AWS Glue (PoC)

2020 December

Rossano Marcos
Big Data Architect

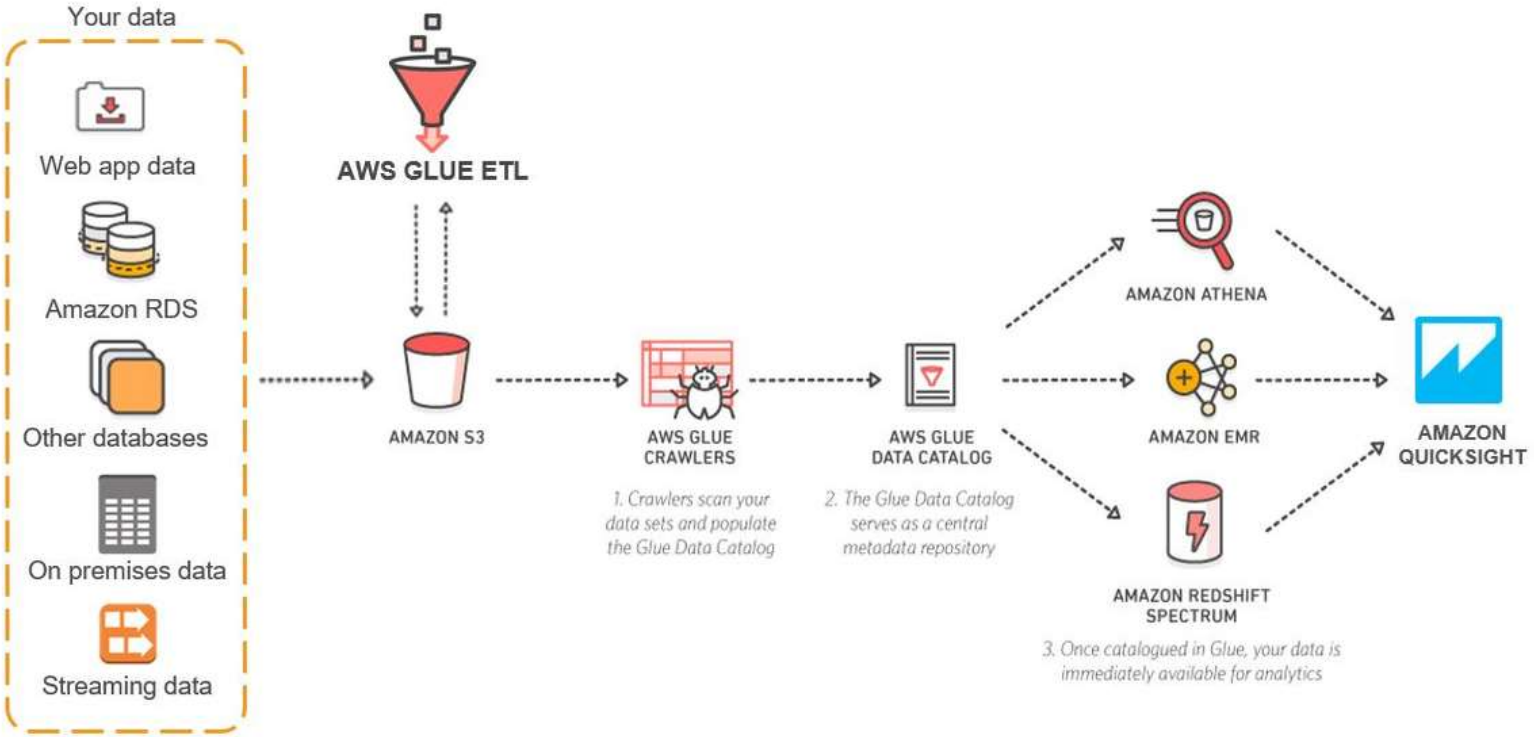
modak

Agenda

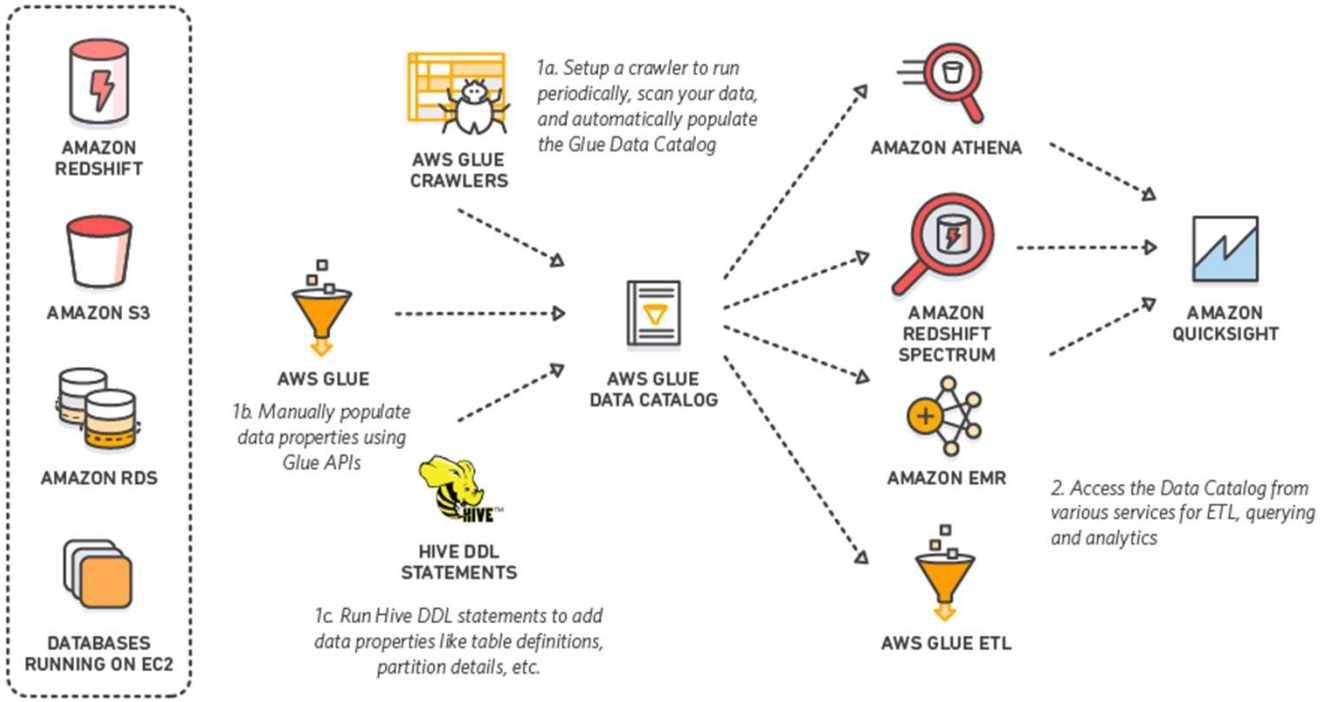
- AWS Glue
  - *Introduction*
  - *Core components*
  - *Demo 1 – Creating a "pure" AWS Glue Job "on the fly"*
    - *Understanding the Code behind Glue*
    - *Job1.py and Job1.Scala*
  - *Demo 2 – Creating a "custom" AWS Job (S3 to Parquet)*
    - *Job2.py - Pyspark*
    - *Job Scala (not almaren job)*
  - *Demo 3 - Creating a "custom" AWS Job (Json to Parquet)*
    - *Pyspark*
    - *Scala (not almaren job)*
  - *Limitations*

