How to Build a Data Lake on Amazon S3 & Amazon Glacier

PD Dutta, Sr. Product Manager, Amazon S3

February 1, 2018

What to expect?

- Defining the AWS data lake
- Why Amazon S3 and Amazon Glacier for your data lake?
- Data cataloging
- Security, performance, and analytics best practices
- Example use case

Defining the AWS data lake

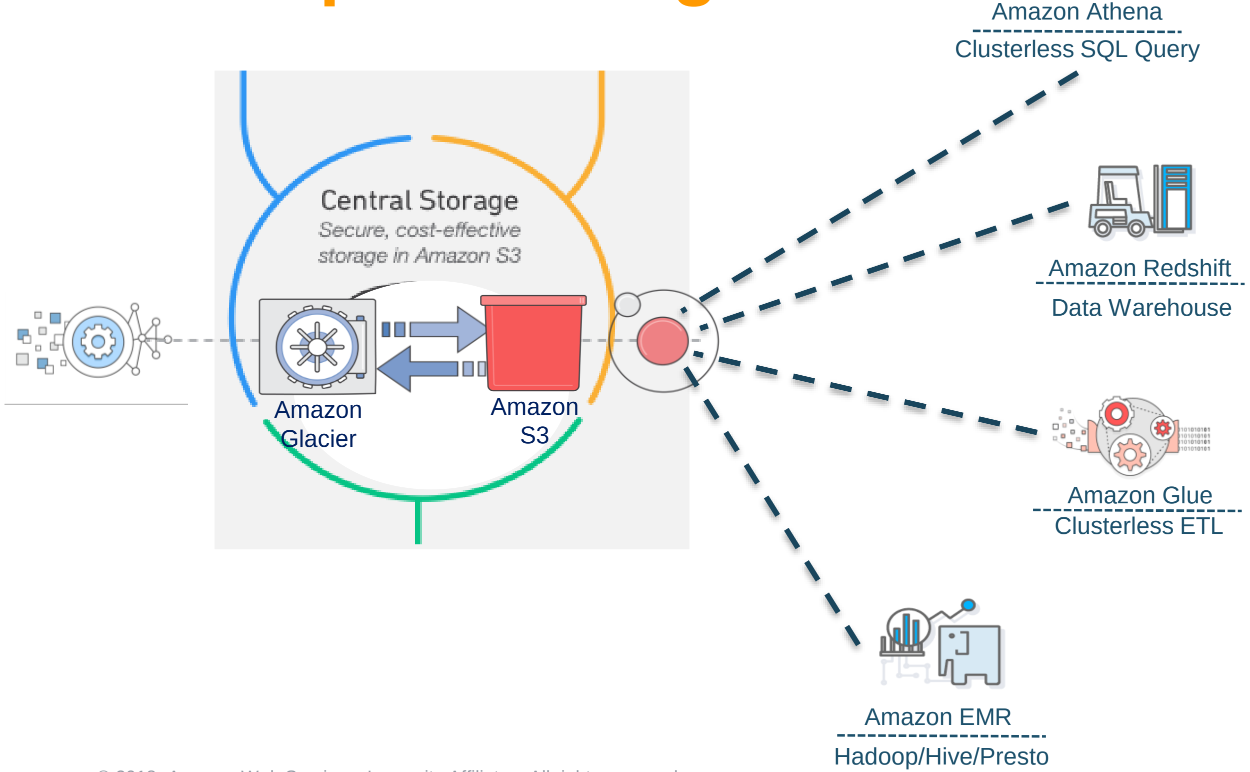
Data lake is an architecture with a virtually limitless centralized storage platform capable of categorization, processing, analysis, and consumption of heterogeneous data sets

Key data lake attributes

- Rapid ingest and transformation
- Decoupled storage and compute
- Secure multi-tenancy
- Query in place
- Schema on read
- Future proofing the data

What can you do with a data lake?

