Introduction to Amazon Athena

Interactive, Serverless, Pay-per-use, Query Service

Presented by:

Raul Frias
Solutions Architect Manager

Kevin Epstein
Head of Cloud Acceleration

© 2016, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

What to Expect from the Session

- Overview of Amazon Athena
- Key Features
- Customer Examples
- Troubleshooting Query errors
- Q&A

Challenges Customers Faced

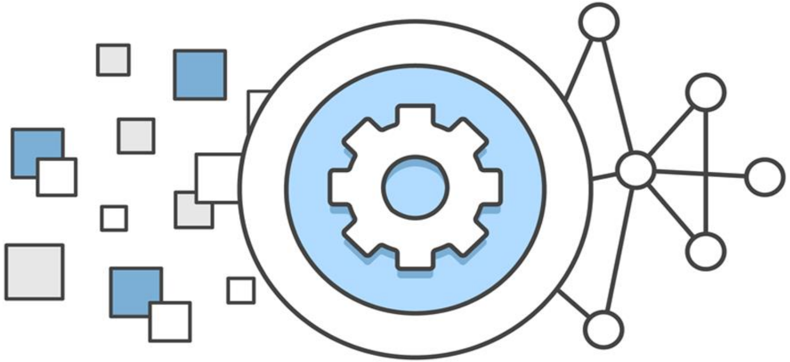
- Significant amount of work required to analyze data in Amazon S3
- Users often only have access to aggregated data sets
- Managing a Hadoop cluster or data warehouse requires expertise

Introducing Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data directly from Amazon S3 using Standard SQL

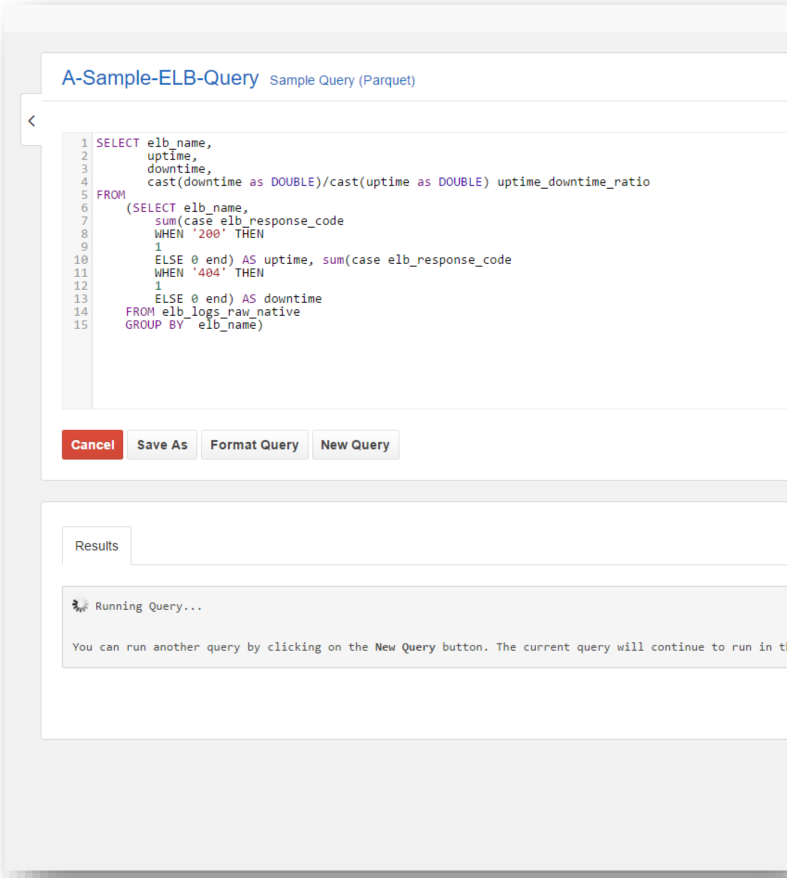
Athena is Serverless

- No Infrastructure or administration
- Zero Spin up time
- Transparent upgrades



Amazon Athena is Easy To Use

- Log into the Console
- Create a table
  - Type in a Hive DDL Statement
  - Use the console Add Table wizard
- Start querying



A-Sample-ELB-Query Sample Query (Parquet)

```
SELECT elb_name,
       uptime,
       downtime,
       cast(downtime as DOUBLE)/cast(uptime as DOUBLE) uptime_downtime_ratio
FROM
    (SELECT elb_name,
        sum(case elb_response_code
        WHEN '200' THEN
        1
        ELSE 0 end) AS uptime, sum(case elb_response_code
        WHEN '404' THEN
        1
        ELSE 0 end) AS downtime
    FROM elb_logs_raw_native
    GROUP BY  elb_name)
```

Running Query...

You can run another query by clicking on the **New Query** button. The current query will continue to run in t

Amazon Athena is Highly Available

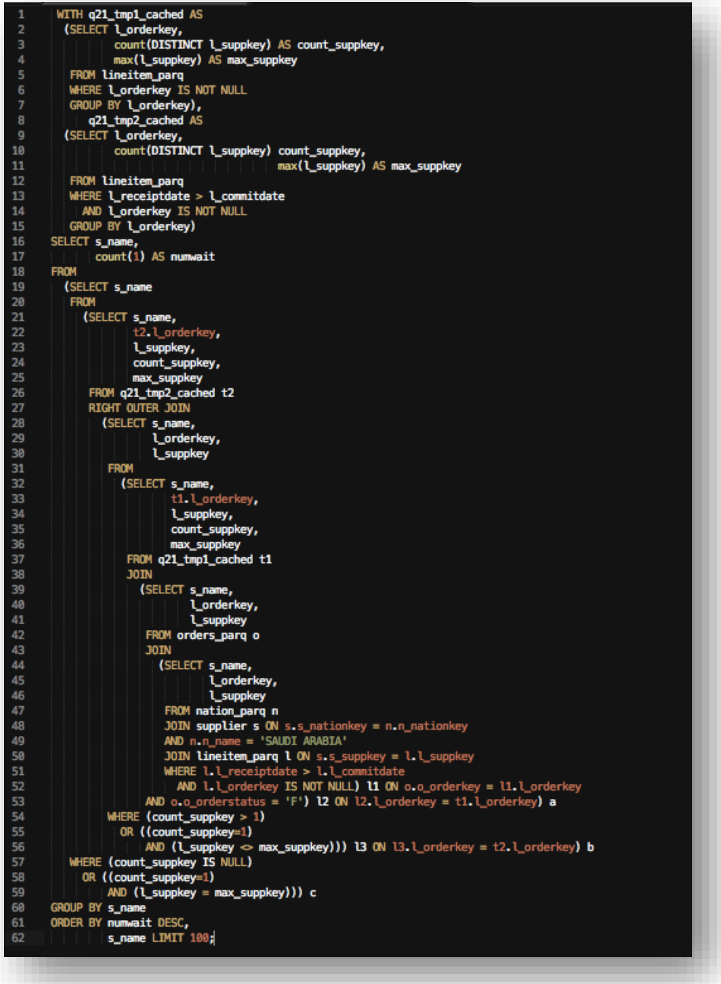
- You connect to a service endpoint or log into the console
- Athena uses warm compute pools across multiple Availability Zones
- Your data is in Amazon S3, which is also highly available and designed for 99.999999999% durability

Query Data Directly from Amazon S3

- No loading of data
- Query data in its raw format
    - Text, CSV, JSON, weblogs, AWS service logs
    - Convert to an optimized form like ORC or Parquet for the best performance and lowest cost
- No ETL required
- Stream data directly from Amazon S3
- Take advantage of Amazon S3 durability and availability

Use ANSI SQL

- Start writing ANSI SQL
- Support for complex joins, nested queries & window functions
- Support for complex data types (arrays, structs)
- Support for partitioning of data by any key
    - (date, time, custom keys)
    - e.g., Year, Month, Day, Hour or Customer Key, Date



```sql
 WITH q21_tmp1_cached AS
  (SELECT l_orderkey,
          count(DISTINCT l_suppkey) AS count_suppkey,
          max(l_suppkey) AS max_suppkey
   FROM lineitem_parq
   WHERE l_orderkey IS NOT NULL
   GROUP BY l_orderkey),
      q21_tmp2_cached AS
  (SELECT l_orderkey,
          count(DISTINCT l_suppkey) count_suppkey,
                                    max(l_suppkey) AS max_suppkey
   FROM lineitem_parq
   WHERE l_receiptdate > l_commitdate
     AND l_orderkey IS NOT NULL
   GROUP BY l_orderkey)
SELECT s_name,
       count(1) AS numwait
FROM
  (SELECT s_name
   FROM
     (SELECT s_name,
             t2.l_orderkey,
             l_suppkey,
             count_suppkey,
             max_suppkey
      FROM q21_tmp2_cached t2
      RIGHT OUTER JOIN
        (SELECT s_name,
                l_orderkey,
                l_suppkey
         FROM
           (SELECT s_name,
                   t1.l_orderkey,
                   l_suppkey,
                   count_suppkey,
                   max_suppkey
            FROM q21_tmp1_cached t1
            JOIN
              (SELECT s_name,
                      l_orderkey,
                      l_suppkey
               FROM orders_parq o
               JOIN
                 (SELECT s_name,
                         l_orderkey,
                         l_suppkey
                  FROM nation_parq n
                  JOIN supplier s ON s.s_nationkey = n.n_nationkey
                  AND n.n_name = 'SAUDI ARABIA'
                  JOIN lineitem_parq l ON s.s_suppkey = l.l_suppkey
                  WHERE l.l_receiptdate > l.l_commitdate
                    AND l.l_orderkey IS NOT NULL) l1 ON o.o_orderkey = l1.l_orderkey
               AND o.o_orderstatus = 'F') l2 ON l2.l_orderkey = t1.l_orderkey) a
         WHERE (count_suppkey > 1)
           OR ((count_suppkey=1)
               AND (l_suppkey <> max_suppkey))) l3 ON l3.l_orderkey = t2.l_orderkey) b
   WHERE (count_suppkey IS NULL)
     OR ((count_suppkey=1)
         AND (l_suppkey = max_suppkey))) c
GROUP BY s_name
ORDER BY numwait DESC,
         s_name LIMIT 100;
```

