AWS DAS-C01 readiness domain3

Tuesday, 25 May 2021 11:28 PM

Amazon Glue ETL Process

Data source

AWS Glue natively supports the following data stores by using the JDBC protocol:

- Amazon Redshift
- Amazon RDS, all six database engines
- Non-AWS database engines
    - MariaDB
    - Microsoft SQL Server
    - MySQL
    - Oracle
    - PostgreSQL

Data transform

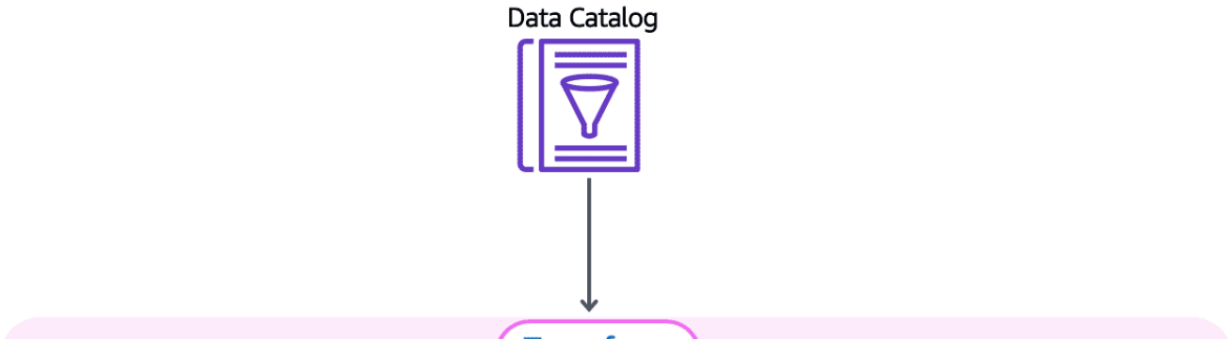
AWS Glue can generate ETL code in **Scala** or **Python** to extract data from the source and transform your data. You can also provide your own script in the AWS Glue console or API.

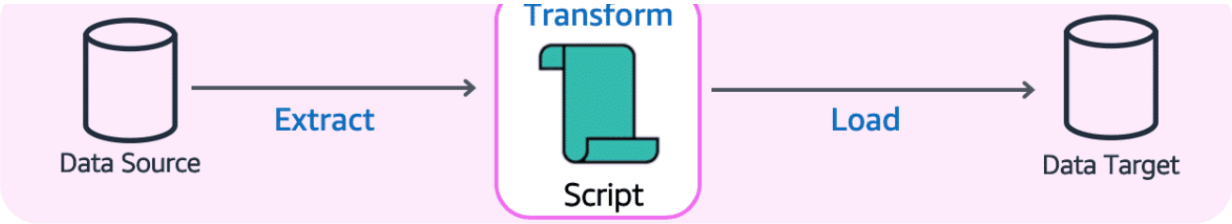
Data load

Once the data is transformed to match the target schema, the data is loaded into the target datastore.

Data catalog

The AWS Glue Data Catalog contains table definitions and metadata that is required to define ETL jobs. You use this metadata when you define a job to transform your data.



