

Name	Database	Location	Classification	Last updated
foursquare_airports	adx2	s3://processed-external-d...	parquet	19 May 2020 2:23 PM UTC-5
foursquare_casual_dining_chains	adx2	s3://processed-external-d...	parquet	15 May 2020 1:33 AM UTC-5
foursquare_casual_dining_chains	adx	s3://processed-external-d...	csv	2 June 2020 10:29 AM UT...
foursquare_fast_food	adx	s3://processed-external-d...	csv	2 June 2020 10:28 AM UT...
foursquare_fast_food	adx2	s3://processed-external-d...	parquet	15 May 2020 1:33 AM UTC-5

You can create the database and tables in the following ways:

- The AWS Glue crawler
- An AWS Glue ETL job
- AWS Glue console
- The `CreateTable` operation in the AWS Glue API
- AWS CloudFormation templates
- A migrated Apache Hive metastore

Currently, AWS Glue supports Amazon Simple Storage Service (Amazon S3), Amazon DynamoDB, and Java Database Connectivity (JDBC) data sources.

AWS Glue crawlers and classifiers

A crawler helps create and update the Data Catalog tables. It can crawl both file-based and table-based data stores.

Crawlers can crawl the following data stores through their respective native interfaces:

- Amazon S3

- DynamoDB

Crawlers can crawl the following data stores through a JDBC connection:

- Amazon Redshift
- Amazon Relational Database Service (Amazon RDS)
  - Amazon Aurora
  - Microsoft SQL Server
  - MySQL
  - Oracle
  - PostgreSQL
- Publicly accessible databases (on-premises or on another cloud provider environment)
  - Aurora
  - Microsoft SQL Server
  - MySQL
  - Oracle
  - PostgreSQL

In the AWS Glue crawler, a classifier recognizes the format of the data and generates the schema. AWS Glue comes with set of built-in classifiers, but you can also create custom classifiers.

The built-in classifiers for various formats include JavaScript Object Notation (JSON), comma-separated values (CSV), web logs, and many database systems.

If AWS Glue doesn't find a custom classifier that fits the input data format with 100 percent certainty, it invokes the built-in classifiers. The classifiers are invoked in the order that is shown in the table in Built-in classifiers in AWS Glue. The built-in classifiers return a result to indicate whether the format matches (certainty=1.0) or does not match (certainty=0.0). The first classifier that has certainty=1.0 provides the classification string and schema for a metadata table in your Data Catalog.

AWS Glue connections

You can use connections to define connection information, such as login credentials and virtual private cloud (VPC) IDs, in one place. This saves time, because you don’t need to provide connection information every time you create a crawler or job.

The following connection types are available:

- JDBC
- Amazon RDS
- Amazon Redshift
- MongoDB, including Amazon DocumentDB (with MongoDB compatibility)
- Network (designates a connection to a data source within a VPC environment on AWS)

AWS Glue Schema Registry

The AWS Glue Schema Registry enables disparate systems to share a schema for serialization and deserialization. For example, assume that you have a producer and a consumer of data. The producer knows the schema when it publishes the serialized data. The consumer uses the Schema Registry

deserializer library that parses the schema version ID from the record payload. The consumer then uses the schema to deserialize the data. For examples of use cases regarding producer and consumer resources, see Integrating with AWS Glue Schema Registry.

With the AWS Glue Schema Registry, you can manage and enforce schemas on your data streaming applications using convenient integrations with the following data input sources:

- Apache Kafka
- Amazon Managed Streaming for Apache Kafka
- Amazon Kinesis Data Streams
- Amazon Kinesis Data Analytics for Apache Flink
- AWS Lambda

Schema Registry consists of the following components:

