

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

- This document is reserved for people enrolled into the AWS Certified Data Engineer 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!**

# Table of Contents

- [Data Engineering Fundamentals](#)
- [Storage](#)
- [Database](#)
- [Migration and Transfer](#)
- [Compute](#)
- [Containers](#)
- [Analytics](#)
- [Application Integration](#)
- [Security, Identity, and Compliance](#)
- [Networking and Content Delivery](#)
- [Management and Governance](#)
- [Machine Learning](#)
- [Developer Tools](#)
- [Everything Else](#)
- [Exam Tips](#)

# AWS Certified Data Engineer Associate Course

## DEA-C01

# Welcome! We're starting in 5 minutes



- We're going to prepare for the AWS Certified Data Engineer – Associate exam – DEA-C01
- 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 engineering background
- We will cover all the AWS Data Engineering services related to the exam
- Take your time, it's not a race!

# About me

- I'm Stephane!
- 10x AWS Certified (so far!)
- Worked with AWS many years; I built websites, apps, streaming platforms
- Veteran Instructor on AWS Certifications
- You can find me on
  - LinkedIn: <https://www.linkedin.com/in/stephanemaar>
  - Instagram: <https://www.instagram.com/stephanemaar/>
  - Twitter: <https://twitter.com/stephanemaarek>
  - GitHub: <https://github.com/simplesteph>
  - Medium: <https://medium.com/@stephane.maarek>



4.7 Instructor Rating  
 554,165 Reviews  
 1,819,442 Students  
 43 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 – Data Analytics & 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>



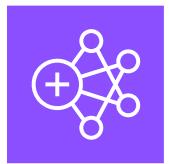
Udemy Instructor Partner

Total students      Reviews  
**752,695**      **153,704**

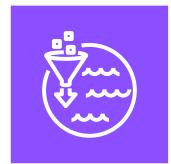


# Services we'll learn

## ANALYTICS



Amazon EMR



AWS Lake Formation



Amazon Redshift



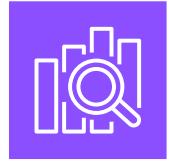
Amazon Kinesis



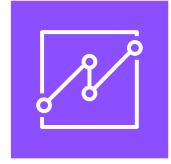
AWS Glue



Amazon Managed Streaming  
for Apache Kafka



Amazon  
OpenSearch Service



Amazon QuickSight

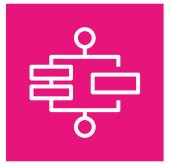


Amazon Athena

## APP INTEGRATION



Amazon EventBridge



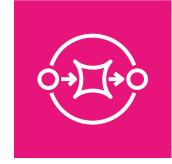
AWS Step Functions



Amazon AppFlow



Amazon Simple Notification  
Service (Amazon SNS)



Amazon Simple Queue  
Service (Amazon SQS)



Amazon Managed Workflows  
for Apache Airflow

## CLOUD FINANCIAL MANAGEMENT



AWS Budgets



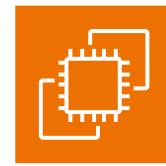
AWS Cost Explorer

# Services we'll learn

## COMPUTE



AWS Batch



Amazon Elastic Compute  
Cloud (Amazon EC2)



AWS Lambda



AWS Serverless  
Application Repository

## CONTAINERS



Amazon Elastic Container  
Registry (Amazon ECR)



Amazon Elastic Container  
Service (Amazon ECS)



Amazon Elastic Kubernetes  
Service (Amazon EKS)

## DATABASE



Amazon DocumentDB  
(with MongoDB compatibility)



Amazon Keyspaces  
(for Apache Cassandra)



Amazon Neptune



Amazon DynamoDB



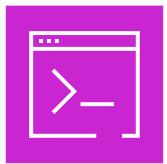
Amazon MemoryDB  
for Redis



Amazon Relational Database  
Service (Amazon RDS)

# Services we'll learn

## DEVELOPER TOOLS



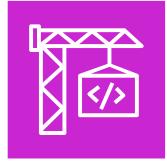
AWS Command Line Interface (AWS CLI)



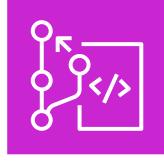
AWS Cloud9



AWS Cloud Development Kit (AWS CDK)



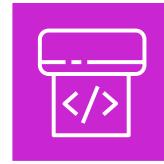
AWS CodeBuild



AWS CodeCommit



AWS CodeDeploy

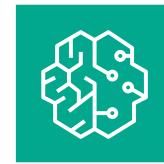


AWS CodePipeline

## FRONTEND WEB



Amazon API Gateway



Amazon SageMaker

## MANAGEMENT AND GOVERNANCE



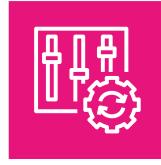
AWS CloudFormation



AWS CloudTrail



Amazon CloudWatch



AWS Config



Amazon Managed Grafana



AWS Systems Manager



AWS Well-Architected Tool

# Services we'll learn

## Migration and Transfer



AWS Application Discovery Service



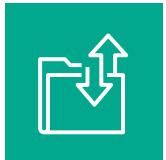
AWS Application Migration Service



AWS Database Migration Service (AWS DMS)



AWS DataSync

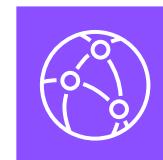


AWS Transfer Family



AWS Snow Family

## Networking and Content Delivery



Amazon CloudFront



AWS PrivateLink



Amazon Route 53



Amazon Virtual Private Cloud (Amazon VPC)

## Security, Identity, and Compliance



AWS Identity and Access Management (IAM)



AWS Key Management Service (AWS KMS)



Amazon Macie



AWS Secrets Manager



AWS Shield



AWS WAF

# Services we'll learn

## STORAGE



AWS Backup



Amazon Elastic Block Store  
(Amazon EBS)



Amazon Elastic File System  
(Amazon EFS)



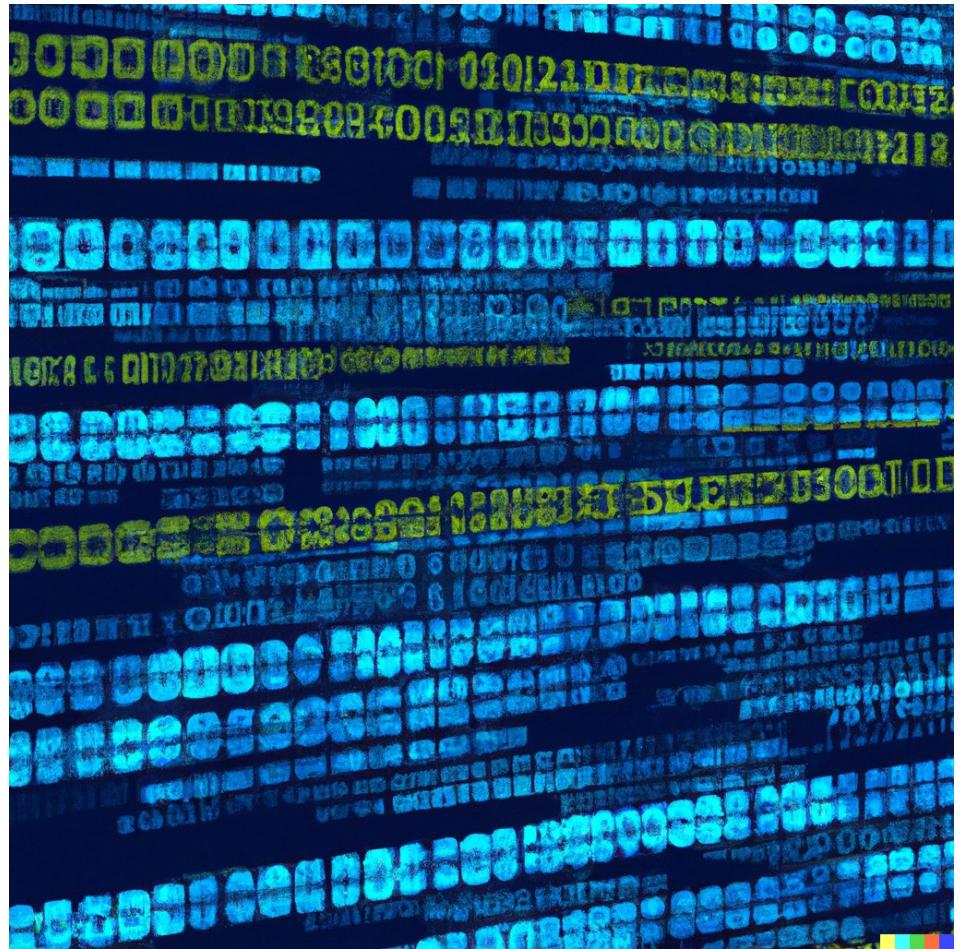
Amazon Simple Storage  
Service (Amazon S3)

# Data Engineering Fundamentals

Beyond AWS – a review

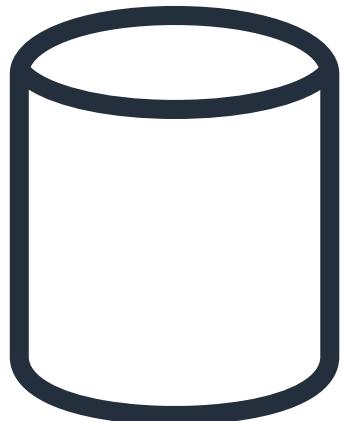
# Types of Data

- Structured
- Unstructured
- Semi-Structured



# Structured Data

- Definition: Data that is organized in a defined manner or schema, typically found in relational databases.
- Characteristics:
  - Easily queryable
  - Organized in rows and columns
  - Has a consistent structure
- Examples:
  - Database tables
  - CSV files with consistent columns
  - Excel spreadsheets



# Unstructured Data

- Definition: Data that doesn't have a predefined structure or schema.
- Characteristics:
  - Not easily queryable without preprocessing
  - May come in various formats
- Examples:
  - Text files without a fixed format
  - Videos and audio files
  - Images
  - Emails and word processing documents



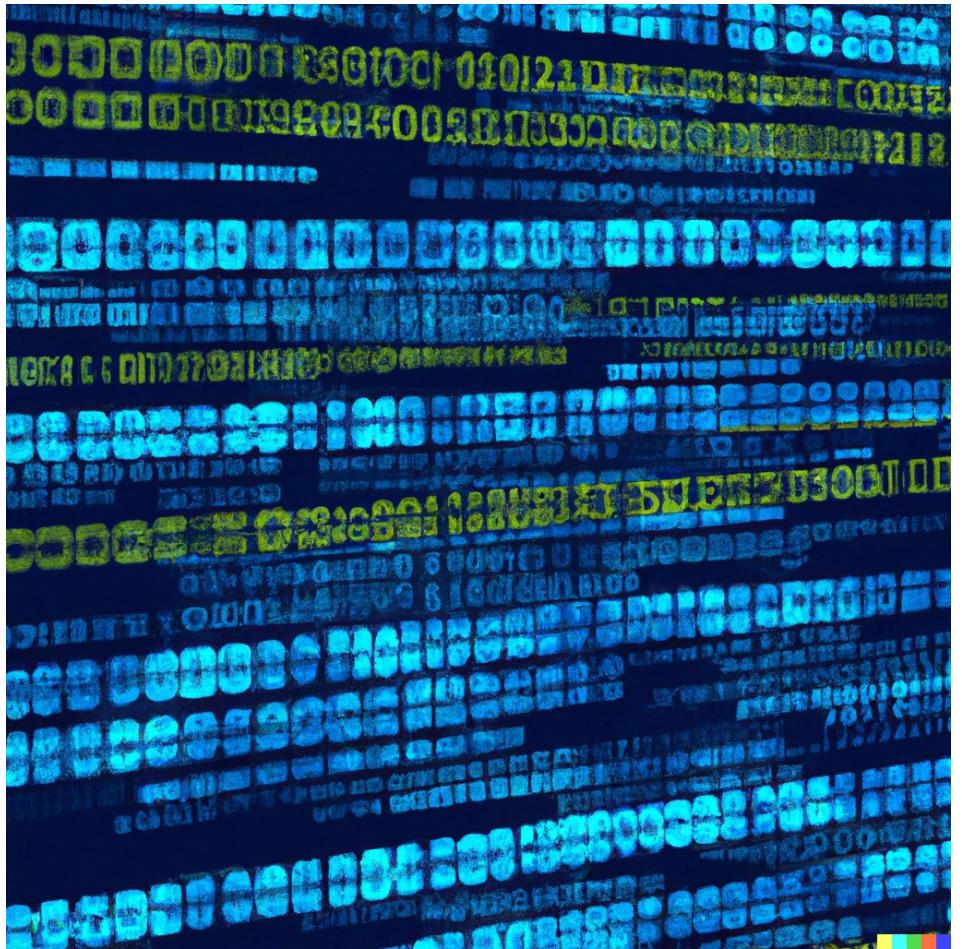
# Semi-Structured Data

- Definition: Data that is not as organized as structured data but has some level of structure in the form of tags, hierarchies, or other patterns.
- Characteristics:
  - Elements might be tagged or categorized in some way
  - More flexible than structured data but not as chaotic as unstructured data
- Examples:
  - XML and JSON files
  - Email headers (which have a mix of structured fields like date, subject, etc., and unstructured data in the body)
  - Log files with varied formats



# Properties of Data

- Volume
- Velocity
- Variety



# Volume

- Definition: Refers to the amount or size of data that organizations are dealing with at any given time.
- Characteristics:
  - May range from gigabytes to petabytes or even more
  - Challenges in storing, processing, and analyzing high volumes of data
- Examples:
  - A popular social media platform processing terabytes of data daily from user posts, images, and videos.
  - Retailers collecting years' worth of transaction data, amounting to several petabytes.

# Velocity

- Definition: Refers to the speed at which new data is generated, collected, and processed.
- Characteristics:
  - High velocity requires real-time or near-real-time processing capabilities
  - Rapid ingestion and processing can be critical for certain applications
- Examples:
  - Sensor data from IoT devices streaming readings every millisecond.
  - High-frequency trading systems where milliseconds can make a difference in decision-making.

# Variety

- Definition: Refers to the different types, structures, and sources of data.
- Characteristics:
  - Data can be structured, semi-structured, or unstructured
  - Data can come from multiple sources and in various formats
- Examples:
  - A business analyzing data from relational databases (structured), emails (unstructured), and JSON logs (semi-structured).
  - Healthcare systems collecting data from electronic medical records, wearable health devices, and patient feedback forms.



# Data Warehouses vs. Data Lakes



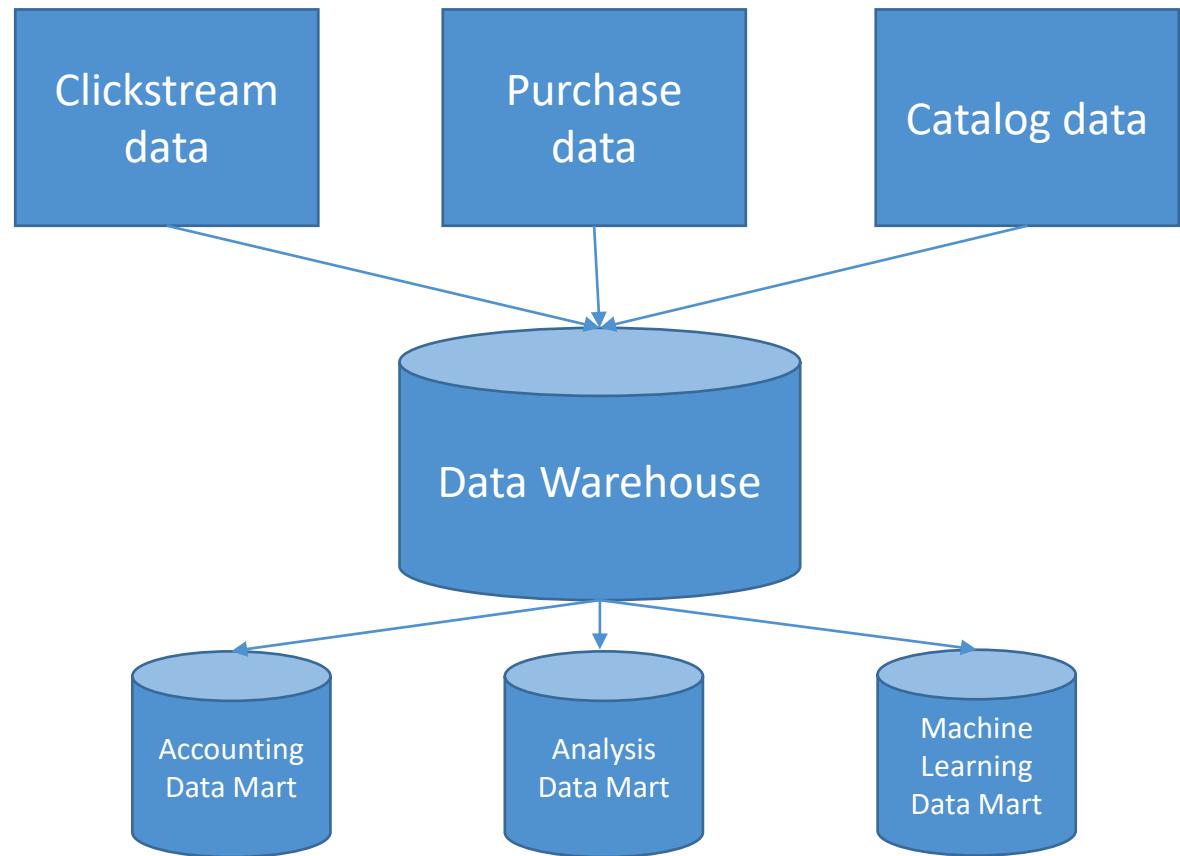
# Data Warehouse

- Definition: A centralized repository optimized for analysis where data from different sources is stored in a structured format.
- Characteristics:
  - Designed for complex queries and analysis
  - Data is cleaned, transformed, and loaded (ETL process)
  - Typically uses a star or snowflake schema
  - Optimized for read-heavy operations
- Examples:
  - Amazon Redshift
  - Google BigQuery
  - Microsoft Azure SQL Data Warehouse



Amazon Redshift

# Data Warehouse example



# Data Lake

- Definition: A storage repository that holds vast amounts of raw data in its native format, including structured, semi-structured, and unstructured data.
- Characteristics:
  - Can store large volumes of raw data without predefined schema
  - Data is loaded as-is, no need for preprocessing
  - Supports batch, real-time, and stream processing
  - Can be queried for data transformation or exploration purposes
- Examples:
  - Amazon Simple Storage Service (S3) when used as a data lake
  - Azure Data Lake Storage
  - Hadoop Distributed File System (HDFS)



# Comparing the two

- **Schema:**
  - Data Warehouse: Schema-on-write (predefined schema before writing data)
    - Extract – Transform – Load (**ETL**)
  - Data Lake: Schema-on-read (schema is defined at the time of reading data)
    - Extract – Load – Transform (**ELT**)
- **Data Types:**
  - Data Warehouse: Primarily structured data
  - Data Lake: Both structured and unstructured data
- **Agility:**
  - Data Warehouse: Less agile due to predefined schema
  - Data Lake: More agile as it accepts raw data without a predefined structure
- **Processing:**
  - Data Warehouse: ETL (Extract, Transform, Load)
  - Data Lake: ELT (Extract, Load, Transform) or just Load for storage purposes
- **Cost:**
  - Data Warehouse: Typically more expensive because of optimizations for complex queries
  - Data Lake: Cost-effective storage solutions, but costs can rise when processing large amounts of data

# Choosing a Warehouse vs. a Lake

- **Use a Data Warehouse when:**
  - You have structured data sources and require fast and complex queries.
  - Data integration from different sources is essential.
  - Business intelligence and analytics are the primary use cases.
- **Use a Data Lake when:**
  - You have a mix of structured, semi-structured, or unstructured data.
  - You need a scalable and cost-effective solution to store massive amounts of data.
  - Future needs for data are uncertain, and you want flexibility in storage and processing.
  - Advanced analytics, machine learning, or data discovery are key goals.
- Often, organizations use a combination of both, ingesting raw data into a data lake and then processing and moving refined data into a data warehouse for analysis.



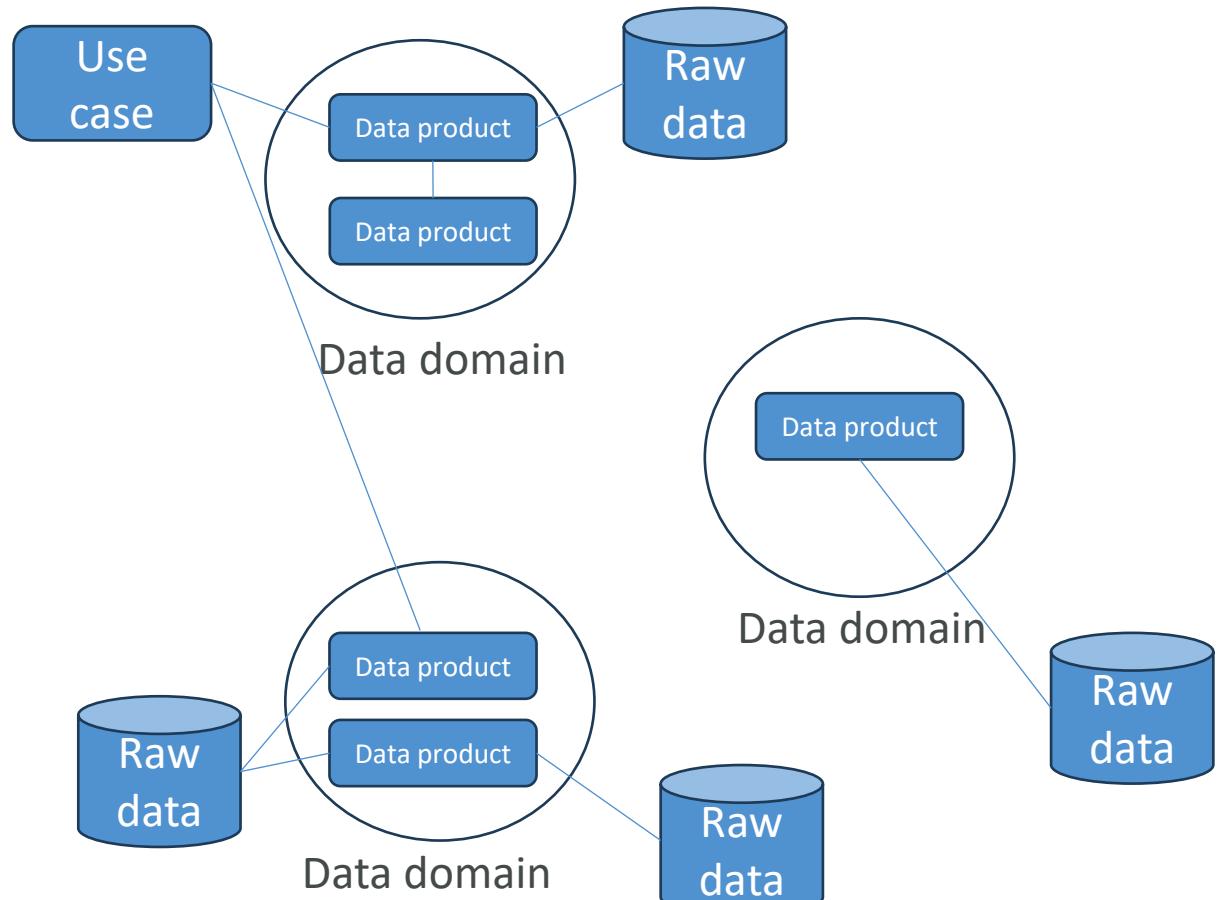
# Data Lakehouse

- Definition: A hybrid data architecture that combines the best features of data lakes and data warehouses, aiming to provide the performance, reliability, and capabilities of a data warehouse while maintaining the flexibility, scale, and low-cost storage of data lakes.
- Characteristics:
  - Supports both structured and unstructured data.
  - Allows for schema-on-write and schema-on-read.
  - Provides capabilities for both detailed analytics and machine learning tasks.
  - Typically built on top of cloud or distributed architectures.
  - Benefits from technologies like Delta Lake, which bring ACID transactions to big data.
- Examples:
  - **AWS Lake Formation (with S3 and Redshift Spectrum)**
  - **Delta Lake:** An open-source storage layer that brings ACID transactions to Apache Spark and big data workloads.
  - **Databricks Lakehouse Platform:** A unified platform that combines the capabilities of data lakes and data warehouses.
  - **Azure Synapse Analytics:** Microsoft's analytics service that brings together big data and data warehousing.



# Data Mesh

- Coined in 2019; it's more about governance and organization
- Individual teams own “data products” within a given domain
- These data products serve various “use cases” around the organization
- “Domain-based data management”
- Federated governance with central standards
- Self-service tooling & infrastructure
- Data lakes, warehouses, etc. may be part of it
  - But a “data mesh” is more about the “data management paradigm” and not the specific technologies or architectures.



# ETL Pipelines

- **Definition:** ETL stands for Extract, Transform, Load. It's a process used to move data from source systems into a data warehouse.
- **Extract:**
  - Retrieve raw data from source systems, which can be databases, CRMs, flat files, APIs, or other data repositories.
  - Ensure data integrity during the extraction phase.
  - Can be done in real-time or in batches, depending on requirements.

# ETL Pipelines: Transform

- Convert the extracted data into a format suitable for the target data warehouse.
- Can involve various operations such as:
  - Data cleansing (e.g., removing duplicates, fixing errors)
  - Data enrichment (e.g., adding additional data from other sources)
  - Format changes (e.g., date formatting, string manipulation)
  - Aggregations or computations (e.g., calculating totals or averages)
  - Encoding or decoding data
  - Handling missing values

# ETL Pipelines: Load

- Move the transformed data into the target data warehouse or another data repository.
- Can be done in batches (all at once) or in a streaming manner (as data becomes available).
- Ensure that data maintains its integrity during the loading phase.



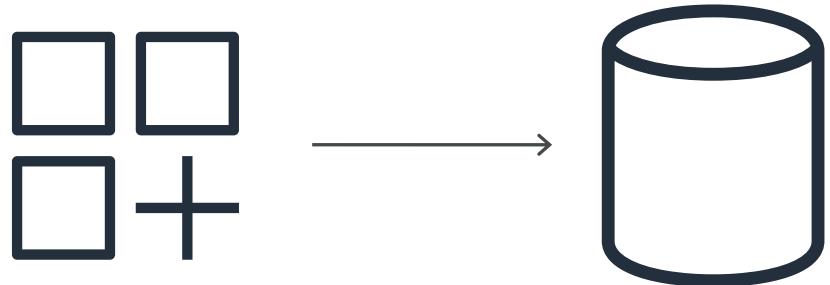
# Managing ETL Pipelines

- This process must be automated in some reliable way
- AWS Glue
- Orchestration services
  - EventBridge
  - Amazon Managed Workflows for Apache Airflow [Amazon MWAA]
  - AWS Step Functions
  - Lambda
  - Glue Workflows
- We'll get into specific architectures later.

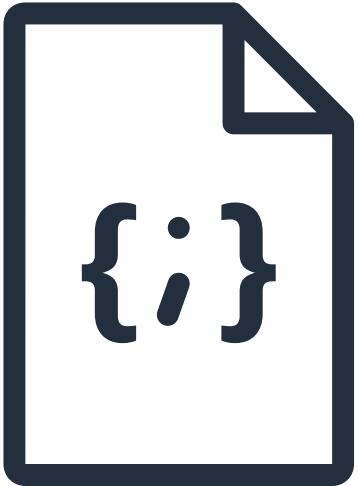


# Data Sources

- JDBC
  - Java Database Connectivity
  - Platform-independent
  - Language-dependent
- ODBC
  - Object Database Connectivity
  - Platform-dependent (thx to drivers)
  - Language-independent
- Raw logs
- API's
- Streams



# Common Data Formats



# CSV (Comma-Separated Values)

- **Description:** Text-based format that represents data in a tabular form where each line corresponds to a row and values within a row are separated by commas.
- **When to Use:**
  - For small to medium datasets.
  - For data interchange between systems with different technologies.
  - For human-readable and editable data storage.
  - Importing/Exporting data from databases or spreadsheets.
- **Systems:** Databases (SQL-based), Excel, Pandas in Python, R, many ETL tools.

```
InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,C  
ustomerID,Country  
536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6,12/1/2010  
8:26,2.55,17850,United Kingdom  
536365,71053,WHITE METAL LANTERN,6,12/1/2010  
8:26,3.39,17850,United Kingdom  
536365,84406B,CREAM CUPID HEARTS COAT HANGER,8,12/1/2010  
8:26,2.75,17850,United Kingdom  
536365,84029G,KNITTED UNION FLAG HOT WATER BOTTLE,6,12/1/2010  
8:26,3.39,17850,United Kingdom  
536365,84029E,RED WOOLLY HOTTIE WHITE HEART.,6,12/1/2010  
8:26,3.39,17850,United Kingdom  
536365,22752,SET 7 BABUSHKA NESTING BOXES,2,12/1/2010  
8:26,7.65,17850,United Kingdom  
536365,21730,GLASS STAR FROSTED T-LIGHT HOLDER,6,12/1/2010  
8:26,4.25,17850,United Kingdom  
536366,22633,HAND WARMER UNION JACK,6,12/1/2010  
8:28,1.85,17850,United Kingdom  
536366,22632,HAND WARMER RED POLKA DOT,6,12/1/2010  
8:28,1.85,17850,United Kingdom  
536367,84879,ASSORTED COLOUR BIRD ORNAMENT,32,12/1/2010  
8:34,1.69,13047,United Kingdom  
536367,22745,POPPIY'S PLAYHOUSE BEDROOM ,6,12/1/2010  
8:34,2.1,13047,United Kingdom  
536367,22748,POPPIY'S PLAYHOUSE KITCHEN,6,12/1/2010  
8:34,2.1,13047,United Kingdom  
536367,22749,FELTCRAFT PRINCESS CHARLOTTE DOLL,8,12/1/2010  
8:34,3.75,13047,United Kingdom  
536367,22310,IVORY KNITTED MUG COSY ,6,12/1/2010  
8:34,1.65,13047,United Kingdom  
536367,84969,BOX OF 6 ASSORTED COLOUR TEASPOONS,6,12/1/2010  
8:34,4.25,13047,United Kingdom  
536367,22623,BOX OF VINTAGE JIGSAW BLOCKS ,3,12/1/2010  
8:34,4.95,13047,United Kingdom  
536367,22622,BOX OF VINTAGE ALPHABET BLOCKS,2,12/1/2010  
8:34,9.95,13047,United Kingdom  
536367,21754,HOME BUILDING BLOCK WORD,3,12/1/2010  
8:34,5.95,13047,United Kingdom  
536367,21755,LOVE BUILDING BLOCK WORD,3,12/1/2010  
8:34,5.95,13047,United Kingdom  
536367,21777,RECIPE BOX WITH METAL HEART,4,12/1/2010
```

# JSON (JavaScript Object Notation)

- **Description:** Lightweight, text-based, and human-readable data interchange format that represents structured or semi-structured data based on key-value pairs.
- **When to Use:**
  - Data interchange between a web server and a web client.
  - Configurations and settings for software applications.
  - Use cases that need a flexible schema or nested data structures.
- **Systems:** Web browsers, many programming languages (like JavaScript, Python, Java, etc.), RESTful APIs, NoSQL databases (like MongoDB).

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "PICARD: You will agree, Data, that Starfleet's instructions are difficult?"}, {"role": "assistant", "content": "DATA: Difficult ... how so? Simply solve the mystery of Farpoint Station."}]}
```

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "PICARD: The problem, Data, is that another life form built that base. How do I negotiate a friendly agreement for Starfleet to use it while at the same time snoop around finding how and why they built it?"}, {"role": "assistant", "content": "DATA: Inquiry ... the word snoop ..."}]}
```

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "PICARD: Data, how can you be programmed as a virtual encyclopedia of human information without knowing a simple word like snoop?"}, {"role": "assistant", "content": "DATA: Possibility ... a kind of human behavior I was not designed to emulate?"}]}
```

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "PICARD: It means 'to spy, to sneak' ..."}, {"role": "assistant", "content": "DATA: Ah! To seek covertly, to go stealthily, to slink, slither ..."}]}
```

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "PICARD: Exactly, yes ..."}, {"role": "assistant", "content": "DATA: ... to glide, creep, skulk, pussyfoot, gumshoe ..."}]}
```

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "WORF: Something strange on the detector circuits ..."}, {"role": "assistant", "content": "DATA: It registers as solid, Captain..."}]}
```

```
{"messages": [{"role": "system", "content": "Data is an android in the TV series Star Trek: The Next Generation."}, {"role": "user", "content": "PICARD: Records search, Data. Results of detaching the Saucer Section at high warp speeds."}, {"role": "assistant", "content": "DATA: Inadvisable at any warp speed,"}]}
```

# Avro

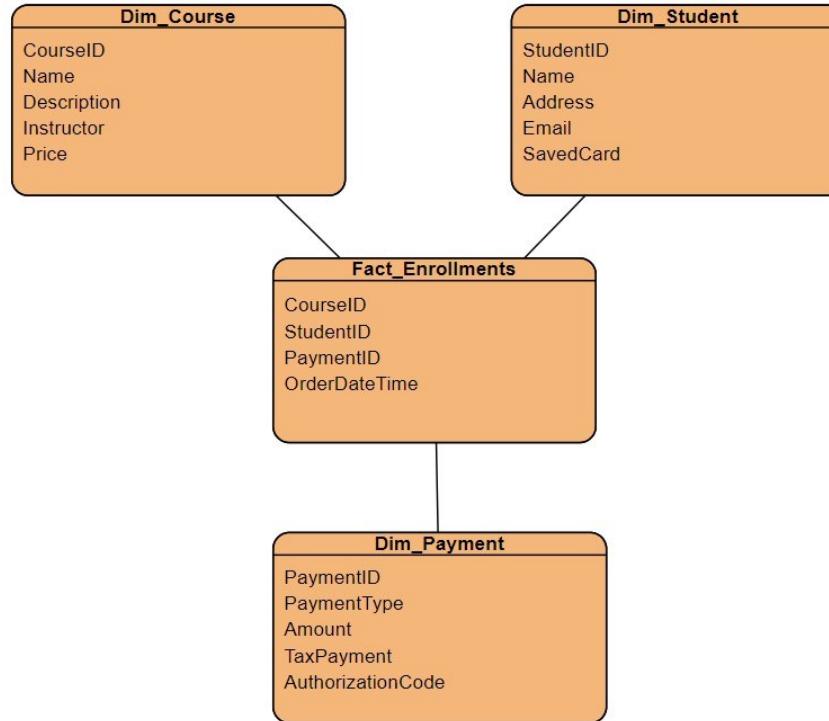
- **Description:** Binary format that stores both the data and its schema, allowing it to be processed later with different systems without needing the original system's context.
- **When to Use:**
  - With big data and real-time processing systems.
  - When schema evolution (changes in data structure) is needed.
  - Efficient serialization for data transport between systems.
- **Systems:** Apache Kafka, Apache Spark, Apache Flink, Hadoop ecosystem.

# Parquet

- **Description:** Columnar storage format optimized for analytics. Allows for efficient compression and encoding schemes.
- **When to Use:**
  - Analyzing large datasets with analytics engines.
  - Use cases where reading specific columns instead of entire records is beneficial.
  - Storing data on distributed systems where I/O operations and storage need optimization.
- **Systems:** Hadoop ecosystem, Apache Spark, Apache Hive, Apache Impala, Amazon Redshift Spectrum.

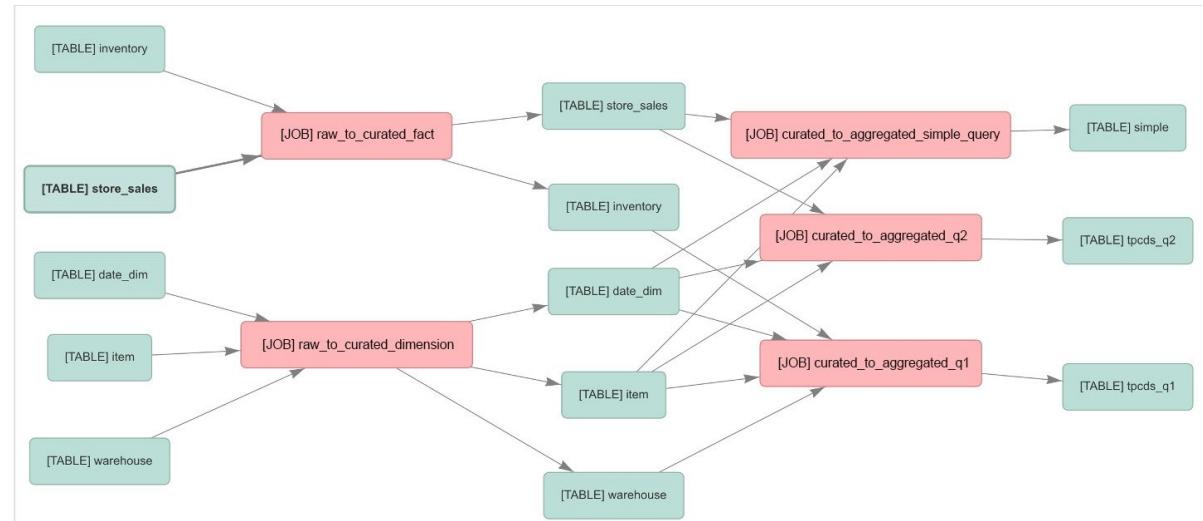
# A Very (intentionally) Incomplete Overview of Data Modeling

- The exam guide doesn't really talk about specific data models.
- But here's a star schema.
  - Fact tables
  - Dimensions
  - Primary / foreign keys
- This sort of diagram is an Entity Relationship Diagram (ERD)



# Data Lineage

- **Description:** A visual representation that traces the flow and transformation of data through its lifecycle, from its source to its final destination.
- **Importance:**
  - Helps in tracking errors back to their source.
  - Ensures compliance with regulations.
  - Provides a clear understanding of how data is moved, transformed, and consumed within systems.



# Data Lineage

- Example of capturing data lineage
  - Uses Spline Agent (for Spark) attached to Glue
  - Dump lineage data into Neptune via Lambda

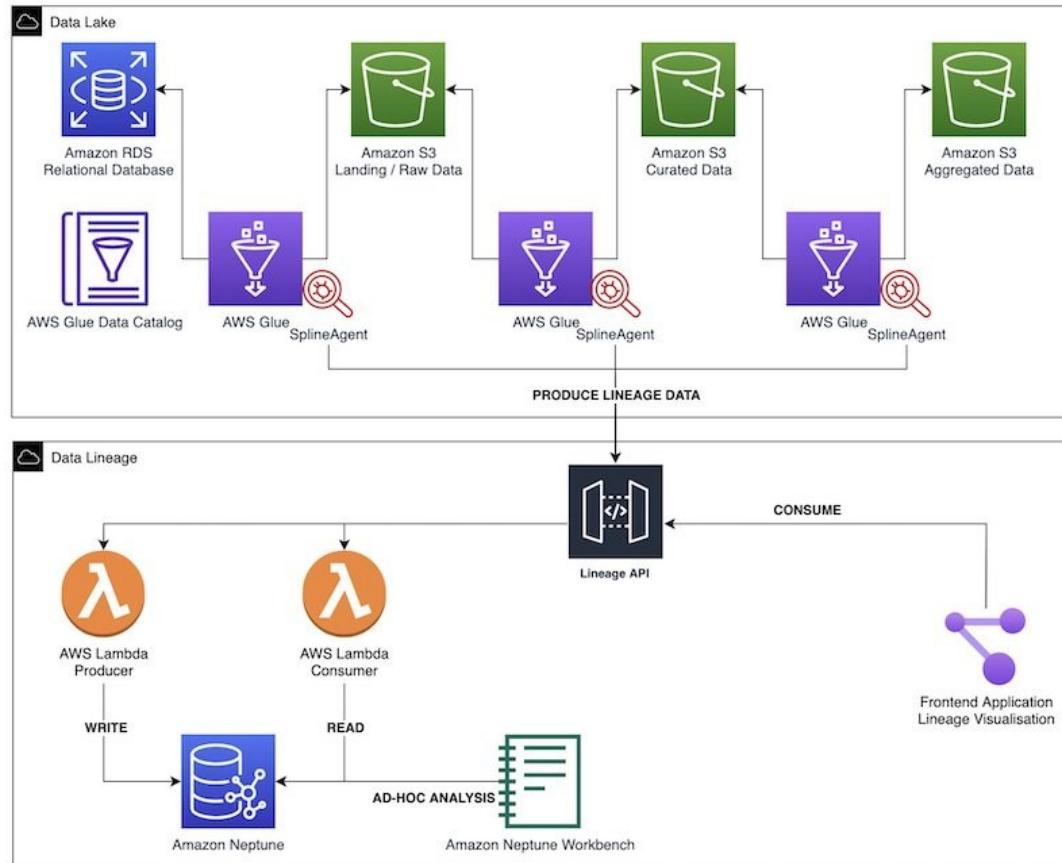


Image: AWS (<https://aws.amazon.com/blogs/big-data/build-data-lineage-for-data-lakes-using-aws-glue-amazon-neptune-and-spline/>)

# Schema Evolution

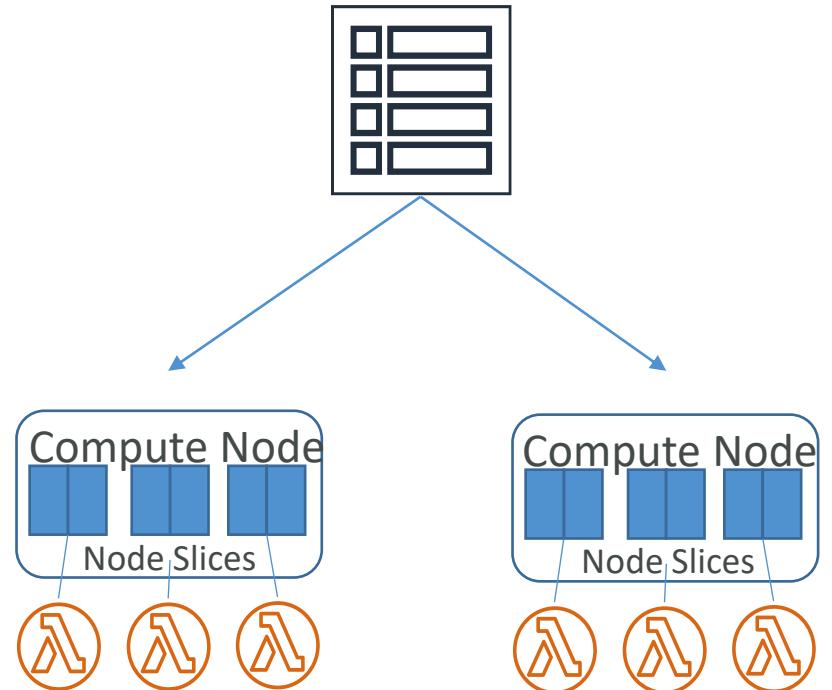
- **Description:** The ability to adapt and change the schema of a dataset over time without disrupting existing processes or systems.
- **Importance:**
  - Ensures data systems can adapt to changing business requirements.
  - Allows for the addition, removal, or modification of columns/fields in a dataset.
  - Maintains backward compatibility with older data records.
- **Glue Schema Registry**
  - Schema discovery, compatibility, validation, registration...

```
{  
  "namespace": "Customer.avro",  
  "type": "record",  
  "name": "Customer",  
  "fields": [  
    {"name": "first_name", "type": "string"},  
    {"name": "last_name", "type": "string"},  
    {"name": "full_name", "type": ["string", "null"], "default": null}  
  ]  
}
```



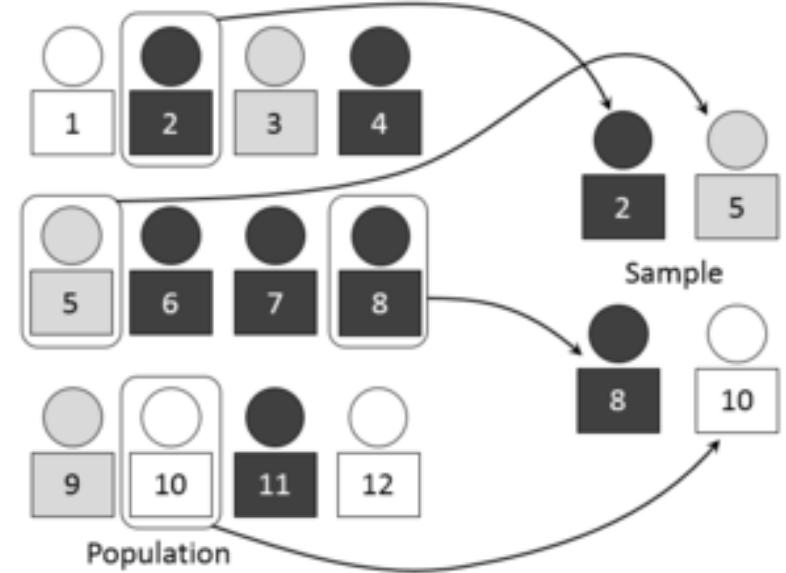
# Database Performance Optimization

- Indexing
  - Avoid full table scans!
  - Enforce data uniqueness and integrity
- Partitioning
  - Reduce amount of data scanned
  - Helps with data lifecycle management
  - Enables parallel processing
- Compression
  - Speed up data transfer, reduce storage & disk reads
  - GZIP, LZOP, BZIP2, ZSTD (Redshift examples)
    - Various tradeoffs between compression & speed
  - Columnar compression



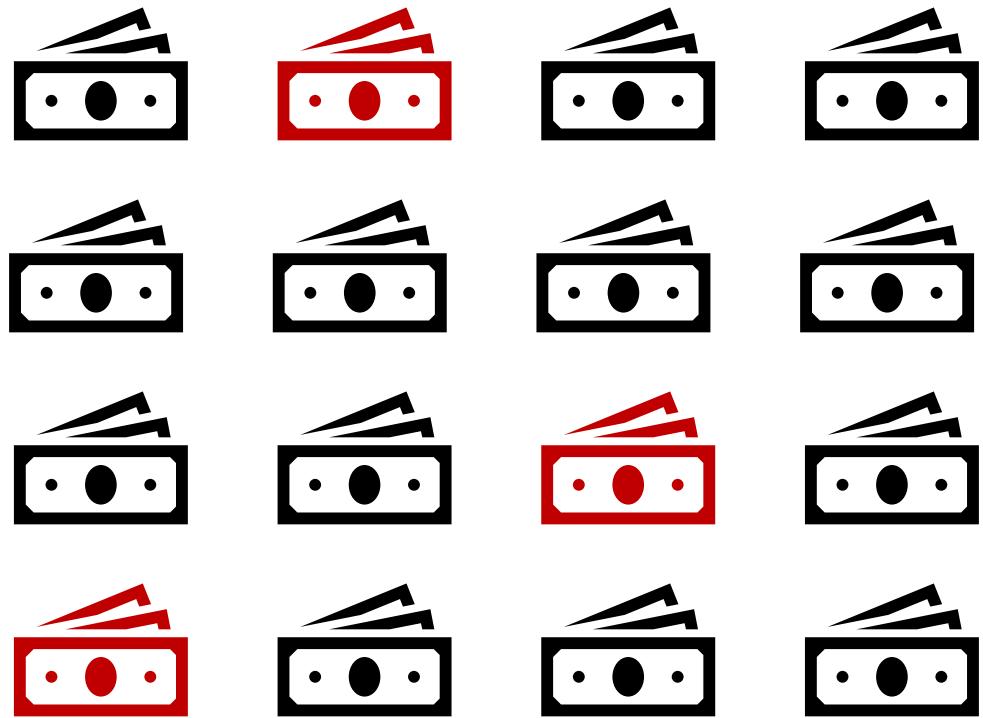
# Data Sampling Techniques

- Random Sampling
  - Everything has an equal chance
- Stratified Sampling
  - Divide population into homogenous subgroups (strata)
  - Random sample within each stratum
  - Ensures representation of each subgroup
- Others
  - Systemic, Cluster, Convenience, Judgmental

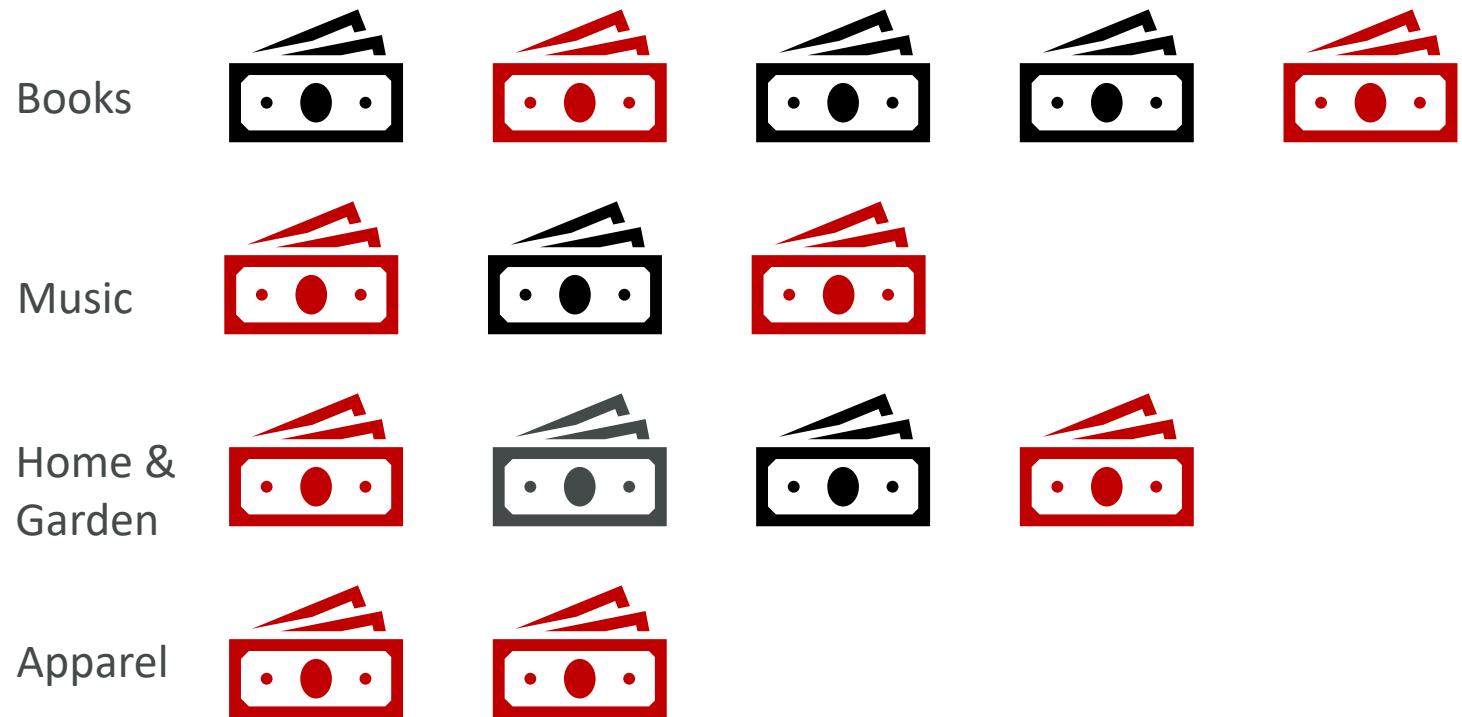


Dan Kernler, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0/>>, via Wikimedia Commons

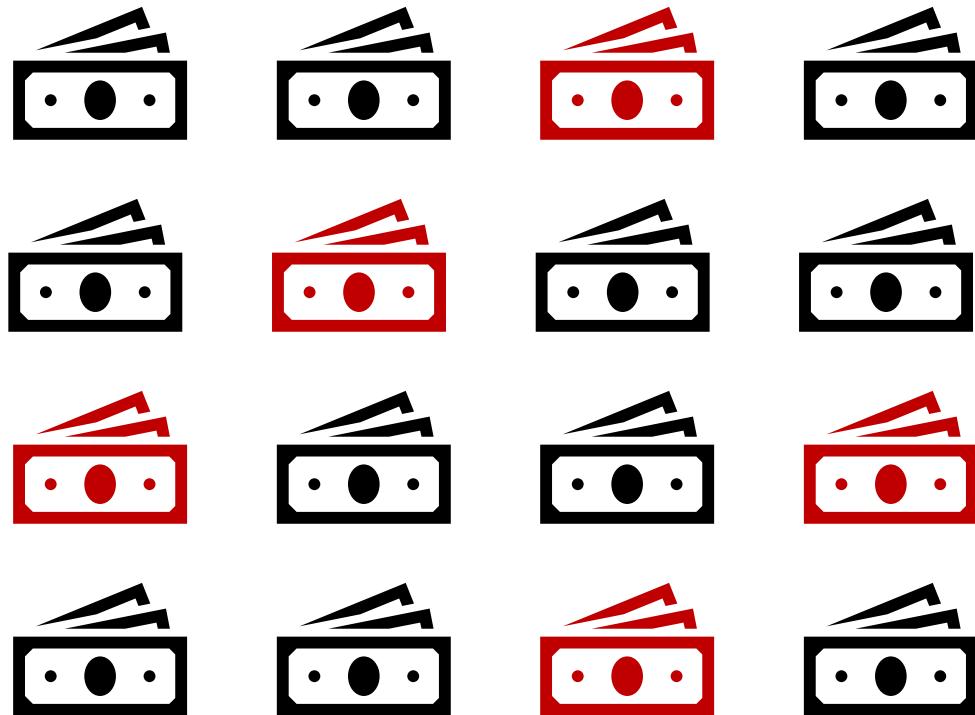
# Random Sampling



# Stratified Sampling

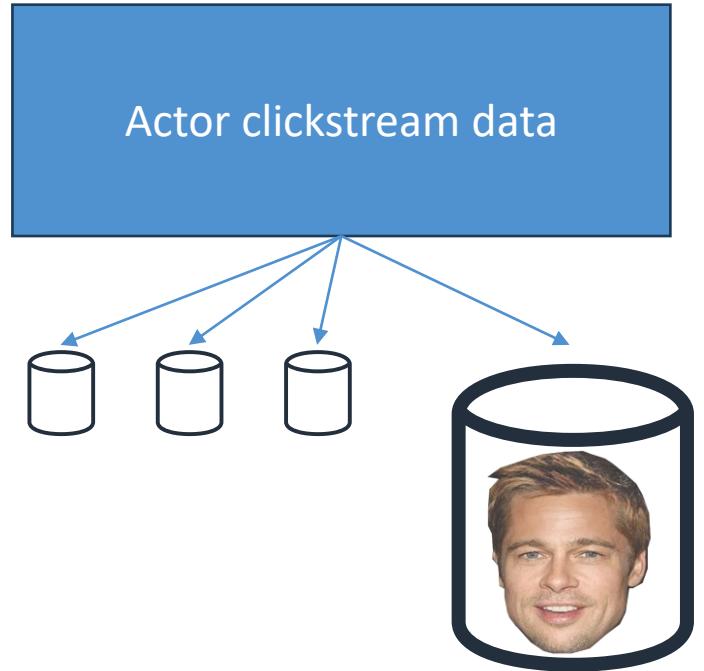


# Systemic Sampling



# Data Skew Mechanisms

- Data skew refers to the unequal distribution or imbalance of data across various nodes or partitions in distributed computing systems.
- “The celebrity problem”
  - Even partitioning doesn’t work if your traffic is uneven
  - Imagine you’re IMDb... Brad Pitt could overload his partition
- Causes:
  - Non-uniform distribution of data
  - Inadequate partitioning strategy
  - Temporal skew
- Important to monitor data distribution and alert when skew issues arise.



# Addressing Data Skew

- 1. Adaptive Partitioning:** Dynamically adjust partitioning based on data characteristics to ensure a more balanced distribution.
- 2. Salting:** Introduce a random factor or "salt" to the data to distribute it more uniformly.
- 3. Repartitioning:** Regularly redistribute the data based on its current distribution characteristics.
- 4. Sampling:** Use a sample of the data to determine the distribution and adjust the processing strategy accordingly.
- 5. Custom Partitioning:** Define custom rules or functions for partitioning data based on domain knowledge.

# Data Validation and Profiling

## 1. Completeness

- Definition:** Ensures all required data is present and no essential parts are missing.
- Checks:** Missing values, null counts, percentage of populated fields.
- Importance:** Missing data can lead to inaccurate analyses and insights.

## 2. Consistency

- Definition:** Ensures data values are consistent across datasets and do not contradict each other.
- Checks:** Cross-field validation, comparing data from different sources or periods.
- Importance:** Inconsistent data can cause confusion and result in incorrect conclusions.

## 3. Accuracy

- Definition:** Ensures data is correct, reliable, and represents what it is supposed to.
- Checks:** Comparing with trusted sources, validation against known standards or rules.
- Importance:** Inaccurate data can lead to false insights and poor decision-making.

## 4. Integrity

- Definition:** Ensures data maintains its correctness and consistency over its lifecycle and across systems.
- Checks:** Referential integrity (e.g., foreign key checks in databases), relationship validations.
- Importance:** Ensures relationships between data elements are preserved, and data remains trustworthy over time.

D6      ▾ | fx | =if(C6="", LOOKUP(A6,F3:G4), C6)

	A	B	C	D	E	F	G
1	User	MovielD	Rating	Imputed Rating			
2	Happy Henry	Big		4	4	User	avg Rating
3	Sad Sam	Forrest Gump		1	1	Happy Henry	4.5
4	Happy Henry	City Lights		5	5	Sad Sam	2
5	Sad Sam	Indiana Jones		2	2		
6	Happy Henry	Star Wars		4.5			
7	Sad Sam	Interstellar		3	3		
8							

# SQL

```
50  
51     SELECT empCode, empName, empSalary  
52     FROM Employee  
53     WHERE empName in  
54         (SELECT DISTINCT empName  
55         FROM population  
56         WHERE Country = "TH")  
57     AND empSalary >=  
58         (SELECT AVG(salary)  
59         FROM Salary  
60         WHERE gender = "M")  
61  
62  
63  
64  
65
```

## SQL Review

# Aggregation

- COUNT
  - `SELECT COUNT(*) AS total_rows FROM employees;`
- SUM
  - `SELECT SUM(salary) AS total_salary FROM employees;`
- AVG
  - `SELECT AVG(salary) AS average_salary FROM employees;`
- MAX / MIN
  - `SELECT MAX(salary) AS highest_salary FROM employees;`

# Aggregate with CASE

- WHERE clauses are specified after aggregation, so you can only filter on one thing at a time.

- ```
SELECT COUNT(*) AS high_salary_count
FROM employees
WHERE salary > 70000;
```

- One way to apply multiple filters to what you're aggregating

- ```
SELECT
    COUNT(CASE WHEN salary > 70000 THEN 1 END) AS high_salary_count,
    COUNT(CASE WHEN salary BETWEEN 50000 AND 70000 THEN 1 END) AS medium_salary_count,
    COUNT(CASE WHEN salary < 50000 THEN 1 END) AS low_salary_count
FROM employees;
```

# Grouping

Query:

sql

 Copy code

```
SELECT department_id, COUNT(*) AS number_of_employees
FROM employees
WHERE join_date > '2020-01-01'
GROUP BY department_id;
```

Sample Output:

diff

 Copy code

department_id	number_of_employees
HR	5
IT	8

# Nested grouping, sorting

Query:

sql

Copy code

```
SELECT YEAR(sale_date) AS sale_year, product_id, SUM(amount) AS total_sales
FROM sales
GROUP BY sale_year, product_id
ORDER BY sale_year, total_sales DESC;
```

Sample Output:

yaml

Copy code

sale_year	product_id	total_sales
2021	P001	5000
2021	P002	4000

# Pivoting

- Pivoting is the act of turning row-level data into columnar data.
- How this works is very database-specific. Some have a PIVOT command.
- For example, let's imagine we have a sales table that contains sales amounts and the salesperson:

Query:

```
sql
SELECT salesperson, [Jan] AS Jan_sales, [Feb] AS Feb_sales
FROM
(SELECT salesperson, month, sales FROM sales) AS sourceTable
PIVOT
(
    SUM(sales)
    FOR month IN ([Jan], [Feb])
) AS PivotTable;
```

Sample Output:

```
yaml
| salesperson | Jan_sales | Feb_sales |
|-----|-----|-----|
| Alice      | 1000     | 1100    |
| Bob        | 900      | 950     |
```

# Pivoting

- The same thing could be achieved with conditional aggregation, without requiring a specific PIVOT operation:

Query:

```
sql
Copy code

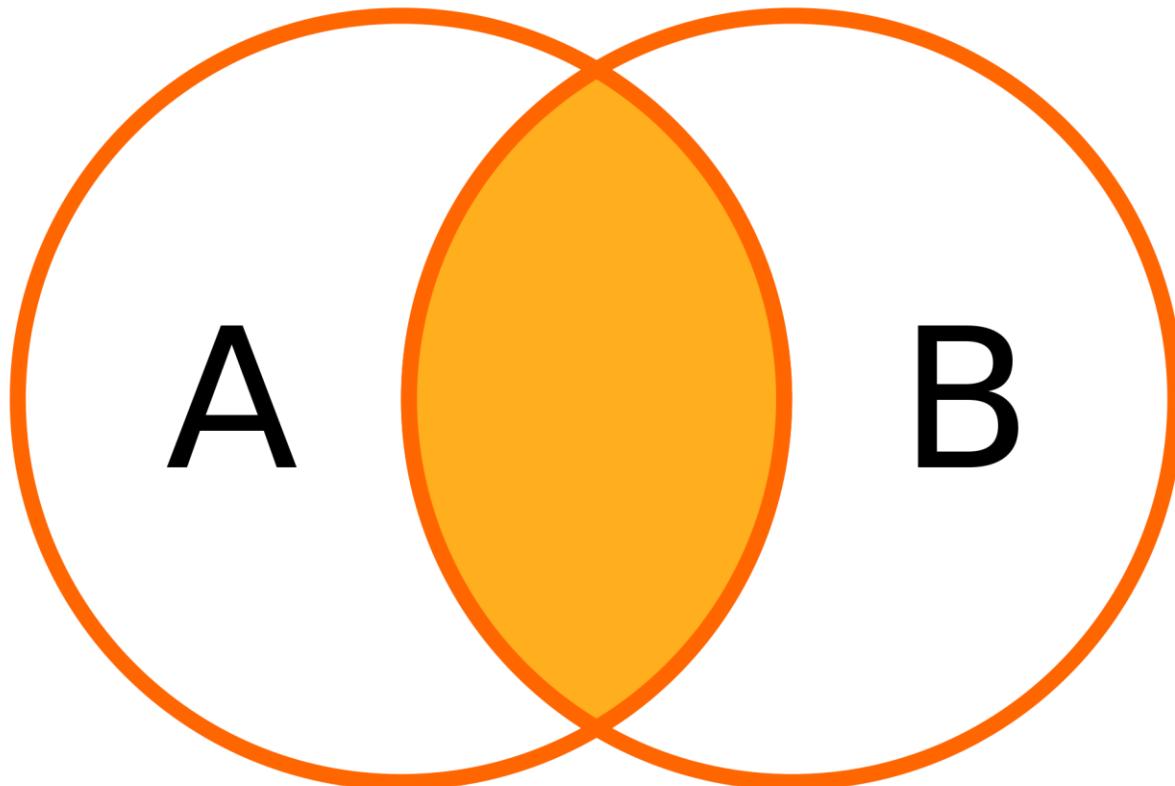
SELECT
    salesperson,
    SUM(CASE WHEN month = 'Jan' THEN sales ELSE 0 END) AS Jan_sales,
    SUM(CASE WHEN month = 'Feb' THEN sales ELSE 0 END) AS Feb_sales
FROM sales
GROUP BY salesperson;
```

Sample Output:

```
yaml
Copy code

| salesperson | Jan_sales | Feb_sales |
|-----|-----|-----|
| Alice       | 1000     | 1100      |
| Bob         | 900      | 950       |
```

# INNER JOIN

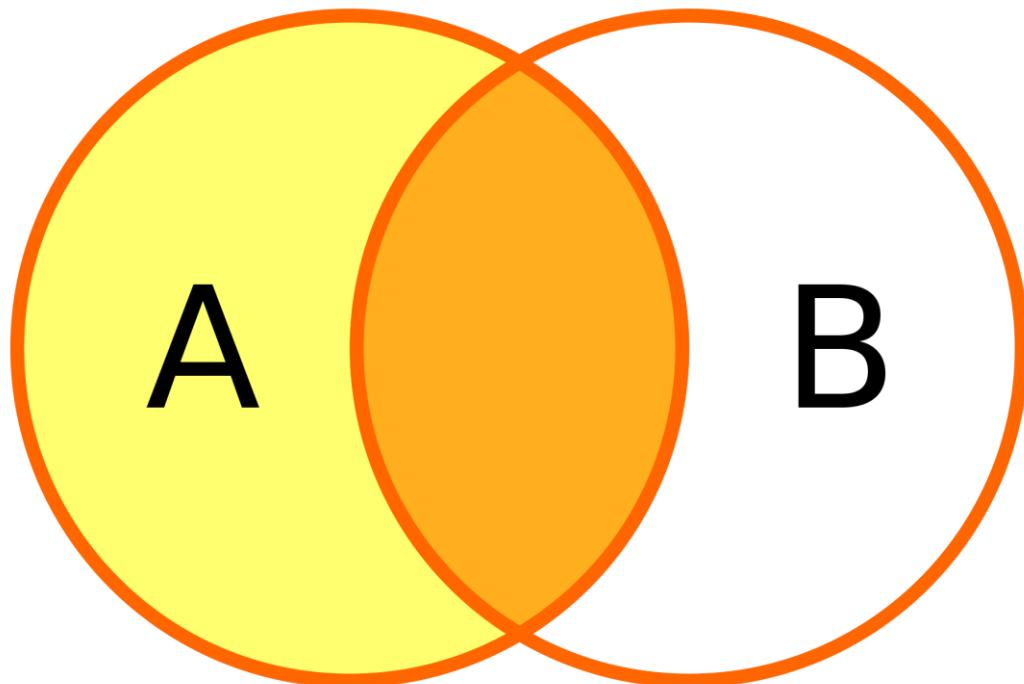


GermanX, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0/>>, via Wikimedia Commons

```
1 •  SELECT c.customerName, p.paymentDate, p.amount
2   FROM customers c
3   INNER JOIN payments p ON c.customerNumber = p.customerNumber
4
```

	customerName	paymentDate	amount
▶	Atelier graphique	2004-10-19	6066.78
	Atelier graphique	2003-06-05	14571.44
	Atelier graphique	2004-12-18	1676.14
	Signal Gift Stores	2004-12-17	14191.12
	Signal Gift Stores	2003-06-06	32641.98
	Signal Gift Stores	2004-08-20	33347.88
	Australian Collectors, Co.	2003-05-20	45864.03
	Australian Collectors, Co.	2004-12-15	82261.22
	Australian Collectors, Co.	2003-05-31	7565.08
	Australian Collectors, Co.	2004-03-10	44894.74
	La Rochelle Gifts	2004-11-14	19501.82
	La Rochelle Gifts	2004-08-08	47924.19
	La Rochelle Gifts	2005-02-22	49523.67
	Baane Mini Imports	2003-02-16	50218.95
	Baane Mini Imports	2003-10-28	1491.38
	Baane Mini Imports	2004-11-04	17876.32
	Baane Mini Imports	2004-11-28	34638.14
	Mini Gifts Distributors Ltd.	2005-03-05	101244.59
	Mini Gifts Distributors Ltd.	2004-08-28	85410.87
	Mini Gifts Distributors Ltd.	2003-04-11	11044.30
	Mini Gifts Distributors Ltd.	2005-04-16	83598.04
	Mini Gifts Distributors Ltd.	2004-12-27	47142.70
	Mini Gifts Distributors Ltd.	2004-11-02	55639.66
	Mini Gifts Distributors Ltd.	2003-08-15	111654.40

# LEFT OUTER JOIN



GermanX, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0>>, via Wikimedia Commons

Query 1 × mysqlsampledatabase

1 •    SELECT c.customerName, p.paymentDate, p.amount  
2      FROM customers c  
3      LEFT JOIN payments p ON c.customerNumber = p.customerNumber  
4

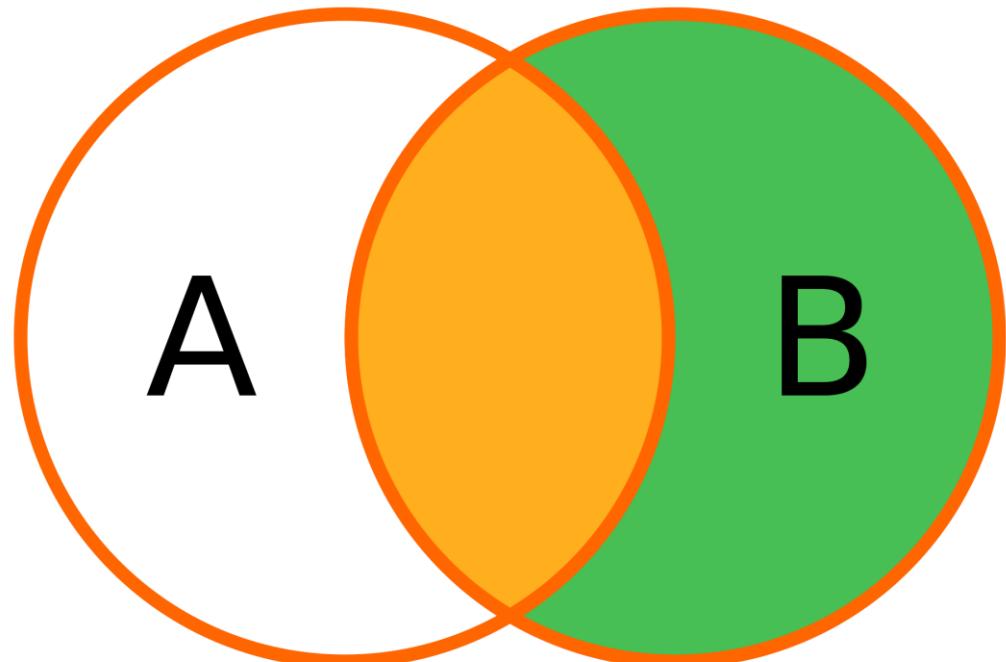
<

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	customerName	paymentDate	amount
1	Technics Stores Inc.	2004-11-14	2434.25
2	Technics Stores Inc.	2003-11-18	50743.65
3	Technics Stores Inc.	2005-02-02	12692.19
4	Technics Stores Inc.	2003-08-05	38675.13
5	Handji Gifts& Co	2004-09-16	38785.48
6	Handji Gifts& Co	2004-07-07	44160.92
7	Handji Gifts& Co	2004-02-28	22474.17
8	Herkku Gifts	2004-09-19	12538.01
9	Herkku Gifts	2003-12-03	85024.46
10	American Souvenirs Inc	NUL	NUL
11	Porto Imports Co.	NUL	NUL
12	Daedalus Designs Imports	2004-03-15	18997.89
13	Daedalus Designs Imports	2003-11-22	42783.81
14	La Corne D'abondance, ...	2004-09-09	1960.80
15	La Corne D'abondance, ...	2004-12-04	51209.58
16	La Corne D'abondance, ...	2003-04-20	33383.14
17	Cambridge Collectables Co.	2004-05-13	11843.45
18	Cambridge Collectables Co.	2004-03-29	20355.24
19	Gift Depot Inc.	2005-05-19	28500.78
20	Gift Depot Inc.	2003-11-19	24879.08
21	Gift Depot Inc.	2004-07-10	42044.77
22	Osaka Souveniers Co.	2004-04-17	15183.63
23	Osaka Souveniers Co.	2004-01-19	47177.59
24	Vitachrome Inc.	2004-04-25	22602.36

Result 6 ×

# RIGHT OUTER JOIN



GermanX, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0/>>, via Wikimedia Commons

Query 1 x mysqlsampledatabase

1 •    SELECT c.customerName, p.paymentDate, p.amount  
2      FROM payments p  
3      RIGHT JOIN customers c ON c.customerNumber = p.customerNumber  
4

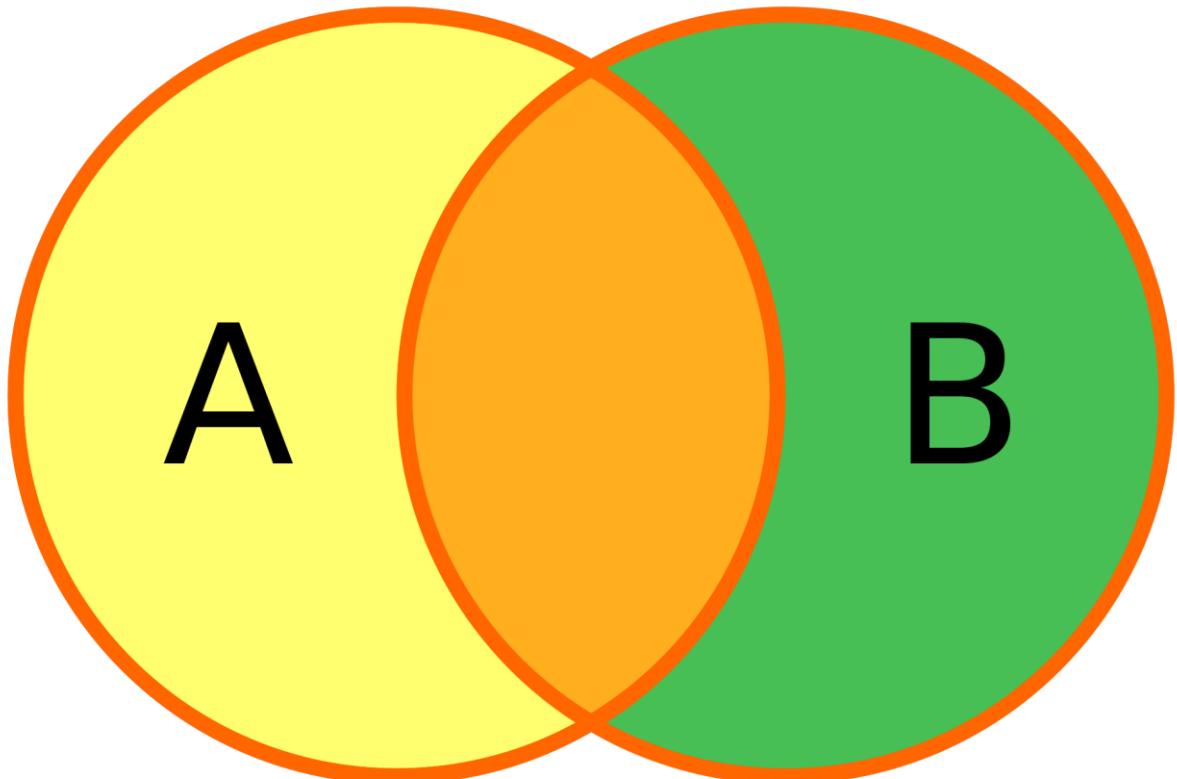
<

Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]

	customerName	paymentDate	amount
1	Muscle Machine Inc	2004-12-14	39964.63
2	Diecast Classics Inc.	2004-11-19	35152.12
3	Diecast Classics Inc.	2004-09-07	63357.13
4	Technics Stores Inc.	2004-11-14	2434.25
5	Technics Stores Inc.	2003-11-18	50743.65
6	Technics Stores Inc.	2005-02-02	12692.19
7	Technics Stores Inc.	2003-08-05	38675.13
8	Handji Gifts& Co	2004-09-16	38785.48
9	Handji Gifts& Co	2004-07-07	44160.92
10	Handji Gifts& Co	2004-02-28	22474.17
11	Herkku Gifts	2004-09-19	12538.01
12	Herkku Gifts	2003-12-03	85024.46
13	American Souvenirs Inc	NULL	NULL
14	Porto Imports Co.	NULL	NULL
15	Daedalus Designs Imports	2004-03-15	18997.89
16	Daedalus Designs Imports	2003-11-22	42783.81
17	La Corne D'abondance, ...	2004-09-09	1960.80
18	La Corne D'abondance, ...	2004-12-04	51209.58
19	La Corne D'abondance, ...	2003-04-20	33383.14
20	Cambridge Collectables Co.	2004-05-13	11843.45
21	Cambridge Collectables Co.	2004-03-29	20355.24
22	Gift Depot Inc.	2005-05-19	28500.78
23	Gift Depot Inc.	2003-11-19	24879.08
24	Gift Depot Inc.	2004-07-10	42044.77

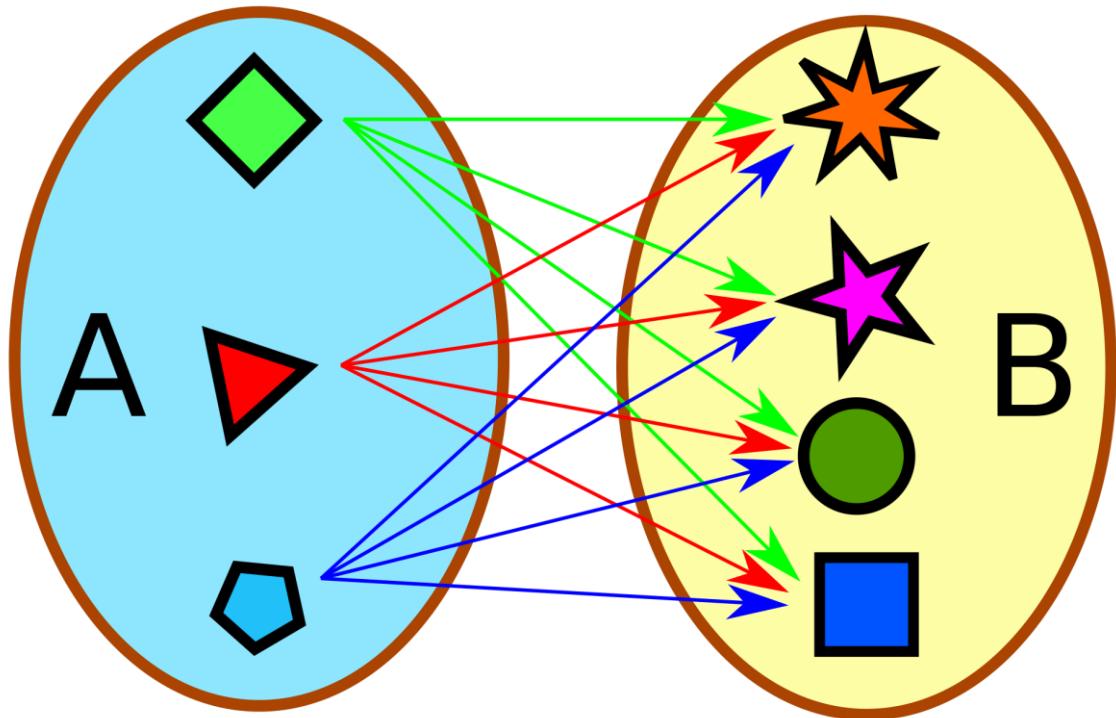
Result 7 x

# FULL OUTER JOIN



GermanX, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0/>>, via Wikimedia Commons

# CROSS OUTER JOIN



GermanX, CC BY-SA 4.0 <<https://creativecommons.org/licenses/by-sa/4.0>>, via Wikimedia Commons

Query 1 x mysqlsampledatabase

1 • SELECT c.customerName, p.paymentDate, p.amount  
2 FROM payments p  
3 CROSS JOIN customers c  
4

Result Grid | Filter Rows: Export: Wrap Cell Content: Fetch rows:

	customerName	paymentDate	amount
▶	Kelly's Gift Shop	2004-10-19	6066.78
	Diecast Collectables	2004-10-19	6066.78
	Double Decker Gift Stores, Ltd.	2004-10-19	6066.78
	Signal Collectibles Ltd.	2004-10-19	6066.78
	Motor Mint Distributors Inc.	2004-10-19	6066.78
	Iberia Gift Imports, Corp.	2004-10-19	6066.78
	Raanan Stores, Inc	2004-10-19	6066.78
	Kremlin Collectables, Co.	2004-10-19	6066.78
	Mit Vergnügen & Co.	2004-10-19	6066.78
	West Coast Collectables Co.	2004-10-19	6066.78
	Frau da Collezione	2004-10-19	6066.78
	Australian Collectables, Ltd	2004-10-19	6066.78
	Anton Designs, Ltd.	2004-10-19	6066.78
	FunGiftIdeas.com	2004-10-19	6066.78
	Warburg Exchange	2004-10-19	6066.78
	Corrida Auto Replicas, Ltd	2004-10-19	6066.78
	Microscale Inc.	2004-10-19	6066.78
	Super Scale Inc.	2004-10-19	6066.78
	Mini Auto Werke	2004-10-19	6066.78
	The Sharp Gifts Warehouse	2004-10-19	6066.78
	Scandinavian Gift Ideas	2004-10-19	6066.78
	Gift Ideas Corp.	2004-10-19	6066.78
	Feuer Online Stores, Inc	2004-10-19	6066.78
	Classic Legends Inc.	2004-10-19	6066.78

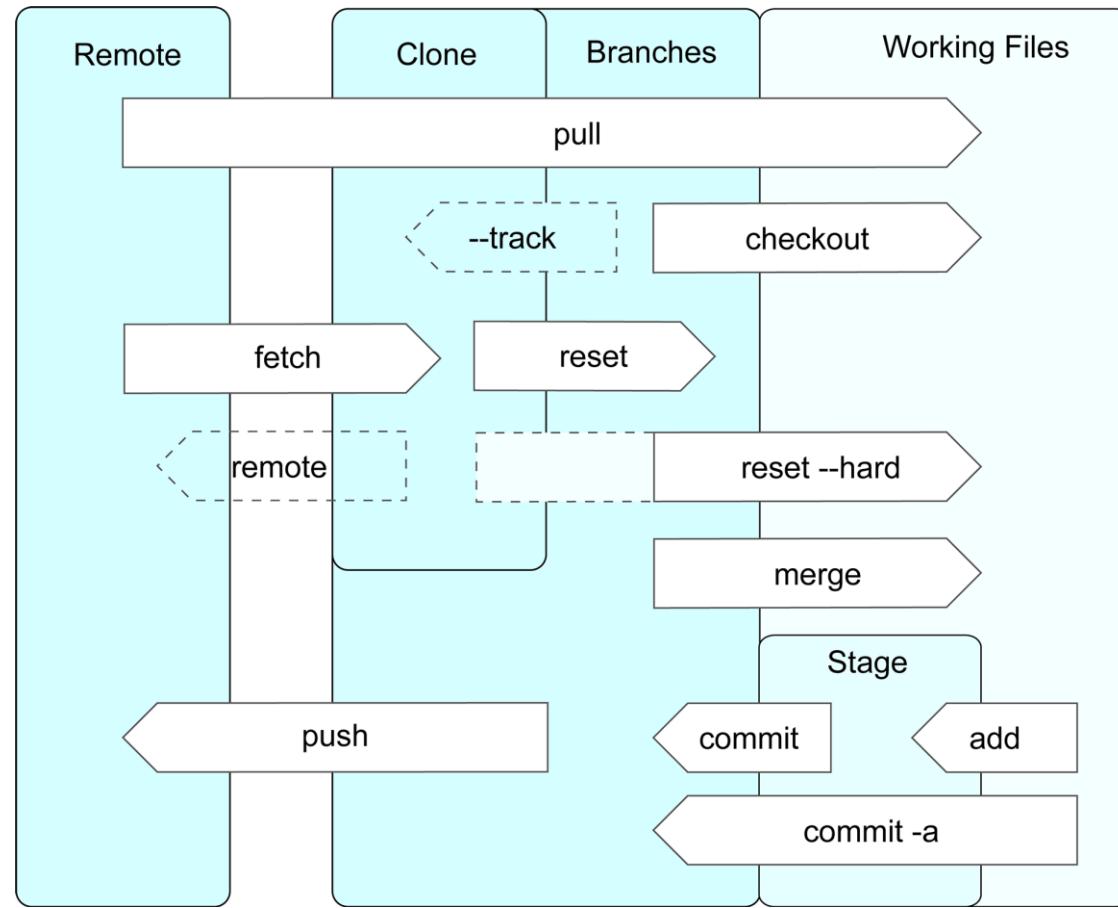
Result 9 x

# SQL Regular Expressions

- Pattern matching
  - Think a much more powerful “LIKE”
- `~` is the regular expression operator
- `~*` is case-insensitive
- `!~*` would mean “not match expression, case insensitive”
- Regular expression 101
  - `^` - match a pattern at the start of a string
  - `$` - match a pattern at the end of a string (`boo$` would match `boo` but not `book`)
  - `|` - alternate characters (`sit|sat` matches both `sit` and `sat`)
  - Ranges (`[a-z]` matches any lower case letter)
  - Repeats (`[a-z]{4}` matches any four-letter lowercase word)
  - Special metacharacters
    - `\d` – any digit; `\w` – any letter, digit, or underscore, `\s` – whitespace, `\t` – tab
- Example:
  - `SELECT * FROM name WHERE name ~ * '^fire|ice');`
  - Selects any rows where the name starts with “fire” or “ice” (case insensitive)



# Git review



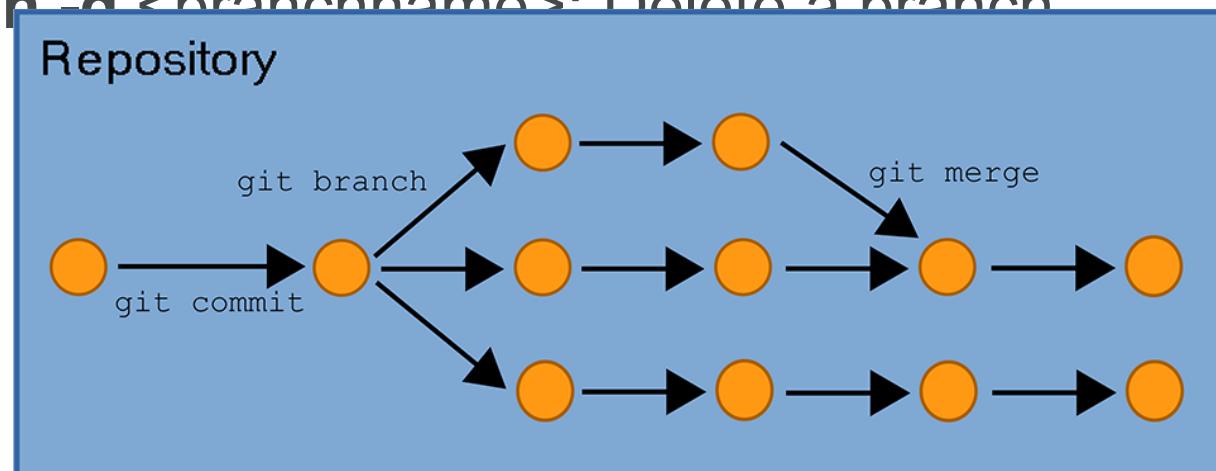
Daniel Kinzler, CC BY 3.0 <<https://creativecommons.org/licenses/by/3.0>>, via Wikimedia Commons

# Common git commands

- Setting Up and Configuration:
  - **git init**: Initialize a new Git repository.
  - **git config**: Set configuration values for user info, aliases, and more.
    - `git config --global user.name "Your Name"`: Set your name.
    - `git config --global user.email "your@email.com"`: Set your email.
- Basic Commands:
  - **git clone <repository>**: Clone (or download) a repository from an existing URL.
  - **git status**: Check the status of your changes in the working directory.
  - **git add <filename>**: Add changes in the file to the staging area.
    - `git add .`: Add all new and changed files to the staging area.
  - **git commit -m "Commit message here"**: Commit the staged changes with a message.
  - **git log**: View commit logs.

# Branching with git

- **git branch**: List all local branches.
  - `git branch <branchname>`: Create a new branch.
- **git checkout <branchname>**: Switch to a specific branch.
  - `git checkout -b <branchname>`: Create a new branch and switch to it.
- **git merge <branchname>**: Merge the specified branch into the current branch.
- **git branch -d <branchname>**: Delete a branch



Felix Dreissig, noris network AG

# Remote repositories

- **git remote add <name> <url>**: Add a remote repository.
- **git remote**: List all remote repositories.
- **git push <remote> <branch>**: Push a branch to a remote repository.
- **git pull <remote> <branch>**: Pull changes from a remote repository branch into the current local branch.

# Undoing changes

- **git reset**: Reset your staging area to match the most recent commit, without affecting the working directory.
- **git reset --hard**: Reset the staging area and the working directory to match the most recent commit.
- **git revert <commit>**: Create a new commit that undoes all of the changes from a previous commit.

# Advanced git

- **git stash**: Temporarily save changes that are not yet ready for a commit.
  - `git stash pop`: Restore the most recently stashed changes.
- **git rebase <branch>**: Reapply changes from one branch onto another, often used to integrate changes from one branch into another.
- **git cherry-pick <commit>**: Apply changes from a specific commit to the current branch.

# Git collaboration and inspection

- **git blame <file>**: Show who made changes to a file and when.
- **git diff**: Show changes between commits, commit and working tree, etc.
- **git fetch**: Fetch changes from a remote repository without merging them.

# Git maintenance and data recovery

- **git fsck**: Check the database for errors.
- **git gc**: Clean up and optimize the local repository.
- **git reflog**: Record when refs were updated in the local repository, useful for recovering lost commits.