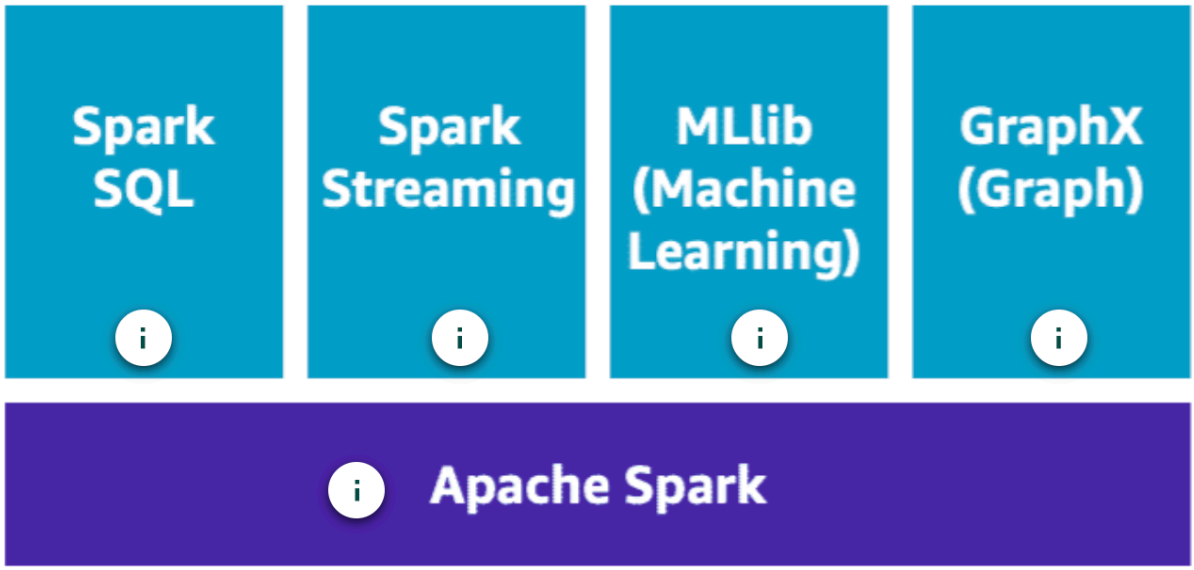


`AWS Glue Triggers

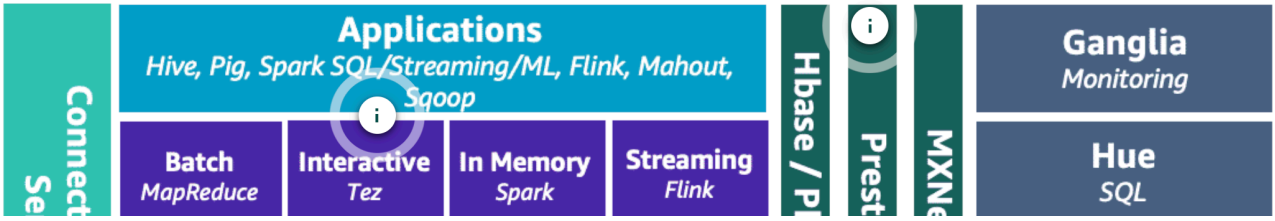
A trigger can be one of the following types:

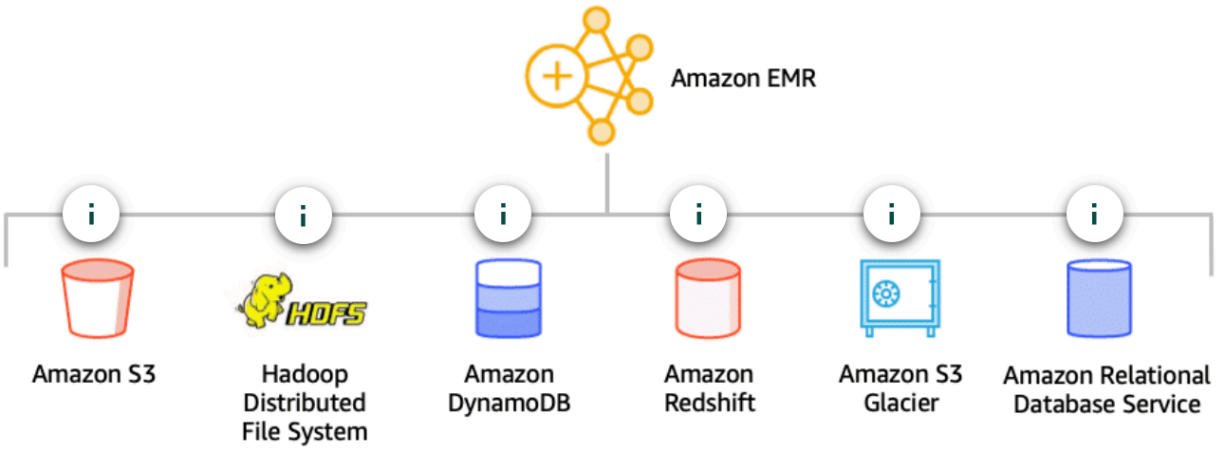
- Schedule
  A time-based trigger based on cron.
- Job events (conditional)
  An event-based trigger that fires when a previous job or multiple jobs satisfy a list of conditions. You provide a list of job events to watch for when their run state changes to succeeded, failed, stopped, or timeout. This trigger waits to fire until any or all the conditions are satisfied.
- On-demand
  The trigger fires when you start it. As jobs complete, any triggers watching for completion are also fired and dependent jobs are started.
- Other AWS Services:
  You can use Amazon CloudWatch Events, AWS Lambda, or AWS Step Functions to trigger an AWS Glue workflow.

