AWS Whitepaper

AWS Glue Best Practices: Building an Operationally Efficient Data Pipeline

AWS Glue Best Practices: Building an Operationally Efficient Data Pipeline: AWS Whitepaper

Table of Contents

AWS Glue Best Practices: Building an Operationally Efficient Data Pipeline

Publication date: **August 26, 2022** (*Document revisions*)

Abstract

Data integration is a critical element in building a data lake and a data warehouse. Data integration enables data from different sources to be cleaned, harmonized, transformed, and finally loaded. In the process of building a data warehouse, most of the development efforts are required for building a data integration pipeline. Data integration is one of the most critical elements in data analytics ecosystems. An efficient and well-designed data integration pipeline is critical for making the data available and trusted amongst the analytics consumers.

This whitepaper shows you some of the considerations and best practices for building and efficiently operating your data pipeline with AWS Glue.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems in the cloud. The six pillars of the Framework allow you to learn architectural best practices for designing and operating reliable, secure, efficient, cost-effective, and sustainable systems. Using the AWS Well-Architected Tool, available at no charge in the AWS Management Console, you can review your workloads against these best practices by answering a set of questions for each pillar.

For more expert guidance and best practices for your cloud architecture—reference architecture deployments, diagrams, and whitepapers—refer to the AWS Architecture Center.

Introduction

Data volumes and complexities are increasing at an unprecedented rate, exploding from terabytes to petabytes or even exabytes of data. Traditional on-premises based approaches for bundling a data pipeline do not work well with a cloud-based strategy, and most of the time, do not provide the elasticity and cost effectiveness of cloud native approaches.

AWS hears from customers that they want to extract more value from their data, but struggle to capture, store, and analyze all the data generated by today's modern and digital businesses.

Data is growing exponentially, coming from new sources. It is increasingly diverse, and needs to be securely accessed and analyzed by any number of applications and people.

With changing data and business needs, the focus on building a high performing, cost effective, and low maintenance data pipeline is paramount. Introduced in 2017, AWS Glue is a fully managed, serverless data integration service that allows customers to scale based on their workload, with no infrastructures to manage.

The next section discusses common best practices for building and efficiently operating your data pipeline with AWS Glue. This document is intended for advanced users, data engineers and architects.

To get the most out of this whitepaper, it's helpful to be familiar with AWS Glue, AWS Glue DataBrew, Amazon Simple Storage Service (Amazon S3), AWS Lambda, and AWS Step Functions.

- Refer to AWS Glue Best Practices: Building a Secure and Reliable Data Pipeline for best practices around security and reliability for your data pipelines with AWS Glue.
- Refer to AWS Glue Best Practices: Building a Performant and Cost Optimized Data Pipeline for best practices around performance efficiency and cost optimization for your data pipelines with AWS Glue.

AWS Glue product family

AWS Glue is a serverless, fully managed data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning (ML), and application development. You simply point AWS Glue to your data stored on AWS, and AWS Glue discovers your data and stores the associated metadata (such as table definition and schema) in the AWS Glue Data Catalog.

Once cataloged, your data is immediately searchable, queryable, and available for extract, transform, load (ETL). One of the most difficult tasks in building a data pipeline is to integrate data from various sources which could be structured, semi-structured, or even un-structured; and that is where AWS Glue shines. AWS Glue provides both visual and code-based interfaces to help build ETL jobs and data pipelines faster.

The AWS Glue product family includes several services that cater to varying user personas and allows them to catalog, transform, clean, enrich, and deliver data in a consistent and reliable way.

The AWS Glue product family consists of AWS Glue for cataloging and ETL transformation, and AWS Glue DataBrew for self-service, no-code data preparation.

The AWS Glue product family

When should I use AWS Glue?

AWS Glue provides both visual and code-based interfaces to make data integration easier. Users can easily find and access data using the AWS Glue Data Catalog. Data engineers and ETL developers can visually create, run, and monitor ETL workflows with a few clicks in AWS Glue Studio.

You can use AWS Glue to organize, cleanse, validate, and format data for storage in a data warehouse or data lake. You can transform and move AWS Cloud data into your data store. You can also load data from disparate static or streaming data sources into your data warehouse or data lake for regular reporting and analysis. By storing data in a data warehouse or data lake, you integrate information from different parts of your business and provide a common source of data for decision making.

AWS Glue simplifies many tasks when you are building a data warehouse or data lake:

- Discovers and catalogs metadata about your data stores into a central catalog. You can process semi-structured data, such as clickstream or process logs.
- Populates the AWS Glue Data Catalog with table definitions from scheduled crawler programs. Crawlers call classifier logic to infer the schema, format, and data types of your data. This metadata is stored as tables in the AWS Glue Data Catalog, and used in the authoring process of your ETL jobs.
- Generates ETL scripts to transform, flatten, and enrich your data from source to target.
- Detects schema changes and adapts based on your preferences.
- Triggers your ETL jobs based on a schedule or event. You can initiate jobs automatically to move your data into your data warehouse or data lake. Triggers can be used to create a dependency flow between jobs.
- Gathers runtime metrics to monitor the activities of your data warehouse or data lake.
- Handles errors and retries automatically.
- Scales resources, as needed, to run your jobs.

You can use AWS Glue when you run serverless queries against your Amazon S3 data lake. AWS Glue can catalog your Amazon S3 data, making it available for querying with Amazon Athena and Amazon Redshift Spectrum. With crawlers, your metadata stays in sync with the underlying data. Athena and Redshift Spectrum can directly query your S3 data lake using the AWS Glue

Data Catalog. With AWS Glue, you access and analyze data through one unified interface without loading it into multiple data silos.

You can create event-driven ETL pipelines with AWS Glue. You can run your ETL jobs as soon as new data becomes available in S3 by invoking your AWS Glue ETL jobs from an AWS Lambda function or event driven workflows. You can also register this new dataset in the AWS Glue Data Catalog as part of your ETL jobs.

You can use AWS Glue to understand your data assets. You can store your data using various AWS services and still maintain a unified view of your data using the AWS Glue Data Catalog. View the Data Catalog to quickly search and discover the datasets that you own, and maintain the relevant metadata in one central repository. The Data Catalog also serves as a drop-in replacement for your external Apache Hive Metastore.

What is AWS Glue Studio?

AWS Glue Studio is a new graphical interface that makes it easy to create, run, and monitor ETL jobs in AWS Glue. You can visually compose data transformation workflows, and seamlessly run them on the AWS Glue Apache Spark-based serverless ETL engine. You can inspect the schema and data results in each step of the job. **Use AWS Glue Studio for a simple visual interface to create ETL workflows for data cleaning and transformation, and run them on AWS Glue.** AWS Glue Studio makes it easy for ETL developers to create repeatable processes to move and transform large-scale, semi-structured datasets, and load them into data lakes and data warehouses. It provides a boxes-and-arrows style visual interface for developing and managing AWS Glue ETL workflows that you can optionally customize with code. AWS Glue Studio combines the ease of use of traditional ETL tools, and the power and flexibility of the big AWS Glue data processing engine.