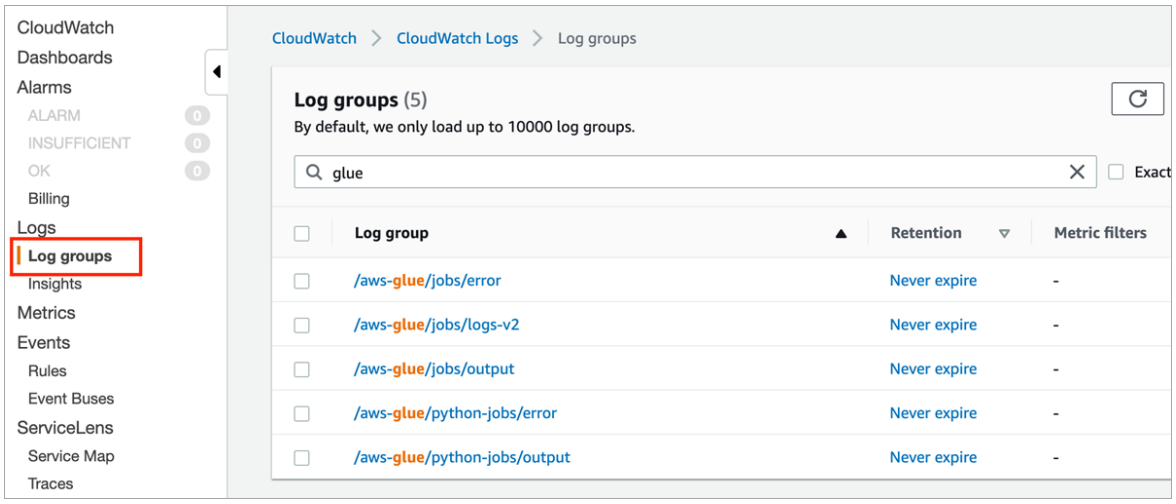
- **Schemas** – A schema is the abstraction to represent the structure and format of a data record.
- **Registry** – A registry is a logical container of schemas. Registries allow you to organize your schemas and manage access control for your applications.

Important features and concepts

With AWS Glue, you can debug, monitor performance, and automate and run jobs in the same S3 bucket without re-running a job on the same set of data.

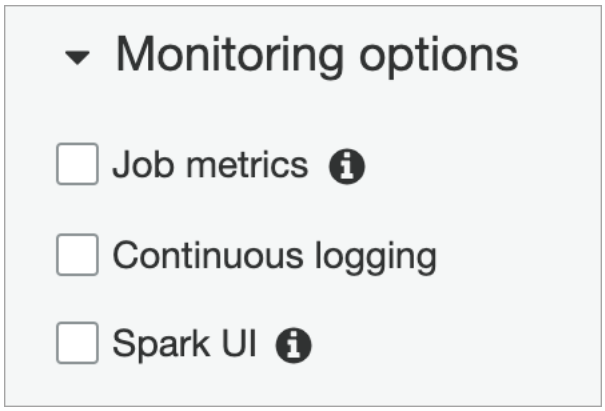
Logging and monitoring

By default, AWS Glue sends logs to Amazon CloudWatch under the `aws-glue` log group.

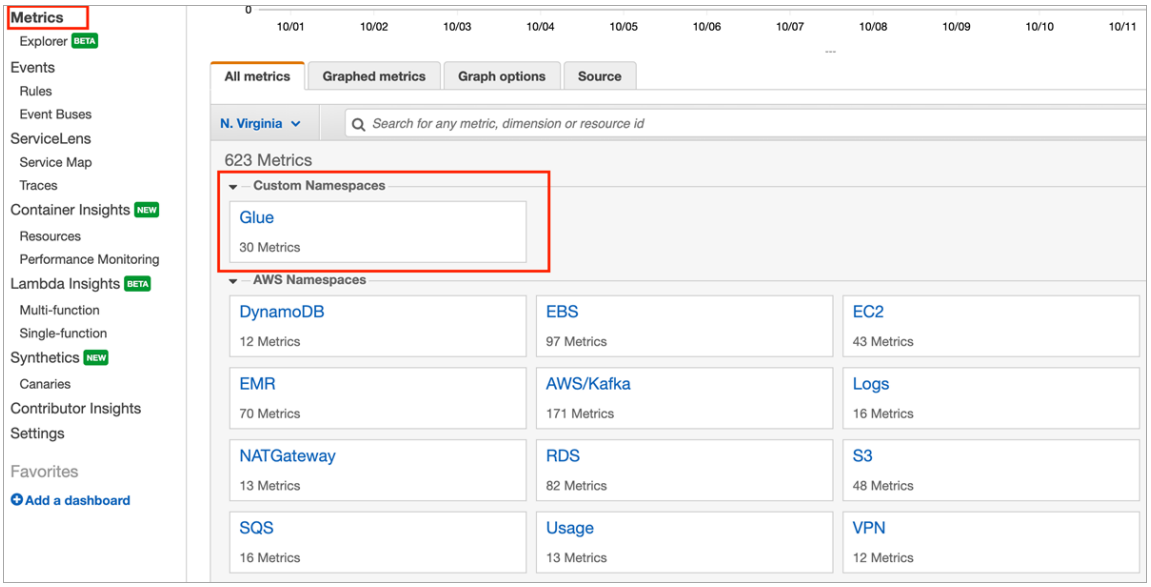


Monitoring options

In addition to the default logs, AWS Glue provides the following options for more advanced level monitoring. Check that the options are appropriate for your use case during or after creating each job. Note that Monitoring options are not available for Python shell jobs.

