Creating a Big Data Pipeline

Using AWS Serverless Technologies

Overview

- My background
- Business problem
- Approach
- Tools and technologies
- Walkthrough

My Background

- Alex Clark
- My Career
  - Data and business analysis
  - Business Intelligence
  - Data Engineering

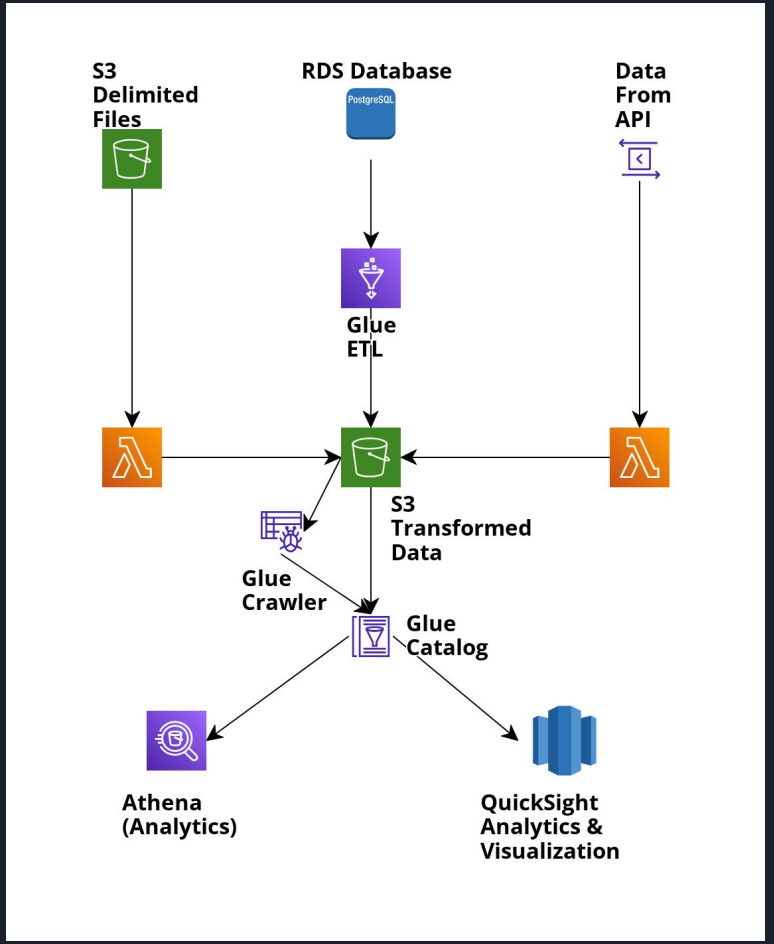
Business Problem

- Data from disparate sources
  - Raw files
  - SQL database
  - API
- How do we bring everything together?
- Finance data

Approach - Overview

- AWS Cloud Development Kit (CDK)
  - Infrastructure as code
  - Languages: Javascript, TypeScript, Python, Java, Go.
- Serverless technologies
  - Affordable
  - Low Maintenance
  - Use what you consume
  - Scalable

Approach - Diagram



Tools and Technologies

- CDK
- Cloud 9 - Web based IDE
- S3 - Scalable Storage
- Glue - Serverless data integration
  - ETL Jobs
  - Crawlers
- Lambda - Serverless computing
  - Layers
- Athena - serverless, interactive analytics service
- Quicksight - Serverless business intelligence service

AWS Lambda

- Serverless Computing
- Deploy code without deploying resources
- Many different programming languages
  - Java, Go, PowerShell, Node. js, C#, Python, and Ruby
- Lambda Layers
  - Package and share code
  - Package code on Linux

Layers

- Layer 1
  - Pandas, Numpy, Pyarrow,
    - Pyarrow used for writing parquet files
      - Open source
      - Column oriented
      - Efficient and cost effective
- Layer 2
  - Alpha Vantage
    - Stock data
    - Forex data
    - Free API Key available

AWS Glue

- Like a standard database
- Serverless
  - Data is in S3
- Data can be accessed with
  - Athena
  - Quicksight
- No need to maintain servers
  - Economical

AWS CDK

- Stack
  - Unit of deployment
- Infrastructure as code
- We’re using multiple smaller stacks
- Great for defining and deploy infrastructure
- Not meant for executing code on an adhoc basis

Transforming the data in JSON

```
{
   "USD":{
      "GBP":{
         "2010-01-01": {"KEYS":"VALUES"},
         "2010-01-02": {"KEYS":"VALUES"},
         "2010-01-03": {"KEYS":"VALUES"}
      },
      "BTC":{
         "2010-01-01": {"KEYS":"VALUES"},
         "2010-01-02": {"KEYS":"VALUES"},
         "2010-01-03": {"KEYS":"VALUES"}
      }
   },
   "INR":{
      "USD":{
         "2010-01-01": {"KEYS":"VALUES"},
         "2010-01-02": {"KEYS":"VALUES"},
         "2010-01-03": {"KEYS":"VALUES"}
      },
      "CNY":{
         "2010-01-01": {"KEYS":"VALUES"},
         "2010-01-02": {"KEYS":"VALUES"},
         "2010-01-03": {"KEYS":"VALUES"}
      }
   }
}
```

FROM_CURRENCY	TO_CURRENCY	DATE	OPEN	HIGH	LOW	CLOSE	ADJ_CLOSE	VOLUME
USD	GBP	2010-01-01	0.8	0.9	0.7	0.8	0.85	0.0
USD	GBP	2010-01-02	0.8	0.9	0.7	0.8	0.85	0.0
USD	GBP	2010-01-03	0.8	0.9	0.7	0.8	0.85	0.0
...								