AWS Glue Studio provides multiple ways to customize your ETL scripts, including adding nodes that represent code snippets in the visual editor.

Use AWS Glue Studio for easier job management. AWS Glue Studio provides you with job and job run management interfaces that make it clear how jobs relate to each other, and give an overall picture of your job runs. The job management page makes it easy to do bulk operations on jobs (previously difficult to do in the AWS Glue console). All job runs are available in a single interface where you can search and filter. This gives you a constantly updated view of your ETL operations and the resources you use. You can use the near real-time dashboard in AWS Glue Studio to monitor your job runs and validate that they are operating as intended.

When should I use AWS Glue DataBrew?

Data analysts and **data scientists** can use AWS Glue DataBrew to visually enrich, clean, and normalize data without writing code. Using DataBrew helps reduce the time it takes to prepare data for analytics and ML by up to 80 percent, compared to custom developed data preparation. You can choose from over 250 ready-made transformations to automate data preparation tasks, such as filtering anomalies, converting data to standard formats, and correcting invalid values.

Use AWS Glue DataBrew to interactively discover, visualize, clean, and transform raw data. With the intuitive DataBrew interface, you can interactively discover, visualize, clean, and transform raw data. DataBrew makes smart suggestions to help you identify data quality issues that can be difficult to find and time-consuming to fix. With DataBrew preparing your data, you can use your time to act on the results and iterate more quickly. You can save transformation as steps in a recipe, which you can update or reuse later with other datasets, and deploy on a continuing basis.

Challenges in building a data pipeline

Building a well-architected and high performing data pipeline requires upfront planning and design of multiple aspects of data storage, including data structure, schema design, schema change handling, storage optimization, and quick scaling to meet the unexpected increase in application data volume and so on. This often requires an ETL mechanism that is designed to orchestrate the transformation of data in multiple steps. You also need to ensure that the ingested data is validated for the data quality or data loss, and monitored for job failures and data exceptions that are not handled with ETL job design.

Here are some common challenges a data engineer typically faces:

- Increase in data volume for processing
- Change of structure of source data
- Poor data quality
- Poor data integrity in source data
- Duplicate data
- Timeliness of source data files
- Lack of available developer interface for testing

Benefits of using AWS Glue for data integration

AWS Glue is a fully managed ETL service that makes it easy for customers to prepare and load their data for analytics. You can create and run an ETL job with a few clicks in the AWS Management Console. You simply point AWS Glue to your data stored on AWS, and AWS Glue discovers your data and stores the associated metadata (such as table definition and schema) in the AWS Glue Data Catalog. Once cataloged, your data is immediately searchable, queryable, and available for ETL.

Following are some benefits of using AWS Glue:

- **Less hassle** — AWS Glue is integrated across a wide range of AWS services, meaning less hassle for you when onboarding. AWS Glue natively supports data stored in Amazon Aurora and all other Amazon Relational Database Service (RDS) engines, Amazon Redshift, and Amazon S3, as well as common database engines and databases in your Virtual Private Cloud (Amazon VPC) running on Amazon Elastic Compute Cloud (Amazon EC2) or your on-premises environment.
- **Cost effective** — AWS Glue is serverless. Because there is no infrastructure to provision or manage, total cost of ownership is lower. AWS Glue handles provisioning, configuration, and scaling of the resources required to run your ETL jobs on a fully managed, scale-out Apache Spark environment. You pay only for the resources used while your jobs are running.
- **More power** — AWS Glue automates much of the effort in building, maintaining, and running ETL jobs. AWS Glue crawls your data sources, identifies data formats, and suggests schemas and transformations. AWS Glue automatically generates the code to run your data transformations and loading processes.

AWS Glue also brings number of important features that provide numerous benefits to your enterprise.

- **Discover and search across all your AWS datasets** — The AWS Glue Data Catalog is your persistent metadata store for all your data assets, regardless of where the data assets are located. The Data Catalog contains table definitions, job definitions, schemas, and other control information to help you manage your AWS Glue environment. It automatically computes statistics, and registers partitions to make queries against your data efficient and cost-effective. It also maintains a comprehensive schema version history so you can understand how your data has changed over time.

- **Automatic schema discovery** — AWS Glue crawlers connect to your source or target data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata in your AWS Glue Data Catalog. The metadata is stored in tables in your data catalog and used in the authoring process of your ETL jobs. You can run crawlers on a schedule, on-demand, or trigger them based on an event to ensure that your metadata is up-to-date.
- **Manage and enforce schemas for data streams** — AWS Glue Schema Registry, a feature of AWS Glue, enables you to validate and control the evolution of streaming data using registered Apache Avro schemas, at no additional charge. Through Apache-licensed serializers and de-serializers, the Schema Registry integrates with Java applications developed for Apache Kafka, Amazon Managed Streaming for Apache Kafka (Amazon MSK), Amazon Kinesis Data Streams, Apache Flink, Amazon Managed Service for Apache Flink for Apache Flink, and AWS Lambda. When data streaming applications are integrated with the Schema Registry, you can improve data quality and safeguard against unexpected changes using compatibility checks that govern schema evolution. Additionally, you can create or update AWS Glue tables and partitions using schemas stored within the registry.
- **Visually transform data with a drag-and-drop interface** — AWS Glue Studio allows you to author highly scalable ETL jobs for distributed processing without becoming an Apache Spark expert. Define your ETL process in the drag-and-drop job editor, and AWS Glue automatically generates the code to extract, transform, and load your data. The code is generated in Scala or Python and written for Apache Spark.
- **Build complex ETL pipelines with simple job scheduling** — AWS Glue jobs can be invoked on a schedule, on-demand, or based on an event. You can start multiple jobs in parallel or specify dependencies across jobs to build complex ETL pipelines. AWS Glue handles all inter-job dependencies, filters bad data, and retries jobs if they fail. All logs and notifications are pushed to Amazon CloudWatch so you can monitor and get alerts from a central service.
- **Clean and transform streaming data in transit** — Serverless streaming ETL jobs in AWS Glue continuously consume data from streaming sources, including Amazon Kinesis and Amazon MSK, clean and transform it in transit, and make it available for analysis in seconds in your target data store. Use this feature to process event data like Internet of Things (IoT) event streams, clickstreams, and network logs. AWS Glue streaming ETL jobs can enrich and aggregate data, join batch and streaming sources, and run a variety of complex analytics and ML operations.
- **Deduplicate and cleanse data with built-in ML** — AWS Glue helps clean and prepare your data for analysis without becoming an ML expert. Its FindMatches feature deduplicates and finds records that are imperfect matches of each other. For example, use FindMatches to find duplicate

records in your database of restaurants, when one record lists “Joe's Pizza” at “121 Main St.” and another shows a “Joseph's Pizzeria” at “121 Main”. FindMatches will ask you to label sets of records as either “matching” or “not matching.” The system will then learn your criteria for calling a pair of records a match, and will build an ETL job that you can use to find duplicate records within a database, or matching records across two databases.