Familiar Technologies Under the Covers

presto

Used for SQL Queries

In-memory distributed query engine

ANSI-SQL compatible with extensions

HIVE

Used for DDL functionality

Complex data types

Multitude of formats

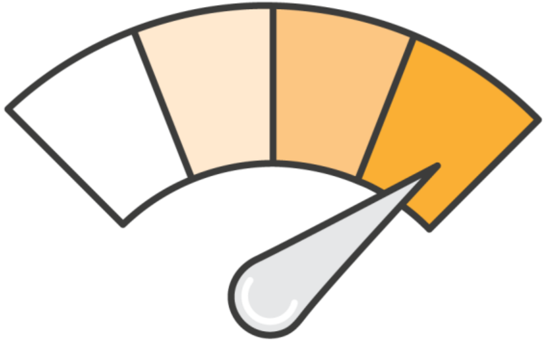
Supports data partitioning

Amazon Athena Supports Multiple Data Formats

- Text files, e.g., CSV, raw logs
- Apache Web Logs, TSV files
- JSON (simple, nested)
- Compressed files
- Columnar formats such as Apache Parquet & Apache ORC
- AVRO support – coming soon

Amazon Athena is Fast

- Tuned for performance
- Automatically parallelizes queries
- Results are streamed to console
- Results also stored in S3
- Improve Query performance
    - Compress your data
    - Use columnar formats



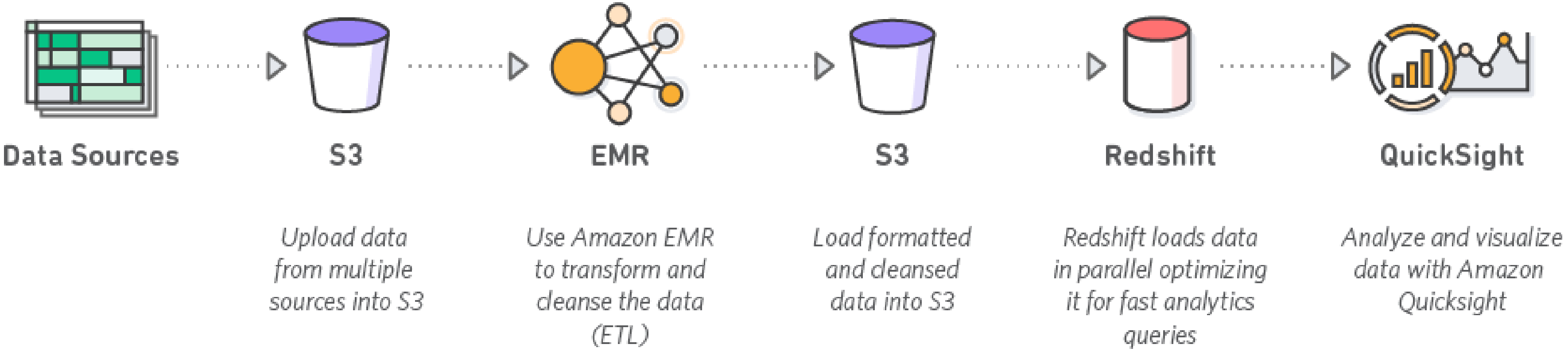
Amazon Athena is Cost Effective

- Pay per query
- $5 per TB scanned from S3
- DDL Queries and failed queries are free
- Save by using compression, columnar formats, partitions

Who is Athena for ?

- Any one looking to process data stored in Amazon S3
    - Data coming IOT Devices, Apache Logs, Omniture logs, CF logs, Application Logs
- Anyone who knows SQL
    - Both developers or Analysts
- Ad-hoc exploration of data and data discovery
- Customers looking to build a data lake on Amazon S3

A Sample Pipeline



Data Sources

S3

Upload data from multiple sources into S3

EMR

Use Amazon EMR to transform and cleanse the data (ETL)

S3

Load formatted and cleansed data into S3

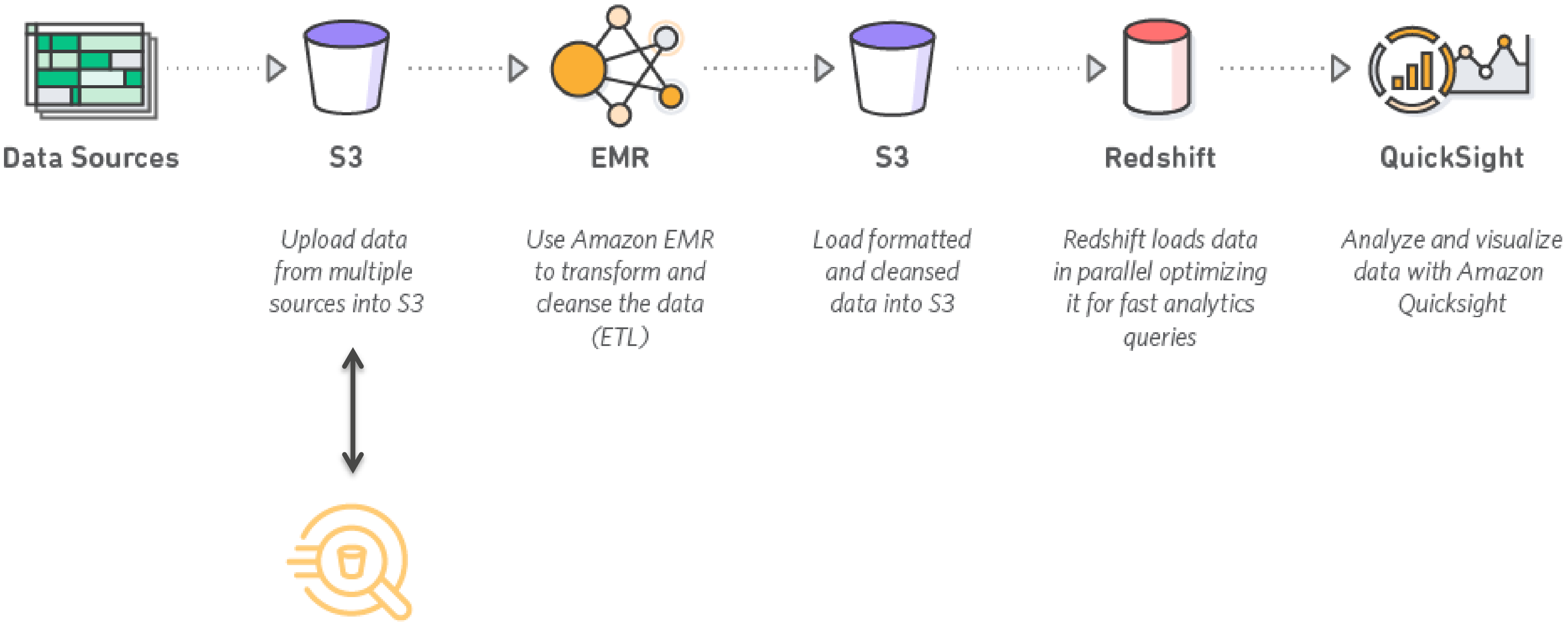
Redshift

Redshift loads data in parallel optimizing it for fast analytics queries

QuickSight

Analyze and visualize data with Amazon Quicksight

A Sample Pipeline



Data Sources

S3

Upload data from multiple sources into S3

EMR

Use Amazon EMR to transform and cleanse the data (ETL)

S3

Load formatted and cleansed data into S3

Redshift

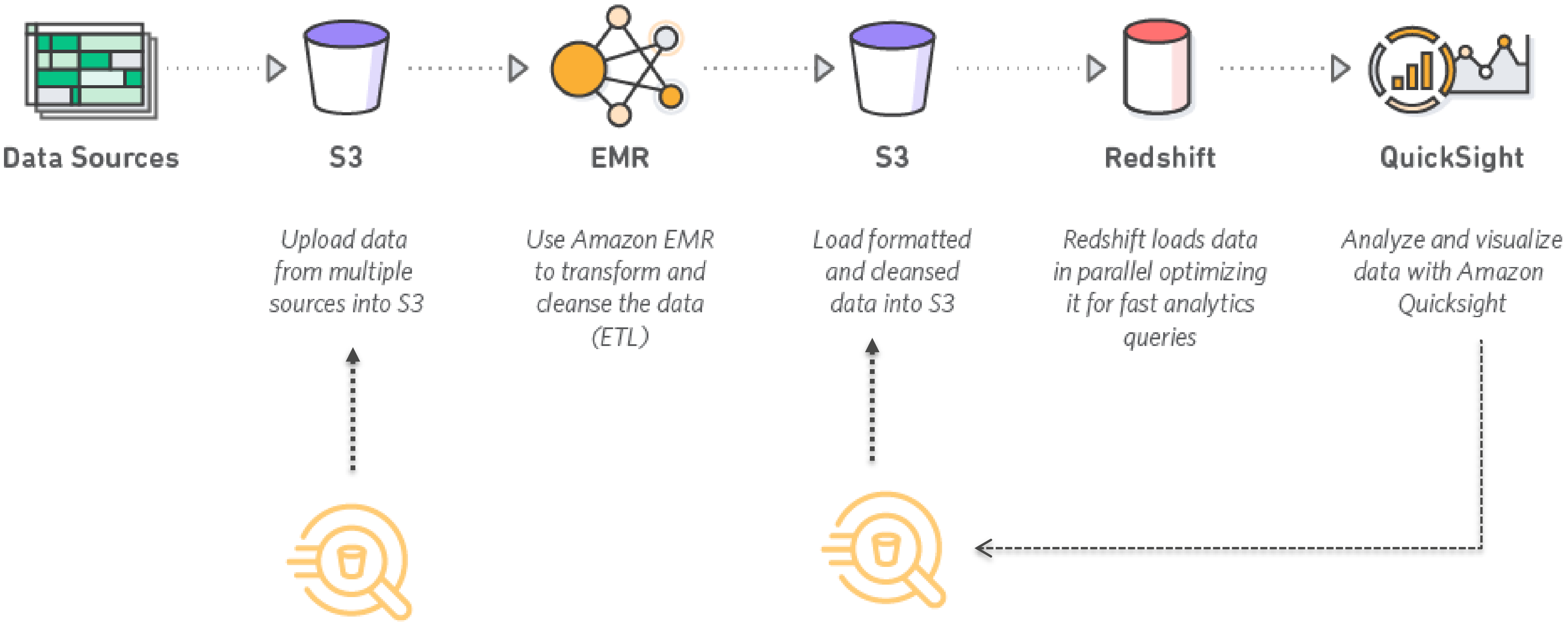
Redshift loads data in parallel optimizing it for fast analytics queries

