

# Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the [AWS Certified Data Analytics Specialty course by Stephane Maarek and Frank Kane.](#)
- **Please do not share this document**, it is intended for personal use and exam preparation only, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to [piracy@datacumulus.com](mailto:piracy@datacumulus.com). Thanks!
- **Best of luck for the exam and happy learning!**

# AWS Certified Data Analytics Specialty Course

## DAS-C01

# Welcome! We're starting in 5 minutes



- We're going to prepare for the Big Data Specialty exam – BDS-C00
- It's a challenging certification, so this course will be long and interesting
- Recommended to have previous AWS knowledge (EC2, networking...)
- Preferred to have some data / analytics background
- We will cover all the AWS Big Data services related to the exam
- Take your time, it's not a race!

# My certification: 94%

Overall Score: 94%

Topic Level Scoring:

- 1.0 Collection: 100%
- 2.0 Storage: 100%
- 3.0 Processing: 100%
- 4.0 Analysis: 87%
- 5.0 Visualization: 100%
- 6.0 Data Security: 80%



# About me

- I'm Stephane!
- Working as in IT consultant and AWS Big Data Architect, Developer & SysOps
- Worked with AWS many years: built websites, apps, streaming platforms
- Veteran Instructor on AWS (Certifications, CloudFormation, Lambda, EC2...)
- You can find me on
  - GitHub: <https://github.com/simplesteph>
  - LinkedIn: <https://www.linkedin.com/in/stephanemaarek>
  - Medium: <https://medium.com/@stephane.maarek>
  - Twitter: <https://twitter.com/stephanemaarek>



★ 4.6 Instructor Rating  
💬 22,710 Reviews  
👤 78,119 Students  
▶ 20 Courses

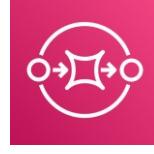
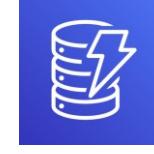
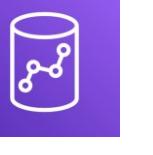
# About me

- I'm Frank!
- 9 years at Amazon as a Sr. Software Engineer and Sr. Manager
- Focused on machine learning / recommender systems in big data environment
- Owner of Sundog Education – Big Data & ML
- You can find me on
  - LinkedIn: <https://www.linkedin.com/in/fkane/>
  - Twitter: <http://www.twitter.com/SundogEducation>
  - Facebook: <https://www.facebook.com/SundogEdu>



★ 4.5 Instructor Rating  
💬 48,404 Reviews  
👤 238,457 Students  
🌐 15 Courses

# Services we'll learn

COLLECTION	STORAGE	PROCESSING	ANALYSIS	VISUALIZATION	SECURITY
					
Amazon Kinesis	AWS IoT Core	S3 + Glacier	AWS Lambda	Amazon ML	Elasticsearch
					
AWS Snowball	Amazon SQS	DynamoDB	AWS Glue	Amazon SageMaker	Amazon Athena
					
Amazon DMS	AWS Direct Connect	ElastiCache	Amazon EMR	AWS Data Pipeline	Amazon Redshift
					
					Amazon KMS
					
					AWS CloudHSM

# Course Cost (please budget \$10)

The screenshot shows the AWS Billing Dashboard. On the left, a sidebar lists various billing-related options. The main area displays two sections: 'AWS summary' and 'Highest cost'.

**AWS summary**

Current month's total forecast	Current MTD balance	Prior month for the same period with trend
<b>USD 4.62</b>	No data to display	No data to display <b>↓ 0.0%</b>

Total number of active services: **12**

Total number of active AWS accounts: No data to display

Total number of active AWS Regions: **4**

**Highest cost**

Service name	Trend compared to prior month	Current MTD balance	Prior month for the same period
<b>Kinesis Analytics</b>	<b>↓ 0.0%</b>	No data to display	No data to display

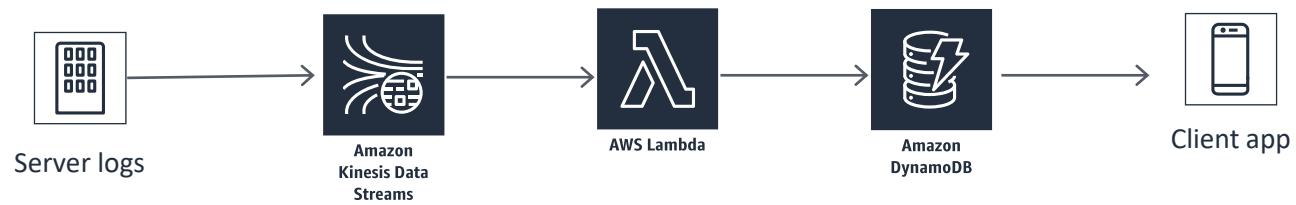
[View your bill](#)

# Introducing our case study

# Our case study: cadabra.com



# Requirement 1: Order history app



# Requirement 2:

## Product recommendations



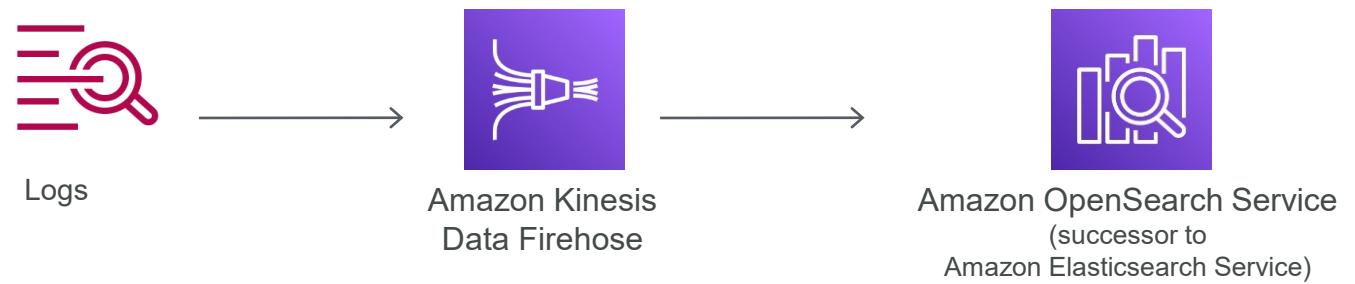
# Requirement 3:

## Transaction rate alarm

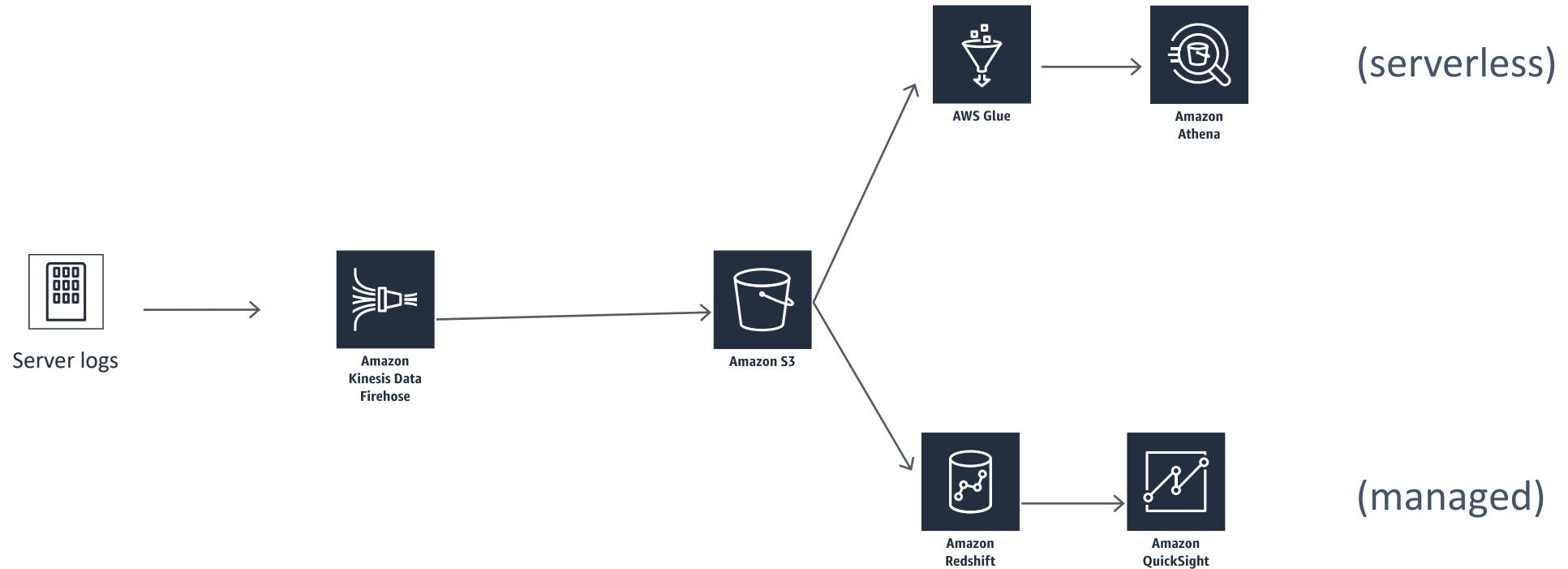


# Requirement 4:

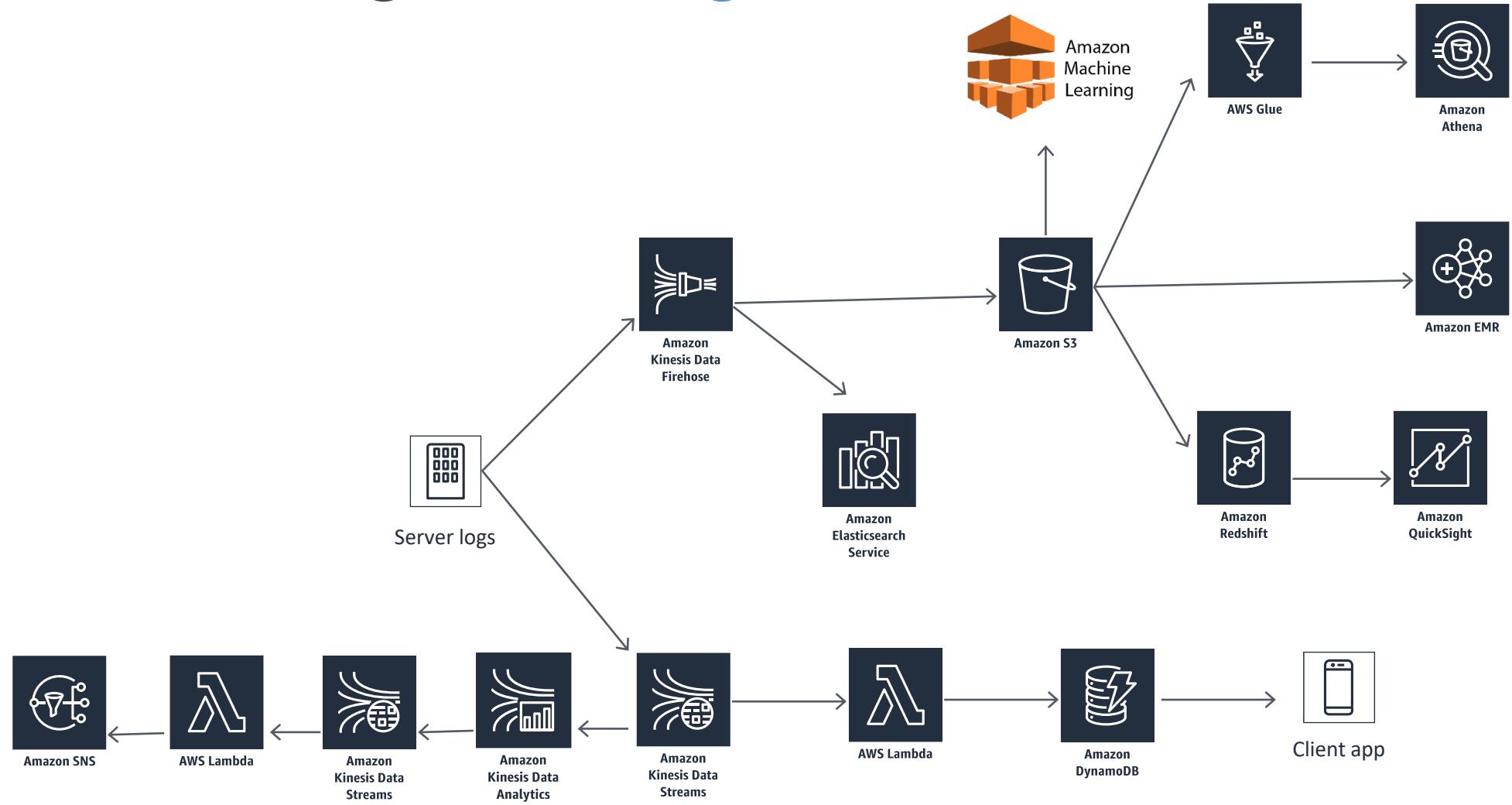
## Near-real-time log analysis



# Requirement 5: Data warehousing & visualization



# Putting it all together



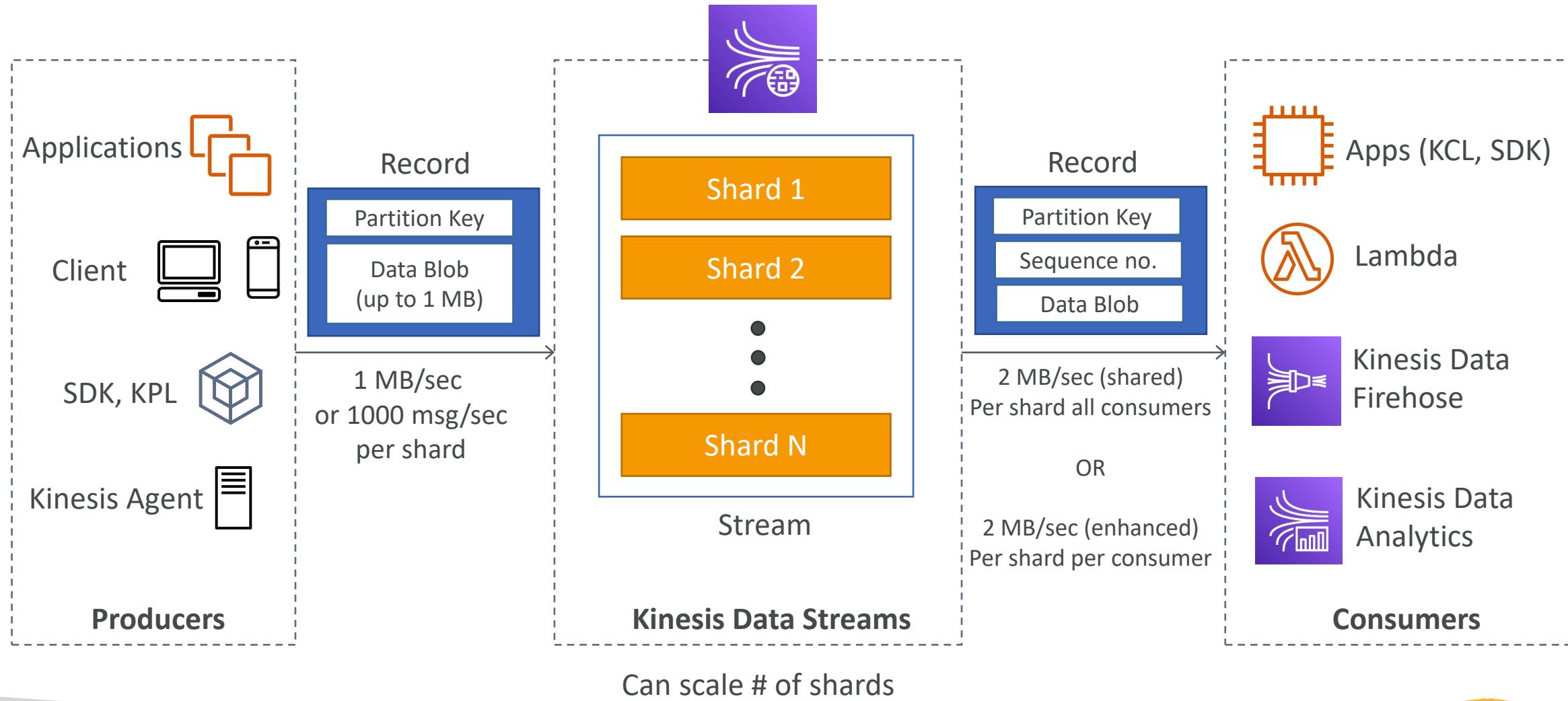
# Collection

Moving data into AWS

# Collection Introduction

- Real Time - Immediate actions
  - Kinesis Data Streams (KDS)
  - Simple Queue Service (SQS)
  - Internet of Things (IoT)
- Near-real time - Reactive actions
  - Kinesis Data Firehose (KDF)
  - Database Migration Service (DMS)
- Batch - Historical Analysis
  - Snowball
  - Data Pipeline

# Kinesis Data Streams





# Kinesis Data Streams

- Retention between 1 day to 365 days
- Ability to reprocess (replay) data
- Once data is inserted in Kinesis, it can't be deleted (immutability)
- Data that shares the same partition goes to the same shard (ordering)
- Producers: AWS SDK, Kinesis Producer Library (KPL), Kinesis Agent
- Consumers:
  - Write your own: Kinesis Client Library (KCL), AWS SDK
  - Managed: AWS Lambda, Kinesis Data Firehose, Kinesis Data Analytics,

# Kinesis Data Streams – Capacity Modes

- **Provisioned mode:**

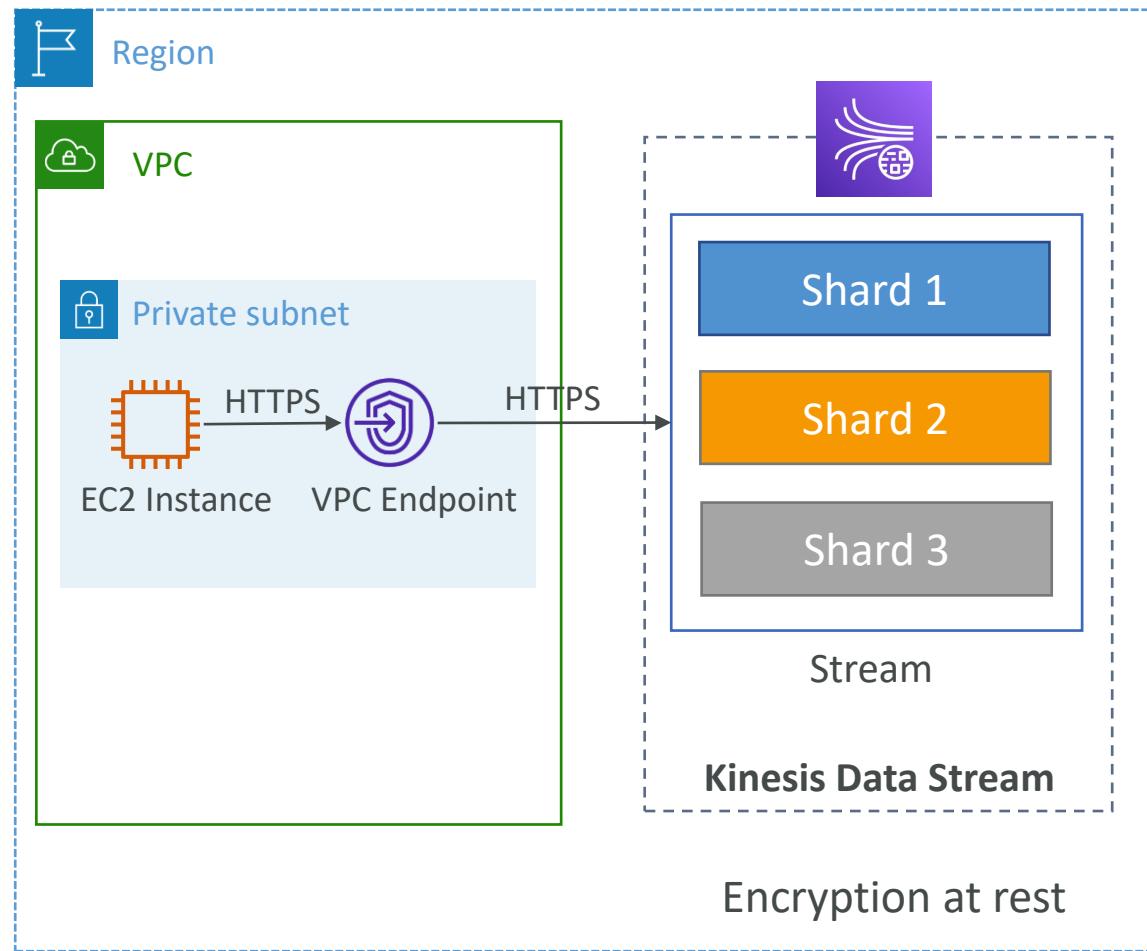
- You choose the number of shards provisioned, scale manually or using API
- Each shard gets 1MB/s in (or 1000 records per second)
- Each shard gets 2MB/s out (classic or enhanced fan-out consumer)
- You pay per shard provisioned per hour

- **On-demand mode:**

- No need to provision or manage the capacity
- Default capacity provisioned (4 MB/s in or 4000 records per second)
- Scales automatically based on observed throughput peak during the last 30 days
- Pay per stream per hour & data in/out per GB

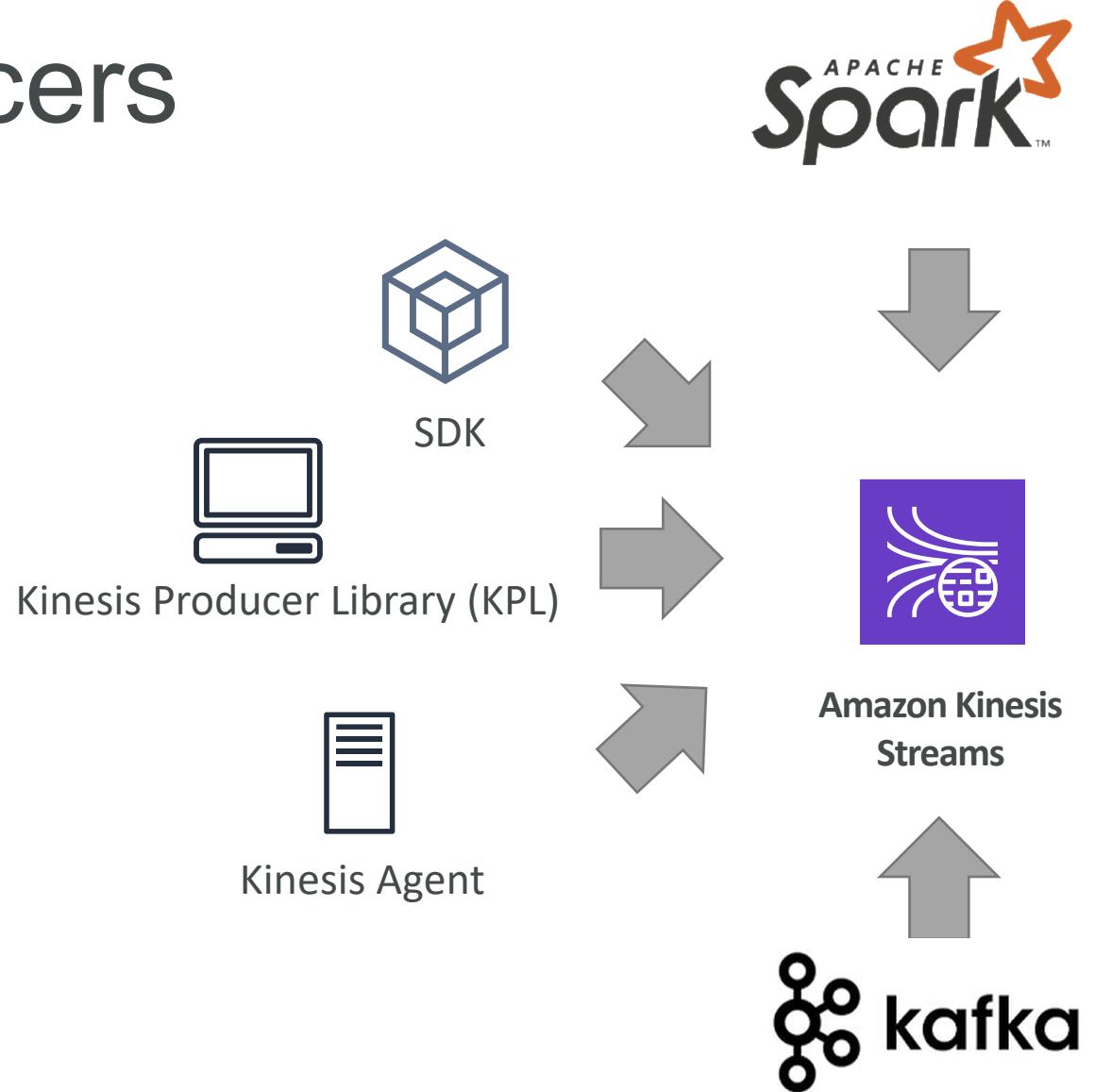
# Kinesis Data Streams Security

- Control access / authorization using IAM policies
- Encryption in flight using HTTPS endpoints
- Encryption at rest using KMS
- You can implement encryption/decryption of data on client side (harder)
- VPC Endpoints available for Kinesis to access within VPC
- Monitor API calls using CloudTrail



# Kinesis Producers

- Kinesis SDK
- Kinesis Producer Library (KPL)
- Kinesis Agent
- 3<sup>rd</sup> party libraries:  
Spark, Log4J  
Appenders, Flume,  
Kafka Connect,  
NiFi...



# Kinesis Producer SDK - PutRecord(s)

- APIs that are used are PutRecord (one) and PutRecords (many records)
- **PutRecords** uses batching and increases throughput => less HTTP requests
- **ProvisionedThroughputExceeded** if we go over the limits
- + AWS Mobile SDK: Android, iOS, etc...
- Use case: low throughput, higher latency, simple API, AWS Lambda
- Managed AWS sources for Kinesis Data Streams:
  - CloudWatch Logs
  - AWS IoT
  - Kinesis Data Analytics

# AWS Kinesis API – Exceptions

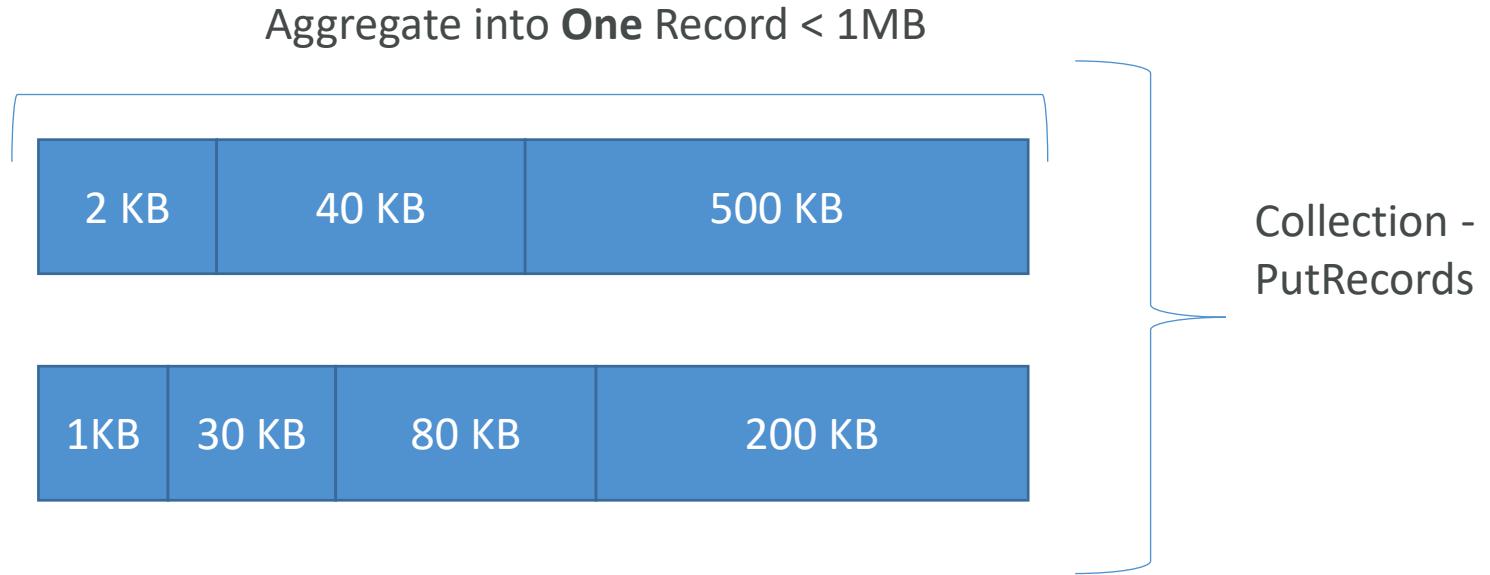
- ProvisionedThroughputExceeded Exceptions
  - Happens when sending more data (exceeding MB/s or TPS for any shard)
  - Make sure you don't have a hot shard (such as your partition key is bad and too much data goes to that partition)
- Solution:
  - Retries with backoff
  - Increase shards (scaling)
  - Ensure your partition key is a good one

# Kinesis Producer Library (KPL)

- Easy to use and highly configurable C++ / Java library
- Used for building high performance, long-running producers
- Automated and configurable **retry** mechanism
- **Synchronous or Asynchronous API** (better performance for async)
- Submits metrics to CloudWatch for monitoring
- **Batching** (both turned on by default) – increase throughput, decrease cost:
  - **Collect** Records and Write to multiple shards in the same PutRecords API call
  - **Aggregate** – increased latency
    - Capability to store multiple records in one record (go over 1000 records per second limit)
    - Increase payload size and improve throughput (maximize 1MB/s limit)
- Compression must be implemented by the user
- KPL Records must be de-coded with KCL or special helper library

# Kinesis Producer Library (KPL)

## Batching



- We can influence the batching efficiency by introducing some delay with `RecordMaxBufferedTime` (default 100ms)

# Kinesis Producer Library – When not to use

- The KPL can incur an additional processing delay of up to **RecordMaxBufferedTime** within the library (user-configurable)
- Larger values of **RecordMaxBufferedTime** results in higher packing efficiencies and better performance
- **Applications that cannot tolerate this additional delay may need to use the AWS SDK directly**

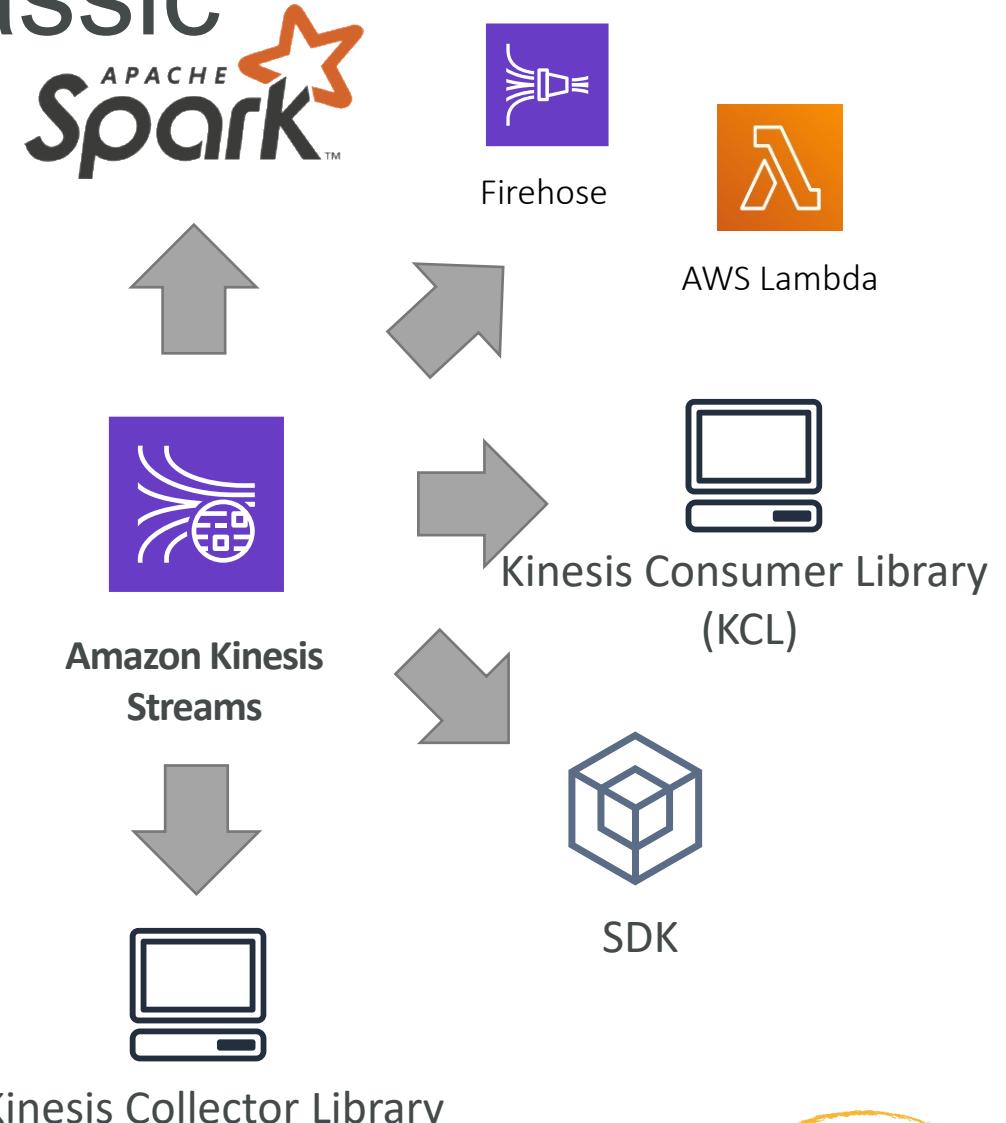


# Kinesis Agent

- Monitor Log files and sends them to Kinesis Data Streams
- Java-based agent, built on top of KPL
- Install in Linux-based server environments
- Features:
  - Write from multiple directories and write to multiple streams
  - Routing feature based on directory / log file
  - Pre-process data before sending to streams (single line, csv to json, log to json...)
  - The agent handles file rotation, checkpointing, and retry upon failures
  - Emits metrics to CloudWatch for monitoring

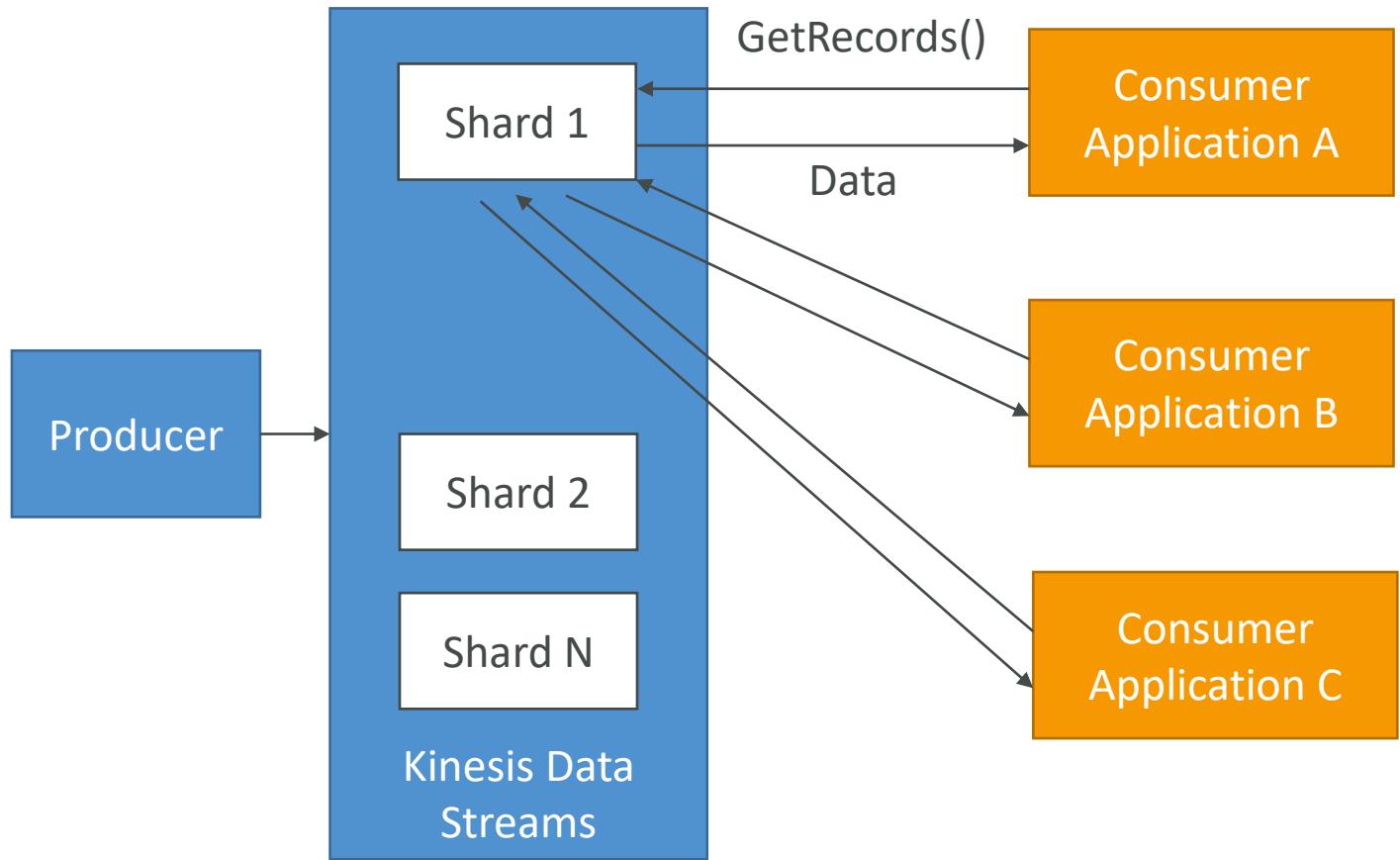
# Kinesis Consumers - Classic

- Kinesis SDK
- Kinesis Client Library (KCL)
- Kinesis Connector Library
- 3<sup>rd</sup> party libraries: Spark, Log4J Appenders, Flume, Kafka Connect...
- Kinesis Firehose
- AWS Lambda
- (Kinesis Consumer Enhanced Fan-Out discussed in the next lecture)



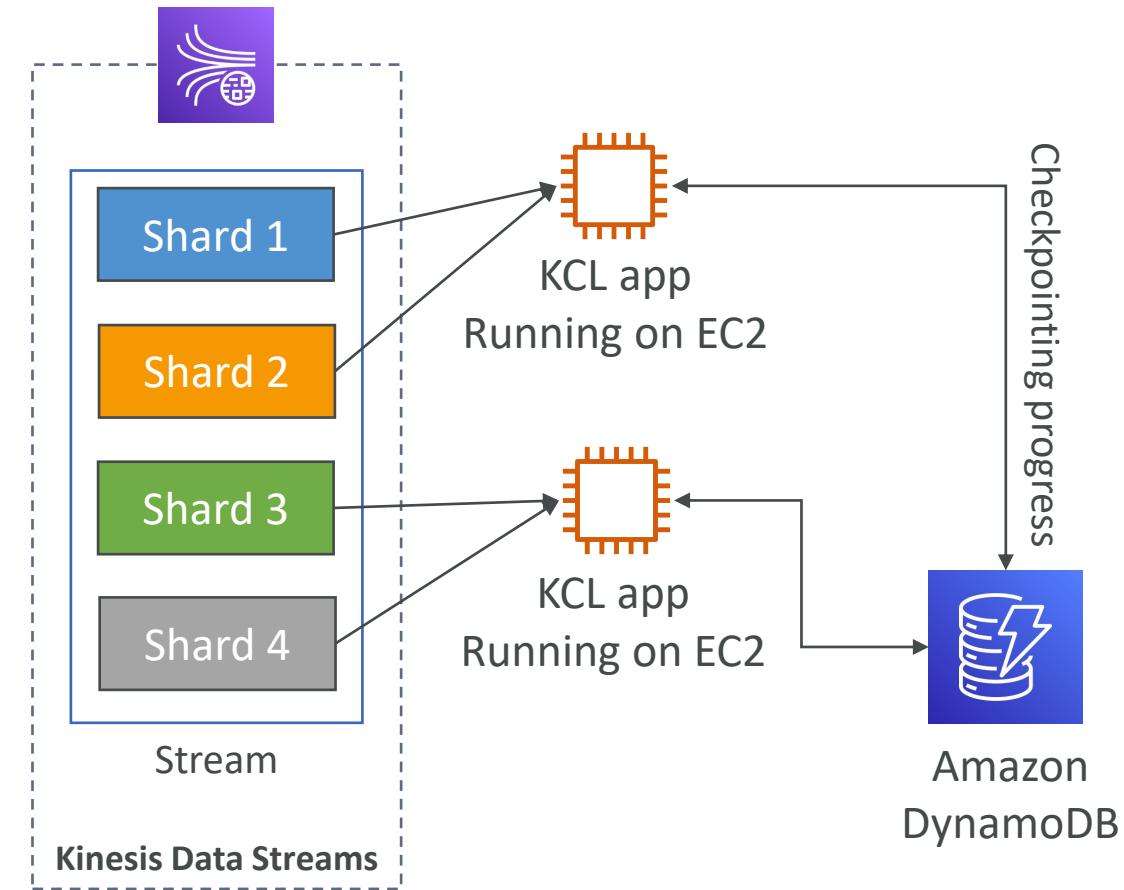
# Kinesis Consumer SDK - GetRecords

- **Classic Kinesis** - Records are polled by consumers from a shard
- **Each shard has 2 MB total aggregate throughput**
- GetRecords returns up to 10MB of data (then throttle for 5 seconds) or up to 10000 records
- Maximum of 5 GetRecords API calls per shard per second = 200ms latency
- If 5 consumers application consume from the same shard, means every consumer can poll once a second and receive less than 400 KB/s



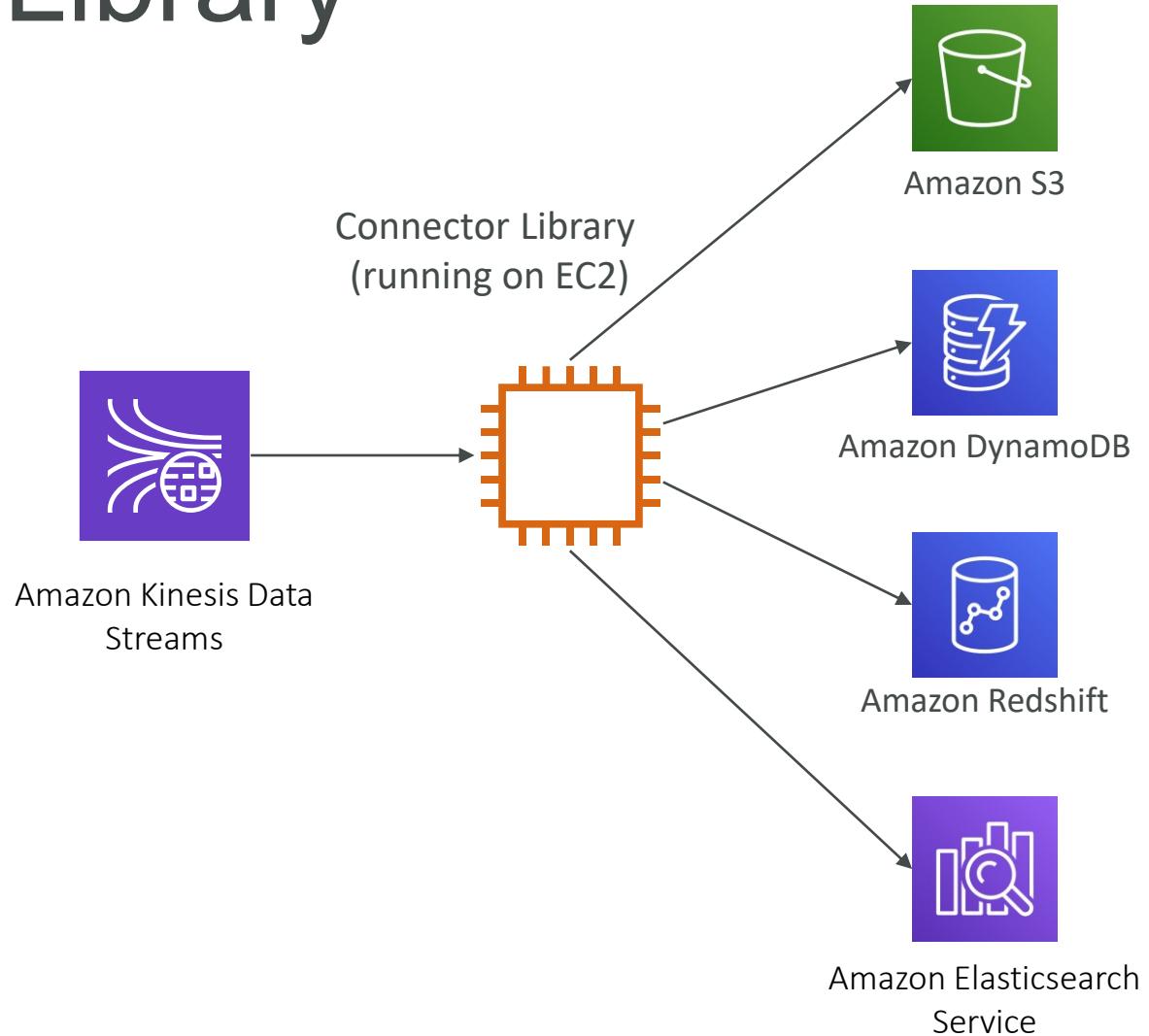
# Kinesis Client Library (KCL)

- Java-first library but exists for other languages too (Golang, Python, Ruby, Node, .NET ...)
- Read records from Kinesis produced with the KPL (de-aggregation)
- Share multiple shards with multiple consumers in one “group”, **shard discovery**
- **Checkpointing** feature to resume progress
- Leverages DynamoDB for coordination and checkpointing (one row per shard)
  - Make sure you provision enough WCU / RCU
  - Or use On-Demand for DynamoDB
  - Otherwise DynamoDB may slow down KCL
- Record processors will process the data
- **ExpiredIteratorException => increase WCU**



# Kinesis Connector Library

- Older Java library (2016), leverages the KCL library
- Write data to:
  - Amazon S3
  - DynamoDB
  - Redshift
  - ElasticSearch
- Kinesis Firehose replaces the Connector Library for a few of these targets, Lambda for the others

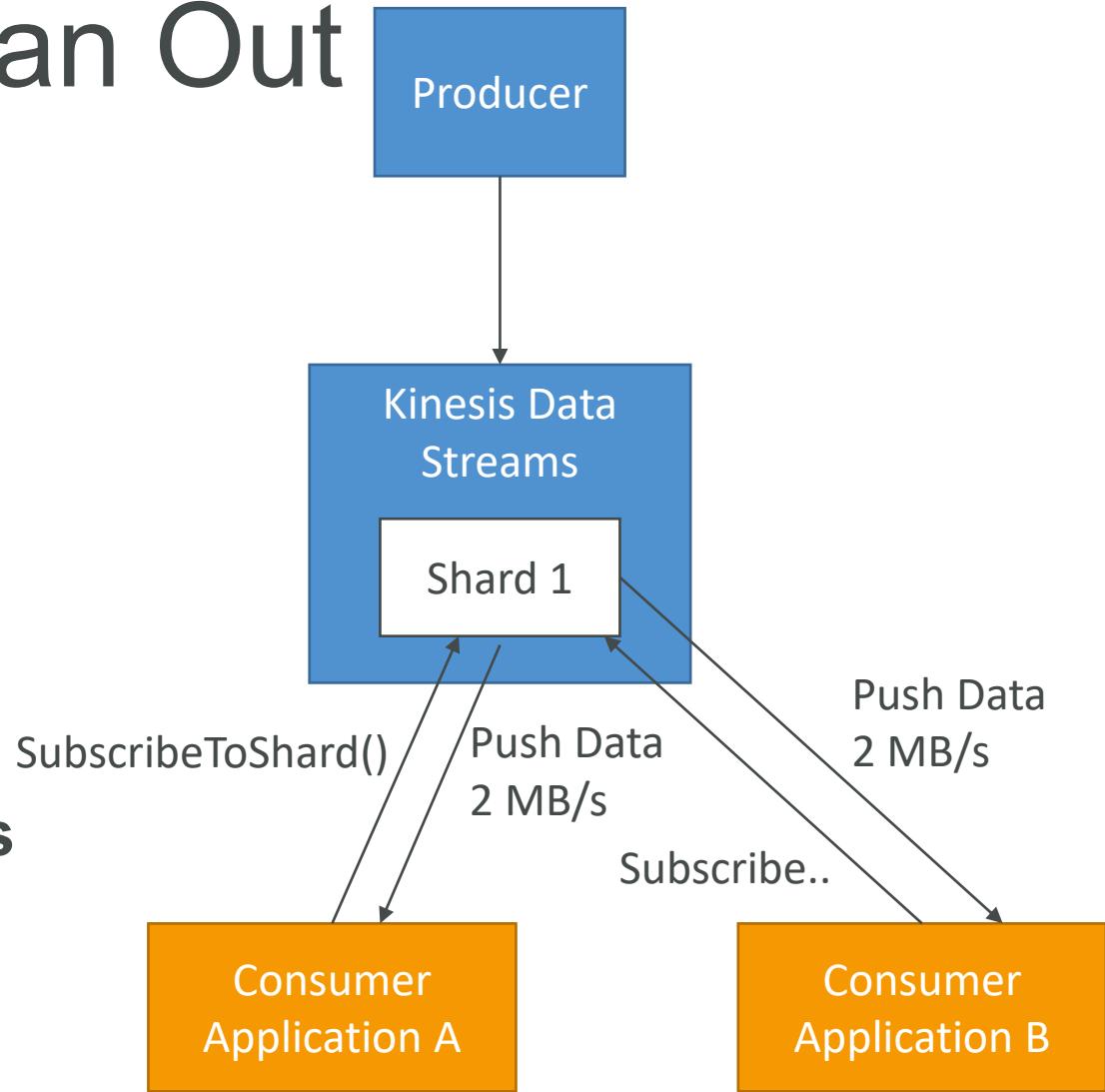


# AWS Lambda sourcing from Kinesis

- AWS Lambda can source records from Kinesis Data Streams
- Lambda consumer has a library to de-aggregate record from the KPL
- Lambda can be used to run lightweight ETL to:
  - Amazon S3
  - DynamoDB
  - Redshift
  - ElasticSearch
  - Anywhere you want
- Lambda can be used to trigger notifications / send emails in real time
- Lambda has a configurable batch size (more in Lambda section)

# Kinesis Enhanced Fan Out

- New **game-changing** feature from August 2018.
- Works with KCL 2.0 and AWS Lambda (Nov 2018)
- Each Consumer get 2 MB/s of provisioned throughput per shard
- That means 20 consumers will get 40MB/s per shard aggregated
- No more 2 MB/s limit!
- Enhanced Fan Out: Kinesis **pushes** data to consumers over HTTP/2
- Reduce latency (~70 ms)

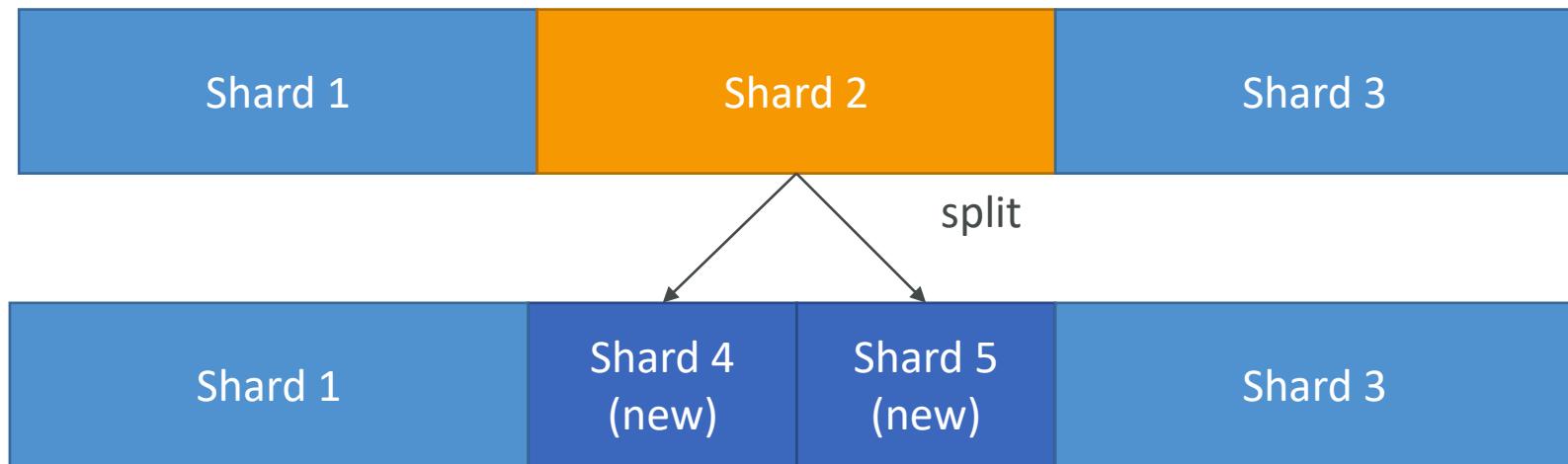


# Enhanced Fan-Out vs Standard Consumers

- Standard consumers:
  - Low number of consuming applications (1,2,3...)
  - Can tolerate ~200 ms latency
  - Minimize cost
- Enhanced Fan Out Consumers:
  - Multiple Consumer applications for the same Stream
  - Low Latency requirements ~70ms
  - Higher costs (see Kinesis pricing page)
  - Default limit of 20 consumers using enhanced fan-out per data stream

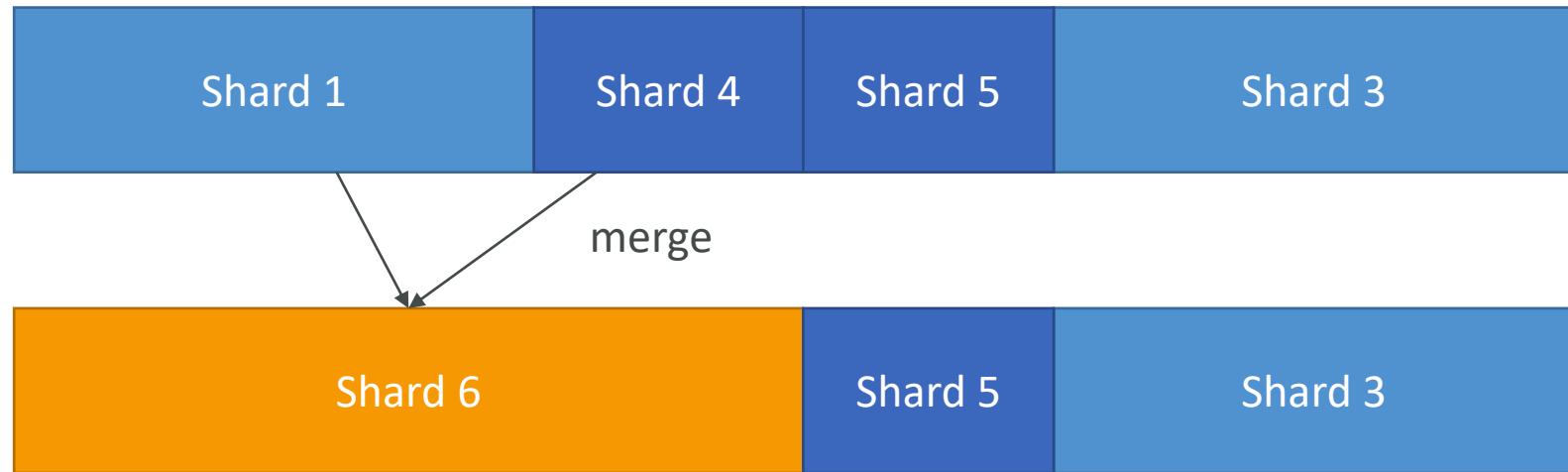
# Kinesis Operations – Adding Shards

- Also called “Shard Splitting”
- Can be used to increase the Stream capacity (1 MB/s data in per shard)
- Can be used to divide a “hot shard”
- The old shard is closed and will be deleted once the data is expired



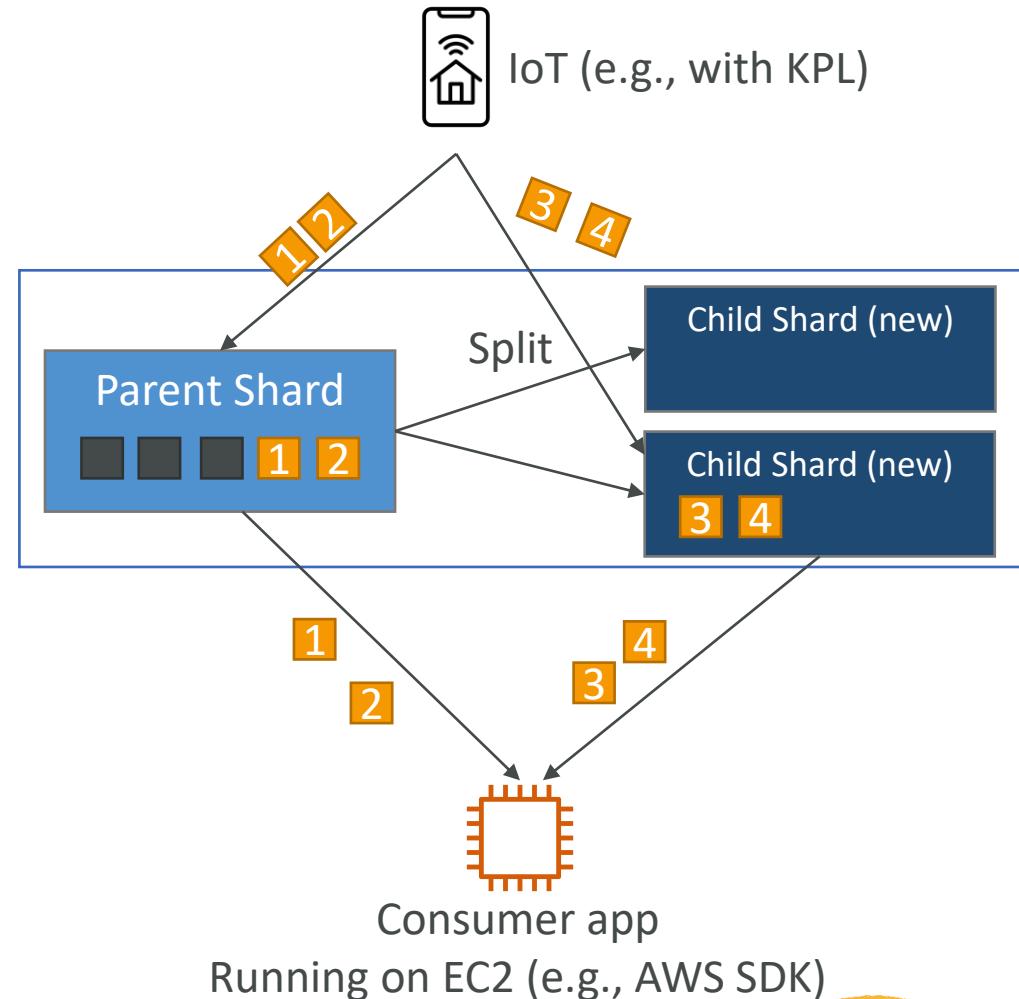
# Kinesis Operations – Merging Shards

- Decrease the Stream capacity and save costs
- Can be used to group two shards with low traffic
- Old shards are closed and deleted based on data expiration



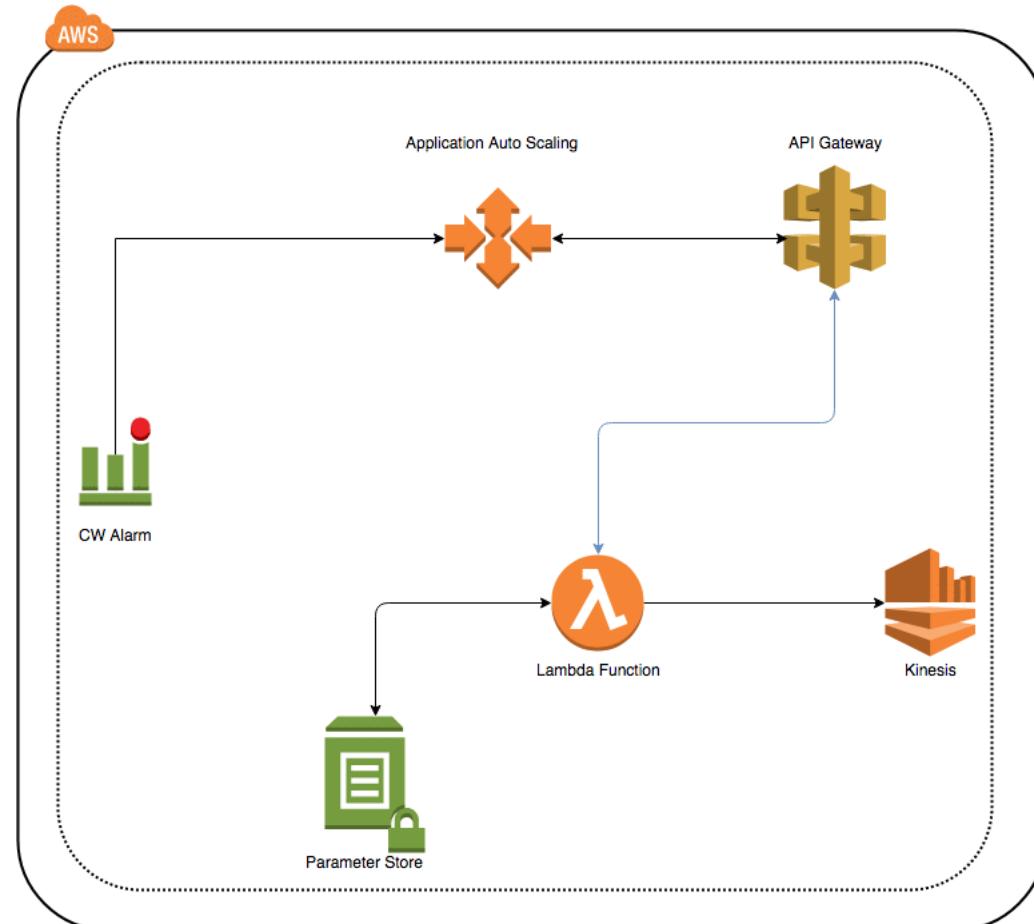
# Out-of-order records after resharding

- After a reshard, you can read from child shards
- However, data you haven't read yet could still be in the parent
- If you start reading the child before completing reading the parent, **you could read data for a particular hash key out of order**
- After a reshard, read entirely from the parent until you don't have new records
- Note: The Kinesis Client Library (KCL) has this logic already built-in, even after resharding operations



# Kinesis Operations – Auto Scaling

- Auto Scaling is not a native feature of Kinesis
- The API call to change the number of shards is `UpdateShardCount`
- We can implement Auto Scaling with AWS Lambda
- See:  
<https://aws.amazon.com/blogs/big-data/scaling-amazon-kinesis-data-streams-with-aws-application-auto-scaling/>



# Kinesis Scaling Limitations

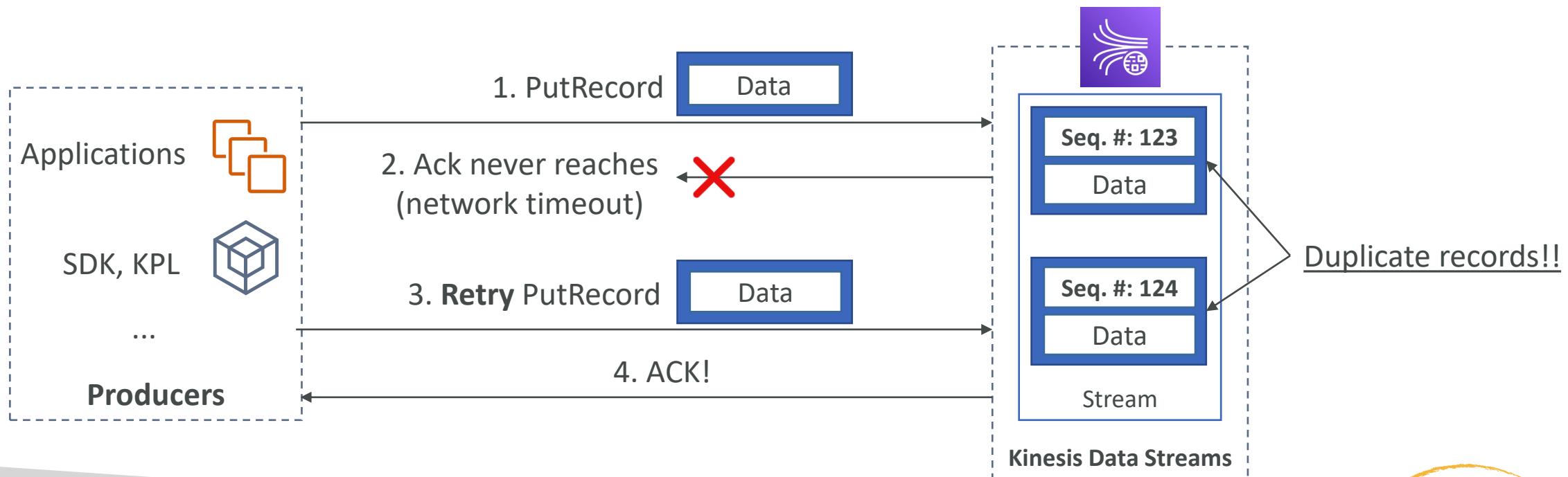
- Resharding cannot be done in parallel. Plan capacity in advance
- You can only perform one resharding operation at a time and it takes a few seconds
- For 1000 shards, it takes 30K seconds (8.3 hours) to double the shards to 2000
- **You can't do the following:**
  - Scale more than 10x for each rolling 24-hour period for each stream
  - Scale up to more than double your current shard count for a stream
  - Scale down below half your current shard count for a stream
  - Scale up to more than 10,000 shards in a stream
  - Scale a stream with more than 10,000 shards down unless the result is less than 10,000 shards
  - Scale up to more than the shard limit for your account

# Kinesis Security

- Control access / authorization using IAM policies
- Encryption in flight using HTTPS endpoints
- Encryption at rest using KMS
- Client side encryption must be manually implemented (harder)
- VPC Endpoints available for Kinesis to access within VPC

# Kinesis Data Streams – Handling Duplicates For Producers

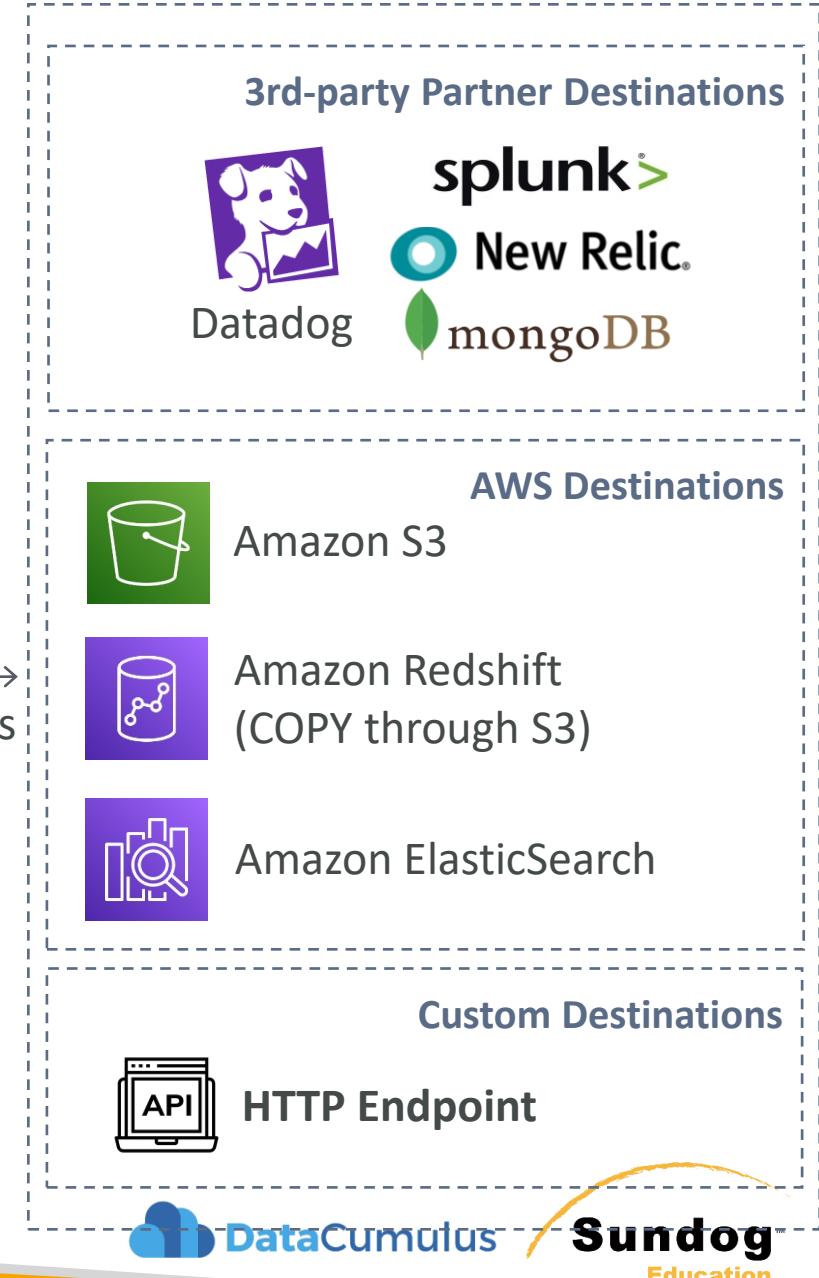
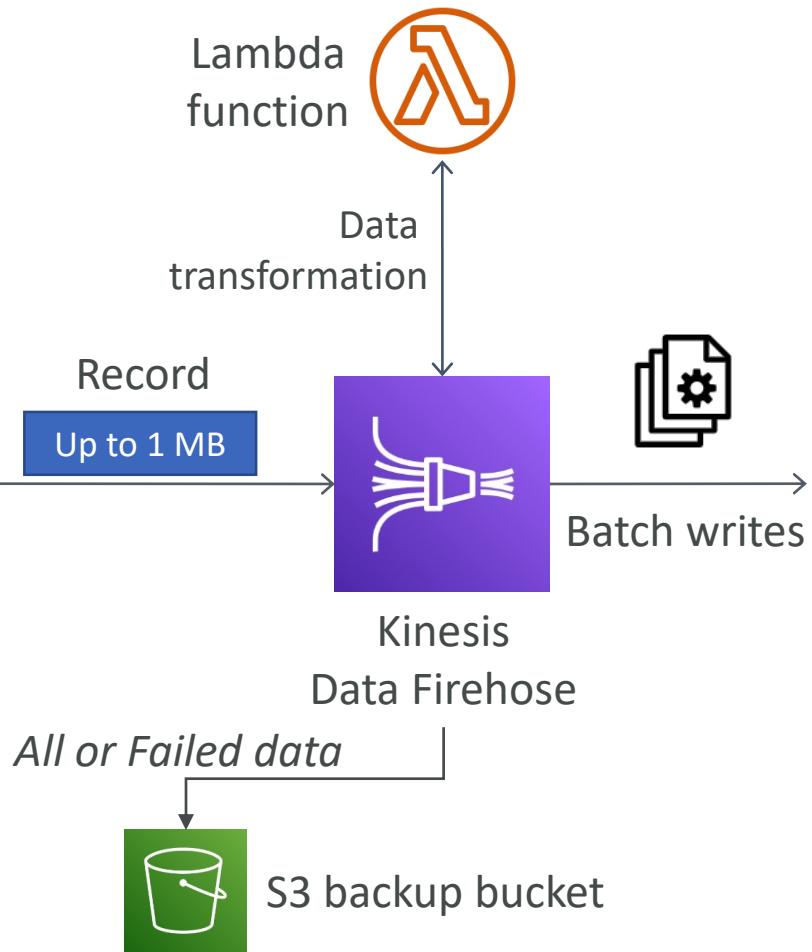
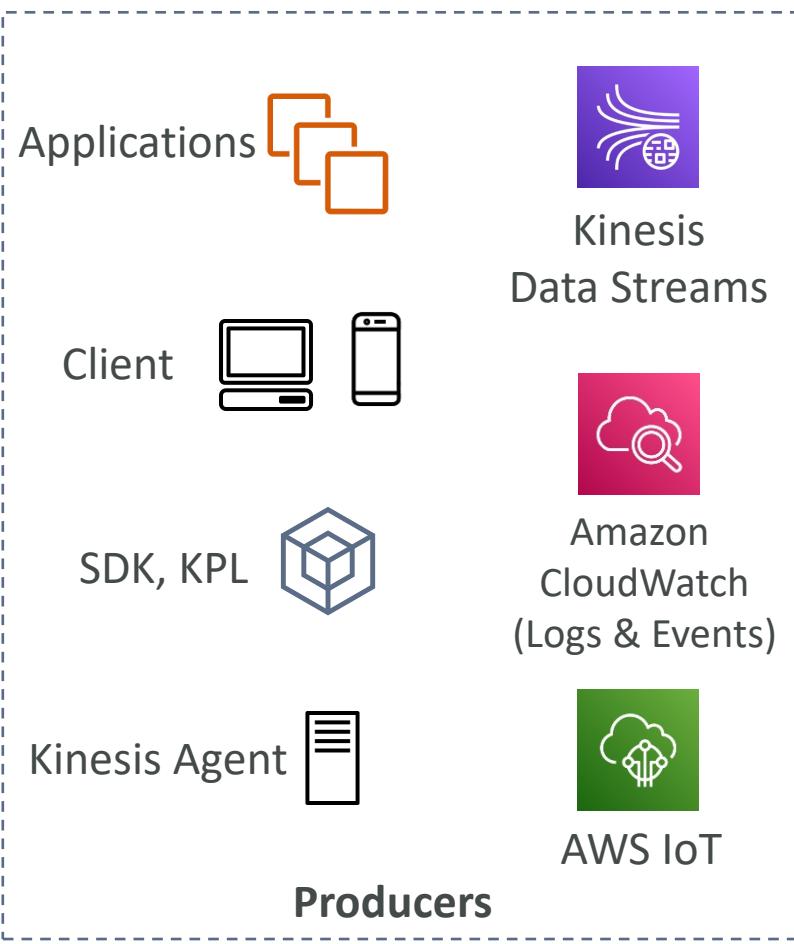
- Producer retries can create duplicates due to **network timeouts**
- Although the two records have identical data, they also have unique sequence numbers
- Fix: **embed unique record ID** in the data to de-duplicate on the consumer side



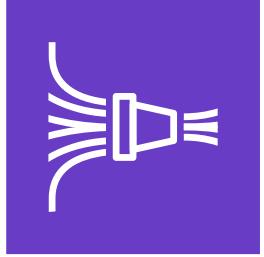
# Kinesis Data Streams – Handling Duplicates For Consumers

- Consumer retries can make your application read the same data twice
- Consumer retries happen when record processors restart:
  1. A worker terminates unexpectedly
  2. Worker instances are added or removed
  3. Shards are merged or split
  4. The application is deployed
- Fixes:
  - Make your consumer application idempotent
  - If the final destination can handle duplicates, it's recommended to do it there
- More info: <https://docs.aws.amazon.comstreams/latest/dev/kinesis-record-processor-duplicates.html>

# Kinesis Data Firehose

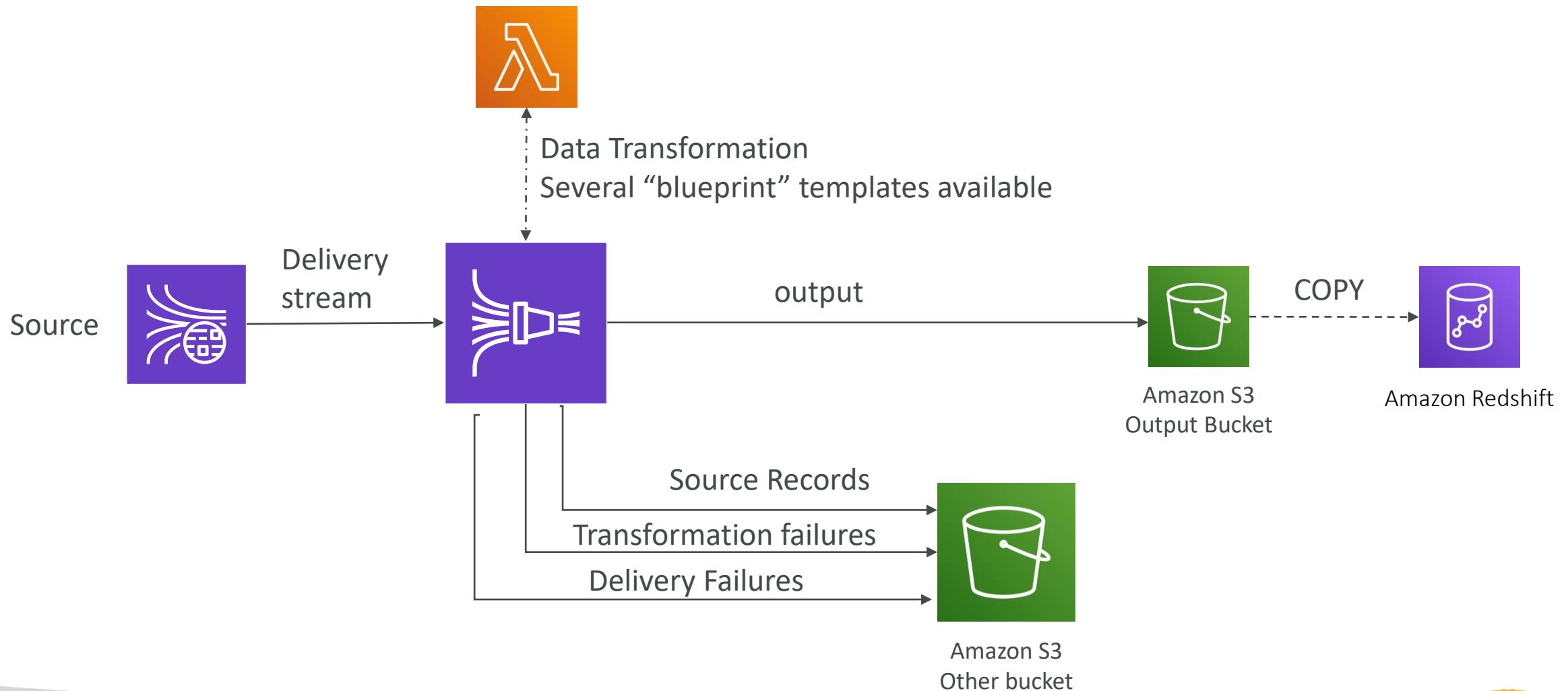


# AWS Kinesis Data Firehose



- Fully Managed Service, no administration
- **Near Real Time** (60 seconds latency minimum for non full batches)
- Load data into Redshift / Amazon S3 / ElasticSearch / Splunk
- Automatic scaling
- Supports many data formats
- Data Conversions from JSON to Parquet / ORC (only for S3)
- Data Transformation through AWS Lambda (ex: CSV => JSON)
- Supports **compression** when target is Amazon S3 (GZIP, ZIP, and SNAPPY)
- Only GZIP is the data is further loaded into Redshift
- Pay for the amount of data going through Firehose
- Spark / KCL do *not* read from KDF

# Kinesis Data Firehose Delivery Diagram



# Firehose Buffer Sizing

- Firehose accumulates records in a buffer
- The buffer is flushed based on time and size rules
- Buffer Size (ex: 32MB): if that buffer size is reached, it's flushed
- Buffer Time (ex: 2 minutes): if that time is reached, it's flushed
- Firehose can automatically increase the buffer size to increase throughput
- High throughput => Buffer Size will be hit
- Low throughput => Buffer Time will be hit

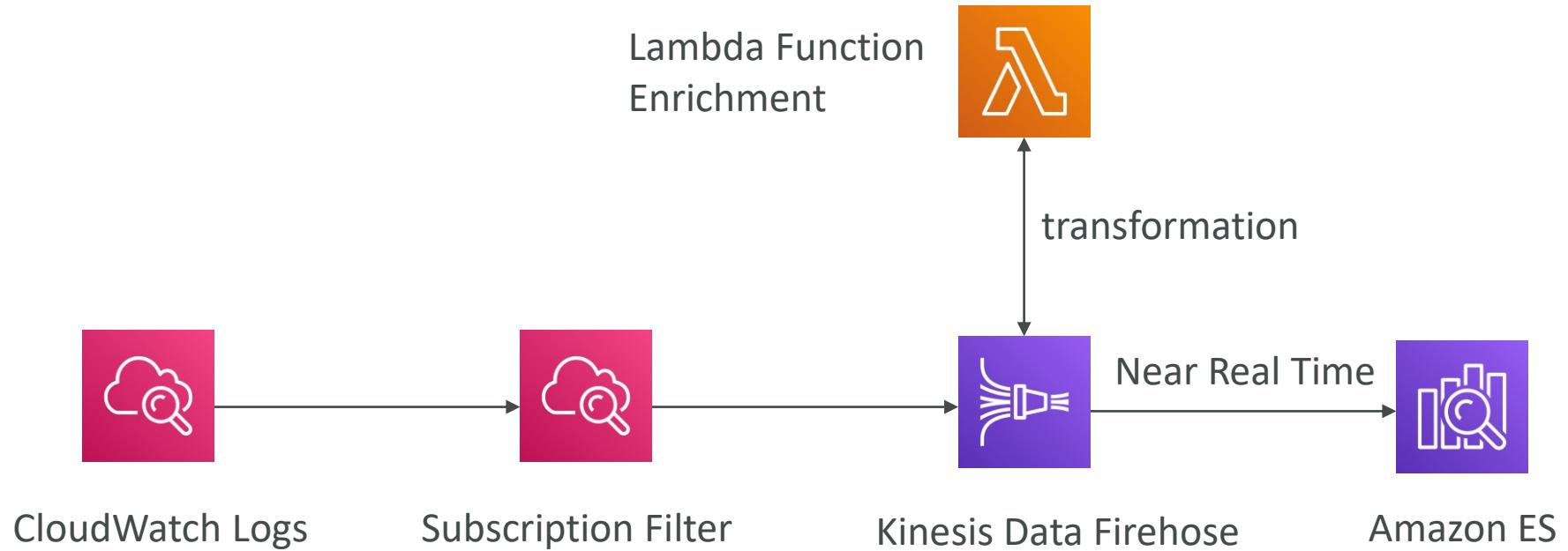
# Kinesis Data Streams vs Firehose

- Streams
  - Going to write custom code (producer / consumer)
  - Real time (~200 ms latency for classic, ~70 ms latency for enhanced fan-out)
  - Must manage scaling (shard splitting / merging)
  - Data Storage for 1 to 365 days, replay capability, multi consumers
  - Use with Lambda to insert data in real-time to ElasticSearch (for example)
- Firehose
  - Fully managed, send to S3, Splunk, Redshift, ElasticSearch
  - Serverless data transformations with Lambda
  - **Near** real time (lowest buffer time is 1 minute)
  - Automated Scaling
  - No data storage