- **Edit, debug, and test ETL code using AWS Glue interactive sessions** — AWS Glue supports interactive application development that assists data engineers to rapidly build, test, and run data preparation and analytics applications. This is achieved using AWS Glue interactive sessions. AWS Glue interactive sessions provide you with on-demand access to a remote Spark runtime environment.

  Flexibility of interactive session lets you interact with it in many ways – the AWS Command Line Interface (AWS CLI), APIs, AWS Glue Studio notebooks, or local Jupyter-compatible notebooks. It provides an open-source Jupyter kernel that integrates almost anywhere that Jupyter does, including integrating with integrated development environments (IDEs) such as PyCharm, IntelliJ, and VS Code. This enables you to author code in your local environment and run it seamlessly on the interactive session backend.

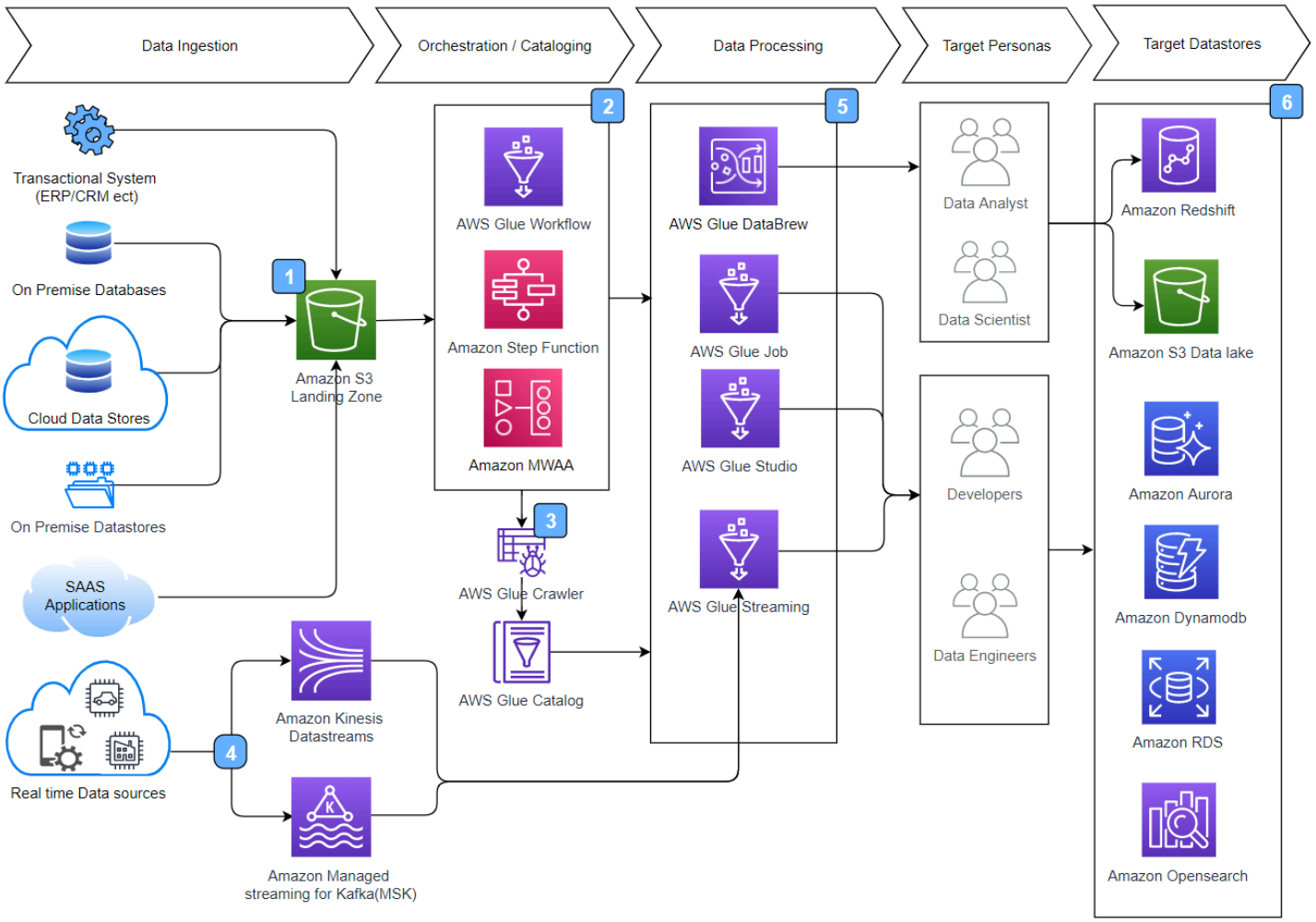
  Interactive sessions provide a faster, cheaper, more-flexible way to build and run data preparation and analytics applications.
- **Normalize data without code using a visual interface** — AWS Glue DataBrew provides an interactive, point-and-click visual interface for users such as data analysts and data scientists to clean and normalize data without writing code. You can easily visualize, clean, and normalize data directly from your data lake, data warehouses, and databases, including Amazon S3, Amazon Redshift, Amazon Aurora, and Amazon RDS. You can choose from over 250 built-in transformations to combine, pivot, and transpose the data, and automate data preparation tasks by applying saved transformations directly to the new incoming data.

Reference architecture with the AWS Glue product family

While working with customers, AWS encountered several different architecture patterns in which the customer used services from the AWS Glue product family to build their data pipelines. Following are some of the common architecture patterns based on user personas.

Building a data pipeline

Here is a reference architecture for building a data pipeline with AWS Glue product family.



