From Data Collection to Actionable Insights in 60 Seconds

Alex Casalboni

Technical Evangelist, AWS

@alex_casalboni

About me

- Software Engineer & Web Developer
- Startupper for 4.5 years
- Serverless Lover & AI Enthusiast
- AWS Customer since 2013

Agenda

1. Data Challenges
2. Columnar Formats
3. Data Lakes vs. Data Warehouses
4. Serverless Analytics
5. Demo time

Data Challenges

Data variety and **data volumes** are increasing rapidly

Multiple **Consumers** and **Applications**

Ingest
Discover
Catalog
Understand
Curate
Find insights

aws

UP TO 75 BILLION EVENTS PER DAY

Monitors 99% EQUITIES & 65% OPTIONS in the US

FINRA

Market Reconstruction Containing TRILLIONS of nodes & edges

Over 20 PETABYTES of storage

Investor PROTECTION

Market INTEGRITY

THINK BIG

Customer Needs Come First

Purpose-built engines

Right tool for the job

aws

Purpose-Built Analytics on AWS

Collect

- Amazon Kinesis Firehose
- AWS Direct Connect
- Amazon Kinesis Streams
- Amazon Snowball

Store

- Amazon S3
- Amazon Glacier
- Amazon Dynamo DB
- Amazon RDS, Amazon Aurora
- Amazon CloudSearch
- Amazon Elasticsearch

Analyze

- Amazon EMR
- Amazon Kinesis Analytics
- Amazon Redshift
- Amazon Athena
- Amazon QuickSight
- Amazon AI

AWS Database Migration Service

AWS Glue

aws

Columnar data

Open-source standards (Apache)

Parquet, ORC, etc.

Optimize Performance

Optimize Costs

Analytical queries

aws

Under the hood

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

Row layout

a1	b1	c1	a2	b2	c2	a3	b3	c3	a4	b4	c4	a5	b5	c5

Column layout

a1	a2	a3	a4	a5	b1	b2	b3	b4	b5	c1	c2	c3	c4	c5

encoded chunk	encoded chunk	encoded chunk

aws

Why it matters

Big Data Analytics

Real-time Analytics

Data exploration

aws

Traditional Data Warehouse

Relational data

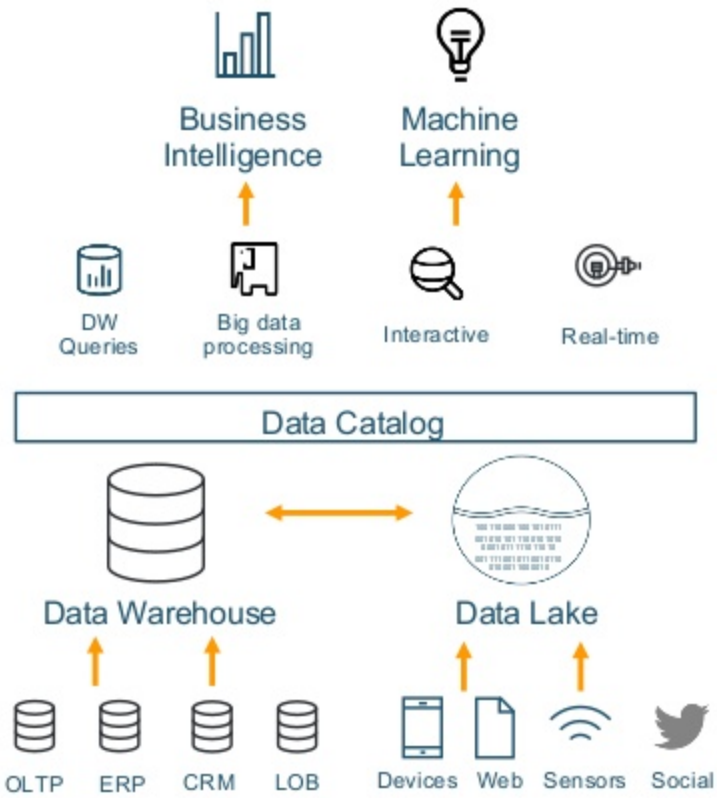
Terabytes to Petabytes scale

Schema defined prior to data load

Operational reporting
and ad-hoc analysis

aws

Data Lakes extend traditional warehouses



Relational and non-relational data

Terabytes to Exabytes scale

Schema defined during analysis (Schema on Read)