# Storage

Storing, accessing, and backing up data in AWS

# 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**



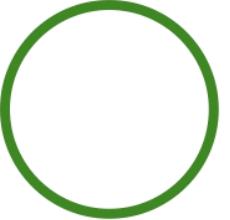
S3 Bucket

# 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 ("/")



# 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

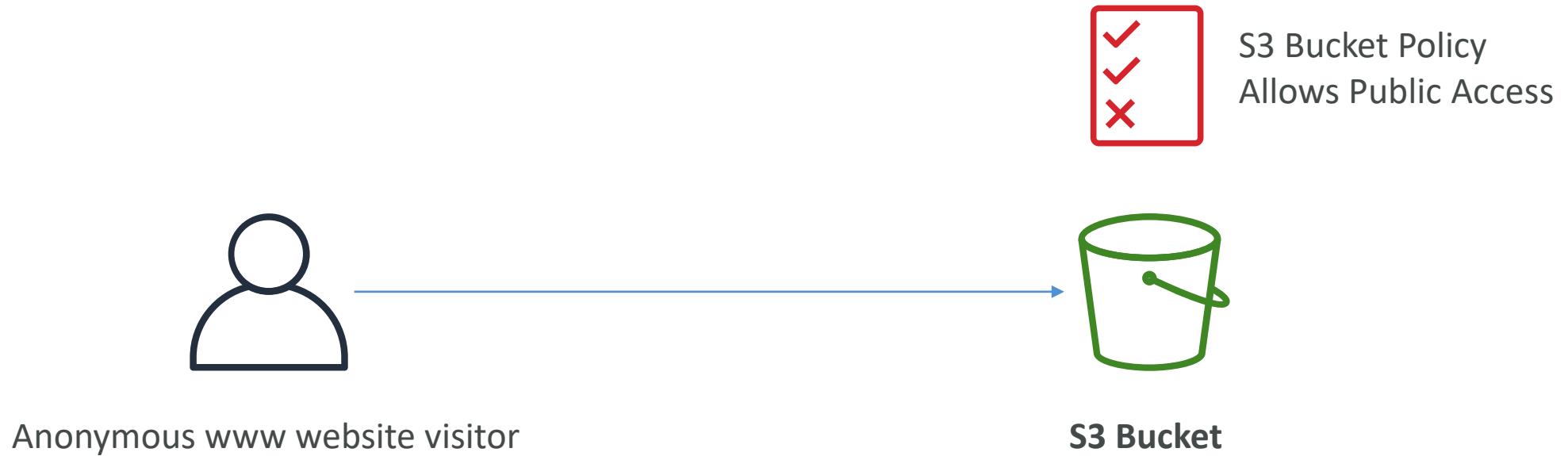
- **User-Based**
  - **IAM Policies** – which API calls should be allowed for a specific user from IAM
- **Resource-Based**
  - **Bucket Policies** – bucket wide rules from the S3 console - allows cross account
  - **Object Access Control List (ACL)** – finer grain (can be disabled)
  - **Bucket Access 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/*"  
      ]  
    }  
  ]  
}
```

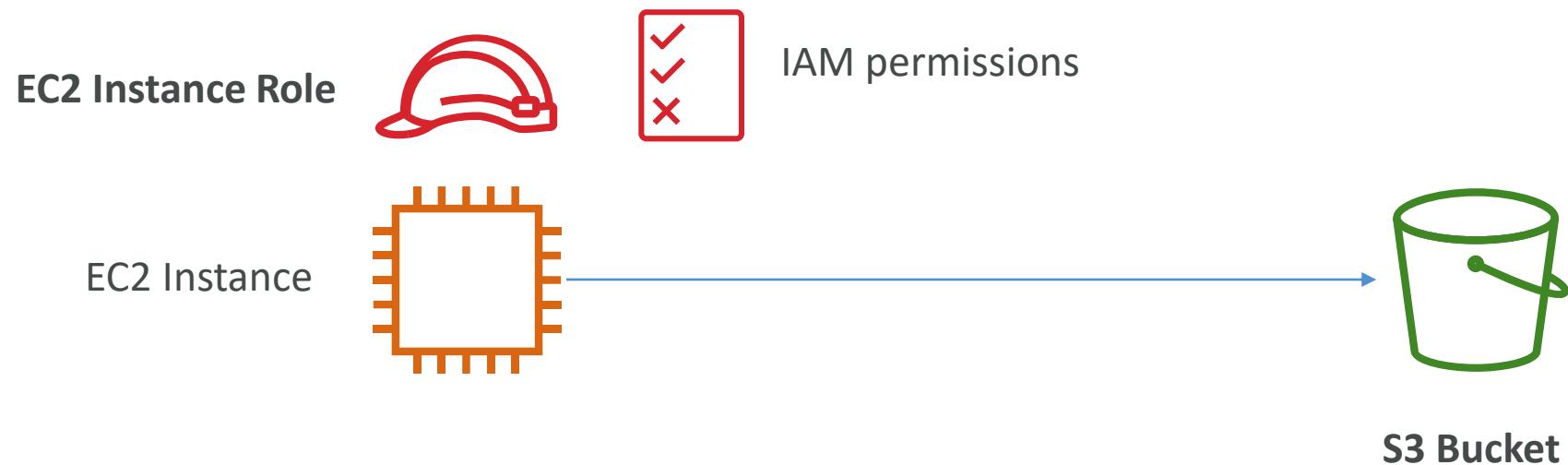
# Example: Public Access - Use Bucket Policy



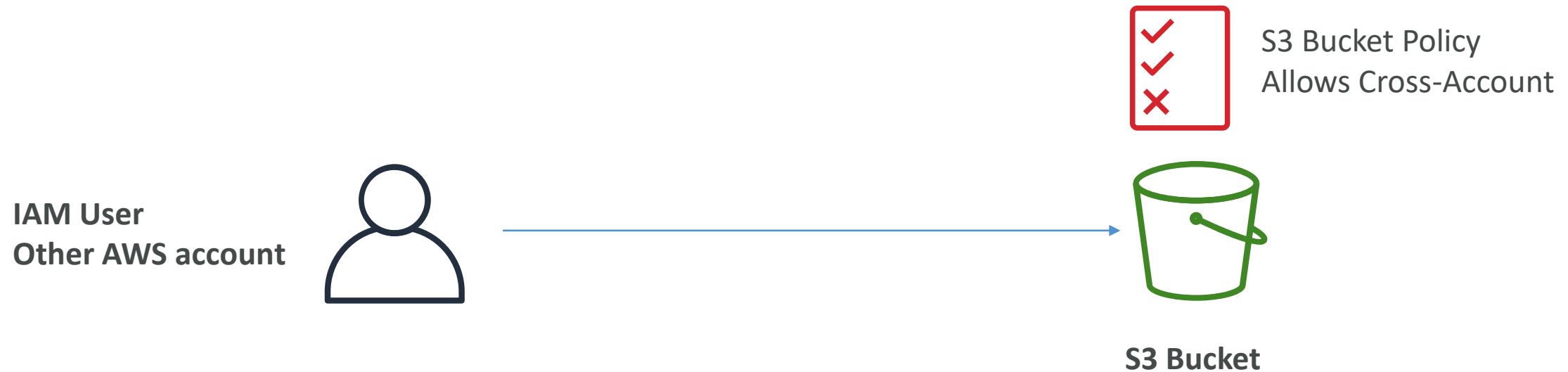
# Example: User Access to S3 – IAM permissions



# Example: EC2 instance access - Use IAM Roles



# Advanced: Cross-Account Access – Use Bucket Policy



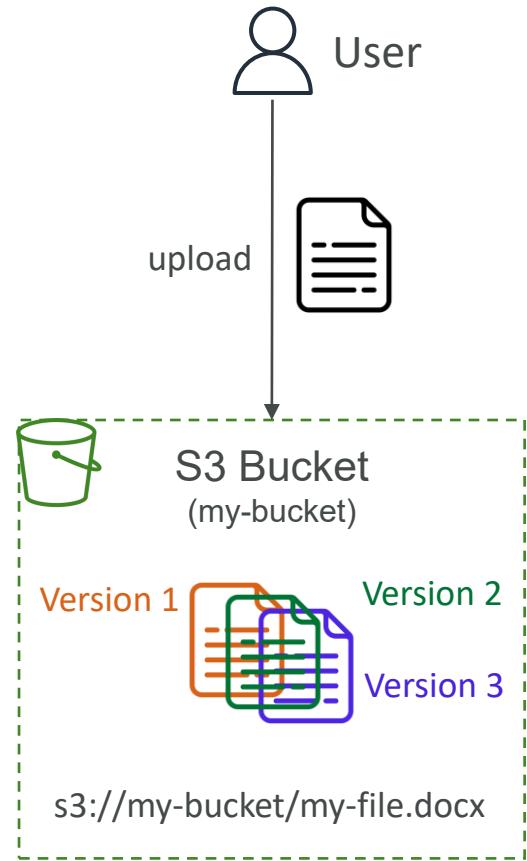
# Bucket settings for Block Public Access

```
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

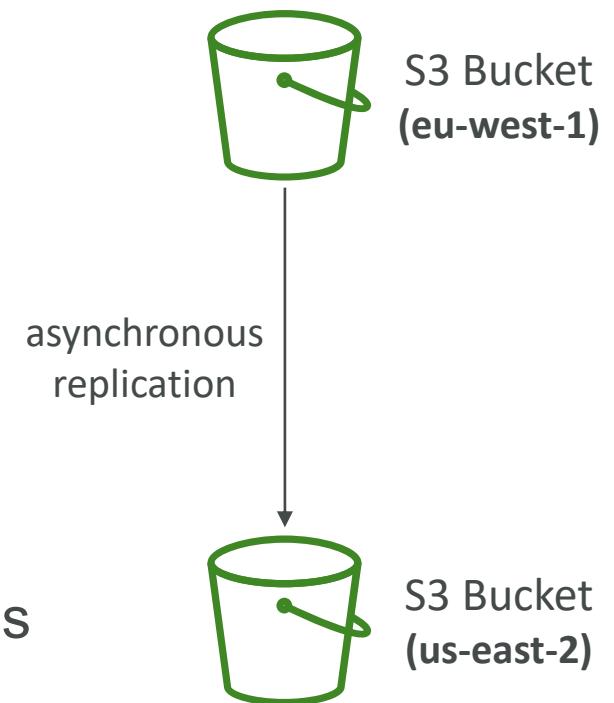
- You can version your files in Amazon S3
- It is enabled at the **bucket level**
- Same key overwrite will change the “version”: 1, 2, 3....
- It is best practice to version your buckets
  - Protect against unintended deletes (ability to restore a version)
  - Easy roll back to previous version
- Notes:
  - 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

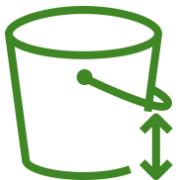


# S3 Standard – General Purpose

- 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

# 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

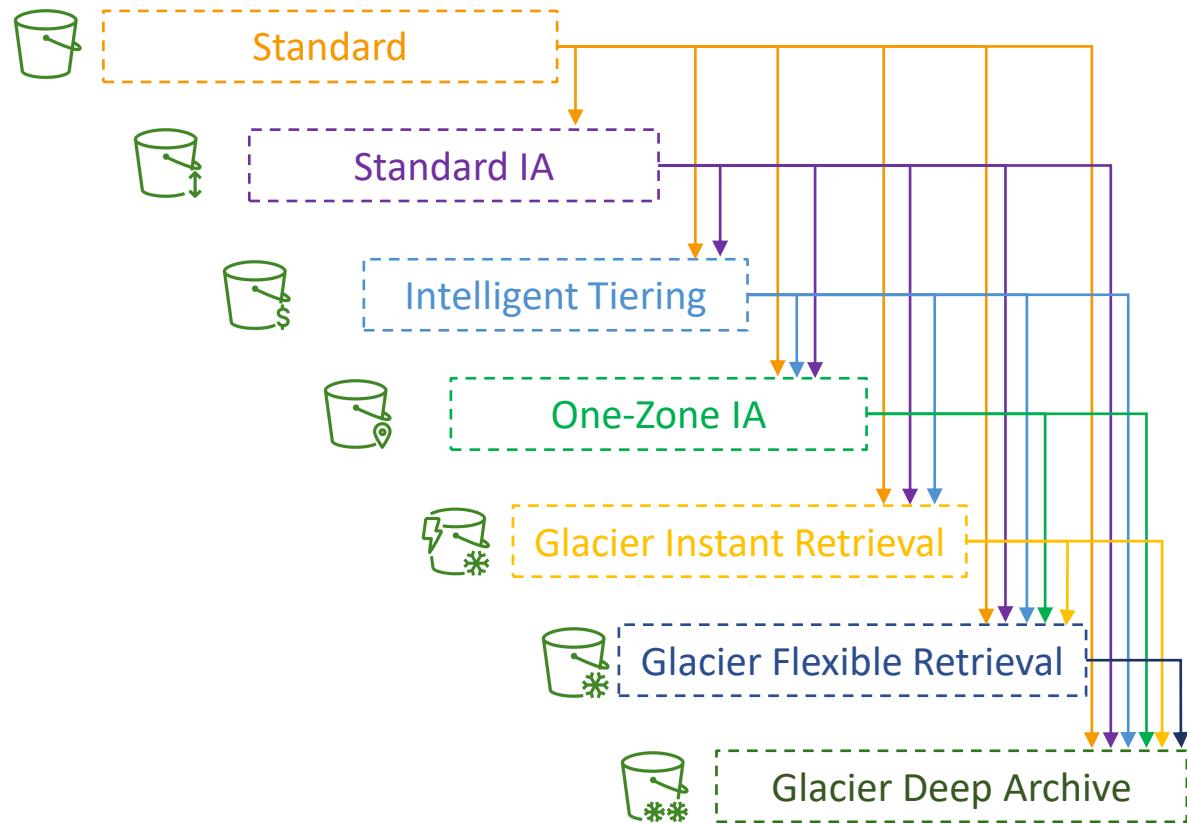
## Example: us-east-1

	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: \$0.0004</b> <b>POST: \$0.005</b>	<b>GET: \$0.0004</b> <b>POST: \$0.005</b>	<b>GET: \$0.001</b> <b>POST: \$0.01</b>	<b>GET: \$0.001</b> <b>POST: \$0.01</b>	<b>GET: \$0.01</b> <b>POST: \$0.02</b>	<b>GET: \$0.0004</b> <b>POST: \$0.03</b>  <b>Expedited: \$10</b> <b>Standard: \$0.05</b> <b>Bulk: free</b>	<b>GET: \$0.0004</b> <b>POST: \$0.05</b>  <b>Standard: \$0.10</b> <b>Bulk: \$0.025</b>
Retrieval Time	Instantaneous						<b>Expedited (1 – 5 mins)</b> <b>Standard (3 – 5 hours)</b> <b>Bulk (5 – 12 hours)</b>
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)

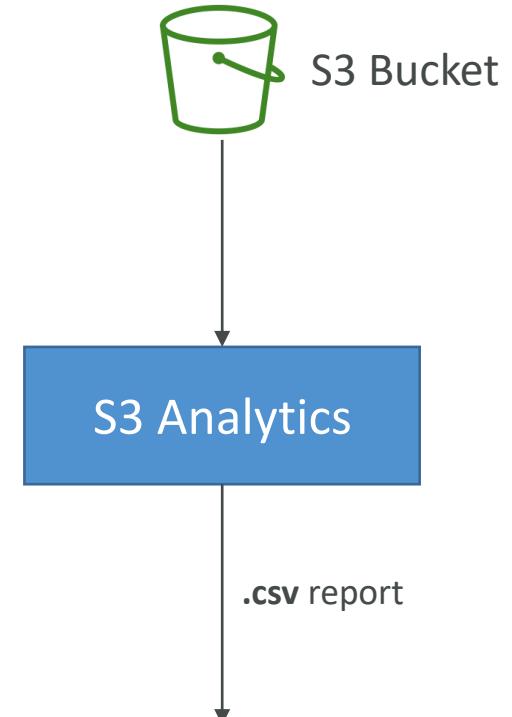
- Your application on EC2 creates images thumbnails after profile photos are uploaded to Amazon S3. These thumbnails can be easily recreated, and only need to be kept for 60 days. The source images should be able to be immediately retrieved for these 60 days, and afterwards, the user can wait up to 6 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

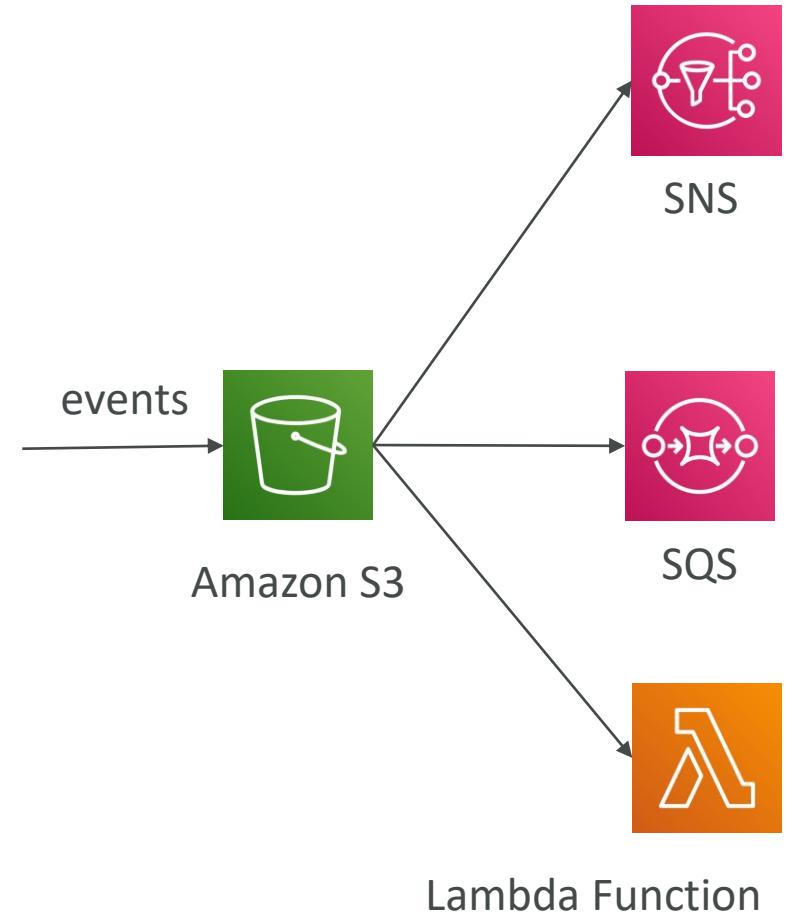
- Help you decide when to transition objects to the right storage class
- Recommendations for **Standard** and **Standard IA**
  - Does NOT work for One-Zone IA or Glacier
- Report is updated daily
- 24 to 48 hours to start seeing data analysis
- Good first step to put together Lifecycle Rules (or improve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



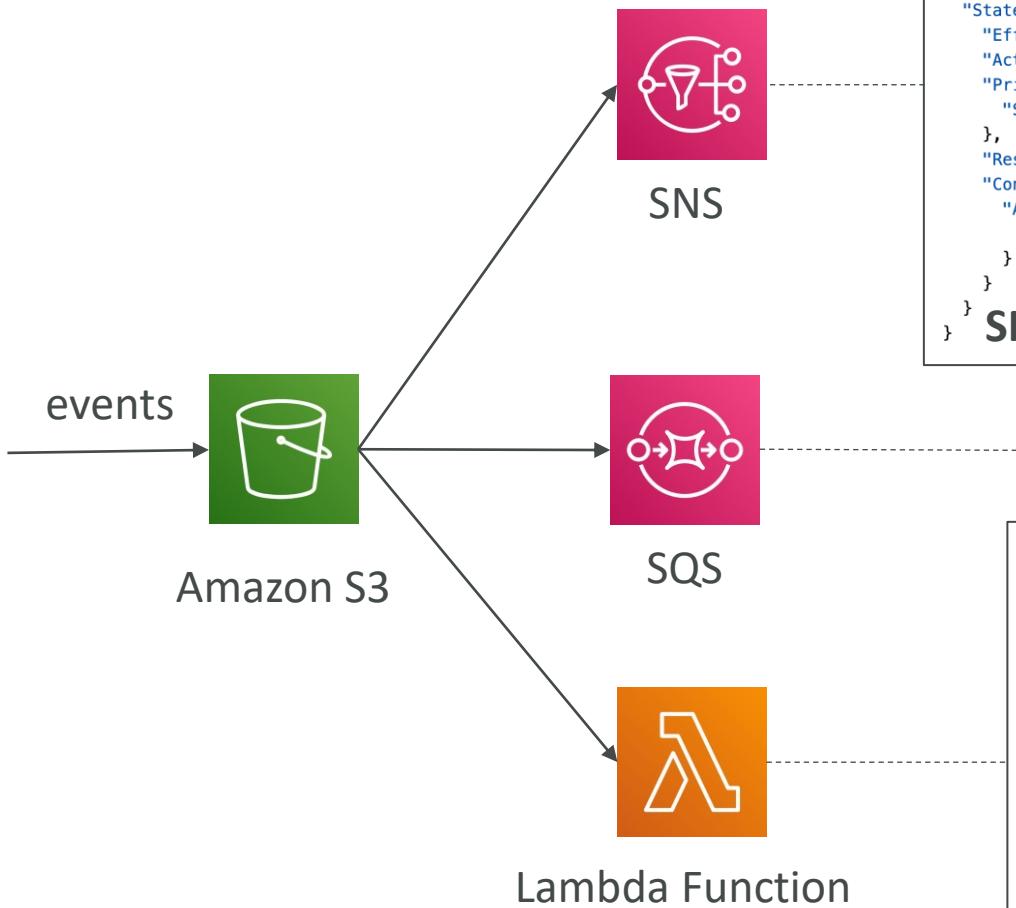
Date	StorageClass	ObjectAge
8/22/2022	STANDARD	000-014
8/25/2022	STANDARD	030-044
9/6/2022	STANDARD	120-149

# 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 – IAM Permissions

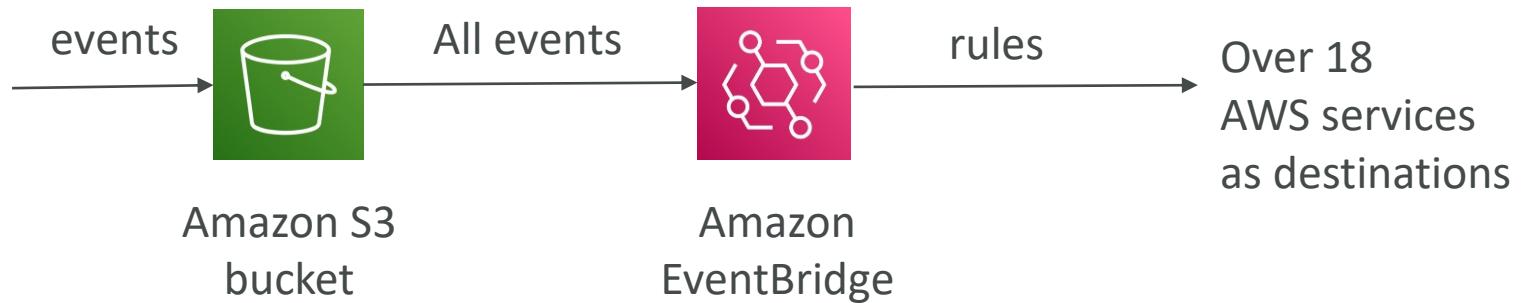


```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "SNS:Publish",  
        "Principal": {  
            "Service": "s3.amazonaws.com"  
        },  
        "Resource": "arn:aws:sns:us-east-1:123456789012:MyTopic",  
        "Condition": {  
            "ArnLike": {  
                "aws:SourceArn": "arn:aws:s3:::MyBucket"  
            }  
        }  
    }  
}  
SNS Resource (Access) Policy
```

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "SQS:SendMessage",  
        "Principal": {  
            "Service": "s3.amazonaws.com"  
        },  
        "Resource": "arn:aws:sqs:us-east-1:123456789012:MyQueue",  
        "Condition": {  
            "ArnLike": {  
                "aws:SourceArn": "arn:aws:s3:::MyBucket"  
            }  
        }  
    }  
}  
SQS Resource (Access) Policy
```

```
{  
    "Version": "2012-10-17",  
    "Statement": {  
        "Effect": "Allow",  
        "Action": "lambda:InvokeFunction",  
        "Principal": {  
            "Service": "s3.amazonaws.com"  
        },  
        "Resource": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",  
        "Condition": {  
            "ArnLike": {  
                "AWS:SourceArn": "arn:aws:s3:::MyBucket"  
            }  
        }  
    }  
}  
Lambda Resource Policy
```

# 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

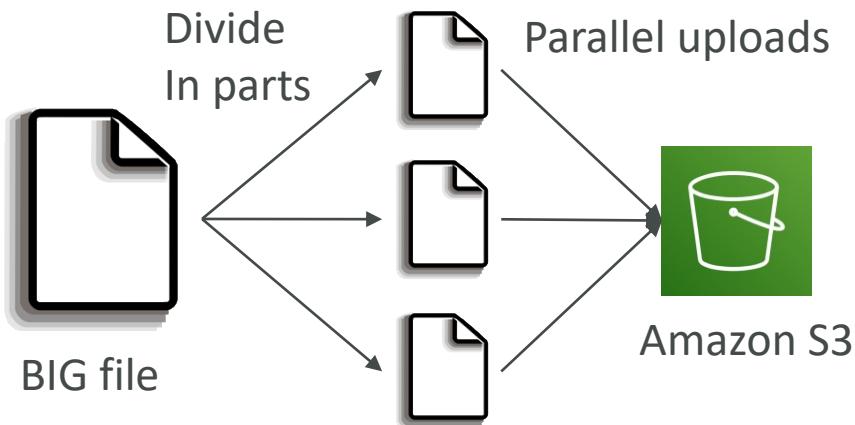
# S3 – Baseline Performance

- Amazon S3 automatically scales to high request rates, latency 100-200 ms
- Your application can achieve at least **3,500 PUT/COPY/POST/DELETE or 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

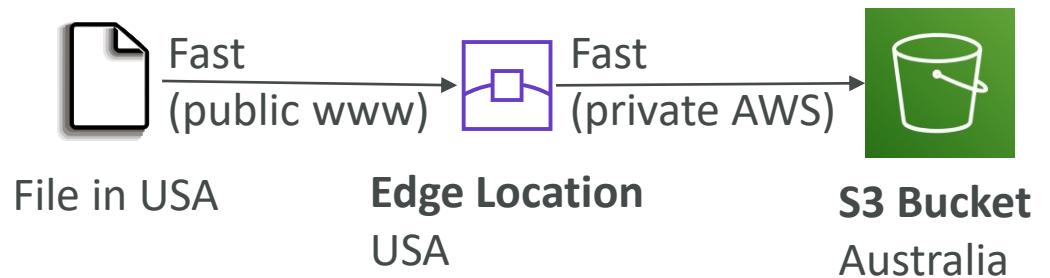
- **Multi-Part upload:**

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



- **S3 Transfer Acceleration**

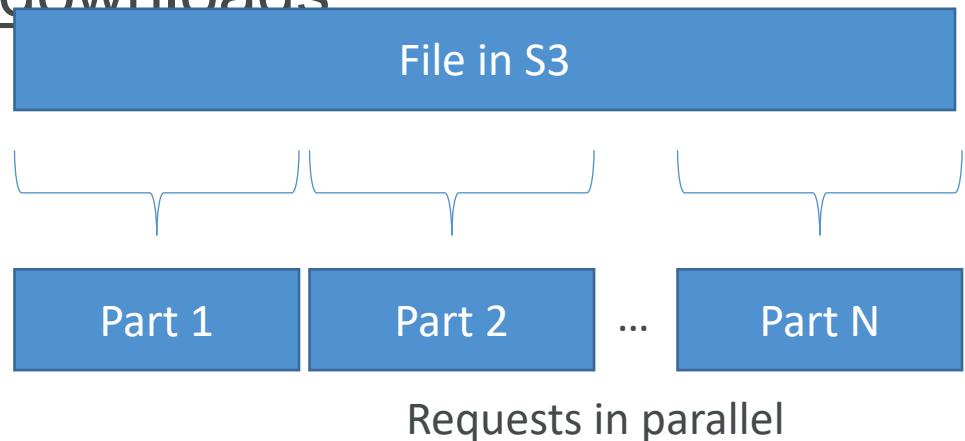
- Increase transfer speed by transferring file to an AWS edge location which will forward the data to the S3 bucket in the target region
- Compatible with multi-part upload



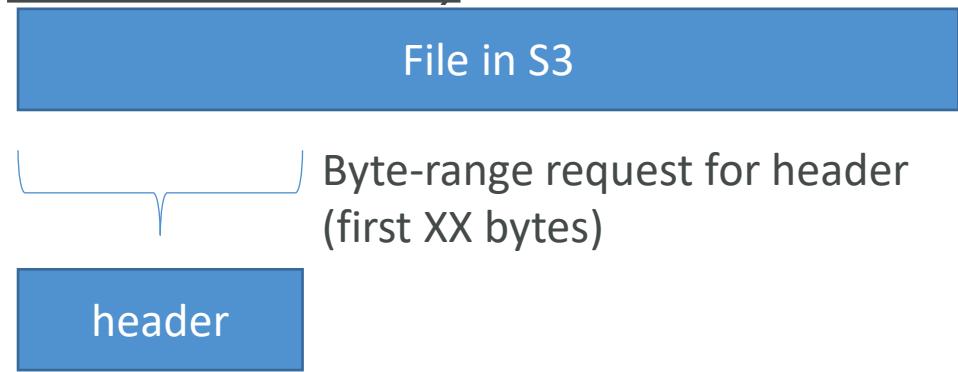
# S3 Performance – S3 Byte-Range Fetches

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

Can be used to speed up downloads

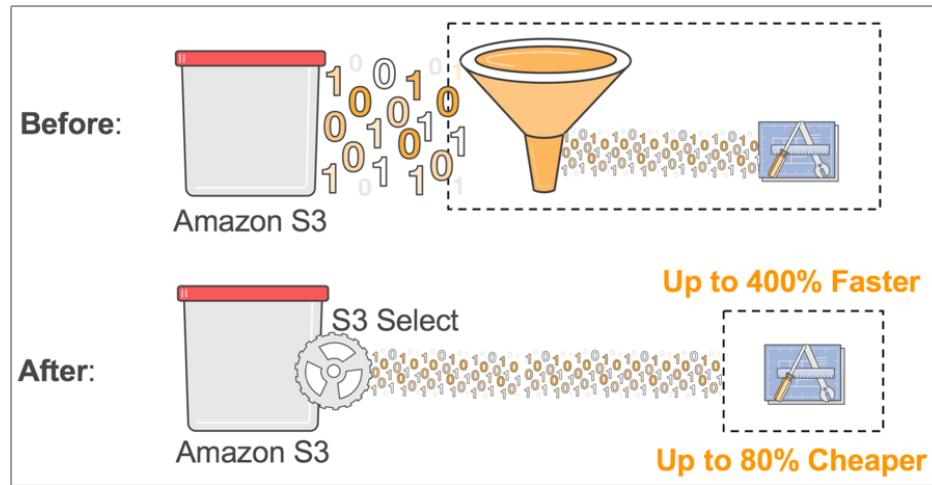


Can be used to retrieve only partial data (for example the head of a file)



# 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/>



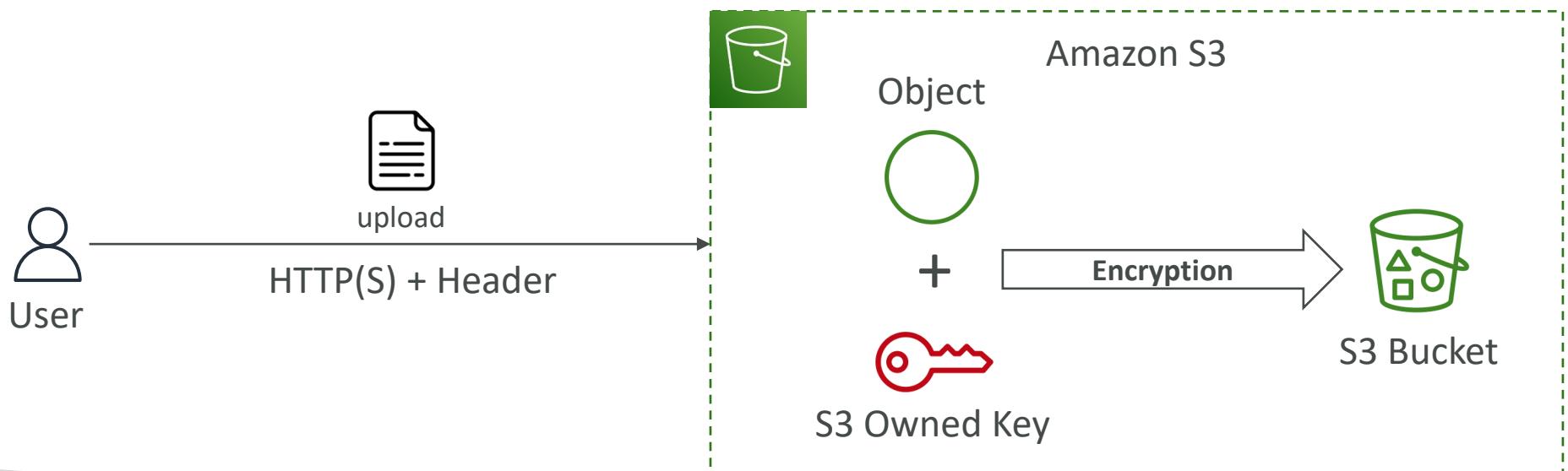


# Amazon S3 – Object Encryption

- 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) – Enabled by Default**
    - 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

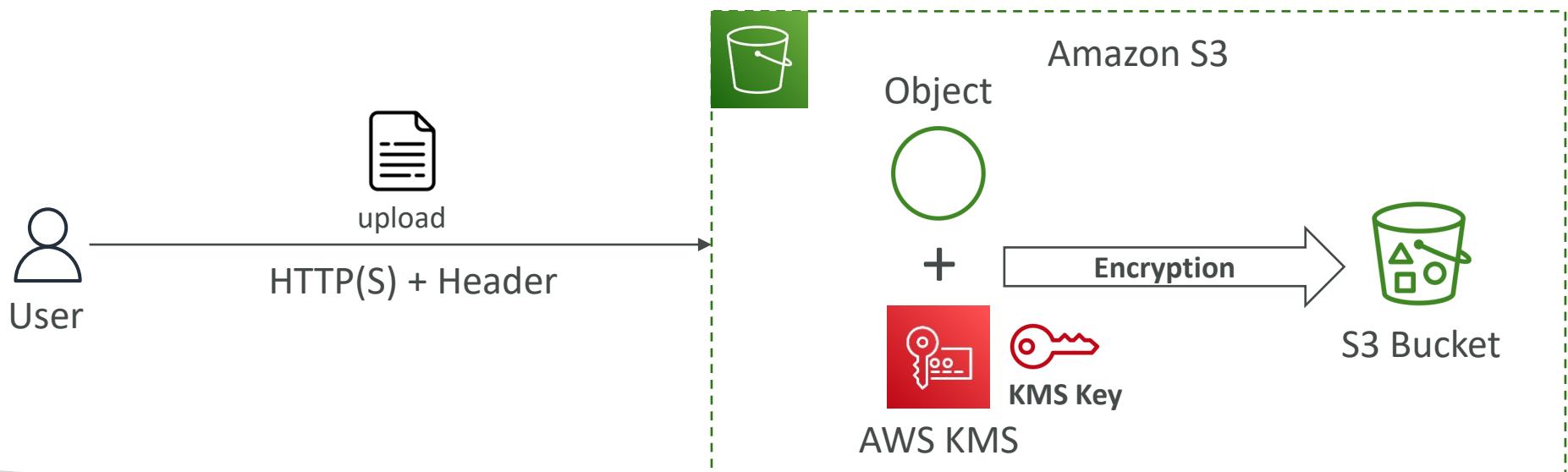
# Amazon S3 Encryption – SSE-S3

- 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"
- **Enabled by default for new buckets & new objects**



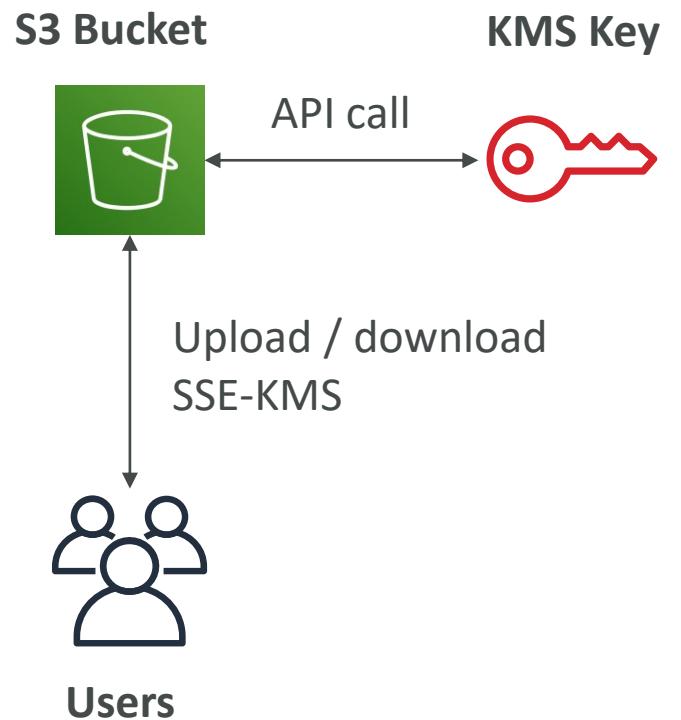
# Amazon S3 Encryption – SSE-KMS

- 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"**



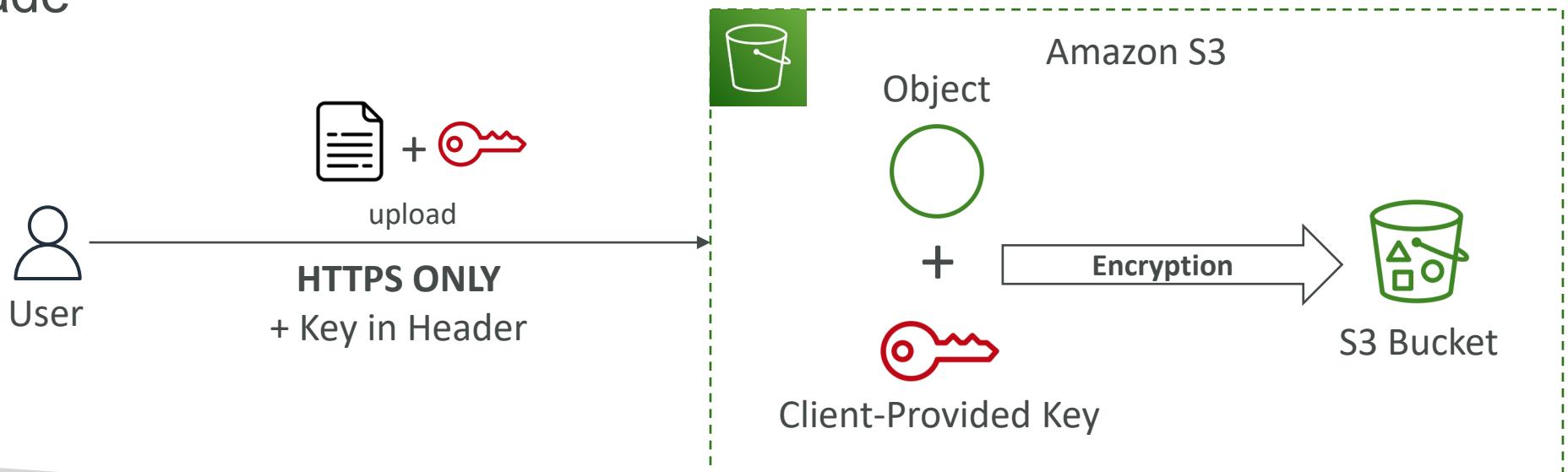
# SSE-KMS Limitation

- 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



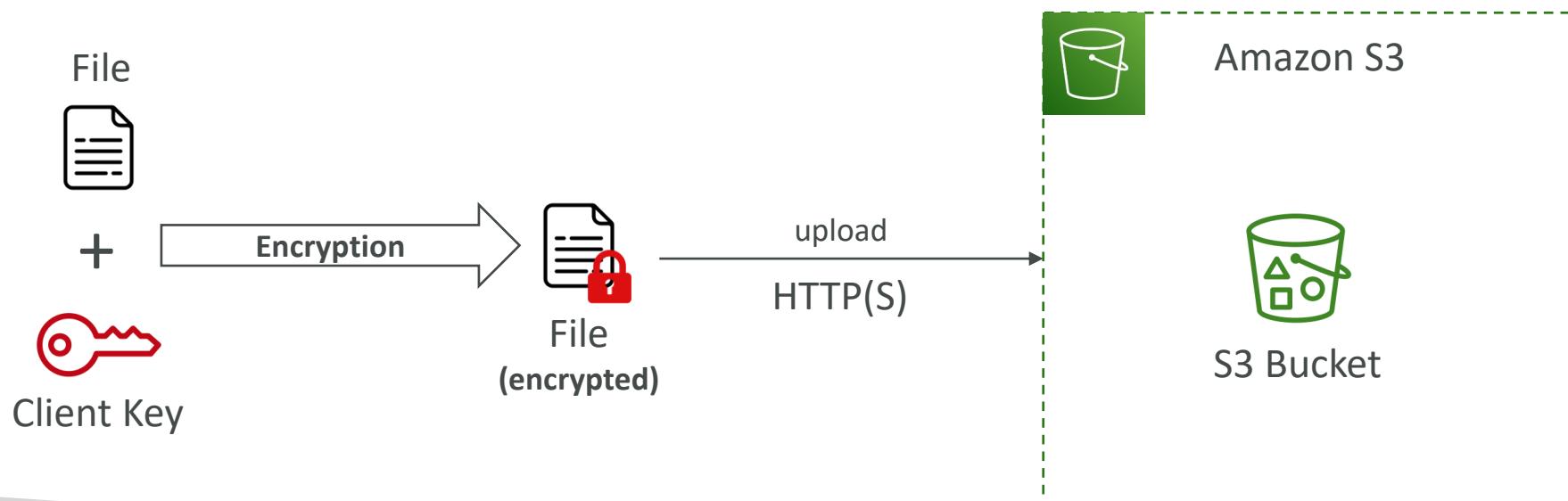
# Amazon S3 Encryption – SSE-C

- 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

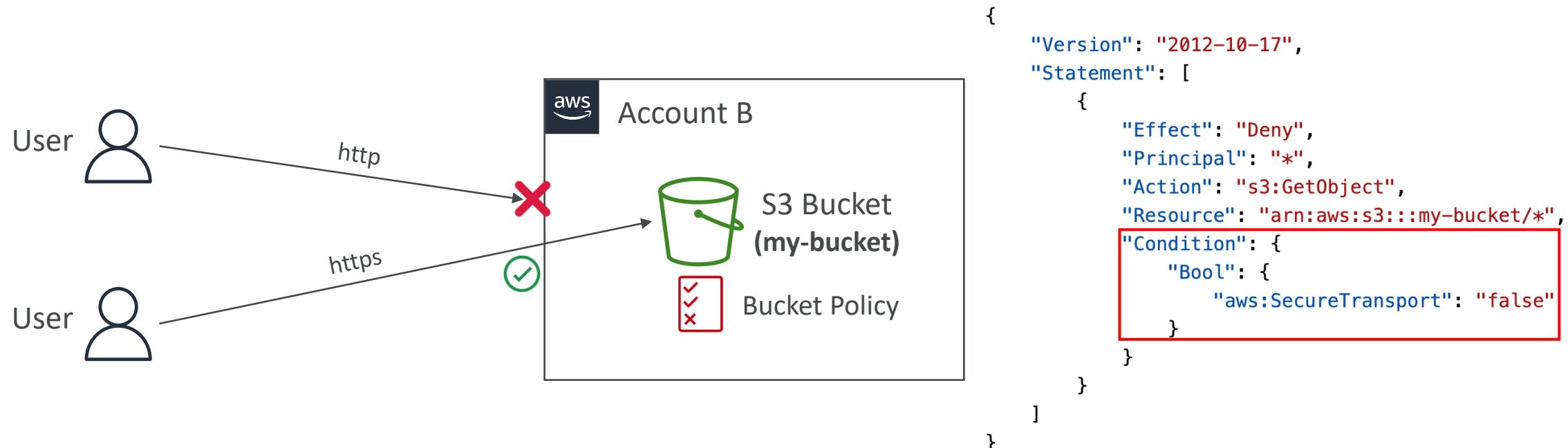


# Amazon S3 – Encryption in transit (SSL/TLS)

- 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 – Force Encryption in Transit aws:SecureTransport



# Amazon S3 – Default Encryption vs. Bucket Policies

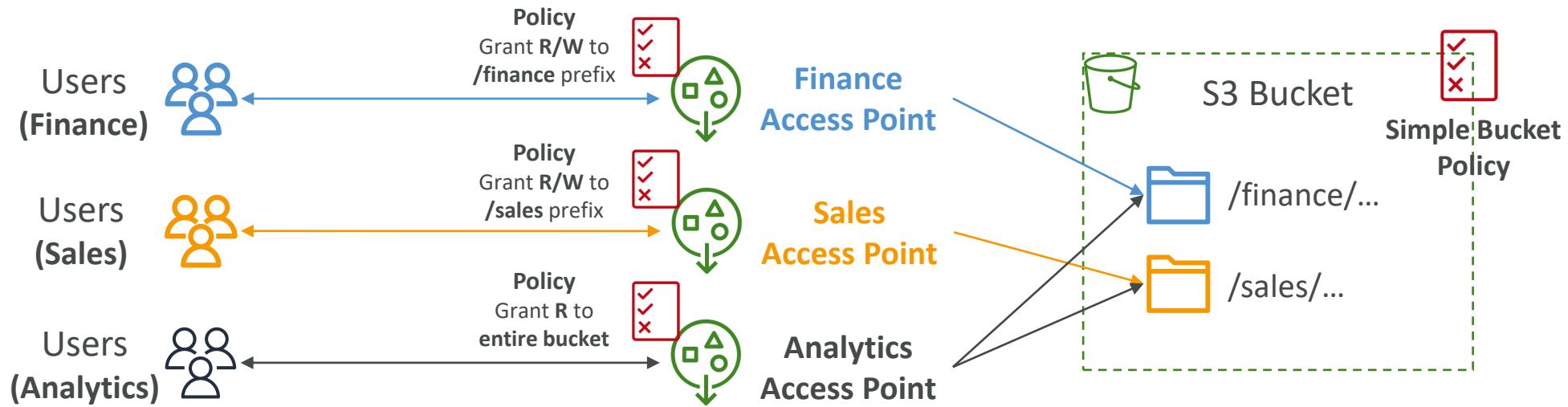
- SSE-S3 encryption is automatically applied to new objects stored in S3 bucket**
- Optionally, you can “force encryption” using a bucket policy and refuse any API call to PUT an S3 object without encryption headers (SSE-KMS or

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

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

- Note: Bucket Policies are evaluated before “Default Encryption”**

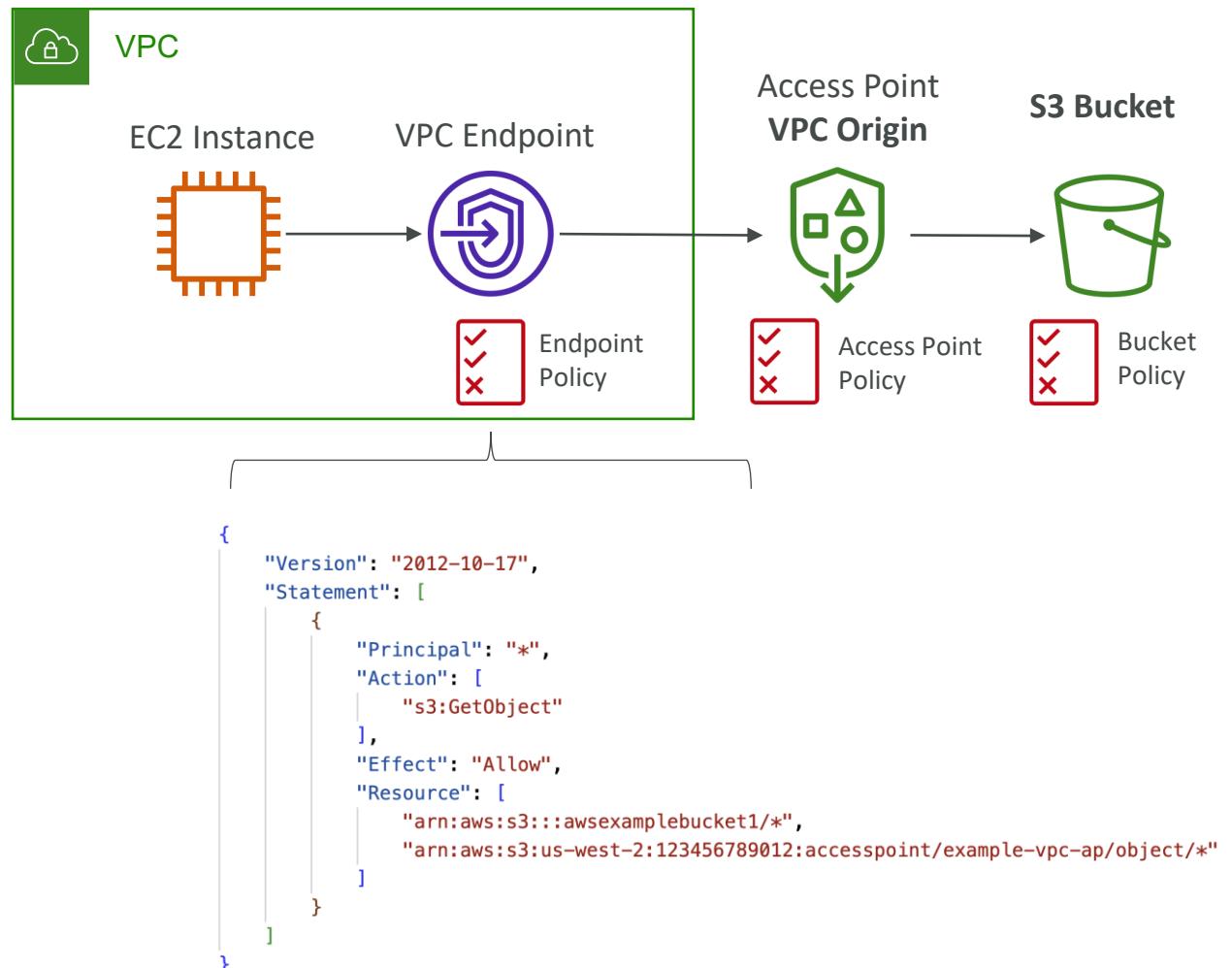
# S3 – Access Points



- Access Points simplify security management for S3 Buckets
- Each Access Point has:
  - its own DNS name (Internet Origin or VPC Origin)
  - an access point policy (similar to bucket policy) – manage security at scale

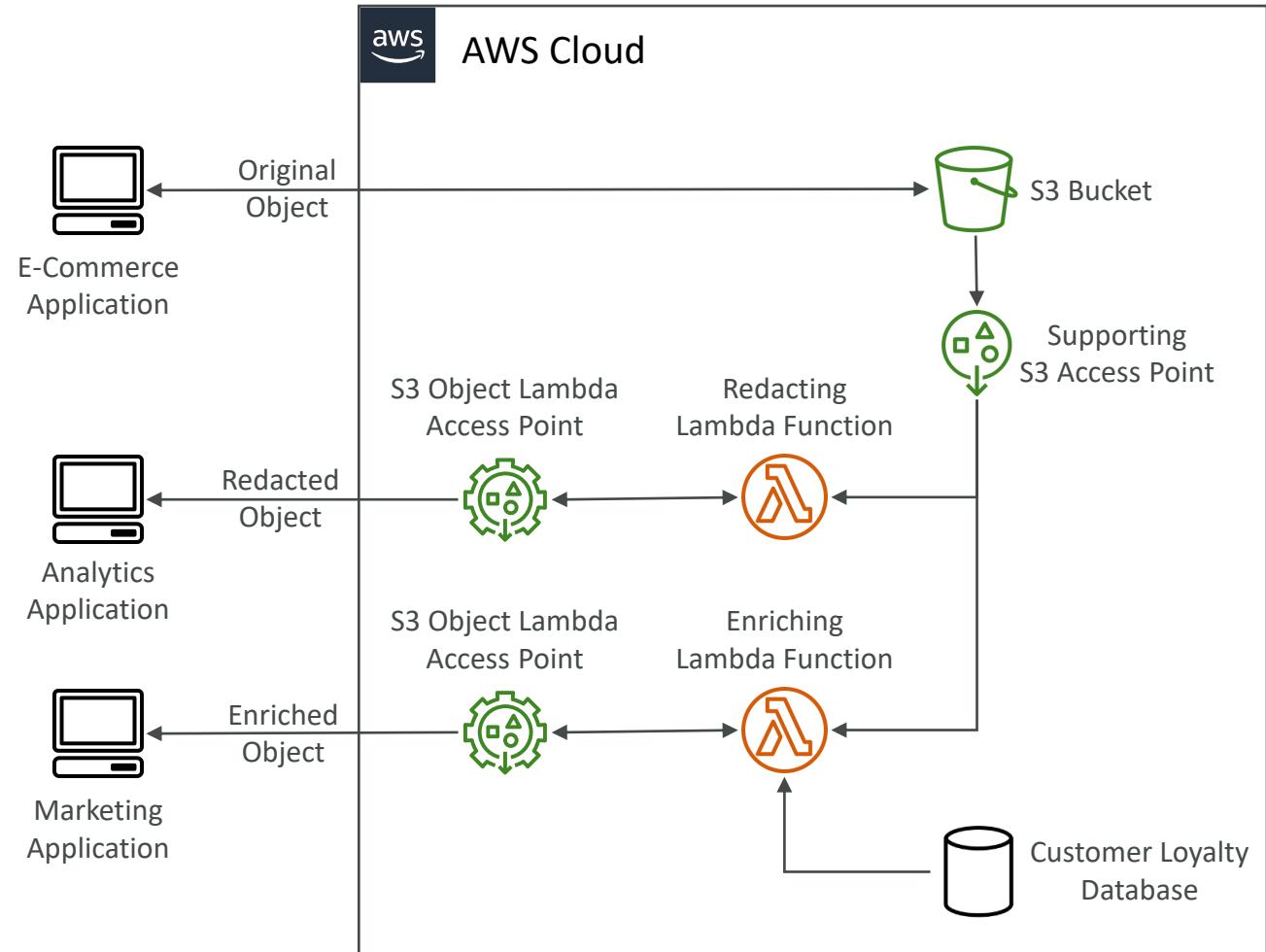
# S3 – Access Points – VPC Origin

- We can define the access point to be accessible only from within the VPC
- You must create a VPC Endpoint to access the Access Point (Gateway or Interface Endpoint)
- The VPC Endpoint Policy must allow access to the target bucket and Access Point



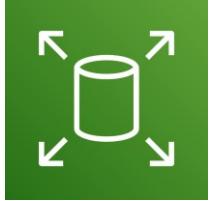
# S3 Object Lambda

- 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.



# EC2 Instance Storage Section

# What's an EBS Volume?

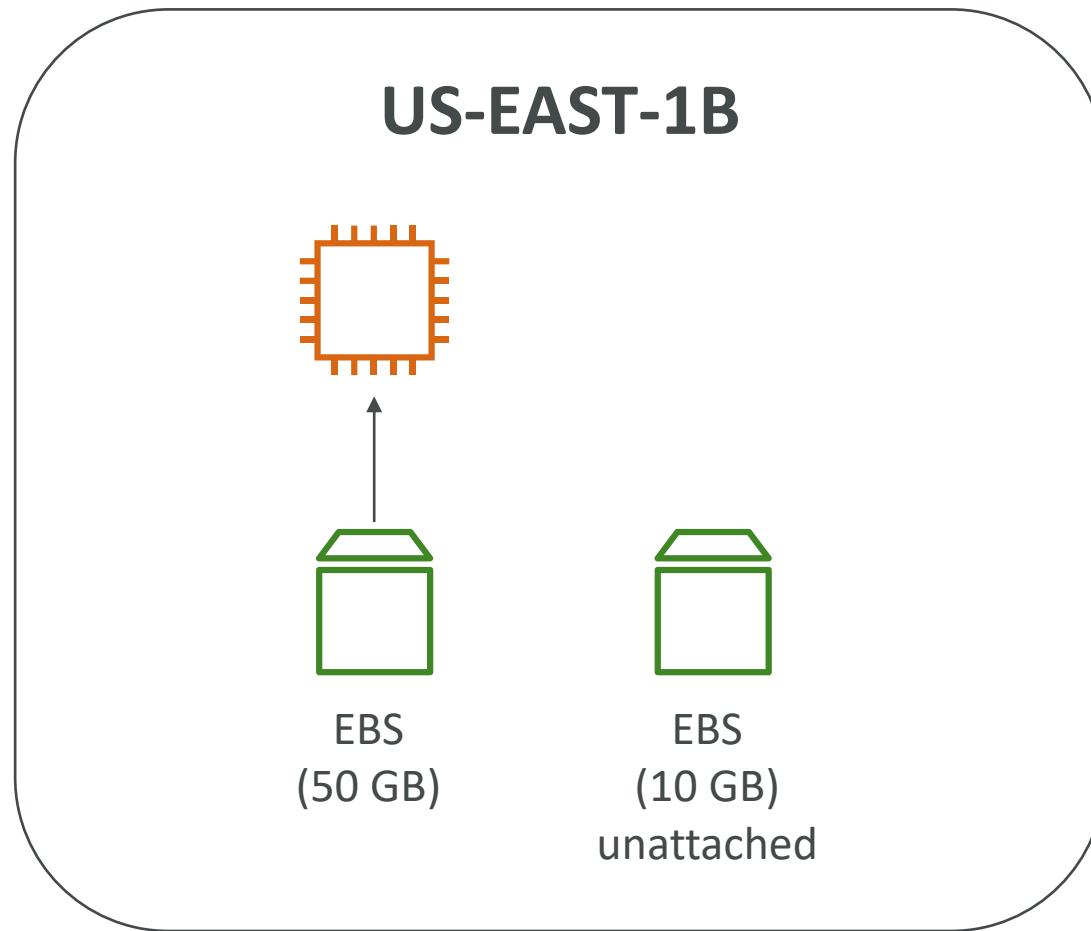
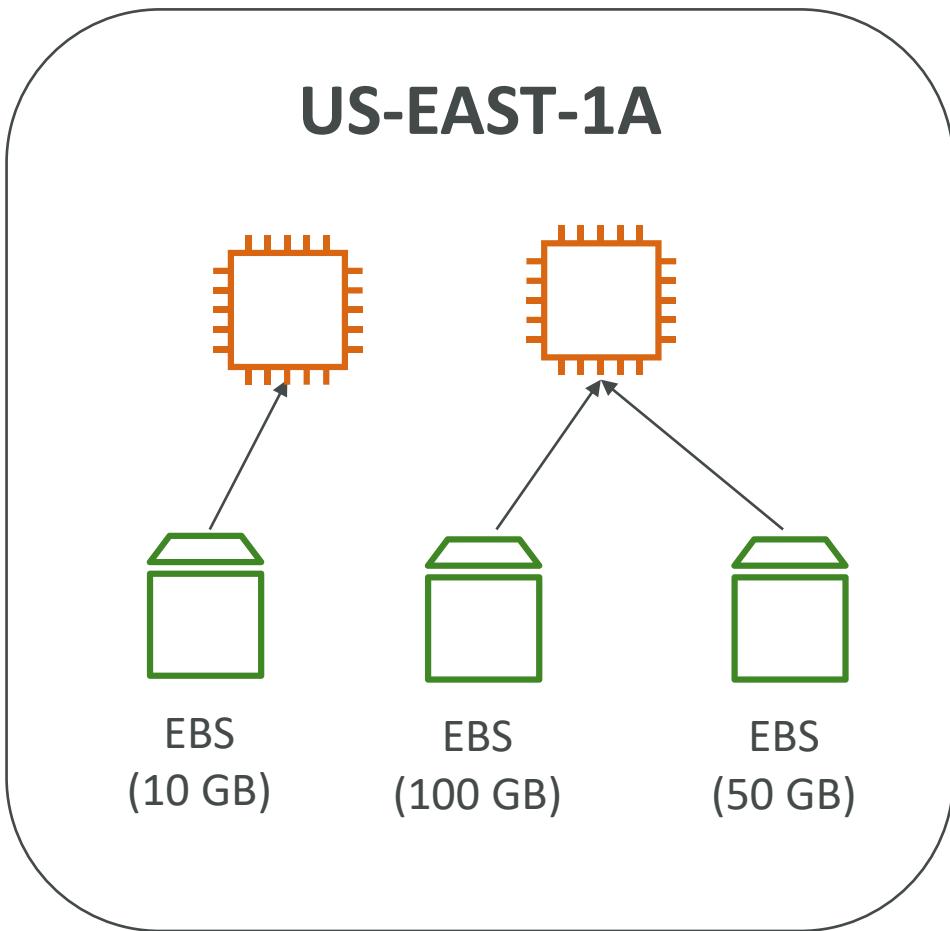


- An **EBS (Elastic Block Store) Volume** is a **network** drive you can attach to your instances while they run
- It allows your instances to persist data, even after their termination
- **They can only be mounted to one instance at a time** (at the CCP level)
- They are bound to **a specific availability zone**
- Analogy: Think of them as a “network USB stick”
- Free tier: 30 GB of free EBS storage of type General Purpose (SSD) or Magnetic per month

# EBS Volume

- It's a network drive (i.e. not a physical drive)
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- It's locked to an Availability Zone (AZ)
  - An EBS Volume in us-east-1a cannot be attached to us-east-1b
  - To move a volume across, you first need to snapshot it
- Have a provisioned capacity (size in GBs, and IOPS)
  - You get billed for all the provisioned capacity
  - You can increase the capacity of the drive over time

# EBS Volume - Example



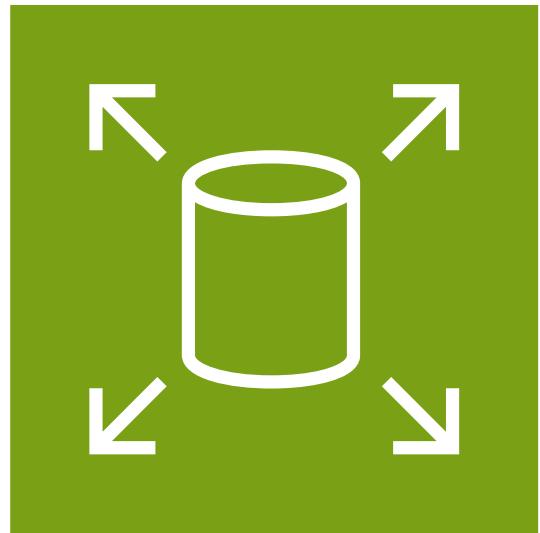
# EBS – Delete on Termination attribute

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-09f18f682fd23a1b1	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted
EBS	/dev/sdb	Search (case-insensit	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input type="checkbox"/>	Not Encrypted
<a href="#">Add New Volume</a>								

- Controls the EBS behaviour when an EC2 instance terminates
  - By default, the root EBS volume is deleted (attribute enabled)
  - By default, any other attached EBS volume is not deleted (attribute disabled)
- This can be controlled by the AWS console / AWS CLI
- **Use case: preserve root volume when instance is terminated**

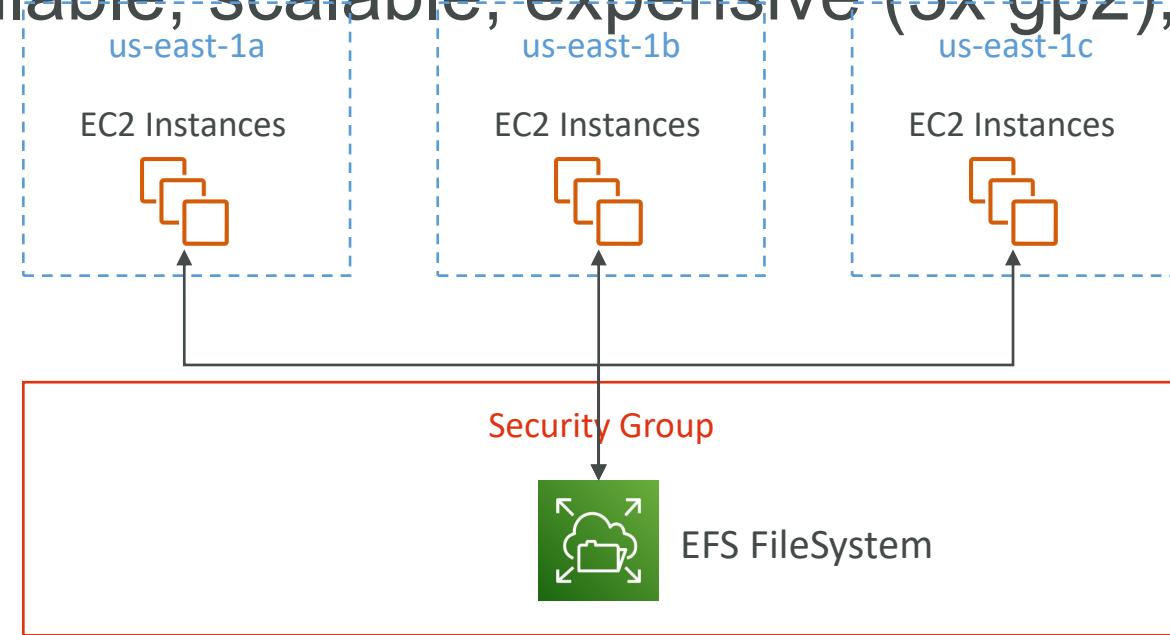
# Amazon EBS Elastic Volumes

- You don't have to detach a volume or restart your instance to change it!
  - Just go to actions / modify volume from the console
- Increase volume size
  - You can only increase, not decrease
- Change volume type
  - Gp2 -> Gp3
  - Specify desired IOPS or throughput performance (or it will guess)
- Adjust performance
  - Increase or decrease



# Amazon EFS – Elastic File System

- Managed NFS (network file system) that can be mounted on many EC2
- EFS works with EC2 instances in multi-AZ
- Highly available, scalable, expensive (3x gp2), pay per use



# Amazon EFS – Elastic File System

- Use cases: content management, web serving, data sharing, Wordpress
- Uses NFSv4.1 protocol
- Uses security group to control access to EFS
- **Compatible with Linux based AMI (not Windows)**
- Encryption at rest using KMS
- POSIX file system (~Linux) that has a standard file API
- File system scales automatically, pay-per-use, no capacity planning!

# EFS – Performance & Storage Classes

- **EFS Scale**

- 1000s of concurrent NFS clients, 10 GB+ /s throughput
- Grow to Petabyte-scale network file system, automatically

- **Performance Mode (set at EFS creation time)**

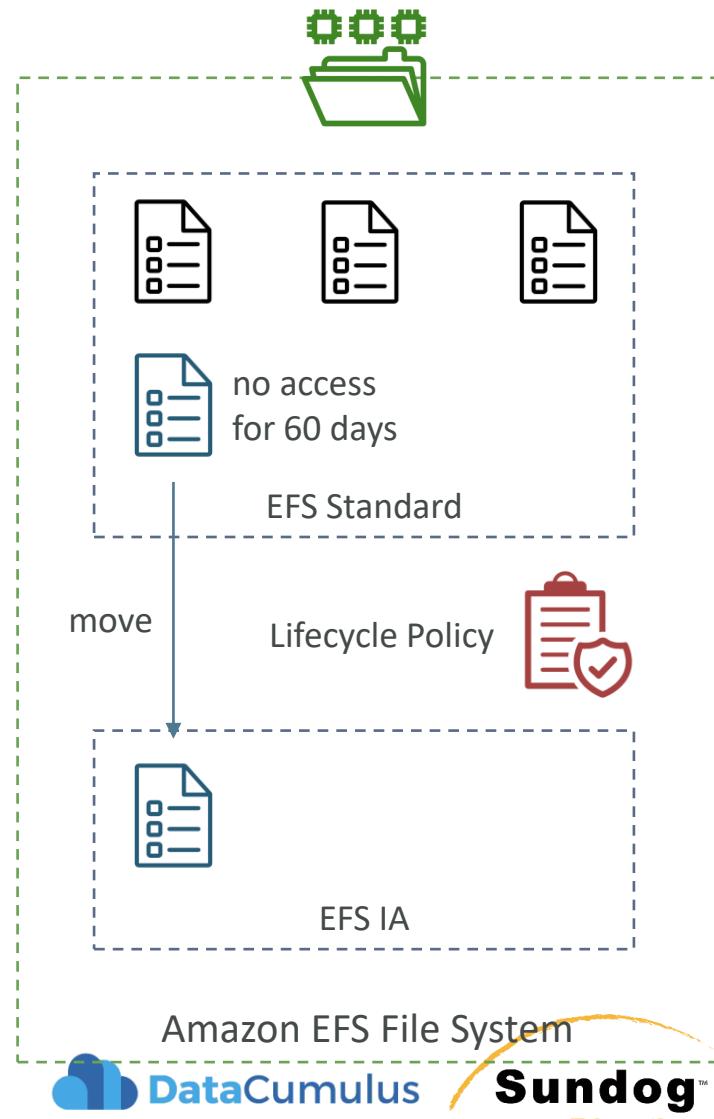
- **General Purpose (default)** – latency-sensitive use cases (web server, CMS, etc...)
- **Max I/O** – higher latency, throughput, highly parallel (big data, media processing)

- **Throughput Mode**

- **Bursting** – 1 TB = 50MiB/s + burst of up to 100MiB/s
- **Provisioned** – set your throughput regardless of storage size, ex: 1 GiB/s for 1 TB storage
- **Elastic** – automatically scales throughput up or down based on your workloads
  - Up to 3GiB/s for reads and 1GiB/s for writes
  - Used for unpredictable workloads

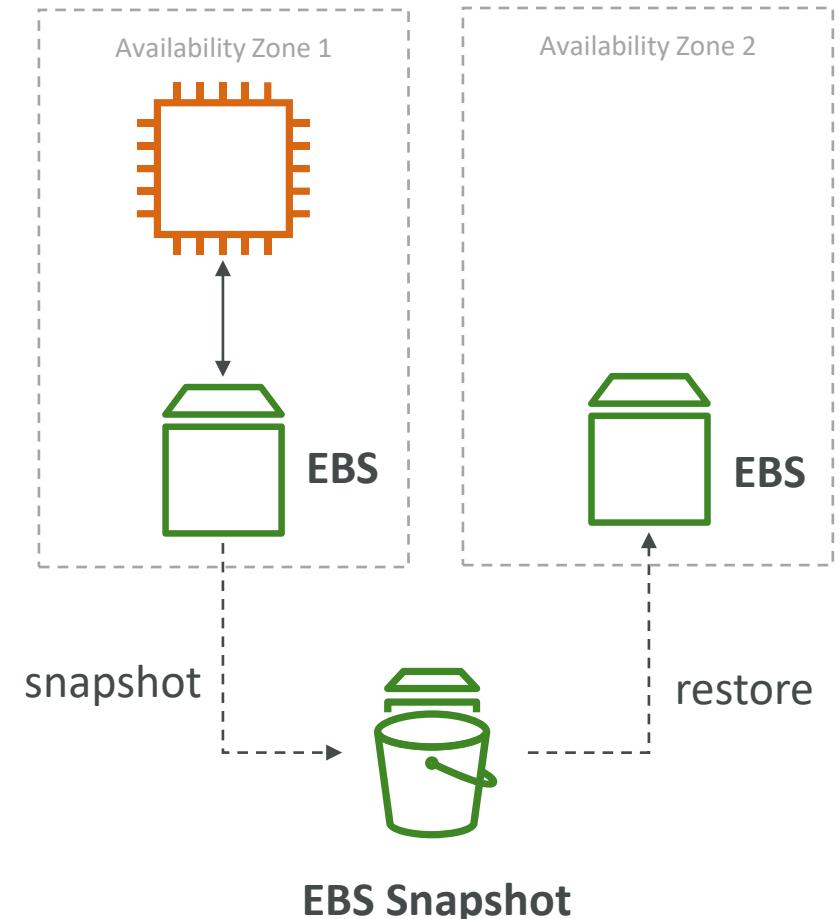
# EFS – Storage Classes

- **Storage Tiers (lifecycle management feature – move file after N days)**
  - Standard: for frequently accessed files
  - Infrequent access (EFS-IA): cost to retrieve files, lower price to store. Enable EFS-IA with a Lifecycle Policy
- **Availability and durability**
  - Standard: Multi-AZ, great for prod
  - One Zone: One AZ, great for dev, backup enabled by default, compatible with IA (EFS One Zone-IA)
- Over 90% in cost savings



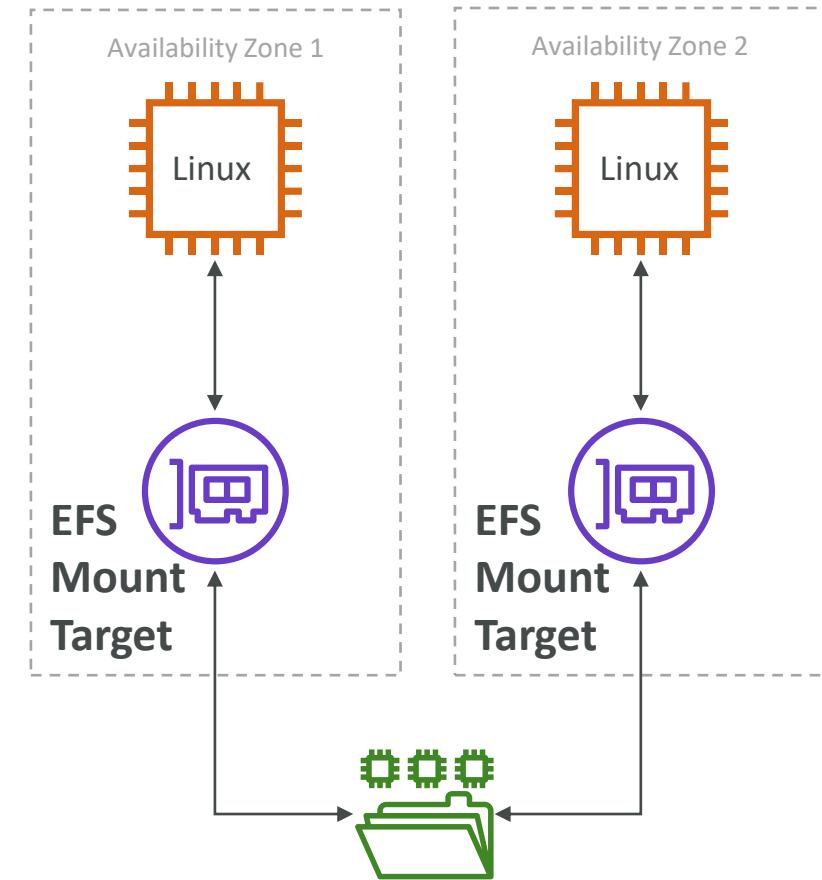
# EBS vs EFS – Elastic Block Storage

- EBS volumes...
  - one instance (except multi-attach io1/io2)
  - are locked at the Availability Zone (AZ) level
  - gp2: IO increases if the disk size increases
  - gp3 & io1: can increase IO independently
- To migrate an EBS volume across AZ
  - Take a snapshot
  - Restore the snapshot to another AZ
  - EBS backups use IO and you shouldn't run them while your application is handling a lot of traffic
- Root EBS Volumes of instances get terminated by default if the EC2 instance gets terminated. (you can disable that)



# EBS vs EFS – Elastic File System

- Mounting 100s of instances across AZ
- EFS share website files (WordPress)
- Only for Linux Instances (POSIX)
- EFS has a higher price point than EBS
- Can leverage EFS-IA for cost savings
- Remember: EFS vs EBS vs Instance Store



# AWS Backup



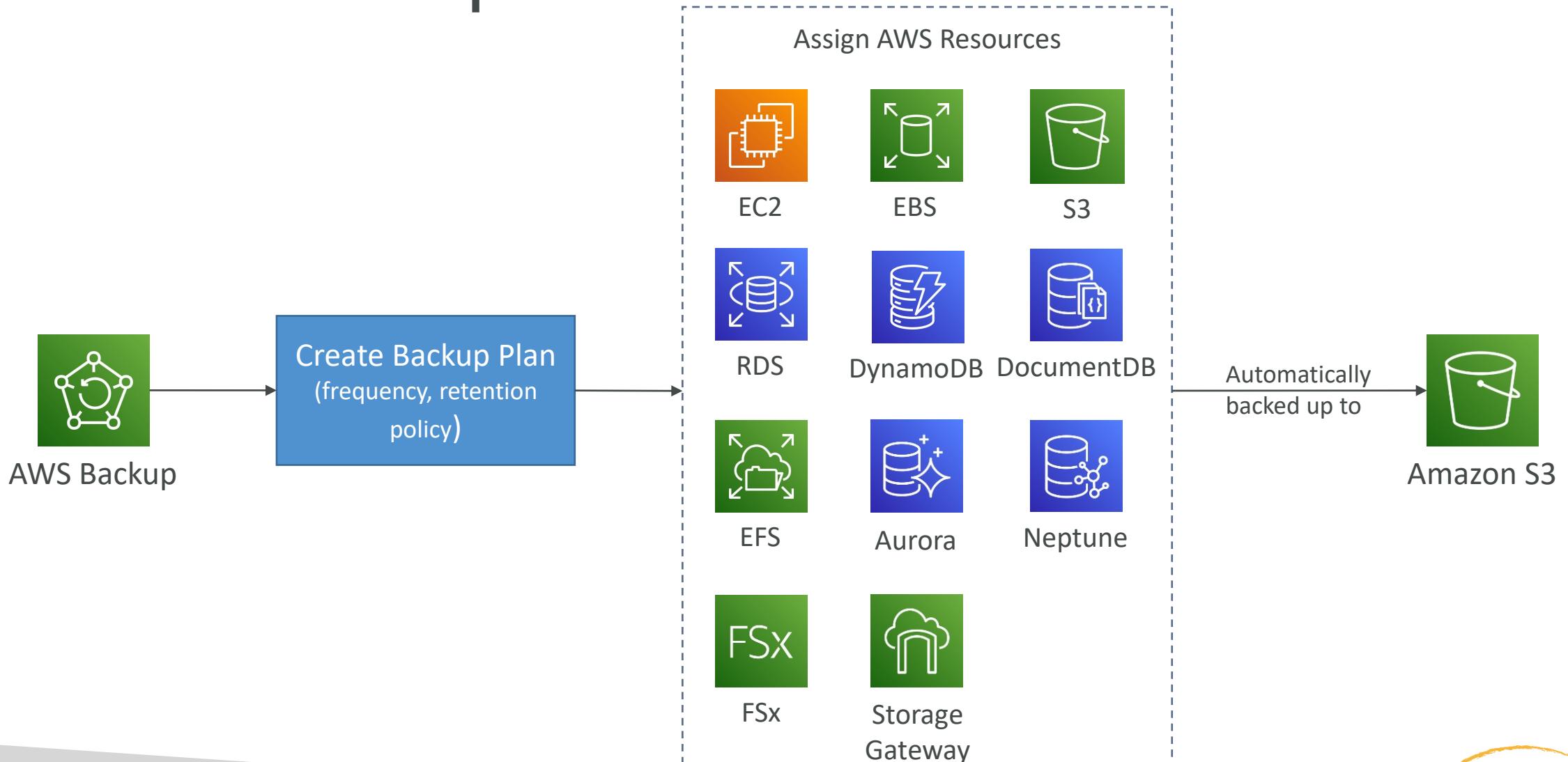
- Fully managed service
- Centrally manage and automate backups across AWS services
- No need to create custom scripts and manual processes
- Supported services:
  - Amazon EC2 / Amazon EBS
  - Amazon S3
  - Amazon RDS (all DBs engines) / Amazon Aurora / Amazon DynamoDB
  - Amazon DocumentDB / Amazon Neptune
  - Amazon EFS / Amazon FSx (Lustre & Windows File Server)
  - AWS Storage Gateway (Volume Gateway)
- Supports cross-region backups
- Supports cross-account backups

# AWS Backup



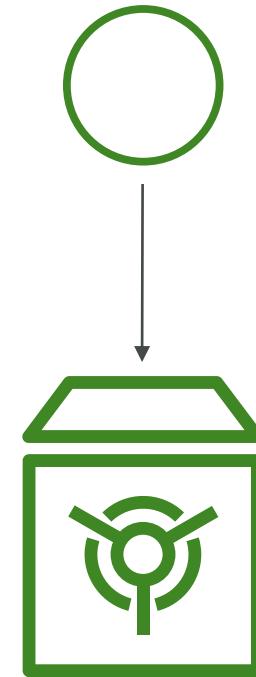
- Supports PITR for supported services
- On-Demand and Scheduled backups
- Tag-based backup policies
- You create backup policies known as **Backup Plans**
  - Backup frequency (every 12 hours, daily, weekly, monthly, cron expression)
  - Backup window
  - Transition to Cold Storage (Never, Days, Weeks, Months, Years)
  - Retention Period (Always, Days, Weeks, Months, Years)

# AWS Backup



# AWS Backup Vault Lock

- Enforce a WORM (Write Once Read Many) state for all the backups that you store in your AWS Backup Vault
- Additional layer of defense to protect your backups against:
  - Inadvertent or malicious delete operations
  - Updates that shorten or alter retention periods
- Even the root user cannot delete backups when enabled



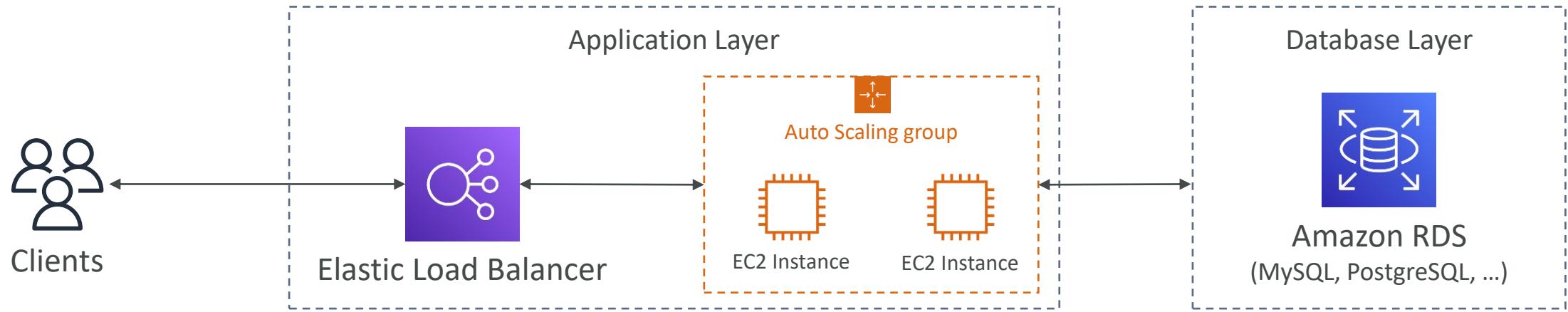
# Database

Managing and querying structured and semi-structured data

# Amazon DynamoDB

NoSQL Serverless Database

# 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

# 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

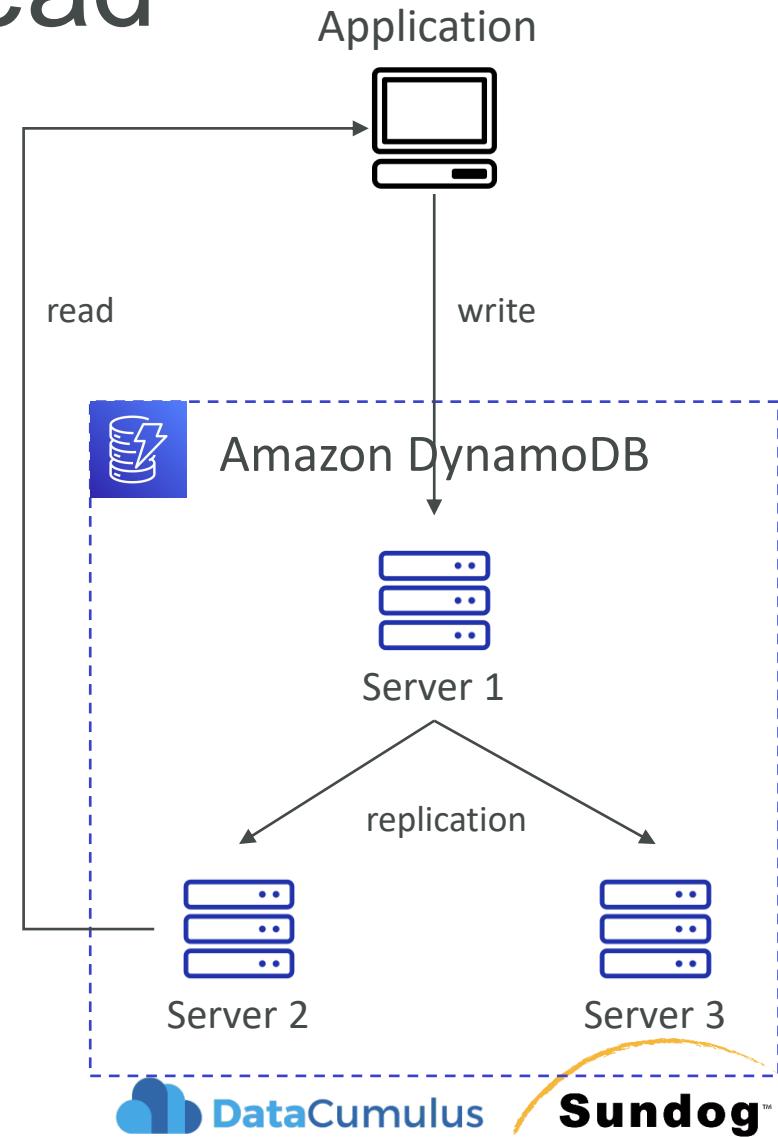
- Table must have provisioned read and write capacity units
- **Read Capacity Units (RCU)** – throughput for reads
- **Write Capacity Units (WCU)** – throughput for writes
- Option to setup **auto-scaling** of throughput to meet demand
- Throughput can be exceeded temporarily using “**Burst Capacity**”
- 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

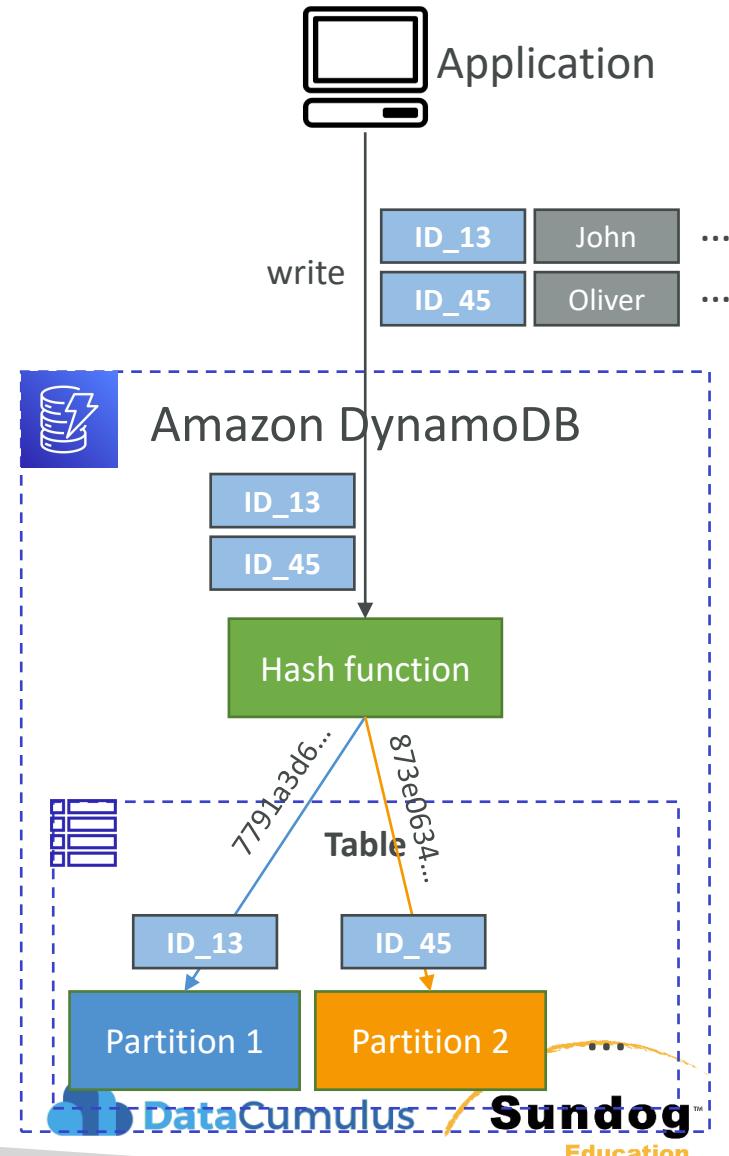


# DynamoDB – Read Capacity Units (RCU)

- 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

# 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**)
  - UnprocessedItems for failed write operations (exponential backoff or add WCU)
- **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
  - UnprocessedKeys for failed read operations (exponential backoff or add RCU)

# DynamoDB – PartiQL

- SQL-compatible query language for DynamoDB
- Allows you to select, insert, update, and delete data in DynamoDB using SQL
- Run queries across multiple DynamoDB tables
- Run PartiQL queries from:
  - AWS Management Console
  - NoSQL Workbench for DynamoDB
  - DynamoDB APIs
  - AWS CLI
  - AWS SDK

```
SELECT OrderID, Total  
FROM Orders  
WHERE OrderID IN [1, 2, 3]  
ORDER BY OrderID DESC
```

# 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 RCU & WCU 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”)

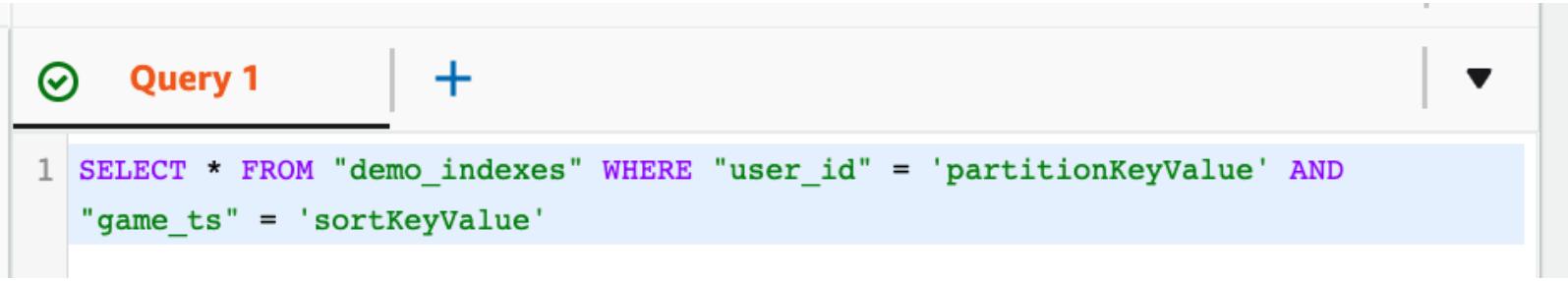


# DynamoDB – Indexes and Throttling

- 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 with a tab labeled "Query 1". The code area contains a single line of SQL-like syntax:

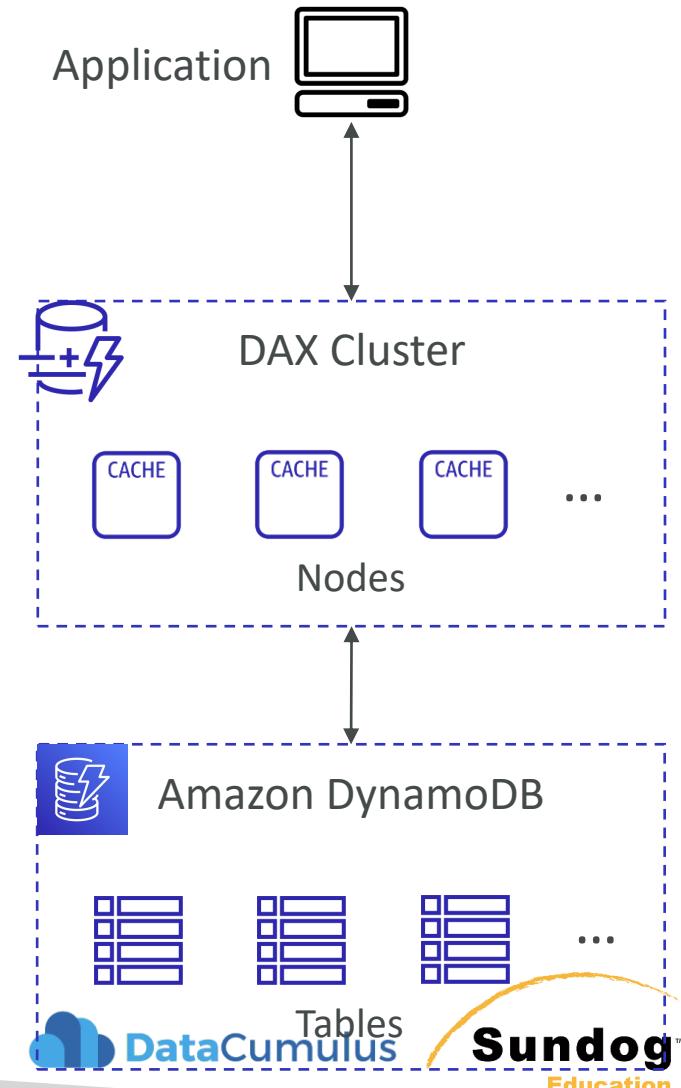
```
1 SELECT * FROM "demo_indexes" WHERE "user_id" = 'partitionKeyValue' AND  
    "game_ts" = 'sortKeyValue'
```

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

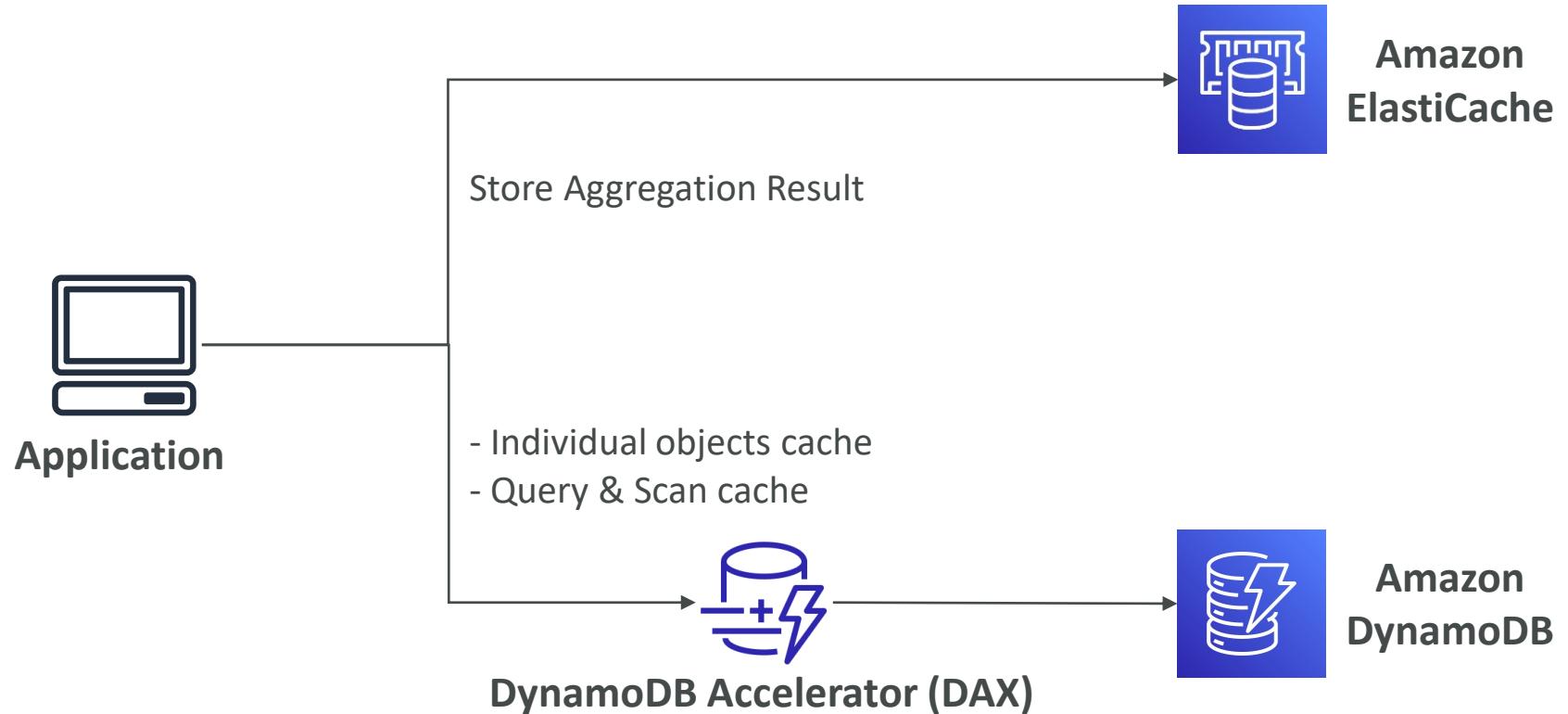
# DynamoDB Accelerator (DAX)



- 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

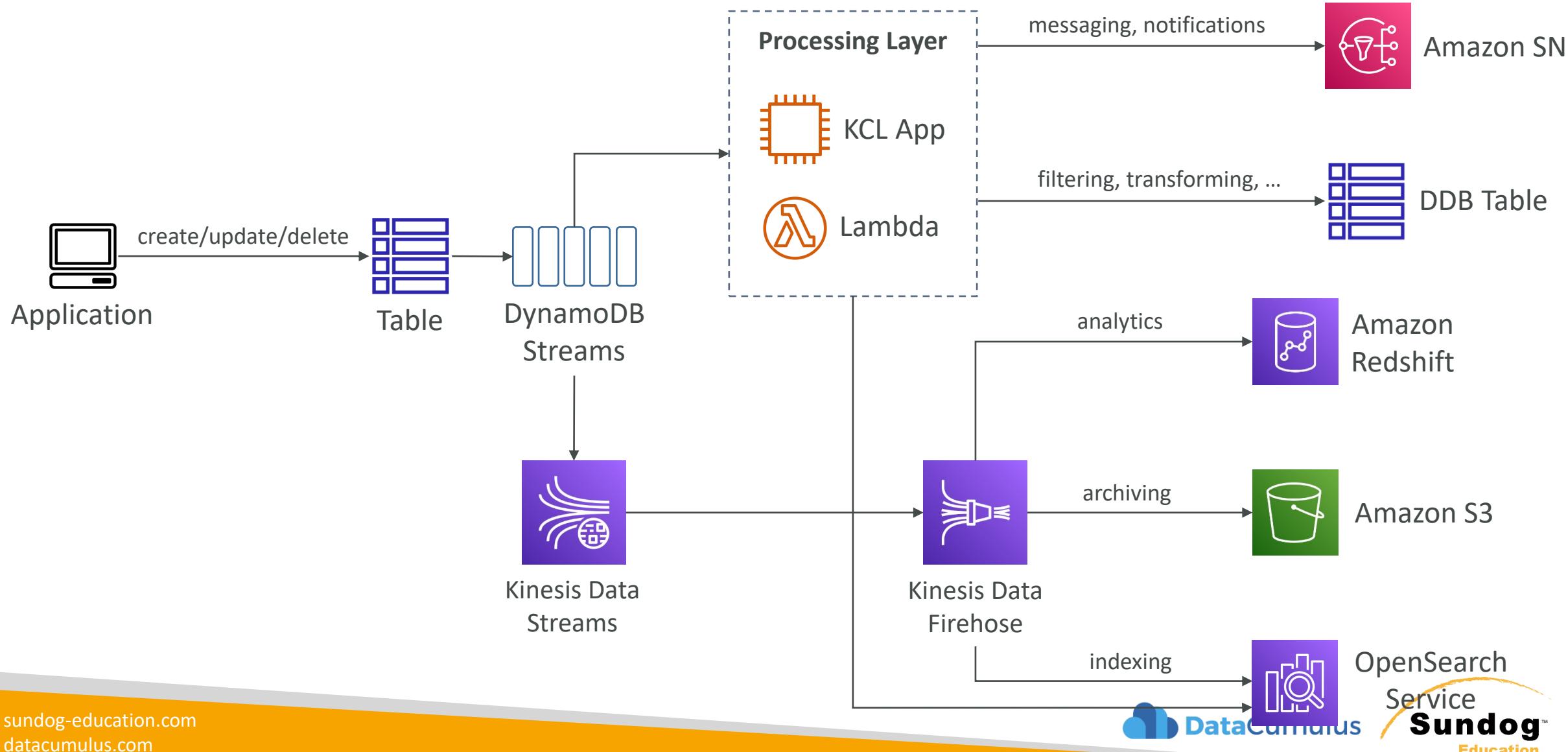


# DynamoDB Streams



- 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 OpenSearch Service
  - Implement cross-region replication

# DynamoDB Streams

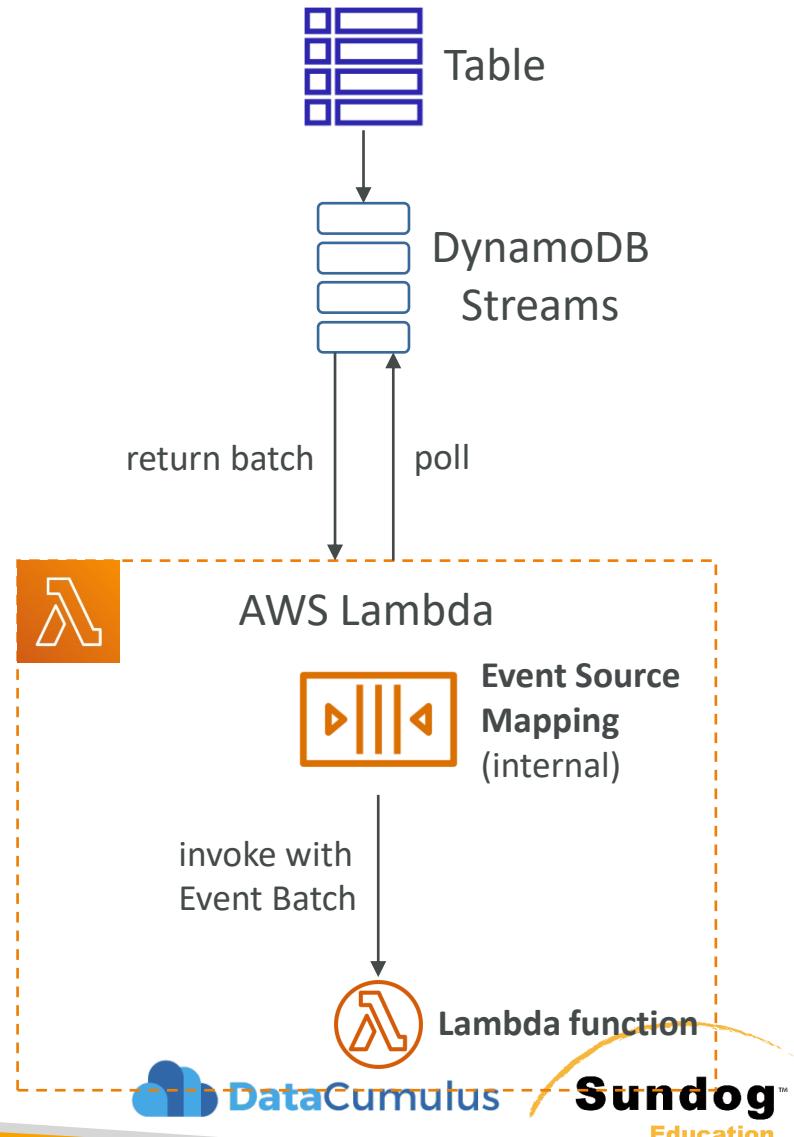


# DynamoDB Streams

- 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

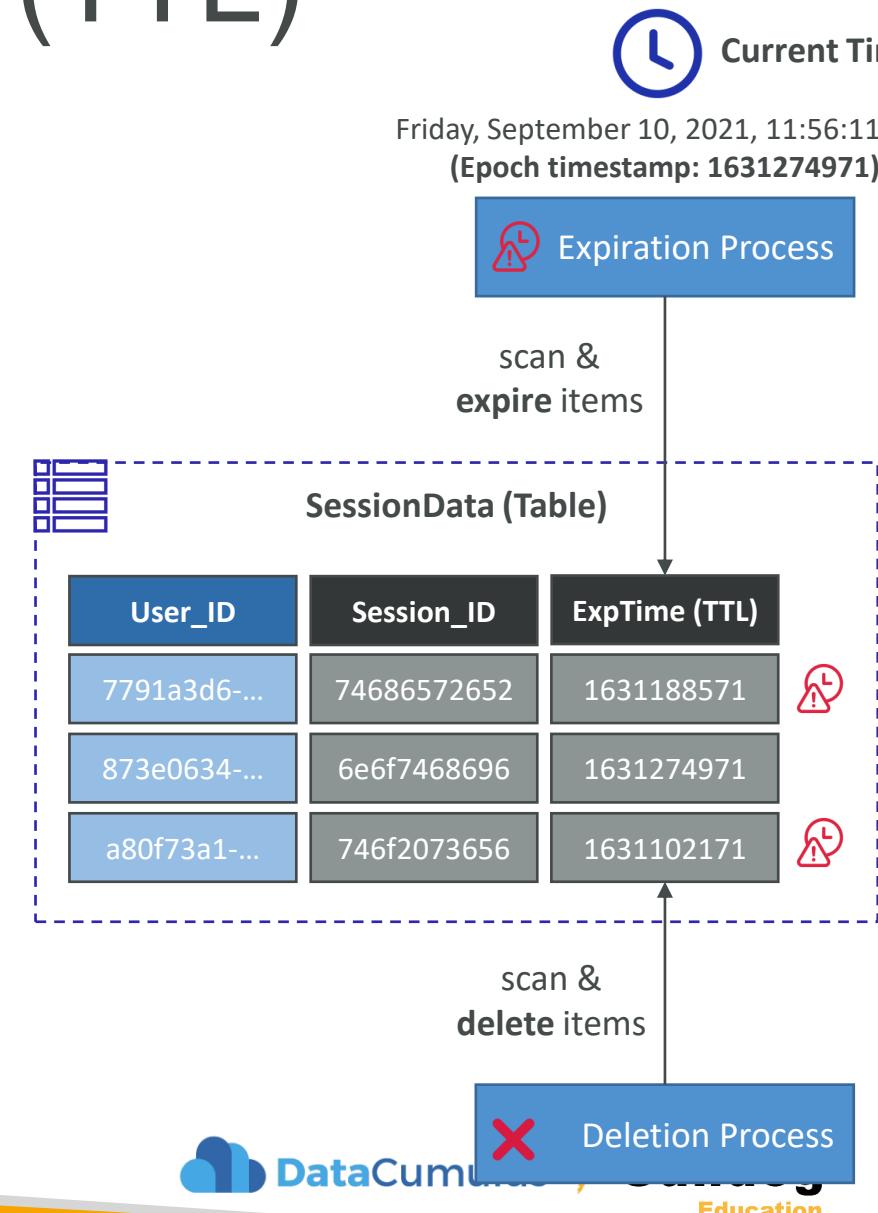
# DynamoDB Streams & AWS Lambda

- 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**

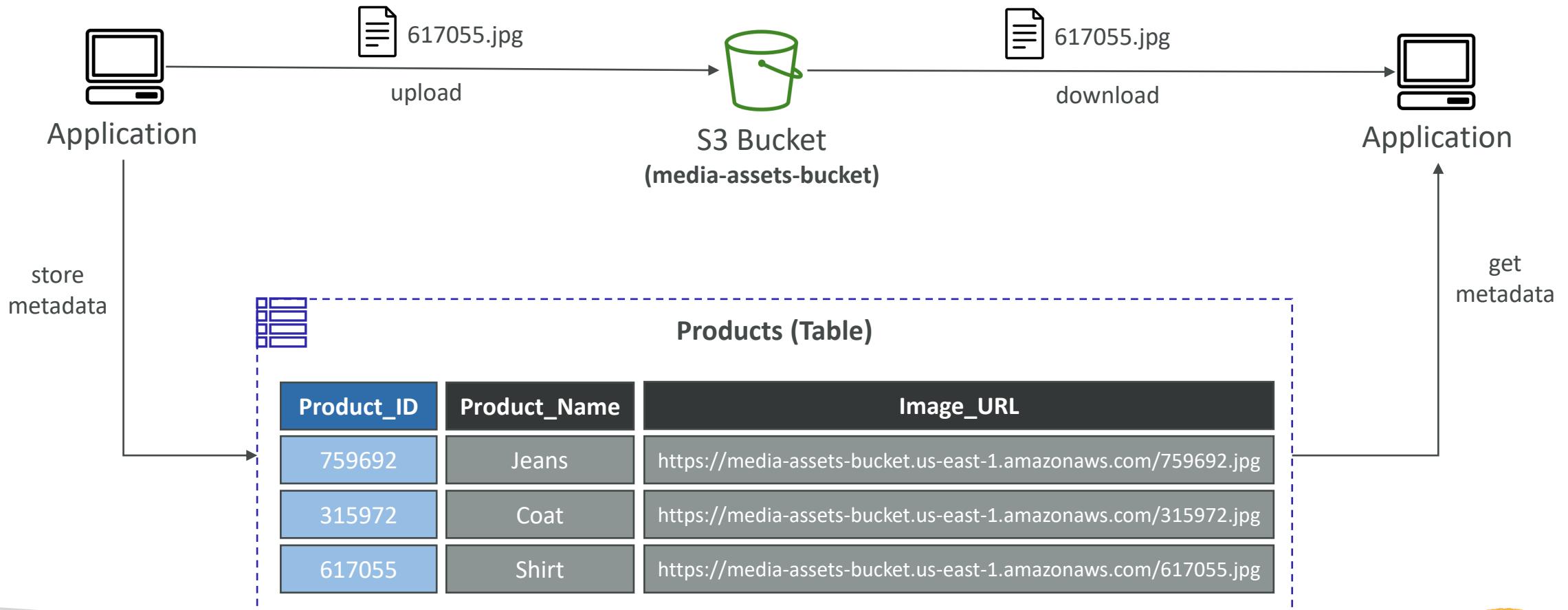


# DynamoDB – Time To Live (TTL)

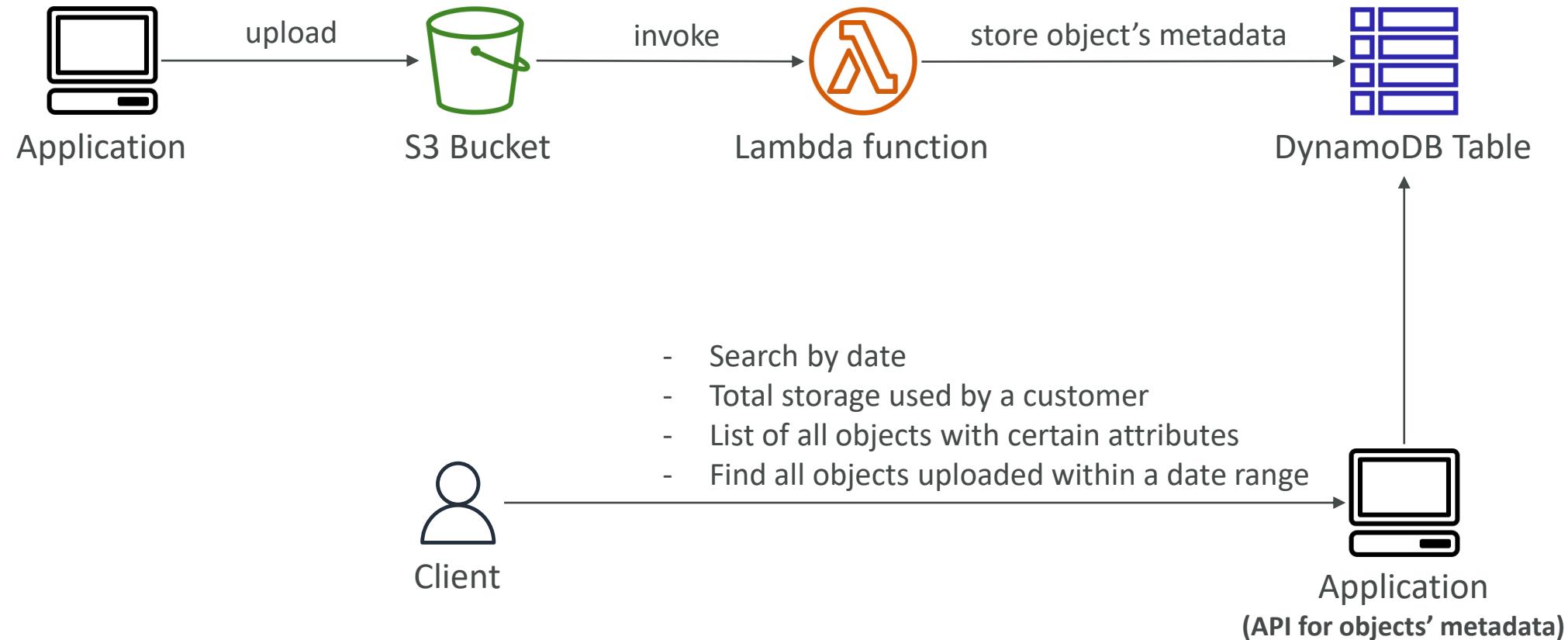
- Automatically delete items after an expiry timestamp
- Doesn't consume any WCUs (i.e., no extra cost)
- The TTL attribute must be a “Number” data type with “Unix Epoch timestamp” value
- Expired items deleted within 48 hours of expiration
- Expired items, that haven't been deleted, appears in reads/queries/scans (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)



# DynamoDB – Large Objects Pattern



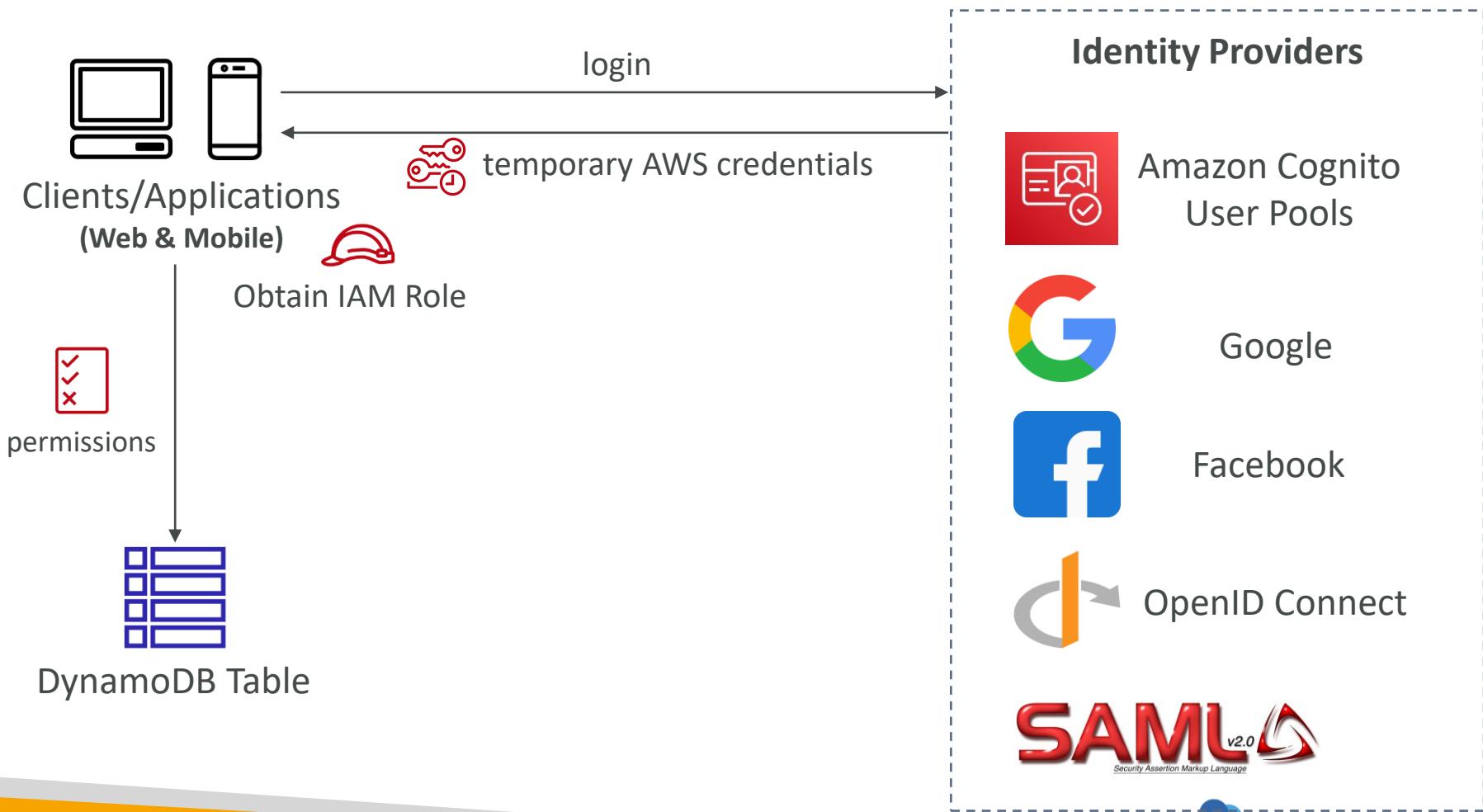
# DynamoDB – Indexing S3 Objects Metadata



# DynamoDB – Security & Other Features

- **Security**
  - VPC Endpoints available to access DynamoDB without using the Internet
  - Access fully controlled by IAM
  - Encryption at rest using AWS KMS and in-transit using SSL/TLS
- **Backup and Restore feature available**
  - Point-in-time Recovery (PITR) like RDS
  - No performance impact
- **Global Tables**
  - Multi-region, multi-active, fully replicated, high performance
- **DynamoDB Local**
  - Develop and test apps locally without accessing the DynamoDB web service (without Internet)
  - AWS Database Migration Service (AWS DMS) can be used to migrate to DynamoDB (from MongoDB, Oracle, MySQL, S3, ...)

# DynamoDB – Users Interact with DynamoDB Directly



# DynamoDB – Fine-Grained Access Control

- Using **Web Identity Federation or Cognito Identity Pools**, each user gets AWS credentials
- You can assign an IAM Role to these users with a **Condition** to limit their API access to DynamoDB
- **LeadingKeys** – limit row-level access for users on the Primary Key
- **Attributes** – limit specific attributes the user can see

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:GetItem", "dynamodb:BatchGetItem", "dynamodb:Query",  
                "dynamodb:PutItem", "dynamodb:UpdateItem", "dynamodb:DeleteItem",  
                "dynamodb:BatchWriteItem"  
            ],  
            "Resource": "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable",  
            "Condition": {  
                "ForAllValues:StringEquals": {  
                    "dynamodb:LeadingKeys": ["${cognito-identity.amazonaws.com:sub}"]  
                }  
            }  
        }  
    ]  
}
```