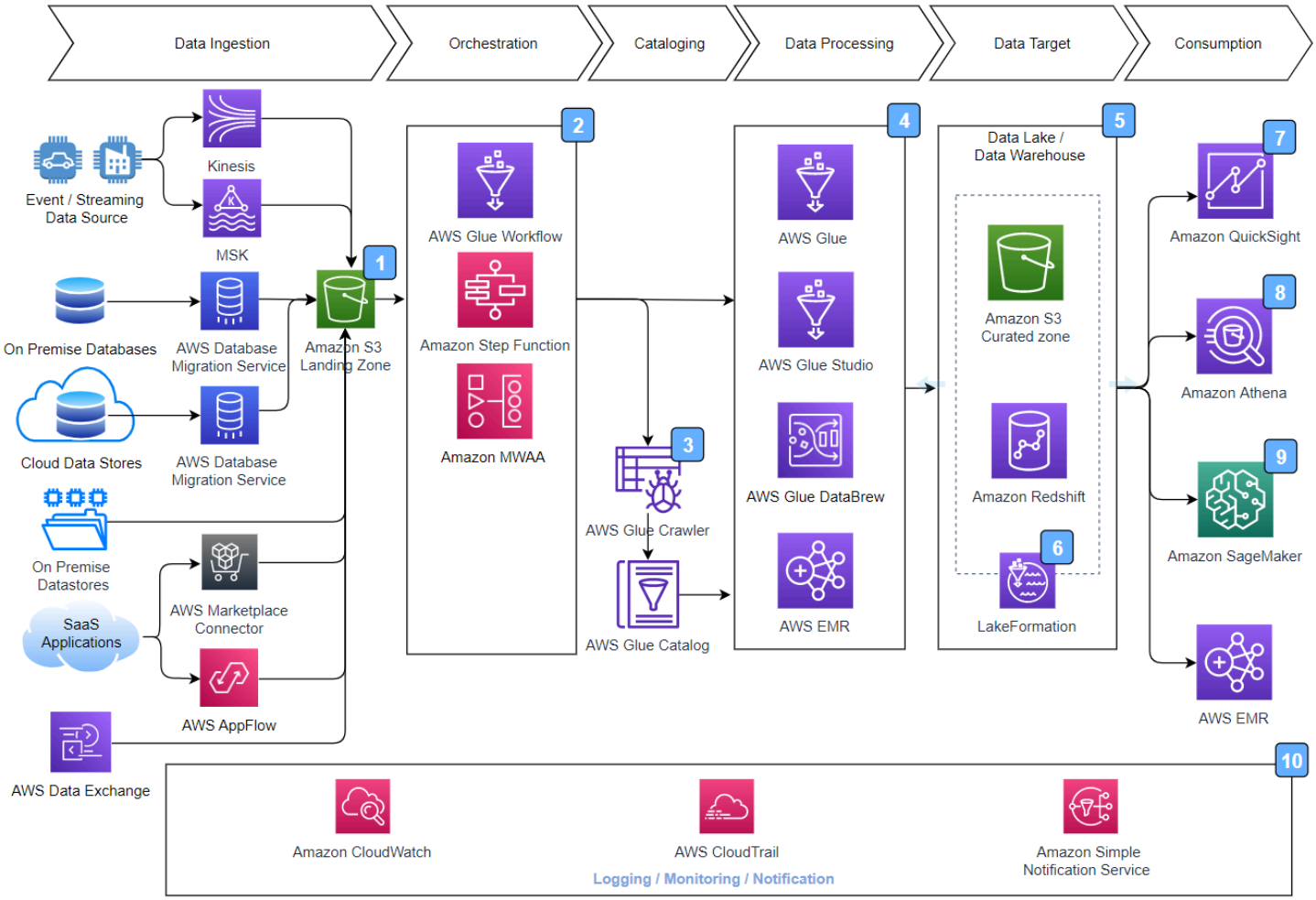
Reference architecture for data pipeline with user personas

The steps the data takes in the architecture shown in the preceding figure are as follows:

1. **Data ingestion** — Data is extracted from various data sources, including transactional data sources such as customer relationship management/enterprise resource planning (CRM/ERP), on-premises databases such as Oracle and SQL Server, on-premises data stores, Sales as a Service (SaaS) applications such as Salesforce, SAP Concur, and so on for further processing.
2. **Job orchestration** — As a new file uploaded into an S3 landing zone or a time-based schedule is triggered, an orchestration workflow is trigger using AWS Step Functions, Amazon Managed Workflow for Apache Airflow (MWAA), or AWS Glue workflow. Depending on the business requirement, workflows are also triggered using a predefined and time-based schedule to process file at certain intervals.
3. **Data cataloging (optional)** — The job orchestrator triggers an AWS Glue workflow to crawl the location of the file and build or update AWS Glue Data Catalog. The AWS Glue Data Catalog contains references to data that is used as sources and targets of your ETL jobs in AWS Glue.
4. **Data streaming** — To process streaming data in near real time, customers commonly ingest the data into Amazon Kinesis Data Streams or Amazon MSK. This data can then be consumed by an AWS Glue streaming ETL application for further processing.
5. **Data processing** — Processes and transforms data and data format, data quality and integrity checks, deduplications, transformation, and so on.
6. **Data loading** — Loads data after processing and transforming to data targets, including data lakes such as S3 locations, relational targets such as Amazon Redshift, Amazon RDS, Amazon Aurora, Amazon OpenSearch Service, or Amazon DynamoDB.

Building a data lake

Following is a reference architecture for building a data lake with the AWS Glue product family:



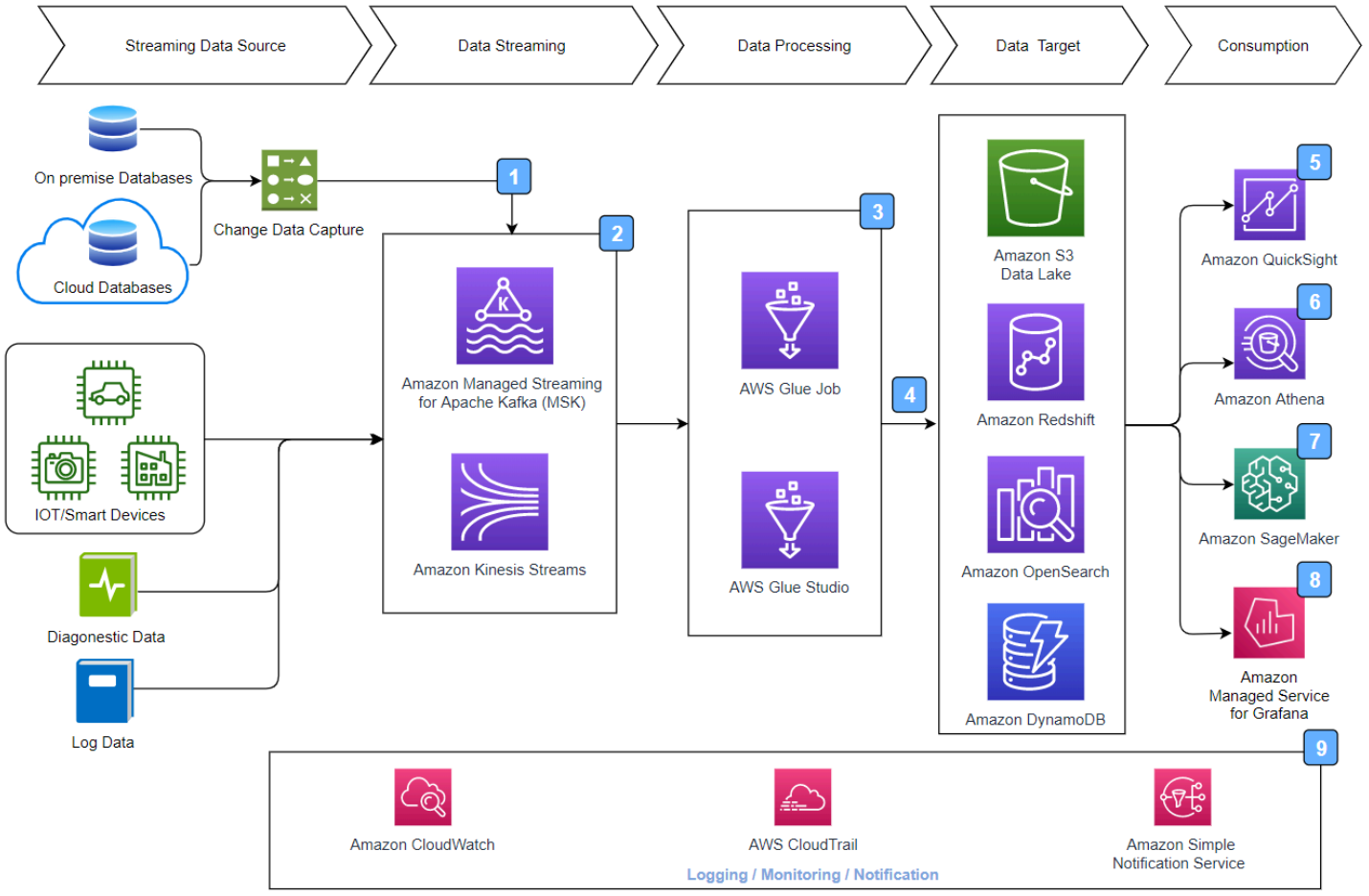
Reference architecture for building a data lake