Diverse analytical engines to gain insights

Designed for low cost storage and analytics

aws

Data Lakes on AWS

Wide variety of ways to bring data in

Durability and availability at Exabyte scale

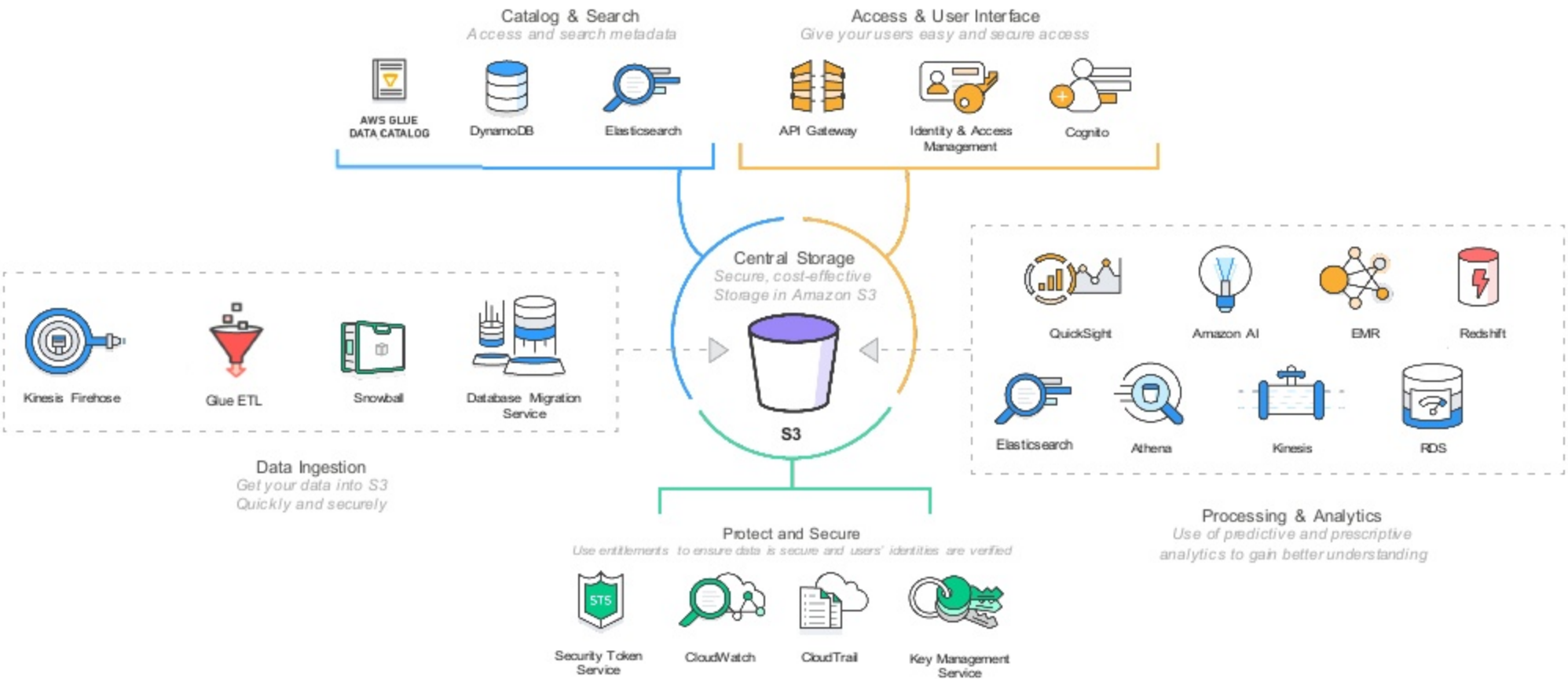
Security, compliance, and audit capabilities

Run any analytics on the same data without movement

Scale storage and compute independently

Store at $0.023 / GB-month
Query for $0.05 / GB scanned

Data Lake Components



Serverless Analytics

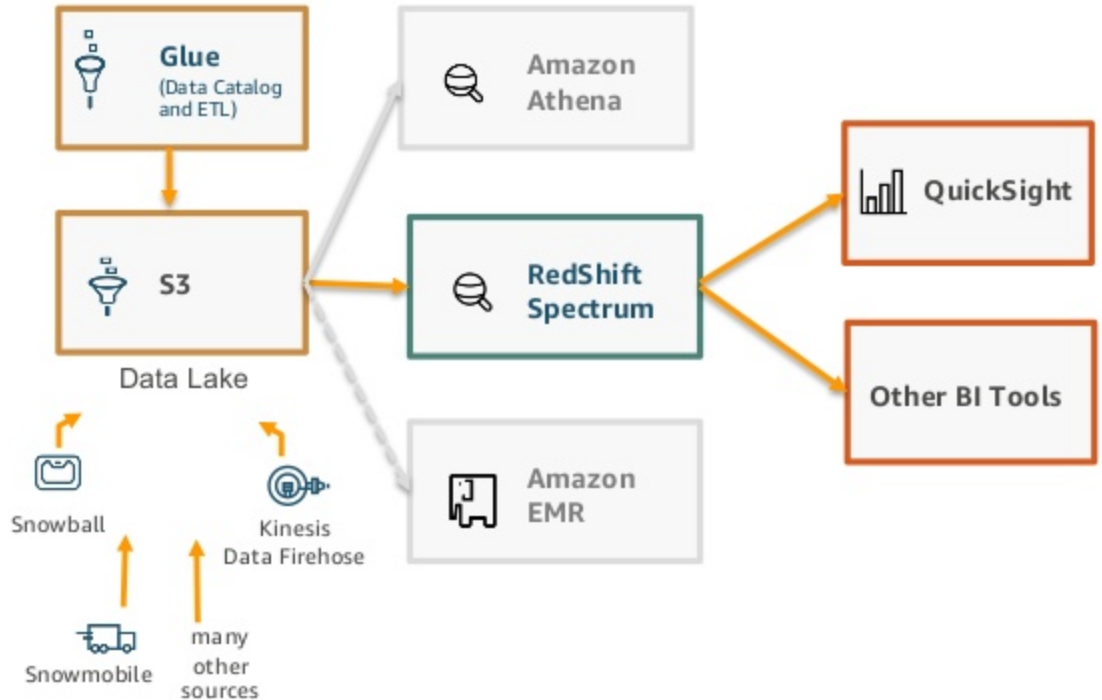
Deliver cost-effective analytic solutions faster