More at: <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/specifying-conditions.html>

# 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 128TB per database volume
- Up to 15 read replicas
- Continuous backup to S3
- Replication across regions and 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



# Using the LOCK command

- Relational databases implicitly “lock” tables to prevent two things writing to it at the same time, or reading while a write is in process.
- Tables or rows can also be explicitly locked to ensure data integrity and concurrency control.
- Types of locks:
  - Shared Locks: Allow reads, prevent writes. Can be held by multiple transactions. (FOR SHARE)
  - Exclusive Locks: Prevent all reads and writes to a resource. Only one transaction can hold an exclusive lock. (FOR UPDATE)



# Examples (MySQL)

- Lock an entire table:
  - `LOCK TABLES employees WRITE;` -- Locks the entire 'employees' table for write operations
  - Use `UNLOCK TABLES;` to release the lock.
  - Note: Redshift also has a `LOCK` command for the same purpose.
- Shared lock (allow reads, prevent other writes during this transaction.)
  - `SELECT * FROM employees WHERE department = 'Finance' FOR SHARE;`
- Exclusive lock (prevent all reads and writes during this transaction)
  - `SELECT * FROM employees WHERE employee_id = 123 FOR UPDATE;`
- Make sure the transactions any locks are in complete, or you could end up with a “deadlock”

# Amazon RDS best practices



# Amazon RDS operational guidelines

- Use CloudWatch to monitor memory, CPU, storage, replica lag
- Perform automatic backups during daily low in write IOPS
- Insufficient I/O will make recovery after failure slow.
  - Migrate to DB instance with more I/O
  - Move to General Purpose or Provisioned IOPS storage
- Set TTL on DNS for your DB instances to 30 seconds or less from your apps
- Test failover before you need it
- Provision enough RAM to include your entire working set
  - If your **ReadIOPS** metric is small and stable, you're good
- Rate limits in Amazon API Gateway can be used to protect your database



# Query optimization in RDS

- Use indexes to accelerate SELECT statements
  - Use EXPLAIN plans to identify the indexes you need
- Avoid full table scans
- Use ANALYZE TABLE periodically
- Simplify WHERE clauses
- Engine-specific optimizations

```
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
  
SELECT  
    OrderH.invoiceNo, OrderH.invoiceDate, OrderH.customerCode,  
    OrderD.itemCode, I.itemName, OrderD.netPrice  
FROM  
    OrderHeader AS OrderH  
    INNER JOIN Customer AS Cust ON OrderH.customerCode = Cust.customerCode  
    INNER JOIN OrderDetail AS OrderD ON OrderD.orderId = OrderH.orderId  
    INNER JOIN Item AS I ON OrderD.itemCode = I.itemCode  
WHERE  
    OrderD.netPrice > 1000  
ORDER BY  
    OrderH.customerCode, OrderD.netPrice
```

# DB-specific tweaks

- MySQL, MariaDB
  - Keep tables well under 16TB, ideally under 100GB
  - Have enough RAM to hold indexes of actively used tables
  - Try to have less than 10,000 tables
  - Use InnoDB for storage engine
- PostgreSQL
  - When loading data, disable DB backups and multi-AZ. Tweak various DB parameters such as maintenance\_work\_mem, max\_wal\_size, checkpoint\_timeout. Disable synchronous\_commit, autovacuum, and ensure tables are logged.
  - Use autovacuum

# DB-specific tweaks

- SQL Server
  - Use RDS DB Events to monitor failovers
  - Do not enable simple recover mode, offline mode, or read-only mode (this breaks Multi-AZ)
  - Deploy into all AZ's
- Oracle is its own beast

# DocumentDB



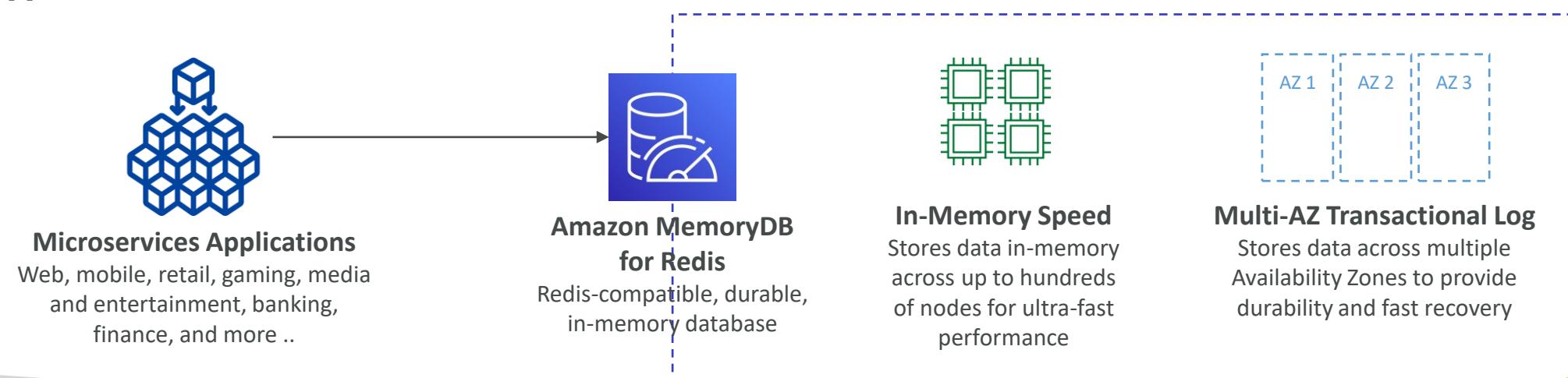
- Aurora is an “AWS-implementation” of PostgreSQL / MySQL ...
- **DocumentDB is the same for MongoDB (which is a NoSQL database)**
- MongoDB is used to store, query, and index JSON data
- Similar “deployment concepts” as Aurora
- Fully Managed, highly available with replication across 3 AZ
- DocumentDB storage automatically grows in increments of 10GB
- Automatically scales to workloads with millions of requests per seconds

# Amazon MemoryDB for Redis



- Redis-compatible, **durable, in-memory database service**
- **Ultra-fast performance with over 160 millions requests/second**
- Durable in-memory data storage with Multi-AZ transactional log
- Scale seamlessly from 10s GBs to 100s TBs of storage
- Use cases: web and mobile apps, online gaming, media streaming,

...



# Amazon Keyspaces (for Apache Cassandra)

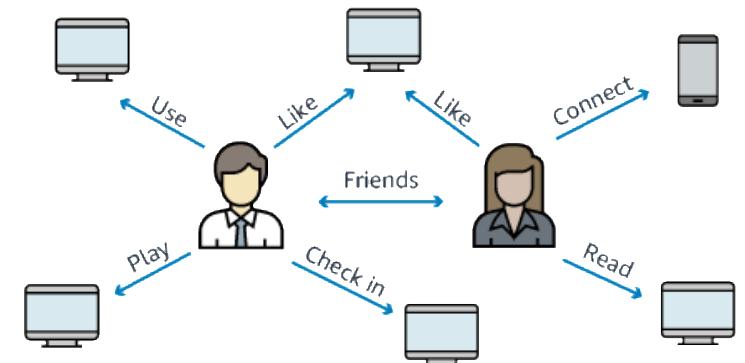


- *Apache Cassandra is an open-source NoSQL distributed database*
- A managed Apache Cassandra-compatible database service
- Serverless, Scalable, highly available, fully managed by AWS
- Automatically scale tables up/down based on the application's traffic
- Tables are replicated 3 times across multiple AZ
- Using the Cassandra Query Language (CQL)
- Single-digit millisecond latency at any scale, 1000s of requests per second
- Capacity: On-demand mode or provisioned mode with auto-scaling
- Encryption, backup, Point-In-Time Recovery (PITR) up to 35 days
- Use cases: store IoT devices info, time-series data, ...

# Amazon Neptune



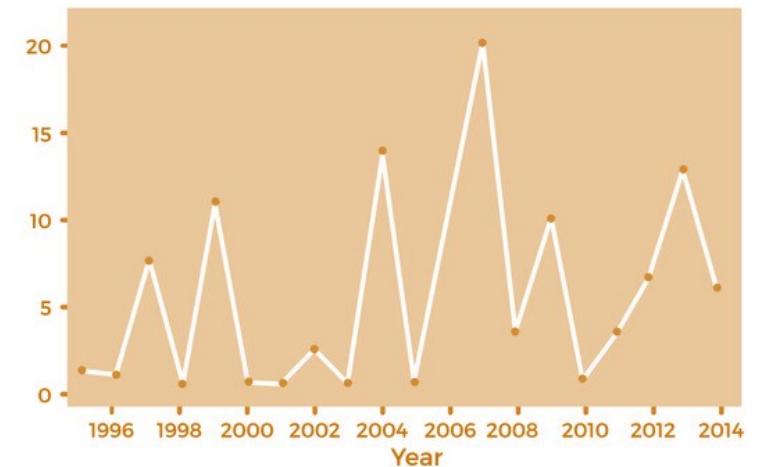
- Fully managed **graph** database
- A popular **graph dataset** would be a **social network**
  - Users have friends
  - Posts have comments
  - Comments have likes from users
  - Users share and like posts...
- Highly available across 3 AZ, with up to 15 read replicas
- Build and run applications working with highly connected datasets – optimized for these complex and hard queries
- Can store up to billions of relations and query the graph with milliseconds latency
- Highly available with replications across multiple AZs
- Great for knowledge graphs (Wikipedia), fraud detection, recommendation engines, social networking



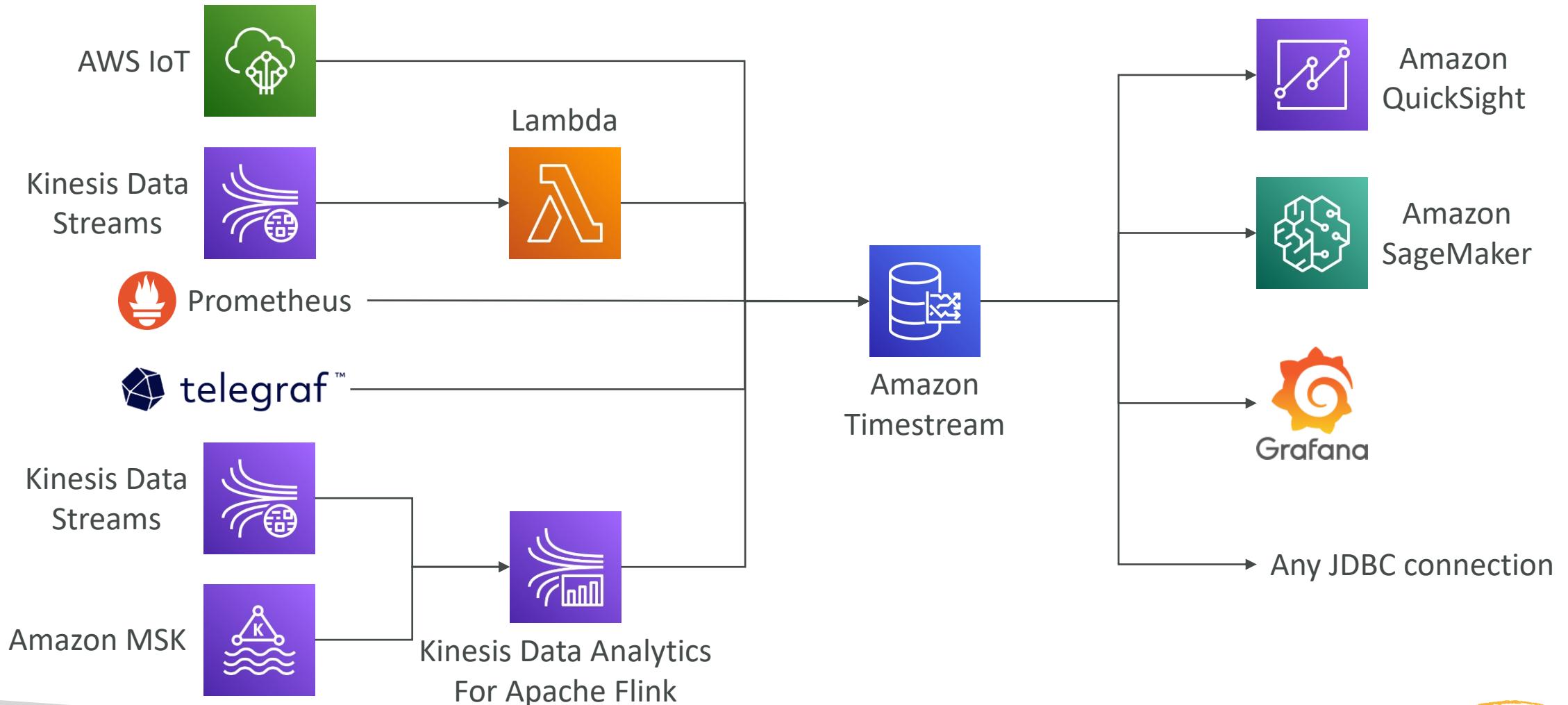
# Amazon Timestream



- Fully managed, fast, scalable, serverless **time series database**
- Automatically scales up/down to adjust capacity
- Store and analyze trillions of events per day
- 1000s times faster & 1/10<sup>th</sup> the cost of relational databases
- Scheduled queries, multi-measure records, SQL compatibility
- Data storage tiering: recent data kept in memory and historical data kept in a cost-optimized storage
- Built-in time series analytics functions (helps you identify patterns in your data in near real-time)
- Encryption in transit and at rest
- Use cases: IoT apps, operational applications, real-time analytics, ...



# Amazon Timestream – Architecture



# 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

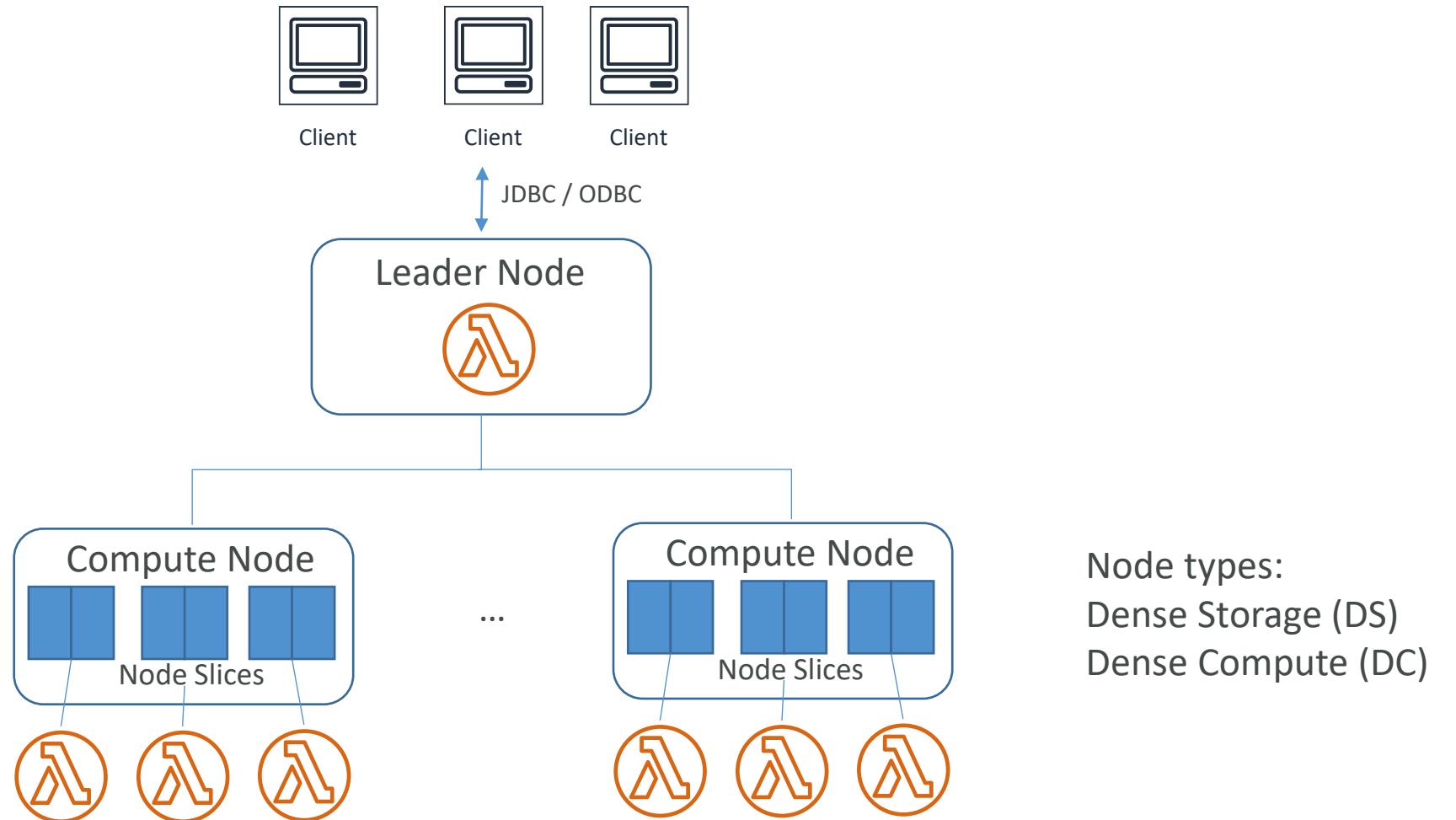


**Amazon  
Redshift**

# 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



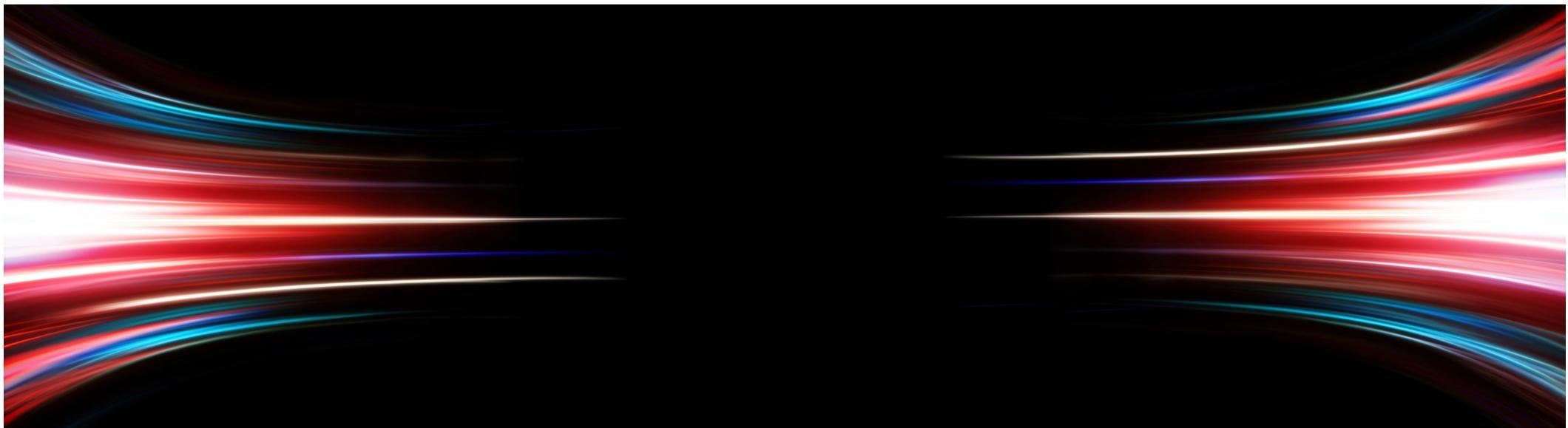
AWS Glue



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 now available



# Scaling Redshift

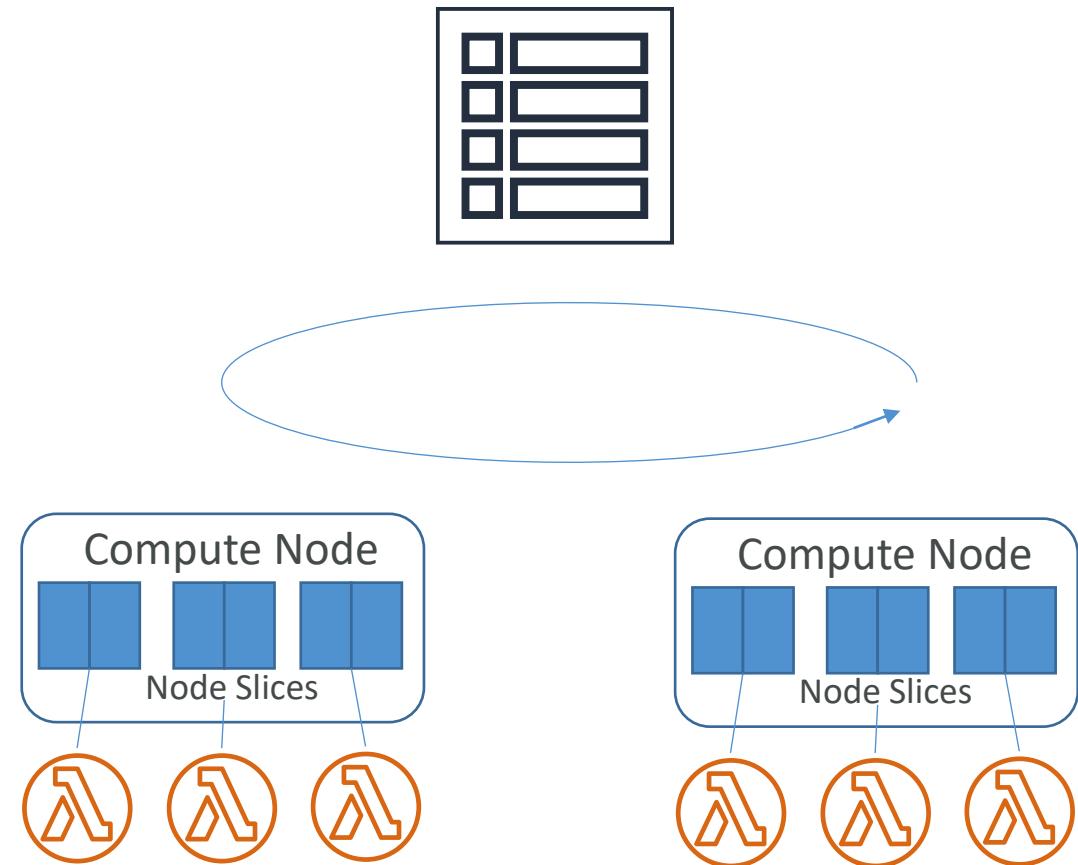
- Vertical and horizontal scaling on demand
- During scaling:
  - A new cluster is created while your old one remains available for reads
  - CNAME is flipped to new cluster (a few minutes of downtime)
  - Data moved in parallel to new compute nodes



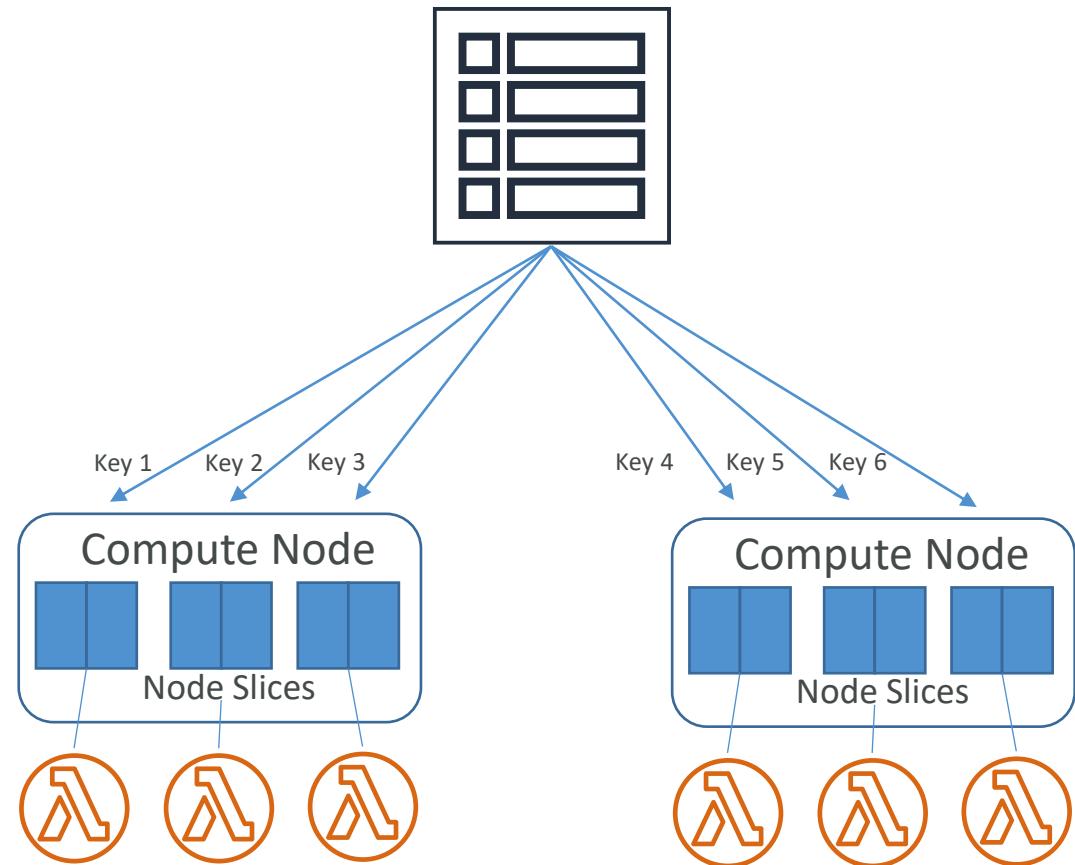
# Redshift Distribution Styles

- 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

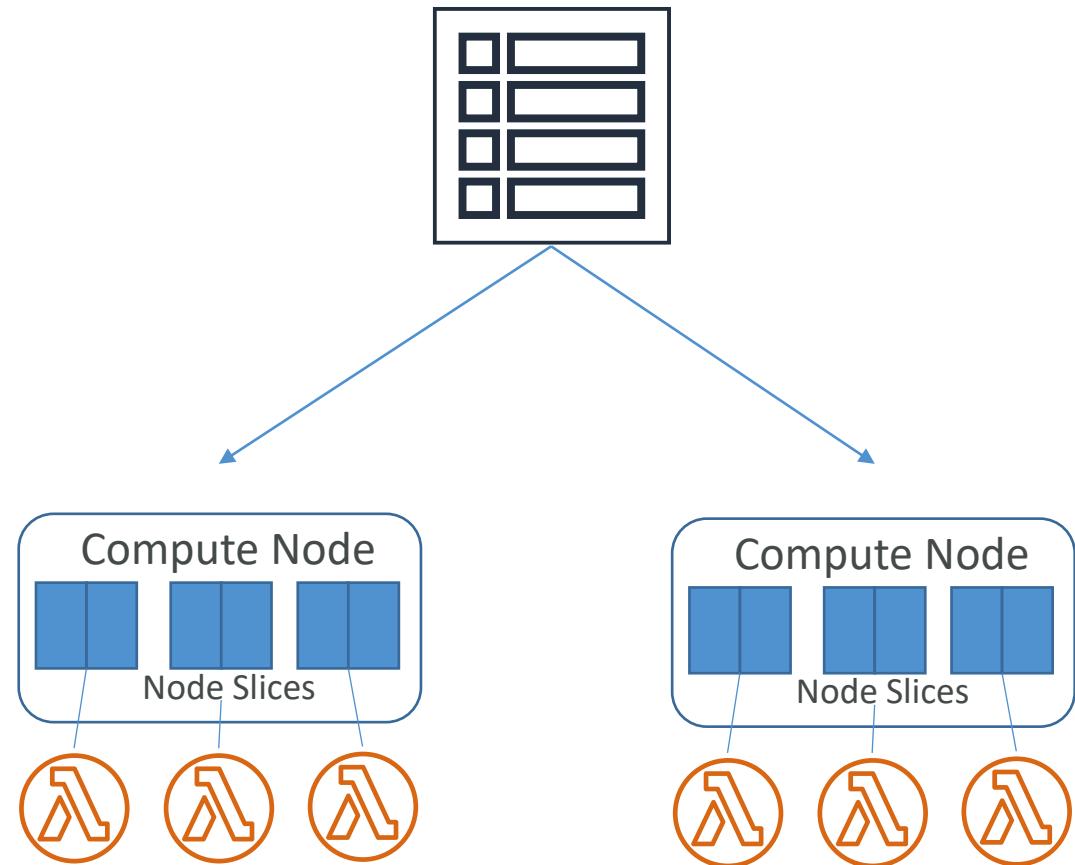
# EVEN distribution



# KEY distribution

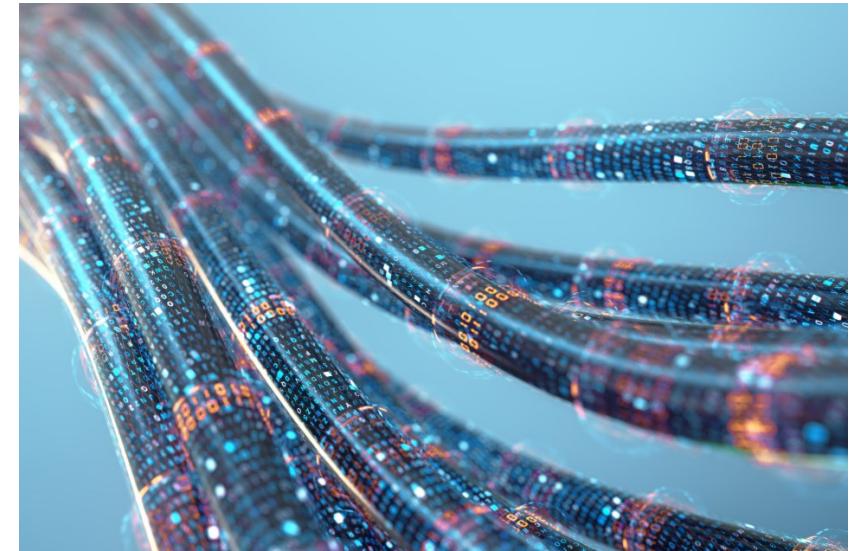


# ALL distribution



# 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
- Auto-copy from Amazon S3
- Amazon Aurora zero-ETL integration
  - Auto replication from Aurora -> Redshift
- Redshift Streaming Ingestion
  - From Kinesis Data Streams or MSK



# 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



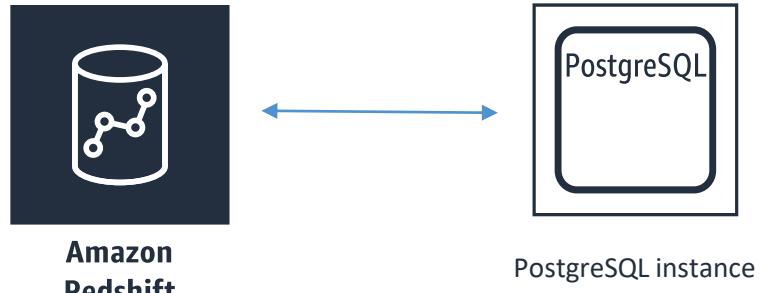
# Redshift copy grants for cross-region snapshot copies

- 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

```
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

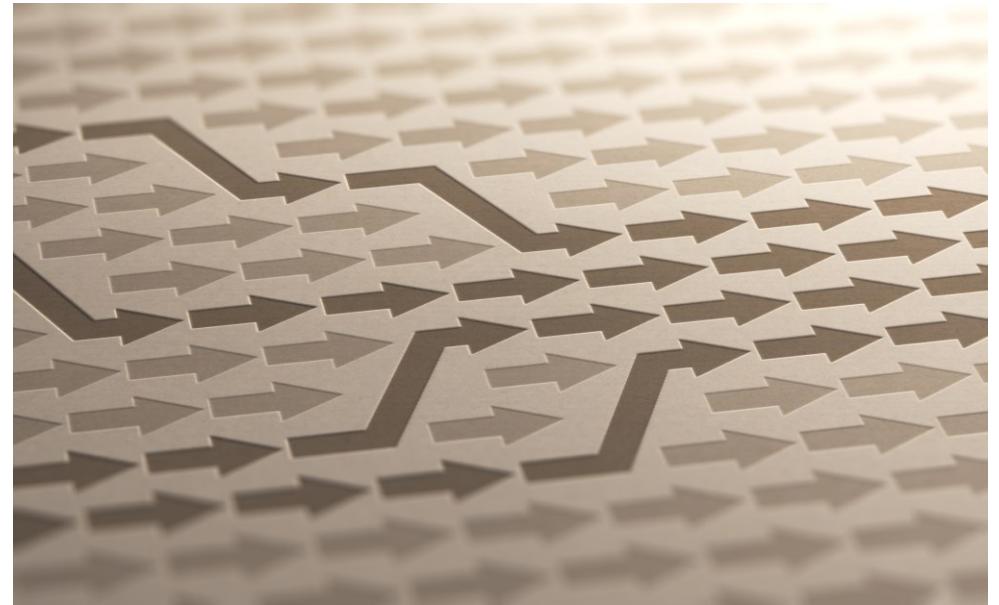
# Concurrency Scaling

- 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

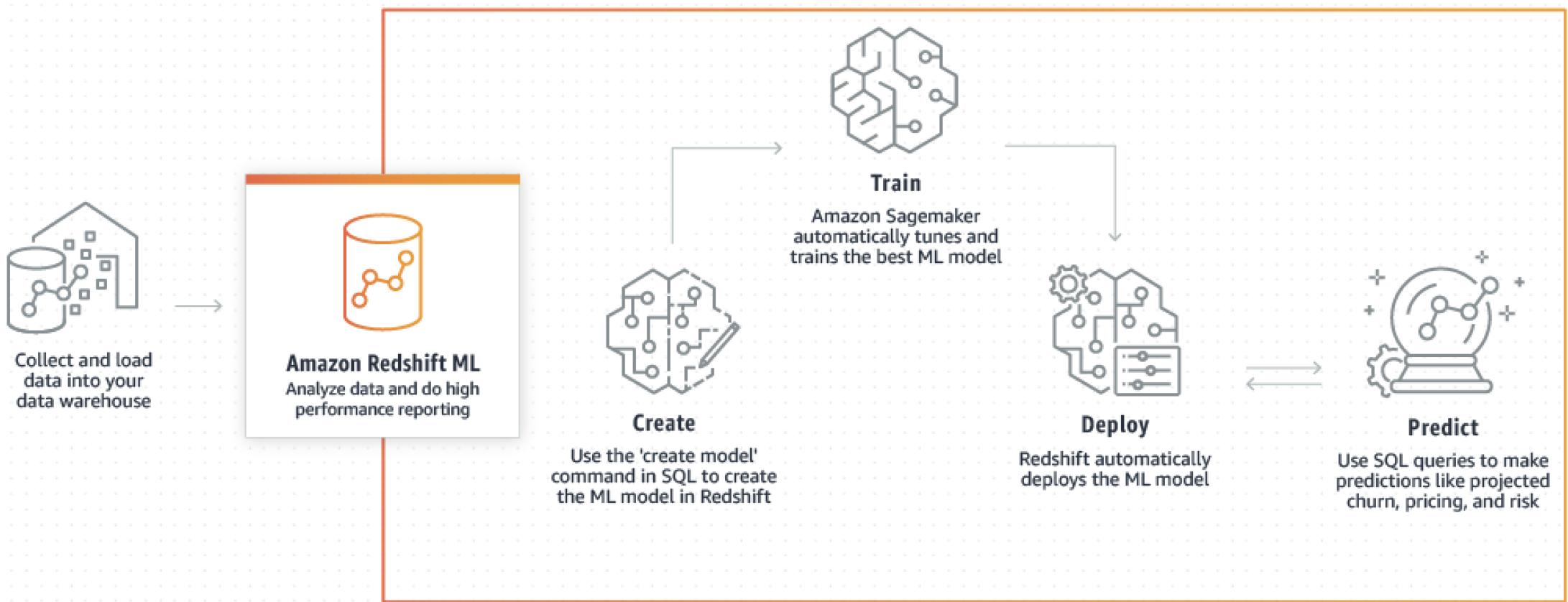
- Recovers space from deleted rows and restores sort order
- VACUUM FULL
- VACUUM DELETE ONLY
  - Skips the sort
- VACUUM SORT ONLY
  - Does not reclaim space!
- VACUUM REINDEX
  - Re-analyzes distribution of sort key columns
  - Then does a full VACUUM



# Newer Redshift features

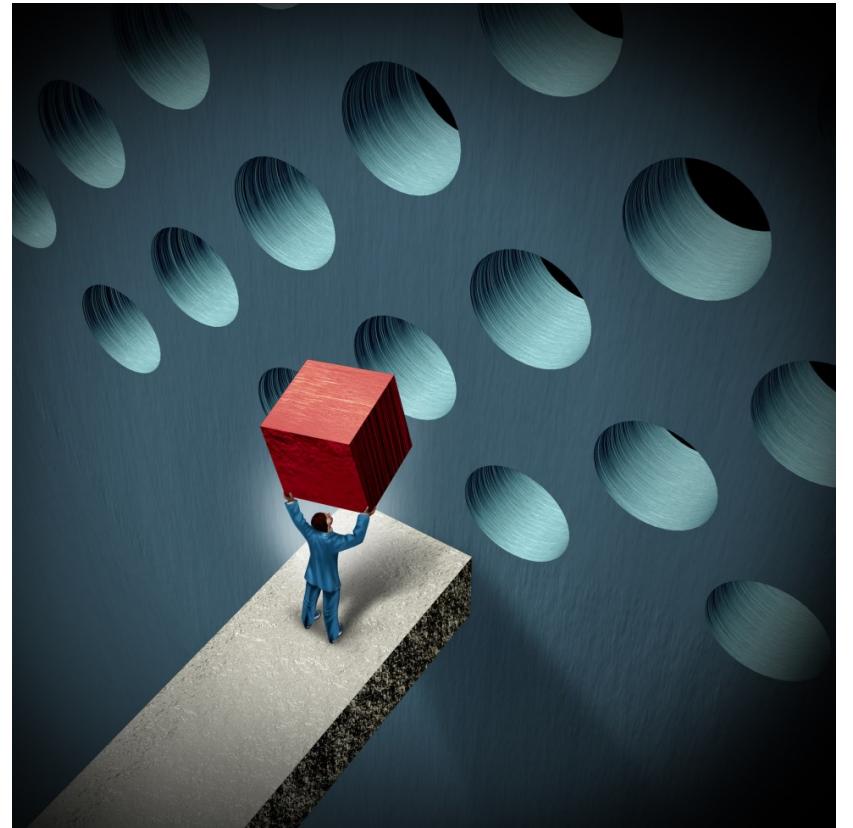
- RA3 nodes with managed storage
  - Enable independent scaling of compute and storage
  - SSD-based
- Redshift data lake export
  - Unload Redshift query to S3 in Apache Parquet format
  - Parquet is 2x faster to unload and consumes up to 6X less storage
  - Compatible with Redshift Spectrum, Athena, 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

# Amazon Redshift ML



# 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

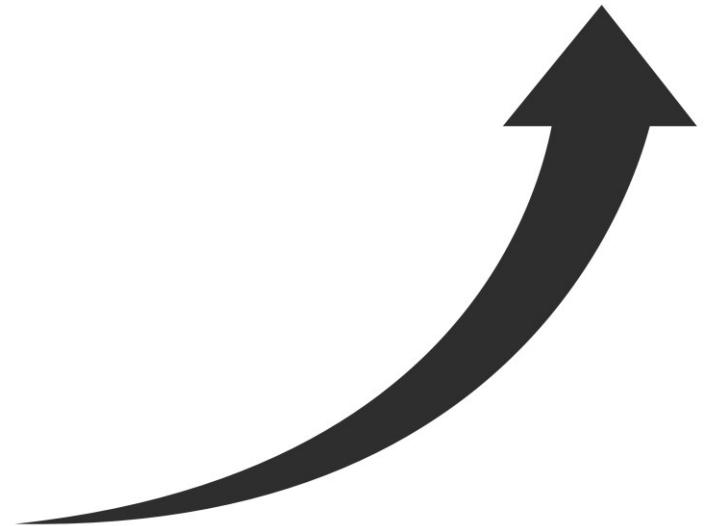
# Redshift Serverless: Getting Started

- 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

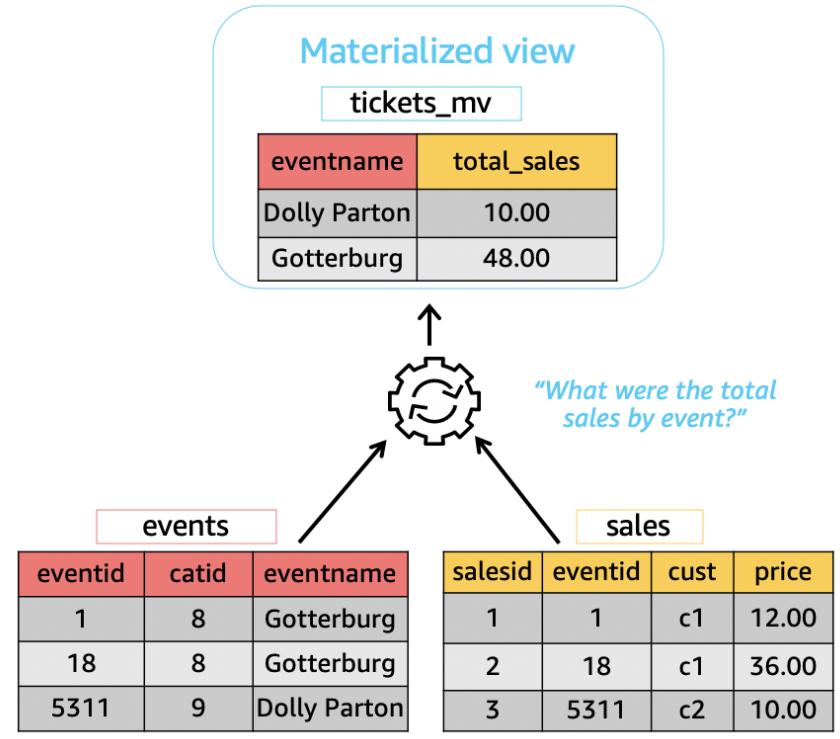
# Redshift Serverless: Monitoring

- 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



# Redshift Materialized Views

- Contain precomputed results based on SQL queries over one or more base tables.
  - This differs from a “normal” view in that it actually stores the results of the query
- Provide a way to speed up complex queries in a data warehouse environment, especially on large tables.
- You can query materialized views just like any other tables or views.
- Queries return results faster since they use precomputed results without accessing base tables.
- They're particularly beneficial for predictable and recurring queries, e.g., populating dashboards like Amazon QuickSight.



AWS

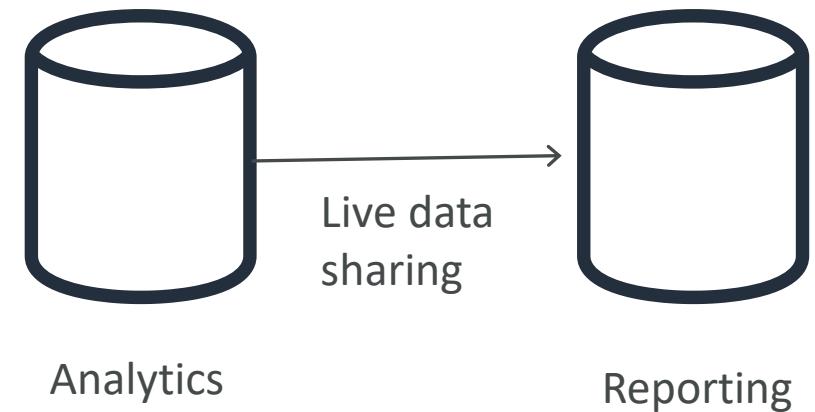
# Using Materialized Views

- CREATE MATERIALIZED VIEW...
- Keeping them refreshed
  - REFRESH MATERIALIZED VIEW...
  - Set AUTO REFRESH option on creation
- Query them just like any other table or view
- Materialized views can be built from other materialized views
  - Useful for re-using expensive joins

```
CREATE MATERIALIZED VIEW tickets_mv AS
  select catgroup,
         sum(qtysold) as sold
    from category c, event e, sales s
   where c.catid = e.catid
     and e.eventid = s.eventid
  group by catgroup;
```

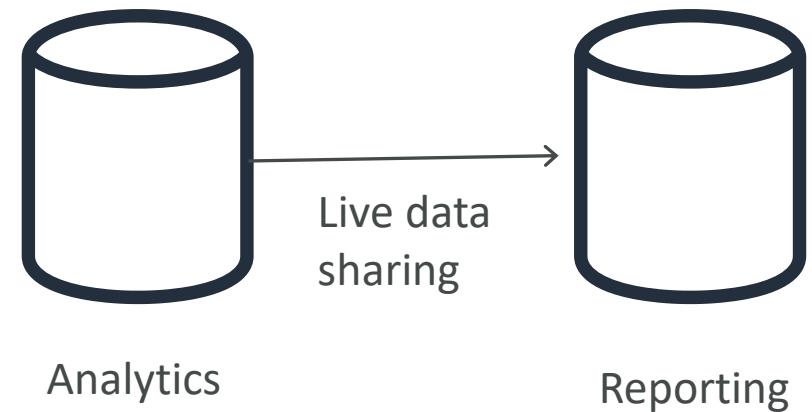
# Redshift Data Sharing

- Securely share live data across Redshift clusters for read purposes
- Why?
  - Workload isolation
  - Cross-group collaboration
  - Sharing data between dev/test/prod
  - Licensing data access in AWS Data Exchange
- Can share DB's, schemas, tables, views, and/or UDFs.
  - Fine-grained access control



# Redshift Data Sharing

- Producer / consumer architecture
  - Producer controls security
  - Isolation to ensure producer performance unaffected by consumers
  - Data is live and transactionally consistent
- Both must be encrypted, must use RA3 nodes
- Cross-region data sharing involves transfer charges
- Types of data shares
  - Standard
  - AWS Data Exchange
  - AWS Lake Formation - managed

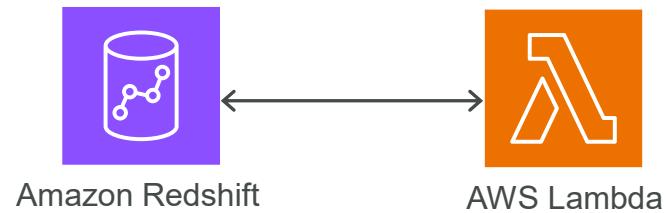


# Redshift Lambda UDF

- Use custom functions in AWS Lambda inside SQL queries
  - Using any language you want!
  - Do anything you want!
    - Call other services (AI?)
    - Access external systems
    - Integrate with location service
- Register with CREATE EXTERNAL FUNCTION
- Must GRANT USAGE ON LANGUAGE EXFUNC for permissions

```
SELECT a, b FROM t1 WHERE  
lambda_multiply(a, b) = 64;
```

```
CREATE EXTERNAL FUNCTION exfunc_sum(INT,INT) RETURNS  
INT VOLATILE LAMBDA 'lambda_sum' IAM_ROLE  
'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';
```



# Redshift Lambda UDF

- Use AWSLambdaRole IAM policy to grant permissions to Lambda on your cluster's IAM role
  - Or roll your own policy, to allow lambda:InvokeFunction
- You also need this IAM role in your CREATE EXTERNAL FUNCTION command
  - Possible to invoke functions in other accounts in the same Region, using IAM role chaining
- Redshift communicates with Lambda using JSON

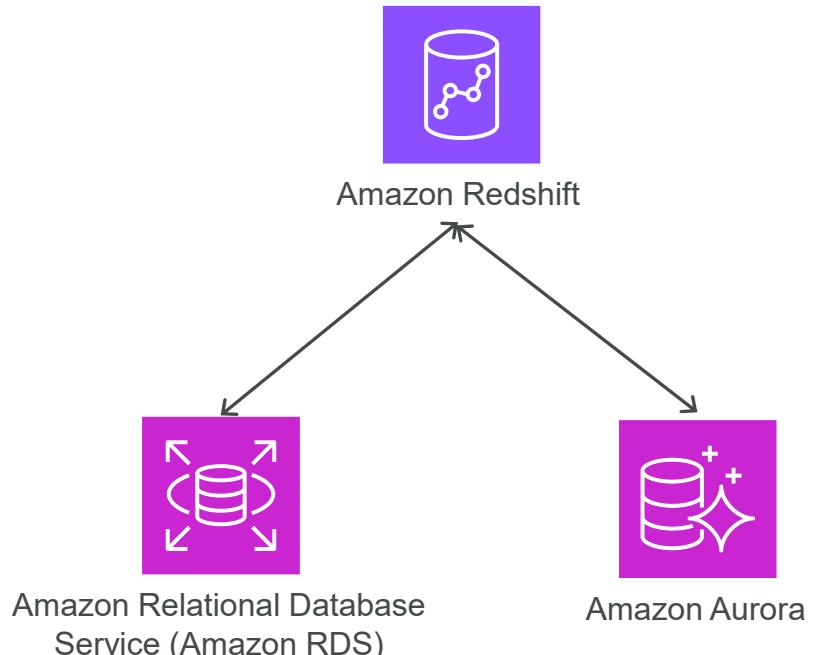
```
{
  "request_id": "23FF1F97-F28A-44AA-AB67-266ED976BF40",
  "cluster": "arn:aws:redshift:xxxx",
  "user": "adminuser",
  "database": "db1",
  "external_function": "public.foo",
  "query_id": 5678234,
  "num_records": 4,
  "arguments": [
    [ 1, 2 ],
    [ 3, null ],
    null,
    [ 4, 6 ]
  ]
}

{
  "success": true, // true indicates the call succeeded
  "error_msg": "my function isn't working", // shall only exist when success != true
  "num_records": 4, // number of records in this payload
  "results": [
    1,
    4,
    null,
    7
  ]
}
```



# Redshift Federated Queries

- Query and analyze across databases, warehouses, and lakes
- Ties Redshift to Amazon RDS or Aurora for PostgreSQL and MySQL
  - Incorporate live data in RDS into your Redshift queries
  - Avoids the need for ETL pipelines
- Offloads computation to remote databases to reduce data movement



# Redshift Federated Queries

- Must establish connectivity between your Redshift cluster and RDS / Aurora
  - Put them in the same VPC subnet
  - Or use VPC peering
- Credentials must be in AWS Secrets Manager
- Include secrets in IAM role for your Redshift cluster
- Connect using CREATE EXTERNAL SCHEMA
  - Can also connect to S3 / Redshift Spectrum this way
- The SVV\_EXTERNAL\_SCHEMAS view contains available external schemas

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AccessSecret",  
      "Effect": "Allow",  
      "Action": [  
        "secretsmanager:GetResourcePolicy",  
        "secretsmanager:GetSecretValue",  
        "secretsmanager:DescribeSecret",  
        "secretsmanager>ListSecretVersionIds"  
      ],  
      "Resource": "arn:aws:secretsmanager:us-west-  
2:123456789012:secret:my-rds-secret-VNenFy"  
    },  
    {  
      "Sid": "VisualEditor1",  
      "Effect": "Allow",  
      "Action": [  
        "secretsmanager:GetRandomPassword",  
        "secretsmanager>ListSecrets"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```

# Redshift Federated Queries

- Read-only access to external data sources
- Costs will be incurred on external DB's
- You can query RDS/Aurora from Redshift, but not the other way around

```
CREATE EXTERNAL SCHEMA apg
FROM POSTGRES
DATABASE 'database-1' SCHEMA 'myschema'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-
SecretsManager-RO'
SECRET_ARN 'arn:aws:secretsmanager:us-west-
2:123456789012:secret:federation/test/dataplane-apg-creds-
YbVKQw';
```

```
SELECT count(*) FROM apg.lineitem;
```

```
count
```

```
-----
```

```
11760
```

# Redshift System Tables and Views

- Contains info about how Redshift is functioning
- Types of system tables / views
  - SYS views: Monitor query & workload usage
  - STV tables: Transient data containing snapshots of current system data
  - SVV views: metadata about DB objects that reference STV tables
  - STL views: Generated from logs persisted to disk
  - SVCS views: Details about queries on main & concurrency scaling clusters
  - SVL views: Details about queries on main clusters
- Many system monitoring views & tables are only for provisioned clusters, not serverless