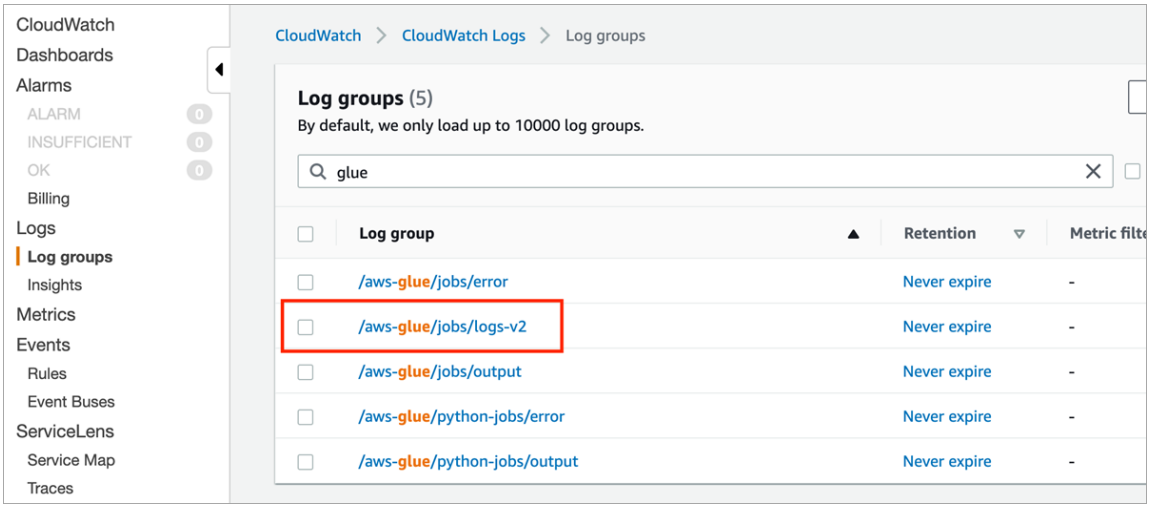


- **Job metrics** – After you turn on **Job metrics**, metrics are reported every 30 seconds to the `Glue` namespace.



For an overview of the metrics that are sent to CloudWatch, see AWS Glue CloudWatch metrics.

- **Continuous logging** – AWS Glue provides real-time, continuous logging for AWS Glue jobs. You can view real-time Apache Spark job logs in CloudWatch, including driver logs, executor logs, and an Apache Spark job progress bar. After you turn on **Continuous logging**, logs appear under the **/aws-glue/jobs/logs-v2** log group.



- **Spark UI** – When your AWS Glue ETL job runs using Spark, you can store Spark UI logs. You can then deploy the Spark history server to view them on an Amazon Elastic Compute Cloud (Amazon EC2) instance. You could also view them locally using Docker. For more information, see Launching the Spark history server.

Automation

AWS Glue provides two ways for you to automate ETL jobs: triggers and workflows.