Batch processing



BI & Visualization

Amazon QuickSight

looker

tableau SOFTWARE

kibana

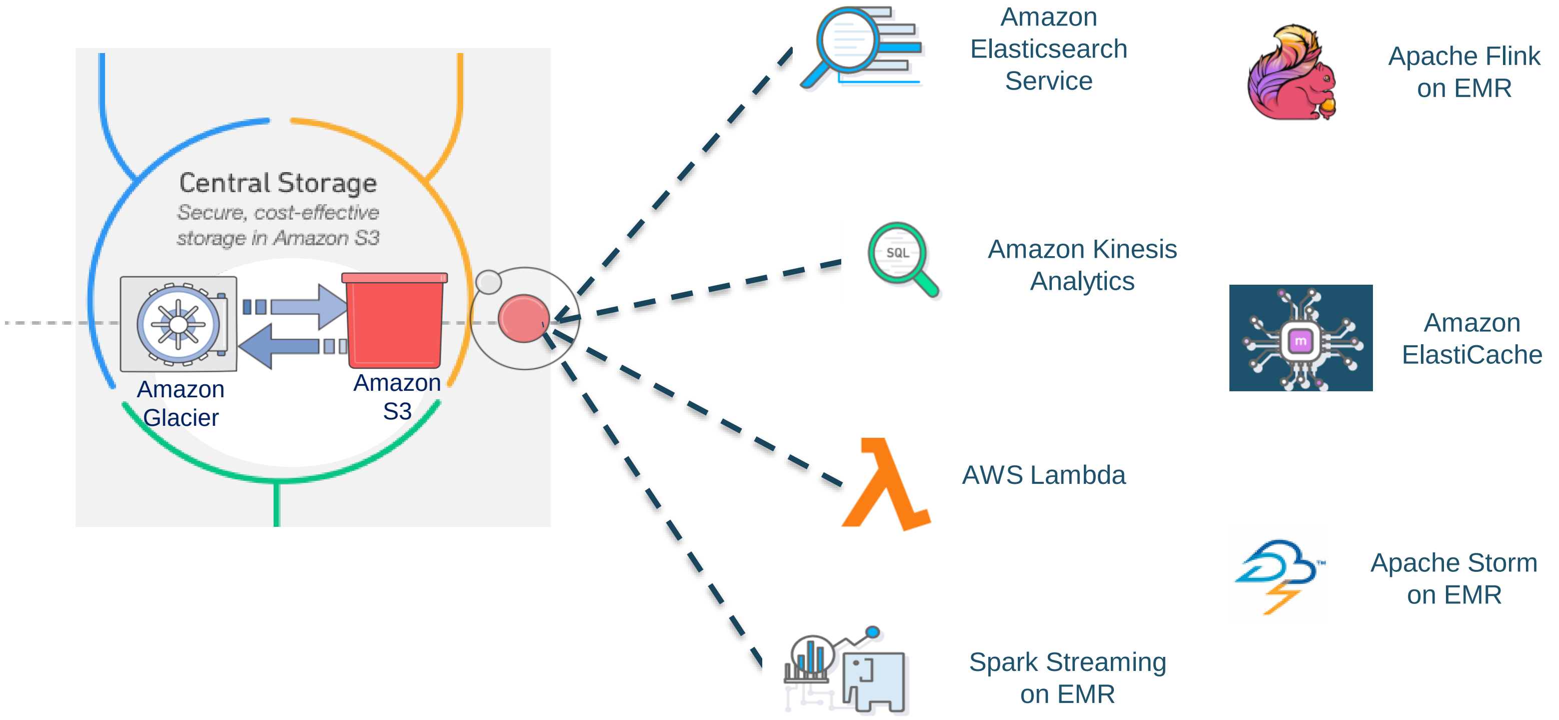
COGNOS

SAP Business Objects

MicroStrategy

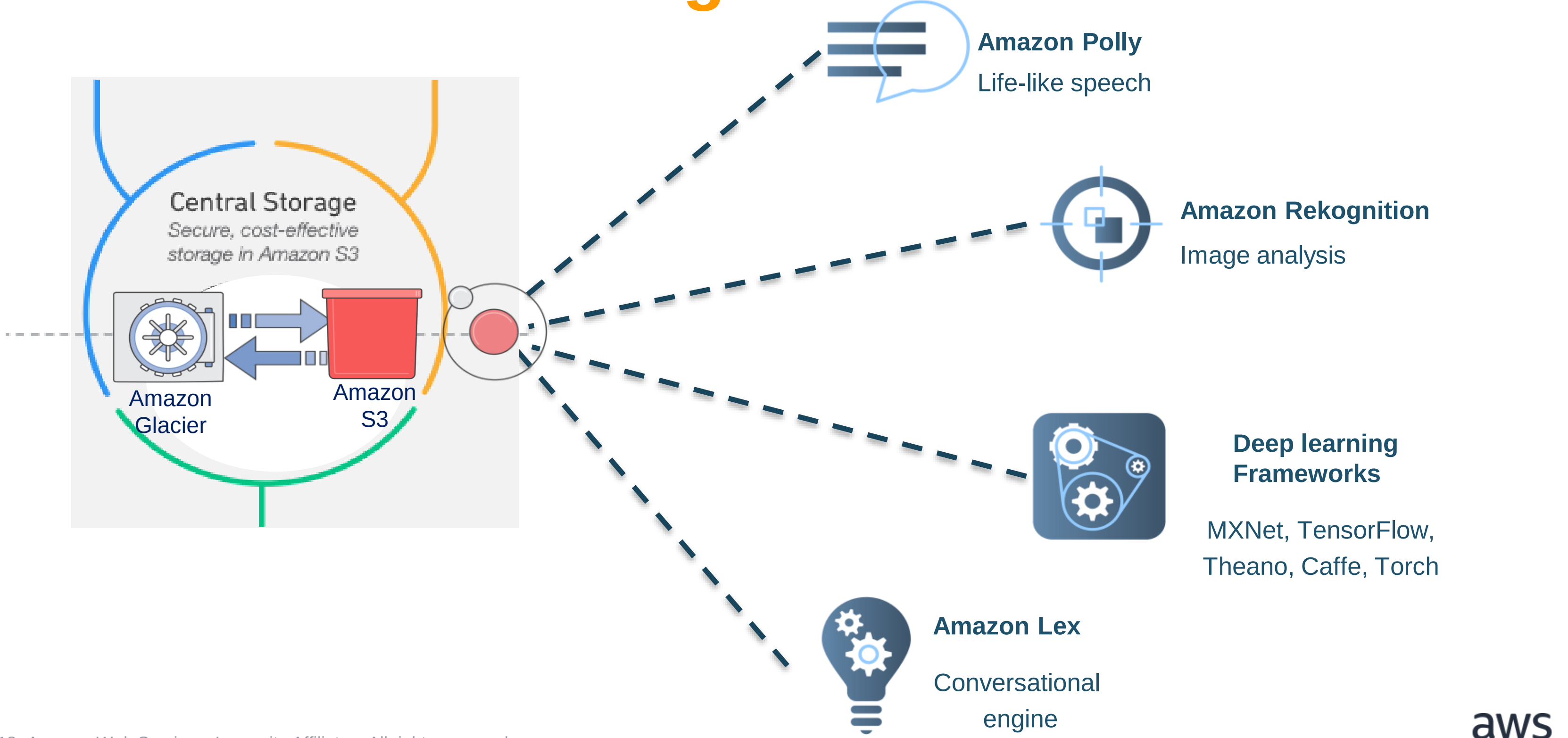
What can you do with a data lake?

Streaming and real-time analytics



What can you do with a data lake?

