**Amazon Redshift**

Amazon Redshift is a fully managed, cloud-native data warehouse from AWS designed for large-scale data storage and analysis. It is a columnar database that does massive parallel processing.

The SQL INSERT statement is inefficient and should not be used in redshift. Instead, to do any sort of bulk loading, use the COPY command. Amazon Redshift automatically parallelizes data ingestion when using a single COPY command with multiple sources.

The most common source for loading data into Amazon Redshift seems to be S3. Data can also be loaded from an Amazon EMR Cluster, DynamoDB, an EC2 instance, or remote hosts that are accessible using SSH.

NOLOAD command checks the data file's validity without loading the data. It's like doing a dry run.

VACUUM command reclaims the space used by any rows that are already marked for deletion which improves the performance of your queries by reorganizing the underlying physical storage of the data within your tables. VACUUM command has these parameters FULL, SORT ONLY, DELETE ONLY, REINDEX, and RECLUSTER.

Four ways to scale Redshift: Classic Resize, Elastic Resize, Concurrency Scaling, Redshift Spectrum.

Amazon Redshift clusters can be scaled both horizontally and vertically.

Vertical scaling makes Redshift cluster larger or smaller. Horizontal scaling adds resources to Redshift clusters to address expected temporary spikes in utilization with minimal disruption.

The Classic Resize operation is best for those operations that change the size and shape of the Redshift cluster. This includes operations that would either cut the size of the cluster in half or double it. It is for changes that will be--more or less--permanent. It won't retain system tables and records marked for deletion.

The Elastic Resize operation is best used to address expected spikes in demand. The Elastic resize retains system log tables.

The Concurrency Scaling feature temporarily adds more compute power--on demand--to process queries.

Amazon Redshift Spectrum allows data stored in Amazon S3 to be queried by the Redshift cluster using standard SQL.

Amazon Redshift has three types of distribution styles for tables: EVEN, KEY, and ALL. AUTO (default) is the fourth one which is a combination of EVEN and ALL.

The EVEN distribution is good for tables with many unique values. KEY distribution is best for joins. ALL distribution is appropriate only for static or relatively slow-moving tables that are used in joins or as lookup tables.  Do not use the ALL distribution for: Tables that will be used once. Tables are updated frequently. Tables have 10 million or more rows.

Amazon Redshift spectrum works in such a way that data stored in S3 can be queried directly. For Data Cataloging/Processing/Analytical purposes, it is moved from S3 to AWS Glue or AWS EMR or Athena. Then Amazon Redshift spectrum can do SQL queries on AWS Glue or AWS EMR or Athena. There is no need to run crawlers on the data.

AWS Glue is used because it stores metadata such as table and column names which becomes the databases, tables, and views that you see in the Athena query editor.

Amazon Redshift Serverless allows you to run Amazon Redshift and create your own data warehouse. It is a fully managed AWS service.

AWS Lakeformation helps in providing data (structured and unstructured) access to authorized users. It takes data from S3 and catalogs, crawls the data.

Partitioning data, compressing data, parquet data file format aids in performance and reduces costs. Example for Partitioning data is “MSCK REPAIR TABLE” and “ALTER TABLE EMPLOYEES ADD PARTITION (age=20)”

Data lake stores all structured and unstructured information. You cannot derive meaningful insights out of it. Data warehouse allows you to perform analysis on a portion of data, to derive meaningful insights.

Amazon Athena is a serverless service that allows you to quickly query unstructured, semi-structured, and structured data.

Amazon Athena for Apache Spark is an open-source engine that allows you to perform data processing on a very large scale.


AWS Glue is a fully managed ETL serverless architecture and tool that makes it simple and cost effective to categorize your data, clean it, enrich it and move it reliably between various data sources.