# CloudWatch Logs Subscriptions Filters

- You can stream CloudWatch Logs into
  - Kinesis Data Streams
  - Kinesis Data Firehose
  - AWS Lambda
- Using CloudWatch Logs Subscriptions Filters
- You can enable them using the AWS CLI

# CloudWatch Logs Subscription Filter Patterns Near Real Time into Amazon ES



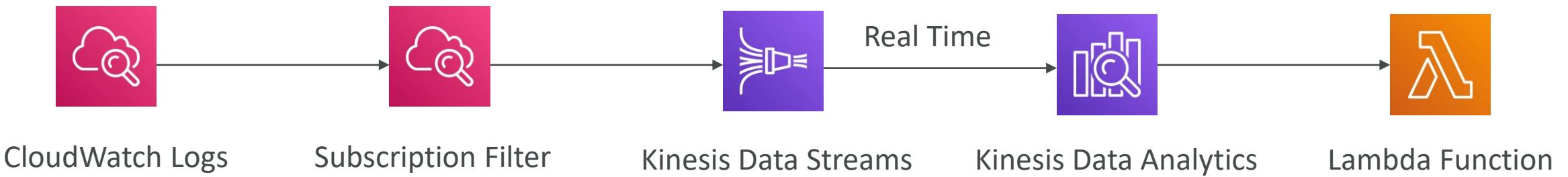
# CloudWatch Logs Subscription Filter Patterns

## Real Time Load into Amazon ES



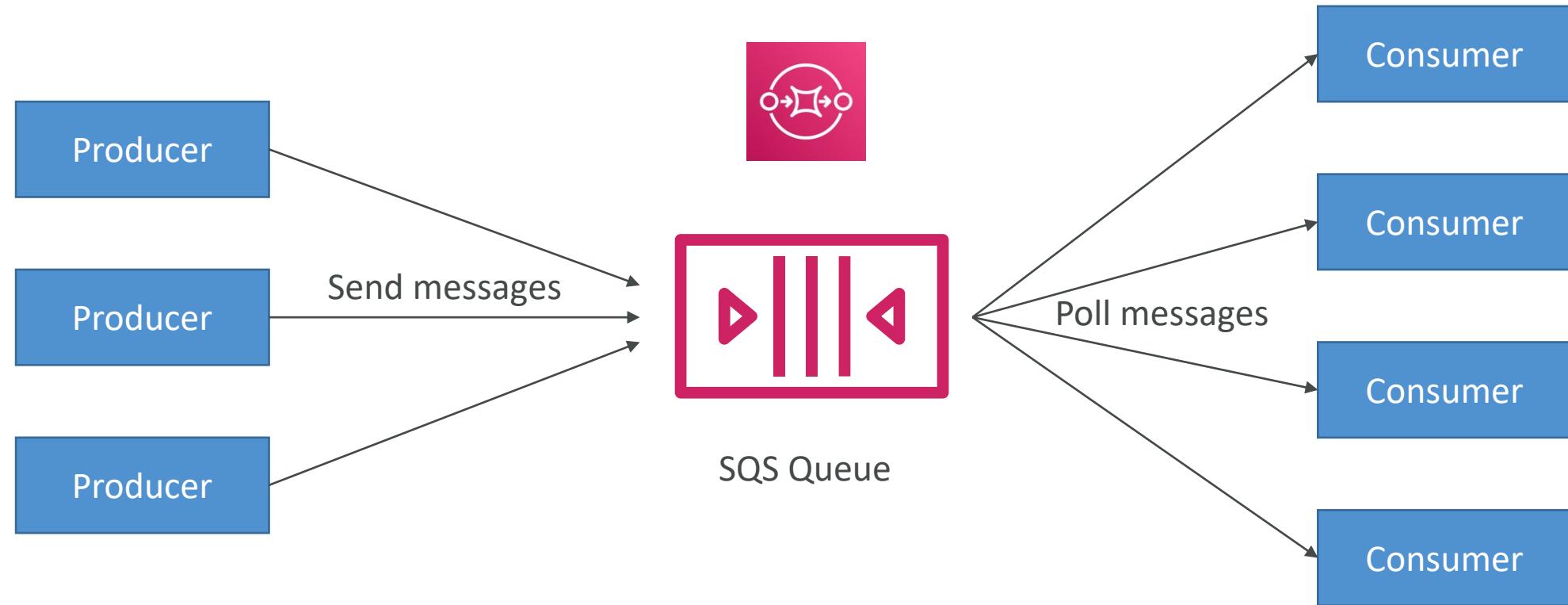
# CloudWatch Logs Subscription Filter Patterns

## Real Time Analytics

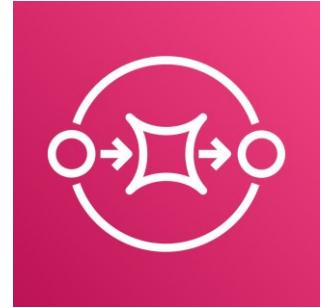


# AWS SQS

## What's a queue?



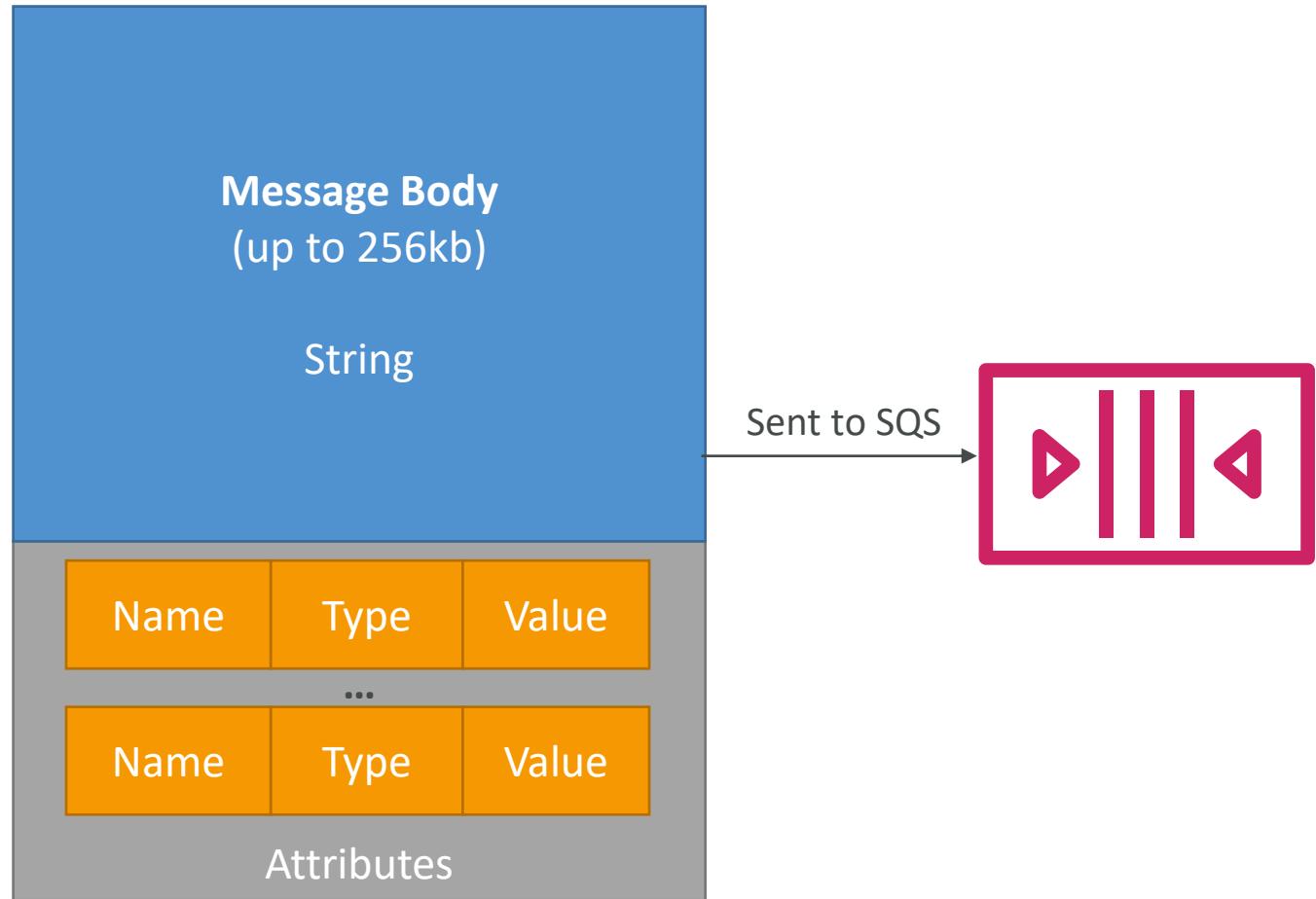
# AWS SQS – Standard Queue



- Oldest offering (over 10 years old)
- Fully managed
- Scales from 1 message per second to 10,000s per second
- Default retention of messages: 4 days, maximum of 14 days
- No limit to how many messages can be in the queue
- Low latency (<10 ms on publish and receive)
- Horizontal scaling in terms of number of consumers
- Can have duplicate messages (at least once delivery, occasionally)
- Can have out of order messages (best effort ordering)
- Limitation of 256KB per message sent

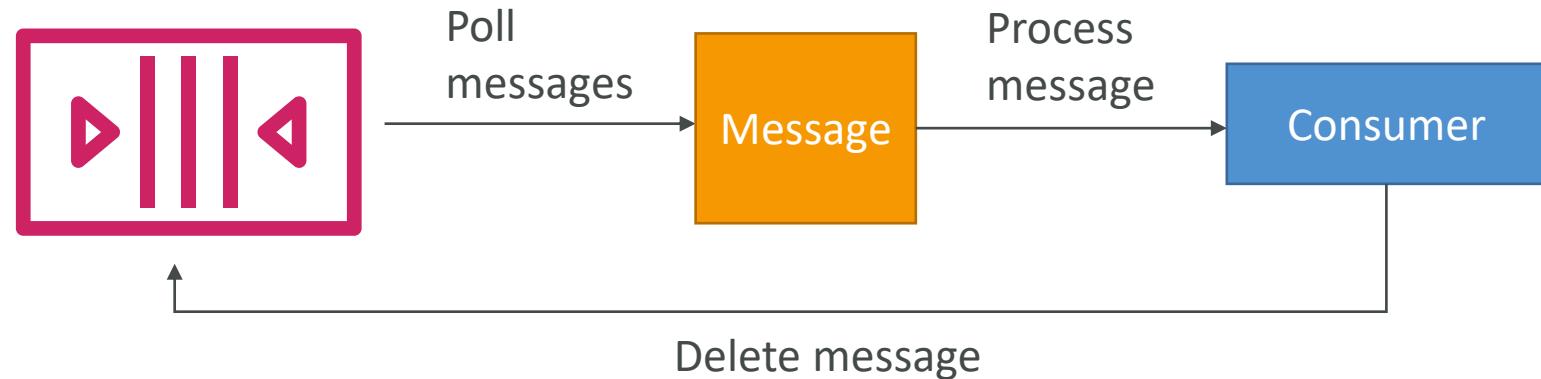
# SQS – Producing Messages

- Define Body
- Add message attributes (metadata – optional)
- Provide Delay Delivery (optional)
- Get back
  - Message identifier
  - MD5 hash of the body



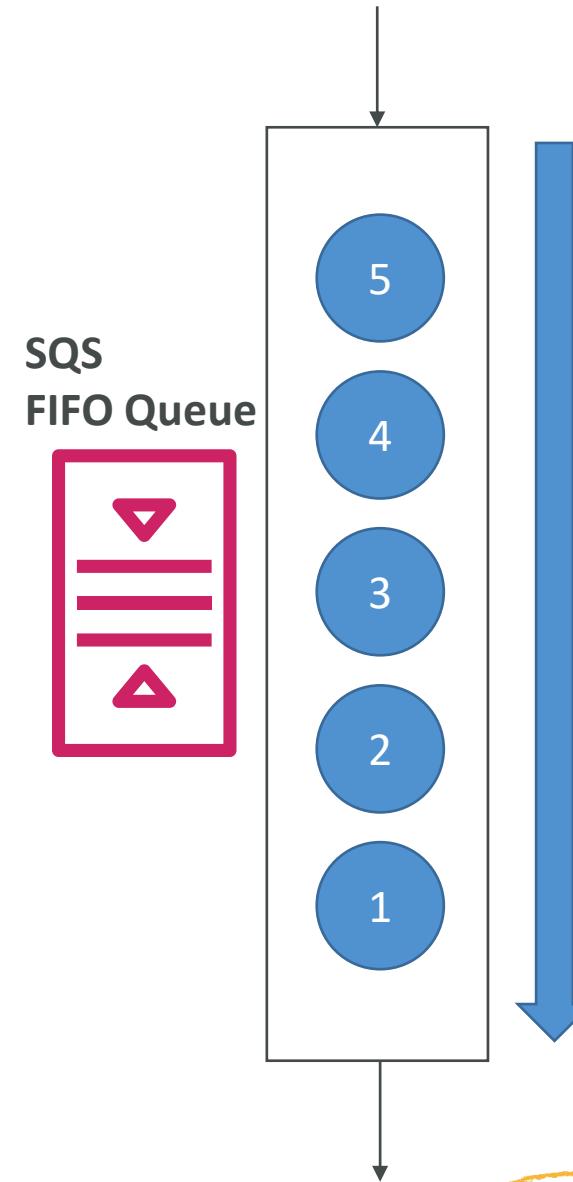
# SQS – Consuming Messages

- Consumers...
- Poll SQS for messages (receive up to 10 messages at a time)
- Process the message within the visibility timeout
- Delete the message using the message ID & receipt handle



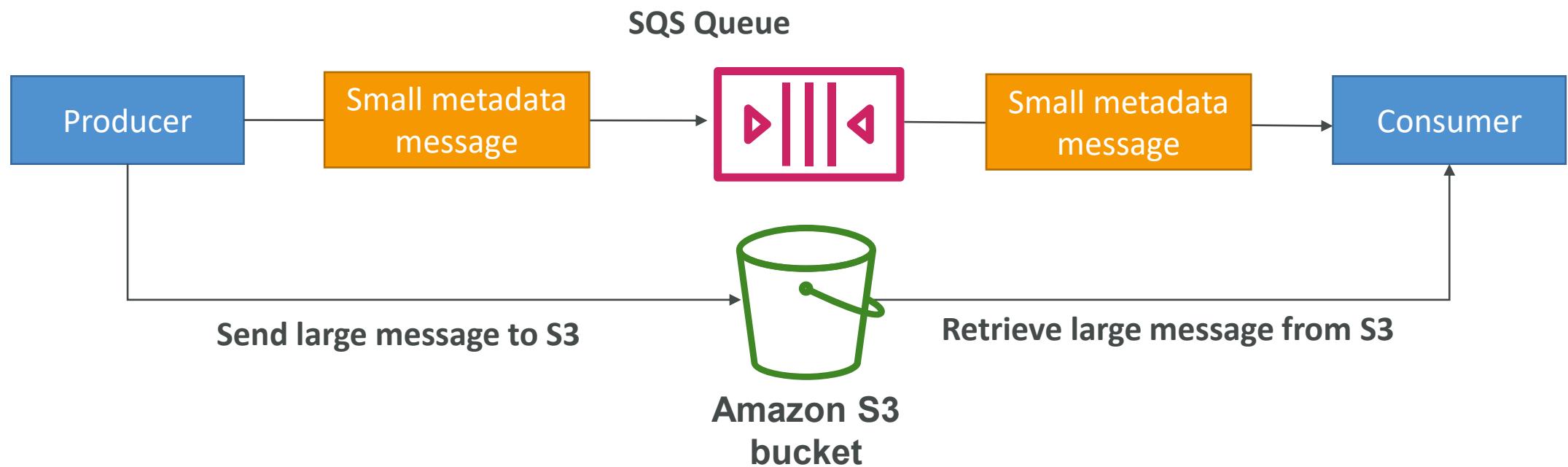
# AWS SQS – FIFO Queue

- Newer offering (First In - First out) – not available in all regions!
- Name of the queue must end in .fifo
- Lower throughput (up to 3,000 per second with batching, 300/s without)
- Messages are processed in order by the consumer
- Messages are sent exactly once
- 5-minute interval de-duplication using “Duplication ID”



# SQS Extended Client

- Message size limit is 256KB, how to send large messages?
- Using the SQS Extended Client (Java Library)



# AWS SQS Use Cases

- Decouple applications  
(for example to handle payments asynchronously)
- Buffer writes to a database  
(for example a voting application)
- Handle large loads of messages coming in  
(for example an email sender)
- SQS can be integrated with Auto Scaling through CloudWatch!

# SQS Limits

- Maximum of 120,000 in-flight messages being processed by consumers
- Batch Request has a maximum of 10 messages – max 256KB
- Message content is XML, JSON, Unformatted text
- Standard queues have an unlimited TPS
- FIFO queues support up to 3,000 messages per second (using batching)
- Max message size is 256KB (or use Extended Client)
- Data retention from 1 minute to 14 days
- Pricing:
  - Pay per API Request
  - Pay per network usage

# AWS SQS Security

- Encryption in flight using the HTTPS endpoint
- Can enable SSE (Server Side Encryption) using KMS
  - Can set the CMK (Customer Master Key) we want to use
  - SSE only encrypts the body, not the metadata (message ID, timestamp, attributes)
- IAM policy must allow usage of SQS
- SQS queue access policy
  - Finer grained control over IP
  - Control over the time the requests come in

# Kinesis Data Stream vs SQS

- **Kinesis Data Stream:**

- Data can be consumed many times
- Data is deleted after the retention period
- Ordering of records is preserved (at the shard level) – even during replays
- Build multiple applications reading from the same stream independently (Pub/Sub)
- “Streaming MapReduce” querying capability (Spark, Flink...)
- Checkpointing needed to track progress of consumption (ex: KCL with DynamoDB)
- Provisioned mode or on-demand mode

- **SQS:**

- Queue, decouple applications
- One application per queue
- Records are deleted after consumption (ack / fail)
- Messages are processed independently for standard queue
- Ordering for FIFO queues (decreased throughput)
- Capability to “delay” messages
- Dynamic scaling of load (no-ops)

# Kinesis Data Streams vs SQS

	Kinesis Data Streams	Kinesis Data Firehose	Amazon SQS Standard	Amazon SQS FIFO
Managed by AWS	yes	yes	yes	yes
Ordering	Shard / Key	No	No	Specify Group ID
Delivery	At least once	At least once	At least once	Exactly Once
Replay	Yes	No	No	No
Max Data Retention	365 days	No	14 days	14 days
Scaling	Provision Shards: 1MB/s producer 2MB/s consumer  On-demand mode	No limit	No limit	With batching: - 3,000 msg/s - 30,000 msg/s in high throughput mode
Max Object Size	1MB	128 MB at destination	256KB (more if using extended lib)	256KB (more if using extended lib)

# SQS vs Kinesis – Use cases

- **SQS use cases:**

- Order processing
- Image Processing
- Auto scaling queues according to messages.
- Buffer and Batch messages for future processing.
- Request Offloading

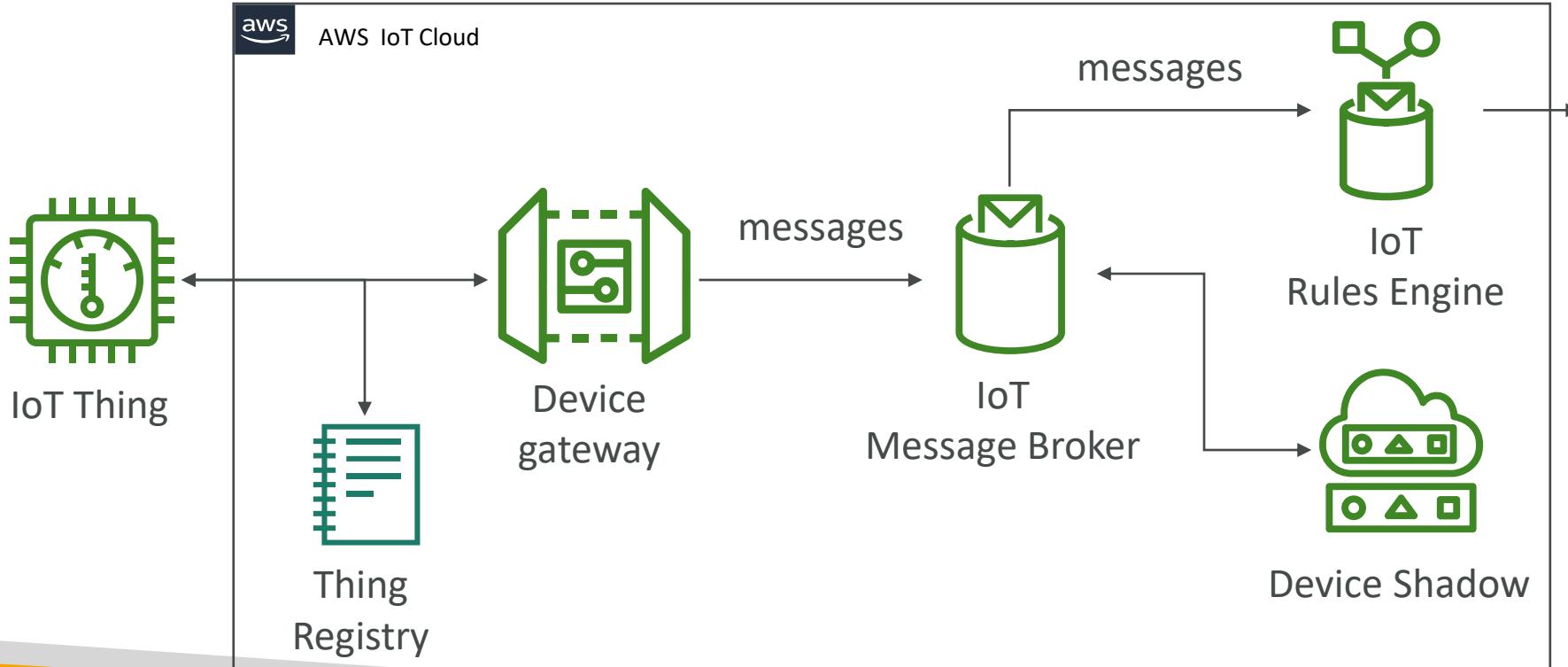
- **Kinesis Data Streams use cases:**

- Fast log and event data collection and processing
- Real Time metrics and reports
- Mobile data capture
- Real Time data analytics
- Gaming data feed
- Complex Stream Processing
- Data Feed from “Internet of Things”

# IoT Overview

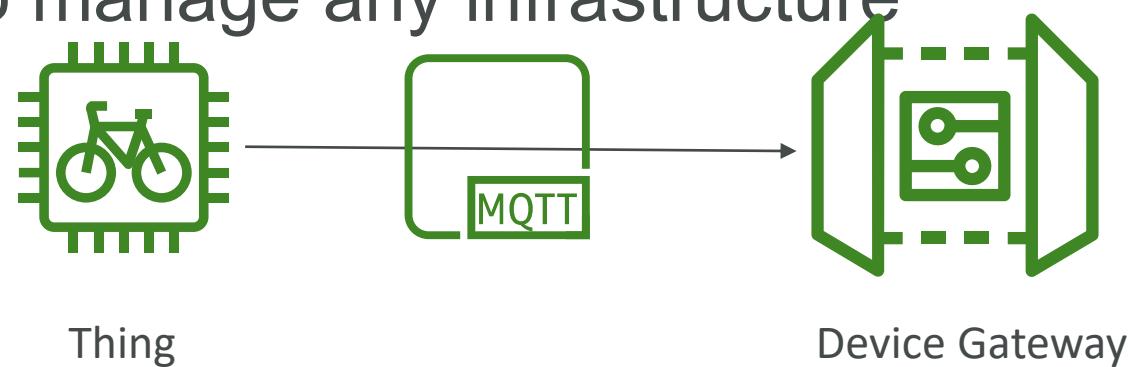


- We deploy IoT devices (“Things”)
- We configure them and retrieve data from them



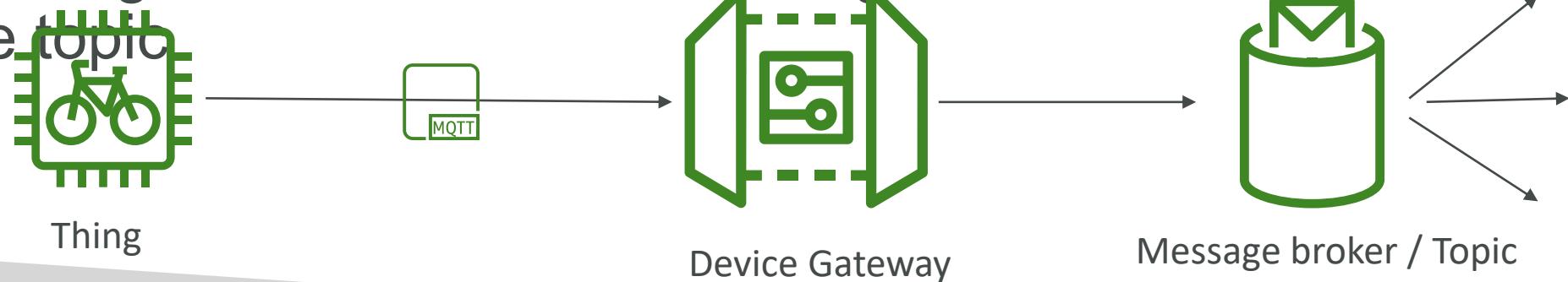
# IoT Device Gateway

- Serves as the entry point for IoT devices connecting to AWS
- Allows devices to securely and efficiently communicate with AWS IoT
- Supports the MQTT, WebSockets, and HTTP 1.1 protocols
- Fully managed and scales automatically to support over a billion devices
- No need to manage any infrastructure



# IoT Message Broker

- Pub/sub (publishers/subscribers) messaging pattern - low latency
- Devices can communicate with one another this way
- Messages sent using the MQTT, WebSockets, or HTTP 1.1 protocols
- Messages are published into topics (just like SNS)
- Message Broker forwards messages to all clients connected to the topic



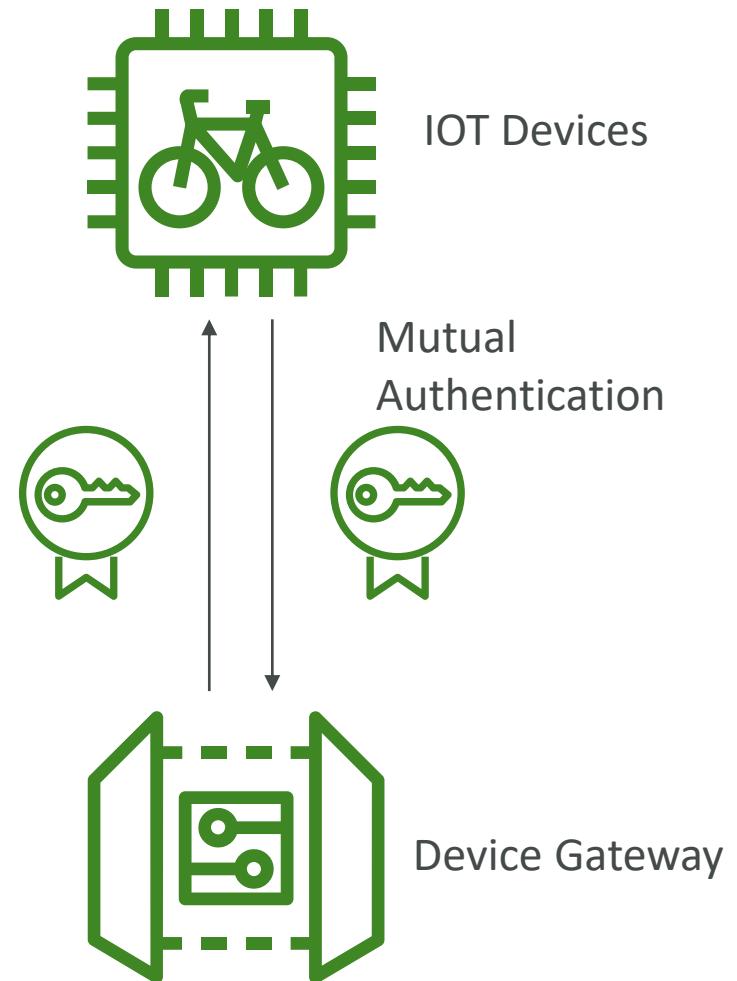
# IoT Thing Registry = IAM of IoT



- All connected IoT devices are represented in the AWS IoT registry
- Organizes the resources associated with each device in the AWS Cloud
- Each device gets a unique ID
- Supports metadata for each device (ex: Celsius vs Fahrenheit, etc...)
- Can create X.509 certificate to help IoT devices connect to AWS
- IoT Groups: group devices together and apply permissions to the group

# Authentication

- 3 possible authentication methods for Things:
  - Create X.509 certificates and load them securely onto the Things
  - AWS SigV4
  - Custom tokens with Custom authorizers
- For mobile apps:
  - Cognito identities (extension to Google, Facebook login, etc...)
- Web / Desktop / CLI:
  - IAM
  - Federated Identities

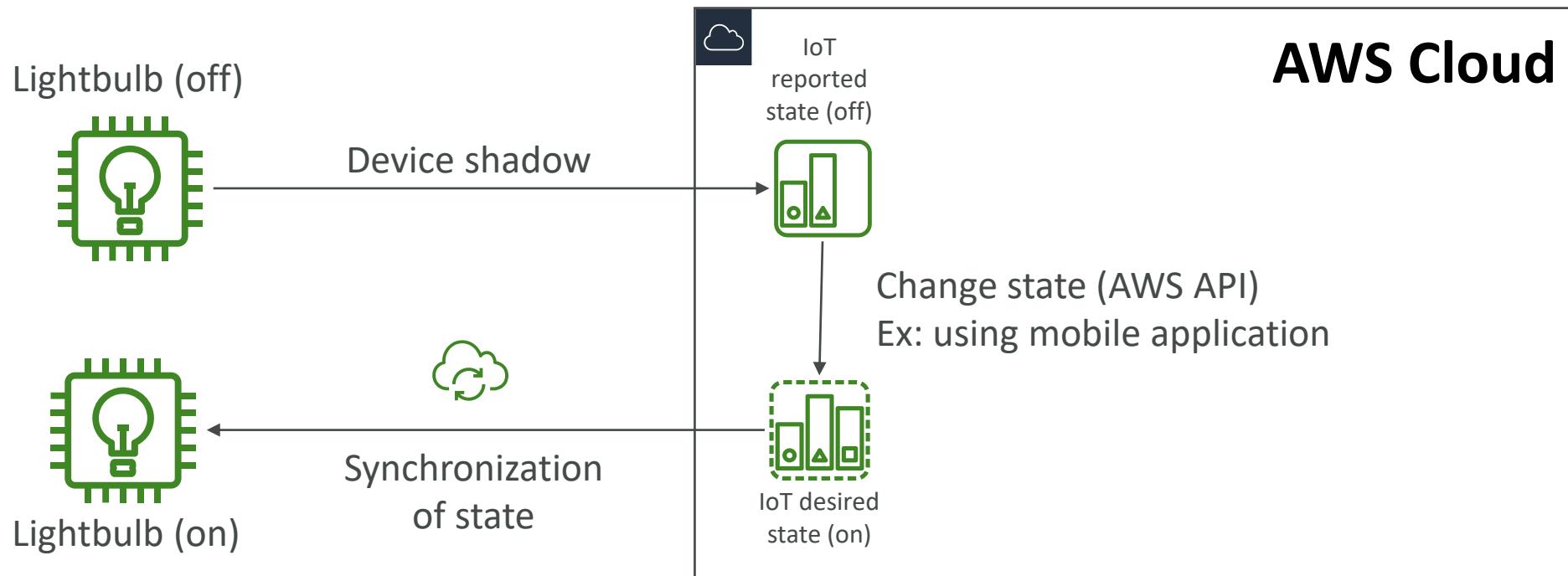


# Authorization

- AWS IoT policies:
  - Attached to X.509 certificates or Cognito Identities
  - Able to revoke any device at any time
  - IoT Policies are JSON documents
  - Can be attached to groups instead of individual Things.
- IAM Policies:
  - Attached to users, group or roles
  - Used for controlling IoT AWS APIs

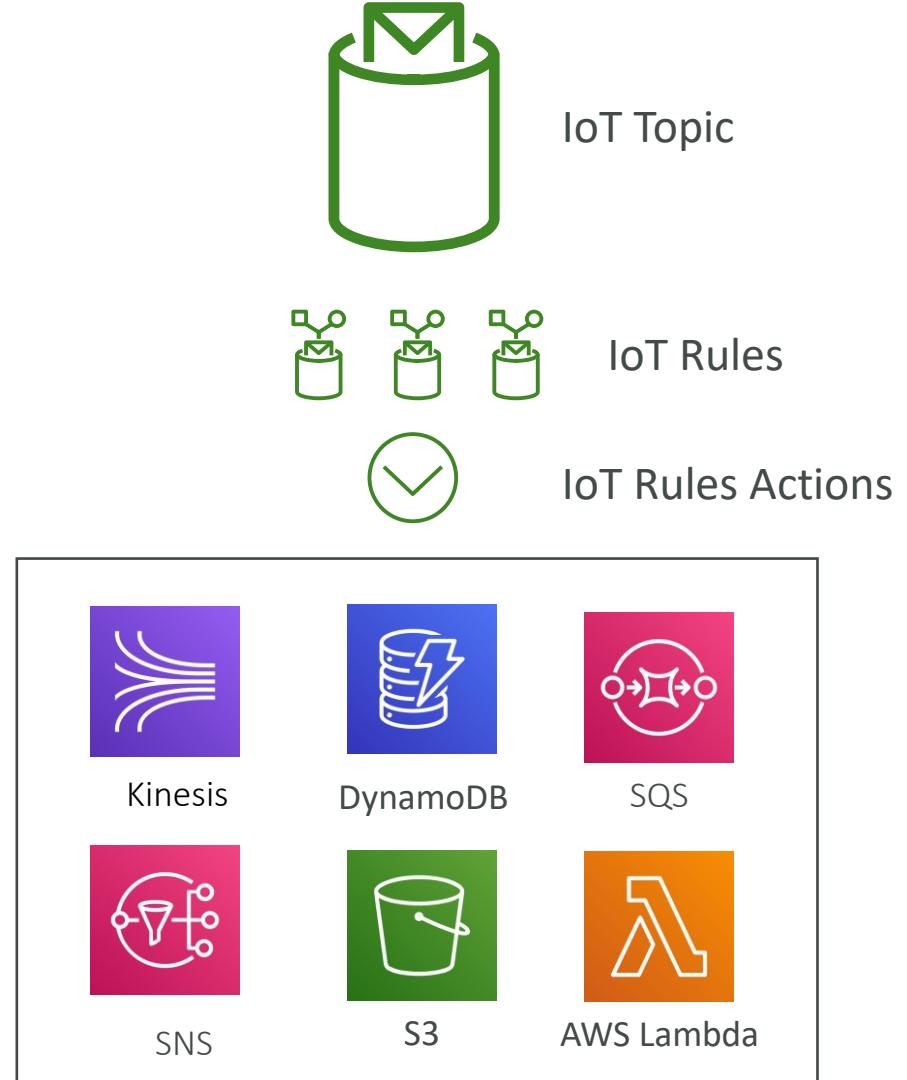
# Device Shadow

- JSON document representing the state of a connected Thing
- We can set the state to a different desired state (ex: light on)
- The IoT thing will retrieve the state when online and adapt



# Rules Engine

- Rules are defined on the MQTT topics
- Rules = when it's triggered | Action = what is does
- Rules use cases:
  - Augment or filter data received from a device
  - Write data received from a device to a **DynamoDB** database
  - Save a file to **S3**
  - Send a push notification to all users using **SNS**
  - Publish data to a **SQS** queue
  - Invoke a **Lambda** function to extract data
  - Process messages from a large number of devices using **Amazon Kinesis**
  - Send data to the Amazon **Elasticsearch Service**
  - Capture a **CloudWatch metric** and Change a **CloudWatch alarm**
  - Send the data from an MQTT message to **Amazon Machine Learning** to make predictions based on an Amazon ML model
  - & more
- Rules need IAM Roles to perform their actions



# IoT Greengrass

- IoT Greengrass brings the compute layer to the device directly
- You can execute AWS Lambda functions on the devices:
  - Pre-process the data
  - Execute predictions based on ML models
  - Keep device data in sync
  - Communicate between local devices
- Operate offline
- Deploy functions from the cloud directly to the devices

## Local Device



AWS IoT Greengrass



Lambda  
function

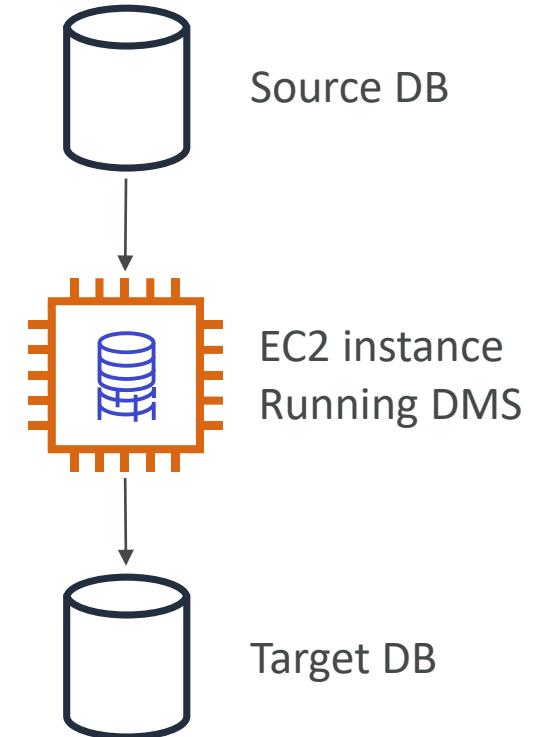


IoT thing  
coffee pot

# DMS – Database Migration Service



- Quickly and securely migrate databases to AWS, resilient, self healing
- The source database remains available during the migration
- Supports:
  - Homogeneous migrations: ex Oracle to Oracle
  - Heterogeneous migrations: ex Microsoft SQL Server to Aurora
- Continuous Data Replication using CDC
- You must create an EC2 instance to perform the replication tasks



# DMS Sources and Targets

## SOURCES:

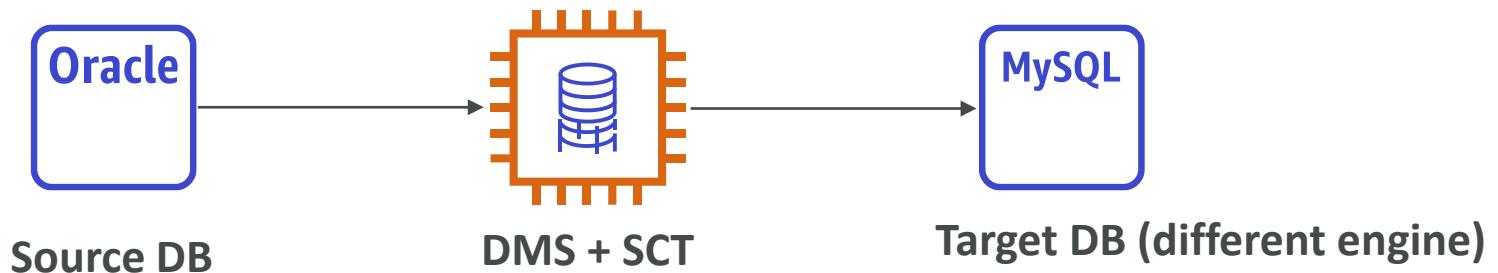
- On-Premise and EC2 instances databases: *Oracle, MS SQL Server, MySQL, MariaDB, PostgreSQL, MongoDB, SAP, DB2*
- Azure: *Azure SQL Database*
- Amazon RDS: all including Aurora
- Amazon S3

## TARGETS:

- On-Premise and EC2 instances databases: Oracle, MS SQL Server, MySQL, MariaDB, PostgreSQL, SAP
- Amazon RDS
- Amazon Redshift
- Amazon DynamoDB
- Amazon S3
- ElasticSearch Service
- Kinesis Data Streams
- DocumentDB

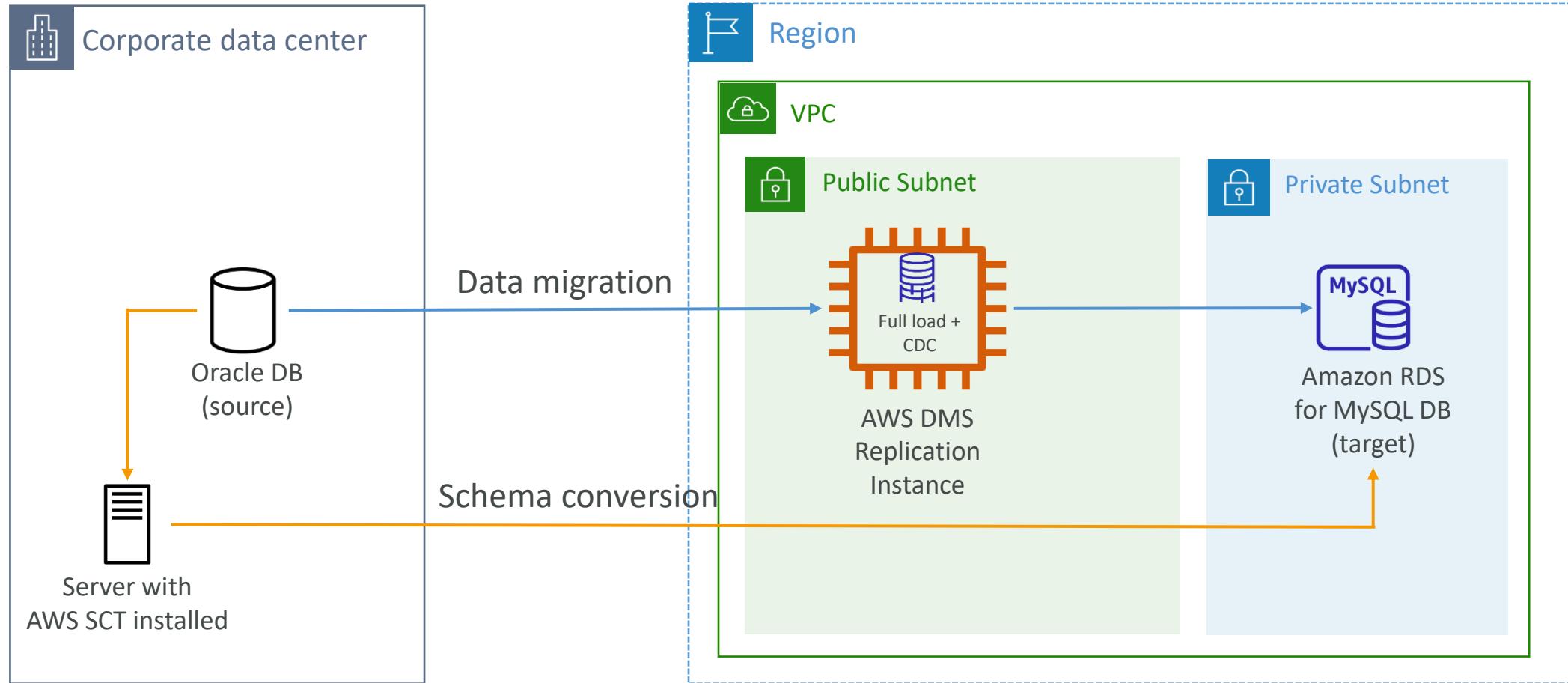
# AWS Schema Conversion Tool (SCT)

- Convert your Database's Schema from one engine to another
- Example OLTP: (SQL Server or Oracle) to MySQL, PostgreSQL, Aurora
- Example OLAP: (Teradata or Oracle) to Amazon Redshift
- Prefer compute-intensive instances to optimize data conversions



- **You do not need to use SCT if you are migrating the same DB engine**
  - Ex: On-Premise PostgreSQL => RDS PostgreSQL
  - The DB engine is still PostgreSQL (RDS is the platform)

# DMS - Continuous Replication

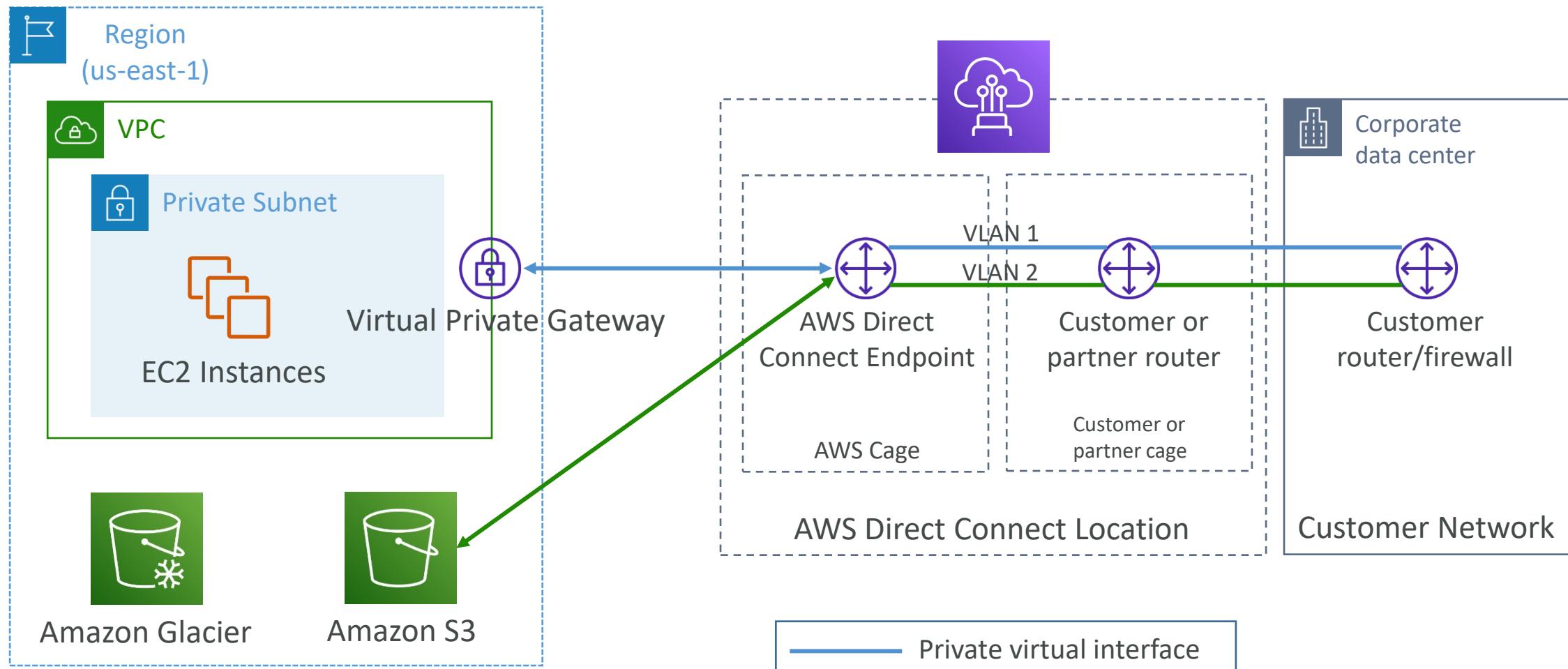


# Direct Connect (DX)



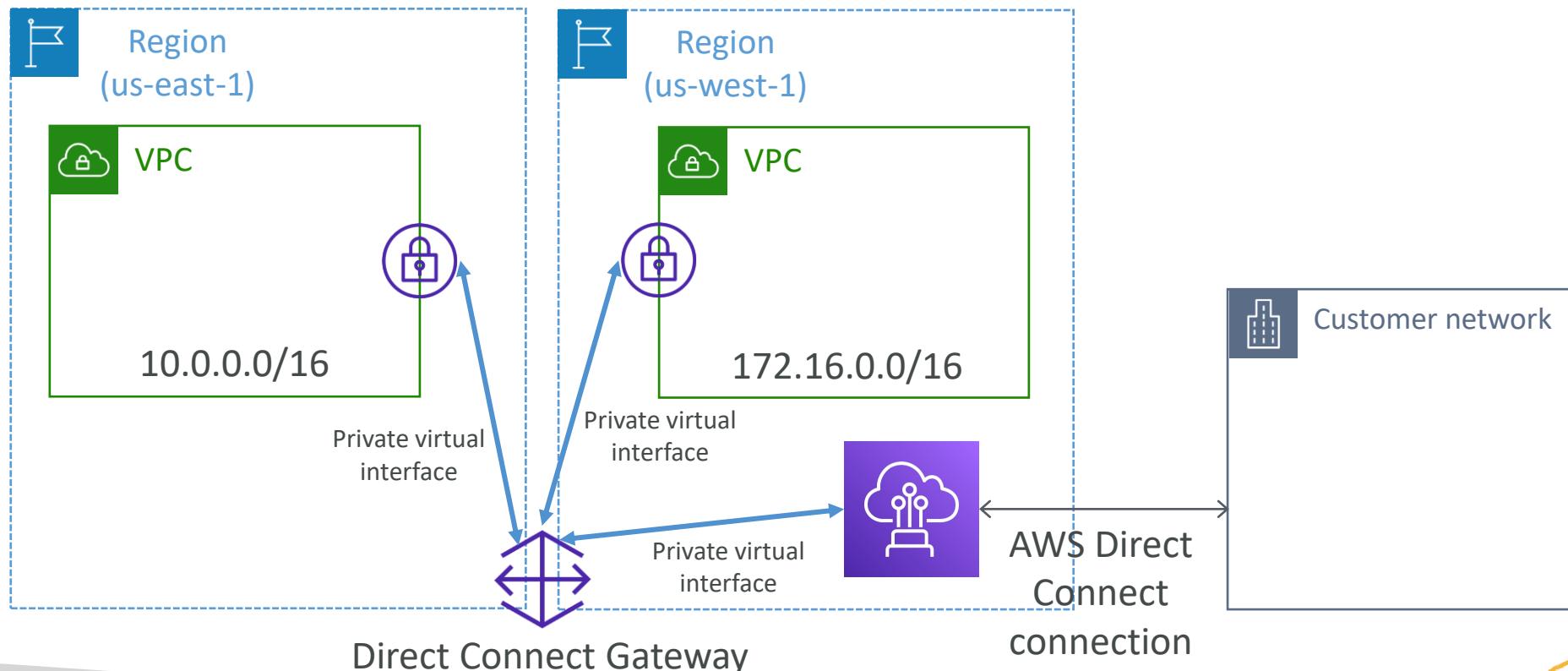
- Provides a dedicated **private** connection from a remote network to your VPC
- Dedicated connection must be setup between your DC and AWS Direct Connect locations
- You need to setup a Virtual Private Gateway on your VPC
- Access public resources (S3) and private (EC2) on same connection
- Use Cases:
  - Increase bandwidth throughput - working with large data sets – lower cost
  - More consistent network experience - applications using real-time data feeds
  - Hybrid Environments (on prem + cloud)
- Supports both IPv4 and IPv6

# Direct Connect Diagram



# Direct Connect Gateway

- If you want to setup a Direct Connect to one or more VPC in many different regions (same account), you must use a Direct Connect Gateway

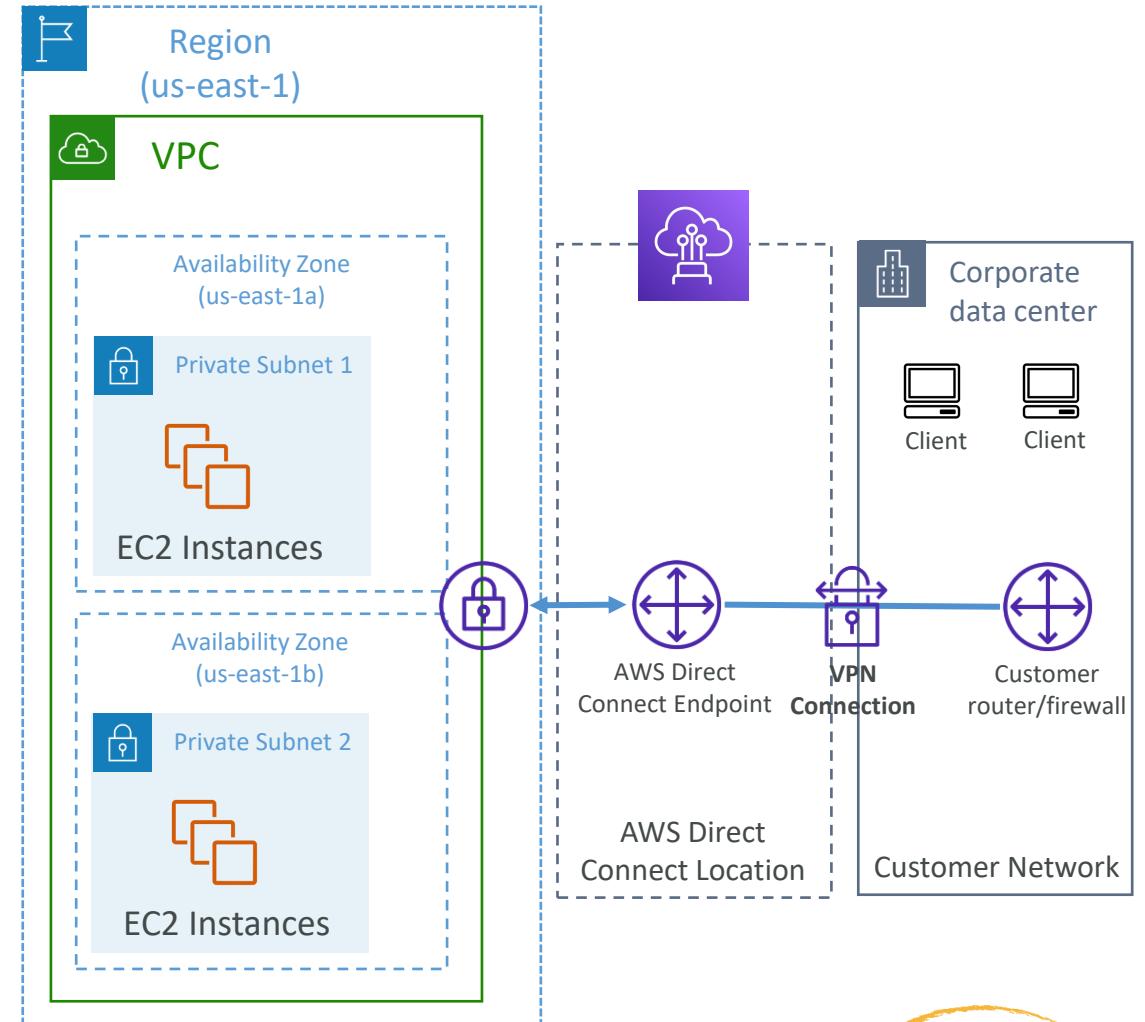


# Direct Connect – Connection Types

- **Dedicated Connections:** 1Gbps, 10 Gbps and 100 Gbps capacity
  - Physical ethernet port dedicated to a customer
  - Request made to AWS first, then completed by AWS Direct Connect Partners
- **Hosted Connections:** 50Mbps, 500 Mbps, to 10 Gbps
  - Connection requests are made via AWS Direct Connect Partners
  - Capacity can be **added or removed on demand**
  - 1, 2, 5, 10 Gbps available at select AWS Direct Connect Partners
- Lead times are often longer than 1 month to establish a new connection

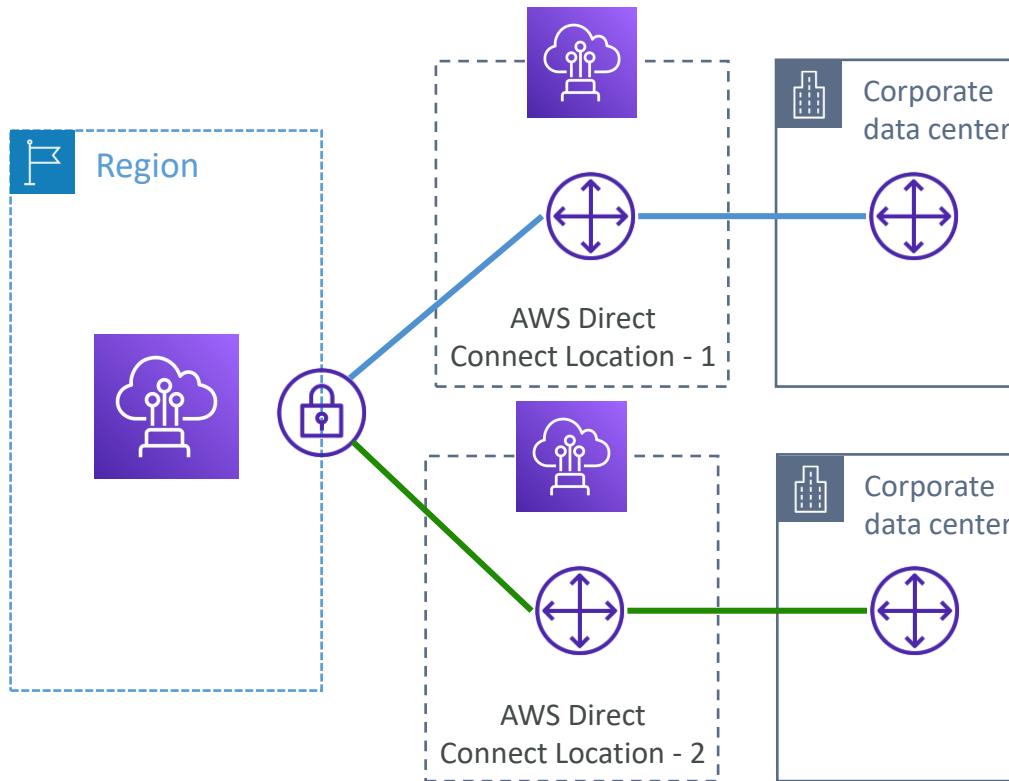
# Direct Connect – Encryption

- Data in transit is not encrypted but is private
- AWS Direct Connect + VPN provides an IPsec-encrypted private connection
- Good for an extra level of security, but slightly more complex to put in place



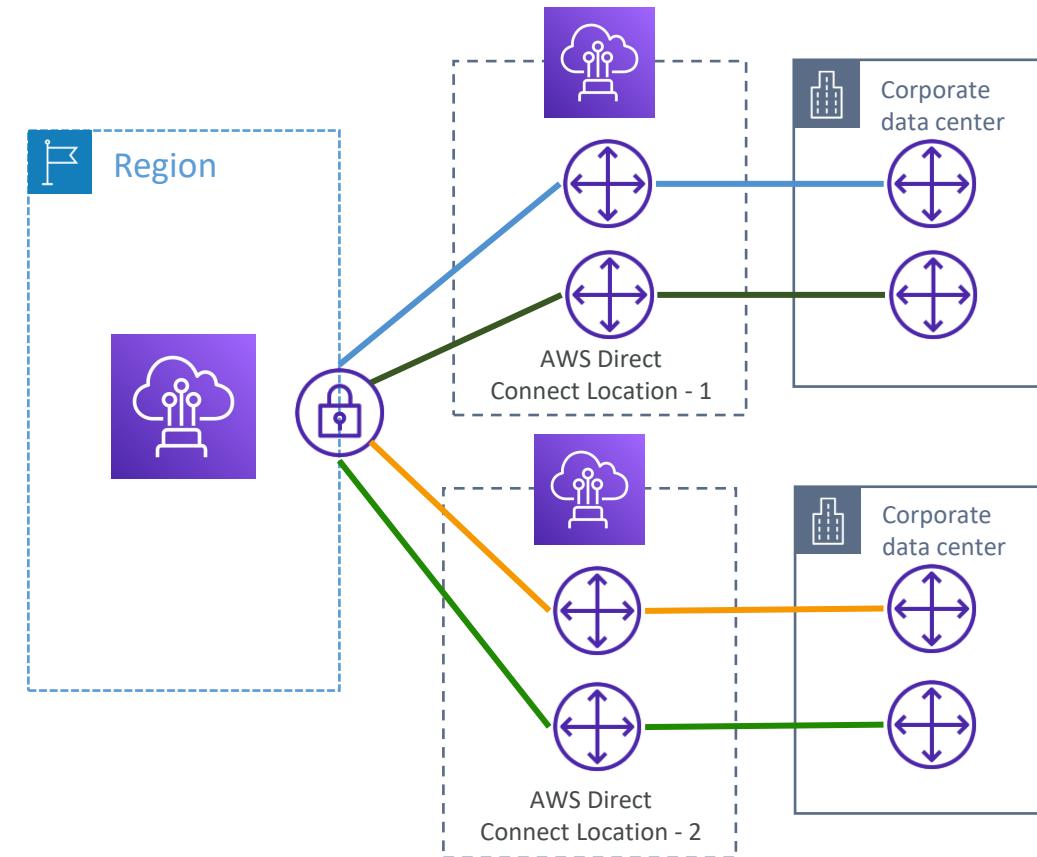
# Direct Connect - Resiliency

## High Resiliency for Critical Workloads



One connection at multiple locations

## Maximum Resiliency for Critical Workloads



Maximum resilience is achieved by separate connections terminating on separate devices in more than one location.

# AWS Snow Family

- Highly-secure, portable devices to **collect and process data at the edge, and migrate data into and out of AWS**

- **Data migration:**



Snowcone



Snowball Edge



Snowmobile

- **Edge computing**



Snowcone



Snowball Edge

# Data Migrations with AWS Snow Family

	Time to Transfer		
	100 Mbps	1Gbps	10Gbps
10 TB	12 days	30 hours	3 hours
100 TB	124 days	12 days	30 hours
1 PB	3 years	124 days	12 days

## Challenges:

- Limited connectivity
- Limited bandwidth
- High network cost
- Shared bandwidth (can't maximize the line)
- Connection stability

**AWS Snow Family: offline devices to perform data migrations**

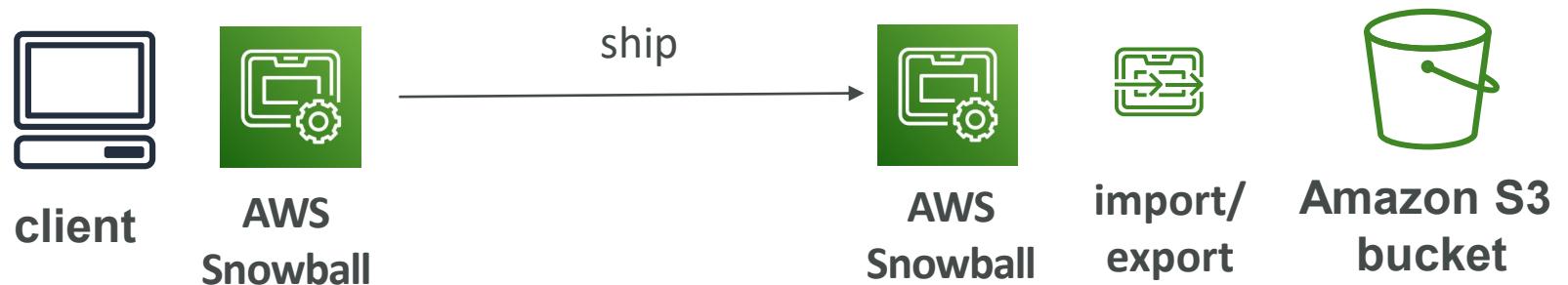
If it takes more than a week to transfer over the network, use Snowball devices!

# Diagrams

- Direct upload to S3:



- With Snow Family:



# Snowball Edge (for data transfers)



- Physical data transport solution: move TBs or PBs of data in or out of AWS
- Alternative to moving data over the network (and paying network fees)
- Pay per data transfer job
- Provide block storage and Amazon S3-compatible object storage
- **Snowball Edge Storage Optimized**
  - 80 TB of HDD capacity for block volume and S3 compatible object storage
- **Snowball Edge Compute Optimized**
  - 42 TB of HDD capacity for block volume and S3 compatible object storage
- Use cases: large data cloud migrations, DC decommission, disaster recovery



# AWS Snowcone



- **Small, portable computing, anywhere, rugged & secure, withstands harsh environments**
- Light (4.5 pounds, 2.1 kg)
- Device used for edge computing, storage, and data transfer
- **8 TBs of usable storage**
- Use Snowcone where Snowball does not fit (space-constrained environment)
- Must provide your own battery / cables
- Can be sent back to AWS offline, or connect it to internet and use **AWS DataSync** to send data



# AWS Snowmobile



- Transfer exabytes of data (1 EB = 1,000 PB = 1,000,000 TBs)
- Each Snowmobile has 100 PB of capacity (use multiple in parallel)
- High security: temperature controlled, GPS, 24/7 video surveillance
- **Better than Snowball if you transfer more than 10 PB**

# AWS Snow Family for Data Migrations



Snowcone



Snowball Edge



Snowmobile

	Snowcone	Snowball Edge Storage Optimized	Snowmobile
Storage Capacity	8 TB usable	80 TB usable	< 100 PB
Migration Size	Up to 24 TB, online and offline	Up to petabytes, offline	Up to exabytes, offline
DataSync agent	Pre-installed		
Storage Clustering		Up to 15 nodes	

# Snow Family – Usage Process

1. Request Snowball devices from the AWS console for delivery
2. Install the snowball client / AWS OpsHub on your servers
3. Connect the snowball to your servers and copy files using the client
4. Ship back the device when you're done (goes to the right AWS facility)
5. Data will be loaded into an S3 bucket
6. Snowball is completely wiped

# What is Edge Computing?

- Process data while it's being created on **an edge location**
  - A truck on the road, a ship on the sea, a mining station underground...



- These locations may have
  - Limited / no internet access
  - Limited / no easy access to computing power
- We setup a **Snowball Edge / Snowcone** device to do edge computing
- Use cases of Edge Computing:
  - Preprocess data
  - Machine learning at the edge
  - Transcoding media streams
- Eventually (if need be) we can ship back the device to AWS (for transferring data for example)

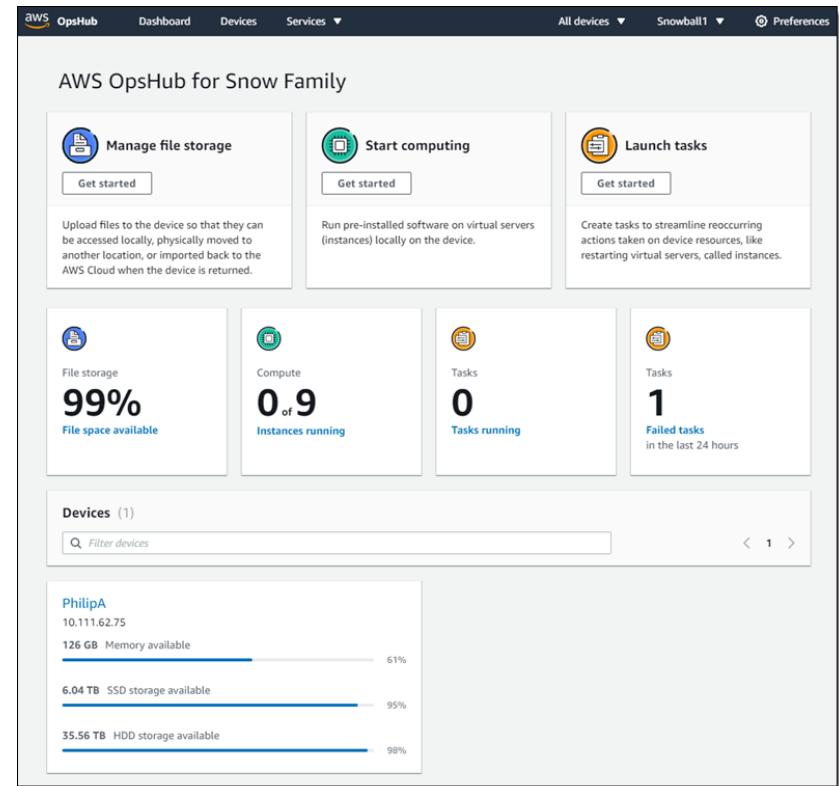
# Snow Family – Edge Computing

- **Snowcone (smaller)**
  - 2 CPUs, 4 GB of memory, wired or wireless access
  - USB-C power using a cord or the optional battery
- **Snowball Edge – Compute Optimized**
  - 52 vCPUs, 208 GiB of RAM
  - Optional GPU (useful for video processing or machine learning)
  - 42 TB usable storage
- **Snowball Edge – Storage Optimized**
  - Up to 40 vCPUs, 80 GiB of RAM
  - Object storage clustering available
- All: Can run EC2 Instances & AWS Lambda functions (using AWS IoT Greengrass)
- Long-term deployment options: 1 and 3 years discounted pricing



# AWS OpsHub

- Historically, to use Snow Family devices, you needed a CLI (Command Line Interface tool)
- Today, you can use **AWS OpsHub** (a software you install on your computer / laptop) to manage your Snow Family Device
  - Unlocking and configuring single or clustered devices
  - Transferring files
  - Launching and managing instances running on Snow Family Devices
  - Monitor device metrics (storage capacity, active instances on your device)
  - Launch compatible AWS services on your devices (ex: Amazon EC2 instances, AWS DataSync, Network File System (NFS))



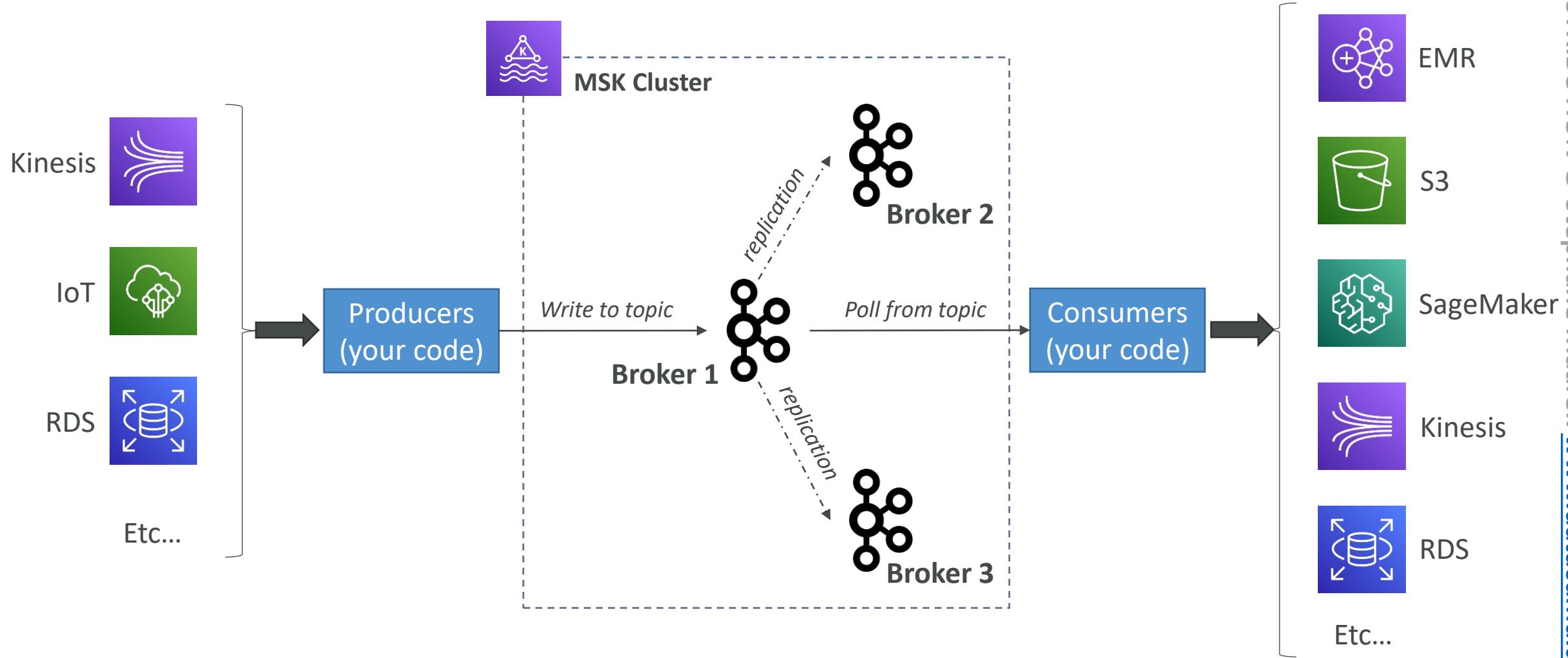
<https://aws.amazon.com/blogs/aws/aws-snowball-edge-update/>

# Amazon Managed Streaming for Apache Kafka (Amazon MSK)



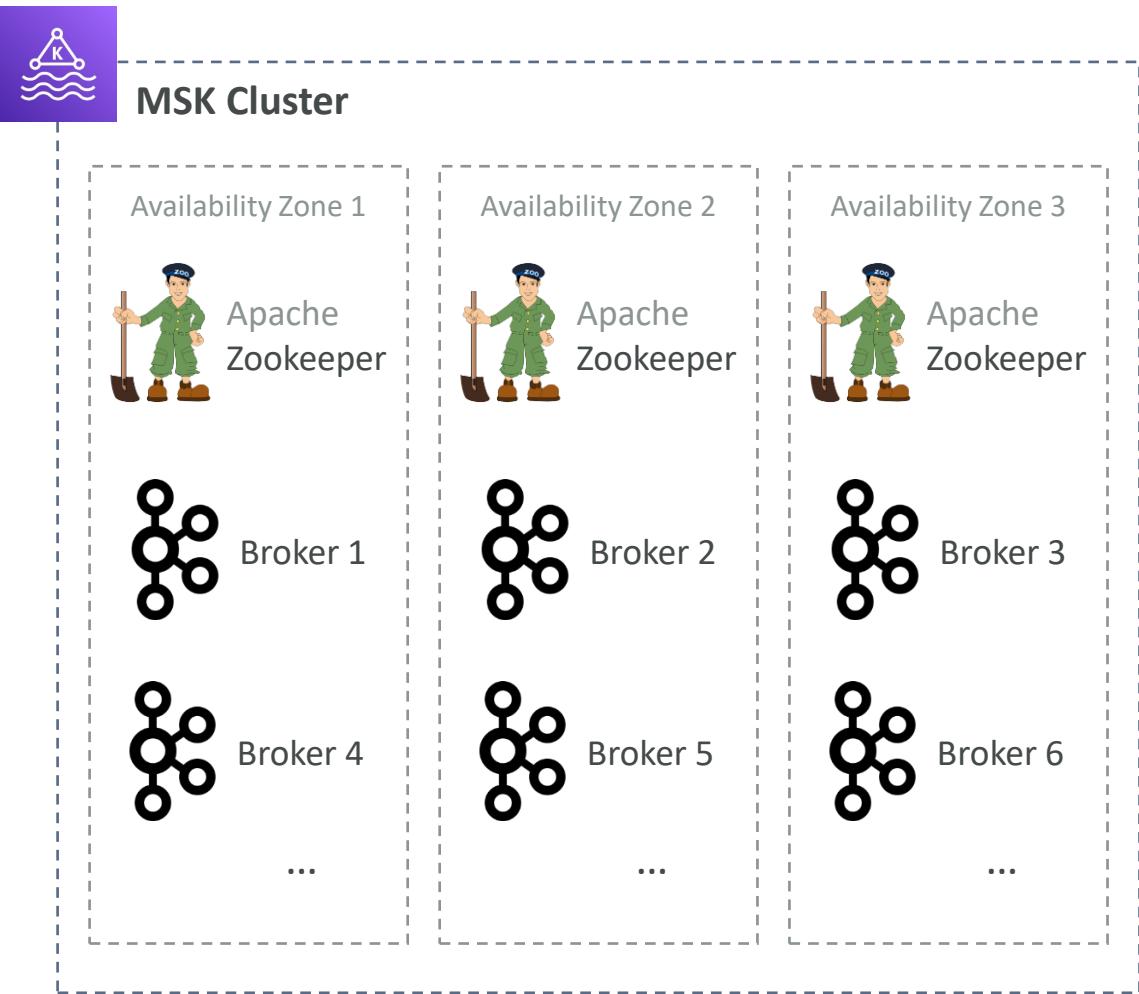
- Alternative to Kinesis (Kafka vs Kinesis next lecture)
- Fully managed Apache Kafka on AWS
  - Allow you to create, update, delete clusters
  - MSK creates & manages Kafka brokers nodes & Zookeeper nodes for you
  - Deploy the MSK cluster in your VPC, multi-AZ (up to 3 for HA)
  - Automatic recovery from common Apache Kafka failures
  - Data is stored on EBS volumes
- You can build producers and consumers of data
- Can create custom configurations for your clusters
  - Default message size of 1MB
  - **Possibilities of sending large messages (ex: 10MB) into Kafka after custom configuration**

# Apache Kafka at a high level



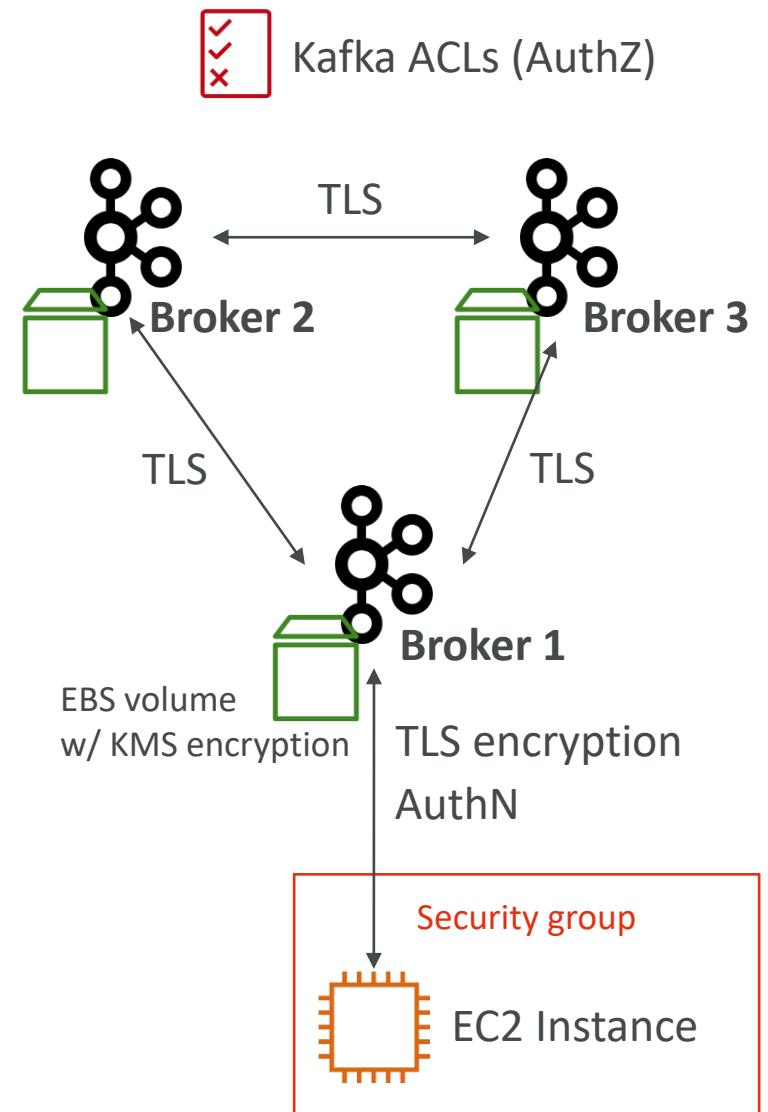
# MSK – Configurations

- Choose the number of AZ (3 – recommended, or 2)
- Choose the VPC & Subnets
- The broker instance type (ex: kafka.m5.large)
- The number of brokers per AZ (can add brokers later)
- Size of your EBS volumes (1GB – 16TB)



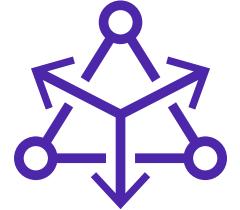
# MSK – Security

- Encryption:**
  - Optional in-flight using TLS between the brokers
  - Optional in-flight with TLS between the clients and brokers
  - At rest for your EBS volumes using KMS
- Network Security:**
  - Authorize specific security groups for your Apache Kafka clients
- Authentication & Authorization (important):**
  - Define who can read/write to which topics
  - Mutual TLS (AuthN) + Kafka ACLs (AuthZ)
  - SASL/SCRAM (AuthN) + Kafka ACLs (AuthZ)
  - IAM Access Control (AuthN + AuthZ)