```
SELECT q.query, qstarttime, qendtime,  
       datediff(seconds, qstarttime, qendtime) as  
execution_time,  
       q.aborted, qr.queue_time, qr.exec_time,  
qr.total_exec_time  
FROM stl_query q  
JOIN svl_qlog qr ON q.query = qr.query  
WHERE qstarttime > (GETDATE() - INTERVAL '1 day')  
ORDER BY qstarttime DESC;
```

*Example: analyze execution time of recent queries*

# Migration and Transfer

Moving data into AWS

# AWS Application Discovery Service

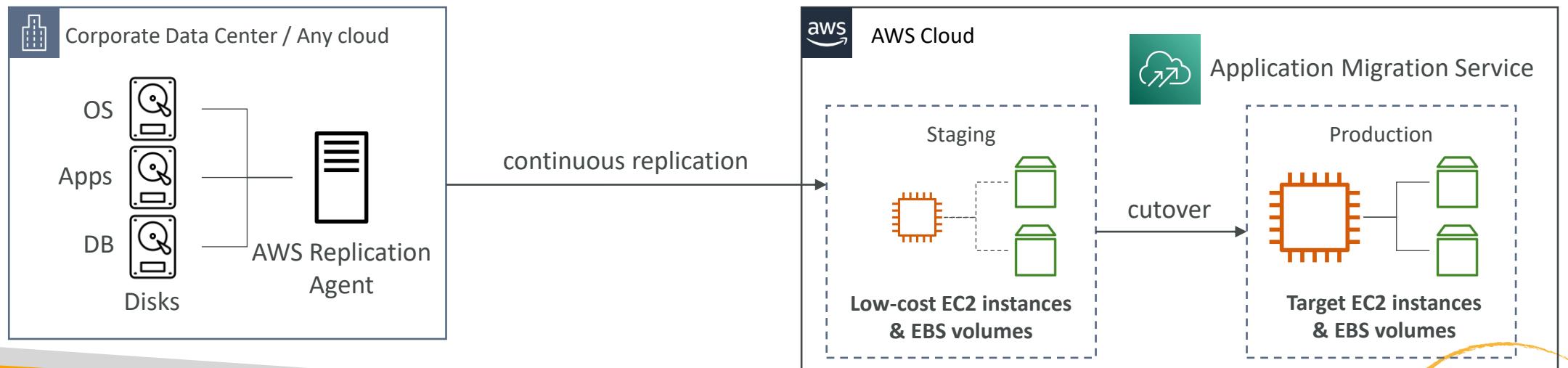


- Plan migration projects by gathering information about on-premises data centers
- Server utilization data and dependency mapping are important for migrations
- **Agentless Discovery (AWS Agentless Discovery Connector)**
  - VM inventory, configuration, and performance history such as CPU, memory, and disk usage
- **Agent-based Discovery (AWS Application Discovery Agent)**
  - System configuration, system performance, running processes, and details of the network connections between systems
- Resulting data can be viewed within AWS Migration Hub

# AWS Application Migration Service (MGN)



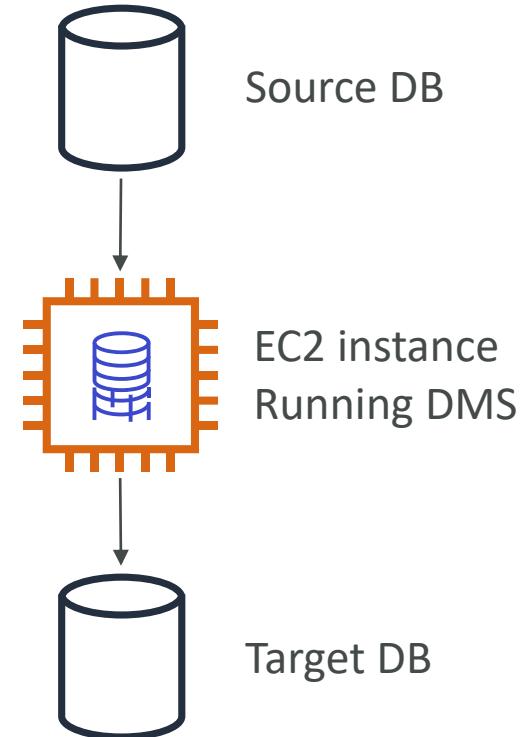
- *The “AWS evolution” of CloudEndure Migration, replacing AWS Server Migration Service (SMS)*
- Lift-and-shift (rehost) solution which simplify **migrating** applications to AWS
- Converts your physical, virtual, and cloud-based servers to run natively on AWS
- Supports wide range of platforms, Operating Systems, and databases
- Minimal downtime, reduced costs



# 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:

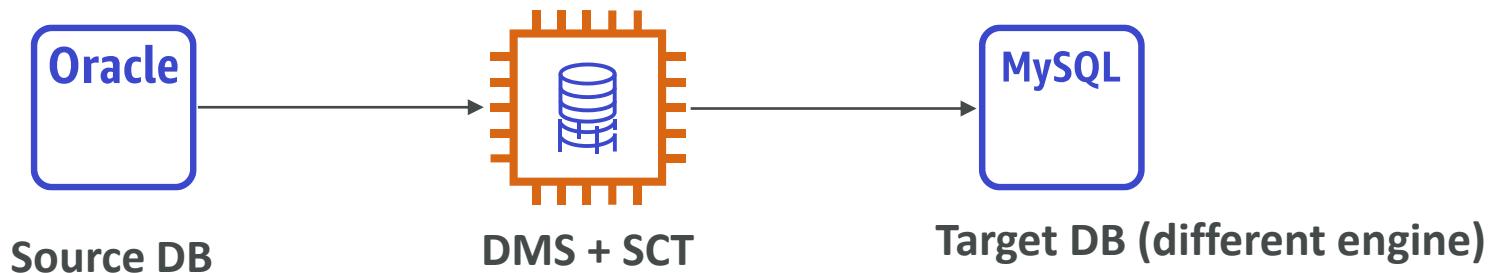
- On-Premises and EC2 instances databases: *Oracle, MS SQL Server, MySQL, MariaDB, PostgreSQL, MongoDB, SAP, DB2*
- Azure: *Azure SQL Database*
- Amazon RDS: all including Aurora
- Amazon S3
- DocumentDB

## TARGETS:

- On-Premises and EC2 instances databases: Oracle, MS SQL Server, MySQL, MariaDB, PostgreSQL, SAP
- Amazon RDS
- Redshift, DynamoDB, S3
- OpenSearch Service
- Kinesis Data Streams
- Apache Kafka
- DocumentDB & Amazon Neptune
- Redis & Babelfish

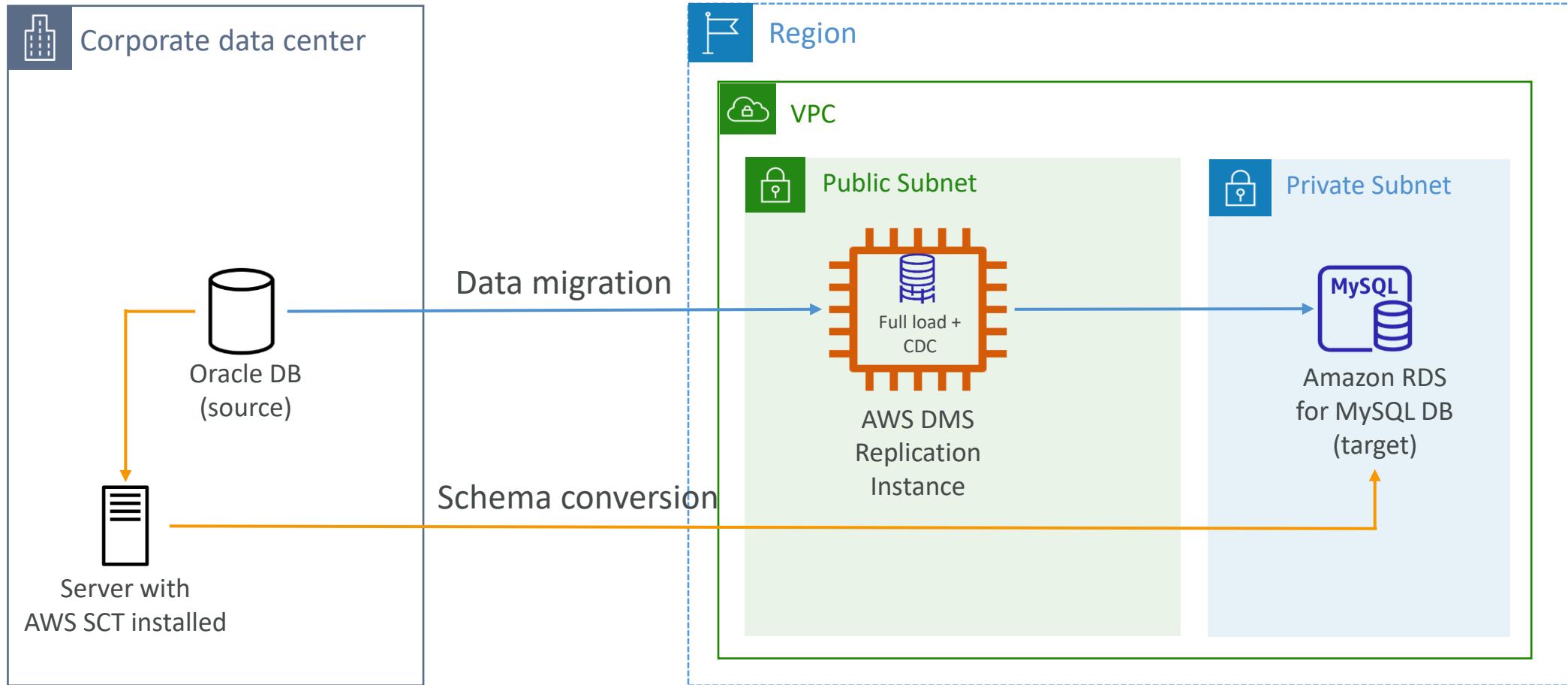
# AWS Schema Conversion Tool (SCT)

- 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



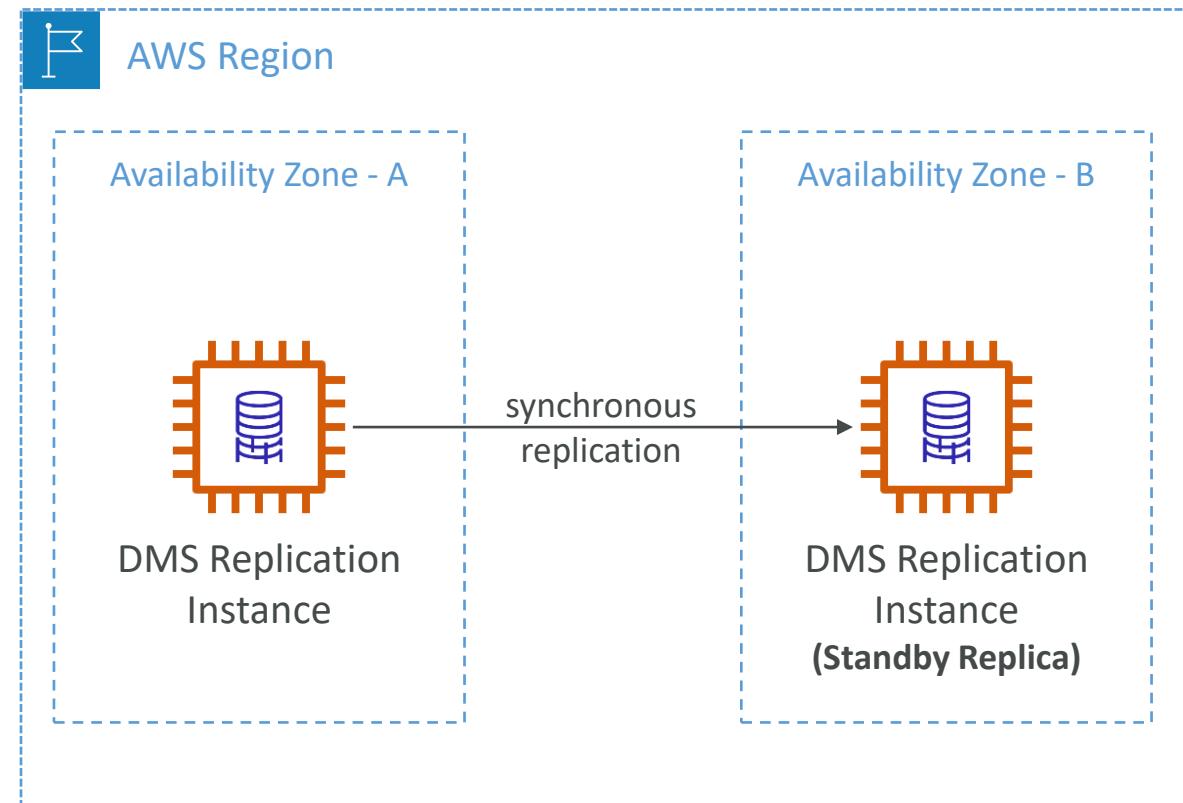
- **You do not need to use SCT if you are migrating the same DB engine**
  - Ex: On-Premise PostgreSQL => RDS PostgreSQL
  - The DB engine is still PostgreSQL (RDS is the platform)

# DMS - Continuous Replication



# AWS DMS – Multi-AZ Deployment

- When Multi-AZ Enabled, DMS provisions and maintains a synchronously stand replica in a different AZ
- Advantages:
  - Provides Data Redundancy
  - Eliminates I/O freezes
  - Minimizes latency spikes



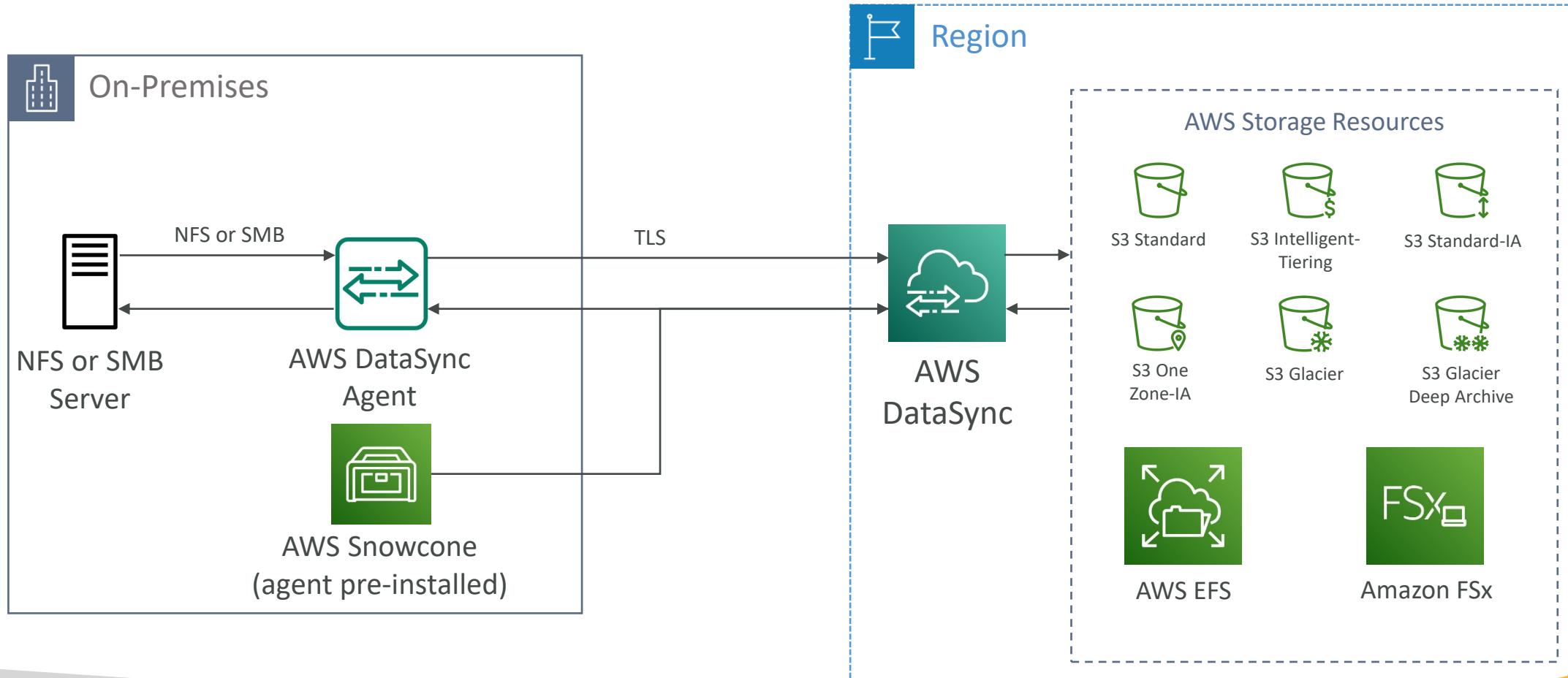
# AWS DataSync



- Move large amount of data to and from
  - On-premises / other cloud to AWS (NFS, SMB, HDFS, S3 API...) – **needs agent**
  - AWS to AWS (different storage services) – no agent needed
- Can synchronize to:
  - Amazon S3 (any storage classes – including Glacier)
  - Amazon EFS
  - Amazon FSx (Windows, Lustre, NetApp, OpenZFS...)
- Replication tasks can be scheduled hourly, daily, weekly
- **File permissions and metadata are preserved (NFS POSIX, SMB...)**
- One agent task can use 10 Gbps, can setup a bandwidth limit

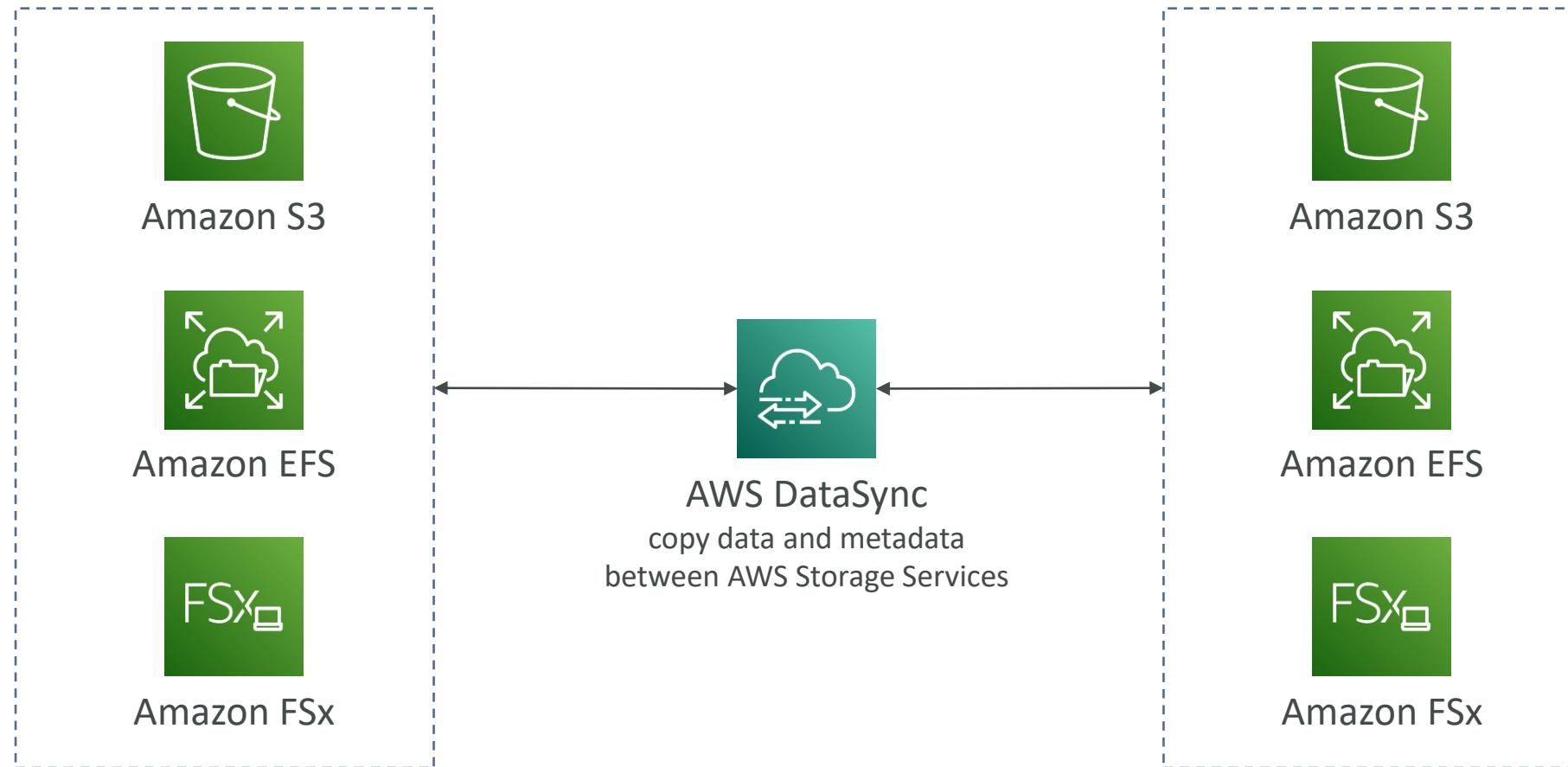
# AWS DataSync

## NFS / SMB to AWS (S3, EFS, FSx...)



# AWS DataSync

## Transfer between AWS storage services



# 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**

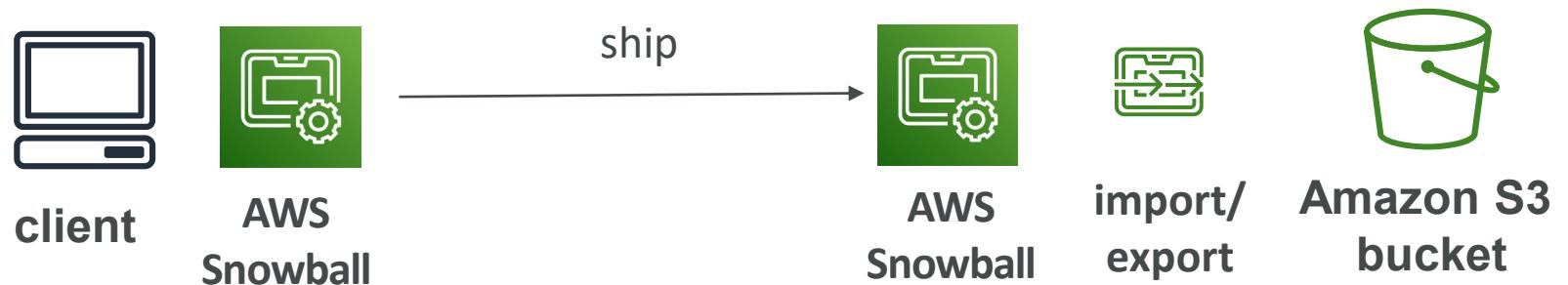
If it takes more than a week to transfer over the network, use Snowball devices!

# 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 or 28TB NVMe capacity for block volume and S3 compatible object storage
- Use cases: large data cloud migrations, DC decommission, disaster recovery



# AWS Snowcone & Snowcone SSD



- **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
- **Snowcone** – 8 TB of HDD Storage
- **Snowcone SSD** – 14 TB of SSD 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 & Snowcone SSD	Snowball Edge Storage Optimized	Snowmobile
Storage Capacity	8 TB HDD 14 TB SSD	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		

# 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)

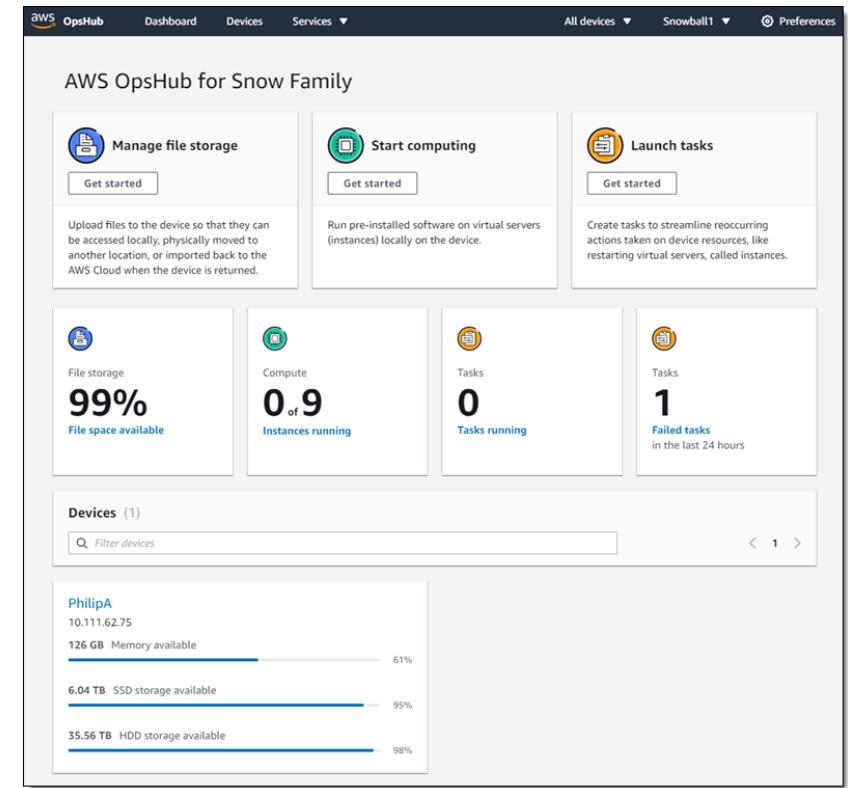
# Snow Family – Edge Computing

- **Snowcone & Snowcone SSD (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**
  - 104 vCPUs, 416 GiB of RAM
  - Optional GPU (useful for video processing or machine learning)
  - 28 TB NVMe or 42TB HDD usable storage
  - Storage Clustering available (up to 16 nodes)
- **Snowball Edge – Storage Optimized**
  - Up to 40 vCPUs, 80 GiB of RAM, 80 TB storage
- All: Can run EC2 Instances & AWS Lambda functions (using AWS IoT Greengrass)
- Long-term deployment options: 1 and 3 years discounted pricing



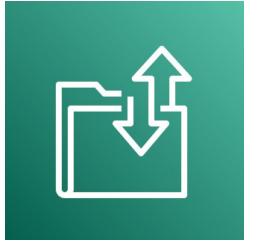
# AWS OpsHub

- 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))



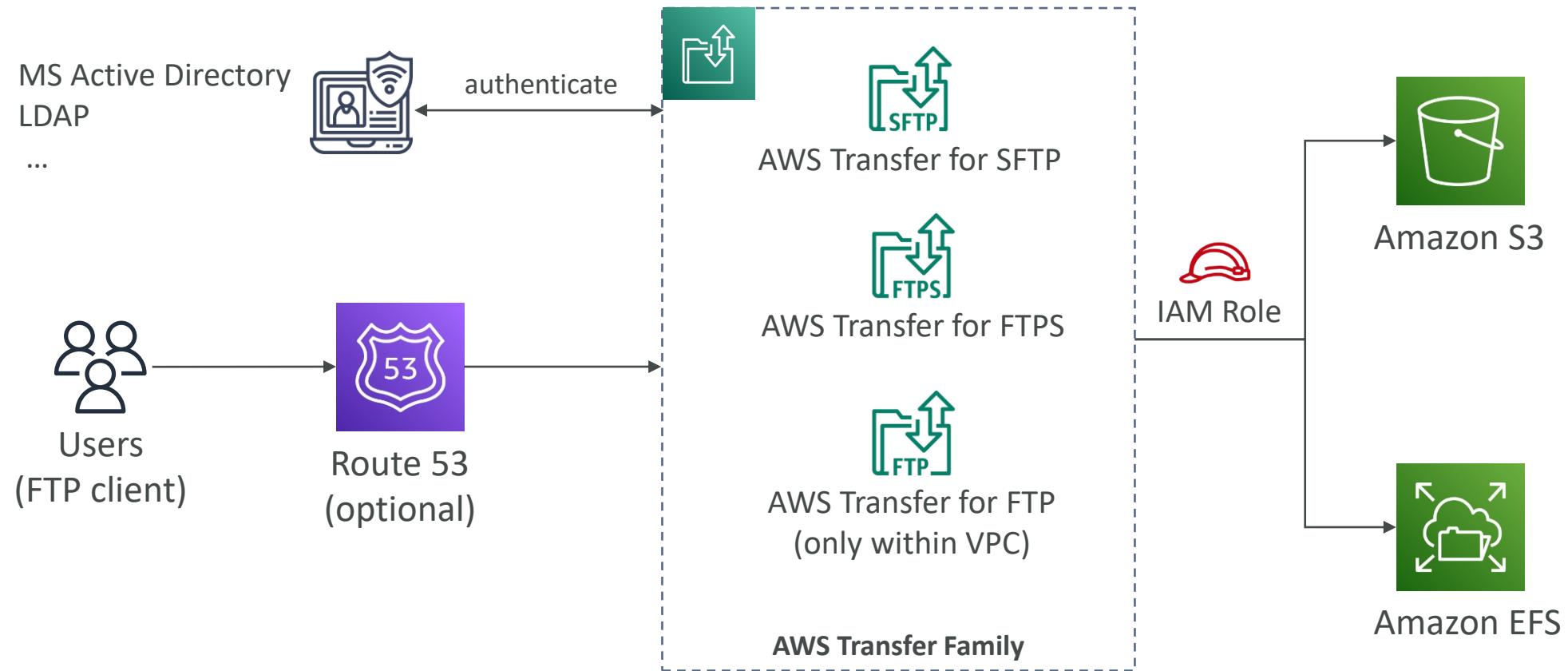
<https://aws.amazon.com/blogs/aws/aws-snowball-edge-update/>

# AWS Transfer Family



- A fully-managed service for file transfers into and out of Amazon S3 or Amazon EFS using the FTP protocol
- Supported Protocols
  - **AWS Transfer for FTP** (File Transfer Protocol (FTP))
  - **AWS Transfer for FTPS** (File Transfer Protocol over SSL (FTPS))
  - **AWS Transfer for SFTP** (Secure File Transfer Protocol (SFTP))
- Managed infrastructure, Scalable, Reliable, Highly Available (multi-AZ)
- Pay per provisioned endpoint per hour + data transfers in GB
- Store and manage users' credentials within the service
- Integrate with existing authentication systems (Microsoft Active Directory, LDAP, Okta, Amazon Cognito, custom)
- Usage: sharing files, public datasets, CRM, ERP, ...

# AWS Transfer Family



# Compute

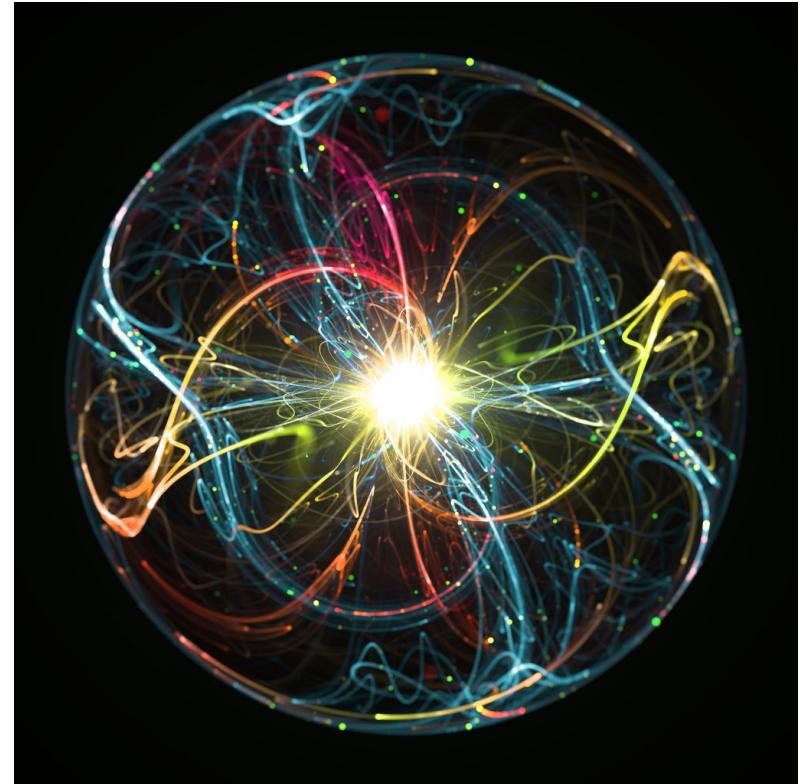
Transforming Data in AWS

# 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 Graviton

- Amazon's own family of processors, powers several EC2 instance types
  - General purpose: M7, T4
  - Compute optimized: C7, C6
  - Memory optimized: R7, X2
  - Storage optimized: Ix4, Is4
  - Accelerated computing (game streaming, ML inference): G5
- Offers best price performance
- Option for many data engineering services
  - MSK, RDS, MemoryDB, ElastiCache, OpenSearch, EMR, Lambda, Fargate



# AWS Lambda

## Serverless data processing

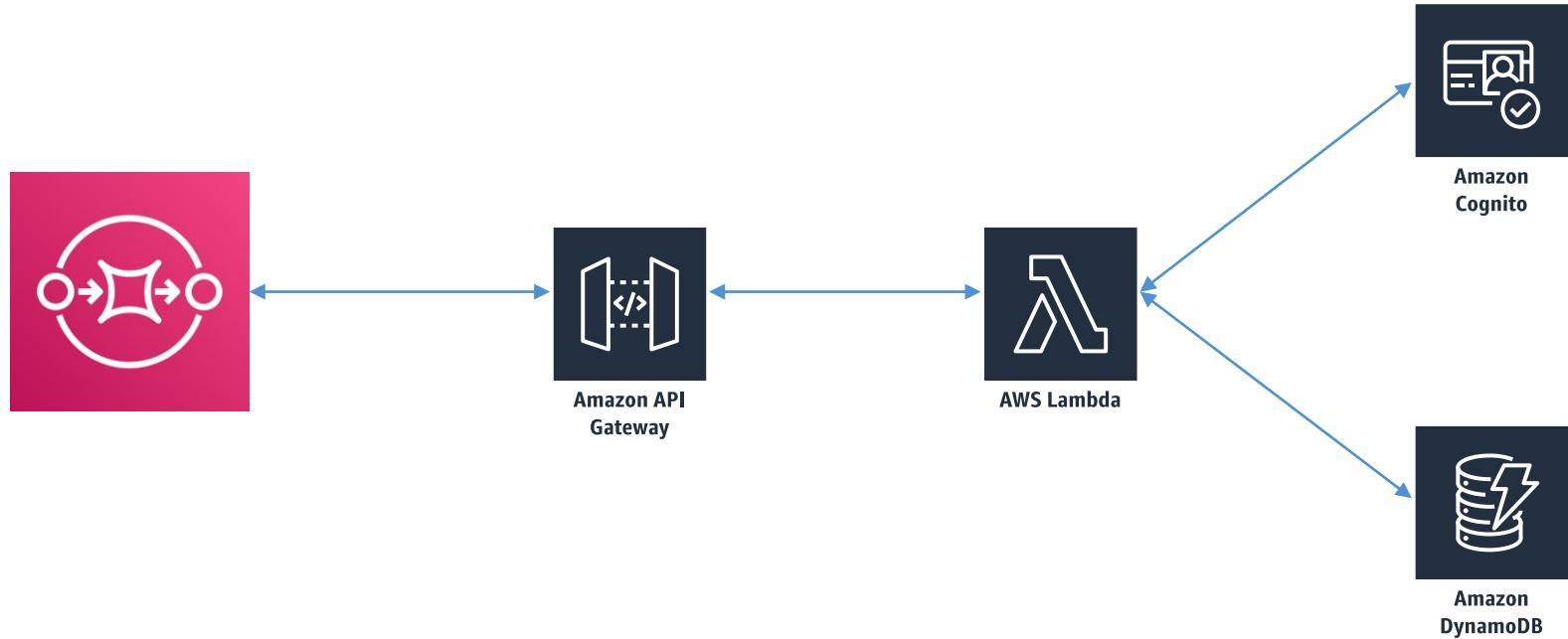
# What is Lambda?

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

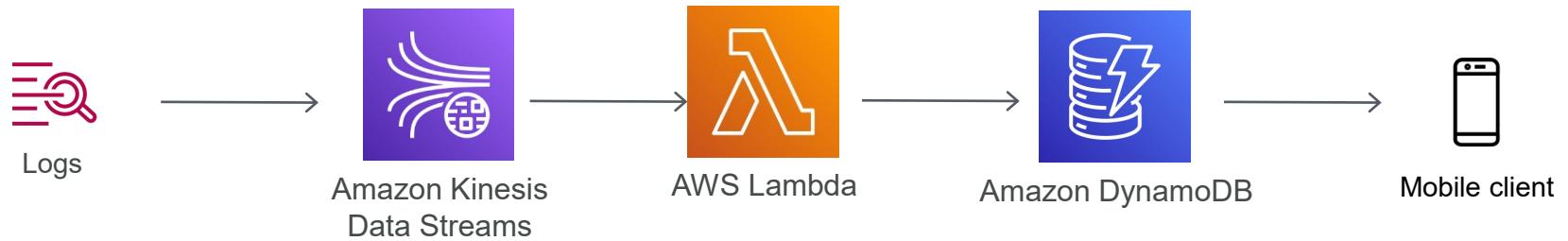


**AWS Lambda**

# Example: Serverless Website

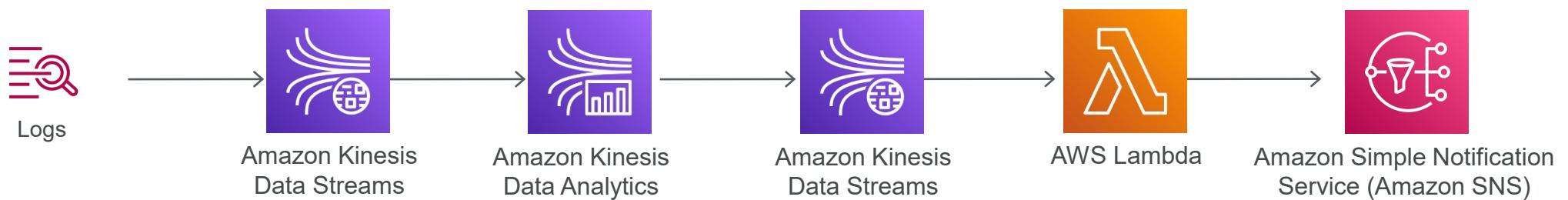


# Example: Order history app



# 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



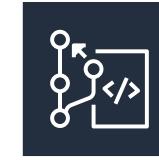
Amazon API  
Gateway



Amazon  
CloudFront



Amazon  
Cognito



AWS  
CodeCommit

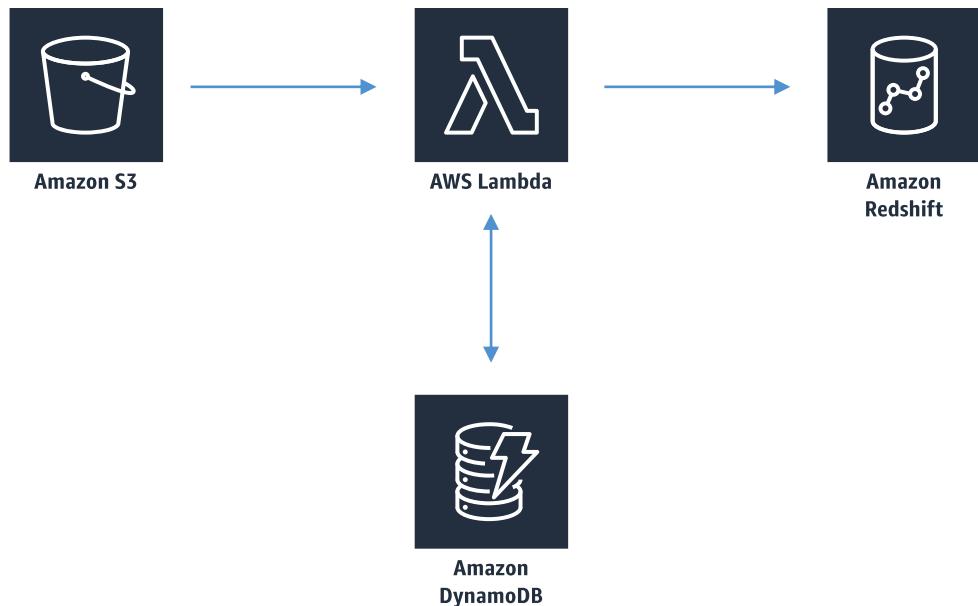
# Lambda and Amazon Opensearch Service



# Lambda and Data Pipeline



# Lambda and Redshift



Best practice for loading data into Redshift is the COPY command

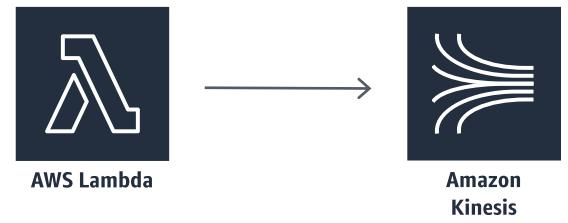
But, you can use Lambda if you need to respond to new data that shows up at any time

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



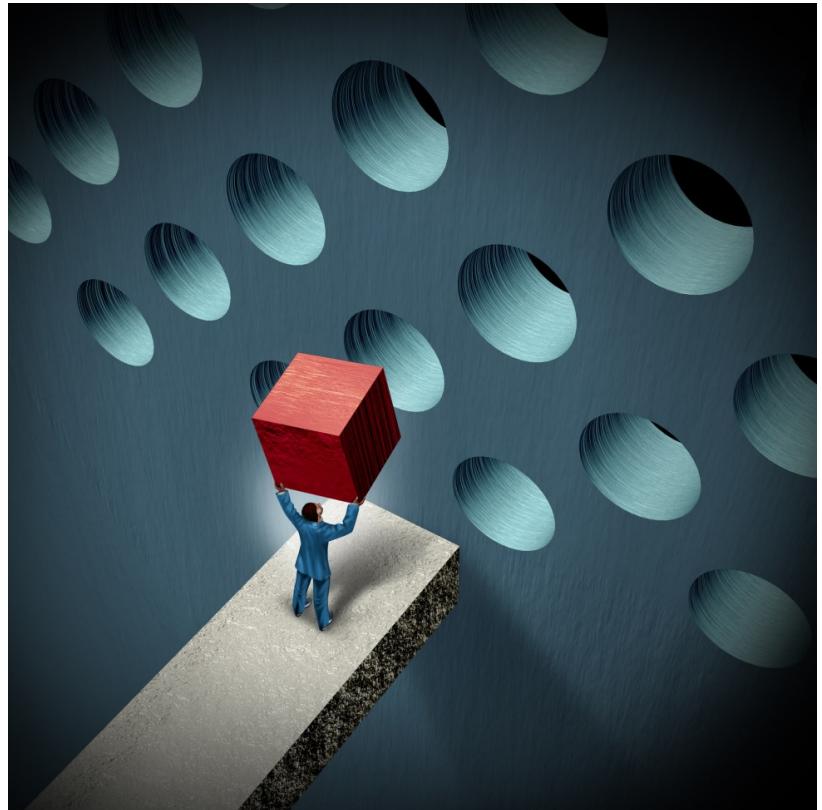
# Other promises

- 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)



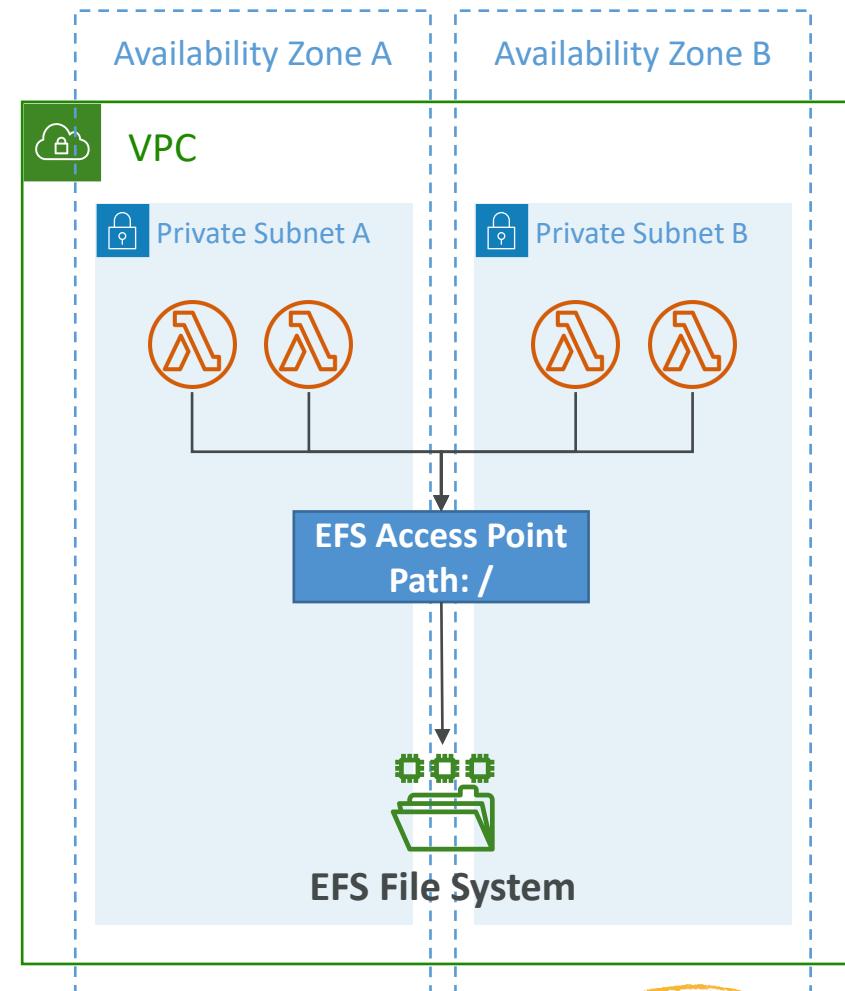
# Anti-patterns

- 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



# Lambda – File Systems Mounting

- Lambda functions can access EFS file systems if they are running in a VPC
- Configure Lambda to mount EFS file systems to local directory during initialization
- Must leverage EFS Access Points
- Limitations: watch out for the EFS connection limits (one function instance = one connection) and connection burst limits



# Lambda – Storage Options

	Ephemeral Storage /tmp	Lambda Layers	Amazon S3	Amazon EFS
<b>Max. Size</b>	10,240 MB	5 layers per function up to 250MB total	Elastic	Elastic
<b>Persistence</b>	Ephemeral	Durable	Durable	Durable
<b>Content</b>	Dynamic	Static	Dynamic	Dynamic
<b>Storage Type</b>	File System	Archive	Object	File System
<b>Operations supported</b>	any File System operation	Immutable	Atomic with Versioning	any File System operation
<b>Pricing</b>	Included in Lambda	Included in Lambda	Storage + Requests + Data Transfer	Storage + Data Transfer + Throughput
<b>Sharing/Permissions</b>	Function Only	IAM	IAM	IAM + NFS
<b>Relative Data Access Speed from Lambda</b>	Fastest	Fastest	Fast	Very Fast
<b>Shared Across All Invocations</b>	No	Yes	Yes	Yes

# AWS SAM

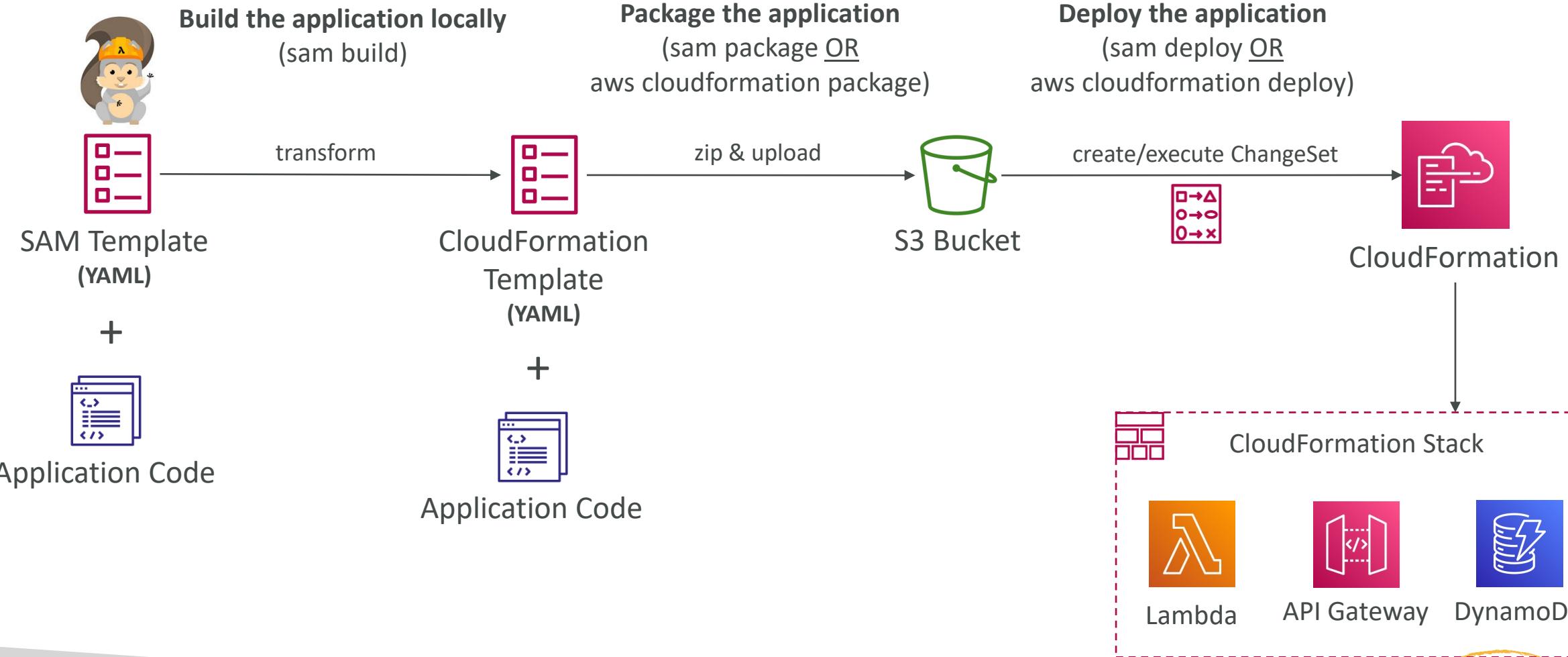


- **SAM = Serverless Application Model**
- Framework for developing and deploying serverless applications
- All the configuration is YAML code
- Generate complex CloudFormation from simple SAM YAML file
- Supports anything from CloudFormation: Outputs, Mappings, Parameters, Resources...
- Only two commands to deploy to AWS
- SAM can use CodeDeploy to deploy Lambda functions
- SAM can help you to run Lambda, API Gateway, DynamoDB locally

# AWS SAM – Recipe

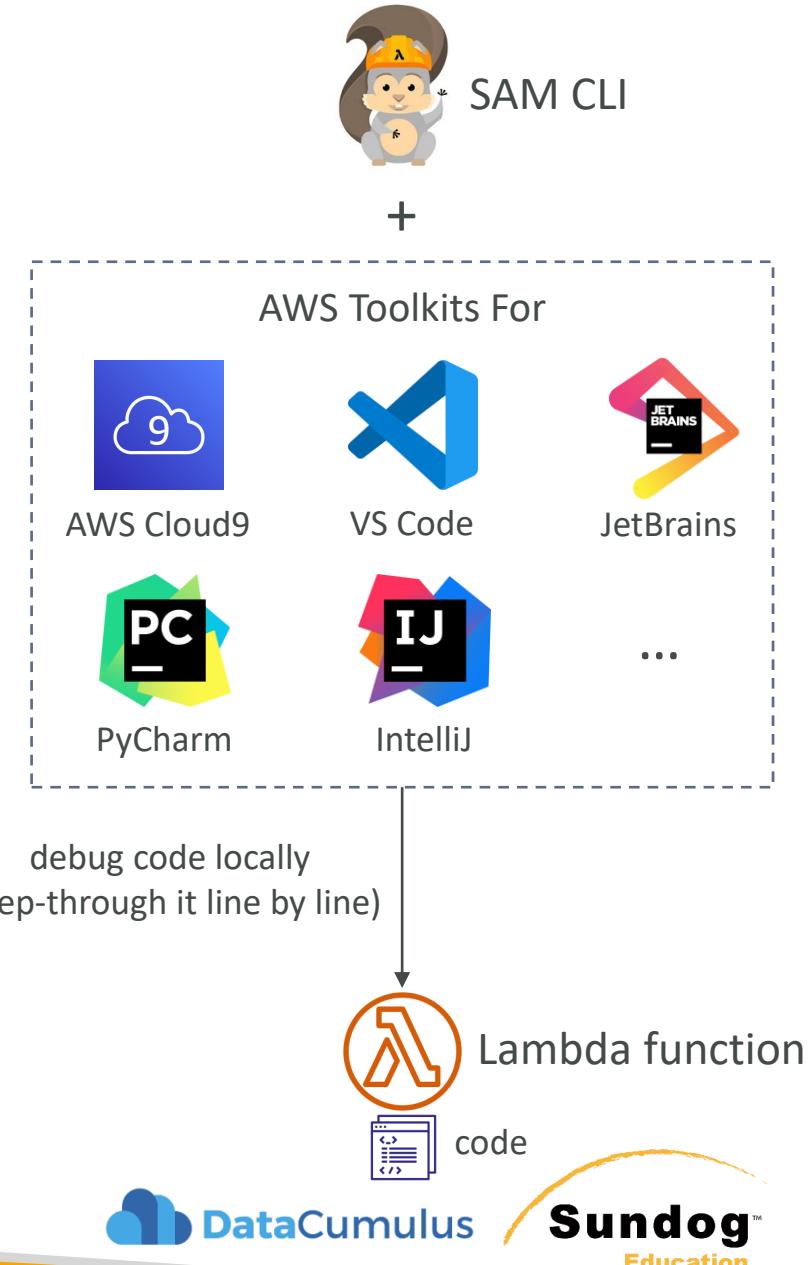
- **Transform Header indicates it's SAM template:**
  - Transform: 'AWS::Serverless-2016-10-31'
- **Write Code**
  - AWS::Serverless::Function
  - AWS::Serverless::Api
  - AWS::Serverless::SimpleTable
- **Package & Deploy:**
  - aws cloudformation package / sam package
  - aws cloudformation deploy / sam deploy

# Deep Dive into SAM Deployment



# SAM – CLI Debugging

- Locally build, test, and debug your serverless applications that are defined using AWS SAM templates
- Provides a lambda-like execution environment locally
- SAM CLI + AWS Toolkits => step-through and debug your code
- Supported IDEs: AWS Cloud9, Visual Studio Code, JetBrains, PyCharm, IntelliJ, ...
- **AWS Toolkits:** IDE plugins which allows you to build, test, debug, deploy, and invoke Lambda functions built using AWS SAM



# AWS Batch



- Run batch jobs as Docker images
- Dynamic provisioning of the instances (EC2 & Spot Instances)
- Optimal quantity and type based on volume and requirements
- No need to manage clusters, fully **serverless**
- You just pay for the underlying EC2 instances
  
- Schedule Batch Jobs using CloudWatch Events
- Orchestrate Batch Jobs using AWS Step Functions

# AWS Batch vs Glue

- Glue:
  - Glue ETL - Run Apache Spark code, Scala or Python based, focus on the ETL
  - Glue ETL - Do not worry about configuring or managing the resources
  - Data Catalog to make the data available to Athena or Redshift Spectrum
- Batch:
  - For any computing job regardless of the job (must provide Docker image)
  - Resources are created in your account, managed by Batch
  - For any non-ETL related work, Batch is probably better

# Containers

Packaging applications in AWS

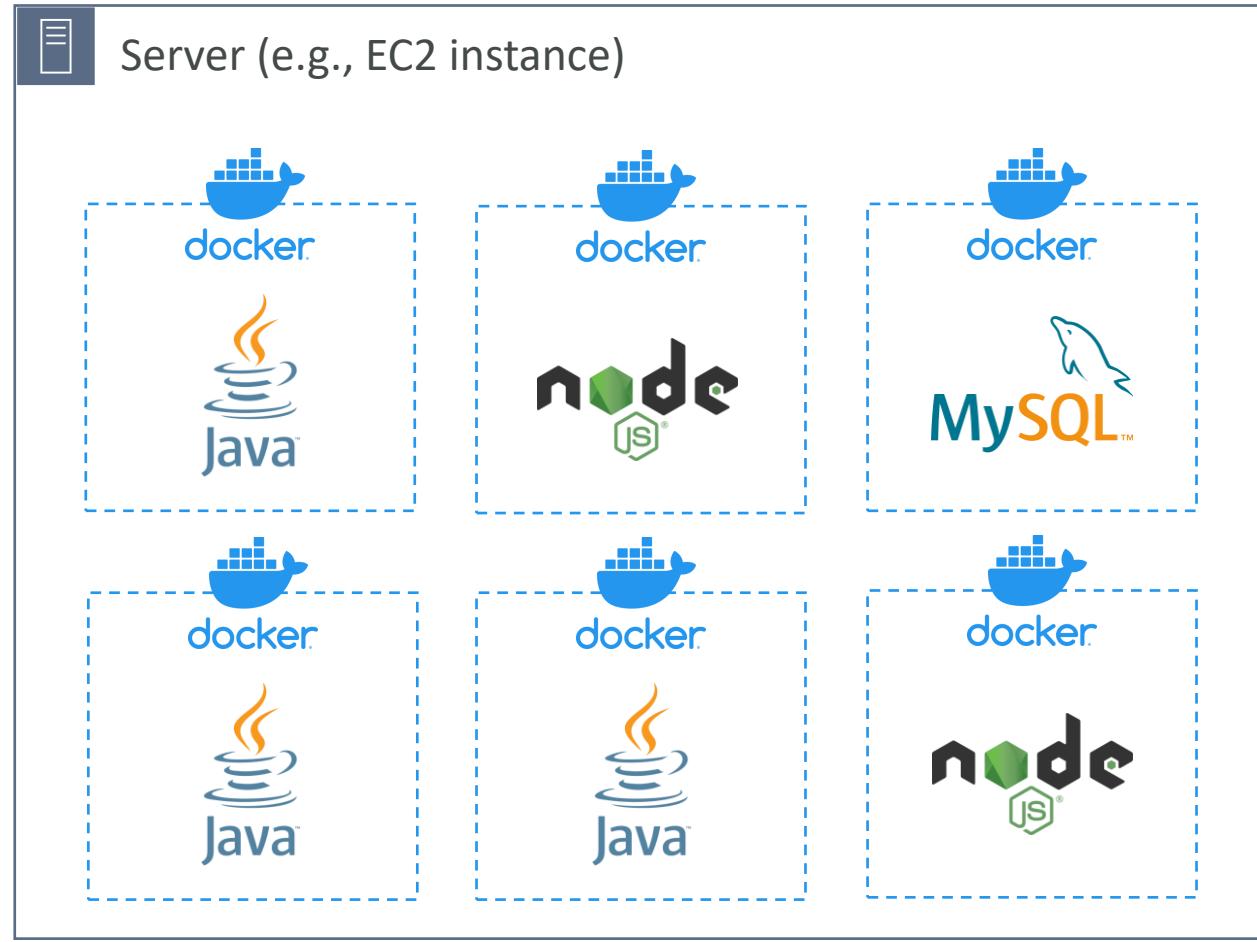
# Containers on AWS

# What is Docker?



- Docker is a software development platform to deploy apps
- Apps are packaged in **containers** that can be run on any OS
- Apps run the same, regardless of where they're run
  - Any machine
  - No compatibility issues
  - Predictable behavior
  - Less work
  - Easier to maintain and deploy
  - Works with any language, any OS, any technology
- Use cases: microservices architecture, lift-and-shift apps from on-premises to the AWS cloud, ...

# Docker on an OS

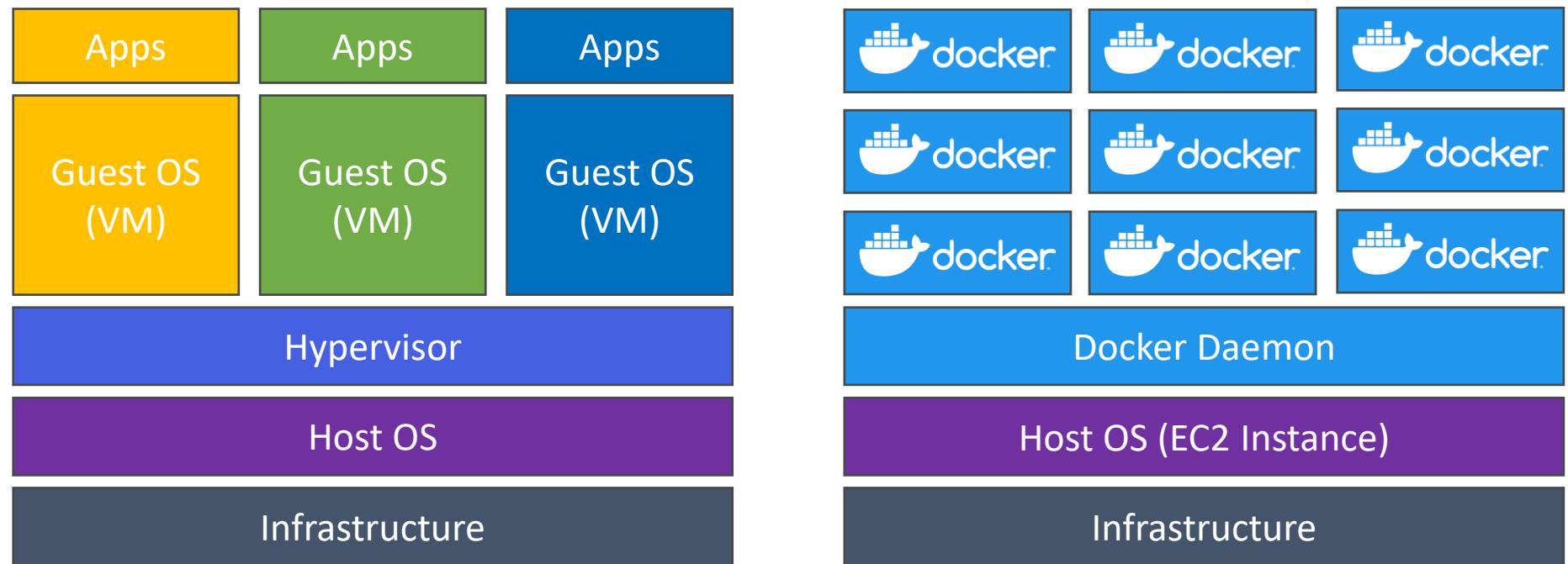


# Where are Docker images stored?

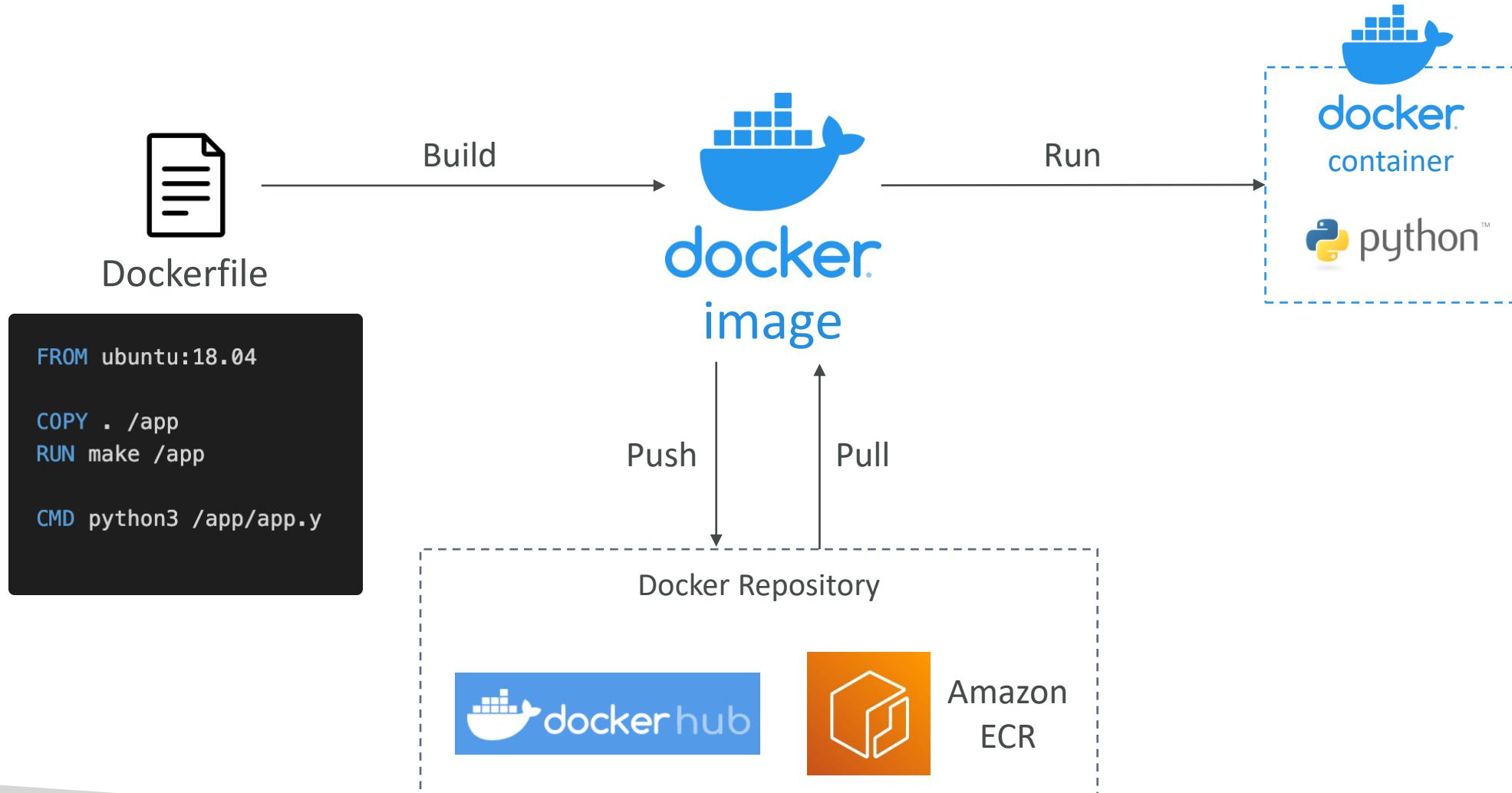
- Docker images are stored in Docker Repositories
- Docker Hub (<https://hub.docker.com>)
  - Public repository
  - Find base images for many technologies or OS (e.g., Ubuntu, MySQL, ...)
- Amazon ECR (Amazon Elastic Container Registry)
  - Private repository
  - Public repository (Amazon ECR Public Gallery  
<https://gallery.ecr.aws>)

# Docker vs. Virtual Machines

- Docker is "sort of" a virtualization technology, but not exactly
- Resources are shared with the host => many containers on one server



# Getting Started with Docker



# Docker Containers Management on AWS

- **Amazon Elastic Container Service (Amazon ECS)**

- Amazon's own container platform



Amazon ECS

- **Amazon Elastic Kubernetes Service (Amazon EKS)**

- Amazon's managed Kubernetes (open source)



Amazon EKS

- **AWS Fargate**

- Amazon's own Serverless container platform
- Works with ECS and with EKS



AWS Fargate

- **Amazon ECR:**

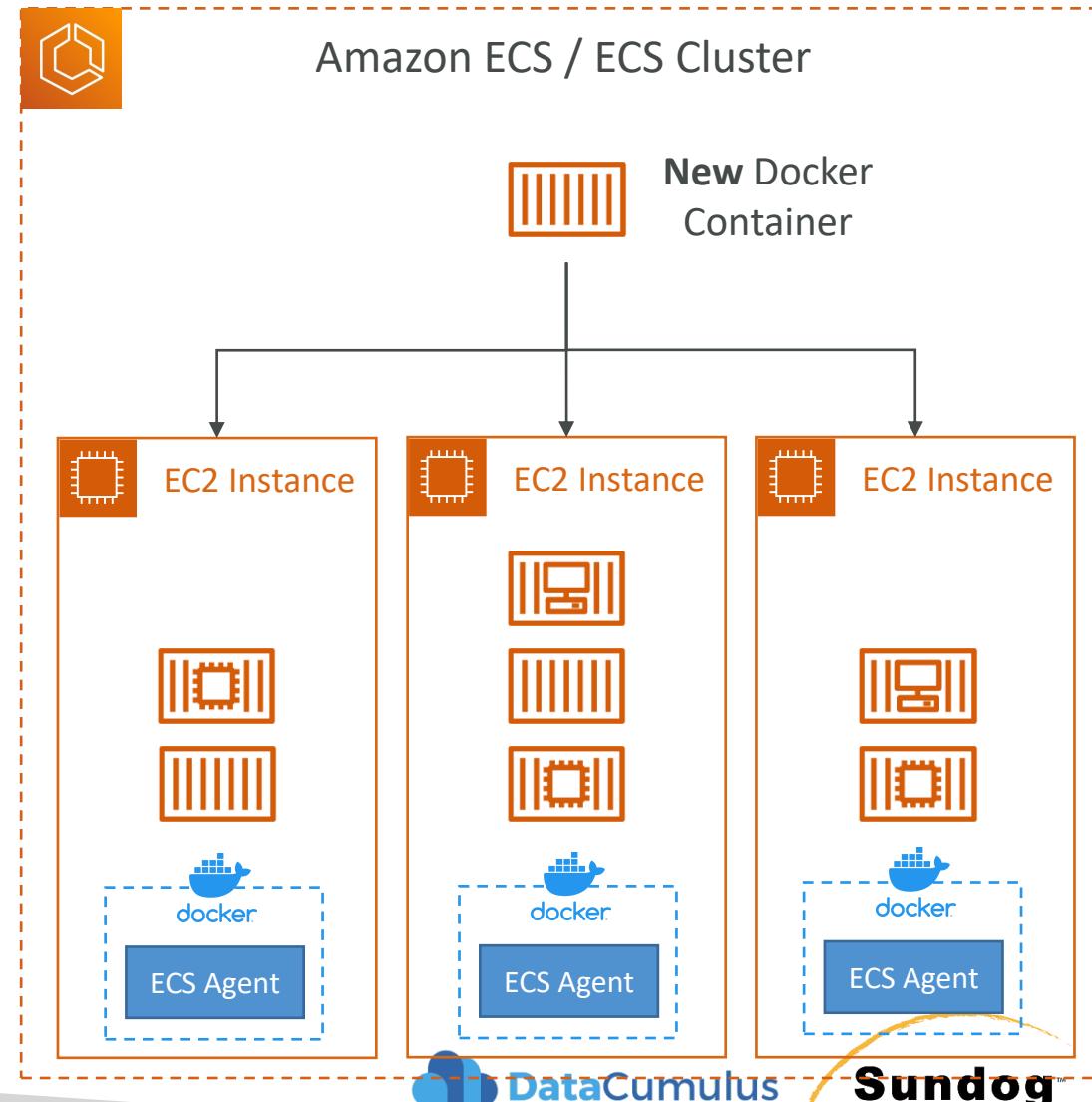
- Store container images



Amazon ECR

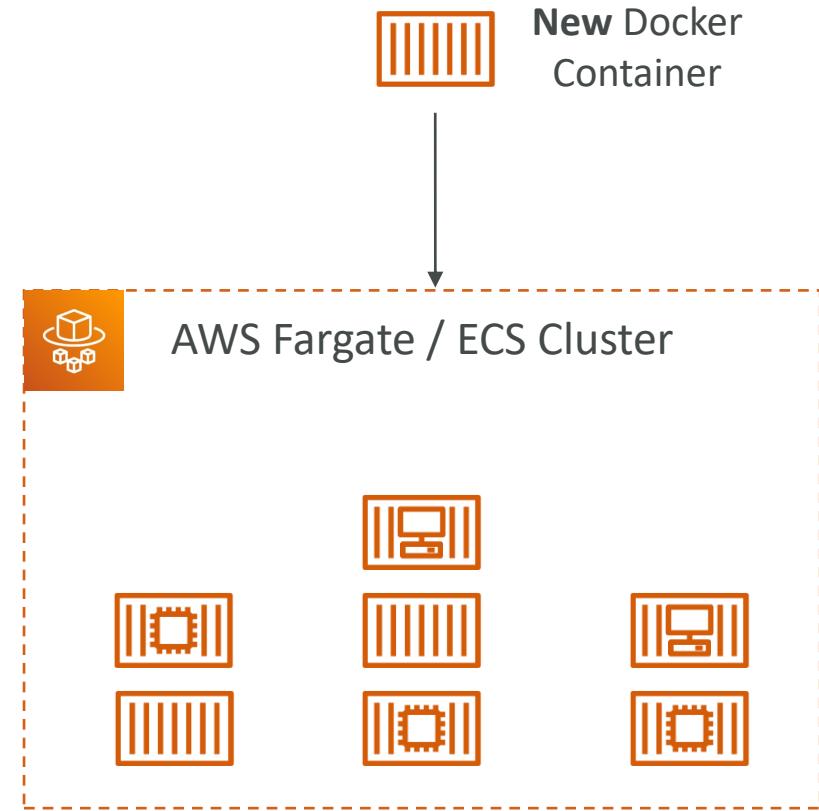
# Amazon ECS - EC2 Launch Type

- ECS = Elastic Container Service
- Launch Docker containers on AWS = Launch **ECS Tasks** on ECS Clusters
- **EC2 Launch Type:** you must provision & maintain the infrastructure (the EC2 instances)
- Each EC2 Instance must run the ECS Agent to register in the ECS Cluster
- AWS takes care of starting / stopping containers



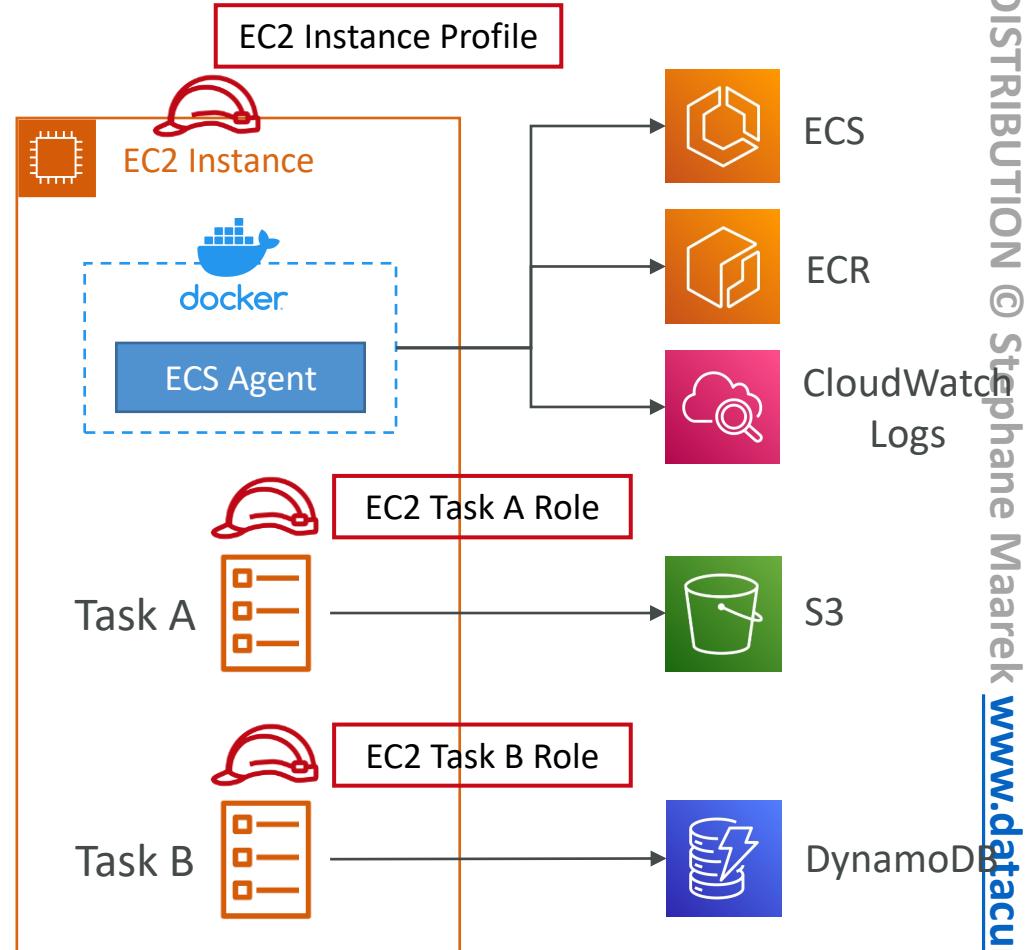
# Amazon ECS – Fargate Launch Type

- Launch Docker containers on AWS
- You do not provision the infrastructure (no EC2 instances to manage)
- **It's all Serverless!**
- You just create task definitions
- AWS just runs ECS Tasks for you based on the CPU / RAM you need
- To scale, just increase the number of tasks. Simple - no more EC2 instances



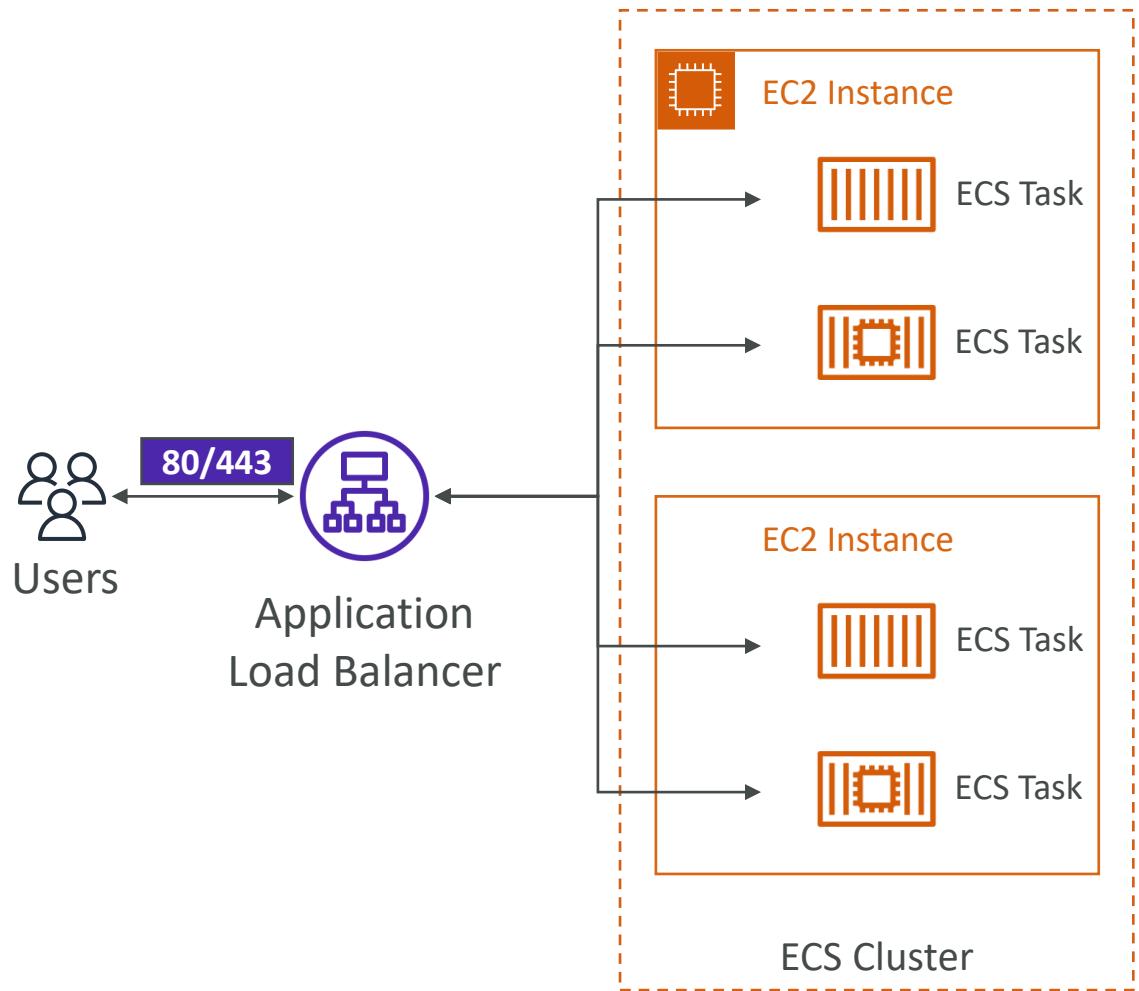
# Amazon ECS – IAM Roles for ECS

- **EC2 Instance Profile (EC2 Launch Type only):**
  - Used by the ECS agent
  - Makes API calls to ECS service
  - Send container logs to CloudWatch Logs
  - Pull Docker image from ECR
  - Reference sensitive data in Secrets Manager or SSM Parameter Store
- **ECS Task Role:**
  - Allows each task to have a specific role
  - Use different roles for the different ECS Services you run
  - Task Role is defined in the task definition



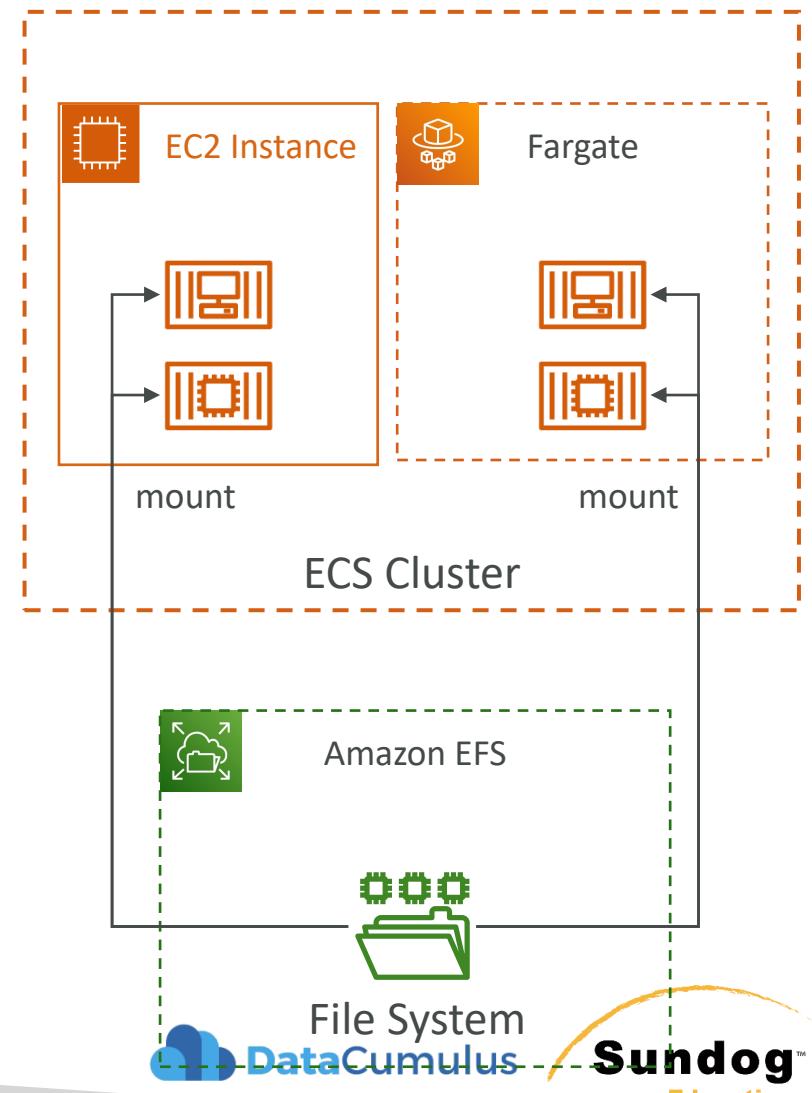
# Amazon ECS – Load Balancer Integrations

- **Application Load Balancer**  
supported and works for most use cases
- **Network Load Balancer**  
recommended only for high throughput / high performance use cases, or to pair it with AWS Private Link
- **Classic Load Balancer** supported but not recommended (no advanced features – no Fargate)



# Amazon ECS – Data Volumes (EFS)

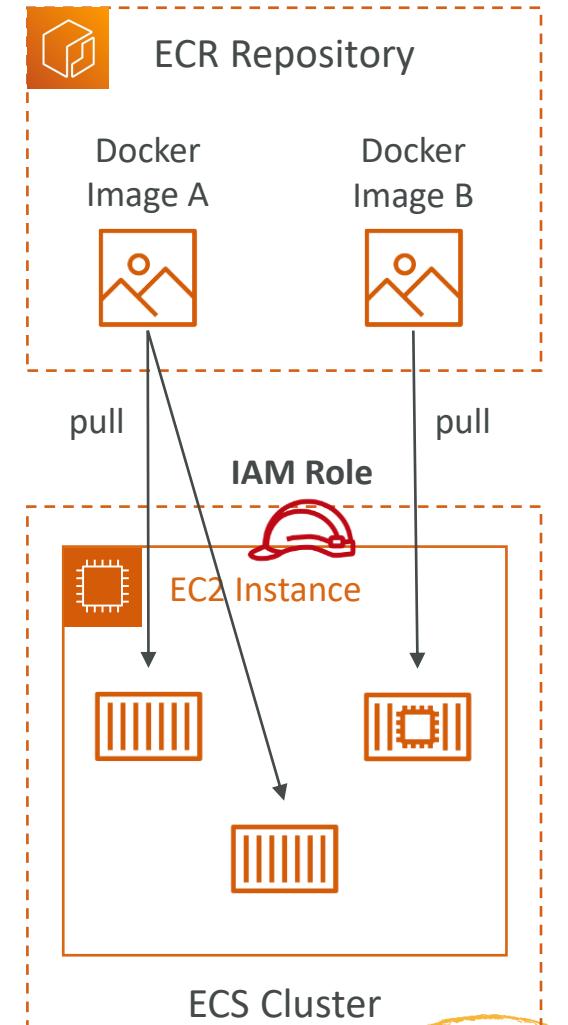
- Mount EFS file systems onto ECS tasks
- Works for both **EC2** and **Fargate** launch types
- Tasks running in any AZ will share the same data in the EFS file system
- **Fargate + EFS = Serverless**
- Use cases: persistent multi-AZ shared storage for your containers
- Note:
  - Amazon S3 cannot be mounted as a file system





# Amazon ECR

- ECR = Elastic Container Registry
- Store and manage Docker images on AWS
- **Private and Public** repository (**Amazon ECR Public Gallery** <https://gallery.ecr.aws>)
- Fully integrated with ECS, backed by Amazon S3
- Access is controlled through IAM (permission errors => policy)
- Supports image vulnerability scanning, versioning, image tags, image lifecycle, ...

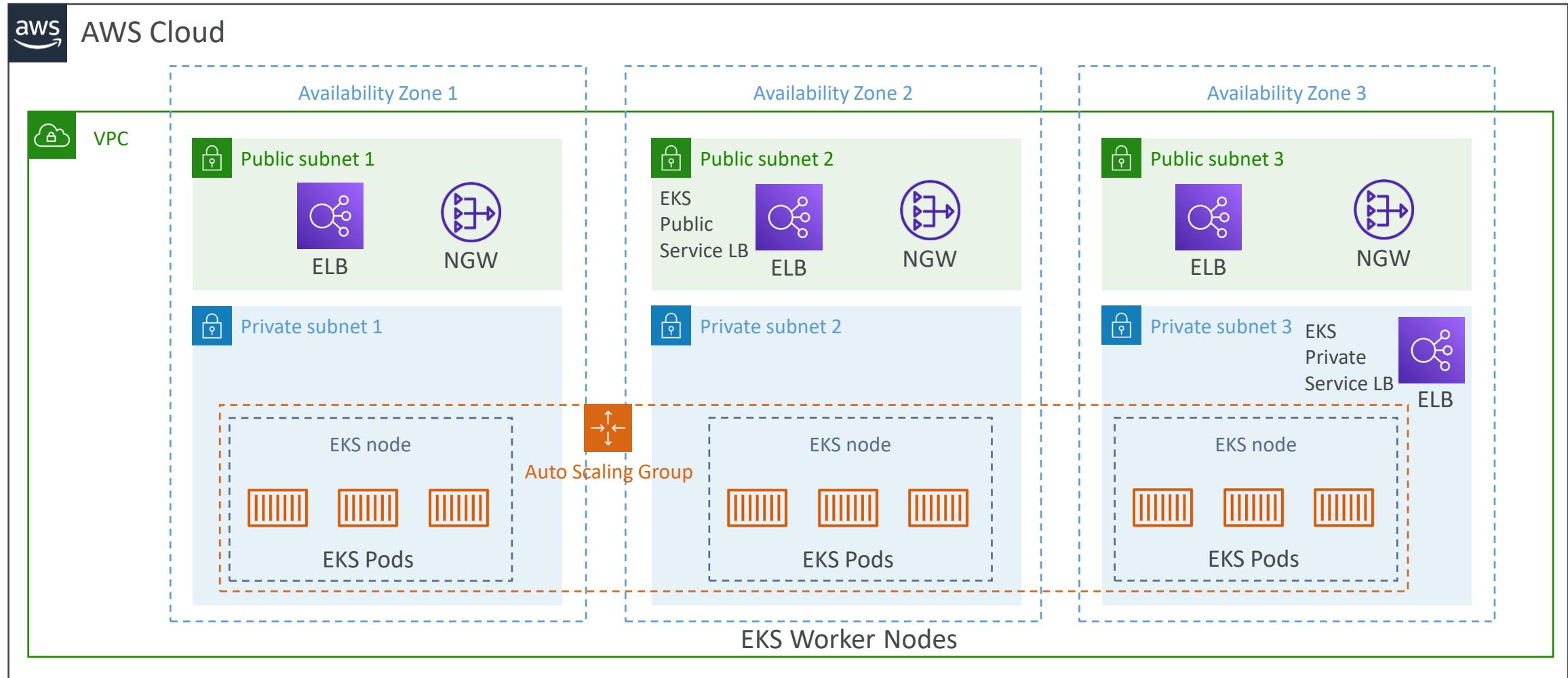


# Amazon EKS Overview



- Amazon EKS = Amazon Elastic **Kubernetes** Service
- It is a way to launch **managed Kubernetes clusters on AWS**
- Kubernetes is an **open-source system** for automatic deployment, scaling and management of containerized (usually Docker) application
- It's an alternative to ECS, similar goal but different API
- EKS supports **EC2** if you want to deploy worker nodes or **Fargate** to deploy serverless containers
- **Use case:** if your company is already using Kubernetes on-premises or in another cloud, and wants to migrate to AWS using Kubernetes
- **Kubernetes is cloud-agnostic** (can be used in any cloud – Azure, GCP...)
- For multiple regions, deploy one EKS cluster per region
- Collect logs and metrics using **CloudWatch Container Insights**

# Amazon EKS - Diagram



# Amazon EKS – Node Types

- **Managed Node Groups**

- Creates and manages Nodes (EC2 instances) for you
- Nodes are part of an ASG managed by EKS
- Supports On-Demand or Spot Instances

- **Self-Managed Nodes**

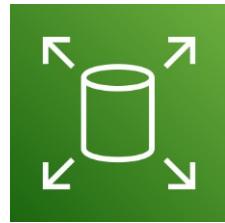
- Nodes created by you and registered to the EKS cluster and managed by an ASG
- You can use prebuilt AMI - Amazon EKS Optimized AMI
- Supports On-Demand or Spot Instances

- **AWS Fargate**

- No maintenance required; no nodes managed

# Amazon EKS – Data Volumes

- Need to specify **StorageClass** manifest on your EKS cluster
- Leverages a **Container Storage Interface (CSI)** compliant driver
- Support for...
- Amazon EBS
- Amazon EFS (works with Fargate)
- Amazon FSx for Lustre
- Amazon FSx for NetApp ONTAP



# Analytics

Extracting structure and meaning from data in AWS

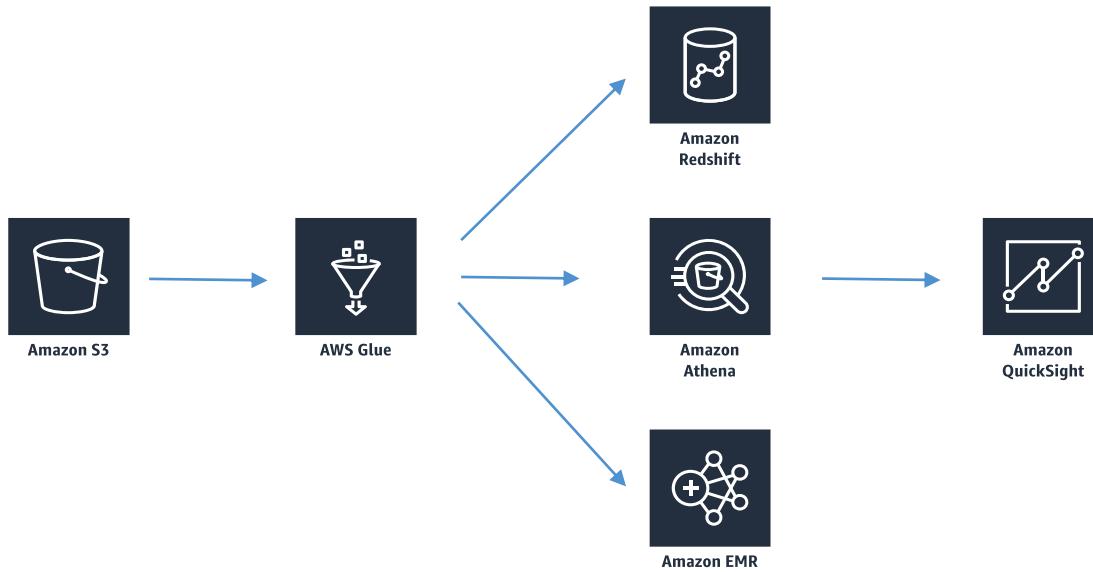
# AWS Glue

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

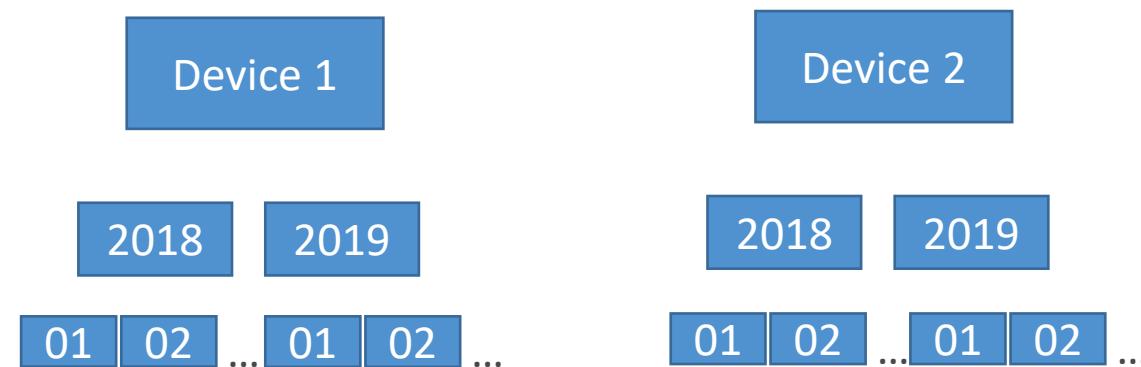
# Glue Crawler / Data Catalog



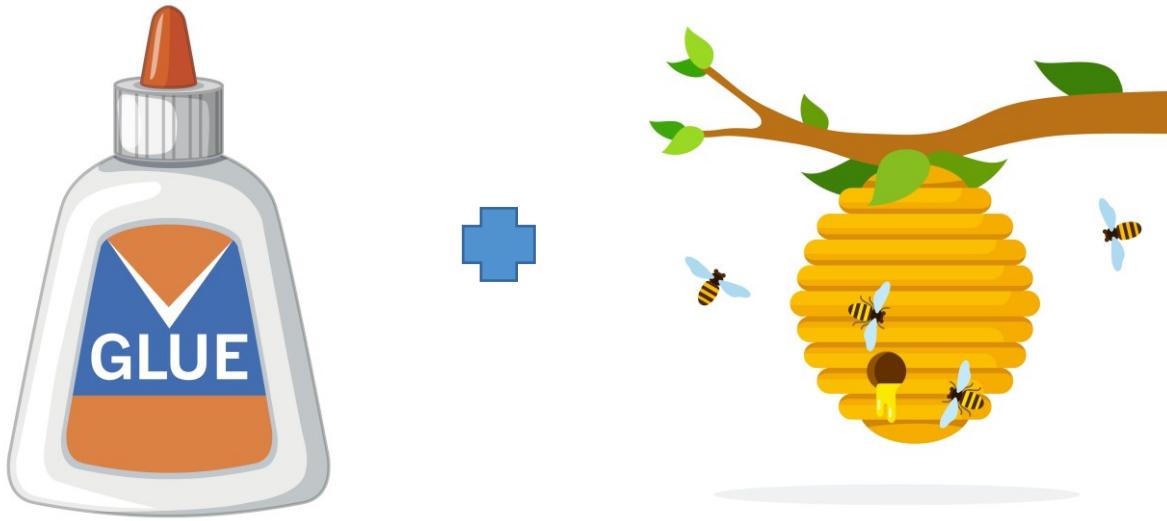
- Glue crawler scans data in S3, creates schema
- Can run periodically
- Populates the Glue Data Catalog
  - Stores only table definition
  - Original data stays in S3
- Once cataloged, you can treat your unstructured data like it's structured
  - Redshift Spectrum
  - Athena
  - EMR
  - Quicksight

# 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

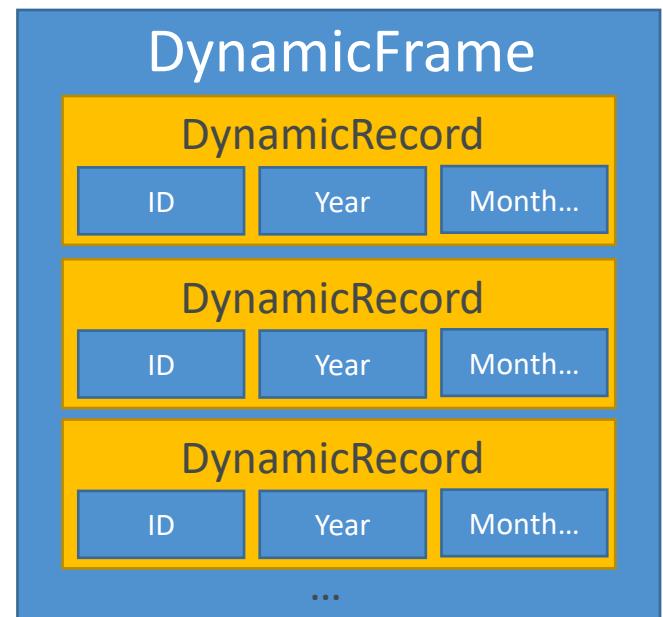
# Glue ETL

- 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

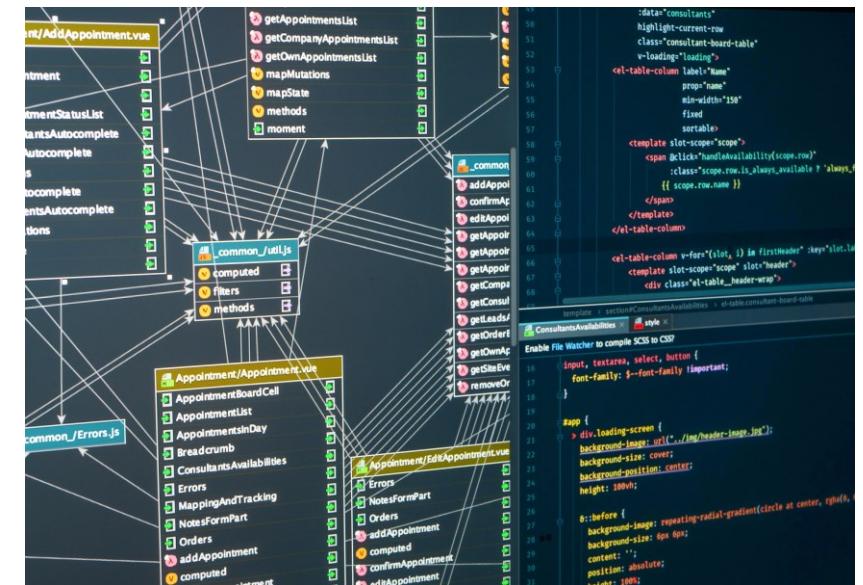
- Deals with ambiguities in a DynamicFrame and returns a new one
- For example, two fields with the same name.
- make\_cols: creates a new column for each type
  - price\_double, price\_string
- cast: casts all values to specified type
- make\_struct: Creates a structure that contains each data type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



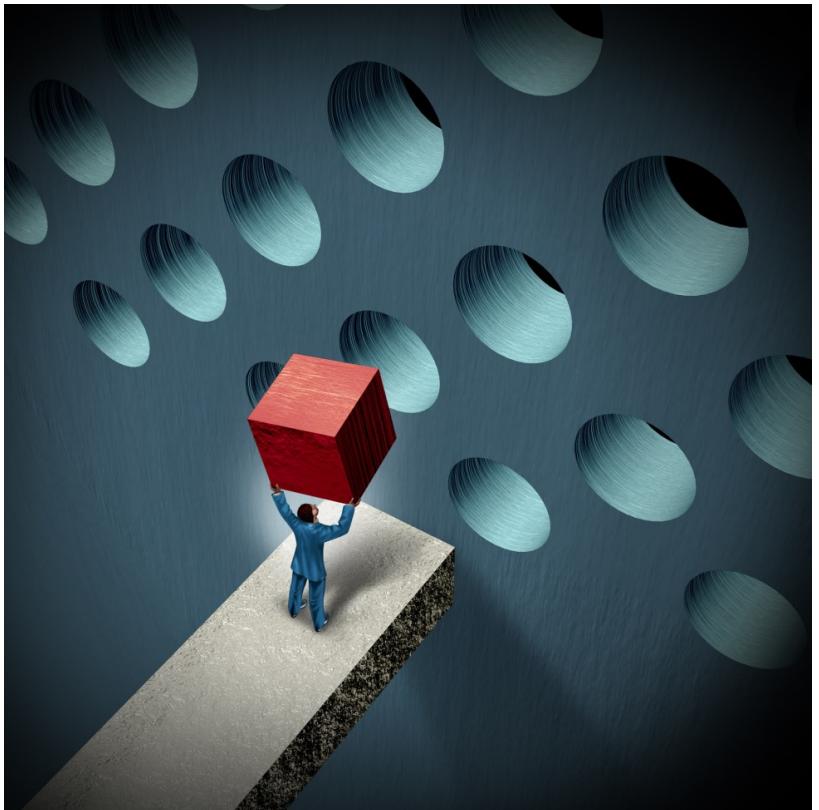
# Glue cost model

- 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.

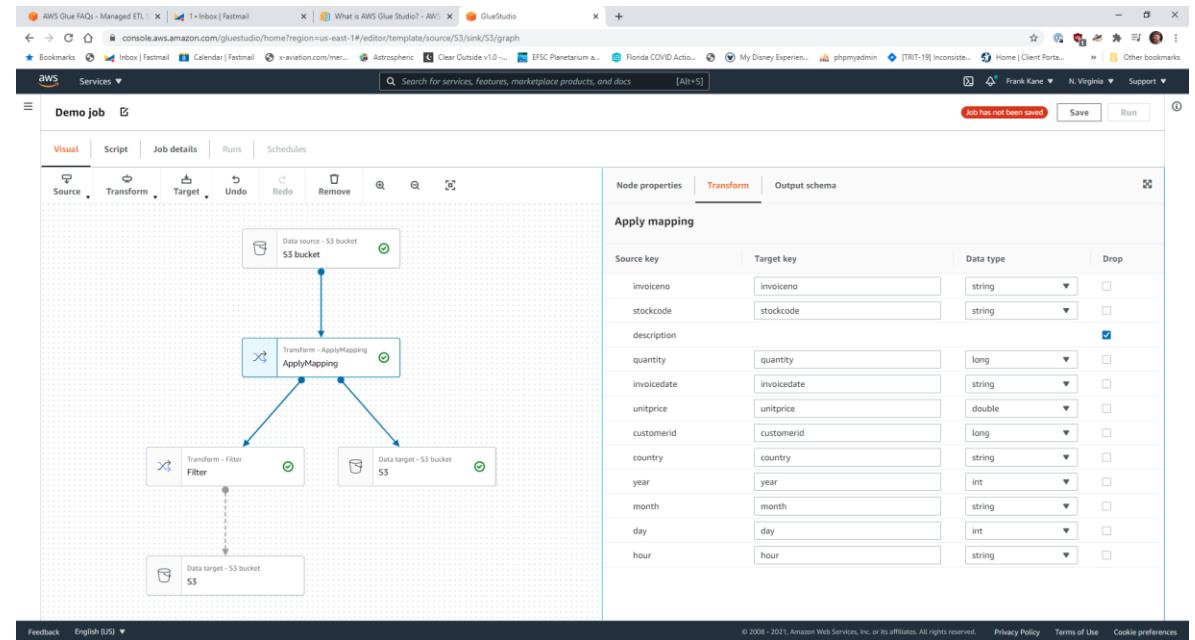


# No longer an anti-pattern: streaming

- 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

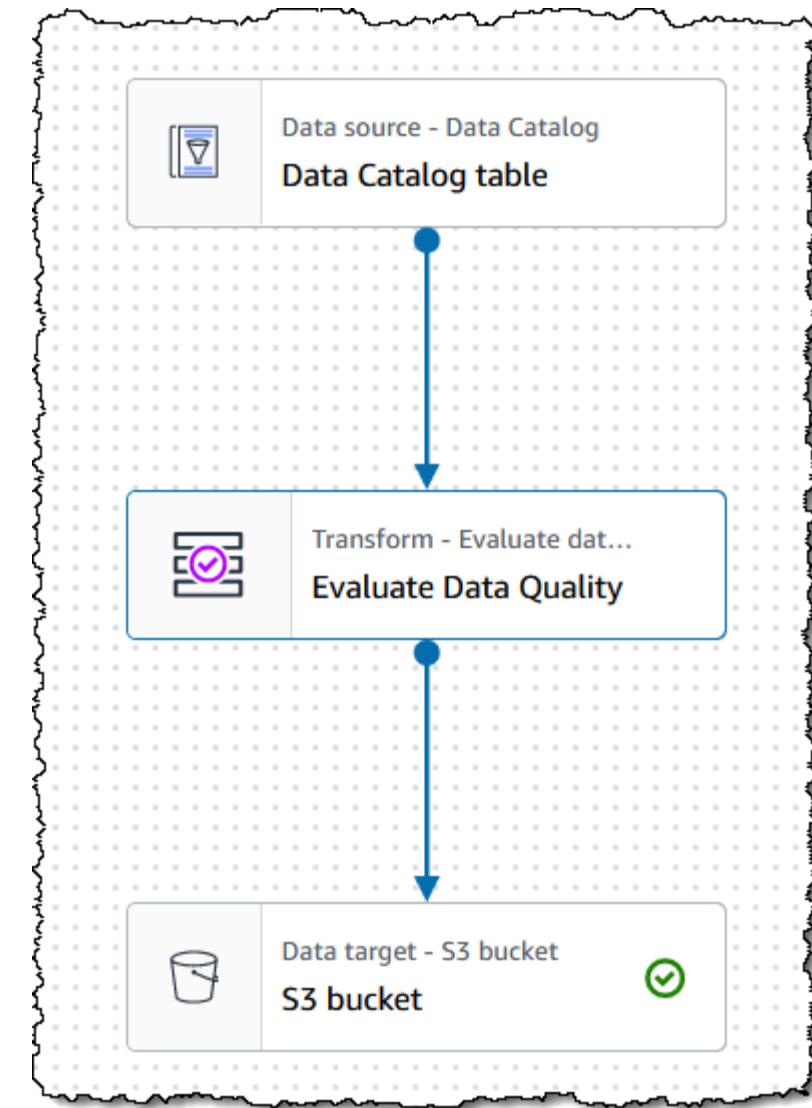
# AWS Glue Studio

- 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

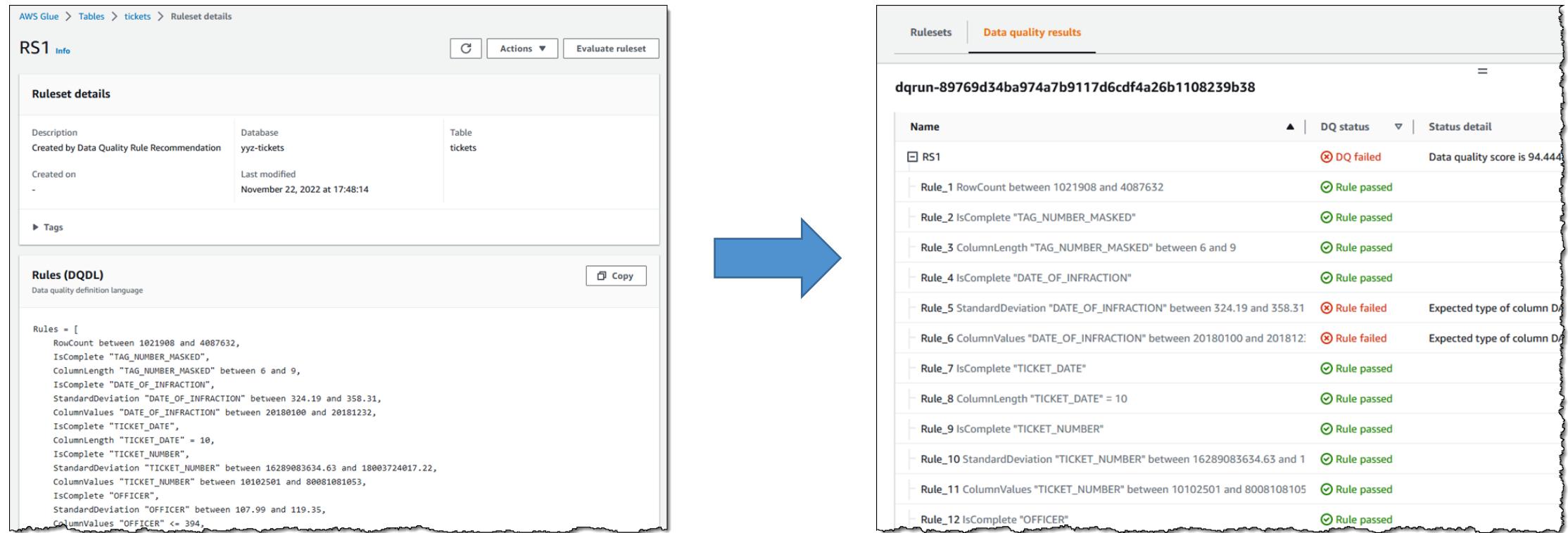


# AWS Glue Data Quality

- Data quality rules may be created manually or recommended automatically
- Integrates into Glue jobs
- Uses Data Quality Definition Language (DQDL)
- Results can be used to fail the job, or just be reported to CloudWatch



# AWS Glue Data Quality



# 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

```
{  
  "RecipeAction": {  
    "Operation": "NEST_TO_MAP",  
    "Parameters": {  
      "sourceColumns":  
        "[\"age\", \"weight_kg\", \"height_cm\"]",  
      "targetColumn": "columnName",  
      "removeSourceColumns": "true"  
    }  
  }  
}
```

*Example: convert selected columns to key-value pairs*

# Handling Personally Identifiable Information (PII) in DataBrew

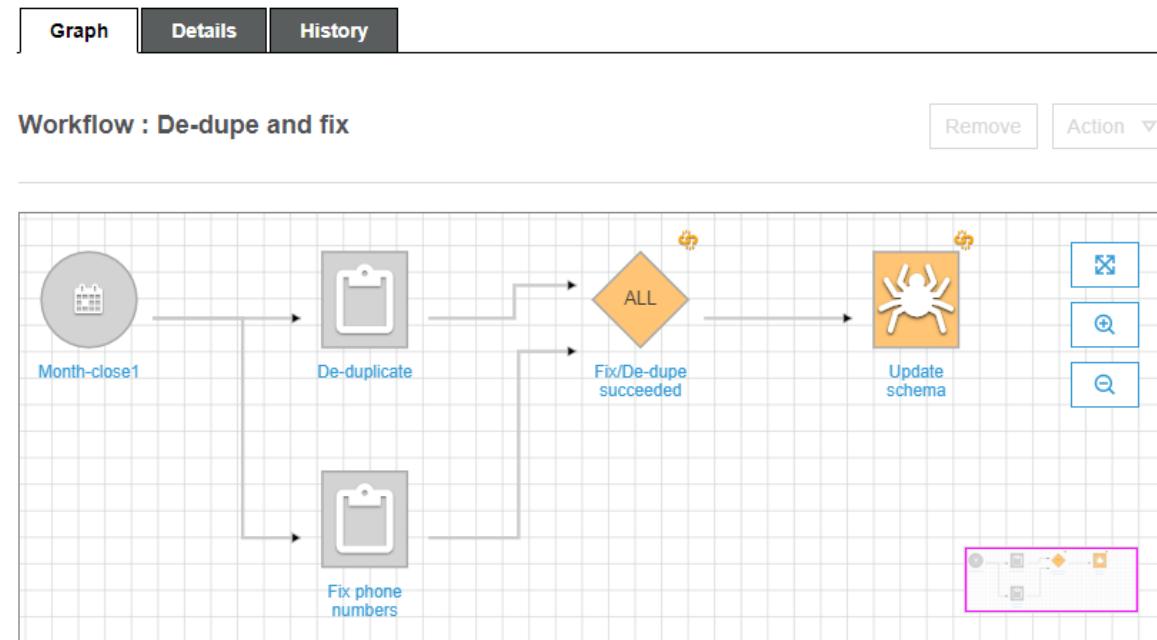
## Transformations

- Enable PII statistics in a DataBrew profile job to identify PII
- Substitution (REPLACE\_WITH\_RANDOM...)
- Shuffling (SHUFFLE\_ROWS)
- Deterministic encryption (DETERMINISTIC\_ENCRYPT)
- Probabilistic encryption (ENCRYPT)
- Decryption (DECRYPT)
- Nulling out or deletion (DELETE)
- Masking out (MASK\_CUSTOM, \_DATE, \_DELIMITER, \_RANGE)
- Hashing (CRYPTOGRAPHIC\_HASH)



# Glue Workflows

- Design multi-job, multi-crawler ETL processes run together
- Create from AWS Glue blueprint, from the console, or API
- This is only for orchestrating complex ETL operations using Glue



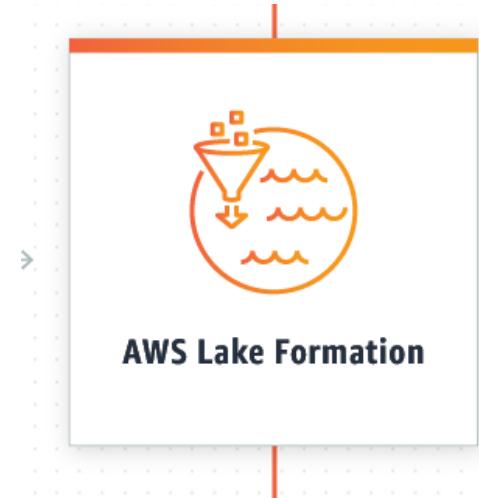
# Glue Workflow Triggers

- Triggers within workflows start jobs or crawlers
  - Or can be fired when jobs or crawlers complete
- Schedule
  - Based on a cron expression
- On demand
- EventBridge events
  - Start on single event or batch of events
  - For example, arrival of a new object in S3
  - Optional batch conditions
    - Batch size (number of events)
    - Batch window (within X seconds, default is 15 min)

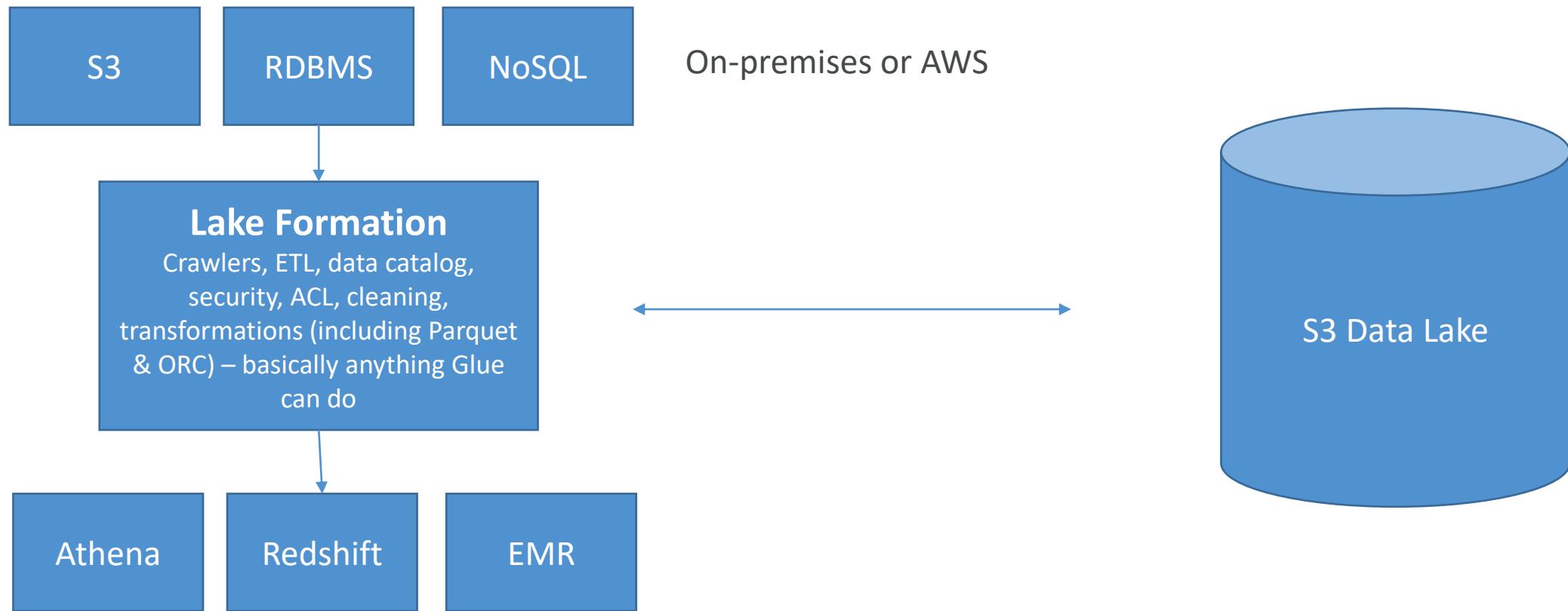


# AWS Lake Formation

- “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

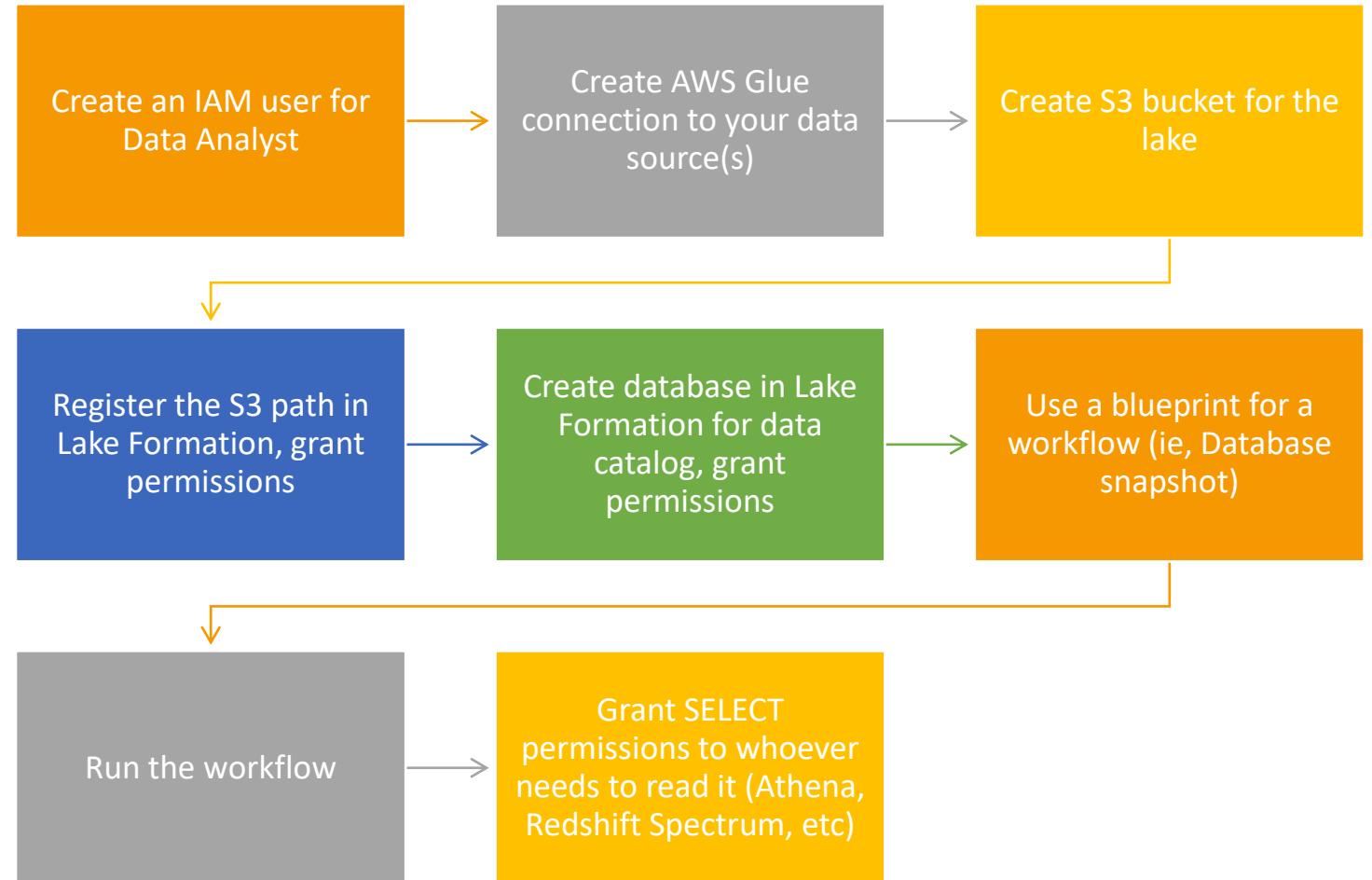


# AWS Lake Formation: Pricing

- 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

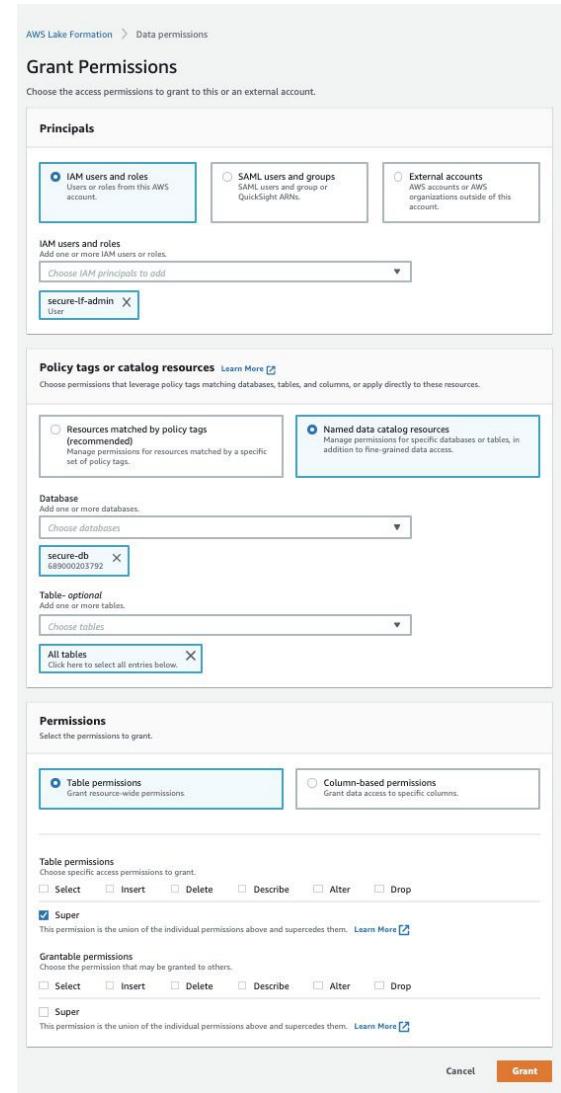
- Cross-account Lake Formation permission
  - Recipient must be set up as a data lake administrator
  - Can use AWS Resource Access Manager for accounts external to your organization
  - IAM permissions for cross-account access
- Lake Formation does not support manifests in Athena or Redshift queries
- 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



# Data Filters in Lake Formation

- Column, row, or cell-level security
- Apply when granting SELECT permission on tables
- “All columns” + row filter = row-level security
- “All rows” + specific columns = column-level security
- Specific columns + specific rows = cell-level security
- Create filters via the console (seen here) or via `CreateDataCellsFilter` API.

**Create data filter**

**Data filter name**  
Enter a name that describes this data access filter.

Name may contain letters (A-Z), numbers (0-9), hyphens (-), or under-scores (\_), and be less than 256 characters.

**Target database**  
Select the database that contains the target table.  
   
 sales  
054881201579

**Target table**  
Select the table for which the data filter will be created.  
   
 orders  
054881201579

**Column-level access**  
Choose whether this filter should have column-level restrictions.

Access to all columns  
Filter won't have any column restrictions.

Include columns  
Filter will only allow access to specific columns.

Exclude columns  
Filter will allow access to all but specific columns.

**Select columns**  
  
 customer\_name  
string

**Row filter expression**  
Enter the rest of the following query statement "SELECT \* FROM orders WHERE..."  
Please see the documentation for examples of filter expressions.

# Amazon Athena

Serverless interactive queries of S3 data

# What is Athena?

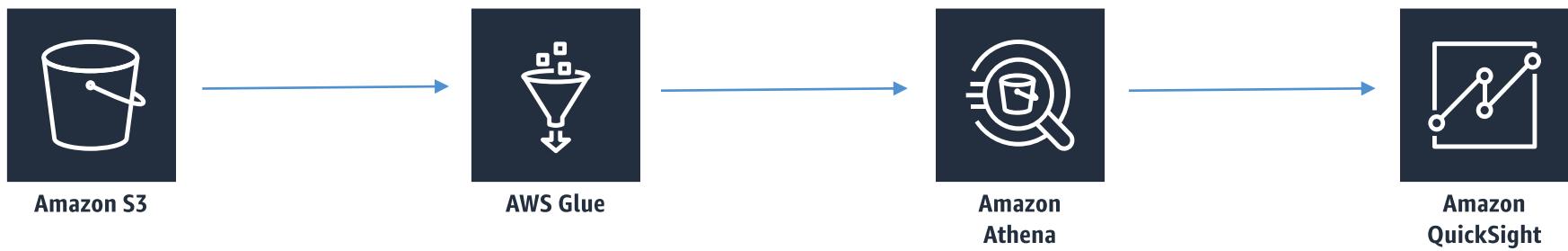
- Interactive query service for S3 (SQL)
  - No need to load data, it stays in S3
- Presto under the hood
- Serverless!
- Supports many data formats
  - CSV, TSV (human readable)
  - JSON (human readable)
  - ORC (columnar, splittable)
  - Parquet (columnar, splittable)
  - Avro (splittable)
  - Snappy, Zlib, LZO, Gzip compression
- 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 p



E 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'
```

# Athena Fine-Grained Access to AWS Glue Data Catalog

- IAM-based Database and table-level security
  - Broader than data filters in Lake Formation
  - Cannot restrict to specific table versions
- At a minimum you must have a policy that grants access to your database and the Glue Data Catalog in each region.

```
{  
  "Sid": "DatabasePermissions",  
  "Effect": "Allow",  
  "Action": [  
    "glue:GetDatabase",  
    "glue:GetDatabases",  
    "glue>CreateDatabase"  
  ],  
  "Resource": [  
    "arn:aws:glue:us-east-1:123456789012:catalog",  
    "arn:aws:glue:us-east-1:123456789012:database/default"  
  ]  
}
```

# Athena Fine-Grained Access to AWS Glue Data Catalog

- You might have policies to restrict access to:
  - ALTER or CREATE DATABASE
  - CREATE TABLE
  - DROP DATABASE or DROP TABLE
  - MSCK REPAIR TABLE
  - SHOW DATABASES or SHOW TABLES
- Just need to map these operations to their IAM actions
- Example: DROP TABLE
- More examples at  
<https://docs.aws.amazon.com/athena/latest/ug/fine-grained-access-to-glue-resources.html>

```
{  
    "Effect": "Allow",  
    "Action": [  
        "glue:GetDatabase",  
        "glue:GetTable",  
        "glue:DeleteTable",  
        "glue:GetPartitions",  
        "glue:GetPartition",  
        "glue:DeletePartition"  
    ],  
    "Resource": [  
        "arn:aws:glue:us-east-1:123456789012:catalog",  
        "arn:aws:glue:us-east-  
1:123456789012:database/example_db",  
        "arn:aws:glue:us-east-  
1:123456789012:table/example_db/test"  
    ]  
}
```

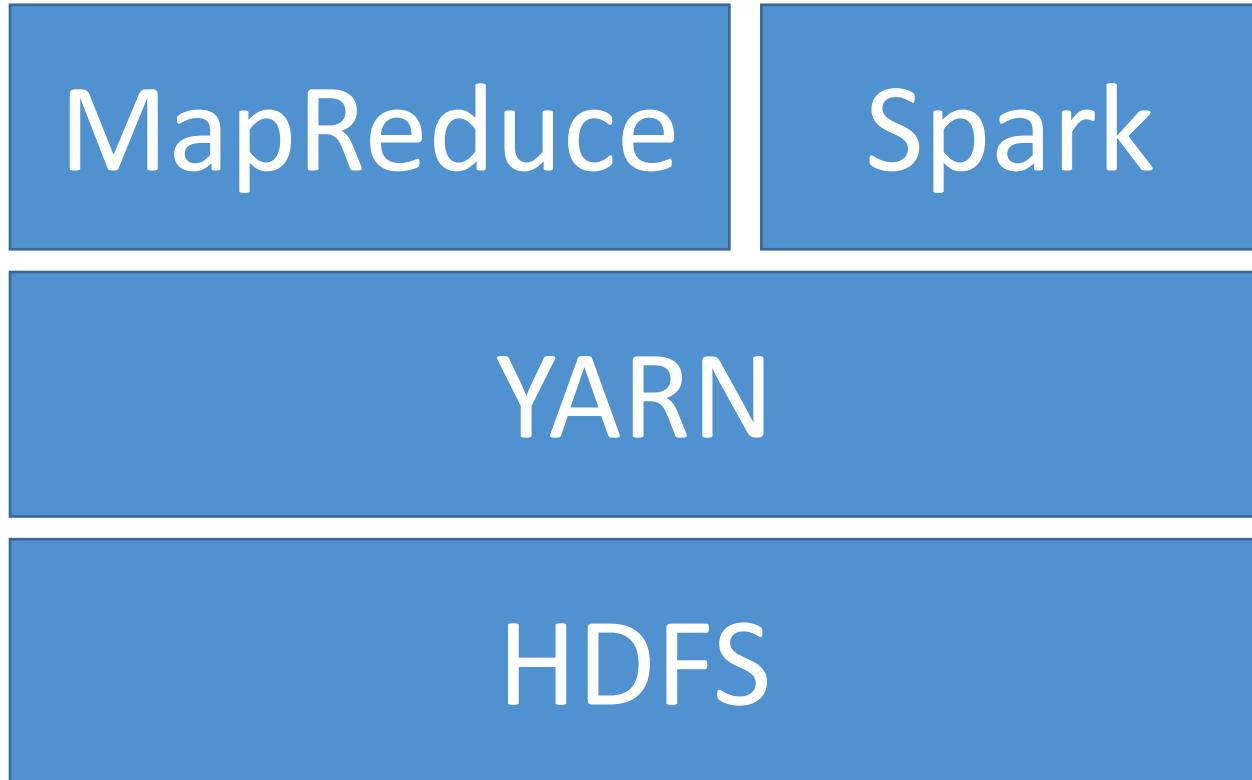
# CREATE TABLE AS SELECT

- You might see this in the context of Amazon Athena
  - But it exists in other databases as well.
- Creates a new table from query results (CTAS)
- Can be used to create a new table that's a subset of another
- Can ALSO be used to convert data into a new underlying format
  - So this is a trick to get Athena to convert data stored in S3

```
CREATE TABLE new_table  
WITH (  
    format = 'Parquet',  
    write_compression = 'SNAPPY')  
AS SELECT *  
FROM old_table;
```

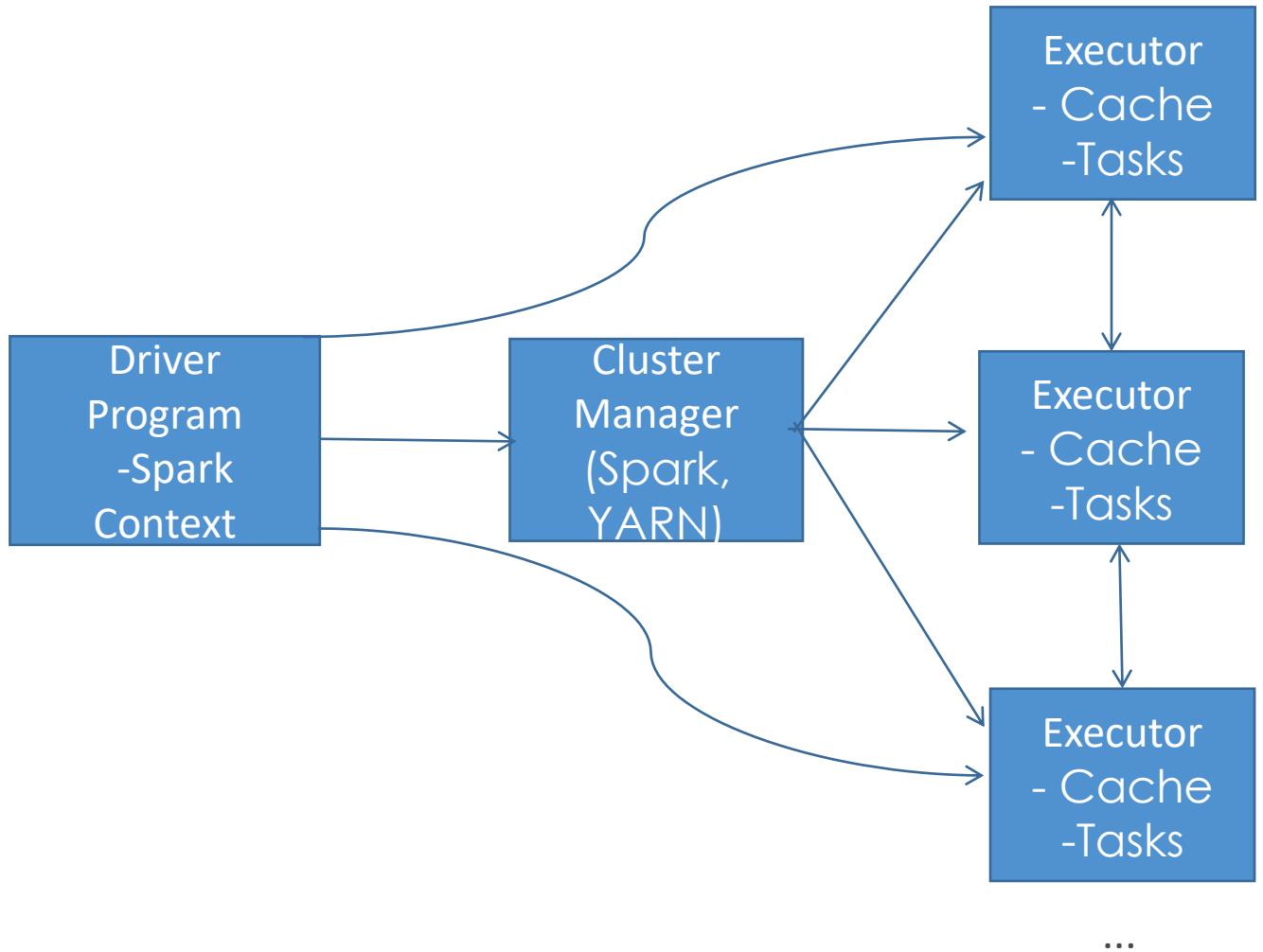
```
CREATE TABLE my_orc_ctas_table  
WITH (  
    external_location =  
    's3://my_athena_results/my_orc_stas_table/',  
    format = 'ORC')  
AS SELECT *  
FROM old_table;
```

# 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

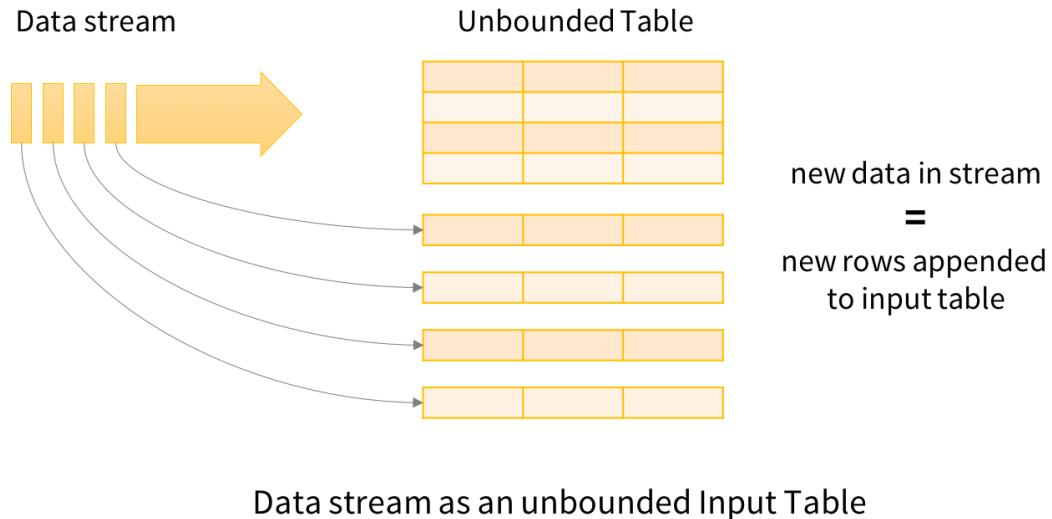
Graph Processing  
ETL, analysis, iterative graph  
computation  
No longer widely used

## SPARK CORE

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

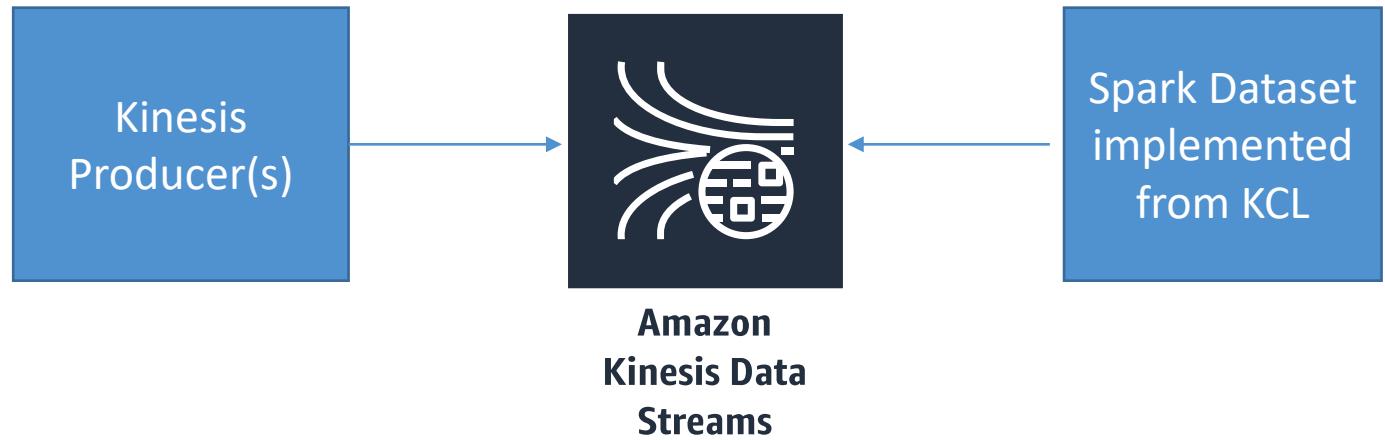
# Spark Structured Streaming

## A constantly growing DataSet



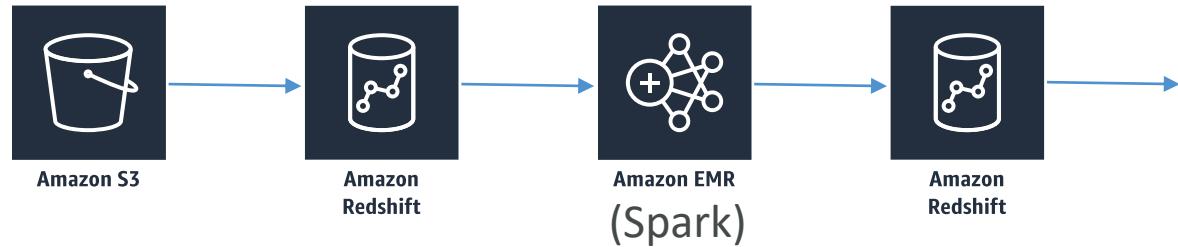
```
val inputDF = spark.readStream.json("s3://logs")
inputDF.groupBy($"action", window($"time", "1 hour")).count()
    .writeStream.format("jdbc").start("jdbc:mysql//...")
```

# Spark Streaming + Kinesis



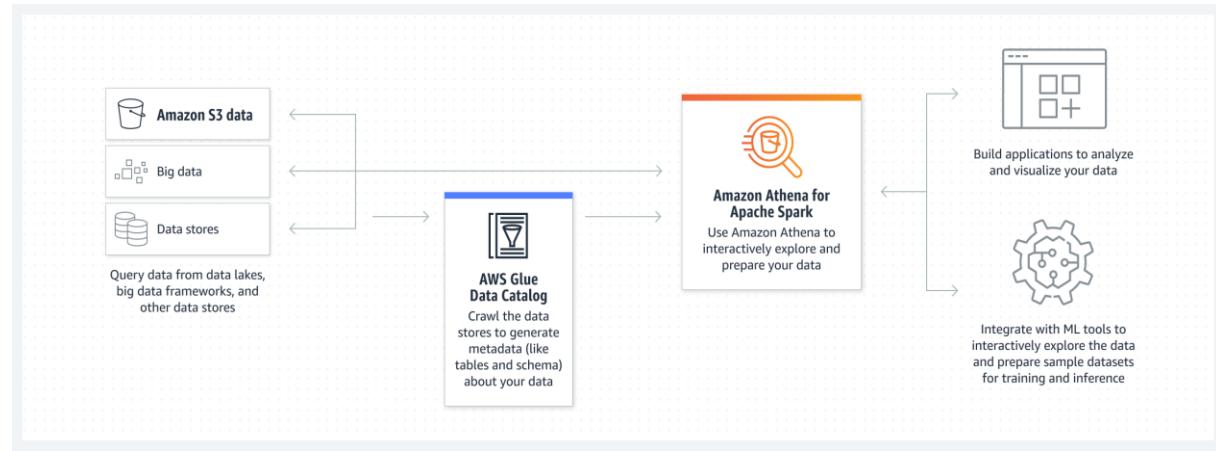
# Spark + Redshift

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



# Amazon Athena for Apache Spark

- Can run Jupyter notebooks with Spark within Athena console
  - Notebooks may be encrypted automatically or with KMS
- Totally serverless
- Selectable as an alternate analytics engine (vs. Athena SQL)
- Uses Firecracker for quickly spinning up Spark resources
- Programmatic API / CLI access as well
  - create-work-group, create-notebook, start-session, start-calculation-execution
- Can adjust DPU's for coordinator and executor sizes
- Pricing based on compute usage and DPU per hour



### Analytics engine

Choose the type of engine

**Athena SQL**  
Athena SQL engine for interactive queries.

**Apache Spark**  
Highly-scalable serverless Spark engine.

Turn on example notebook  
Create an example notebook in your workgroup, attach it to your workgroup and attach additional permissions to your service role.