AWS Glue runs on a fully managed, scale-out Apache Spark environment



If you want more flexibility, you can use Amazon EMR





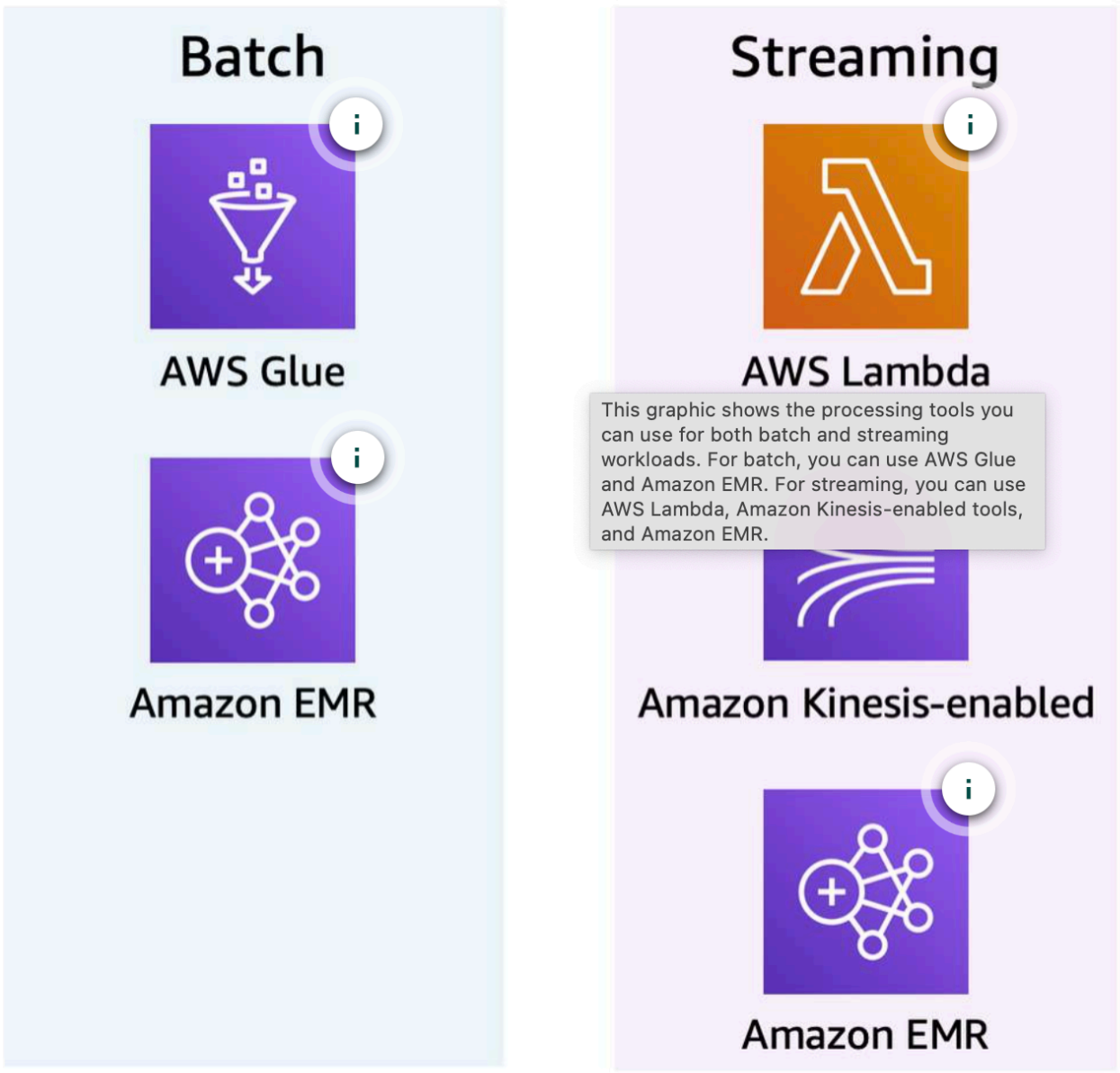
To ensure that your ETL requirements are met, you'll want to assess additional design considerations, such as ***scalability, reliability, and the cost*** of the ETL tooling.