# MSK – Monitoring

- **CloudWatch Metrics**
  - Basic monitoring (cluster and broker metrics)
  - Enhanced monitoring (++enhanced broker metrics)
  - Topic-level monitoring (++enhanced topic-level metrics)
- **Prometheus (Open-Source Monitoring)**
  - Opens a port on the broker to export cluster, broker and topic-level metrics
  - Setup the JMX Exporter (metrics) or Node Exporter (CPU and disk metrics)
- **Broker Log Delivery**
  - Delivery to CloudWatch Logs
  - Delivery to Amazon S3
  - Delivery to Kinesis Data Streams



# MSK Connect

- Managed Kafka Connect workers on AWS
- Auto-scaling capabilities for workers
- You can deploy any Kafka Connect connectors to MSK Connect as a **plugin**
  - Amazon S3, Amazon Redshift, Amazon OpenSearch, Debezium, etc...
- Example pricing: Pay \$0.11 per worker per hour



# MSK Serverless

- Run Apache Kafka on MSK without managing the capacity
- MSK automatically provisions resources and scales compute & storage
- You just define your topics and your partitions and you're good to go!
- Security: IAM Access Control for all clusters
- Example Pricing:
  - \$0.75 per cluster per hour = \$558 monthly per cluster
  - \$0.0015 per partition per hour = \$1.08 monthly per partition
  - \$0.10 per GB of storage each month
  - \$0.10 per GB in
  - \$0.05 per GB out

# Kinesis Data Streams vs Amazon MSK



## Kinesis Data Streams

- 1 MB message size limit
- Data Streams with Shards
- Shard Splitting & Merging
- TLS In-flight encryption
- KMS At-rest encryption
- Security:
  - IAM policies for AuthN/AuthZ



## Amazon MSK

- 1MB default, configure for higher (ex: 10MB)
- Kafka Topics with Partitions
- Can only add partitions to a topic
- PLAINTEXT or TLS In-flight Encryption
- KMS At-rest encryption
- Security:
  - Mutual TLS (AuthN) + Kafka ACLs (AuthZ)
  - SASL/SCRAM (AuthN) + Kafka ACLs (AuthZ)
  - IAM Access Control (AuthN + AuthZ)

# Amazon S3 Section

# Section introduction



- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage
- Many websites use Amazon S3 as a backbone
- Many AWS services use Amazon S3 as an integration as well
- We'll have a step-by-step approach to S3

# Amazon S3 Use cases

- Backup and storage
- Disaster Recovery
- Archive
- Hybrid Cloud storage
- Application hosting
- Media hosting
- Data lakes & big data analytics
- Software delivery
- Static website



Nasdaq stores 7 years of data into S3 Glacier



Sysco runs analytics on its data and gain business insights

# Amazon S3 - Buckets

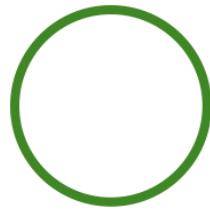
- Amazon S3 allows people to store objects (files) in “buckets” (directories)
- Buckets must have a **globally unique name (across all regions all accounts)**
- Buckets are defined at the region level
- S3 looks like a global service but buckets are created in a region
- Naming convention
  - No uppercase, No underscore
  - 3-63 characters long
  - Not an IP
  - Must start with lowercase letter or number
  - Must NOT start with the prefix **xn--**
  - Must NOT end with the suffix **-s3alias**



S3 Bucket

# Amazon S3 - Objects

- Objects (files) have a Key
- The **key** is the **FULL** path:
  - s3://my-bucket/**my\_file.txt**
  - s3://my-bucket/**my\_folder1/another\_folder/my\_file.txt**
- The key is composed of **prefix** + **object name**
  - s3://my-bucket/**my\_folder1/another\_folder**/**my\_file.txt**
- There's no concept of "directories" within buckets (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("/")

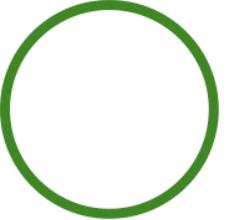


Object



S3 Bucket  
with Objects

# Amazon S3 – Objects (cont.)



- Object values are the content of the body:
  - Max. Object Size is 5TB (5000GB)
  - If uploading more than 5GB, must use “multi-part upload”
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

# Amazon S3 – Security

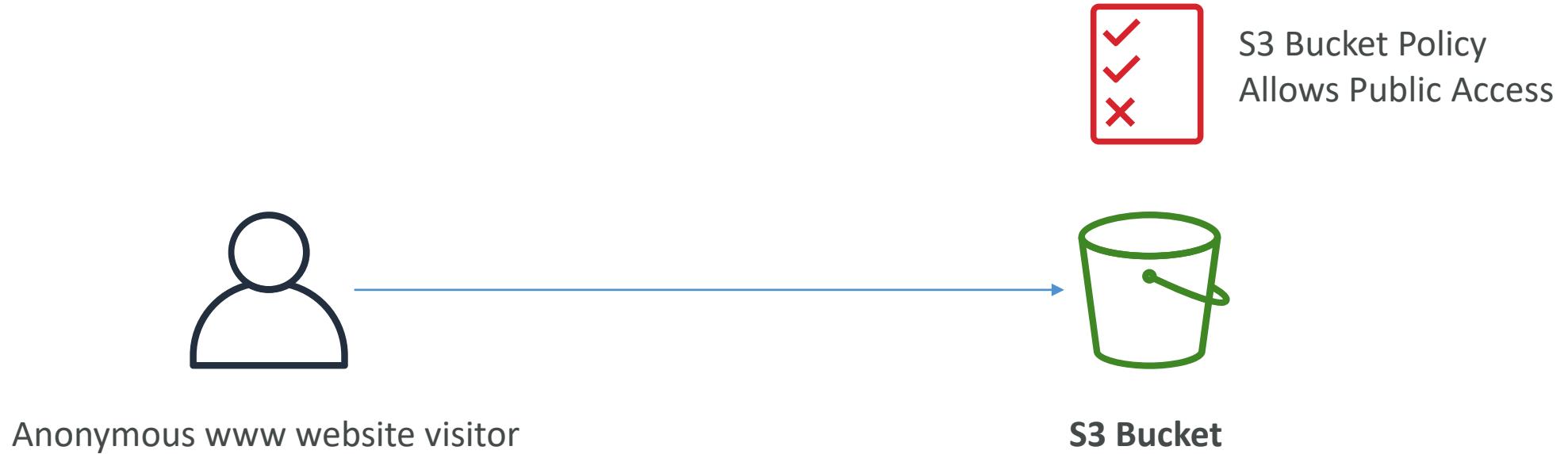
- **User-Based**
  - **IAM Policies** – which API calls should be allowed for a specific user from IAM
- **Resource-Based**
  - **Bucket Policies** – bucket wide rules from the S3 console - allows cross account
  - **Object Access Control List (ACL)** – finer grain (can be disabled)
  - **Bucket Access Control List (ACL)** – less common (can be disabled)
- **Note:** an IAM principal can access an S3 object if
  - The user IAM permissions ALLOW it OR the resource policy ALLOWS it
  - AND there's no explicit DENY
- **Encryption:** encrypt objects in Amazon S3 using encryption keys

# S3 Bucket Policies

- JSON based policies
  - Resources: buckets and objects
  - Effect: Allow / Deny
  - Actions: Set of API to Allow or Deny
  - Principal: The account or user to apply the policy to
- Use S3 bucket for policy to:
  - Grant public access to the bucket
  - Force objects to be encrypted at upload
  - Grant access to another account (Cross Account)

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "PublicRead",  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": [  
        "s3:GetObject"  
      ],  
      "Resource": [  
        "arn:aws:s3:::examplebucket/*"  
      ]  
    }  
  ]  
}
```

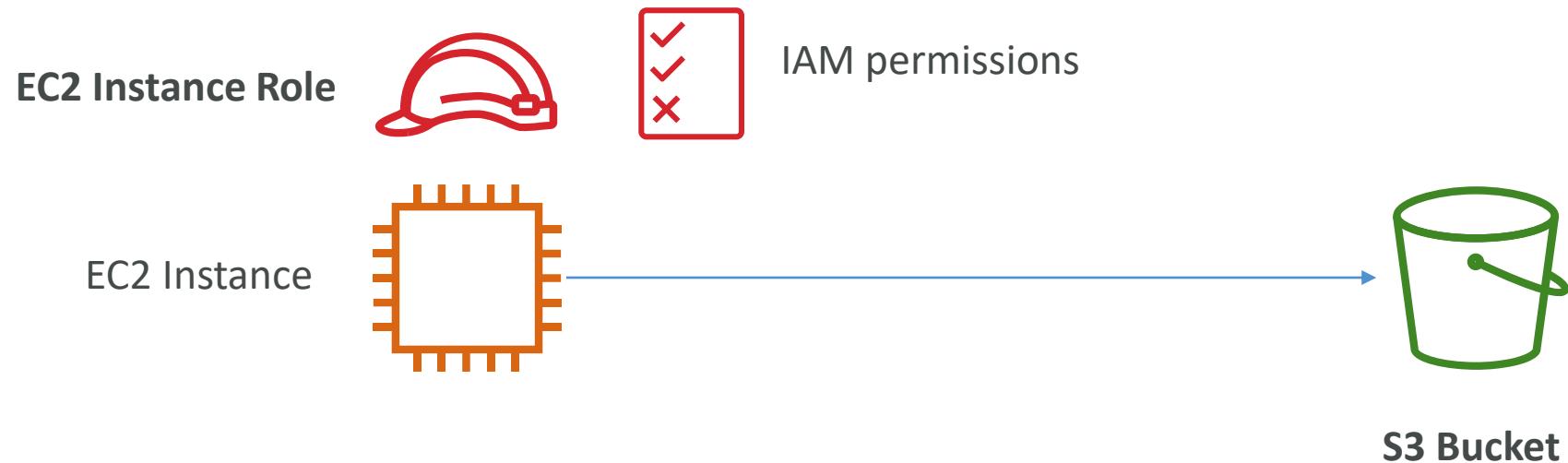
# Example: Public Access - Use Bucket Policy



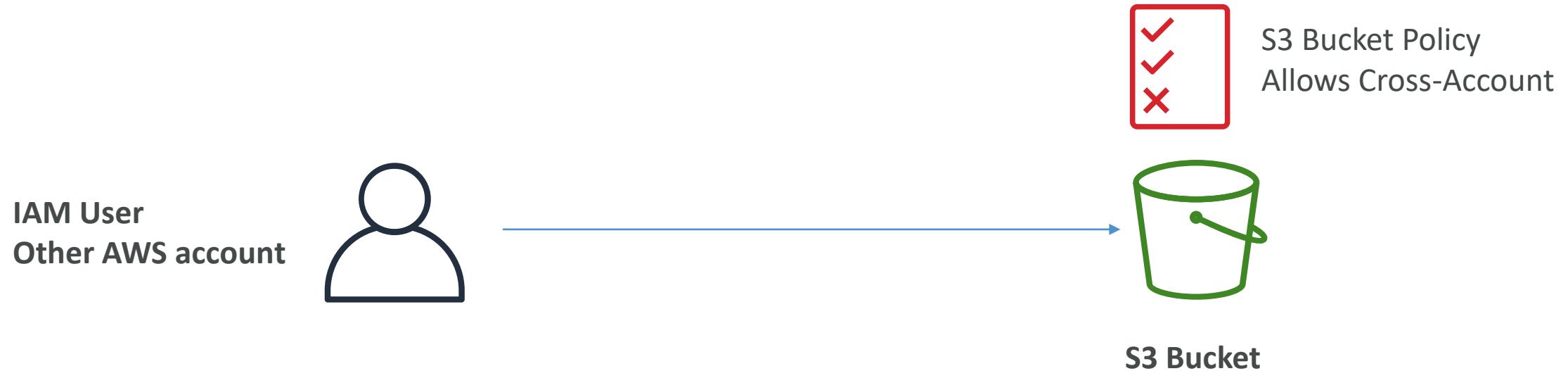
# Example: User Access to S3 – IAM permissions



# Example: EC2 instance access - Use IAM Roles



# Advanced: Cross-Account Access – Use Bucket Policy



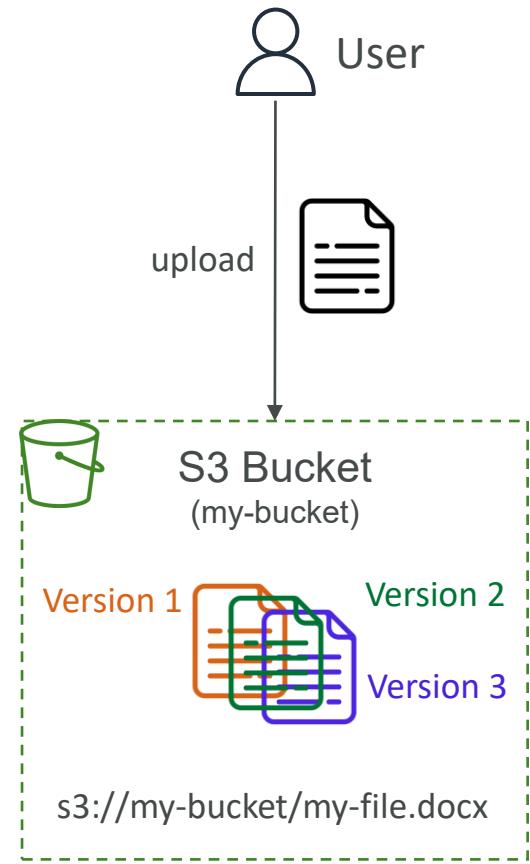
# Bucket settings for Block Public Access

```
Block all public access
On
  └─ Block public access to buckets and objects granted through new access control lists (ACLs)
      On
  └─ Block public access to buckets and objects granted through any access control lists (ACLs)
      On
  └─ Block public access to buckets and objects granted through new public bucket or access point policies
      On
  └─ Block public and cross-account access to buckets and objects through any public bucket or access point policies
      On
```

- These settings were created to prevent company data leaks
- If you know your bucket should never be public, leave these on
- Can be set at the account level

# Amazon S3 - Versioning

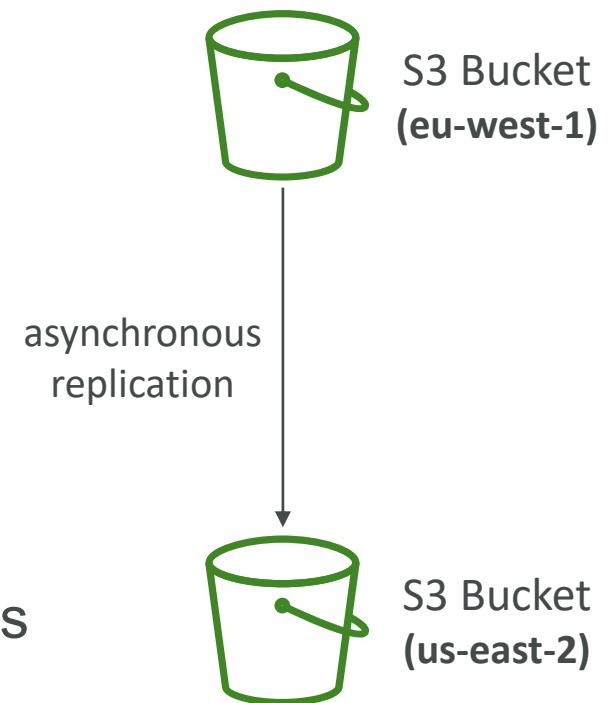
- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will change the “version”: 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - Any file that is not versioned prior to enabling versioning will have version “null”
  - Suspending versioning does not delete the previous versions



# Amazon S3 – Replication (CRR & SRR)



- Must enable Versioning in source and destination buckets
- Cross-Region Replication (CRR)
- Same-Region Replication (SRR)
- Buckets can be in different AWS accounts
- Copying is asynchronous
- Must give proper IAM permissions to S3
- Use cases:
  - CRR – compliance, lower latency access, replication across accounts
  - SRR – log aggregation, live replication between production and test accounts



# Amazon S3 – Replication (Notes)

- After you enable Replication, only new objects are replicated
- Optionally, you can replicate existing objects using **S3 Batch Replication**
  - Replicates existing objects and objects that failed replication
- For DELETE operations
  - **Can replicate delete markers** from source to target (optional setting)
  - Deletions with a version ID are not replicated (to avoid malicious deletes)
- **There is no “chaining” of replication**
  - If bucket 1 has replication into bucket 2, which has replication into bucket 3
  - Then objects created in bucket 1 are not replicated to bucket 3

# S3 Storage Classes

- Amazon S3 Standard - General Purpose
- Amazon S3 Standard-Infrequent Access (IA)
- Amazon S3 One Zone-Infrequent Access
- Amazon S3 Glacier Instant Retrieval
- Amazon S3 Glacier Flexible Retrieval
- Amazon S3 Glacier Deep Archive
- Amazon S3 Intelligent Tiering
  
- Can move between classes manually or using S3 Lifecycle configurations

# S3 Durability and Availability

- Durability:
  - High durability (99.99999999%, 11 9's) of objects across multiple AZ
  - If you store 10,000,000 objects with Amazon S3, you can on average expect to incur a loss of a single object once every 10,000 years
  - Same for all storage classes
- Availability:
  - Measures how readily available a service is
  - Varies depending on storage class
  - Example: S3 standard has 99.99% availability = not available 53 minutes a year



# S3 Standard – General Purpose

- 99.99% Availability
- Used for frequently accessed data
- Low latency and high throughput
- Sustain 2 concurrent facility failures
  
- Use Cases: Big Data analytics, mobile & gaming applications, content distribution...

# S3 Storage Classes – Infrequent Access

- For data that is less frequently accessed, but requires rapid access when needed
- Lower cost than S3 Standard
- **Amazon S3 Standard-Infrequent Access (S3 Standard-IA)**
  - 99.9% Availability
  - Use cases: Disaster Recovery, backups
- **Amazon S3 One Zone-Infrequent Access (S3 One Zone-IA)**
  - High durability (99.99999999%) in a single AZ; data lost when AZ is destroyed
  - 99.5% Availability
  - Use Cases: Storing secondary backup copies of on-premises data, or data you can recreate



# Amazon S3 Glacier Storage Classes

- Low-cost object storage meant for archiving / backup
- Pricing: price for storage + object retrieval cost
- **Amazon S3 Glacier Instant Retrieval**
  - Millisecond retrieval, great for data accessed once a quarter
  - Minimum storage duration of 90 days
- **Amazon S3 Glacier Flexible Retrieval** (formerly Amazon S3 Glacier):
  - Expedited (1 to 5 minutes), Standard (3 to 5 hours), Bulk (5 to 12 hours) – free
  - Minimum storage duration of 90 days
- **Amazon S3 Glacier Deep Archive – for long term storage:**
  - Standard (12 hours), Bulk (48 hours)
  - Minimum storage duration of 180 days





# S3 Intelligent-Tiering

- Small monthly monitoring and auto-tiering fee
  - Moves objects automatically between Access Tiers based on usage
  - There are no retrieval charges in S3 Intelligent-Tiering
- 
- *Frequent Access tier (automatic)*: default tier
  - *Infrequent Access tier (automatic)*: objects not accessed for 30 days
  - *Archive Instant Access tier (automatic)*: objects not accessed for 90 days
  - *Archive Access tier (optional)*: configurable from 90 days to 700+ days
  - *Deep Archive Access tier (optional)*: config. from 180 days to 700+ days

# S3 Storage Classes Comparison

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Durability	99.999999999% == (11 9's)						
Availability	99.99%	99.9%	99.9%	99.5%	99.9%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99%	99.9%	99.9%
Availability Zones	>= 3	>= 3	>= 3	1	>= 3	>= 3	>= 3
Min. Storage Duration Charge	None	None	30 Days	30 Days	90 Days	90 Days	180 Days
Min. Billable Object Size	None	None	128 KB	128 KB	128 KB	40 KB	40 KB
Retrieval Fee	None	None	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved	Per GB retrieved

# S3 Storage Classes – Price Comparison

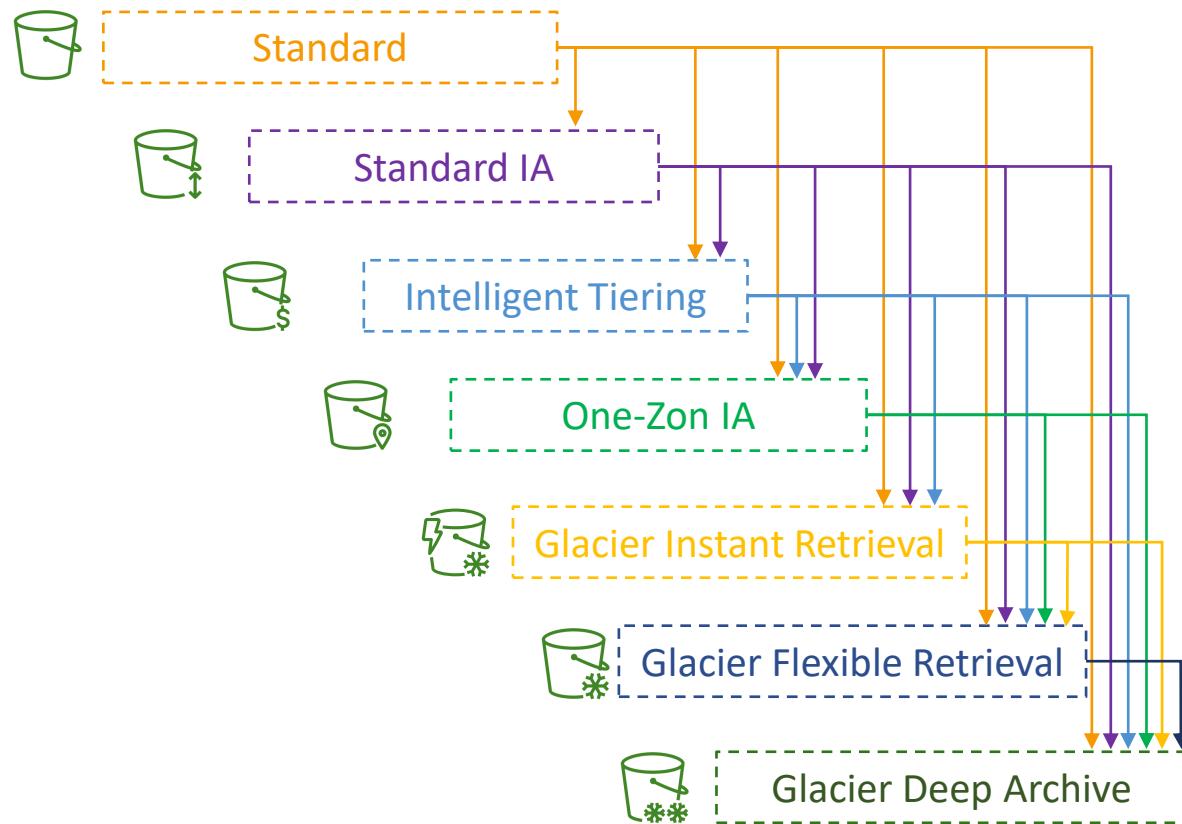
## Example: us-east-1

	Standard	Intelligent-Tiering	Standard-IA	One Zone-IA	Glacier Instant Retrieval	Glacier Flexible Retrieval	Glacier Deep Archive
Storage Cost (per GB per month)	\$0.023	\$0.0025 - \$0.023	%0.0125	\$0.01	\$0.004	\$0.0036	\$0.00099
Retrieval Cost (per 1000 request)	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.005	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.005	<b>GET:</b> \$0.001 <b>POST:</b> \$0.01	<b>GET:</b> \$0.001 <b>POST:</b> \$0.01	<b>GET:</b> \$0.01 <b>POST:</b> \$0.02	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.03  <b>Expedited:</b> \$10 <b>Standard:</b> \$0.05 <b>Bulk:</b> free	<b>GET:</b> \$0.0004 <b>POST:</b> \$0.05  <b>Standard:</b> \$0.10 <b>Bulk:</b> \$0.025
Retrieval Time	Instantaneous						<b>Expedited</b> (1 – 5 mins) <b>Standard</b> (3 – 5 hours) <b>Bulk</b> (5 – 12 hours)
Monitoring Cost (pet 1000 objects)		\$0.0025					

<https://aws.amazon.com/s3/pricing/>

# Amazon S3 – Moving between Storage Classes

- You can transition objects between storage classes
- For infrequently accessed object, move them to **Standard IA**
- For archive objects that you don't need fast access to, move them to **Glacier or Glacier Deep Archive**
- Moving objects can be automated using a **Lifecycle Rules**



# Amazon S3 – Lifecycle Rules



- **Transition Actions** – configure objects to transition to another storage class
  - Move objects to Standard IA class 60 days after creation
  - Move to Glacier for archiving after 6 months
- **Expiration actions** – configure objects to expire (delete) after some time
  - Access log files can be set to delete after a 365 days
  - **Can be used to delete old versions of files (if versioning is enabled)**
  - Can be used to delete incomplete Multi-Part uploads
- Rules can be created for a certain prefix (example: s3://mybucket/mp3/\*)
- Rules can be created for certain objects Tags (example: *Department: Finance*)

# Amazon S3 – Lifecycle Rules (Scenario 1)

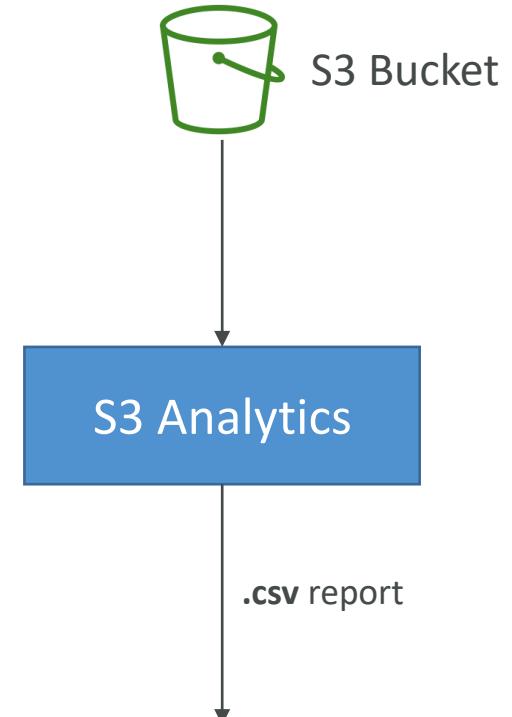
- Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 hours. How would you design this?
- S3 source images can be on **Standard**, with a lifecycle configuration to transition them to **Glacier** after 60 days
- S3 thumbnails can be on **One-Zone IA**, with a lifecycle configuration to expire them (delete them) after 60 days

# Amazon S3 – Lifecycle Rules (Scenario 2)

- A rule in your company states that you should be able to recover your deleted S3 objects immediately for 30 days, although this may happen rarely. After this time, and for up to 365 days, deleted objects should be recoverable within 48 hours.
- **Enable S3 Versioning** in order to have object versions, so that “deleted objects” are in fact hidden by a “delete marker” and can be recovered
- Transition the “noncurrent versions” of the object to **Standard IA**
- Transition afterwards the “noncurrent versions” to **Glacier Deep Archive**

# Amazon S3 Analytics – Storage Class Analysis

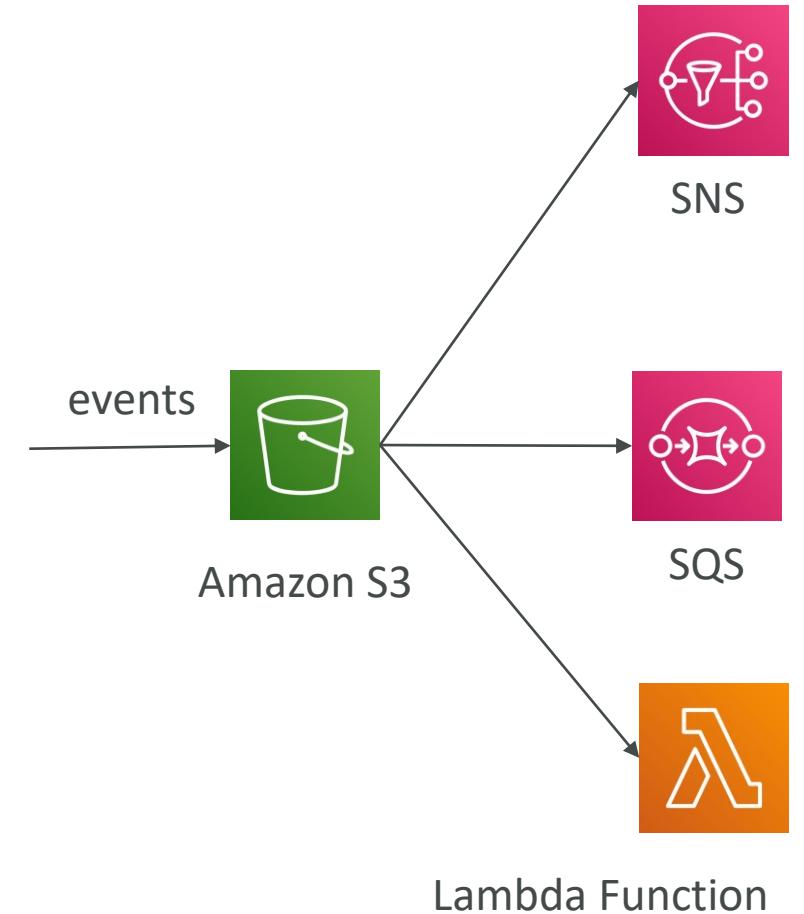
- Help you decide when to transition objects to the right storage class
- Recommendations for **Standard** and **Standard IA**
  - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis
- Good first step to put together Lifecycle Rules (or improve them)!



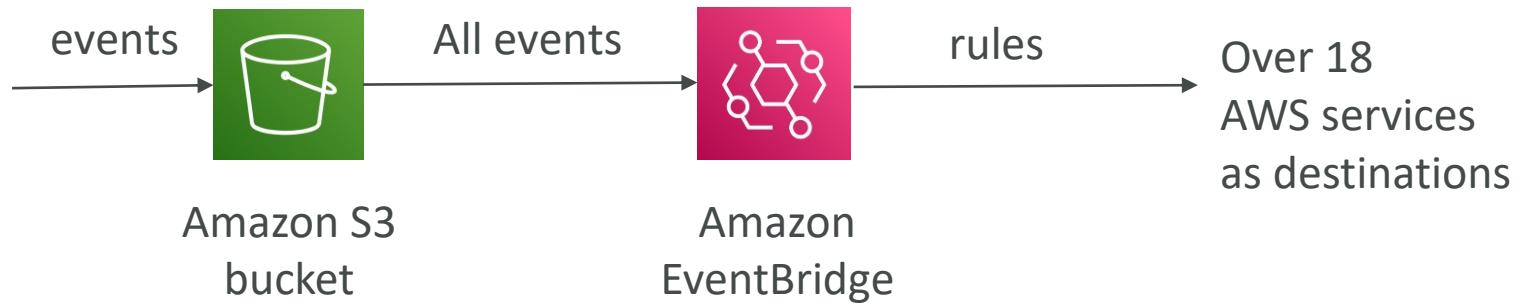
Date	StorageClass	ObjectAge
8/22/2022	STANDARD	000-014
8/25/2022	STANDARD	030-044
9/6/2022	STANDARD	120-149

# S3 Event Notifications

- S3:ObjectCreated, S3:ObjectRemoved, S3:ObjectRestore, S3:Replication...
- Object name filtering possible (\*.jpg)
- Use case: generate thumbnails of images uploaded to S3
- **Can create as many “S3 events” as desired**
- S3 event notifications typically deliver events in seconds but can sometimes take a minute or longer



# S3 Event Notifications with Amazon EventBridge



- **Advanced filtering options with JSON rules (metadata, object size, name...)**
- **Multiple Destinations** – ex Step Functions, Kinesis Streams / Firehose...
- **EventBridge Capabilities** – Archive, Replay Events, Reliable delivery

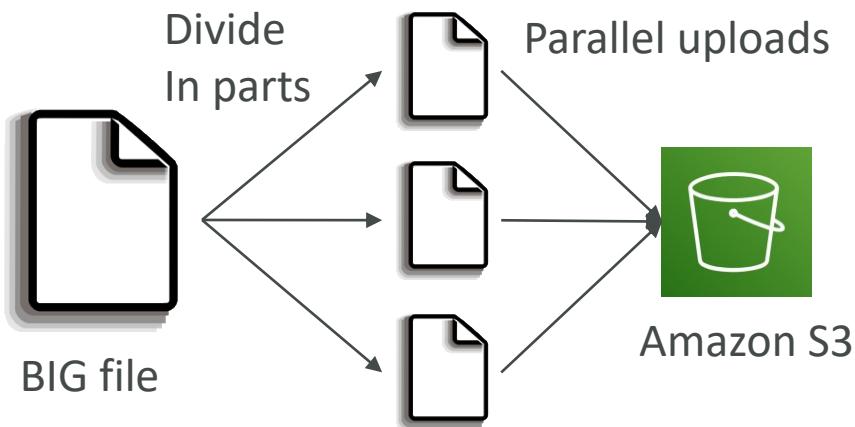
# S3 – Baseline Performance

- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least **3,500 PUT/COPY/POST/DELETE and 5,500 GET/HEAD requests per second per prefix in a bucket.**
- There are no limits to the number of prefixes in a bucket.
- Example (object path => prefix):
  - bucket/folder1/sub1/file => /folder1/sub1/
  - bucket/folder1/sub2/file => /folder1/sub2/
  - bucket/1/file => /1/
  - bucket/2/file => /2/
- If you spread reads across all four prefixes evenly, you can achieve 22,000 requests per second for GET and HEAD

# S3 Performance

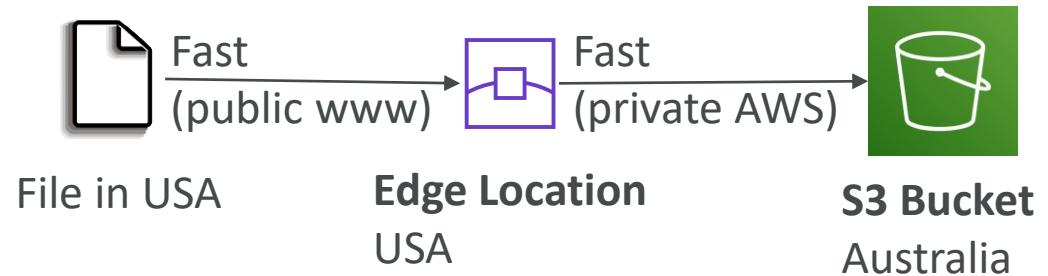
- **Multi-Part upload:**

- recommended for files > 100MB, must use for files > 5GB
- Can help parallelize uploads (speed up transfers)



- **S3 Transfer Acceleration**

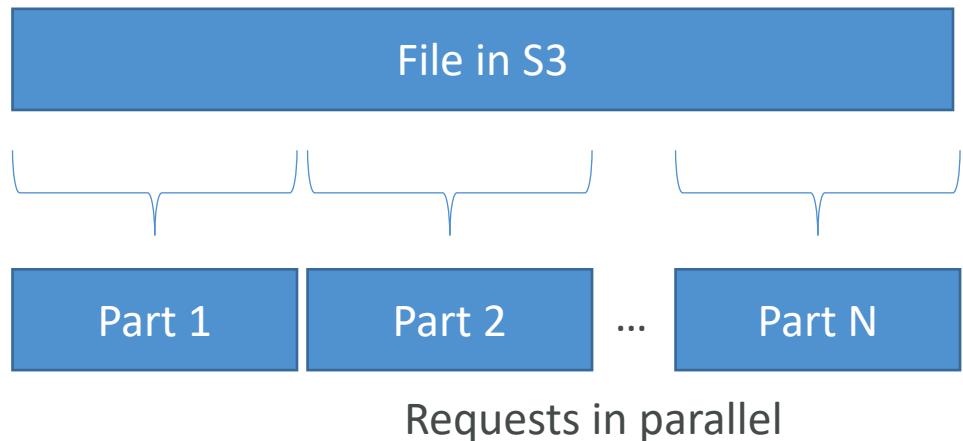
- Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
- Compatible with multi-part upload



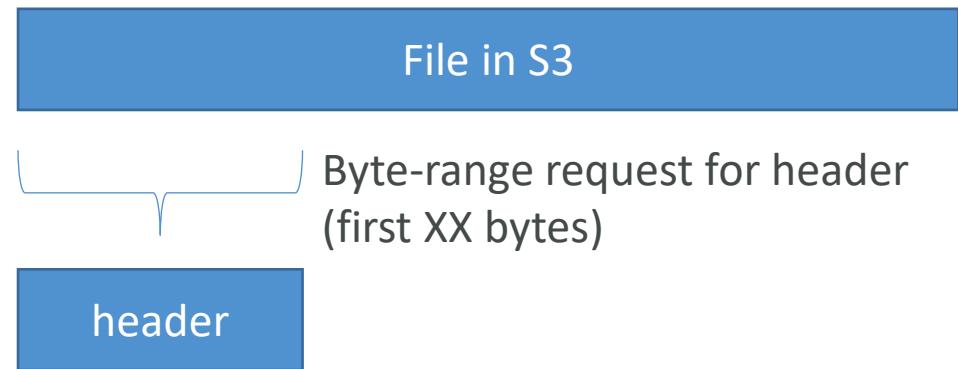
# S3 Performance – S3 Byte-Range Fetches

- Parallelize GETs by requesting specific byte ranges
- Better resilience in case of failures

Can be used to speed up downloads

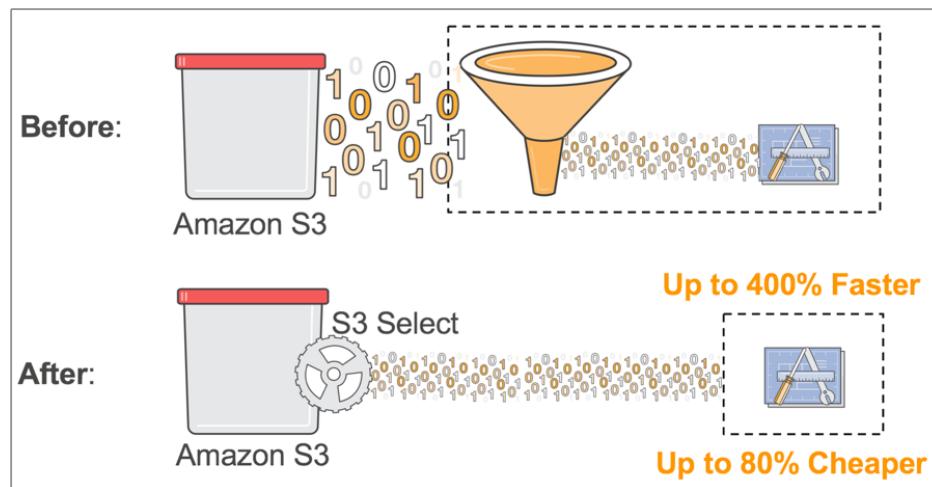


Can be used to retrieve only partial data (for example the head of a file)



# S3 Select & Glacier Select

- Retrieve less data using SQL by performing **server-side filtering**
- Can filter by rows & columns (simple SQL statements)
- Less network transfer, less CPU cost client-side



<https://aws.amazon.com/blogs/aws/s3-glacier-select/>



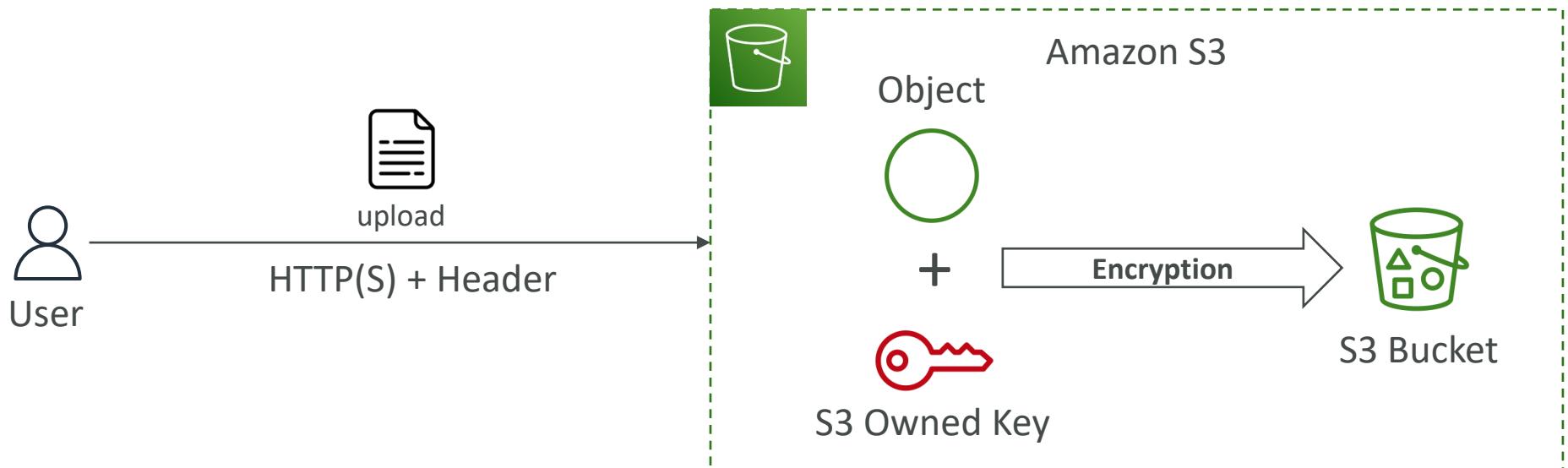


# Amazon S3 – Object Encryption

- You can encrypt objects in S3 buckets using one of 4 methods
- **Server-Side Encryption (SSE)**
  - **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)**
    - Encrypts S3 objects using keys handled, managed, and owned by AWS
  - **Server-Side Encryption with KMS Keys stored in AWS KMS (SSE-KMS)**
    - Leverage AWS Key Management Service (AWS KMS) to manage encryption keys
  - **Server-Side Encryption with Customer-Provided Keys (SSE-C)**
    - When you want to manage your own encryption keys
- **Client-Side Encryption**
- It's important to understand which ones are for which situation for the exam

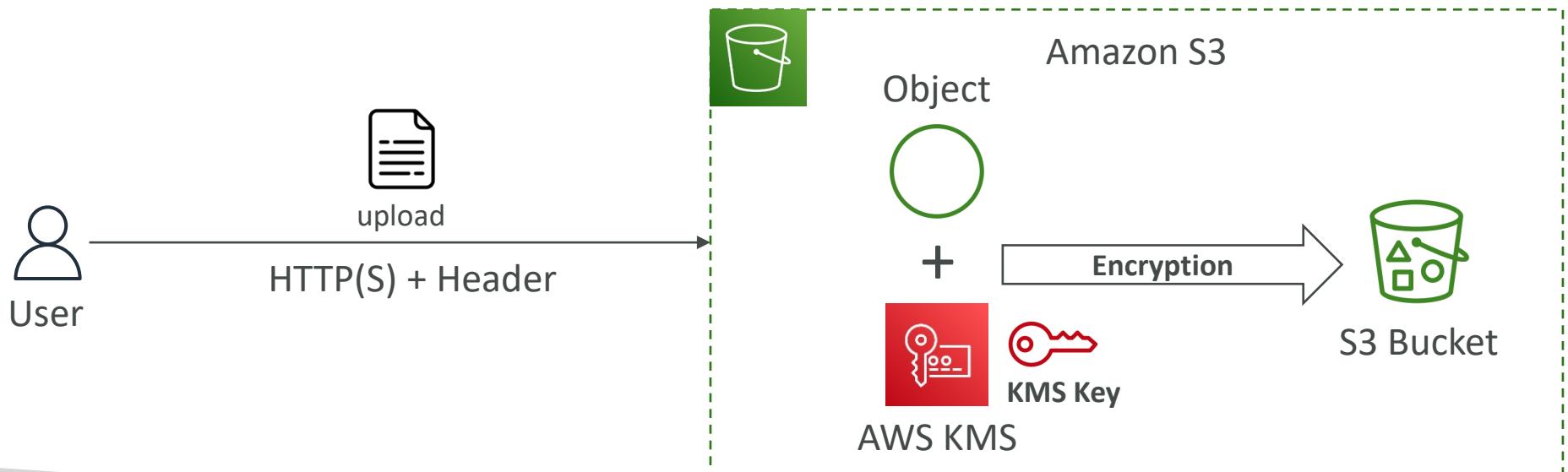
# Amazon S3 Encryption – SSE-S3

- Encryption using keys handled, managed, and owned by AWS
- Object is encrypted server-side
- Encryption type is **AES-256**
- Must set header "**x-amz-server-side-encryption": "AES256"**



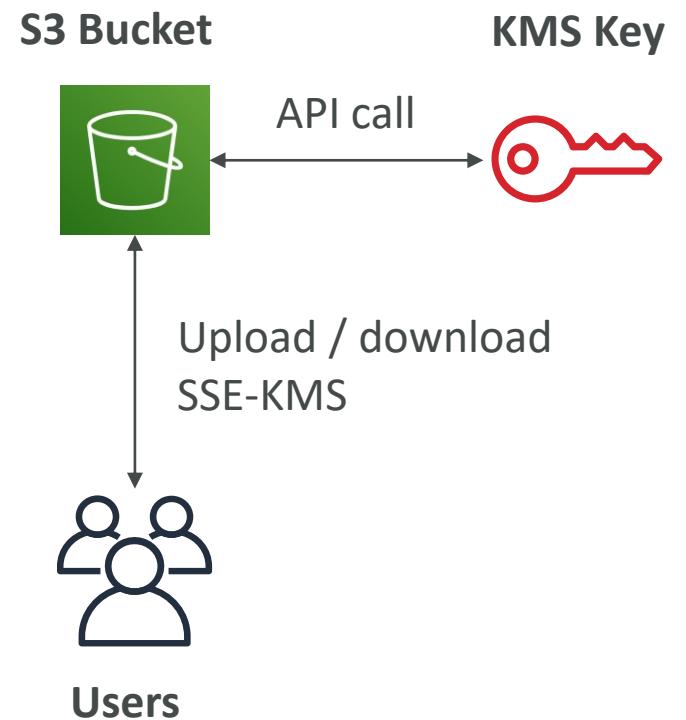
# Amazon S3 Encryption – SSE-KMS

- Encryption using keys handled and managed by AWS KMS (Key Management Service)
- KMS advantages: user control + audit key usage using CloudTrail
- Object is encrypted server side
- Must set header "**x-amz-server-side-encryption": "aws:kms"**



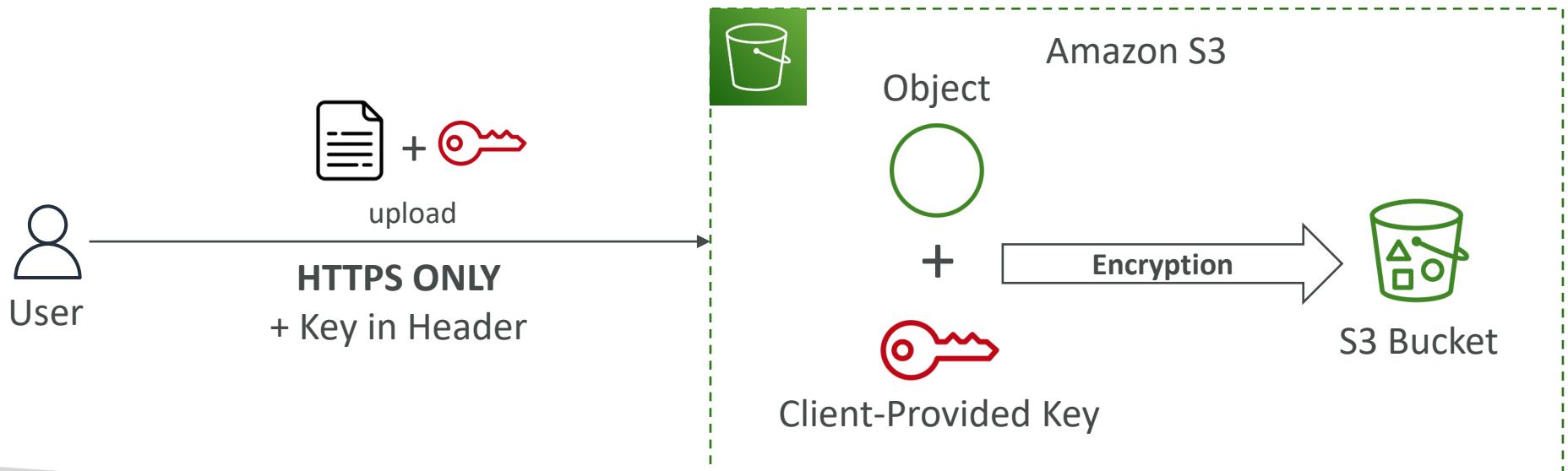
# SSE-KMS Limitation

- If you use SSE-KMS, you may be impacted by the KMS limits
- When you upload, it calls the **GenerateDataKey** KMS API
- When you download, it calls the **Decrypt** KMS API
- Count towards the KMS quota per second (5500, 10000, 30000 req/s based on region)
- You can request a quota increase using the Service Quotas Console



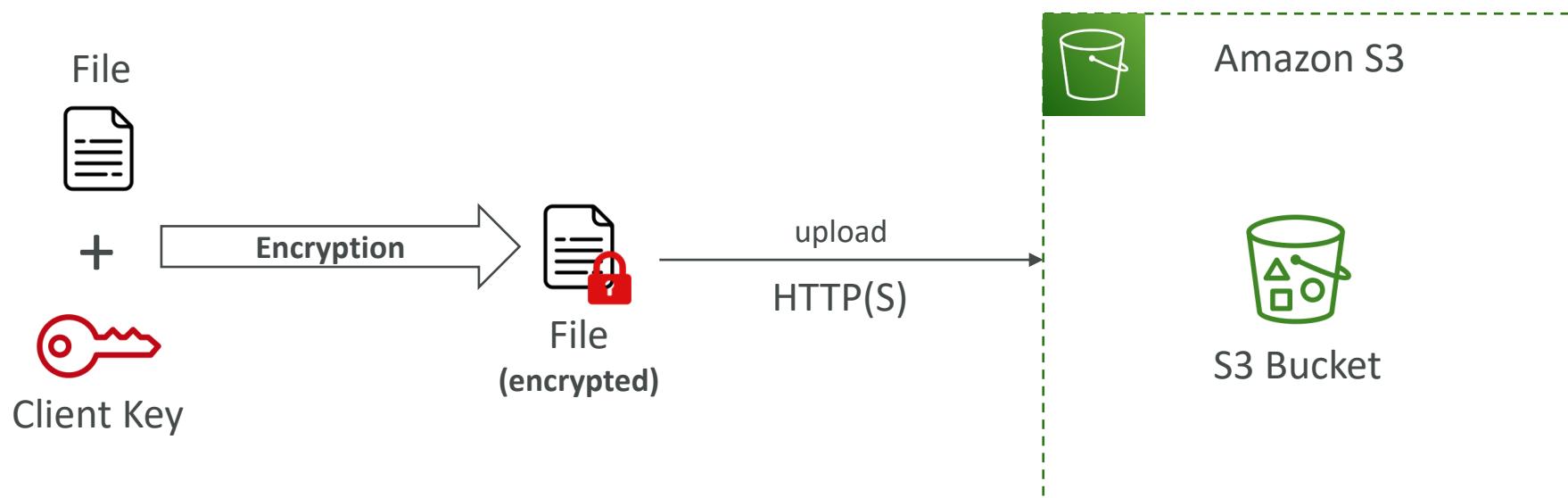
# Amazon S3 Encryption – SSE-C

- Server-Side Encryption using keys fully managed by the customer outside of AWS
- Amazon S3 does **NOT** store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



# Amazon S3 Encryption – Client-Side Encryption

- Use client libraries such as **Amazon S3 Client-Side Encryption Library**
- Clients must encrypt data themselves before sending to Amazon S3
- Clients must decrypt data themselves when retrieving from Amazon S3
- Customer fully manages the keys and encryption cycle



# Amazon S3 – Encryption in transit (SSL/TLS)

- Encryption in flight is also called SSL/TLS
- Amazon S3 exposes two endpoints:
  - **HTTP Endpoint** – non encrypted
  - **HTTPS Endpoint** – encryption in flight
- **HTTPS is recommended**
- **HTTPS is mandatory for SSE-C**
- Most clients would use the HTTPS endpoint by default



# Amazon S3 – Default Encryption vs. Bucket Policies

- One way to “force encryption” is to use a bucket policy and refuse any API call to PUT an S3 object without encryption headers

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DenyIncorrectDecryptionHeader",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": [ "s3:PutObject" ],  
            "Resource": [ "arn:aws:s3:::examplebucket/*" ],  
            "Condition": {  
                "StringNotEquals": {  
                    "s3:x-amz-server-side-encryption": "AES256"  
                }  
            }  
        }  
    ]  
}
```

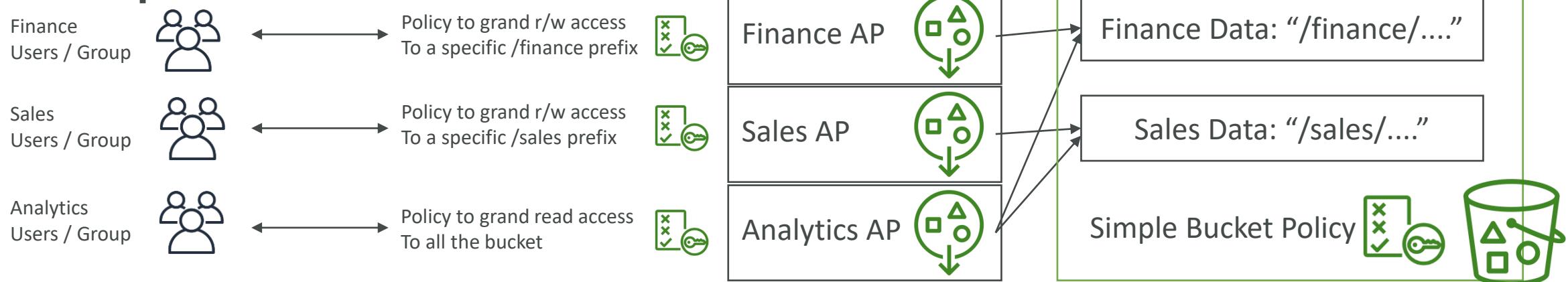
```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "DenyUnencryptedObjectUploads",  
            "Effect": "Deny",  
            "Principal": "*",  
            "Action": [ "s3:PutObject" ],  
            "Resource": [ "arn:aws:s3:::examplebucket/*" ],  
            "Condition": {  
                "Null": {  
                    "s3:x-amz-server-side-encryption": true  
                }  
            }  
        }  
    ]  
}
```

- Another way is to use the “default encryption” option in S3
- **Note: Bucket Policies are evaluated before “default encryption”**



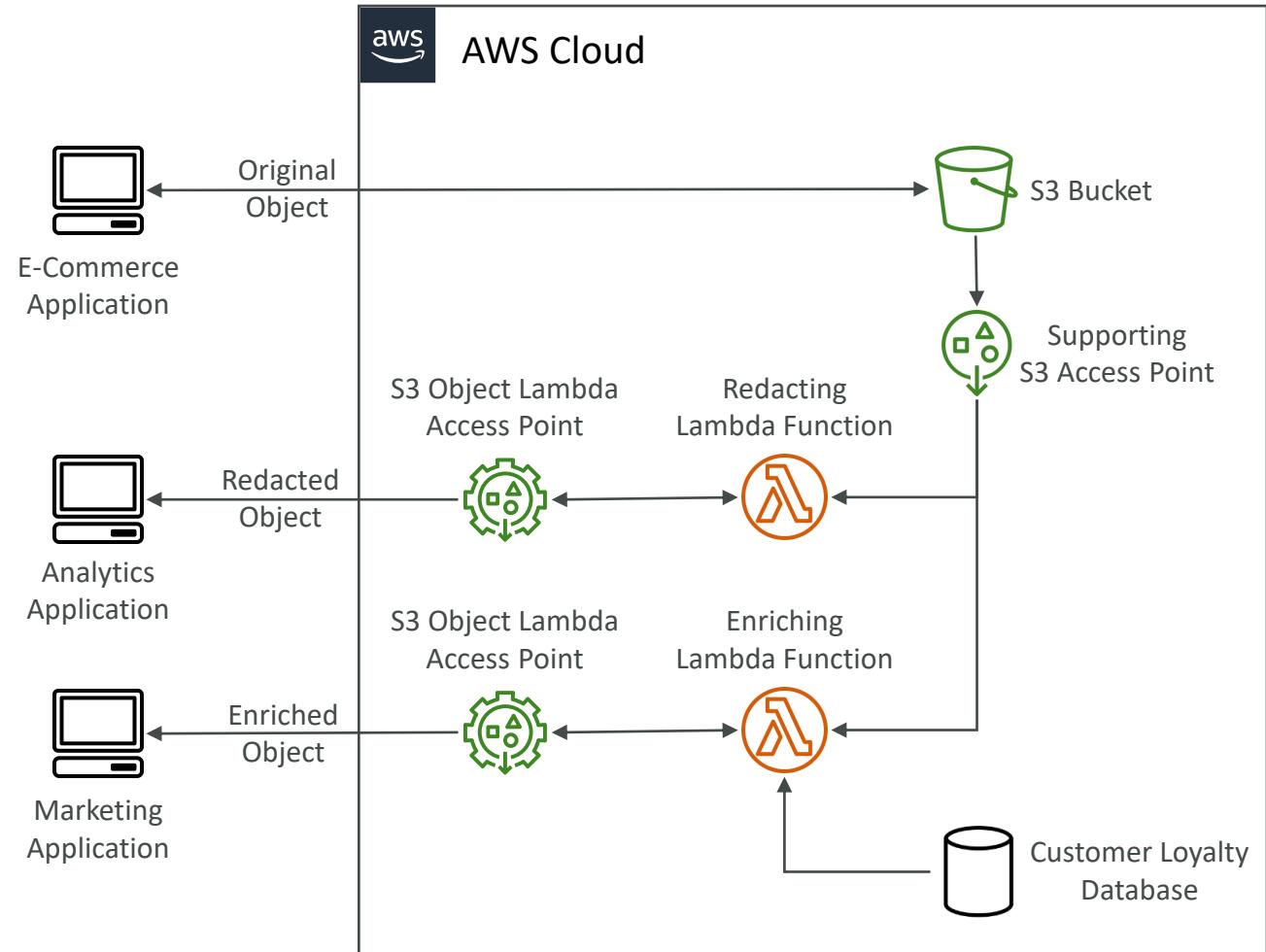
# S3 – Access Points

- Each Access Point gets its own DNS and policy to limit who can access it
  - A specific IAM user / group
  - One policy per Access Point => **Easier to manage than complex bucket policies**

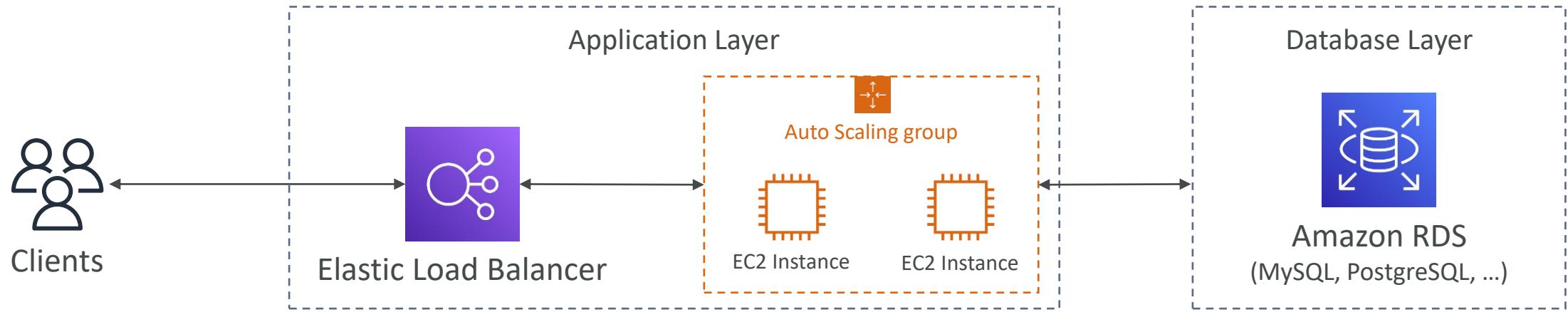


# S3 Object Lambda

- Use AWS Lambda Functions to change the object before it is retrieved by the caller application
- Only one S3 bucket is needed, on top of which we create **S3 Access Point** and **S3 Object Lambda Access Points**.
- Use Cases:
  - Redacting personally identifiable information for analytics or non-production environments.
  - Converting across data formats, such as converting XML to JSON.
  - Resizing and watermarking images on the fly using caller-specific details, such as the user who requested the object.



# Traditional Architecture



- Traditional applications leverage RDBMS databases
- These databases have the SQL query language
- Strong requirements about how the data should be modeled
- Ability to do query joins, aggregations, complex computations
- Vertical scaling (getting a more powerful CPU / RAM / IO)
- Horizontal scaling (increasing reading capability by adding EC2 / RDS Read Replicas)

# NoSQL databases

- NoSQL databases are non-relational databases and are **distributed**
- NoSQL databases include MongoDB, DynamoDB, ...
- NoSQL databases do not support query joins (or just limited support)
- All the data that is needed for a query is present in one row
- NoSQL databases don't perform aggregations such as "SUM", "AVG", ...
- **NoSQL databases scale horizontally**
- There's no "right or wrong" for NoSQL vs SQL, they just require to model the data differently and think about user queries differently

# Amazon DynamoDB



- Fully managed, highly available with replication across multiple AZs
- NoSQL database - not a relational database
- Scales to massive workloads, distributed database
- Millions of requests per seconds, trillions of row, 100s of TB of storage
- Fast and consistent in performance (low latency on retrieval)
- Integrated with IAM for security, authorization and administration
- Enables event driven programming with DynamoDB Streams
- Low cost and auto-scaling capabilities
- Standard & Infrequent Access (IA) Table Class

# DynamoDB - Basics

- DynamoDB is made of **Tables**
- Each table has a **Primary Key** (must be decided at creation time)
- Each table can have an infinite number of items (= rows)
- Each item has **attributes** (can be added over time – can be null)
- Maximum size of an item is **400KB**
- Data types supported are:
  - **Scalar Types** – String, Number, Binary, Boolean, Null
  - **Document Types** – List, Map
  - **Set Types** – String Set, Number Set, Binary Set

# DynamoDB – Primary Keys

- **Option 1: Partition Key (HASH)**

- Partition key must be unique for each item
- Partition key must be “diverse” so that the data is distributed
- Example: “User\_ID” for a users table

Primary Key		Attributes		
Partition Key				
User_ID		First_Name	Last_Name	Age
7791a3d6...		John	William	46
873e0634...		Oliver		24
a80f73a1...		Katie	Lucas	31

# DynamoDB – Primary Keys

- **Option 2: Partition Key + Sort Key (HASH + RANGE)**

- The combination must be unique for each item
- Data is grouped by partition key
- Example: users-games table, “User\_ID” for Partition Key and “Game\_ID” for Sort Key

Primary Key		Attributes	
Partition Key	Sort Key	Score	Result
User_ID	Game_ID	Score	Result
7791a3d6...	4421	92	Win
Same partition key Different sort key 873e0634...	1894	14	Lose
	4521	77	Win

# DynamoDB – Partition Keys (Exercise)

- We're building a movie database
- What is the best Partition Key to maximize data distribution?
  - movie\_id
  - producer\_name
  - leader\_actor\_name
  - movie\_language
- “movie\_id” has the highest cardinality so it's a good candidate
- “movie\_language” doesn't take many values and may be skewed towards English so it's not a great choice for the Partition Key

# DynamoDB in Big Data

- Common use cases include:
  - Mobile apps
  - Gaming
  - Digital ad serving
  - Live voting
  - Audience interaction for live events
  - Sensor networks
  - Log ingestion
  - Access control for web-based content
  - Metadata storage for Amazon S3 objects
  - E-commerce shopping carts
  - Web session management
- Anti Pattern
  - Prewritten application tied to a traditional relational database: use RDS instead
  - Joins or complex transactions
  - Binary Large Object (BLOB) data: store data in S3 & metadata in DynamoDB
  - Large data with low I/O rate: use S3 instead

# DynamoDB – Read/Write Capacity Modes

- Control how you manage your table's capacity (read/write throughput)
- **Provisioned Mode (default)**
  - You specify the number of reads/writes per second
  - You need to plan capacity beforehand
  - Pay for provisioned read & write capacity units
- **On-Demand Mode**
  - Read/writes automatically scale up/down with your workloads
  - No capacity planning needed
  - Pay for what you use, more expensive (\$\$\$)
- You can switch between different modes once every 24 hours

# R/W Capacity Modes – Provisioned

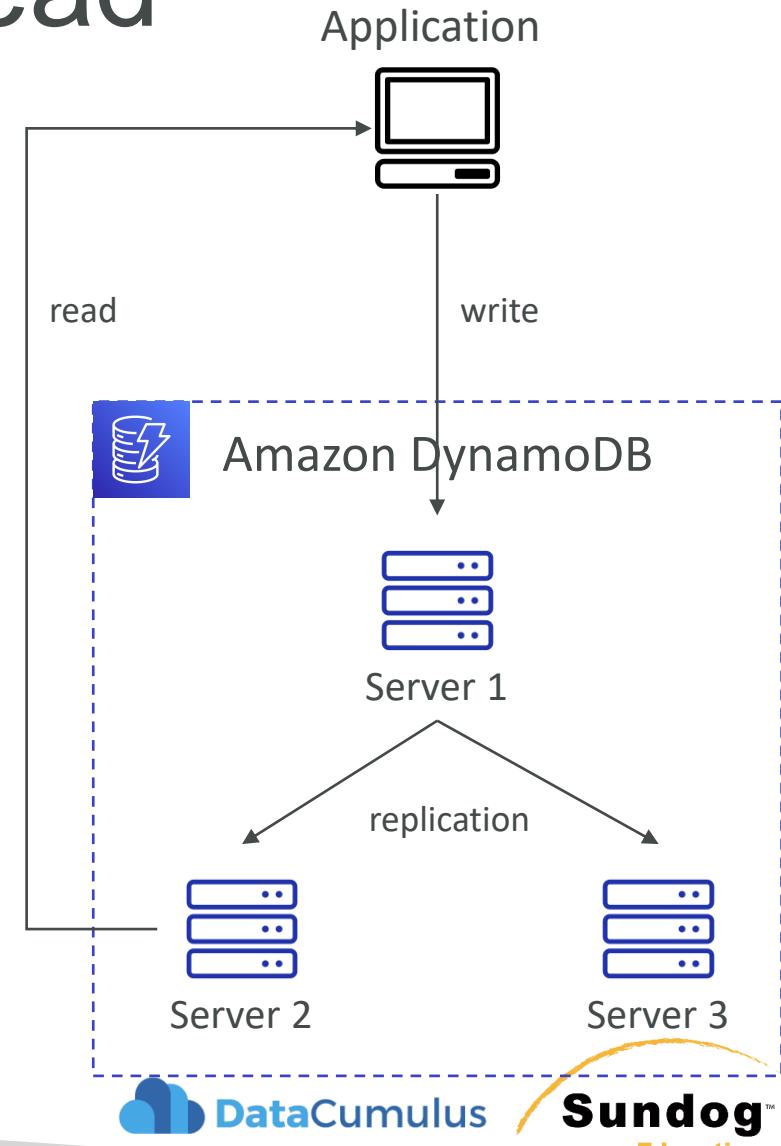
- Table must have provisioned read and write capacity units
- **Read Capacity Units (RCU)** – throughput for reads
- **Write Capacity Units (WCU)** – throughput for writes
- Option to setup **auto-scaling** of throughput to meet demand
- Throughput can be exceeded temporarily using “**Burst Capacity**”
- If Burst Capacity has been consumed, you’ll get a “**ProvisionedThroughputExceededException**”
- It’s then advised to do an **exponential backoff** retry

# DynamoDB – Write Capacity Units (WCU)

- One *Write Capacity Unit (WCU)* represents one write per second for an item up to **1 KB** in size
- If the items are larger than 1 KB, more WCUs are consumed
- **Example 1:** we write 10 items per second, with item size 2 KB
  - We need  $10 * \left(\frac{2 \text{ KB}}{1 \text{ KB}}\right) = 20 \text{ WCUs}$
- **Example 2:** we write 6 items per second, with item size 4.5 KB
  - We need  $6 * \left(\frac{5 \text{ KB}}{1 \text{ KB}}\right) = 30 \text{ WCUs}$  (4.5 gets rounded to the upper KB)
- **Example 3:** we write 120 items per minute, with item size 2 KB
  - We need  $\left(\frac{120}{60}\right) * \left(\frac{2 \text{ KB}}{1 \text{ KB}}\right) = 4 \text{ WCUs}$

# Strongly Consistent Read vs. Eventually Consistent Read

- **Eventually Consistent Read (default)**
  - If we read just after a write, it's possible we'll get some stale data because of replication
- **Strongly Consistent Read**
  - If we read just after a write, we will get the correct data
  - Set “**ConsistentRead**” parameter to **True** in API calls (GetItem, BatchGetItem, Query, Scan)
  - Consumes twice the RCU

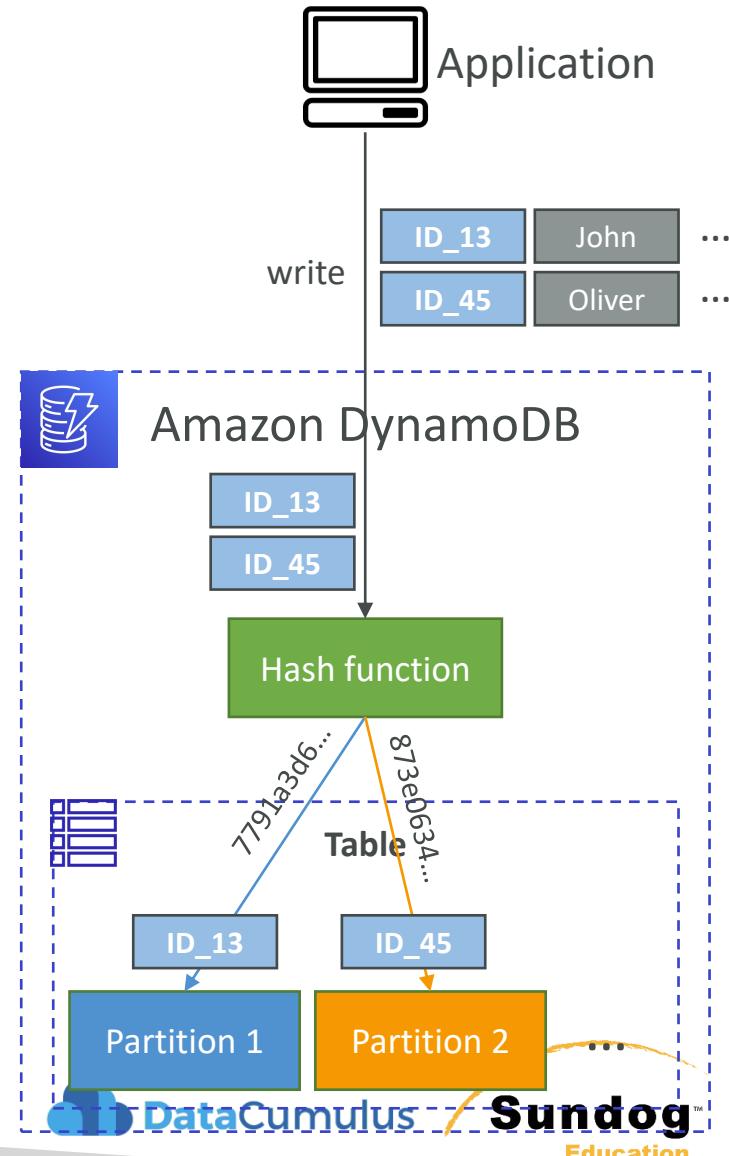


# DynamoDB – Read Capacity Units (RCU)

- One *Read Capacity Unit (RCU)* represents **one Strongly Consistent Read** per second, or **two Eventually Consistent Reads** per second, for an item up to **4 KB** in size
- If the items are larger than 4 KB, more RCUs are consumed
- **Example 1:** 10 Strongly Consistent Reads per second, with item size 4 KB
  - We need  $10 * \left(\frac{4\ KB}{4\ KB}\right) = 10\ RCUs$
- **Example 2:** 16 Eventually Consistent Reads per second, with item size 12 KB
  - We need  $\left(\frac{16}{2}\right) * \left(\frac{12\ KB}{4\ KB}\right) = 24\ RCUs$
- **Example 3:** 10 Strongly Consistent Reads per second, with item size 6 KB
  - We need  $10 * \left(\frac{8\ KB}{4\ KB}\right) = 20\ RCUs$  (we must round up 6 KB to 8 KB)

# DynamoDB – Partitions Internal

- Data is stored in partitions
- Partition Keys go through a hashing algorithm to know to which partition they go to
- To compute the number of partitions:
  - $\# \text{ of partitions}_{\text{by capacity}} = \left( \frac{\text{RCUs}_{\text{Total}}}{3000} \right) + \left( \frac{\text{WCUs}_{\text{Total}}}{1000} \right)$
  - $\# \text{ of partitions}_{\text{by size}} = \frac{\text{Total Size}}{10 \text{ GB}}$
  - $\# \text{ of partitions} = \text{ceil}(\max(\# \text{ of partitions}_{\text{by capacity}}, \# \text{ of partitions}_{\text{by size}}))$
- **WCUs and RCUs are spread evenly across partitions**



# DynamoDB – Throttling

- If we exceed provisioned RCUs or WCUs, we get **“ProvisionedThroughputExceededException”**
- Reasons:
  - **Hot Keys** – one partition key is being read too many times (e.g., popular item)
  - **Hot Partitions**
  - **Very large items**, remember RCU and WCU depends on size of items
- Solutions:
  - **Exponential backoff** when exception is encountered (already in SDK)
  - **Distribute partition keys** as much as possible
  - If RCU issue, we can **use DynamoDB Accelerator (DAX)**

# R/W Capacity Modes – On-Demand

- Read/writes automatically scale up/down with your workloads
- No capacity planning needed (WCU / RCU)
- Unlimited WCU & RCU, no throttle, more expensive
- You're charged for reads/writes that you use in terms of **RRU** and **WRU**
- **Read Request Units (RRU)** – throughput for reads (same as RCU)
- **Write Request Units (WRU)** – throughput for writes (same as WCU)
- 2.5x more expensive than provisioned capacity (use with care)
- Use cases: unknown workloads, unpredictable application traffic, ...

# DynamoDB – Writing Data

- **PutItem**
  - Creates a new item or fully replace an old item (same Primary Key)
  - Consumes WCUs
- **UpdateItem**
  - Edits an existing item's attributes or adds a new item if it doesn't exist
  - Can be used to implement **Atomic Counters** – a numeric attribute that's unconditionally incremented
- **Conditional Writes**
  - Accept a write/update/delete only if conditions are met, otherwise returns an error
  - Helps with concurrent access to items
  - No performance impact

# DynamoDB – Reading Data

- **GetItem**
  - Read based on Primary key
  - Primary Key can be **HASH** or **HASH+RANGE**
  - Eventually Consistent Read (default)
  - Option to use Strongly Consistent Reads (more RCU - might take longer)
  - **ProjectionExpression** can be specified to retrieve only certain attributes

# DynamoDB – Reading Data (Query)

- **Query** returns items based on:
  - **KeyConditionExpression**
    - Partition Key value (**must be = operator**) – required
    - Sort Key value (=, <, <=, >, >=, Between, Begins with) – optional
  - **FilterExpression**
    - Additional filtering after the Query operation (before data returned to you)
    - Use only with non-key attributes (does not allow HASH or RANGE attributes)
- Returns:
  - The number of items specified in **Limit**
  - Or up to 1 MB of data
- Ability to do pagination on the results
- Can query table, a Local Secondary Index, or a Global Secondary Index

# DynamoDB – Reading Data (Scan)

- **Scan** the entire table and then filter out data (inefficient)
- Returns up to 1 MB of data – use pagination to keep on reading
- Consumes a lot of RCU
- Limit impact using **Limit** or reduce the size of the result and pause
- For faster performance, use **Parallel Scan**
  - Multiple workers scan multiple data segments at the same time
  - Increases the throughput and RCU consumed
  - Limit the impact of parallel scans just like you would for Scans
- Can use **ProjectionExpression & FilterExpression** (no changes to RCU)

# DynamoDB – Deleting Data

- **DeleteItem**
  - Delete an individual item
  - Ability to perform a conditional delete
- **DeleteTable**
  - Delete a whole table and all its items
  - Much quicker deletion than calling **DeleteItem** on all items

# DynamoDB – Batch Operations

- Allows you to save in latency by reducing the number of API calls
  - Operations are done in parallel for better efficiency
  - Part of a batch can fail; in which case we need to try again for the failed items
- 
- **BatchWriteItem**
    - Up to 25 **PutItem** and/or **DeleteItem** in one call
    - Up to 16 MB of data written, up to 400 KB of data per item
    - Can't update items (use **UpdateItem**)
  - **BatchGetItem**
    - Return items from one or more tables
    - Up to 100 items, up to 16 MB of data
    - Items are retrieved in parallel to minimize latency

# DynamoDB – Local Secondary Index (LSI)

- Alternative Sort Key for your table (same Partition Key as that of base table)
- The Sort Key consists of one scalar attribute (String, Number, or Binary)
- Up to 5 Local Secondary Indexes per table
- Must be defined at table creation time
- Attribute Projections – can contain some or all the attributes of the base table (**KEYS\_ONLY**, **INCLUDE**, **ALL**)

Primary Key		Attributes		
Partition Key	Sort Key	LSI	Score	Result
User_ID	Game_ID	Game_TS	92	Win
7791a3d6...	4421	"2021-03-15T17:43:08"		Lose
873e0634...	4521	"2021-06-20T19:02:32"		Win
a80f73a1...	1894	"2021-02-11T04:11:31"	77	

# DynamoDB – Global Secondary Index (GSI)

- Alternative Primary Key (**HASH or HASH+RANGE**) from the base table
- Speed up queries on non-key attributes
- The Index Key consists of scalar attributes (String, Number, or Binary)
- **Attribute Projections** – some or all the attributes of the base table (**KEYS\_ONLY, INCLUDE, ALL**)
- Must provision RCUs & WCUs for the index
- **Can be added/modified after table creation**

Partition Key	Sort Key	Attributes
User_ID	Game_ID	Game_TS
7791a3d6-...	4421	"2021-03-15T17:43:08"
873e0634-...	4521	"2021-06-20T19:02:32"
a80f73a1-...	1894	"2021-02-11T04:11:31"

TABLE (query by “User\_ID”)

Partition Key	Sort Key	Attributes
Game_ID	Game_TS	User_ID
4421	"2021-03-15T17:43:08"	7791a3d6-...
4521	"2021-06-20T19:02:32"	873e0634-...
1894	"2021-02-11T04:11:31"	a80f73a1-...