1. **Data ingestion** — Data is extracted from various data sources like transactional data sources such as CRM/ERP, on-premises databases such as Oracle and SQL Servers, on-premises data stores, SaaS applications such as Salesforce, SAP Concur, and so on into a landing zone in S3 for further processing. In this step commonly, Amazon AppFlow is used for SaaS applications, AWS Database Migration Service (AWS DMS) is used for ingesting data from on-premises and cloud databases such as AWS Data Exchange and is used for integrating third-party data into the data lake.
2. **Job orchestration** — As a new file is uploaded into an S3 landing zone, a Lambda function or event driven AWS Glue workflow triggers orchestration workflow using AWS Step Functions, MWAA, or AWS Glue workflow. Depending on the business requirements, workflows are also triggered using a predefined and time-based schedule to process file at certain intervals.

3. **Data cataloging (optional)** — The job orchestrator triggers an AWS Glue workflow to crawl the location of the file and build or update the AWS Glue Data Catalog. The AWS Glue Data Catalog contains references to data that is used as sources and targets for your ETL jobs in AWS Glue.
4. **Data processing** — The data is processes and transforms data that is transforming data and improving data quality, performing integrity checks, and so on.
5. **Data loading** — In this step, the processed and transformed data is loaded into data into an S3-based curated zone with appropriate partitions and data format, which is used as a data lake layer.
6. **Unified governance** — AWS Lake Formation is commonly used for implementation of unified governance on a data lake. Additionally, if you are looking for transactional capability and small file compaction with AWS Lake Formation, governed tables can also be considered.
7. **Amazon QuickSight** — Amazon QuickSight allows everyone in your organization to understand your data by asking questions in their natural language, exploring through interactive dashboards, or automatically looking for patterns and outliers powered by ML.
8. **Amazon Athena** — Amazon Athena provides capability for ad hoc querying capability on the data stored in the data lake.
9. **Amazon SageMaker** — Amazon SageMaker and AWS AI services can be used to build, train, and deploy ML models, and add intelligence to your applications.
10. **Logging, monitoring and notification** — Amazon CloudWatch can be used for monitoring, Amazon Simple Notification Service (Amazon SNS) can be used for notification, and AWS CloudTrail can be used for logging of events.

Building a streaming data pipeline

Here is a reference architecture for building a streaming data pipeline with the AWS Glue product family.



Reference architecture for streaming data pipeline

1. **Data Source** — In the previous architecture, there are multiple data sources. Near real-time data is generated through streaming data sources such as IoT devices, log and diagnostic data from application servers, and change data capture (CDC) from transactional data stores.
2. **Data steaming** — Messages and events are streamed into streaming services such as Amazon Kinesis Data Streams or Amazon MSK.
3. **Stream data processing** — In this step, you can create streaming ETL jobs that run continuously and consume data from streaming sources such as Amazon Kinesis Data Streams and Amazon MSK. The jobs cleanse and transform the data.
4. **Stream data loading** — The processed data is typically loaded into S3 data lakes or joint database connectivity (JDBC) data stores such as Amazon Redshift or NoSQL data sources such as Amazon DynamoDB or Amazon OpenSearch Service. After the data is loaded, the data can be consumed using services such as Amazon QuickSight, Amazon Athena, Amazon SageMaker, Amazon Managed Grafana, and so on.

5. **Amazon QuickSight:** — Amazon QuickSight allows everyone in your organization to understand your data by asking questions in their native language, exploring through interactive dashboards, or automatically looking for patterns and outliers powered by ML.
6. **Amazon Athena** — Amazon Athena provides capability for ad hoc querying capability on the data stored in the data lake.
7. **Amazon SageMaker** — Amazon SageMaker and AWS AI services can be used to build, train, and deploy ML models, and add intelligence to your applications.
8. **Amazon Managed Grafana** — Amazon Managed Grafana is an open-source analytics platform that can be used to query, visualize, alert on, and understand metrics, no matter where they are stored.
9. **Logging, monitoring, and notification** — Amazon CloudWatch can be used for monitoring, Amazon SNS can be used for notification, and AWS CloudTrail can be used for event logging.

Using the AWS Well-Architected Framework for building a data pipeline

Building a well-architected data pipeline is critical for the success of a data engineering project. When designing a well-architected data pipeline, use the guidelines of the AWS Well-Architected Framework. This helps you understand the pros and cons of decisions you make while building applications on AWS.

The Well-Architected Framework guides the architecture considerations in operating reliable, secure, efficient, and cost-effective systems in the cloud. It provides a way for you to consistently measure your architectures against best practices, and identify areas for improvement. AWS believes that having a well-architected data pipeline using the AWS Well-Architected pillars greatly increases the likelihood of success. The AWS Well-Architected Framework is based on six pillars:

- **Operational Excellence** — The Operational Excellence pillar includes the ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value. You can find prescriptive guidance on implementation in the *Operational Excellence Pillar* whitepaper.
- **Security** — The Security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. You can find prescriptive guidance on implementation in the *Security Pillar* whitepaper.
- **Reliability** — The Reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This includes the ability to operate and test the workload through its total lifecycle. You can find prescriptive guidance on implementation in the *Reliability Pillar* whitepaper.
- **Performance Efficiency** —The Performance Efficiency pillar includes the ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve. You can find prescriptive guidance on implementation in the *Performance Efficiency Pillar* whitepaper.
- **Cost Optimization** — The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point. You can find prescriptive guidance on implementation in the *Cost Optimization Pillar* whitepaper.
- **Sustainability** — The Sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct

action to reduce resource usage. You can find prescriptive guidance on implementation in the *Sustainability Pillar* whitepaper.

For best practices around Security and Reliability for your data pipelines, refer to AWS Glue Best Practices: Building a Secure and Reliable Data Pipeline.

For best practices around Performance Efficiency and Cost Optimization for your data pipelines, refer to AWS Glue Best Practices: Building a Performant and Cost Optimized Data Pipeline.