Attribute	AWS Glue	Amazon EMR	AWS Lambda	Amazon Kinesis Client Library (KCL)	Kinesis Data Analytics
Scale	Data Processing Units (DPUs)	Nodes	Automatic	Nodes	Automatic
Reliability	Handles retries and errors automatically	Replaces failed instances	Managed by Lambda	KCL Checkpoints	Managed by Kinesis
Cost	Hourly rate, billed by the second	Per second + Amazon Elastic Compute Cloud (Amazon EC2)	Number of requests + duration	Nodes + DynamoDB	KPUs + storage

`

Batch vs streaming

Batch vs Streaming



Batch - AWS Glue
Batch oriented, schedule at minimum 5min. It is completely serverless, ideal for use cases where minimal server maintenance and low operational burden is important.

Batch - AWS EMR
Amazon EMR can use for both streaming and batch. It provides more flexibility in customizing ELT processing.

Streaming - Lambda
Lambda reads data streams and invokes functions synchronously with an event that contains stream record.

Streaming - Kinesis-enabled
KCL, Kinesis data analytics, Kinesis data firehose to process and transform streaming

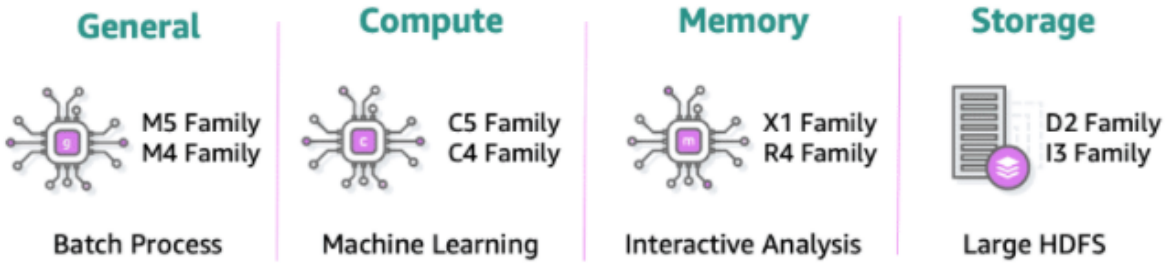
KCL, Kinesis data analytics, Kinesis data firehose to process and transform streaming data.