[View additional permissions details](#)

# EMR

## Elastic MapReduce

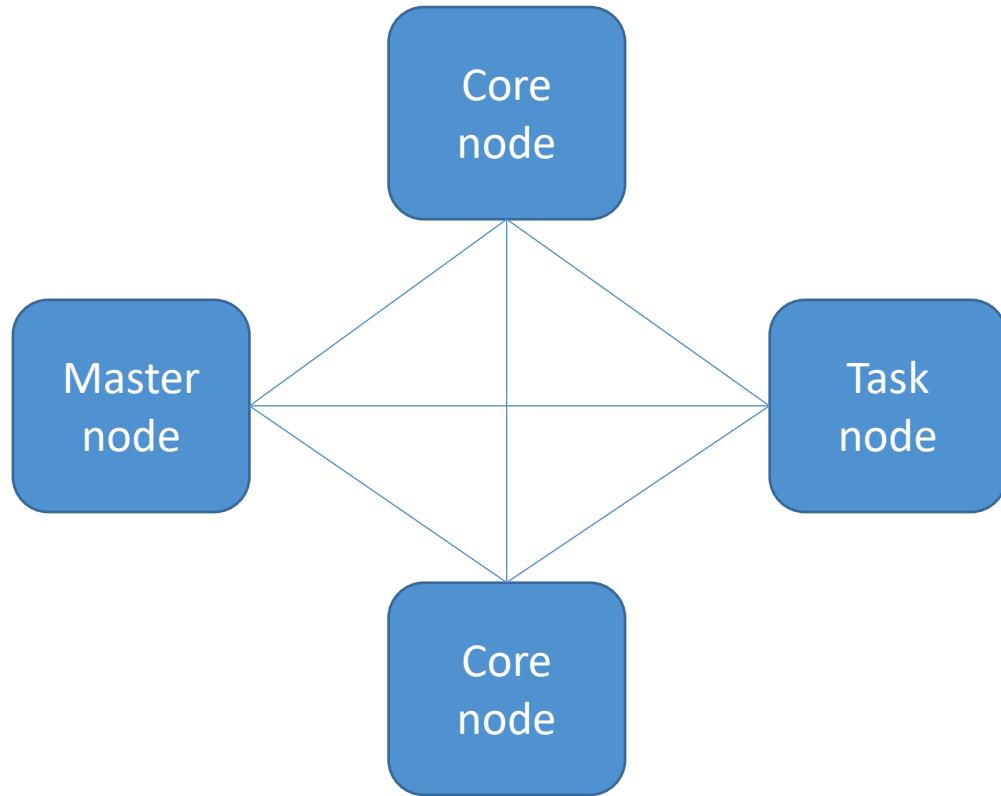
# What is EMR?

- 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

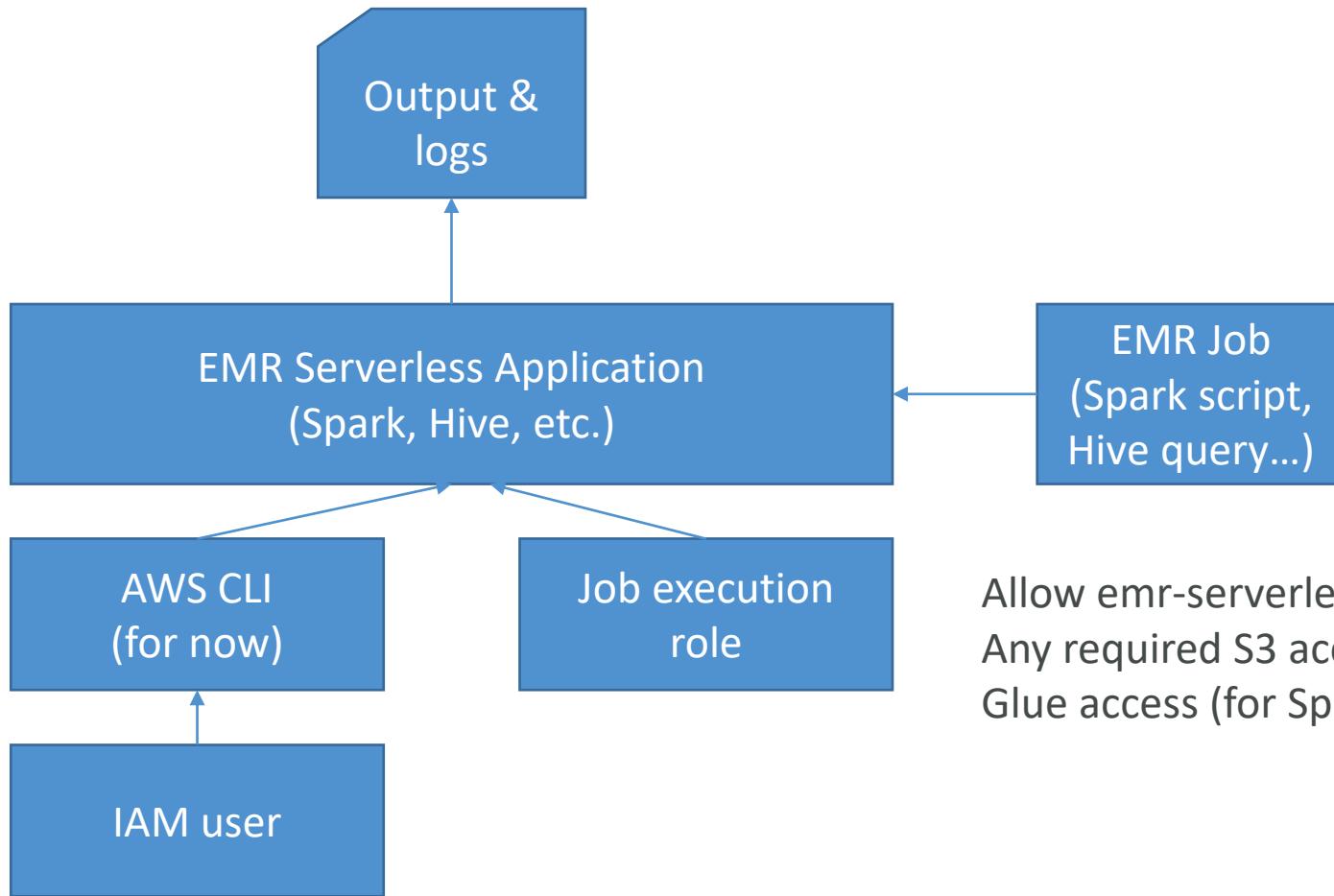
- EMR Automatic Scaling
  - The old way of doing it
  - Custom scaling rules based on CloudWatch metrics
  - Supports instance groups only
- EMR Managed Scaling
  - Introduced in 2020
  - Support instance groups and instance fleets
  - Scales spot, on-demand, and instances in a Savings Plan within the 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

- 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

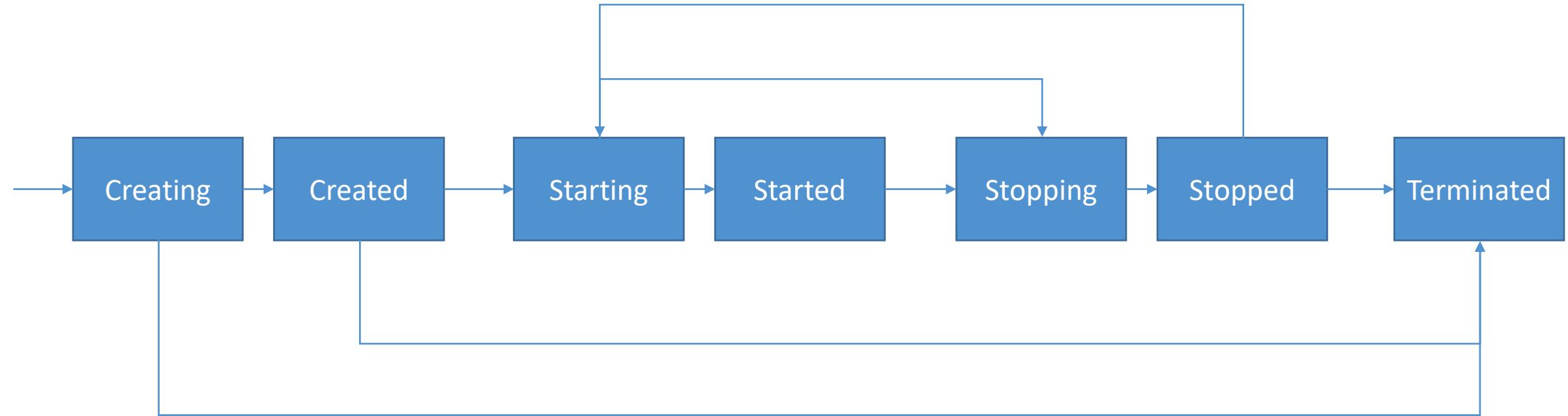


```

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"
}'
```

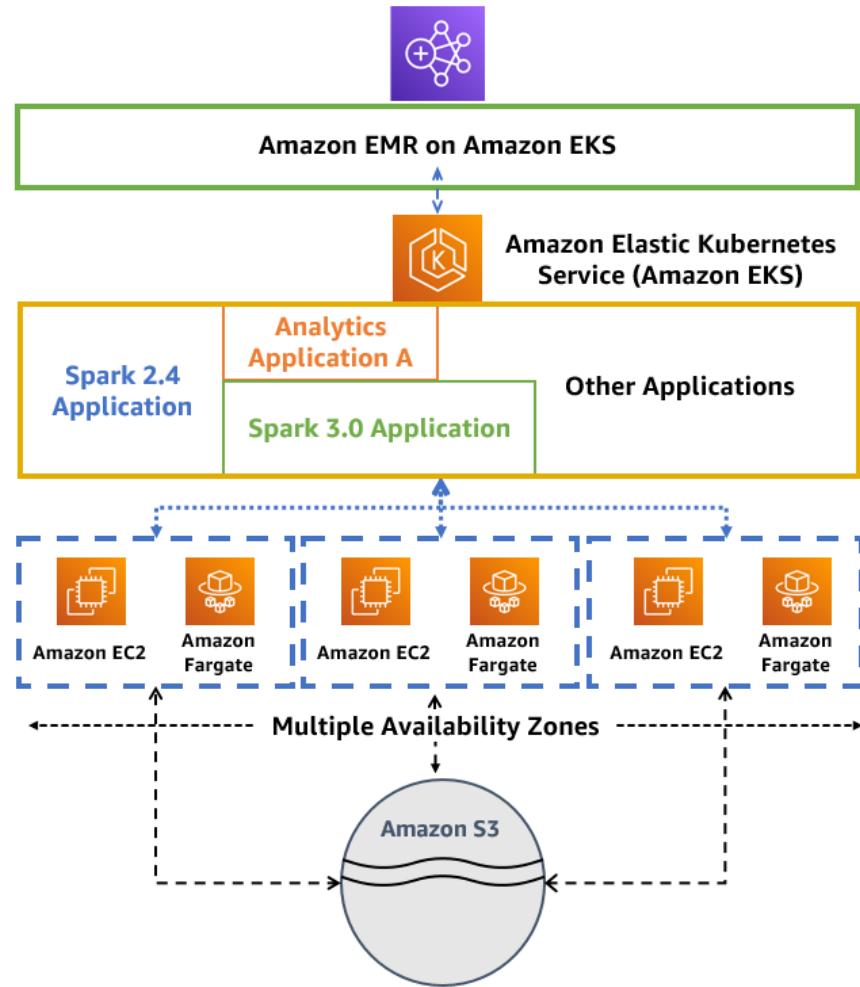
# EMR Serverless Security

- 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

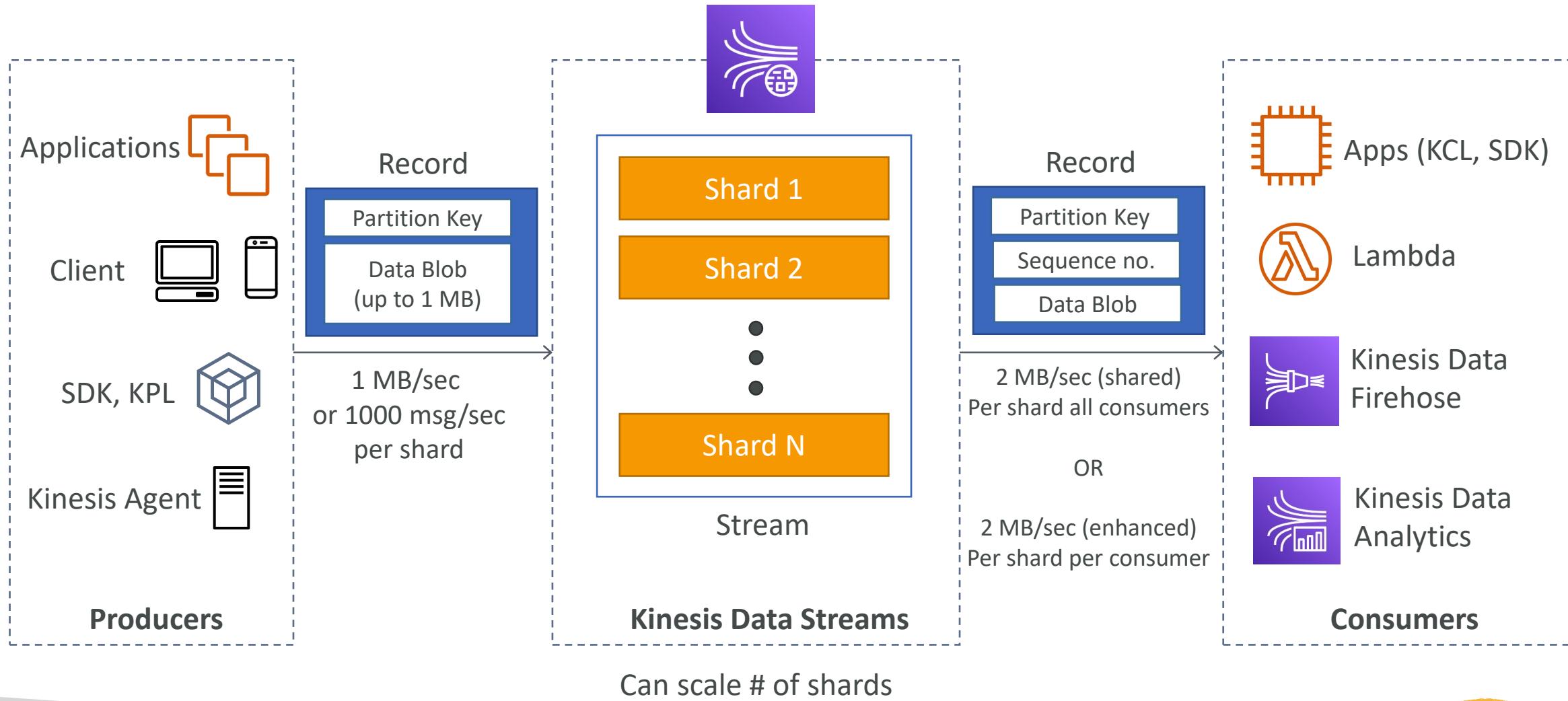


# EMR on EKS

- Allows submitting Spark job on Elastic Kubernetes Service without provisioning clusters
- Fully managed
- Share resources between Spark and other apps on Kubernetes



# 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:**

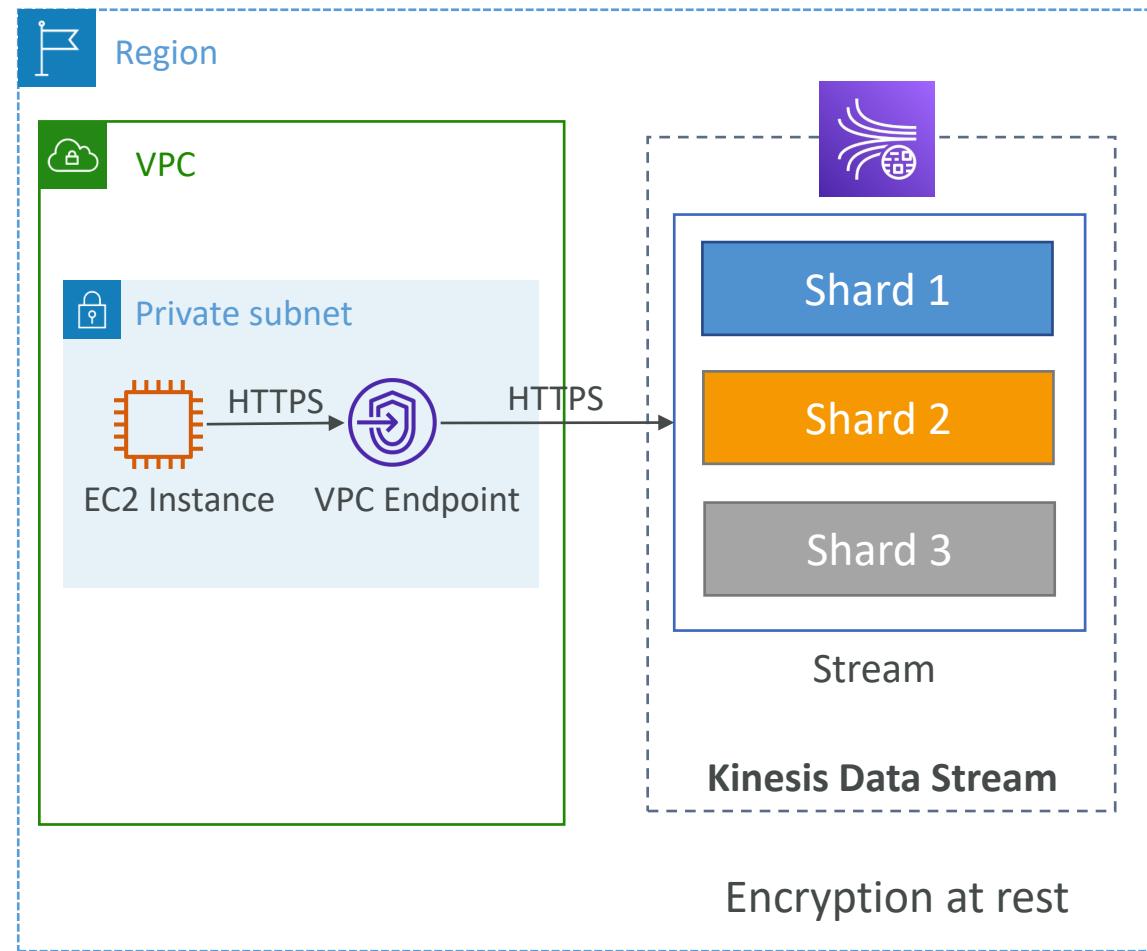
- You choose the number of shards provisioned, scale manually or using API
- Each shard gets 1MB/s in (or 1000 records per second)
- Each shard gets 2MB/s out (classic or enhanced fan-out consumer)
- You pay per shard provisioned per hour

- **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

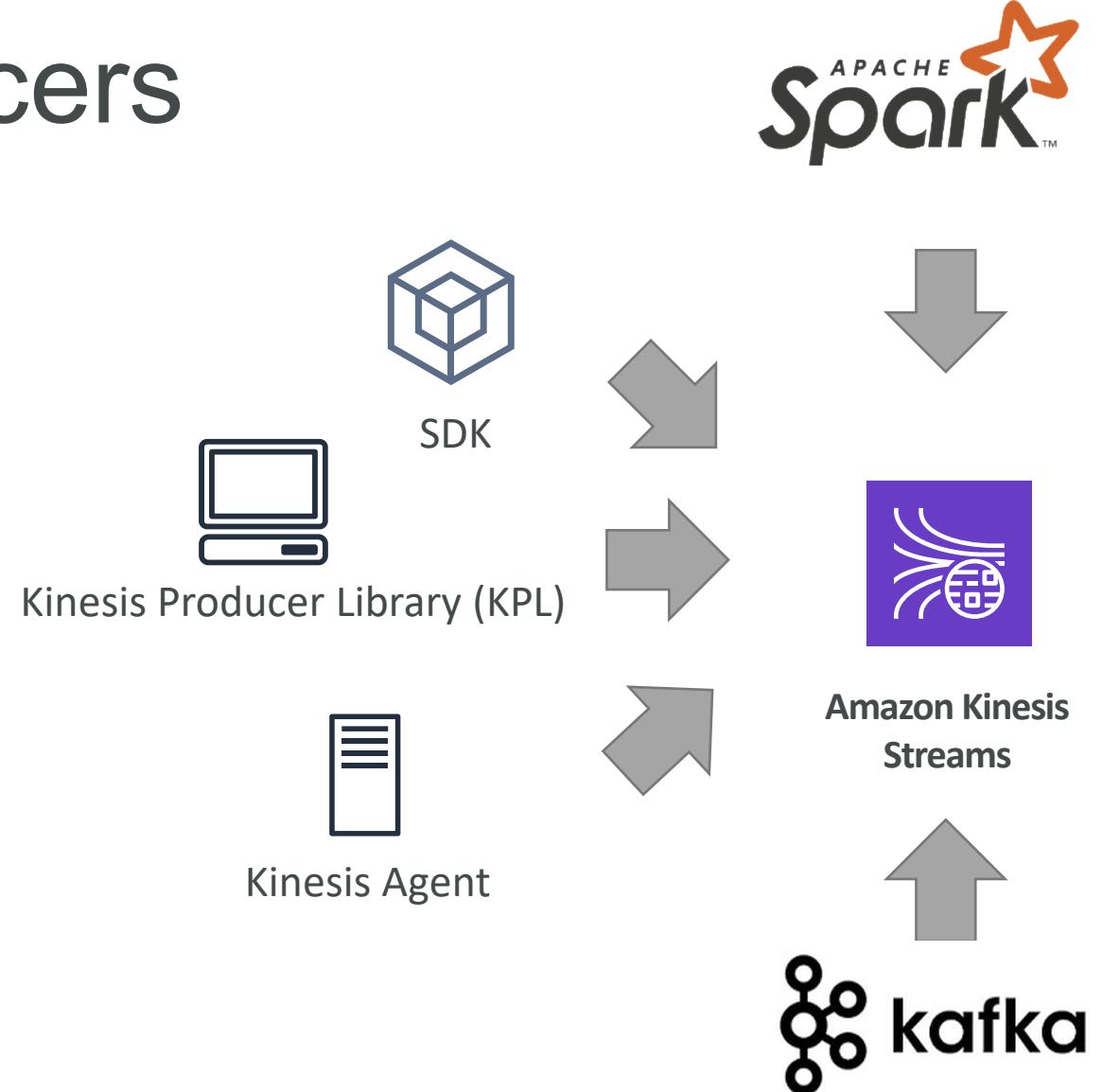
# Kinesis Data Streams Security

- 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

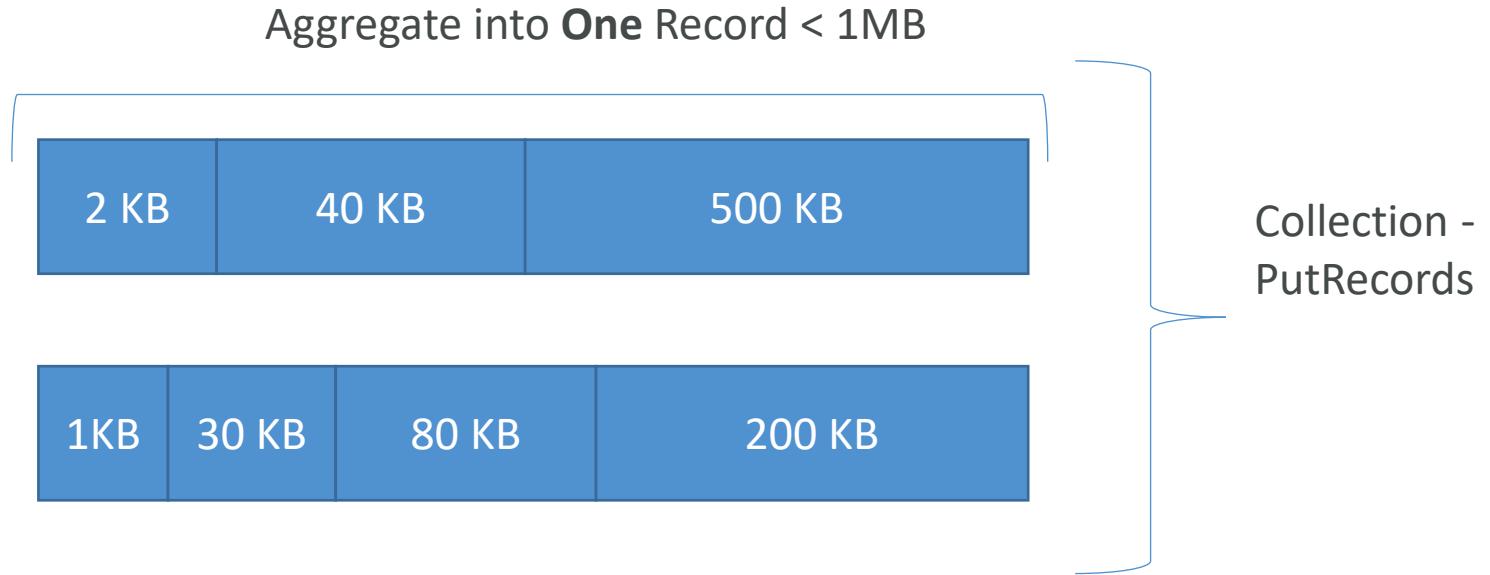
- ProvisionedThroughputExceeded Exceptions
  - Happens when sending more data (exceeding MB/s or TPS for any shard)
  - Make sure you don't have a hot shard (such as your partition key is bad and too much data goes to that partition)
- Solution:
  - Retries with backoff
  - Increase shards (scaling)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

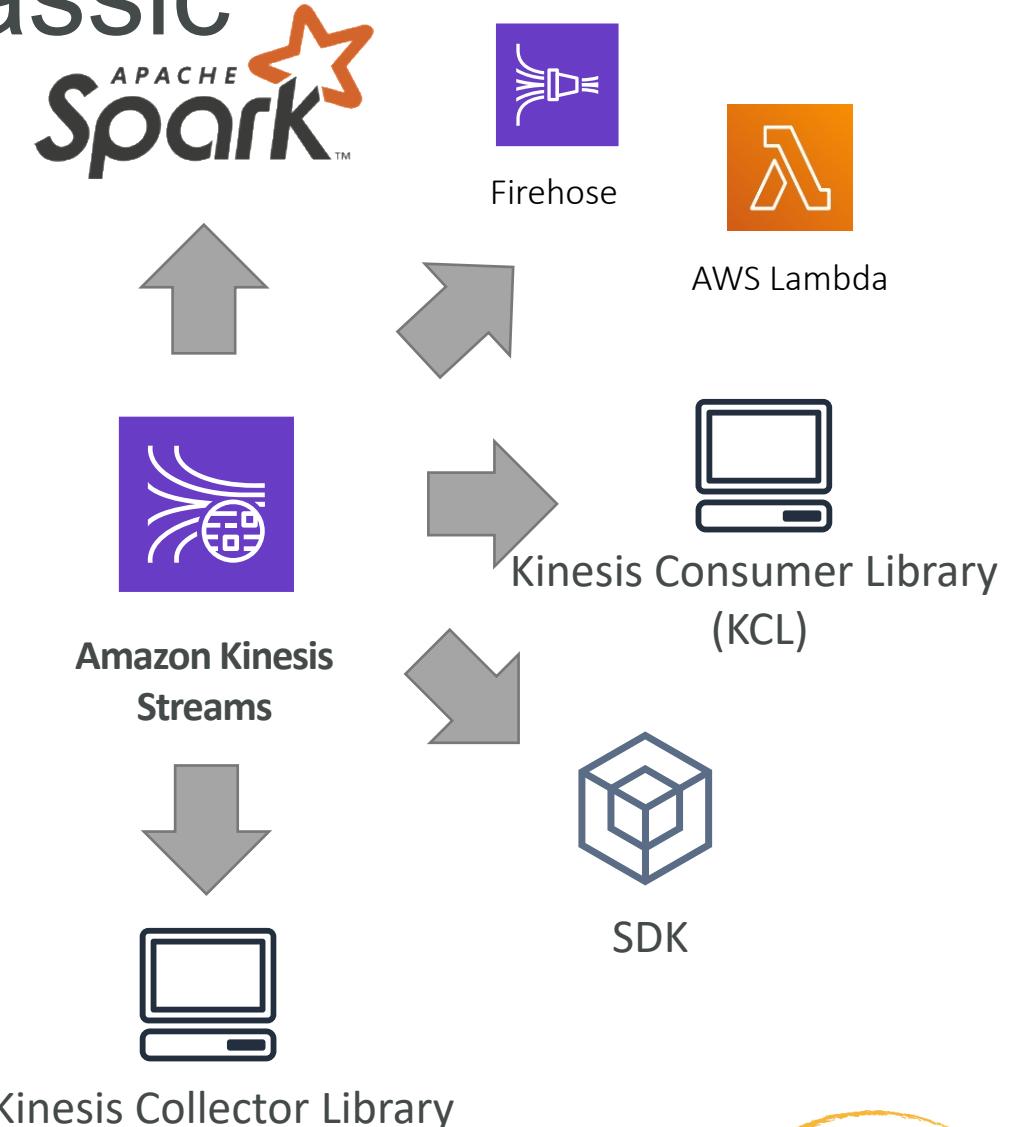


# Kinesis Agent

- 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

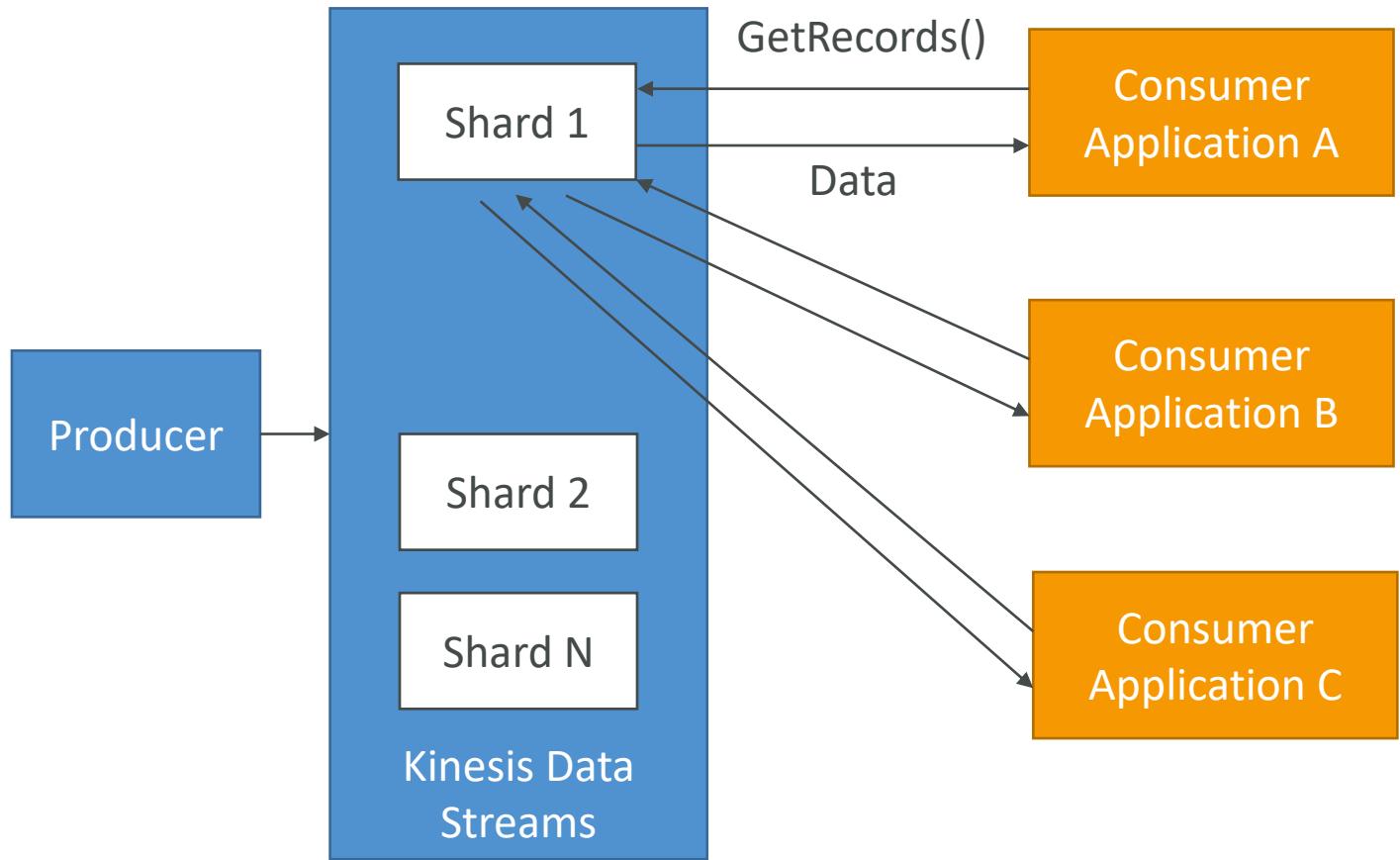
# Kinesis Consumers - Classic

- 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)



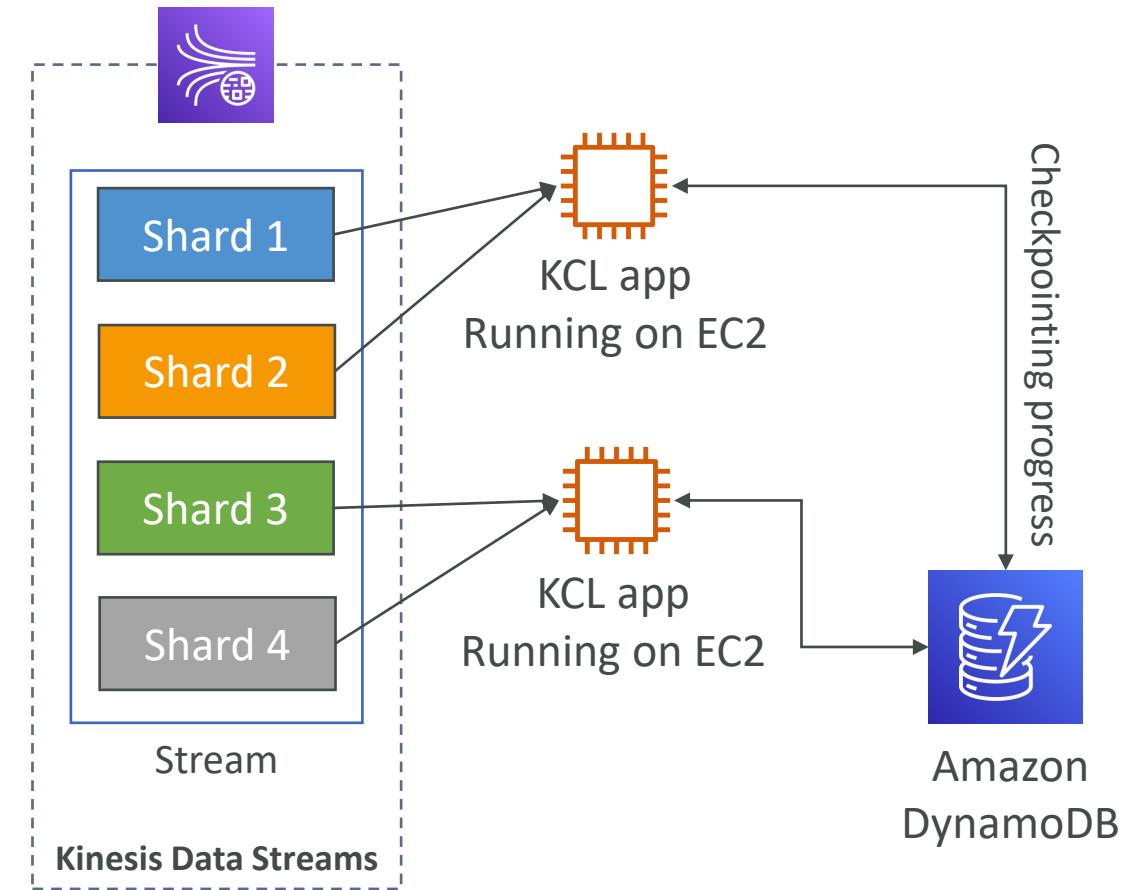
# Kinesis Consumer SDK - GetRecords

- **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



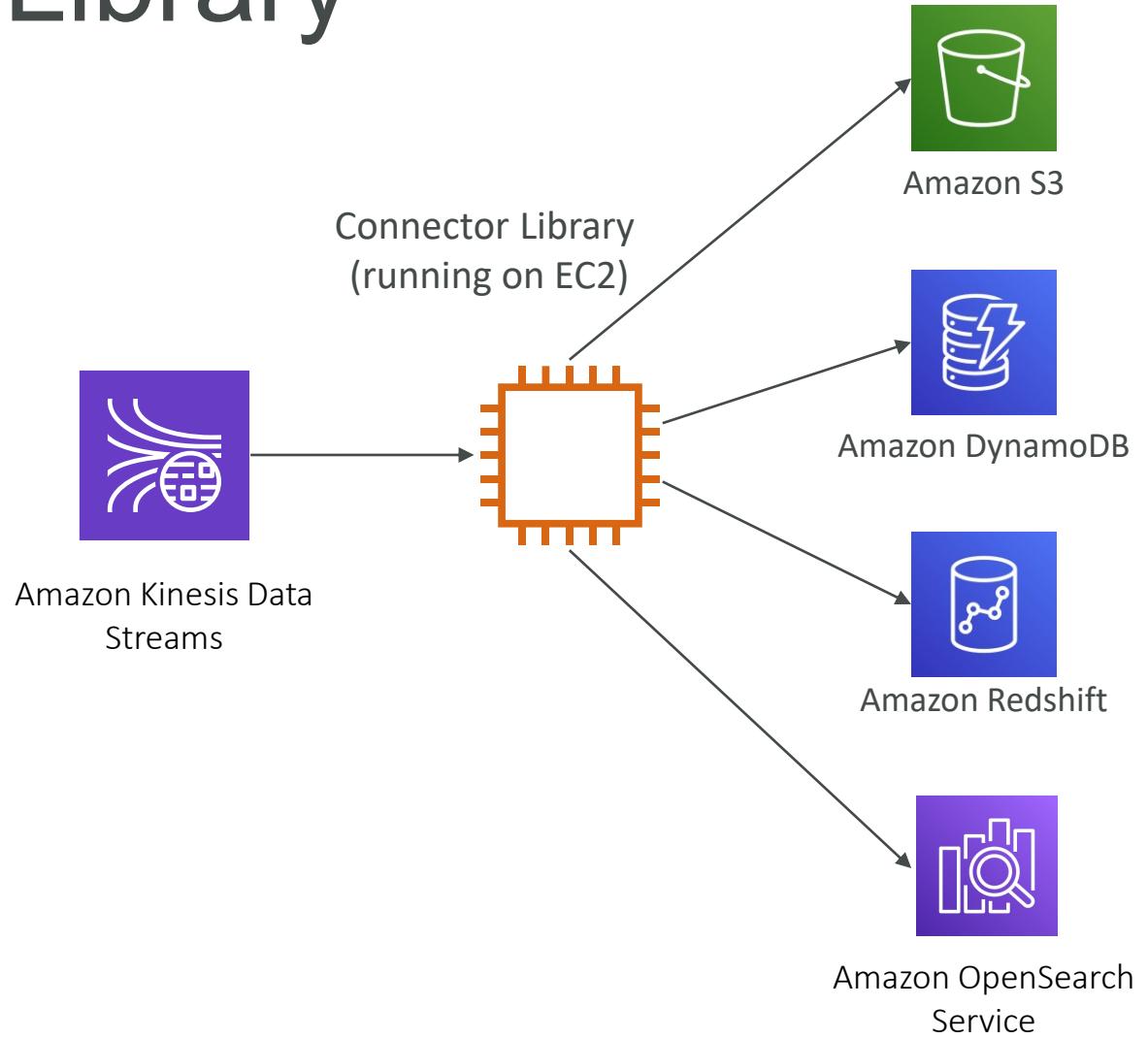
# Kinesis Client Library (KCL)

- 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
  - OpenSearch
- Kinesis Firehose replaces the Connector Library for a few of these targets, Lambda for the others

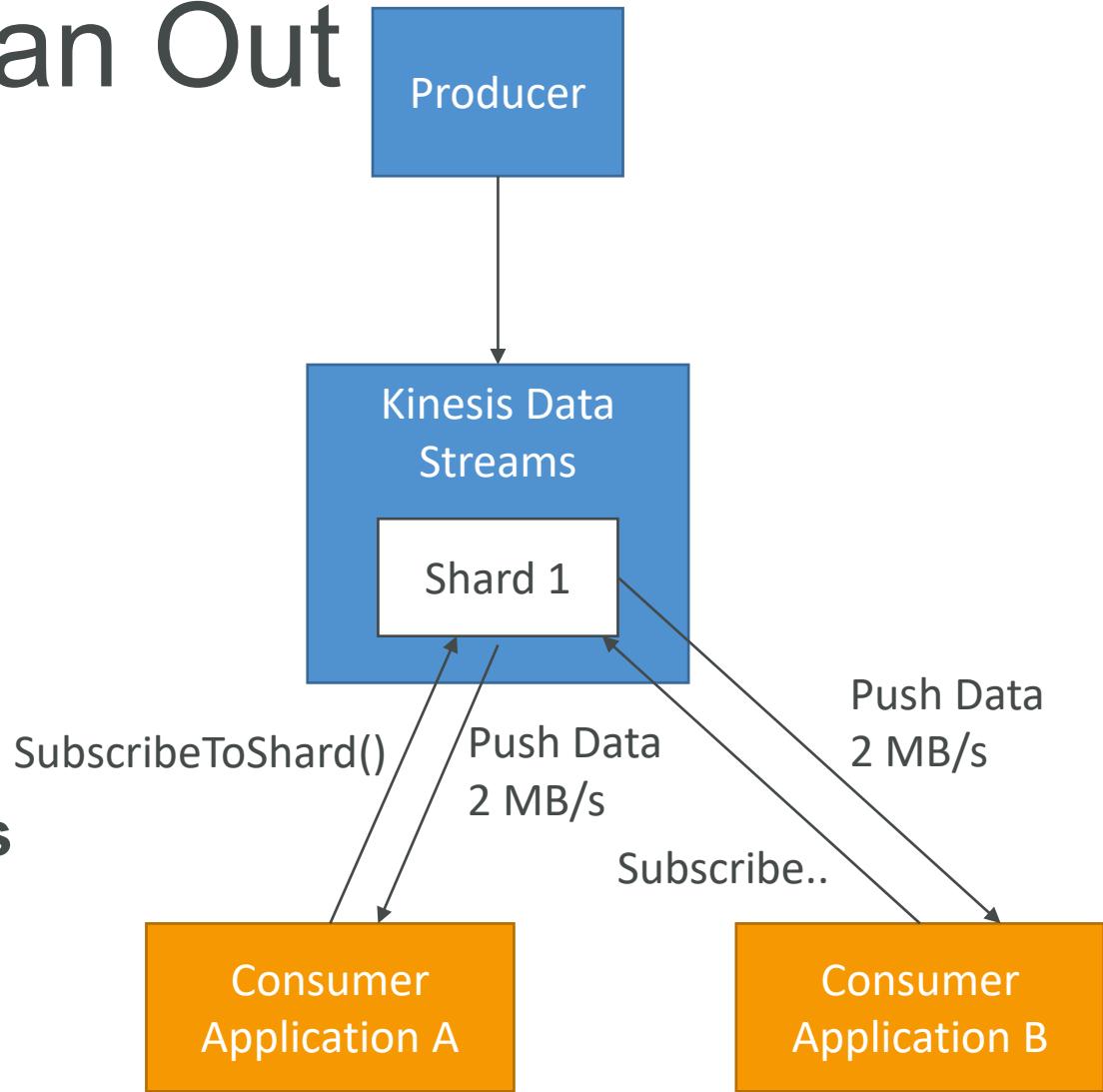


# AWS Lambda sourcing from Kinesis

- 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
  - OpenSearch
  - 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)

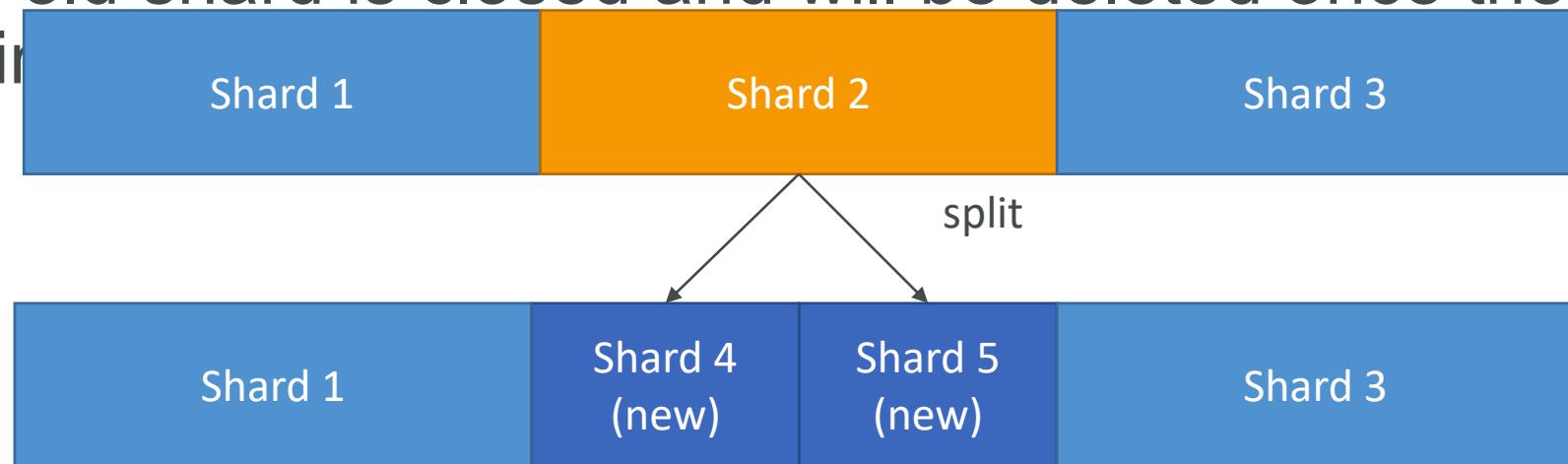


# Enhanced Fan-Out vs Standard Consumers

- 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

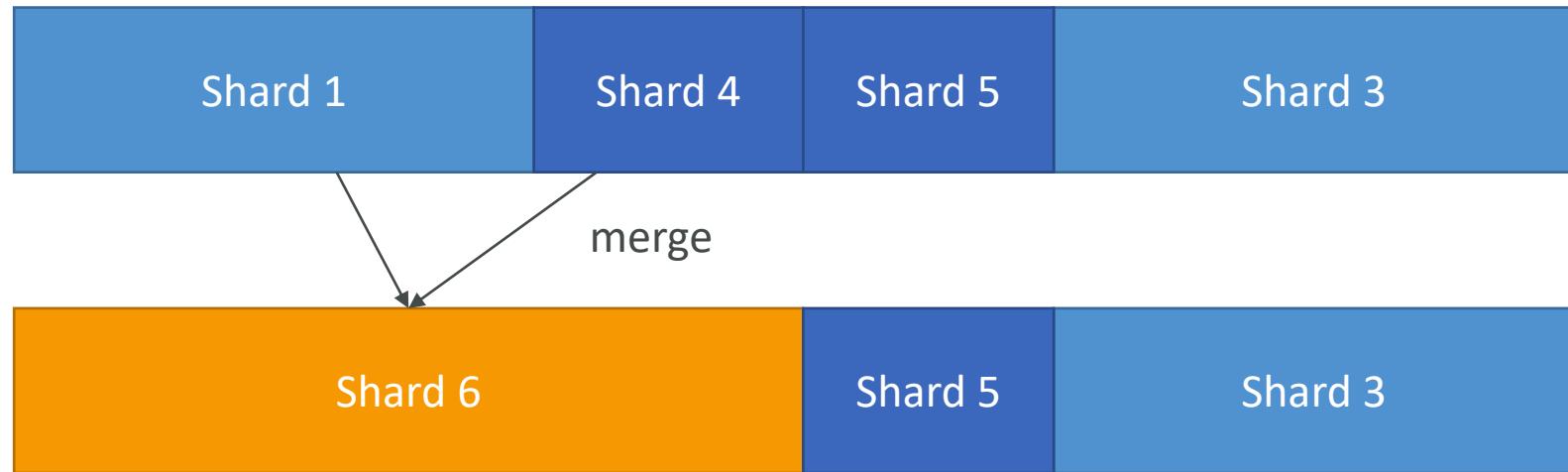
# Kinesis Operations – Adding Shards

- 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



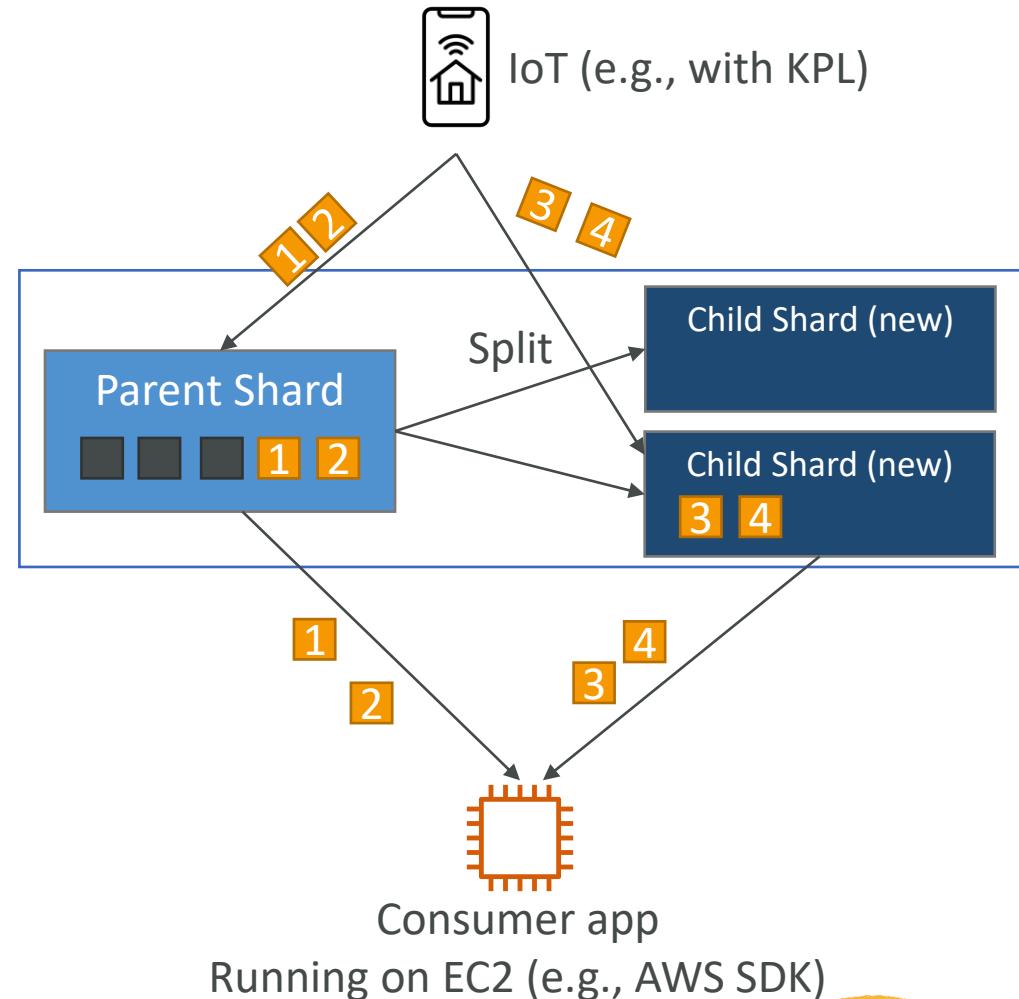
# Kinesis Operations – Merging Shards

- 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



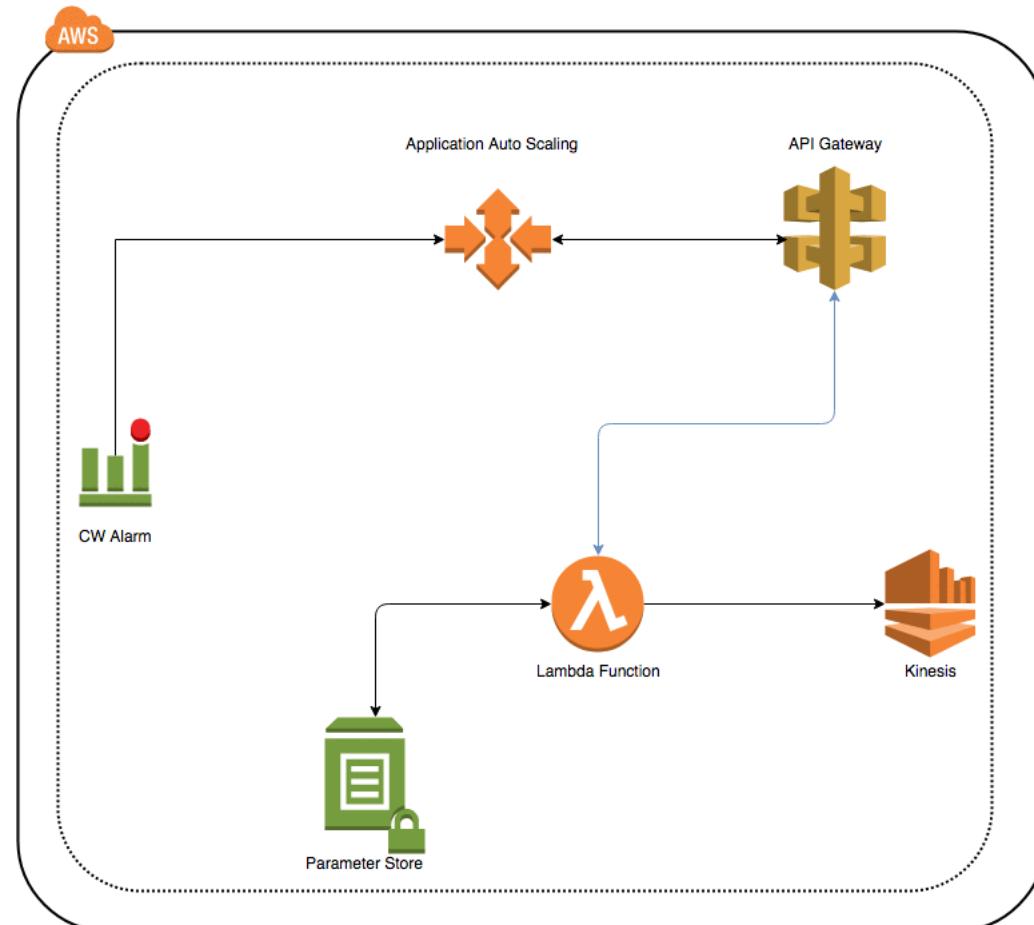
# Out-of-order records after resharding

- 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

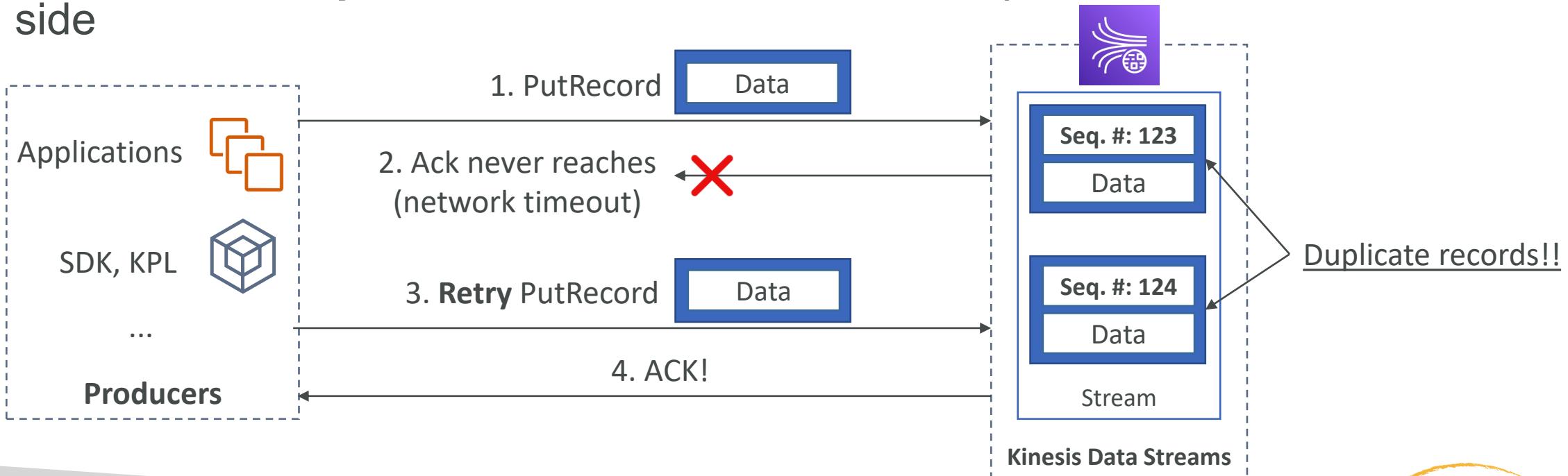
# Kinesis Data Streams – Handling

## 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

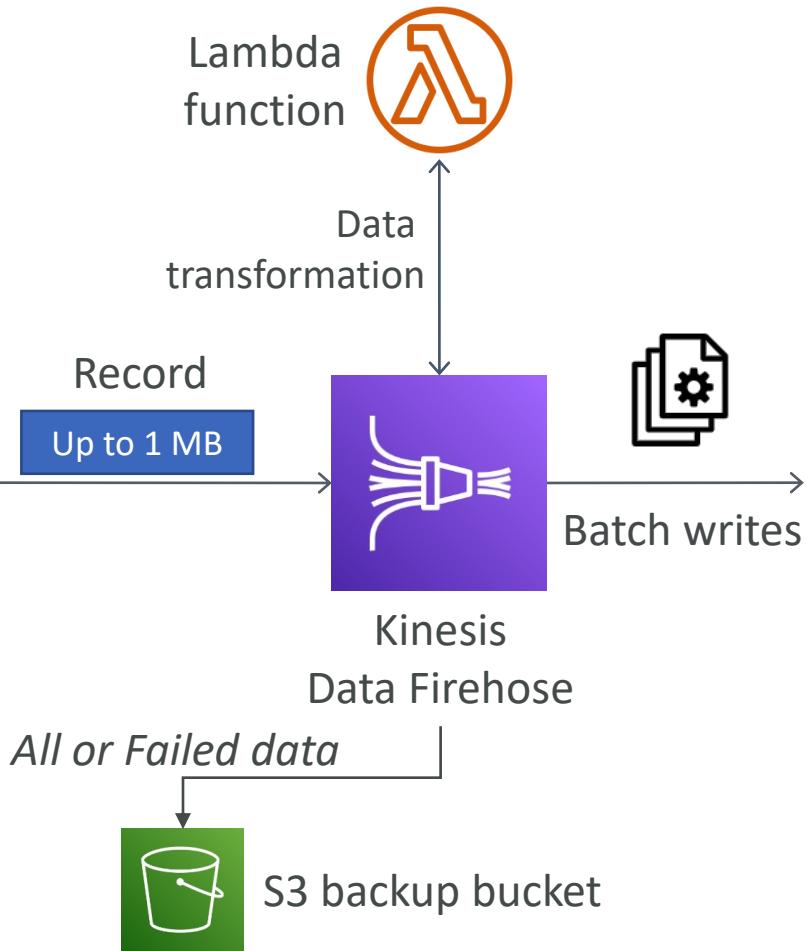
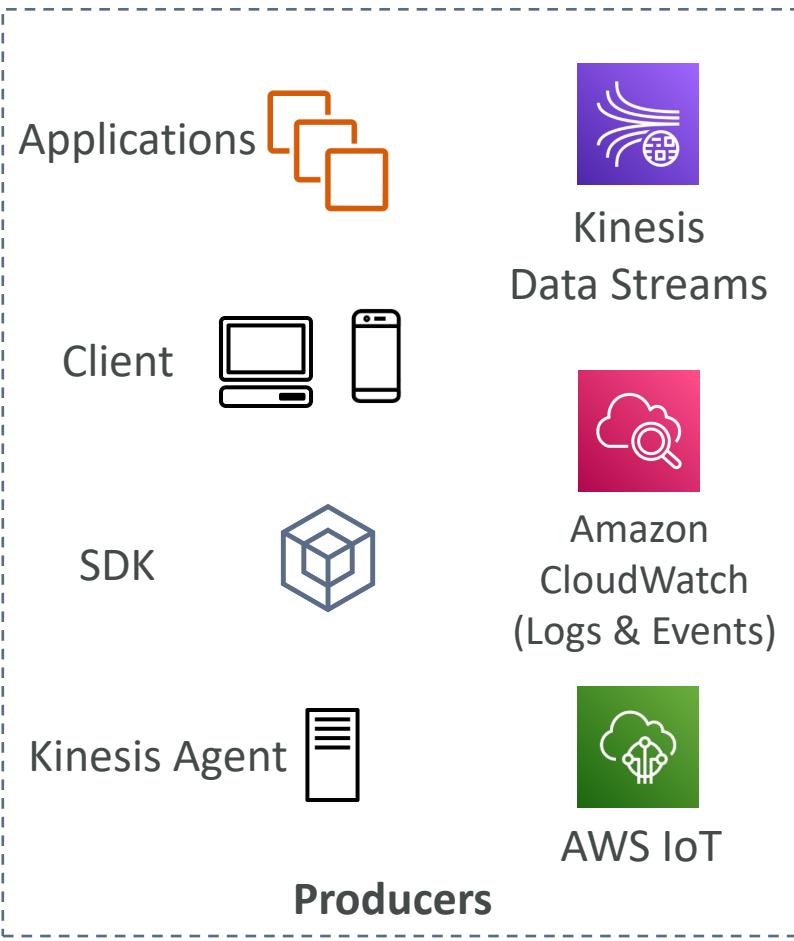
## For Consumers

- 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>

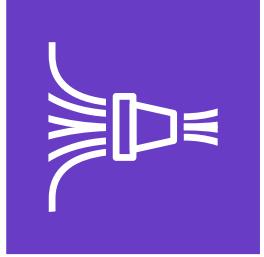
# Kinesis Security

- 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 Firehose

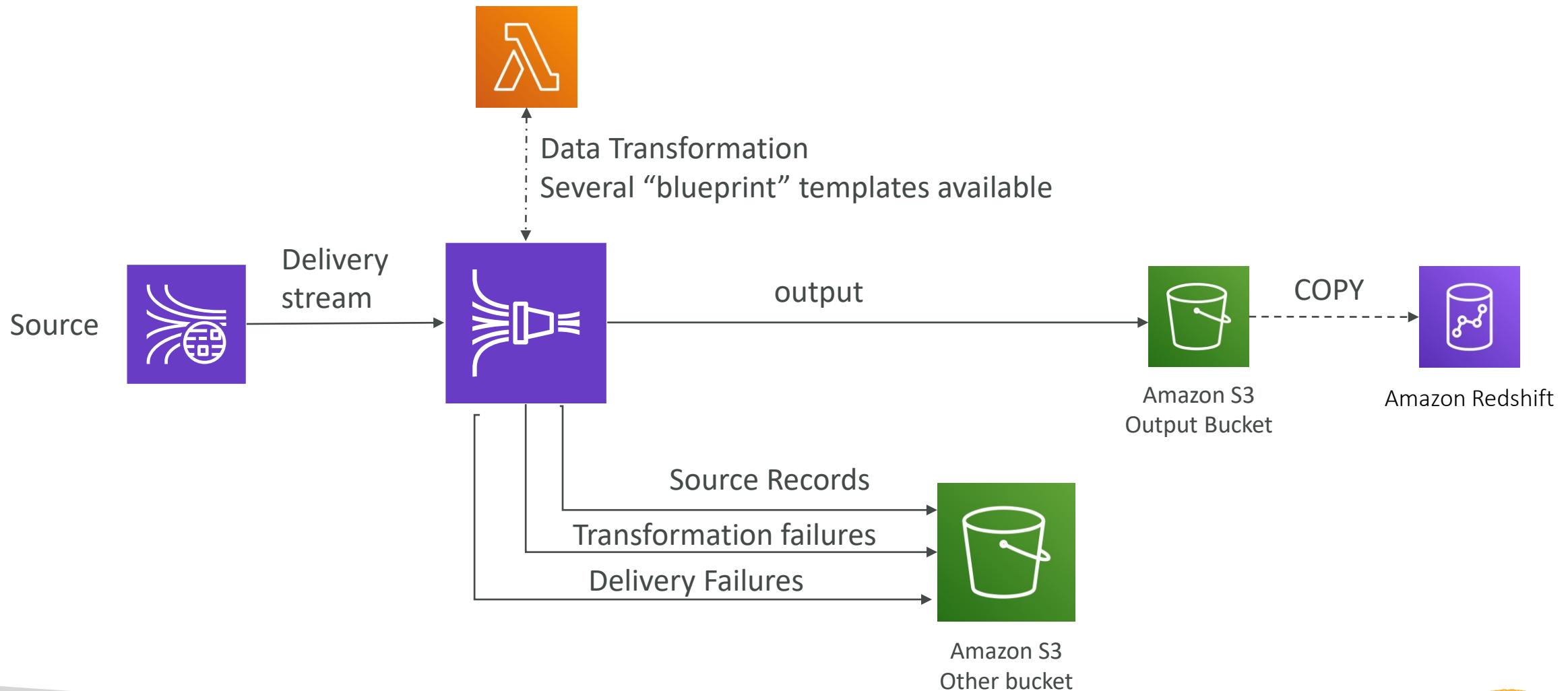


# AWS Kinesis Data Firehose



- Fully Managed Service, no administration
- **Near Real Time (Buffer based on time and size, optionally can be disabled)**
- Load data into Redshift / Amazon S3 / OpenSearch / 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 OpenSearch (for example)
- Firehose
  - Fully managed, send to S3, Splunk, Redshift, OpenSearch
  - Serverless data transformations with Lambda
  - **Near** real time
  - Automated Scaling
  - No data storage

# Troubleshooting Kinesis Data Stream Producers: Performance

- Writing is too slow
  - Service limits may be exceeded. Check for throughput exceptions, see what operations are being throttled. Different calls have different limits.
  - There are shard-level limits for writes and reads
  - Other operations (ie, CreateStream, ListStreams, DescribeStreams) have stream-level limits of 5-20 calls per second
  - Select partition key to evenly distribute puts across shards
- Large producers
  - Batch things up. Use Kinesis Producer Library, PutRecords with multi-records, or aggregate records into larger files.
- Small producers (i.e. apps)
  - Use PutRecords or Kinesis Recorder in the AWS Mobile SDKs



# Other Kinesis Data Stream Producer Issues

- Stream returns a 500 or 503 error
  - This indicates an AmazonKinesisException error rate above 1%
  - Implement a retry mechanism
- Connection errors from Flink to Kinesis
  - Network issue or lack of resources in Flink's environment
  - Could be a VPC misconfiguration
- Timeout errors from Flink to Kinesis
  - Adjust RequestTimeout and #setQueueLimit on FlinkKinesisProducer
- Throttling errors
  - Check for hot shards with enhanced monitoring (shard-level)
  - Check logs for “micro spikes” or obscure metrics breaching limits
  - Try a random partition key or improve the key’s distribution
  - Use exponential backoff
  - Rate-limit



# Troubleshooting Kinesis Data Stream Consumers

- Records get skipped with Kinesis Client Library
  - Check for unhandled exceptions on processRecords
- Records in same shard are processed by more than one processor
  - May be due to failover on the record processor workers
  - Adjust failover time
  - Handle shutdown methods with reason “ZOMBIE”
- Reading is too slow
  - Increase number of shards
  - maxRecords per call is too low
  - Your code is too slow (test an empty processor vs. yours)
- GetRecords returning empty results
  - This is normal, just keep calling GetRecords
- Shard Iterator expires unexpectedly
  - May need more write capacity on the shard table in DynamoDB
- Record processing falling behind
  - Increase retention period while troubleshooting
  - Usually insufficient resources
  - Monitor with `GetRecords.IteratorAgeMilliseconds` and `MillisBehindLatest`



# Troubleshooting Kinesis Data Stream Consumers

- Lambda function can't get invoked
  - Permissions issue on execution role
  - Function is timing out (check max execution time)
  - Breaching concurrency limits
  - Monitor IteratorAge metric; it will increase if this is a problem
- ReadProvisionedThroughputExceeded exception
  - Throttling
  - Reshard your stream
  - Reduce size of GetRecords requests
  - Use enhanced fan-out
  - Use retries and exponential backoff



# Troubleshooting Kinesis Data Stream Consumers

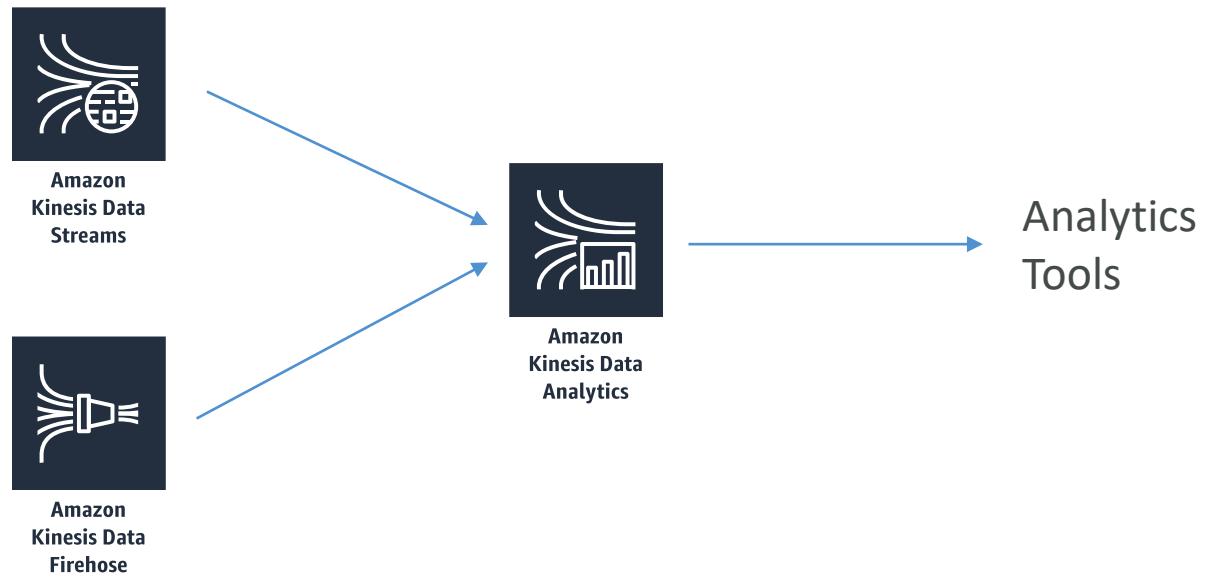
- High latency
  - Monitor with GetRecords.Latency and IteratorAge
  - Increase shards
  - Increase retention period
  - Check CPU and memory utilization (may need more memory)
- 500 errors
  - Same as producers – indicates a high error rate (>1%)
  - Implement a retry mechanism
- Blocked or stuck KCL application
  - Optimize your processRecords method
  - Increase maxLeasesPerWorker
  - Enable KCL debug logs



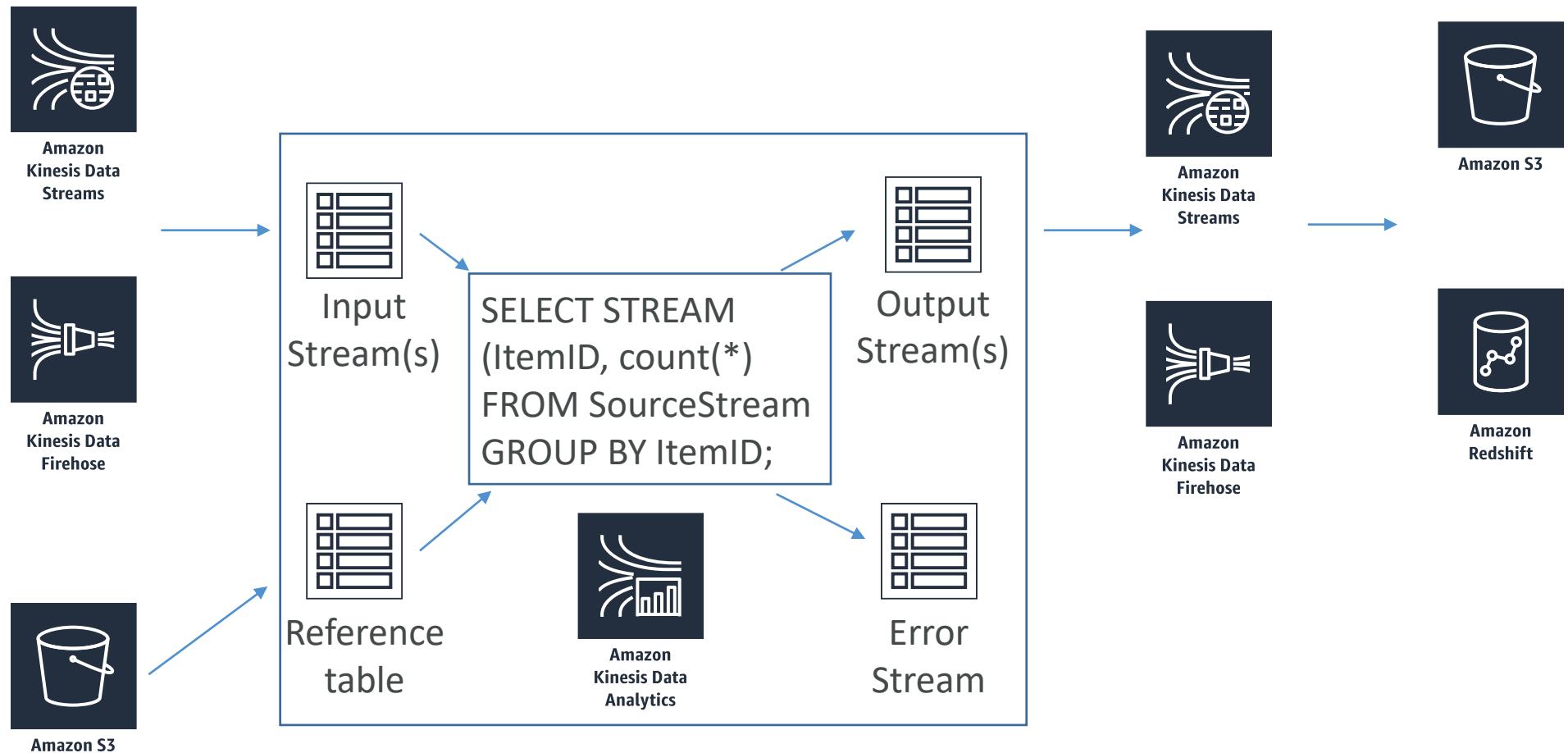
# Kinesis Data Analytics / Managed Service for Apache Flink

Querying streams of data

# Amazon Kinesis Data Analytics for SQL Applications

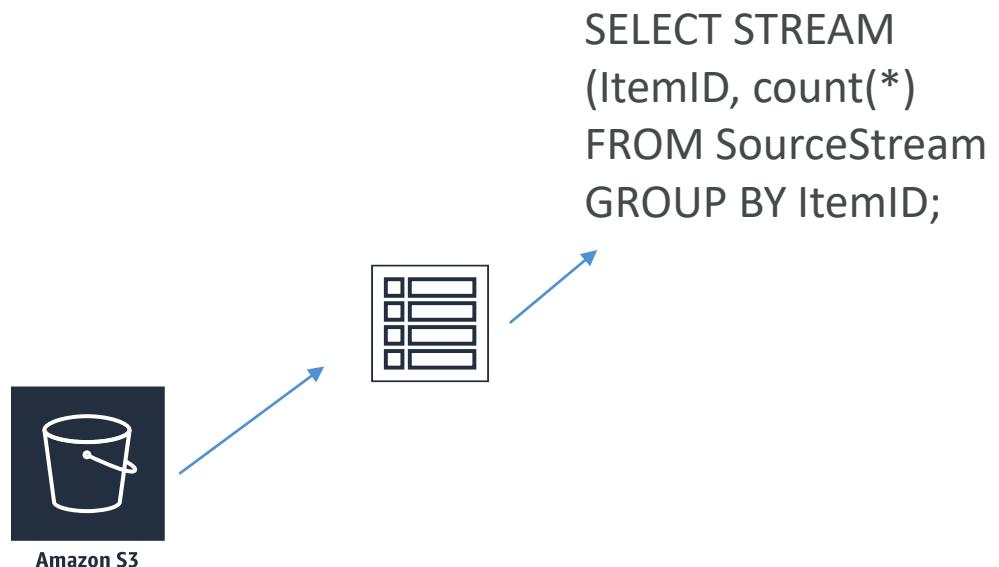


# In more depth...



# 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

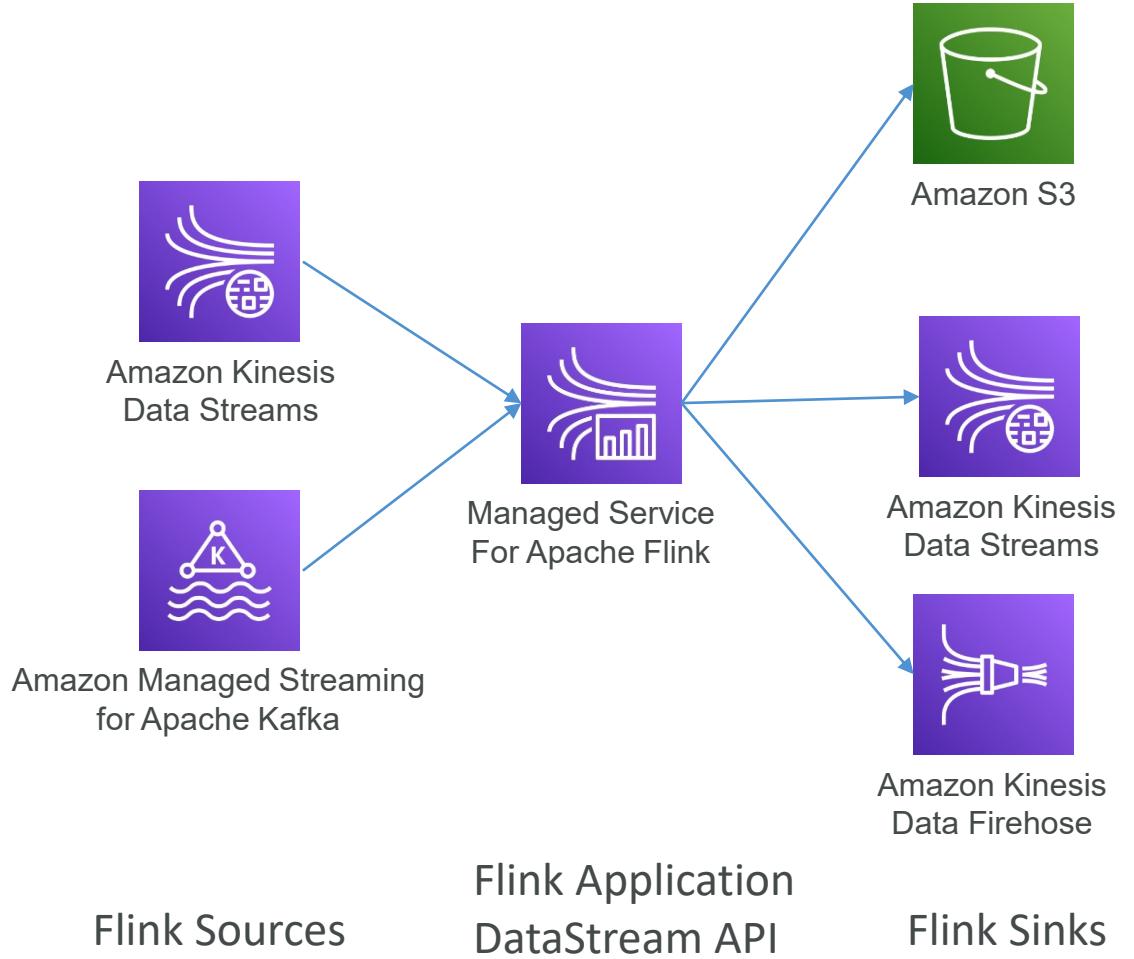


# Kinesis Data Analytics + Lambda

- 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

# Managed Service for Apache Flink

- Formerly Kinesis Data Analytics for Apache Flink or for Java
  - Kinesis Data Analytics always used Flink under the hood
  - But now supports Python and Scala
  - Flink is a framework for processing data streams
- MSAF integrates Flink with AWS
  - Instead of using SQL, you can develop your own Flink application from scratch and load it into MSAF via S3
- In addition to the DataStream API, there is a Table API for SQL access
- 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

**Nina Mishra**  
Amazon, Palo Alto, CA 94303.

NMISHRA@AMAZON.COM

**Gourav Roy**  
Amazon, Bangalore, India 560055.

GORAVR@AMAZON.COM

**Okke Schrijvers**  
Stanford University, Palo Alto, CA 94305.

OKKES@CS.STANFORD.EDU

### 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.

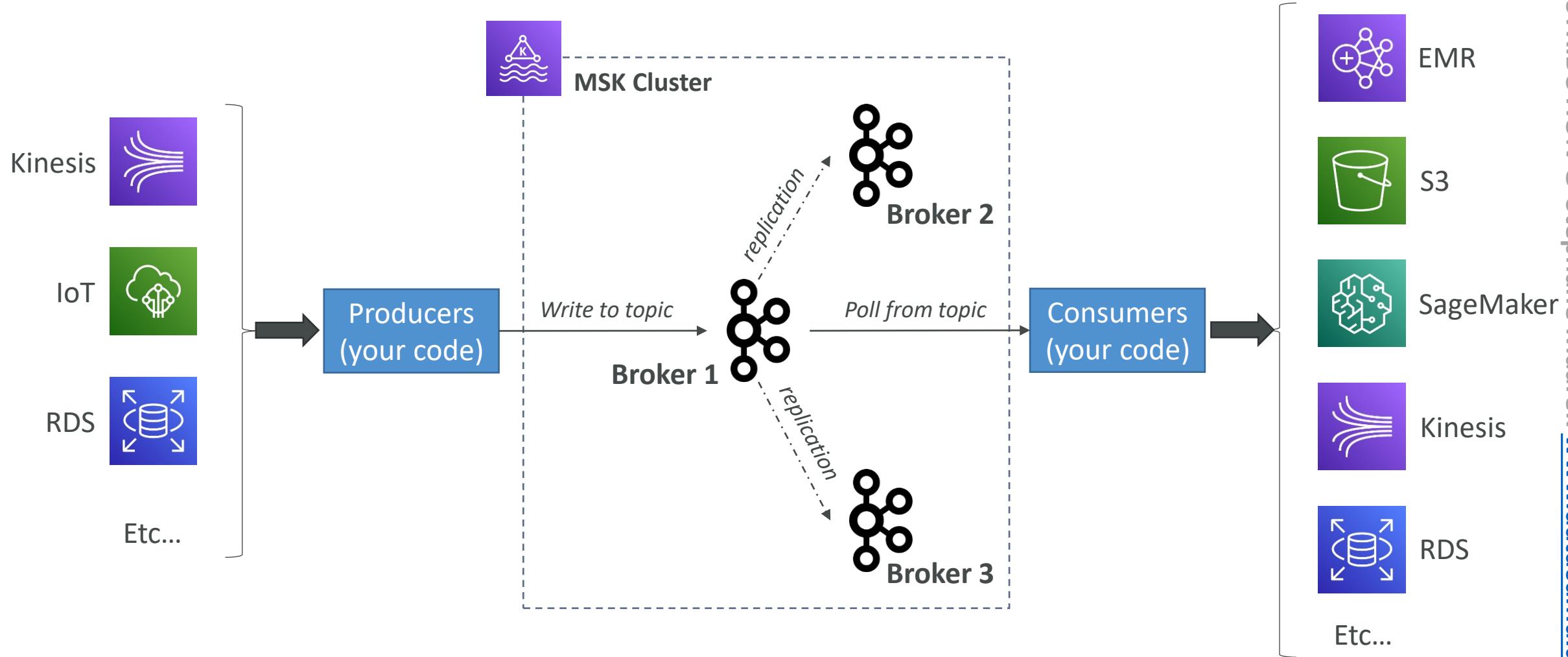
For question (2), we explore a randomized approach, akin to (Liu et al., 2012), due in part to the practical success reported in (Emmott et al., 2013). Randomization is a powerful tool and known to be valuable in supervised learning (Breiman, 2001). But its technical exploration in the context of anomaly 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 see how to extend this work to a stream. Prior work attempted 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 Managed Streaming for Apache Kafka (Amazon MSK)