Serverless
Zero infrastructure
Zero administration

Pay only for what you use, not for idle resources

Automatically scales resources with usage

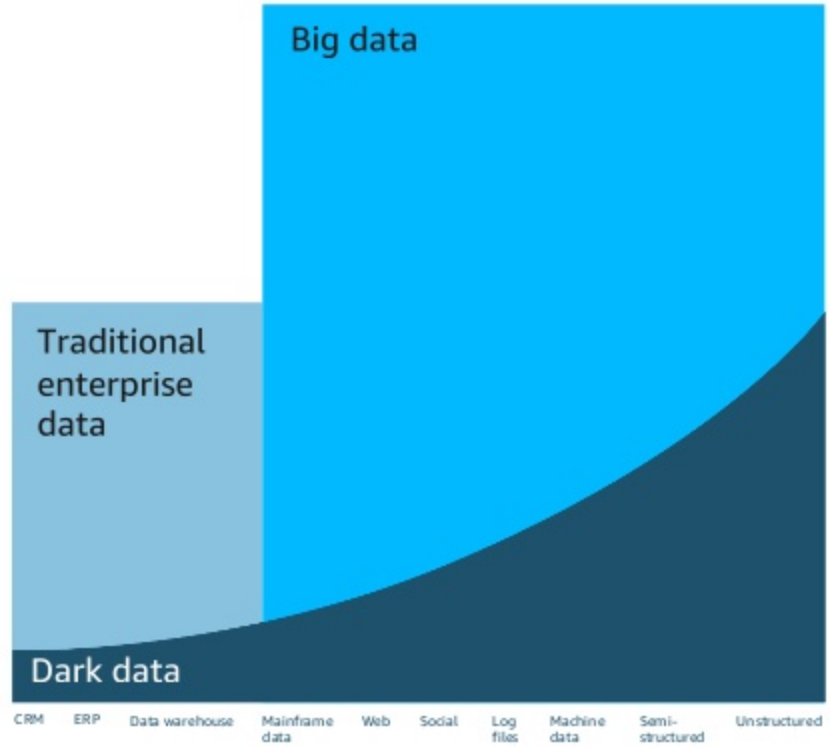
Availability and fault tolerance built in



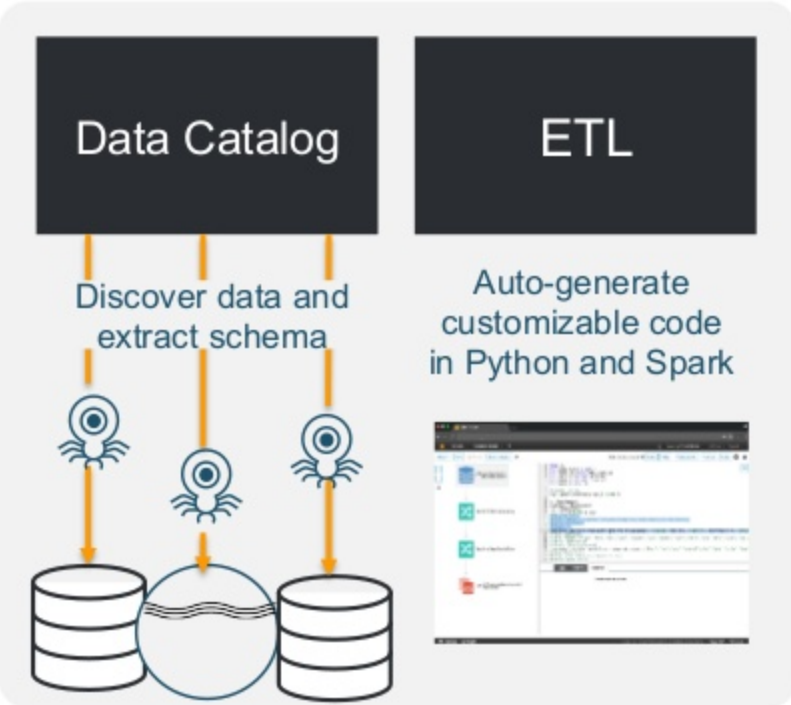
aws

"Dark data are the information assets organizations collect, process, and store during regular business activities, but generally fail to use for other purposes (for example, analytics, business relationships and direct monetizing)."