QuickSight

Analyze and visualize data with Amazon Quicksight

Ad-hoc access to raw data using SQL

A Sample Pipeline



Data Sources

S3

Upload data from multiple sources into S3

EMR

Use Amazon EMR to transform and cleanse the data (ETL)

S3

Load formatted and cleansed data into S3

Redshift

Redshift loads data in parallel optimizing it for fast analytics queries

QuickSight

Analyze and visualize data with Amazon Quicksight

Ad-hoc access to data using Athena

Athena can query aggregated datasets as well

Re-visiting Challenges

~~Significant amount of work required to analyze data in Amazon S3~~

No ETL required. No loading of data. Query data where it lives

~~Users often only have access to aggregated data sets~~

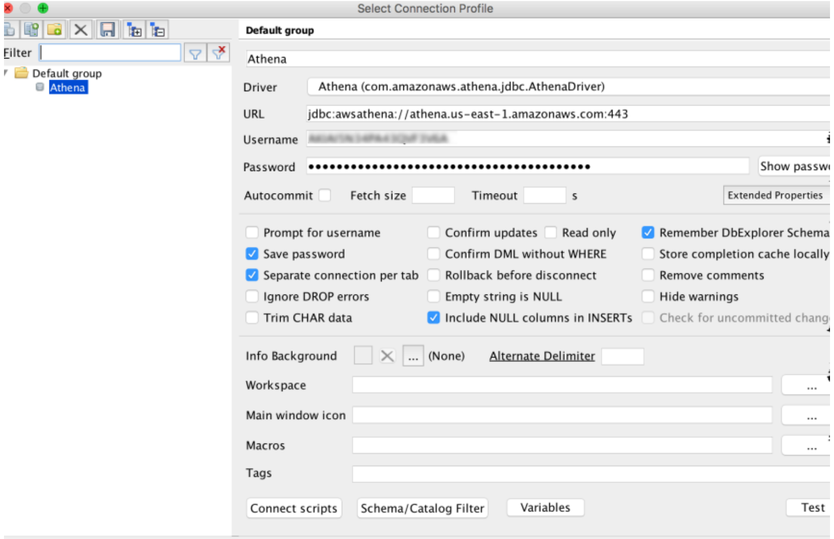
Query data at whatever granularity you want

~~Managing a Hadoop cluster or data warehouse requires expertise~~

No infrastructure to manage

Accessing Amazon Athena

Use the JDBC Driver



Select Connection Profile

Filter

Default group
Athena

Default group

Athena

Driver: Athena (com.amazonaws.athena.jdbc.AthenaDriver)

URL: jdbc:awsathena://athena.us-east-1.amazonaws.com:443

Username: [redacted]

Password: ••• Show passw

Autocommit ☐ Fetch size Timeout s Extended Properties

- ☐ Prompt for username
- ☑ Save password
- ☑ Separate connection per tab
- ☐ Ignore DROP errors
- ☐ Trim CHAR data
- ☐ Confirm updates
- ☐ Read only
- ☐ Confirm DML without WHERE
- ☐ Rollback before disconnect
- ☐ Empty string is NULL
- ☑ Include NULL columns in INSERTs
- ☑ Remember DbExplorer Schema
- ☐ Store completion cache locally
- ☐ Remove comments
- ☐ Hide warnings
- ☐ Check for uncommitted chang

Info Background ... (None) Alternate Delimiter

Workspace ...

Main window icon ...

Macros ...

Tags

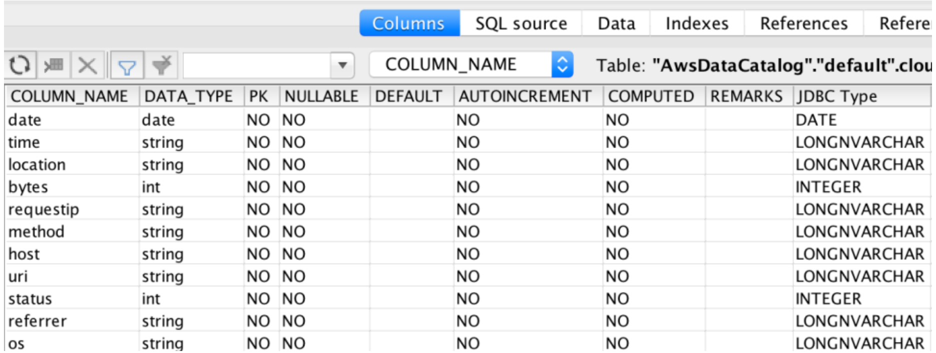
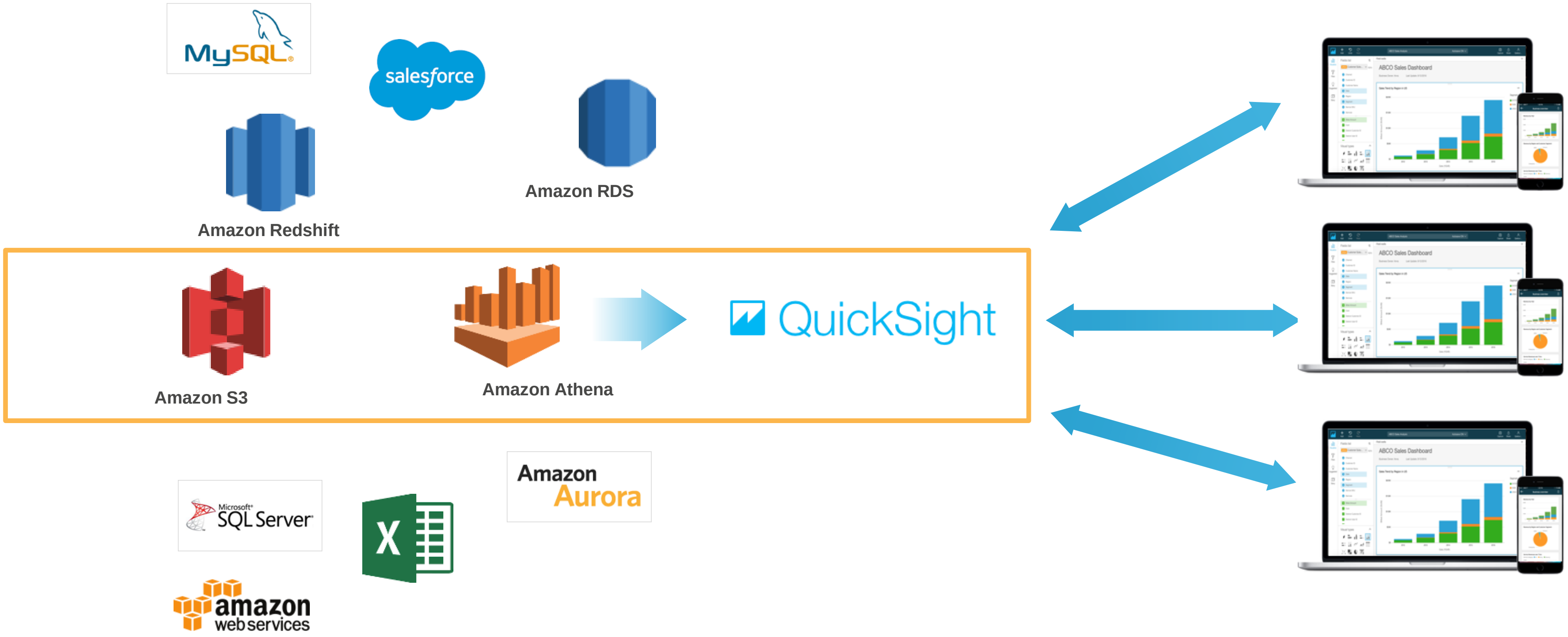