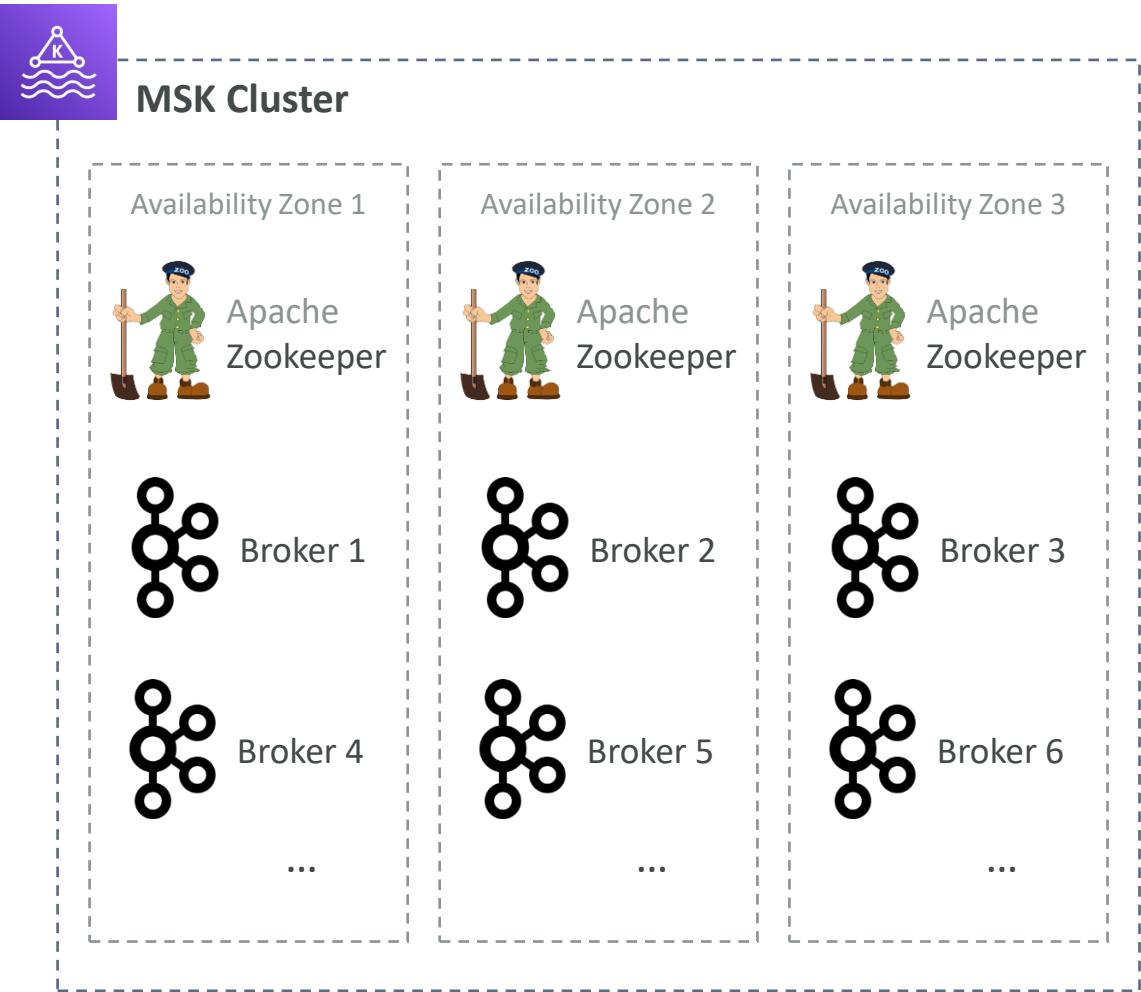
- Alternative to Kinesis (Kafka vs Kinesis next lecture)
- Fully managed Apache Kafka on AWS
  - Allow you to create, update, delete clusters
  - MSK creates & manages Kafka brokers nodes & Zookeeper nodes for you
  - Deploy the MSK cluster in your VPC, multi-AZ (up to 3 for HA)
  - Automatic recovery 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



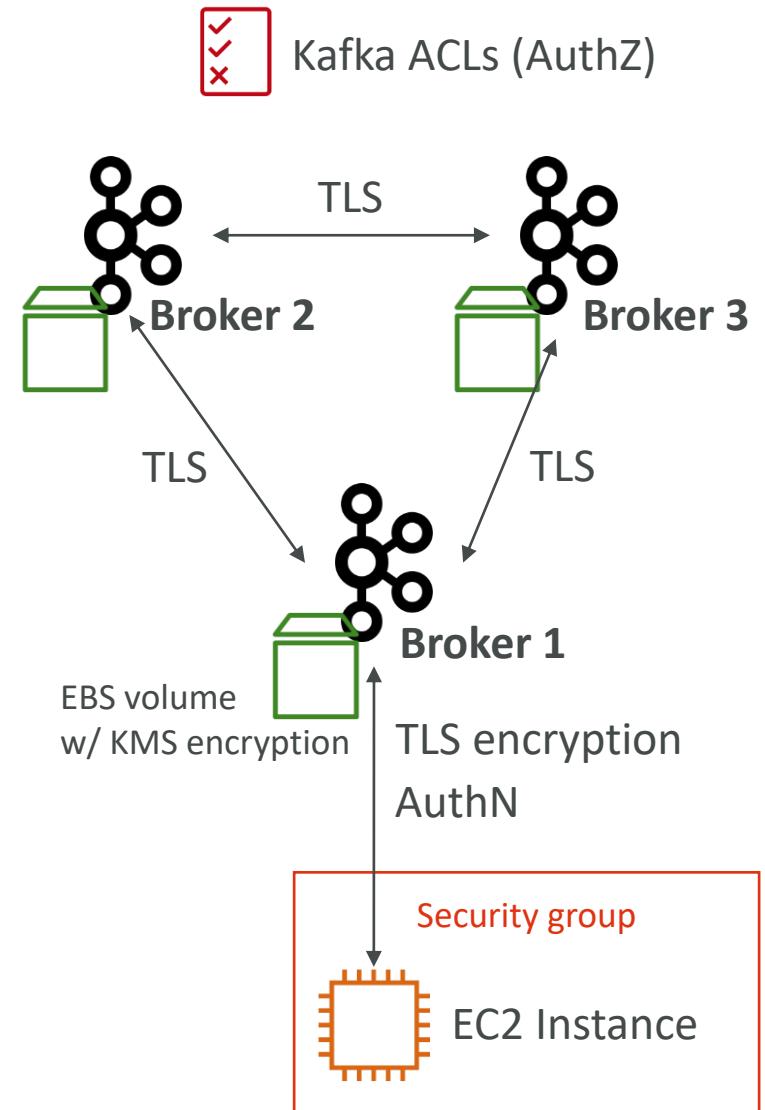
# MSK – Configurations

- 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)



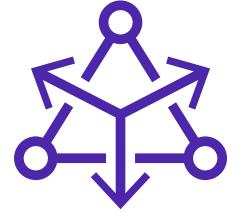
# MSK – Security

- 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

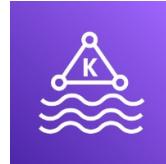
- Run Apache Kafka on MSK without managing the capacity
- MSK automatically provisions resources and scales compute & storage
- You just define your topics and your partitions and you're good to go!
- Security: IAM Access Control for all clusters
- Example Pricing:
  - \$0.75 per cluster per hour = \$558 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)

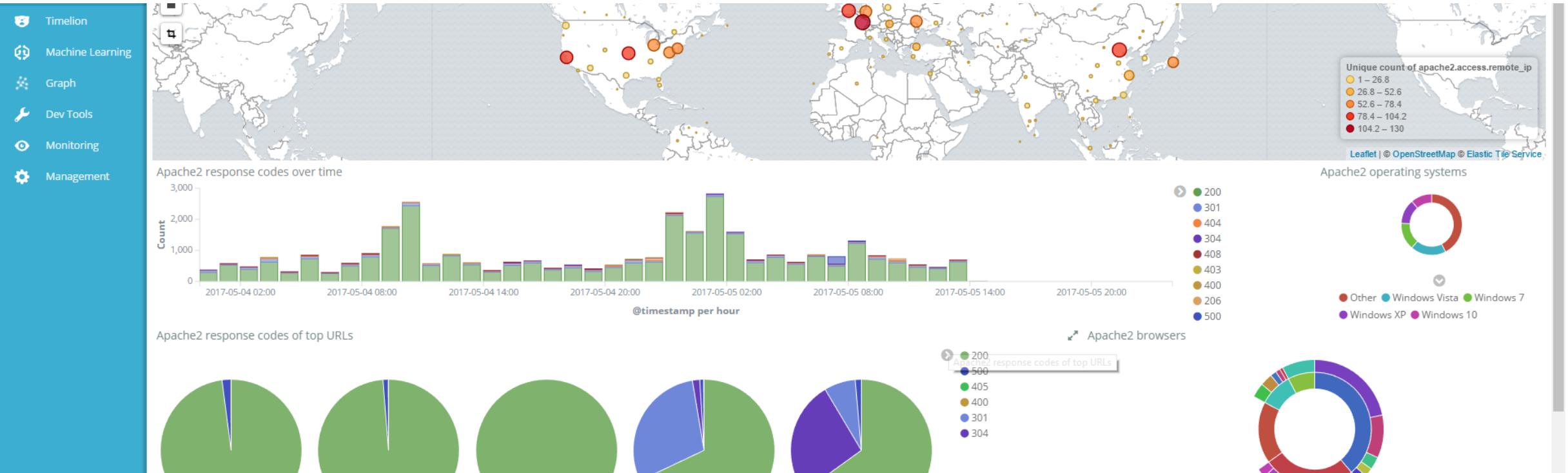
# Amazon Opensearch Service (formerly Elasticsearch)

Petabyte-scale analysis and reporting

# What is OpenSearch?

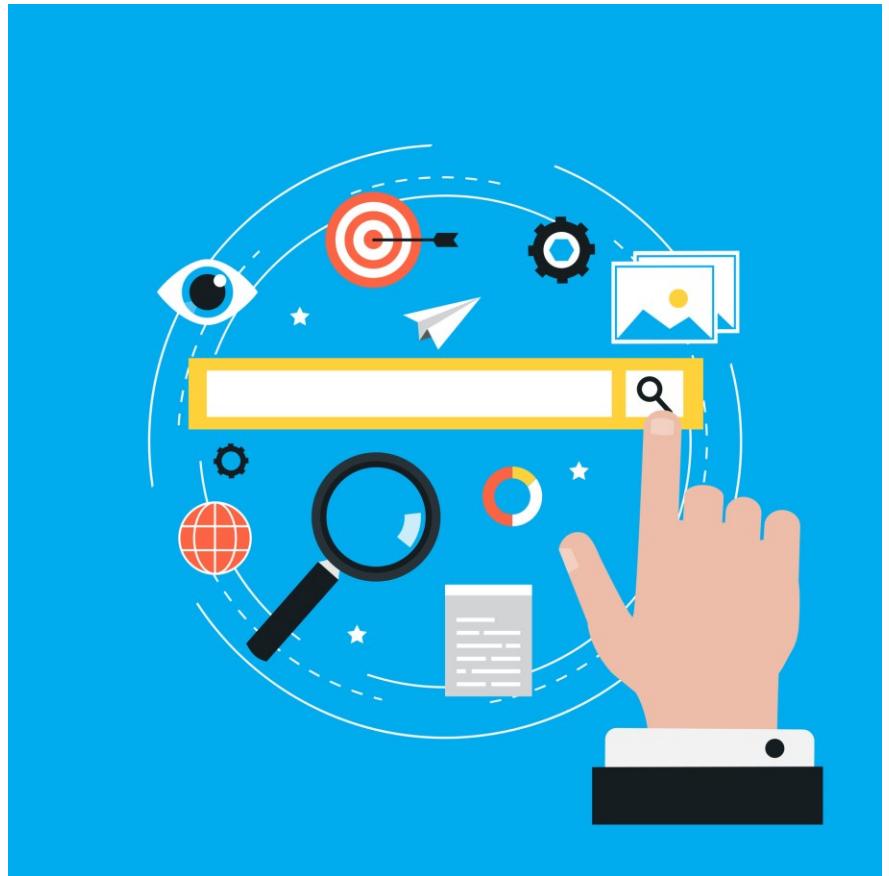
- A fork of Elasticsearch and Kibana
- A search engine
- An analysis tool
- A visualization tool (Dashboards = Kibana)
- A data pipeline
  - Kinesis replaces Beats & Logstash
- Horizontally scalable

# 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.)

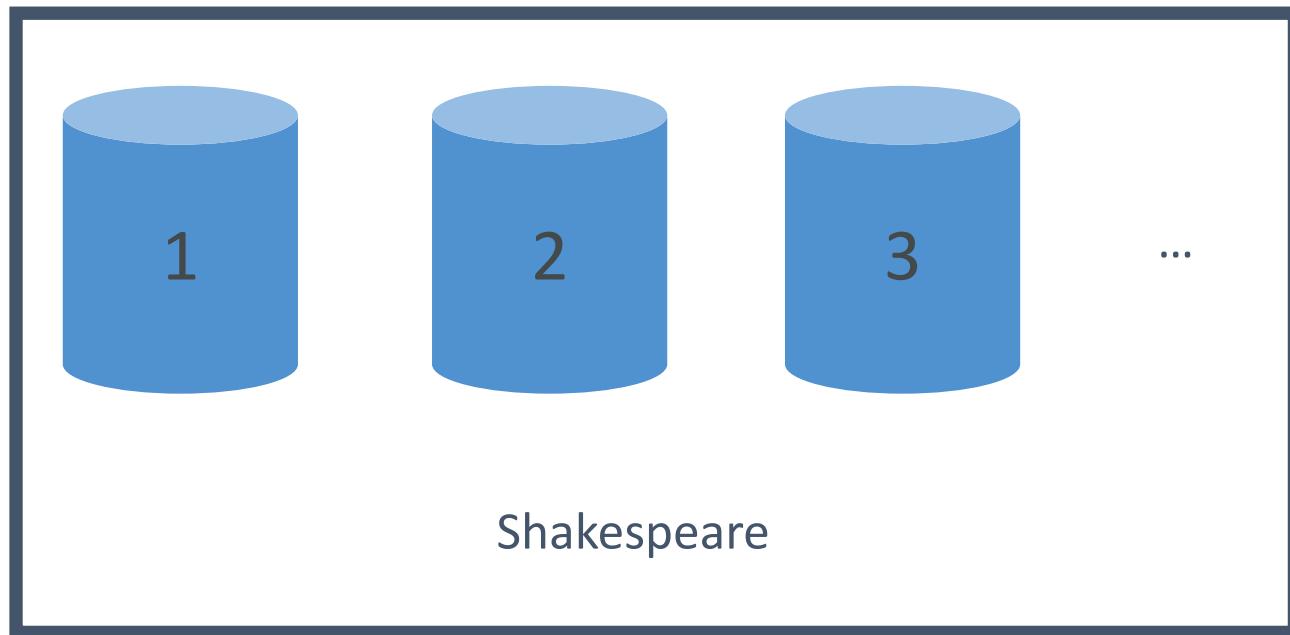


## indices

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.



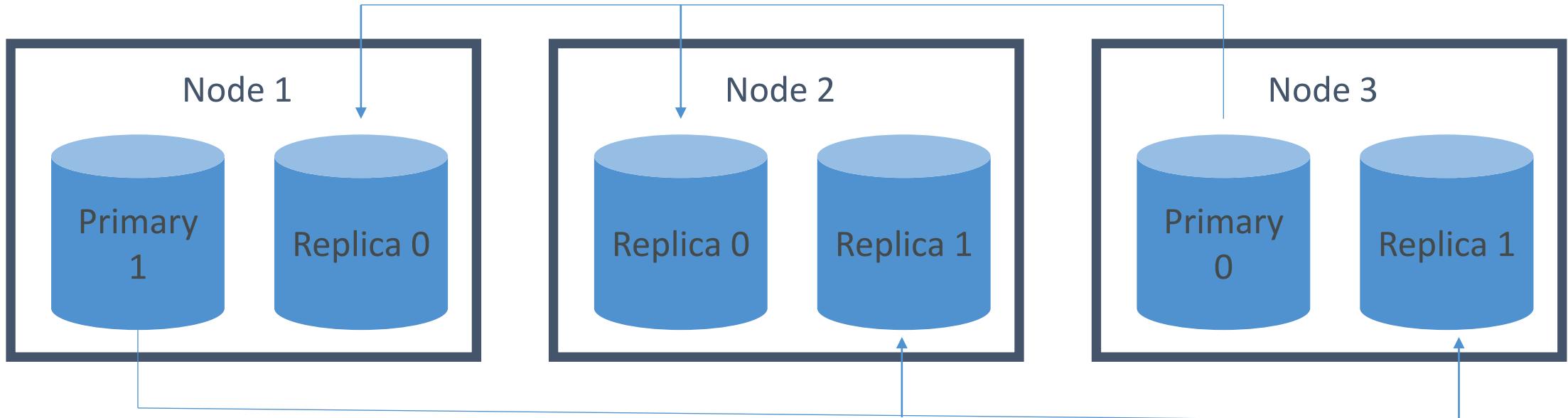
Each shard may be on a different **node** in a **cluster**.

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 (Managed)

- Fully-managed (but not serverless)
  - There is a separate serverless option now
- 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



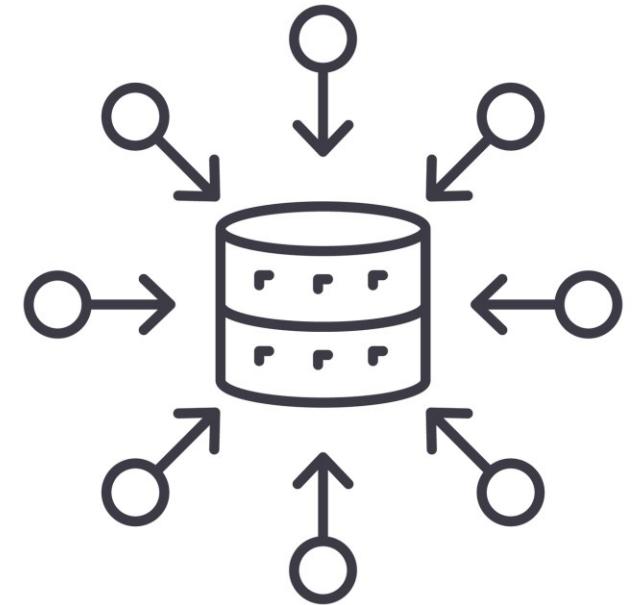
# Index State Management

- 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



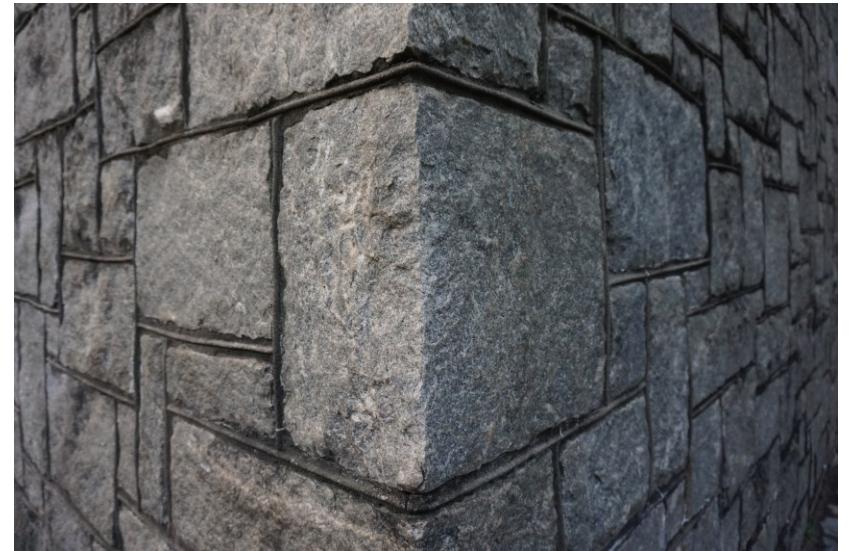
# Cross-cluster replication

- 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
  - (source data + room to grow) \* (1 + indexing overhead) / 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



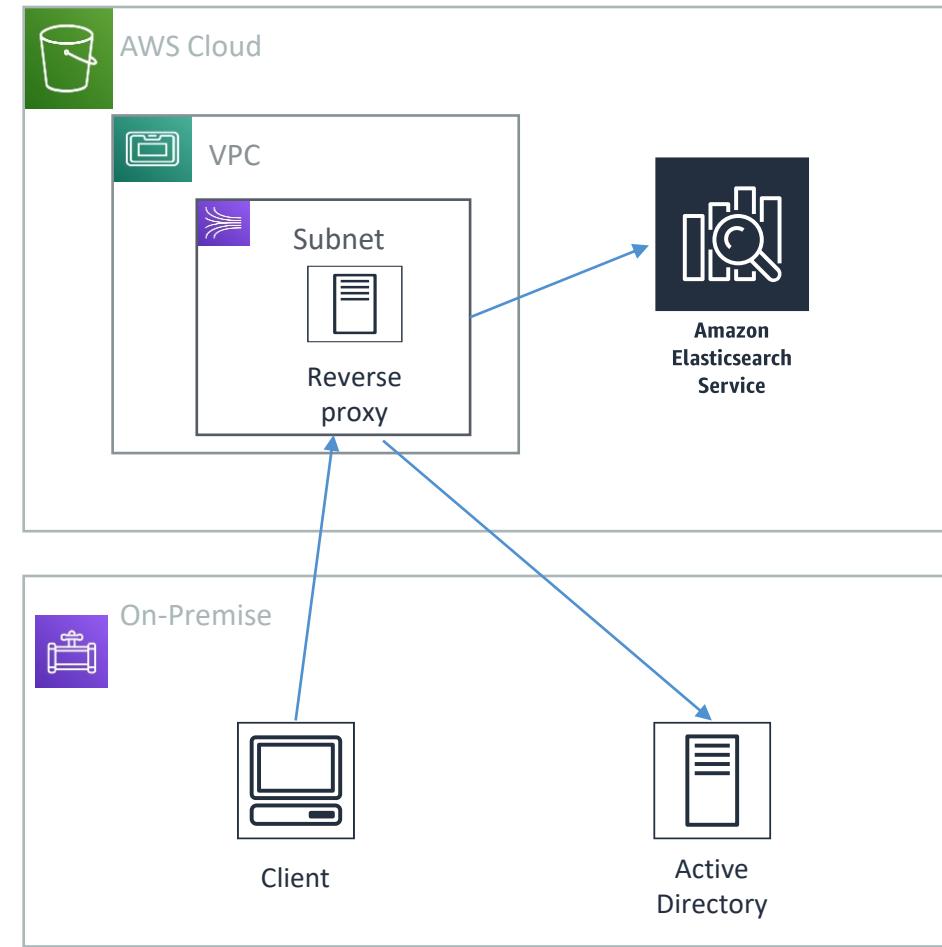
# Amazon Opensearch Security

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



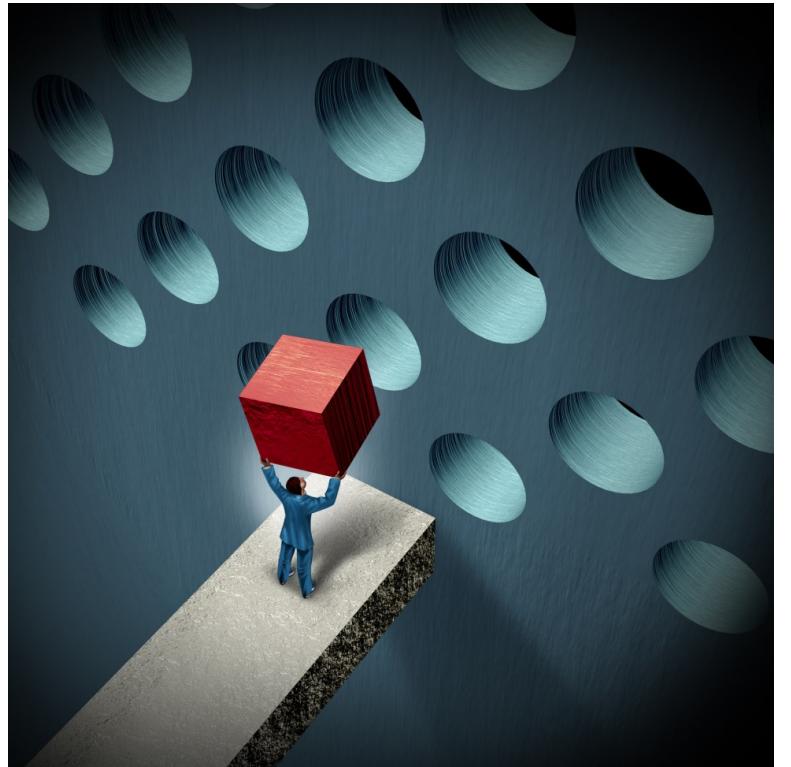
# Securing Dashboards

- 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 OpenSearch Serverless

- On-demand autoscaling!
- Works against “collections” instead of provisioned domains
  - May be “search” or “time series” type
- Always encrypted with your KMS key
  - Data access policies
  - Encryption at rest is required
  - May configure security policies across many collections
- Capacity measured in Opensearch Compute Units (OCUs)
  - Can set an upper limit, lower limit is always 2 for indexing, 2 for search

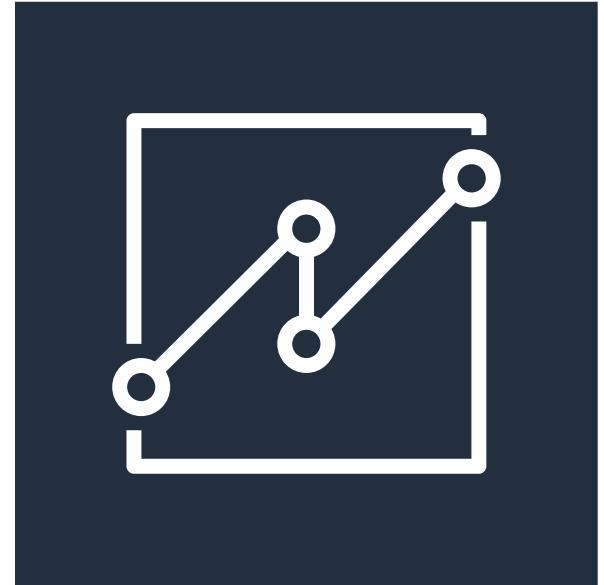


# Amazon QuickSight

Business analytics and visualizations in the cloud

# What is QuickSight?

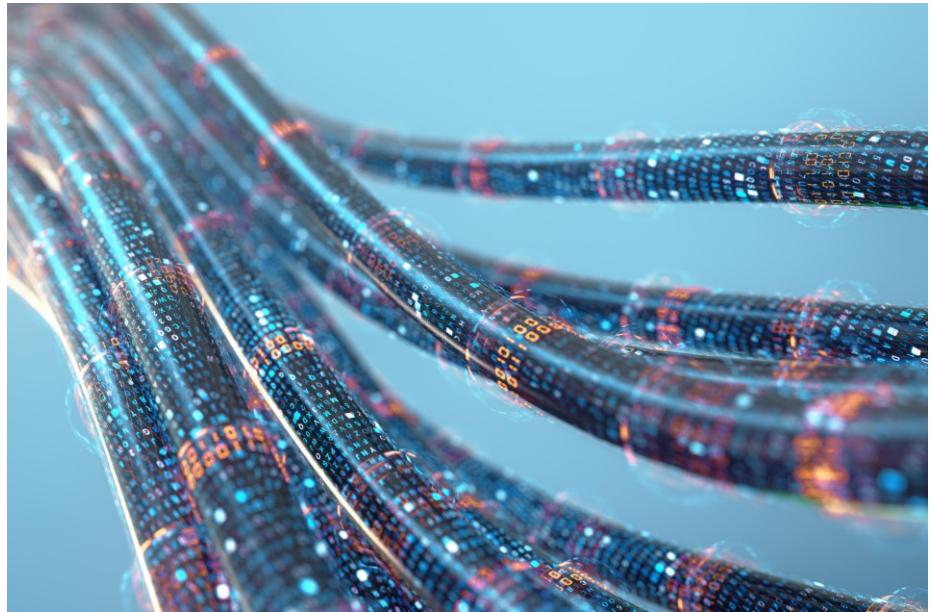
- Fast, easy, cloud-powered business analytics service
- Allows all employees in an organization to:
  - Build visualizations
  - Build paginated reports
  - Perform ad-hoc analysis
  - Get alerts on detected anomalies
  - Quickly get business insights from data
  - Anytime, on any device (browsers,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



**Amazon  
QuickSight**

# QuickSight Data Sources

- Redshift
- Aurora / RDS
- Athena
- OpenSearch
- IoT Analytics
- 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

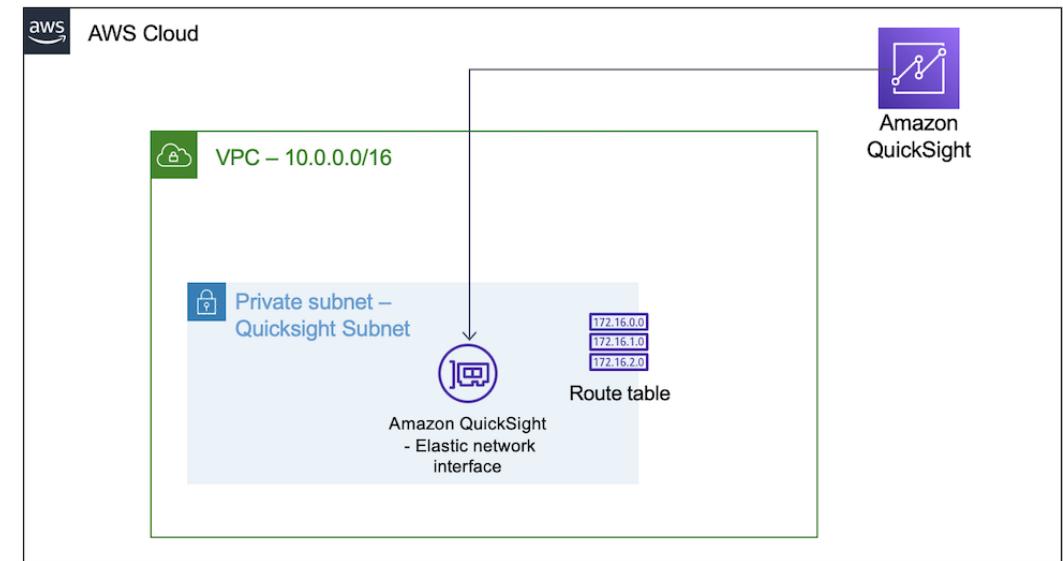


# Quicksight + Redshift: Security

- 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>

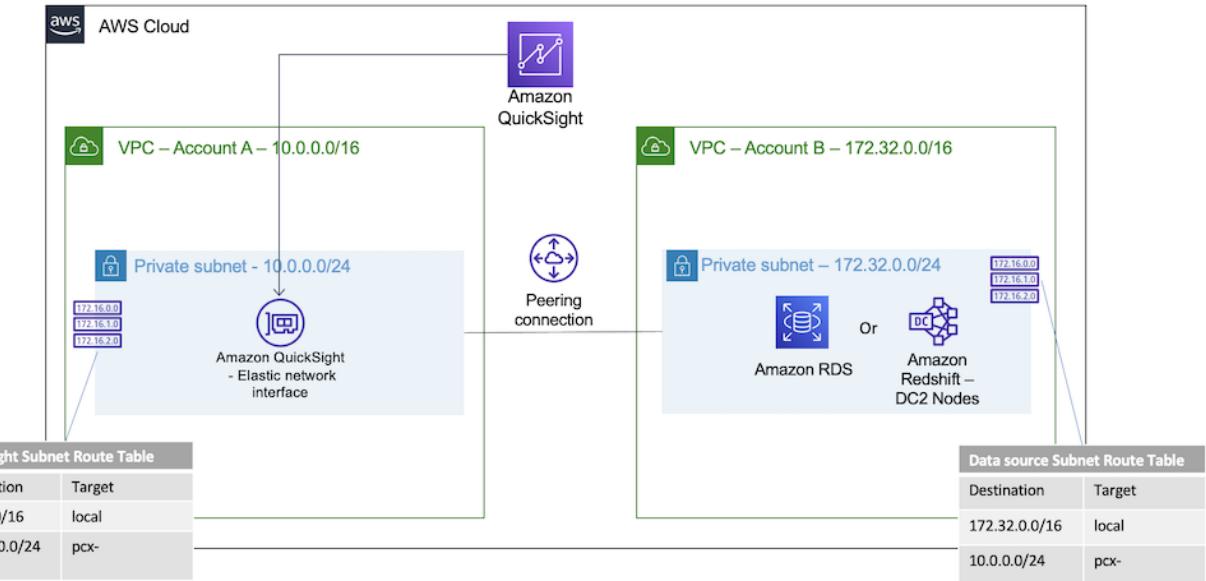
# Quicksight / Redshift / RDS cross-region

- Other ways to do it (if you have Enterprise Edition)
- Create a private subnet in a VPC
- Use Elastic Network Interface to put Quicksight in the subnet
  - This is the part that requires Enterprise Edition
- This can also enable cross-account access



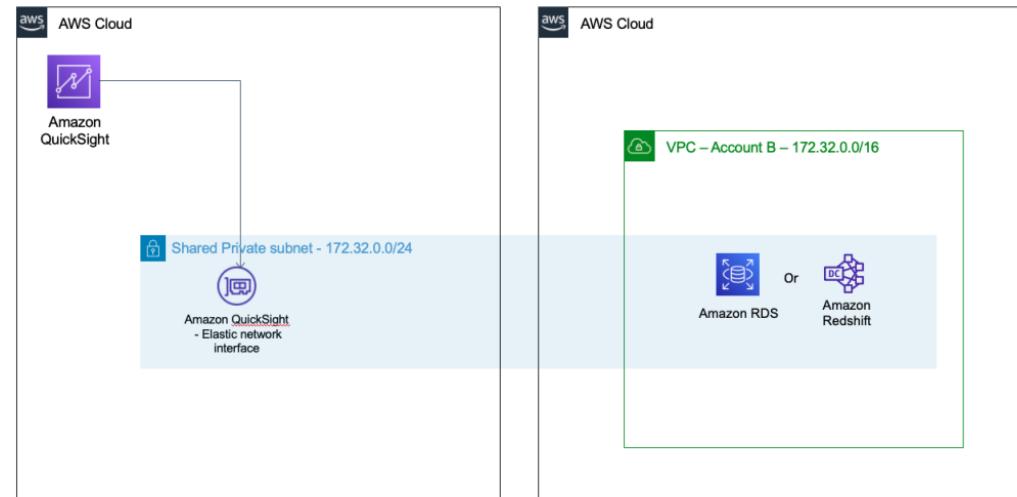
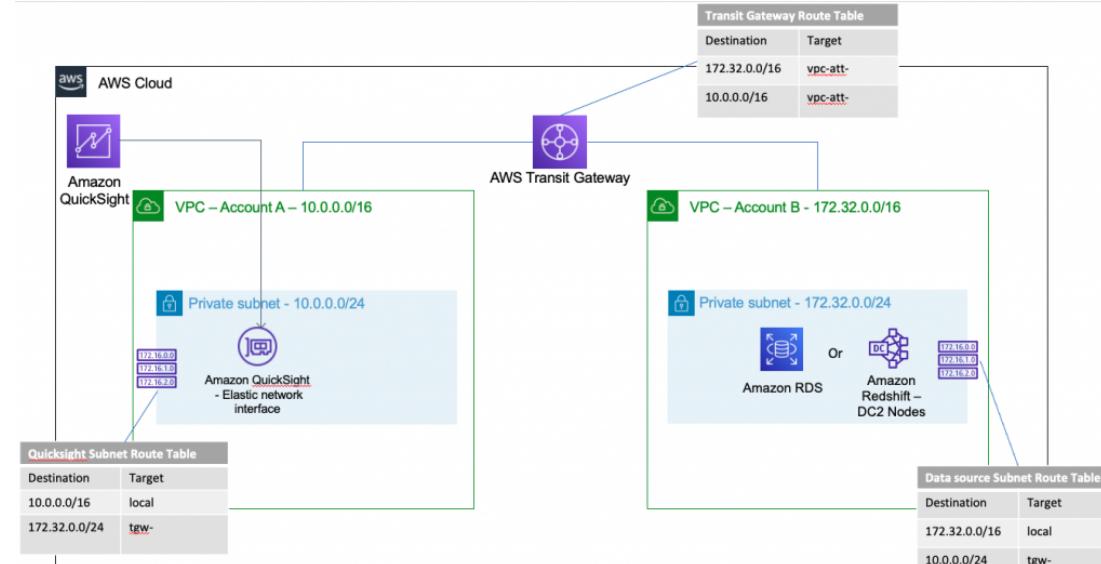
# Quicksight / Redshift / RDS cross-region

- Now you can create a **peering connection** between your private subnet and another private subnet containing your data
  - This also works for cross-account access



# Quicksight / Redshift / RDS cross-account

- Can use an AWS Transit Gateway to connect your subnets
  - But this must be in the same org & region
  - Or you can peer Transit Gateways in different regions
- OR use AWS PrivateLink to connect them
- OR use VPC sharing to connect them



# QuickSight User Management

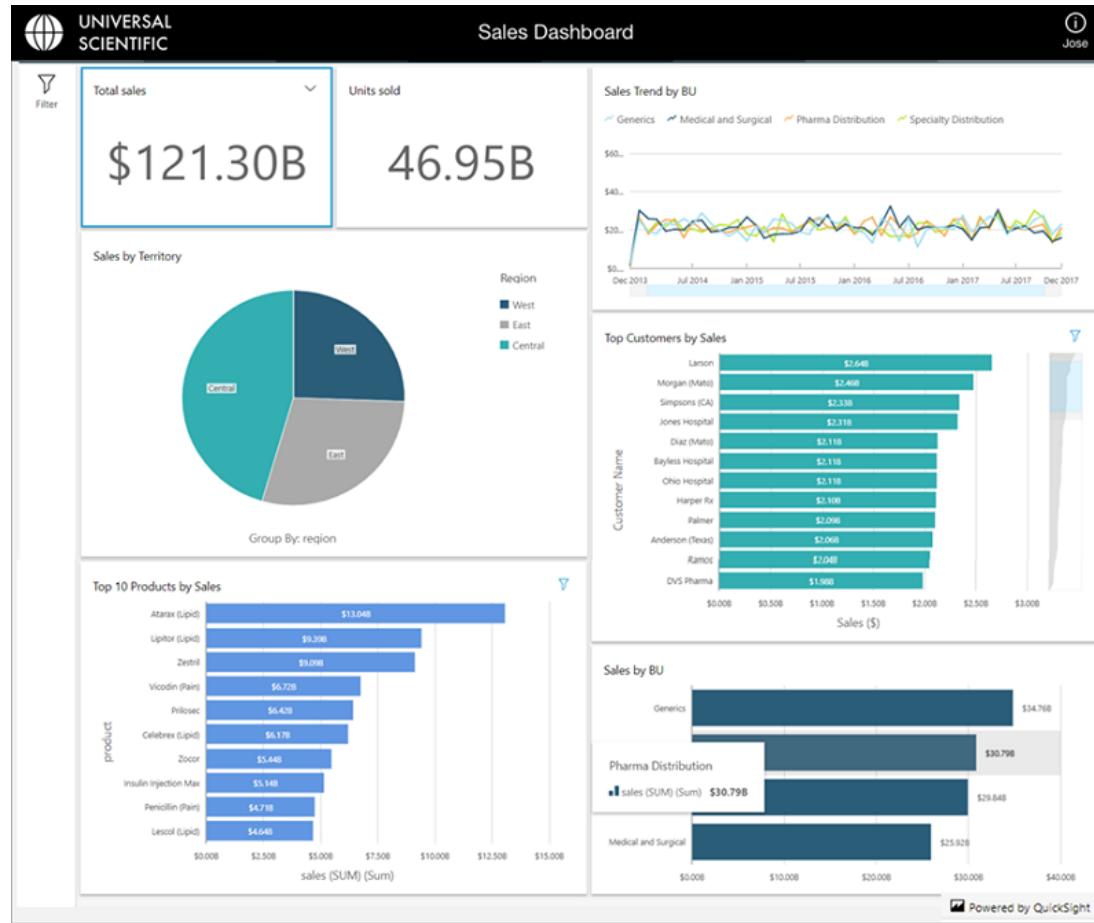
- Users defined via IAM, or email signup
- Active Directory connector with QuickSight Enterprise Edition
  - All keys are managed by AWS; you CANNOT use customer-provided keys
  - Enterprise edition only!
  - Can



# QuickSight Pricing

- Annual subscription
  - Standard: \$9 / user / month
  - Enterprise: \$18 / user / month
  - With Quicksight Q: \$28 / user / month
- Extra SPICE capacity (beyond 10GB)
  - \$0.25 (standard) \$0.38 (enterprise) / GB / month
- Month to month
  - Standard: \$12 / user / month
  - Enterprise: \$24 / user / month
  - With Quicksight Q: \$34 / user / month
- Enterprise edition
  - Encryption at rest
  - Microsoft Active Directory integration

# QuickSight Dashboards

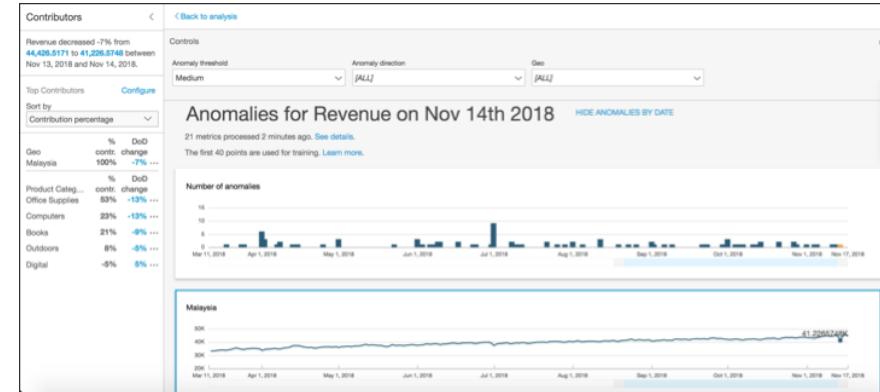


- Read-only snapshots of an analysis
- Can share with others with Quicksight access
- Can share even more widely with *embedded dashboards*
  - Embed within an application
  - Authenticate with Active Directory / Cognito / SSO
  - QuickSight Javascript SDK / QuickSight API
  - Whitelist domains where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



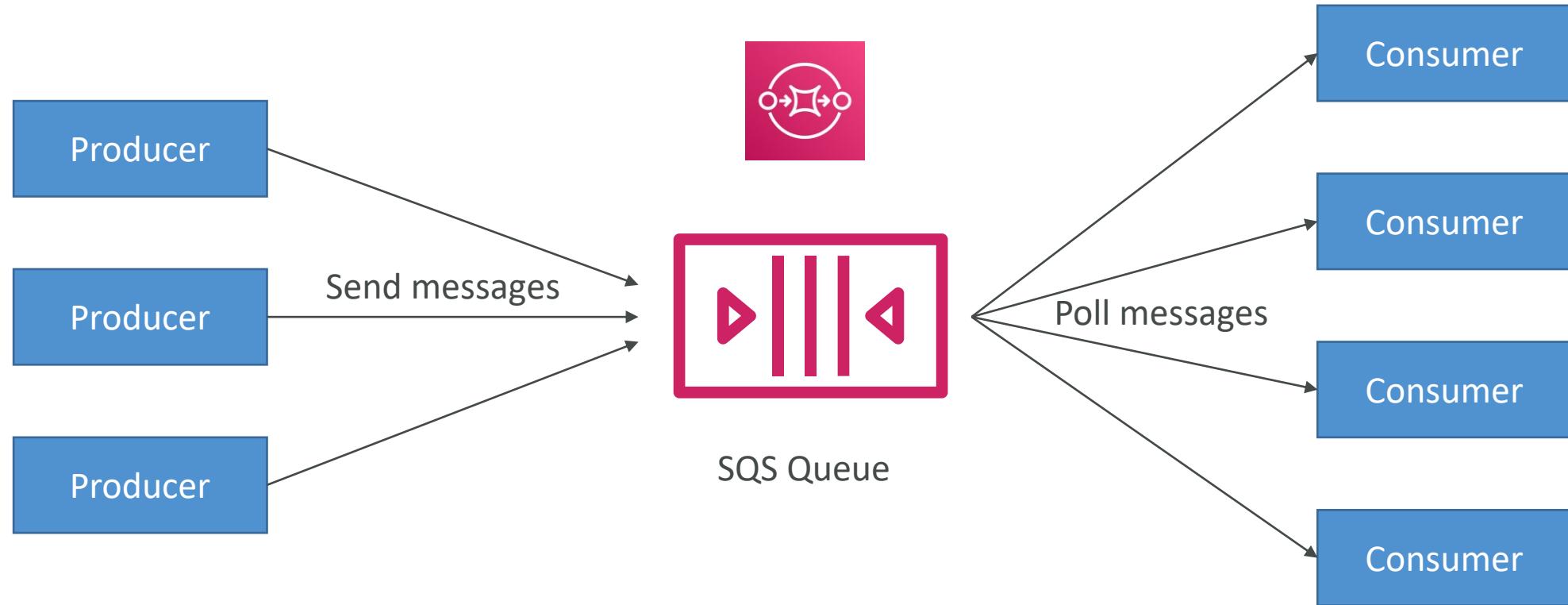
Total Revenue for Nov 17, 2018 **decreased by 2.15%**  
**(-131,031.34720000066)** from 6,100,697.1616 to  
 5,969,665.8144. Compounded growth rate for the last 4 days is  
**-0.26% worse than expected.**

# Application Integration

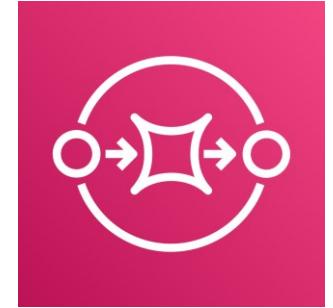
Moving data between systems in AWS

# AWS SQS

## What's a queue?



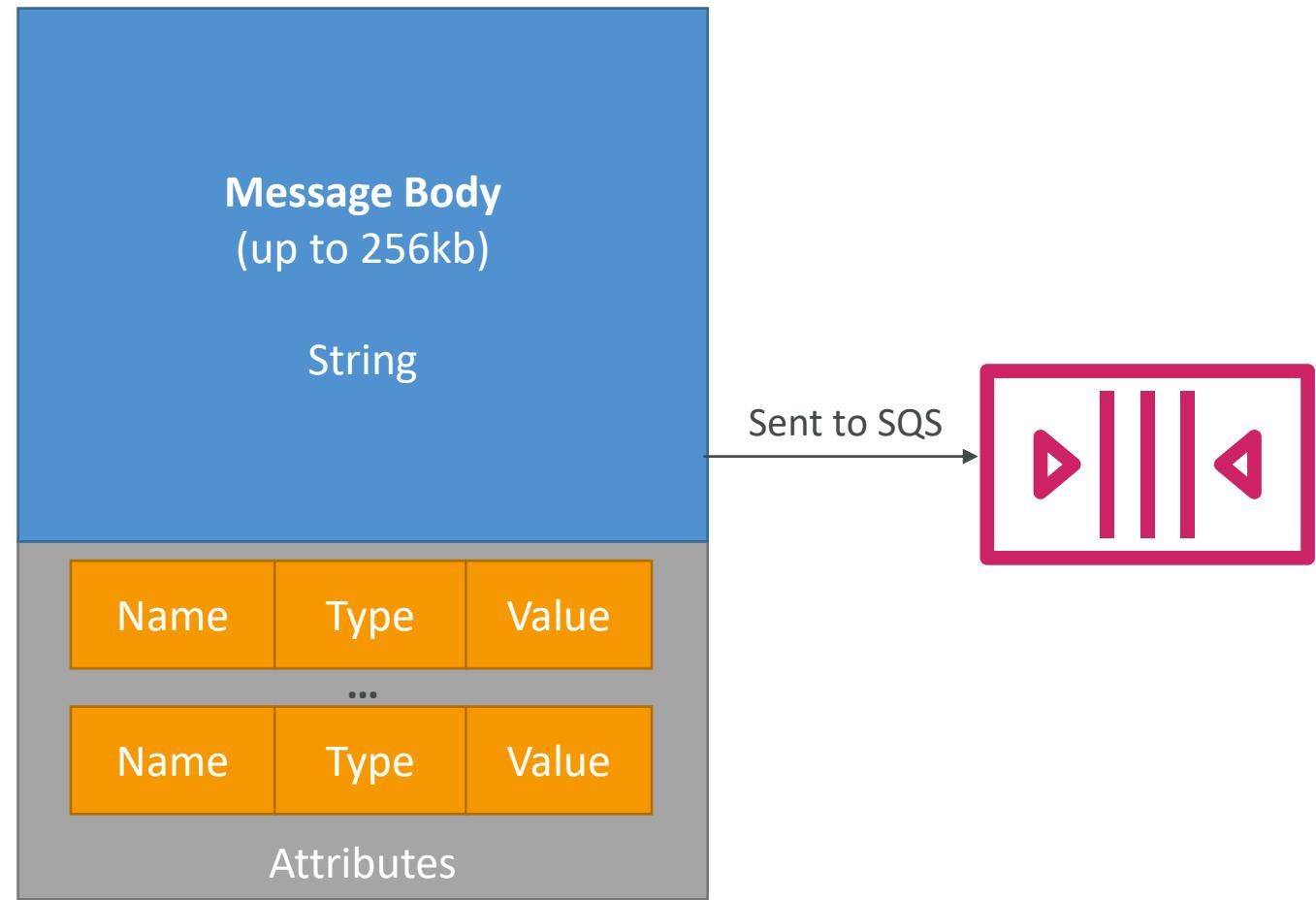
# AWS SQS – Standard Queue



- Oldest offering (over 10 years old)
- Fully managed
- Scales from 1 message per second to 15,000s 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

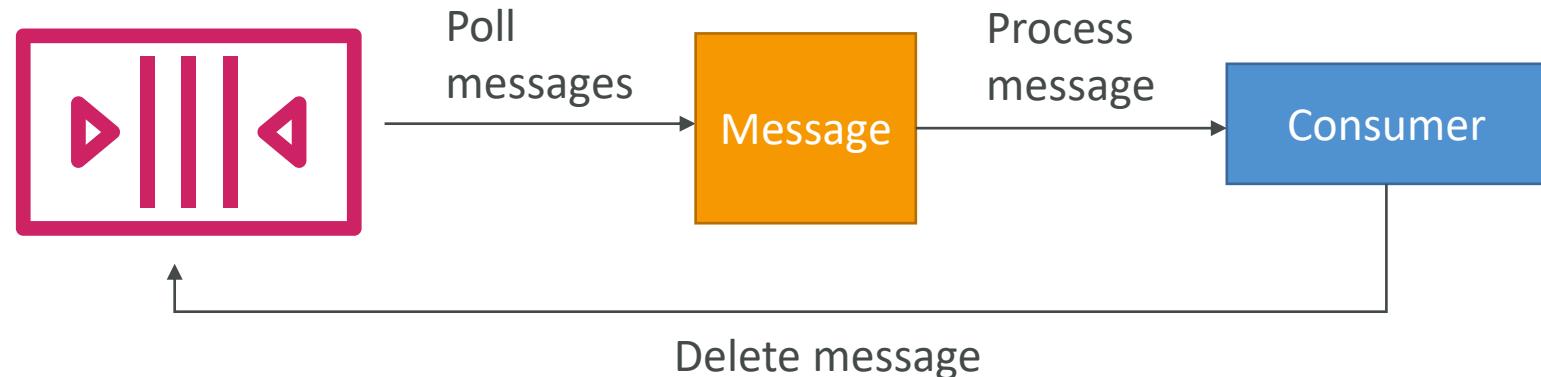
# SQS – Producing Messages

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



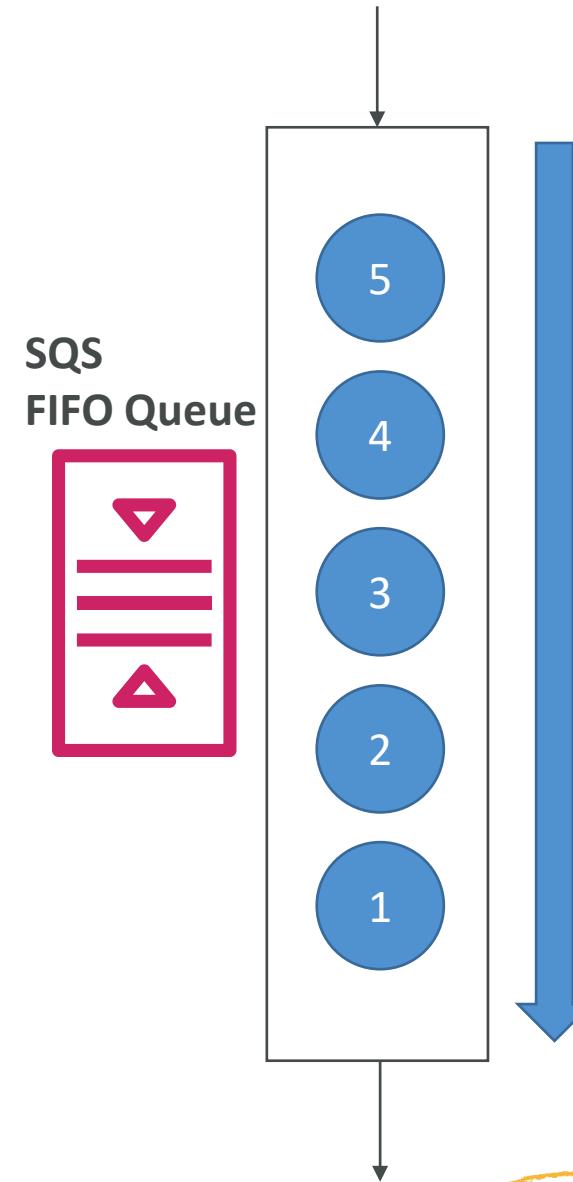
# SQS – Consuming Messages

- 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



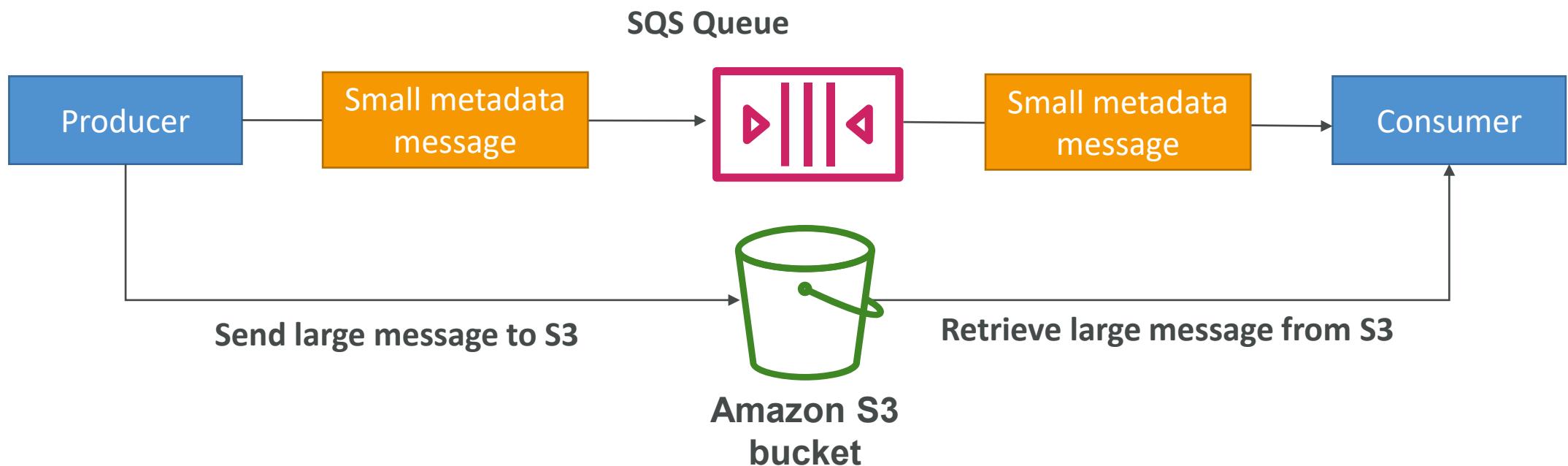
# AWS SQS – FIFO Queue

- 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:**

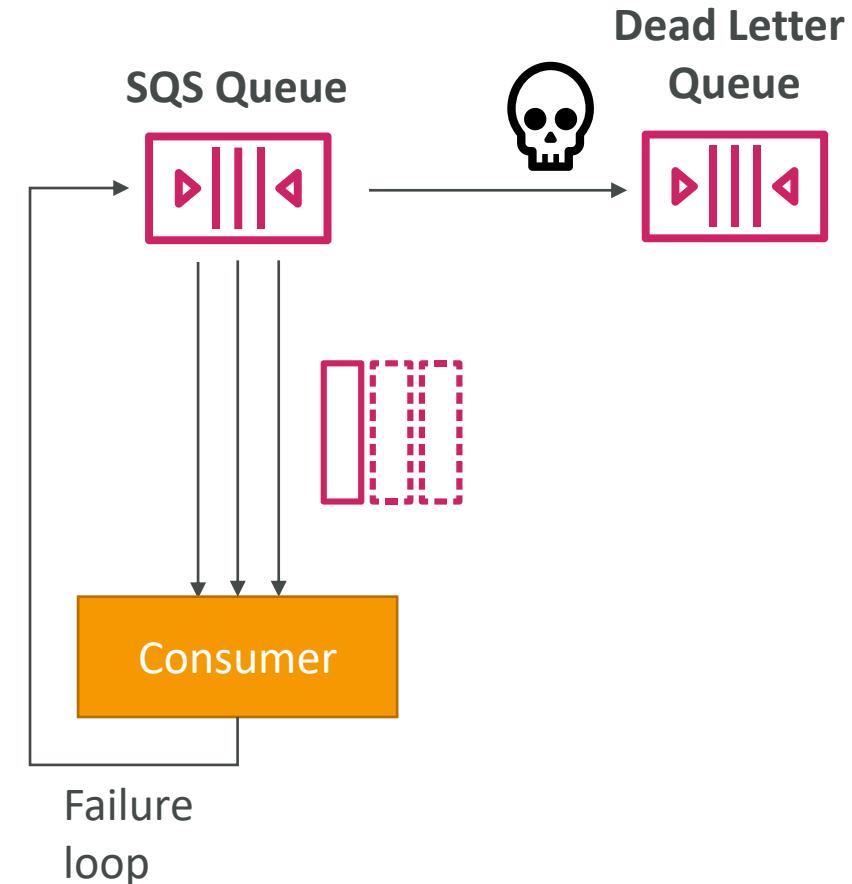
- Order processing
- Image Processing
- Auto scaling queues according to messages.
- Buffer and Batch messages for future processing.
- Request Offloading

- **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”

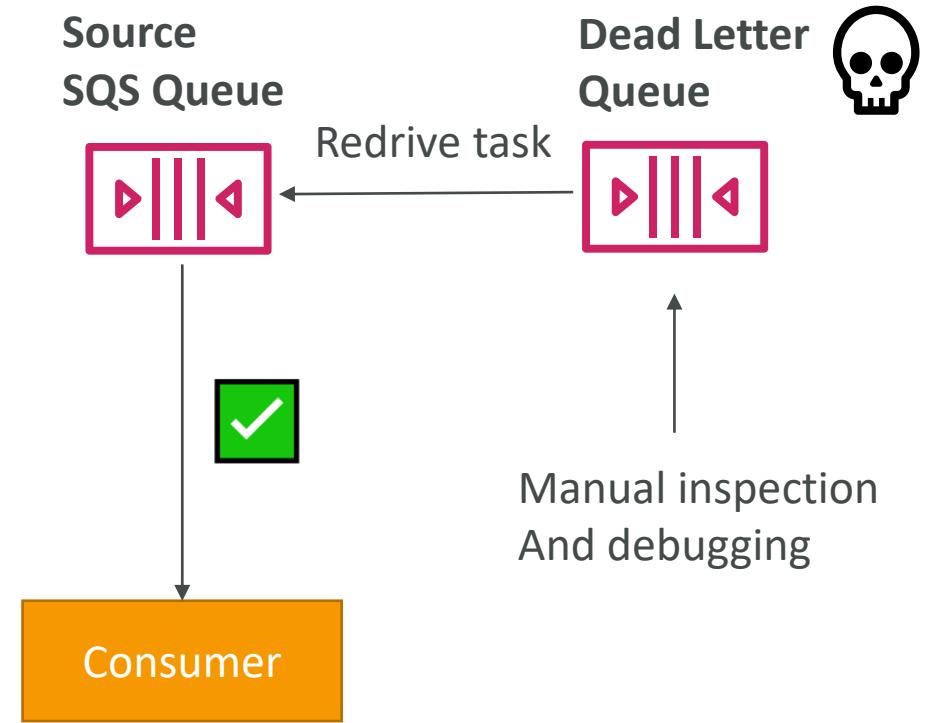
# Amazon SQS – Dead Letter Queue (DLQ)

- If a consumer fails to process a message within the Visibility Timeout...  
the message goes back to the queue!
- We can set a threshold of how many times a message can go back to the queue
- After the **MaximumReceives** threshold is exceeded, the message goes into a Dead Letter Queue (DLQ)
- Useful for debugging!
- **DLQ of a FIFO queue must also be a FIFO queue**
- **DLQ of a Standard queue must also be a Standard queue**
- Make sure to process the messages in the DLQ before they expire:



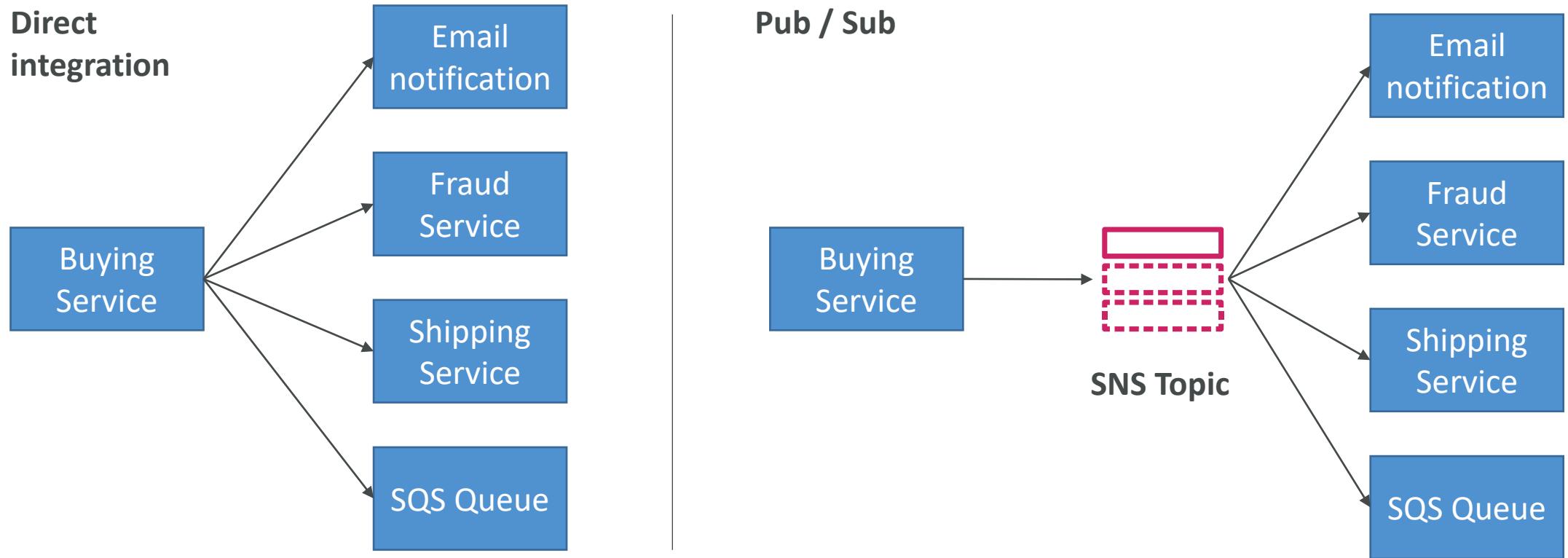
# SQS DLQ – Redrive to Source

- Feature to help consume messages in the DLQ to understand what is wrong with them
- When our code is fixed, we can redrive the messages from the DLQ back into the source queue (or any other queue) in batches without writing custom code



# Amazon SNS

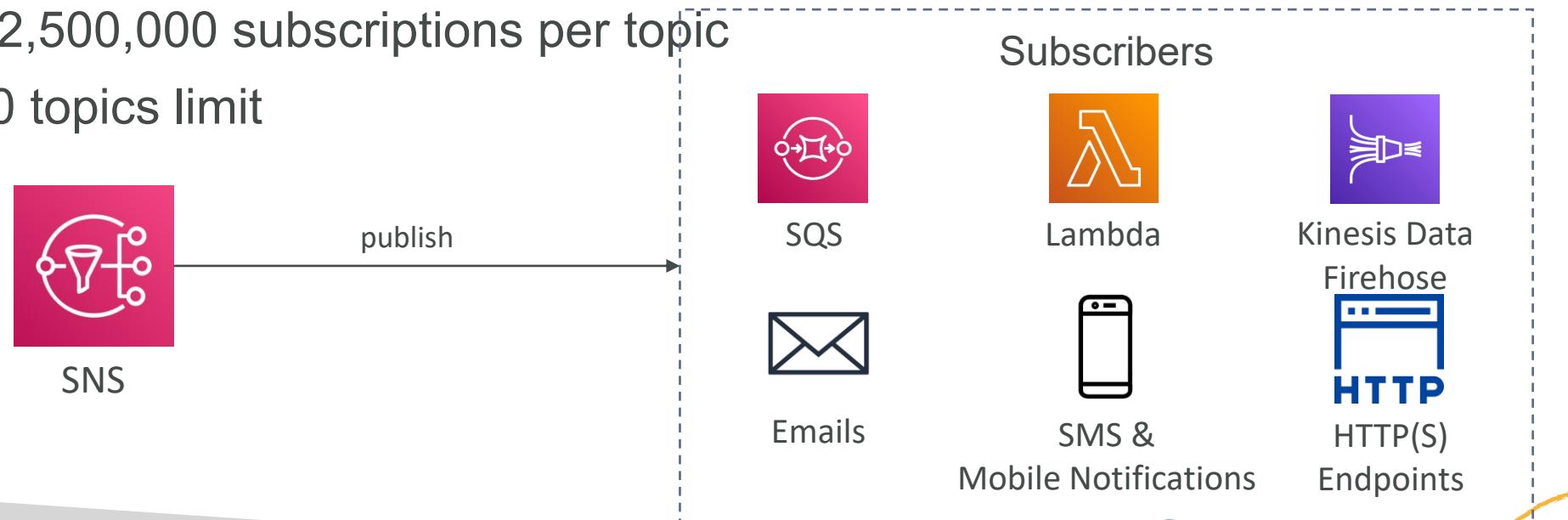
- What if you want to send one message to many receivers?



# Amazon SNS

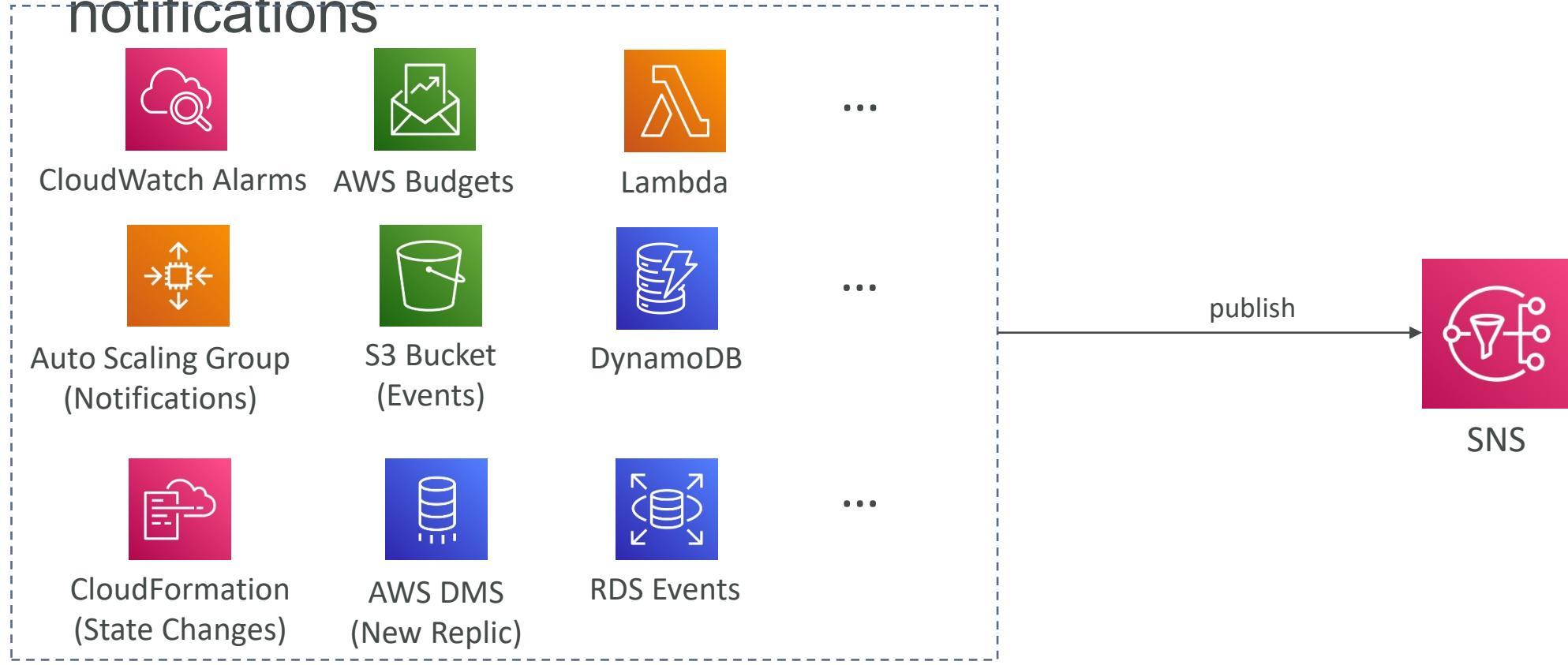


- The “event producer” only sends message to one SNS topic
- As many “event receivers” (subscriptions) as we want to listen to the SNS topic notifications
- Each subscriber to the topic will get all the messages (note: new feature to filter messages)
- Up to 12,500,000 subscriptions per topic
- 100,000 topics limit



# SNS integrates with a lot of AWS services

- Many AWS services can send data directly to SNS for notifications



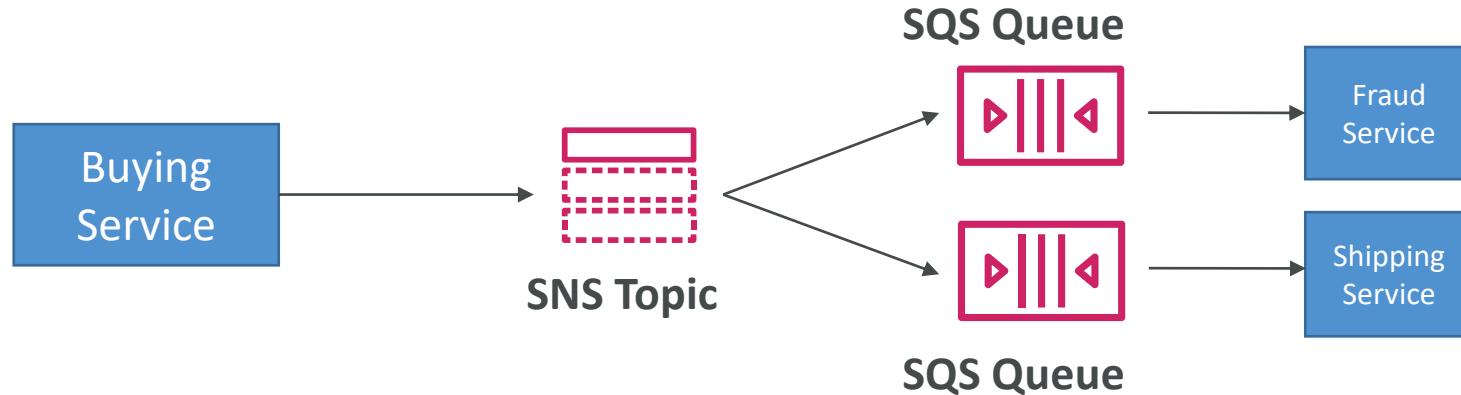
# Amazon SNS – How to publish

- Topic Publish (using the SDK)
  - Create a topic
  - Create a subscription (or many)
  - Publish to the topic
- Direct Publish (for mobile apps SDK)
  - Create a platform application
  - Create a platform endpoint
  - Publish to the platform endpoint
  - Works with Google GCM, Apple APNS, Amazon ADM...

# Amazon SNS – Security

- **Encryption:**
  - In-flight encryption using HTTPS API
  - At-rest encryption using KMS keys
  - Client-side encryption if the client wants to perform encryption/decryption itself
- **Access Controls:** IAM policies to regulate access to the SNS API
- **SNS Access Policies** (similar to S3 bucket policies)
  - Useful for cross-account access to SNS topics
  - Useful for allowing other services ( S3...) to write to an SNS topic

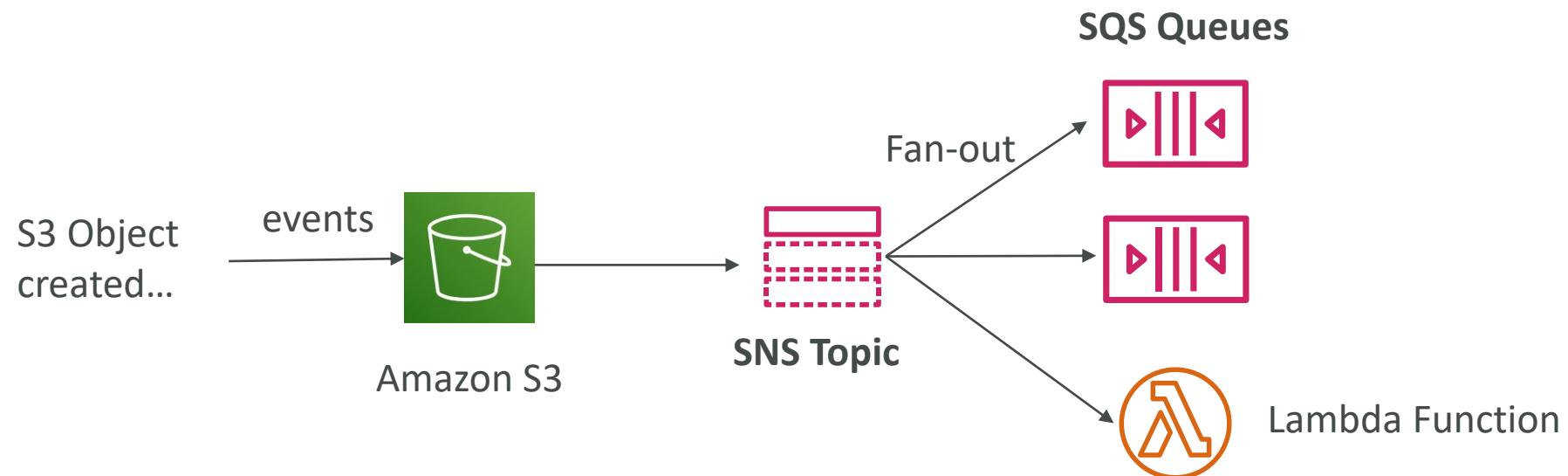
# SNS + SQS: Fan Out



- Push once in SNS, receive in all SQS queues that are subscribers
- Fully decoupled, no data loss
- SQS allows for: data persistence, delayed processing and retries of work
- Ability to add more SQS subscribers over time
- Make sure your SQS queue **access policy** allows for SNS to

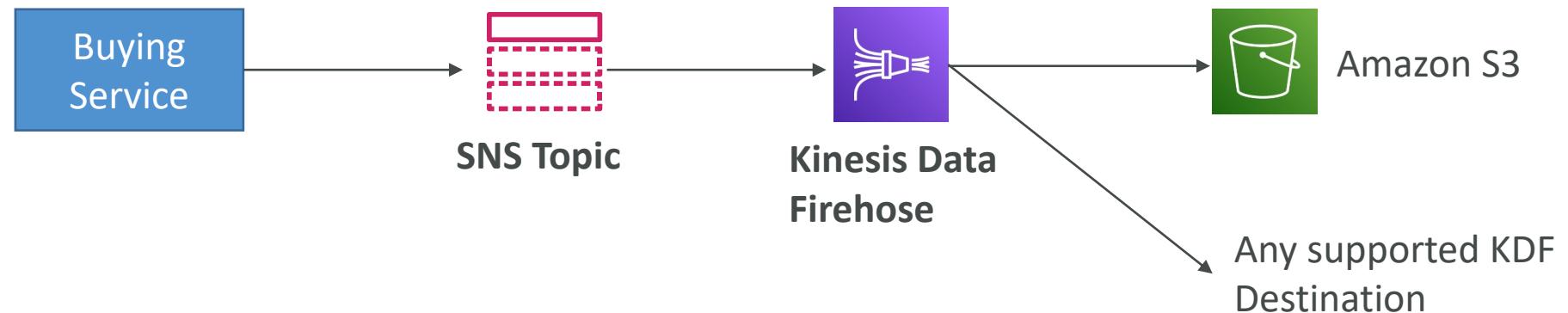
# Application: S3 Events to multiple queues

- For the same combination of: **event type** (e.g. object create) and **prefix** (e.g. images/) you can only have one S3 Event rule
- If you want to send the same S3 event to many SQS queues, use fan-out



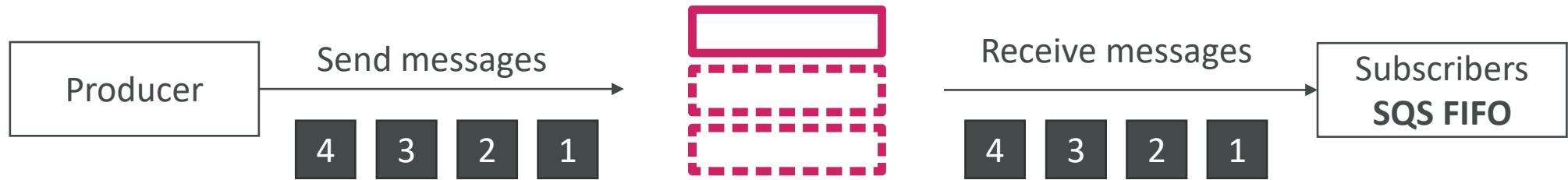
# Application: SNS to Amazon S3 through Kinesis Data Firehose

- SNS can send to Kinesis and therefore we can have the following solutions architecture:



# Amazon SNS – FIFO Topic

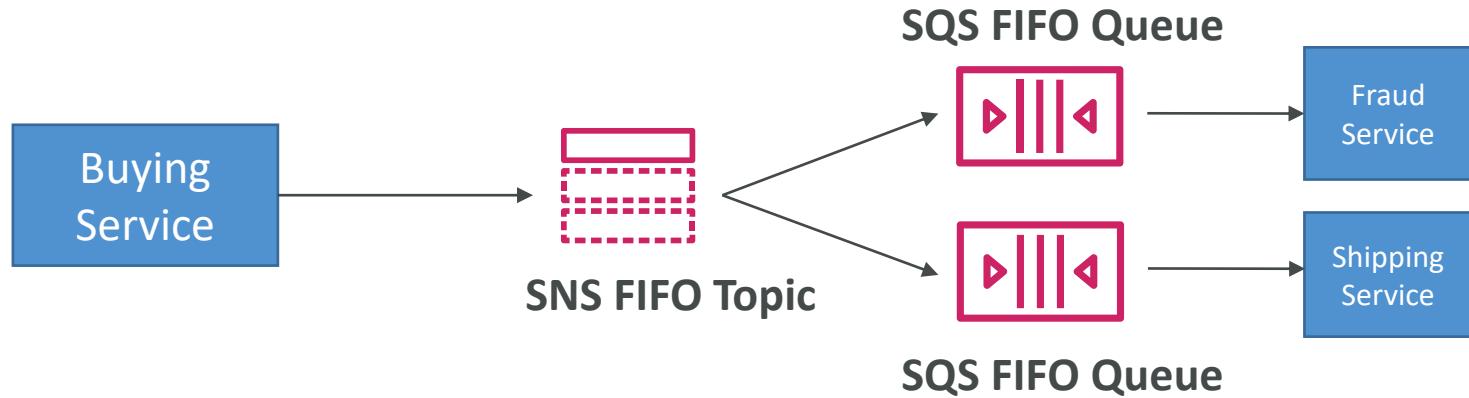
- FIFO = First In First Out (ordering of messages in the topic)



- Similar features as SQS FIFO:
  - **Ordering** by Message Group ID (all messages in the same group are ordered)
  - **Deduplication** using a Deduplication ID or Content Based Deduplication
- **Can have SQS Standard and FIFO queues as subscribers**
- **Limited throughput** (same throughput as SQS FIFO)

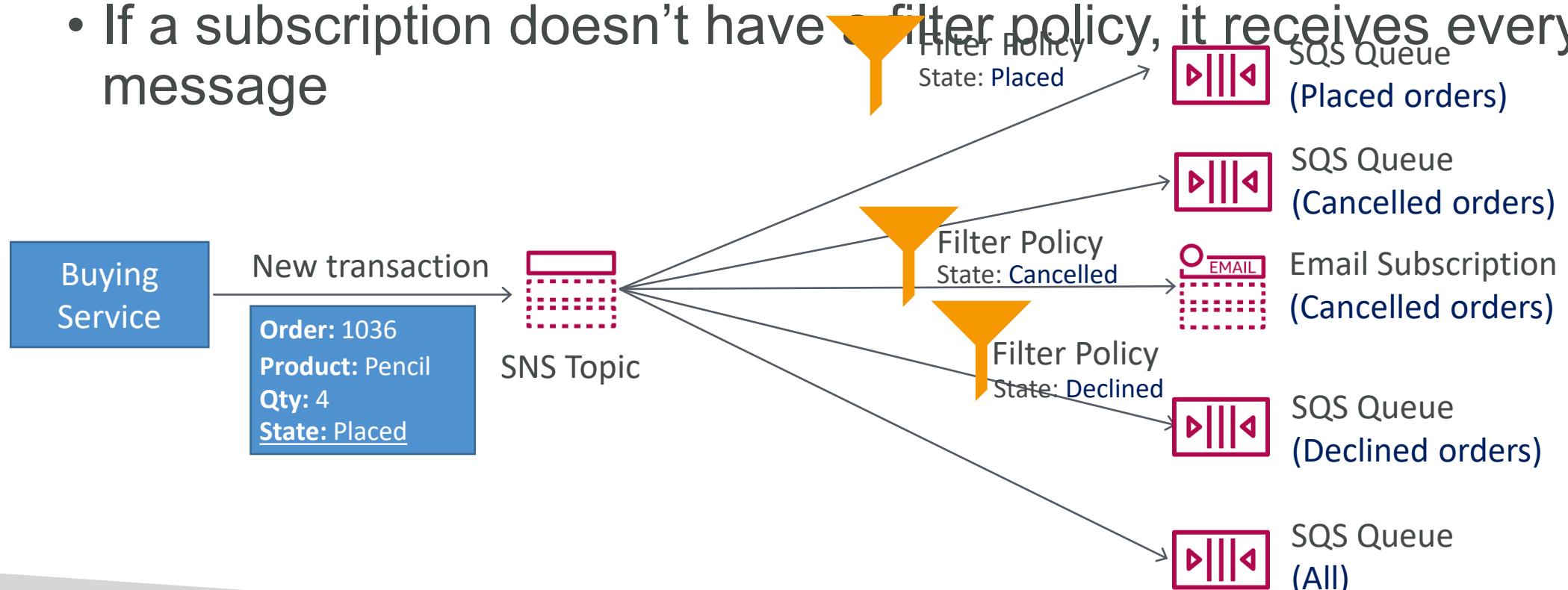
# SNS FIFO + SQS FIFO: Fan Out

- In case you need fan out + ordering + deduplication

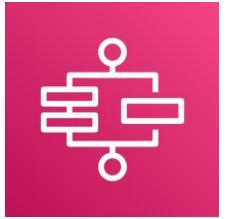


# SNS – Message Filtering

- JSON policy used to filter messages sent to SNS topic's subscriptions
- If a subscription doesn't have a filter policy, it receives every message



# AWS Step Functions



- **Use to design workflows**
- Easy visualizations
- Advanced Error Handling and Retry mechanism outside the code
- Audit of the history of workflows
- Ability to “Wait” for an arbitrary amount of time
- Max execution time of a State Machine is 1 year

# Step Functions – Examples

## Train a Machine Learning Model

**Definition**

Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). [Learn more](#)

```

1  {
2    "StartAt": "Generate dataset",
3    "States": {
4      "Generate dataset": {
5        "Resource": "<GENERATE_LAMBDA_FUNCTION_ARN>",
6        "Type": "Task",
7        "Next": "Train model (XGBoost)"
8      },
9      "Train model (XGBoost)": {
10        "Resource": "arn:
<PARTITION>:states:::sagemaker:createTrainingJob.sync",
11        "Parameters": {
12          "AlgorithmSpecification": {
13            "TrainingImage": "<SAGEMAKER_TRAINING_IMAGE>",
14            "TrainingInputMode": "File"
15          },
16          "OutputDataConfig": {
17            "S3OutputPath": "s3://<S3_BUCKET>/models"
18          },
19          "StoppingCondition": {
20            "MaxRuntimeInSeconds": 86400
21          },
22          "ResourceConfig": {
23            "InstanceCount": 1,
24            "InstanceType": "ml.m4.xlarge",

```

```

graph TD
    Start((Start)) --> Generate[Generate dataset]
    Generate --> Train[Train model (XGBoost)]
    Train --> Save[Save Model]
    Save --> Batch[Batch transform]
    Batch --> End((End))

```

# Step Functions – Examples

## Tune a Machine Learning Model

**Definition**  
Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). [Learn more](#)

```

1  {
2      "StartAt": "Generate Training Dataset",
3      "States": {
4          "Generate Training Dataset": {
5              "Resource": "<GENERATE_LAMBDA_FUNCTION_ARN>",
6              "Type": "Task",
7              "Next": "HyperparameterTuning (XGBoost)"
8          },
9          "HyperparameterTuning (XGBoost)": {
10             "Resource": "arn:
<PARTITION>:states:::sagemaker:createHyperParameterTuningJob.sync",
11             "Parameters": {
12                 "HyperParameterTuningJobName.$": "
<JOB_NAME_FROM_LAMBDA>",
13                 "HyperParameterTuningJobConfig": {
14                     "Strategy": "Bayesian",
15                     "HyperParameterTuningJobObjective": {
16                         "Type": "Minimize",
17                         "MetricName": "validation:rmse"
18                     },
19                     "ResourceLimits": {
20                         "MaxNumberOfTrainingJobs": 2,
21                         "MaxParallelTrainingJobs": 2
22                     },
23                     "ParameterRanges": {

```

The screenshot shows the AWS Step Functions console. On the left, there is a code editor window displaying the ASL code for the state machine. On the right, there is a visual representation of the state machine with nodes: Start, Generate Training Dataset, HyperparameterTuning (XGBoost), Extract Model Path, HyperparameterTuning - Save Model, Extract Model Name, Batch transform, and End. Arrows indicate the flow between these nodes. To the left of the code editor, there are buttons for creating (+), deleting (-), and cloning (C) the state machine.