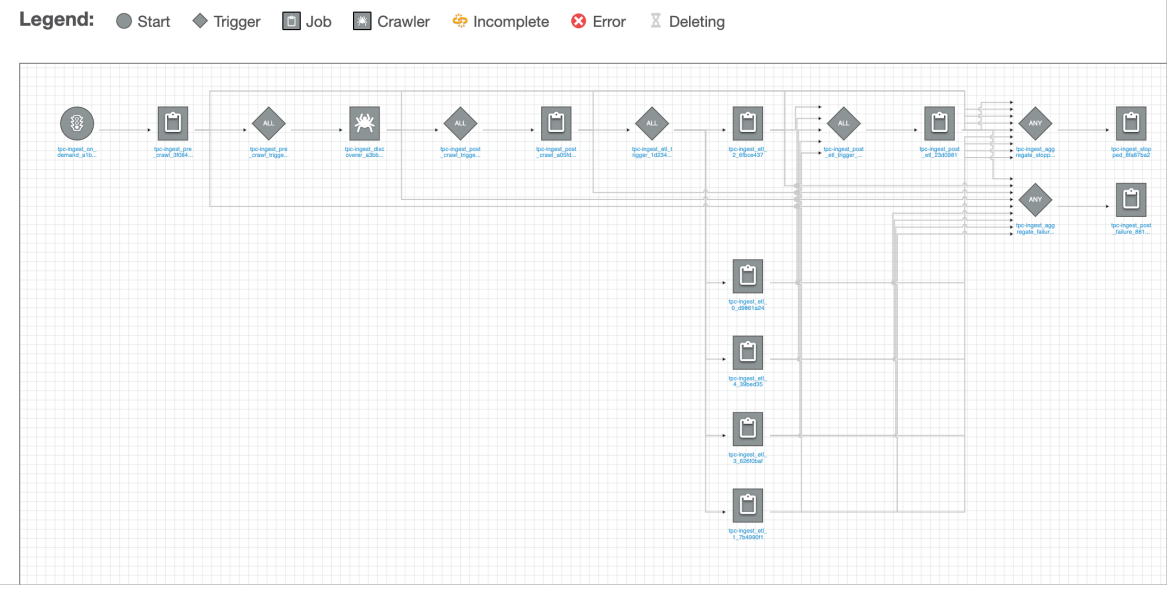
AWS Glue triggers

You can design a basic chain of dependent jobs and crawlers based on a specific schedule or condition by using AWS Glue triggers. For more information, see AWS Glue triggers.

AWS Glue workflows

For more complex workloads, you can use workflows to create and visualize complex ETL activities that involve multiple crawlers, jobs, and triggers. Each workflow manages running and monitoring all its components.

The following diagram shows a sample AWS Glue workflow.



Integration with other AWS services

AWS Glue also integrates with other AWS services, such as Lambda and AWS Step Functions, for more automation options.

Job bookmarks

To avoid reprocessing data when a scheduled ETL job runs, AWS Glue uses job bookmarks. AWS Glue tracks data that has already been processed during a previous run of an ETL job by persisting state information from the job run. The persisted state information is the job bookmark. Job bookmarks help AWS Glue maintain state information and prevent the reprocessing of old data. For more information, see Tracking processed data using job bookmarks.

AWS Glue DataBrew

AWS Glue DataBrew differs from AWS Glue ETL in that you don't have write code to work with it. DataBrew is available in a separate console view from AWS Glue. It works with the following services:

- AWS Data Exchange
- AWS Glue Data Catalog
- AWS Lake Formation
- Amazon RDS
- Amazon Redshift
- Amazon S3

DataBrew is based on the following six core concepts:

Project

The entire data preparation workspace in DataBrew

Dataset

A set of data

Recipe

A set of instructions containing many steps; each step can contain many actions

Job

A set of instructions to run a recipe or a data profile job

Data lineage

The tracking of data in a visual interface to identify its origin

Data profile

A summary view of the shape of your data

To get started with DataBrew, use the AWS Glue DataBrew sample project tutorial.

Best practices

Following are some of the best practices, from beginner level to advanced level, from the AWS Glue documentation and related blog posts.

Run locally first

To save on cost and time while building your ETL jobs, start the development locally to test your code and business logic. This is a best practice even if you are experienced. For instructions on setting up a Docker container that can help write AWS Glue ETL jobs both in a shell and in an integrated development environment (IDE), see the blog post Developing AWS Glue ETL jobs locally using a container.

Use a development endpoint

You can use a development endpoint with AWS Glue ETL for PySpark and Scala. By using the development endpoint, you can test your code on the real dataset. Use a sample size from your original dataset. To get started, follow the steps in Adding a development endpoint.

Use partitioning to query exactly what you need

You can use partitioning to filter the data at loading time instead loading the entire dataset. The blog post Work with partitioned data in AWS Glue illustrates partitioning data and the predicate pushdown feature of AWS Glue.

Optimize memory management

One of the most crucial factors in writing AWS Glue jobs is memory management because AWS Glue uses Apache Spark for its Spark-related ETL jobs. For more information, see the blog post Optimize memory management in AWS Glue, which talks about how you can use the following:

- Amazon S3 list implementation of AWS Glue
- Grouping
- Excluding Amazon S3 paths that are not required for the job
- Filtering on Amazon S3 storage class
- Spark read partitioning
- AWS Glue read partitioning
- Bulk inserts
- JDBC optimizations
- Join optimizations
- PySpark user-defined functions (UDFs)
- Incremental processing

Use the appropriate type of scaling

Understanding how to scale is crucial in writing performant ETL jobs. Compute-intensive AWS Glue jobs that possess a high degree of data parallelism can benefit from horizontal scaling (more standard or G1.X workers). ETL jobs that need high memory or ample disk space to store intermediate shuffle output can benefit from vertical scaling (more G1.X or G2.X workers). The blog post Best practices to scale Apache Spark jobs and partition data with AWS Glue talks about worker types and scaling in AWS Glue.