AI and machine learning



Benefits of Amazon S3 & Amazon Glacier

Durable, Available, & Scalable

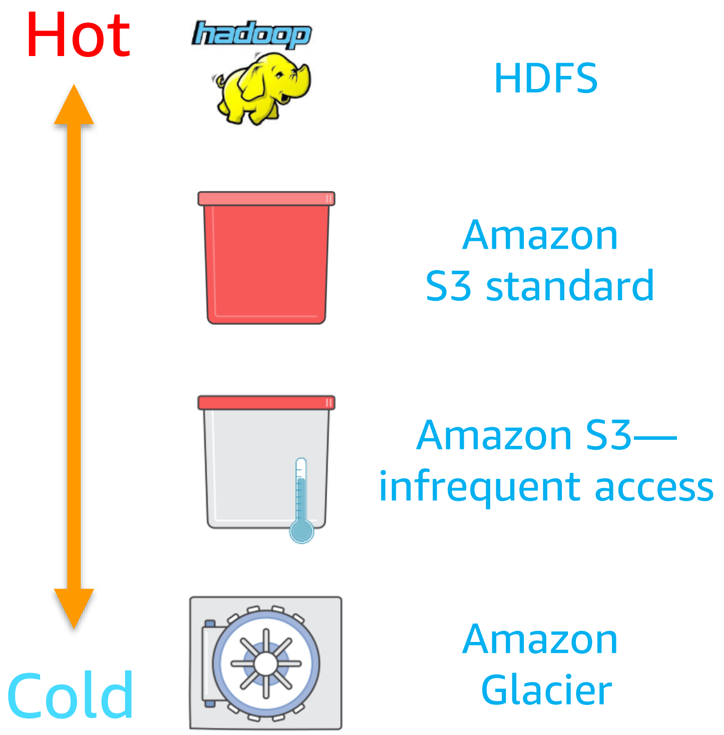
Security & Compliance

Query In Place

Flexible Management

Ecosystem

Optimize costs with data tiering



- ✓ Use EMR/Hadoop with local HDFS for hottest data sets
- ✓ Store cooler data in S3 and Glacier to reduce costs
- ✓ Use S3 Analytics to optimize tiering strategy