Streaming - AWS EMR

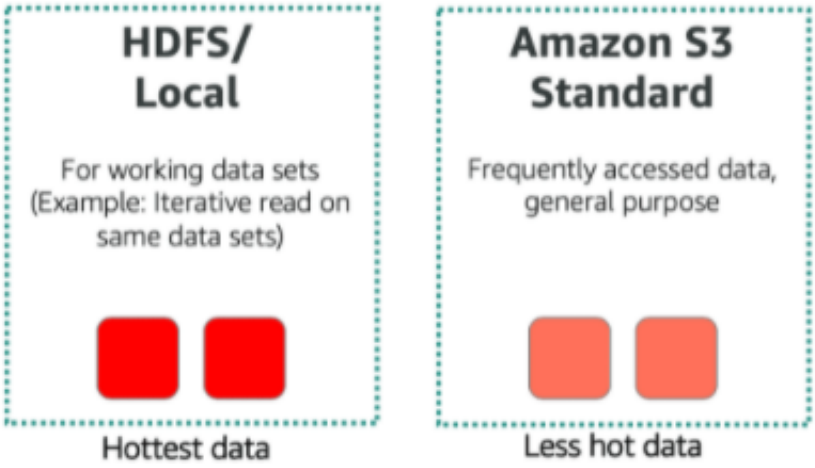
You can use Spark Streaming on Amazon EMR for streaming records.

EMR options

1. Instance Right sizing



2. Amazon S3 vs. local HDFS
   HDFS is used by the master and core nodes. While HDFS provides fast performance, it is considered ephemeral storage that is reclaimed when the cluster is terminated. It is best used for **caching the results** produced by intermediate job-flow steps. **HDFS** is useful for the **hottest** data sets.



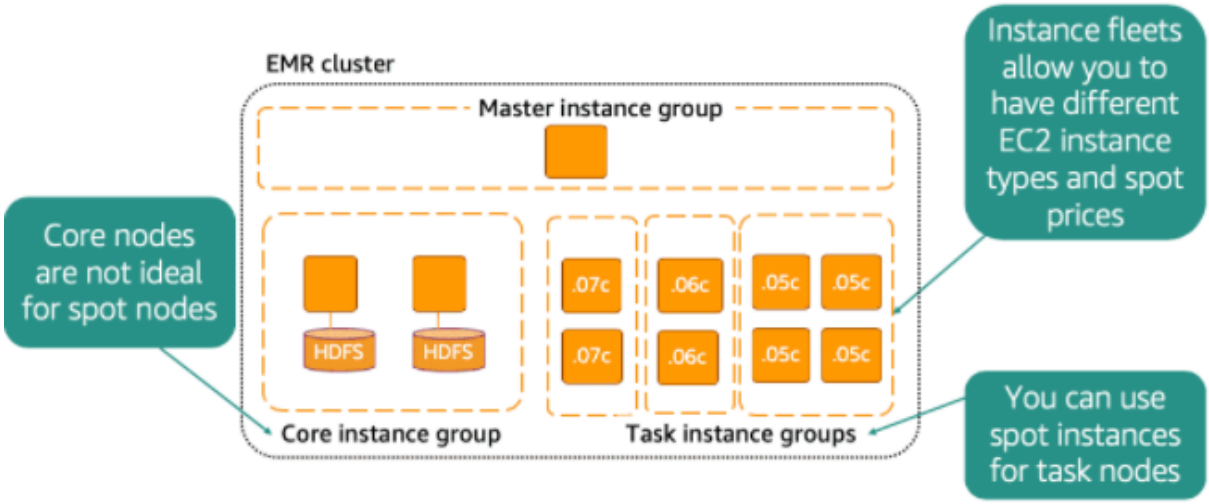
3. Transient vs. persistent clusters
   You can lower the operational cost of your solution by running the Amazon EMR cluster only when it is needed.

   A global solution that ingests data 24/7 might not provide an opportunity for a transient cluster. However, if you have a question that involves business analysts doing work on a set schedule, then a **transient cluster** will likely be part of the answer.