Table: "AwsDataCatalog"."default".clou

COLUMN_NAME	DATA_TYPE	PK	NULLABLE	DEFAULT	AUTOINCREMENT	COMPUTED	REMARKS	JDBC Type
date	date	NO	NO		NO	NO		DATE
time	string	NO	NO		NO	NO		LONGNVARCHAR
location	string	NO	NO		NO	NO		LONGNVARCHAR
bytes	int	NO	NO		NO	NO		INTEGER
requestip	string	NO	NO		NO	NO		LONGNVARCHAR
method	string	NO	NO		NO	NO		LONGNVARCHAR
host	string	NO	NO		NO	NO		LONGNVARCHAR
uri	string	NO	NO		NO	NO		LONGNVARCHAR
status	int	NO	NO		NO	NO		INTEGER
referrer	string	NO	NO		NO	NO		LONGNVARCHAR
os	string	NO	NO		NO	NO		LONGNVARCHAR

Using Amazon Athena with Amazon QuickSight

QuickSight allows you to connect to data from a wide variety of AWS, third-party, and on-premises sources including Amazon Athena

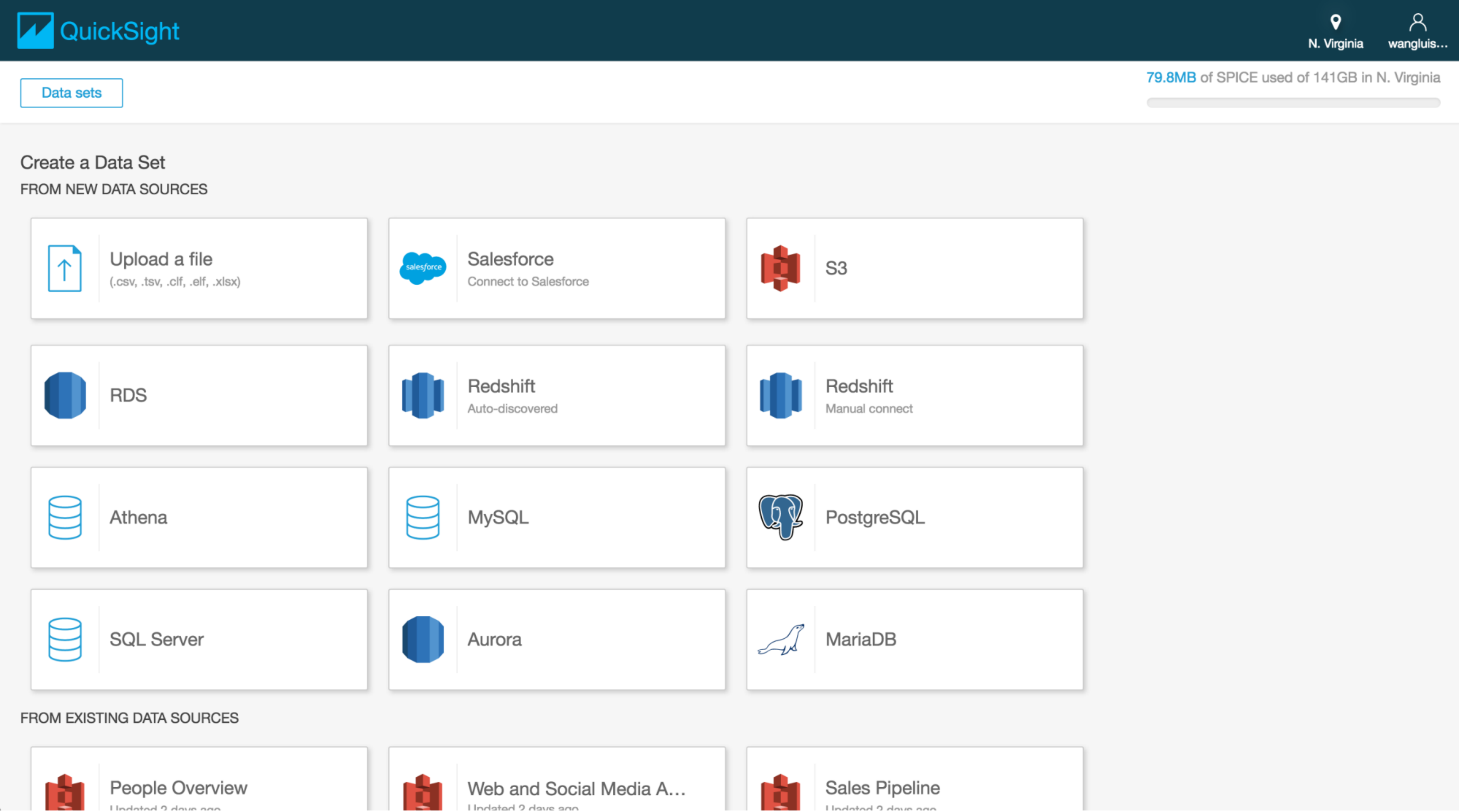


Amazon Redshift

Amazon RDS

Amazon S3

Amazon Athena



QuickSight

N. Virginia

wangluis...

Data sets

79.8MB of SPICE used of 141GB in N. Virginia

Create a Data Set

FROM NEW DATA SOURCES

Upload a file
(.csv, .tsv, .clf, .elf, .xlsx)

Salesforce
Connect to Salesforce

S3

RDS

Redshift
Auto-discovered

Redshift
Manual connect

Athena

MySQL

PostgreSQL

SQL Server

Aurora

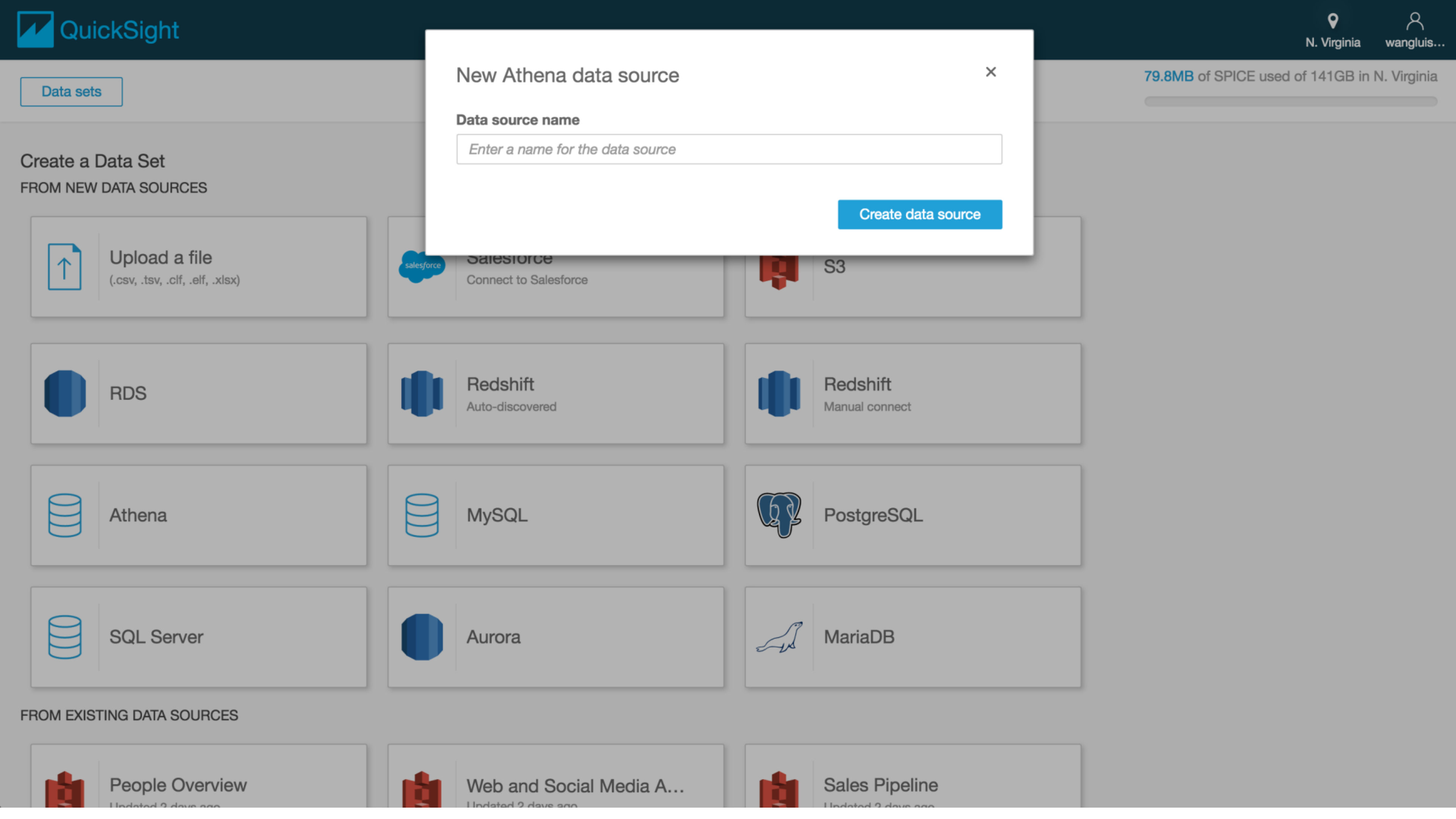
MariaDB

FROM EXISTING DATA SOURCES

People Overview
Updated 2 days ago

Web and Social Media A...
Updated 2 days ago

Sales Pipeline
Updated 2 days ago



QuickSight

N. Virginia

wangluis...