S3 Analytics

Multiple data lake ingestion methods

AWS Snowball and AWS Snowmobile

- PB-scale migration

Amazon Kinesis Firehose

- Ingest device streams
- Transform and store on Amazon S3

AWS Storage Gateway

- Migrate legacy files

AWS Direct Connect

- On-premises integration

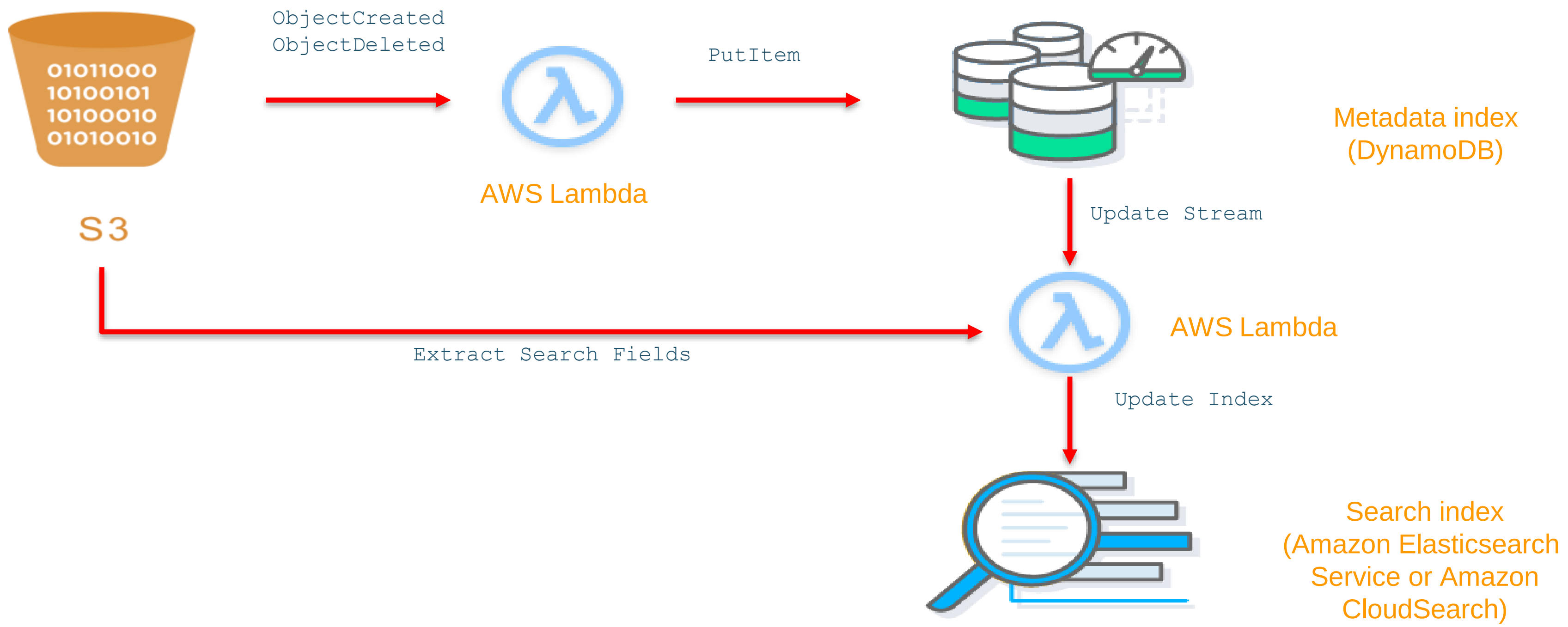
Native/ISV Connectors

- Ecosystem integration

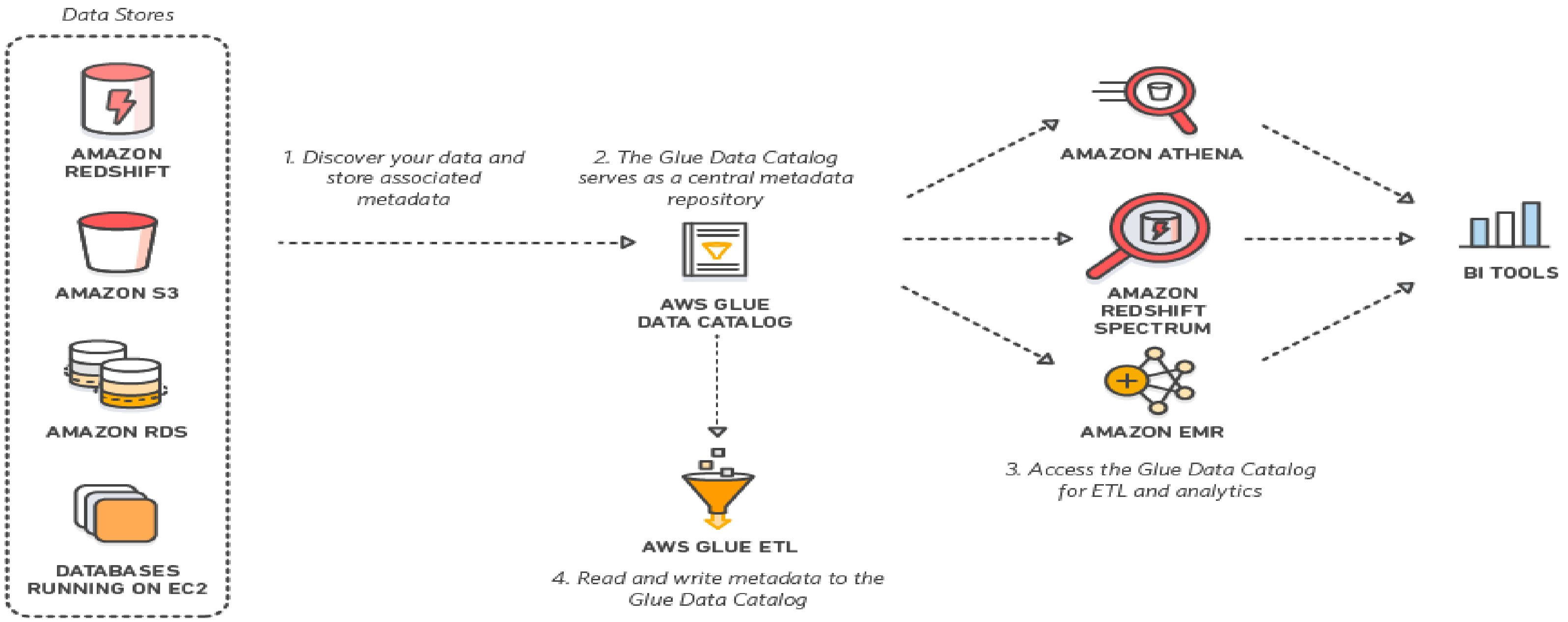
Amazon S3 Transfer Acceleration

- Long-distance data transfer

Catalog your S3 data



AWS Glue analytics data catalog


Data Stores
AMAZON REDSHIFT
AMAZON S3
AMAZON RDS
DATABASES RUNNING ON EC2
1. Discover your data and store associated metadata
2. The Glue Data Catalog serves as a central metadata repository
AWS GLUE DATA CATALOG
AWS GLUE ETL
4. Read and write metadata to the Glue Data Catalog
AMAZON ATHENA
AMAZON REDSHIFT SPECTRUM
AMAZON EMR
3. Access the Glue Data Catalog for ETL and analytics
BI TOOLS

AWS Glue analytics data catalog

Manage table metadata through a Hive metastore API or Hive SQL. Supported by tools like Hive, Presto, Spark, etc.

We added a few extensions:

- **Search** over metadata for data discovery
- **Connection info** JDBC URLs, credentials
- **Classification** for identifying and parsing files
- **Versioning** of table metadata as schemas evolve and other metadata are updated

Populate using Hive DDL, bulk import, or automatically through **crawlers**

Populating the AWS Glue data catalog

Crawlers automatically build your data catalog and keep it in sync

- Automatically discover new data, extracts schema definitions
    - Detect schema changes and version tables
    - Detect Hive style partitions on Amazon S3
- Built-in classifiers for popular types; custom classifiers using Grok expressions
- Run via Lambda triggers or scheduled; serverless—only pay when crawler runs

Securing your data on Amazon S3

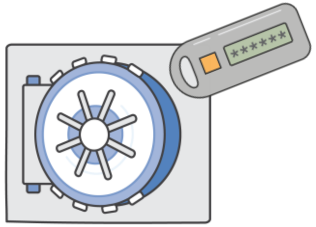
AWS data lake security entitlements

Encryption

- Default encryption New
- Server-side encryption
- Client-side encryption
- SSL endpoints
- Encryption status in inventory New
- CRR with KMS New

Identity and access

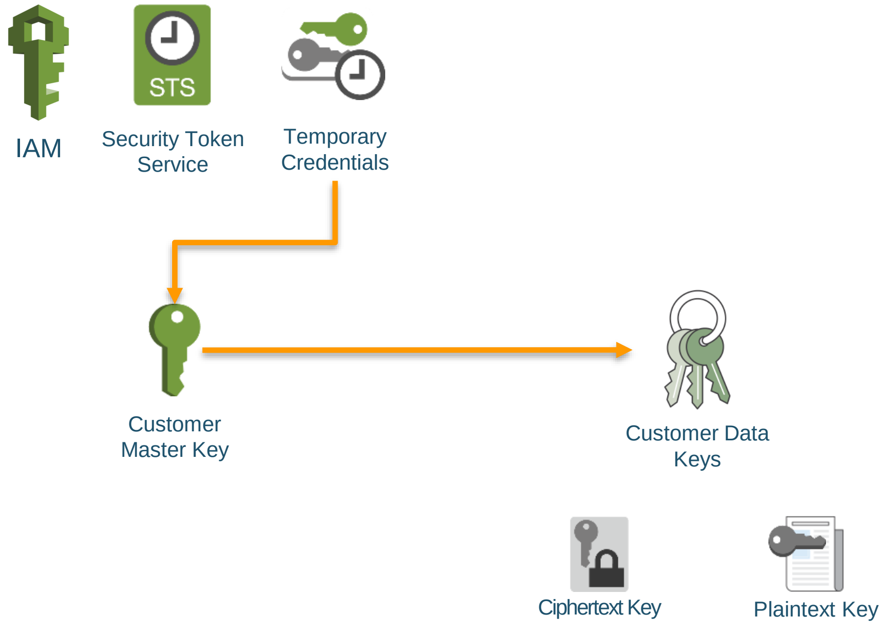
- Amazon Macie New
- Permission checks New
- AWS Config Rules New
- IAM & bucket policies
- Access control lists