4. Using Spot instances

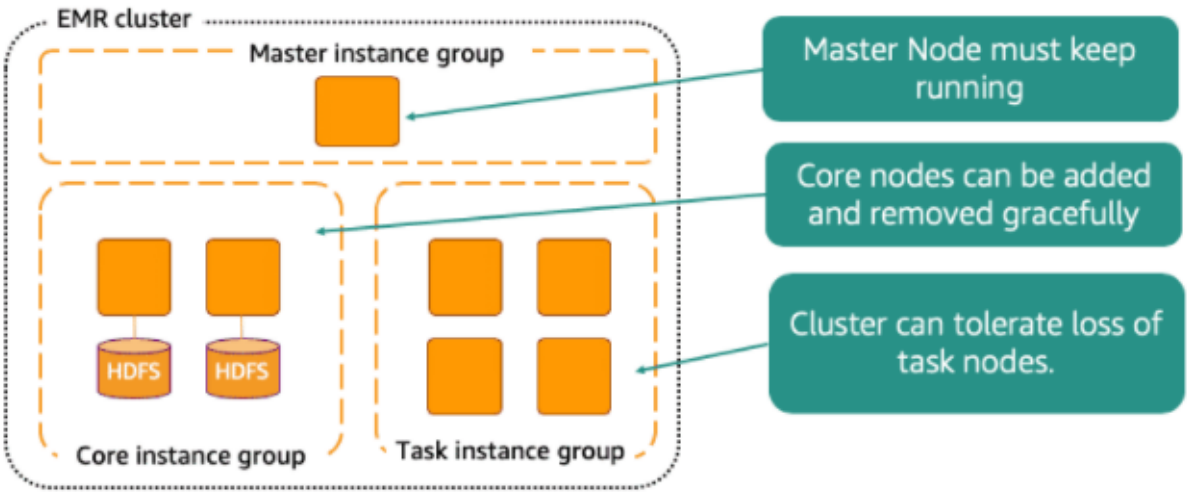
4. Using Spot instances

Since core nodes run HDFS and task nodes do not, Amazon EMR task nodes are ideal for Spot instances. In this case, if the Spot price increases and you lose task nodes, you will not lose data stored in HDFS.