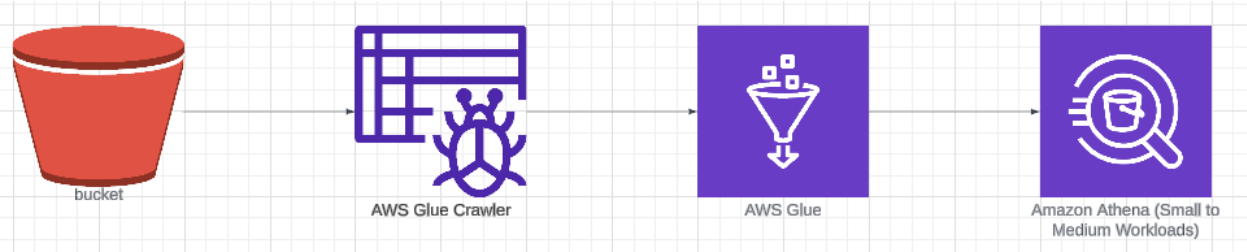
Benefits of AWS Glue: Crawlers detect and infer schemas from data sources with very little configuring and crawling can be scheduled and can also trigger job runs to perform ETL. Auto code generation gives you what you need in Python or Scala code to run a simple job or to extend within your own code additions.

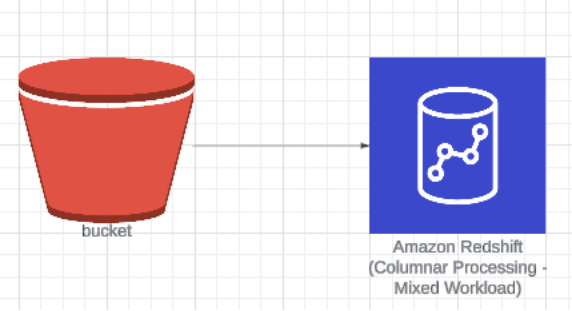

AWS Glue Interactive Sessions can be initiated from Local environment or AWS Glue Studio Notebooks (jupyter).

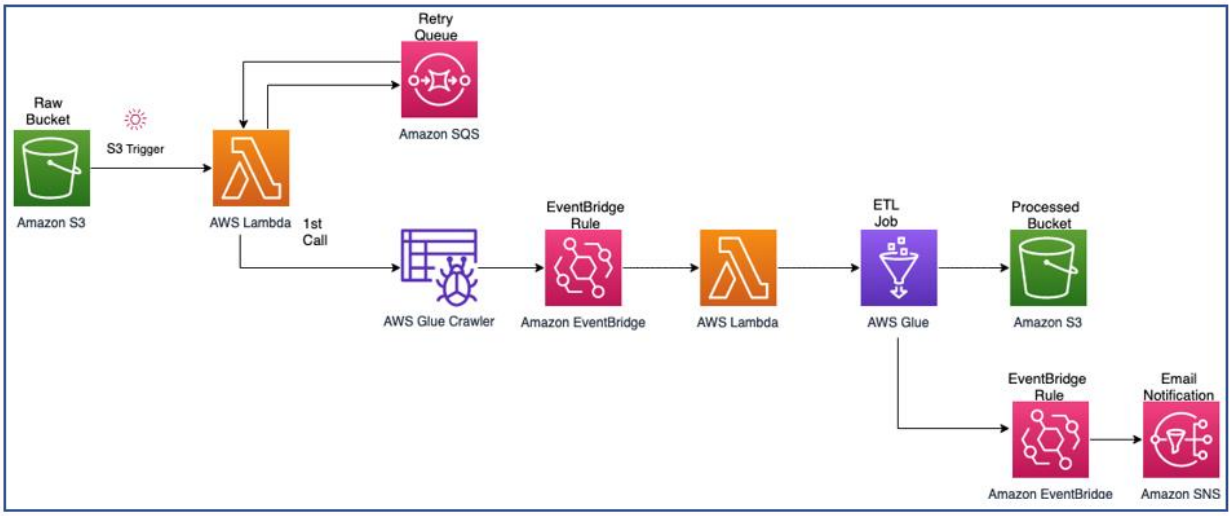
AWS EMR is based on Apache Hadoop framework, an open-source distributed processing framework intended for big data processing.