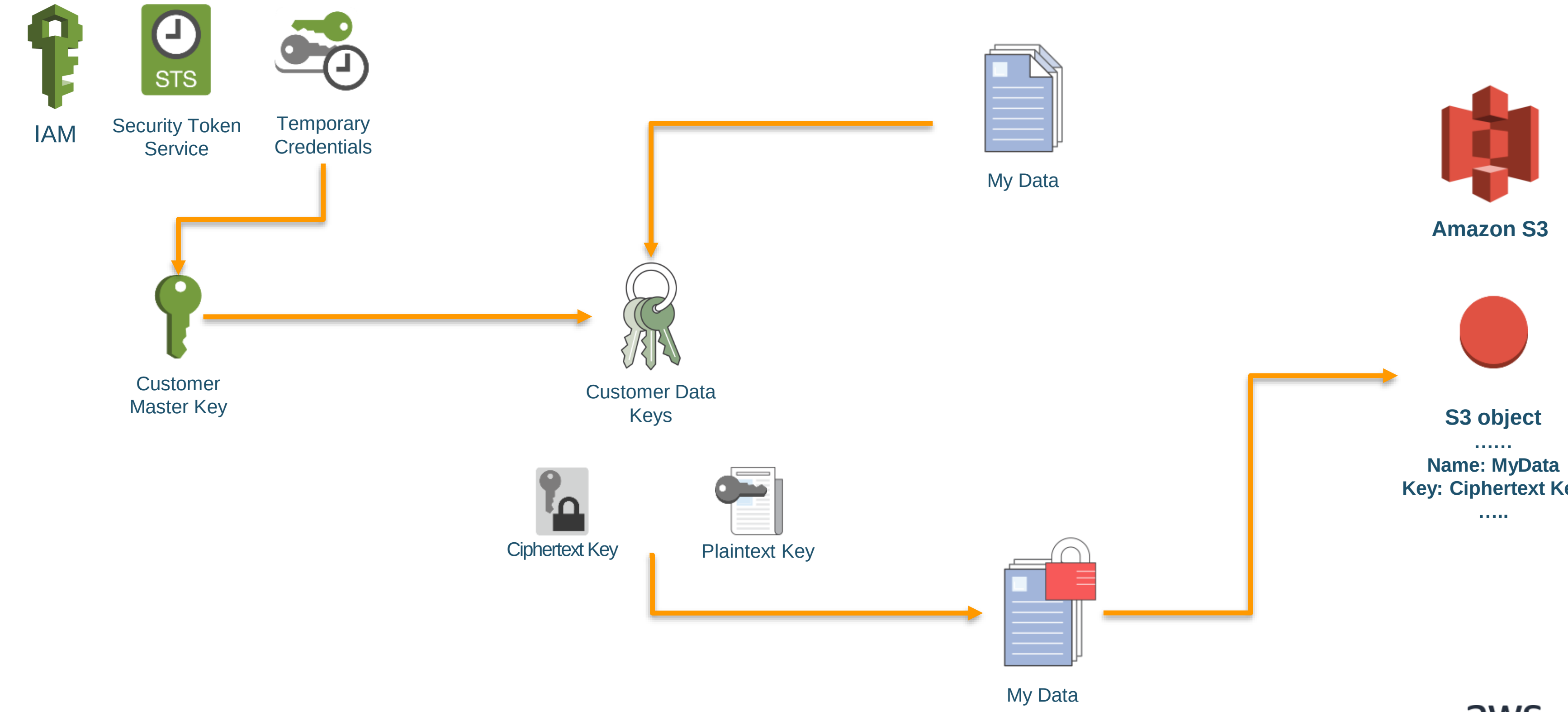
Compliance

- Certifications—HIPAA, FedRAMP, PCI-DSS
- Cloud HSM integration
- Versioning & MFA deletes
- Audit logging

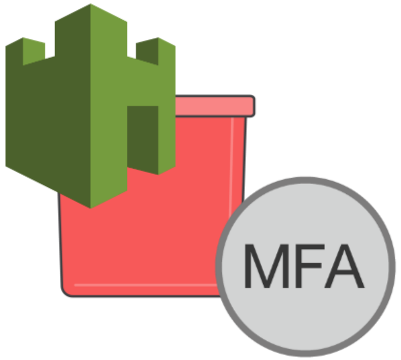
Security: Access to encryption keys



Security: Access to encryption keys



Security for your data lake



IAM best practices

SSL/TLS connections

Server-side encryption

Bucket policies

Versioning; recycle bin

MFA deletes

Optimizing performance on Amazon S3

Getting high throughput with Amazon S3

Most customers need not worry about introducing entropy in key names

Consider 3-4 character hash for higher requests per second

examplebucket/*232a-2017-26-05-15-00-00*/cust1234234/photo1.jpg
examplebucket/*7b54-2017-26-05-15-00-00*/cust3857422/photo2.jpg
examplebucket/*921c-2017-26-05-15-00-00*/cust1248473/photo2.jpg

✓

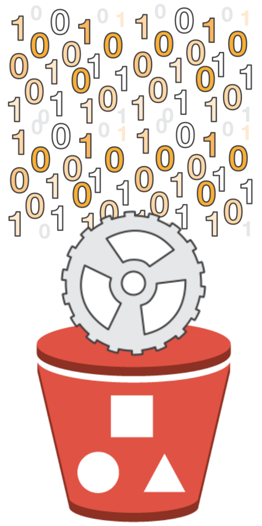
A bit more LIST friendly:

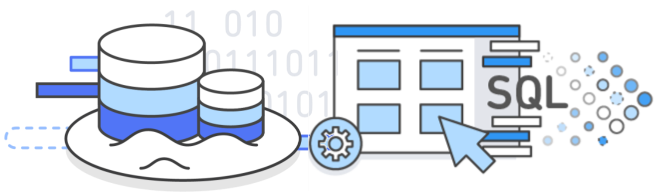
examplebucket/*animations/232a*-2017-26-05-15-00-00/cust1234234/animation1.obj
examplebucket/*videos/ba65*-2017-26-05-15-00-00/cust8474937/video2.mpg
examplebucket/*photos/8761*-2017-26-05-15-00-00/cust1248473/photo3.jpg