INDEX GSI (query by “Game\_ID”)



# DynamoDB – Indexes and Throttling

- Global Secondary Index (GSI):
  - **If the writes are throttled on the GSI, then the main table will be throttled!**
  - Even if the WCU on the main tables are fine
  - Choose your GSI partition key carefully!
  - Assign your WCU capacity carefully!
- Local Secondary Index (LSI):
  - Uses the WCUs and RCUs of the main table
  - No special throttling considerations

# DynamoDB - PartiQL

- Use a SQL-like syntax to manipulate DynamoDB tables



The screenshot shows a query editor interface titled "Query 1". The code area contains the following SQL-like query:

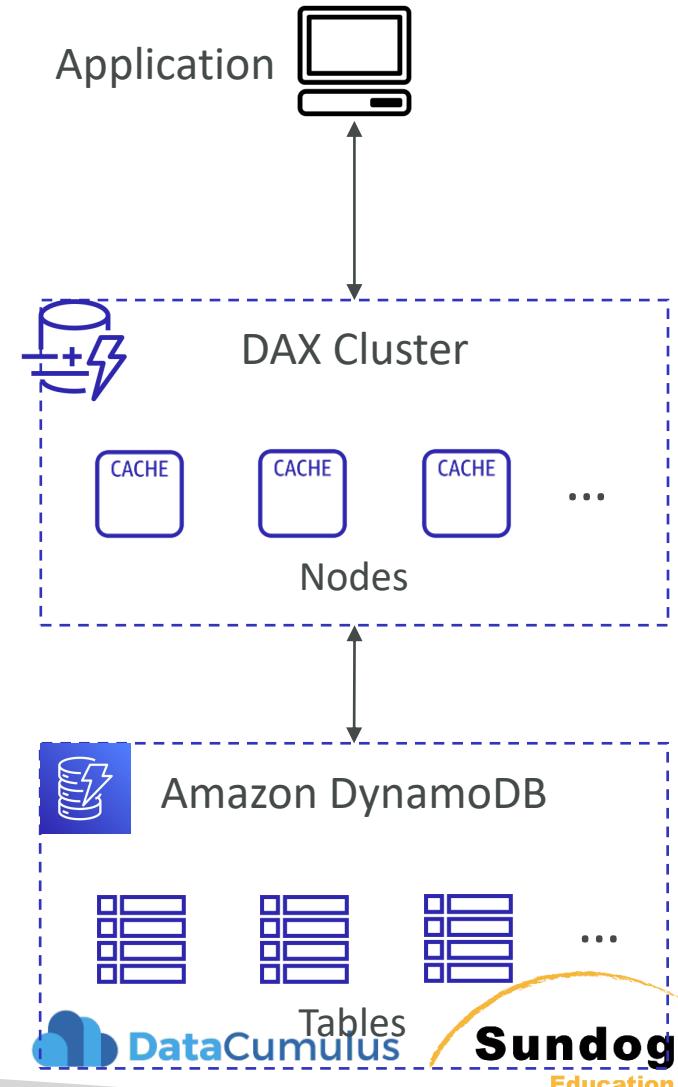
```
1 SELECT * FROM "demo_indexes" WHERE "user_id" = 'partitionKeyValue' AND  
    "game_ts" = 'sortKeyValue'
```

- Supports some (but not all) statements:
  - INSERT
  - UPDATE
  - SELECT
  - DELETE
- It supports Batch operations

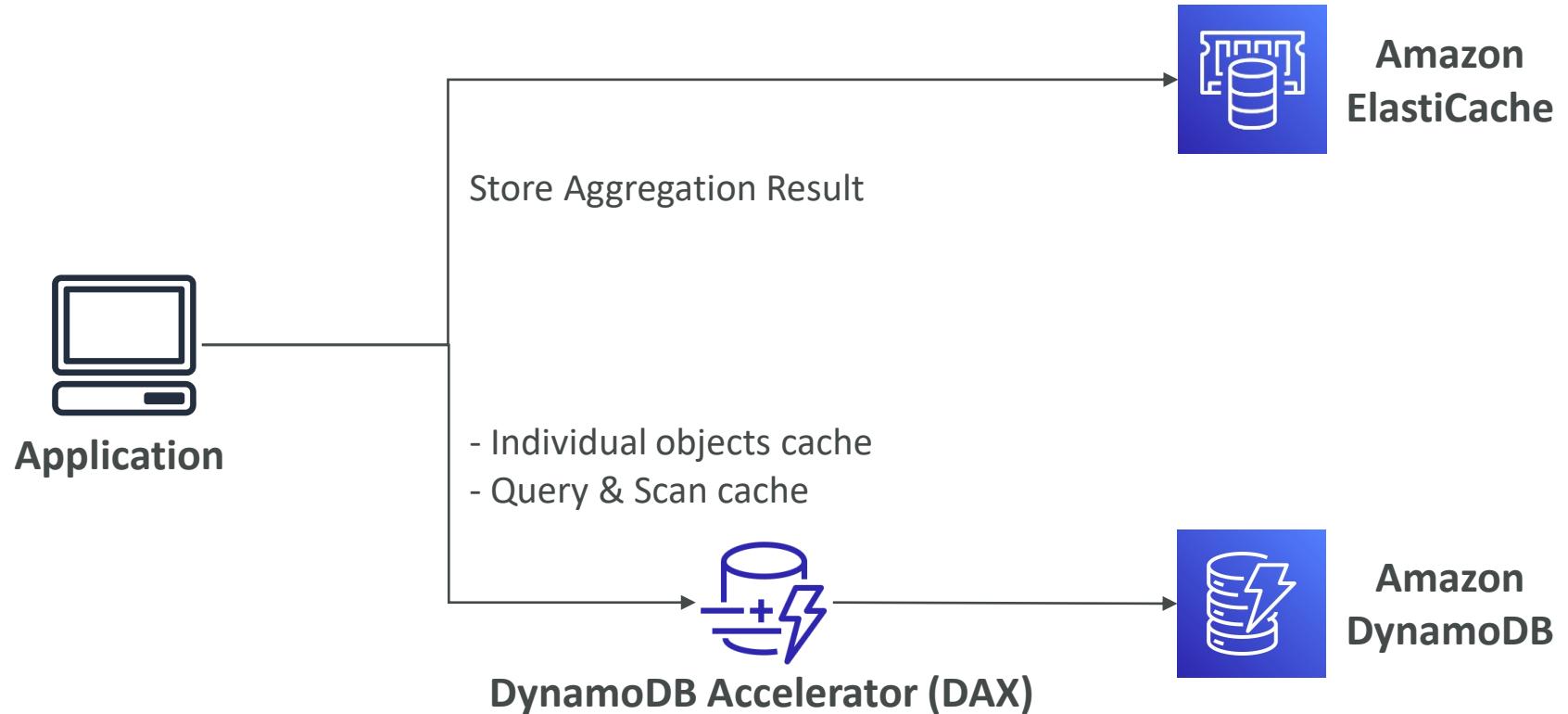
# DynamoDB Accelerator (DAX)



- Fully-managed, highly available, seamless in-memory cache for DynamoDB
- Microseconds latency for cached reads & queries
- Doesn't require application logic modification (compatible with existing DynamoDB APIs)
- Solves the “Hot Key” problem (too many reads)
- 5 minutes TTL for cache (default)
- Up to 10 nodes in the cluster
- Multi-AZ (3 nodes minimum recommended for production)
- Secure (Encryption at rest with KMS, VPC, IAM, CloudTrail, ...)



# DynamoDB Accelerator (DAX) vs. ElastiCache

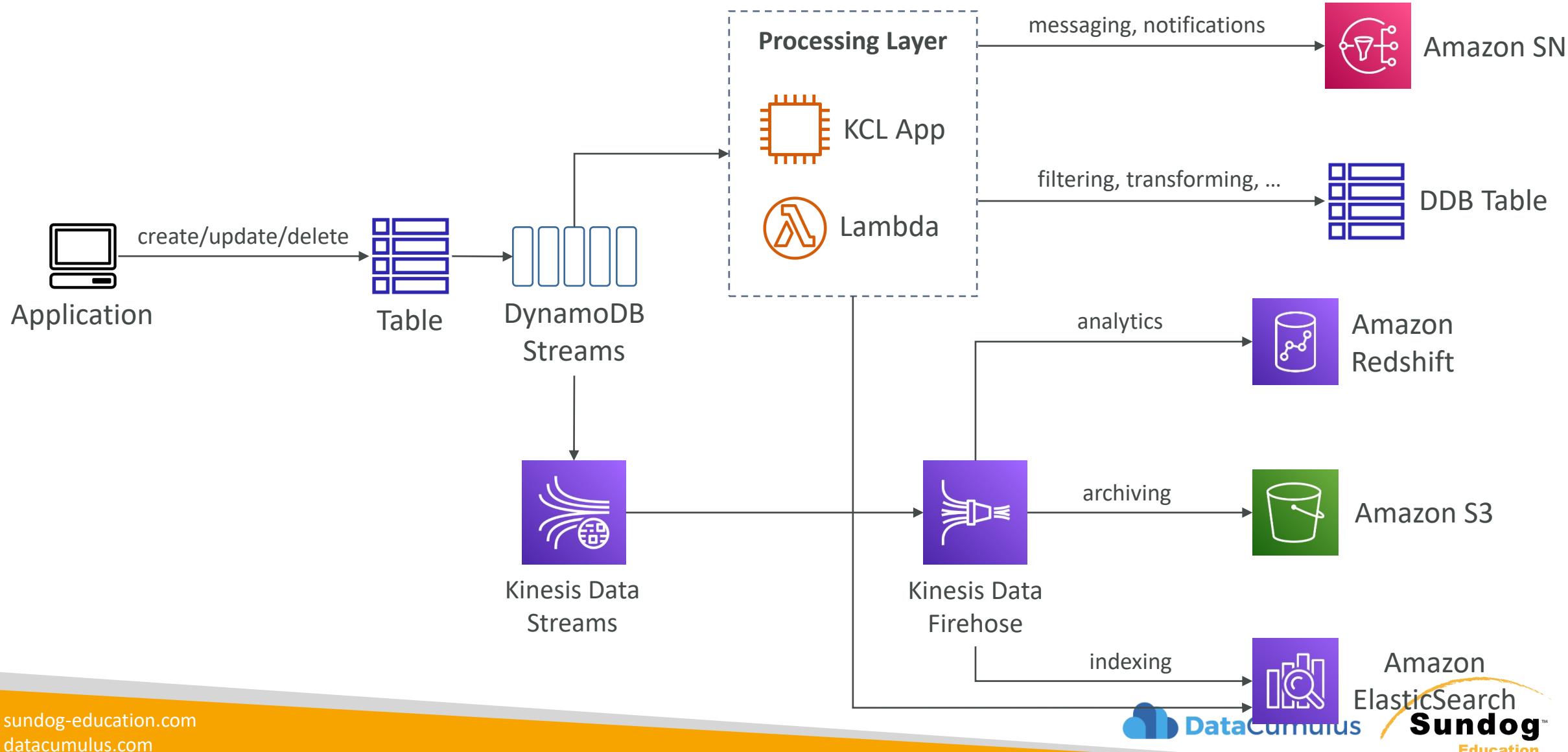


# DynamoDB Streams



- Ordered stream of item-level modifications (create/update/delete) in a table
- Stream records can be:
  - Sent to **Kinesis Data Streams**
  - Read by **AWS Lambda**
  - Read by **Kinesis Client Library applications**
- Data Retention for up to 24 hours
- Use cases:
  - react to changes in real-time (welcome email to users)
  - Analytics
  - Insert into derivative tables
  - Insert into ElasticSearch
  - Implement cross-region replication

# DynamoDB Streams

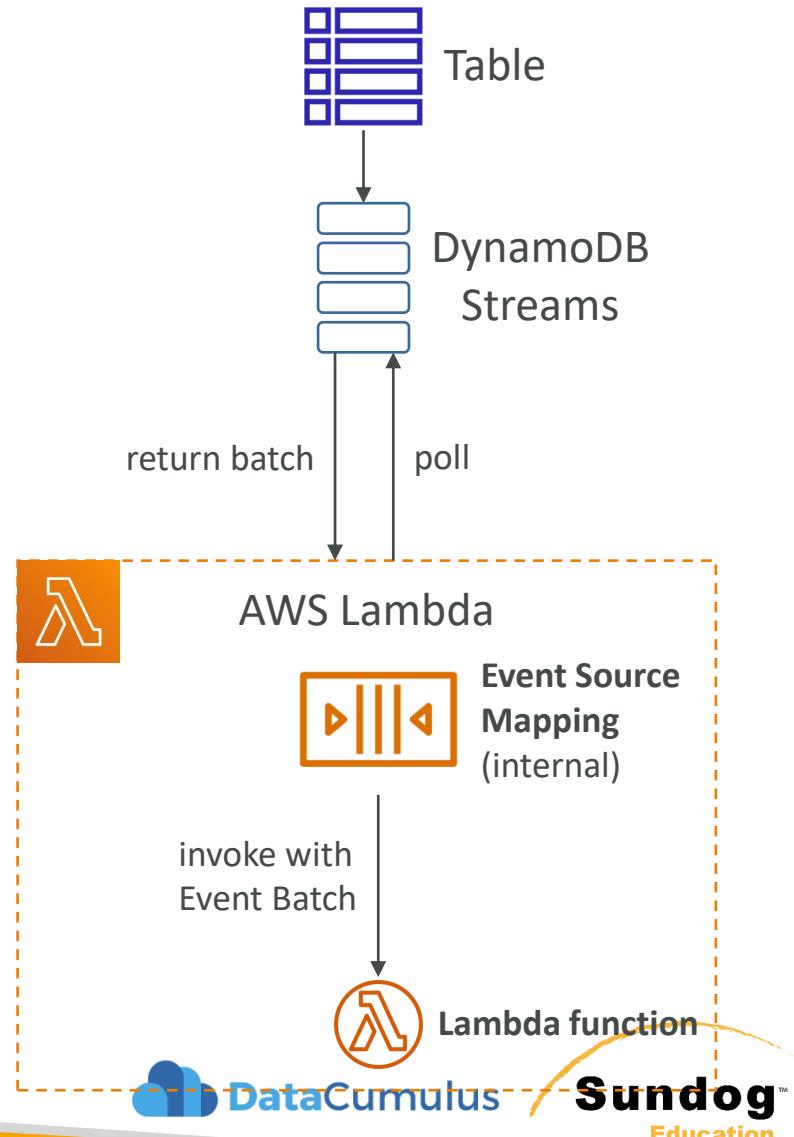


# DynamoDB Streams

- Ability to choose the information that will be written to the stream:
  - **KEYS\_ONLY** – only the key attributes of the modified item
  - **NEW\_IMAGE** – the entire item, as it appears after it was modified
  - **OLD\_IMAGE** – the entire item, as it appeared before it was modified
  - **NEW\_AND\_OLD\_IMAGES** – both the new and the old images of the item
- DynamoDB Streams are made of shards, just like Kinesis Data Streams
- You don't provision shards, this is automated by AWS
- **Records are not retroactively populated in a stream after enabling it**

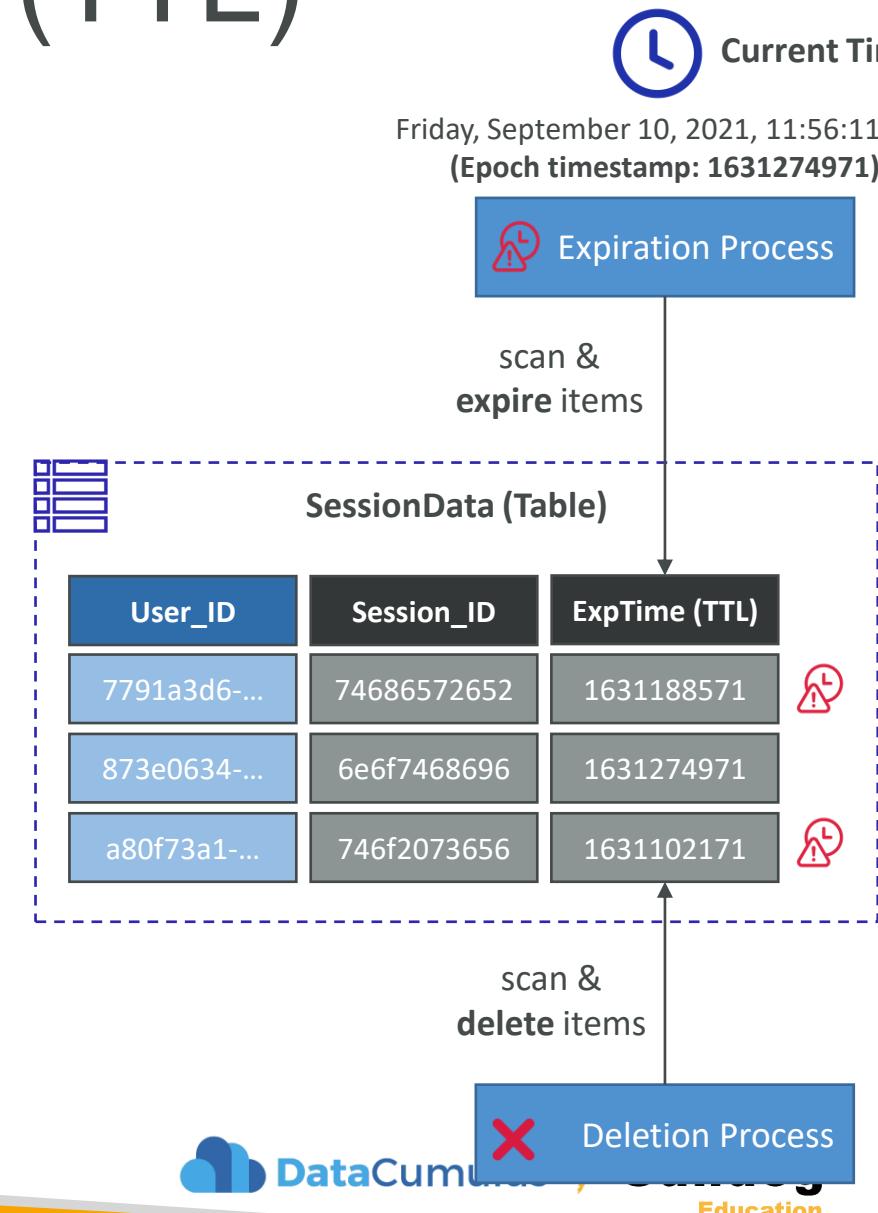
# DynamoDB Streams & AWS Lambda

- You need to define an **Event Source Mapping** to read from a DynamoDB Streams
- You need to ensure the Lambda function has the **appropriate permissions**
- Your Lambda function is invoked **synchronously**

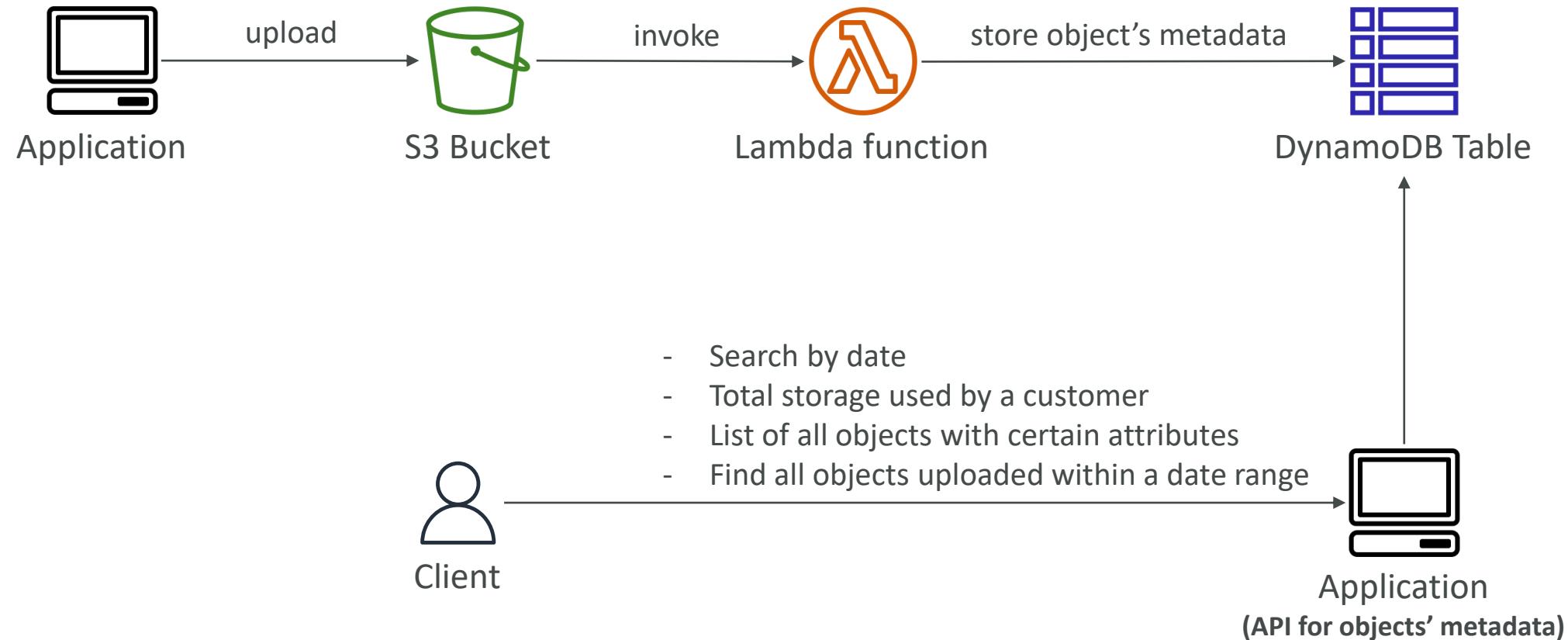


# DynamoDB – Time To Live (TTL)

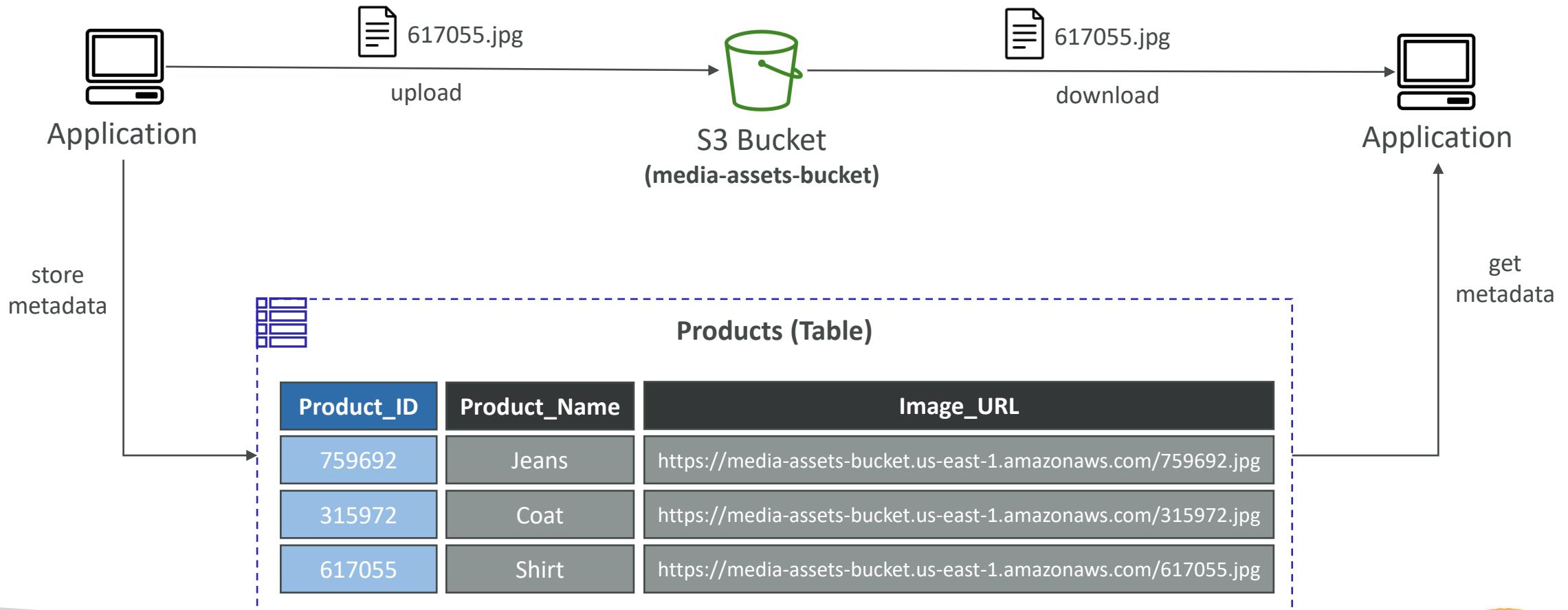
- Automatically delete items after an expiry timestamp
- Doesn't consume any WCUs (i.e., no extra cost)
- The TTL attribute must be a “Number” data type with “Unix Epoch timestamp” value
- Expired items deleted within 48 hours of expiration
- Expired items, that haven't been deleted, appears in reads/queries/scans (if you don't want them, filter them out)
- Expired items are deleted from both LSIs and GSIs
- A delete operation for each expired item enters the DynamoDB Streams (can help recover expired items)
- Use cases: reduce stored data by keeping only current items, adhere to regulatory obligations, ...



# DynamoDB – Indexing S3 Objects Metadata



# DynamoDB – Large Objects Pattern



# DynamoDB – Security & Other Features

- Security:
  - VPC Endpoints available to access DynamoDB without internet
  - Access fully controlled by IAM
  - Encryption at rest using KMS
  - Encryption in transit using SSL / TLS
- Backup and Restore feature available
  - Point in time restore like RDS
  - No performance impact
- Global Tables
  - Multi region, fully replicated, high performance
- Amazon Database Migration Service (DMS) can be used to migrate to DynamoDB (from Mongo, Oracle, MySQL, S3, etc...)
- You can launch a local DynamoDB on your computer for development purposes

# AWS ElastiCache Overview



- The same way RDS is to get managed Relational Databases...
- ElastiCache is to get managed Redis or Memcached
- Caches are in-memory databases with really high performance, low latency
- Helps reduce load off of databases for read intensive workloads
- Helps make your application stateless
- Write Scaling using sharding
- Read Scaling using Read Replicas
- Multi AZ with Failover Capability
- AWS takes care of OS maintenance / patching, optimizations, setup, configuration, monitoring, failure recovery and backups

# Redis Overview

- Redis is an in-memory key-value store
- Super low latency (sub ms)
- Cache survive reboots by default (it's called persistence)
- Great to host
  - User sessions
  - Leaderboard (for gaming)
  - Distributed states
  - Relieve pressure on databases (such as RDS)
  - Pub / Sub capability for messaging
- Multi AZ with Automatic Failover for disaster recovery if you don't want to lose your cache data
- Support for Read Replicas

# Memcached Overview

- Memcached is an in-memory object store
- Cache doesn't survive reboots
- Use cases:
  - Quick retrieval of objects from memory
  - Cache often accessed objects
- Overall, Redis has largely grown in popularity and has better feature sets than Memcached.
- I would personally only use Redis for caching needs.

# AWS Lambda

## Serverless data processing

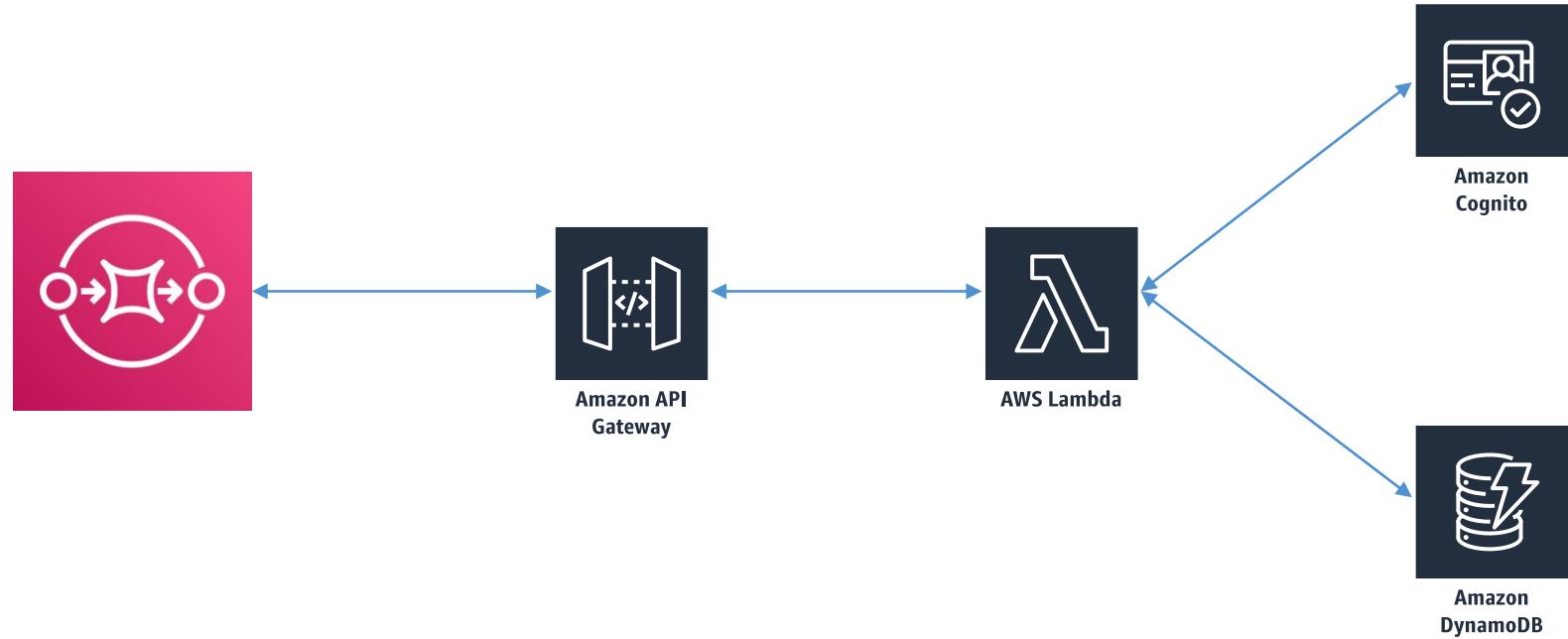
# What is Lambda?

- A way to run code snippets “in the cloud”
  - Serverless
  - Continuous scaling
- Often used to process data as it’s moved around



**AWS Lambda**

# Example: Serverless Website

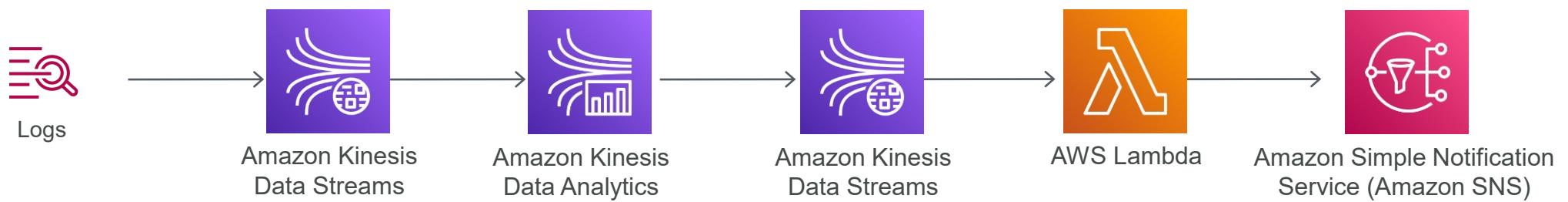


# Example: Order history app



# Example:

## Transaction rate alarm



# Why not just run a server?

- Server management (patches, monitoring, hardware failures, etc.)
- Servers can be cheap, but scaling gets expensive really fast
- You don't pay for processing time you don't use
- Easier to split up development between front-end and back-end

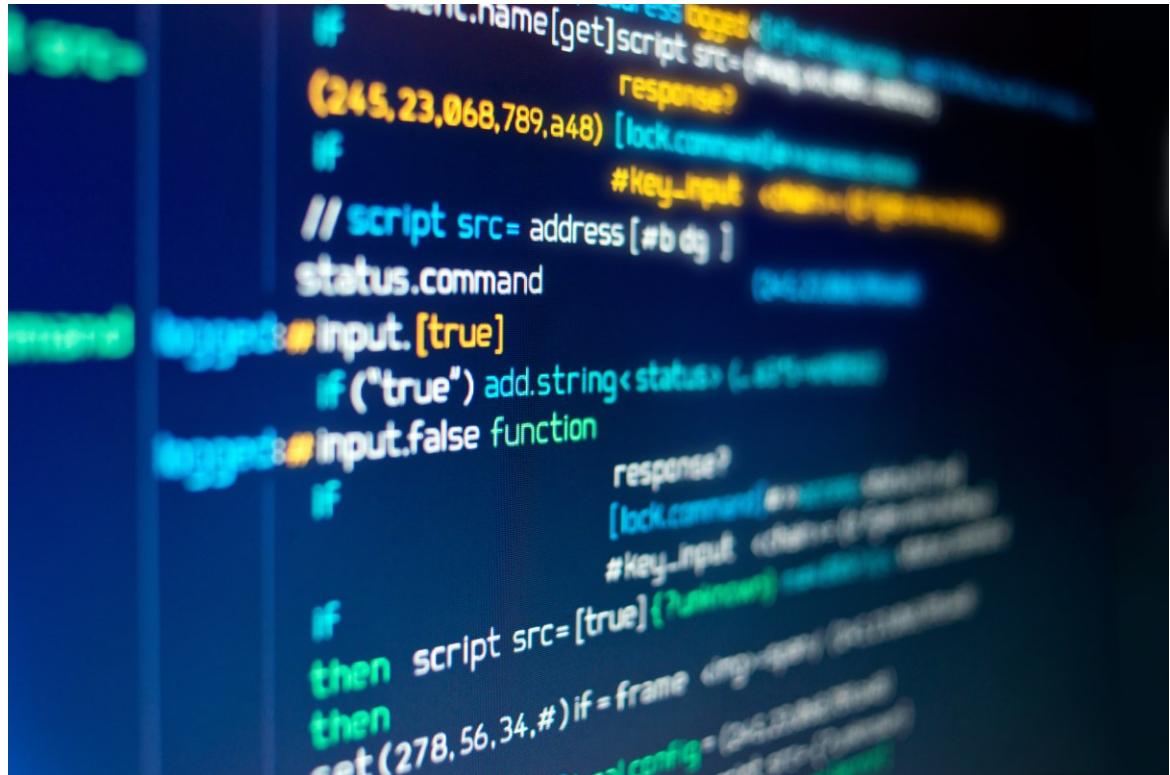
# Main uses of Lambda

- Real-time file processing
- Real-time stream processing
- ETL
- Cron replacement
- Process AWS events



# Supported languages

- Node.js
- Python
- Java
- C#
- Go
- Powershell
- Ruby



# Lambda triggers



Amazon S3



Amazon Simple Email Service



Amazon Kinesis Data Firehose



Amazon Kinesis Data Streams



Amazon DynamoDB



Amazon SNS



Amazon SQS



AWS Config



AWS IoT  
Button



Amazon Lex



Amazon  
CloudWatch



AWS  
CloudFormation



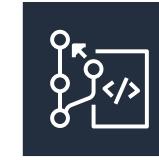
Amazon API  
Gateway



Amazon  
CloudFront



Amazon  
Cognito



AWS  
CodeCommit

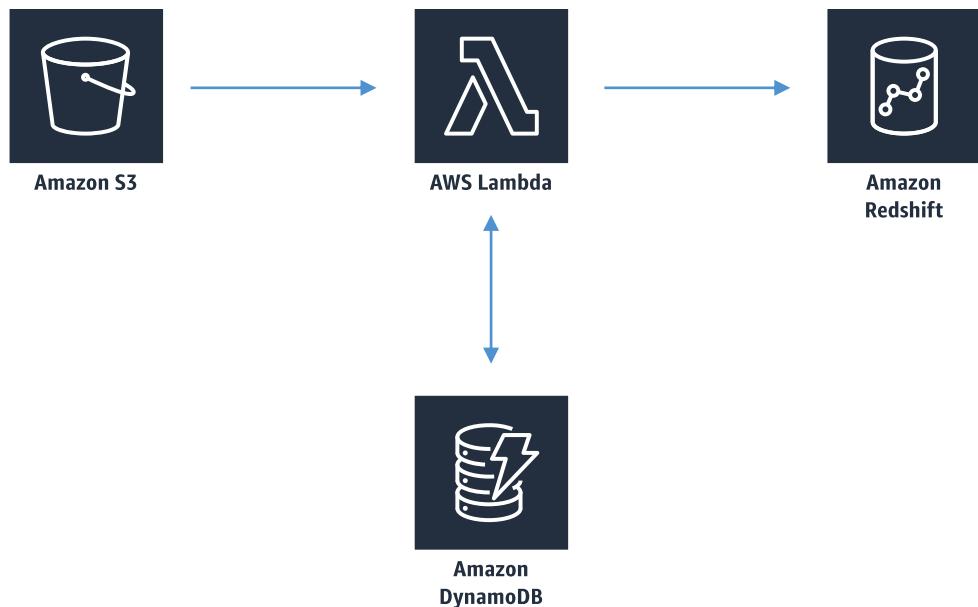
# Lambda and Amazon Elasticsearch Service



# Lambda and Data Pipeline



# Lambda and Redshift



Best practice for loading data into Redshift is the COPY command

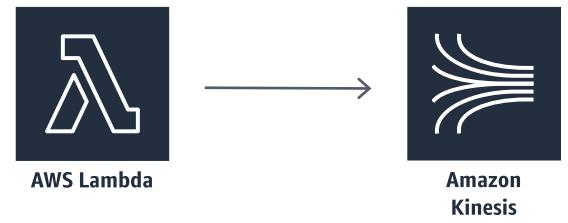
But, you can use Lambda if you need to respond to new data that shows up at any time

Can use DynamoDB to keep track of what's been loaded

Lambda can batch up new data and load them with COPY

# Lambda + Kinesis

- Your Lambda code receives an event with a **batch** of stream records
  - You specify a batch size when setting up the trigger (up to 10,000 records)
  - Too large a batch size can cause timeouts!
  - Batches may also be split beyond Lambda's payload limit (6 MB)
- Lambda will retry the batch until it succeeds or the data expires
  - This can stall the shard if you don't handle errors properly
  - Use more shards to ensure processing isn't totally held up by errors
- Lambda processes shard data synchronously



# Cost Model

- “Pay for what you use”
- Generous free tier (1M requests / month, 400K GB-seconds compute time)
- \$0.20 / million requests
- \$.00001667 per GB/second



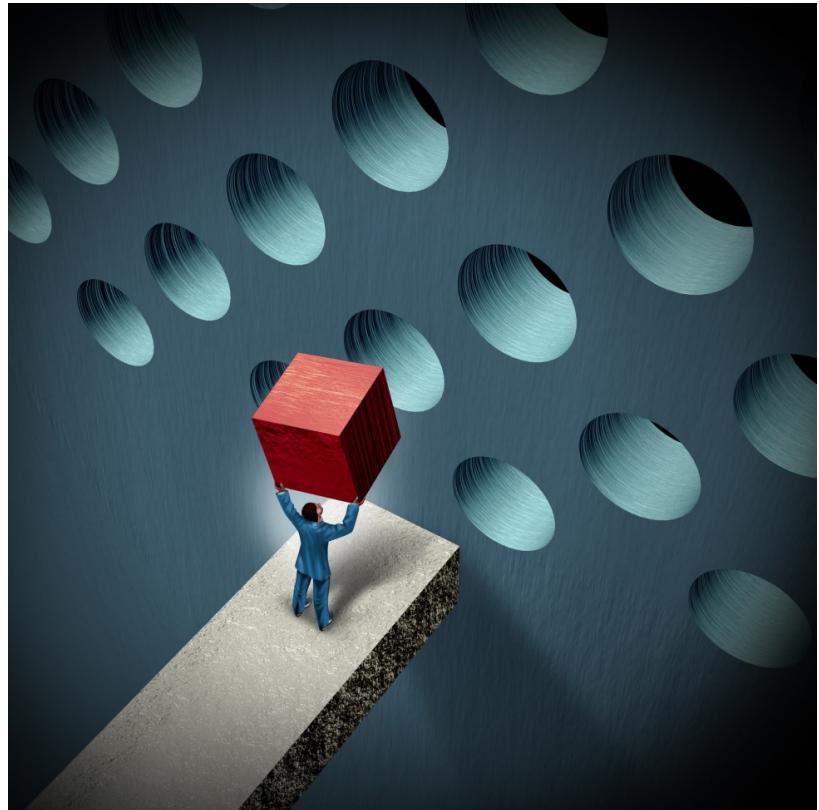
# Other promises

- High availability
  - No scheduled downtime
  - Retries failed code 3 times
- Unlimited scalability\*
  - Safety throttle of 1,000 concurrent executions per region
- High performance
  - New functions callable in seconds
  - Events processed in milliseconds
  - Code is cached automatically
  - But you do specify a timeout! This can cause problems. Max is 900 seconds (15 min)



# Anti-patterns

- Long-running applications
  - Use EC2 instead, or chain functions
- Dynamic websites
  - Although Lambda can be used to develop “serverless” apps that rely on client-side AJAX
- Stateful applications
  - But you can work in DynamoDB or S3 to keep track of state



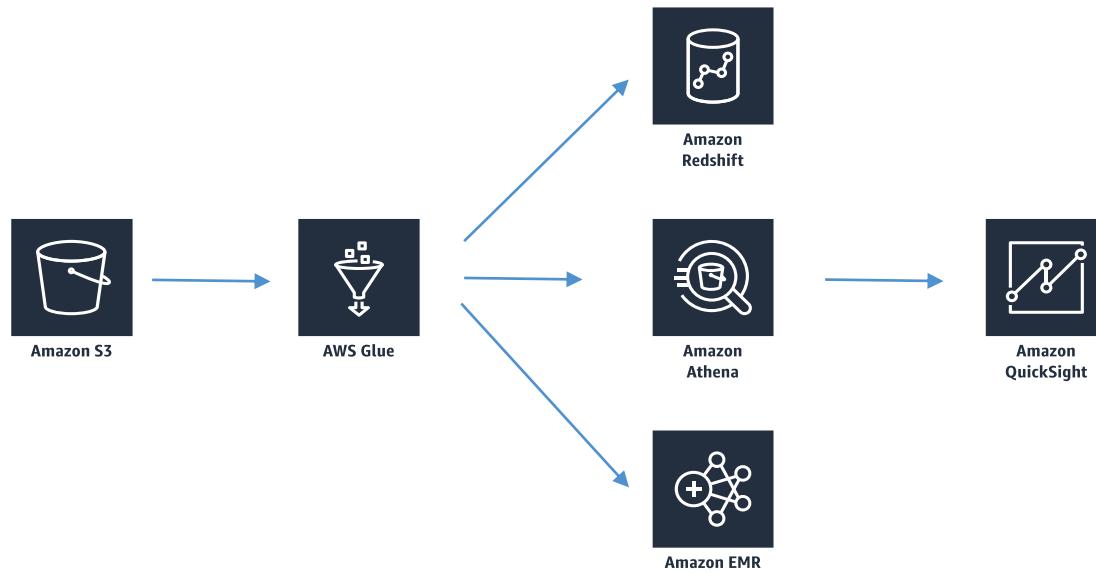
# AWS Glue

Table definitions and ETL

# What is Glue?

- Serverless discovery and definition of table definitions and schema
  - S3 “data lakes”
  - RDS
  - Redshift
  - DynamoDB
  - Most other SQL databases
- Custom ETL jobs
  - Trigger-driven, on a schedule, or on demand
  - Fully managed

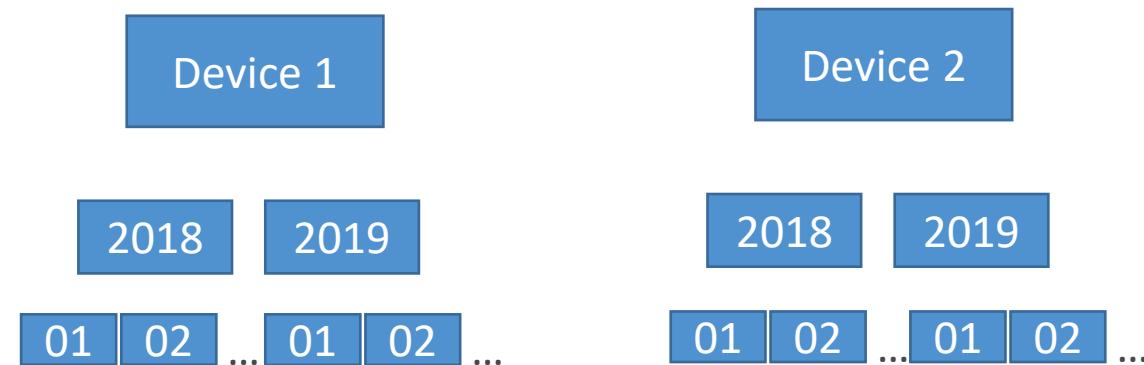
# Glue Crawler / Data Catalog



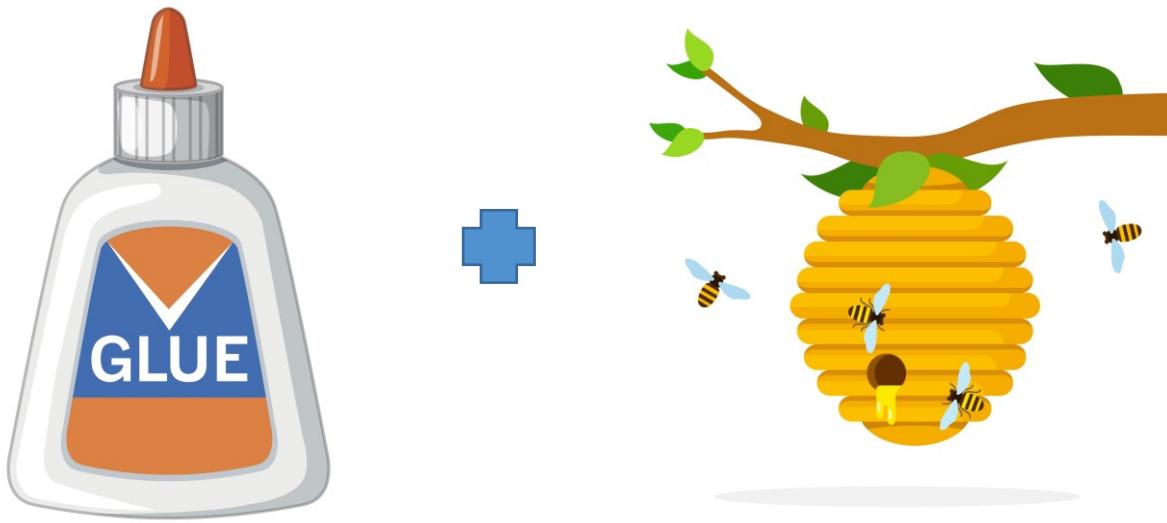
- Glue crawler scans data in S3, creates schema
- Can run periodically
- Populates the Glue Data Catalog
  - Stores only table definition
  - Original data stays in S3
- Once cataloged, you can treat your unstructured data like it's structured
  - Redshift Spectrum
  - Athena
  - EMR
  - Quicksight

# Glue and S3 Partitions

- Glue crawler will extract partitions based on how your S3 data is organized
- Think up front about how you will be querying your data lake in S3
- Example: devices send sensor data every hour
- Do you query primarily by time ranges?
  - If so, organize your buckets as yyyy/mm/dd/device
- Do you query primarily by device?
  - If so, organize your buckets as device/yyyy/mm/dd



# Glue + Hive



- Hive lets you run SQL-like queries from EMR
- The Glue Data Catalog can serve as a Hive “metastore”
- You can also import a Hive metastore into Glue

# Glue ETL

- Automatic code generation
- Scala or Python
- Encryption
  - Server-side (at rest)
  - SSL (in transit)
- Can be event-driven
- Can provision additional “DPU’s” (data processing units) to increase performance of underlying Spark jobs
  - Enabling job metrics can help you understand the maximum capacity in DPU’s you need
- Errors reported to CloudWatch
  - Could tie into SNS for notification

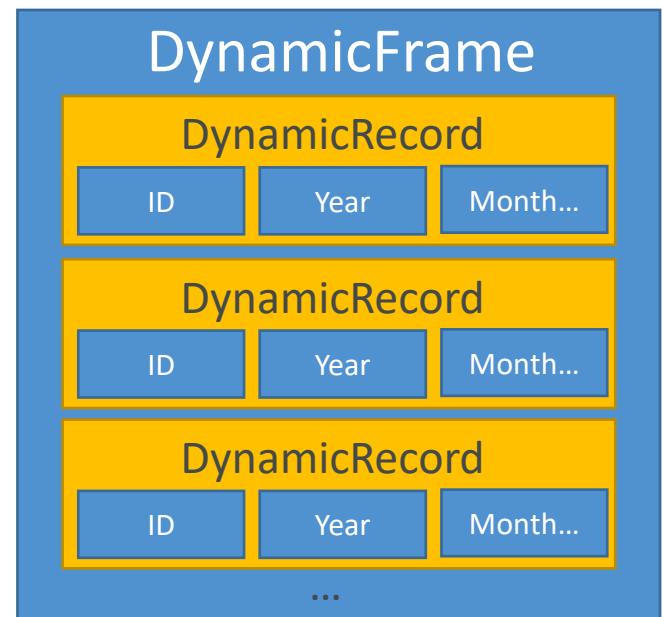
# Glue ETL

- Transform data, Clean Data, Enrich Data (before doing analysis)
  - Generate ETL code in Python or Scala, you can modify the code
  - Can provide your own Spark or PySpark scripts
  - Target can be S3, JDBC (RDS, Redshift), or in Glue Data Catalog
- Fully managed, cost effective, pay only for the resources consumed
- Jobs are run on a serverless Spark platform
- Glue Scheduler to schedule the jobs
- Glue Triggers to automate job runs based on “events”

# Glue ETL: The DynamicFrame

- A DynamicFrame is a collection of DynamicRecords
  - DynamicRecords are self-describing, have a schema
  - Very much like a Spark DataFrame, but with more ETL stuff
  - Scala and Python APIs

```
val pushdownEvents = glueContext.getCatalogSource(  
    database = "githubarchive_month", tableName = "data")  
  
val projectedEvents = pushdownEvents.applyMapping(Seq(  
    ("id", "string", "id", "long"), ("type", "string", "type",  
    "string"), ("actor.login", "string", "actor", "string"),  
    ("repo.name", "string", "repo", "string"),  
    ("payload.action", "string", "action", "string"),  
    ("org.login", "string", "org", "string"), ("year",  
    "string", "year", "int"), ("month", "string", "month",  
    "int"), ("day", "string", "day", "int") ))
```



# Glue ETL - Transformations

- Bundled Transformations:
  - DropFields, DropNullFields – remove (null) fields
  - Filter – specify a function to filter records
  - Join – to enrich data
  - Map - add fields, delete fields, perform external lookups
- Machine Learning Transformations:
  - **FindMatches ML:** identify duplicate or matching records in your dataset, even when the records do not have a common unique identifier and no fields match exactly.
- Format conversions: CSV, JSON, Avro, Parquet, ORC, XML
- Apache Spark transformations (example: K-Means)
  - Can convert between Spark DataFrame and Glue DynamicFrame

# Glue ETL: ResolveChoice

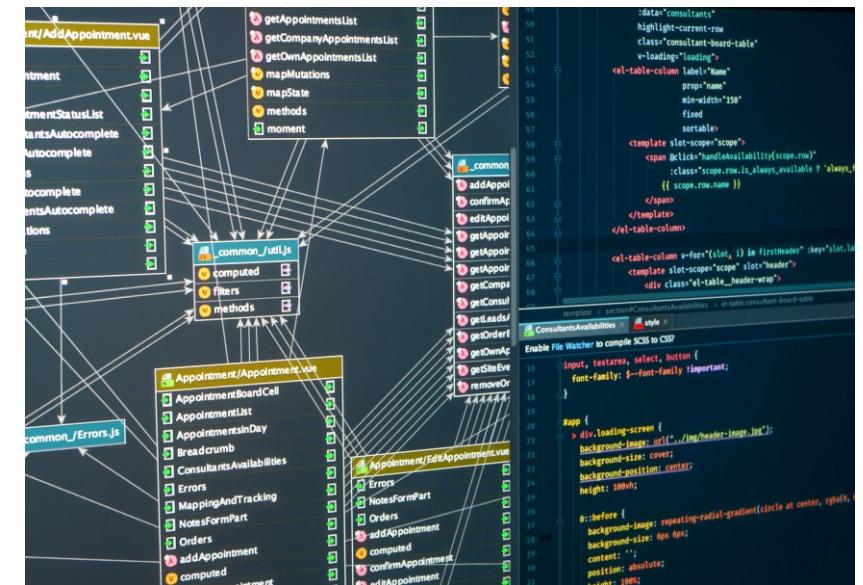
- Deals with ambiguities in a DynamicFrame and returns a new one
- For example, two fields with the same name.
- make\_cols: creates a new column for each type
  - price\_double, price\_string
- cast: casts all values to specified type
- make\_struct: Creates a structure that contains each data type
- project: Projects every type to a given type, for example project:string

```
"myList": [ { "price": 100.00 }, { "price": "$100.00" } ]
```

```
df1 = df.resolveChoice(choice = "make_cols")
df2 = df.resolveChoice(specs = [("myList[]].price",
"make_struct"), ("columnA", "cast:double")])
```

# Glue ETL: Modifying the Data Catalog

- ETL scripts can update your schema and partitions if necessary
- Adding new partitions
  - Re-run the crawler, or
  - Have the script use enableUpdateCatalog and partitionKeys options
- Updating table schema
  - Re-run the crawler, or
  - Use enableUpdateCatalog / updateBehavior from script
- Creating new tables
  - enableUpdateCatalog / updateBehavior with setCatalogInfo
- Restrictions
  - S3 only
  - Json, csv, avro, parquet only
  - Parquet requires special code
  - Nested schemas are not supported



# AWS Glue Development Endpoints

- Develop ETL scripts using a notebook
  - Then create an ETL job that runs your script (using Spark and Glue)
- Endpoint is in a VPC controlled by security groups, connect via:
  - Apache Zeppelin on your local machine
  - Zeppelin notebook server on EC2 (via Glue console)
  - SageMaker notebook
  - Terminal window
  - PyCharm professional edition
  - Use Elastic IP's to access a private endpoint address



# Running Glue jobs

- Time-based schedules (cron style)
- Job bookmarks
  - Persists state from the job run
  - Prevents reprocessing of old data
  - Allows you to process new data only when re-running on a schedule
  - Works with S3 sources in a variety of formats
  - Works with relational databases via JDBC (if PK's are in sequential order)
    - Only handles new rows, not updated rows
- CloudWatch Events
  - Fire off a Lambda function or SNS notification when ETL succeeds or fails
  - Invoke EC2 run, send event to Kinesis, activate a Step Function



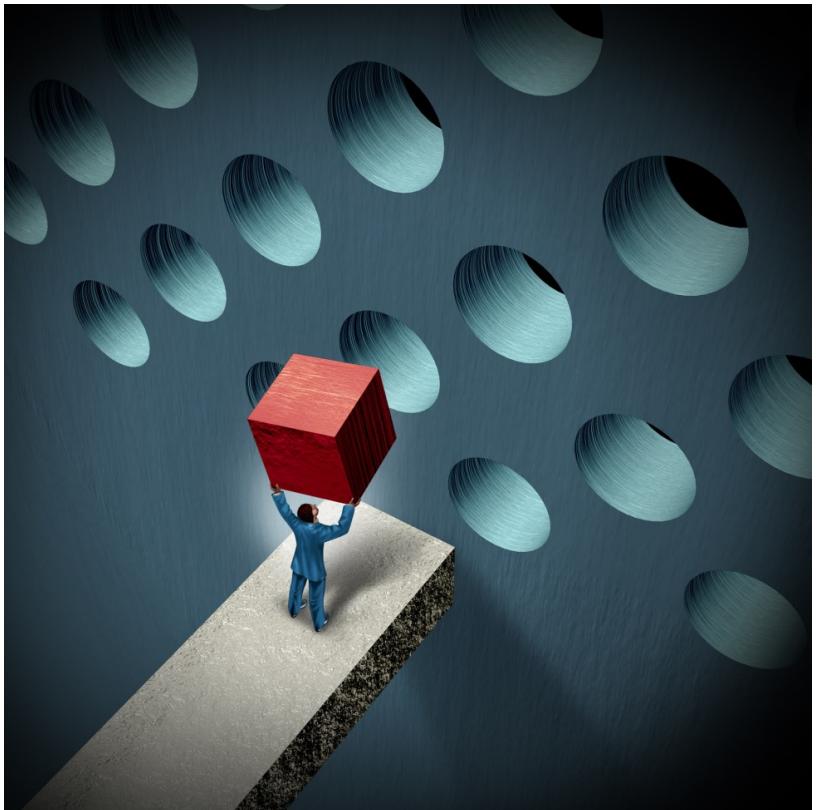
# Glue cost model

- Billed by the second for crawler and ETL jobs
- First million objects stored and accesses are free for the Glue Data Catalog
- Development endpoints for developing ETL code charged by the minute



# Glue Anti-patterns

- Multiple ETL engines
  - Glue ETL is based on Spark
  - If you want to use other engines (Hive, Pig, etc) Data Pipeline EMR would be a better fit.

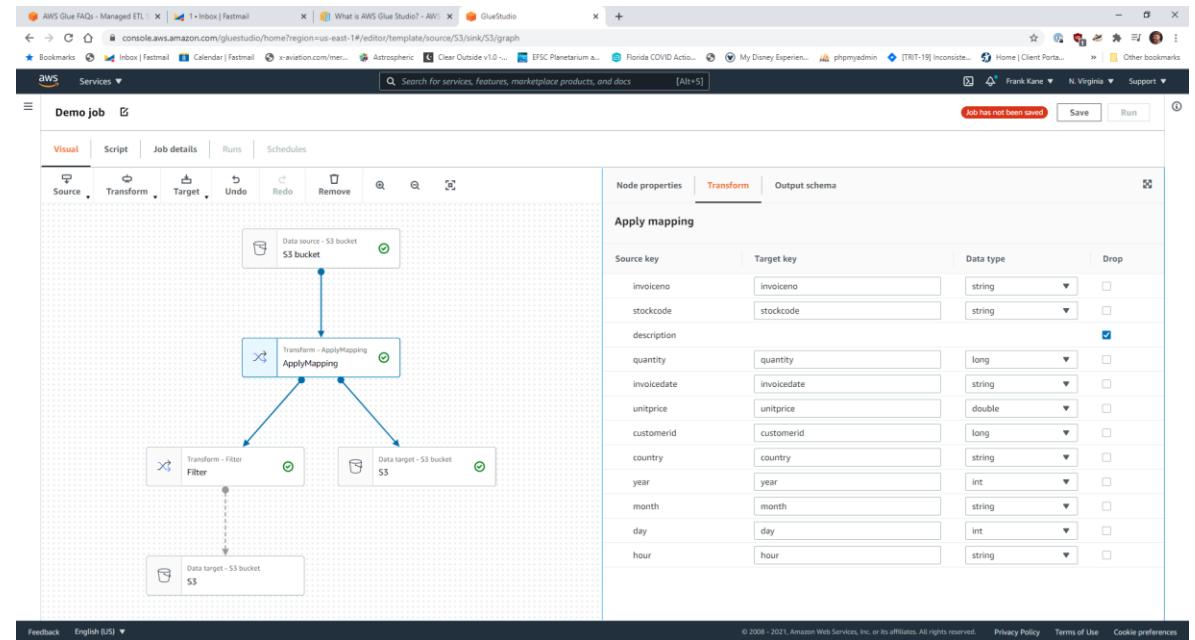


# No longer an anti-pattern: streaming

- As of April 2020, Glue ETL supports serverless streaming ETL
  - Consumes from Kinesis or Kafka
  - Clean & transform in-flight
  - Store results into S3 or other data stores
- Runs on Apache Spark Structured Streaming

# AWS Glue Studio

- Visual interface for ETL workflows
- Visual job editor
  - Create DAG's for complex workflows
  - Sources include S3, Kinesis, Kafka, JDBC
  - Transform / sample / join data
  - Target to S3 or Glue Data Catalog
  - Support partitioning
- Visual job dashboard
  - Overviews, status, run times



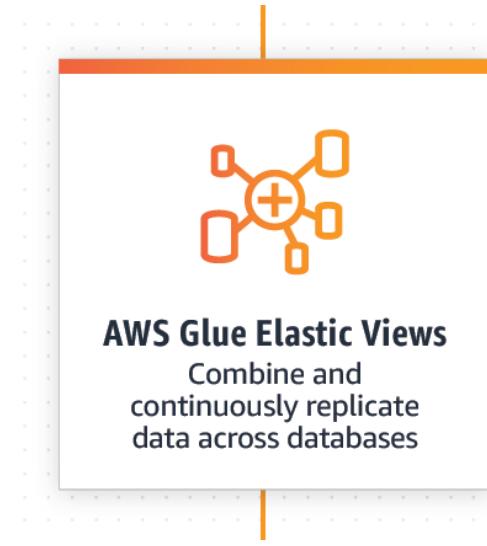
# AWS Glue DataBrew

- A visual data preparation tool
  - UI for pre-processing large data sets
  - Input from S3, data warehouse, or database
  - Output to S3
- Over 250 ready-made transformations
- You create “recipes” of transformations that can be saved as jobs within a larger project
- May define data quality rules
- May create datasets with custom SQL from Redshift and Snowflake
- Security
  - Can integrate with KMS (with customer master keys only)
  - SSL in transit
  - IAM can restrict who can do what
  - CloudWatch & CloudTrail



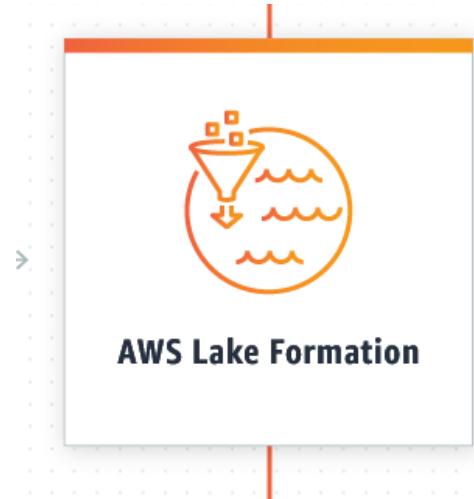
# AWS Glue Elastic Views

- Coming soon!
- Builds materialized views from Aurora, RDS, DynamoDB
- Those views can be used by Redshift, Elasticsearch, S3, DynamoDB, Aurora, RDS
- SQL interface
- Handles any copying or combining / replicating data needed
- Monitors for changes and continuously updates
- Serverless

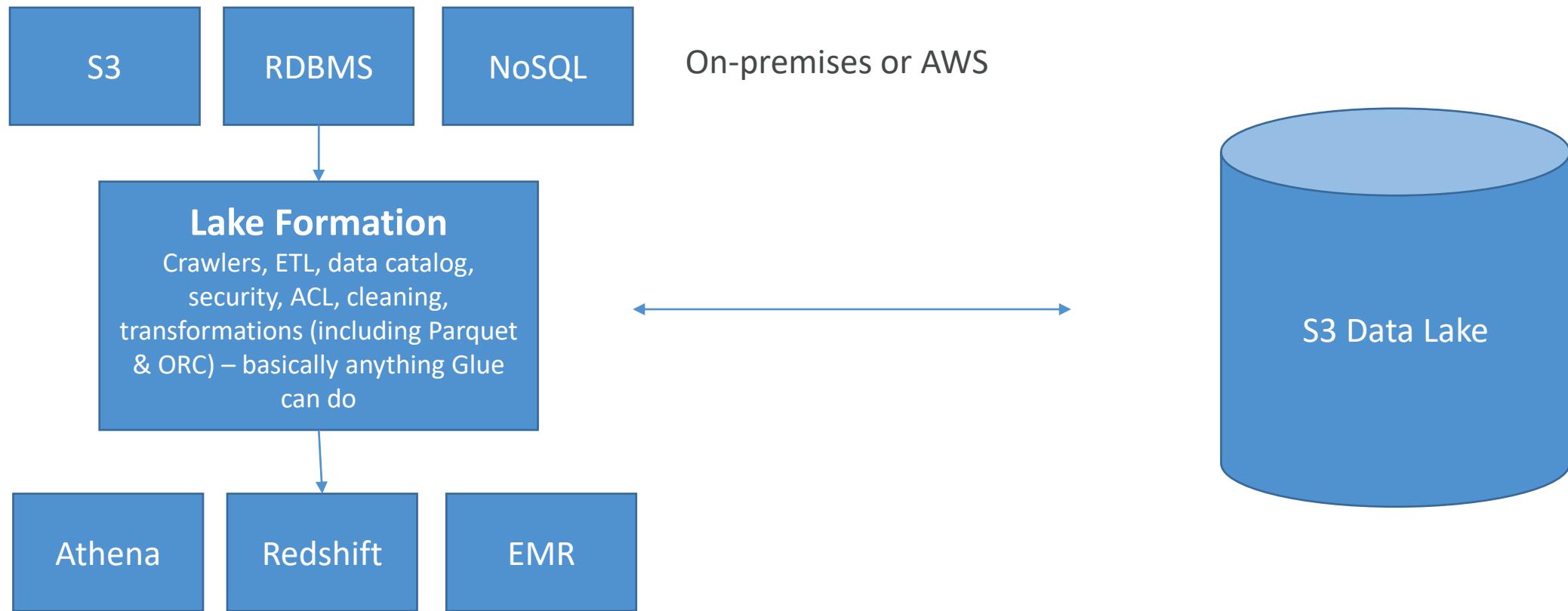


# AWS Lake Formation

- “Makes it easy to set up a secure data lake in days”
- Loading data & monitoring data flows
- Setting up partitions
- Encryption & managing keys
- Defining transformation jobs & monitoring them
- Access control
- Auditing
- Built on top of Glue



# AWS Lake Formation

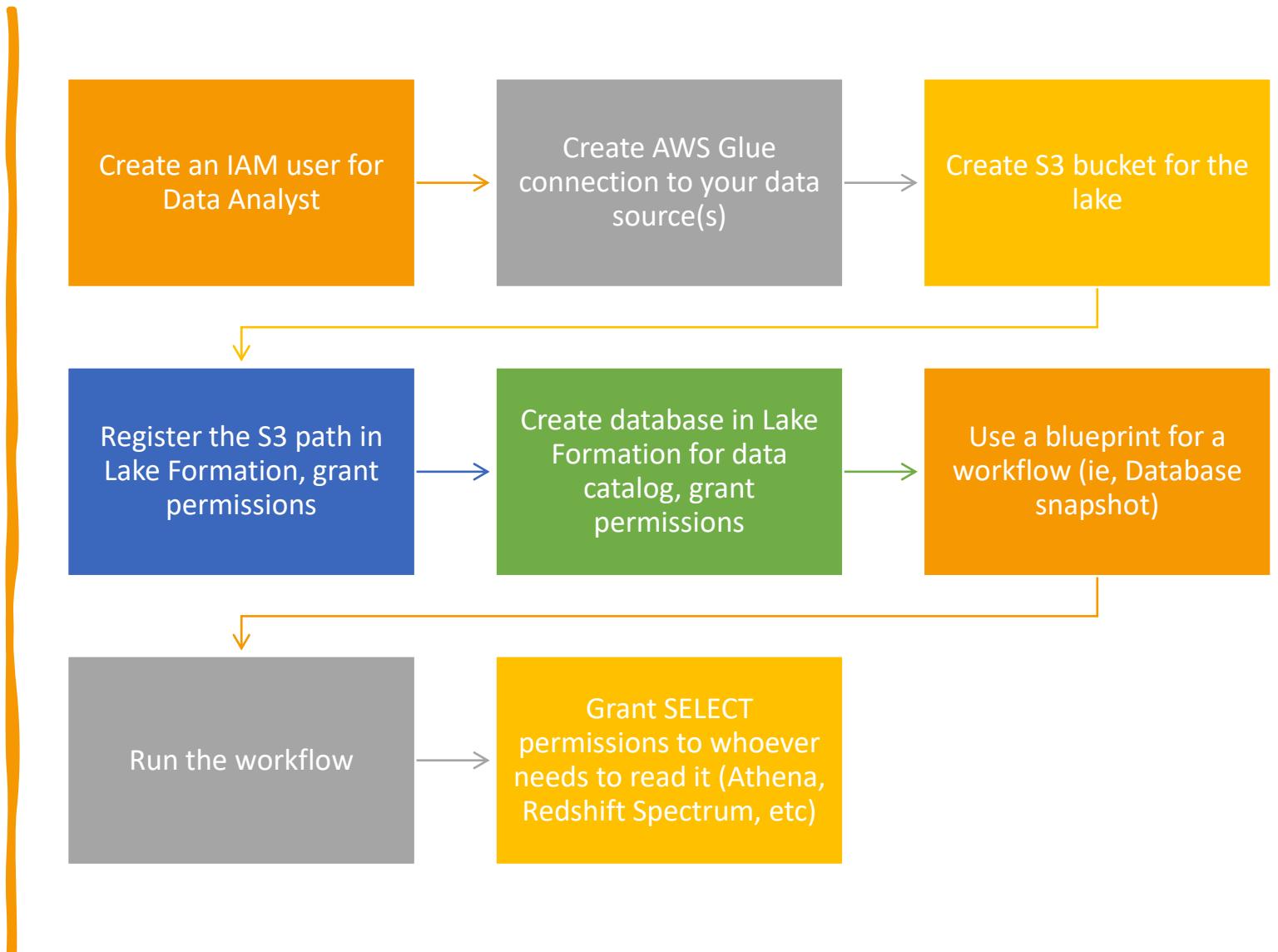


# AWS Lake Formation: Pricing

- No cost for Lake Formation itself
- But underlying services incur charges
  - Glue
  - S3
  - EMR
  - Athena
  - Redshift



# AWS Lake Formation: Building a Data Lake



# AWS Lake Formation: The Finer Points

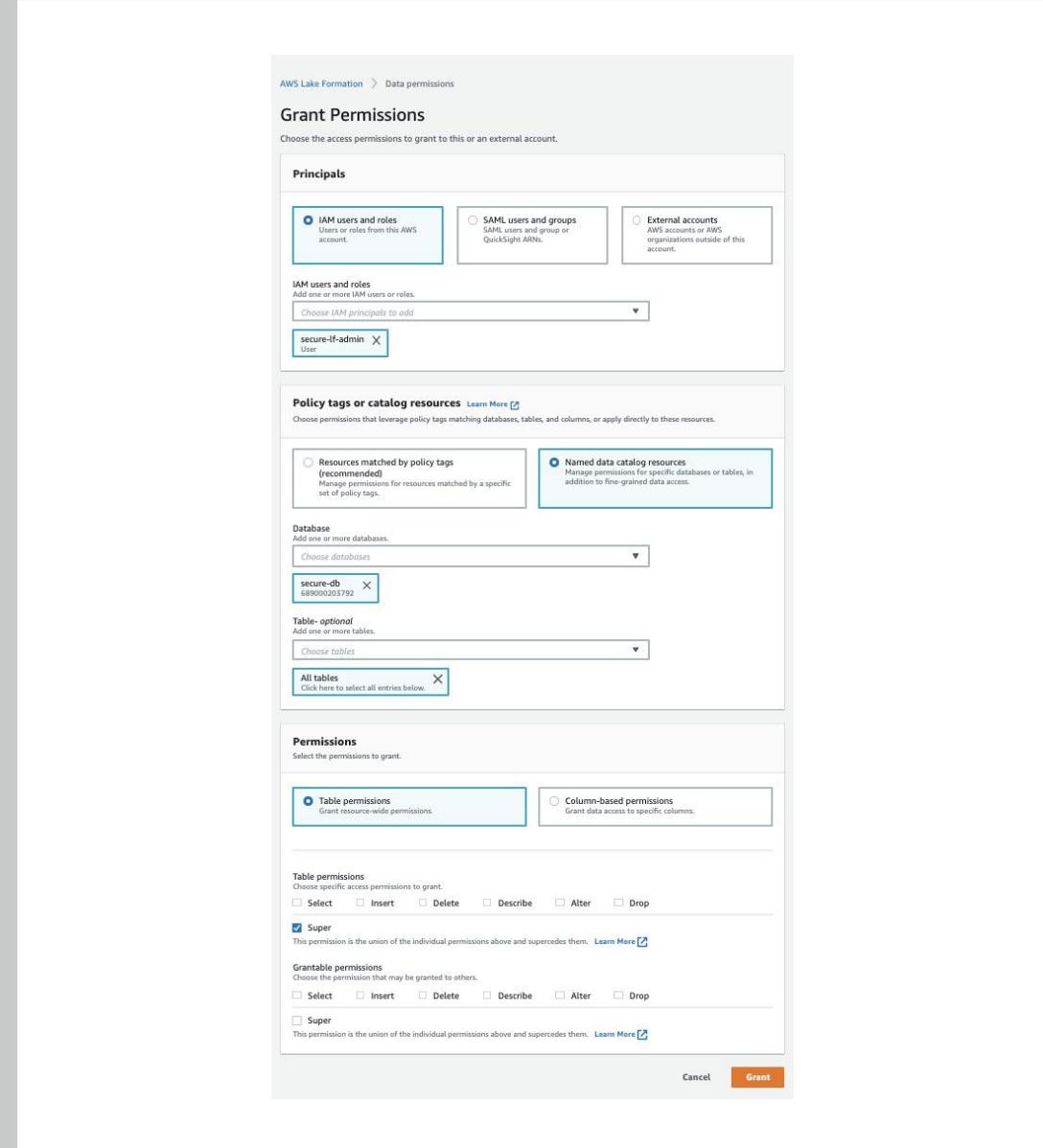
- Cross-account Lake Formation permission
  - Recipient must be set up as a data lake administrator
  - Can use AWS Resource Access Manager for accounts external to your organization
  - IAM permissions for cross-account access
- Lake Formation does not support manifests in Athena or Redshift queries
- IAM permissions on the KMS encryption key are needed for encrypted data catalogs in Lake Formation
- IAM permissions needed to create blueprints and workflows

# AWS Lake Formation: Governed Tables and Security

- Now supports “Governed Tables” that support ACID transactions across multiple tables
  - New type of S3 table
  - Can’t change choice of governed afterwards
  - Works with streaming data too (Kinesis)
  - Can query with Athena
- Storage Optimization with Automatic Compaction
- Granular Access Control with Row and Cell-Level Security
  - Both for governed and S3 tables
- Above features incur additional charges based on usage

# Data Permissions in Lake Formation

- Can tie to IAM users/roles, SAML, or external AWS accounts
- Can use policy tags on databases, tables, or columns
- Can select specific permissions for tables or columns



# EMR

## Elastic MapReduce

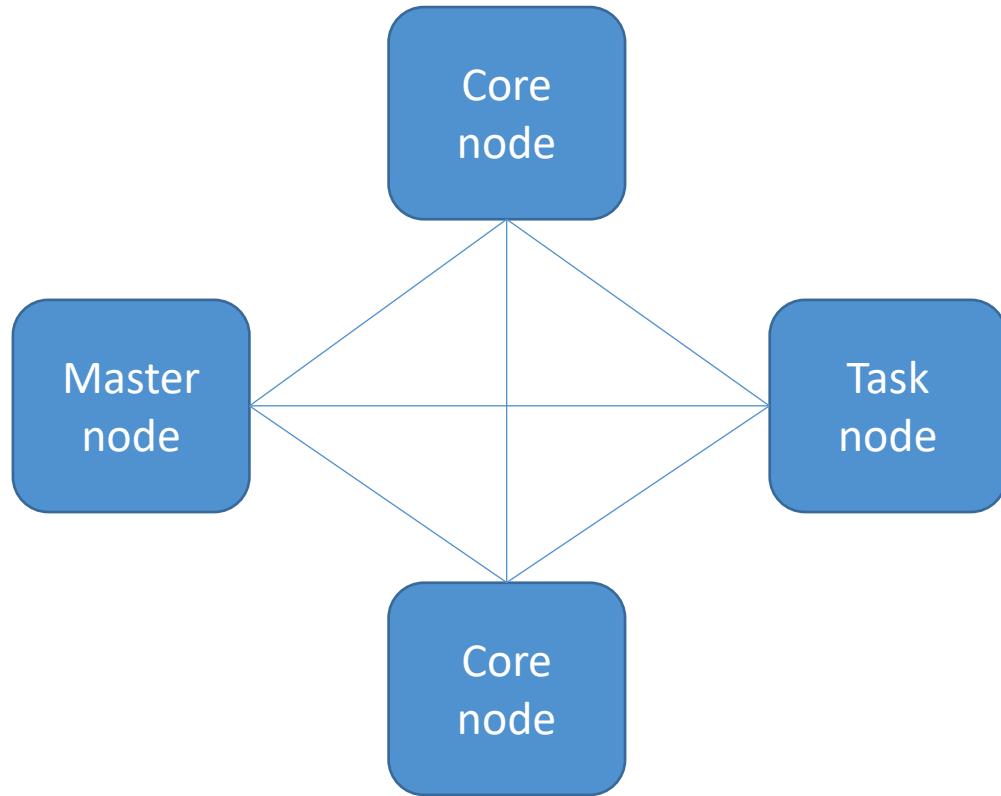
# What is EMR?

- Elastic MapReduce
- Managed Hadoop framework on EC2 instances
- Includes Spark, HBase, Presto, Flink, Hive & more
- EMR Notebooks
- Several integration points with AWS



**Amazon EMR**

# An EMR Cluster



- **Master node:** manages the cluster
  - Tracks status of tasks, monitors cluster health
  - Single EC2 instance (it can be a single node cluster even)
  - AKA “leader node”
- **Core node:** Hosts HDFS data and runs tasks
  - Can be scaled up & down, but with some risk
  - Multi-node clusters have at least one
- **Task node:** Runs tasks, does not host data
  - Optional
  - No risk of data loss when removing
  - Good use of **spot instances**

# EMR Usage

- Transient vs Long-Running Clusters
  - Transient clusters terminate once all steps are complete
    - Loading data, processing, storing – then shut down
    - Saves money
  - Long-running clusters must be manually terminated
    - Basically a data warehouse with periodic processing on large datasets
    - Can spin up task nodes using Spot instances for temporary capacity
    - Can use reserved instances on long-running clusters to save \$
    - Termination protection on by default, auto-termination off

# EMR Usage

- Frameworks and applications are specified at cluster launch
- Connect directly to master to run jobs directly
- Or, submit ordered steps via the console
  - Process data in S3 or HDFS
  - Output data to S3 or somewhere
  - Once defined, steps can be invoked via the console

# EMR / AWS Integration

- Amazon EC2 for the instances that comprise the nodes in the cluster
- Amazon VPC to configure the virtual network in which you launch your instances
- Amazon S3 to store input and output data
- Amazon CloudWatch to monitor cluster performance and configure alarms
- AWS IAM to configure permissions
- AWS CloudTrail to audit requests made to the service
- AWS Data Pipeline to schedule and start your clusters

# EMR Storage

- HDFS
  - Hadoop Distributed File System
  - Multiple copies stored across cluster instances for redundancy
  - Files stored as blocks (128MB default size)
  - Ephemeral – HDFS data is lost when cluster is terminated!
  - But, useful for caching intermediate results or workloads with significant random I/O
  - Hadoop tries to process data where it is stored on HDFS
- EMRFS: access S3 as if it were HDFS
  - Allows persistent storage after cluster termination
  - **EMRFS Consistent View** – Optional for S3 consistency
    - Uses DynamoDB to track consistency
    - May need to tinker with read/write capacity on DynamoDB
  - **New in 2021: S3 is Now Strongly Consistent!**



# EMR Storage

- Local file system
  - Suitable only for temporary data (buffers, caches, etc)
- EBS for HDFS
  - Allows use of EMR on EBS-only types (M4, C4)
  - Deleted when cluster is terminated
  - EBS volumes can only be attached when launching a cluster
  - If you manually detach an EBS volume, EMR treats that as a failure and replaces it

# EMR promises

- EMR charges by the hour
  - Plus EC2 charges
- Provisions new nodes if a core node fails
- Can add and remove tasks nodes on the fly
  - Increase processing capacity, but not HDFS capacity
- Can resize a running cluster's core nodes
  - Increases both processing and HDFS capacity
- Core nodes can also be added or removed
  - But removing risks data loss.



# EMR Managed Scaling

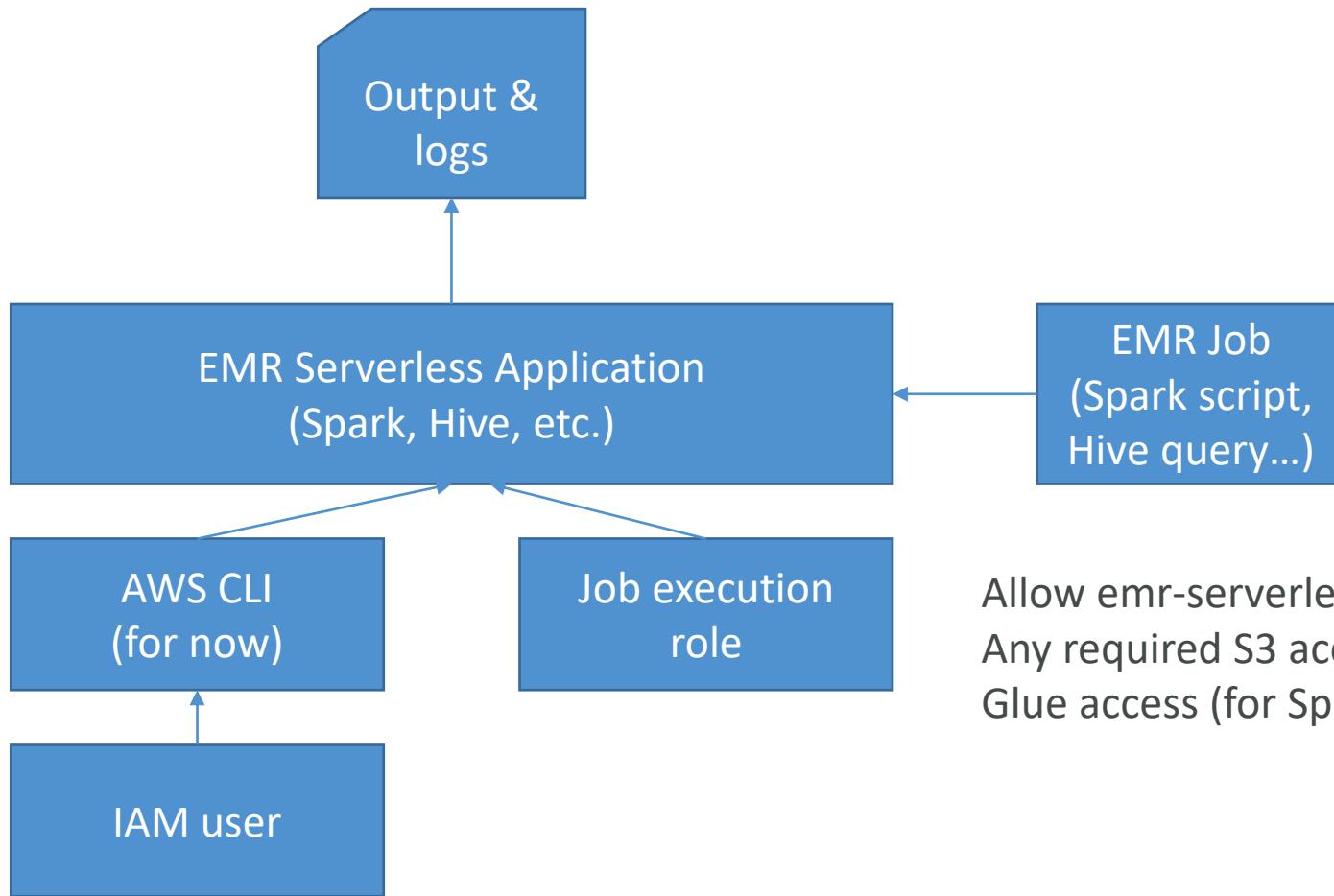
- EMR Automatic Scaling
  - The old way of doing it
  - Custom scaling rules based on CloudWatch metrics
  - Supports instance groups only
- EMR Managed Scaling
  - Introduced in 2020
  - Support instance groups and instance fleets
  - Scales spot, on-demand, and instances in a Savings Plan within the same cluster
  - Available for Spark, Hive, YARN workloads
- Scale-up Strategy
  - First adds core nodes, then task nodes, up to max units specified
- Scale-down Strategy
  - First removes task nodes, then core nodes, no further than minimum constraints
  - Spot nodes always removed before on-demand instances



# EMR Serverless

- Coming in 2022
- Choose an EMR Release and Runtime (Spark, Hive, Presto)
- Submit queries / scripts via job run requests
- EMR manages underlying capacity
  - But you can specify default worker sizes & pre-initialized capacity
  - EMR computes resources needed for your job & schedules workers accordingly
  - All within one region (across multiple AZ's)
- Why is this a big deal?
  - You no longer have to estimate how many workers are needed for your workloads – they are provisioned as needed, automatically.
- Serverless? Really?
  - TBH you still need to think about worker nodes and how they are configured.