Gartner



AWS Glue—Serverless Data Catalog & ETL



Automatically discovers data and stores schema

Data is immediately searchable and available for ETL

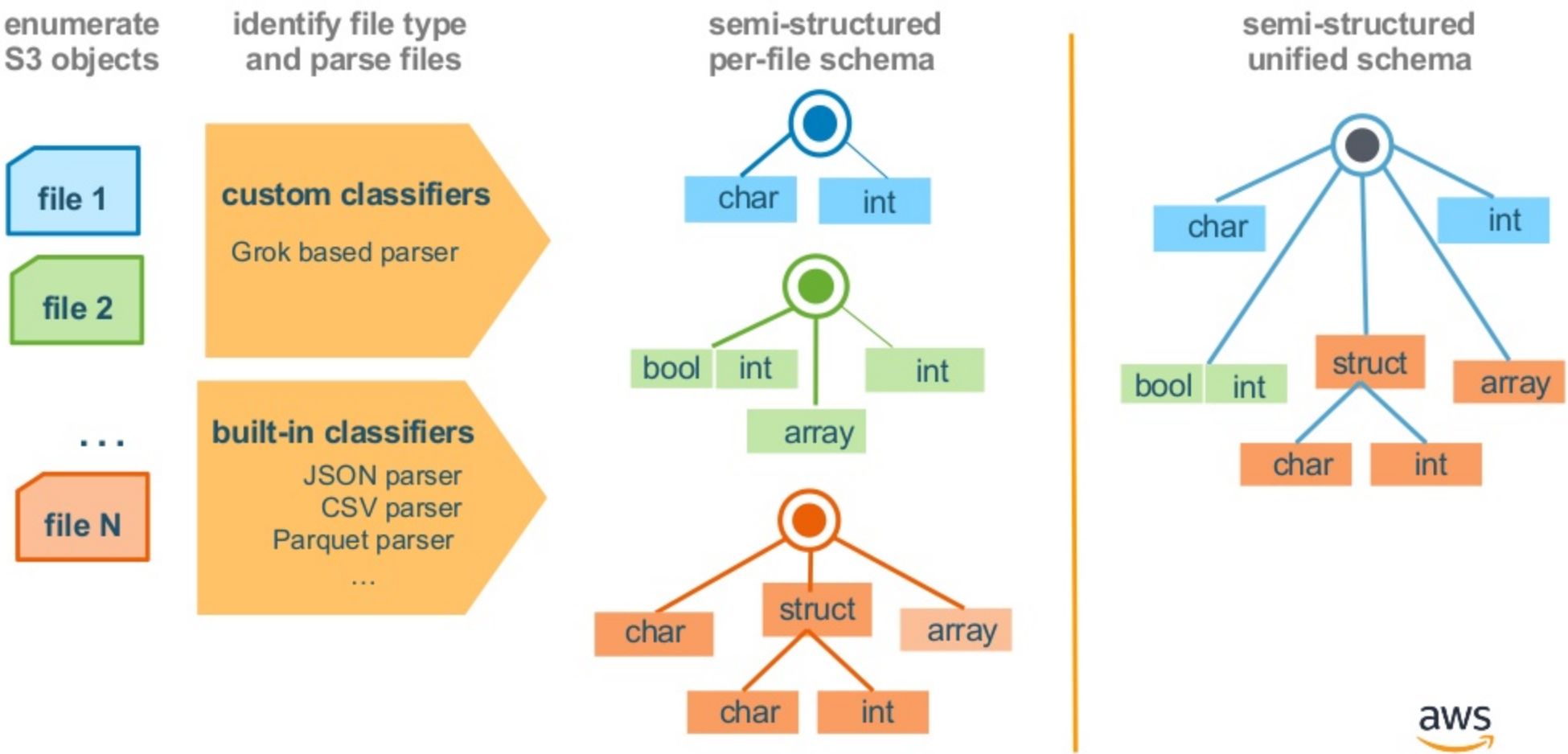
Generates customizable code