Building an operationally excellent data pipeline

The Operational Excellence pillar includes the ability to support development and run data pipelines effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value. Here are some considerations to review when designing data pipelines using the guidelines of the Operational Excellence pillar of the AWS Well-Architected Framework.

Using AWS Glue blueprints

Blueprints in AWS Glue provide organizations with a mechanism to develop, share and reuse complex ETL workflows. They assist ETL developers to generate and publish templates of commonly asked ETL workflows that analysts or other non-developers can consume and run jobs without having to write code.

A blueprint in AWS Glue is a zip archive that consists of the following files:

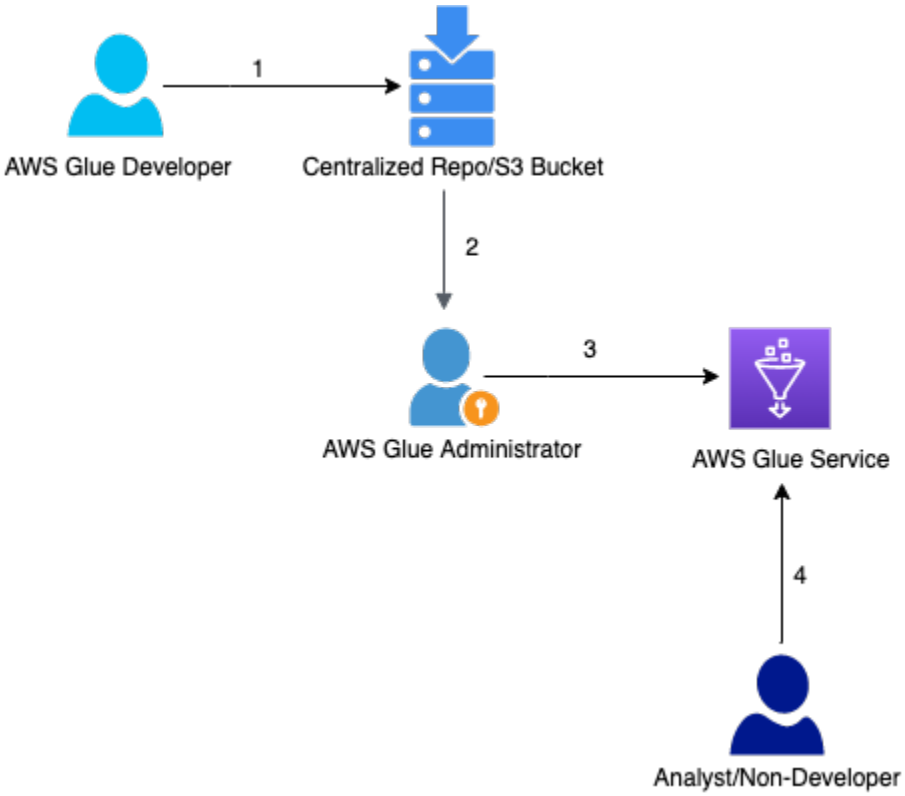
- **A blueprint configuration file** — This file describes the name and data type of all the blueprint parameters. Parameter names could be IAM roles, input/output (I/O) locations, workflow names, and so on. The configuration file also contains a reference to the workflow layout generator function definition.
- **A layout script** — This consists of the implementation of the layout generator. The script performs the prerequisite tasks such as creating the catalog tables, temporary paths, and so on. It also creates the AWS Glue workflow, which may be a collection of jobs, triggers, and crawlers.
- **Scripts (optional)** – These are supporting scripts or AWS Glue ETL jobs that constitute the workflow.
- **Supporting documents (optional)** — Readme files or any documents supporting the blueprints.

Blueprints can be parametrized, which means that a single blueprint can be used to solve multiple use cases. A few use cases are:

- Compacting small files on S3
- Partitioning datasets
- Creating a database or table snapshot
- Converting data from one file format to another

Blueprint lifecycle

The following diagram summarizes the lifecycle of an AWS Glue blueprint.



Lifecycle of an AWS Glue blueprint

1. An AWS Glue developer develops and test the workflows, and bundles them into a blueprint structure. They publish it to the organization's central repository. This repository could be a file system, S3, or a code repository such as GitHub.
2. The AWS Glue admin user exports these blueprints from the centralized store. The central store may be a private repository within the organization, or it can be a public repository that hosts blueprints authored by a community of developers.
3. The administrator then registers the blueprint with the AWS Glue services and provides the analysts or other users the necessary access.
4. The analyst or any users can now use the blueprint, configure it to their business needs, create and run workflow, and consume results without having to write any code.

Orchestrating AWS Glue jobs

You have several mechanisms to orchestrate and automate your AWS Glue jobs using AWS native and managed orchestration services such as:

- AWS Glue workflows
- AWS Step Functions
- MWAA

AWS Glue workflows

AWS Glue workflows are a built-in feature of AWS Glue ETL which enables you to create workflows of your jobs and crawlers, and lets you add triggers to begin the orchestration process. AWS Glue provides a graphical user interface (GUI) for building your workflow that is simple to use, and provides core orchestration capability, which is ideal for simple workflows that do not require other AWS services such as SNS. It is simple to set up, and provides core orchestration capability, which is ideal for simple workflows that do not involve any additional services such as SNS.

AWS Step Function

AWS Step Functions is a low-code, serverless visual workflow service used to orchestrate AWS services such as AWS Glue to automate and orchestrate ETL jobs and crawlers, and integrate with additional AWS services such as SNS for notification or AWS Lambda for generation of trigger of workflow for a file upload event into S3. Using AWS Step Functions, you can manage failures, retries, parallelization, service integrations, and observability. Being complete serverless makes Step Functions an ideal choice when you don't want to manage infrastructure for orchestration purposes. An AWS Step Function can be created using both GUI and Amazon States Language, which is a JSON-based language used to describe state machines declaratively. Following are some best practices for using AWS Step Functions:

- **Use timeouts to avoid stuck job runs:** The Amazon States Language doesn't set timeouts in state machine definitions by default. If there is no explicit timeout, Step Functions often relies solely on a response from an activity worker to know that a task is complete. If something goes wrong and `TimeoutSeconds` isn't specified, a job run is stuck waiting for a response from the ETL job which may come much later. To avoid this, specify a reasonable timeout when you create a task in your state machine. Here is an example of timeout in a task:

```
"ActivityState": {
"Type": "Task",
"Resource": "arn:aws:states:us-east-1:123456789012:activity:HelloWorld",
"TimeoutSeconds": 300,
"HeartbeatSeconds": 60,
"Next": "NextState"
}
```

- **Use Amazon S3 ARNs instead of passing large payloads:** Job runs that pass large payloads of data between states can be ended. If the data you are passing between states might grow to over 262,144 bytes, use S3 to store the data, and parse the Amazon Resource Name (ARN) of the bucket in the `Payload` parameter to get the bucket name and key value. Alternatively, adjust your implementation so that you pass smaller payloads in your job runs.
- **Amazon CloudWatch Logs resource policy size restrictions:** CloudWatch Logs resource policies are limited to 5120 characters. When CloudWatch Logs detects that a policy approaches this size limit, it automatically enables log groups that start with `/aws/vendedlogs/`. When you create a state machine with logging enabled, Step Functions must update your CloudWatch Logs resource policy with the log group you specify. To avoid reaching the CloudWatch Logs resource policy size limit, prefix your CloudWatch Logs log group names with `/aws/vendedlogs/`. When you create a log group in the Step Functions console, the log group names are prefixed with `/aws/vendedlogs/states`. For more information, refer to Enabling Logging from Certain AWS Services.