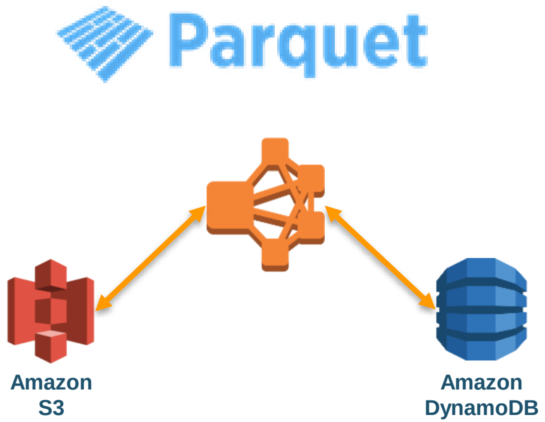
✓

Random hash should come before patterns such as dates and sequential IDs
Always first ensure that your application can accommodate

Optimizing data lake performance







Aggregate small files

EMR: S3distcp

Amazon Kinesis Firehose

S3 Select

Big data cheaper, faster

Up to 400% faster

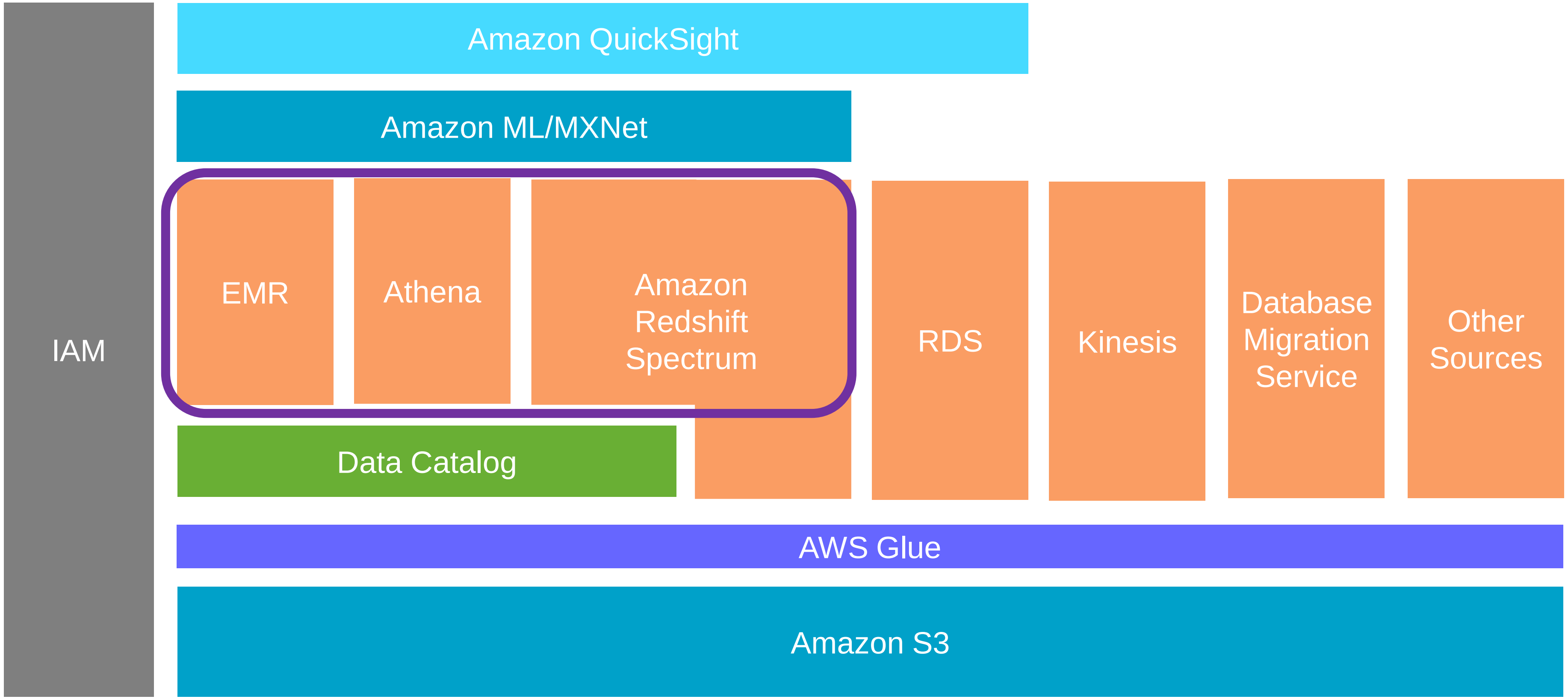
Data Formats

Columnar formats

EMRFS consistent view

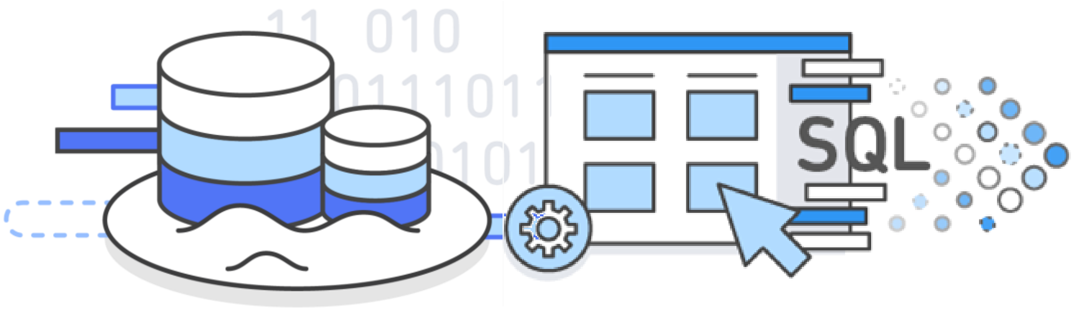
Big data analytics & query in place

Amazon analytics end-to-end architecture



Introducing Amazon S3 Select New

Simple API to retrieve subset of data based on a SQL expression

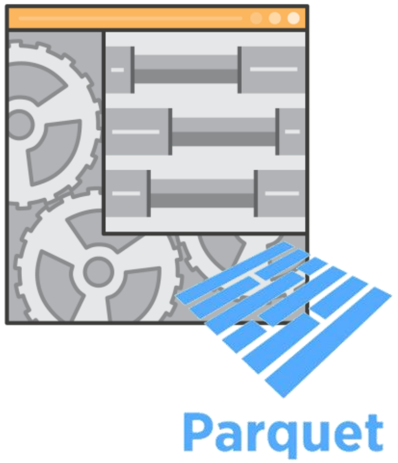


Accelerate performance for data retrieval and processing by up to 400%

Simplify compute by retrieving subset of data in a common format

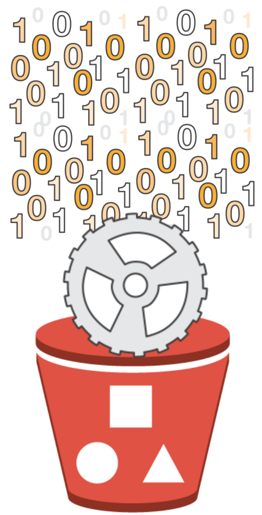
Amazon **EMR**: Decouple compute & storage