Schedules and runs your ETL jobs

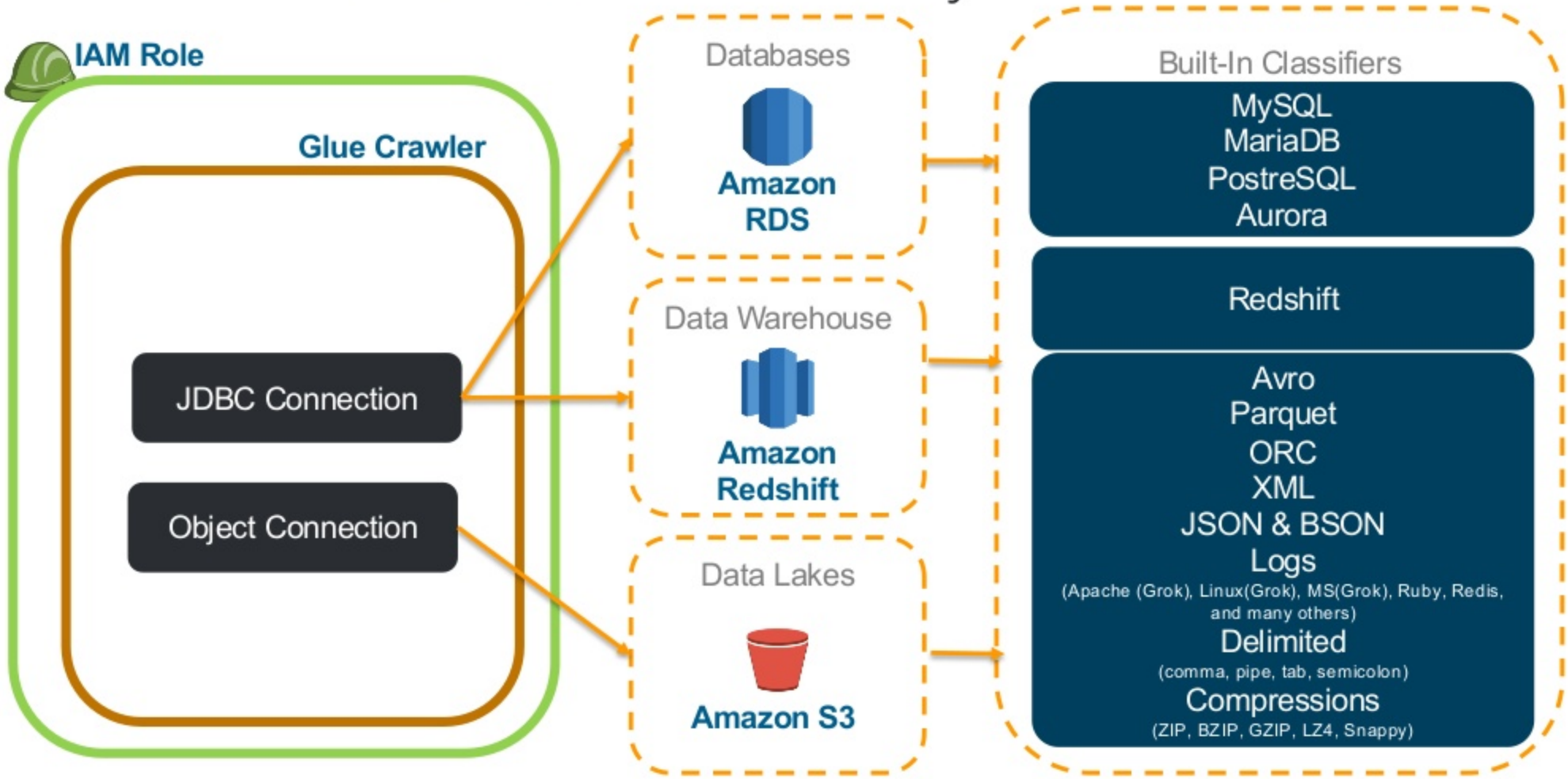
Serverless Model

aws

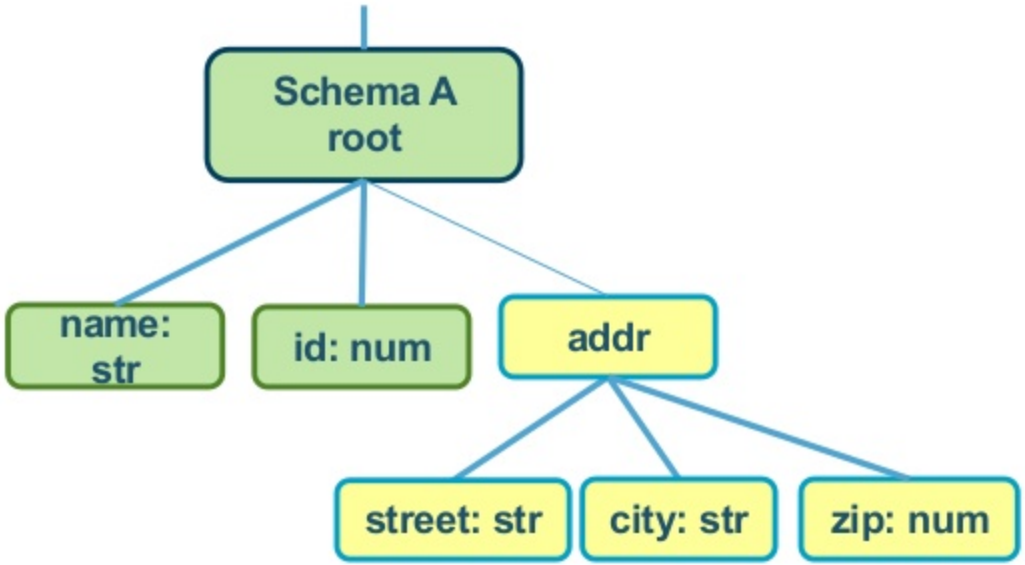
Crawlers: Automatic Schema Inference

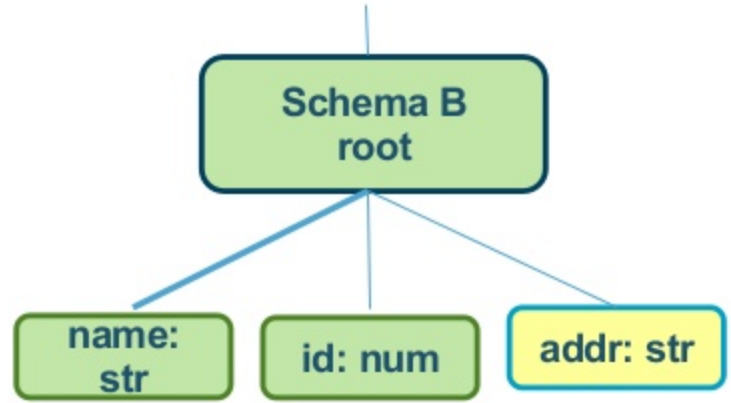


What can Crawlers Classify?