If you need flexibility with how you manage the engine or the underlying infrastructure, EMR on EC2 is best for you. Otherwise, if you need to run short-lived jobs that will run in an Apache Spark environment, Glue or EMR Serverless will save you time by managing the infrastructure for you.

**Use Cases:**




**Redshift with S3:**

```
create table venue(
   venueid smallint not null distkey sortkey,
   venuename varchar(100),
   venuecity varchar(30),
   venuestate char(2),
   venueseats integer);

)

// Inserts data into venue table from s3

copy venue
from 's3://awssampledbuswest2/tickit/venue_pipe.txt'
credentials 'aws_iam_role=arn:aws:iam::909737842772:role/caredshiftlab'
delimiter '|' region 'us-west-2';
```

**Compression in Redshift with S3:**

```
// Create table with compression encoding

CREATE TABLE example_table1 (id INT ENCODE AZ64, name VARCHAR(100) ENCODE
ZSTD, description TEXT ENCODE ZSTD,  created_at TIMESTAMP ENCODE ZSTD);


// Compress Data Locally Before Uploading to S3

gzip data.csv


// Upload the Compressed File to S3

aws s3 cp data.csv.gz s3://mybucket/mydata/


// Inserts data into table from s3

COPY my_compressed_table FROM 's3://mybucket/mydata/data.csv.gz' CREDENTIALS
'aws_iam_role=arn:aws:iam::123456789012:role/MyRedshiftRole' CSV GZIP;
```

// Once a table is created normally, you can alter a table with below compression encoding

ALTER TABLE mytable

ALTER COLUMN id ENCODE AZ64;

ALTER TABLE mytable

ALTER COLUMN name ENCODE ZSTD;

ALTER TABLE mytable

ALTER COLUMN description ENCODE ZSTD;

ALTER TABLE mytable

ALTER COLUMN created_at ENCODE ZSTD;

**Data Partition in S3:**

aws s3 mv "s3://$bucket_name/colors/red.csv" "s3://$bucket_name/colors/color=red/red.csv"

aws s3 mv "s3://$bucket_name/colors/blue.csv" "s3://$bucket_name/colors/color=blue/blue.csv"

aws s3 mv "s3://$bucket_name/colors/green.csv" "s3://$bucket_name/colors/color=green/green.csv"

**Athena Query:**

```
CREATE EXTERNAL TABLE `lab_db`.`colors`(
  `count` string,
  `timestamp` string,
  `data_field` string)
PARTITIONED BY (`color` string)
ROW FORMAT DELIMITED
```

```
  FIELDS TERMINATED BY ', '
  LINES TERMINATED BY '\n'
LOCATION
  's3://ca-labs-xxxx/colors'
TBLPROPERTIES (
  'skip.header.line.count'='1')
```

```
// This command looks for new partitions and updates the metadata for your table
MSCK REPAIR TABLE lab_db.colors;
```

**Convert CSV to Parquet:**

```
// This command converts csv to parquet
csv2parquet data.csv
```

```
CREATE EXTERNAL TABLE `lab_db`.`data_parquet`(
  `count` string,
  `timestamp` string,
  `data_field_one` string,
  `data_field_two` string)
STORED AS PARQUET
LOCATION
  's3://ca-labs-xxxx/convert/parquet'
tblproperties ("parquet.compress"="SNAPPY")
```

**Data Compression in S3:**

When working with text-based files (CSV, TSV, JSON), Athena supports the following compression formats:

- LZO
- BZIP2
- GZIP

For Parquet files, SNAPPY compression is the default in Athena. And for Apache ORC data files, the Athena default is ZLIB.

Redshift Spectrum benefits from compressing files and storing them in more efficient formats.