Data sets

79.8MB of SPICE used of 141GB in N. Virginia

New Athena data source

×

Data source name

Enter a name for the data source

Create data source

Create a Data Set

FROM NEW DATA SOURCES

Upload a file
(.csv, .tsv, .clf, .elf, .xlsx)

salesforce

Salesforce
Connect to Salesforce

S3

RDS

Redshift
Auto-discovered

Redshift
Manual connect

Athena

MySQL

PostgreSQL

SQL Server

Aurora

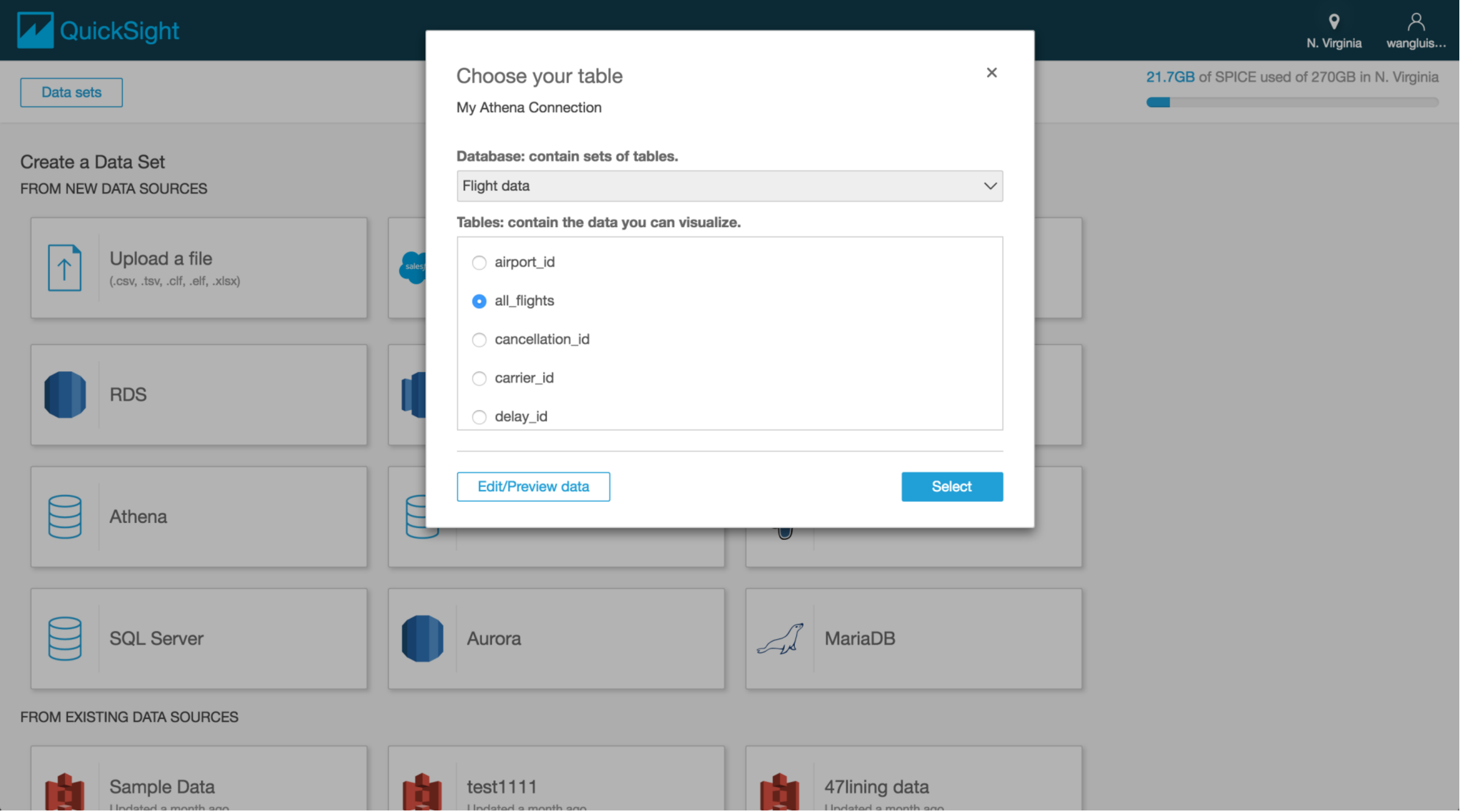
MariaDB

FROM EXISTING DATA SOURCES

People Overview
Updated 2 days ago

Web and Social Media A...
Updated 2 days ago

Sales Pipeline
Updated 2 days ago



Choose your table ×

My Athena Connection

Database: contain sets of tables.

Flight data

Tables: contain the data you can visualize.