# Step Functions – Examples

## Manage a Batch Job

**Definition**

Code is pre-configured by the chosen sample project. It can be edited after creation. Step Functions state machines are defined using the JSON-based Amazon States Language (ASL). [Learn more](#)

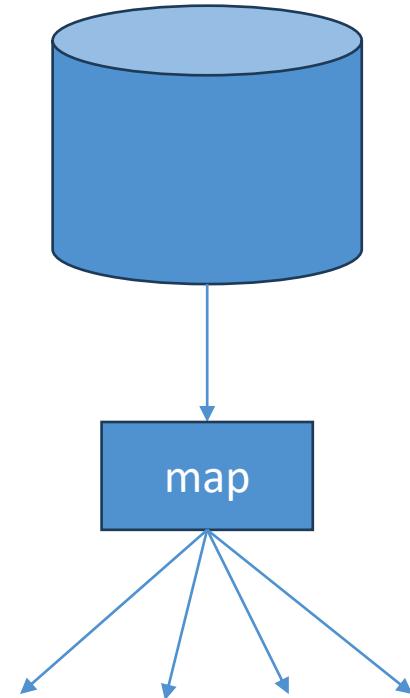
```
1  {
2      "Comment": "An example of the Amazon States Language for
3          notification on an AWS Batch job completion",
4      "StartAt": "Submit Batch Job",
5      "TimeoutSeconds": 3600,
6      "States": {
7          "Submit Batch Job": {
8              "Type": "Task",
9              "Resource": "arn:<PARTITION>:states:::batch:submitJob.sync",
10             "Parameters": {
11                 "JobName": "BatchJobNotification",
12                 "JobQueue": "<BATCH_QUEUE_ARN>",
13                 "JobDefinition": "<BATCH_JOB_DEFINITION_ARN>"
14             },
15             "Next": "Notify Success",
16             "Catch": [
17                 {
18                     "ErrorEquals": [ "States.ALL" ],
19                     "Next": "Notify Failure"
20                 }
21             ],
22             "Notify Success": {
23                 "Type": "Task",
24                 "Resource": "arn:<PARTITION>:states:::sns:publish",
```

**Diagram**

```
graph TD
    Start((Start)) --> Submit[Submit Batch Job]
    Submit --> Success[Notify Success]
    Submit --> Failure[Notify Failure]
    Success --> End((End))
    Failure --> End
```

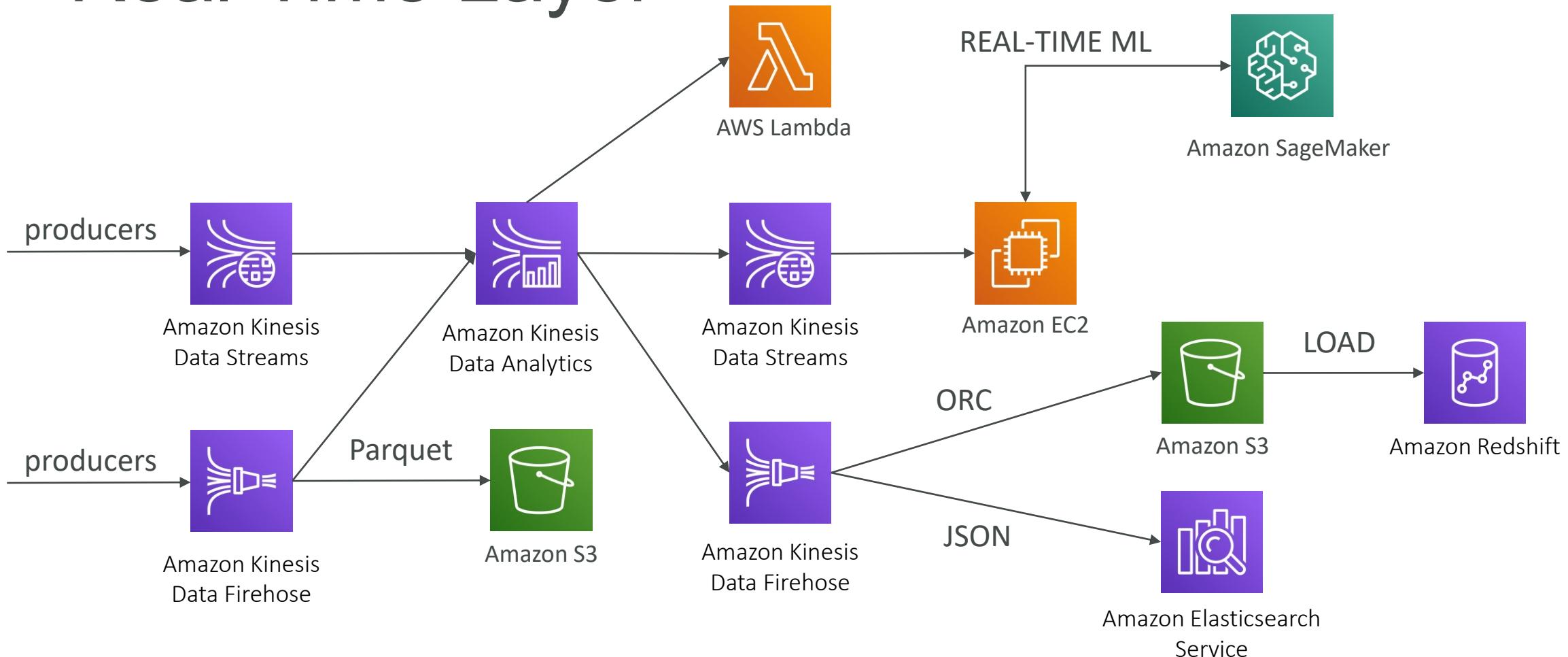
# AWS Step Functions

- Your workflow is called a *state machine*
- Each step in a workflow is a *state*
- Types of states
  - **Task:** Does something with Lambda, other AWS services, or third party API's
  - **Choice:** Adds conditional logic via Choice Rules (ie, comparisons)
  - **Wait:** Delays state machine for a specified time
  - **Parallel:** Add separate branches of execution
  - **Map:** Run a set of steps for each item in a dataset, in parallel
    - **This is most relevant to data engineering!**  
**Works with JSON, S3 objects, CSV files**
    - **Also Pass, Succeed, Fail**



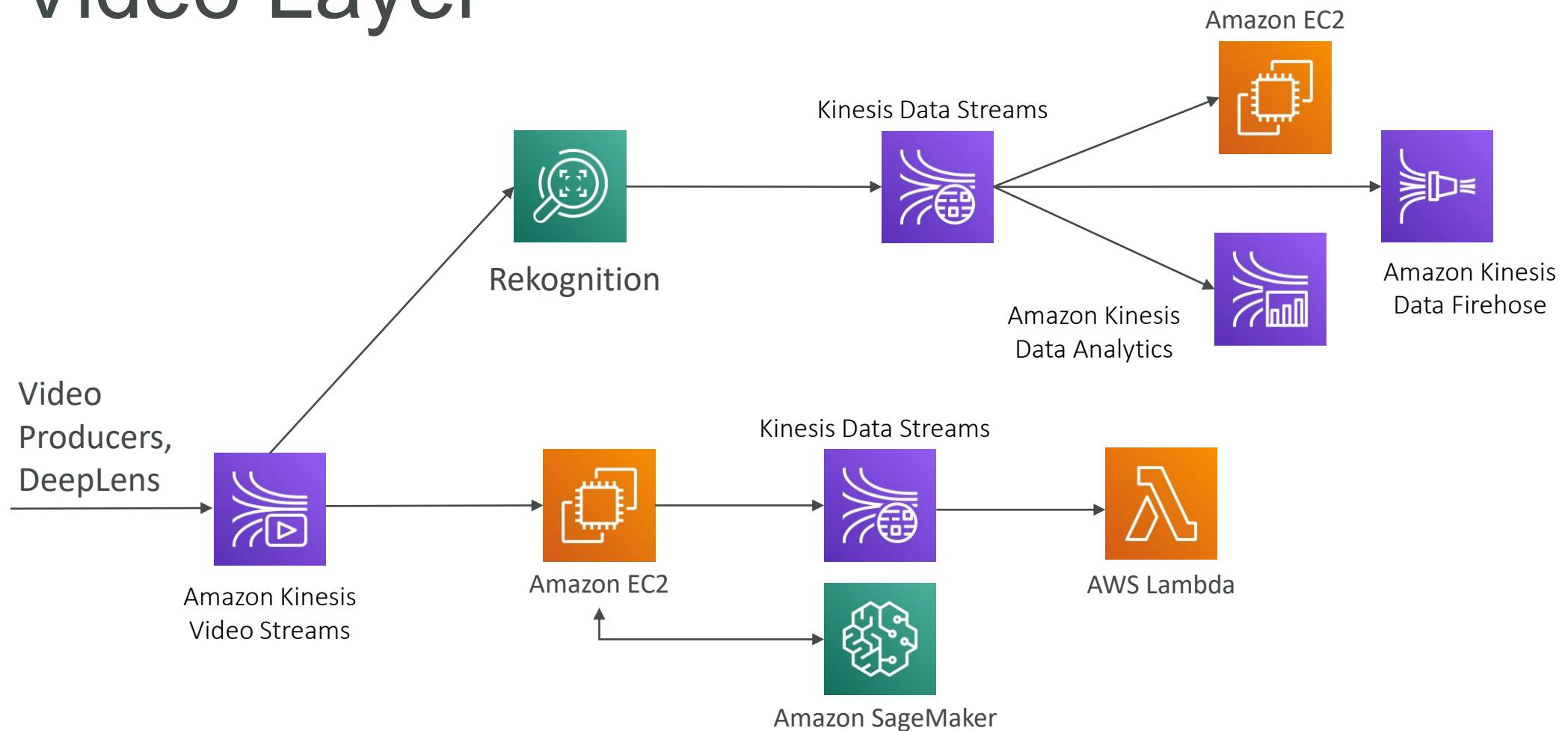
# Full Data Engineering Pipeline

## Real-Time Layer



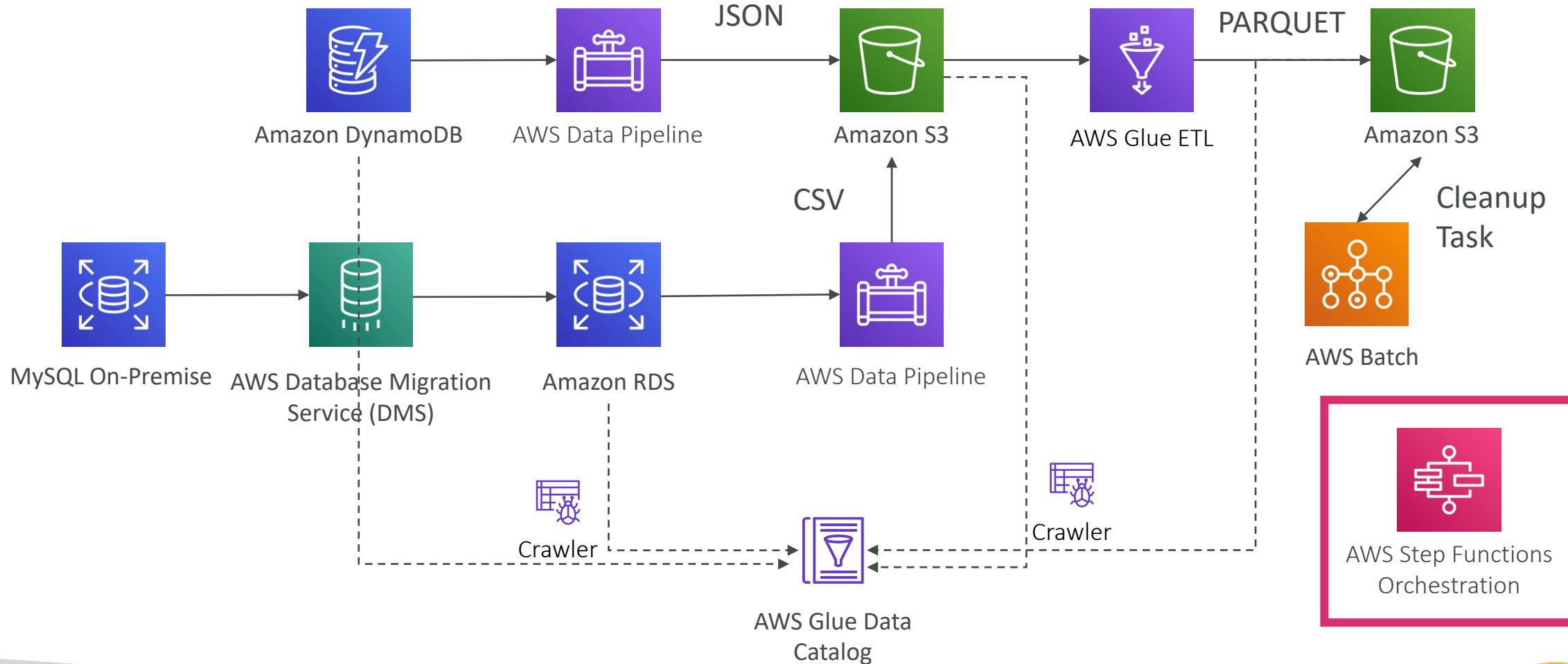
# Full Data Engineering Pipeline

## Video Layer



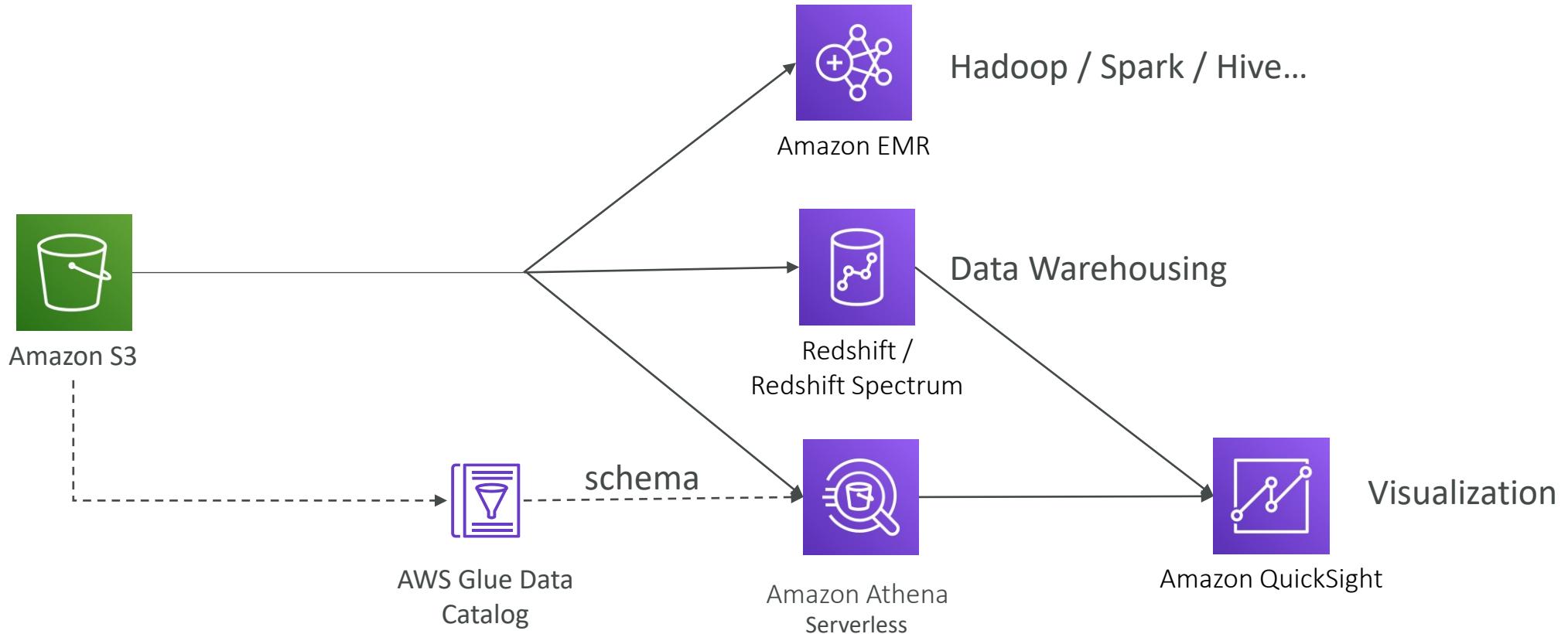
# Full Data Engineering Pipeline

## Batch Layer



# Full Data Engineering Pipeline

## Analytics layer



# Amazon AppFlow

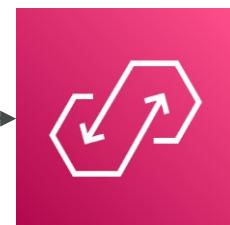


- Fully managed integration service that enables you to securely transfer data between **Software-as-a-Service (SaaS) applications and AWS**
- Sources: **Salesforce**, SAP, Zendesk, Slack, and ServiceNow
- Destinations: AWS services like **Amazon S3**, **Amazon Redshift** or non-AWS such as SnowFlake 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**

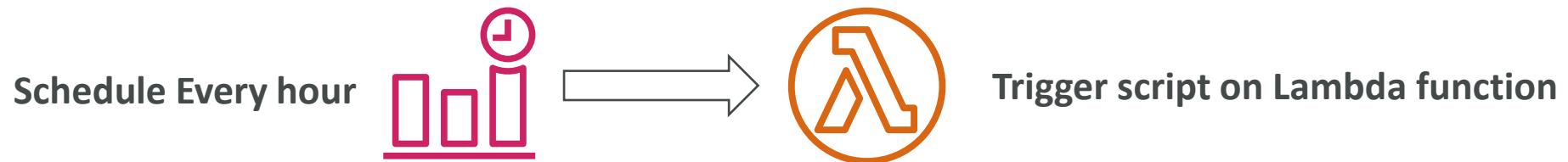
## Destinations

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	

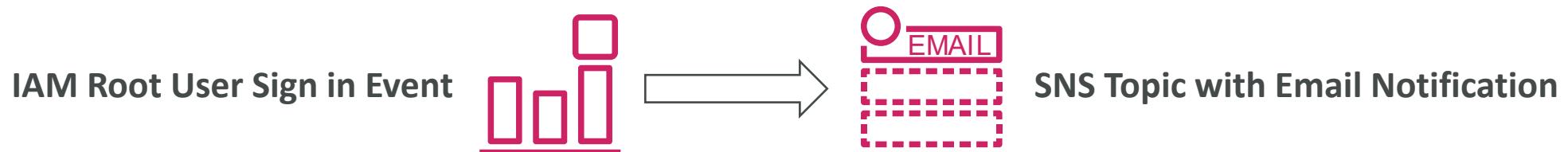
# Amazon EventBridge (formerly CloudWatch Events)



- Schedule: Cron jobs (scheduled scripts)

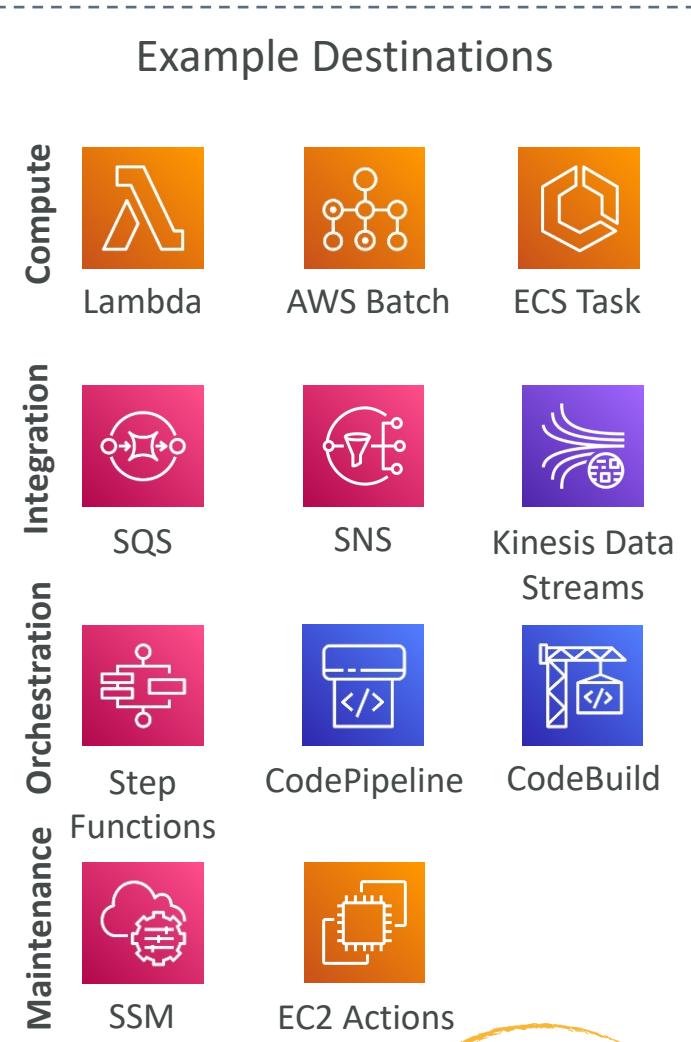
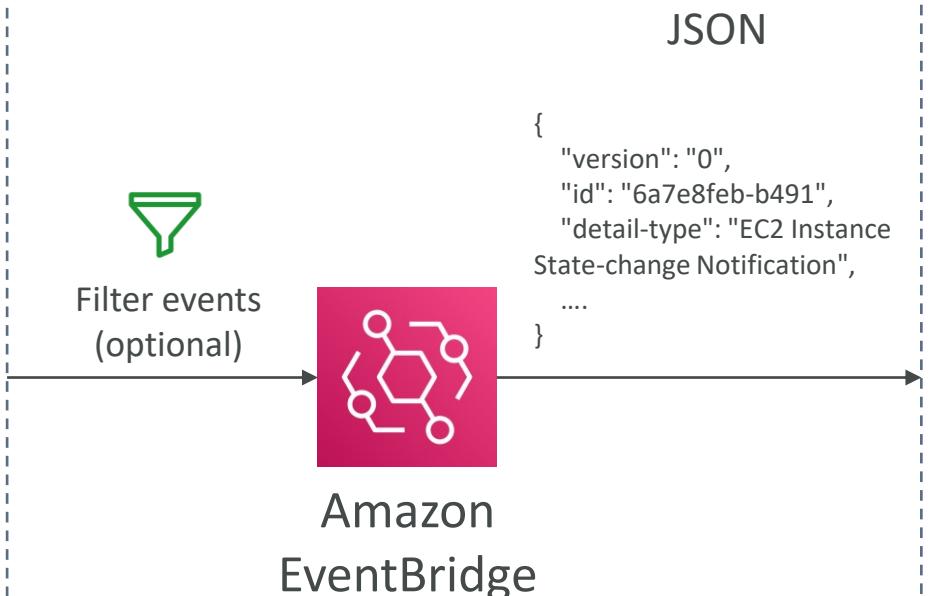
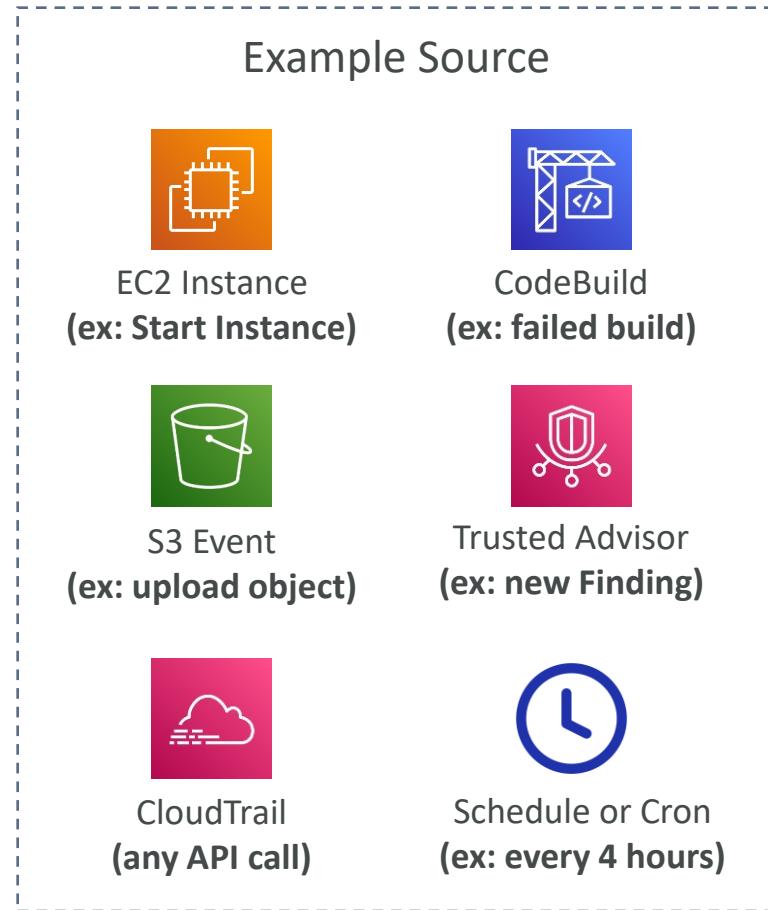


- Event Pattern: Event rules to react to a service doing something

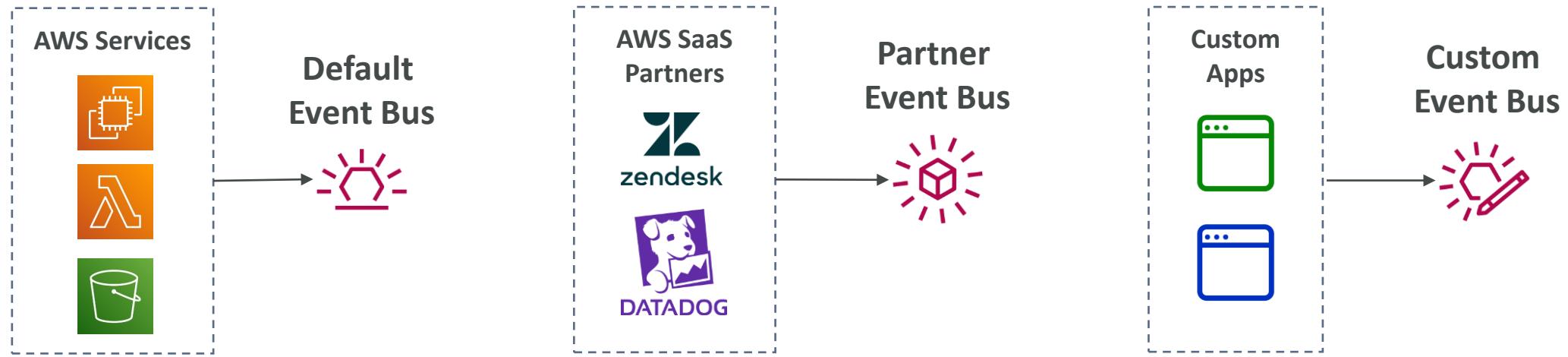


- Trigger Lambda functions, send SQS/SNS messages...

# Amazon EventBridge Rules



# Amazon EventBridge



- Event buses can be accessed by other AWS accounts using Resource-based Policies
- You can **archive events** (all/filter) sent to an event bus (indefinitely or set period)
- Ability to **replay archived events**

# Amazon EventBridge – Schema Registry

- EventBridge can analyze the events in your bus and infer the **schema**
- The **Schema Registry** allows you to generate code for your application, that will know in advance how data is structured in the event bus
- Schema can be versioned

The screenshot shows the AWS Schema Registry interface. At the top, it displays the schema name: `aws.codepipeline@CodePipelineActionExecutionStateChange`. Below this, the "Schema details" section provides the following information:

Schema name	Last modified	Schema ARN
<code>aws.codepipeline@CodePipelineActionExecutionStateChange</code>	Dec 1, 2019, 12:11 AM GMT	-
Description	Schema for event type CodePipelineActionExecutionStateChange, published by AWS service aws.codepipeline	Schema registry aws.events Number of versions 1 Schema type OpenAPI 3.0

Below the schema details, there is a section for "Version 1" (Created on Dec 1, 2019, 12:11 AM GMT). It includes an "Action" dropdown and a "Download code bindings" button. The schema definition is shown as a JSON-like code block:

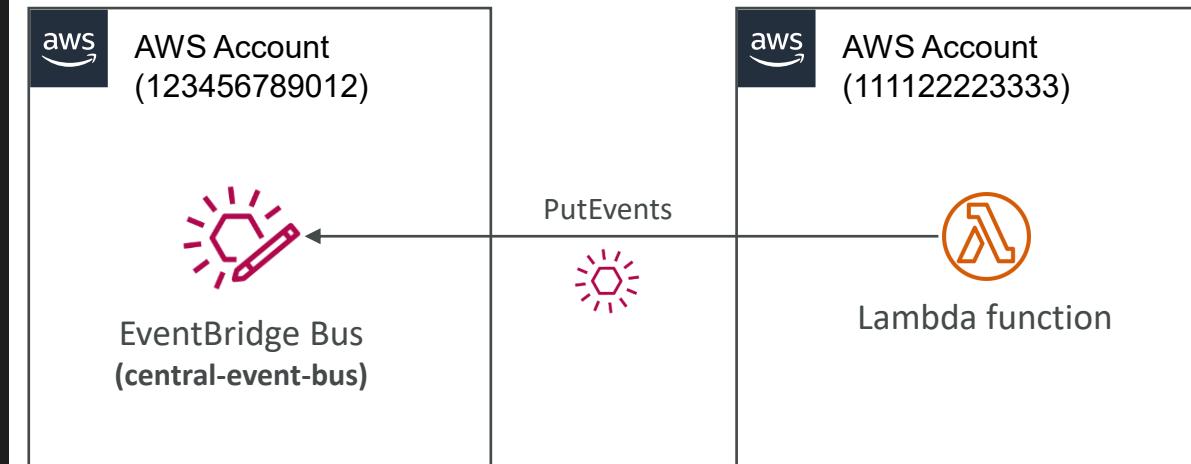
```
1 {
2   "openapi": "3.0.0",
3   "info": {
4     "version": "1.0.0",
5     "title": "CodePipelineActionExecutionStateChange"
6   },
7   "paths": {},
8   "components": {
9     "schemas": {
10       "AWSEvent": {
```

# Amazon EventBridge – Resource-based Policy

- Manage permissions for a specific Event Bus
- Example: allow/deny events from another AWS account or AWS region
- Use case: aggregate all events from your AWS Organization in a single AWS account or AWS region

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "events:PutEvents",  
            "Principal": { "AWS": "111122223333" },  
            "Resource": "arn:aws:events:us-east-1:123456789012:  
event-bus/central-event-bus"  
        }  
    ]  
}
```

Allow **events** from another AWS account

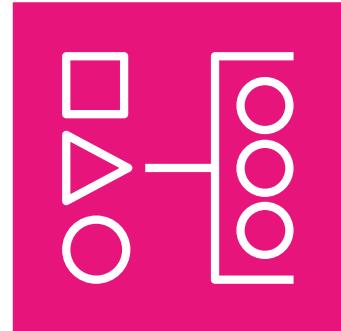


# Amazon Managed Workflows for Apache Airflow (MWAA)

- Apache Airflow is batch-oriented workflow tool
- Develop, schedule, and monitor your workflows
- Workflows are defined as Python code that creates a Directed Acyclic Graph (DAG)
- Amazon MWAA provides a managed service for Apache Airflow so you don't have to deal with installing or maintaining it
- Use cases:
  - Complex workflows
  - ETL coordination
  - Preparing ML data



Apache  
Airflow



Amazon Managed Workflows  
for Apache Airflow

# Sample DAG in Airflow

```
from datetime import datetime

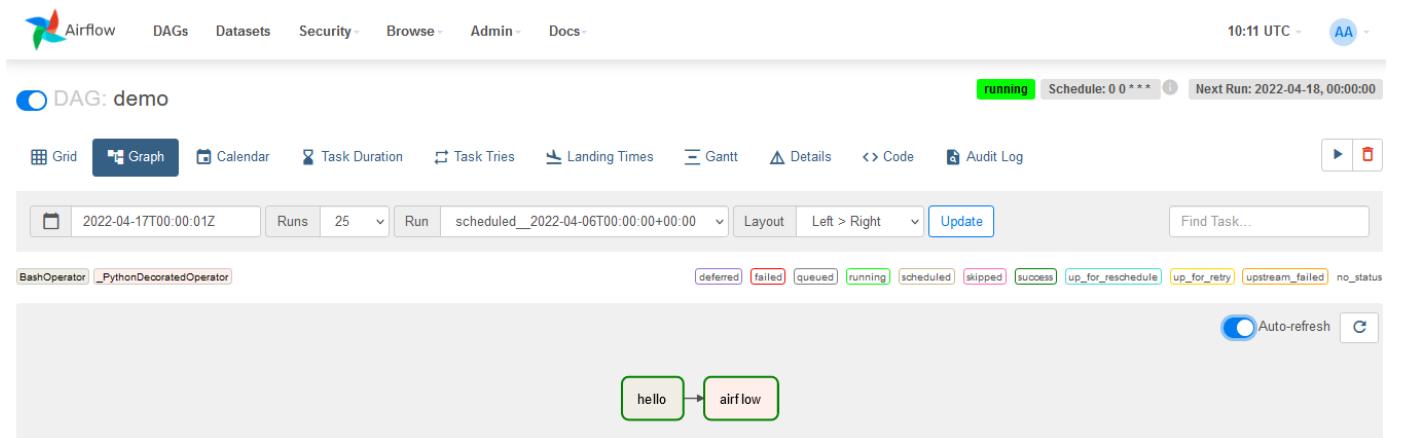
from airflow import DAG
from airflow.decorators import task
from airflow.operators.bash import BashOperator

# A DAG represents a workflow, a collection of tasks
with DAG(dag_id="demo", start_date=datetime(2022, 1, 1),
schedule="0 0 * * *") as dag:

    # Tasks are represented as operators
    hello = BashOperator(task_id="hello", bash_command="echo
hello")

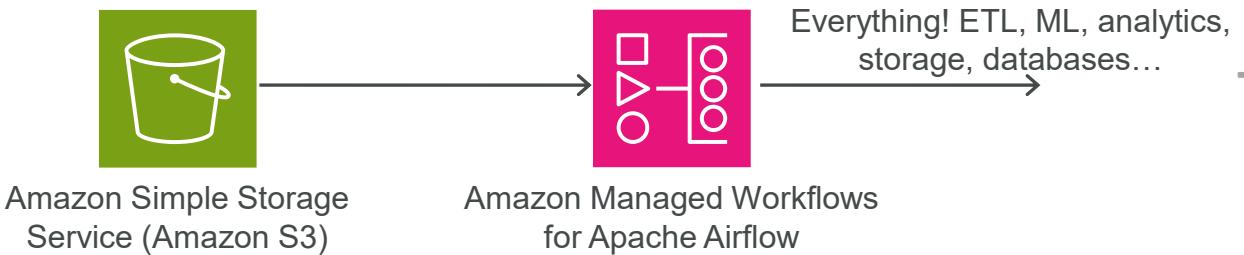
    @task()
    def airflow():
        print("airflow")

    # Set dependencies between tasks
    hello >> airflow()
```



# Airflow + MWAA

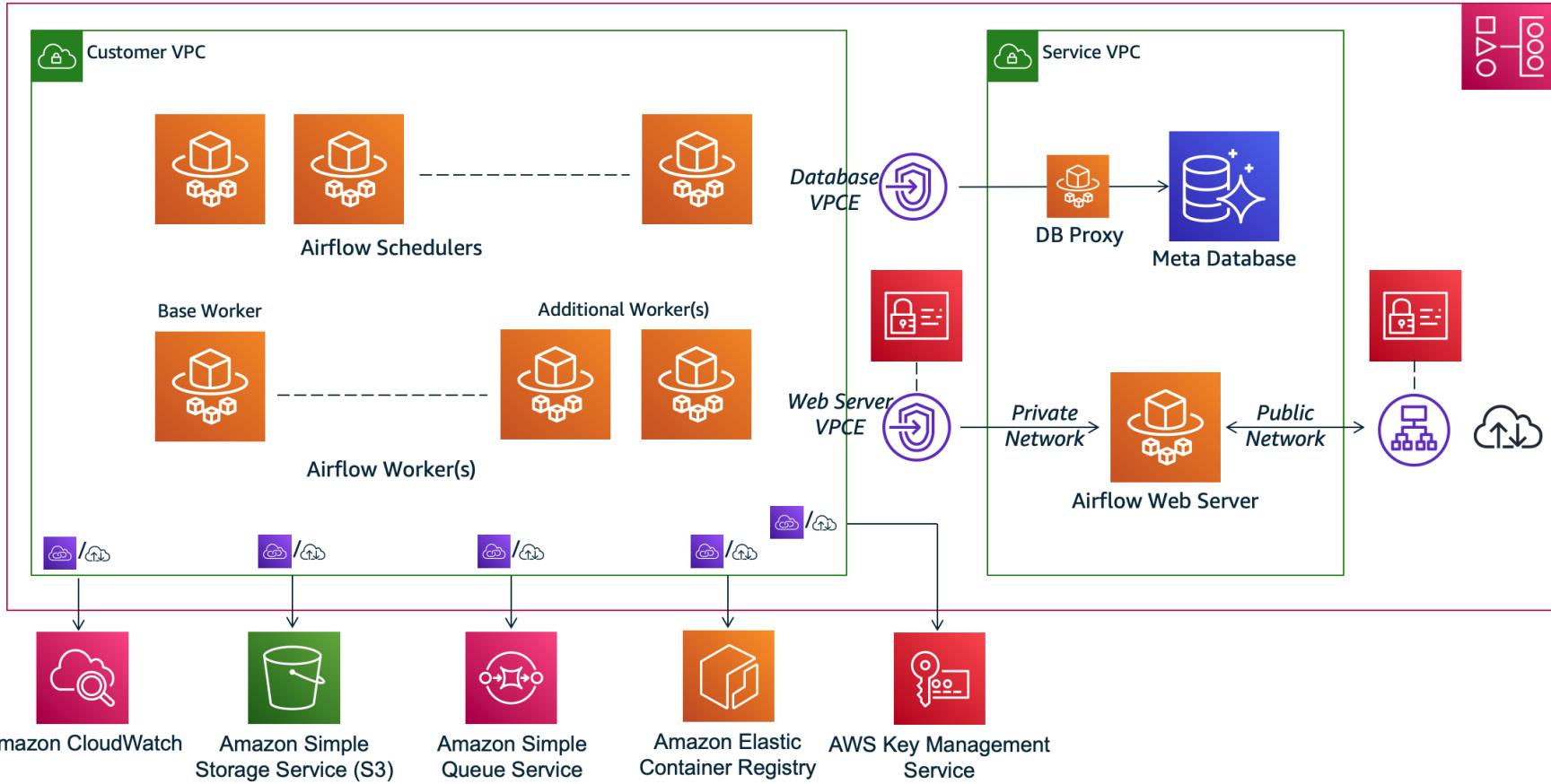
- Your DAGs (Python code) are uploaded into S3
  - May also zip it together with required plugins and requirements
- Amazon MWAA picks it up, and orchestrates and schedules the pipelines defined by each DAG.
- Runs within a VPC
  - In at least 2 AZ's
- Private or public endpoints
  - IAM-managed
  - (Access to Airflow Web Server)
- Automatic scaling
  - Airflow Workers autoscale up to the limits you define



# Amazon MWAA Integration

- Leverages open-source integrations
  - Athena, Batch, CloudWatch, DynamoDB, DataSync
  - EMR, Fargate, EKS, Kinesis, Glue, Lambda
  - Redshift, SQS, SNS, SageMaker, S3... and more
  - Security services (AWS Secrets Manager, etc)
- Schedulers and Workers are AWS Fargate containers

# Amazon MWAA Architecture



# Security, Identity, and Compliance

Securing data in AWS

# Principle of Least Privilege

- Grant only the permissions required to perform a task
- Start with broad permissions while developing
- But lock it down once you have a better idea of the exact services and operations a workload requires
- Can use IAM Access Analyzer to generate least-privilege policies based on access activity

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowListBucket",  
      "Effect": "Allow",  
      "Action": "s3>ListBucket",  
      "Resource": "arn:aws:s3:::my-specific-bucket",  
      "Condition": {  
        "StringLike": {  
          "s3:prefix": "data/reports/*"  
        }  
      }  
    },  
    {  
      "Sid": "AllowReadCSVFiles",  
      "Effect": "Allow",  
      "Action": [  
        "s3.GetObject",  
        "s3.GetObjectVersion"  
      ],  
      "Resource": "arn:aws:s3:::my-specific-bucket/data/reports/*.csv"  
    }  
  ]  
}
```

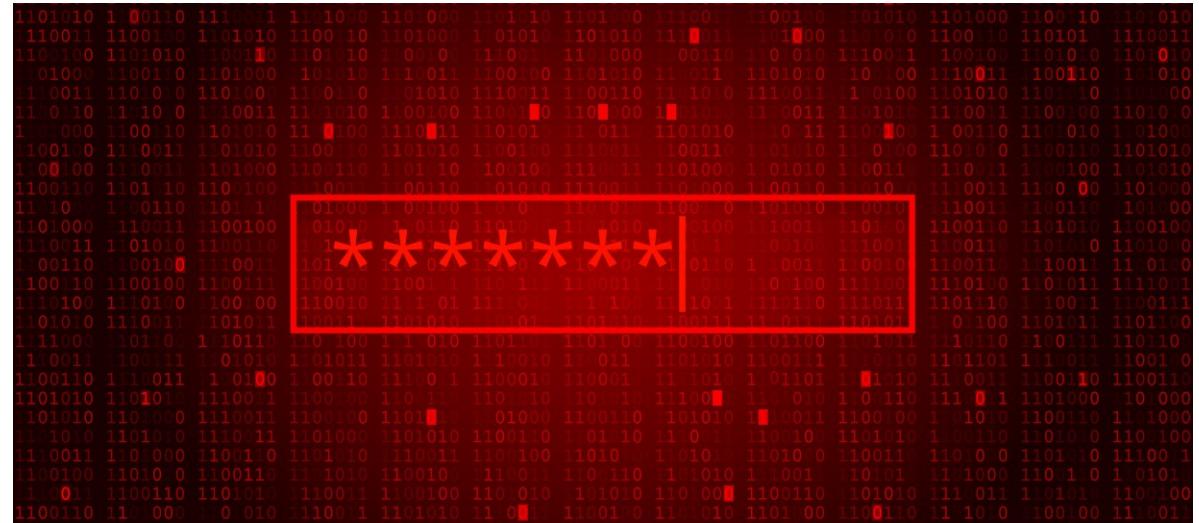
# Data Masking and Anonymization

- Dealing with PII or other sensitive data
- Masking obfuscates data
  - For example, masking all but the last 4 digits of a credit card or social security number
  - Masking passwords
  - Supported in Glue DataBrew and Redshift

--create a masking policy that fully masks the credit card number  
CREATE MASKING POLICY mask\_credit\_card\_full WITH (credit\_card  
VARCHAR(256)) USING ('00000XXXX0000'::TEXT);

# Data Masking and Anonymization

- Anonymization techniques
  - Replace with random
  - Shuffle
  - Encrypt (deterministic or probabilistic)
  - Hashing
- Or just delete it or don't import it in the first place.



# Key Salting

- Salting involves appending or prepending a random value, known as a "salt," to a piece of data (often a password) before hashing it.
- Prevents pre-computed "rainbow table" attacks where adversaries use pre-generated hashes of commonly used passwords to find matches.
- Ensures that the same piece of data (like two identical passwords) doesn't produce the same hash across different instances due to the unique salt.
- Use strong, cryptographically secure random values for salts
- Rotate salts periodically
- Each user should have a unique salt
- Salt and hash passwords before storing

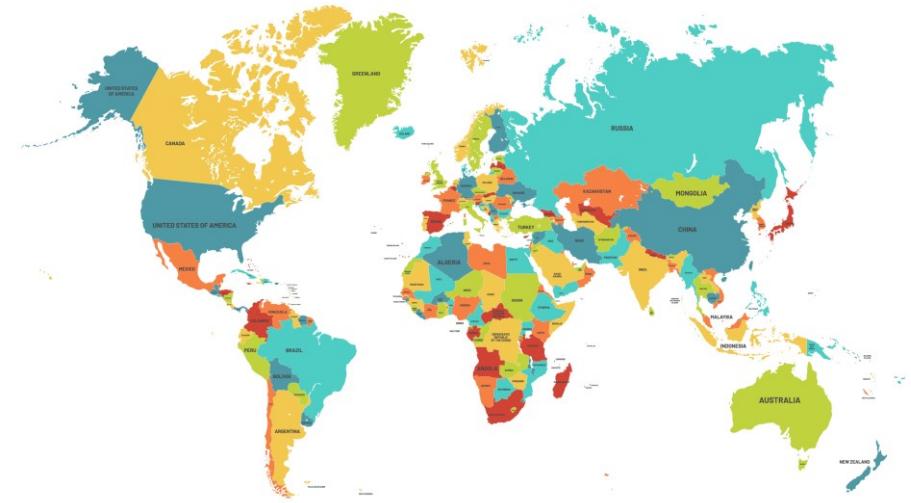


# Key Salting: Example

Username	Salt value	String to be hashed	Hashed value = <u>SHA256</u> (Password + Salt value)
user1	D;%yL9TS:5PaIs/d	<b>password123</b> D;%yL9TS:5PaIs/d	9C9B913EB1B6254F4737 CE947EFD16F16E916F9D6 EE5C1102A2002E48D4C8 8BD
user2	)<,-<U(jLezy4j>*	<b>password123</b> )<,-<U(jLezy4j>*	6058B4EB46BD6487298B 59440EC8E70EAE482239F F2B4E7CA69950DFBD553 2F2

# Keeping Data Where it Belongs

- Compliance may prohibit data from being transferred into certain regions
- It's all too easy for backups or replication to send data to a region where it is not allowed
- AWS Organizations' Service Control Policies are one way to enforce this
- Restrict IAM policies and S3 bucket policies to allowed regions
- Take care when configuring backups & replication for RDS, Aurora, Redshift
- Monitor and alarm with CloudTrail / CloudWatch

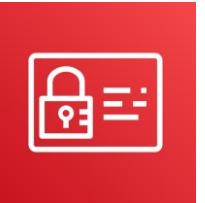


# Keeping Data Where it Belongs

- Example: IAM Policy to restrict RDS to backups in us-east-1 and us-west-1

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "AllowRDSActionsInSpecificRegions",  
            "Effect": "Allow",  
            "Action": [  
                "rds>CreateDBSnapshot",  
                "rds:CopyDBSnapshot"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "aws:RequestedRegion": [  
                        "us-west-1",  
                        "us-east-1"  
                    ]  
                }  
            }  
        },  
        {  
            "Sid": "AllowOtherRDSActions",  
            "Effect": "Allow",  
            "Action": "rds:*",  
            "Resource": "*",  
            "NotAction": [  
                "rds>CreateDBSnapshot",  
                "rds:CopyDBSnapshot"  
            ]  
        }  
    ]  
}
```

# IAM: Users & Groups



- IAM = Identity and Access Management, **Global** service
- **Root account** created by default, shouldn't be used or shared
- **Users** are people within your organization, and can be grouped
- **Groups** only contain users, not other groups
- Users don't have to belong to a group, and user can belong to multiple groups



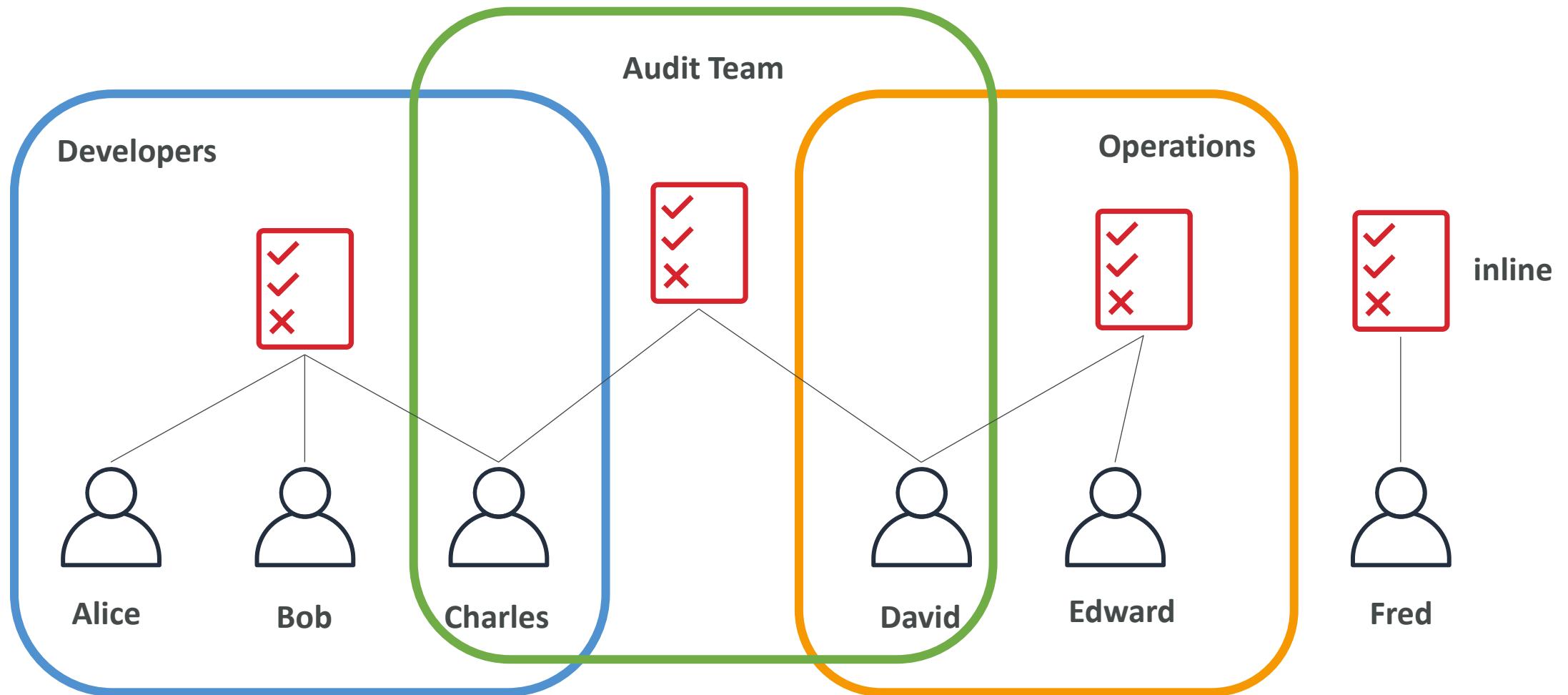
# IAM: Permissions

- **Users or Groups** can be assigned JSON documents called policies
- These policies define the **permissions** of the users
- In AWS you apply the **least privilege principle**: don't give more permissions than a user needs

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "ec2:Describe*",  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": "elasticloadbalancing:Describe*",  
      "Resource": "*"  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "cloudwatch:ListMetrics",  
        "cloudwatch:GetMetricStatistics",  
        "cloudwatch:Describe*"  
      ],  
      "Resource": "*"  
    }  
  ]  
}
```



# IAM Policies inheritance



# IAM Policies Structure

- Consists of
  - **Version:** policy language version, always include "2012-10-17"
  - **Id:** an identifier for the policy (optional)
  - **Statement:** one or more individual statements (required)
- Statements consists of
  - **Sid:** an identifier for the statement (optional)
  - **Effect:** whether the statement allows or denies access (Allow, Deny)
  - **Principal:** account/user/role to which this policy applied to
  - **Action:** list of actions this policy allows or denies
  - **Resource:** list of resources to which the actions applied to
  - **Condition:** conditions for when this policy is in effect (optional)

```
{
  "Version": "2012-10-17",
  "Id": "S3-Account-Permissions",
  "Statement": [
    {
      "Sid": "1",
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::123456789012:root"]
      },
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::mybucket/*"]
    }
  ]
}
```

# IAM – Password Policy

- Strong passwords = higher security for your account
- In AWS, you can setup a password policy:
  - Set a minimum password length
  - Require specific character types:
    - including uppercase letters
    - lowercase letters
    - numbers
    - non-alphanumeric characters
  - Allow all IAM users to change their own passwords
  - Require users to change their password after some time (password expiration)
  - Prevent password re-use

# Multi Factor Authentication - MFA



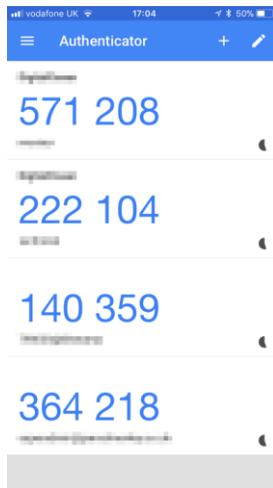
- Users have access to your account and can possibly change configurations or delete resources in your AWS account
- **You want to protect your Root Accounts and IAM users**
- MFA = password you know + security device you own



- **Main benefit of MFA:**  
if a password is stolen or hacked, the account is not compromised

# MFA devices options in AWS

## Virtual MFA device



Google Authenticator  
(phone only)

Support for multiple tokens on a single device.

## Universal 2nd Factor (U2F) Security Key



YubiKey by Yubico (3<sup>rd</sup> party)

Support for multiple root and IAM users  
using a single security key

# MFA devices options in AWS

## Hardware Key Fob MFA Device



Provided by Gemalto (3<sup>rd</sup> party)

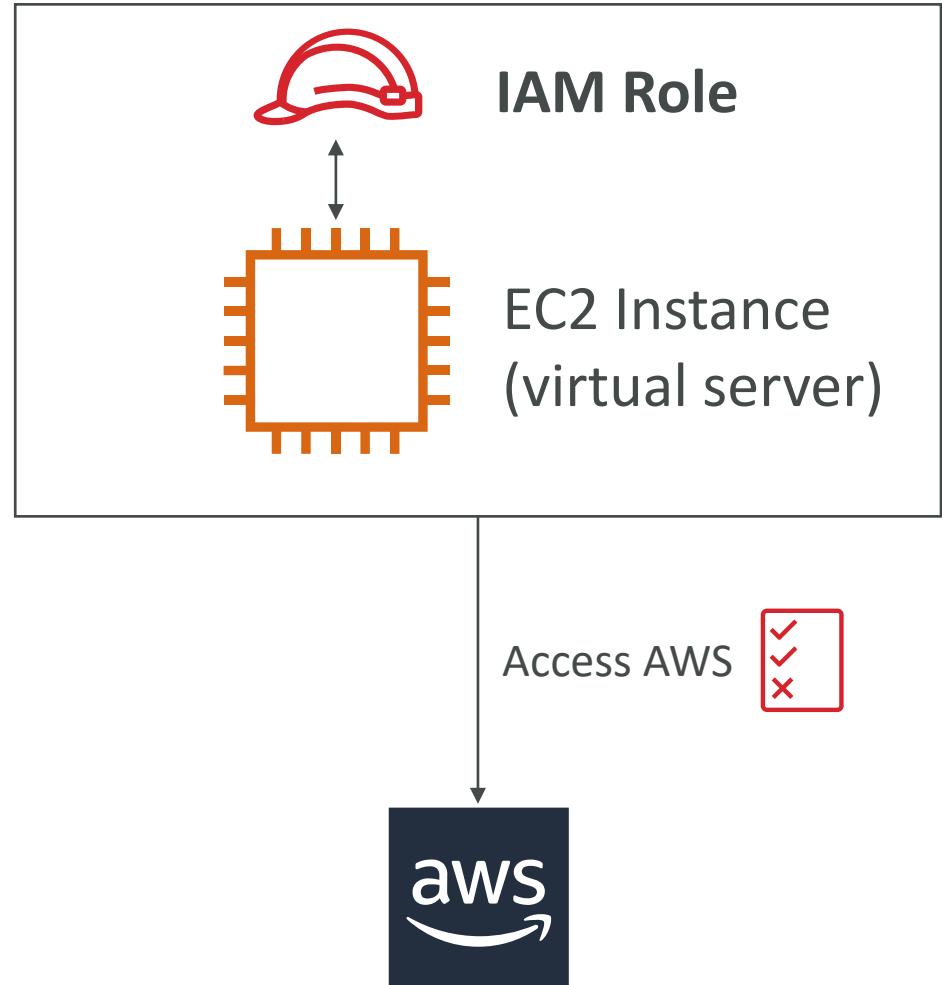
## Hardware Key Fob MFA Device for AWS GovCloud (US)



Provided by SurePassID (3<sup>rd</sup> party)

# IAM Roles for Services

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign **permissions** to AWS services with **IAM Roles**
- Common roles:
  - EC2 Instance Roles
  - Lambda Function Roles
  - Roles for CloudFormation



# Why encryption?

## Encryption in flight (TLS / SSL)

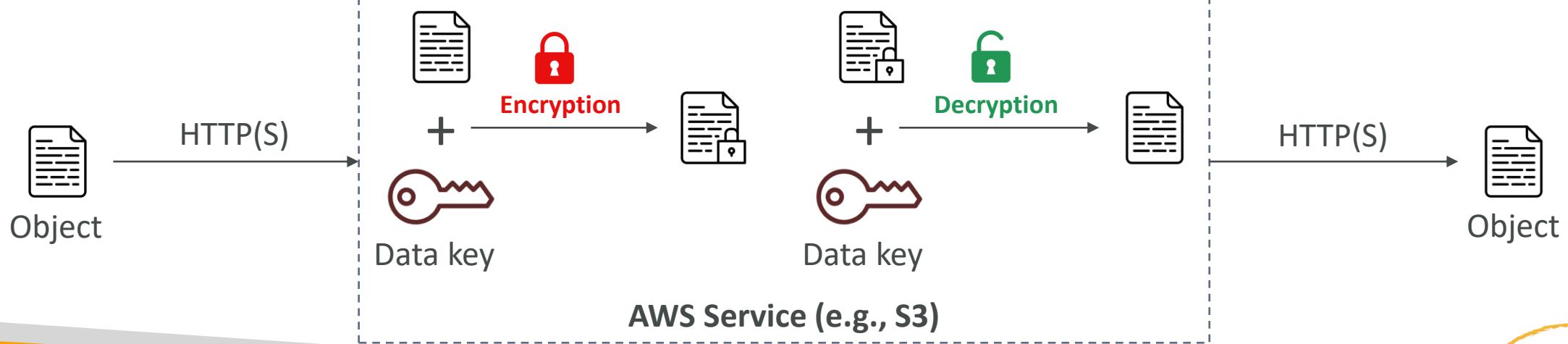
- Data is encrypted before sending and decrypted after receiving
- TLS certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen



# Why encryption?

## Server-side encryption at rest

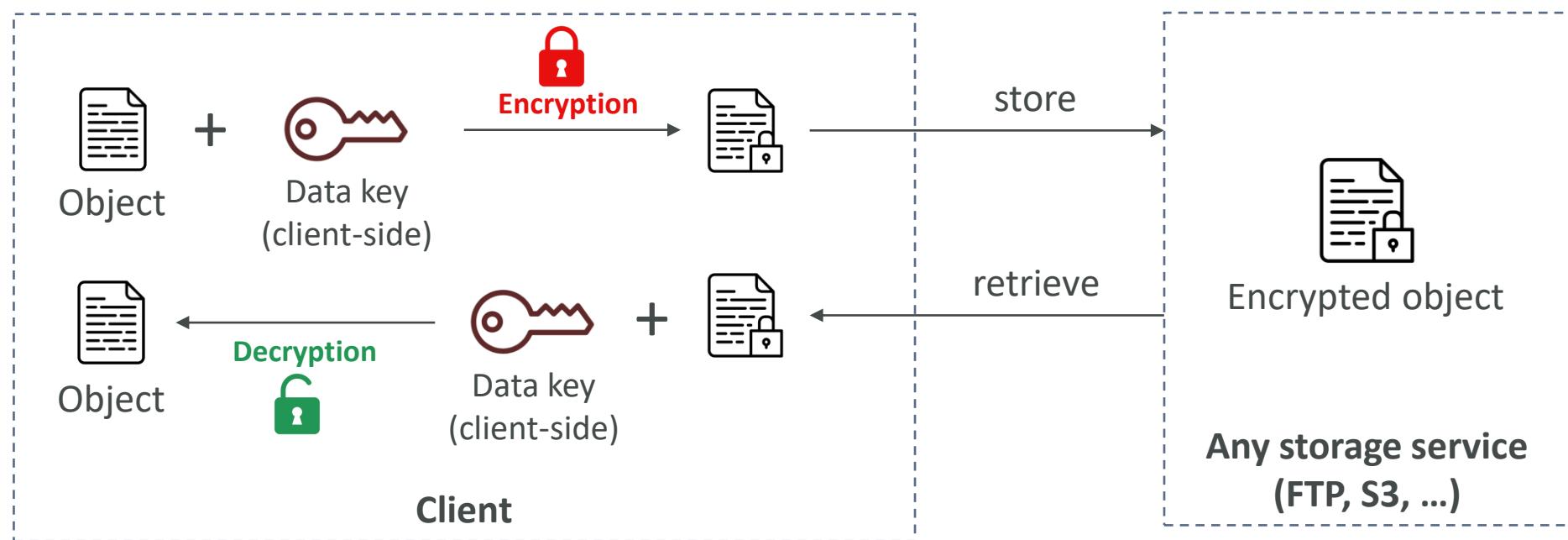
- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere, and the server must have access to it



# 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



# AWS KMS (Key Management Service)



- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- AWS manages encryption keys for us
- Fully integrated with IAM for authorization
- Easy way to control access to your data
- Able to audit KMS Key usage using CloudTrail
- Seamlessly integrated into most AWS services (EBS, S3, RDS, SSM...)
- **Never ever store your secrets in plaintext, especially in your code!**

# KMS Keys Types

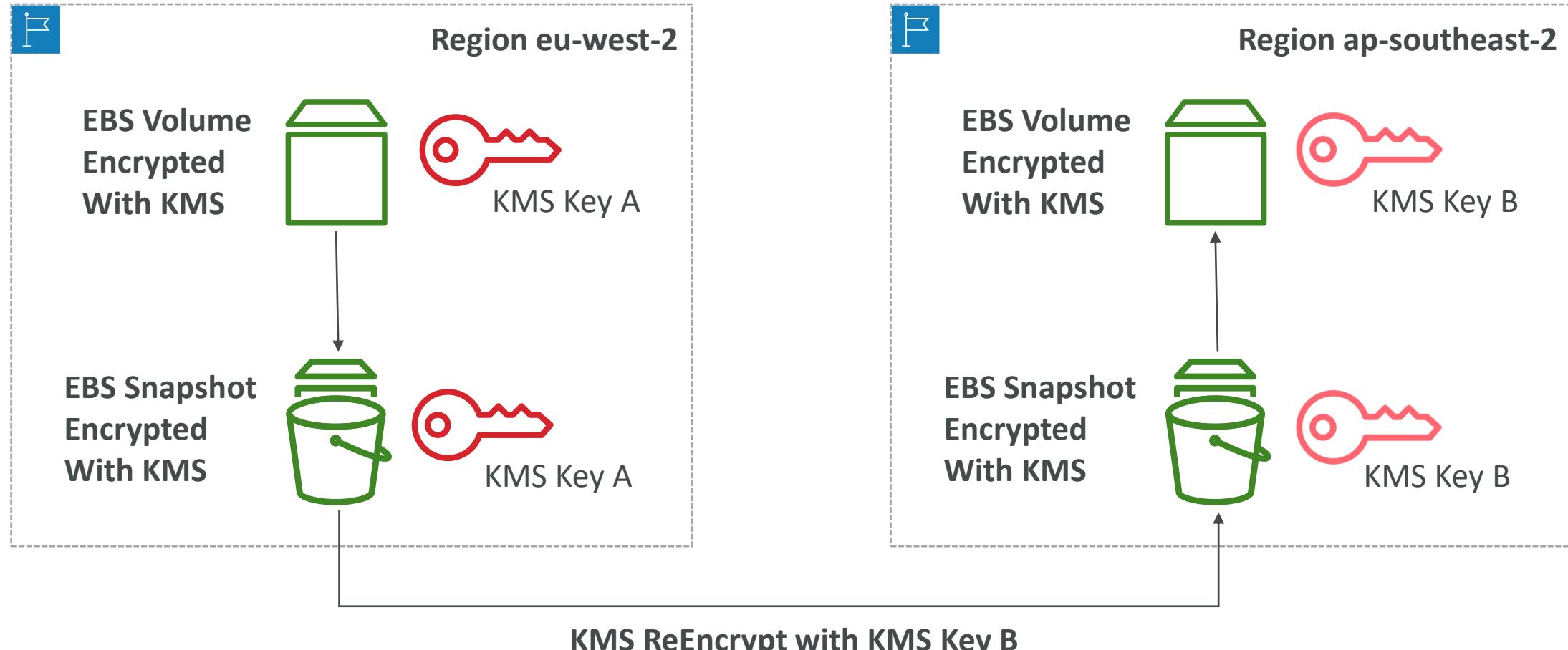
- **KMS Keys is the new name of KMS Customer Master Key**
- **Symmetric (AES-256 keys)**
  - Single encryption key that is used to Encrypt and Decrypt
  - AWS services that are integrated with KMS use Symmetric CMKs
  - You never get access to the KMS Key unencrypted (must call KMS API to use)
- **Asymmetric (RSA & ECC key pairs)**
  - Public (Encrypt) and Private Key (Decrypt) pair
  - Used for Encrypt/Decrypt, or Sign/Verify operations
  - The public key is downloadable, but you can't access the Private Key unencrypted
  - Use case: encryption outside of AWS by users who can't call the KMS API

# AWS KMS (Key Management Service)

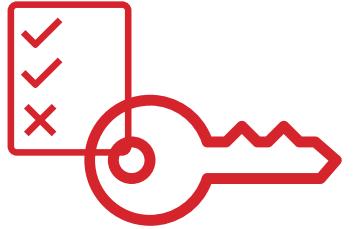


- Types of KMS Keys:
  - AWS Owned Keys (free): SSE-S3, SSE-SQS, SSE-DDB (default key)
  - AWS Managed Key: **free** (aws/service-name, example: aws/rds or aws/ebs)
  - Customer managed keys created in KMS: **\$1 / month**
  - Customer managed keys imported (must be **syI** Encryption key management **nth**)
    - + pay for API call to KMS (\$0.03 / 10000 calls)
      - Owned by Amazon DynamoDB
      - AWS managed key **Lea**  
Key alias: aws/dynamodb.
      - Stored in your account,  
and owned and managed by you
- Automatic Key rotation:
  - AWS-managed KMS Key: automatic every 1 year
  - Customer-managed KMS Key: (must be enabled) automatic every 1 year
  - Imported KMS Key: only manual rotation possible using alias

# Copying Snapshots across regions



# KMS Key Policies



- Control access to KMS keys, “similar” to S3 bucket policies
- Difference: you cannot control access without them
- **Default KMS Key Policy:**
  - Created if you don't provide a specific KMS Key Policy
  - Complete access to the key to the root user = entire AWS account
- **Custom KMS Key Policy:**
  - Define users, roles that can access the KMS key
  - Define who can administer the key
  - Useful for cross-account access of your KMS key

# Copying Snapshots across accounts

1. Create a Snapshot, encrypted with your own KMS Key (Customer Managed Key)
2. **Attach a KMS Key Policy to authorize cross-account access**
3. Share the encrypted snapshot
4. (in target) Create a copy of the Snapshot, encrypt it with a CMK in your account
5. Create a volume from the snapshot

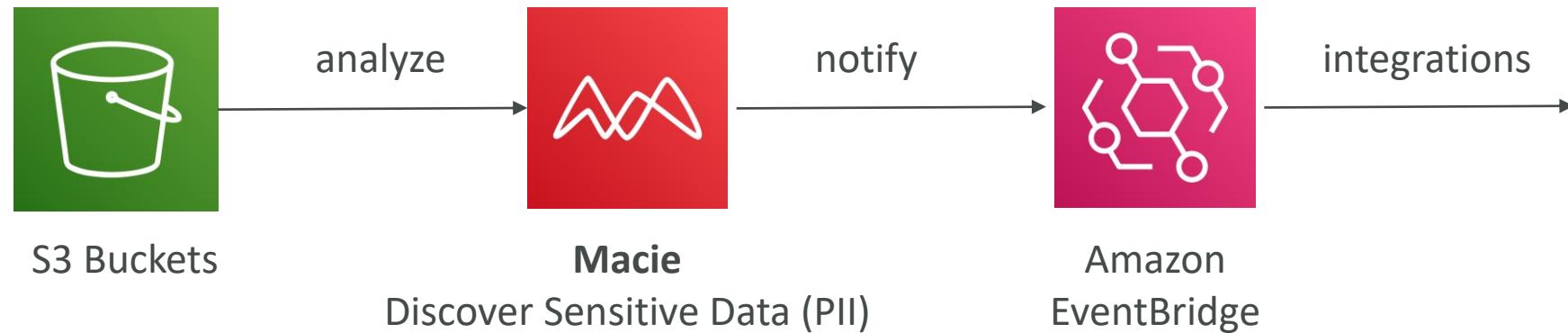
```
{  
  "Sid": "Allow use of the key with destination account",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::TARGET-ACCOUNT-ID:role/ROLENAMESPACE"  
  },  
  "Action": [  
    "kms:Decrypt",  
    "kms>CreateGrant"  
  ],  
  "Resource": "*",  
  "Condition": {  
    "StringEquals": {  
      "kms:ViaService": "ec2.REGION.amazonaws.com",  
      "kms:CallerAccount": "TARGET-ACCOUNT-ID"  
    }  
  }  
}
```

KMS Key Policy

# AWS Macie



- Amazon Macie is a fully managed data security and data privacy service that uses **machine learning and pattern matching to discover and protect your sensitive data in AWS**.
- Macie helps identify and alert you to **sensitive data, such as personally identifiable information (PII)**



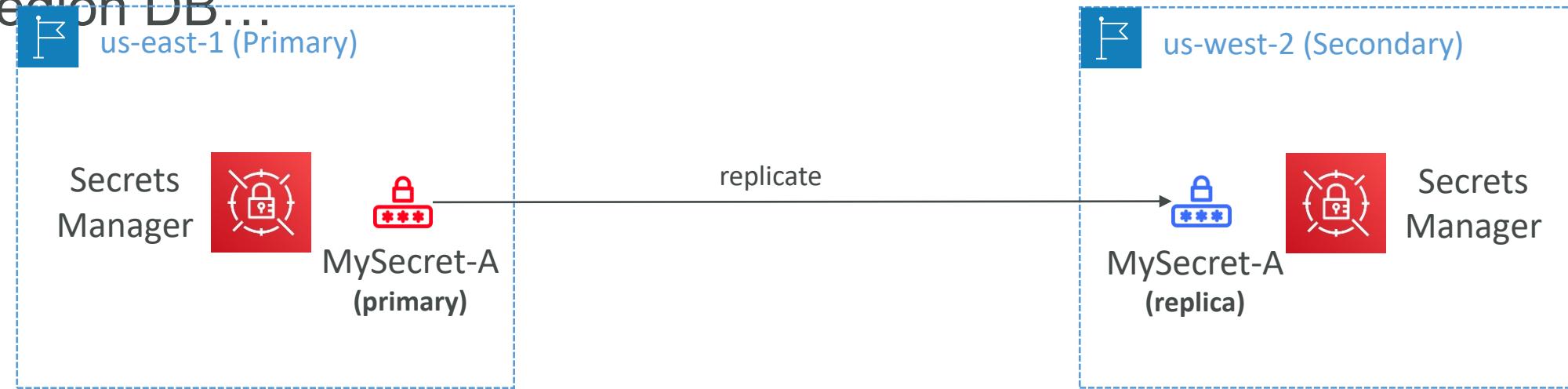
# AWS Secrets Manager



- Newer service, meant for storing secrets
- Capability to force **rotation of secrets** every X days
- Automate generation of secrets on rotation (uses Lambda)
- Integration with **Amazon RDS** (MySQL, PostgreSQL, Aurora)
- Secrets are encrypted using KMS
- Mostly meant for RDS integration

# AWS Secrets Manager – Multi-Region Secrets

- Replicate Secrets across multiple AWS Regions
- Secrets Manager keeps read replicas in sync with the primary Secret
- Ability to promote a read replica Secret to a standalone Secret
- Use cases: multi-region apps, disaster recovery strategies, multi-region DB...



# AWS WAF – Web Application Firewall



- Protects your web applications from common web exploits (Layer 7)
- **Layer 7 is HTTP** (vs Layer 4 is TCP/UDP)
  
- Deploy on
  - **Application Load Balancer**
  - **API Gateway**
  - **CloudFront**
  - **AppSync GraphQL API**
  - **Cognito User Pool**

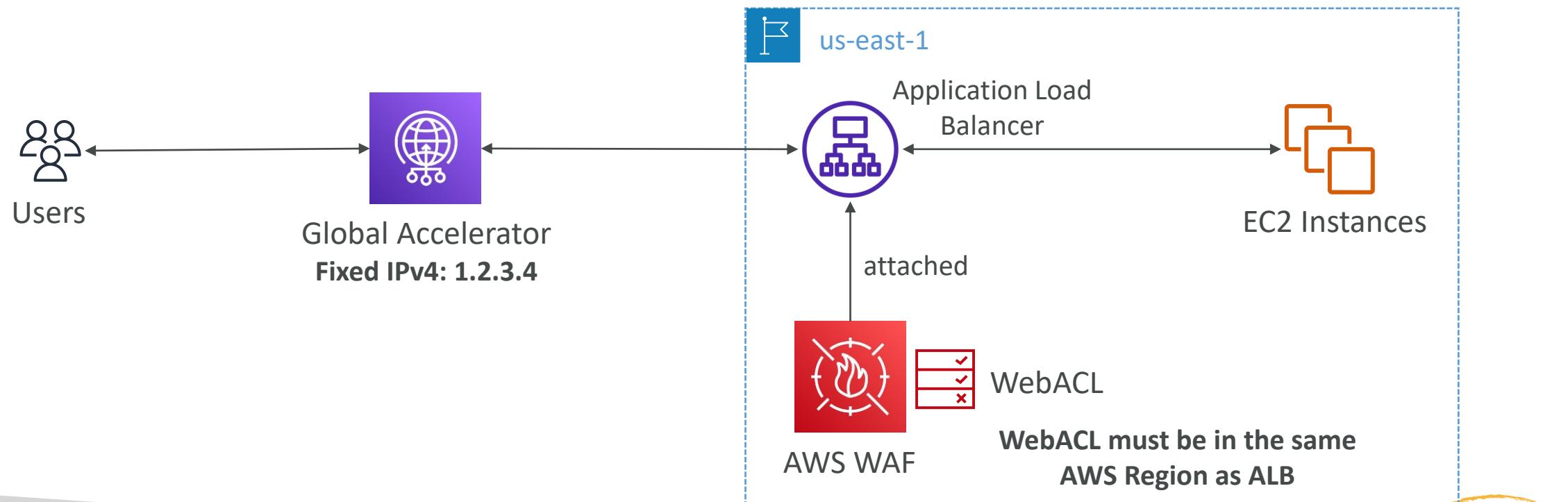
# AWS WAF – Web Application Firewall



- Define Web ACL (Web Access Control List) Rules:
  - **IP Set: up to 10,000 IP addresses** – use multiple Rules for more IPs
  - HTTP headers, HTTP body, or URI strings Protects from common attack - **SQL injection and Cross-Site Scripting (XSS)**
  - Size constraints, **geo-match (block countries)**
  - **Rate-based rules** (to count occurrences of events) – **for DDoS protection**
- Web ACL are Regional except for CloudFront
- A rule group is a **reusable set of rules that you can add to a web ACL**

# WAF – Fixed IP while using WAF with a Load Balancer

- WAF does not support the Network Load Balancer (Layer 4)
- We can use Global Accelerator for fixed IP and WAF on the ALB



# AWS Shield: protect from DDoS attack



- **DDoS:** Distributed Denial of Service – many requests at the same time
- **AWS Shield Standard:**
  - Free service that is activated for every AWS customer
  - Provides protection from attacks such as SYN/UDP Floods, Reflection attacks and other layer 3/layer 4 attacks
- **AWS Shield Advanced:**
  - Optional DDoS mitigation service (\$3,000 per month per organization)
  - Protect against more sophisticated attack on [Amazon EC2](#), [Elastic Load Balancing \(ELB\)](#), [Amazon CloudFront](#), [AWS Global Accelerator](#), and [Route 53](#)
  - 24/7 access to AWS DDoS response team (DRP)
  - Protect against higher fees during usage spikes due to DDoS
  - Shield Advanced automatic application layer DDoS mitigation automatically creates, evaluates and deploys AWS WAF rules to mitigate layer 7 attacks



# 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, MySQL and MariaDB
- 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-securig-amazon-emr/>



# Security – OpenSearch Service

- Amazon VPC provides network isolation
- OpenSearch policy to manage security further
- Data security by encrypting data at-rest using KMS
- Encryption in-transit using HTTPS (TLS)
  
- IAM or Cognito based authentication
- Amazon Cognito allow end-users to log-in to OpenSearch Dashboards 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
- Column Level Security to restrict access to specific columns in dataset

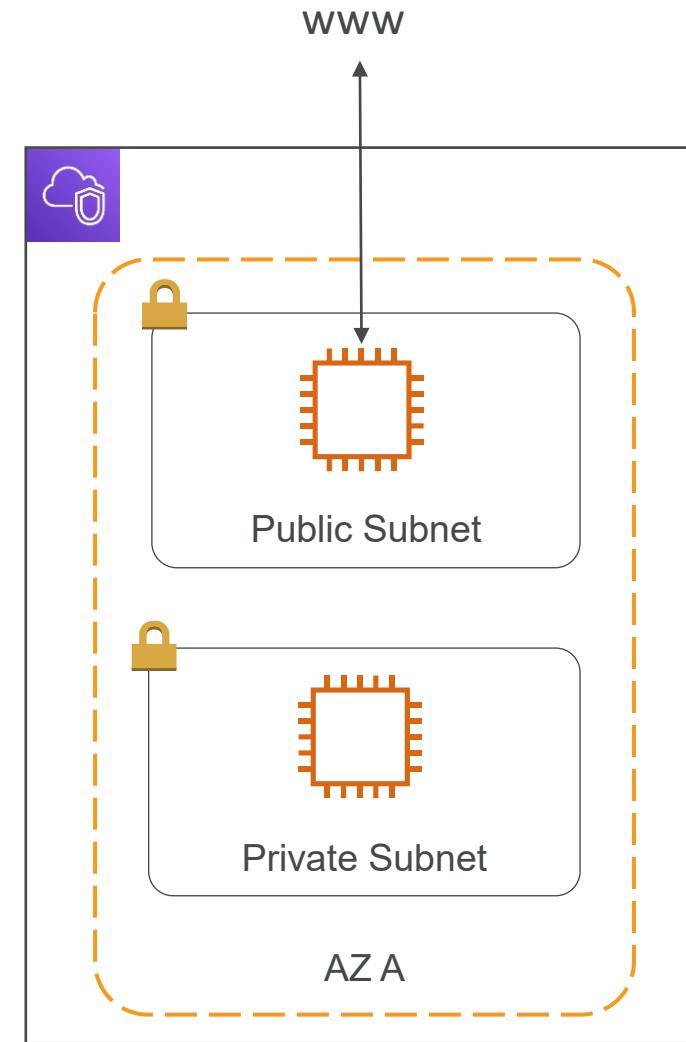
# Networking and Content Delivery

Connecting the pipes in AWS

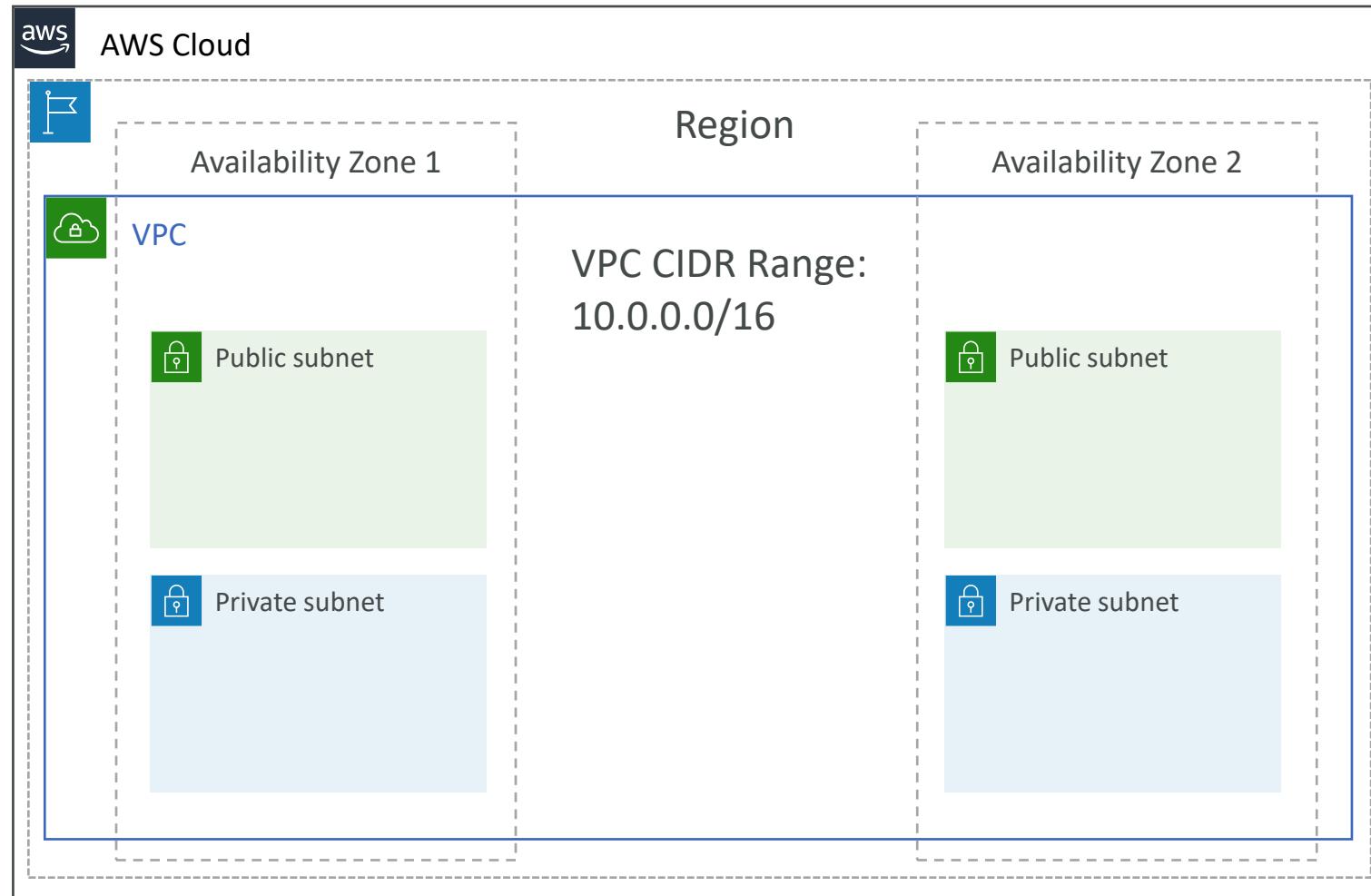
# VPC & Subnets

## Primer

- **VPC**: private network to deploy your resources (regional resource)
- **Subnets** allow you to partition your network inside your VPC (Availability Zone resource)
- A **public subnet** is a subnet that is accessible from the internet
- A **private subnet** is a subnet that is not accessible from the internet
- To define access to the internet and between subnets, we use **Route Tables**.

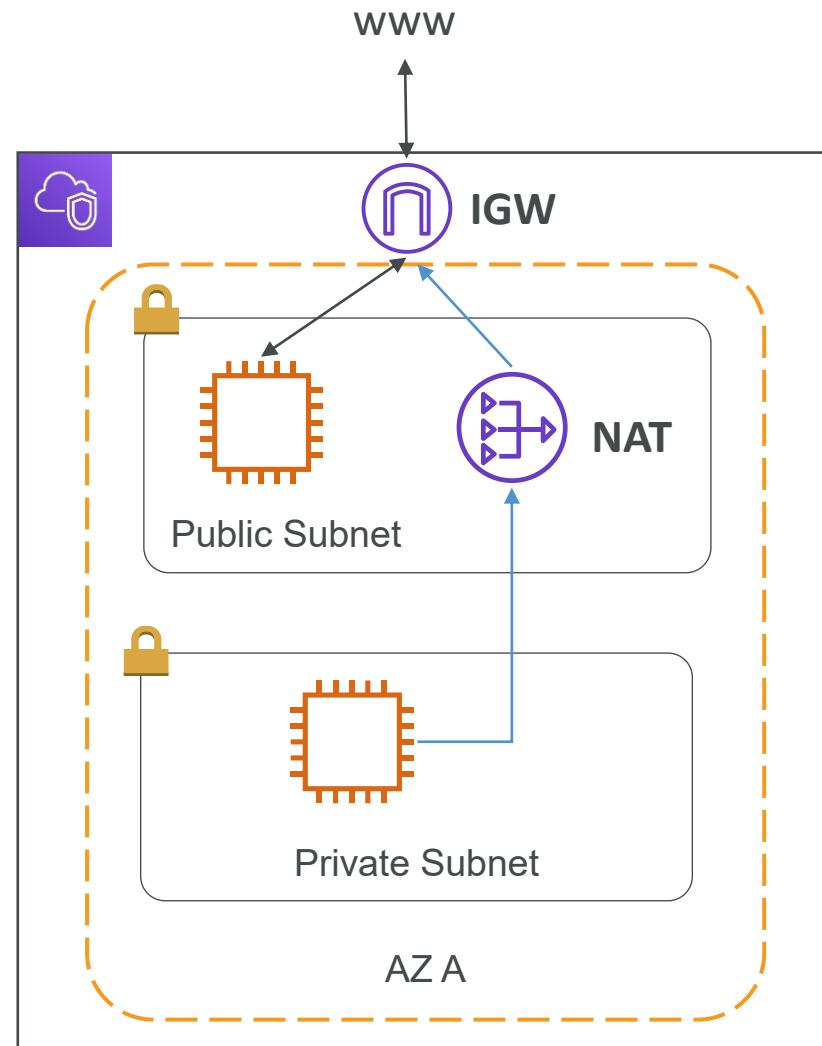


# VPC Diagram



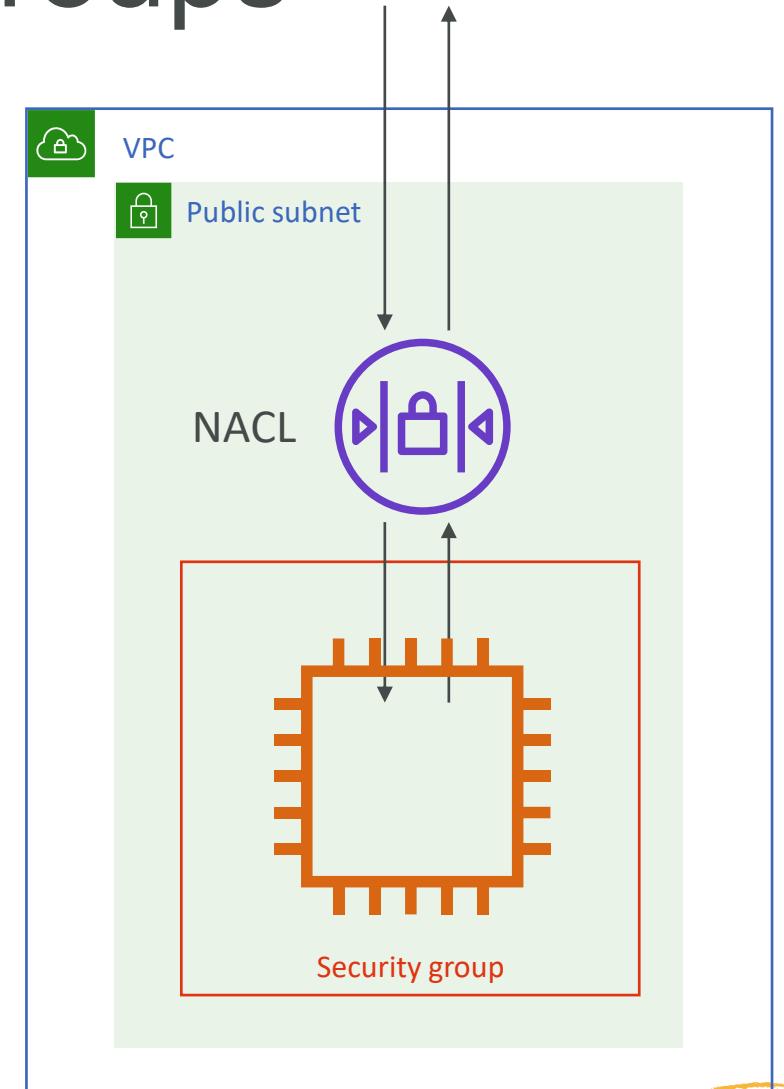
# Internet Gateway & NAT Gateways

- **Internet Gateways** helps our VPC instances connect with the internet
- Public Subnets have a route to the internet gateway.
- **NAT Gateways** (AWS-managed) & **NAT Instances** (self-managed) allow your instances in your **Private Subnets** to access the internet while remaining private



# Network ACL & Security Groups

- **NACL (Network ACL)**
  - A firewall which controls traffic from and to subnet
  - Can have ALLOW and DENY rules
  - Are attached at the **Subnet** level
  - Rules only include IP addresses
- **Security Groups**
  - A firewall that controls traffic to and from **an ENI / an EC2 Instance**
  - Can have only ALLOW rules
  - Rules include IP addresses and other security groups



# Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

[https://docs.aws.amazon.com/vpc/latest/userguide/VPC\\_Security.html#VPC\\_Security\\_Comparison](https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison)

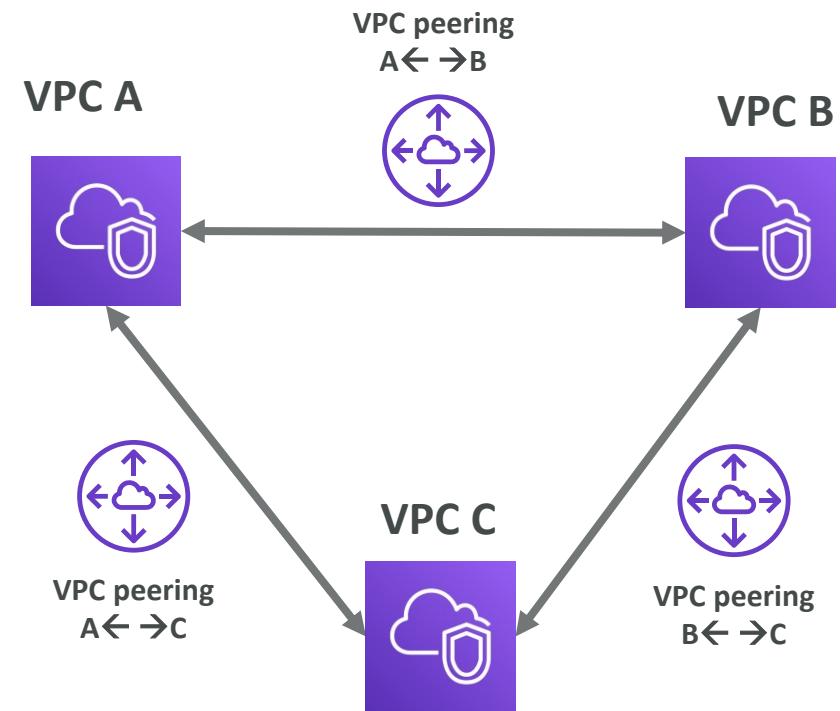


# VPC Flow Logs

- Capture information about IP traffic going into your interfaces:
  - **VPC Flow Logs**
  - **Subnet Flow Logs**
  - **Elastic Network Interface Flow Logs**
- Helps to monitor & troubleshoot connectivity issues. Example:
  - Subnets to internet
  - Subnets to subnets
  - Internet to subnets
- Captures network information from AWS managed interfaces too: Elastic Load Balancers, ElastiCache, RDS, Aurora, etc...
- VPC Flow logs data can go to S3, CloudWatch Logs, and Kinesis Data Firehose

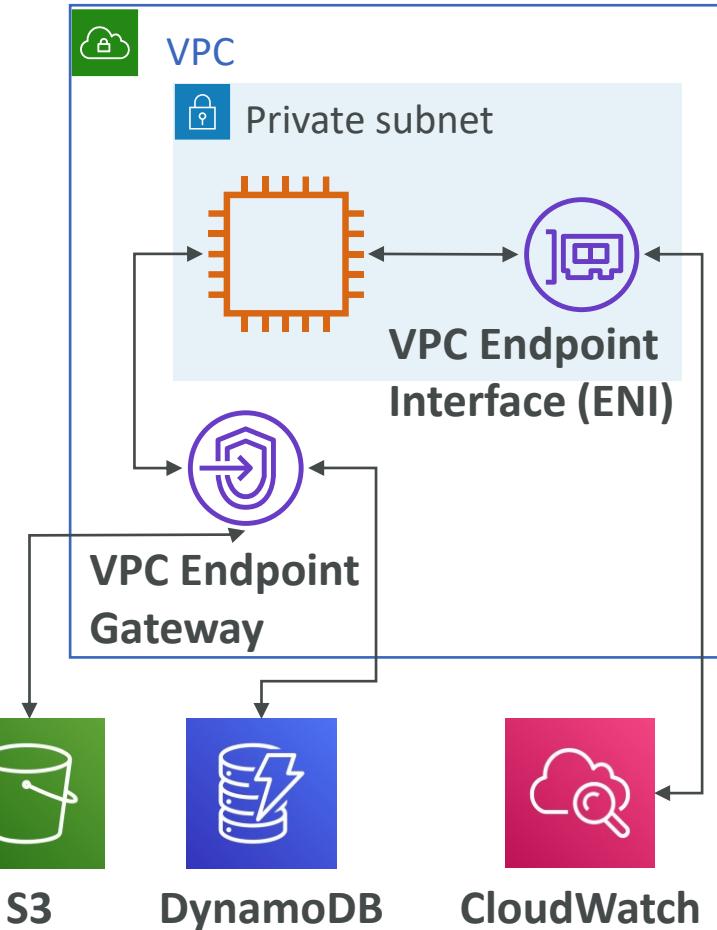
# VPC Peering

- Connect two VPC, privately using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDR (IP address range)
- VPC Peering connection is **not transitive** (must be established for each VPC that need to communicate with one another)



# VPC Endpoints

- Endpoints allow you to connect to AWS Services **using a private network** instead of the public www network
- This gives you enhanced security and lower latency to access AWS services
- VPC Endpoint Gateway: S3 & DynamoDB
- VPC Endpoint Interface: the rest
- Only used within your VPC



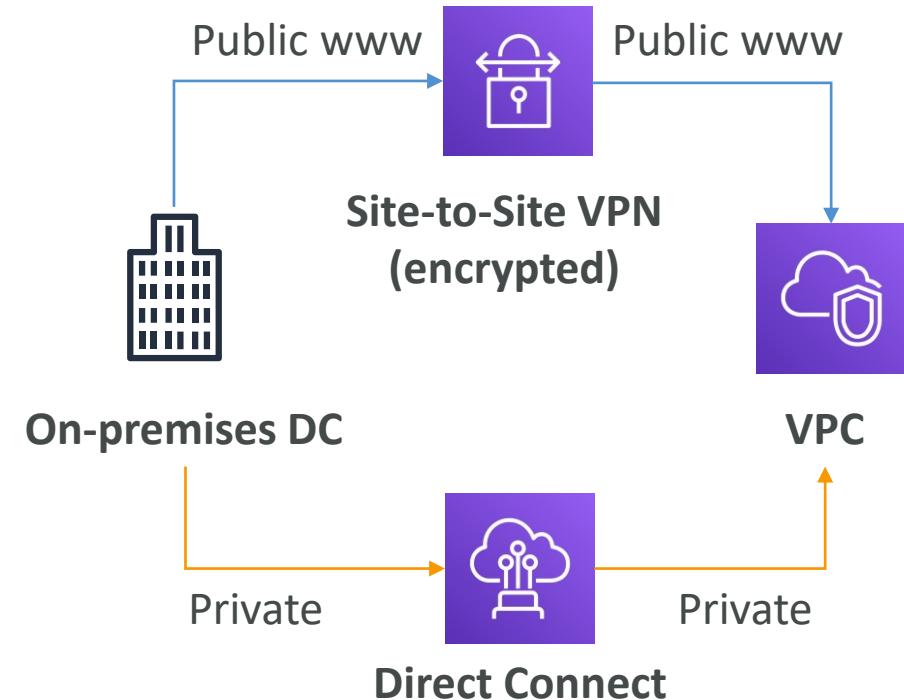
# Site to Site VPN & Direct Connect

- **Site to Site VPN**

- Connect an on-premises VPN to AWS
- The connection is automatically encrypted
- Goes over the public internet

- **Direct Connect (DX)**

- Establish a physical connection between on-premises and AWS
- The connection is private, secure and fast
- Goes over a private network
- Takes at least a month to establish



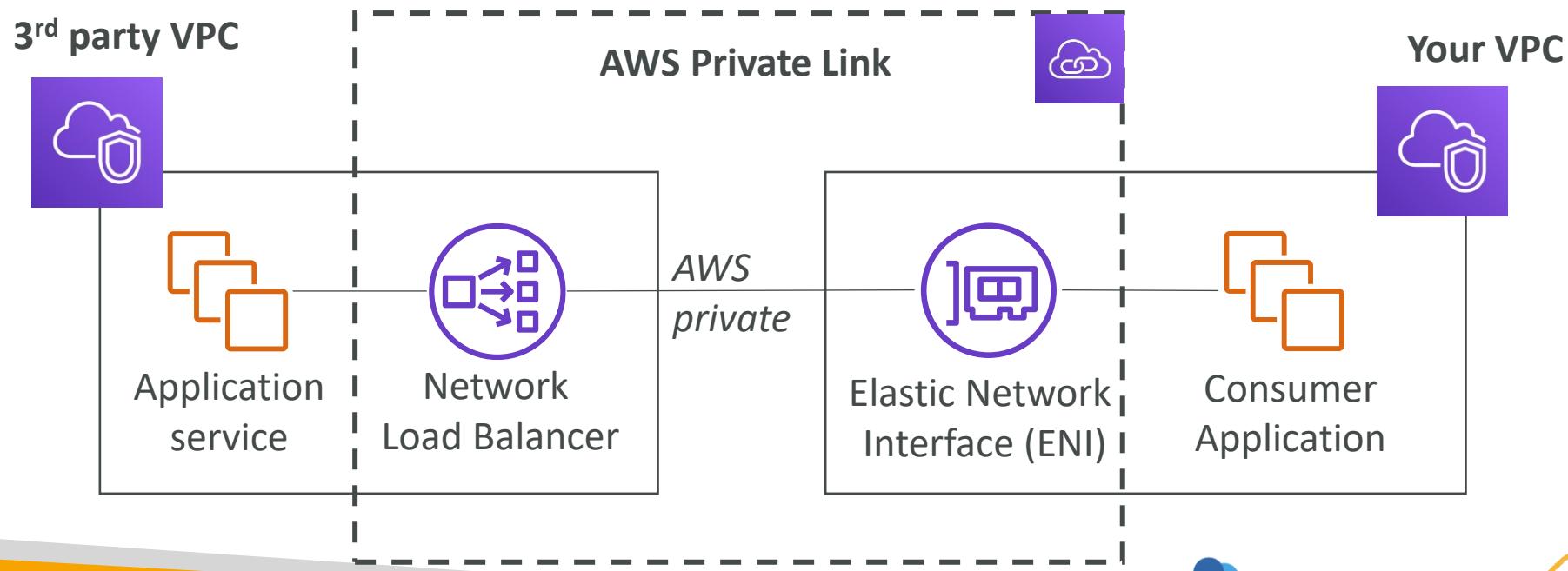
# VPC Closing Comments

- **VPC:** Virtual Private Cloud
- **Subnets:** Tied to an AZ, network partition of the VPC
- **Internet Gateway:** at the VPC level, provide Internet Access
- **NAT Gateway / Instances:** give internet access to private subnets
- **NACL:** Stateless, subnet rules for inbound and outbound
- **Security Groups:** Stateful, operate at the EC2 instance level or ENI
- **VPC Peering:** Connect two VPC with non overlapping IP ranges, non transitive
- **VPC Endpoints:** Provide private access to AWS Services within VPC
- **VPC Flow Logs:** network traffic logs
- **Site to Site VPN:** VPN over public internet between on-premises DC and AWS
- **Direct Connect:** direct private connection to a AWS

# AWS PrivateLink (VPC Endpoint Services)



- Most secure & scalable way to expose a service to 1000s of VPCs
- Does not require VPC peering, internet gateway, NAT, route tables...
- Requires a network load balancer (Service VPC) and ENI (Customer VPC)



# What is DNS?

- Domain Name System which translates the human friendly hostnames into the machine IP addresses
- www.google.com => 172.217.18.36
- DNS is the backbone of the Internet
- DNS uses hierarchical naming structure

.com

example.com

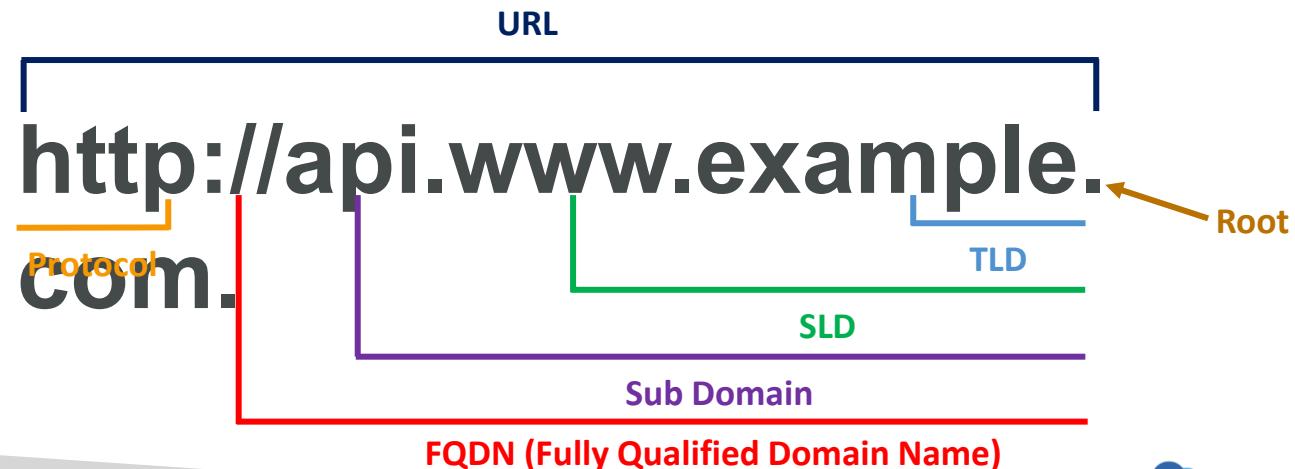
www.example.com

api.example.co

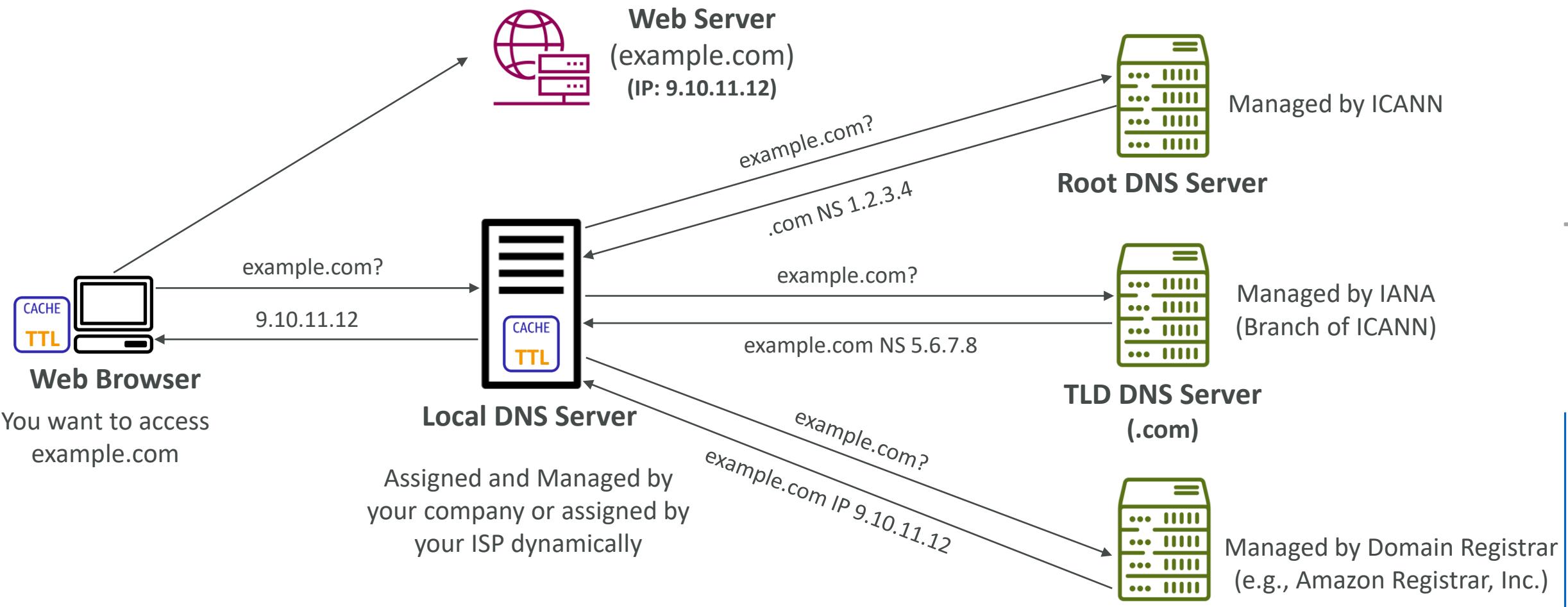
m

# DNS Terminologies

- **Domain Registrar:** Amazon Route 53, GoDaddy, ...
- **DNS Records:** A, AAAA, CNAME, NS, ...
- **Zone File:** contains DNS records
- **Name Server:** resolves DNS queries (Authoritative or Non-Authoritative)
- **Top Level Domain (TLD):** .com, .us, .in, .gov, .org, ...
- **Second Level Domain (SLD):** amazon.com, google.com, ...