# Using EMR Serverless

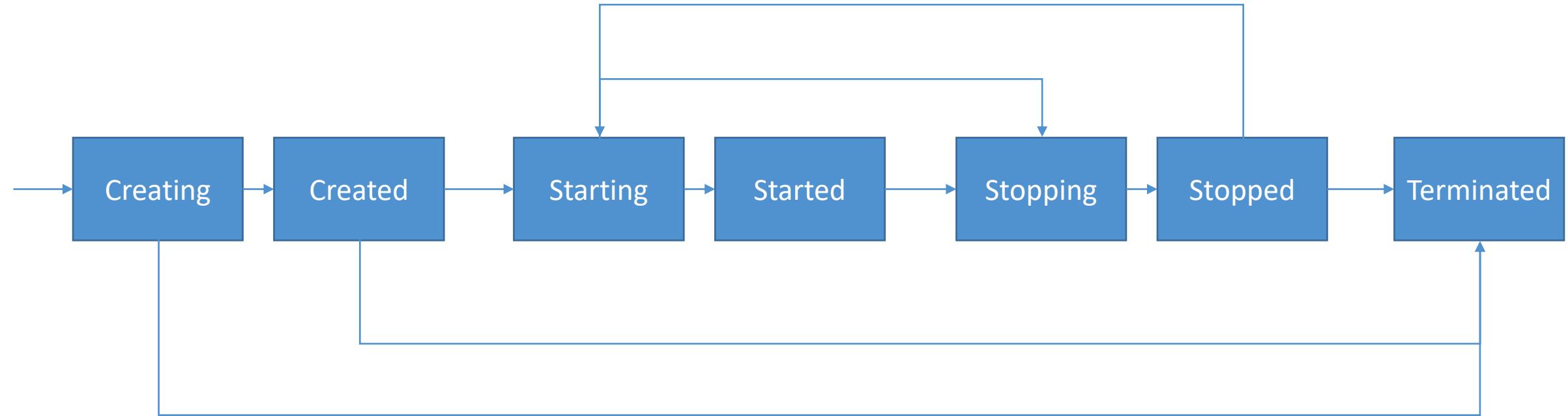


```

aws emr-serverless start-job-run \
--application-id <application_id> \
--execution-role-arn <execution_role_arn> \
--job-driver '{'
  "sparkSubmit": {
    "entryPoint": "s3://us-east-1.elasticmapreduce/emr-
containers/samples/wordcount/scripts/wordcount.py",
    "entryPointArguments": ["s3://DOC-EXAMPLE-
BUCKET/output"],
    "sparkSubmitParameters": "--conf spark.executor.cores=1 --conf
spark.executor.memory=4g --conf spark.driver.cores=1 --conf
spark.driver.memory=4g --conf spark.executor.instances=1"
  }
}' \
--configuration-overrides '{
  "monitoringConfiguration": {
    "s3MonitoringConfiguration": {
      "logUri": "s3://DOC-EXAMPLE-BUCKET/logs"
    }
  }
}'
  
```

Allow emr-serverless.amazonaws.com service  
Any required S3 access for scripts & data  
Glue access (for SparkSQL), KMS keys

# EMR Serverless Application Lifecycle



This isn't all automatic!

Must call API's such as CreateApplication, StartApplication, StopApplication, and importantly DeleteApplication to avoid excess charges.

# Pre-Initialized Capacity

- Spark adds 10% overhead to memory requested for drivers & executors
- Be sure initial capacity is at least 10% more than requested by the job

```
aws emr-serverless create-application \
--type "SPARK" \
--name <"my_application_name"> \
--release-label "emr-6.5.0-preview" \
--initial-capacity'{
  "DRIVER": {
    "workerCount": 5,
    "resourceConfiguration": {
      "cpu": "2vCPU",
      "memory": "4GB"
    }
  },
  "EXECUTOR": {
    "workerCount": 50,
    "resourceConfiguration": {
      "cpu": "4vCPU",
      "memory": "8GB"
    }
  }
}' \
--maximum-capacity'{
  "cpu": "400vCPU",
  "memory": "1024GB"
}'
```

# EMR Serverless Security

- Basically the same as EMR
- EMRFS
  - S3 encryption (SSE or CSE) at rest
  - TLS in transit between EMR nodes and S3
- S3
  - SSE-S3, SSE-KMS
- Local disk encryption
- Spark communication between drivers & executors is encrypted
- Hive communication between Glue Metastore and EMR uses TLS
- Force HTTPS (TLS) on S3 policies with aws:SecureTransport



# So... what's Hadoop?

## MapReduce

Framework for distributed data processing  
Maps data to key/value pairs  
Reduces intermediate results to final output  
Largely supplanted by Spark these days

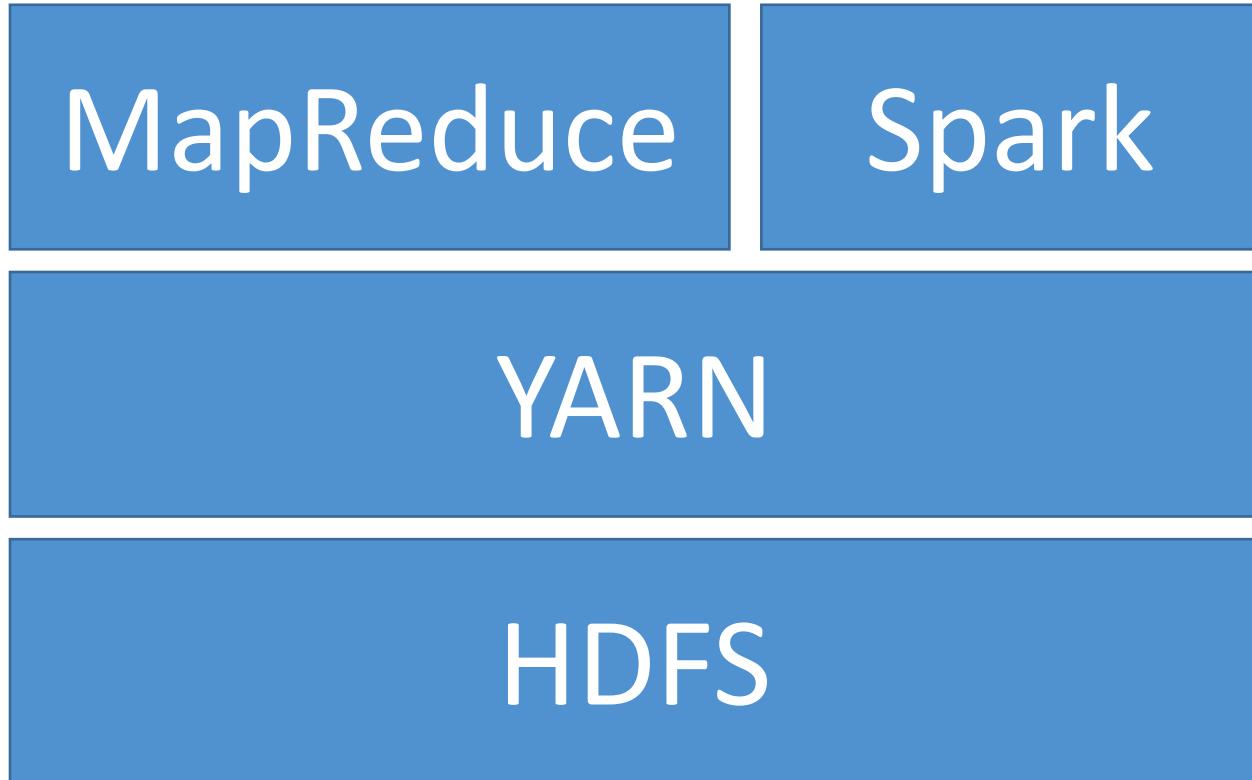
## YARN

Yet Another Resource Negotiator  
Manages cluster resources for multiple data processing frameworks

## HDFS

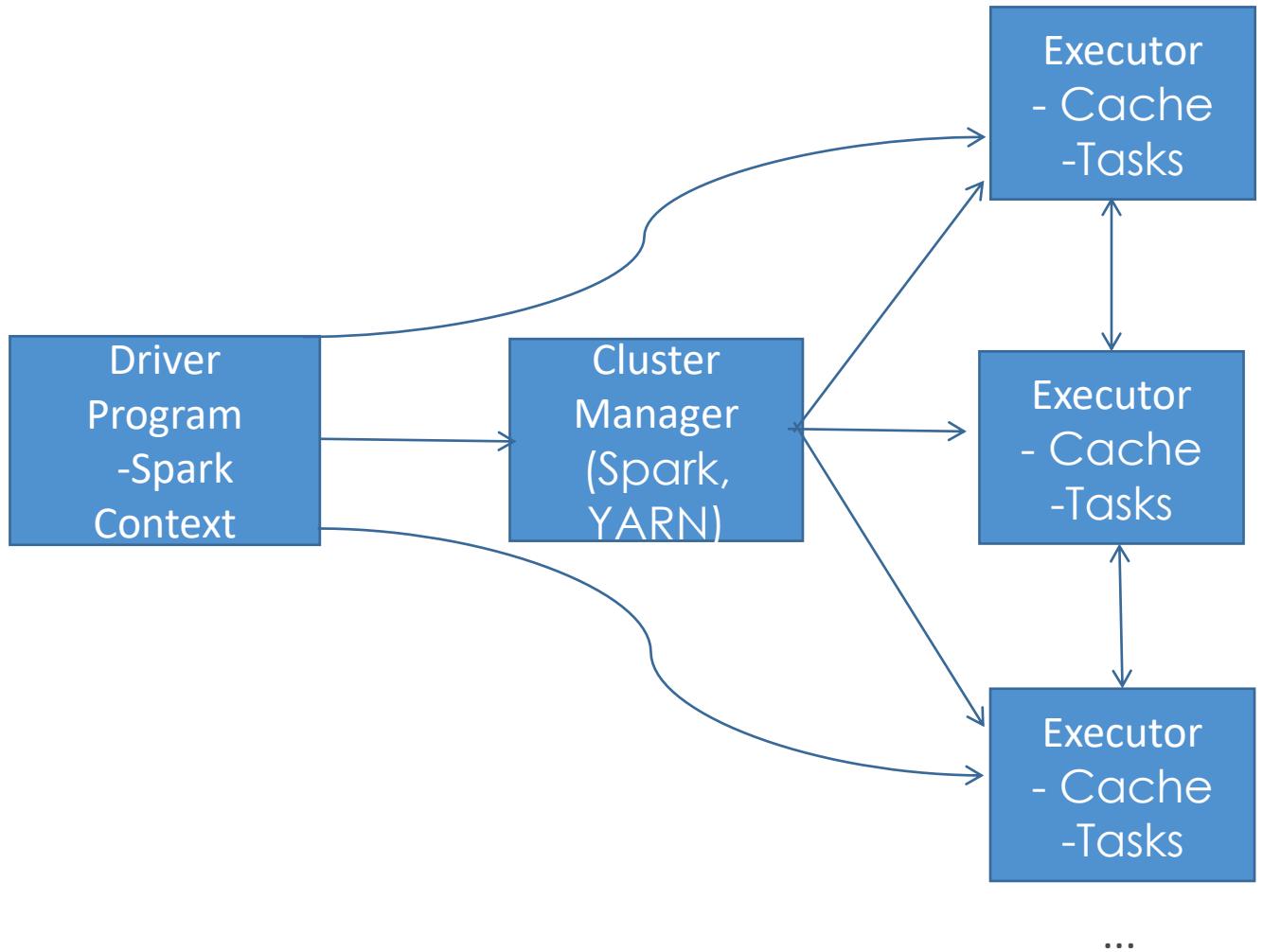
Hadoop Distributed File System  
Distributes data blocks across cluster in a redundant manner  
Ephemeral in EMR; data lost on termination

# Apache Spark



- Distributed processing framework for big data
- In-memory caching, optimized query execution
- Supports Java, Scala, Python, and R
- Supports code reuse across
  - Batch processing
  - Interactive Queries
    - Spark SQL
  - Real-time Analytics
  - Machine Learning
    - MLLib
  - Graph Processing
- Spark Streaming
  - Integrated with Kinesis, Kafka, on EMR
- Spark is NOT meant for OLTP

# How Spark Works



- Spark apps are run as independent processes on a cluster
- The `SparkContext` (driver program) coordinates them
- `SparkContext` works through a Cluster Manager
- Executors run computations and store data
- `SparkContext` sends application code and tasks to executors

# Spark Components

## Spark Streaming

Real-time streaming analytics  
Structured streaming  
Twitter, Kafka, Flume, HDFS,  
ZeroMQ

## Spark SQL

Up to 100x faster than  
MapReduce  
JDBC, ODBC, JSON, HDFS, ORC,  
Parquet, HiveQL

## MLLib

Classification, regression,  
clustering, collaborative  
filtering, pattern mining  
Read from HDFS, HBase...

## GraphX

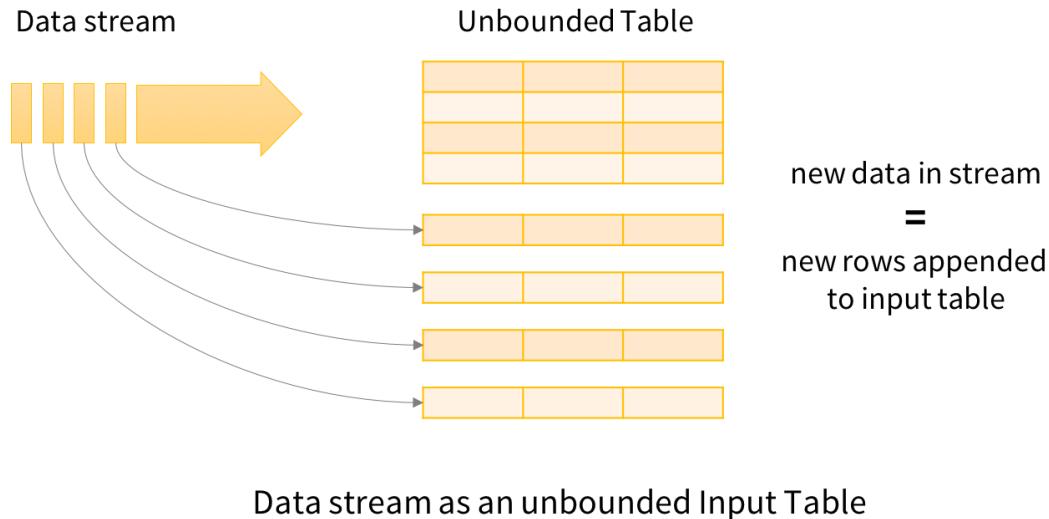
Graph Processing  
ETL, analysis, iterative graph  
computation  
No longer widely used

## SPARK CORE

Memory management, fault recovery, scheduling, distribute & monitor jobs, interact with storage  
Scala, Python, Java, R

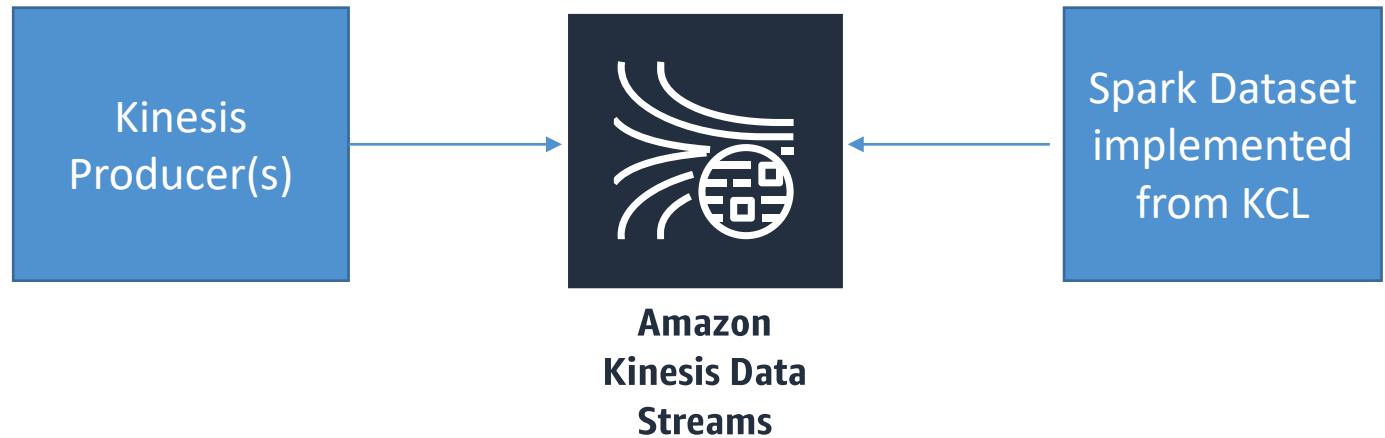
# Spark Structured Streaming

## A constantly growing DataSet



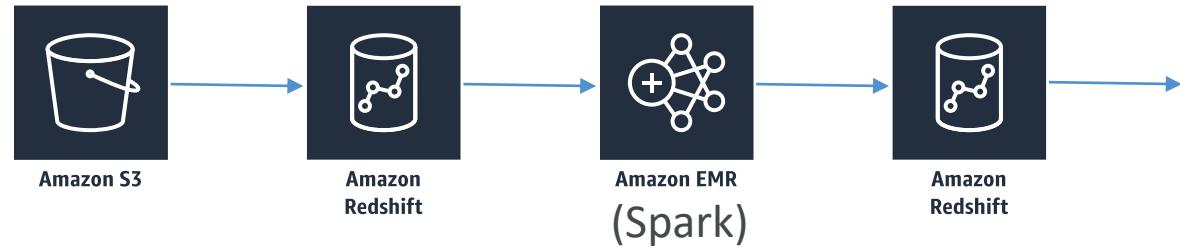
```
val inputDF = spark.readStream.json("s3://logs")
inputDF.groupBy($"action", window($"time", "1 hour")).count()
    .writeStream.format("jdbc").start("jdbc:mysql//...")
```

# Spark Streaming + Kinesis

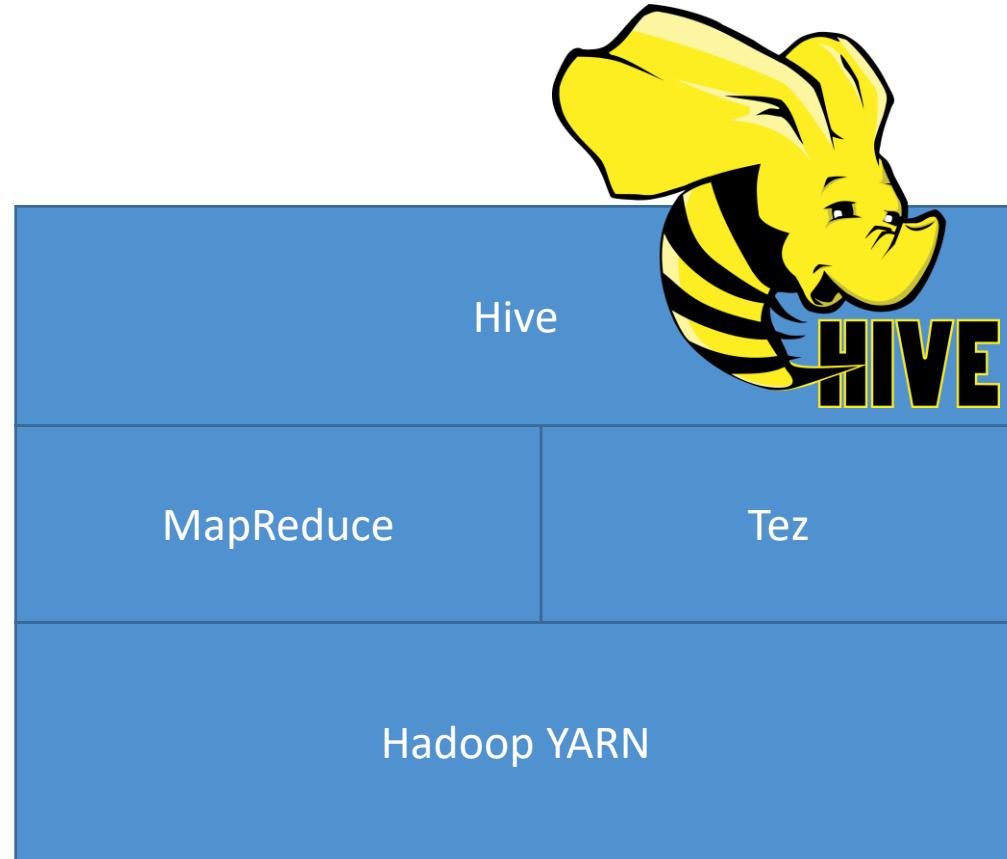


# Spark + Redshift

- spark-redshift package allows Spark datasets from Redshift
  - It's a Spark SQL data source
- Useful for ETL using Spark

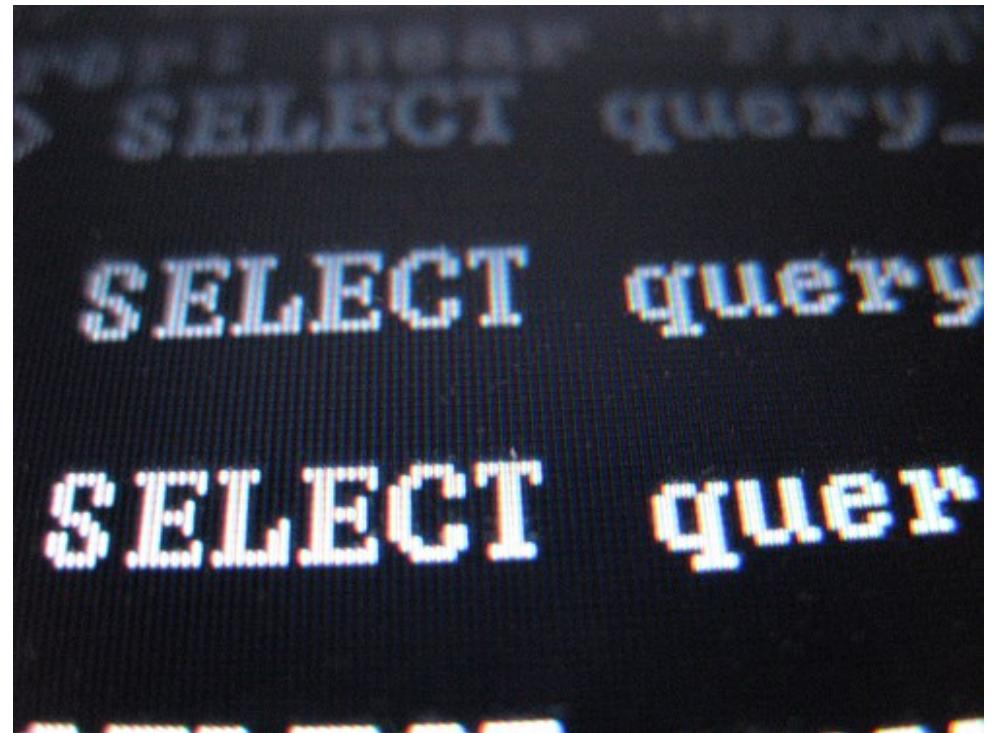


# Apache Hive



# Why Hive?

- Uses familiar SQL syntax (HiveQL)
- Interactive
- Scalable – works with “big data” on a cluster
  - Really most appropriate for data warehouse applications
- Easy OLAP queries – WAY easier than writing MapReduce in Java
- Highly optimized
- Highly extensible
  - User defined functions
  - Thrift server
  - JDBC / ODBC driver



# The Hive Metastore

- Hive maintains a “metastore” that imparts a structure you define on the unstructured data that is stored on HDFS etc.

```
CREATE TABLE ratings (
    userID INT,
    movieID      INT,
    rating INT,
    time   INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '${env:HOME}/ml-100k/u.data'
OVERWRITE INTO TABLE ratings;
```

# External Hive Metastores

- Metastore is stored in MySQL on the master node by default
- External metastores offer better resiliency / integration
  - AWS Glue Data Catalog
    - Shares schema across EMR and other AWS services
    - Tie Glue to EMR using the console, CLI, or API
  - Amazon RDS / Aurora
    - Need to override default Hive configuration values for external database location



# Other Hive / AWS integration points

- Load table partitions from S3
  - “alter table recover partitions”
- Write tables directly to S3
- Load scripts from S3
- DynamoDB as an external table
  - Read/write access
  - Copy to/from HDFS
  - Perform JOIN's on DynamoDB

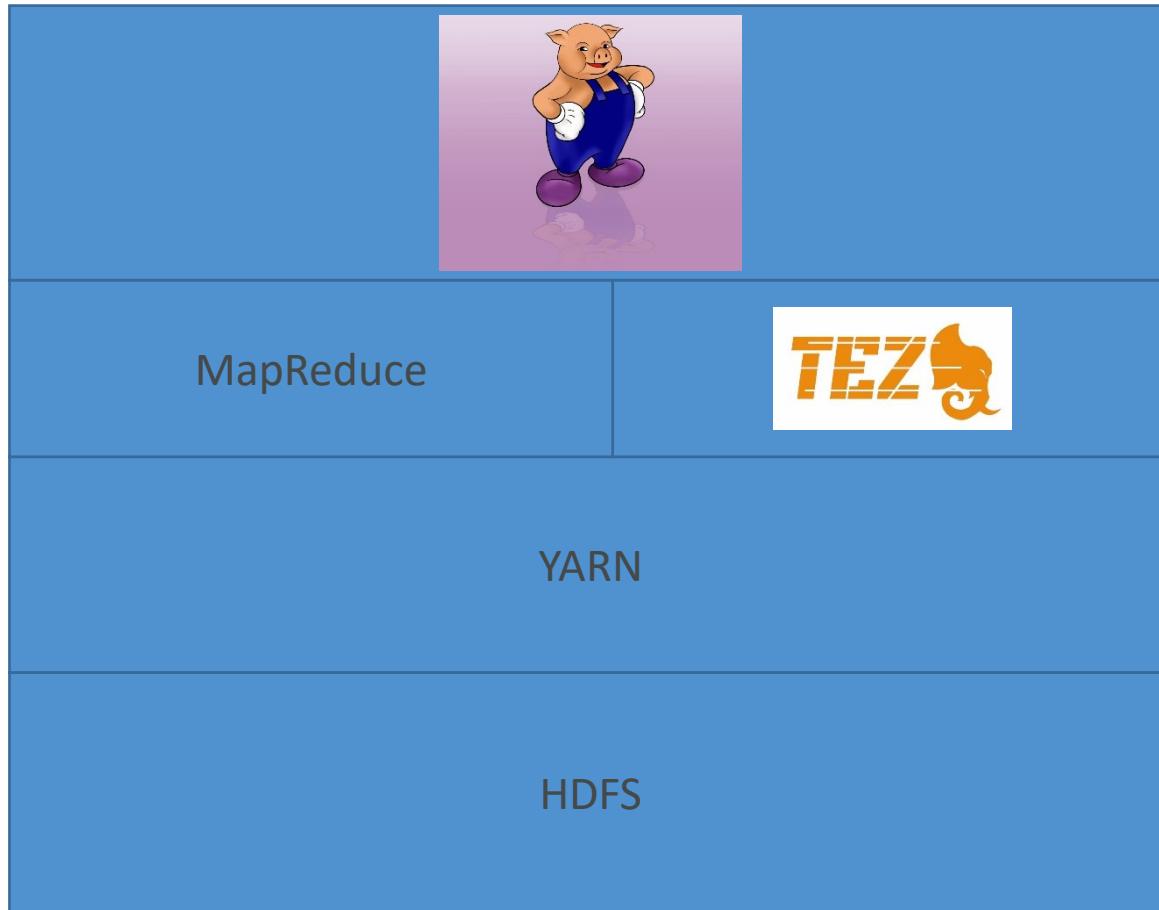


# Apache Pig

- Writing mappers and reducers by hand takes a long time.
- Pig introduces *Pig Latin*, a scripting language that lets you use SQL-like syntax to define your map and reduce steps.
- Highly extensible with user-defined functions (UDF's)



# How Pig Works



```

ratings = LOAD '/user/maria_dev/ml-100k/u.data' AS (userID:int, movieID:int, rating:int, ratingTime:int);

metadata = LOAD '/user/maria_dev/ml-100k/u.item' USING PigStorage('|)
    AS (movieID:int, movieTitle:chararray, releaseDate:chararray, videoRelease:chararray, imdbLink:chararray);

nameLookup = FOREACH metadata GENERATE movieID, movieTitle,
    ToUnixTime(ToDate(releaseDate, 'dd-MMM-yyyy')) AS releaseTime;

ratingsByMovie = GROUP ratings BY movieID;

avgRatings = FOREACH ratingsByMovie GENERATE group AS movieID, AVG(ratings.rating) AS avgRating;

fiveStarMovies = FILTER avgRatings BY avgRating > 4.0;

fiveStarsWithData = JOIN fiveStarMovies BY movieID, nameLookup BY movieID;

oldestFiveStarMovies = ORDER fiveStarsWithData BY nameLookup::releaseTime;

DUMP oldestFiveStarMovies;

```

# Pig / AWS Integration

- Ability to use multiple file systems (not just HDFS)
  - i.e., query data in S3
- Load JAR's and scripts from S3



Amazon S3

# HBase

- Non-relational, petabyte-scale database
- Based on Google's BigTable, on top of HDFS
- In-memory
- Hive integration



# Sounds a lot like DynamoDB

- Both are NoSQL databases intended for the same sorts of things
- But if you're all-in with AWS anyhow, DynamoDB has advantages
  - Fully managed (auto-scaling)
  - More integration with other AWS services
  - Glue integration
- HBase has some advantages though:
  - Efficient storage of sparse data
  - Appropriate for high frequency counters (consistent reads & writes)
  - High write & update throughput
  - More integration with Hadoop

# HBase / AWS integration

- Can store data (StoreFiles and metadata) on S3 via EMRFS
- Can back up to S3



Amazon S3

# Presto

- It can connect to many different “big data” databases and data stores at once, and query across them
- **Interactive queries at petabyte scale**
- Familiar SQL syntax
- Optimized for OLAP – analytical queries, data warehousing
- Developed, and still partially maintained by Facebook
- This is what Amazon Athena uses under the hood
- Exposes JDBC, Command-Line, and Tableau interfaces



# Presto connectors

- HDFS
- S3
- Cassandra
- MongoDB
- HBase
- SQL
- Redshift
- Teradata



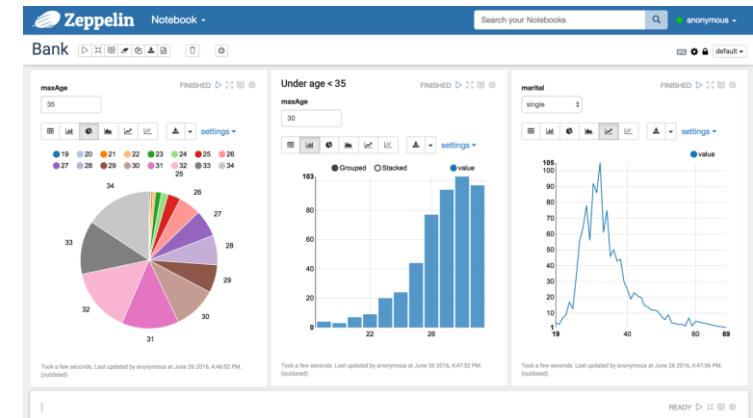
# Apache Zeppelin



- If you're familiar with iPython notebooks – it's like that
  - Lets you interactively run scripts / code against your data
  - Can interleave with nicely formatted notes
  - Can share notebooks with others on your cluster
- Spark, Python, JDBC, HBase, Elasticsearch + more

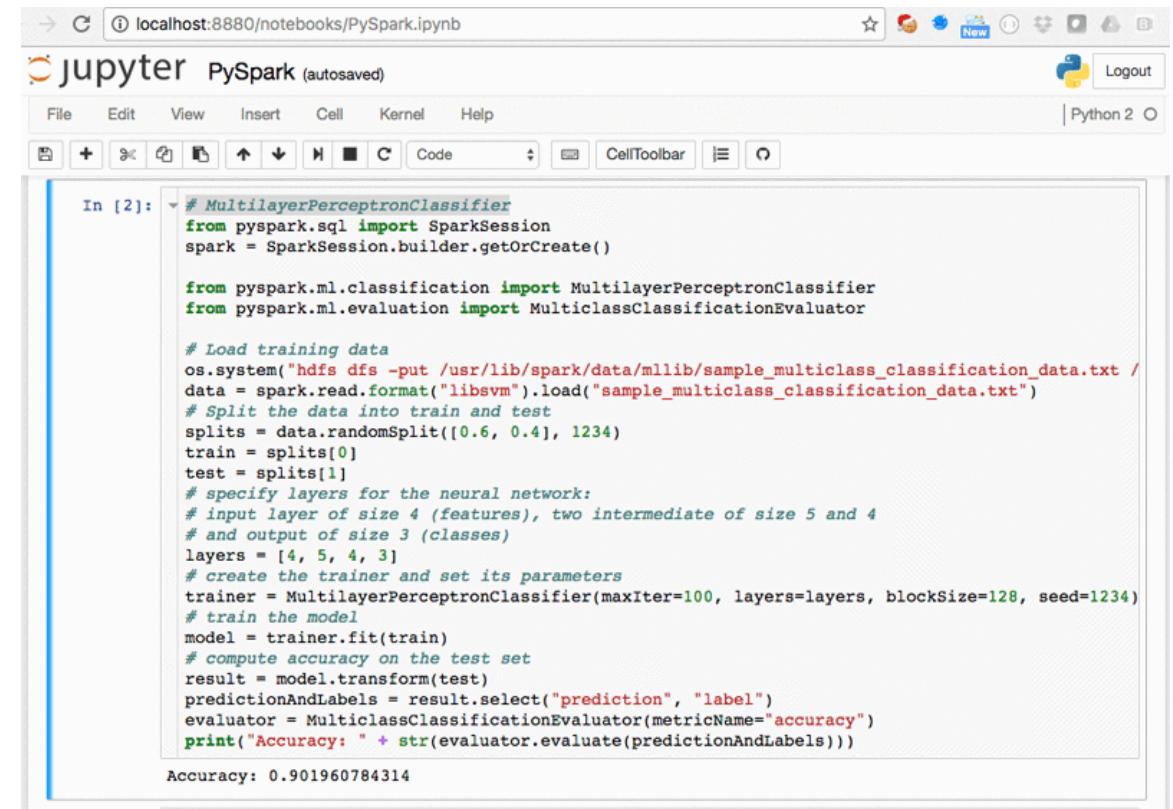
# Zeppelin + Spark

- Can run Spark code interactively (like you can in the Spark shell)
  - This speeds up your development cycle
  - And allows easy experimentation and exploration of your big data
- Can execute SQL queries directly against SparkSQL
- Query results may be visualized in charts and graphs
- Makes Spark feel more like a data science tool!



# EMR Notebook

- Similar concept to Zeppelin, with more AWS integration
- Notebooks backed up to S3
- Provision clusters from the notebook!
- Hosted inside a VPC
- Accessed only via AWS console



The screenshot shows a Jupyter Notebook interface with the title "PySpark" in the header. The notebook URL is "localhost:8880/notebooks/PySpark.ipynb". The code in cell [2] is as follows:

```
# MultilayerPerceptronClassifier
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

from pyspark.ml.classification import MultilayerPerceptronClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Load training data
os.system("hdfs dfs -put /usr/lib/spark/data/mllib/sample_multiclass_classification_data.txt /")
data = spark.read.format("libsvm").load("sample_multiclass_classification_data.txt")

# Split the data into train and test
splits = data.randomSplit([0.6, 0.4], 1234)
train = splits[0]
test = splits[1]

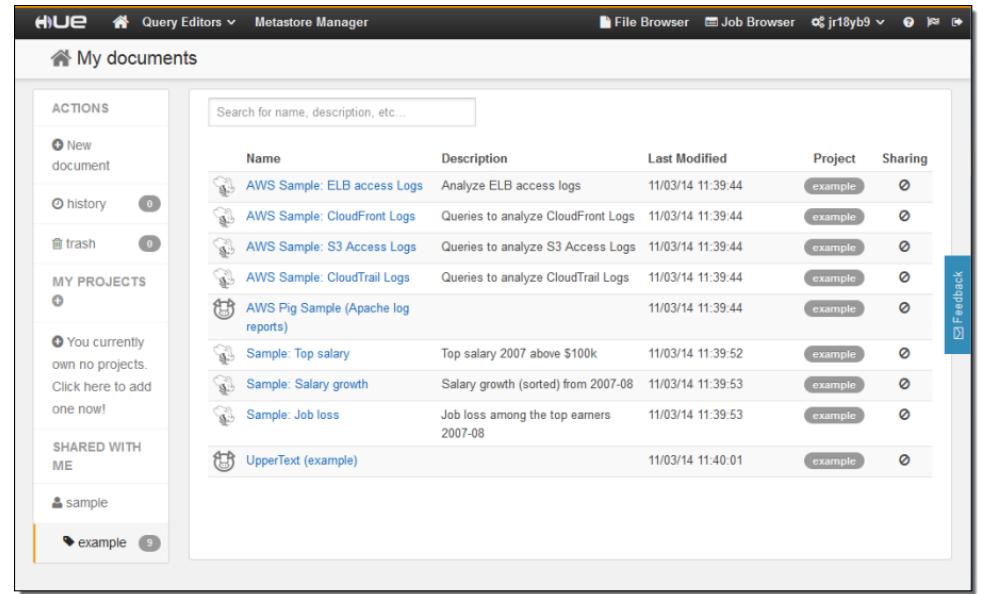
# specify layers for the neural network:
# input layer of size 4 (features), two intermediate of size 5 and 4
# and output of size 3 (classes)
layers = [4, 5, 4, 3]
# create the trainer and set its parameters
trainer = MultilayerPerceptronClassifier(maxIter=100, layers=layers, blockSize=128, seed=1234)
# train the model
model = trainer.fit(train)
# compute accuracy on the test set
result = model.transform(test)
predictionAndLabels = result.select("prediction", "label")
evaluator = MulticlassClassificationEvaluator(metricName="accuracy")
print("Accuracy: " + str(evaluator.evaluate(predictionAndLabels)))
```

The output of the code is:

```
Accuracy: 0.901960784314
```

# Hue

- Hadoop User Experience
- Graphical front-end for applications on your EMR cluster
- IAM integration: Hue Super-users inherit IAM roles
- S3: Can browse & move data between HDFS and S3

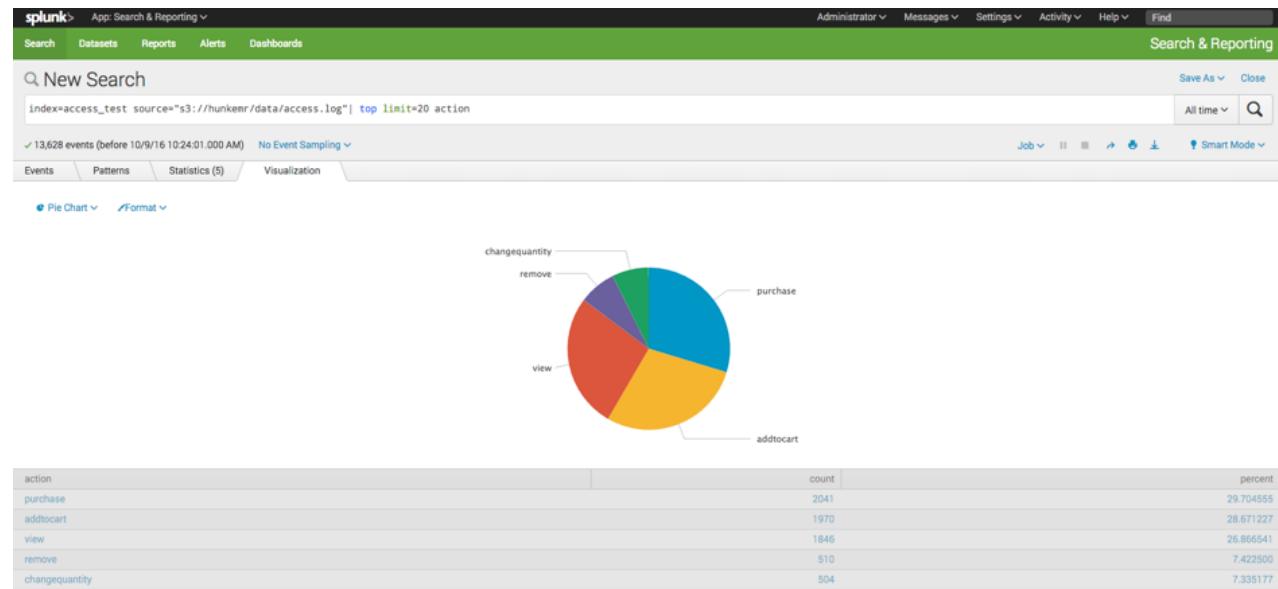


The screenshot shows the Hue web interface with the title 'HUE' at the top. The main area is titled 'My documents' and contains a table of data. The table has columns for Name, Description, Last Modified, Project, and Sharing. The data includes various sample queries and logs from AWS services like ELB, CloudFront, S3, and CloudTrail, along with some Pig samples and salary-related queries. A sidebar on the left shows options for 'New document', 'history', 'trash', 'MY PROJECTS', and 'SHARED WITH ME'.

Name	Description	Last Modified	Project	Sharing
AWS Sample: ELB access Logs	Analyze ELB access logs	11/03/14 11:39:44	example	0
AWS Sample: CloudFront Logs	Queries to analyze CloudFront Logs	11/03/14 11:39:44	example	0
AWS Sample: S3 Access Logs	Queries to analyze S3 Access Logs	11/03/14 11:39:44	example	0
AWS Sample: CloudTrail Logs	Queries to analyze CloudTrail Logs	11/03/14 11:39:44	example	0
AWS Pig Sample (Apache log reports)		11/03/14 11:39:44	example	0
Sample: Top salary	Top salary 2007 above \$100k	11/03/14 11:39:52	example	0
Sample: Salary growth	Salary growth (sorted) from 2007-08	11/03/14 11:39:53	example	0
Sample: Job loss	Job loss among the top earners 2007-08	11/03/14 11:39:53	example	0
UpperText (example)		11/03/14 11:40:01	example	0

# Splunk

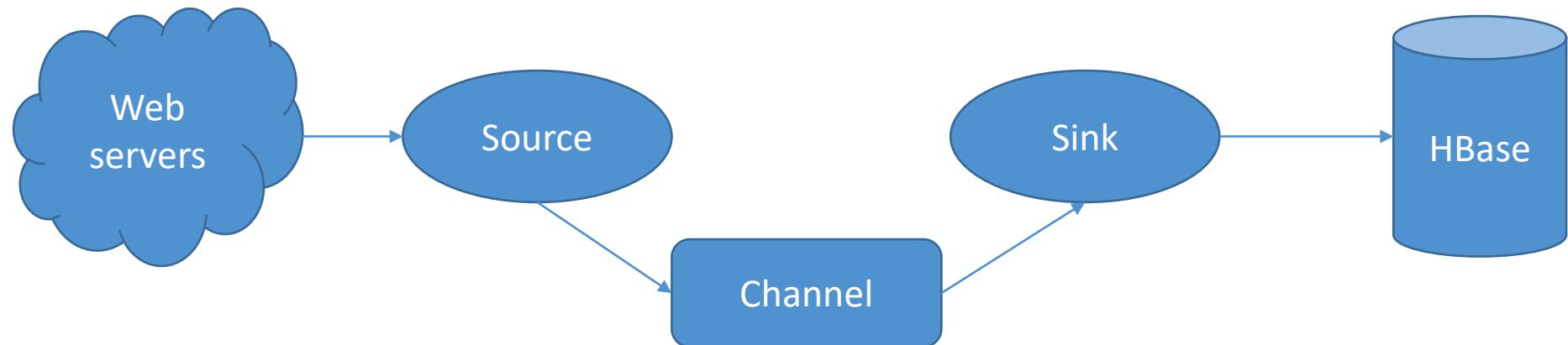
- Splunk / Hunk “makes machine data accessible, usable, and valuable to everyone”
- Operational tool – can be used to visualize EMR and S3 data using your EMR Hadoop cluster.
- Reserved instances on 64-bit OS recommended
  - A public AMI of Splunk Enterprise is available



# Flume



- Another way to stream data into your cluster
- Made from the start with Hadoop in mind
  - Built-in sinks for HDFS and HBase
- Originally made to handle log aggregation



# MXNet

- Like Tensorflow, a library for building and accelerating neural networks
- Included on EMR



# S3DistCP

- Tool for copying large amounts of data
  - From S3 into HDFS
  - From HDFS into S3
- Uses MapReduce to copy in a distributed manner
- Suitable for parallel copying of large numbers of objects
  - Across buckets, across accounts



# Other EMR / Hadoop Tools

- **Ganglia** (monitoring)
- **Mahout** (machine learning)
- **Accumulo** (another NoSQL database)
- **Sqoop** (relational database connector)
- **HCatalog** (table and storage management for Hive metastore)
- **Kinesis Connector** (directly access Kinesis streams in your scripts)
- **Tachyon** (accelerator for Spark)
- **Derby** (open-source relational DB in Java)
- **Ranger** (data security manager for Hadoop)
- Install whatever you want

# EMR Security

- IAM policies
  - Grant or deny permissions
  - Allow user actions
  - Combine with tagging to control access per cluster
- Kerberos
  - Secure user authentication
- SSH
  - Secure connection to command line
  - Tunneling for web interfaces
  - Can use Kerberos or EC2 key pairs
- IAM roles
  - Control access to EMRFS data based on user, group, location of data
  - Each cluster must have a service role and a role for the EC2 instance profile
  - IAM policies attached to roles
  - Auto-scaling role
  - Service-linked roles



# EMR Security

- “Block public access”
  - Easy way to prevent public access to data stored on your EMR cluster
  - Can set at the account level before creating the cluster.



# EMR: Choosing Instance Types

- Master node:
  - m5.xlarge if < 50 nodes, larger if > 50 nodes
- Core & task nodes:
  - m5.xlarge is usually good
  - If cluster waits a lot on external dependencies (i.e. a web crawler), t2.medium
  - Improved performance: m4.xlarge
  - Computation-intensive applications: high CPU instances
  - Database, memory-caching applications: high memory instances
  - Network / CPU-intensive (NLP, ML) – cluster computer instances
- Spot instances
  - Good choice for task nodes
  - Only use on core & master if you're testing or very cost-sensitive; you're risking partial data loss

# Amazon Machine Learning

ML with linear and logistic regression

# Machine Learning 101

- Machine learning systems predict some unknown property of an item, given its other properties
- Examples:
  - How much will this house sell for?
  - What is this a picture of?
  - Is this biopsy result malignant?
  - Is this financial transaction fraudulent?



# Supervised Learning

- Supervised machine learning systems are **trained**
  - The property we want to predict is called a **label**
  - Our **training data set** contains labels known to be correct, together with the other attributes of the data (i.e., known house sale price given its location, # of bedrooms, square feet, etc.)
  - This training data is used to build a **model** that can then make **predictions** of unknown labels



# Train / Test

- Your training data can be randomly split into a **training set** and a **test set**
- Only the training set is used to train the model
- The model is then used on the test set
- We can then measure the **accuracy** of the predicted labels vs. their actual labels



# Types of models in Amazon ML

Example	Model type
What price will this house sell for?	Regression
What is this a picture of?	Multiclass Classification
Is this biopsy result malignant?	Binary Classification
Is this financial transaction fraudulent?	Binary Classification



# Confusion Matrix

- A way to visualize the accuracy of multiclass classification predictive models

Predicted label	Dog	Cat	Fish
True label	1.00	0.00	0.00
Dog	0.00	0.62	0.38
Cat	0.00	0.00	1.00
Fish	0.00	0.00	1.00



# Hyperparameters

- Machine learning models often depend on tuning the parameters of the model itself
  - This is called **hyperparameter tuning**
- Parameters in Amazon ML include:
  - Learning rate
  - Model size
  - Number of passes
  - Data shuffling
  - Regularization



# Amazon Machine Learning (ML)

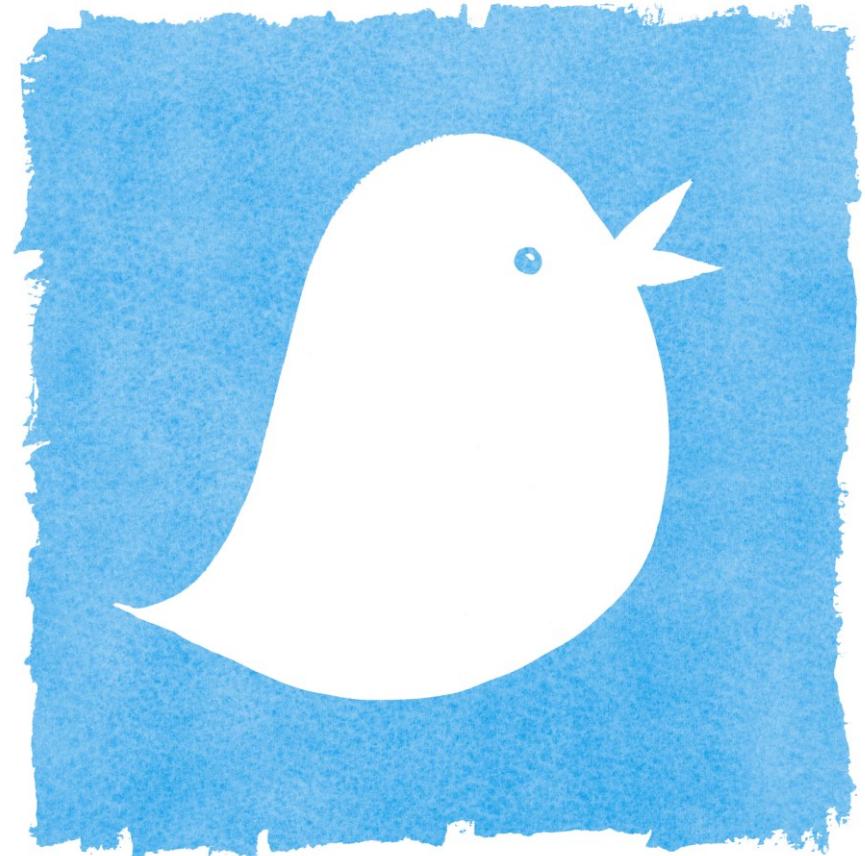
- Provides visualization tools & wizards to make creating a model easy
- You point it to training data in S3, Redshift, or RDS
- It builds a model than can make predictions using batches or a low-latency API
- Can do train/test and evaluate your model
- Fully managed
- Honestly it's a bit outdated now



Amazon  
Machine  
Learning

# “Ideal Usage Patterns”

- Flag suspicious transactions (fraud detection)
- Forecasting product demand
- Personalization – predict items a user will be interested in
- Predict user activity (we'll do this)
- Classify social media (does this Tweet require my attention?)



# Amazon ML: Cost Model

- “Pay for what you use”
- Charged for compute time
- Number of predictions
- Memory used to run your model
- Compute-hours for training

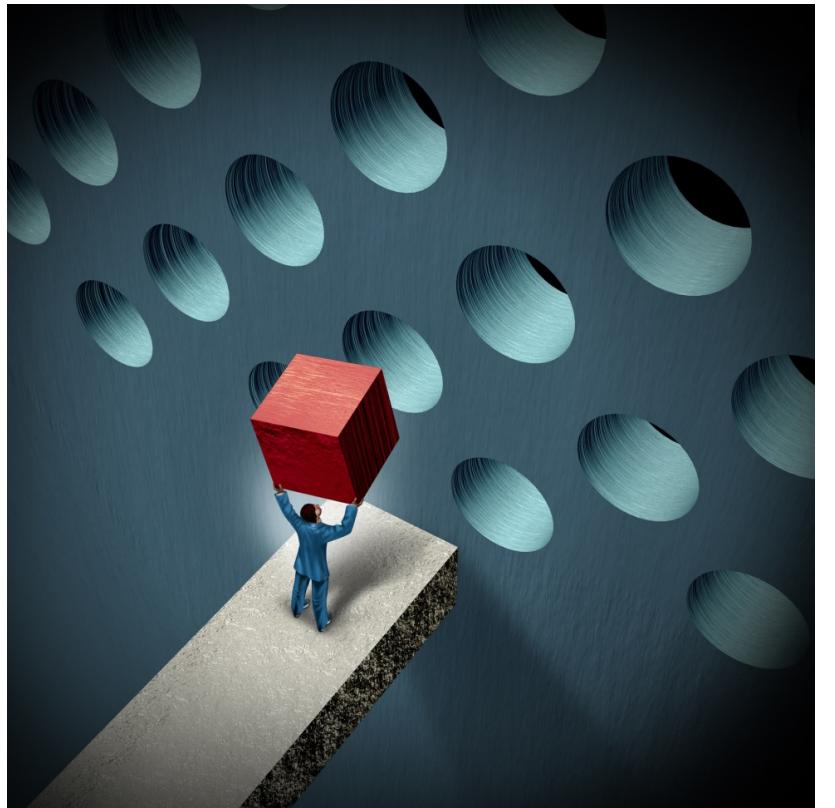
# Amazon ML: Promises & Limitations

- No downtime
- Up to 100GB training data  
(more via support ticket)
- Up to 5 simultaneous jobs  
(more via support ticket)



# Amazon ML: Anti-Patterns

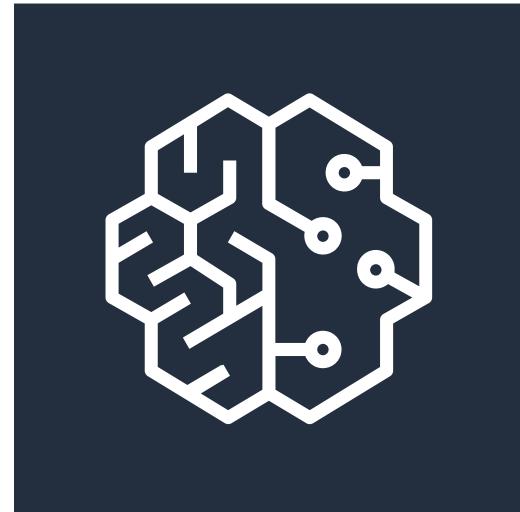
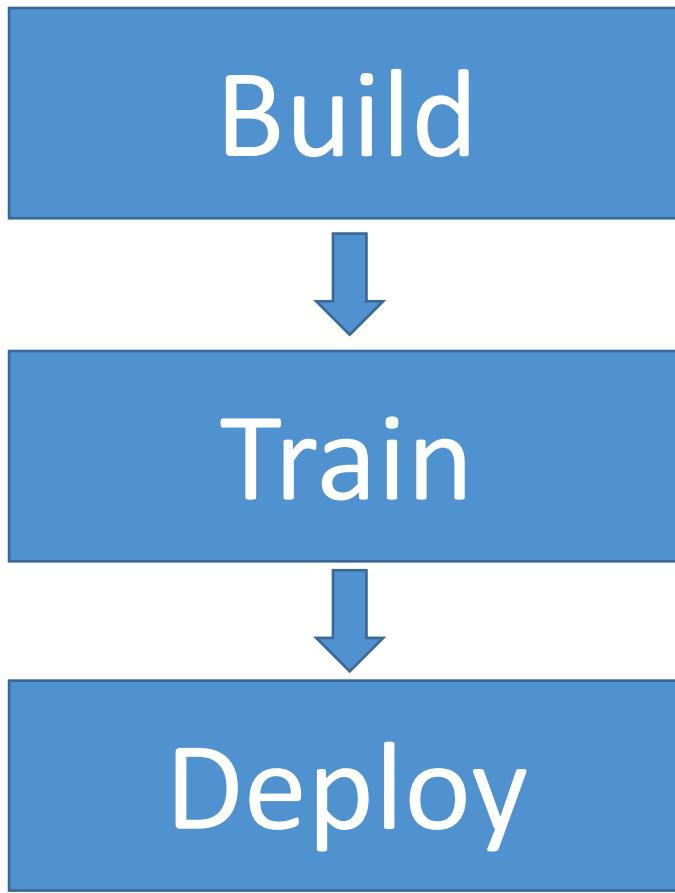
- Terabyte-scale data
- Unsupported learning tasks
  - Sequence prediction
  - Unsupervised clustering
  - Deep learning
- EMR / Spark is an (unmanaged) alternative.



# Amazon SageMaker

Scalable, fully-managed machine learning

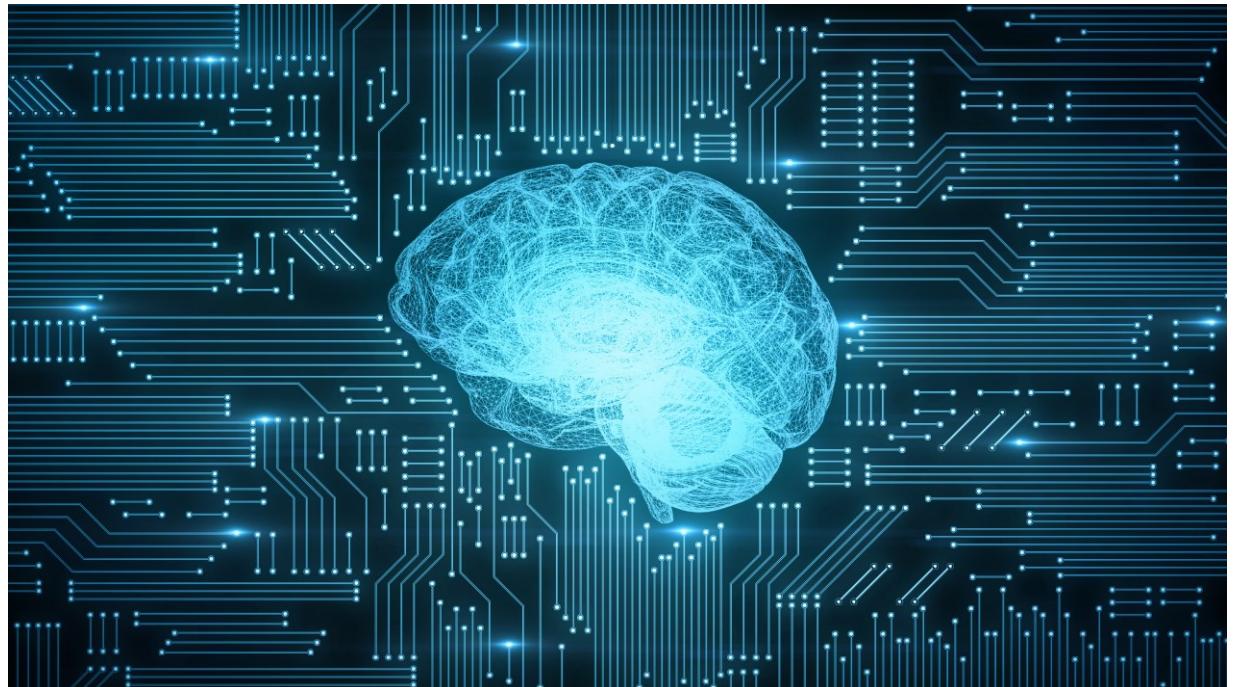
# SageMaker modules



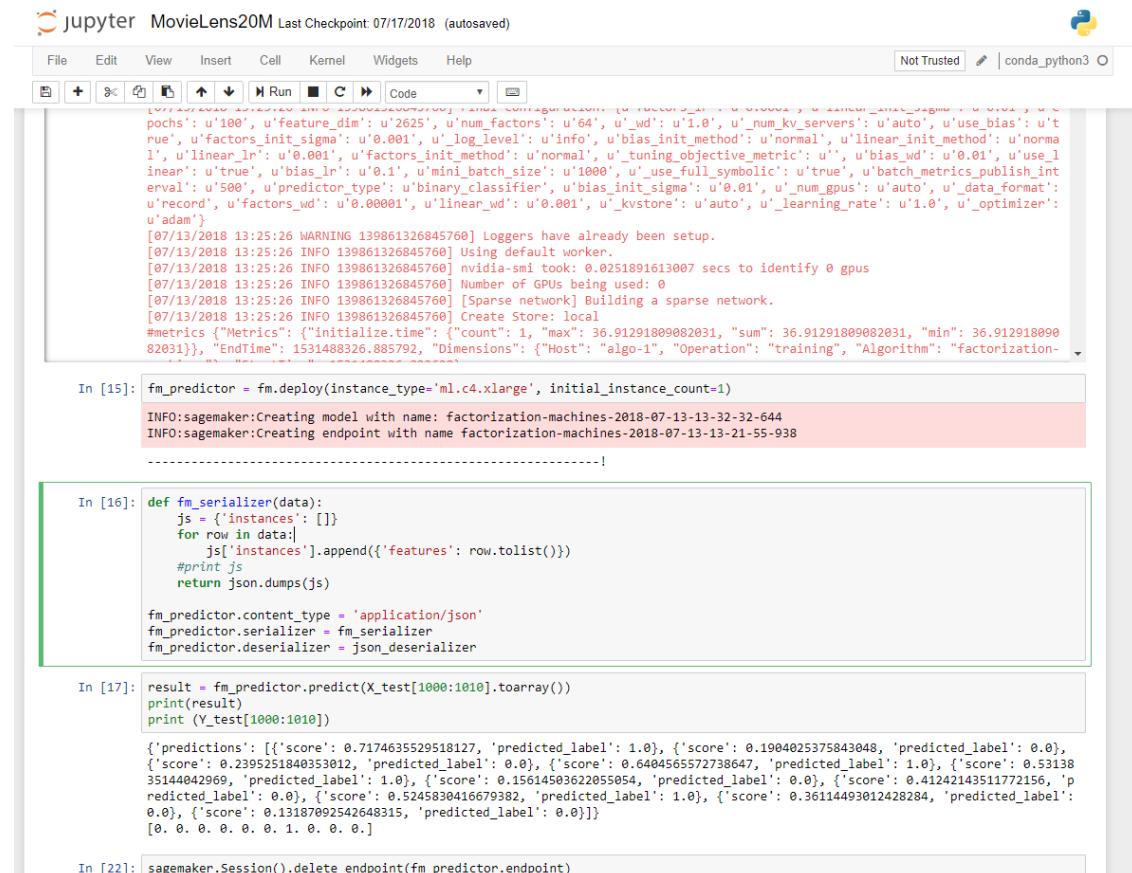
Amazon  
SageMaker

# SageMaker is powerful

- Tensorflow
- Apache MXNet
- GPU accelerated deep learning
- Scaling effectively unlimited
- Hyperparameter tuning jobs



# Jupyter Notebooks



The screenshot shows a Jupyter Notebook window with the title "jupyter MovieLens20M Last Checkpoint: 07/17/2018 (autosaved)". The notebook has tabs for File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar above the cells includes icons for file operations, run, and code. The status bar indicates "Not Trusted" and "conda\_python3".

Cell In [15] contains the following Python code:

```
fm_predictor = fm.deploy(instance_type='ml.c4.xlarge', initial_instance_count=1)
```

Output for In [15] shows INFO messages from sagemaker:

```
INFO:sagemaker:Creating model with name: factorization-machines-2018-07-13-13-32-644
INFO:sagemaker:Creating endpoint with name factorization-machines-2018-07-13-13-21-55-938
```

Cell In [16] contains the following Python code:

```
def fm_serializer(data):
    js = {'instances': []}
    for row in data:
        js['instances'].append({'features': row.tolist()})
    #print js
    return json.dumps(js)

fm_predictor.content_type = 'application/json'
fm_predictor.serializer = fm_serializer
fm_predictor.deserializer = json_deserializer
```

Cell In [17] contains the following Python code:

```
result = fm_predictor.predict(X_test[1000:1010].toarray())
print(result)
print(Y_test[1000:1010])
```

Output for In [17] shows the predicted results and actual labels for the first 10 rows of the test set.

Cell In [22] contains the following Python code:

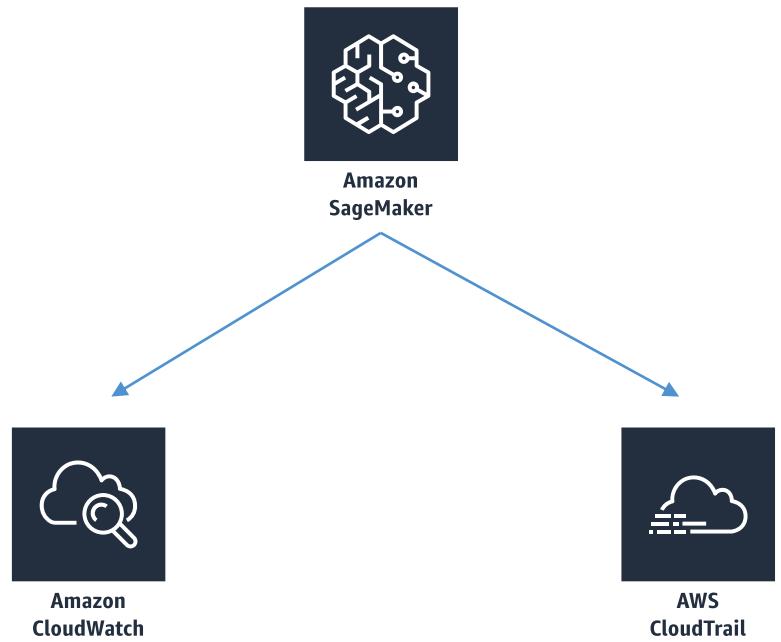
```
sagemaker.Session().delete_endpoint(fm_predictor.endpoint)
```

# SageMaker Security

- Code stored in “ML storage volumes”
  - Controlled by security groups
  - Optionally encrypted at rest
- All artifacts encrypted in transit and at rest
- API & console secured by SSL
- IAM roles
- Encrypted S3 buckets for data
- KMS integration for SageMaker notebooks, training jobs, endpoints



# SageMaker Operations



# Deep Learning 101

## And AWS Best Practices

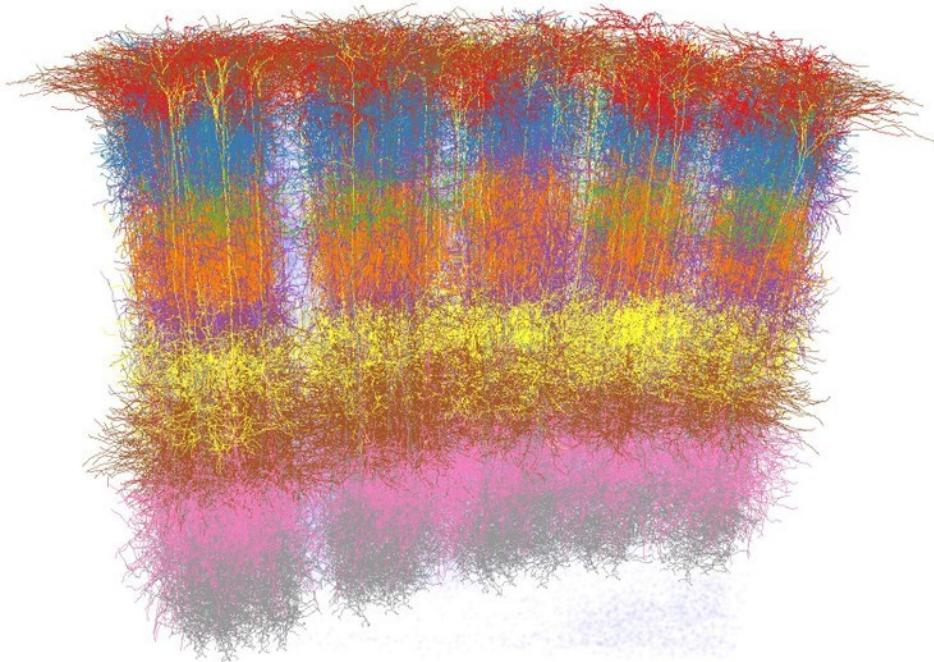
# The biological inspiration

- Neurons in your cerebral cortex are connected via axons
- A neuron “fires” to the neurons it’s connected to, when enough of its input signals are activated.
- Very simple at the individual neuron level – but layers of neurons connected in this way can yield learning behavior.
- Billions of neurons, each with thousands of connections, yields a mind



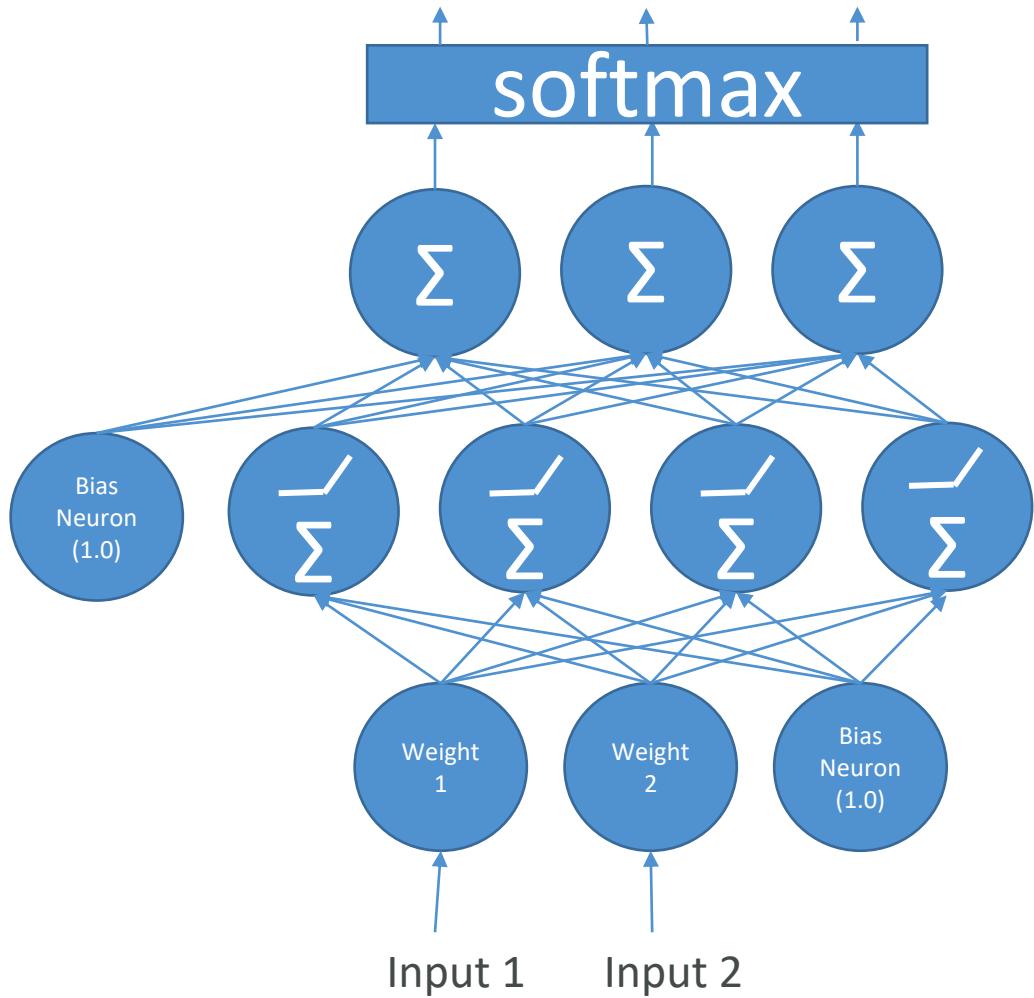
# Cortical columns

- Neurons in your cortex seem to be arranged into many stacks, or “columns” that process information in parallel
- “mini-columns” of around 100 neurons are organized into larger “hyper-columns”. There are 100 million mini-columns in your cortex
- This is coincidentally similar to how GPU’s work...