Detecting Schema Similarity





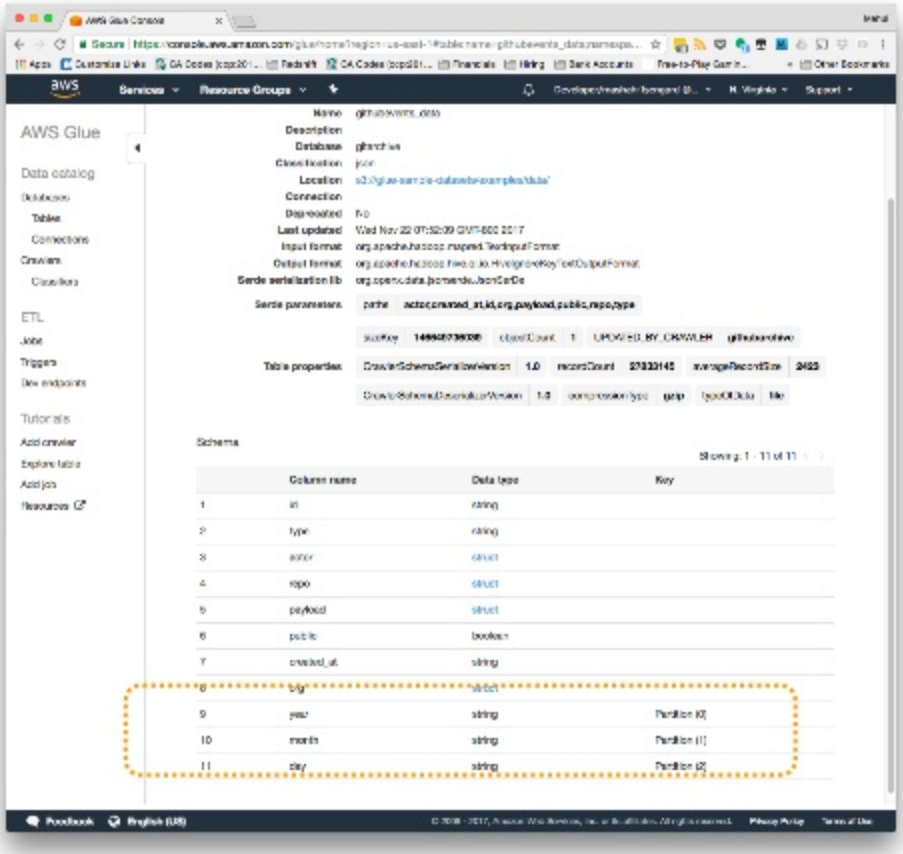
Schema similarity heuristic

- 1 point for matching name
- 1 point for matching data type
- Match when similarity index > 0.7

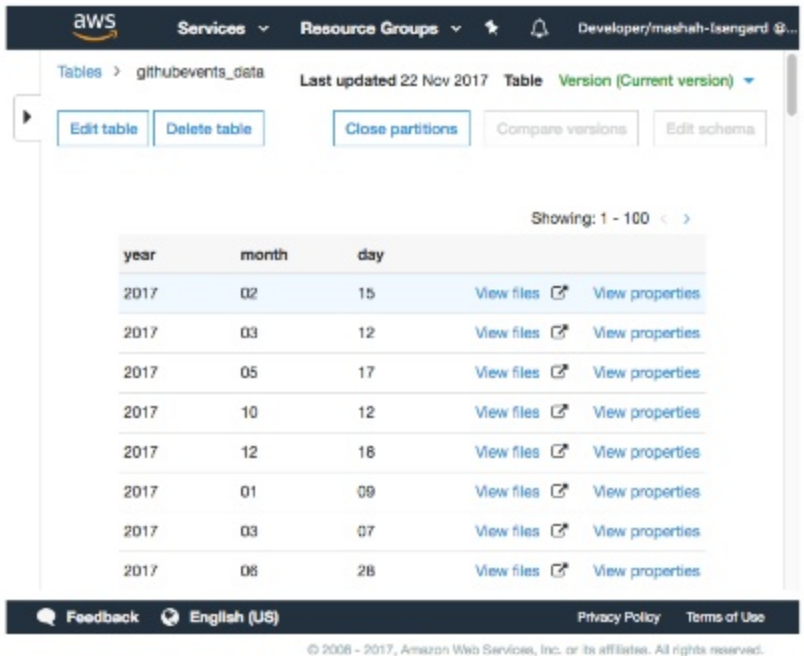
$$\text{sim} = \frac{\text{intersection}}{\min(A,B)} = \frac{7}{8} = .875$$

aws

Automatically Detect Partitions



Available partitions



Automatic Schema Versioning

Automatically update table version as data evolves

Last updated 21 Aug 2017 **Table** Version 1

Name	simpletweets_json
Description	
Database	simpletweets_json
Classification	json
Location	s3://gluesampledata/simpletweets.json
Connection	
Deprecated	No
Last updated	Mon Aug 21 15:23:42 GMT-700 2017
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	org.openx.data.jsonserde.JsonSerDe
Serde parameters	paths entities,id,retweeted,text,user
Table properties	sizeKey 456580 objectCount 1 UPDATED_BY_CRAWLER TestS3Crawler mycustom abc CrawlerSchemaSerializerVersion 1.0 recordCount 1001 averageRecordSize 456 CrawlerSchemaDeserializerVersion 1.0 compressionType none typeOfData file

Change	Column name	Data type	Key
	id	bigint	
	retweeted	boolean	
	text	string	
	user	struct	