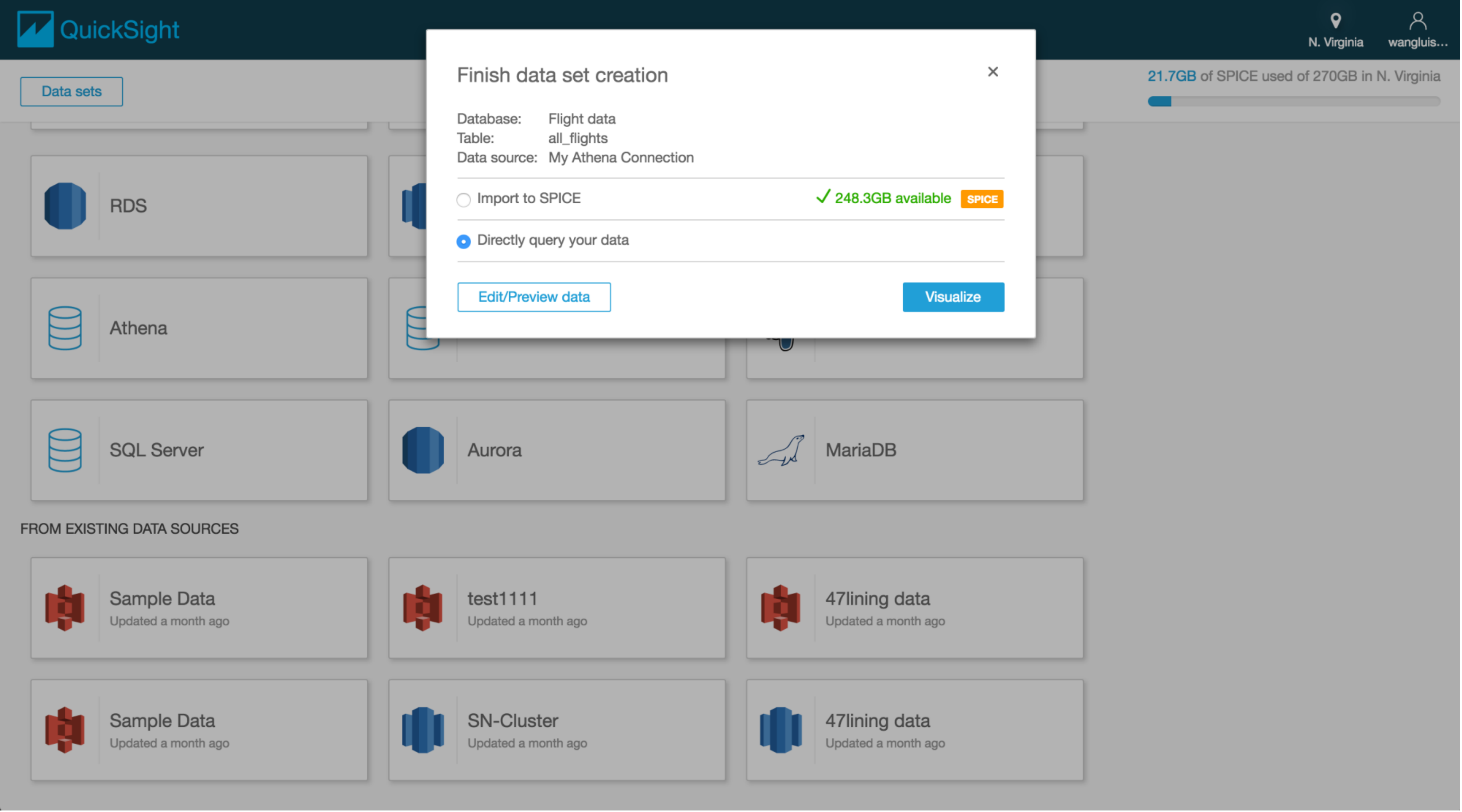
○ airport_id

● all_flights

○ cancellation_id

○ carrier_id

○ delay_id

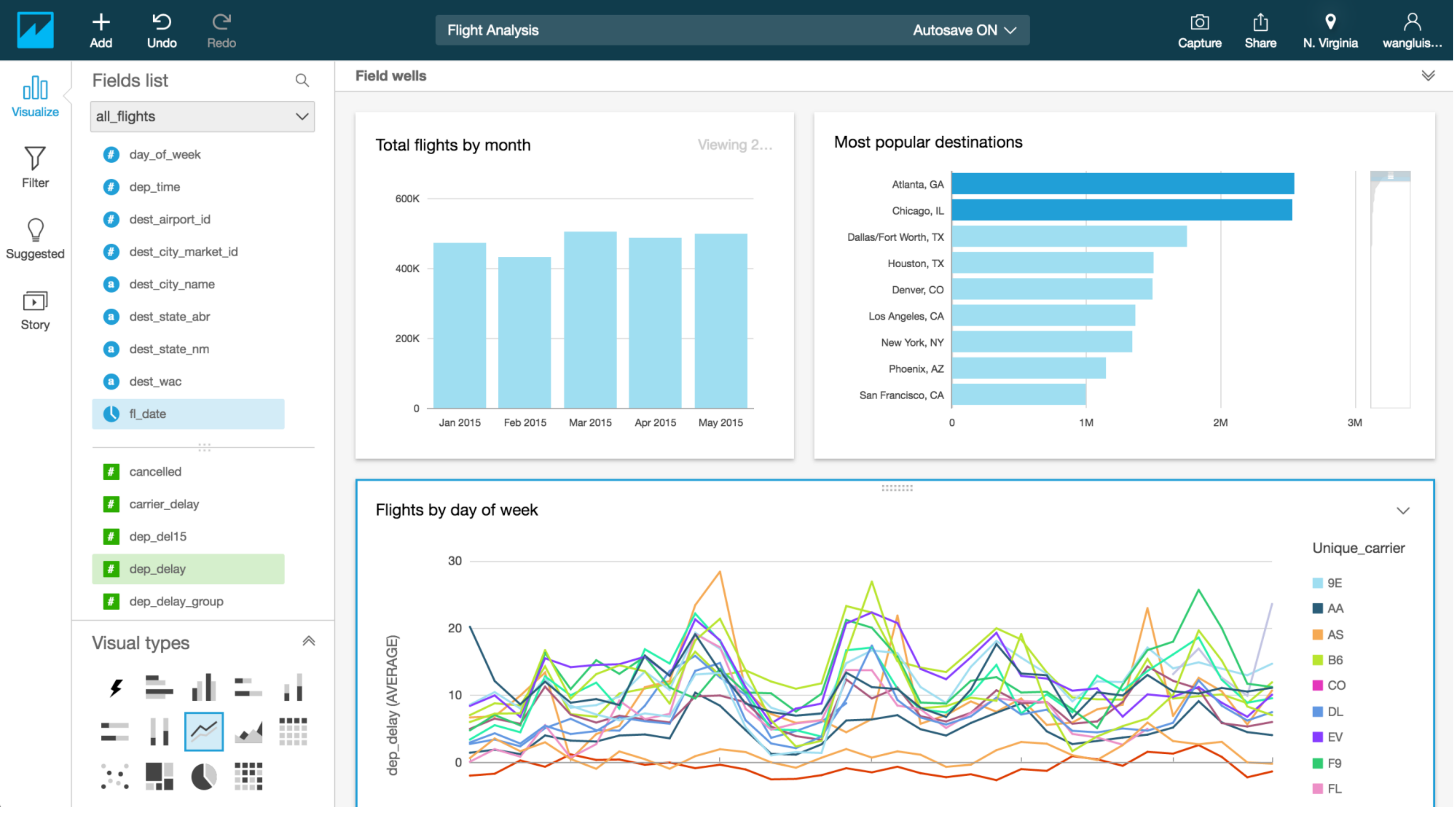


Finish data set creation ×

Database: Flight data
Table: all_flights
Data source: My Athena Connection

○ Import to SPICE ✓ 248.3GB available SPICE

● Directly query your data


Add
Undo
Redo
Flight Analysis
Autosave ON
Capture
Share
N. Virginia
wangluis...
Visualize
Filter
Suggested
Story
Fields list
all_flights
day_of_week
dep_time
dest_airport_id
dest_city_market_id
dest_city_name
dest_state_abr
dest_state_nm
dest_wac
fl_date
cancelled
carrier_delay
dep_del15
dep_delay
dep_delay_group
Visual types
Field wells
Total flights by month
Viewing 2...
600K
400K
200K
0
Jan 2015
Feb 2015
Mar 2015
Apr 2015
May 2015
Most popular destinations
Atlanta, GA
Chicago, IL
Dallas/Fort Worth, TX
Houston, TX
Denver, CO
Los Angeles, CA
New York, NY
Phoenix, AZ
San Francisco, CA
0
1M
2M
3M
Flights by day of week
30
20
10
0
dep_delay (AVERAGE)
Unique_carrier
9E
AA
AS
B6
CO
DL
EV
F9
FL

JDBC also Provides Programmatic Access

```
/* Setup the driver */
Properties info = new Properties();
info.put("user", "AWSAccessKey");
info.put("password", "AWSSecretAccessKey");
info.put("s3_staging_dir", "s3://S3 Bucket Location/");

Class.forName("com.amazonaws.athena.jdbc.AthenaDriver");

Connection connection = DriverManager.getConnection("jdbc:awsathena://athena.us-east-1.amazonaws.com:443/", info);
```

Creating a Table and Executing a Query

```
/* Create a table */
Statement statement = connection.createStatement();
ResultSet queryResults = statement.executeQuery("CREATE EXTERNAL TABLE tableName ( Col1 String ) LOCATION ‘s3://bucket/tableLocation");

/* Execute a Query */
Statement statement = connection.createStatement();
ResultSet queryResults = statement.executeQuery("SELECT * FROM cloudfront_logs");
```

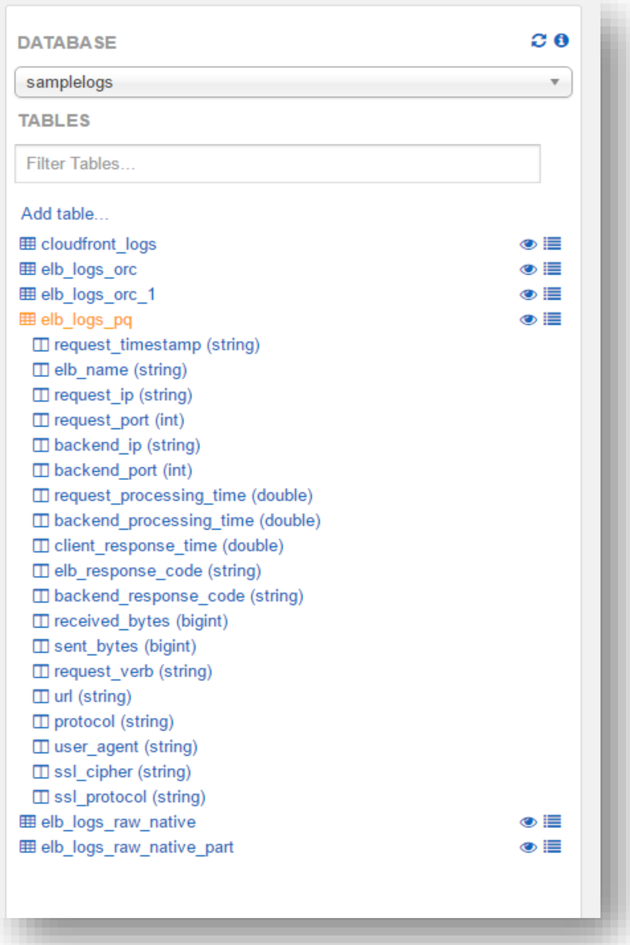
Creating Tables and Querying Data

Creating Tables - Concepts

- Create Table Statements (or DDL) are written in Hive
    - High degree of flexibility
    - Schema on Read
    - Hive is SQL like but allows other concepts such “external tables” and partitioning of data
    - Data formats supported – JSON, TXT, CSV, TSV, Parquet and ORC (via Serdes)
    - Data in stored in Amazon S3
    - Metadata is stored in an a metadata store

Athena’s Internal Metadata Store

- Stores Metadata
  - Table definition, column names, partitions
- Highly available and durable
- Requires no management
- Access via DDL statements
- Similar to a Hive Metastore



DATABASE

samplelogs

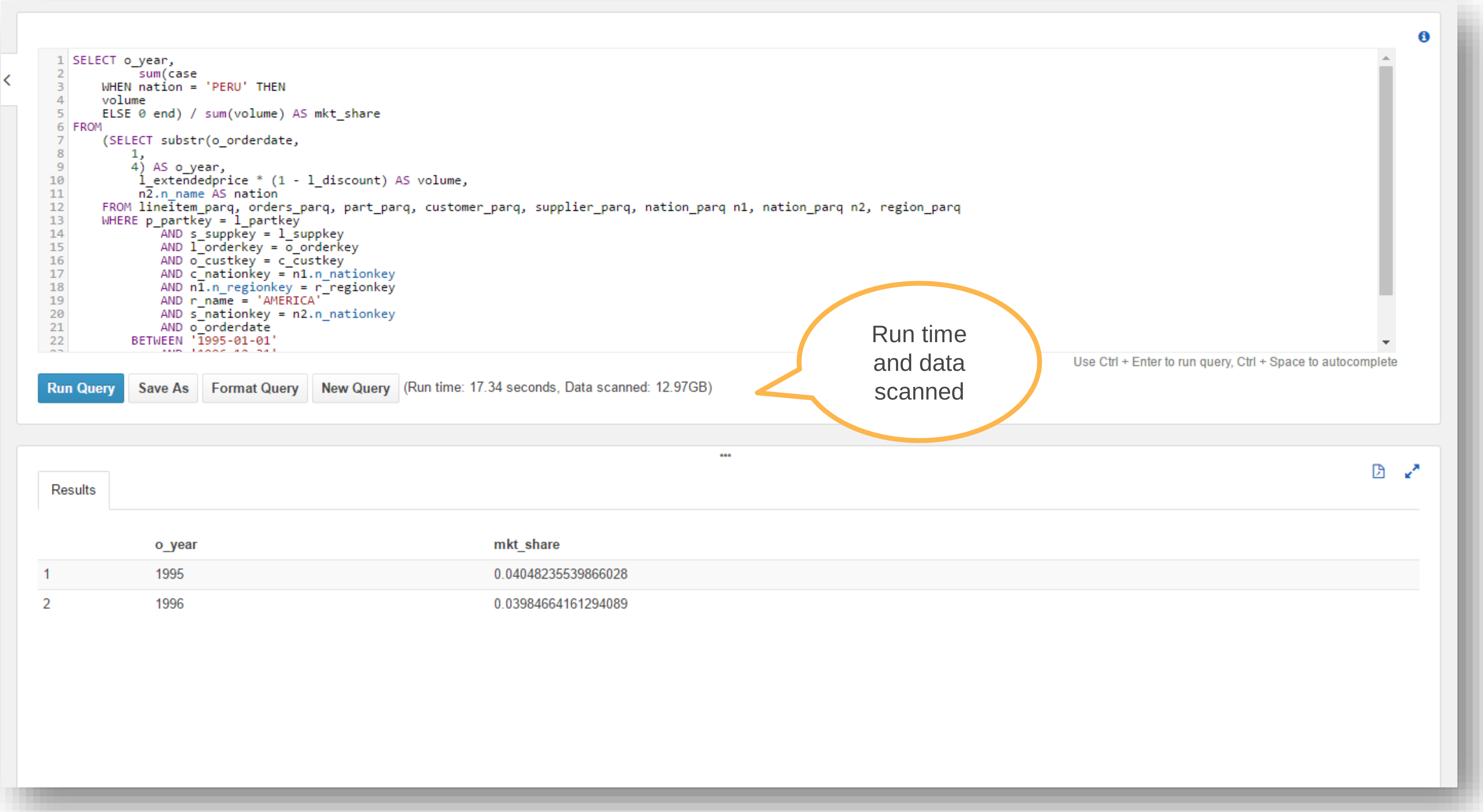
TABLES

Filter Tables...

