Diving into AWS Athena

Craig Godden-Payne
Feb 7 · 3 min read ★

Benefits of Athena

Athena is serverless, which means there is less management and maintainance overhead to run day to day tasks, such as updates, failovers etc. It also means that it is almost infinitely scalable, which is great if you want to start small, and want to have the ability to easily grow. It is also highly available, which means it is very durable and is built with security in mind. You can use IAM policies, ACLs and S3 bucket policies to secure Athena, which works well with anyone who is used to securing AWS applications.

Athena is really easy to run simple queries, but also complex analyses, it allows you to partition queries on any column, which improves query performance and also cost savings. It also has integration with Quicksight, which is Amazons Business Insight Analytics, which provides intelligent insights through machine learning.

Athena is ACID compliant, and all tables are external. This means when a table is dropped, only the metadata for the schema is removed, and not the underlaying data. Athena uses Apache Hive to define tables and databases. Databases are just logical namespaces of tables. Essentially, you are just describing the location and schema of the files within S3.

When you create a table, you include the location clause, specifying where the underlaying data lives. You can also specify glue crawlers, or other connections using DDL format.

e.g.

```
CREATE EXTERNAL TABLE webserver_access_logs (
    ip_address string,
    path string,
    request_time Timestamp
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION 's3://my-bucket/access-logs/'
```

A good breakdown of syntax can be found here: Amazon Docs or DDL Statements

Supported formats for Athena

- CSV, TSV, custom seperated values
- JSON
- Hadoop related formats, such as Parquet, ORC, Avro
- Log related formats, such as Logstash, Cloudtrail and Apache web server logs

Efficient Data Querying Formats

Parquet or ORC formats are efficient columnar formats. Their main benefits are its abilities with:

- Compression by column, this is based on the data type of the column, not the overall data set or row.
- This equates to less disk usage, and less I/O when running queries using Athena.
- Predicate pushdown, this enables only the blocks needed to be fetched by using statistics such as min/max to determine whether to fetch the block during the query.
- Parallelism, due to the format of the date, when querying using athena, it is possible for multiple readers to split the reading of data into readers to increase query speed.

SerDes

Athena uses SerDes (Serialise Deserialise) to intepret data that it reads from S3. By default, it supports parsing a variety of data formats, such as:

- RegexSerDes, useful for apache logs
- LazySimpleSerDes, useful for CSV, TSV and Custom Seperated values
- GrokSerDes, useful for logging formats
- HiveJsonSerDes, useful for json formats
- ParquetSerDes, useful for parquet format, works will snappy compression

Schema on read VS Schema on write

Schema on read

- Create the Schema
- Fit the data to the schema
- Great for performance or repetition

Schema on write

- Schema applied ‘just in time’
- Good for experimentation and exploration

Written on December 1, 2019.

Originally published on https://craig.goddenpayne.co.uk/athena/