*(credit: Marcel Oberlaender et al.)*

# Deep Neural Networks



# Deep Learning Frameworks

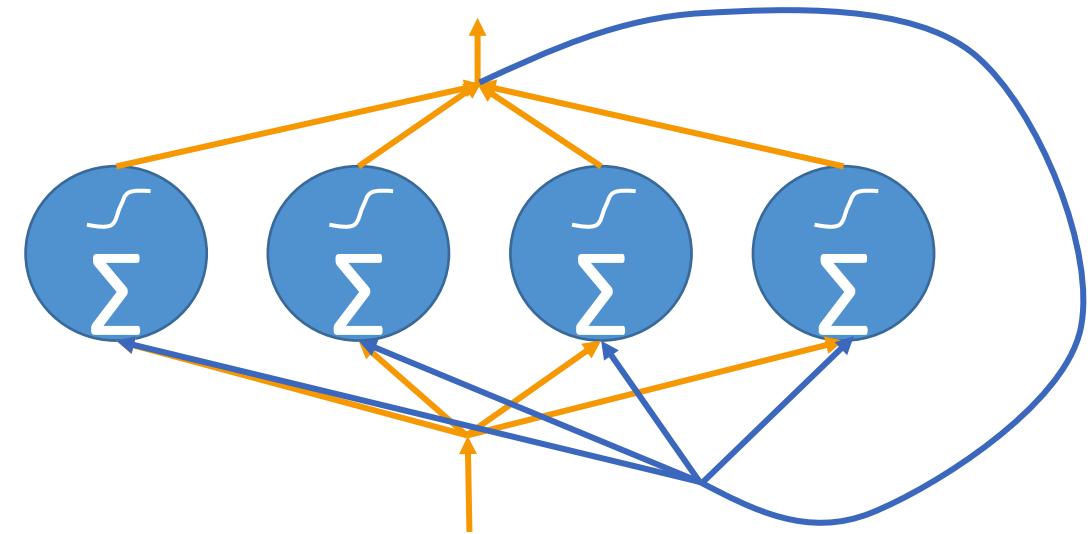
- Tensorflow / Keras
- MXNet

```
model = Sequential()

model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9,
           nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd, metrics=['accuracy'])
```

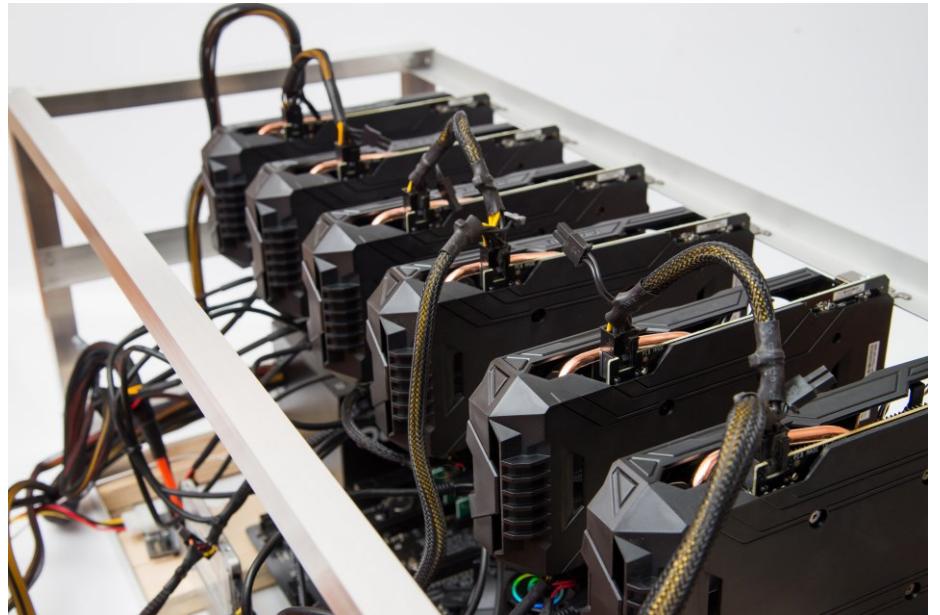
# Types of Neural Networks

- Feedforward Neural Network
- Convolutional Neural Networks (CNN)
  - Image classification (is there a stop sign in this image?)
- Recurrent Neural Networks (RNNs)
  - Deals with sequences in time (predict stock prices, understand words in a sentence, etc)
  - **LSTM, GRU**



# Deep Learning on EC2 / EMR

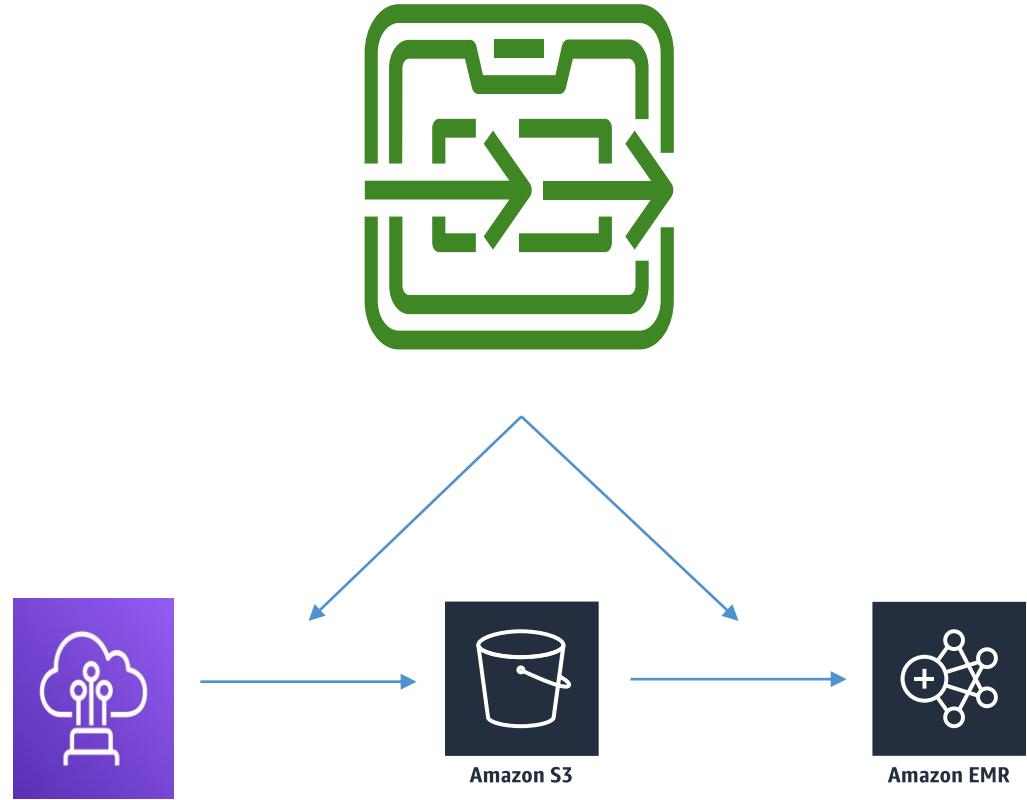
- EMR supports Apache MXNet and GPU instance types
- Appropriate instance types for deep learning:
  - P3: 8 Tesla V100 GPU's
  - P2: 16 K80 GPU's
  - G3: 4 M60 GPU's (all Nvidia chips)
- Deep Learning AMI's



# AWS Data Pipeline

A high-level overview

# Data Pipeline example



# Data Pipeline Features

- Destinations include S3, RDS, DynamoDB, Redshift and EMR
- Manages task dependencies
- Retries and notifies on failures
- Cross-region pipelines
- Precondition checks
- Data sources may be on-premises
- Highly available



# Data Pipeline Activities

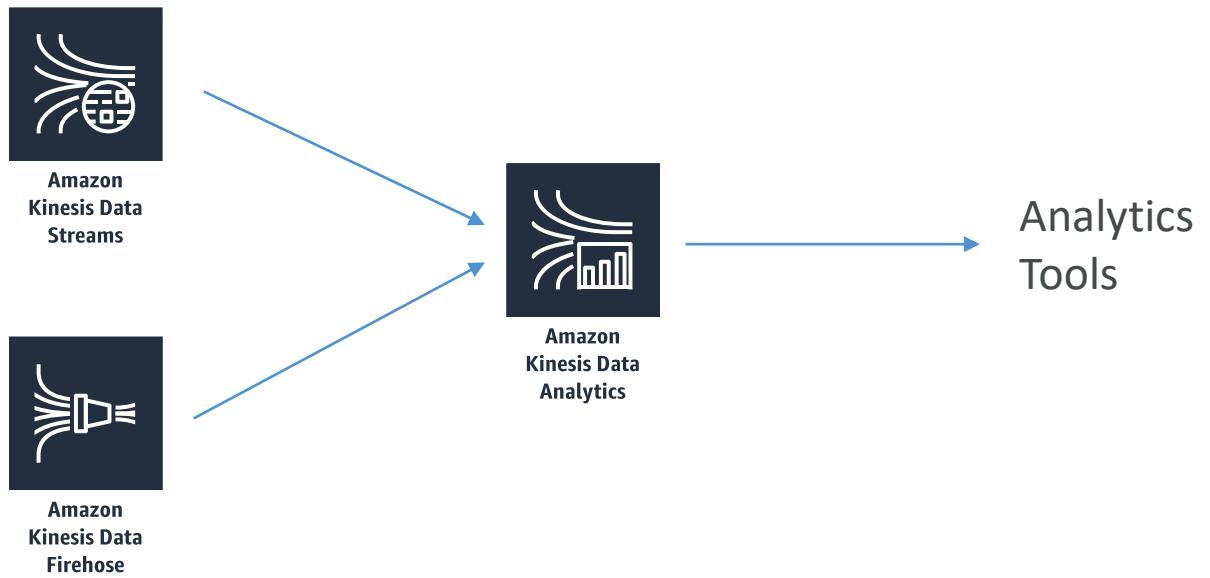
- EMR
- Hive
- Copy
- SQL
- Scripts



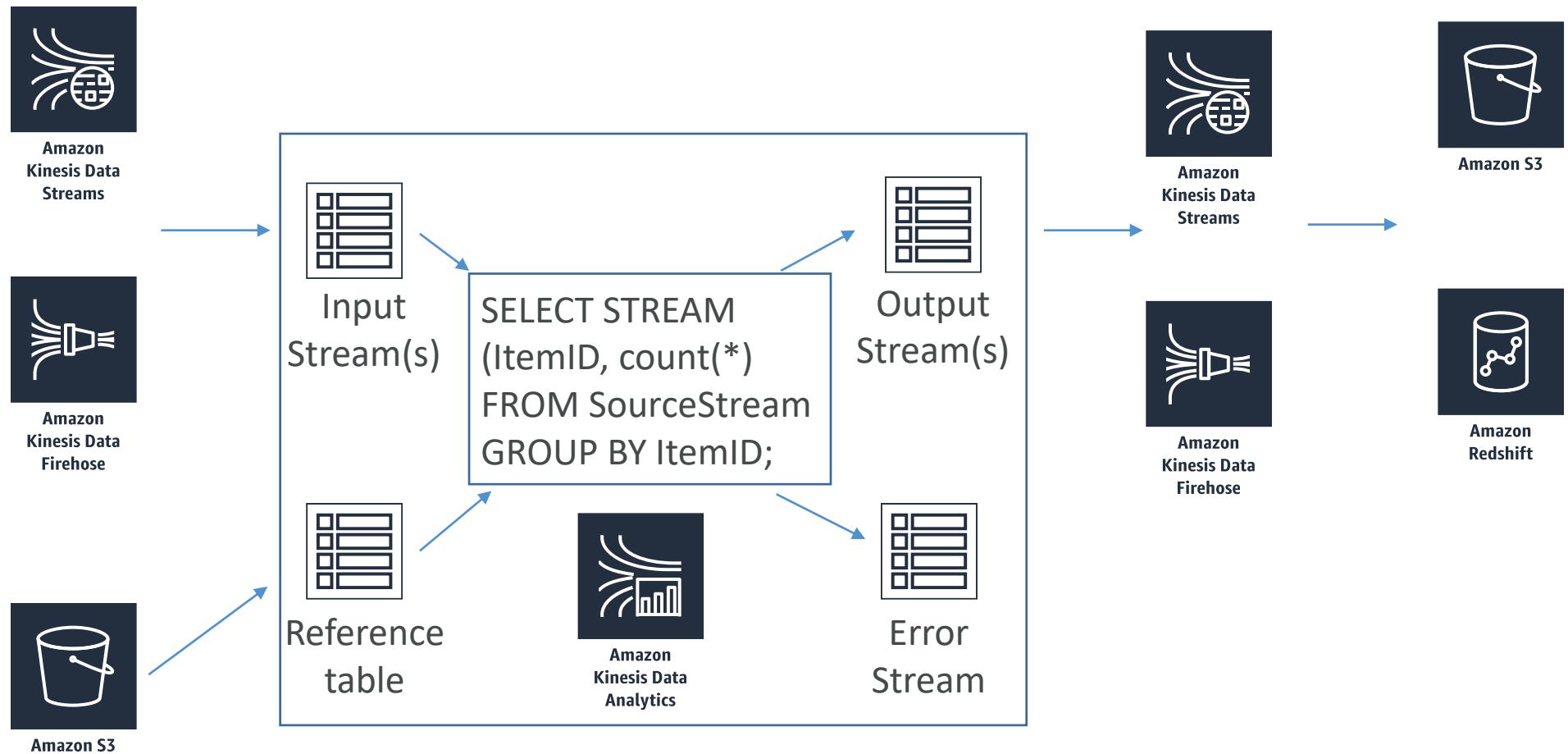
# Kinesis Data Analytics

Querying streams of data

# Conceptually...

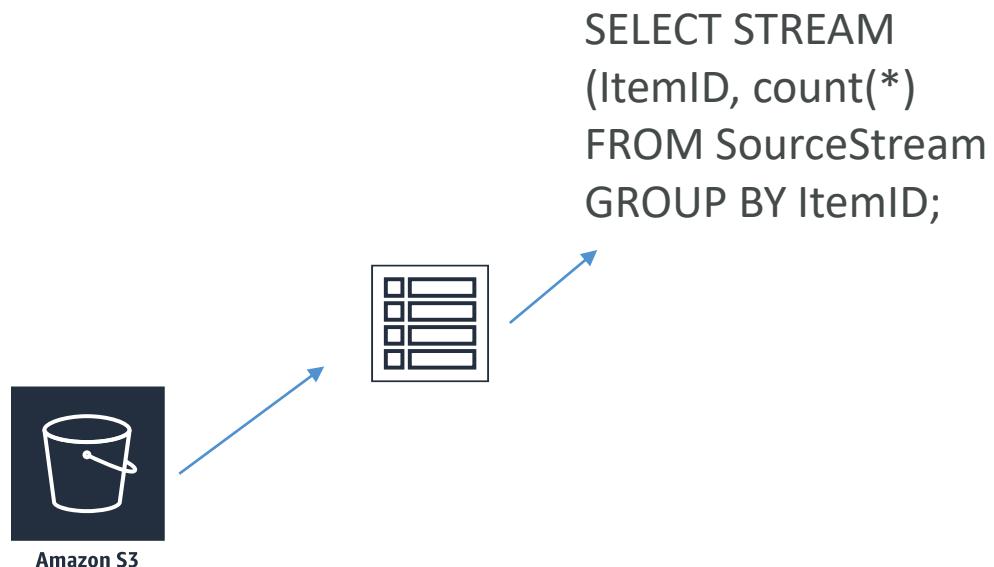


# In more depth...



# Reference tables are cool

- Inexpensive way to “join” data for quick lookups
  - i.e., look up the city associated with a zip code
  - Mapping is stored in S3 which is very inexpensive
  - Just use a “JOIN” command to use the data in your queries

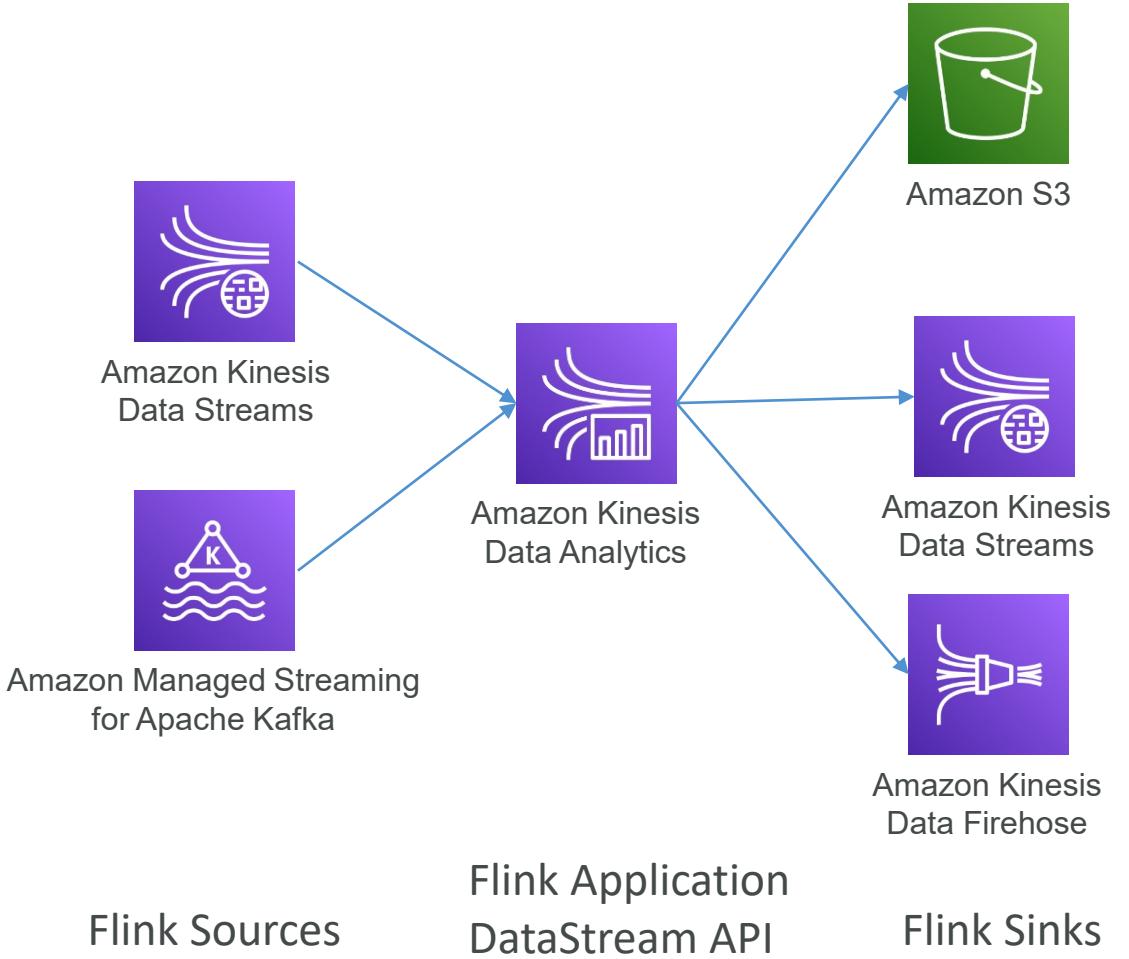


# Kinesis Data Analytics + Lambda

- AWS Lambda can be a destination as well
- Allows lots of flexibility for post-processing
  - Aggregating rows
  - Translating to different formats
  - Transforming and enriching data
  - Encryption
- Opens up access to other services & destinations
  - S3, DynamoDB, Aurora, Redshift, SNS, SQS, CloudWatch

# Kinesis Data Analytics for Apache Flink

- Formerly Kinesis Data Analytics for Java
  - Kinesis Data Analytics always used Flink under the hood
  - But now supports Scala as well as Java
- Flink is a framework for processing data streams
- Kinesis Data Analytics integrates Flink with AWS
  - Instead of using SQL, you can develop your own Flink application from scratch and load it into KDA via S3
- Serverless



# Common use-cases

- Streaming ETL
- Continuous metric generation
- Responsive analytics



**Amazon  
Kinesis Data  
Analytics**

# Kinesis Analytics

- Pay only for resources consumed (but it's not cheap)
  - Charged by Kinesis Processing Units (KPU's) consumed per hour
  - 1 KPU = 1 vCPU + 4GB
- Serverless; scales automatically
- Use IAM permissions to access streaming source and destination(s)
- Schema discovery

# RANDOM\_CUT\_FOREST

- SQL function used for anomaly detection on numeric columns in a stream
- They're especially proud of this because they published a paper on it
- It's a novel way to identify outliers in a data set so you can handle them however you need to
- Example: detect anomalous subway ridership during the NYC marathon

---

## Robust Random Cut Forest Based Anomaly Detection On Streams

---

Sudipto Guha  
University of Pennsylvania, Philadelphia, PA 19104.

SUDIPTO@CIS.UPENN.EDU  
NMISHRA@AMAZON.COM

Nina Mishra  
Amazon, Palo Alto, CA 94303.

GOURAVR@AMAZON.COM

Gourav Roy  
Amazon, Bangalore, India 560055.

OKKES@CS.STANFORD.EDU

Okke Schrijvers  
Stanford University, Palo Alto, CA 94305.

### Abstract

In this paper we focus on the anomaly detection problem for dynamic data streams through the lens of random cut forests. We investigate a robust random cut data structure that can be used as a sketch or synopsis of the input stream. We provide a plausible definition of non-parametric anomalies based on the influence of an unseen point on the remainder of the data, i.e., the externalities of the data points. We show how the sketch can be efficiently updated in a dynamic data stream. We demonstrate the viability of the algorithm on publicly available real data.

### 1. Introduction

Anomaly detection is one of the cornerstone problems in data mining. Even though the problem has been well studied over the last few decades, the emerging explosion of data from the internet of things and sensors lead us to reconsider the problem. In most of these contexts, data is streaming and well-understood prior models do not exist. Furthermore the input streams need not be append-only; there may be corrections, updates and a variety of other dynamic changes. Two central questions in this regard are (1) how do we define anomalies? and (2) what data struc-

a point is data dependent and corresponds to the externalities induced by the point in according to the model of the data. We extend this notion of externality to handle “outlier masking” that often arises from duplicates and near duplicate records. Note that the notion of model complexity has to be amenable to efficient computation in dynamic data streams. This relates question (1) to question (2) which we discuss in greater detail next. However it is worth noting that anomaly detection is not well understood even in the simpler context of static batch processing and (2) remains relevant in the batch setting as well.

For question (2), we explore a randomized approach, akin to (Liu et al., 2012), due in part to the practical success reported in (Emmott et al., 2013). Randomization is a powerful tool and known to be valuable in supervised learning (Breiman, 2001). But its technical exploration in the context of anomaly detection is not well-understood and the same comment applies to the algorithm put forth in (Liu et al., 2012). Moreover that algorithm has several limitations as described in Section 4.1. In particular, we show that the presence of important dimensions, crucial for outlier detection, makes it difficult to extend the algorithm to a stream. Prior work attempted to extend this work to a stream. Prior work attempted to solutions (Tan et al., 2011) that extend to streaming, however those were not found to be effective (Emmott et al., 2013). To address these limitations, we put forward a sketch or synopsis termed *robust random cut forest* (RRCF) formally

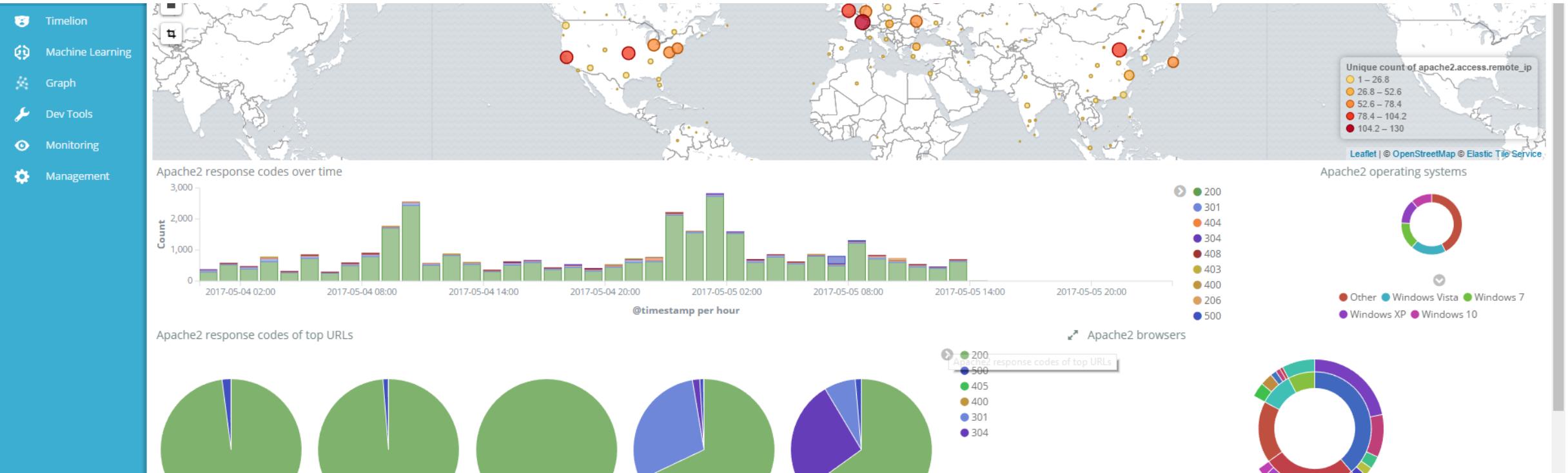
# Amazon OpenSearch Service (formerly Elasticsearch)

Petabyte-scale analysis and reporting

# What is OpenSearch?

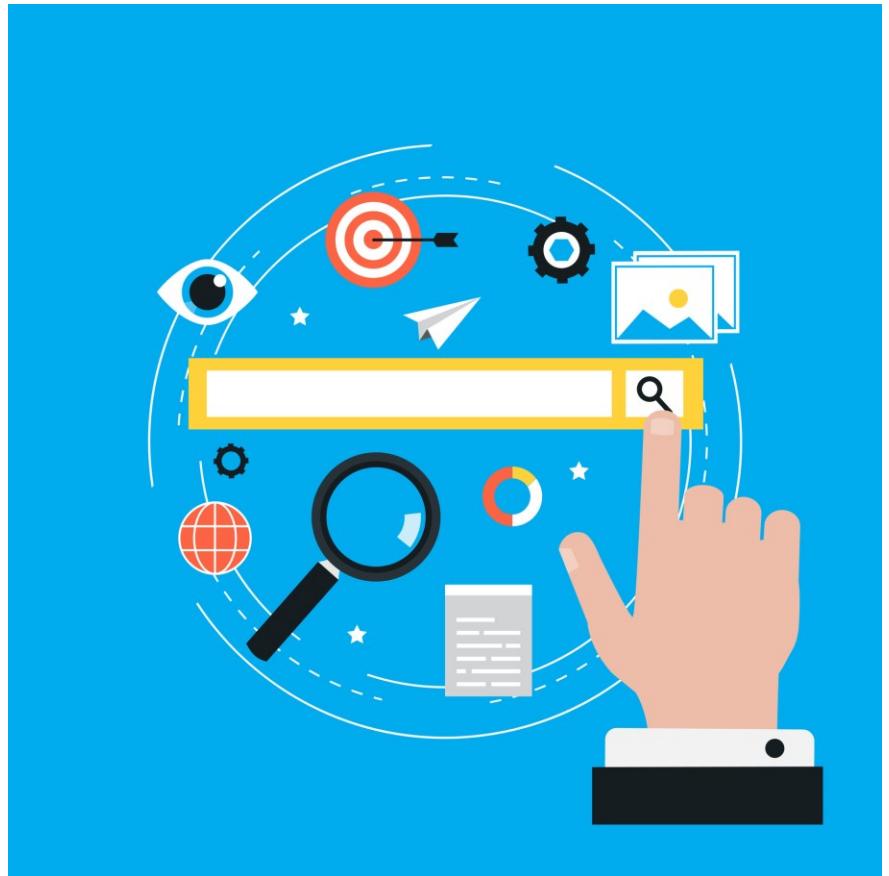
- A fork of Elasticsearch and Kibana
- A search engine
- An analysis tool
- A visualization tool (Dashboards = Kibana)
- A data pipeline
  - Kinesis replaces Beats & Logstash
- Horizontally scalable

# What are Dashboards?



# Opensearch applications

- Full-text search
- Log analytics
- Application monitoring
- Security analytics
- Clickstream analytics



# Opensearch concepts



## documents

Documents are the things you're searching for. They can be more than text – any structured JSON data works. Every document has a unique ID, and a type.



## types

A type defines the schema and mapping shared by documents that represent the same sort of thing. (A log entry, an encyclopedia article, etc.)

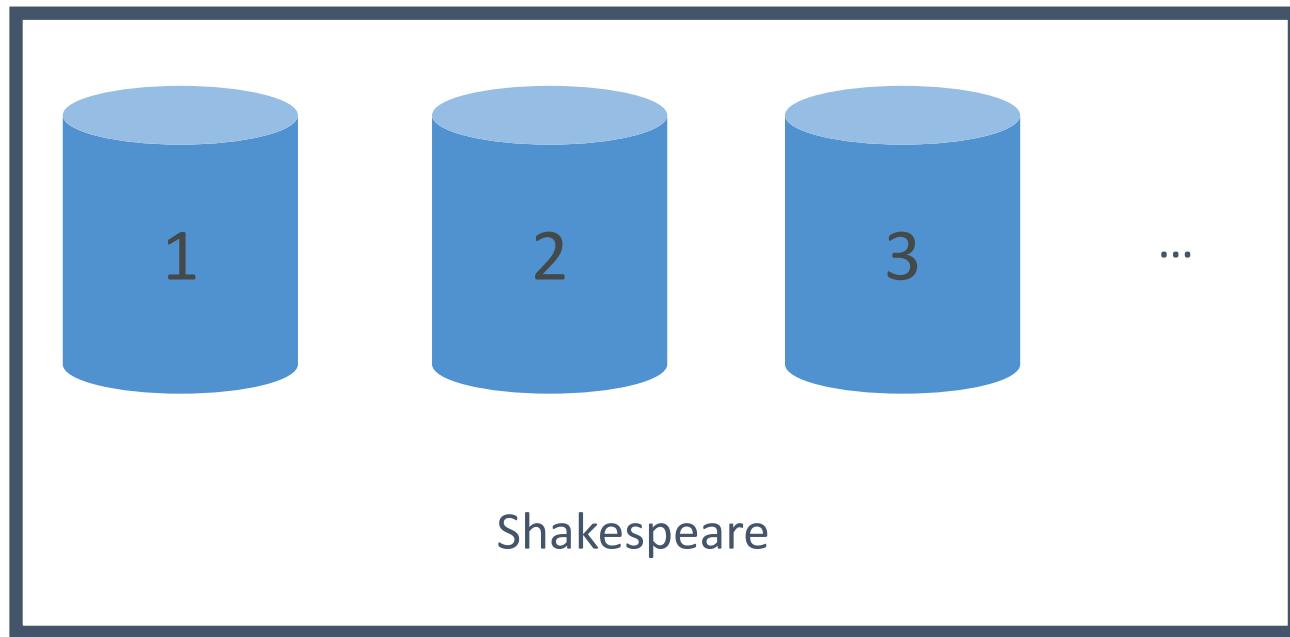


## indices

An index powers search into all documents within a collection of types. They contain inverted indices that let you search across everything within them at once.

# An index is split into shards

Documents are **hashed** to a particular shard.



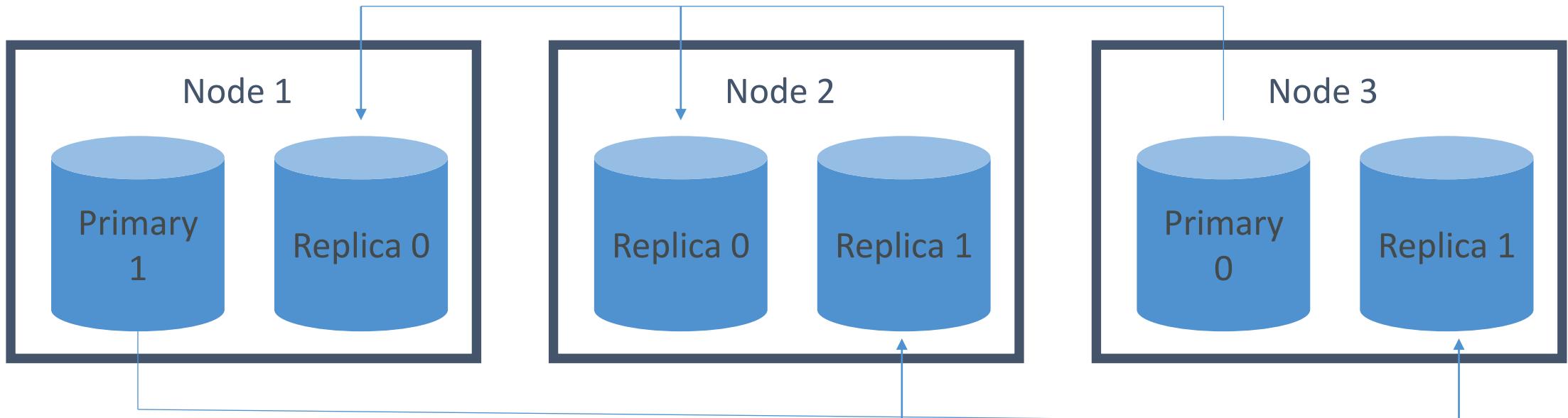
Each shard may be on a different **node** in a **cluster**.

Every shard is a self-contained Lucene index of its own.

# Redundancy

This index has two primary shards and two replicas.

Your application should round-robin requests amongst nodes.



Write requests are routed to the primary shard, then replicated

Read requests are routed to the primary or any replica

# Amazon Opensearch Service

- Fully-managed (but not serverless)
- Scale up or down without downtime
  - But this isn't automatic
- Pay for what you use
  - Instance-hours, storage, data transfer
- Network isolation
- AWS integration
  - S3 buckets (via Lambda to Kinesis)
  - Kinesis Data Streams
  - DynamoDB Streams
  - CloudWatch / CloudTrail
  - Zone awareness

# Amazon Opensearch options

- Dedicated master node(s)
  - Choice of count and instance types
- “Domains”
- Snapshots to S3
- Zone Awareness

# Cold / warm / ultrawarm / hot storage

- Standard data nodes use “hot” storage
  - Instance stores or EBS volumes / fastest performance
- UltraWarm (warm) storage uses S3 + caching
  - Best for indices with few writes (like log data / immutable data)
  - Slower performance but much lower cost
  - Must have a dedicated master node
- Cold storage
  - Also uses S3
  - Even cheaper
  - For “periodic research or forensic analysis on older data”
  - Must have dedicated master and have UltraWarm enabled too.
  - Not compatible with T2 or T3 instance types on data nodes
  - If using fine-grained access control, must map users to cold\_manager role in OpenSearch Dashboards
- Data may be migrated between different storage types



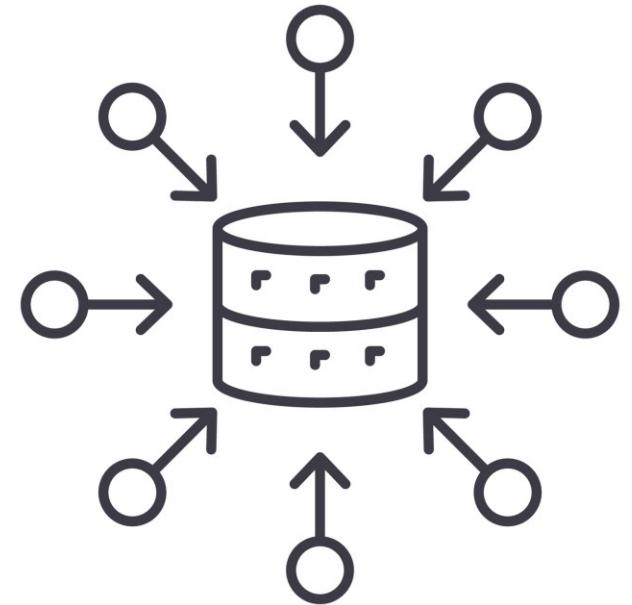
# Index State Management

- Automates index management policies
- Examples
  - Delete old indices after a period of time
  - Move indices into read only state after a period of time
  - Move indices from hot -> UltraWarm -> cold storage over time
  - Reduce replica count over time
  - Automate index snapshots
- ISM policies are run every 30-48 minutes
  - Random jitter to ensure they don't all run at once
- Can even send notifications when done



# More Index Management

- Index rollups
  - Periodically roll up old data into summarized indices
  - Saves storage costs
  - New index may have fewer fields, coarser time buckets
- Index transforms
  - Like rollups, but purpose is to create a different view to analyze data differently.
  - Groupings and aggregations



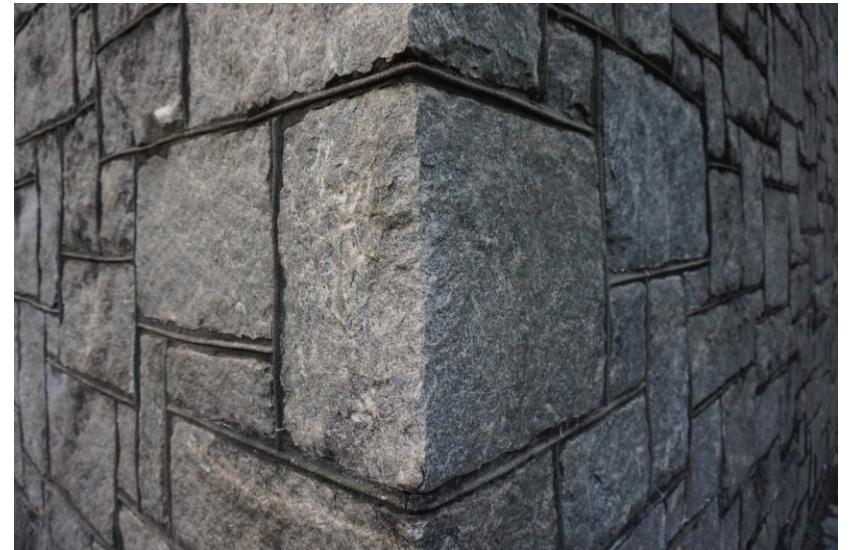
# Cross-cluster replication

- Replicate indices / mappings / metadata across domains
- Ensures high availability in an outage
- Replicate data geographically for better latency
- “Follower” index pulls data from “leader” index
- Requires fine-grained access control and node-to-node encryption
- “Remote Reindex” allows copying indices from one cluster to another on demand



# Opensearch Stability

- 3 dedicated master nodes is best
  - Avoids “split brain”
- Don’t run out of disk space
  - Minimum storage requirement is roughly:  
Source Data \* (1 + Number of Replicas) \* 1.45
- Choosing the number of shards
  - $(\text{source data} + \text{room to grow}) * (1 + \text{indexing overhead}) / \text{desired shard size}$
  - In rare cases you may need to limit the number of shards per node
    - You usually run out of disk space first.
- Choosing instance types
  - At least 3 nodes
  - Mostly about storage requirements
  - i.e., m6g.large.search, i3.4xlarge.search, i3.16xlarge.search



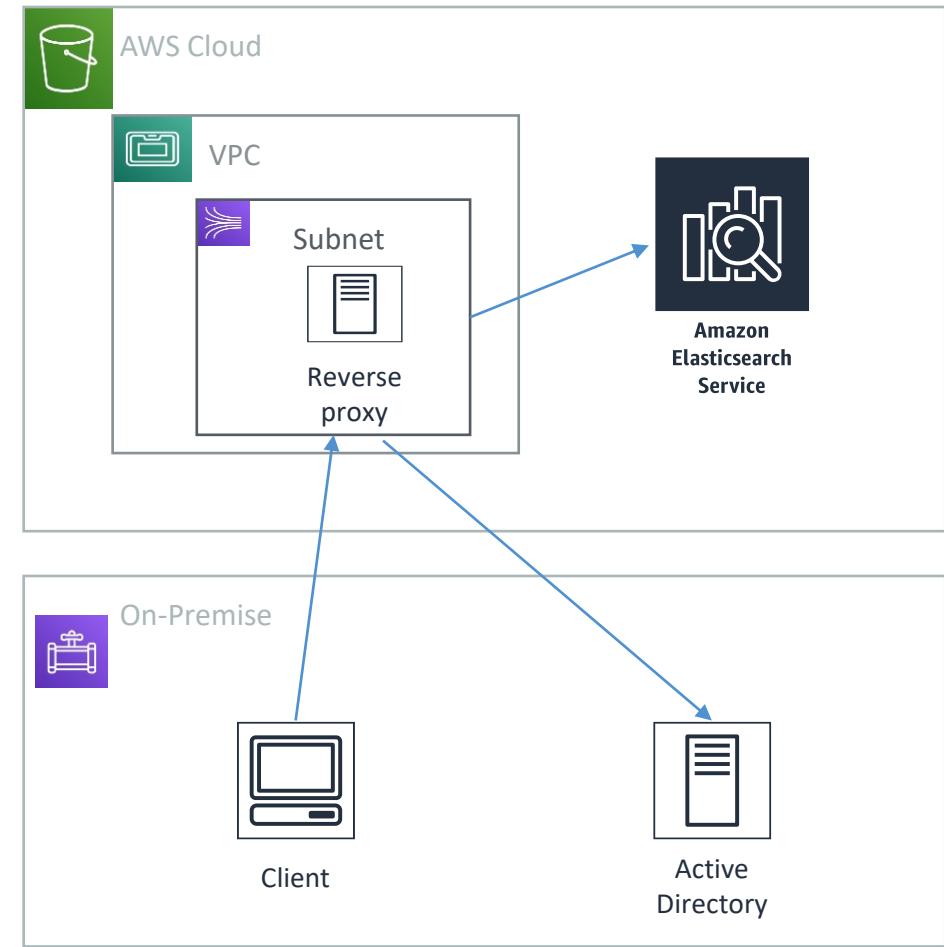
# Amazon Opensearch Security

- Resource-based policies
- Identity-based policies
- IP-based policies
- Request signing
- VPC
- Cognito



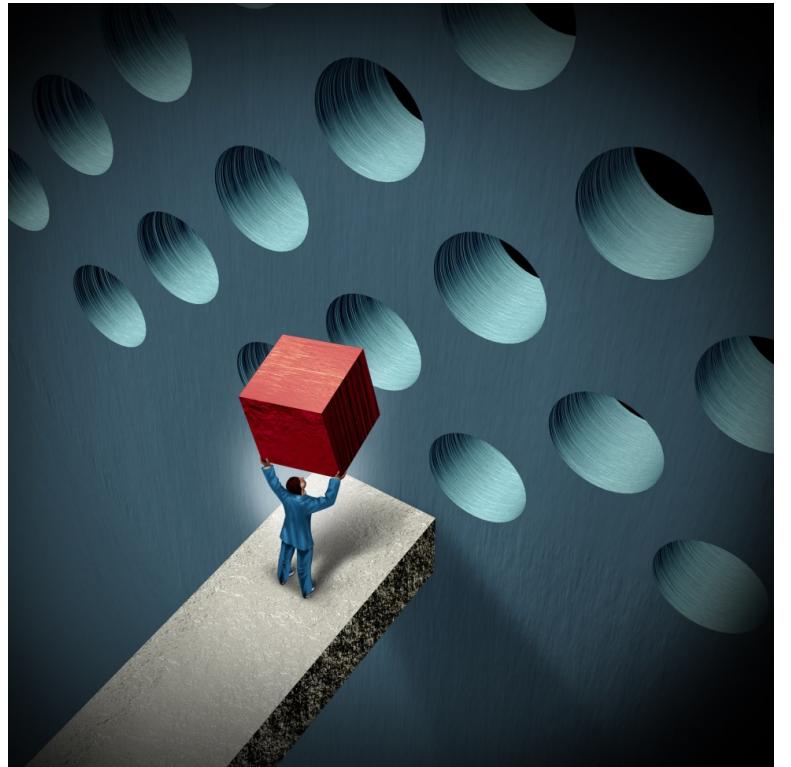
# Securing Dashboards

- Cognito
- Getting inside a VPC from outside is hard...
  - Nginx reverse proxy on EC2 forwarding to ES domain
  - SSH tunnel for port 5601
  - VPC Direct Connect
  - VPN



# Amazon Opensearch anti-patterns

- OLTP
  - No transactions
  - RDS or DynamoDB is better
- Ad-hoc data querying
  - Athena is better
- Remember Opensearch is primarily for search & analytics



# Amazon Opensearch performance

- Memory pressure in the JVM can result if:
  - You have unbalanced shard allocations across nodes
  - You have too many shards in a cluster
- Fewer shards can yield better performance if JVMMemoryPressure errors are encountered
  - Delete old or unused indices



# Amazon Athena

Serverless interactive queries of S3 data

# What is Athena?

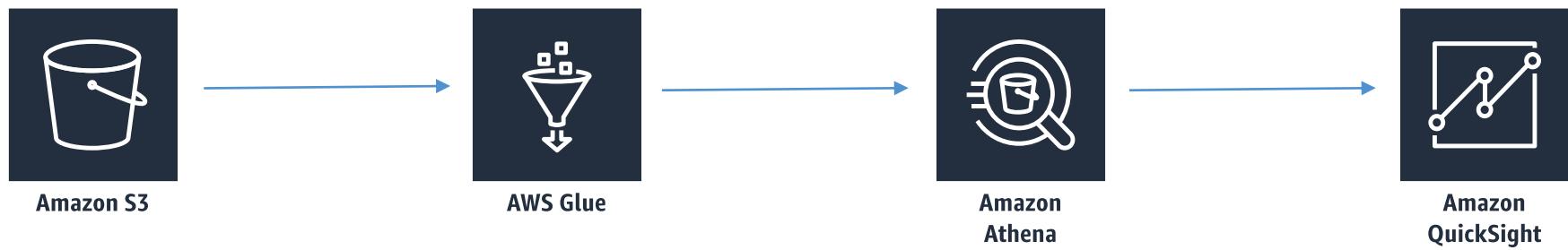
- Interactive query service for S3 (SQL)
  - No need to load data, it stays in S3
- Presto under the hood
- Serverless!
- Supports many data formats
  - CSV (human readable)
  - JSON (human readable)
  - ORC (columnar, splittable)
  - Parquet (columnar, splittable)
  - Avro (splittable)
- Unstructured, semi-structured, or structured

# Some examples

- Ad-hoc queries of web logs
- Querying staging data before loading to Redshift
- Analyze CloudTrail / CloudFront / VPC / ELB etc logs in S3
- Integration with Jupyter, Zeppelin, RStudio notebooks
- Integration with QuickSight
- Integration via ODBC / JDBC with other visualization tools

```
43 USE AdventureworksLT;
44 GO
45 SELECT p.Name AS ProductName,
46 NonDiscountSales = (OrderQty * UnitPrice)
47 Discounts = ((OrderQty * UnitPrice) * TaxRate)
48 FROM Production.Product AS p
49 INNER JOIN Sales.SalesOrderDetail sod
50 ON p.ProductID = sod.ProductID
51 ORDER BY ProductName DESC;
52 GO
```

# Athena + Glue



# Athena Workgroups

- Can organize users / teams / apps / workloads into Workgroups
- Can control query access and track costs by Workgroup
- Integrates with IAM, CloudWatch, SNS
- Each workgroup can have its own:
  - Query history
  - Data limits (*you can limit how much data queries may scan by workgroup*)
  - IAM policies
  - Encryption settings



# Athena cost model

- Pay-as-you-go
  - \$5 per TB scanned
  - Successful or cancelled queries count, failed queries do not.
  - No charge for DDL (CREATE/ALTER/DROP etc.)
- Save LOTS of money by using columnar formats
  - ORC, Parquet
  - Save 30-90%, and get better performance
- Glue and S3 have their own charges



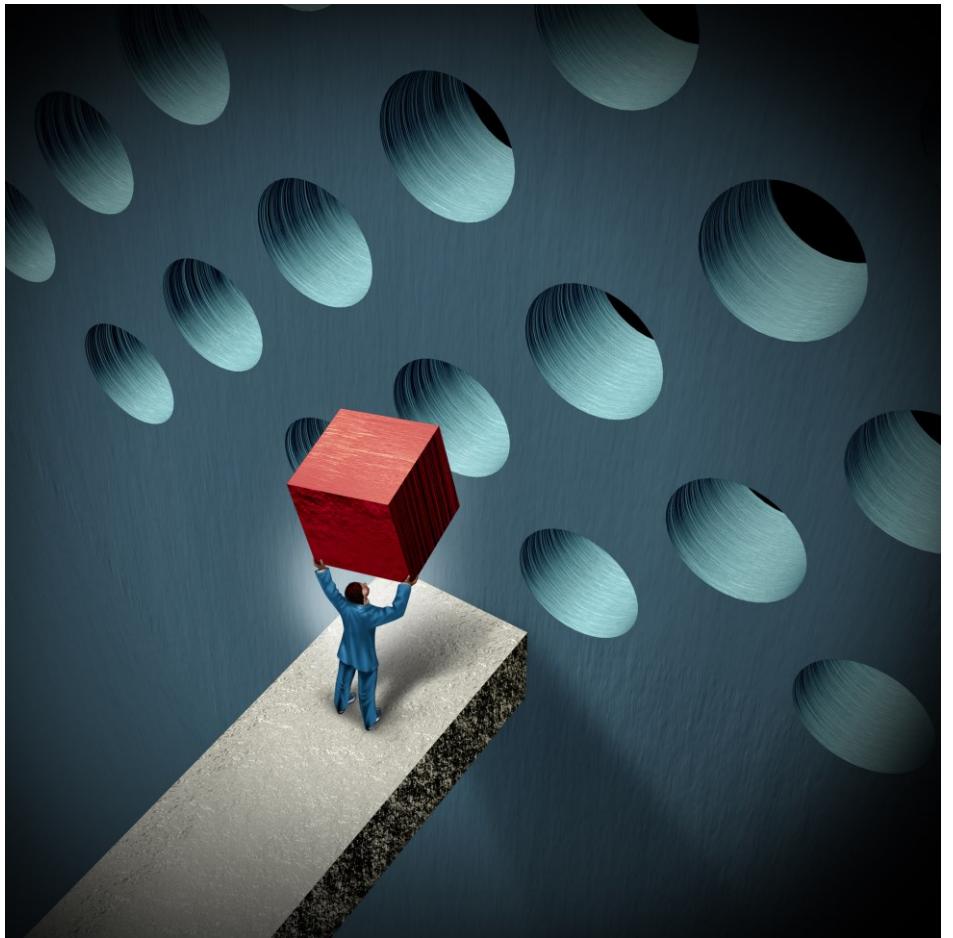
# Athena Security

- Access control
  - IAM, ACLs, S3 bucket policies
  - AmazonAthenaFullAccess / AWSQuicksightAthenaAccess
- Encrypt results at rest in S3 staging directory
  - Server-side encryption with S3-managed key (SSE-S3)
  - Server-side encryption with KMS key (SSE-KMS)
  - Client-side encryption with KMS key (CSE-KMS)
- Cross-account access in S3 bucket policy possible
- Transport Layer Security (TLS) encrypts in-transit (between Athena and S3)



# Athena anti-patterns

- Highly formatted reports / visualization
  - That's what QuickSight is for
- ETL
  - Use Glue instead



# Athena: Optimizing performance

- Use columnar data (ORC, Parquet)
- Small number of large files performs better than large number of small files
- Use partitions
  - If adding partitions after the fact, use MSCK REPAIR TABLE command



# Athena ACID transactions

- Powered by Apache Iceberg
  - Just add 'table\_type' = 'ICEBERG' in your CREATE TABLE command
- Concurrent users can safely make row-level modifications
- Compatible with EMR, Spark, anything that supports Iceberg table format.
- Removes need for custom record locking
- Time travel operations
  - Recover data recently deleted with a SELECT statement
- Remember governed tables in Lake Formation? This is another way of getting ACID features in Athena.
- Benefits from periodic compaction to preserve performance

```
OPTIMIZE table REWRITE DATA  
USING BIN_PACK  
WHERE catalog = 'c1'
```

# Amazon Redshift

Fully-managed, petabyte-scale data warehouse

# What is Redshift?

- Fully-managed, petabyte scale data warehouse service
- 10X better performance than other DW's
  - Via machine learning, massively parallel query execution, columnar storage
- Designed for OLAP, not OLTP
- Cost effective
- SQL, ODBC, JDBC interfaces
- Scale up or down on demand
- Built-in replication & backups
- Monitoring via CloudWatch / CloudTrail

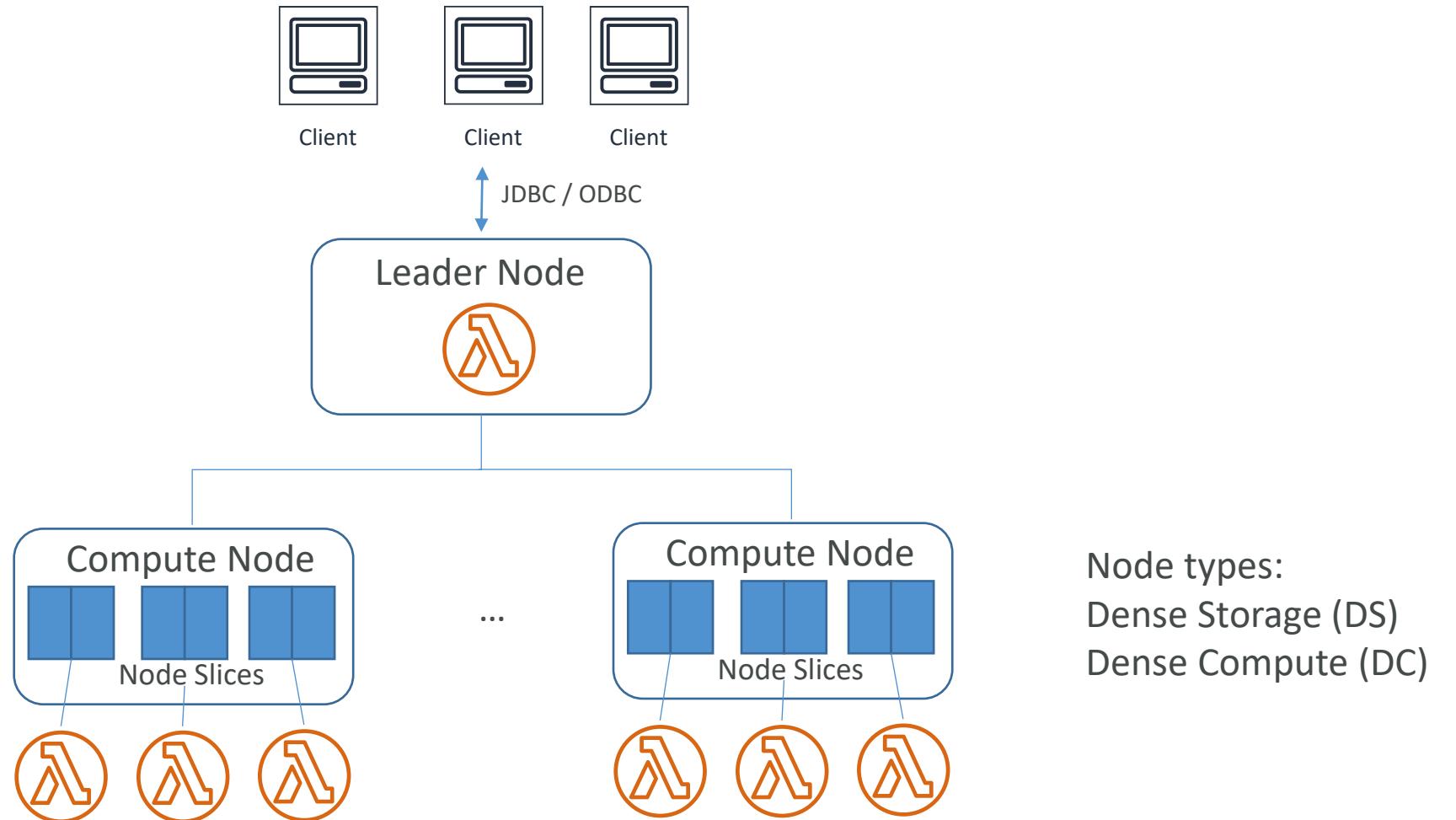


**Amazon  
Redshift**

# Redshift Use-Cases

- Accelerate analytics workloads
- Unified data warehouse & data lake
- Data warehouse modernization
- Analyze global sales data
- Store historical stock trade data
- Analyze ad impressions & clicks
- Aggregate gaming data
- Analyze social trends

# Redshift architecture



# Redshift Spectrum

- Query exabytes of unstructured data in S3 without loading
- Limitless concurrency
- Horizontal scaling
- Separate storage & compute resources
- Wide variety of data formats
- Support of Gzip and Snappy compression



Amazon  
Redshift



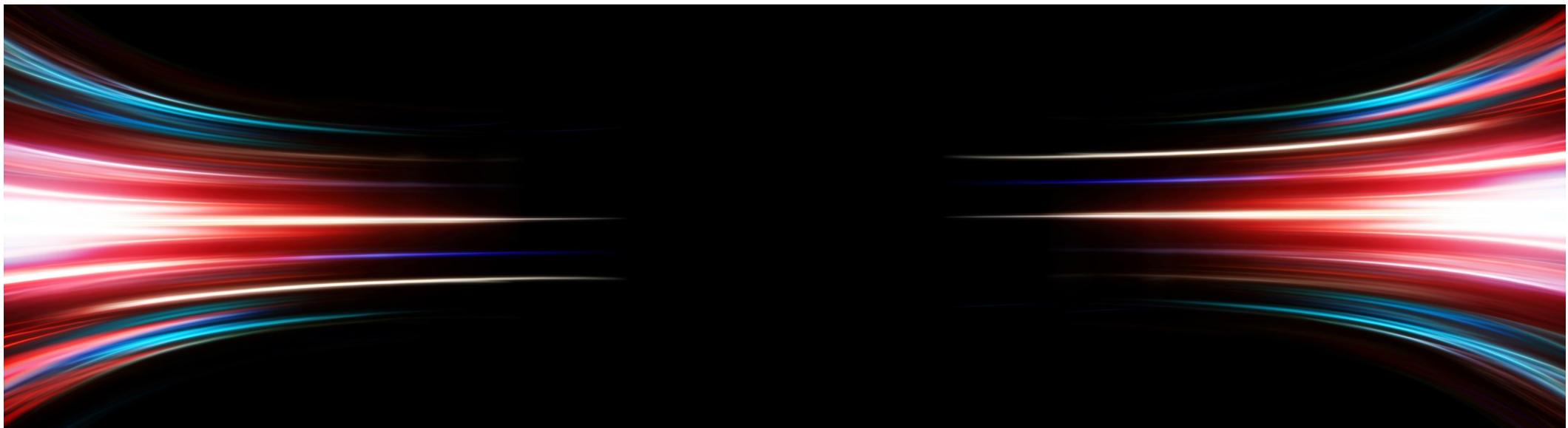
AWS Glue



Amazon S3

# Redshift Performance

- Massively Parallel Processing (MPP)
- Columnar Data Storage
- Column Compression



# Redshift Durability

- Replication within cluster
- Backup to S3
  - Asynchronously replicated to another region
- Automated snapshots
- Failed drives / nodes automatically replaced
- However – limited to a single availability zone (AZ)
  - Multi-AZ for RA3 clusters is in Preview



# Scaling Redshift

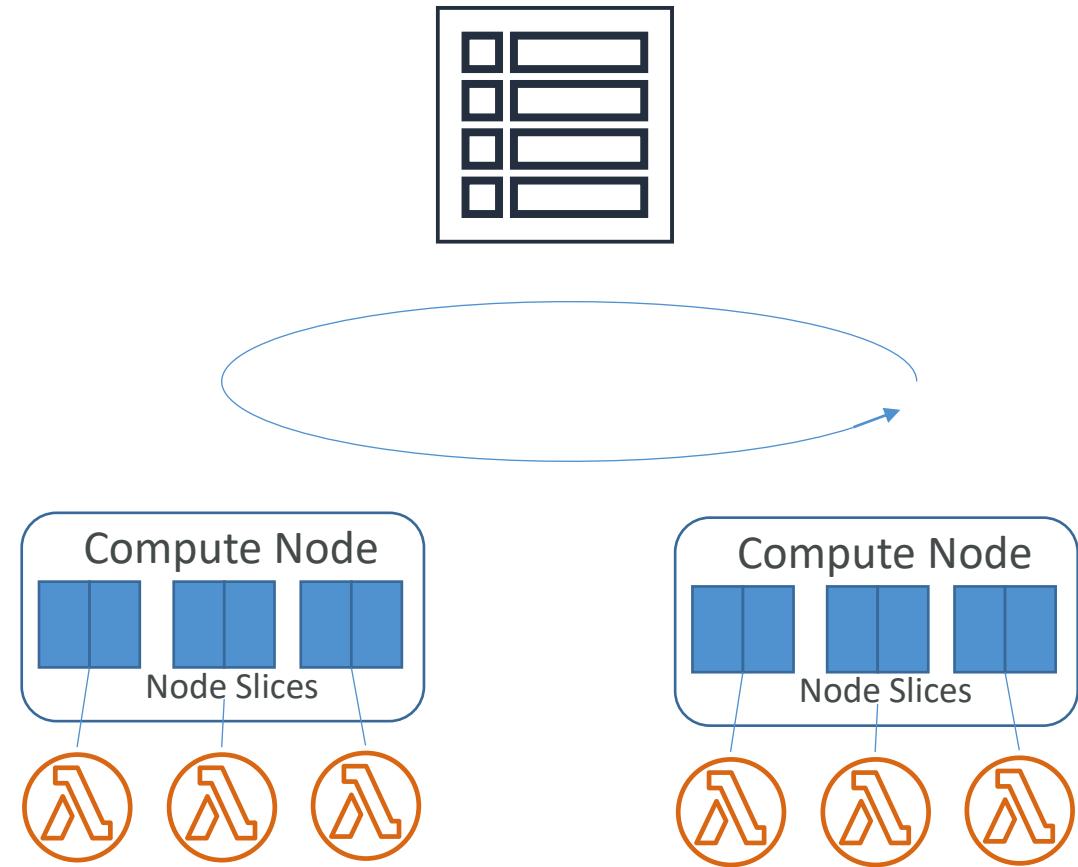
- Vertical and horizontal scaling on demand
- During scaling:
  - A new cluster is created while your old one remains available for reads
  - CNAME is flipped to new cluster (a few minutes of downtime)
  - Data moved in parallel to new compute nodes



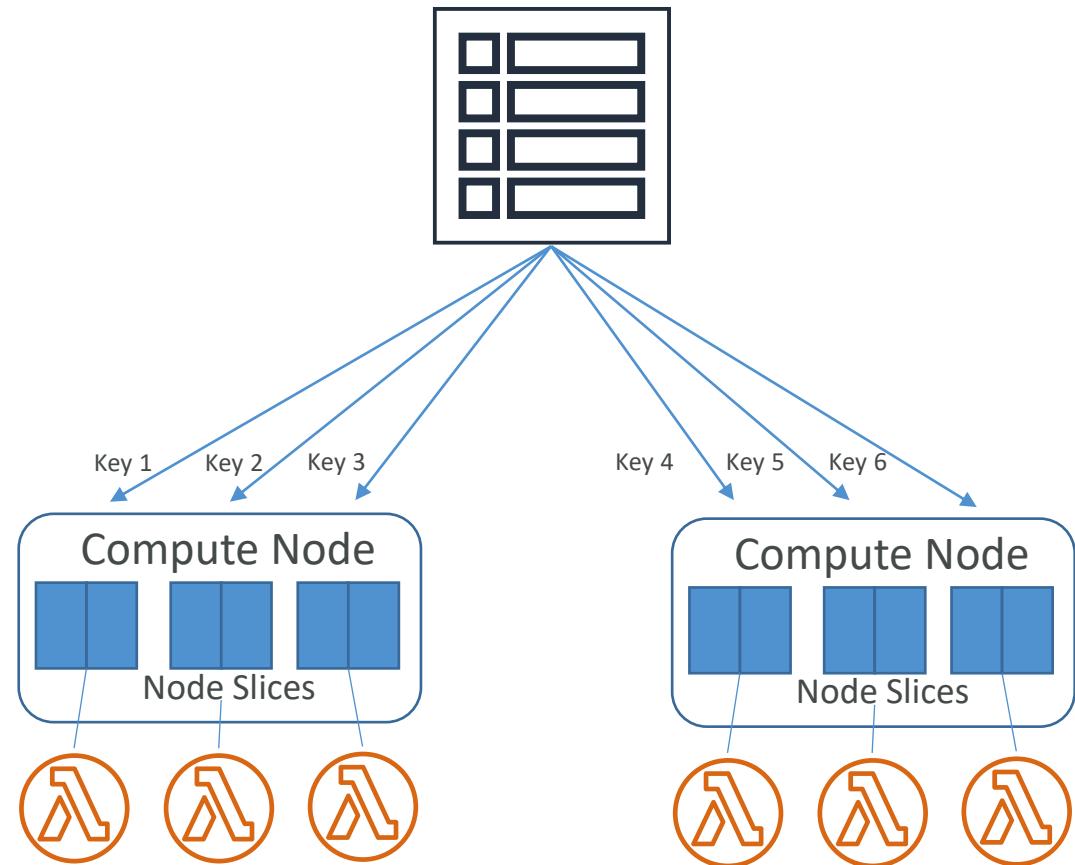
# Redshift Distribution Styles

- AUTO
  - Redshift figures it out based on size of data
- EVEN
  - Rows distributed across slices in round-robin
- KEY
  - Rows distributed based on one column
- ALL
  - Entire table is copied to every node

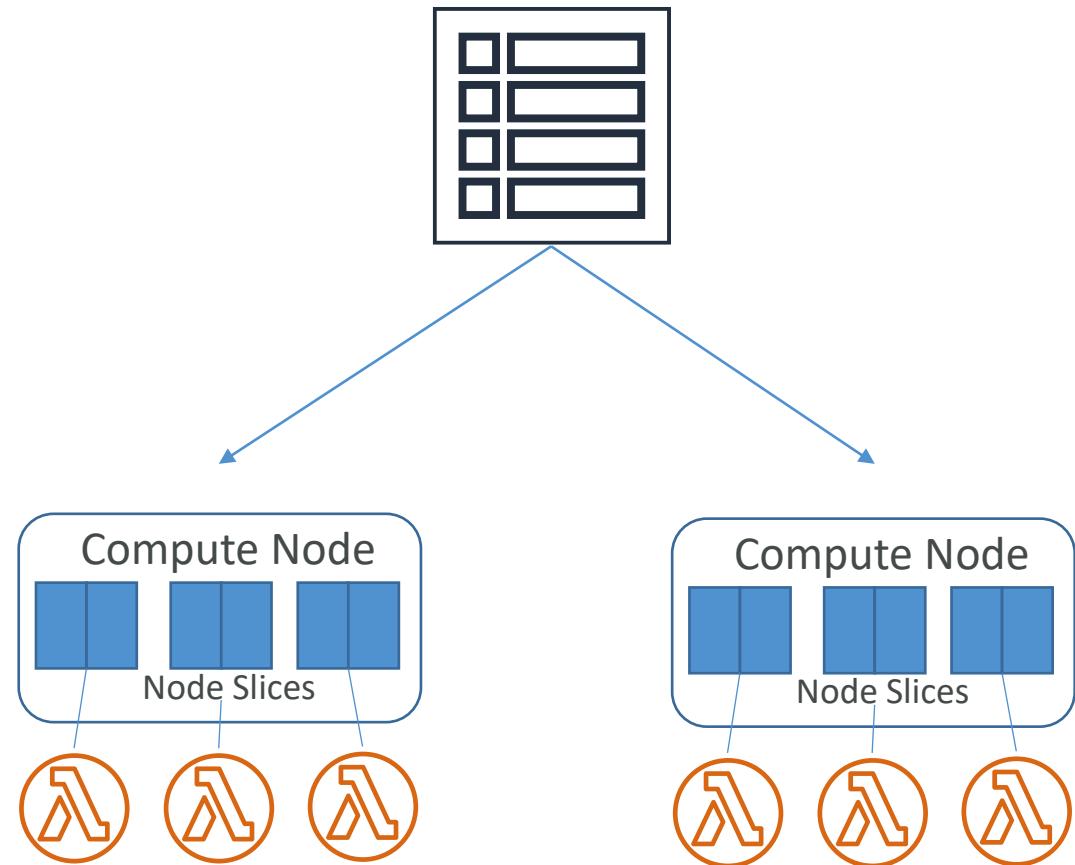
# EVEN distribution



# KEY distribution



# ALL distribution



# Redshift Sort Keys

- Rows are stored on disk in sorted order based on the column you designate as a sort key
- Like an index
- Makes for fast range queries
- Choosing a sort key
  - Recency? Filtering? Joins?
- Single vs. Compound vs Interleaved sort keys



# Sort Keys: Single Column

Date	Genre	Movie
3/18/2019	Comedy	Monty Python and the Holy Grail
3/18/2019	Adventure	Indiana Jones and the Temple of Doom
3/18/2019	Drama	Interstellar
3/18/2019	Drama	The Dark Knight
3/19/2019	Fantasy	The Lord of the Rings
3/19/2019	Drama	12 Angry Men
3/19/2019	Adventure	Inception

# Sort Keys: Compound

Date	Genre	Movie
3/18/2019	Adventure	Indiana Jones and the Temple of Doom
3/18/2019	Comedy	Monty Python and the Holy Grail
3/18/2019	Drama	Interstellar
3/18/2019	Drama	The Dark Knight
3/19/2019	Adventure	Inception
3/19/2019	Drama	12 Angry Men
3/19/2019	Fantasy	The Lord of the Rings

# Sort Keys: Interleaved

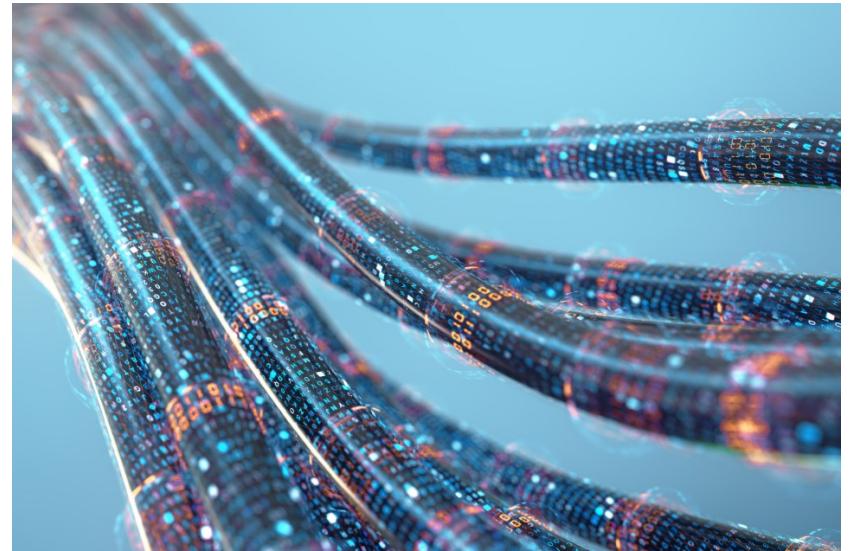
Date	Genre	Movie
3/18/2019	Adventure	Indiana Jones and the Temple of Doom
3/18/2019	Comedy	Monty Python and the Holy Grail
3/18/2019	Drama	Interstellar
3/18/2019	Drama	The Dark Knight
3/19/2019	Adventure	Inception
3/19/2019	Drama	12 Angry Men
3/19/2019	Fantasy	The Lord of the Rings

Date	Genre	Movie
3/19/2019	Drama	12 Angry Men
3/19/2019	Adventure	Inception
3/18/2019	Adventure	Indiana Jones and the Temple of Doom
3/18/2019	Drama	Interstellar
3/18/2019	Comedy	Monty Python and the Holy Grail
3/18/2019	Drama	The Dark Knight
3/19/2019	Fantasy	The Lord of the Rings

Date	Genre	Movie
3/18/2019	Adventure	Indiana Jones and the Temple of Doom
3/19/2019	Adventure	Inception
3/18/2019	Comedy	Monty Python and the Holy Grail
3/18/2019	Drama	Interstellar
3/18/2019	Drama	The Dark Knight
3/19/2019	Drama	12 Angry Men
3/19/2019	Fantasy	The Lord of the Rings

# Importing / Exporting data

- COPY command
  - Parallelized; efficient
  - From S3, EMR, DynamoDB, remote hosts
  - S3 requires a manifest file and IAM role
- UNLOAD command
  - Unload from a table into files in S3
- Enhanced VPC routing



# COPY command: More depth

- Use COPY to load large amounts of data from outside of Redshift
- If your data is already in Redshift in another table,
  - Use INSERT INTO ...SELECT
  - Or CREATE TABLE AS
- COPY can decrypt data as it is loaded from S3
  - Hardware-accelerated SSL used to keep it fast
- Gzip, Izop, and bzip2 compression supported to speed it up further
- Automatic compression option
  - Analyzes data being loaded and figures out optimal compression scheme for storing it
- Special case: narrow tables (lots of rows, few columns)
  - Load with a single COPY transaction if possible
  - Otherwise hidden metadata columns consume too much space

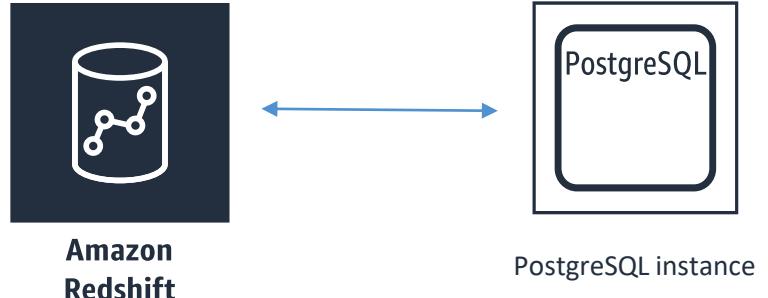


# Redshift copy grants for cross-region snapshot copies

- Let's say you have a KMS-encrypted Redshift cluster and a snapshot of it
- You want to copy that snapshot to another region for backup
- In the destination AWS region:
  - Create a KMS key if you don't have one already
  - Specify a unique name for your snapshot copy grant
  - Specify the KMS key ID for which you're creating the copy grant
- In the source AWS region:
  - Enable copying of snapshots to the copy grant you just created

# DBLINK

- Connect Redshift to PostgreSQL (possibly in RDS)
- Good way to copy and sync data between PostgreSQL and Redshift



PostgreSQL instance

```
CREATE EXTENSION postgres_fdw;
CREATE EXTENSION dblink;
CREATE SERVER foreign_server
    FOREIGN DATA WRAPPER postgres_fdw
    OPTIONS (host '<amazon_redshift_ip>', port '<port>', dbname '<database_name>', sslmode
'require');
CREATE USER MAPPING FOR <rds_postgresql_username>
    SERVER foreign_server
    OPTIONS (user '<amazon_redshift_username>', password '<password>');
```

# Integration with other services

- S3
- DynamoDB
- EMR / EC2
- Data Pipeline
- Database Migration Service



Amazon S3



AWS Database  
Migration  
Service



Amazon  
DynamoDB



Amazon EMR

# Redshift Workload Management (WLM)

- Prioritize short, fast queries vs. long, slow queries
- Query queues
- Via console, CLI, or API

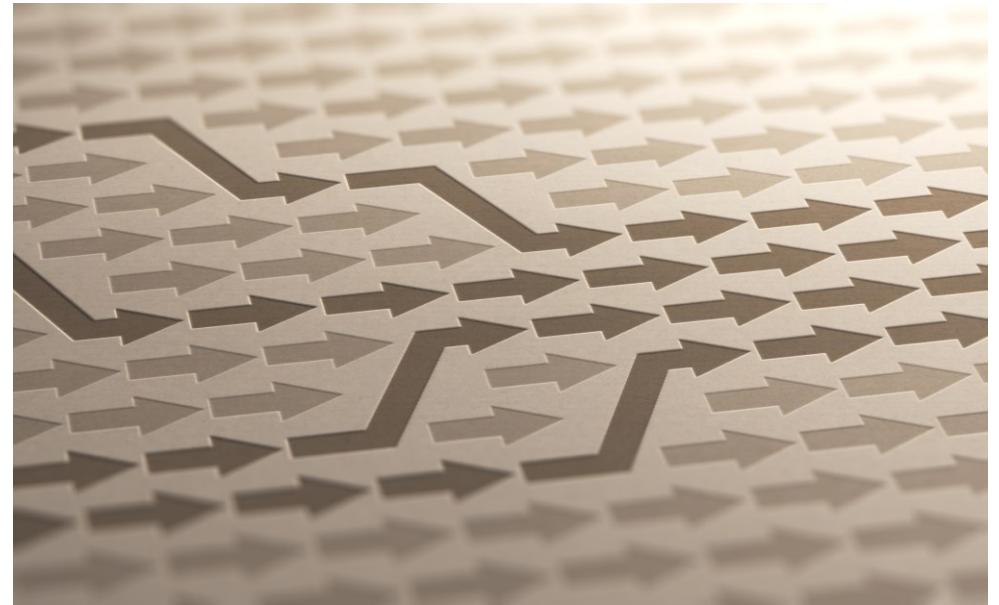
# Concurrency Scaling

- Automatically adds cluster capacity to handle increase in concurrent **read** queries
- Support virtually unlimited concurrent users & queries
- WLM queues manage which queries are sent to the concurrency scaling cluster



# Automatic Workload Management

- Creates up to 8 queues
- Default 5 queues with even memory allocation
- Large queries (ie big hash joins) -> concurrency lowered
- Small queries (ie inserts, scans, aggregations) -> concurrency raised
- Configuring query queues
  - Priority
  - Concurrency scaling mode
  - User groups
  - Query groups
  - Query monitoring rules



# Manual Workload Management

- One default queue with concurrency level of 5 (5 queries at once)
- Superuser queue with concurrency level 1
- Define up to 8 queues, up to concurrency level 50
  - Each can have defined concurrency scaling mode, concurrency level, user groups, query groups, memory, timeout, query monitoring rules
  - Can also enable query queue hopping
    - Timed out queries “hop” to next queue to try again

# Short Query Acceleration (SQA)

- Prioritize short-running queries over longer-running ones
- Short queries run in a dedicated space, won't wait in queue behind long queries
- Can be used in place of WLM queues for short queries
- Works with:
  - CREATE TABLE AS (CTAS)
  - Read-only queries (SELECT statements)
- Uses machine learning to predict a query's execution time
- Can configure how many seconds is “short”

# Resizing Redshift Clusters

- Elastic resize
  - Quickly add or remove nodes of same type
    - (It \*can\* change node types, but not without dropping connections – it creates a whole new cluster)
  - Cluster is down for a few minutes
  - Tries to keep connections open across the downtime
  - Limited to doubling or halving for some dc2 and ra3 node types.
- Classic resize
  - Change node type and/or number of nodes
  - Cluster is read-only for hours to days
- Snapshot, restore, resize
  - Used to keep cluster available during a classic resize
  - Copy cluster, resize new cluster



# VACUUM command

- Recovers space from deleted rows
- VACUUM FULL
- VACUUM DELETE ONLY
- VACUUM SORT ONLY
- VACUUM REINDEX



# Newer Redshift features

- RA3 nodes with managed storage
  - Enable independent scaling of compute and storage
  - SSD-based
- Redshift data lake export
  - Unload Redshift query to S3 in Apache Parquet format
  - Parquet is 2x faster to unload and consumes up to 6X less storage
  - Compatible with Redshift Spectrum, Athena, EMR, SageMaker
  - Automatically partitioned
- Spatial data types
  - GEOMETRY, GEOGRAPHY
- Cross-Region Data Sharing
  - Share live data across Redshift clusters without copying
  - Requires new RA3 node type
  - Secure, across regions and across accounts

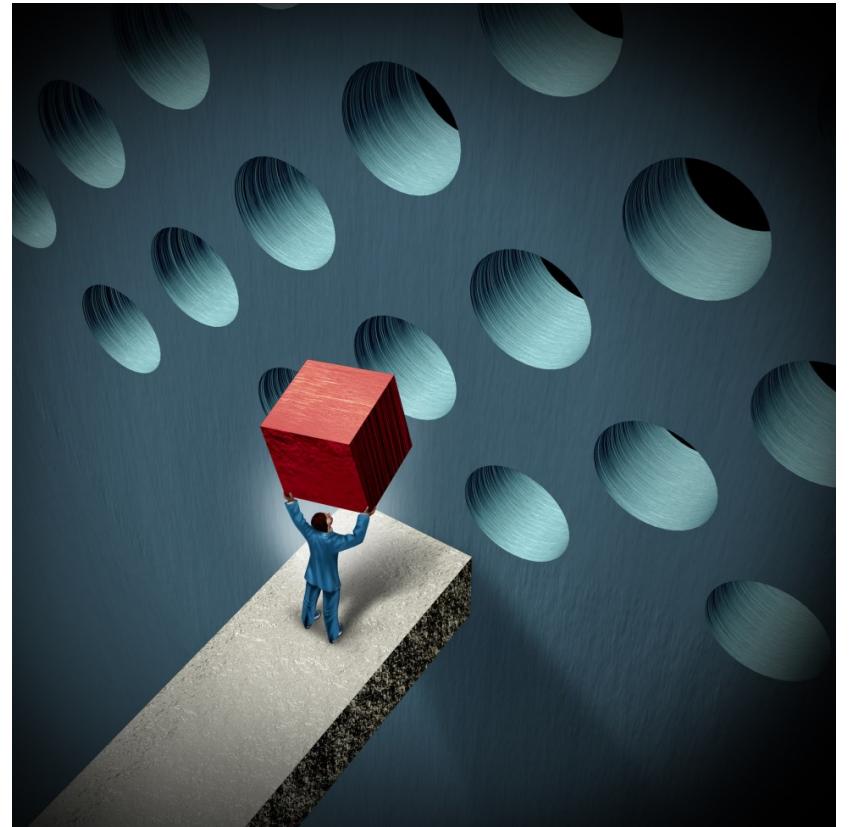
# AQUA

- Advanced Query Accelerator
- Available on ra3.4x1, ra3.16x1
- Pushes reduction and aggregation queries closer to the data
- Up to 10X faster, no extra cost, no code changes.
- Also benefits from high-bandwidth connection to S3
- All you have to do is turn it on in your cluster configuration (when using the supported node types)



# Redshift anti-patterns

- Small data sets
  - Use RDS instead
- OLTP
  - Use RDS or DynamoDB instead
- Unstructured data
  - ETL first with EMR etc.
- BLOB data
  - Store references to large binary files in S3, not the files themselves.



# Redshift security concerns

- Using a Hardware Security Module (HSM)
  - Must use a client and server certificate to configure a trusted connection between Redshift and the HSM
  - If migrating an unencrypted cluster to an HSM-encrypted cluster, you must create the new encrypted cluster and then move data to it.
- Defining access privileges for user or group
  - Use the GRANT or REVOKE commands in SQL
  - Example: grant select on table foo to bob;

# Redshift Serverless

- Automatic scaling and provisioning for your workload
- Optimizes costs & performance
  - Pay only when in use
- Uses ML to maintain performance across variable & sporadic workloads
- Easy spinup of development and test environments
- Easy ad-hoc business analysis
- You get back a serverless endpoint, JDBC/ODBC connection, or just query via the console's query editor.

[Amazon Redshift Serverless \(Preview\)](#) > [Get started with Amazon Redshift Serverless \(Preview\)](#)

## Get started with Amazon Redshift Serverless (Preview)

To start using Amazon Redshift Serverless (Preview), set up your Serverless endpoint and create a database.

### Get started with Serverless credit

#### Serverless credit

Get started with Serverless using your Serverless credit. Serverless credit is available for a limited time if your organization has never created a Serverless endpoint. [Learn more](#)

#### Choose starter base configuration

The starter base configuration has a lower base RPU level for slower consumption of your Serverless credit. You can remain at that RPU level or increase as needed.

### Configuration

#### Use default settings

Default settings have been defined to help you get started. You can change them at any time.

#### Customize settings

Customize your settings for your specific needs.

### Database name and password

#### Database name

The name of the first database in the Amazon Redshift Serverless environment.

dev

The name must be 1-64 alphanumeric characters (lowercase only), and it can't be a reserved word.