AWS GLUE

Data lake on Amazon S3 with AWS Glue



Data lake on Amazon S3 with AWS Glue

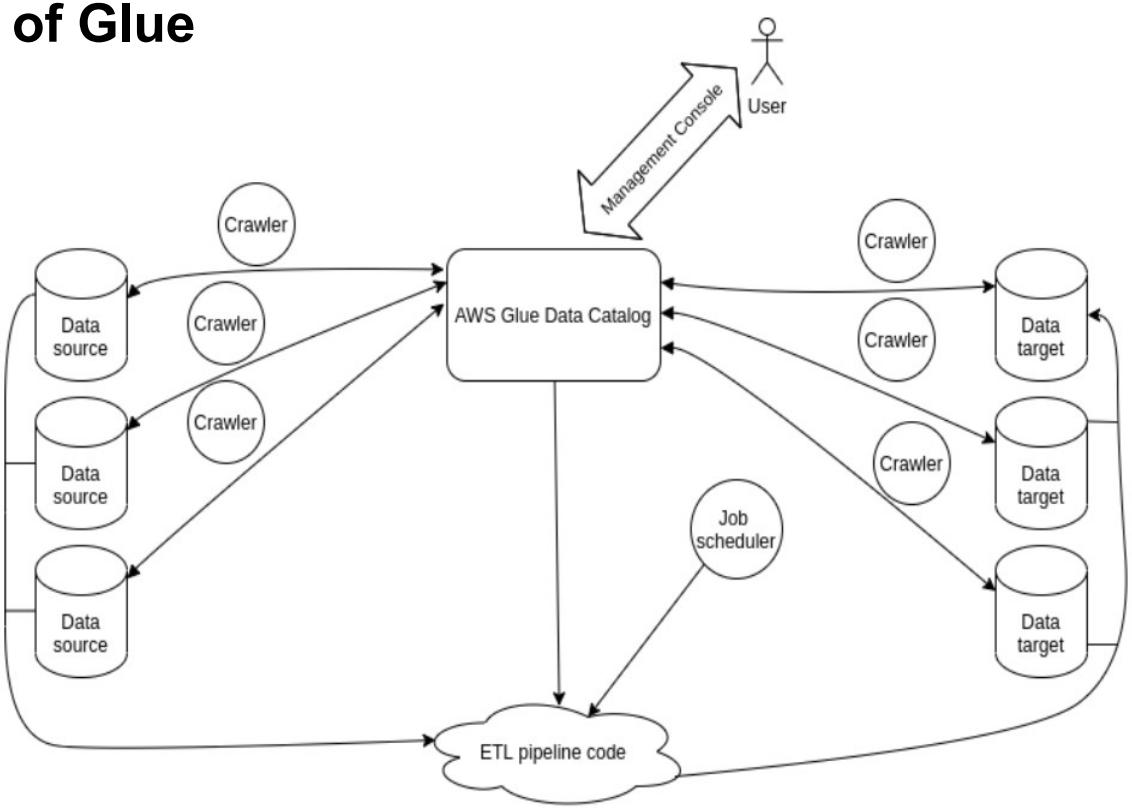


modak

Components of AWS Glue

- **Data catalog:** The data catalog holds the metadata and the structure of the data.
- **Database:** It is used to create or access the database for the sources and targets.
- **Table:** Create one or more tables in the database that can be used by the source and target.
- **Crawler and Classifier:** A crawler is used to retrieve data from the source using built-in or custom classifiers. It creates/uses metadata tables that are pre-defined in the data catalog.
- **Job:** A job is business logic that carries out an ETL task. Internally, Apache Spark with python or scala language writes this business logic.
- **Trigger:** A trigger starts the ETL job execution on-demand or at a specific time.
- **Development endpoint:** It creates a development environment where the ETL job script can be tested, developed and debugged.

Components of Glue



modak

modak

1

modak

DEMO 1 – CSV file to Parquet – (Invoices.CSV)

- Create a S3 Bucket to upload the data sources
- Create a S3 Bucket to write the target data
- Create a Crawler over the “data source”
  - Create a DB
  - Create a Table
  - Profile it
- Create an AWS Glue Job
- Show Script generated
- Show Data converted in Athena