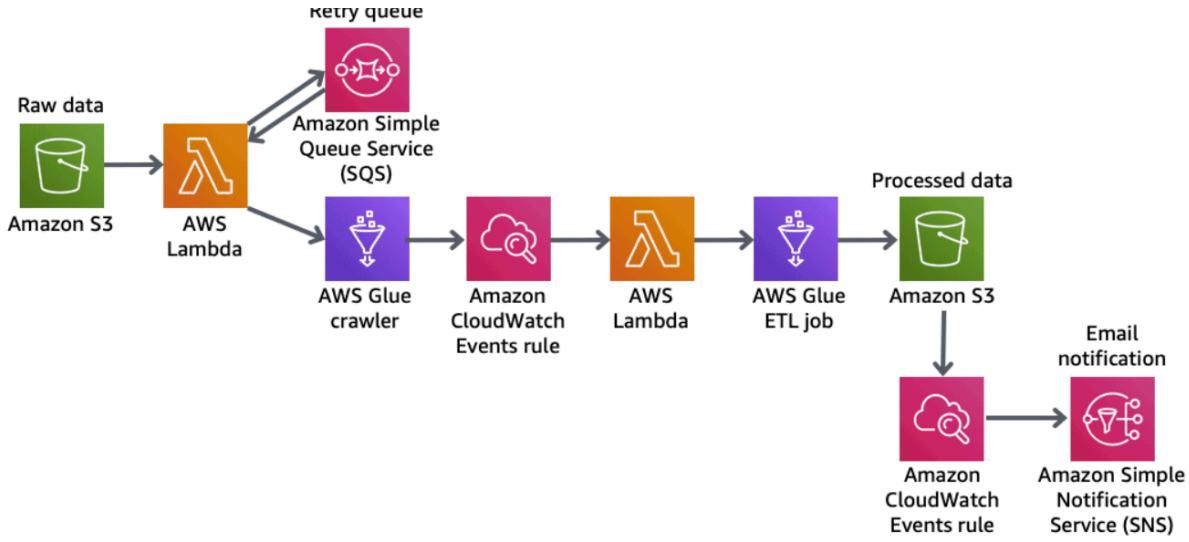
5. Elastic cluster vs. static cluster

- You can configure automatic scaling for the core instance group and task instance groups when you first create them or after the cluster is running. Amazon EMR automatically configures Amazon EC2 Auto Scaling parameters according to rules you specify, and then adds and removes instances based on an Amazon CloudWatch metric.
- You can manually resize the core instance group and task instance groups by manually adding or removing Amazon EC2 instances.
- You can add a new task instance group to the cluster.



Orchestration Patterns for ETL

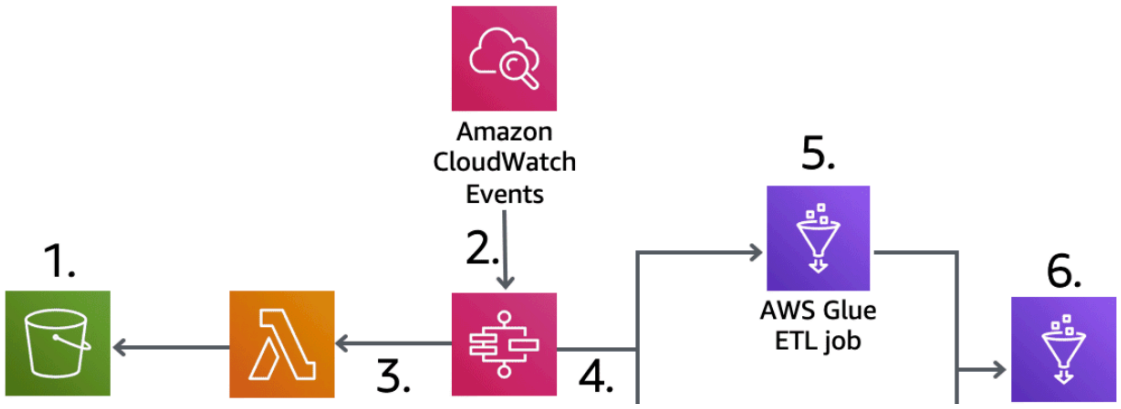
Automating AWS Glue ETL jobs with AWS Lambda



The architecture shown above uses Lambda to trigger an ETL workflow. Here are the general steps of the architecture:

1. First, you build a data lake with Amazon S3 as the primary data store. Ingested data lands in an Amazon S3 bucket that we call the **raw zone**.
2. To make that data available, you have to **catalog its schema in the AWS Glue Data Catalog.** You can do this using a **Lambda** function invoked by an **Amazon S3 trigger** to start an **AWS Glue crawler** that catalogs the data. To handle errors with Lambda, this architecture uses an **Amazon Simple Queue Service** (Amazon SQS) queue for event debugging and retries.
3. When the AWS Glue crawler is finished creating the table definition, you invoke a second **Lambda function** using a CloudWatch Events rule. This step starts an AWS Glue ETL job to convert the data to **Apache Parquet** format.
4. The AWS Glue ETL job stores the converted data into the Amazon S3 bucket that we call the **processed zone**.
5. As soon as the ETL job finishes, another **CloudWatch Events** rule sends you an email notification using an **Amazon Simple Notification Service** (Amazon SNS) topic. This notification indicates that your data was successfully processed.