#### Admin user credentials

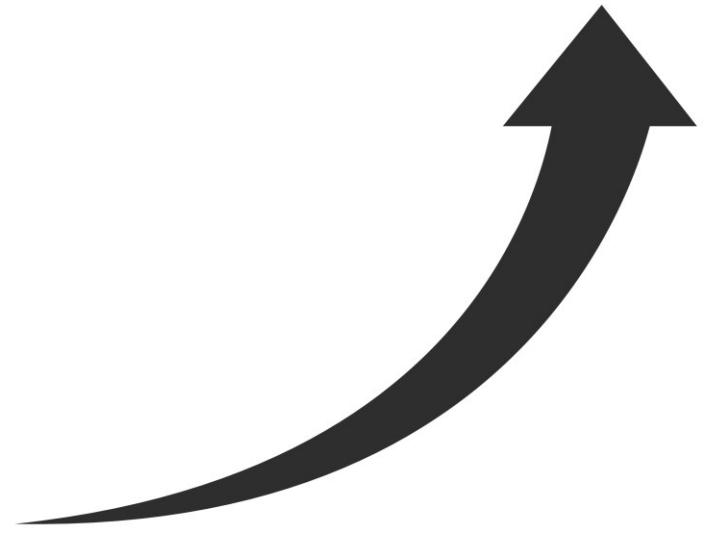
# Redshift Serverless: Getting Started

- Need an IAM role with this policy
- Define your
  - Database name
  - Admin user credentials
  - VPC
  - Encryption settings
    - AWS-owned KMS by default
  - Audit logging
- Can manage snapshots & recovery points after creation