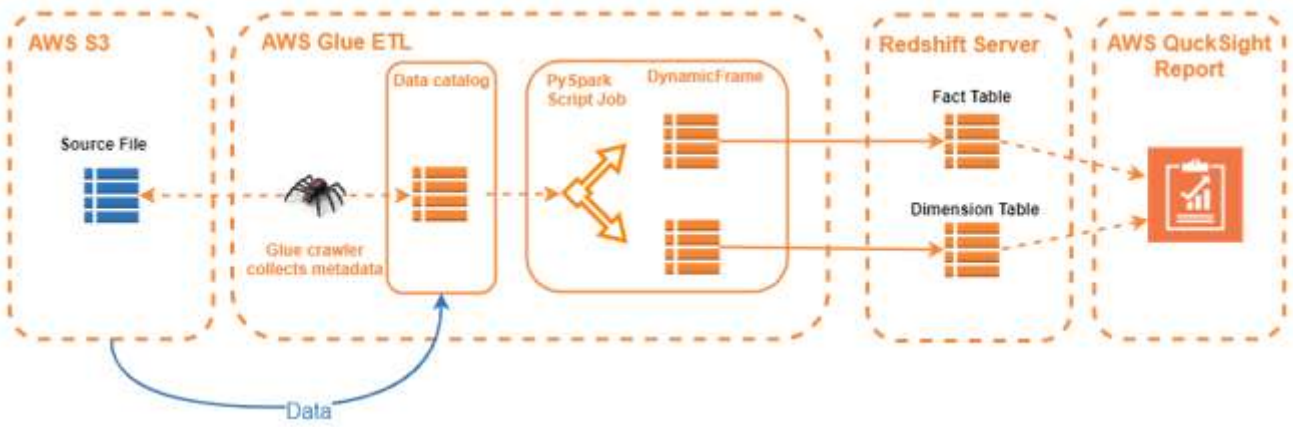
AWS GLUE SCRIPTS

Glue Jobs

- AWS provides **GlueContext** and **DynamicFrame**, an abstraction on top of **SparkContext** and **DataFrame** respectively to easily connect with Glue Catalog and do ETL
- **GlueContext** and **DynamicFrame** are not easy to use in your local development lifecycle.
- The binaries of these libraries are not available in maven/pip. It is not even available to download as a jar on the AWS website. Thus making it harder to code and compile locally.

Spark vs Glue
GlueContext = SparkContext
DynamicFrame = **DataFrame**

AWS Glue pipeline



Spark vs Glue

GlueContext = SparkContext

DynamicFrame = Spark DataFrame

modak

Script AWS Glue - Understanding

- Importing the necessary python libraries that create the ETL Job.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
```

Script AWS Glue - Understanding

- Get the name of Job through the command line.

```
args = getResolvedOptions(sys.argv, ['TempDir','JOB_NAME'])
```

- Initialize the GlueContext and SparkContext for the Job.

```
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
```

Spark vs Glue
GlueContext = SparkContext
DynamicFrame = DataFrame

“glueparquet” format option

glueparquet is a performance-optimized Apache parquet writer type for writing DynamicFrames. It computes and modifies the schema dynamically.

