**AWS**

**AWS GLUE:**

* AWS Glue is a fully managed ETL service that makes it simple and cost effective to categorize your data , clean it , enrich it and move it reliably between various data stores
* It has a central metadata repository: AWS glue catalogue
* It has a ETL engine that generate Python or Scala code and a scheduler for managing dependencies

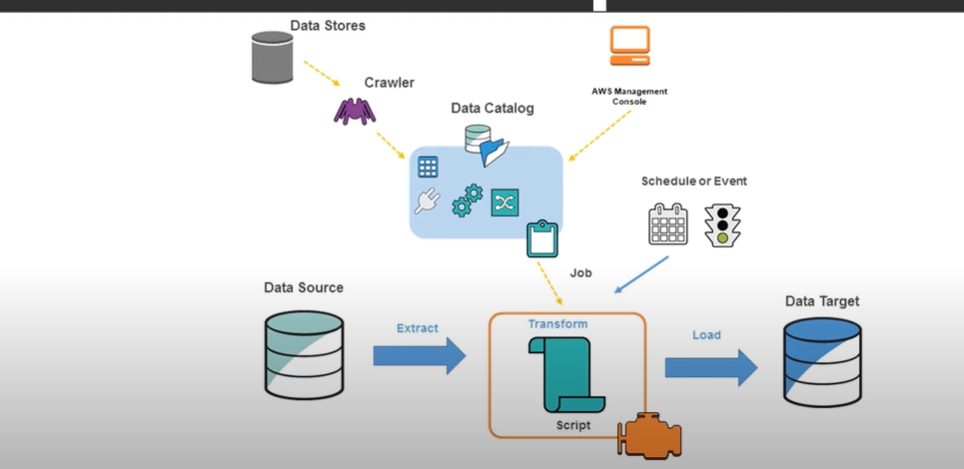
Usage:

1. To build a data warehouse to organize , cleanse, validate and format data
2. To run serverless queries against Amazon S3 data lake (Athena, Redshift spectrum can query S3 data using glue catalogue)
3. When you want to create event driven ETL pipeline (Using AWS lambda create an event that trigger a glue job when file is available)
4. To understand your data assets 🡪 We can store out data from various sources in S3 and still maintain a unified view of all our data in on place using AWS glue catalog. We can also configure the Hive external metastore as glue catalogue so that the external tables created in Hive can be still queried using Athena even though EMR is not running

Benefits:

1. Less Hassle: AWS Glue is integrated across a wide range of AWS services. It natively supports data stored in amazon Aurora , other RDS engines , Redshift , S3 and common data bases engines and databases running in our VPC
2. Cost effectiveness : AWS Glue is serverless fully managed ETL service so that we don’t need to care about setting up cluster resources and mange it (or auto scaling it). That is there is no infrastructure to provision or manage. It handles provisioning, configuration and scaling of the resources required to run ETL job using a fully managed apache spark environment. We need to pay only for the resources that our job consumes
3. More Power : Automatically generate codes, crawl data and suggest schemas etc

Components of an AWS glue ETL job:



* Data Catalog : Persistent metadata store in AWS glue. Table definitions/job definitions/ other control information to control AWS glue environment. Each AWS account has 1 data catalogue per region
* Classifier : A classifier determines the schema of the data. Glue provides classifier for common file format csv/xml/json/avro/parquet etc . It also provides classifiers for common DBMS using a JDBC connection. One can write their own classifier
  + You can provide a custom classifier to classify your data in AWS Glue. You can create a custom classifier using a grok pattern, an XML tag, JavaScript Object Notation (JSON), or comma-separated values (CSV). An AWS Glue crawler calls a custom classifier. If the classifier recognizes the data, it returns the classification and schema of the data to the crawler. You might need to define a custom classifier if your data doesn't match any built-in classifiers, or if you want to customize the tables that are created by the crawler.
  + AWS Glue console 🡪 classifier tab : Here you can find the list of classifiers
  + To add a classifier in the AWS Glue console, choose **Add classifier**.  When you define a classifier, you supply values for the following:
    - **Classifier name** – Provide a unique name for your classifier.
    - **Classifier type** – The classification type of tables inferred by this classifier.
    - **Last updated** – The last time this classifier was updated.
  + Classifier type can be any one of the following:
    - Grok Pattern : This is used to parse your data into a structured schema. The grok pattern is composed of named patterns that describe the format of your data store. You write this grok pattern using the named built-in patterns provided by AWS Glue and custom patterns you write and include in the **Custom patterns** field.
    - XML Row tag : For XML classifiers, this is the name of the XML tag that defines a table row in the XML document. Type the name without angle brackets < >. The name must comply with XML rules for a tag.
    - JSON and CSV classifier
* Connection : This contains configuration/properties to be connected to the data store
* Crawler : This is a program that connects to your data store (source or target). It defines the schema of the data with the help of a classifier . And then it creates metadata tables in data catalogue.
* Database: It is nothing but a data catalog table definitions organized into a logical group in AWS glue
* Data store : To persistently storing the data 🡪 AWS S3 and other RDBMS services
* Data source : It is the data store used as input to process and transform
* Data Target : This is the data store to which a process or a transform writes to
* Development end point : we can use to develop and test scripts
* Job : It’s a combination of Data source , Data Target and corresponding business rules implemented and transformation
* Notebook server : We can set up a notebook server on a development end point to run pyspark statements with AWS Glue statement
* Script : ETL code in pyspark or scala spark
* Table : Schema of the data
* Transform : Core logic to manipulate the data
* Trigger : triggers a glue ETL job . It can be scheduled or event based

**Steps for a simple Glue Job:**