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "redshift-serverless:*",  
      "Resource": "*"  
    }  
  ]  
}
```

# Resource Scaling in Redshift Serverless

- Capacity measured in Redshift Processing Units (RPU's)
- You pay for RPU-hours (per second) plus storage
- Base RPU's
  - You can adjust base capacity
  - Defaults to AUTO
  - But you can adjust from 32-512 RPU's to improve query performance
- Max RPU's
  - Can set a usage limit to control costs
  - Or, increase it to improve throughput



# Redshift Serverless

- Does everything Redshift can, except:
  - Redshift Spectrum
  - Parameter Groups
  - Workload Management
  - AWS Partner integration
  - Maintenance windows / version tracks
- No public endpoints (yet)
  - Must access within a VPC

# Redshift Serverless: Monitoring

- Monitoring views
  - SYS\_QUERY\_HISTORY
  - SYS\_LOAD\_HISTORY
  - SYS\_SERVERLESS\_USAGE
  - ...and many more
- CloudWatch logs
  - Connection & user logs enabled by default
  - Optional user activity log data
  - Under /aws/redshift/serverless/
- CloudWatch metrics
  - QueriesCompletedPerSecond, QueryDuration, QueriesRunning, etc.
  - Dimensions: DatabaseName, latency (short/medium/long), QueryType, stage



# Amazon RDS

## Relational Database Service

# What is RDS?

- Hosted relational database
  - Amazon Aurora
  - MySQL
  - PostgreSQL
  - MariaDB
  - Oracle
  - SQL Server
- Not for “big data”
  - Might appear on exam as an example of what not to use
  - Or in the context of migrating from RDS to Redshift etc.



**Amazon RDS**

# ACID

- RDS databases offer full ACID compliance
  - Atomicity
  - Consistency
  - Isolation
  - Durability



# Amazon Aurora

- MySQL and PostgreSQL – compatible
- Up to 5X faster than MySQL, 3X faster than PostgreSQL
- 1/10 the cost of commercial databases
- Up to 64TB per database instance
- Up to 15 read replicas
- Continuous backup to S3
- Replication across availability zones
- Automatic scaling with Aurora Serverless

# Aurora Security

- VPC network isolation
- At-rest with KMS
  - Data, backup, snapshots, and replicas can be encrypted
- In-transit with SSL



# Amazon QuickSight

Business analytics and visualizations in the cloud

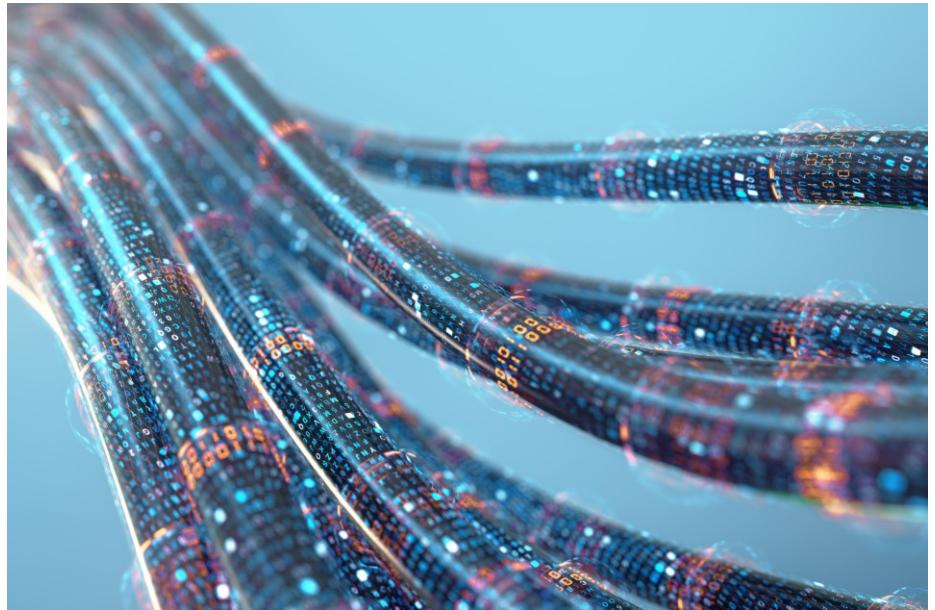
# What is QuickSight?

- Fast, easy, cloud-powered business analytics service
- Allows all employees in an organization to:
  - Build visualizations
  - Perform ad-hoc analysis
  - Quickly get business insights from data
  - Anytime, on any device (browsers, mobile)
- Serverless



# QuickSight Data Sources

- Redshift
- Aurora / RDS
- Athena
- EC2-hosted databases
- Files (S3 or on-premises)
  - Excel
  - CSV, TSV
  - Common or extended log format
- Data preparation allows limited ETL



# SPICE

- Data sets are imported into SPICE
  - Super-fast, Parallel, In-memory Calculation Engine
  - Uses columnar storage, in-memory, machine code generation
  - Accelerates interactive queries on large datasets
- Each user gets 10GB of SPICE
- Highly available / durable
- Scales to hundreds of thousands of users
- Can accelerate large queries that would time out in direct query mode (hitting Athena directly)
  - But if it takes more than 30 minutes to import your data into SPICE it will still time out

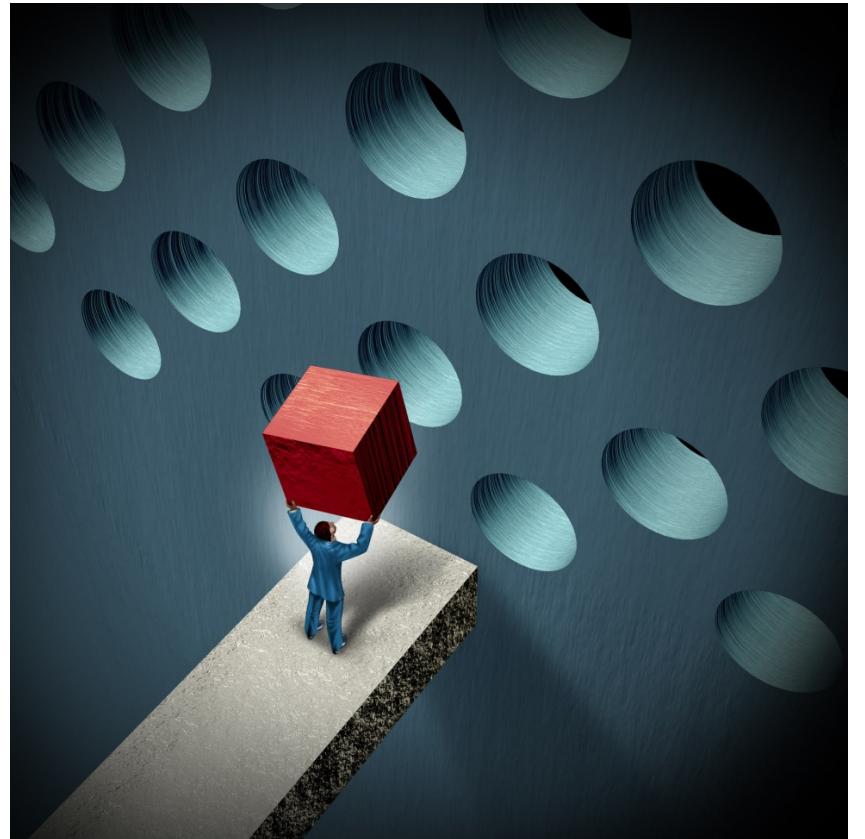


# QuickSight Use Cases

- Interactive ad-hoc exploration / visualization of data
- Dashboards and KPI's
- Analyze / visualize data from:
  - Logs in S3
  - On-premise databases
  - AWS (RDS, Redshift, Athena, S3)
  - SaaS applications, such as Salesforce
  - Any JDBC/ODBC data source

# QuickSight Anti-Patterns

- Highly formatted canned reports
  - QuickSight is for ad-hoc queries, analysis, and visualization
- ETL
  - Use Glue instead, although QuickSight can do some transformations



# QuickSight Security

- Multi-factor authentication on your account
- VPC connectivity
  - Add QuickSight's IP address range to your database security groups
- Row-level security
  - New for 2021: Column-level security too (CLS) – Enterprise edition only
- Private VPC access
  - Elastic Network Interface, AWS Direct Connect



# QuickSight Security

- Resource access
  - Must ensure QuickSight is authorized to use Athena / S3 / your S3 buckets
  - This can be managed within the QuickSight console (Manage Quicksight / Security & Permissions)\
- Data access
  - Can create IAM policies to restrict what data in S3 given QuickSight users can access



# Quicksight + Redshift: Security

- By default Quicksight can only access data stored IN THE SAME REGION as the one Quicksight is running within
- So if Quicksight is running in one region, and Redshift in another, that's a problem
- A VPC configured to work across AWS regions won't work!
- Solution: **create a new security group with an inbound rule authorizing access from the IP range of QuickSight servers in that region**
  - Those ranges are documented at  
<https://docs.aws.amazon.com/quicksight/latest/user/regions.html>

# QuickSight User Management

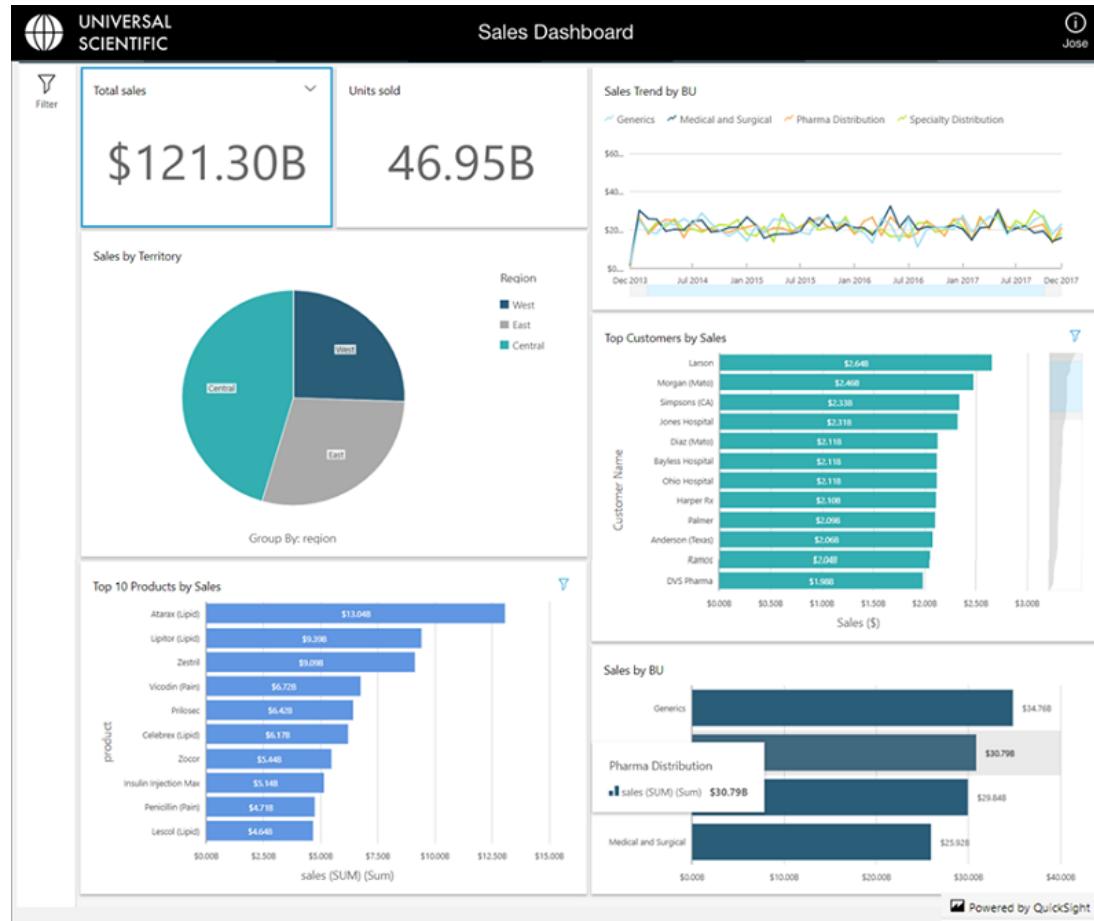
- Users defined via IAM, or email signup
- Active Directory connector with QuickSight Enterprise Edition
  - All keys are managed by AWS; you CANNOT use customer-provided keys
  - Enterprise edition only!
  - Can tweak security access using IAM if needed



# QuickSight Pricing

- Annual subscription
  - Standard: \$9 / user / month
  - Enterprise: \$18 / user / month
- Extra SPICE capacity (beyond 10GB)
  - \$0.25 (standard) \$0.38 (enterprise) / GB / user / month
- Month to month
  - Standard: \$12 / user / month
  - Enterprise: \$24 / user / month
- Enterprise edition
  - Encryption at rest
  - Microsoft Active Directory integration

# QuickSight Dashboards

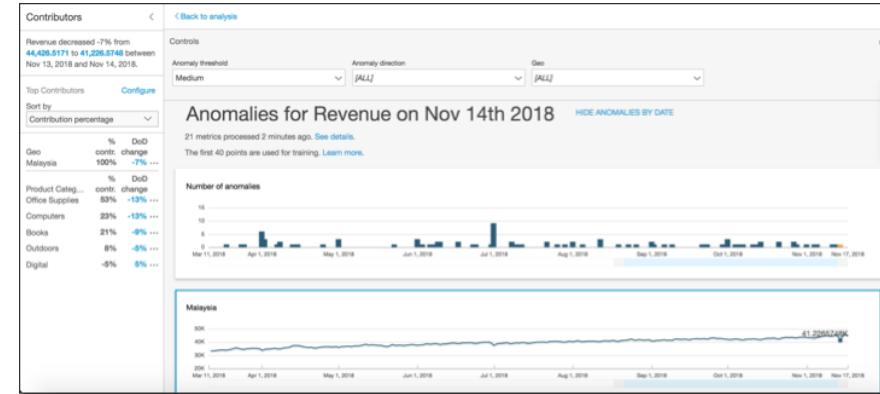


- Read-only snapshots of an analysis
- Can share with others with Quicksight access
- Can share even more widely with *embedded dashboards*
  - Embed within an application
  - Authenticate with Active Directory / Cognito / SSO
  - QuickSight Javascript SDK / QuickSight API
  - Whitelist domains where embedding is allowed

Image: AWS Big Data Blog

# Quicksight Machine Learning Insights

- ML-powered anomaly detection
  - Uses Random Cut Forest
  - Identify top contributors to significant changes in metrics
- ML-powered forecasting
  - Also uses Random Cut Forest
  - Detects seasonality and trends
  - Excludes outliers and imputes missing values
- Autonarratives
  - Adds “story of your data” to your dashboards
- Suggested Insights
  - “Insights” tab displays read-to-use suggested insights



Total Revenue for Nov 17, 2018 **decreased by 2.15%**  
**(-131,031.34720000066)** from 6,100,697.1616 to  
 5,969,665.8144. Compounded growth rate for the last 4 days is  
**-0.26% worse than expected.**

# Quicksight Q

- Machine learning-powered
- Answers business questions with Natural Language Processing
  - “What are the top-selling items in Florida?”
- Offered as an add-on for given regions
- Personal training on how to use it is required
- Must set up *topics* associated with *datasets*
  - Datasets and their fields must be NLP-friendly
  - How to handle dates must be defined



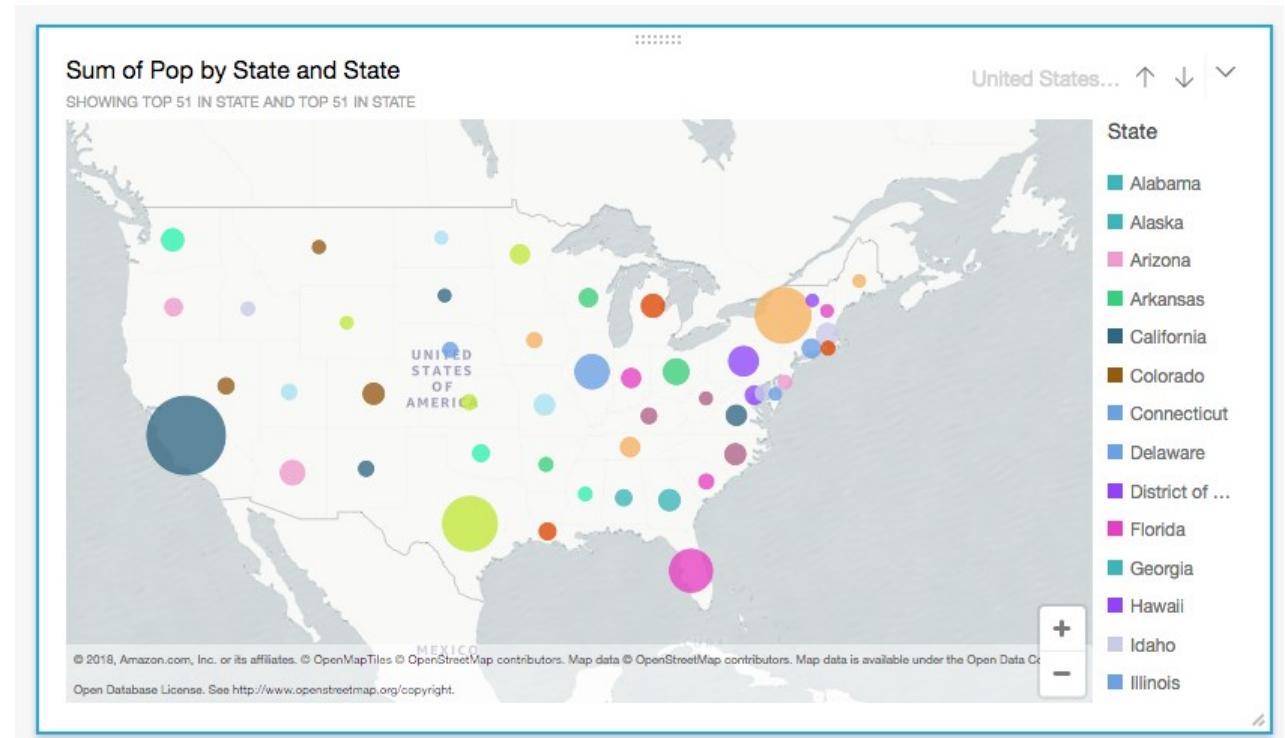
# QuickSight Visual Types

- AutoGraph
- Bar Charts
  - For comparison and distribution (histograms)
- Line graphs
  - For changes over time
- Scatter plots, heat maps
  - For correlation
- Pie graphs, tree maps
  - For aggregation
- Pivot tables
  - For tabular data



# Additional Visual Types

- KPIs
- Geospatial Charts (maps)
- Donut Charts
- Gauge Charts
- Word Clouds



# Bar Charts: Comparison, Distribution

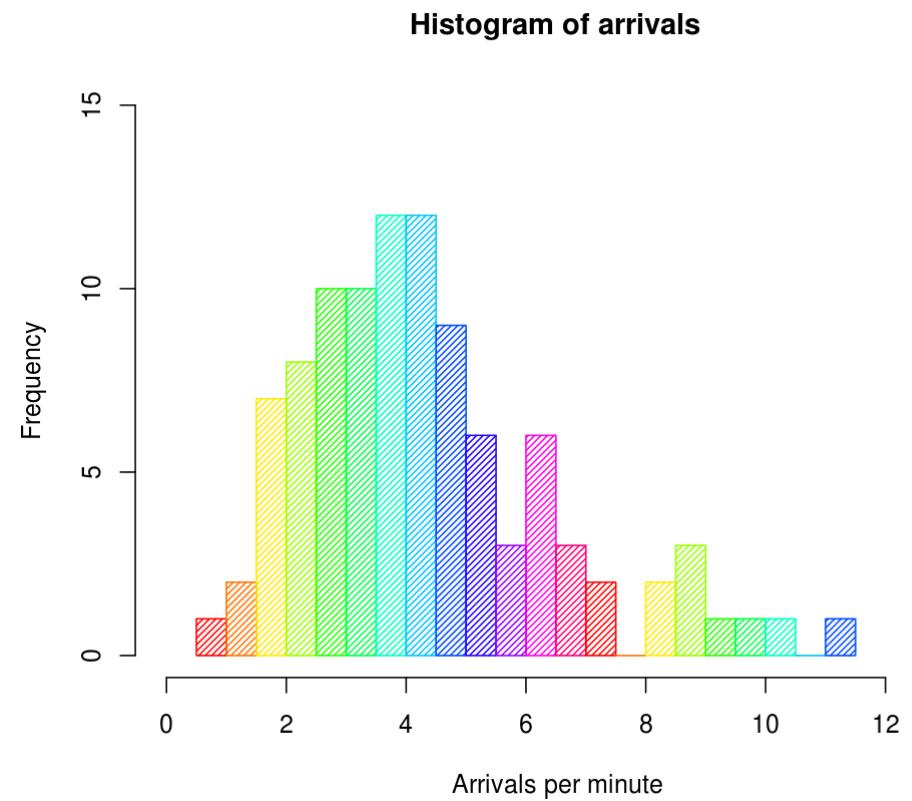
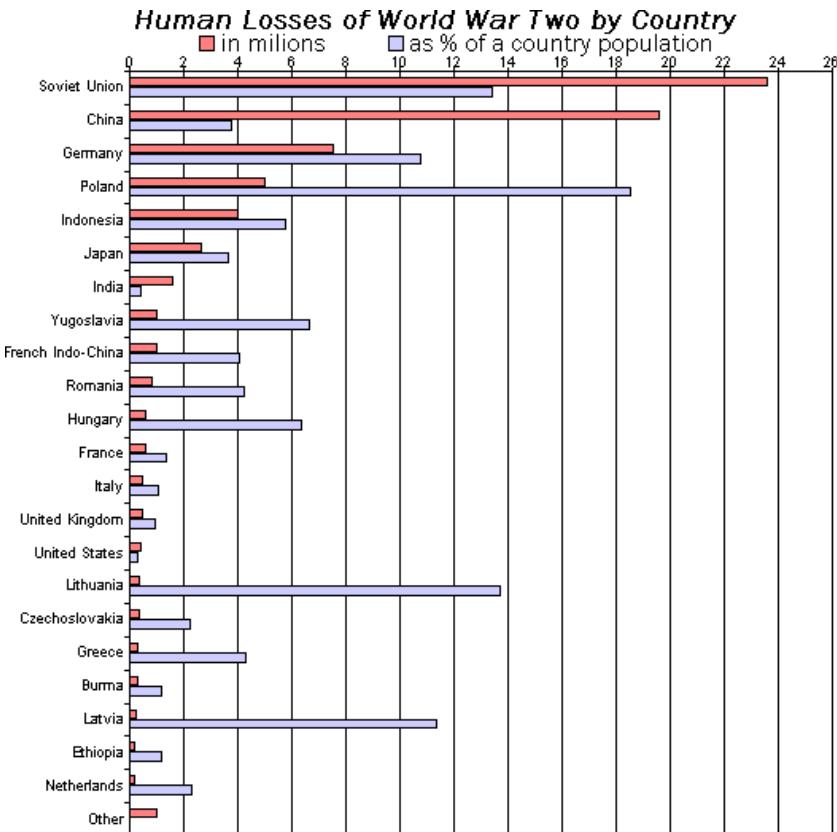
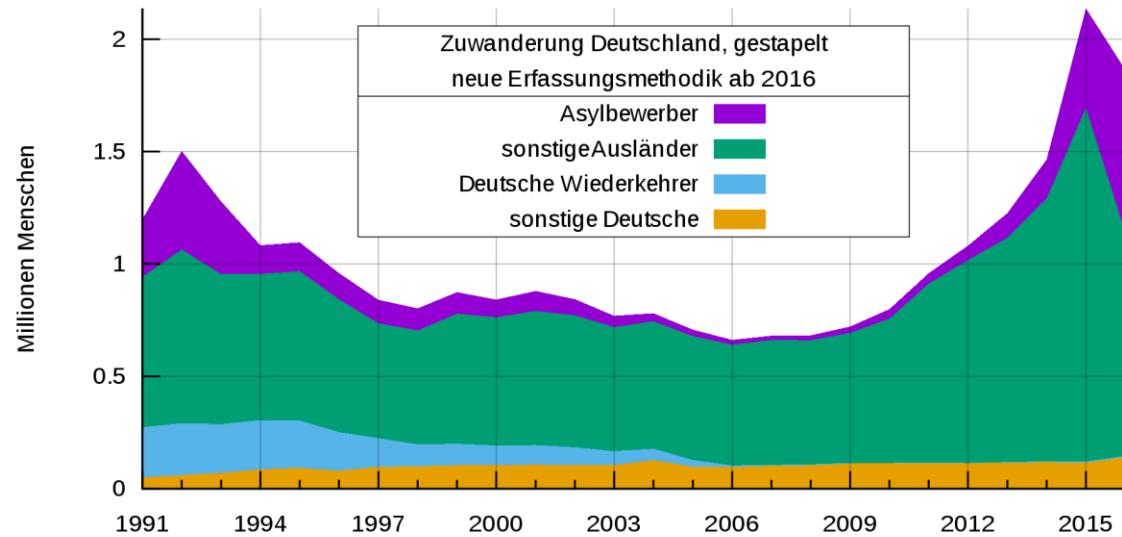
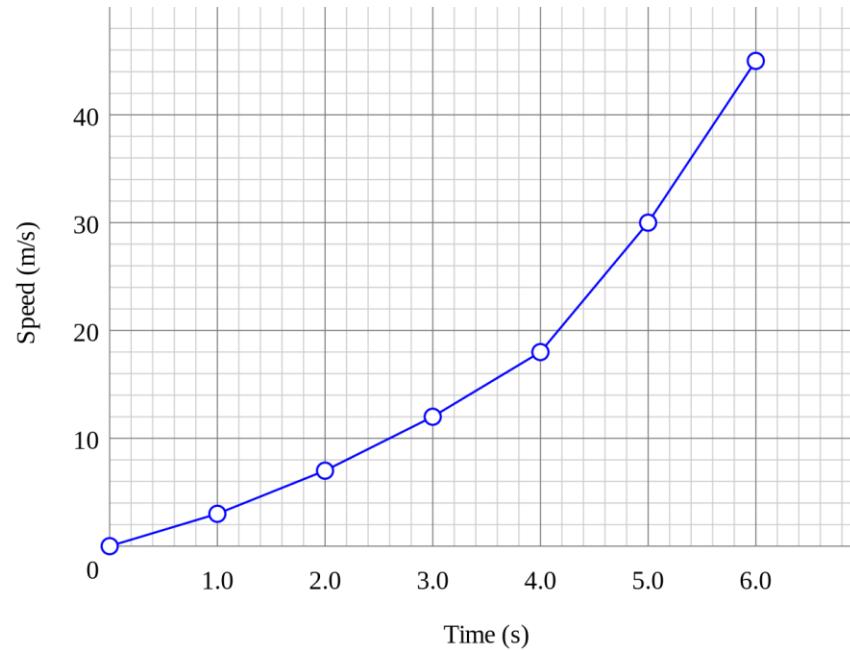
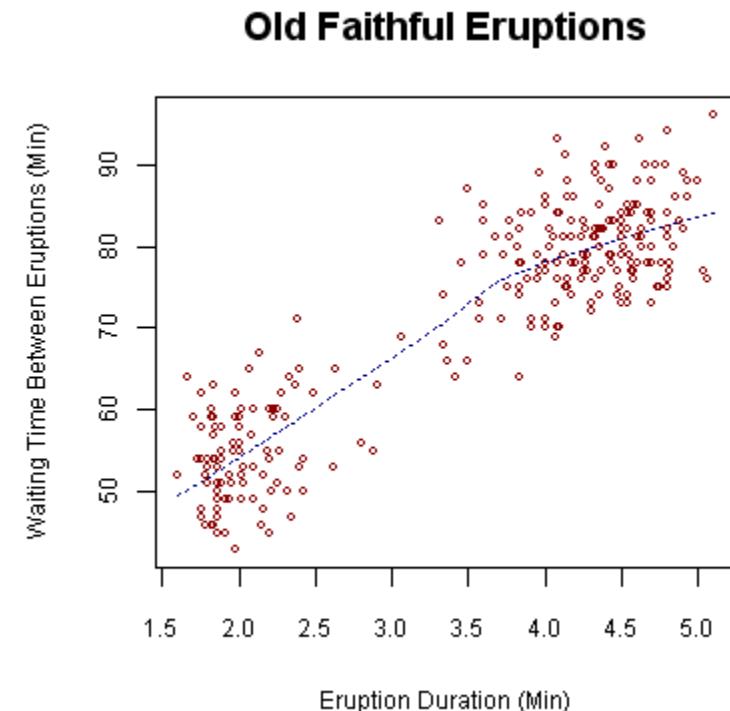


Image: DanielPenfield, Wikipedia CC BY-SA 3.0

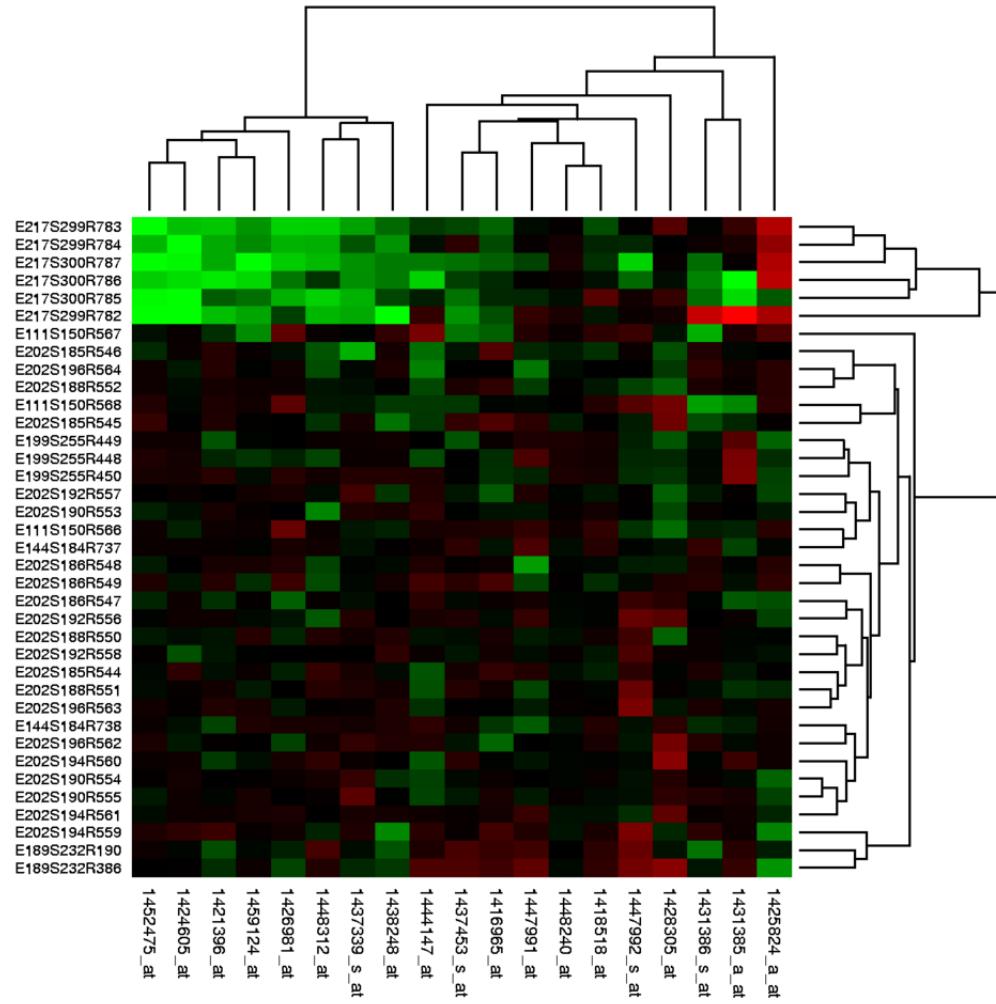
# Line Charts: Changes over Time



# Scatter Plots: Correlation



# Heat Maps: Correlation



# Pie Charts: Aggregation

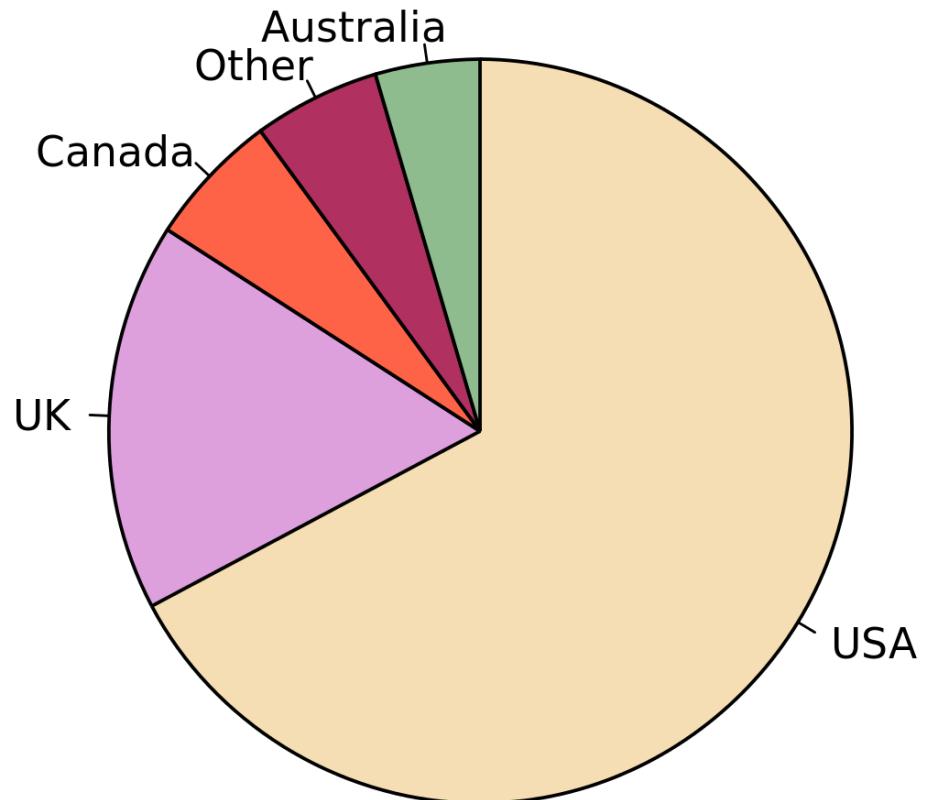
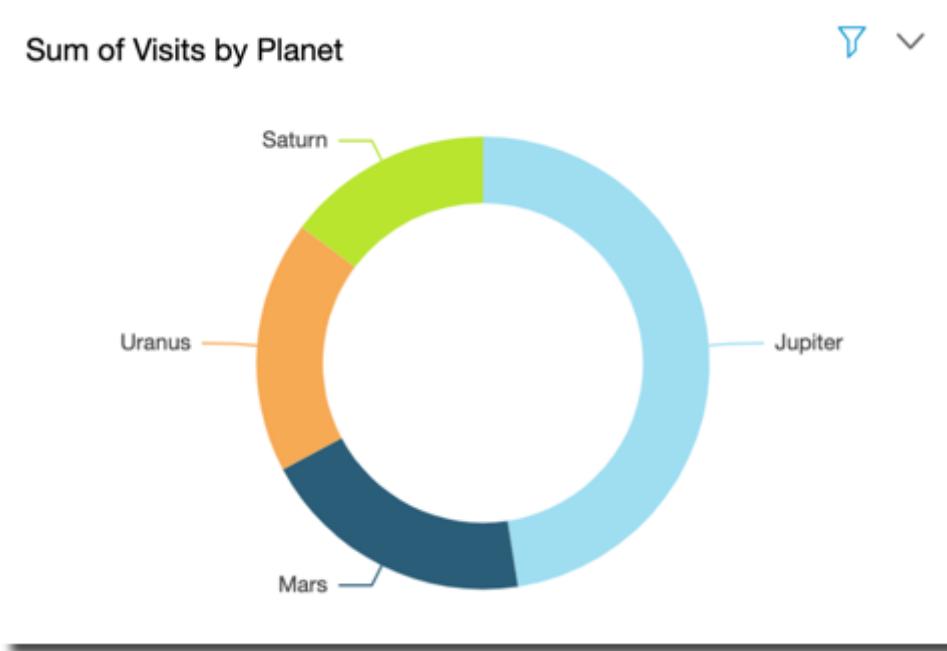
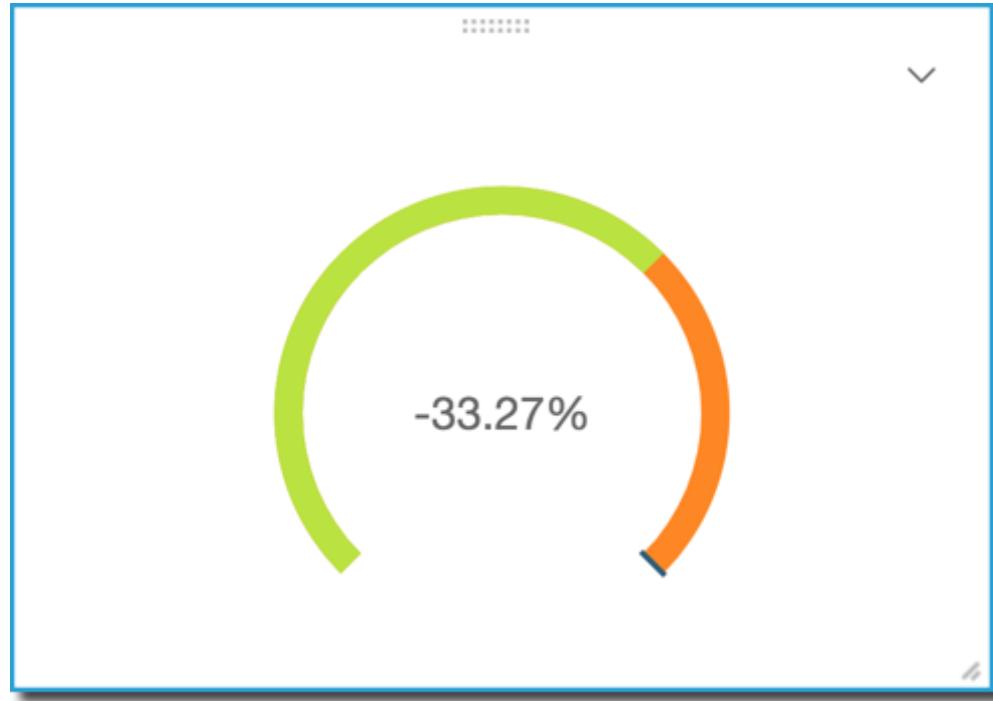


Image: M.W. Toews, Wikipedia, CC BY-SA 4.0

# Donut Charts: Percentage of Total Amount



# Gauge Charts: Compare values in a measure



# Tree Maps: Heirarchical Aggregation

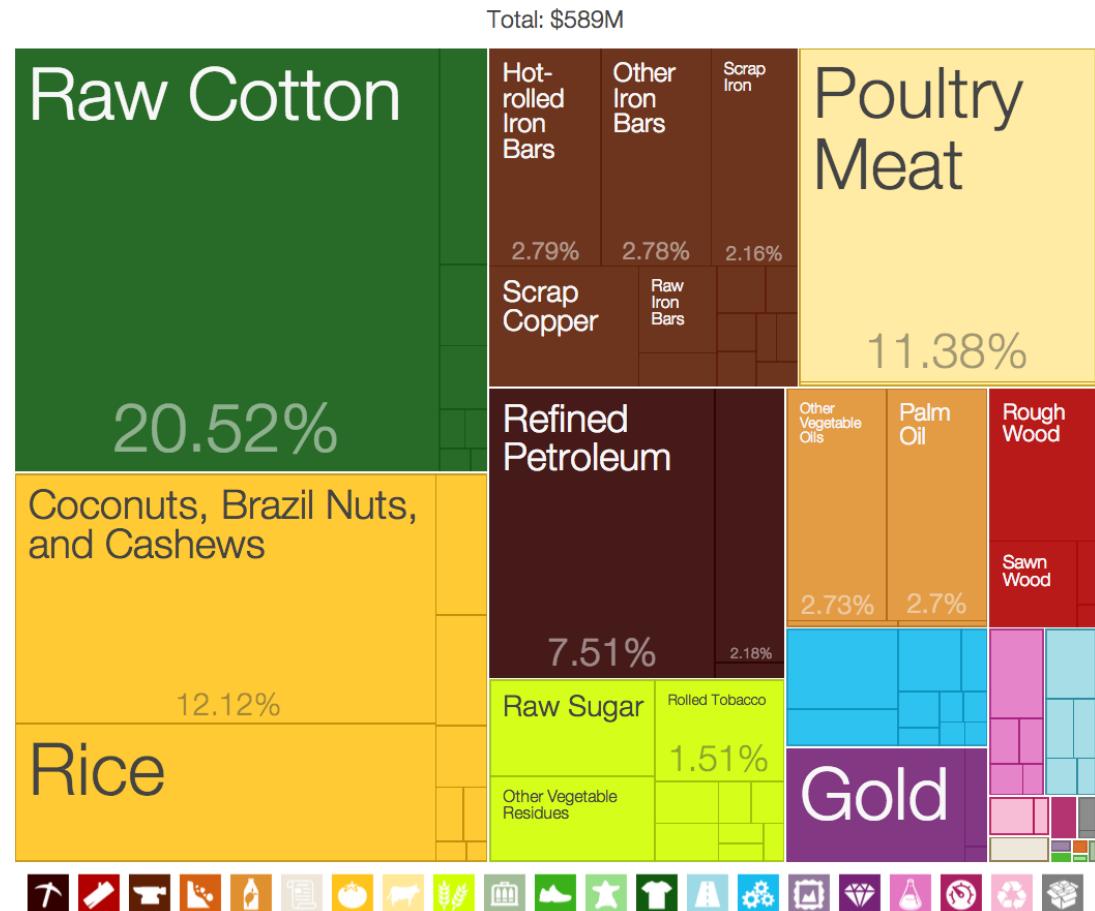


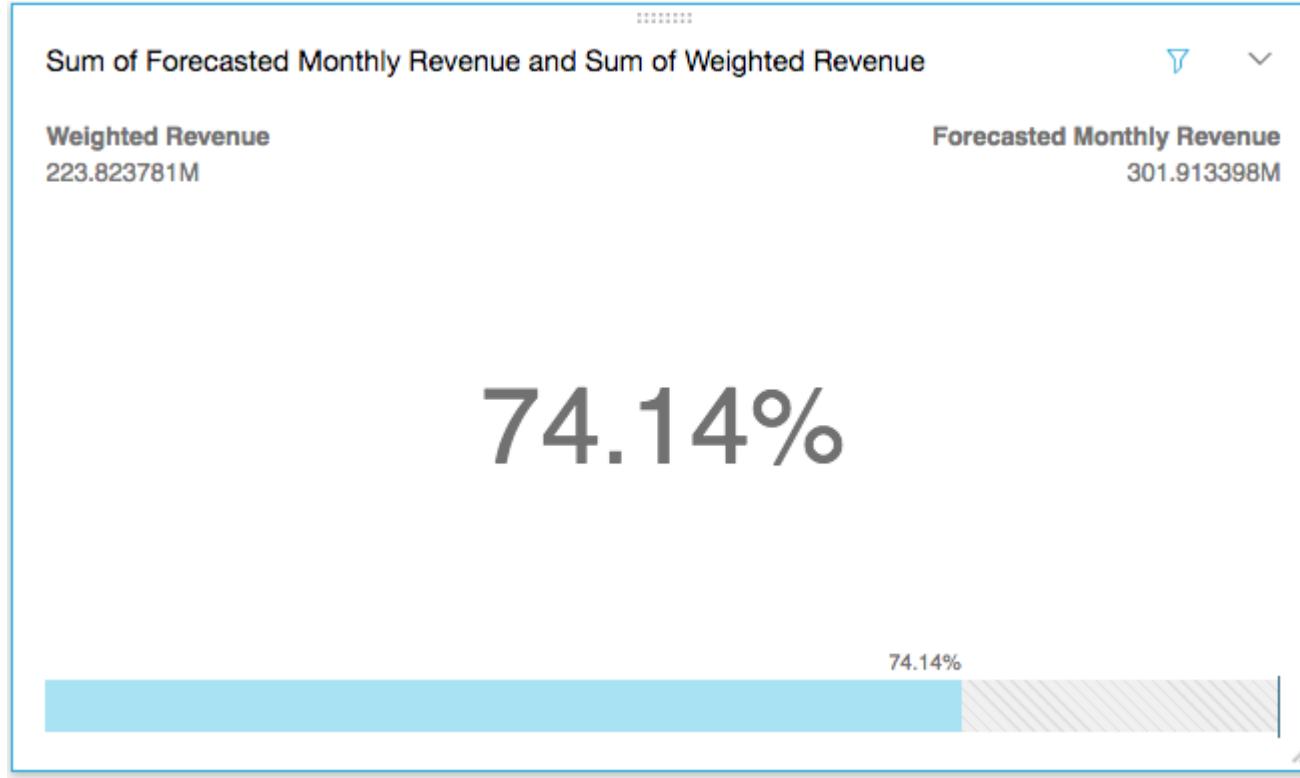
Image: Harvard-MIT Observatory of Economic Complexity, Wikipedia, CC-BY-SA 3.0

# Pivot Tables: Tabular Data

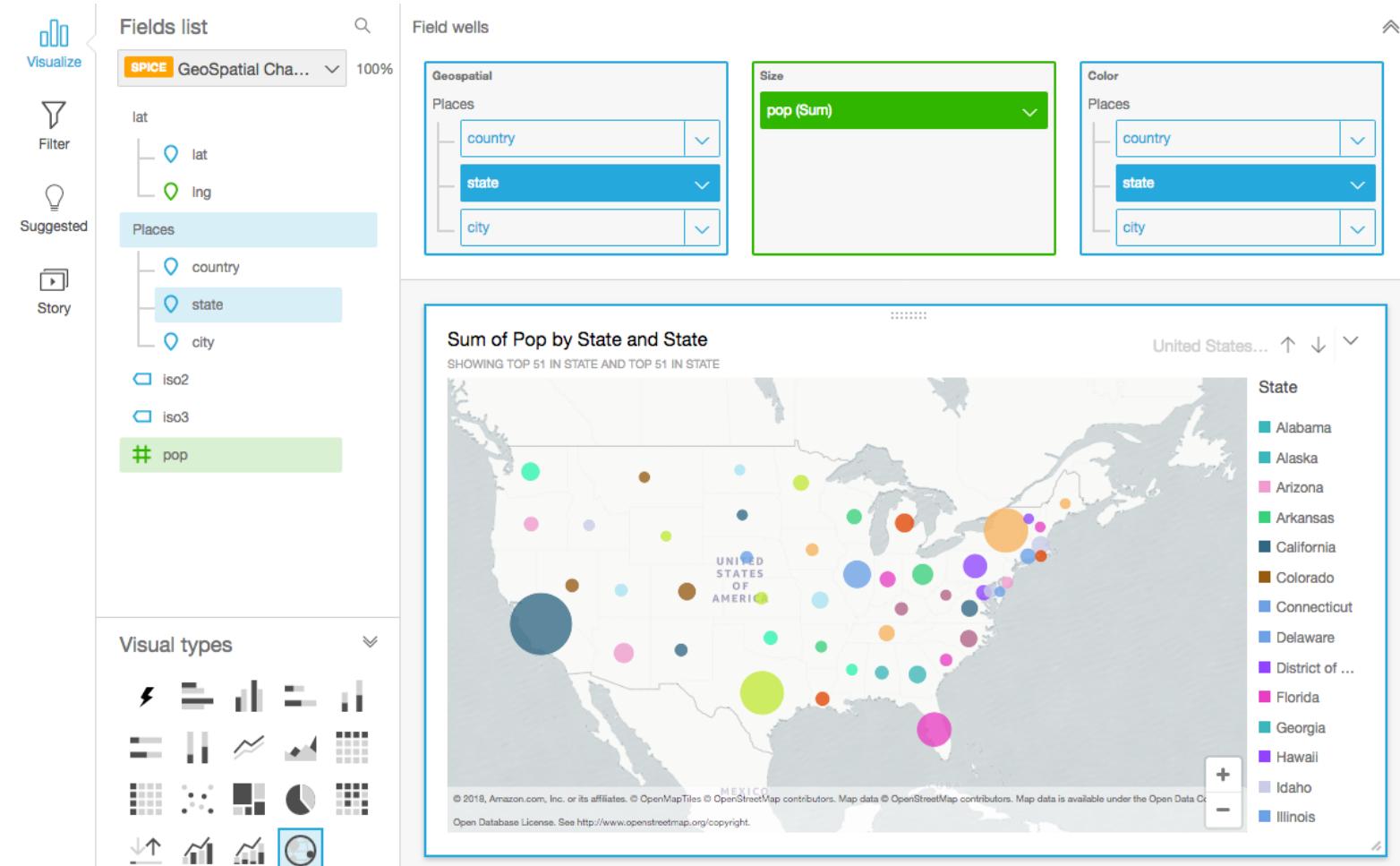
	A	B	C	D	E	F	G
1	Region	Gender	Style	Ship Date	Units	Price	Cost
2	East	Boy	Tee	1/31/2005	12	11.04	10.42
3	East	Boy	Golf	1/31/2005	12	13	12.6
4	East	Boy	Fancy	1/31/2005	12	11.96	11.74
5	East	Girl	Tee	1/31/2005	10	11.27	10.56
6	East	Girl	Golf	1/31/2005	10	12.12	11.95
7	East	Girl	Fancy	1/31/2005	10	13.74	13.33
8	West	Boy	Tee	1/31/2005	11	11.44	10.94
9	West	Boy	Golf	1/31/2005	11	12.63	11.73
10	West	Boy	Fancy	1/31/2005	11	12.06	11.51
11	West	Girl	Tee	1/31/2005	15	13.42	13.29
12	West	Girl	Golf	1/31/2005	15	11.48	10.67

Sum of Units	Ship Date ▼	1/31/2005	2/28/2005	3/31/2005	4/30/2005	5/31/2005	6/30/2005
Region	▼	66	80	102	116	127	125
East		96	117	138	151	154	156
North		123	141	157	178	191	202
South		78	97	117	136	150	157
(blank)							
Grand Total		363	435	514	581	622	640

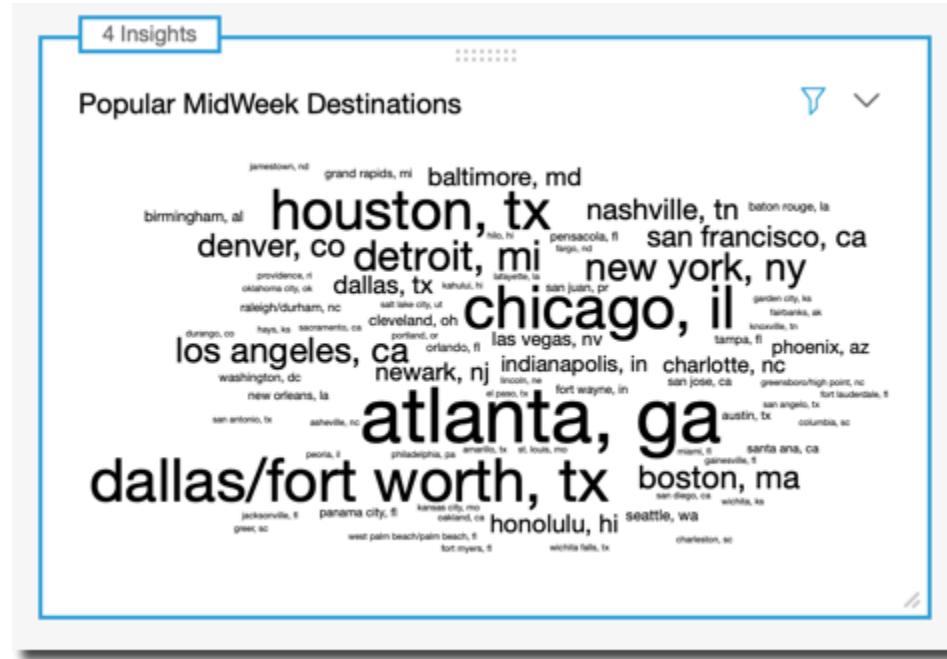
# KPI's: compare key value to its target value



# Geospatial Charts



# Word Clouds: word or phrase frequency



# Alternative Visualization Tools

- Web-based visualizations tools (deployed to the public)
  - D3.js
  - Chart.js
  - Highchart.js
- Business Intelligence Tools
  - Tableau
  - MicroStrategy



# Security

# Why encryption?

## Encryption in flight (TLS / SSL)

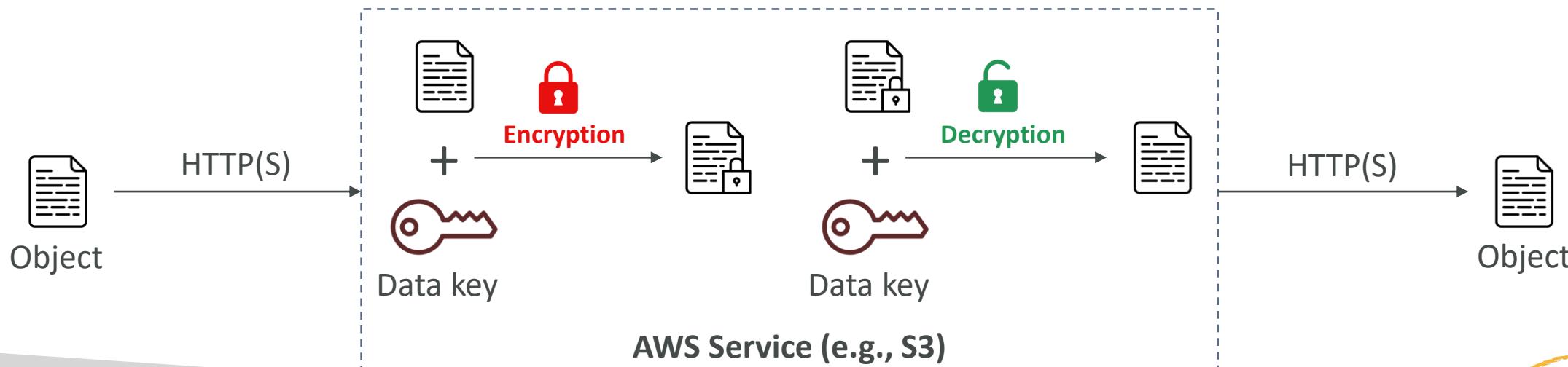
- Data is encrypted before sending and decrypted after receiving
- TLS certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



# Why encryption?

## Server-side encryption at rest

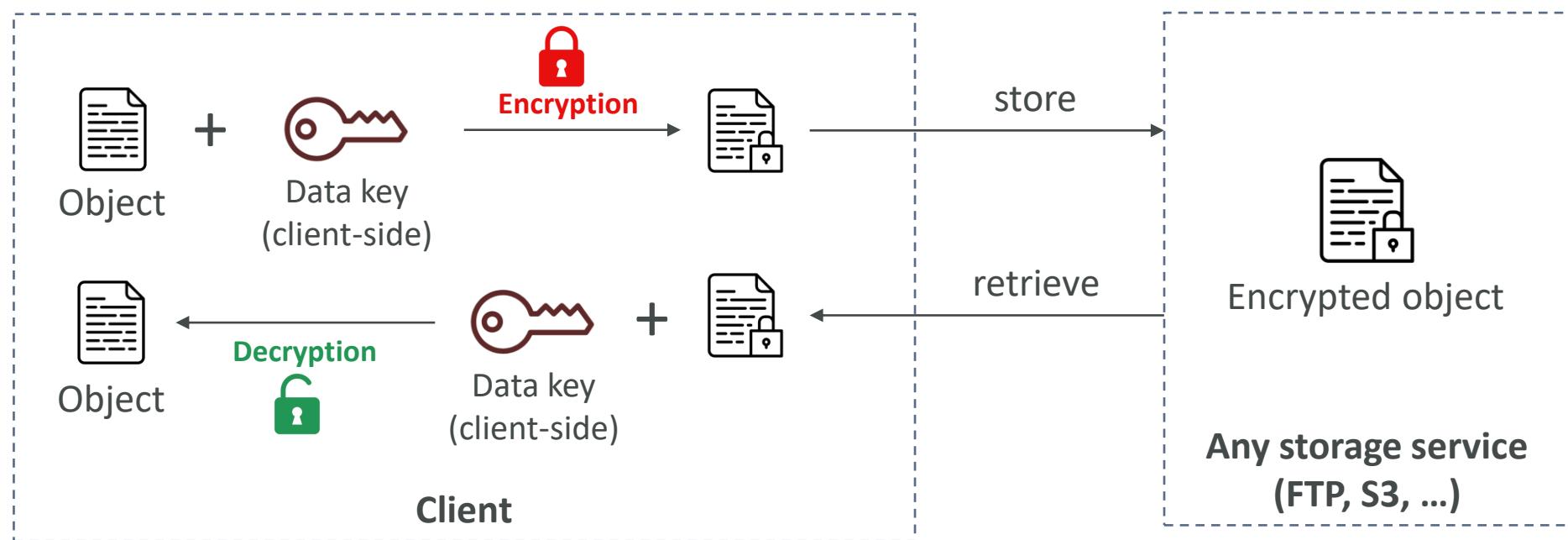
- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere, and the server must have access to it



# Why encryption?

## Client-side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption

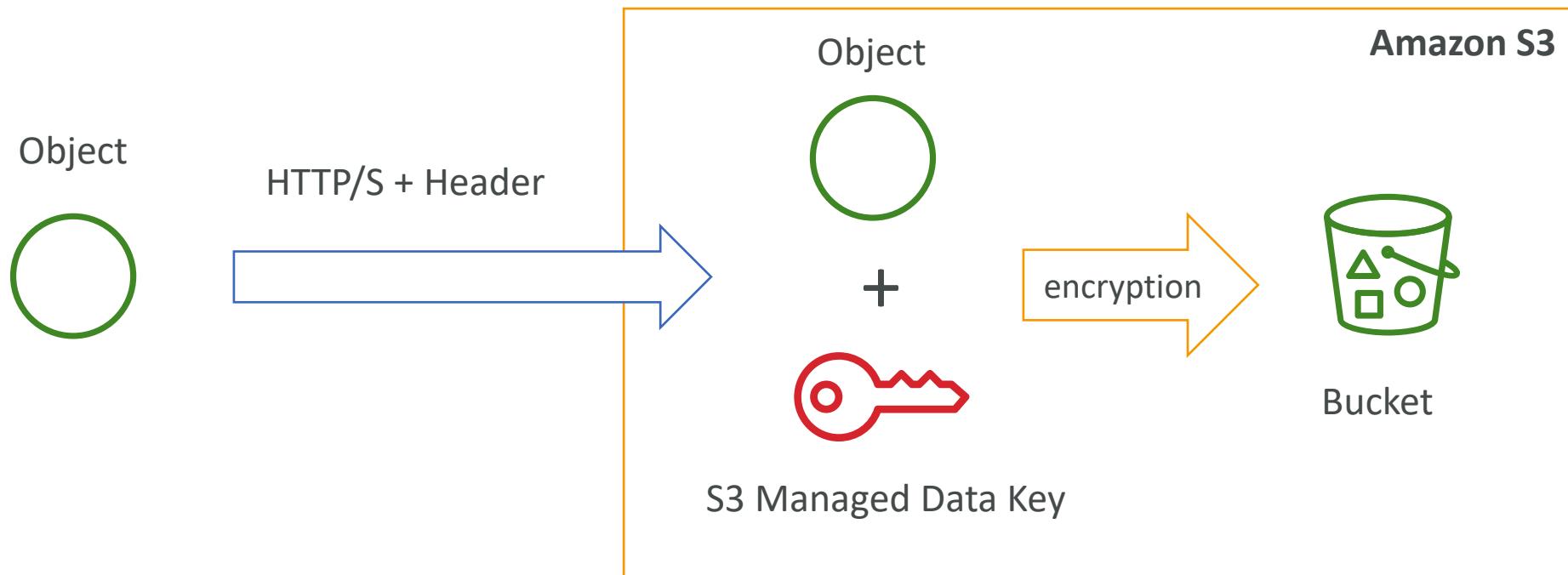


# S3 Encryption for Objects

- There are 4 methods of encrypting objects in S3
  - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
  - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
  - SSE-C: when you want to manage your own encryption keys
  - Client Side Encryption
- It's important to understand which ones are adapted to which situation for the exam

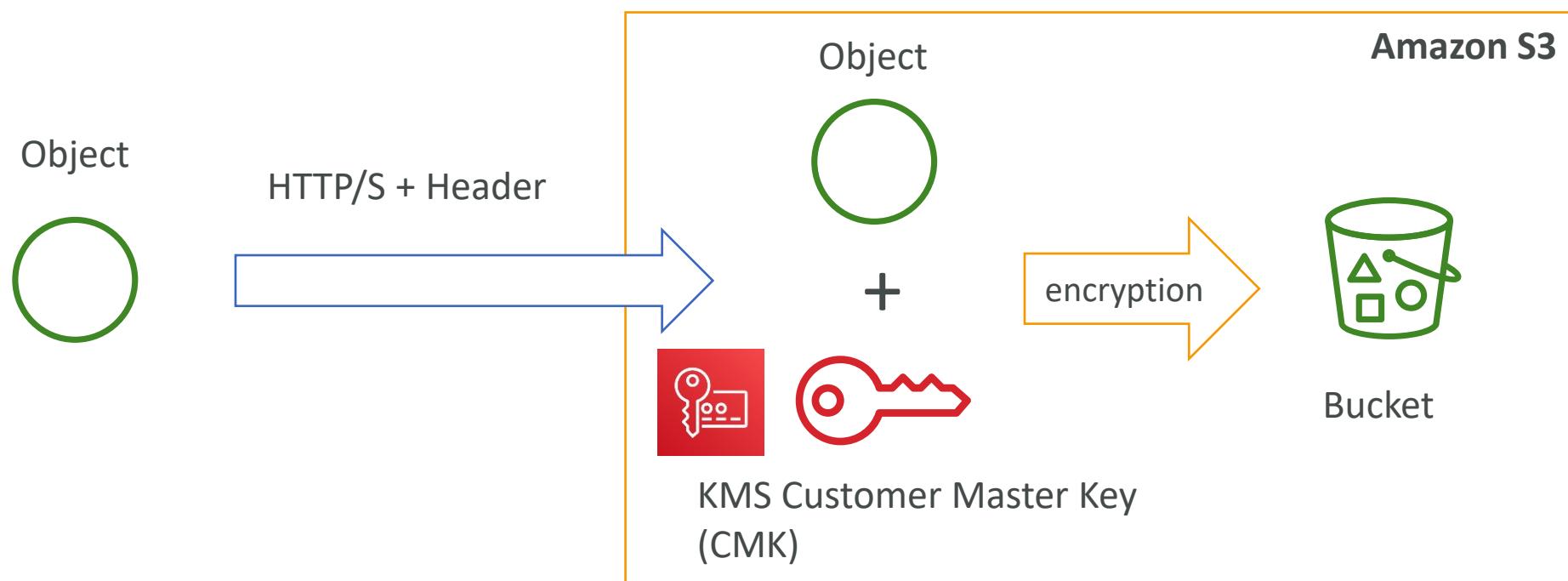
# SSE-S3

- SSE-S3: encryption using keys handled & managed by Amazon S3
- Object is encrypted server side
- AES-256 encryption type
- Must set header: “**x-amz-server-side-encryption**”: "AES256"



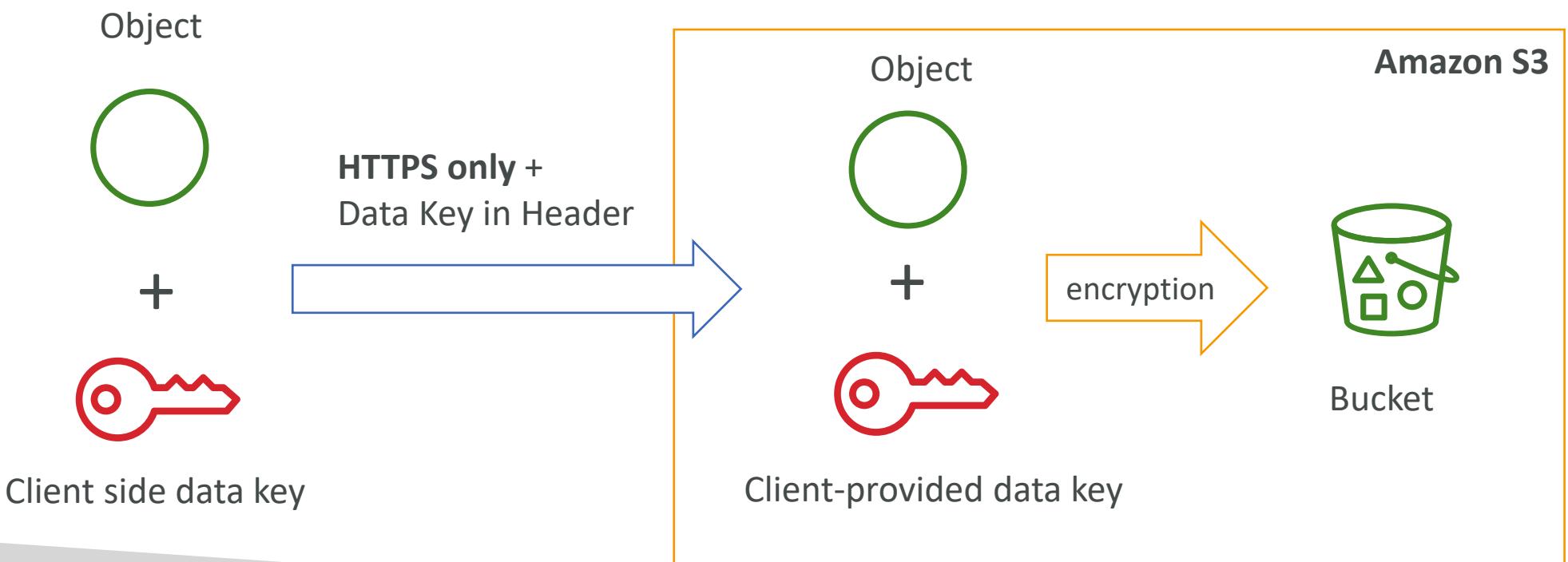
# SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: “**x-amz-server-side-encryption**”: “aws:kms”



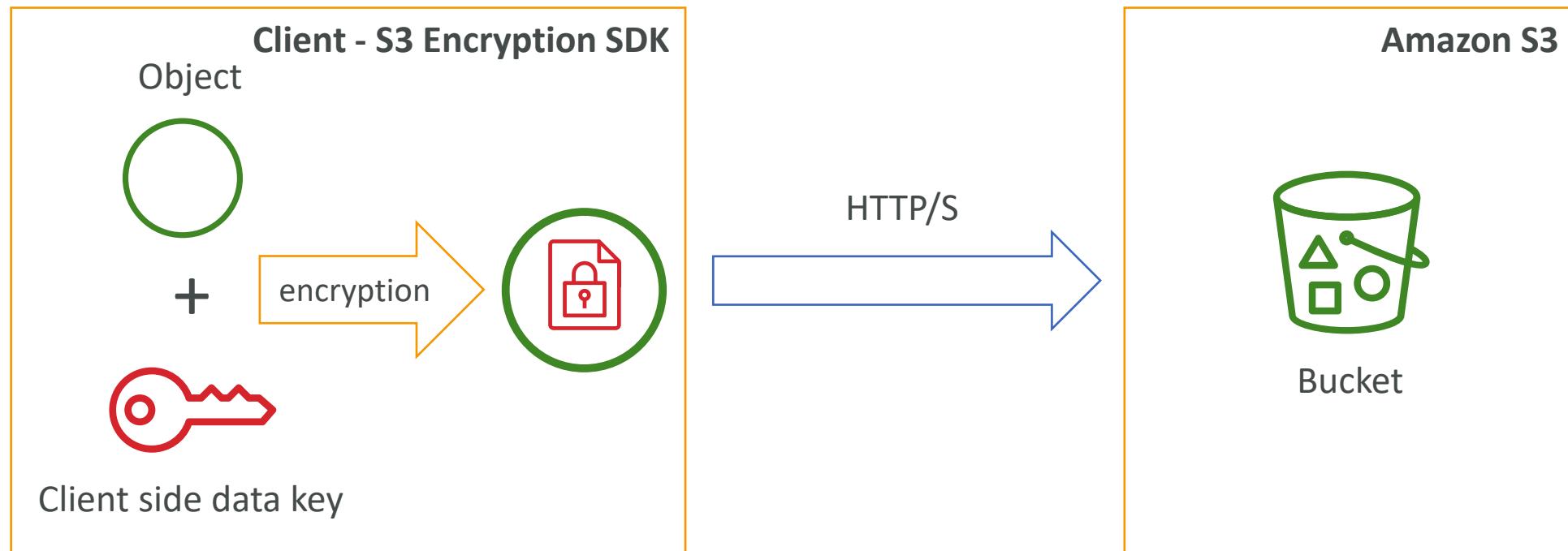
# SSE-C

- SSE-C: server-side encryption using data keys fully managed by the customer outside of AWS
- Amazon S3 does not store the encryption key you provide
- **HTTPS must be used**
- Encryption key must provided in HTTP headers, for every HTTP request made



# Client Side Encryption

- Client library such as the Amazon S3 Encryption Client
- Clients must encrypt data themselves before sending to S3
- Clients must decrypt data themselves when retrieving from S3
- Customer fully manages the keys and encryption cycle



# Encryption in transit (SSL/TLS)



- Amazon S3 exposes:
  - HTTP endpoint: non encrypted
  - HTTPS endpoint: encryption in flight
- You're free to use the endpoint you want, but HTTPS is recommended
- Most clients would use the HTTPS endpoint by default
- HTTPS is mandatory for SSE-C
- Encryption in flight is also called SSL / TLS

# AWS KMS (Key Management Service)



- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
  - Amazon EBS: encrypt volumes
  - Amazon S3: Server side encryption of objects
  - Amazon Redshift: encryption of data
  - Amazon RDS: encryption of data
  - Amazon SSM: Parameter store
  - Etc...
- But you can also use the CLI / SDK

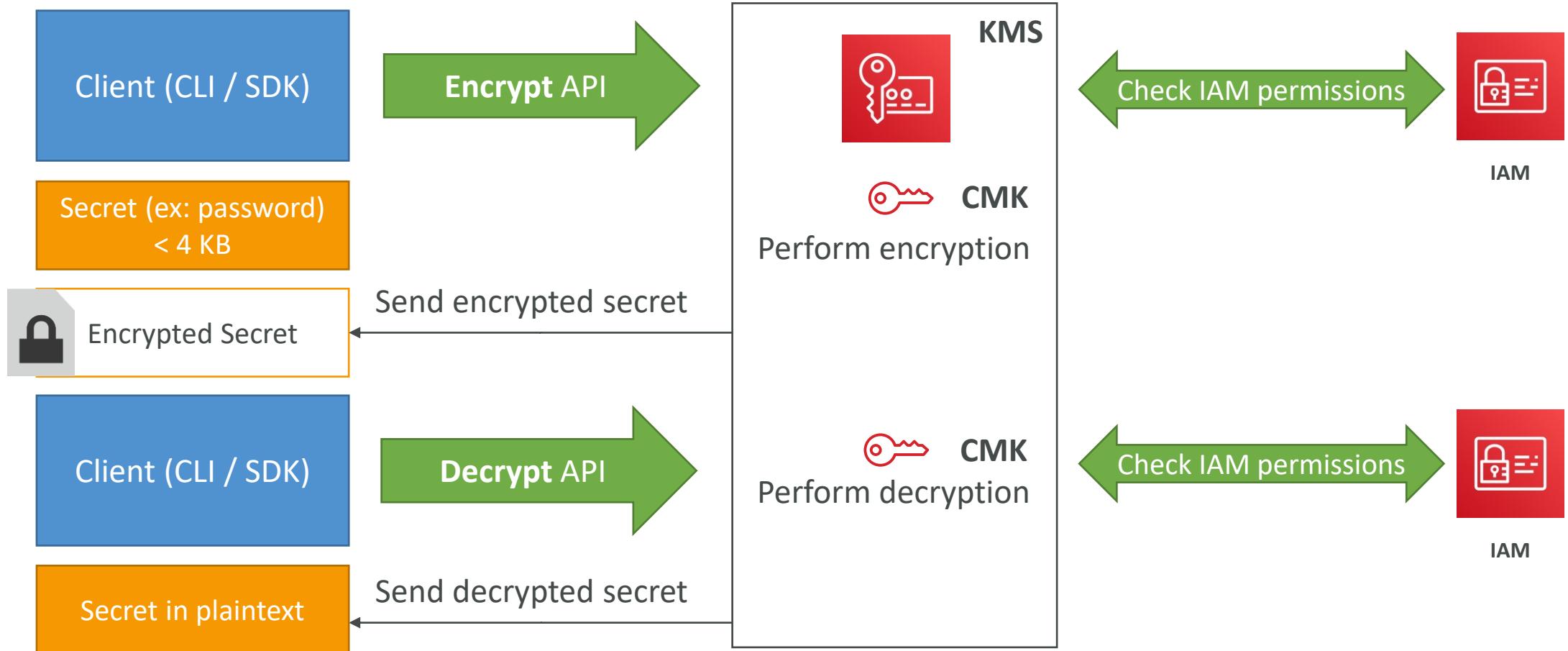
# AWS KMS 101

- Anytime you need to share sensitive information... use KMS
  - Database passwords
  - Credentials to external service
  - Private Key of SSL certificates
- The value in KMS is that the CMK used to encrypt data can never be retrieved by the user, and the CMK can be rotated for extra security
- **Never ever store your secrets in plaintext, especially in your code!**
- Encrypted secrets can be stored in the code / environment variables
- **KMS can only help in encrypting up to 4KB of data per call**
- If data > 4 KB, use envelope encryption
- To give access to KMS to someone:
  - Make sure the Key Policy allows the user
  - Make sure the IAM Policy allows the API calls

# AWS KMS (Key Management Service)

- Able to fully manage the keys & policies:
  - Create
  - Rotation policies
  - Disable
  - Enable
- Able to audit key usage (using CloudTrail)
- Three types of Customer Master Keys (CMK):
  - AWS Managed Service Default CMK: **free**
  - User Keys created in KMS: **\$1 / month**
  - User Keys imported (must be 256-bit symmetric key): **\$1 / month**
- + pay for API call to KMS (**\$0.03 / 10000 calls**)

# How does KMS work? API – Encrypt and Decrypt

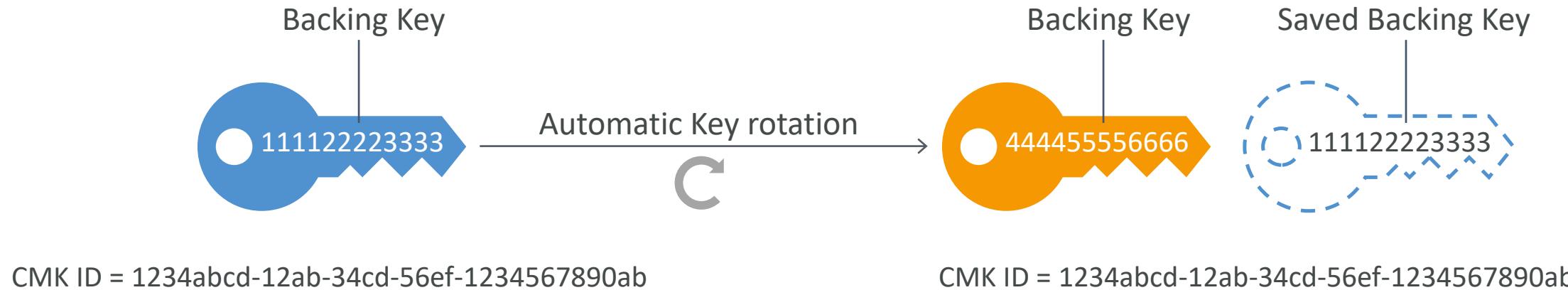


# Encryption in AWS Services

- Requires migration (through Snapshot / Backup):
  - EBS Volumes
  - RDS databases
  - ElastiCache
  - EFS network file system
- In-place encryption:
  - S3

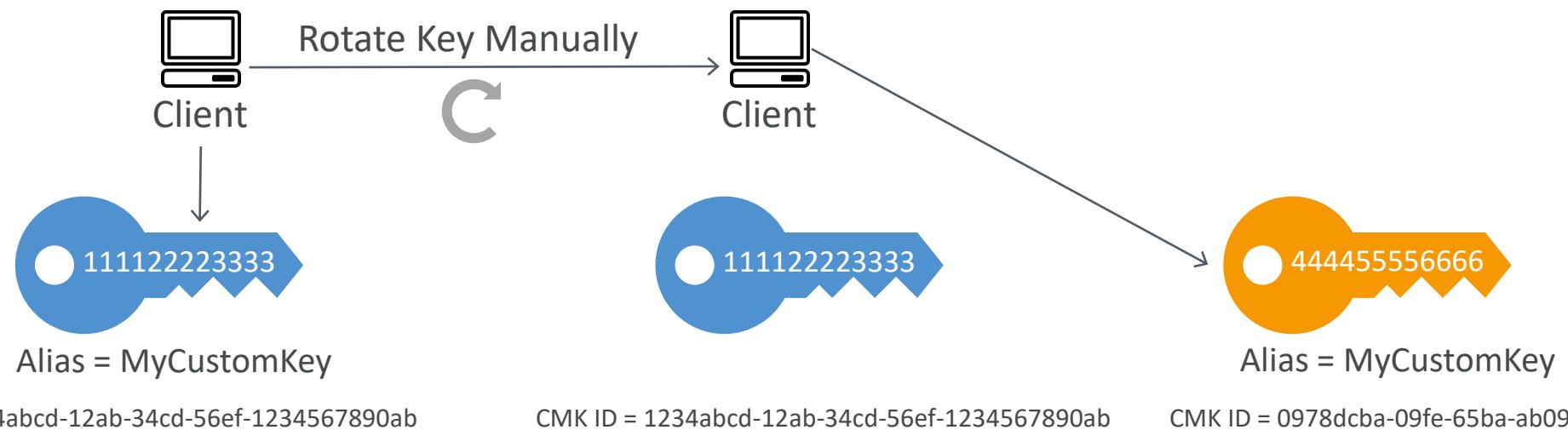
# KMS Automatic Key Rotation

- For Customer-managed CMK (not AWS managed CMK)
- If enabled: automatic key rotation happens every 1 year
- Previous key is kept active so you can decrypt old data
- New Key has the same CMK ID (only the backing key is changed)



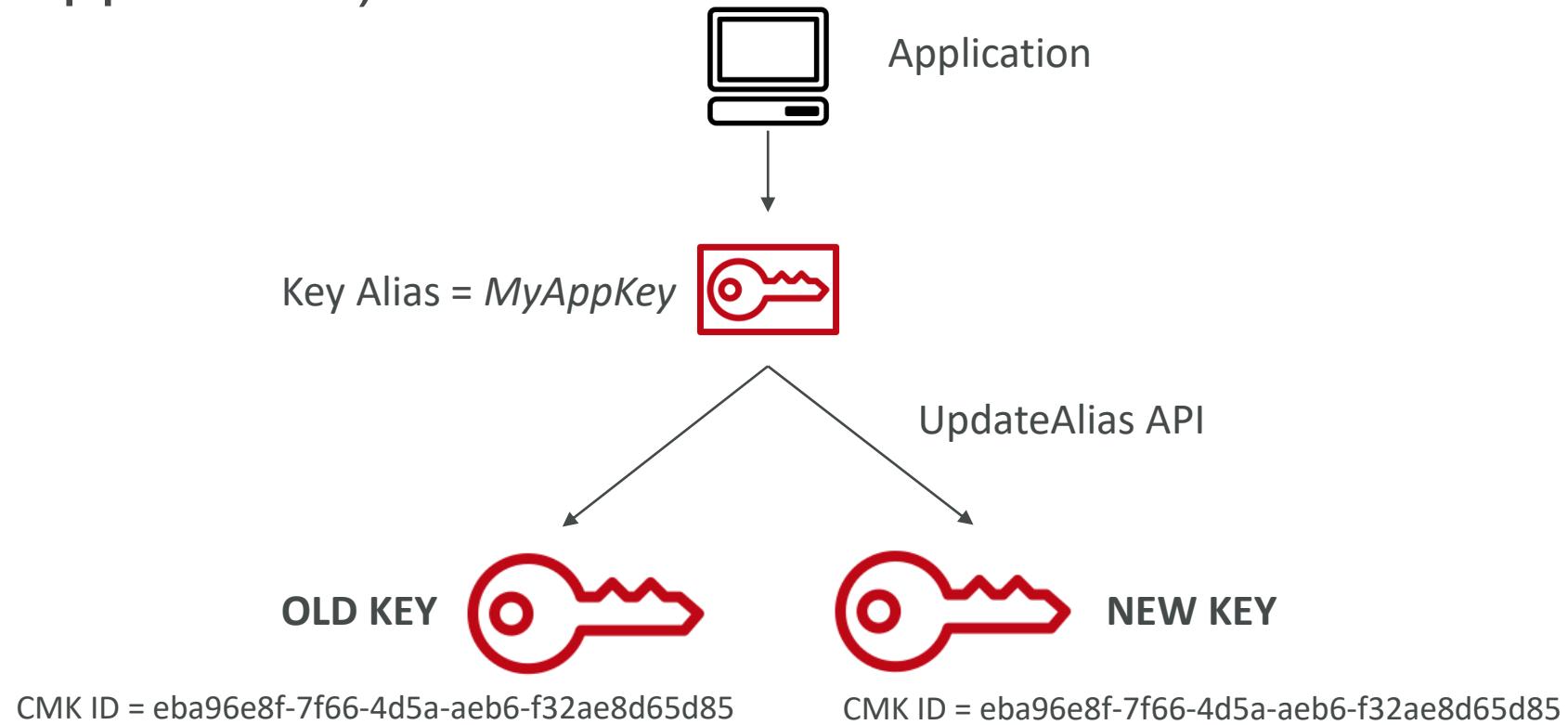
# KMS Manual Key Rotation

- When you want to rotate key every 90 days, 180 days, etc...
- New Key has a different CMK ID
- Keep the previous key active so you can decrypt old data
- Better to use aliases in this case (to hide the change of key for the application)
- Good solution to rotate CMK that are not eligible for automatic rotation (like asymmetric CMK)



# KMS Alias Updating

- Better to use aliases in this case (to hide the change of key for the application)

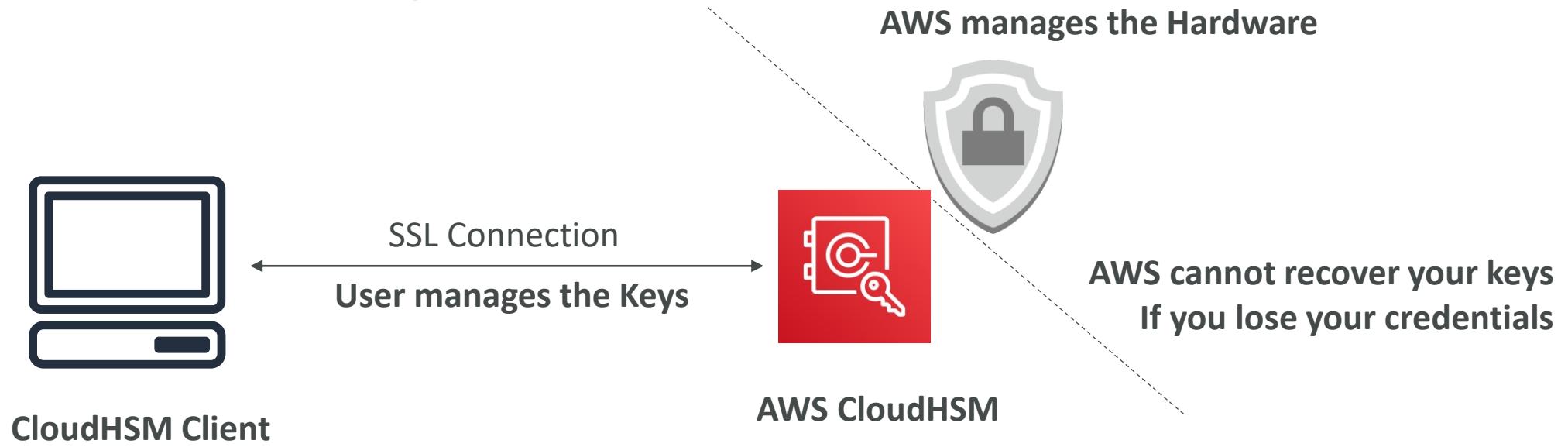


# CloudHSM



- KMS => AWS manages the software for encryption
- CloudHSM => AWS provisions encryption **hardware**
- Dedicated Hardware (HSM = Hardware Security Module)
- You manage your own encryption keys entirely (not AWS)
- HSM device is tamper resistant, FIPS 140-2 Level 3 compliance
- **CloudHSM clusters are spread across Multi AZ (HA) – must setup**
- Supports both symmetric and **asymmetric** encryption (SSL/TLS keys)
- No free tier available
- Must use the CloudHSM Client Software
- Redshift supports CloudHSM for database encryption and key management
- Good option to use with SSE-C encryption

# CloudHSM Diagram



## IAM permissions:

- CRUD an HSM Cluster

## CloudHSM Software:

- Manage the Keys
- Manage the Users



# Security - Kinesis

- Kinesis Data Streams
  - SSL endpoints using the HTTPS protocol to do encryption in flight
  - AWS KMS provides server-side encryption [Encryption at rest]
  - For client side-encryption, you **must** use your own encryption libraries
  - Supported Interface VPC Endpoints / Private Link – access privately
  - KCL – must get read / write access to DynamoDB table
- Kinesis Data Firehose:
  - Attach IAM roles so it can deliver to S3 / ES / Redshift / Splunk
  - Can encrypt the delivery stream with KMS [Server side encryption]
  - Supported Interface VPC Endpoints / Private Link – access privately
- Kinesis Data Analytics
  - Attach IAM role so it can read from Kinesis Data Streams and reference sources and write to an output destination (example Kinesis Data Firehose)



# Security - SQS

- Encryption in flight using the HTTPS endpoint
- Server Side Encryption using KMS
- IAM policy must allow usage of SQS
- SQS queue access policy
  
- Client-side encryption must be implemented manually
- VPC Endpoint is provided through an Interface

# Security – AWS IoT



- AWS IoT policies:
  - Attached to X.509 certificates or Cognito Identities
  - Able to revoke any device at any time
  - IoT Policies are JSON documents
  - Can be attached to groups instead of individual Things.
- IAM Policies:
  - Attached to users, group or roles
  - Used for controlling IoT AWS APIs
- Attach roles to Rules Engine so they can perform their actions



# Security – Amazon S3

- IAM policies
- S3 bucket policies
- Access Control Lists (ACLs)
- Encryption in flight using HTTPS
- Encryption at rest
  - Server-side encryption: SSE-S3, SSE-KMS, SSE-C
  - Client-side encryption – such as Amazon S3 Encryption Client
- Versioning + MFA Delete
- CORS for protecting websites
- VPC Endpoint is provided through a Gateway
- Glacier – vault lock policies to prevent deletes (WORM)



# Security – DynamoDB

- Data is encrypted in transit using TLS (HTTPS)
- DynamoDB tables are encrypted at rest
  - KMS encryption for base tables and secondary indexes
  - AWS owned key (default)
  - AWS managed key (aws/dynamodb)
  - AWS customer managed key (your own)
- Access to tables / API / DAX using IAM
- DynamoDB Streams are encrypted
- VPC Endpoint is provided through a Gateway



# Security - RDS

- VPC provides network isolation
- Security Groups control network access to DB Instances
- KMS provides encryption at rest
- SSL provides encryption in-flight
- IAM policies provide protection for the RDS API
- IAM authentication is supported by PostgreSQL and MySQL
- Must manage user permissions within the database itself
- MSSQL Server and Oracle support TDE (Transparent Data Encryption)



# Security - Aurora

- (very similar to RDS)
- VPC provides network isolation
- Security Groups control network access to DB Instances
- KMS provides encryption at rest
- SSL provides encryption in-flight
- IAM authentication is supported by PostgreSQL and MySQL
- Must manage user permissions within the database itself



# Security - Lambda

- IAM roles attached to each Lambda function
- Sources
- Targets
- KMS encryption for secrets
- SSM parameter store for configurations
- CloudWatch Logs
- Deploy in VPC to access private resources



# Security - Glue

- IAM policies for the Glue service
- Configure Glue to only access JDBC through SSL
- Data Catalog: Encrypted by KMS
- Connection passwords: Encrypted by KMS
- Data written by AWS Glue – Security Configurations:
  - S3 encryption mode: SSE-S3 or SSE-KMS
  - CloudWatch encryption mode
  - Job bookmark encryption mode



# Security - EMR

- Using Amazon EC2 key pair for SSH credentials
- Attach IAM roles to EC2 instances for:
  - proper S3 access
  - for EMRFS requests to S3
  - DynamoDB scans through Hive
- EC2 Security Groups
  - One for master node
  - Another one for cluster node (core node or task node)
- Encrypts data at-rest: EBS encryption, Open Source HDFS Encryption, LUKS + EMRFS for S3
- In-transit encryption: node to node communication, EMRFS, TLS
- Data is encrypted before uploading to S3
- Kerberos authentication (provide authentication from Active Directory)
- Apache Ranger: Centralized Authorization (RBAC – Role Based Access) – setup on external EC2
- <https://aws.amazon.com/blogs/big-data/best-practices-for-securin>



# Security – ElasticSearch Service

- Amazon VPC provides network isolation
- ElasticSearch policy to manage security further
- Data security by encrypting data at-rest using KMS
- Encryption in-transit using SSL
  
- IAM or Cognito based authentication
- Amazon Cognito allow end-users to log-in to Kibana through enterprise identity providers such as Microsoft Active Directory using SAML

# Security - Redshift



- VPC provides network isolation
- Cluster security groups
- Encryption in flight using the JDBC driver enabled with SSL
- Encryption at rest using KMS or an HSM device (establish a connection)
- Supports S3 SSE using default managed key
- Use IAM Roles for Redshift
- To access other AWS Resources (example S3 or KMS)
- Must be referenced in the COPY or UNLOAD command (alternatively paste access key and secret key creds)

# Security - Athena



- IAM policies to control access to the service
- Data is in S3: IAM policies, bucket policies & ACLs
- Encryption of data according to S3 standards: SSE-S3, SSE-KMS, CSE-KMS
- Encryption in transit using TLS between Athena and S3 and JDBC
- Fine grained access using the AWS Glue Catalog



# Security - Quicksight

- Standard edition:
  - IAM users
  - Email based accounts
- Enterprise edition:
  - Active Directory
  - Federated Login
  - Supports MFA (Multi Factor Authentication)
  - Encryption at rest and in SPICE
- Row Level Security to control which users can see which rows

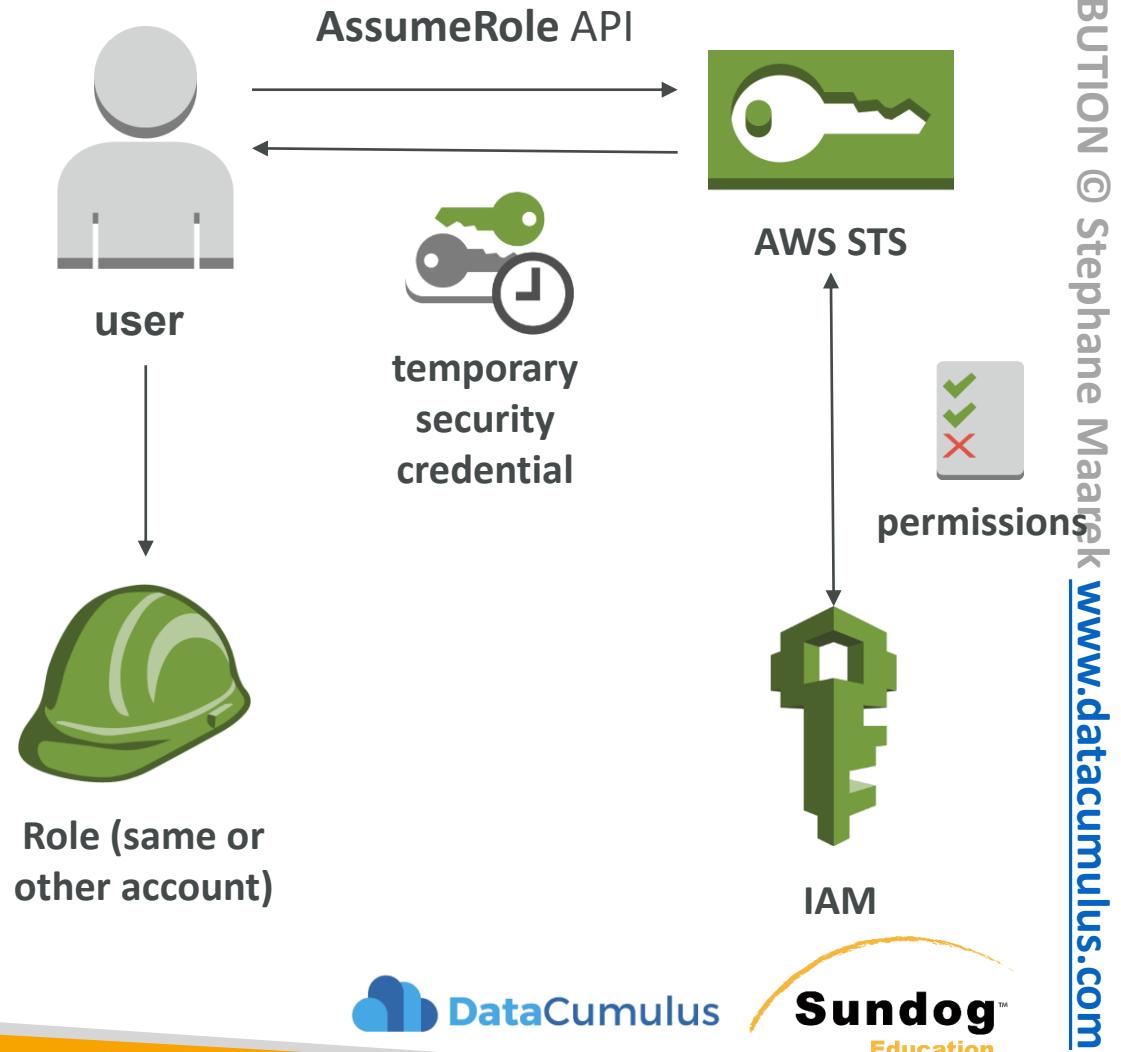
# AWS STS – Security Token Service



- Allows to grant limited and temporary access to AWS resources.
- Token is valid for up to one hour (must be refreshed)
- **Cross Account Access**
  - Allows users from one AWS account access resources in another
- **Federation (Active Directory)**
  - Provides a non-AWS user with temporary AWS access by linking users Active Directory credentials
  - Uses SAML (Security Assertion markup language)
  - Allows Single Sign On (SSO) which enables users to log in to AWS console without assigning IAM credentials
- **Federation with third party providers / Cognito**
  - Used mainly in web and mobile applications
  - Makes use of Facebook/Google/Amazon etc to federate them

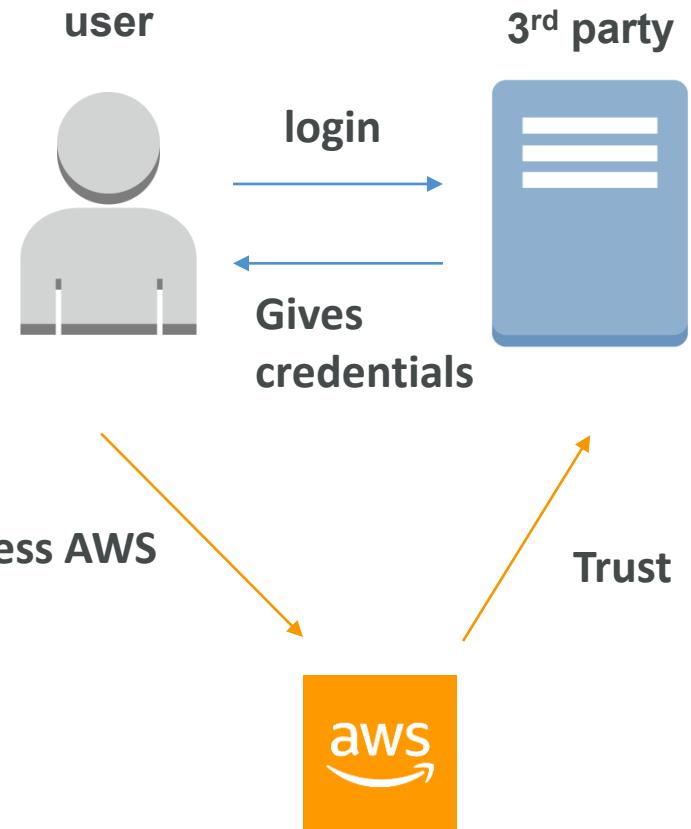
# Cross Account Access

- Define an IAM Role for another account to access
- Define which accounts can access this IAM Role
- Use AWS STS (Security Token Service) to retrieve credentials and impersonate the IAM Role you have access to (**AssumeRole API**)
- Temporary credentials can be valid between 15 minutes to 1 hour



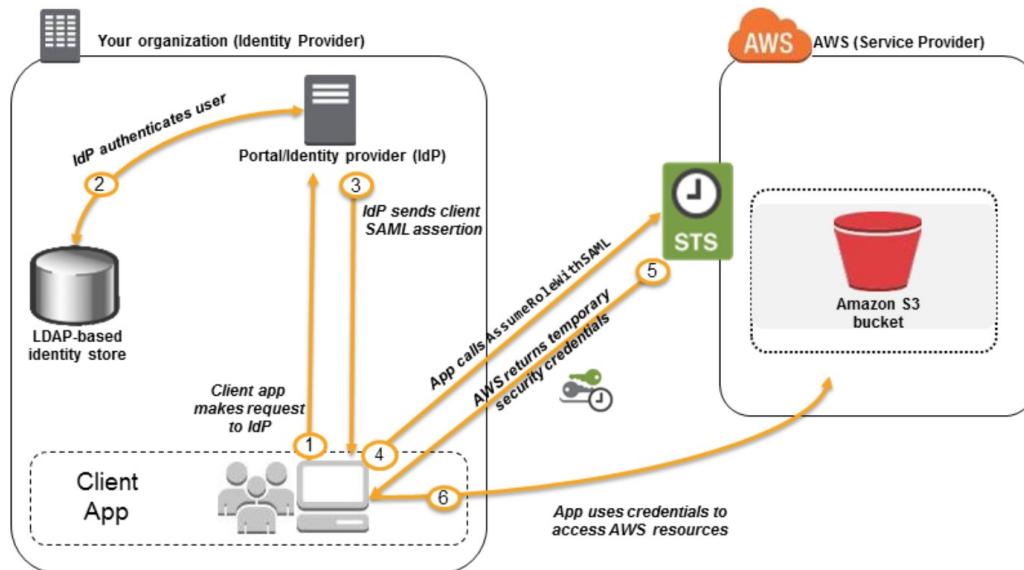
# What's Identity Federation?

- Federation lets users outside of AWS to assume temporary role for accessing AWS resources.
- These users assume identity provided access role.
- **Federation assumes a form of 3rd party authentication**
  - LDAP
  - Microsoft Active Directory (~= SAML)
  - Single Sign On
  - Open ID
  - Cognito
- **Using federation, you don't need to create IAM users (user management is outside of AWS)**

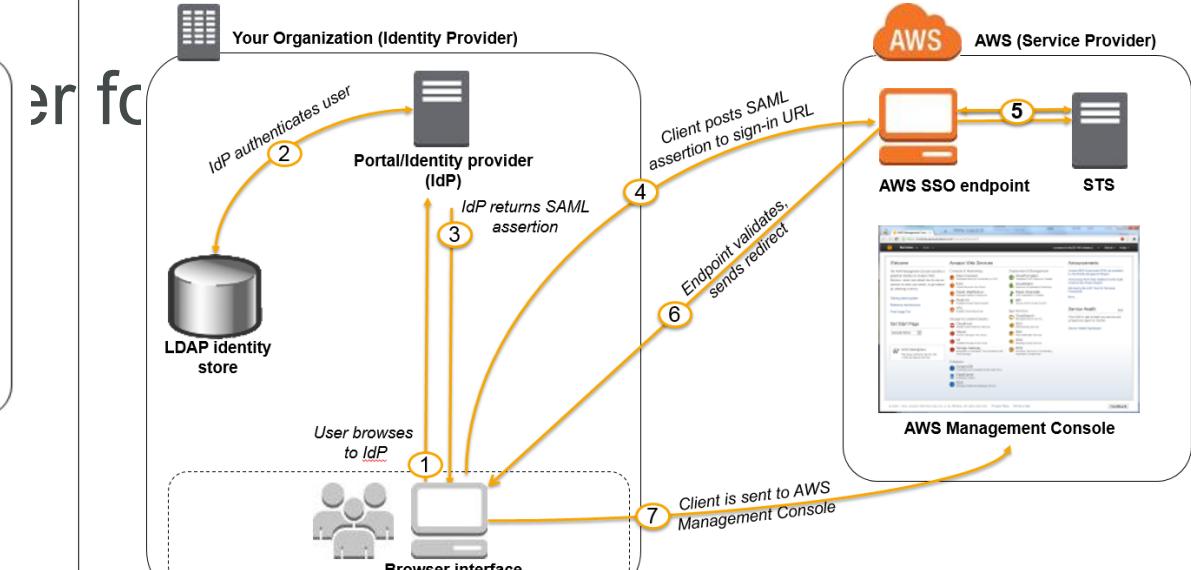


# SAML Federation For Enterprises

- To integrate Active Directory / ADFS with AWS (or any SAML 2.0)
- Provides access to AWS Console or CLI (through temporary security credentials)



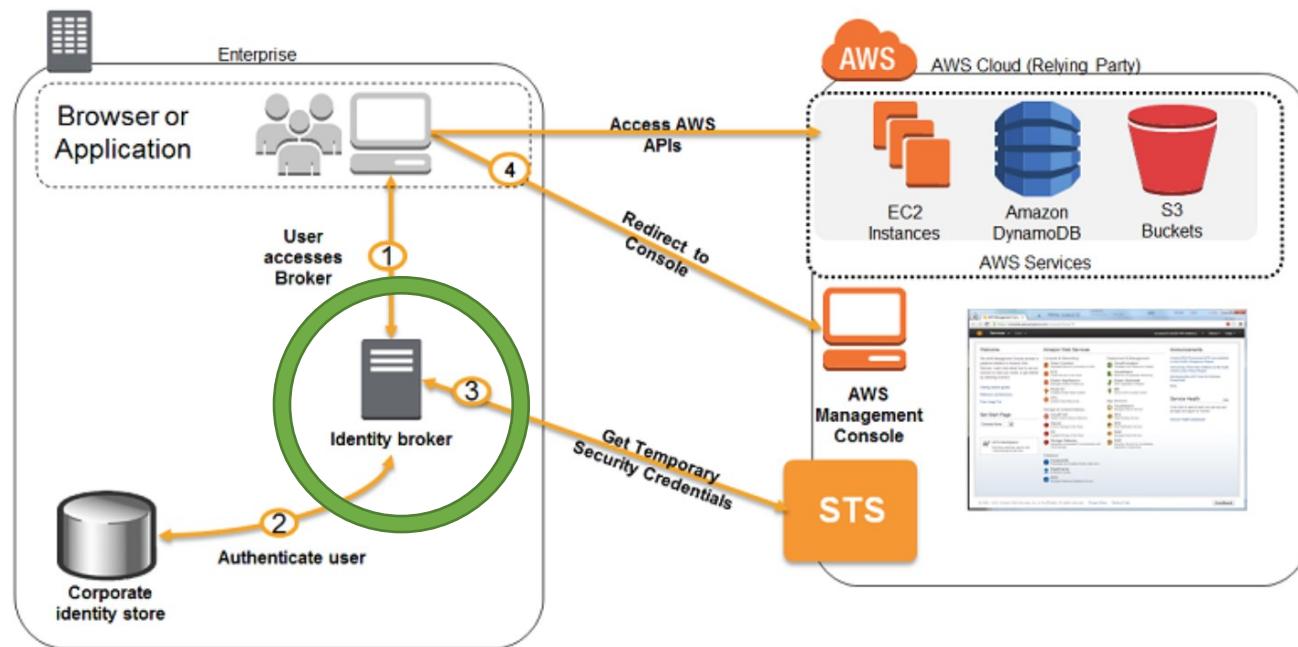
[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_providers\\_saml.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml.html)



[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_providers\\_enable-console-saml.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_enable-console-saml.html)

# Custom Identity Broker Application For Enterprises

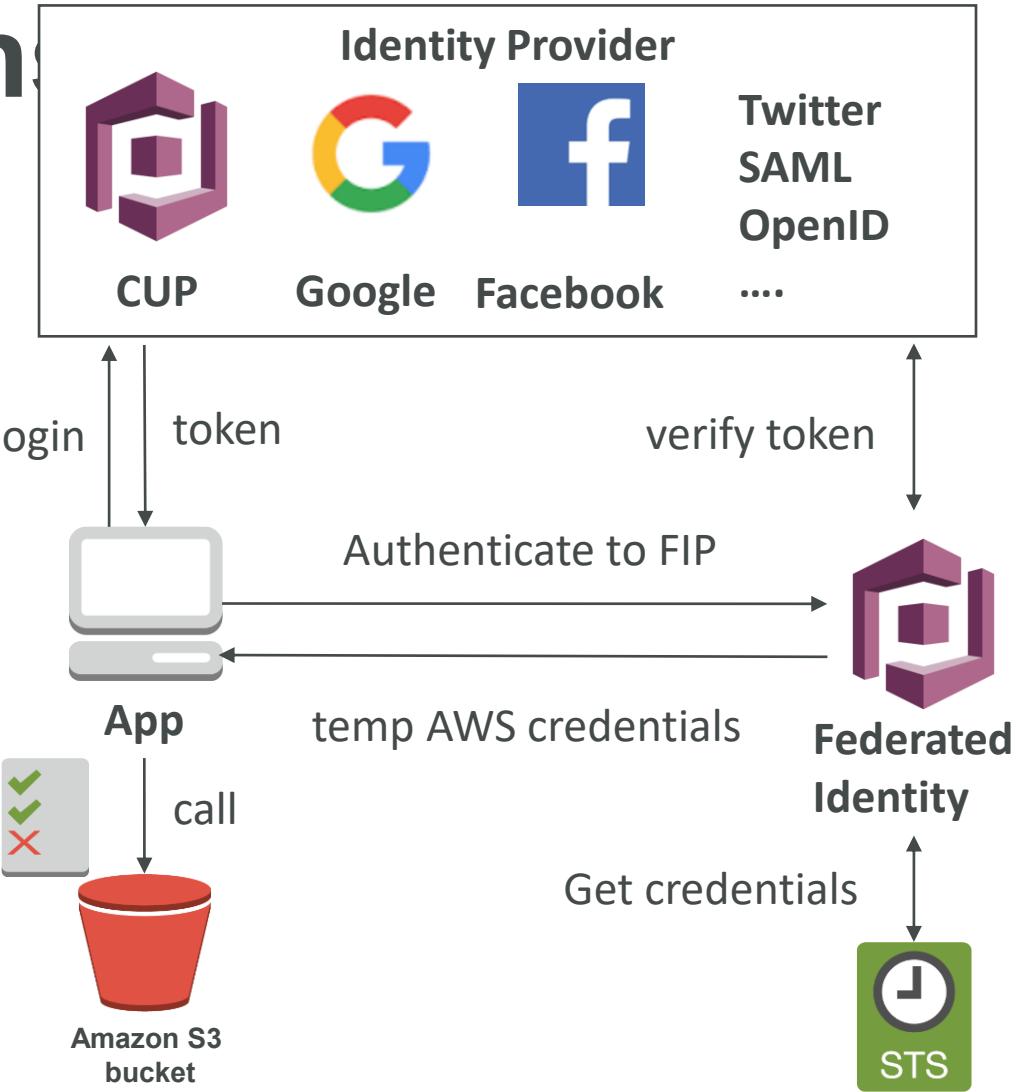
- Use only if identity provider is not compatible with SAML 2.0
- **The identity broker must determine the appropriate IAM policy**



[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles\\_common-scenarios\\_federated-users.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_common-scenarios_federated-users.html)

# AWS Cognito - Federated Identity Pools For Public Applications

- **Goal:**
  - Provide direct access to AWS Resources from the Client Side
- **How:**
  - Log in to federated identity provider – or remain anonymous
  - Get temporary AWS credentials back from the Federated Identity Pool
  - These credentials come with a pre-defined IAM policy stating their permissions
- **Example:**
  - provide (temporary) access to write to S3 bucket using Facebook Login
- **Note:**
  - Web Identity Federation is an alternative to using Cognito but AWS recommends against it



# Policies – leveraging AWS variables

- [https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_variables.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html)
  - \${aws:username} : to restrict users to tables / buckets
  - \${aws:principaltype} : account, user, federated, or assumed role
  - \${aws:PrincipalTag/department} : to restrict using Tags
- [https://docs.aws.amazon.com/IAM/latest/UserGuide/reference\\_policies\\_iam-condition-keys.html#condition-keys-wif](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_iam-condition-keys.html#condition-keys-wif)
  - \${aws:FederatedProvider} : which IdP was used for the user (Cognito, Amazon..)
  - \${www.amazon.com:user\_id} , \${cognito-identity.amazonaws.com:sub} ...
  - \${saml:sub}, \${sts:ExternalId}

# Policies - Advanced

- For S3 - let's analyze the policies at:  
<https://docs.aws.amazon.com/AmazonS3/latest/dev/example-bucket-policies.html>
- For DynamoDB – let's analyze the policies at:  
<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/specifying-conditions.html>
- Note for RDS – IAM policies don't help with **in-database** security, as it's a proprietary technology and we are responsible for users & authorization

# AWS CloudTrail



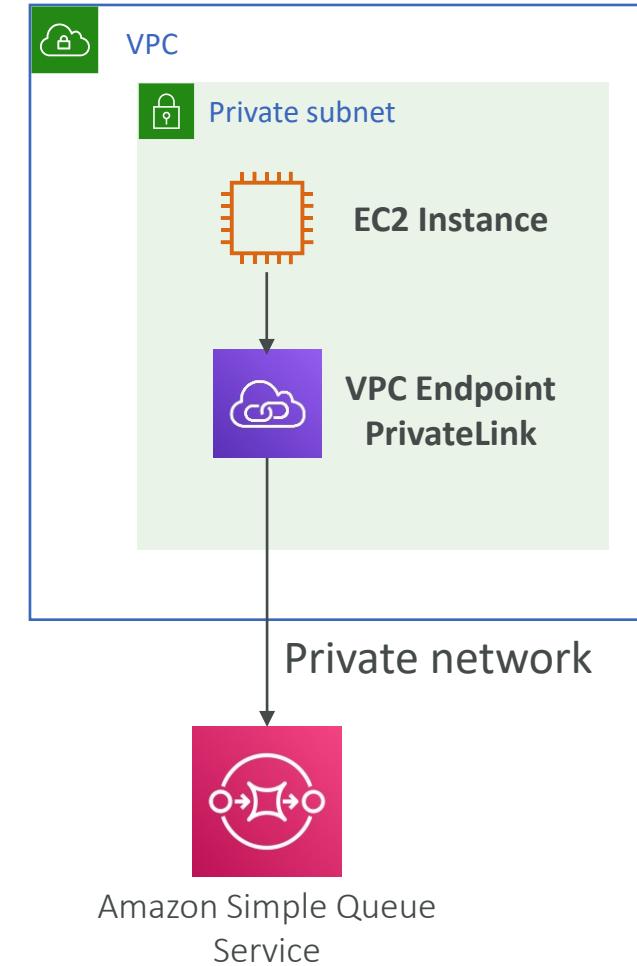
- Provides governance, compliance and audit for your AWS Account
- CloudTrail is enabled by default!
- Get an history of events / API calls made within your AWS Account by:
  - Console
  - SDK
  - CLI
  - AWS Services
- Can put logs from CloudTrail into CloudWatch Logs
- If a resource is deleted in AWS, look into CloudTrail first!

# CloudTrail continued...

- CloudTrail shows the past 90 days of activity
- The default UI only shows “Create”, “Modify” or “Delete” events
- **CloudTrail Trail:**
  - Get a detailed list of all the events you choose
  - Ability to store these events in S3 for further analysis
  - Can be region specific or global
- CloudTrail Logs have SSE-S3 encryption when placed into S3
- Control access to S3 using IAM, Bucket Policy, etc...

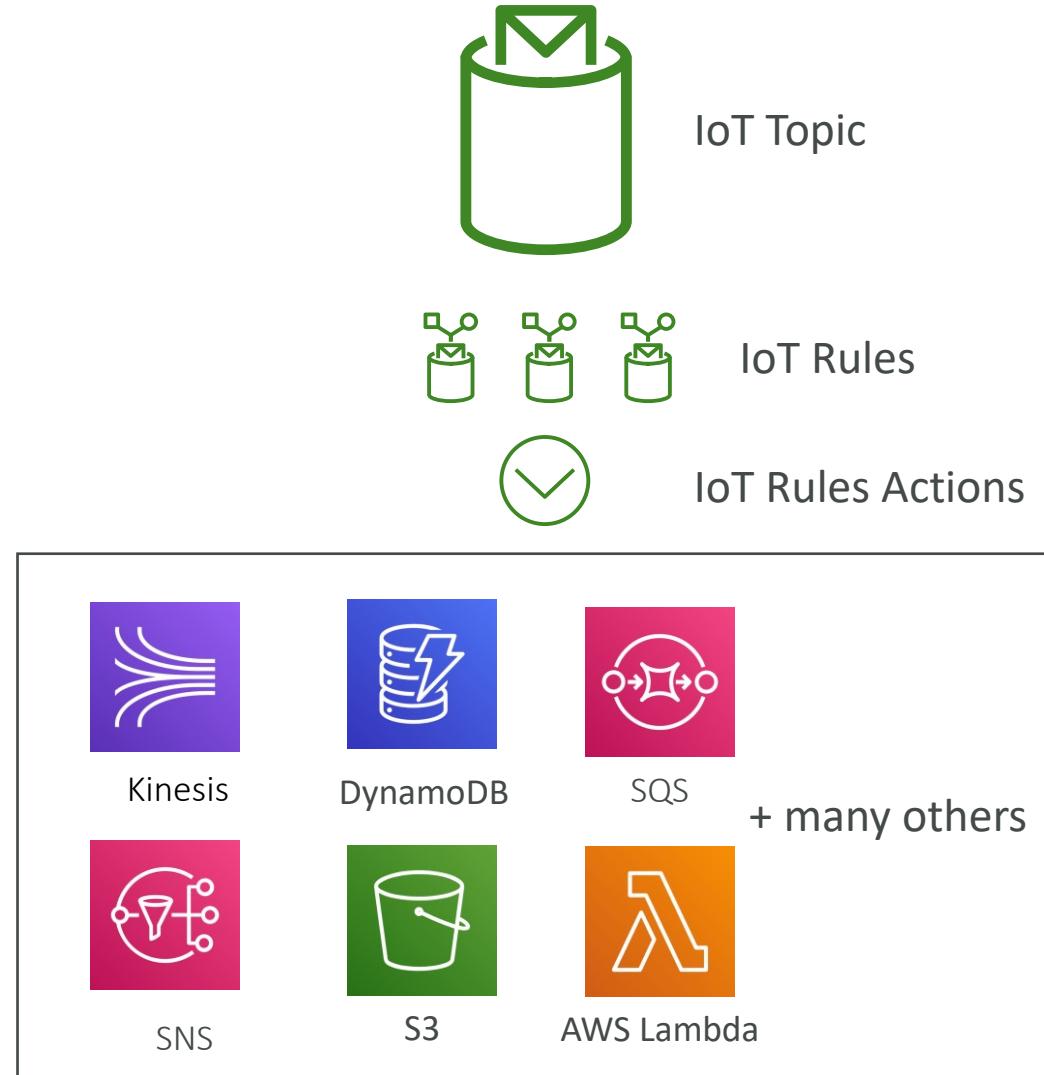
# VPC Endpoints

- Endpoints allow you to connect to AWS Services using a private network instead of the public www network
- They scale horizontally and are redundant
- They remove the need of IGW, NAT, etc... to access AWS Services
- **Gateway:** provisions a target and must be used in a route table  
ONLY S3 and DynamoDB
- **Interface:** provisions an ENI (private IP address) as an entry point (must attach security group) – most AWS services  
Also called VPC PrivateLink

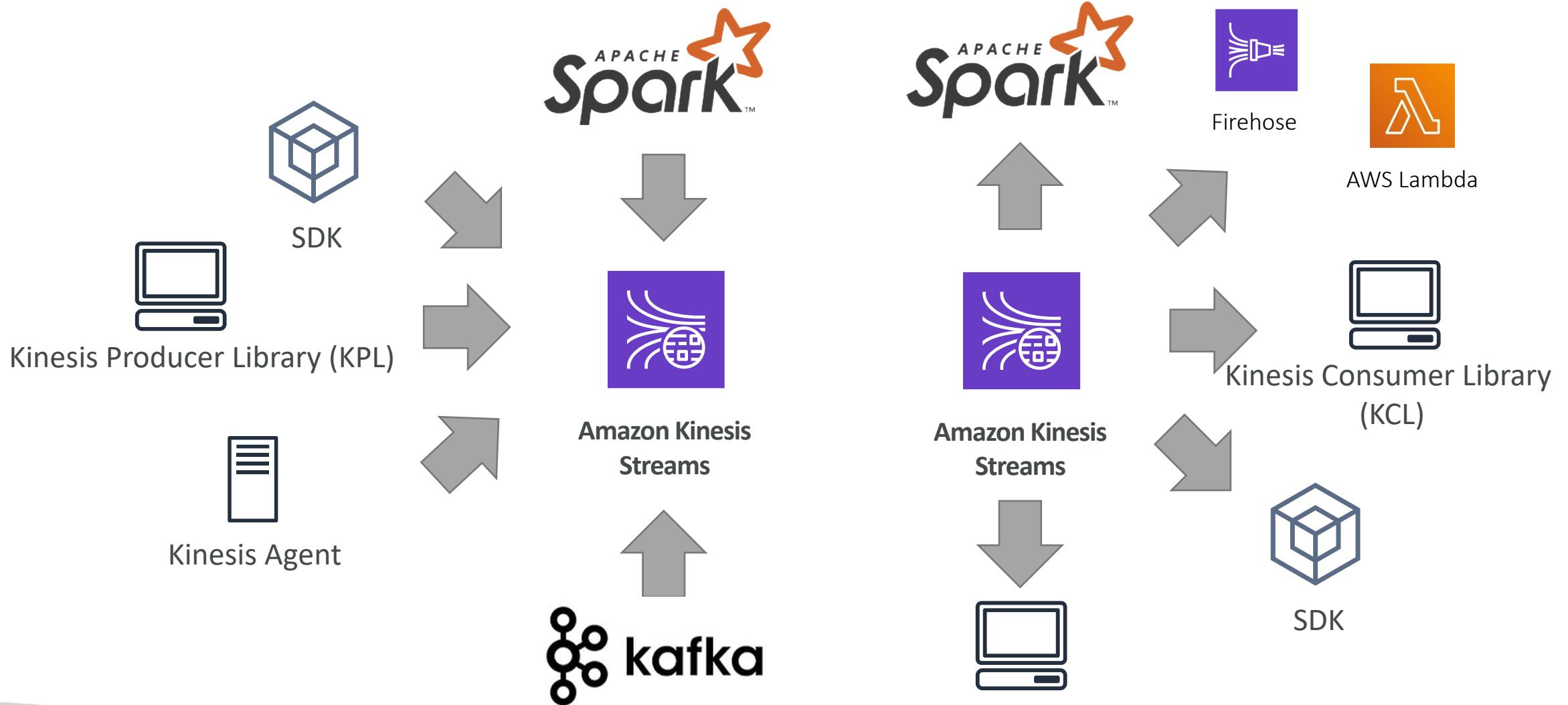


# Everything Else

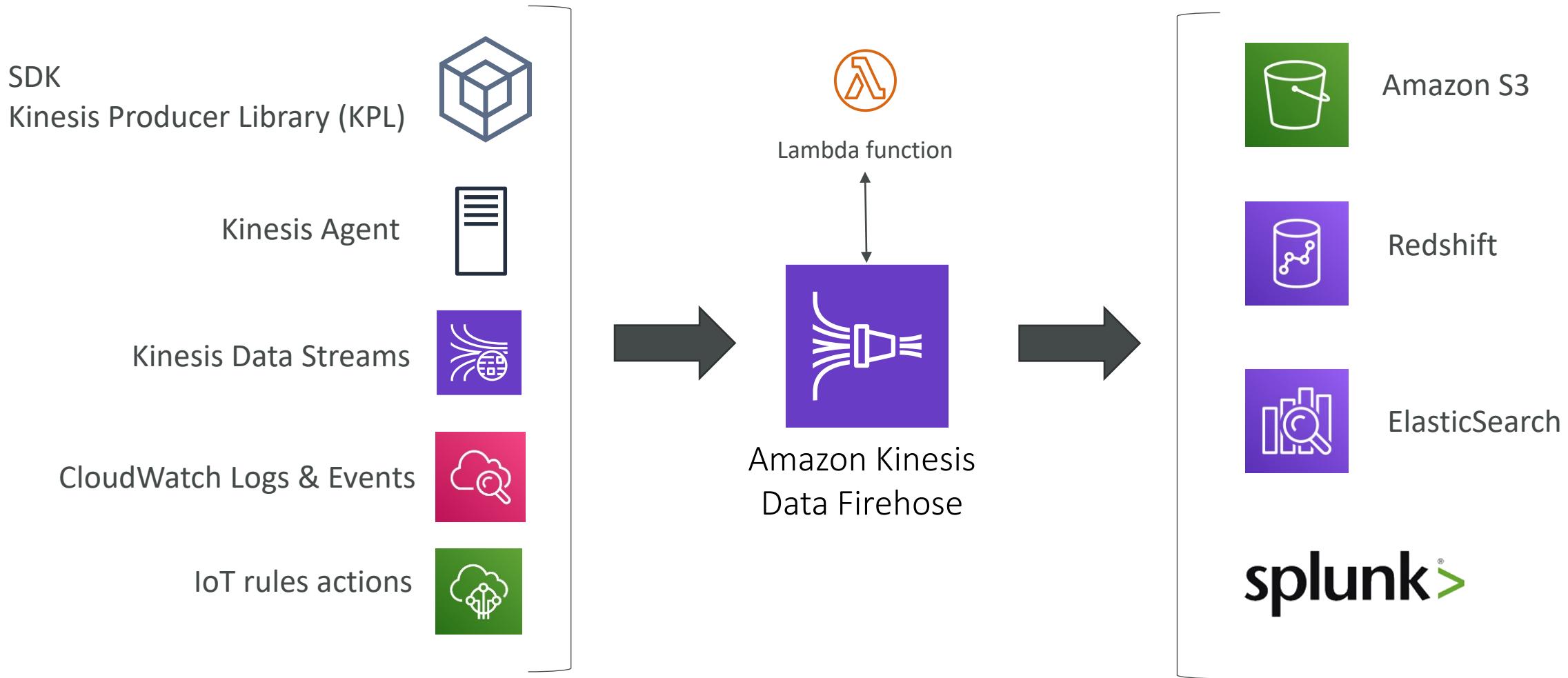
# IoT



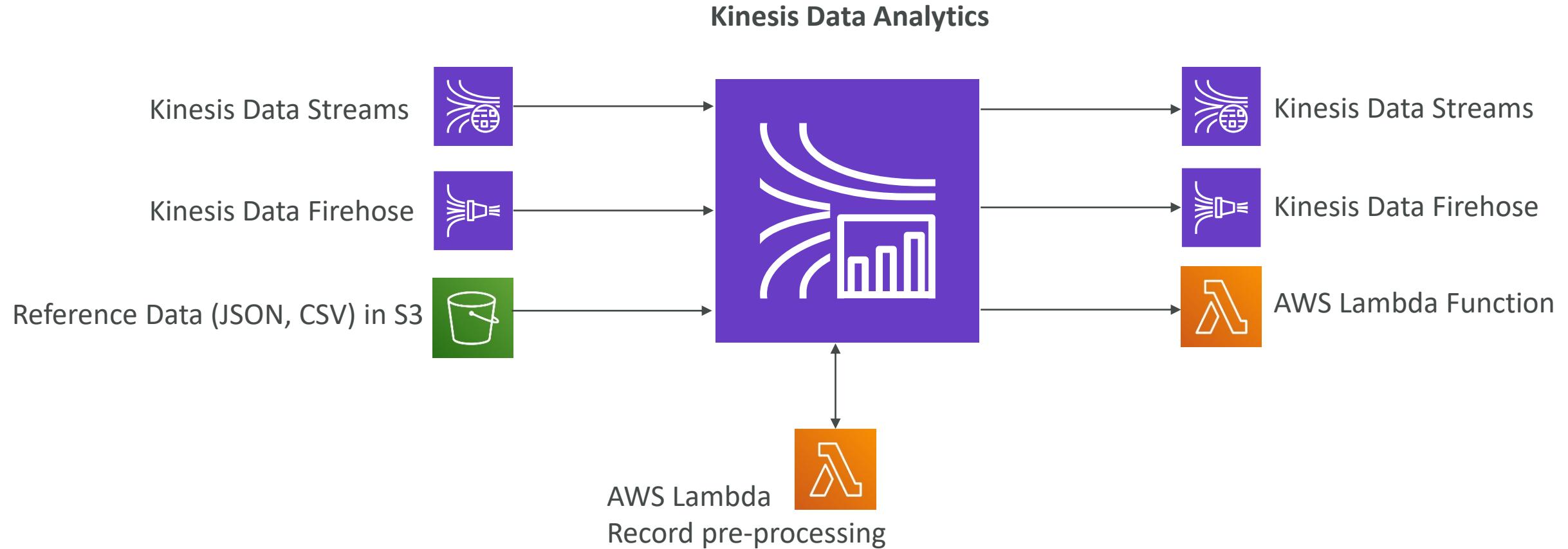
# Kinesis Data Streams



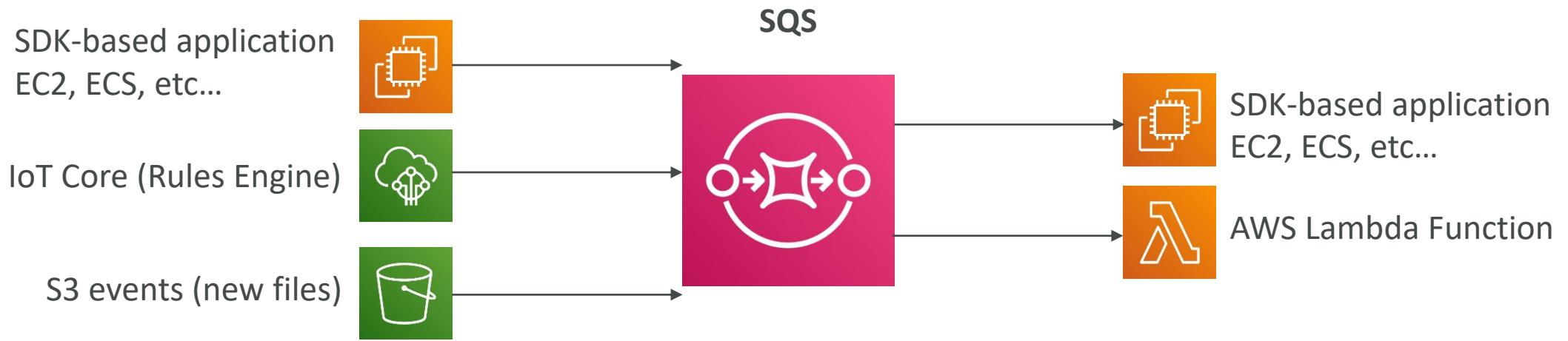
# Kinesis Data Firehose



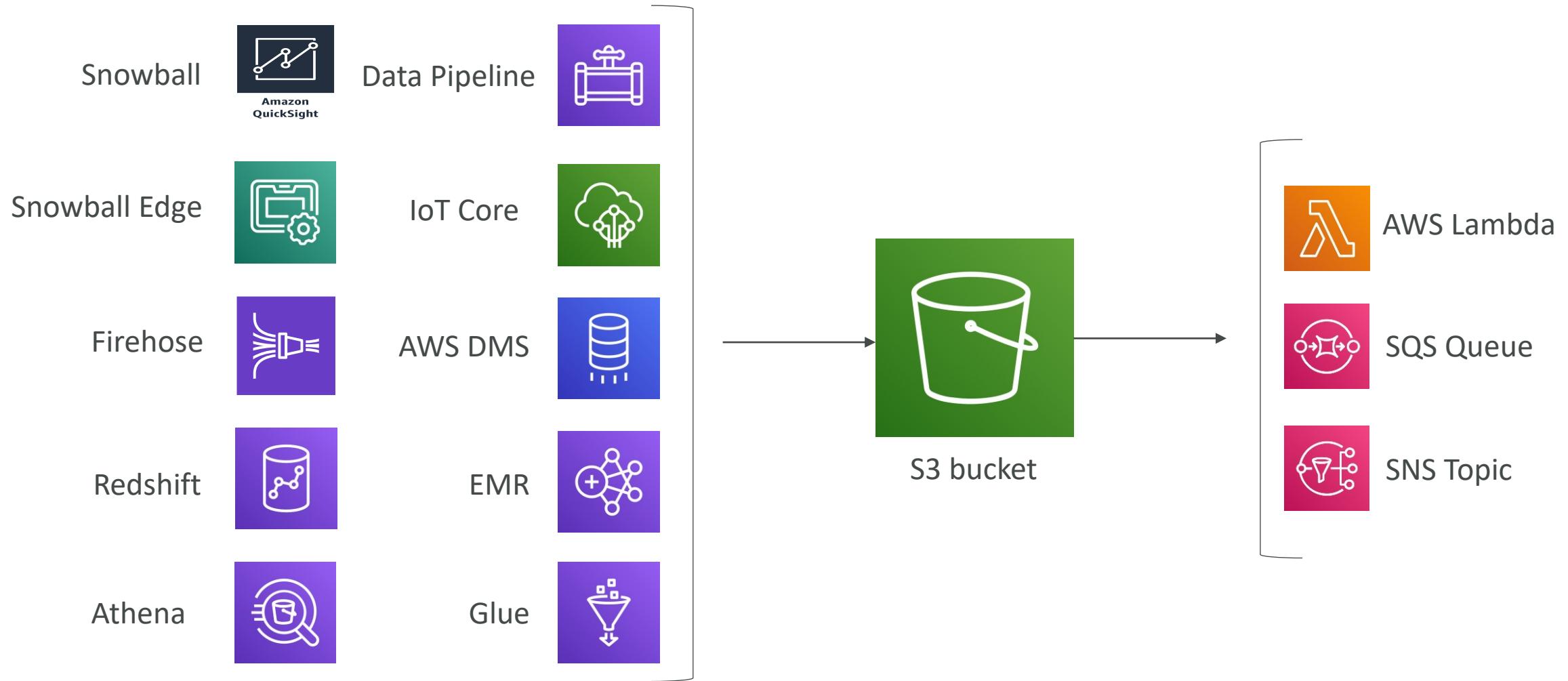
# Kinesis Data Analytics



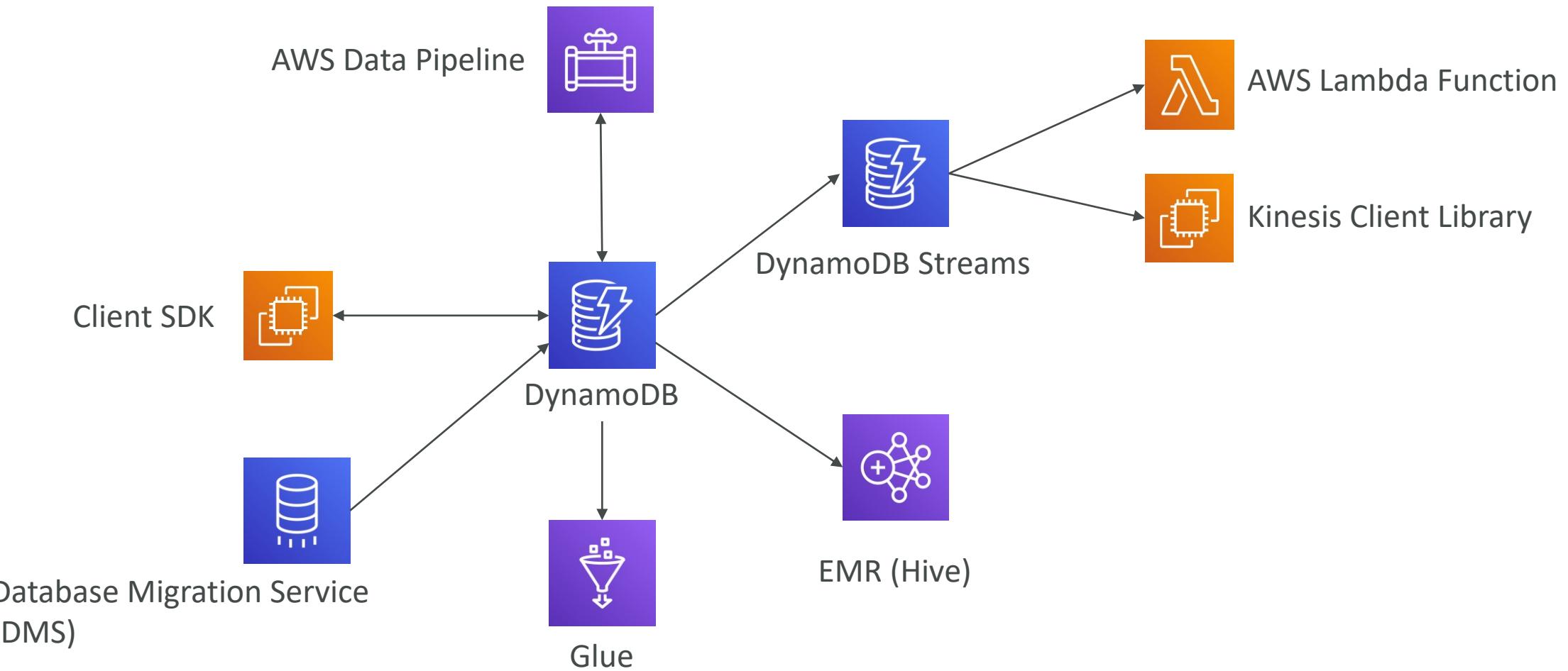
# SQS



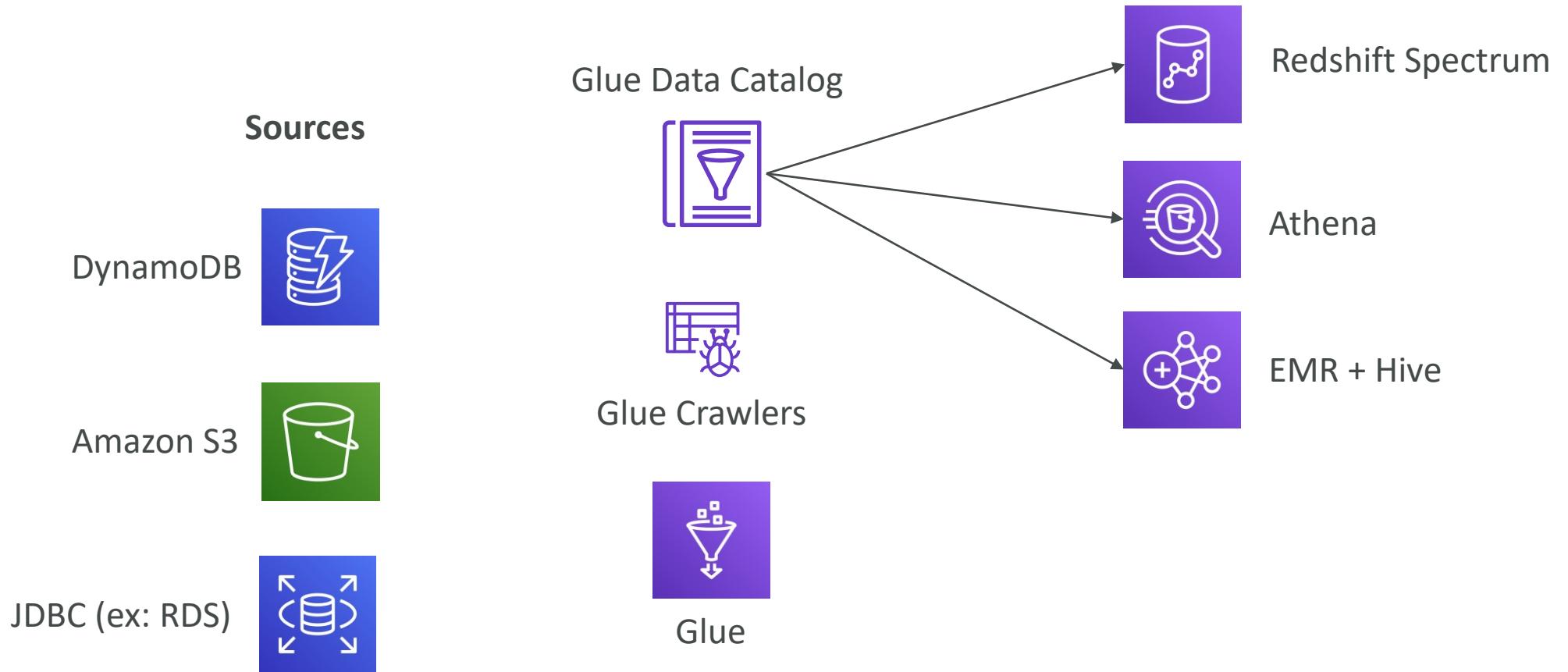
# S3



# DynamoDB



# Glue



# EMR

Glue Data Catalog



Amazon S3 / EMRFS



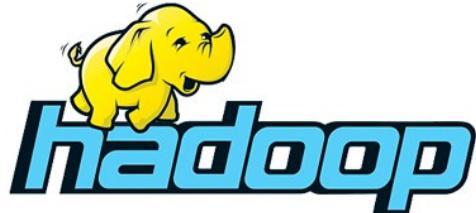
DynamoDB



Apache Ranger on EC2

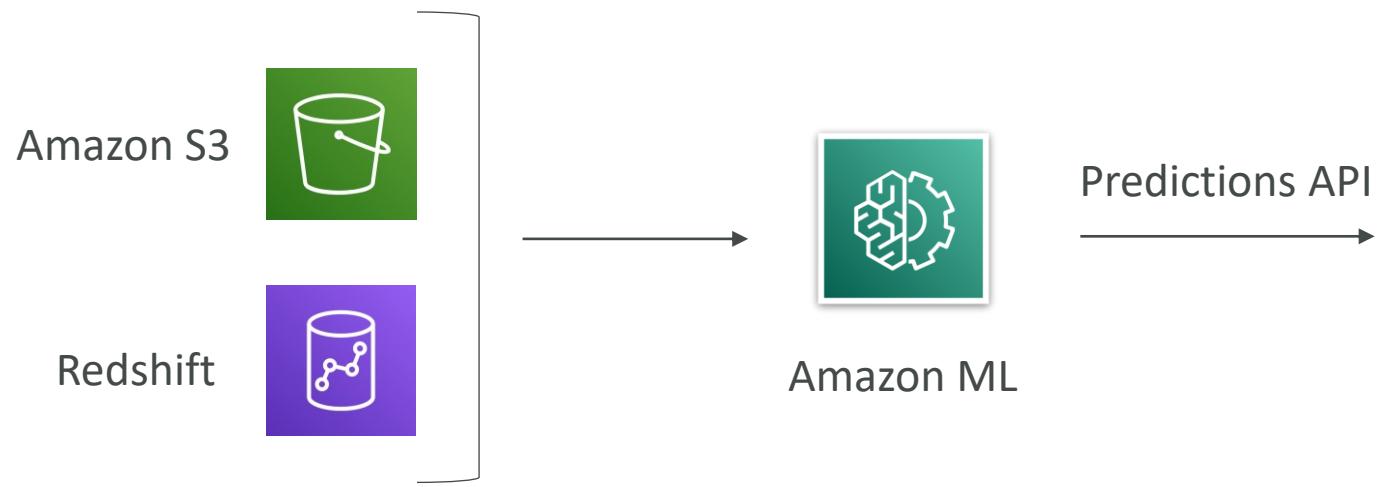


EMR

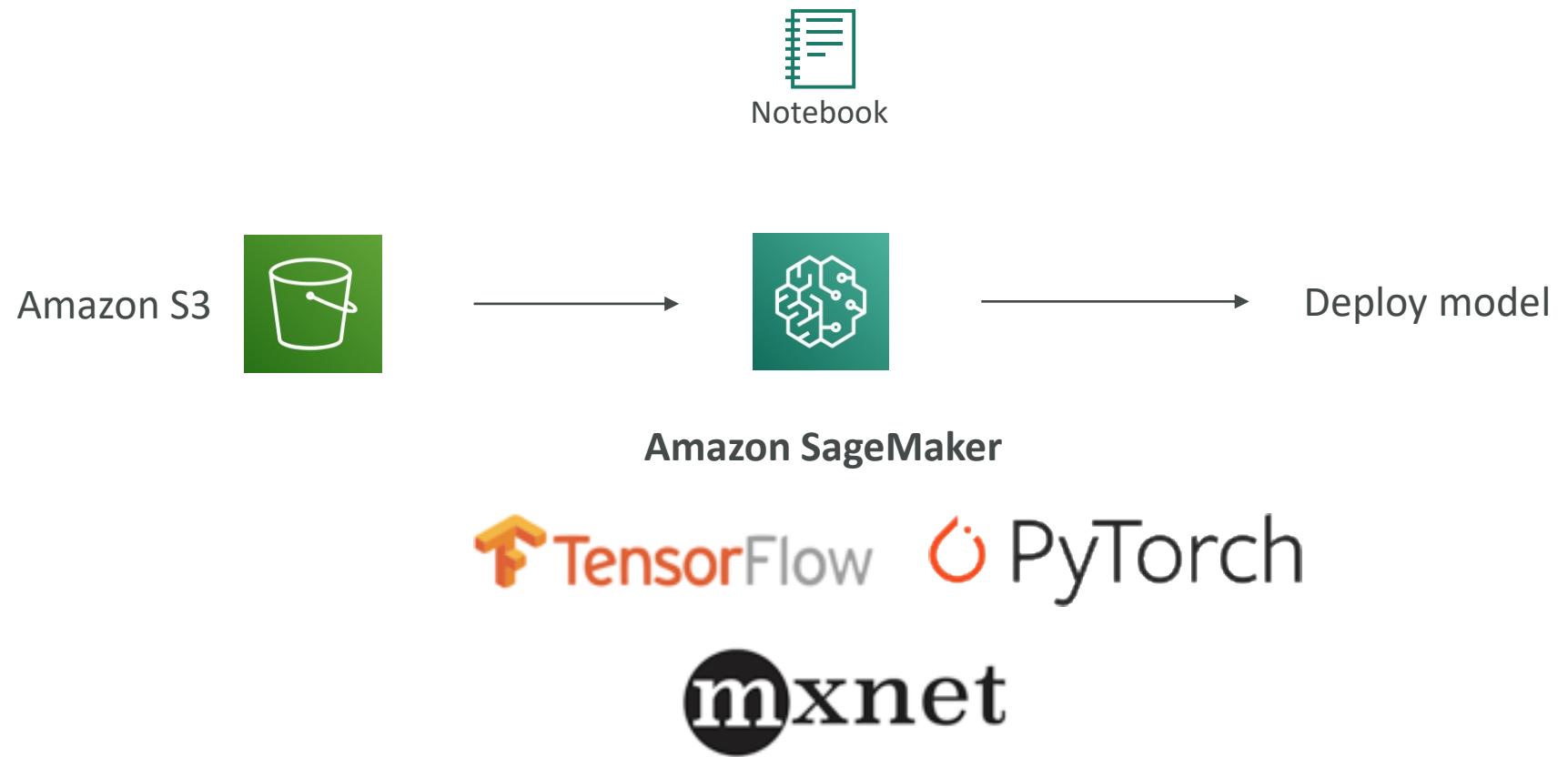


**Flink**

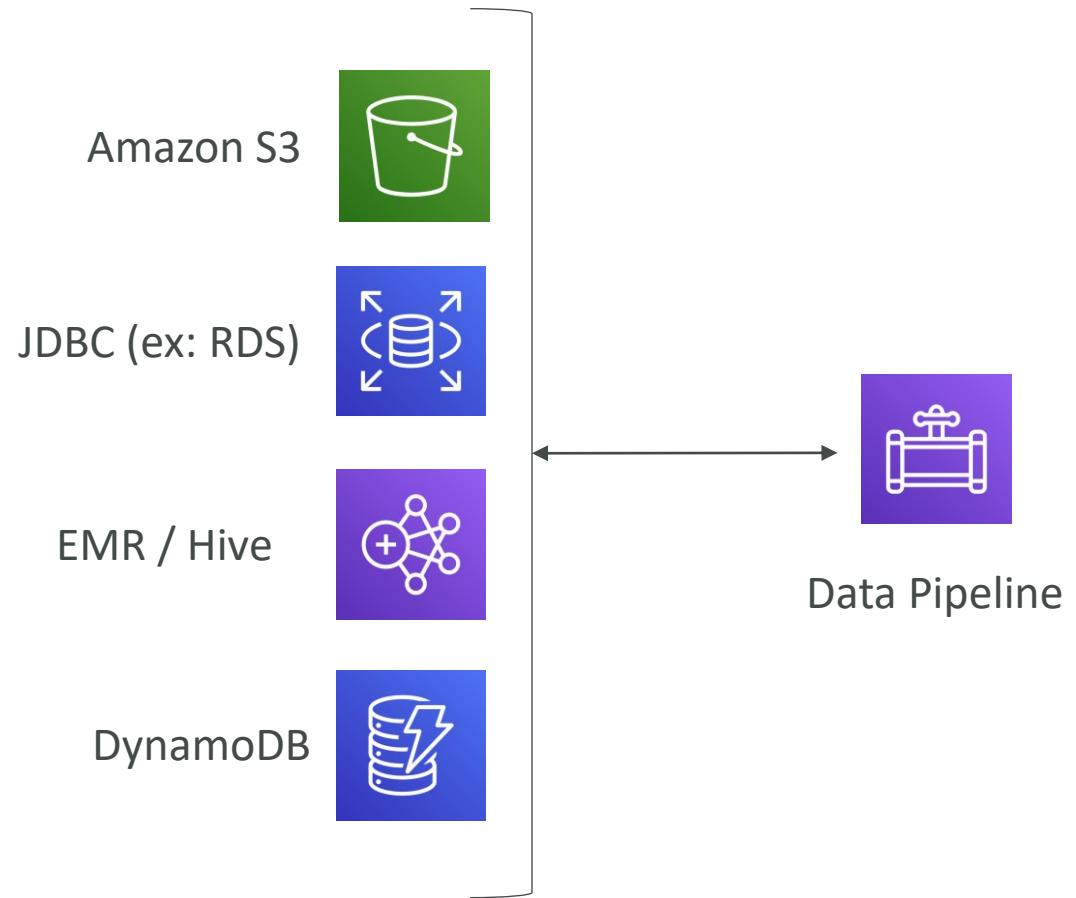
# Amazon Machine Learning (ML) (deprecated)



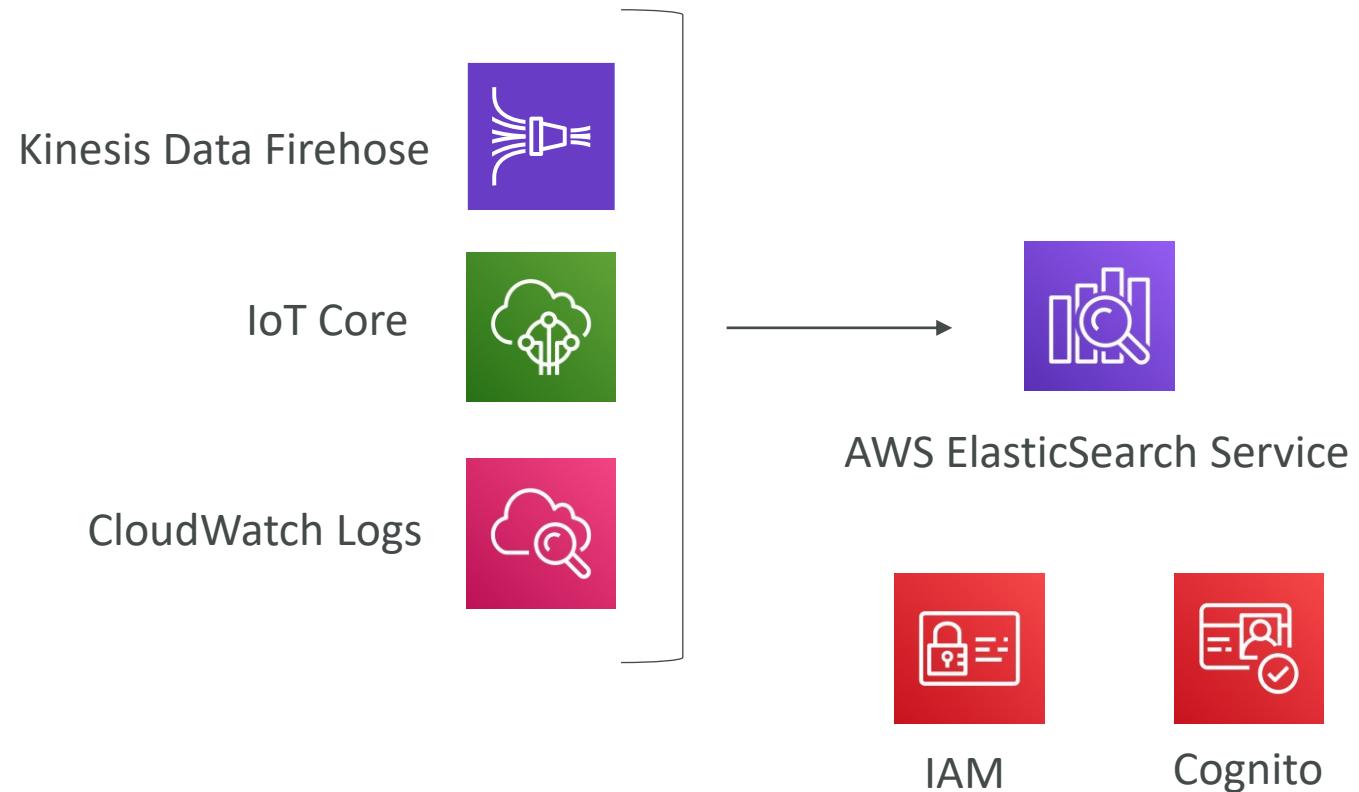
# Amazon SageMaker



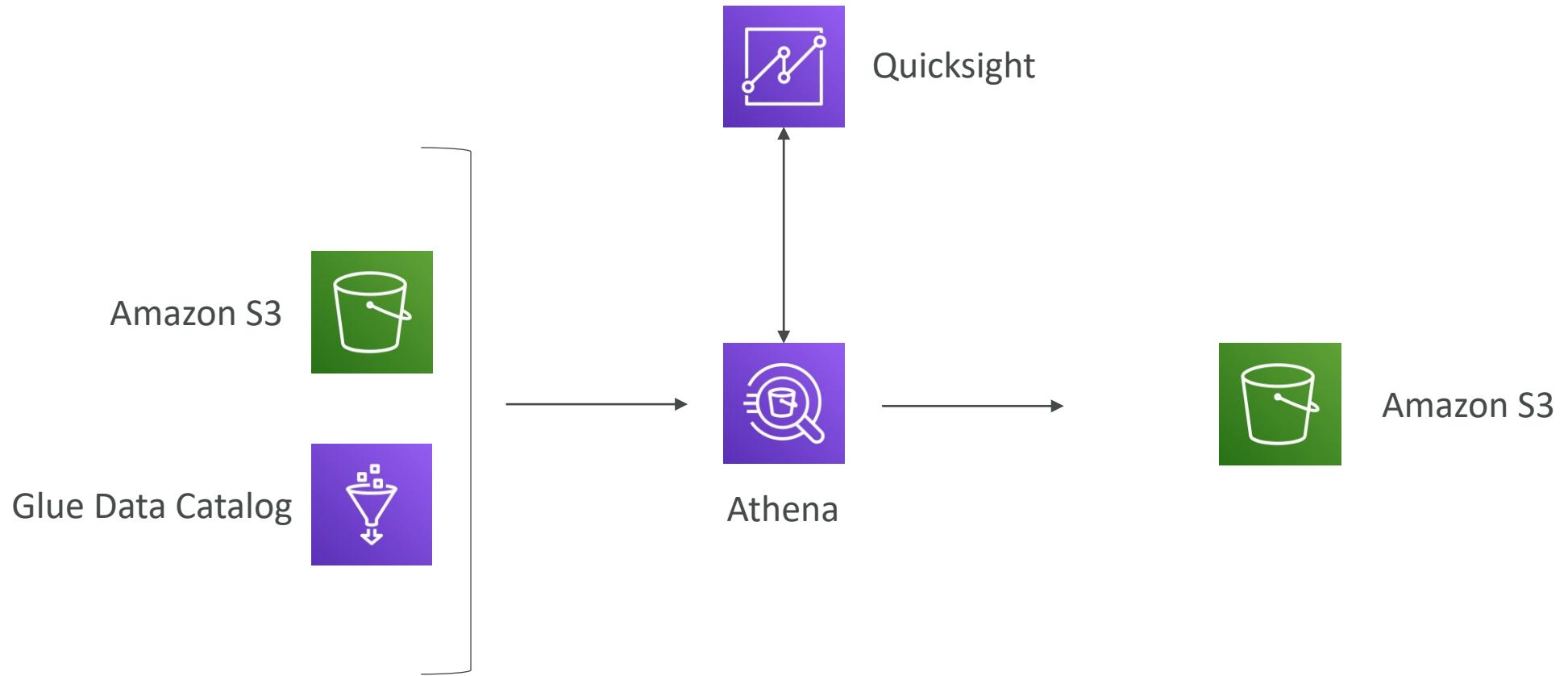
# AWS Data Pipeline



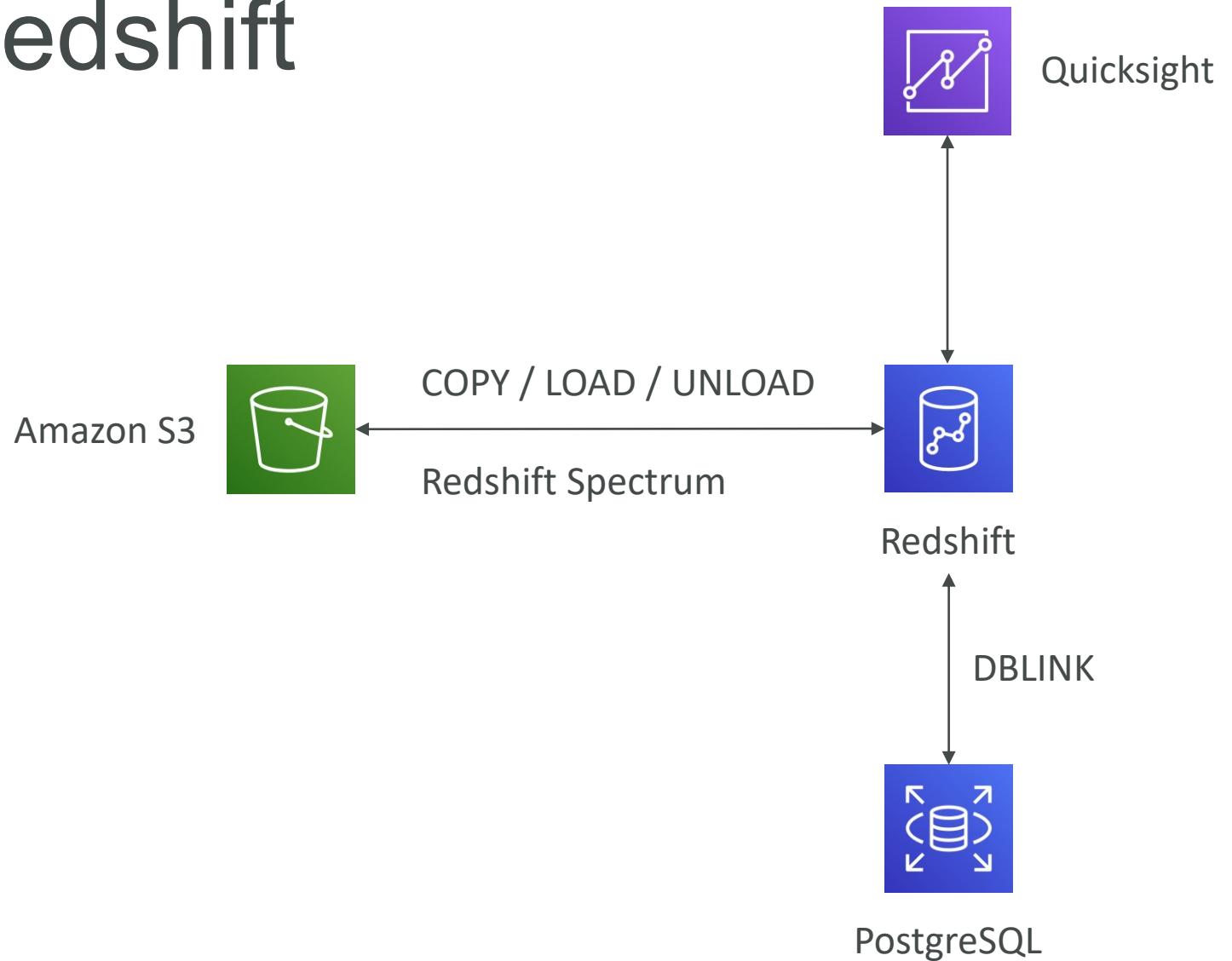
# ElasticSearch Service



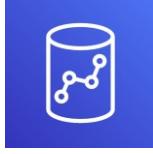
# Athena



# Redshift



# Quicksight

RDS / JDBC	
Redshift	
Athena	
Amazon S3	



# AWS Instance Types

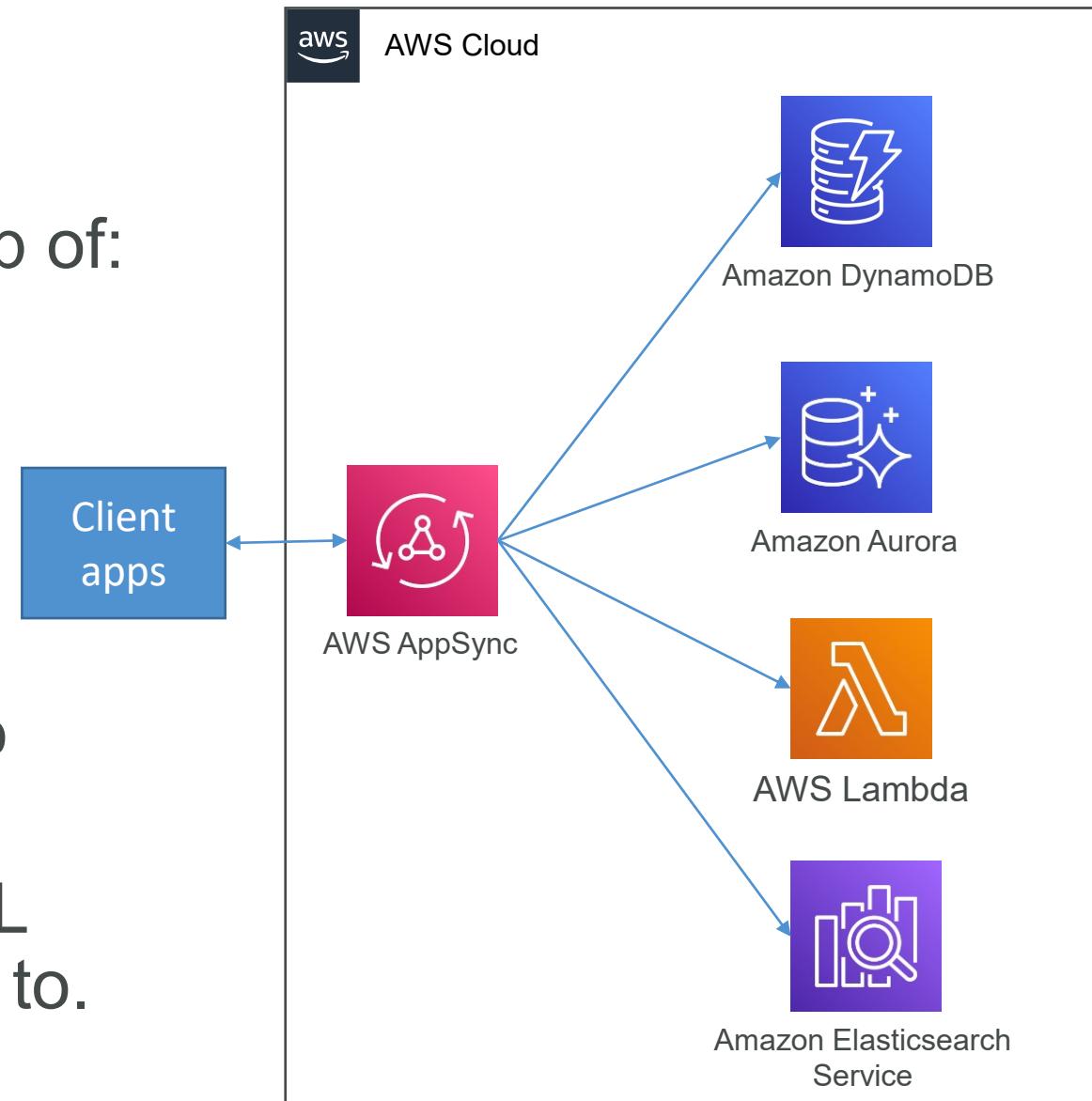
- General Purpose: T2, T3, M4, M5
- Compute Optimized: C4, C5
  - Batch processing, Distributed analytics, Machine / Deep Learning Inference
- Memory Optimized: R4, R5, X1, Z1d
  - High performance database, In memory database, Real time big data analytics
- Accelerated Computing: P2, P3, G3, F1
  - GPU instances, Machine or Deep Learning, High Performance Computing
- Storage Optimized: H1, I3, D2
  - Distributed File System (HDFS), NFS, Map Reduce, Apache Kafka, Redshift

# EC2 in Big Data

- On demand, Spot & Reserved instances:
  - Spot: can tolerate loss, low cost => checkpointing feature (ML, etc)
  - Reserved: long running clusters, databases (over a year)
  - On demand: remaining workloads
- Auto Scaling:
  - Leverage for EMR, etc
  - Automated for DynamoDB, Auto Scaling Groups, etc...
- EC2 is behind EMR
  - Master Nodes
  - Compute Nodes (contain data) + Tasks Nodes (do not contain data)

# AWS AppSync

- Creates GraphQL API on top of:
  - DynamoDB
  - Aurora
  - Lambda
  - Elasticsearch
  - HTTP
- GraphQL is an alternative to REST etc. for applications
- AppSync creates a GraphQL endpoint your apps can talk to.



# Amazon Kendra

- Enterprise search with natural language
- For example, “Where is the IT support desk?” “How do I connect to my VPN?”
- Combines data from file systems, SharePoint, intranet, sharing services (JDBC, S3) into one searchable repository
- ML-powered (of course) – uses thumbs up / down feedback
- Relevance tuning – boost strength of document freshness, view counts, etc.
- Alexa’s sister? I don’t know, but that’s one way to remember it ☺

The screenshot shows the 'Intranet Search' interface. The search bar at the top contains the query 'it support desk'. Below the search bar, there's a 'Your recent searches' dropdown and a link to 'Displaying results 1 - 10 of 21'. On the left, there are filters for 'SEARCH IN:' (Everything (21), Wiki (17), Email List Archive (3)), 'REFINE:' (Categories: Service (1), Team (1); Creator: admin (1), abcde (1), it (1), corp (1)), and 'CATEGORIES' (Service (1), Team (1)). The main results area shows three items:

- IT\_Support\_Training\_Program.Web**: A document from a wiki page about Linux desktop setup.
- Com\_Support\_Wiki.Web**: A document from a wiki page about communication sign-in flow.
- OperationalBestPractices.Event**: A document from a wiki page about dialing into both calls.

On the right side, there's a 'RESULTS PAGE' section with a search bar for 'Where is the it support desk?'. It displays 'Kendra's suggested answer' which is '1st floor'. Below that, there are 'Frequently asked questions' with links to 'Where do I get IT help?', 'What are the IT support hours?', and 'Where can I get IT help corporate campus?'. At the bottom right, there's a note about frequently asked questions.

# AWS Data Exchange



- Find, subscribe to, and use third-party data in the cloud
  - Reuters, who curate data from over 2.2 million unique news stories per year in multiple languages
  - Change Healthcare, who process and anonymize more than 14 billion healthcare transactions and \$1 trillion in claims annually
  - Dun & Bradstreet, who maintain a database of more than 330 million global business records;
  - Foursquare, whose location data is derived from 220 million unique consumers and includes more than 60 million global commercial venues.
- Once subscribed to a data product, you can use the AWS Data Exchange API to load data directly into Amazon S3 and then analyze it with a wide variety of AWS analytics and machine learning services

# AWS Data Exchange



**FOURSQUARE**



AWS Data Exchange



Amazon S3



Amazon SageMaker

Browse catalog [Info](#)

Search  Search

All data products (3,886 results) showing 1 - 36

Sort by most relevant ▾

< 1 ... >

# AWS Data Exchange – Other Products

- **AWS Data Exchange for Redshift**

- Find and subscribe to third-party data in AWS Data Exchange that you can query in an Amazon Redshift data warehouse in minutes
- Easily license your data in Amazon Redshift through AWS Data Exchange

- **AWS Data Exchange for APIs**

- Find and subscribe to third-party APIs with a consistent access using AWS SDKs
- Consistent AWS-native authentication and governance

# Amazon AppFlow

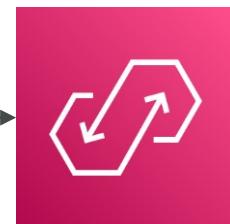


- Fully managed integration service that enables you to securely transfer data between **Software-as-a-Service (SaaS) applications and AWS**
- Sources: **Salesforce**, SAP, Zendesk, Slack, and ServiceNow
- Destinations: AWS services like **Amazon S3**, **Amazon Redshift** or non-AWS such as SnowFlake and Salesforce
- Frequency: on a schedule, in response to events, or on demand
- Data transformation capabilities like filtering and validation
- Encrypted over the public internet or privately over AWS PrivateLink
- Don't spend time writing the integrations and leverage APIs immediately

# Amazon AppFlow

## Sources

Source details <a href="#">Info</a>	
Source name	
 <b>Salesforce</b>	Salesforce is a customer relationship management...
	
 <b>Amazon S3</b>	Amazon Simple Storage Service (Amazon S3) is a...
 <b>Amplitude</b>	Amplitude is a product analytics platform that...
 <b>Datadog</b>	Datadog is a monitoring service for cloud-s...
 <b>Dynatrace</b>	Dynatrace is a software intelligence company...
 <b>Google Analytics</b>	Google Analytics is a web analytics service...
 <b>Infor Nexus</b>	Infor is a global software company that buil...
 <b>Marketo</b>	Marketo is a marketing automation softwar...
 <b>Salesforce Pardot</b>	Pardot is a marketing automation solution t...



**Amazon  
AppFlow**

## Destinations

Amazon Redshift	
	Amazon Redshift is a fast, fully mana...
Amazon S3	
	Amazon Simple Storage Service (Amazon S3) is a...
Amazon Lookout for Metrics	
	Amazon Lookout for Metrics is a machine learning...
Marketo	
	Marketo is a marketing automation software...
Salesforce	
	Salesforce is a customer relationship man...
Snowflake	
	Snowflake is cloud data warehouse service ...
Upsolver	
	Upsolver is a cloud-native data preparati...
Zendesk	

# Preparing for the exam

# Exam Tips

The strategic aspect...

# Take your time reading the question

- Look for key words about requirements

A large news website needs to produce personalized recommendations for articles **to its readers**, by training a machine learning model **on a daily basis** using historical click data. The influx of this data is fairly constant, except during major elections when **traffic to the site spikes considerably**.

Which system would provide the most **cost-effective** and **reliable** solution?

# Pace yourself

- You have 170 minutes and about 65 questions
- That's about 2 ½ minutes per question!
- Try not to get stressed out... that's enough time to read and understand each question



# Flag questions for later review

- If you're stumped on something, don't spend too much time on it
- Select your best guess, and mark it for review
- Then use any time you have at the end to go back and reconsider
- Flag questions you're not totally sure about, too.



# Arrive prepared

- Get a good night's sleep
- Do whatever you need to do to stay alert – this test requires stamina
- Go to the bathroom before arriving
- Arrive early – the exam location may be hard to find



# Additional prep resources

- AWS Big Data White Paper (“Big Data Analytics Options on AWS”)
- AWS’s free online prep course
- White papers on Kinesis, Database Migration Service, Migrating Applications to AWS
- Exam overview from AWS
- This shouldn’t be your first certification exam
- Take our practice exam

# State of learning checkpoint

- Let's look how far we've gone on our learning journey
- <https://aws.amazon.com/certification/certified-big-data-specialty/>

# How will the exam work?

- You'll have to register online at <https://www.aws.training/>
- Fee for the exam is 300 USD
- Provide two identity documents (ID, Credit Card, details are in emails sent to you)
- No notes are allowed, no pen is allowed, no speaking
- ~65 questions will be asked in 170 minutes
- At the end you can optionally review all the questions / answers
  
- You will know right away if you passed / failed the exams
- You will not know which answers were right / wrong
- You will know the overall score a few days later (email notification)
- The pass score is not provided, but some people passed with 60%
- If you fail, you can retake the exam again 14 days later