```
datasink = glueContext.write_dynamic_frame_from_options(
               frame=dynamicframe,
               connection_type="s3",
               connection_options={
                  "path": S3_location,
                  "partitionKeys": ["year", "month", "day", "hour"]
               },
               format="glueparquet",
               format_options = {"compression": "snappy"},
               transformation_ctx ="datasink")
```


modak

2

DEMO 2 – PySpark (Custom) CSV to Parquet

- JOB2.py and JOB2.Scala
  - Create a S3 Bucket to upload the data sources
  - Create a S3 Bucket to write the target data
  - Create a Crawler over the “data source”
    - Create a DB
    - Create a table
    - Profile it
  - Create an AWS Glue Job just to run the custom script

DEMO 3

- JOB3.py and JOB3.Scala
  - Create a S3 Bucket to upload the data sources
  - Create a S3 Bucket to write the target data
  - Create a Crawler over the “data source”
    - Create a DB
    - Create a table
    - Profile it
  - Create an AWS Glue Job just to run the custom script

Special Parameters Used by AWS Glue

AWS Glue recognizes several argument names that you can use to set up the script environment for your jobs and job runs:

- `--job-language` — The script programming language. This value must be either `scala` or `python`. If this parameter is not present, the default is `python`.
- `--class` — The Scala class that serves as the entry point for your Scala script. This applies only if your `--job-language` is set to `scala`.
- `--scriptLocation` — The Amazon Simple Storage Service (Amazon S3) location where your ETL script is located (in the form `s3://path/to/my/script.py`). This parameter overrides a script location set in the `JobCommand` object.
- `--extra-py-files` — The Amazon S3 paths to additional Python modules that AWS Glue adds to the Python path before executing your script. Multiple values must be complete paths separated by a comma (`,`). Only individual files are supported, not a directory path. Currently, only pure Python modules work. Extension modules written in C or other languages are not supported.
- `--extra-jars` — The Amazon S3 paths to additional Java `.jar` files that AWS Glue adds to the Java classpath before executing your script. Multiple values must be complete paths separated by a comma (`,`).
- `--user-jars-first` — When setting this value to `true`, it prioritizes the customer's extra JAR files in the classpath. This option is only available in AWS Glue version 2.0.
- `--extra-files` — The Amazon S3 paths to additional files, such as configuration files that AWS Glue copies to the working directory of your script before executing it. Multiple values must be complete paths separated by a comma (`,`). Only individual files are supported, not a directory path.
- `--job-bookmark-option` — Controls the behavior of a job bookmark. The following option values can be set.

- https://docs.aws.amazon.com/glue/latest/dg/aws-glue-programming-etl-glue-arguments.html

AWS LIMITATIONS

Limitations with AWS Glue – 1 / 2

- Amount of Work Involved in the Customization
  - AWS Glue is a managed ETL service for Apache Spark. And it is not a full-fledged ETL service like Talend, Xplexty, etc.

- Integration with other Platforms
  - AWS Glue is specifically made for the AWS console and its products. And hence it isn't easy to use for other technologies.
  - Also, it supports limited data sources like S3 and JDBC. Hence, you need to move your data to these cloud applications (if it is not there already) for the AWS Glue functioning.
  - This is one of the biggest limitations of the AWS Glue. To overcome this limitation, you need to have the above-mentioned data sources.

- Limitations of Real-time data
  - As AWS Glue only supports a handful of data sources like S3, there is no room to include an incremental synchronization with the data source.
  - Due to the lack of incremental sync, you cannot see the real-time data for complex operations.
  - You can overcome this challenge by portioning your data source sequences into a simplified process and seeing the real-time data.

Limitations with AWS Glue – 2 / 2

Required Skillset

AWS Glue is a serverless application, and it is still a novel technology.

Hence, the skillset required to implement and operate the AWS Glue is on the higher side.

You need to have a team with adequate knowledge expertise in the serverless architecture.

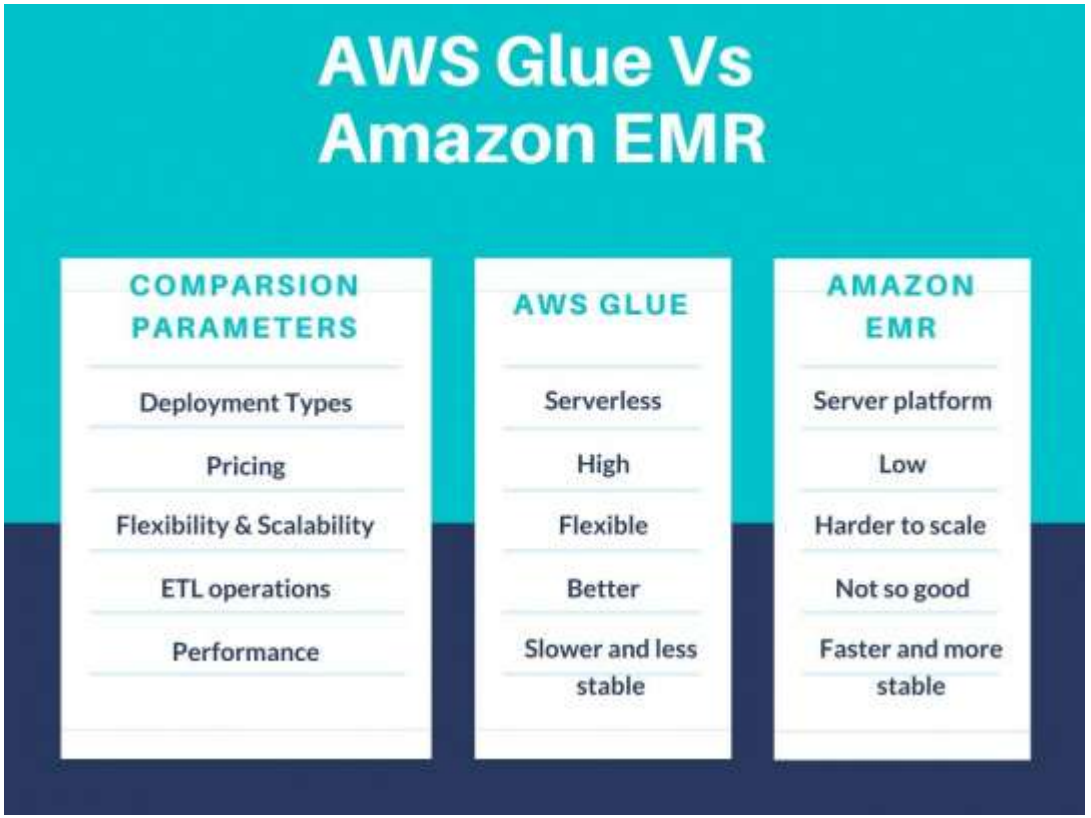
Process Speed and Room for Flexibility

AWS Glue requires you to test the changes in the live environment. It does not provide the test environment to analyze the repercussions of a change.

This slows down the deployment speed of the procedure.

Lack of Available Use Cases and Documentation

AWS Glue is still quite a new concept, and with serverless architecture, there is a lack of information readily available. Also, there are not many use cases and ready documentation that can solve your problems.



https://www.knowledgenile.com/blogs/aws-glue-vs-emr/

modak

Questions & Answers

Thank You !!!