Add table...

- cloudfront_logs
- elb_logs_orc
- elb_logs_orc_1
- elb_logs_pq
  - request_timestamp (string)
  - elb_name (string)
  - request_ip (string)
  - request_port (int)
  - backend_ip (string)
  - backend_port (int)
  - request_processing_time (double)
  - backend_processing_time (double)
  - client_response_time (double)
  - elb_response_code (string)
  - backend_response_code (string)
  - received_bytes (bigint)
  - sent_bytes (bigint)
  - request_verb (string)
  - url (string)
  - protocol (string)
  - user_agent (string)
  - ssl_cipher (string)
  - ssl_protocol (string)
- elb_logs_raw_native
- elb_logs_raw_native_part

Running Queries is Simple



```
SELECT o_year,
         sum(case
    WHEN nation = 'PERU' THEN
    volume
    ELSE 0 end) / sum(volume) AS mkt_share
FROM
    (SELECT substr(o_orderdate,
        1,
        4) AS o_year,
         l_extendedprice * (1 - l_discount) AS volume,
         n2.n_name AS nation
    FROM lineitem_parq, orders_parq, part_parq, customer_parq, supplier_parq, nation_parq n1, nation_parq n2, region_parq
    WHERE p_partkey = l_partkey
            AND s_suppkey = l_suppkey
            AND l_orderkey = o_orderkey
            AND o_custkey = c_custkey
            AND c_nationkey = n1.n_nationkey
            AND n1.n_regionkey = r_regionkey
            AND r_name = 'AMERICA'
            AND s_nationkey = n2.n_nationkey
            AND o_orderdate
        BETWEEN '1995-01-01'
```

(Run time: 17.34 seconds, Data scanned: 12.97GB)

	o_year	mkt_share
1	1995	0.04048235539866028
2	1996	0.03984664161294089

Run time and data scanned

Apache Parquet and Apache ORC – Columnar Formats

PARQUET

- Columnar format
- Schema segregated into footer
- Column major format
- All data is pushed to the leaf
- Integrated compression and indexes
- Support for predicate pushdown

ORC

- Apache Top level project
- Schema segregated into footer
- Column major with stripes
- Integrated compression, indexes, and stats
- Support for Predicate Pushdown

Converting to ORC and PARQUET

- You can use Hive CTAS to convert data
  - CREATE TABLE new_key_value_store
  - STORED AS PARQUET
  - AS
  - SELECT col_1, col2, col3 FROM noncolumartable
  - SORT BY new_key, key_value_pair;

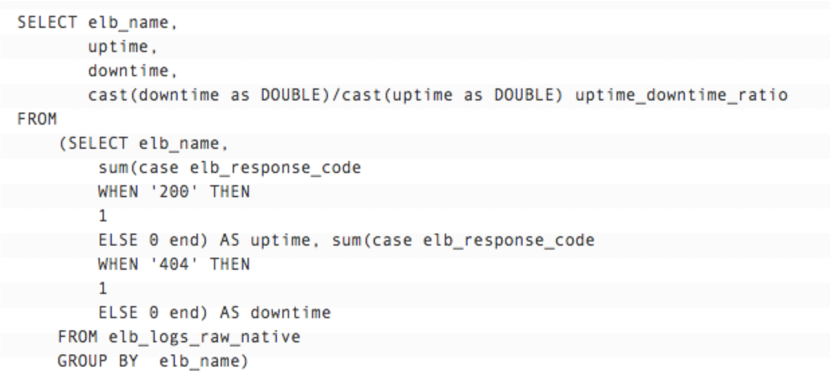
- You can also use Spark to convert the file into PARQUET / ORC

- 20 lines of Pyspark code, running on EMR
  - Converts 1TB of text data into 130 GB of Parquet with snappy conversion
  - Total cost $5

https://github.com/awslabs/aws-big-data-blog/tree/master/aws-blog-spark-parquet-conversion

Pay By the Query - $5/TB Scanned

- Pay by the amount of data scanned per query
- Ways to save costs
  - Compress
  - Convert to Columnar format
  - Use partitioning
- Free: DDL Queries, Failed Queries

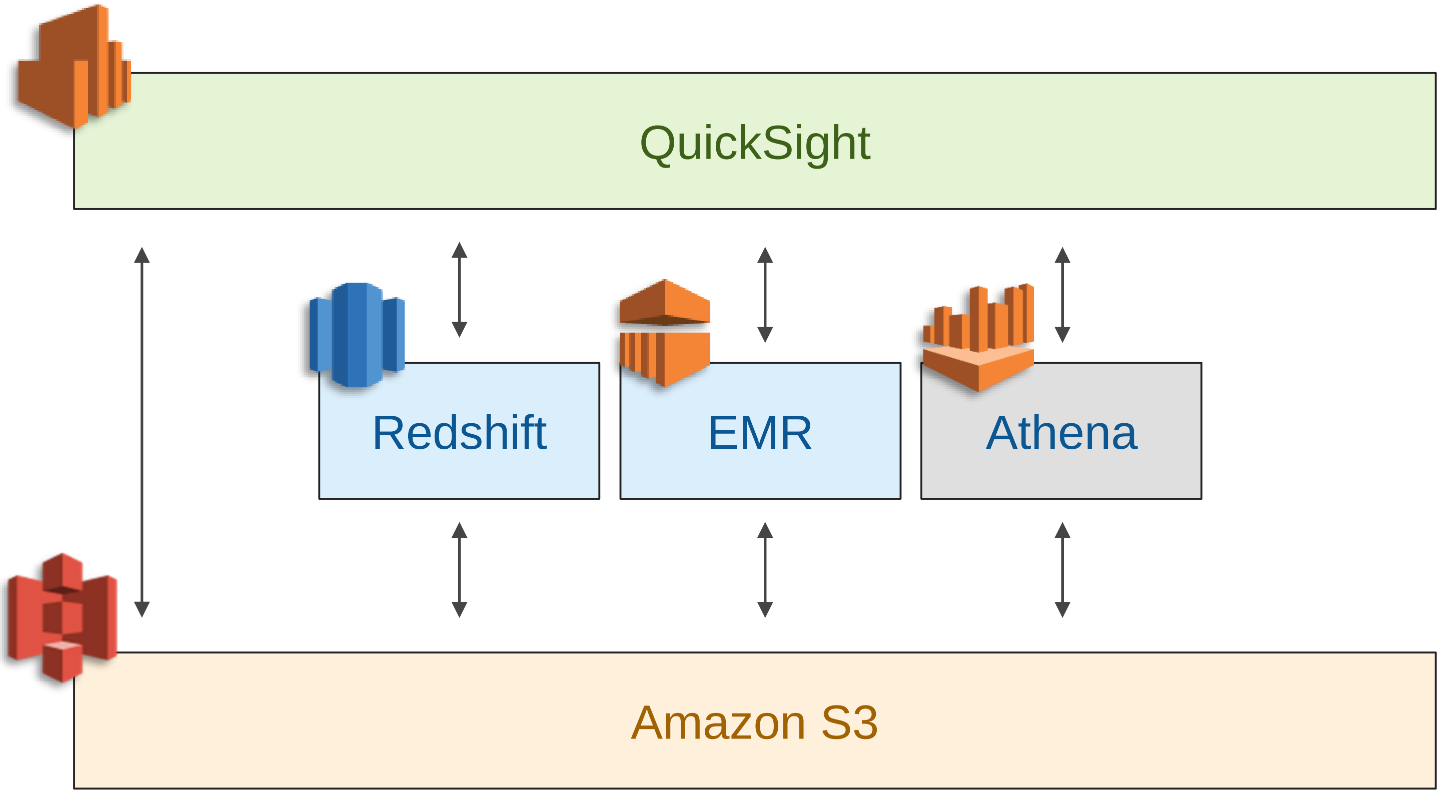


```
SELECT elb_name,
       uptime,
       downtime,
       cast(downtime as DOUBLE)/cast(uptime as DOUBLE) uptime_downtime_ratio
FROM
    (SELECT elb_name,
        sum(case elb_response_code
        WHEN '200' THEN
        1
        ELSE 0 end) AS uptime, sum(case elb_response_code
        WHEN '404' THEN
        1
        ELSE 0 end) AS downtime
    FROM elb_logs_raw_native
    GROUP BY  elb_name)
```

Dataset	**Size on Amazon S3**	**Query Run time**	**Data Scanned**	**Cost**
Logs stored as Text files	1 TB	237 seconds	1.15TB	$5.75
Logs stored in Apache Parquet format*	130 GB	5.13 seconds	2.69 GB	$0.013
Savings	**87% less with Parquet**	**34x faster**	**99% less data scanned**	**99.7% cheaper**

Use Cases

Athena Complements Amazon Redshift & Amazon EMR



Customers Using Athena

LiveIntent

JWPLAYER

mparticle

DataXu®
Data. Insight. Action.