Last updated 25 Nov 2017 **Table** Version 2

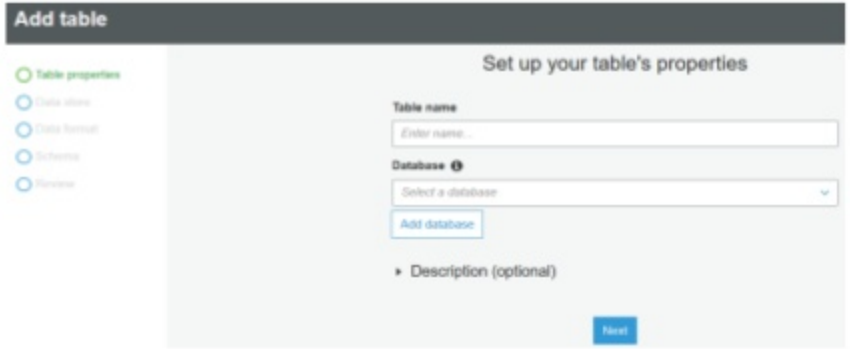
Name	simpletweets_json
Description	
Database	simpletweets_json
Classification	json
Location	s3://gluesampledata/simpletweets.json
Connection	
Deprecated	No
Last updated	Sat Nov 25 12:30:28 GMT-800 2017
Input format	org.apache.hadoop.mapred.TextInputFormat
Output format	org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Serde serialization lib	org.openx.data.jsonserde.JsonSerDe
Serde parameters	paths entities,id,retweeted,text,user
Table properties	sizeKey 456580 objectCount 1 UPDATED_BY_CRAWLER TestS3Crawler mycustom abc CrawlerSchemaSerializerVersion 1.0 recordCount 1001 averageRecordSize 456 CrawlerSchemaDeserializerVersion 1.0 compressionType none typeOfData file

Change	Column name	Data type	Key
	id	bigint	
	retweeted	boolean	
	text	string	
	user	struct	
Added	url	string	

aws

Other Ways of Creating Tables

Create table manually



Run Hive DDL statement

```
CREATE EXTERNAL TABLE IF NOT EXISTS elb_logs_raw_native_part (
  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 )
PARTITIONED BY(year string, month string, day string)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
        'serialization.format' = '1','input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^ ]*):([0-9]*) ([.0-9]*) ([.0-
LOCATION 's3://athena-examples/elb/raw/';
```

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Run Query | Save As | Format Query | New Query (Run time: 2.03 seconds, Data scanned: 0KB)

Call Glue's CreateTable API

Import from Apache Hive Metastore

Amazon Redshift - Data Warehousing

Fast, powerful, simple, and fully managed data warehouse at 1/10 the cost

Massively parallel, scale from gigabytes to petabytes

Fast at any scale

Columnar storage technology to improve I/O efficiency and scale query performance

Open file formats

Analyze optimized data formats on the latest SSD, and all open data formats in Amazon S3

Secure

Audit everything; encrypt data end-to-end; extensive certification and compliance

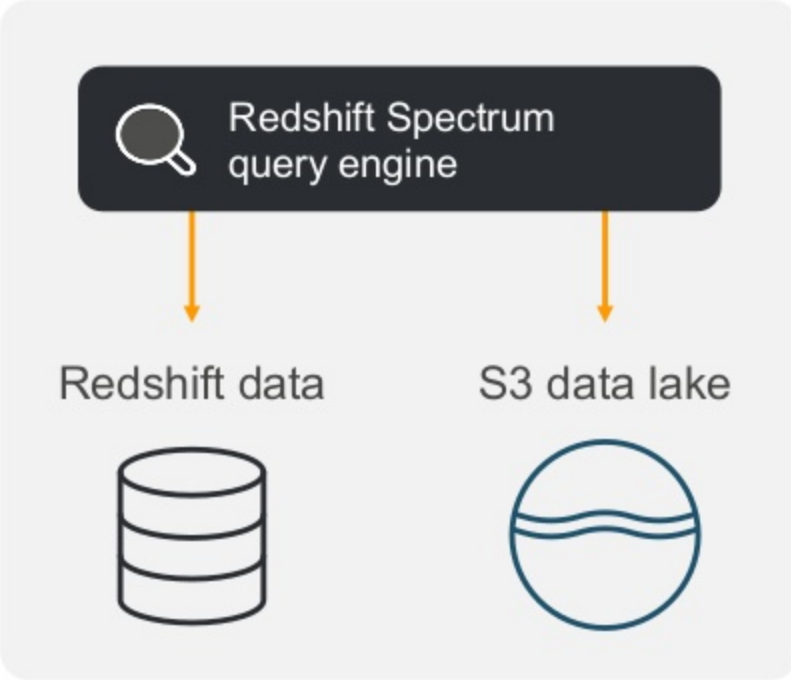
Inexpensive

As low as $1,000 per terabyte per year, 1/10th the cost of traditional data warehouse solutions; Start at $0.25 per hour