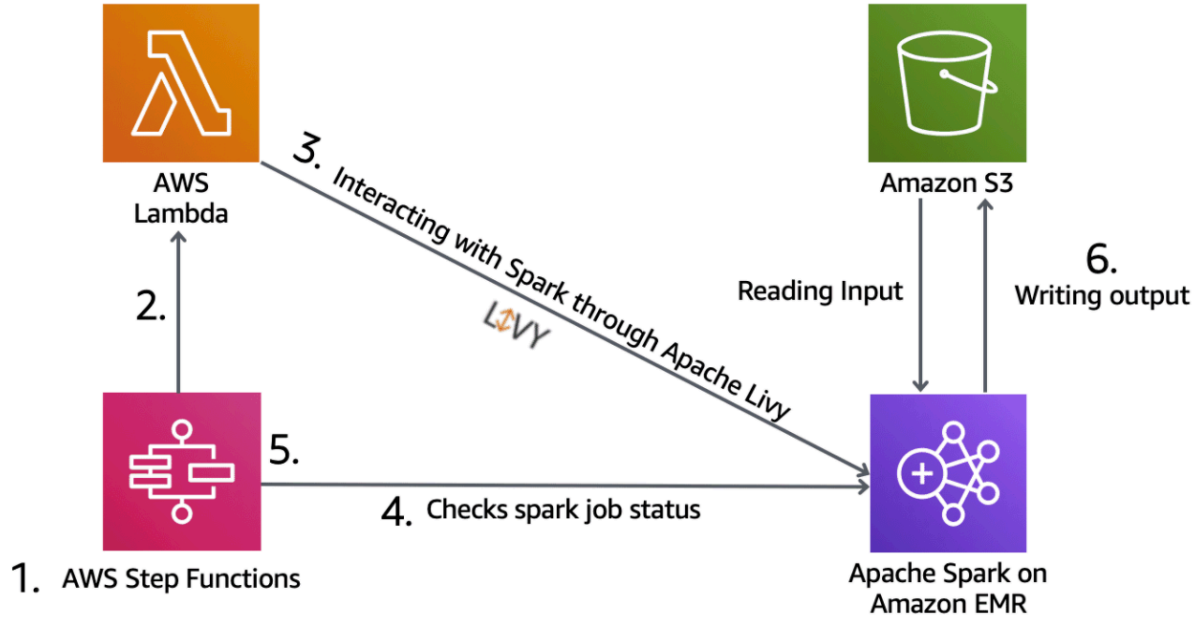
<u>Using Step Functions with AWS Glue</u>



In the diagram above, the services are orchestrated together by AWS Step Functions.

1. First, Amazon S3 acts as the landing zone for all ingested data. In this example, this architecture is used to combine two different data sets, one consisting of sales data and the other consisting of marketing data.
2. CloudWatch Events schedules the AWS Step Functions state machine to run weekly.
3. When the Step Functions state machine is triggered by the CloudWatch Event, it confirms the availability of the datasets through a Lambda function
4. The Step Functions state machine will then begin ETL job orchestration. Step Functions will start two AWS Glue ETL jobs in parallel.
5. One AWS Glue ETL job will begin to process sales data, while the other begins to process marketing data.
6. After the two AWS Glue ETL jobs run, another AWS Glue ETL job combines the two data sets.

Using Step Functions with Amazon EMR



In this diagram, AWS Step Functions and Apache Livy are being used to orchestrate Apache Spark applications. At a high level, the solution includes the following steps:

1. The AWS Step Functions state machine starts executing, and the input file path of the data stored in Amazon S3 is passed to the state machine

of the data stored in Amazon S3 is passed to the state machine.
2. The first stage in the state machine triggers a Lambda function.
3. The Lambda function interacts with Apache Spark running on Amazon EMR using Apache Livy, and submits a Spark job.
4. The state machine waits a few seconds before checking the Spark job status.
5. Based on the job status, the state machine moves to the success or failure state.
6. The output data from the Spark job is stored in Amazon S3.

Logging and auditing

To enable **traceability** in the **API** and at the EMR cluster level, you should be viewing logs in **AWS CloudTrail**. You can also review logs containing information on data written to the Amazon EMR **master node**. In Amazon EMR, you can view logs in the master node, archived logs in Amazon S3, or you can optionally view logs in the debugging tool.