Gunosy

Atlassian

News Corp

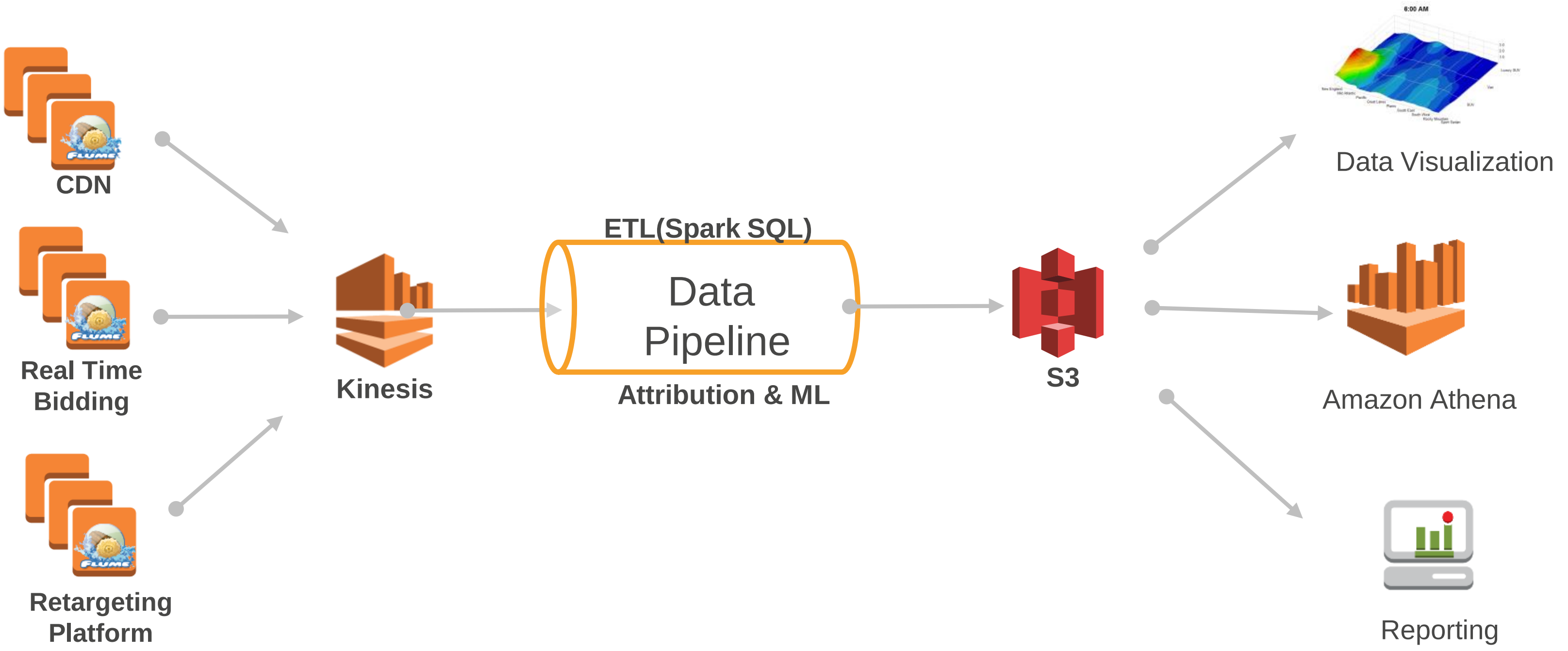
TAXI

Nasdaq

TUNE

INRIX®

DataXu – 180TB of Log Data per Day



CDN

Real Time Bidding

Retargeting Platform

Kinesis

ETL(Spark SQL)

Data Pipeline

Attribution & ML

S3

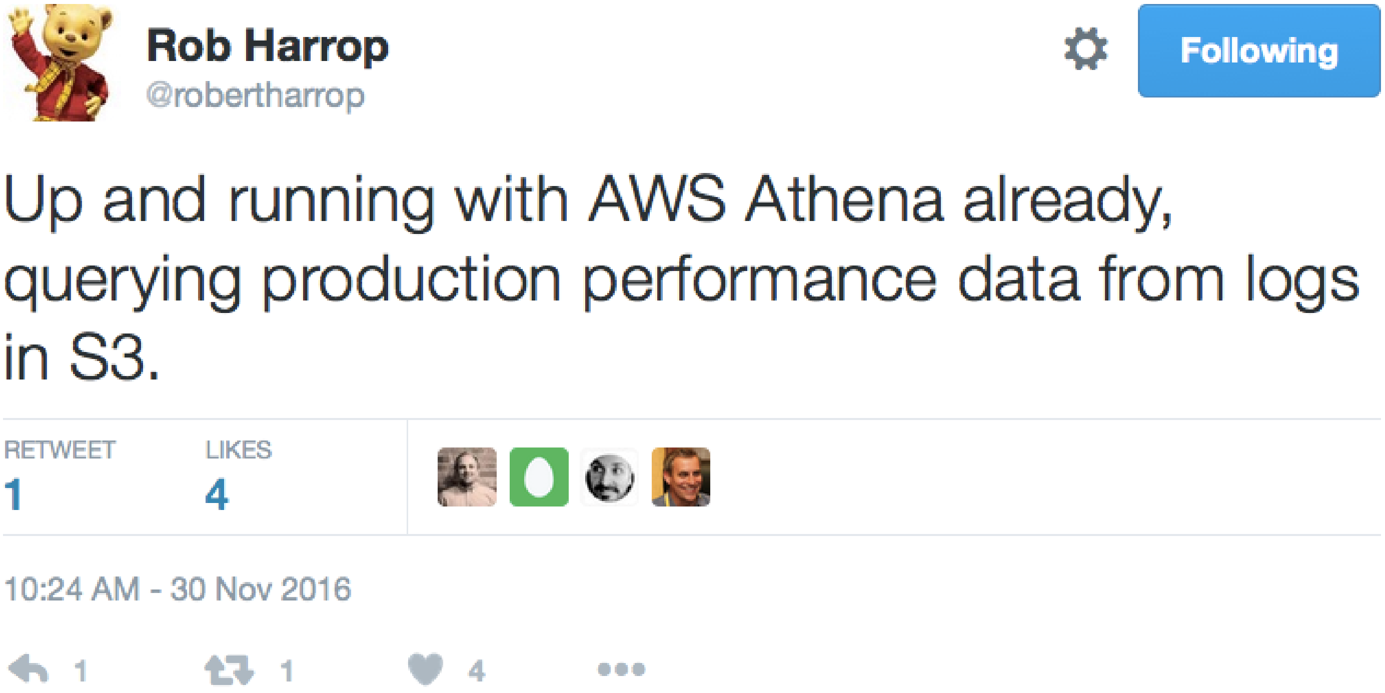
Data Visualization

Amazon Athena

Reporting

AWS

Ecosystem of tools and services



Rob Harrop
@robertharrop

Up and running with AWS Athena already, querying production performance data from logs in S3.

RETWEET 1 LIKES 4

10:24 AM - 30 Nov 2016

Tips and Tricks

Created a Table, but do not see data

- Verify that the input LOCATION is correct on S3
  - Buckets should be specified as s3://name/ or s3://name/subfolder/
  - s3://us-east-1.amazonaws.com/bucket/path will throw an error s3://bucket/path/
- Did the table have partitions ?
  - MSCK Repair Table for
    - *s3://mybucket/athena/inputdata/year=2016/data.csv*
    - *s3://mybucket/athena/inputdata/year=2015/data.csv*
    - *s3://mybucket/athena/inputdata/year=2014/data.csv*
  - Alter Table Add Partition for
    - *s3://mybucket/athena/inputdata/2016/data.csv*
    - *s3://mybucket/athena/inputdata/2015/data.csv*
    - *s3://mybucket/athena/inputdata/2014/data.csv*
  - ALTER TABLE Employee ADD
    - PARTITION (year=2016) LOCATION s3://mybucket/athena/inputdata/2016/'
    - PARTITION (year=2015) LOCATION s3://mybucket/athena/inputdata/2015/'
    - PARTITION (year=2014) LOCATION s3://mybucket/athena/inputdata/2014/'

How did you define your partitions

```
CREATE EXTERNAL TABLE Employee (
        Id INT,
        Name STRING,
        Address STRING
) PARTITIONED BY (year INT)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ','
LOCATION
‘s3://mybucket/athena/inputdata/’;
```

```
CREATE EXTERNAL TABLE Employee (
        Id INT,
        Name STRING,
        Address STRING,
        INT Year
) PARTITIONED BY (year INT)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ','
LOCATION
‘s3://mybucket/athena/inputdata/’;
```

Reading JSON Data

- Make sure you are using the right Serde
  - Native JSON Serde org.apache.hive.hcatalog.data.JsonSerDe
  - OpenX SerDe (org.openx.data.jsonserde.JsonSerDe)
- Make sure JSON record is a single line
- Generate your data in case-insensitive columns
- Provide an option to ignore malformed records

```
CREATE EXTERNAL TABLE json (
    a string,
    b int
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES ( 'ignore.malformed.json' = 'true')
LOCATION 's3://bucket/path/';
```

Access Denied Issues

- Check the IAM Policy
  - Refer to the Getting Started Documentation
- Check the bucket ACL
  - Both the read bucket and write bucket

Table names

- Use backticks if table names begin with an underscore.

  For example:

  ```
  CREATE TABLE myUnderScoreTable (
  `_id` string,
  `_index`string,
  ```

- Table name can only have underscores as special characters

Thank you!