AWS Managed Workflow for Apache Airflow (MWAA)

AWS Managed Workflows for Apache Airflow (MWAA) is a managed orchestration service for Apache Airflow that makes it easier to set up and operate end-to-end data pipelines in the cloud at scale. Apache Airflow is an open-source tool used to programmatically author, schedule, and monitor sequences of processes and tasks referred to as “workflows.” MWAA helps to manage the underlying infrastructure for scalability, availability, and security, so that you can focus more time on developing the workflows rather than managing the operation of the orchestration servers. If your team is already using a workflow that uses Apache Airflow and you are looking for integration of AWS services such as EMR, Amazon Redshift, S3, and so on, then MWAA may be a good choice for your workflow orchestration.

As MWAA is a managed orchestration service of Apache Airflow, many of the best practices of Apache Airflow for developing directed acyclic graphs (DAGs) are also applicable here. Following are some of the considerations for data pipeline orchestration with MWAA:

- **Make DAGs and tasks immutable** — In every workflow run, DAGs should produce similar data. It's a best practice to make read/write to and from a partition idempotent.
- **Grant least privilege required** — In accordance with standard AWS security best practices, grant permissions to only the resources or actions that users need to perform tasks.
- **Monitoring user activity** — Use AWS CloudTrail to monitor user activity in your account.
- Ensure that the S3 bucket policy and object access control lists (ACLs) grant permissions to the users from the associated MWAA environment to put objects in the bucket. This ensures that users with permissions to add workflows to the bucket also have permissions to run the workflows in Airflow.
- Use the S3 buckets associated with Amazon MWAA environments for Amazon MWAA only. It's a best practice not to store other objects in the bucket, or use the bucket with another service.
- For more information on performance tuning best practices, refer to Performance tuning for Apache Airflow on Amazon MWAA.
- For best practices on managing Python dependencies refer to Managing Python dependencies in requirements.txt.

Table 1 — When to use AWS Glue Workflow, AWS Step Functions, or Amazon MWAA

Factor	AWS Glue Workflow	AWS Step Function	Amazon Managed Workflow for Apache Airflow (MWAA)
Use case	Suitable when your pipeline consists of mostly AWS Glue jobs and crawlers.	Suitable when there is a need to integrate with different services, including AWS Lambda, SSM, and so on.	Compatible with open-source Airflow and suitable when you want to reuse existing Airflow assets.
Infrastructure	Serverless	Serverless	Managed service

Factor	AWS Glue Workflow	AWS Step Function	Amazon Managed Workflow for Apache Airflow (MWAA)
User interface and supported language	Simple UI in AWS Glue Console, API/ SDK, CloudFormation, Python (via custom blueprint)	Interactive / rich UI based on Amazon State Language (ASL), JSON, and YAML	Any supported integrated development environment (IDE) for Python
Building a pipeline	Build a data pipeline using an AWS Glue job written in Python or /Scala and crawlers. Possible to integrate with other services using AWS SDK for Python (boto3).	Build a data pipeline using the Step Functions console. Possible to integrate with non-supported services using Lambda.	Workflows are created as DAGs, which are defined within a Python file that defines the DAG's structure as code. You can add or update Apache Airflow DAGs on your Amazon MWAA environment using the DAGs folder in your S3 bucket.
Passing information between tasks/states	Parameters can be shared in the workflow and jobs can refer them.	Parameters can be passed between states.	Global variables are supported.
Resuming failed tasks	Possible to resume failed task.	Possible to resume failed part by defining a new state machine.	Possible to resume failed task.

Factor	AWS Glue Workflow	AWS Step Function	Amazon Managed Workflow for Apache Airflow (MWAA)
Change management of workflow	Not supported. Possible to delete/ re-create workflow using blueprint, or you can modify the jobs and crawlers.	Update DAG and reflect it in API/ Console.	Update DAG and re-deploy DAG.
Cost	No additional cost	Inexpensive with no operational overhead. $0.025 per 1,000 state transitions.	Inexpensive with limited operation al overhead. Detail pricing available on the MWAA Pricing page.

Using parameters

Most applications need to be configured in different ways to function in different environments. For example, there could be cases where the same business logic is submitted at different time of the day, by the same job, or there could be cases where input parameters change across different application environments (dev/test/production and so on) Like any application, AWS Glue jobs should also be designed with this reusability in mind.

AWS Glue jobs support the concept of parameters. These parameters can be used as a part of an independent job, or as a part of an AWS Glue workflow. Being able to pass parameters to the job makes it reusable, and reduces the number of code changes to meet future requirements. Details on how to work with parameters follows:

A utility function, `getResolvedOptions`, within the AWS Glue API lets you access the arguments that are passed to a job. The function is under `awsglue.utils` when working with Python. While using Scala, it is under `com.amazonaws.services.glue.util.GlueArgParser`.

The function prototype follows:

```
getResolvedOptions(args, options)
```

- **args** — This corresponds to the list of arguments, implicitly available in the Scala `args` array or `sys.argv` in Python.
- **options** — this corresponds to the array of argument names or job parameters that you want to retrieve.

Usage

Suppose you pass two parameters, **LOCATION** and **DATE**, to your job. The following section shows you the code snippets to retrieve them both.

Scala

```
import com.amazonaws.services.glue.util.GlueArgParser
...
def main(args: Array[String]) {
...
val jobParams = GlueArgParser.getResolvedOptions(args,
 Seq("JOB_NAME","LOCATION","DATE").toArray)
val location = jobParams("LOCATION")
val date = jobParams("DATE")
...
```

Python

```
import sys
from awsglue.utils import getResolvedOptions
...
args = getResolvedOptions(sys.argv, ['JOB_NAME', 'LOCATION', 'DATE'])
location=args['LOCATION']
date=args['DATE']
...
```