Highly distributed processing frameworks such as **Hadoop/Spark**



Aggregate small files

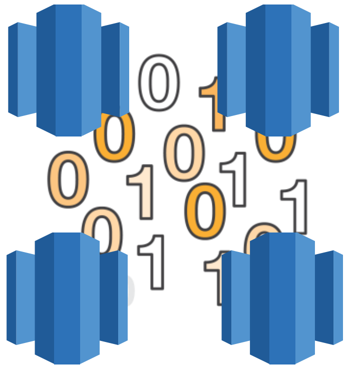
S3distcp "group-by" clause

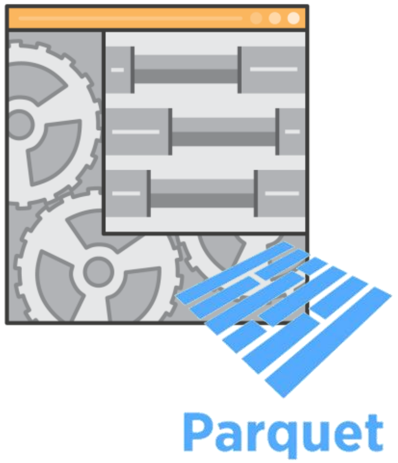


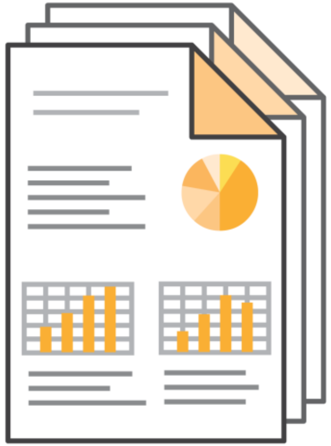
Compress datasets

Columnar file formats

Amazon **Redshift Spectrum**: Exabyte Scale query-in-place







Structured data w/ joins

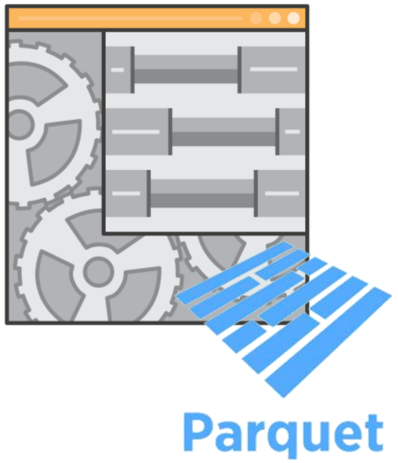
Multiple on-demand clusters - scale concurrency

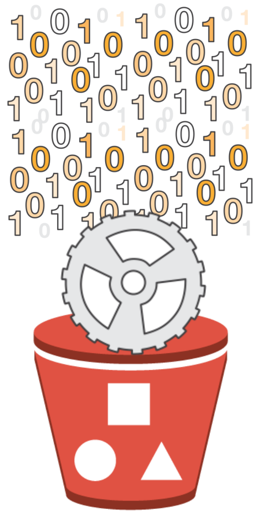
Columnar file formats

Data partitioning

Better query performance with predicate pushdown

Amazon **Athena**: Query without ETL





Serverless service

Schema on read

Compress datasets

Columnar file formats

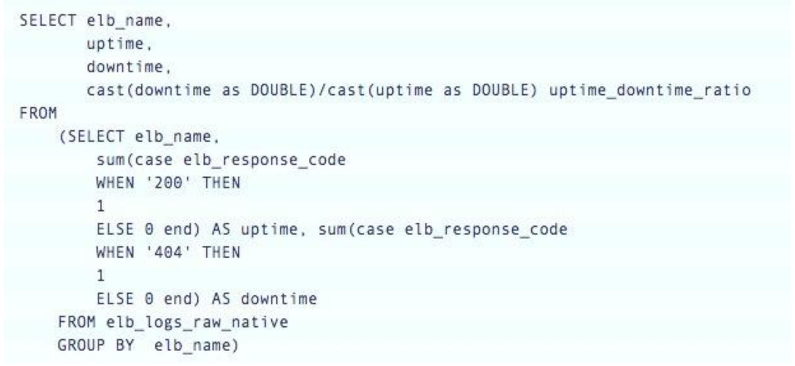
Optimize file sizes

Optimize querying (Presto backend)

Use the right data formats

- Pay by the amount of data scanned per query
- Use compressed columnar formats
    - Parquet
    - ORC