Serverless ETL on AWS GlueFAQ

This section provides answers to commonly raised questions about serverless ETL on AWS Glue.

When should I use AWS Glue Python shell instead of AWS Glue with Spark?

Use AWS Glue Python shell when you do not need too much of a compute power to run light ETL workloads. Use AWS Glue with Spark when you must scale either horizontally, vertically, or both.

What is the difference between AWS Glue version 1.0 and AWS Glue version 2.0?

The major improvement of version 2.0 is the reduced startup time for Spark-related jobs. More feature-related improvements are mentioned in the AWS documentation.

Next steps

Understanding AWS Glue transformations

For more efficient data processing, AWS Glue includes built-in transformation functions. The functions pass from transform to transform in a data structure called a DynamicFrame, which is an extension to an Apache Spark SQL DataFrame. A DynamicFrame is similar to a DataFrame, except that each record is self-describing, so no schema is required initially.

To get acquainted with several AWS Glue PySpark built-in functions, read the blog post Building an AWS Glue ETL pipeline locally without an AWS account.

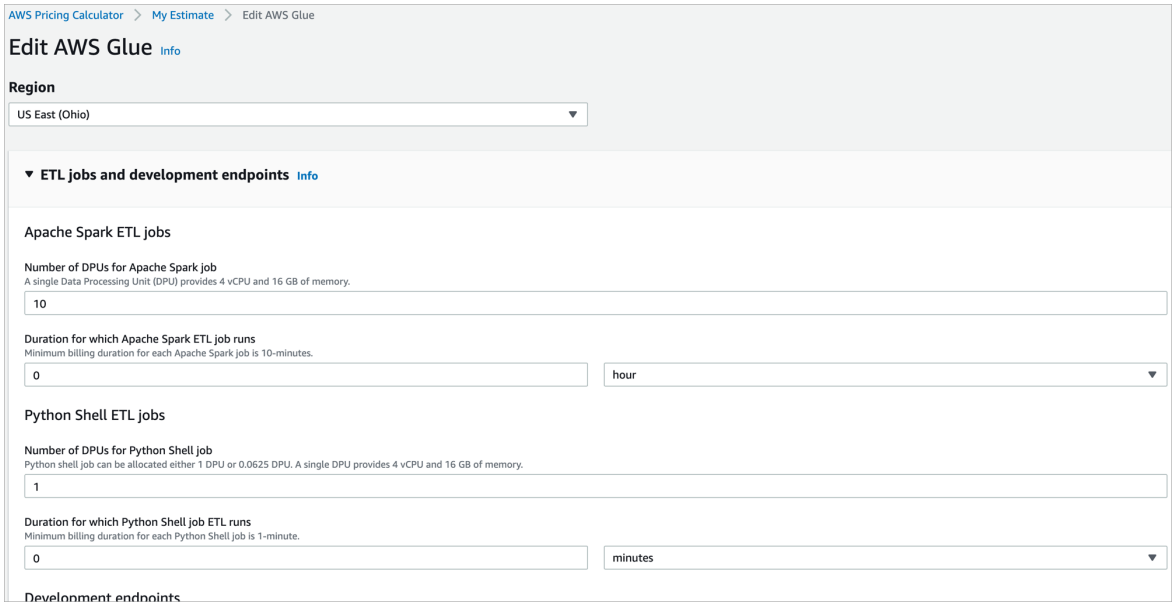
Authoring your first ETL job

If you haven't written an ETL job before, you can get started by using the Three AWS Glue ETL job types for converting data to Apache Parquet pattern.

If you have experience writing ETL jobs, you can use the AWS Glue GitHub samples to explore more deeply.

Pricing

For pricing information, see AWS Glue pricing. You can also use the pricing calculator to estimate your monthly cost for using different AWS Glue components.



Additional resources

References

- Load ongoing data lake changes with AWS DMS and AWS Glue
- Connect to and run ETL jobs across multiple VPCs using a dedicated AWS Glue VPC
- Creating a source to Lakehouse data replication pipe using Apache Hudi, AWS Glue, AWS DMS, and Amazon Redshift

Patterns

- Build an ETL service pipeline to load data incrementally from Amazon S3 to Amazon Redshift using AWS Glue
- Load data from Amazon S3 to Amazon Redshift using AWS Glue
- Deploy and manage a serverless data lake on the AWS Cloud by using infrastructure as code

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

update-history-change	update-history-description	update-history-date
Added a best practice (p. 15)	We added information about using partitioning to query for specific data.	February 4, 2021
Initial publication (p. 15)	—	January 29, 2021