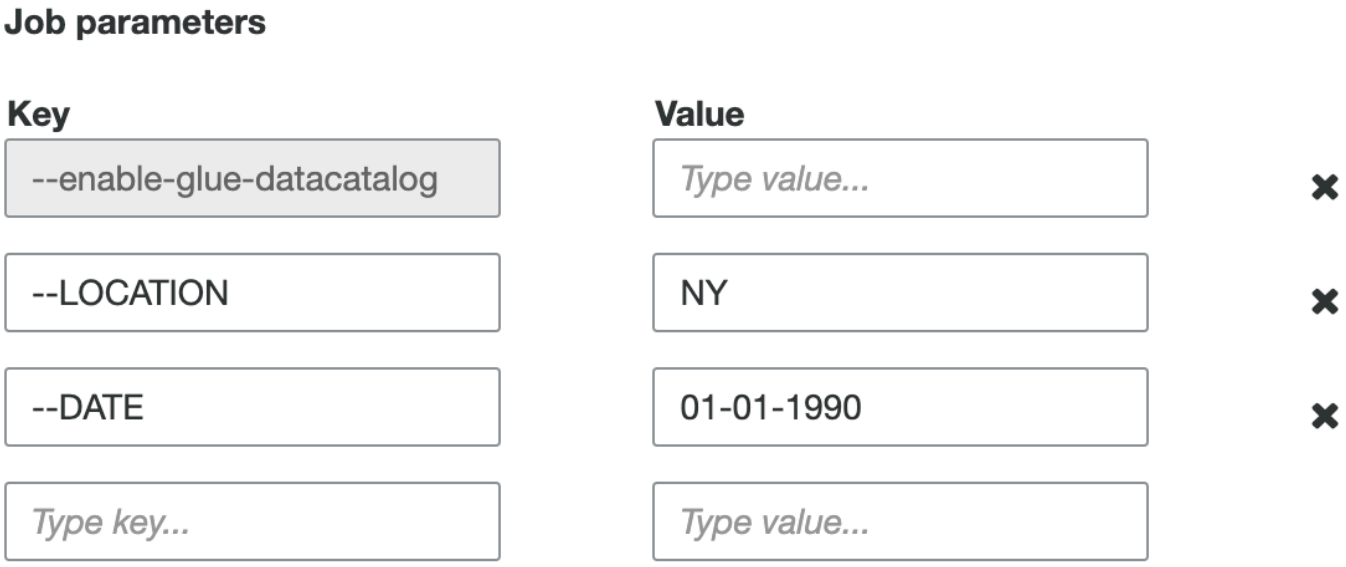
Now that you know how to access the parameters, learn how to pass parameters to a job.

Via the AWS Glue console

By default, the parameters of an AWS Glue job are empty but you can configure and save them via the AWS Glue console or via Glue Studio for future job runs. This way, you can re-run the job with these preset parameters, and you won't have to type the values again.

To configure the parameters, expand the **Security configuration, script libraries, and job parameters (optional)** section of the AWS Glue job:

Add the parameters under the **Job parameters** heading. Notice the double dash (--) before the parameter name.



Adding job parameters using the AWS Glue console

Via API (boto3)

You can pass the parameters via API as well. However, this doesn't save the parameter values and they need to be explicitly passed each time the job is run:

```
response = client.start_job_run(
JobName = 'Job_With_Params',
Arguments = {
'--LOCATION': 'NY',
'--DATE': '01-01-1990' } )
```

Via CLI

You can pass the parameters via CLI as well. However, this doesn't save the parameter values and they need to be explicitly passed each time the job is run:

```
start-job-run
--job-name Job_With_Params
--arguments <value>
```

The arguments can be passed using the following shorthand syntax:

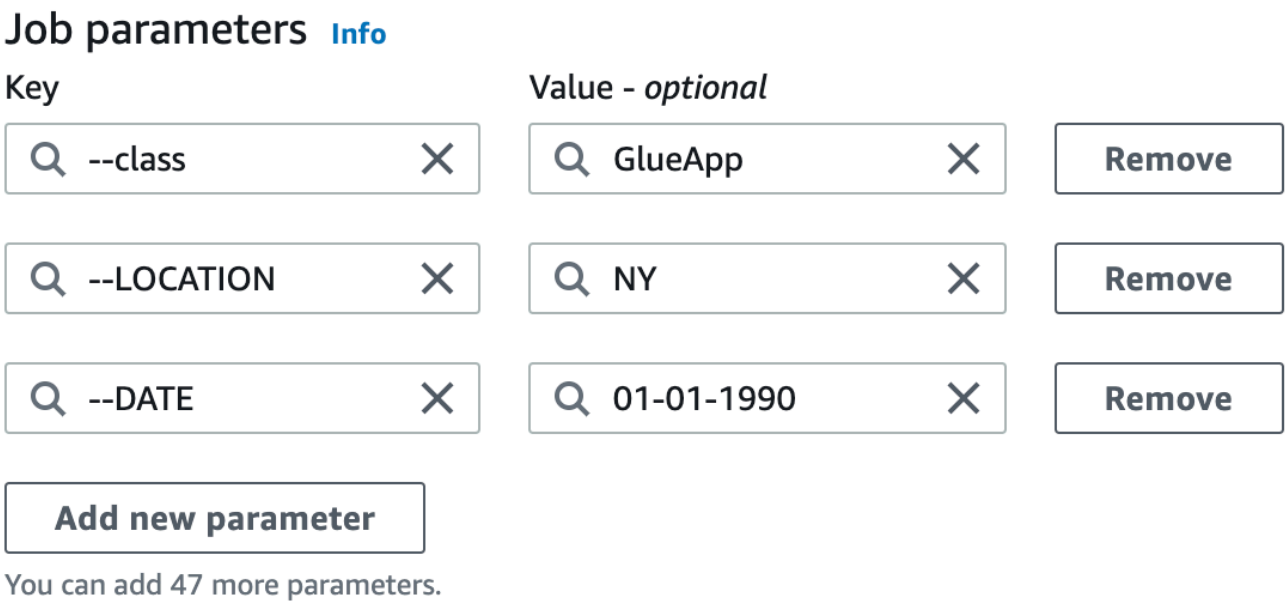
```
--arguments LOCATION=NY,DATE=01-01-1990
```

Or using JSON:

```
--arguments {"LOCATION":"NY","DATE":"01-01-1990"}
```

Using AWS Glue Studio

Similar to the AWS Glue console, AWS Glue Studio also allows the job parameters to be added and saved for future executions. On the AWS Glue Studio console, navigate to **Job Details > Advanced Properties > Job parameters > Add new parameter**:



Adding job parameters using AWS Glue Studio

Conclusion

In this whitepaper we explained what AWS Glue does, showed you some common design patterns where AWS Glue can be used in a data processing pipeline, described some challenges in building an efficient data pipeline, and described some best practices for designing and operating your data pipeline with AWS Glue, using guidance from the AWS Well-Architected Framework.

Contributors

Contributors to this document include:

- Durga Mishra, Sr. Solutions Architect, Amazon Web Services
- Arun A K, Solutions Architect, Amazon Web Services
- Narendra Gupta, Sr. Solutions Architect, Amazon Web Services
- Rajesh Agarwalla, Data Architect, Amazon Web Services

Further reading

For additional information, refer to:

- Top 10 Performance Tuning Tips for Amazon Athena (blog post)
- Load data incrementally and optimized Parquet writer with AWS Glue (blog post)
- *AWS Glue Best Practices: Building a Secure and Reliable Data Pipeline* (AWS whitepaper)
- *AWS Glue Best Practices: Building a Performant and Cost Optimized Data Pipeline* (AWS whitepaper)

Document revisions

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Initial publication	Whitepaper published.	August 26, 2022

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.