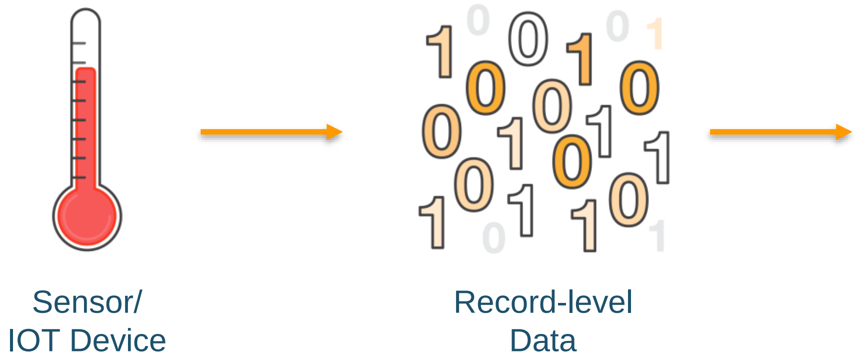
- Easy to integrate with wide variety of tools



```
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**

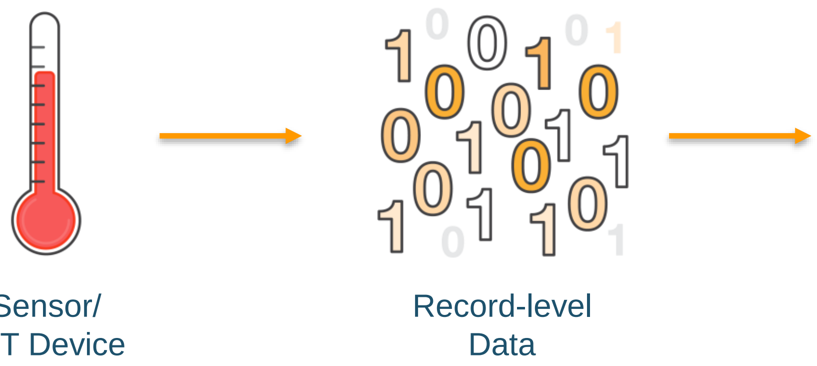
Example Use Case



Business Questions

1. What is going on with a specific sensor?
2. Daily aggregations (device inefficiencies, average temperatures, etc)
3. A real-time view of how many sensors are showing inefficiencies

Example Use Case



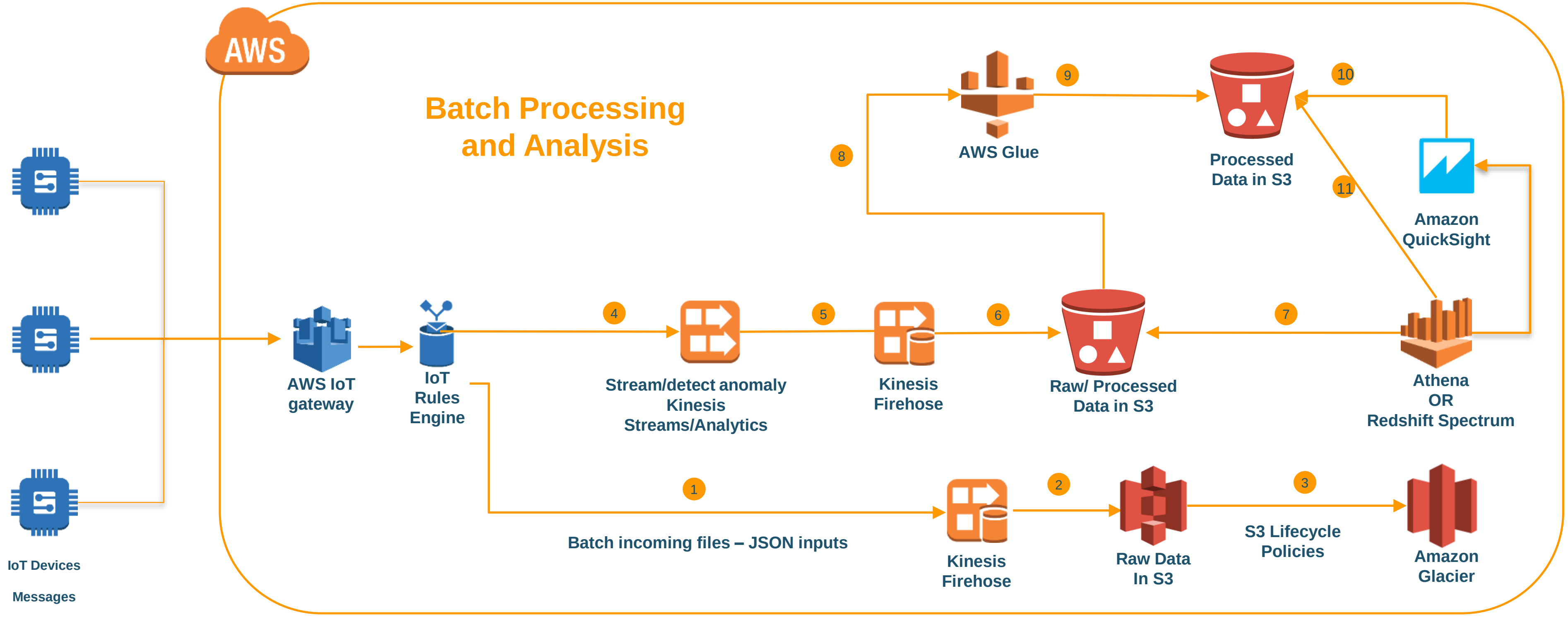
Business Questions

1. What is going on with a specific sensor?
2. Daily aggregations (device inefficiencies, average temperatures, etc)
3. A real-time view of how many sensors are showing inefficiencies

Operations

1. Scale
2. High Availability
3. Less Management Overhead
4. Pay what I need

Example Use Case Architecture



Example Use Case – Characteristics

- ✓ Scales to hundreds of thousands of data sources
- ✓ Virtually infinite storage capability
- ✓ Highly available and durable
- ✓ Real time and batch processing layers
- ✓ Interactive queries
- ✓ Pay only for what you use

- ❖ No servers to manage

Putting it all together…

- ✓ Always store a copy of the raw input
- ✓ Implement the right security controls
- ✓ Use a format that supports your data, rather than forcing a format
- ✓ Partition data to improve performance
- ✓ Apply compression to lower network load and cost

Thank you!