What is Apache Parquet

- Column oriented data store
  - Columns instead of rows
  - Better compression
- Allows splitting
  - 1 file sent to multiple processors

```
4-byte magic number "PAR1"
<Column 1 Chunk 1 + Column Metadata>
<Column 2 Chunk 1 + Column Metadata>
...
<Column N Chunk 1 + Column Metadata>
<Column 1 Chunk 2 + Column Metadata>
<Column 2 Chunk 2 + Column Metadata>
...
<Column N Chunk 2 + Column Metadata>
...
<Column 1 Chunk M + Column Metadata>
<Column 2 Chunk M + Column Metadata>
...
<Column N Chunk M + Column Metadata>
File Metadata
4-byte length in bytes of file metadata
4-byte magic number "PAR1"
```

Source: https://parquet.apache.org/docs/file-format/

Why Apache Parquet?

Row Format:

Scan

	ticker	date	open	high	low	close	adj_close	volume
1	MSFT	2012-11-19	26.7999992371	26.7999992371	26.4699993134	26.7299995422	22.1029281616	57,179,300
2	MSFT	2012-11-20	26.7600002289	26.7999992371	26.4599990845	26.7099990845	22.086391449	47,070,400
3	MSFT	2012-11-21	26.7099990845	27.1700000763	26.6700000763	26.9500007629	22.2848453522	66,360,300
4	MSFT	2012-11-23	27.2299995422	27.7700004578	27.2000007629	27.7000007629	22.9050178528	57,845,700
5	MSFT	2012-11-26	27.5400009155	27.5799999237	27.1700000763	27.3899993896	22.6486759186	85,198,700
6	MSFT	2012-11-27	27.3600006104	27.3799991608	27.0400009155	27.0799999237	22.3923397064	45,018,600
7	MSFT	2012-11-28	27.0100002289	27.3899993896	26.7700004578	27.3600006104	22.623872757	53,018,400
8	MSFT	2012-11-29	27.1100006104	27.3600006104	26.8600006104	26.9500007629	22.2848453522	69,551,400
9	MSFT	2012-11-30	27.0499992371	27.1299991608	26.4899997711	26.6200008392	22.0119686127	83,690,200
10	MSFT	2012-12-03	26.7800006866	26.8199996948	26.3999996185	26.4300003052	21.8548564911	53,173,800
11	MSFT	2012-12-04	26.5	26.6299991608	26.3400001526	26.3700008392	21.8052406311	49,777,500
12	MSFT	2012-12-05	26.3799991608	26.9300003052	26.2600002289	26.6700000763	22.0533103943	68,283,800
13	MSFT	2012-12-06	26.8099994659	26.9799995422	26.6100006104	26.7299995422	22.1029281616	39,182,300
14	MSFT	2012-12-07	26.8199996948	26.8199996948	26.3700008392	26.4599990845	21.8796653748	46,162,100
15	MSFT	2012-12-10	26.5599994659	26.9699993134	26.5200004578	26.9400005341	22.2765808105	47,031,100
16	MSFT	2012-12-11	27.0499992371	27.4899997711	27.0499992371	27.3199996948	22.5907974243	52,282,800
17	MSFT	2012-12-12	27.5300006866	27.6200008392	27.0799999237	27.2399997711	22.5246486664	43,966,300
18	MSFT	2012-12-13	27.3199996948	27.5200004578	26.9500007629	27.1100006104	22.4171485901	45,080,100
19	MSFT	2012-12-14	27.1100006104	27.1299991608	26.7000007629	26.8099994659	22.1690788269	42,077,500
20	MSFT	2012-12-17	26.7900009155	27.2199993134	26.6800003052	27.1000003815	22.4088840485	42,046,100

Why Apache Parquet cont’d

Column Format:

- Header
  - Chunk 1
    - Ticker: [MSFT, MSFT, MSFT..IBM]
    - Date: [2012-11-09, 2012-11-10, 2012-11-11..2022-12-01]
    - Open: [26.7, 26.8, 26.9..30.1]
    - ..
  - Chunk 2
    - ..

Why Apache Parquet cont'd

- Performance
- Cost

Query	select l_orderkey from lineitem where l_partkey = 17766770		Savings Compared to Text Format
Text GZIP data	Runtime	33.06 seconds	
	Data Scanned	22 GB	
	Cost	$0.1	
Parquet GZIP data with no sorting	Runtime	1.72 seconds	~94% faster
	Data Scanned	2 GB	
	Cost	$0.009	~91% cheaper

Source: https://aws.amazon.com/blogs/big-data/top-10-performance-tuning-tips-for-amazon-athena/