aws

Amazon Redshift Spectrum

Extend the data warehouse to exabytes of data in an S3 data lake



Exabyte Redshift SQL queries against S3

Join data across Redshift and S3

Scale compute and storage separately

Stable query performance and unlimited concurrency

CSV, ORC, Grok, Avro, & Parquet data formats

Pay only for the amount of data scanned

aws

Redshift Spectrum

Query your data lake

```
SELECT COUNT(*)
FROM S3.EXT_TABLE
GROUP BY ...
```

JDBC/ODBC

Amazon Redshift

Redshift Spectrum

Scale-out serverless compute

1 2 3 4 ... N

Amazon S3

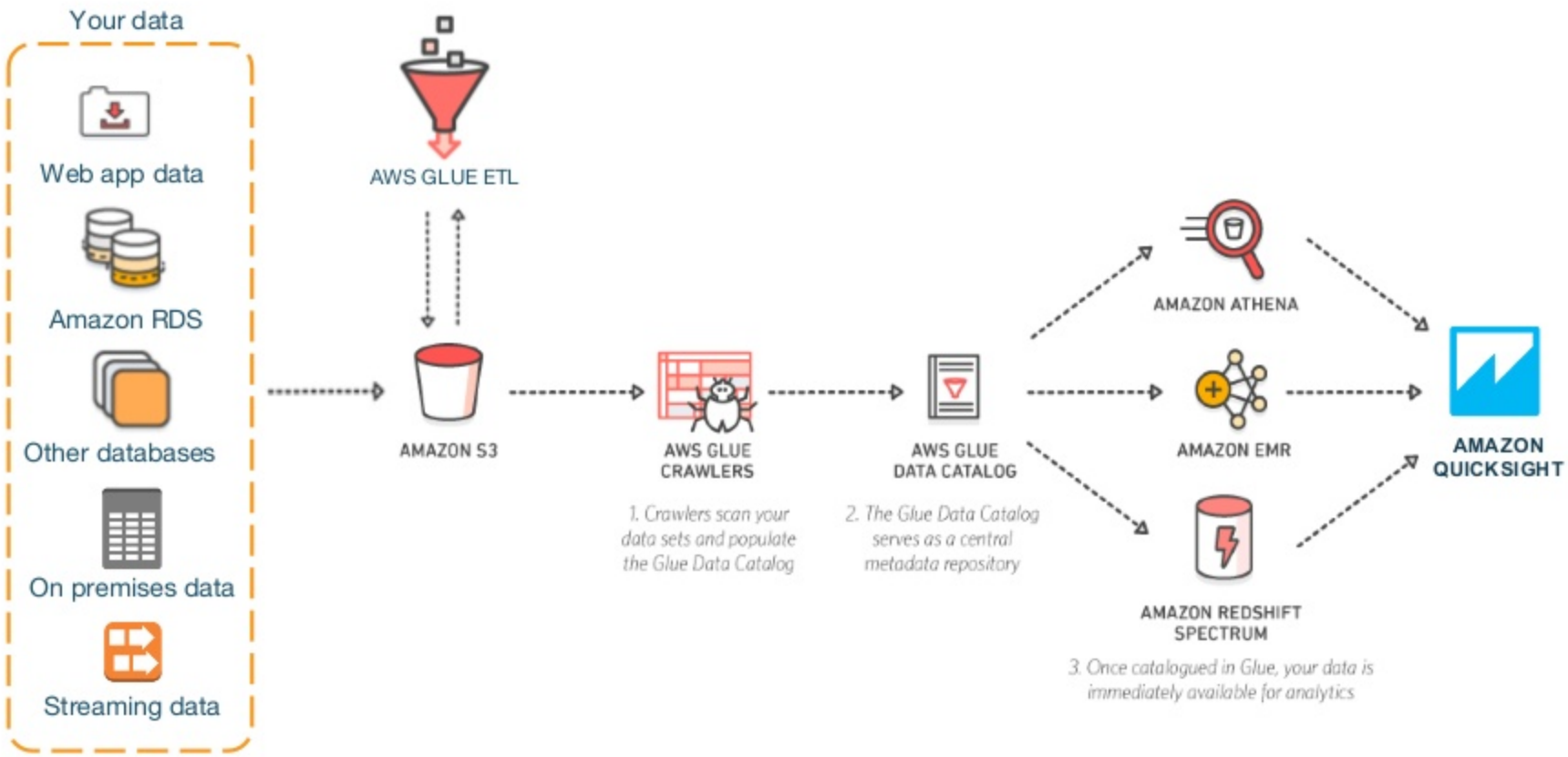
Exabyte-scale object storage

AWS Glue

Data Catalog

aws

Data Lake on Amazon S3 with AWS Glue



Demo Time

```sql
SELECT name, avg(value) as average
FROM table
WHERE action = 'refill'
GROUP BY name;
```

Uncompressed

Compressed (-94%)

Parquet (-70%)

Partitioned (-70%)

Overall 99.5% improvement!

aws

```
SELECT count(*)
FROM table;
```

Uncompressed

Compressed (-94%)

Parquet (-100%)

Partitioned (-100%)

Overall 100% improvement!

aws

```sql
SELECT
  name,
  (
    SELECT count(value)
    FROM table
    WHERE name=t1.name and action='charge'
  ) as charges,
  (
    SELECT count(value)
    FROM table
    WHERE name=t1.name and action='refill'
  ) as refills
FROM table as t1
GROUP BY name;
```

Uncompressed

Compressed (-94%)

Parquet (-72%)

Partitioned (-100%)

Overall 100% improvement!

aws

Amazon QuickSight

aws

Additional Resources

Kinesis Data Generator (KDG)
github.com/awslabs/amazon-kinesis-data-generator

Serverless Data Pipeline powered by AWS SAM
github.com/alexcasalboni/serverless-data-pipeline-sam

AWS Big Data Blog
aws.amazon.com/blogs/big-data

Did We Scan Your Badge?

Remember to opt-in to AWS communications and you will receive a post-event email with a link to:

- **AWS Developer Workshop Slides**
- **$200 in AWS Credits**

Thank you!

Alex Casalboni

Technical Evangelist, AWS

@alex_casalboni