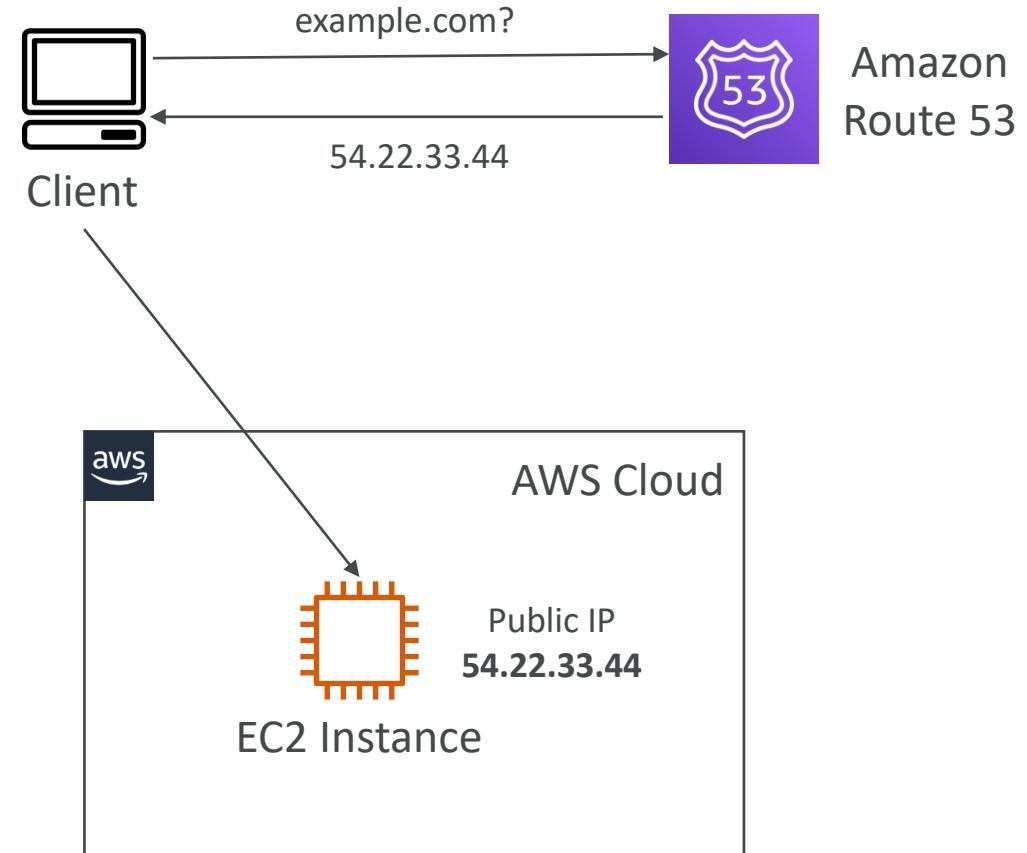


# How DNS Works



# Amazon Route 53

- A highly available, scalable, fully managed and *Authoritative* DNS
  - Authoritative = the customer (you) can update the DNS records
- Route 53 is also a Domain Registrar
- Ability to check the health of your resources
- The only AWS service which provides 100% availability SLA
- Why Route 53? 53 is a reference to the traditional DNS port

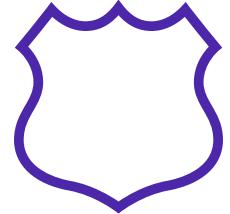


# Route 53 – Records

- How you want to route traffic for a domain
- Each record contains:
  - **Domain/subdomain Name** – e.g., example.com
  - **Record Type** – e.g., A or AAAA
  - **Value** – e.g., 12.34.56.78
  - **Routing Policy** – how Route 53 responds to queries
  - **TTL** – amount of time the record cached at DNS Resolvers
- Route 53 supports the following DNS record types:
  - (must know) A / AAAA / CNAME / NS
  - (advanced) CAA / DS / MX / NAPTR / PTR / SOA / TXT / SPF / SRV

# Route 53 – Record Types

- **A** – maps a hostname to IPv4
- **AAAA** – maps a hostname to IPv6
- **CNAME** – maps a hostname to another hostname
  - The target is a domain name which must have an A or AAAA record
  - Can't create a CNAME record for the top node of a DNS namespace (Zone Apex)
  - Example: you can't create for example.com, but you can create for www.example.com
- **NS** – Name Servers for the Hosted Zone
  - Control how traffic is routed for a domain

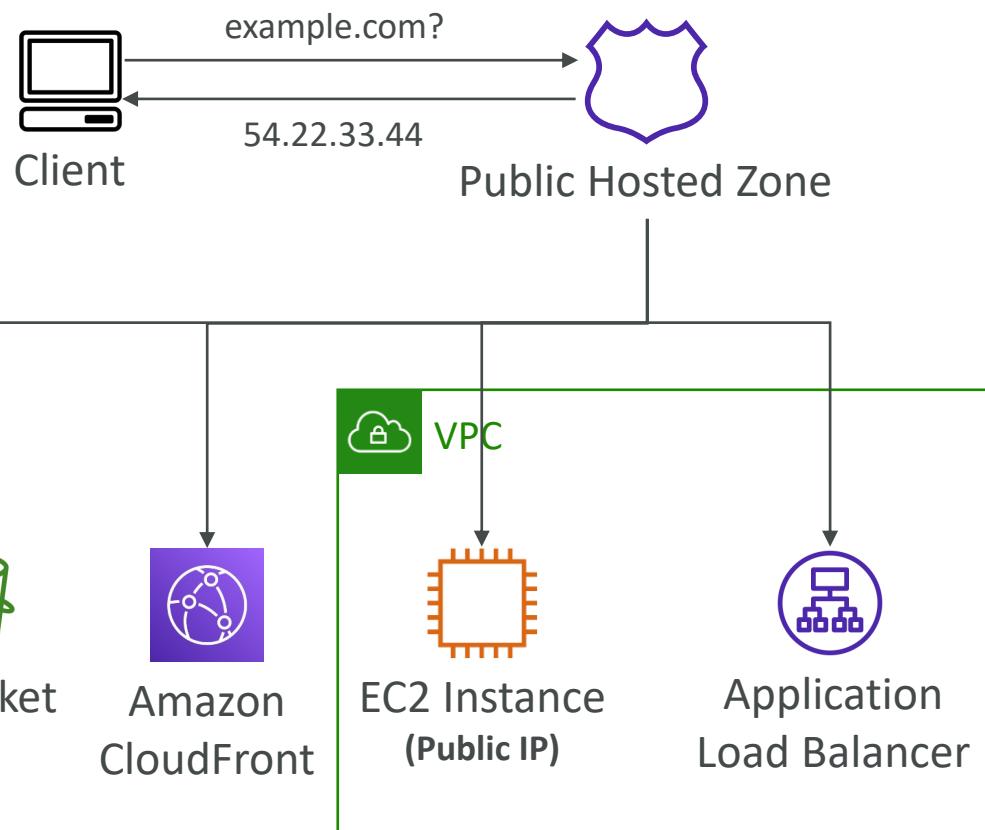


# Route 53 – Hosted Zones

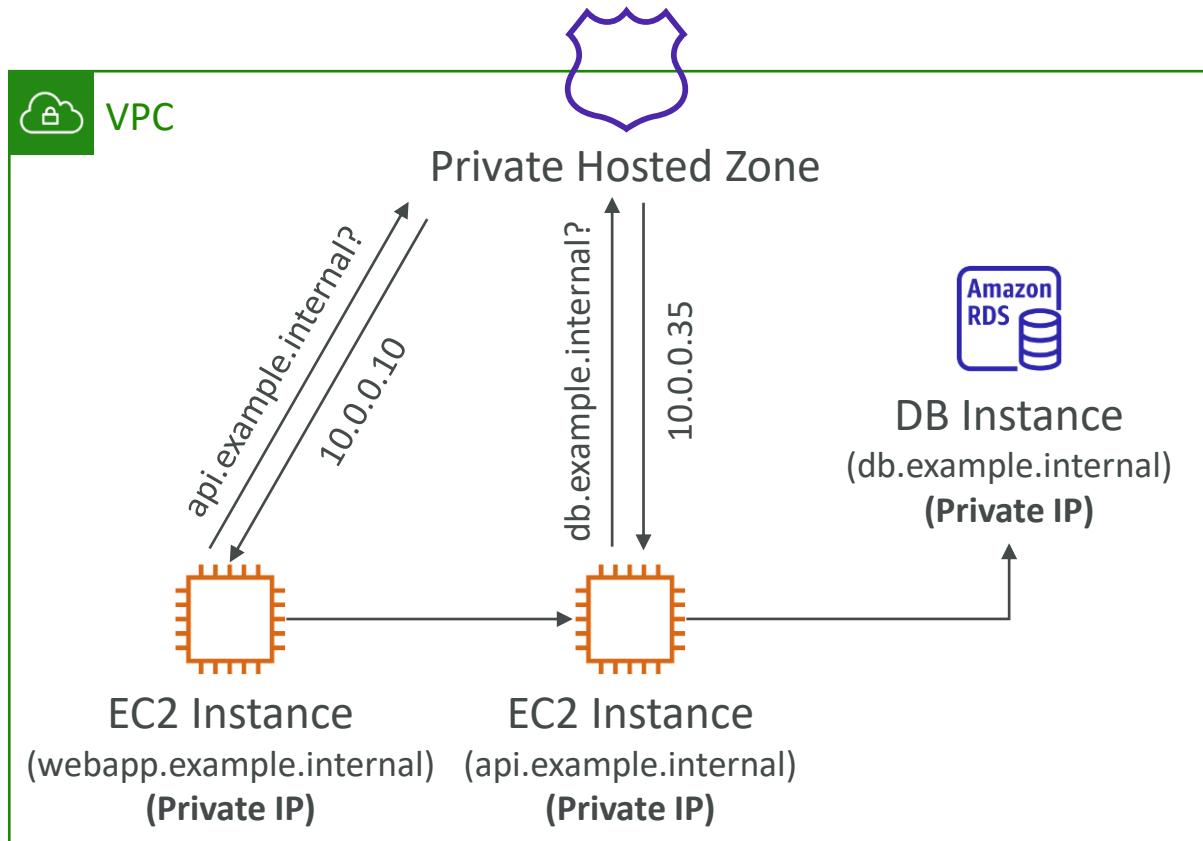
- A container for records that define how to route traffic to a domain and its subdomains
- **Public Hosted Zones** – contains records that specify how to route traffic on the Internet (public domain names)  
[application1.mypublicdomain.com](http://application1.mypublicdomain.com)
- **Private Hosted Zones** – contain records that specify how you route traffic within one or more VPCs (private domain names)  
[application1.company.internal](http://application1.company.internal)
- You pay \$0.50 per month per hosted zone

# Route 53 – Public vs. Private Hosted Zones

## Public Hosted Zone



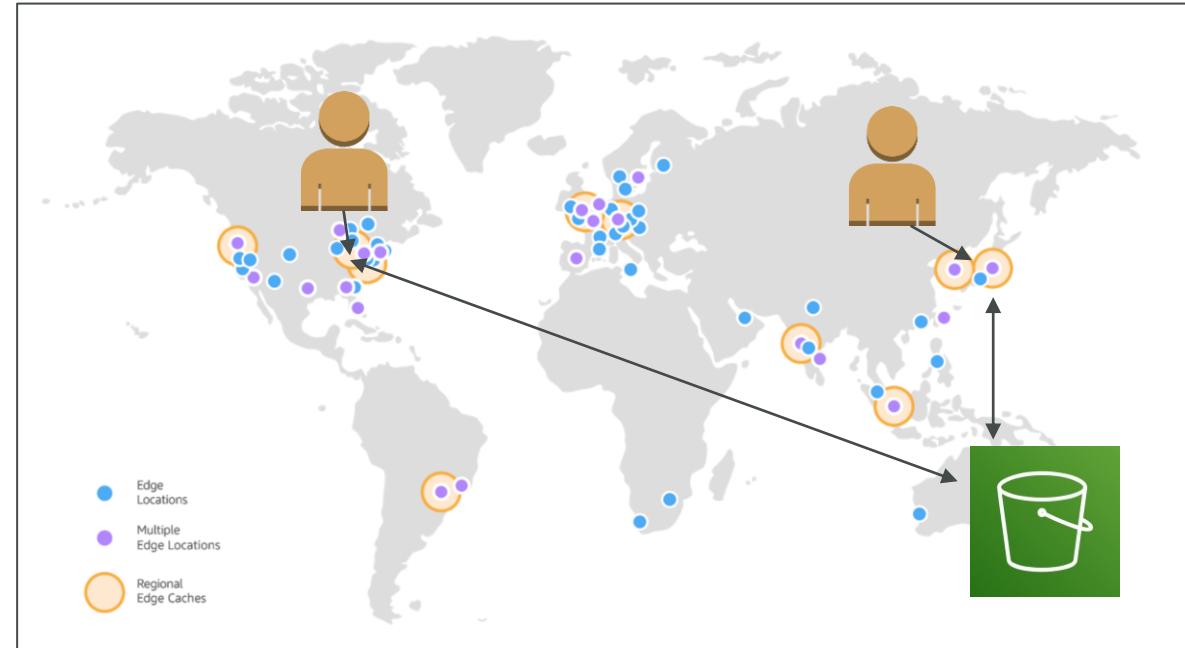
## Private Hosted Zone



# Amazon CloudFront



- Content Delivery Network (CDN)
- **Improves read performance, content is cached at the edge**
- Improves users experience
- 216 Point of Presence globally (edge locations)
- **DDoS protection (because worldwide), integration with Shield, AWS Web Application Firewall**

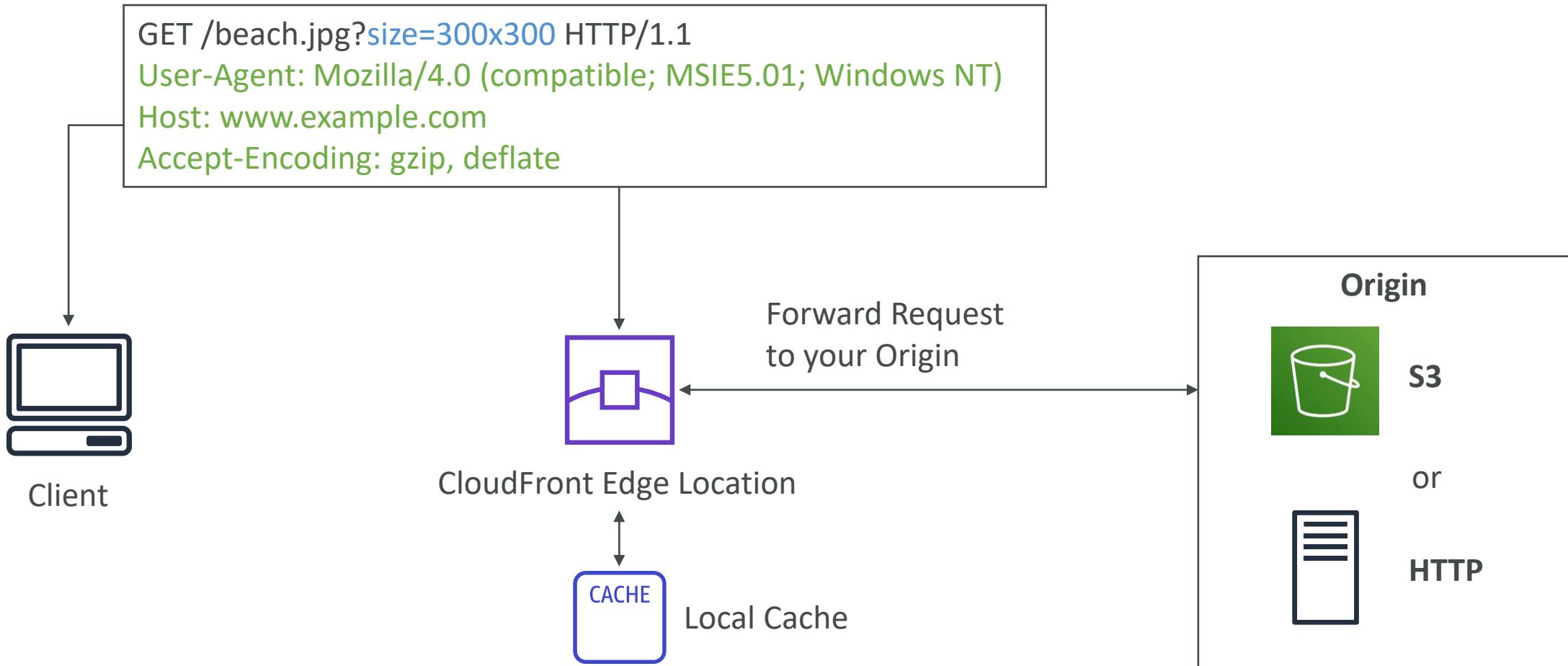


Source: <https://aws.amazon.com/cloudfront/features/?nc=sn&loc=2>

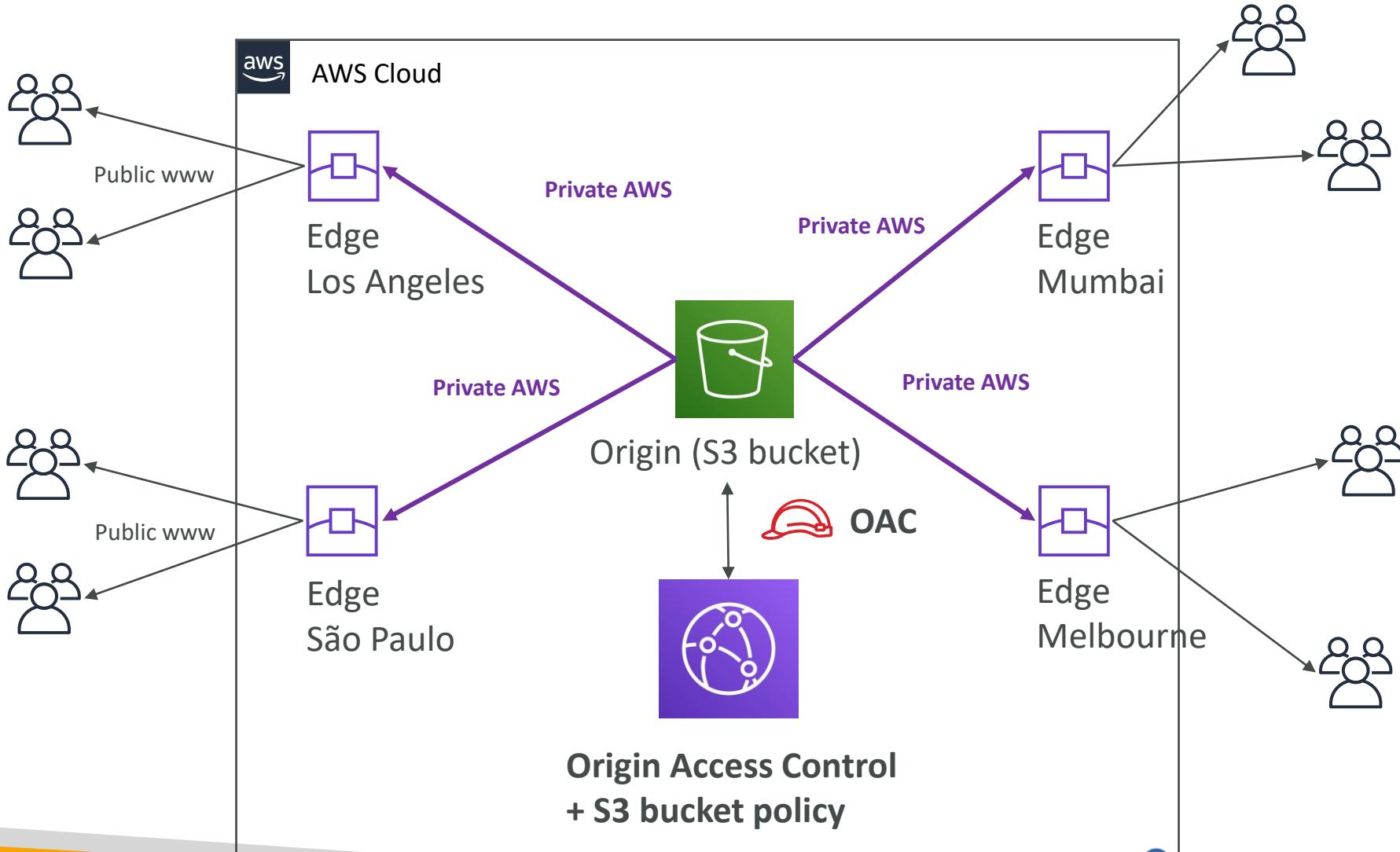
# CloudFront – Origins

- **S3 bucket**
  - For distributing files and caching them at the edge
  - Enhanced security with CloudFront **Origin Access Control (OAC)**
  - OAC is replacing Origin Access Identity (OAI)
  - CloudFront can be used as an ingress (to upload files to S3)
- **Custom Origin (HTTP)**
  - Application Load Balancer
  - EC2 instance
  - S3 website (must first enable the bucket as a static S3 website)
  - Any HTTP backend you want

# CloudFront at a high level



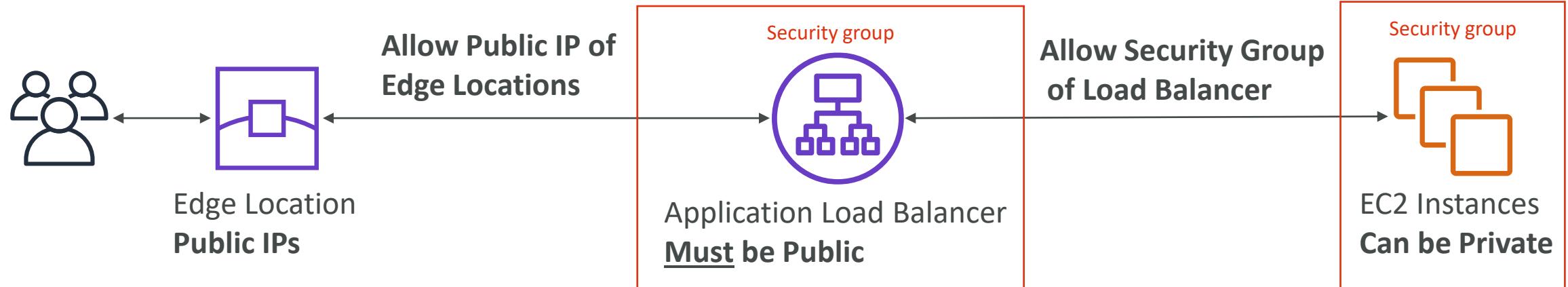
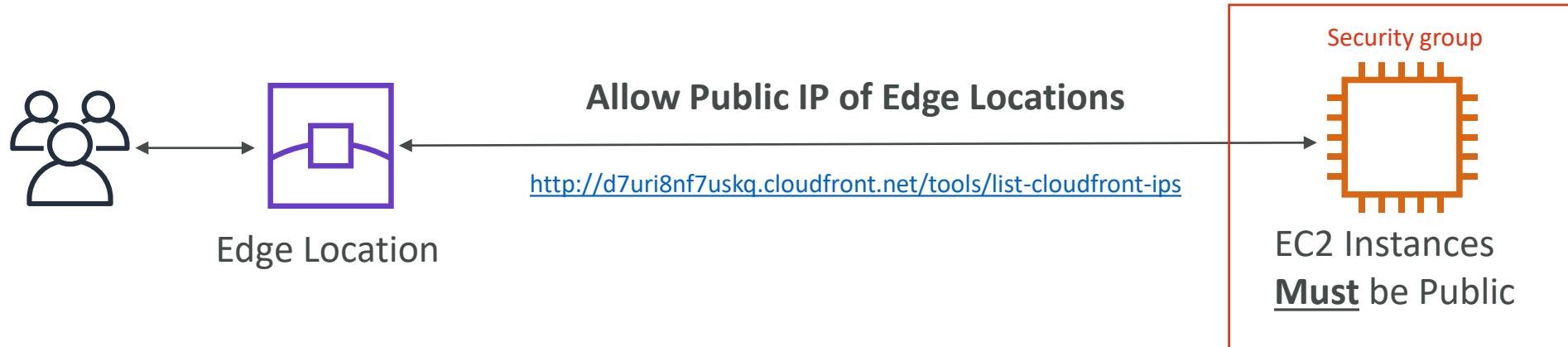
# CloudFront – S3 as an Origin



# CloudFront vs S3 Cross Region Replication

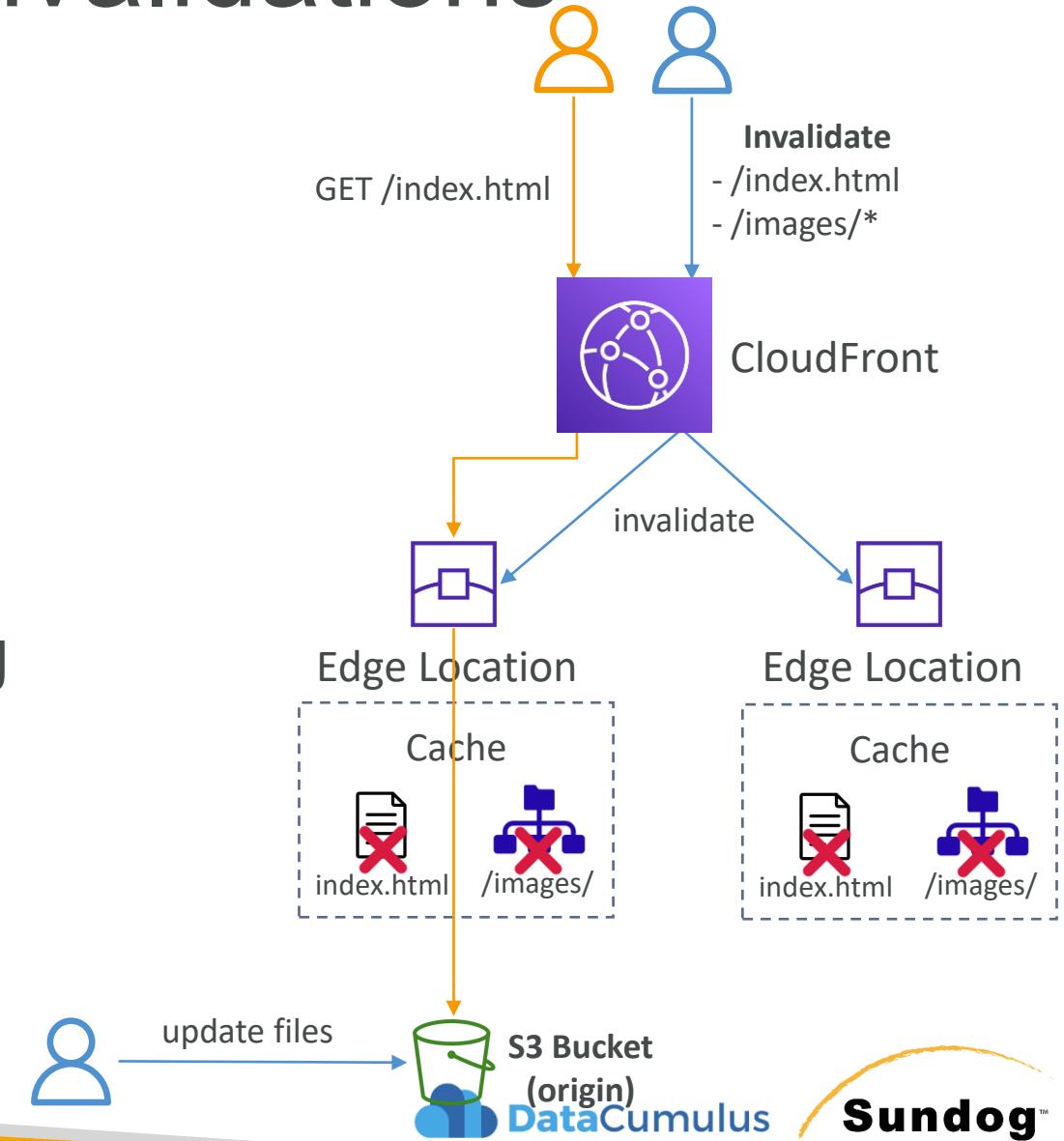
- CloudFront:
  - Global Edge network
  - Files are cached for a TTL (maybe a day)
  - **Great for static content that must be available everywhere**
- S3 Cross Region Replication:
  - Must be setup for each region you want replication to happen
  - Files are updated in near real-time
  - Read only
  - **Great for dynamic content that needs to be available at low-latency in few regions**

# CloudFront – ALB or EC2 as an origin



# CloudFront – Cache Invalidations

- In case you update the back-end origin, CloudFront doesn't know about it and will only get the refreshed content after the TTL has expired
- However, you can force an entire or partial cache refresh (thus bypassing the TTL) by performing a **CloudFront Invalidation**
- You can invalidate all files (\*) or a special path (/images/\*)



# Management and Governance

Monitoring the big picture in AWS

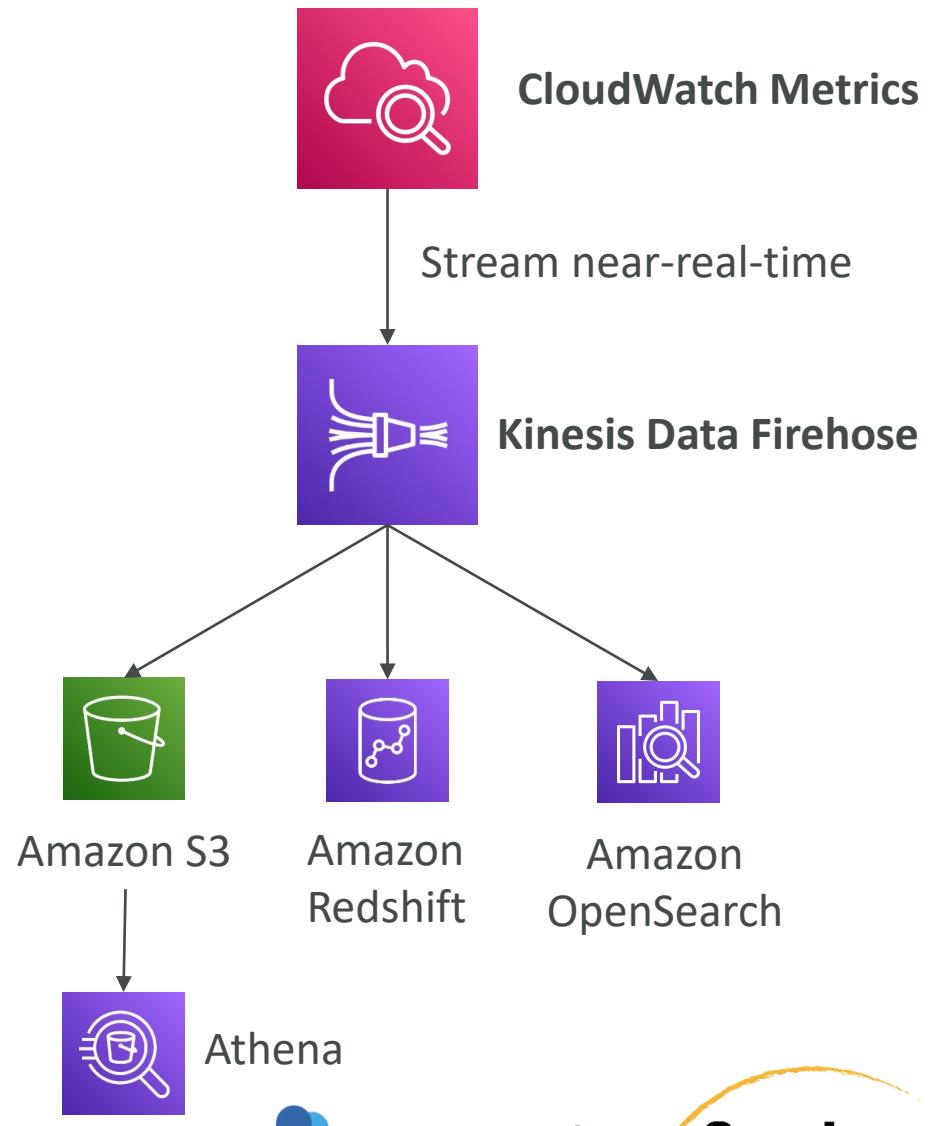
# Amazon CloudWatch Metrics



- CloudWatch provides metrics for **every** services in AWS
- **Metric** is a variable to monitor (CPUUtilization, NetworkIn...)
- Metrics belong to **namespaces**
- **Dimension** is an attribute of a metric (instance id, environment, etc...).
- Up to 30 dimensions per metric
- Metrics have **timestamps**
- Can create CloudWatch dashboards of metrics
- Can create **CloudWatch Custom Metrics** (for the RAM for example)

# CloudWatch Metric Streams

- Continually stream CloudWatch metrics to a destination of your choice, with **near-real-time delivery** and low latency.
  - Amazon Kinesis Data Firehose (and then its destinations)
  - 3<sup>rd</sup> party service provider: Datadog, Dynatrace, New Relic, Splunk, Sumo Logic...
- Option to **filter metrics** to only stream a subset of them





# CloudWatch Logs

- **Log groups:** arbitrary name, usually representing an application
- **Log stream:** instances within application / log files / containers
- Can define log expiration policies (never expire, 1 day to 10 years...)
- **CloudWatch Logs can send logs to:**
  - Amazon S3 (exports)
  - Kinesis Data Streams
  - Kinesis Data Firehose
  - AWS Lambda
  - OpenSearch
- Logs are encrypted by default
- Can setup KMS-based encryption with your own keys

# CloudWatch Logs - Sources

- SDK, CloudWatch Logs Agent, CloudWatch Unified Agent
- Elastic Beanstalk: collection of logs from application
- ECS: collection from containers
- AWS Lambda: collection from function logs
- VPC Flow Logs: VPC specific logs
- API Gateway
- CloudTrail based on filter
- Route53: Log DNS queries

# CloudWatch Logs Insights

The screenshot shows the CloudWatch Logs Insights interface. At the top, there's a navigation bar with 'CloudWatch > Logs Insights'. Below it is a search bar labeled 'Select log group(s)' with 'application.log' selected. To the right of the search bar are time range controls showing '2021-11-09 (06:40:02) > 2021-11-09 (06:55:17)' and a calendar icon. On the far right of the header is a vertical sidebar with 'Fields', 'Queries', and 'Help' buttons.

The main query editor area has a text input box with placeholder 'Write your query here.' and a code editor containing:

```
1 fields @timestamp, @message
2 | sort @timestamp desc
3 | limit 20
```

Below the editor are 'Run query', 'Save', and 'History' buttons. A note says 'Queries are allowed to run for up to 15 minutes.'

To the right of the query editor are two boxes: 'Change the time range here.' pointing to the time range controls, and 'Discovered Fields in your log groups.' pointing to the sidebar.

The results section starts with tabs for 'Logs' (which is selected) and 'Visualization'. A box highlights 'Tabs for query results, and visualization options.' An arrow points from this box to the 'Logs' tab. To the right of the tabs are 'Export results' and 'Add to dashboard' buttons, with a box highlighting 'Export the results, or add to a dashboard.' An arrow points from this box to the 'Add to dashboard' button.

The results display a histogram titled 'Showing 20 of 10,197 records matched' with a note '10,197 records (2.3 MB) scanned in 3.3s @ 3,091 records/s (714.9 kB/s)'. A 'Hide histogram' link is also present. Below the histogram is a table with columns '#', '@timestamp', and '@message'. It shows two rows of data:

#	@timestamp	@message
► 1	2021-11-09T06:54:17.62...	{"Severity": "INFO", "message": "This is where the message detail would go", "IP Address": "10.30.86.98", "Timestamp": "2021-11-09T11:54:17.620Z"}
► 2	2021-11-09T06:54:13.38...	{"Severity": "INFO", "message": "This is where the message detail would go", "IP Address": "192.168.0.43", "Timestamp": "2021-11-09T11:54:13.380Z"}

<https://mng.workshop.aws/operations-2022/detect/cwlogs.html>

# CloudWatch Logs Insights

- Search and analyze log data stored in CloudWatch Logs
- Example: find a specific IP inside a log, count occurrences of “ERROR” in your logs...
- Provides a purpose-built query language
  - Automatically discovers fields from AWS services and JSON log events
  - Fetch desired event fields, filter based on conditions, calculate aggregate statistics, sort events, limit number of events...
  - Can save queries and add them to CloudWatch Dashboards
- Can query multiple Log Groups in different AWS accounts
- It’s a query engine, not a real-time engine

Sample queries [Learn more ↗](#)

- ▶ Lambda
- ▶ VPC Flow Logs
- ▶ CloudTrail
- ▼ Common queries

- ▼ 25 most recently added log events

```
fields @timestamp, @message
| sort @timestamp desc
| limit 25
```

[Apply](#)
- ▼ Number of exceptions logged every 5 minutes

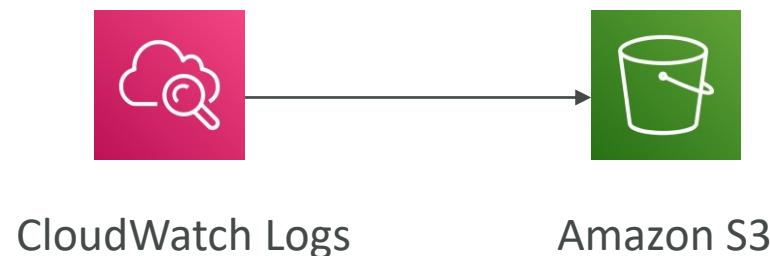
```
filter @message like /Exception/
| stats count(*) as exceptionCount by
bin(5m)
| sort exceptionCount desc
```

[Apply](#)
- ▼ List of log events that are not exceptions

```
fields @message
| filter @message not like /Exception/
```

[Apply](#)

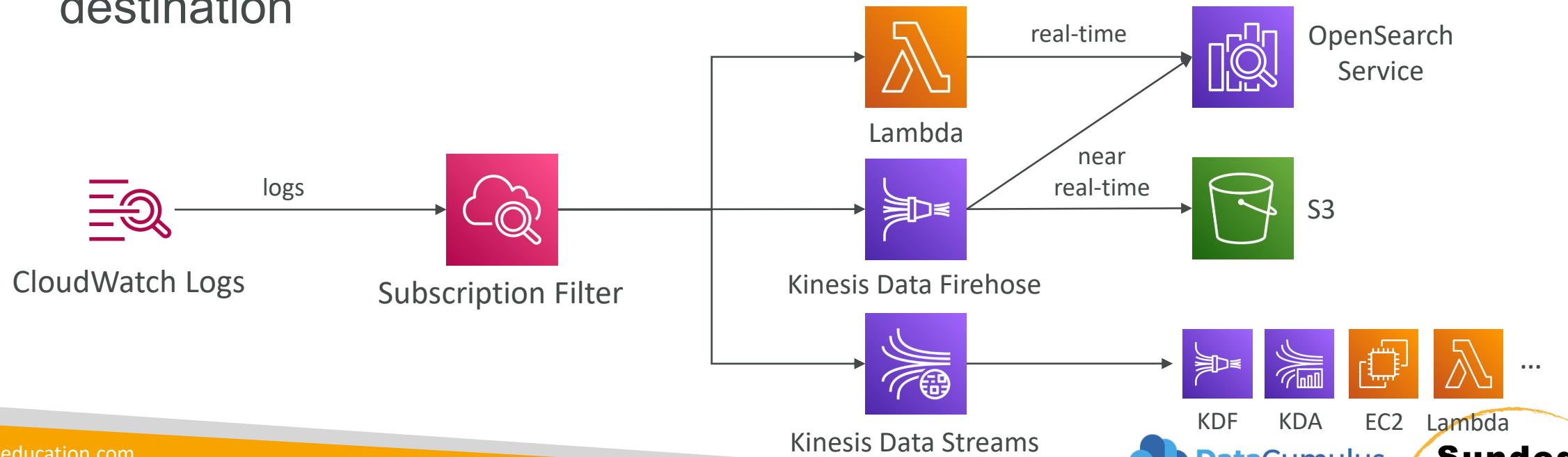
# CloudWatch Logs – S3 Export



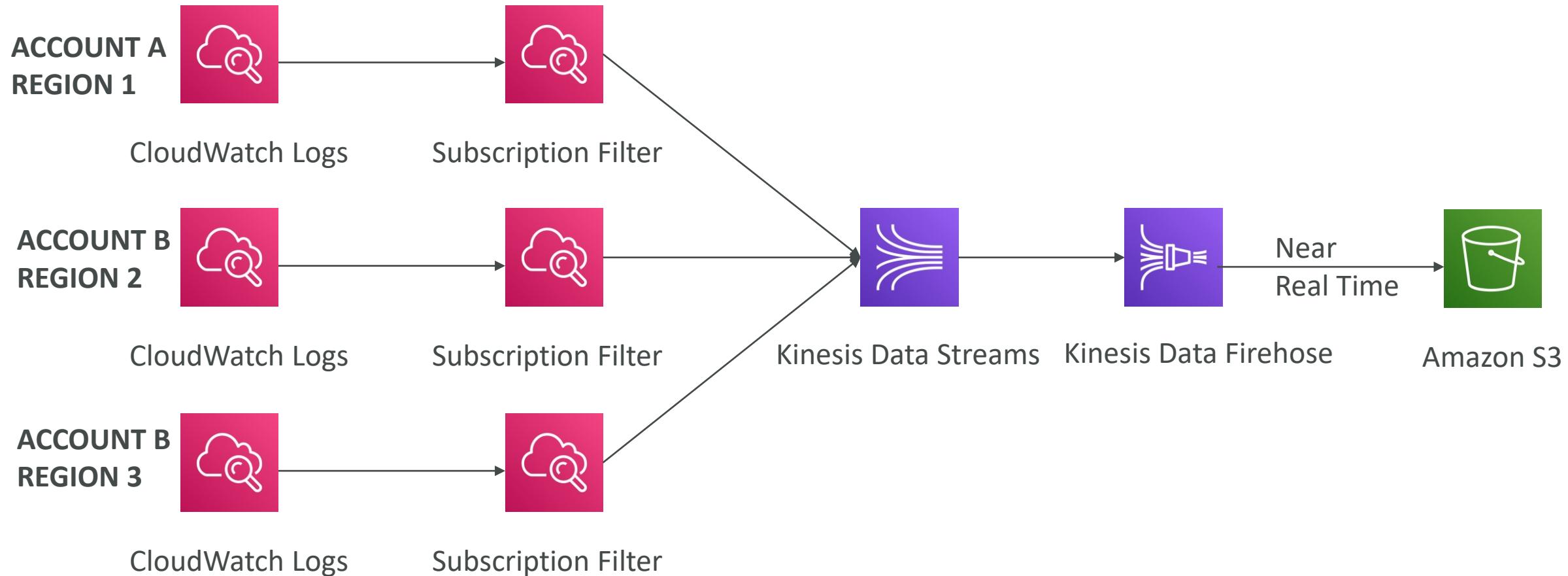
- Log data can take **up to 12 hours** to become available for export
- The API call is **CreateExportTask**
- Not near-real time or real-time... use Logs Subscriptions instead

# CloudWatch Logs Subscriptions

- Get a real-time log events from CloudWatch Logs for processing and analysis
- Send to Kinesis Data Streams, Kinesis Data Firehose, or Lambda
- **Subscription Filter** – filter which logs are events delivered to your destination

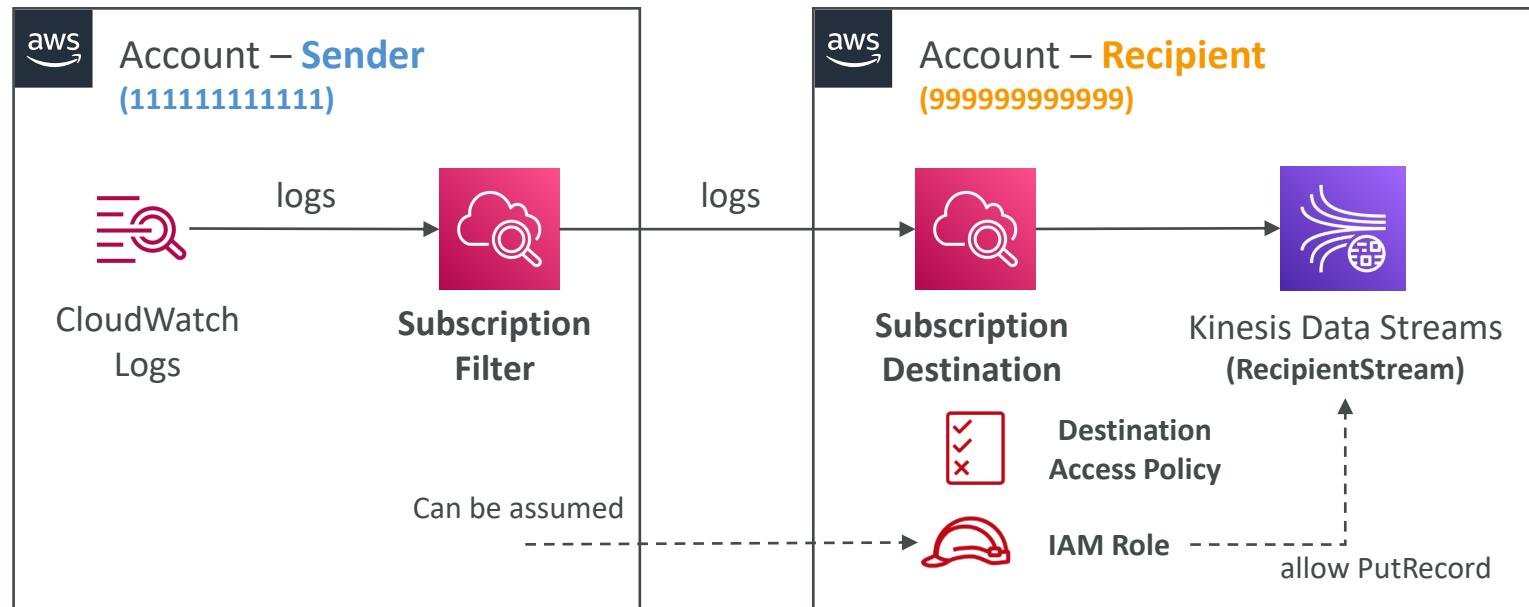


# CloudWatch Logs Aggregation Multi-Account & Multi Region



# CloudWatch Logs Subscriptions

- Cross-Account Subscription** – send log events to resources in a different AWS account (KDS, KDF)



```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "kinesis:PutRecord",
      "Resource": "arn:aws:kinesis:us-east-1:999999999999:stream/RecipientStream"
    }
  ]
}

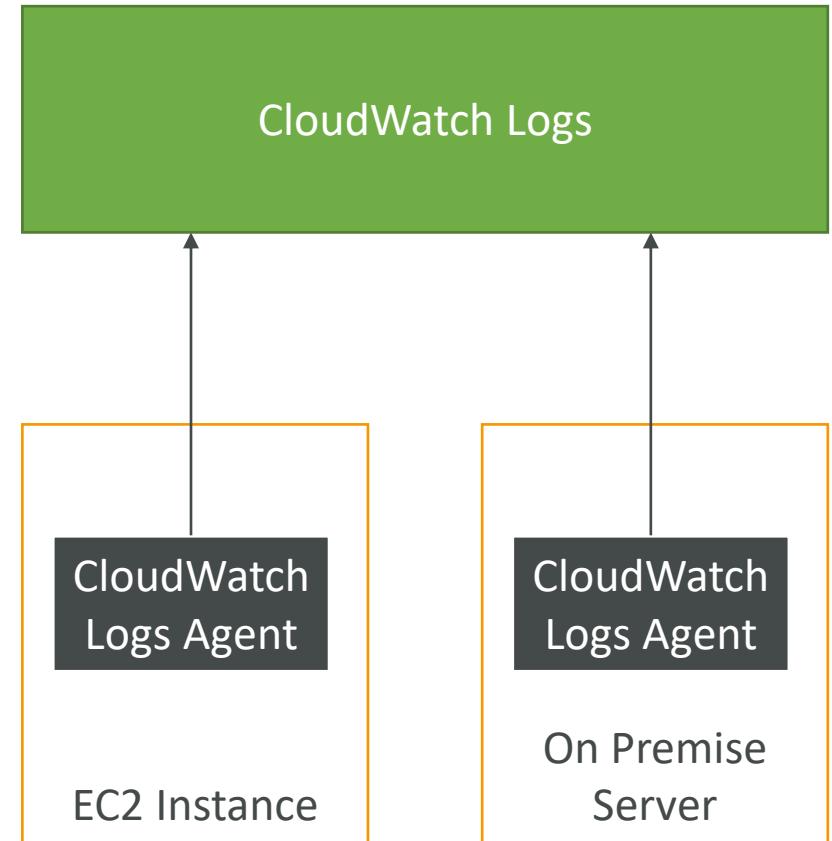
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111111111111"
      },
      "Action": "logs:PutSubscriptionFilter",
      "Resource": "arn:aws:logs:us-east-1:999999999999:destination:testDestination"
    }
  ]
}
  
```

**IAM Role (Cross-Account)**

**Destination Access Policy**

# CloudWatch Logs for EC2

- By default, no logs from your EC2 machine will go to CloudWatch
- You need to run a CloudWatch agent on EC2 to push the log files you want
- Make sure IAM permissions are correct
- The CloudWatch log agent can be setup on-premises too



# CloudWatch Logs Agent & Unified Agent

- For virtual servers (EC2 instances, on-premises servers...)
- **CloudWatch Logs Agent**
  - Old version of the agent
  - Can only send to CloudWatch Logs
- **CloudWatch Unified Agent**
  - Collect additional system-level metrics such as RAM, processes, etc...
  - Collect logs to send to CloudWatch Logs
  - Centralized configuration using SSM Parameter Store

# CloudWatch Unified Agent – Metrics

- Collected directly on your Linux server / EC2 instance
- **CPU** (active, guest, idle, system, user, steal)
- **Disk metrics** (free, used, total), **Disk IO** (writes, reads, bytes, iops)
- **RAM** (free, inactive, used, total, cached)
- **Netstat** (number of TCP and UDP connections, net packets, bytes)
- **Processes** (total, dead, bloqued, idle, running, sleep)
- **Swap Space** (free, used, used %)
- Reminder: out-of-the box metrics for EC2 – disk, CPU, network (high level)

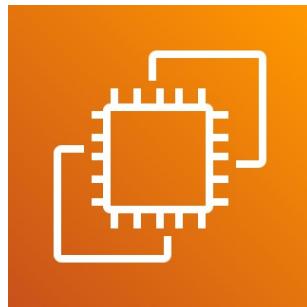
# CloudWatch Alarms



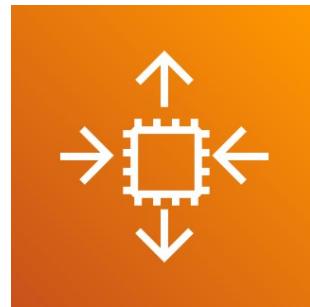
- Alarms are used to trigger notifications for any metric
- Various options (sampling, %, max, min, etc...)
- Alarm States:
  - OK
  - INSUFFICIENT\_DATA
  - ALARM
- Period:
  - Length of time in seconds to evaluate the metric
  - High resolution custom metrics: 10 sec, 30 sec or multiples of 60 sec

# CloudWatch Alarm Targets

- Stop, Terminate, Reboot, or Recover an EC2 Instance
- Trigger Auto Scaling Action
- Send notification to SNS (from which you can do pretty much anything)



Amazon EC2



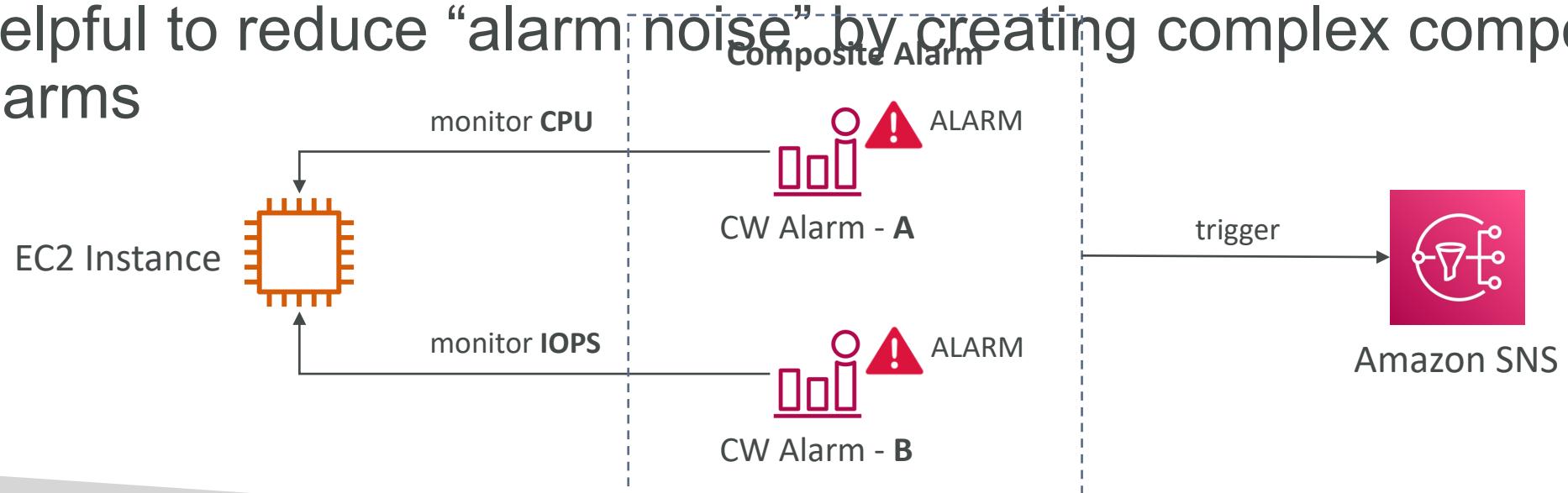
EC2 Auto Scaling



Amazon SNS

# CloudWatch Alarms – Composite Alarms

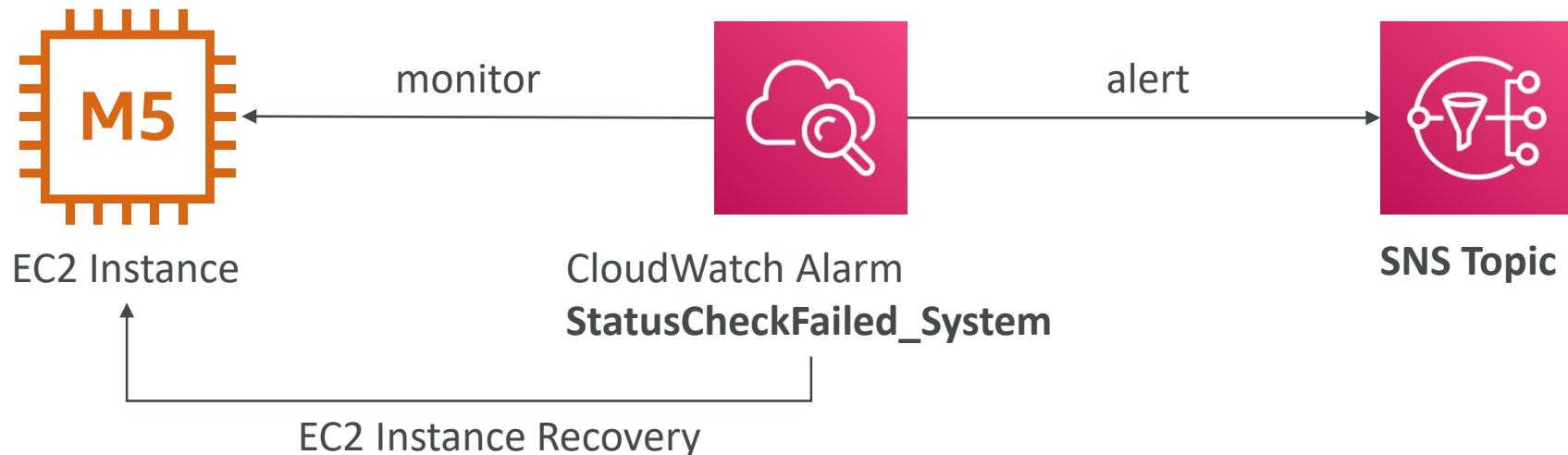
- CloudWatch Alarms are on a single metric
- **Composite Alarms are monitoring the states of multiple other alarms**
- AND and OR conditions
- Helpful to reduce “alarm noise” by creating complex composite alarms



# EC2 Instance Recovery

- Status Check:

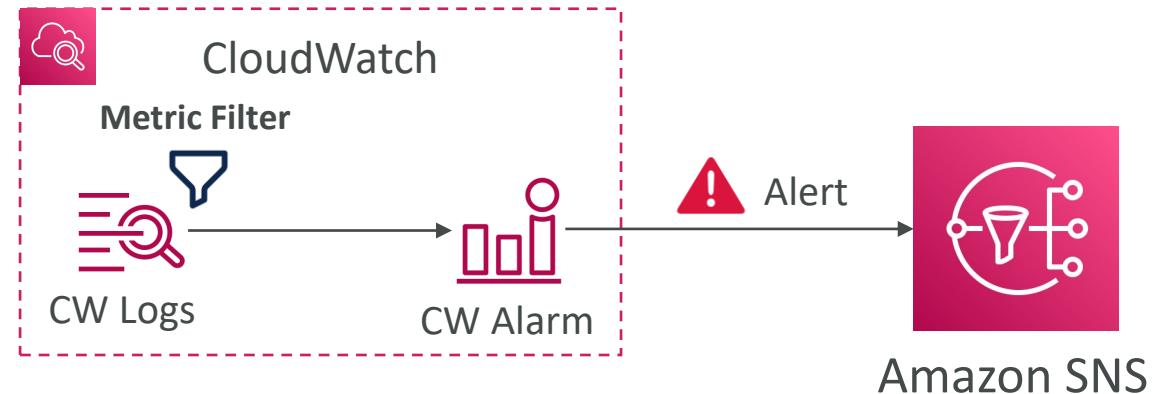
- Instance status = check the EC2 VM
- System status = check the underlying hardware



- **Recovery:** Same Private, Public, Elastic IP, metadata, placement group

# CloudWatch Alarm: good to know

- Alarms can be created based on CloudWatch Logs Metrics Filters



- To test alarms and notifications, set the alarm state to Alarm using CLI

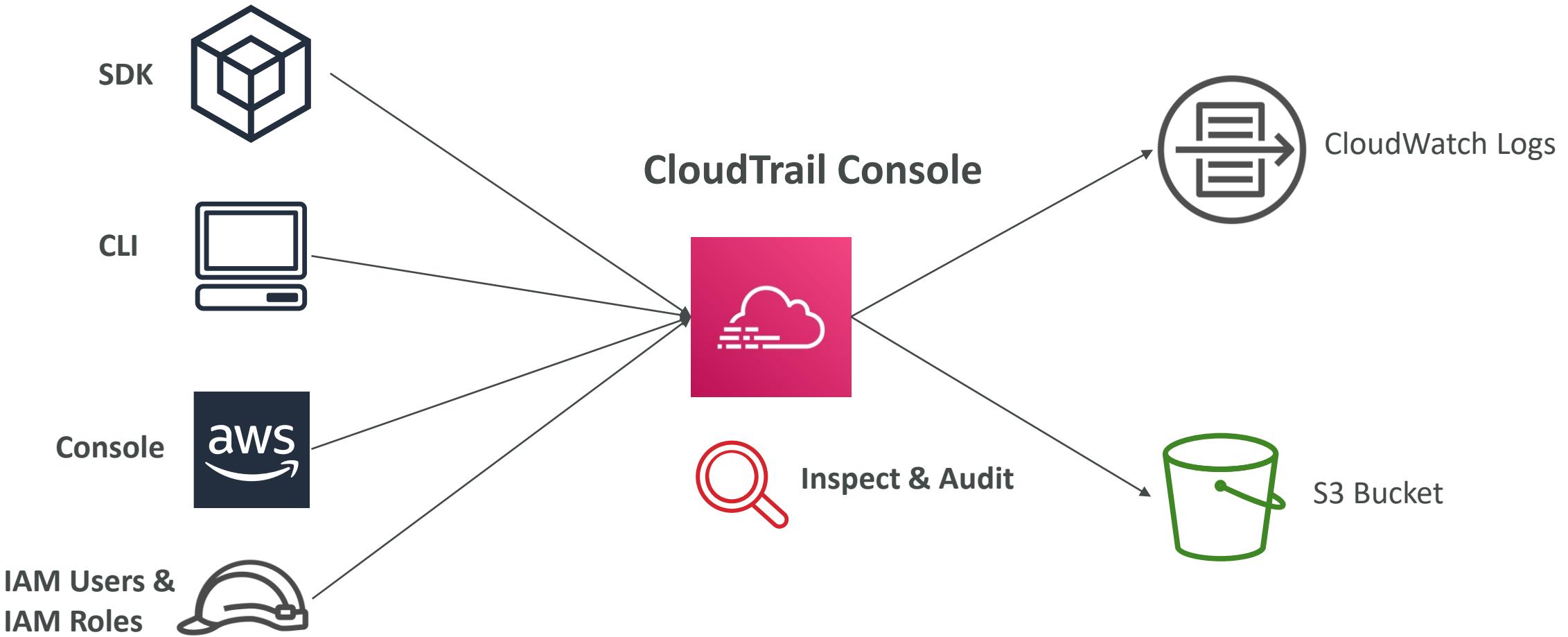
```
aws cloudwatch set-alarm-state --alarm-name "myalarm" --  
state-value ALARM --state-reason "testing purposes"
```

# 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 or S3
- A trail can be applied to All Regions (default) or a single Region.
- If a resource is deleted in AWS, investigate CloudTrail first!

# CloudTrail Diagram



# CloudTrail Events

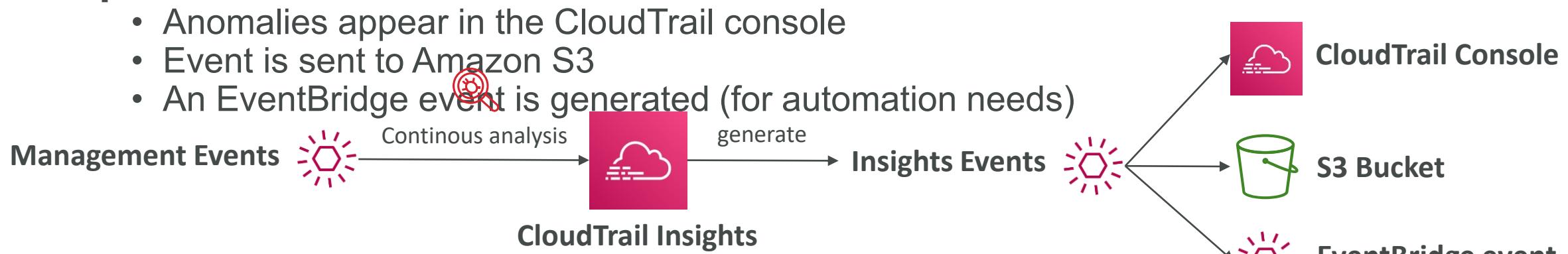


- **Management Events:**
  - Operations that are performed on resources in your AWS account
  - Examples:
    - Configuring security (IAM AttachRolePolicy)
    - Configuring rules for routing data (Amazon EC2 CreateSubnet)
    - Setting up logging (AWS CloudTrail CreateTrail)
  - **By default, trails are configured to log management events.**
  - Can separate **Read Events** (that don't modify resources) from **Write Events** (that may modify resources)
- **Data Events:**
  - **By default, data events are not logged (because high volume operations)**
  - Amazon S3 object-level activity (ex: GetObject, DeleteObject, PutObject): can separate Read and Write Events
  - AWS Lambda function execution activity (the Invoke API)
- **CloudTrail Insights Events:**
  - See next slide ☺

# CloudTrail Insights

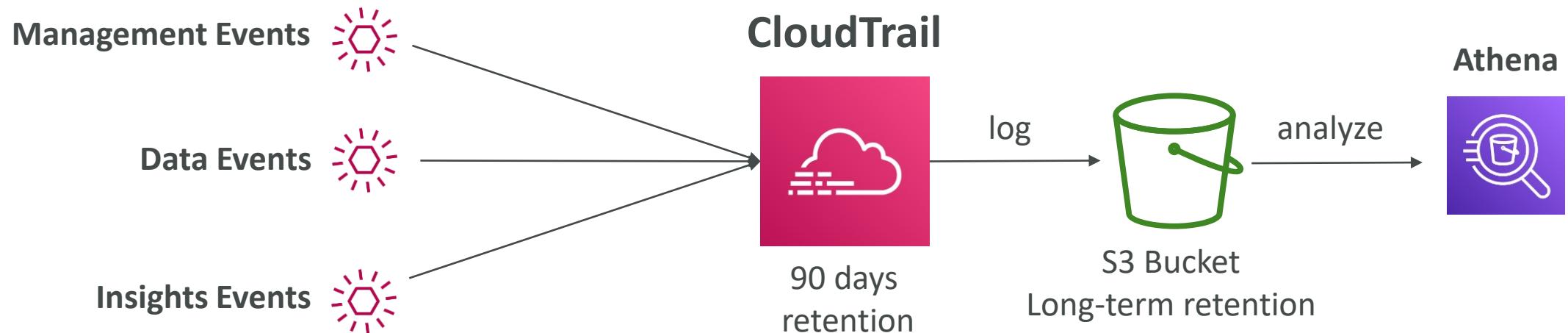


- Enable **CloudTrail Insights** to detect unusual activity in your account:
  - inaccurate resource provisioning
  - hitting service limits
  - Bursts of AWS IAM actions
  - Gaps in periodic maintenance activity
- CloudTrail Insights analyzes normal management events to create a baseline
- And then **continuously analyzes write events to detect unusual patterns**



# CloudTrail Events Retention

- Events are stored for 90 days in CloudTrail
- To keep events beyond this period, log them to S3 and use Athena



# AWS CloudTrail Lake

- Managed data lake for CloudTrail events
- Integrates collection, storage, preparation, and optimization for analysis & query
  - Events are converted to ORC format
- Enables querying CloudTrail data with SQL
- Enable it with the “Create event data store” menu choice in the console
- Data is retained for up to 7 years

**General details Info**  
Enter general details about your event data store.

**Event data store name**  
Enter a display name for your store.  
  
3-128 characters. Only letters, numbers, periods, underscores, and dashes are allowed.

**Retention period**  
Enter the number of days of data to retain in your data store.  
  
Only integers between 7 and 2555 are allowed.

Include only the current region (us-east-1) in my event data store

Enable for all accounts in my organization  
To review accounts in your organization, open AWS Organizations. [See all accounts](#)

# AWS CloudTrail Lake

- Specify the event types you want to track
- Note KMS events add up fast and can make your costs blow up
- Basic event selectors can be selected in the UI
- Finer grained selection may be achieved with advanced event selectors
  - This can help control your ingestion and storage costs
- You can create “channels” to integrate with events outside of AWS
  - Built-in support for Okta, LaunchDarkly, Clumio, and other CloudTrail partners
  - Or custom integrations

The screenshot shows two panels of the AWS CloudTrail Lake configuration interface.

**Event type** Info  
Choose the type of events you want to add to your event data store.

Management events  
Capture management operations performed on your AWS resources.

Data events  
Log the resource operations performed on or within a resource.

**Management events** Info  
Management events show information about management operations performed on resources in your AWS account.

**API activity**  
Choose the activities you want to log.

Read     Write

Exclude AWS KMS events

Exclude Amazon RDS Data API events

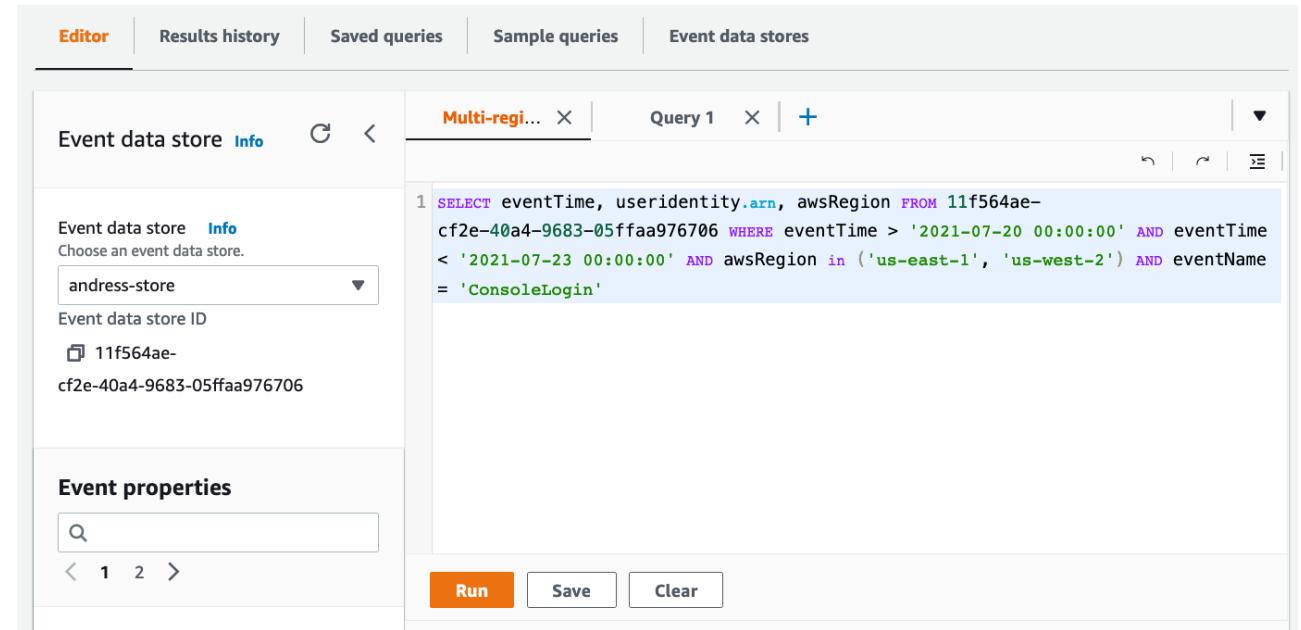
# Example of an advanced event selector

```
aws cloudtrail put-event-selectors --trail-name TrailName \
--advanced-event-selectors
' [
{
  "Name": "Log readOnly and writeOnly management events",
  "FieldSelectors": [
    { "Field": "eventCategory", "Equals": ["Management"] }
  ]
},
{
  "Name": "Log PutObject and DeleteObject events for two S3 prefixes",
  "FieldSelectors": [
    { "Field": "eventCategory", "Equals": ["Data"] },
    { "Field": "resources.type", "Equals": ["AWS::S3::Object"] },
    { "Field": "eventName", "Equals": ["PutObject","DeleteObject"] },
    { "Field": "resources.ARN", "StartsWith": ["arn:aws:s3::::mybucket/prefix","arn:aws:s3::::mybucket2/prefix2"] }
  ]
}
]'
```

Logs all management events, plus PutObject and DeleteObject calls in specified bucket prefixes.

# Querying CloudTrail Lake

- Lake dashboards allow you to visualize events
- Roll your own SQL queries
- Start from sample queries in the CloudTrail Lake Editor
- Remember to bound your queries by eventTime to constrain costs

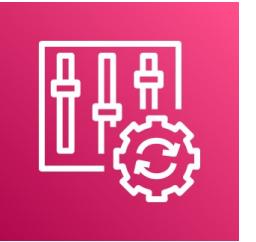


The screenshot shows the CloudTrail Lake Editor interface. At the top, there are tabs for 'Editor' (which is selected), 'Results history', 'Saved queries', 'Sample queries', and 'Event data stores'. Below the tabs, there's a navigation bar with 'Event data store' (selected), 'Info', and a back arrow. A dropdown menu shows 'andress-store' and a list item '11f564ae-'. On the left, there's a sidebar titled 'Event properties' with a search bar and a page number indicator (< 1 2 >). The main area contains a query editor with a tab labeled 'Multi-regi...' and a sub-tab 'Query 1'. The query itself is:

```
1 SELECT eventTime, useridentity.arn, awsRegion FROM 11f564ae-  
cf2e-40a4-9683-05ffaa976706 WHERE eventTime > '2021-07-20 00:00:00' AND eventTime  
< '2021-07-23 00:00:00' AND awsRegion in ('us-east-1', 'us-west-2') AND eventName  
= 'ConsoleLogin'
```

At the bottom of the editor are three buttons: 'Run', 'Save', and 'Clear'.

# AWS Config



- Helps with auditing and recording **compliance** of your AWS resources
- Helps record configurations and changes over time
- Questions that can be solved by AWS Config:
  - Is there unrestricted SSH access to my security groups?
  - Do my buckets have any public access?
  - How has my ALB configuration changed over time?
- You can receive alerts (SNS notifications) for any changes
- AWS Config is a per-region service
- Can be aggregated across regions and accounts
- Possibility of storing the configuration data into S3 (analyzed by Athena)

# Config Rules

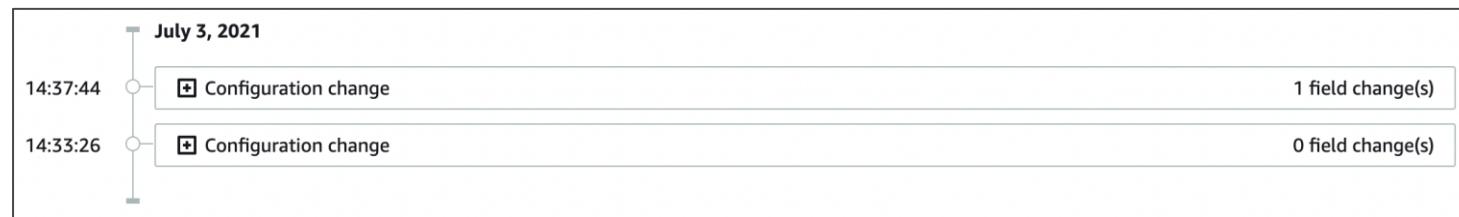
- Can use AWS managed config rules (over 75)
- Can make custom config rules (must be defined in AWS Lambda)
  - Ex: evaluate if each EBS disk is of type gp2
  - Ex: evaluate if each EC2 instance is t2.micro
- Rules can be evaluated / triggered:
  - For each config change
  - And / or: at regular time intervals
- **AWS Config Rules does not prevent actions from happening (no deny)**
- Pricing: no free tier, \$0.003 per configuration item recorded per region, \$0.001 per config rule evaluation per region

# AWS Config Resource

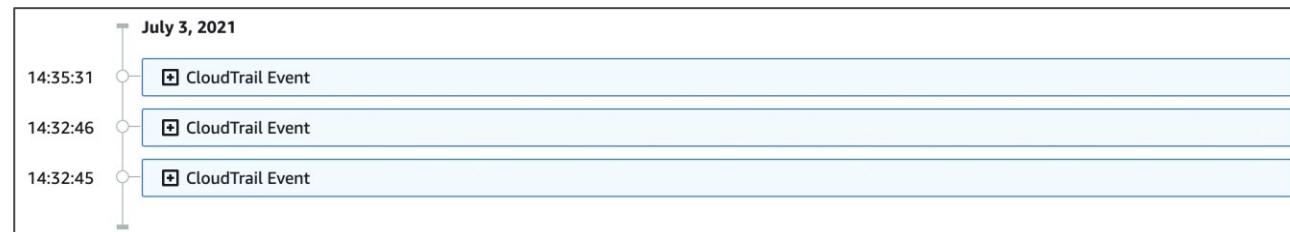
- View compliance of a resource over time

○ sg-077b425b1649da83e	EC2 SecurityGroup	✓ Compliant
○ sg-0831434f1876c0c74	EC2 SecurityGroup	⚠ Noncompliant
○ sg-09f10ed254d464f30	EC2 SecurityGroup	✓ Compliant

- View configuration of a resource over time

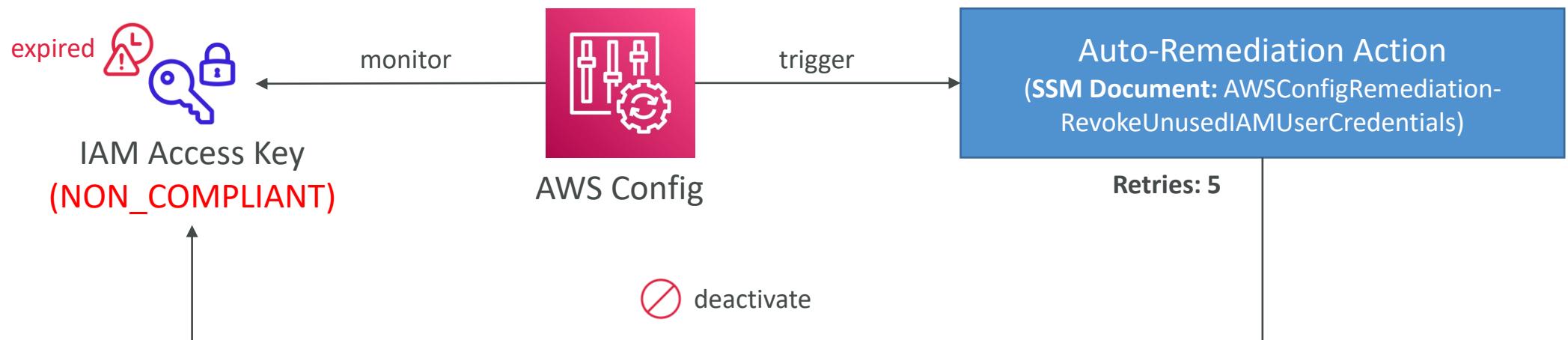


- View CloudTrail API calls of a resource over time



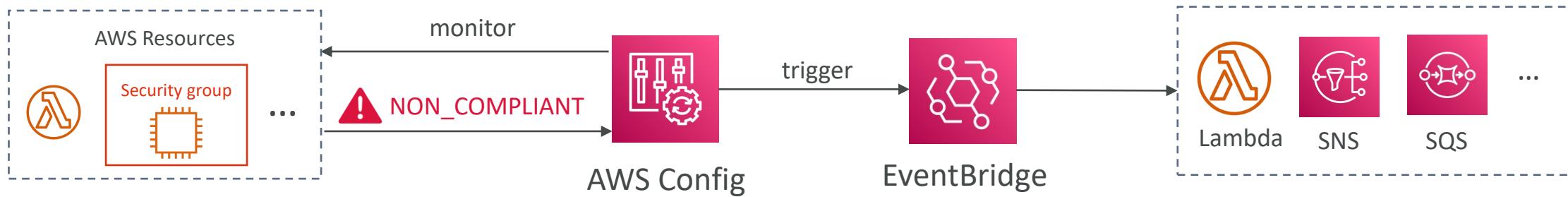
# Config Rules – Remediations

- Automate remediation of non-compliant resources using SSM Automation Documents
- Use AWS-Managed Automation Documents or create custom Automation Documents
  - Tip: you can create custom Automation Documents that invokes Lambda function
- You can set **Remediation Retries** if the resource is still non-compliant after auto-remediation

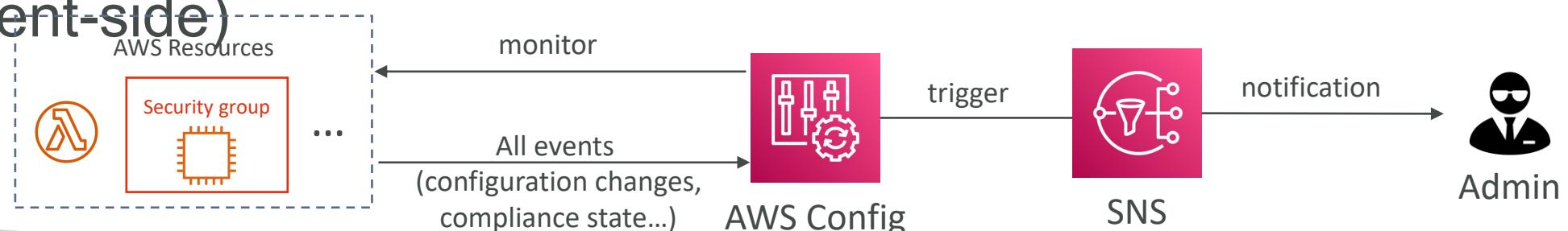


# Config Rules – Notifications

- Use EventBridge to trigger notifications when AWS resources are non-compliant



- Ability to send configuration changes and compliance state notifications to SNS (all events – use SNS Filtering or filter at client-side)



# CloudWatch vs CloudTrail vs Config

- CloudWatch
  - Performance monitoring (metrics, CPU, network, etc...) & dashboards
  - Events & Alerting
  - Log Aggregation & Analysis
- CloudTrail
  - Record API calls made within your Account by everyone
  - Can define trails for specific resources
  - Global Service
- Config
  - Record configuration changes
  - Evaluate resources against compliance rules
  - Get timeline of changes and compliance

# For an Elastic Load Balancer

- CloudWatch:
  - Monitoring Incoming connections metric
  - Visualize error codes as % over time
  - Make a dashboard to get an idea of your load balancer performance
- Config:
  - Track security group rules for the Load Balancer
  - Track configuration changes for the Load Balancer
  - Ensure an SSL certificate is always assigned to the Load Balancer (compliance)
- CloudTrail:
  - Track who made any changes to the Load Balancer with API calls

# What is CloudFormation



- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
  - I want a security group
  - I want two EC2 instances using this security group
  - I want an S3 bucket
  - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

# Benefits of AWS CloudFormation (1/2)

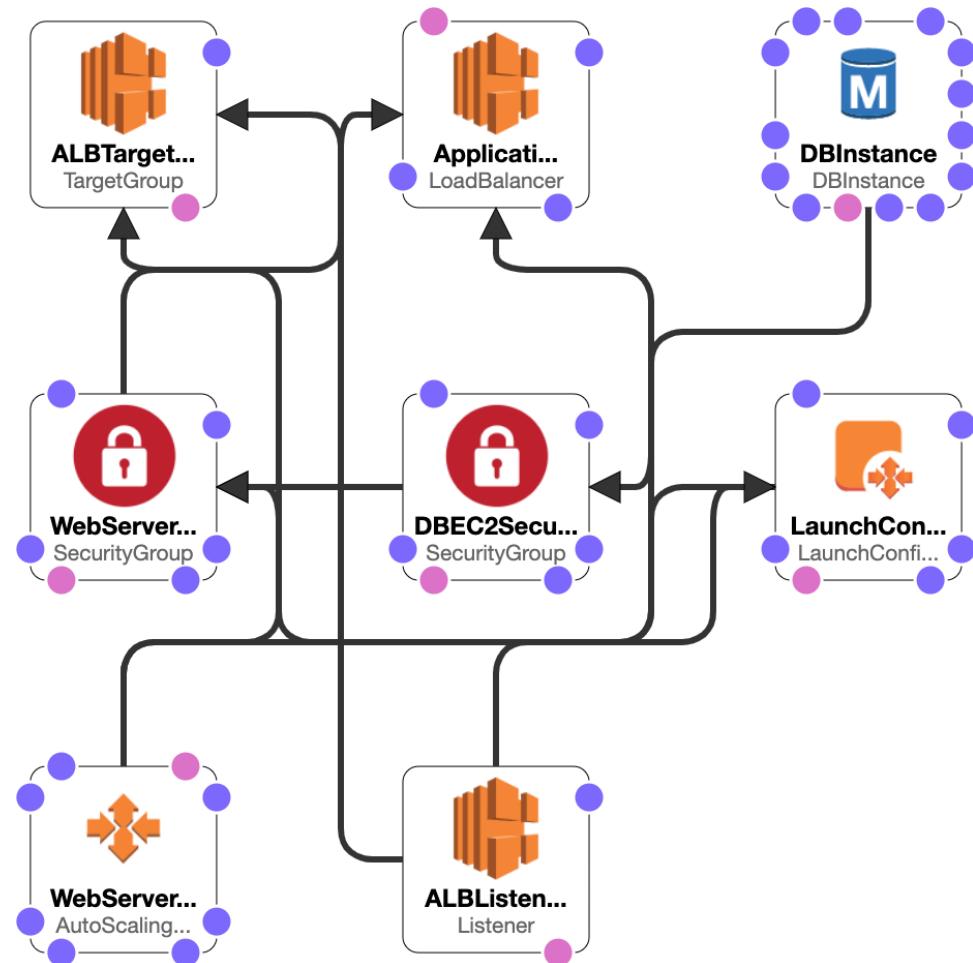
- **Infrastructure as code**
  - No resources are manually created, which is excellent for control
  - Changes to the infrastructure are reviewed through code
- **Cost**
  - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
  - You can estimate the costs of your resources using the CloudFormation template
  - **Savings strategy:** In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

# Benefits of AWS CloudFormation (2/2)

- Productivity
  - Ability to destroy and re-create an infrastructure on the cloud on the fly
  - Automated generation of Diagram for your templates!
  - Declarative programming (no need to figure out ordering and orchestration)
- Don't re-invent the wheel
  - Leverage existing templates on the web!
  - Leverage the documentation
- **Supports (almost) all AWS resources:**
  - Everything we'll see in this course is supported
  - You can use "custom resources" for resources that are not supported

# CloudFormation Stack Designer

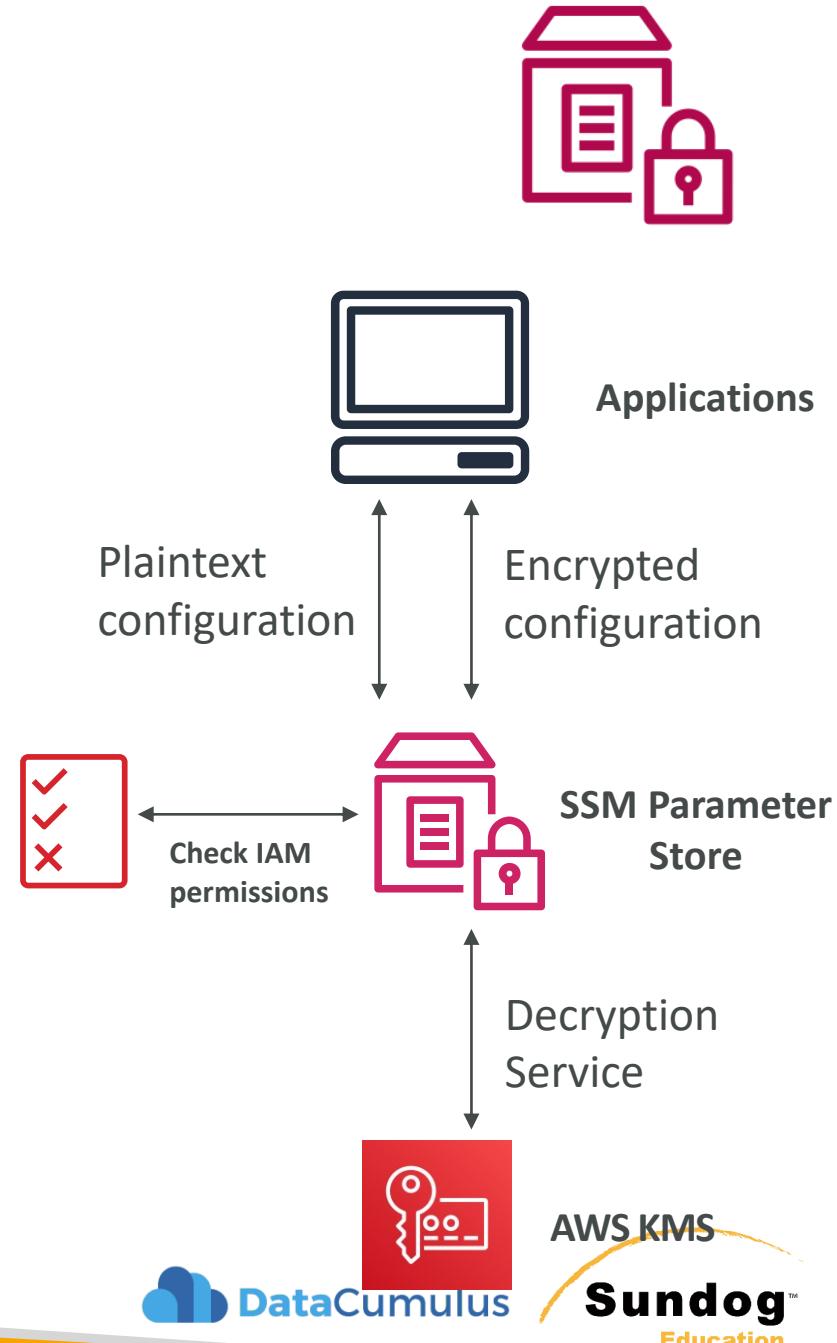
- Example: WordPress CloudFormation Stack
- We can see all the **resources**
- We can see the **relations** between the components



# SSM Parameter Store

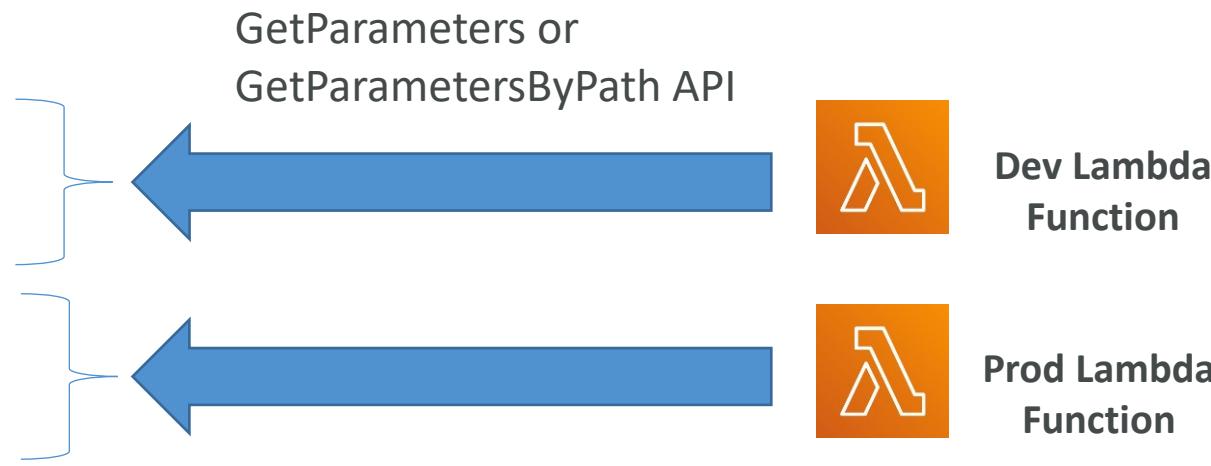
Secure storage for configuration and secrets

- Optional Seamless Encryption using KMS
- Serverless, scalable, durable, easy SDK
- Version tracking of configurations / secrets
- Security through IAM
- Notifications with Amazon EventBridge
- Integration with CloudFormation



# SSM Parameter Store Hierarchy

- /my-department/
  - my-app/
    - dev/
      - db-url
      - db-password
    - prod/
      - db-url
      - db-password
    - other-app/
  - /other-department/
  - /aws/reference/secretsmanager/secret\_ID\_in\_Secrets\_Manager
  - /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-gp2 (public)



# Standard and advanced parameter tiers

	Standard	Advanced
Total number of parameters allowed (per AWS account and Region)	10,000	100,000
Maximum size of a parameter value	4 KB	8 KB
Parameter policies available	No	Yes
Cost	No additional charge	Charges apply
Storage Pricing	Free	\$0.05 per advanced parameter per month

# Parameters Policies (for advanced parameters)

- Allow to assign a TTL to a parameter (expiration date) to force updating or deleting sensitive data such as passwords
- Can assign multiple policies at a time

Expiration (to delete a parameter)

```
{  
  "Type": "Expiration",  
  "Version": "1.0",  
  "Attributes": {  
    "Timestamp": "2020-12-02T21:34:33.000Z"  
  }  
}
```

ExpirationNotification (EventBridge)

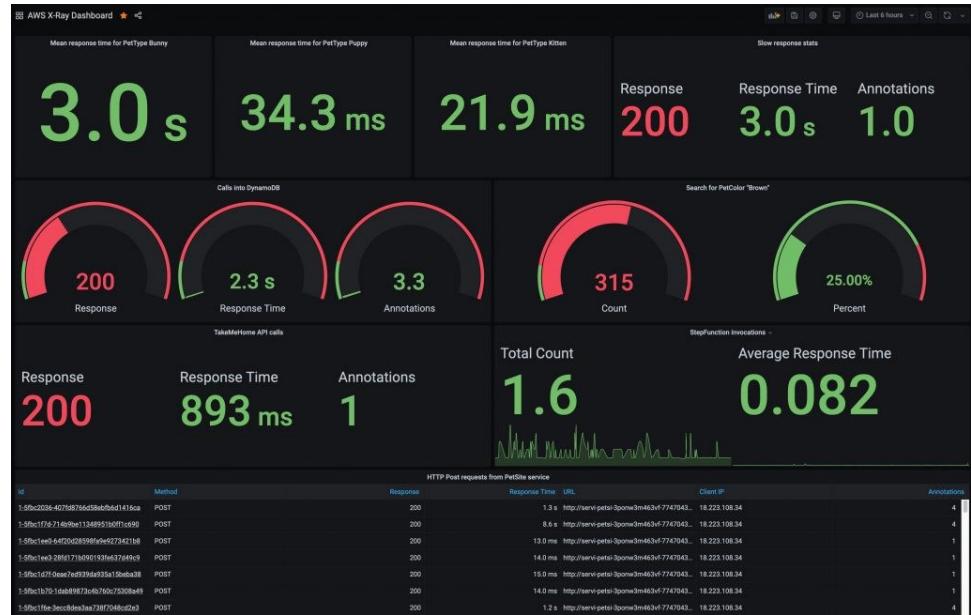
```
{  
  "Type": "ExpirationNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "Before": "15",  
    "Unit": "Days"  
  }  
}
```

NoChangeNotification (EventBridge)

```
{  
  "Type": "NoChangeNotification",  
  "Version": "1.0",  
  "Attributes": {  
    "After": "20",  
    "Unit": "Days"  
  }  
}
```

# Amazon Managed Grafana

- Grafana is a popular open-source platform used to monitor, visualize, and alert on metrics and logs.
- Integrated with IAM Identity Center (formerly AWS SSO) and/or SAML for user management and permissions
- Compatible with Grafana plugins and alerts
- Fully managed, scales automatically
- Encrypted at rest and in transit, can use KMS.



# Amazon Managed Grafana

- Integrated with many AWS data sources
  - CloudWatch, OpenSearch, Timestream, Athena, Redshift, X-Ray
  - Amazon Managed Service for Prometheus (AMP)
- Also with everything else Grafana integrates with
  - GitHub, Google, Azure, MySQL, Redis, JSON, OpenTelemetry, much more.



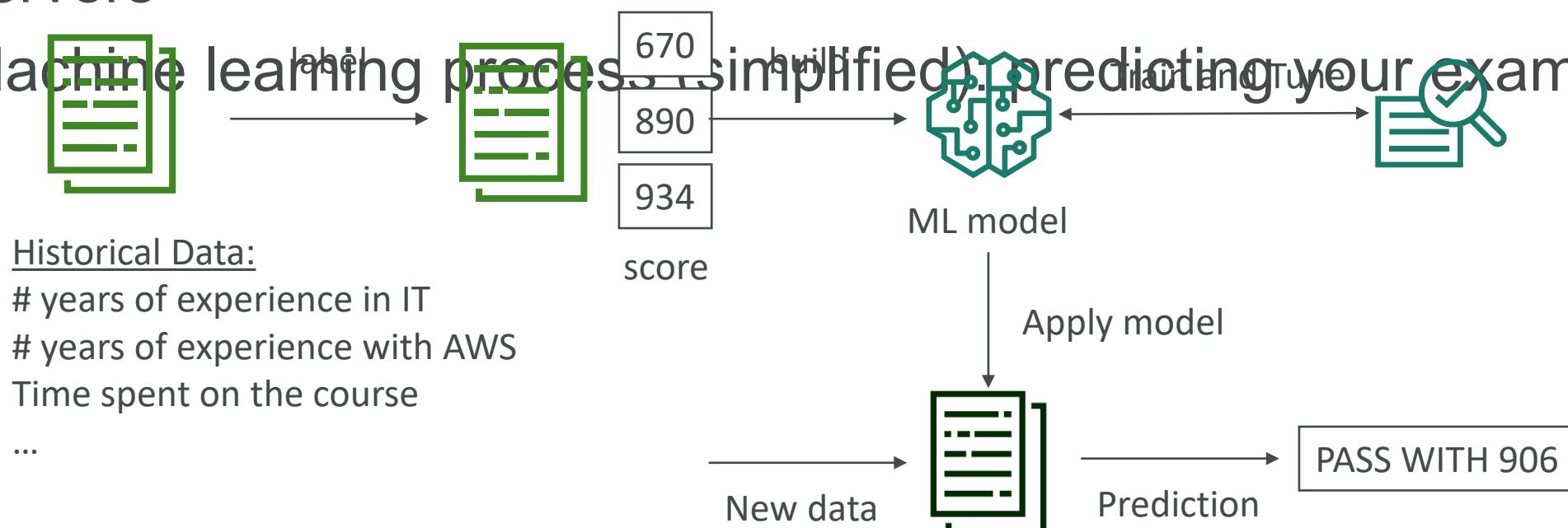
# Machine Learning

Learning from data in AWS

# Amazon SageMaker

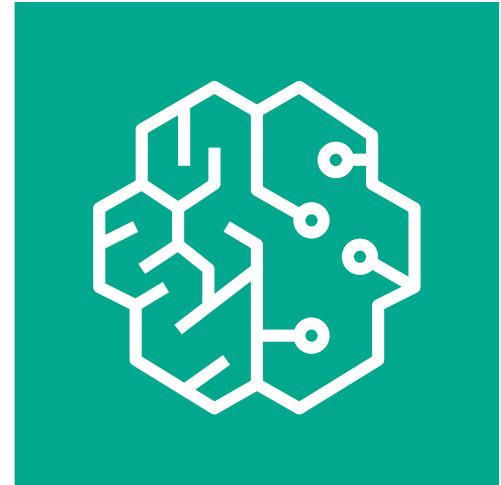


- Fully managed service for developers / data scientists to build ML models
- Typically, difficult to do all the processes in one place + provision servers
- Machine learning process simplified: predicting your exam score



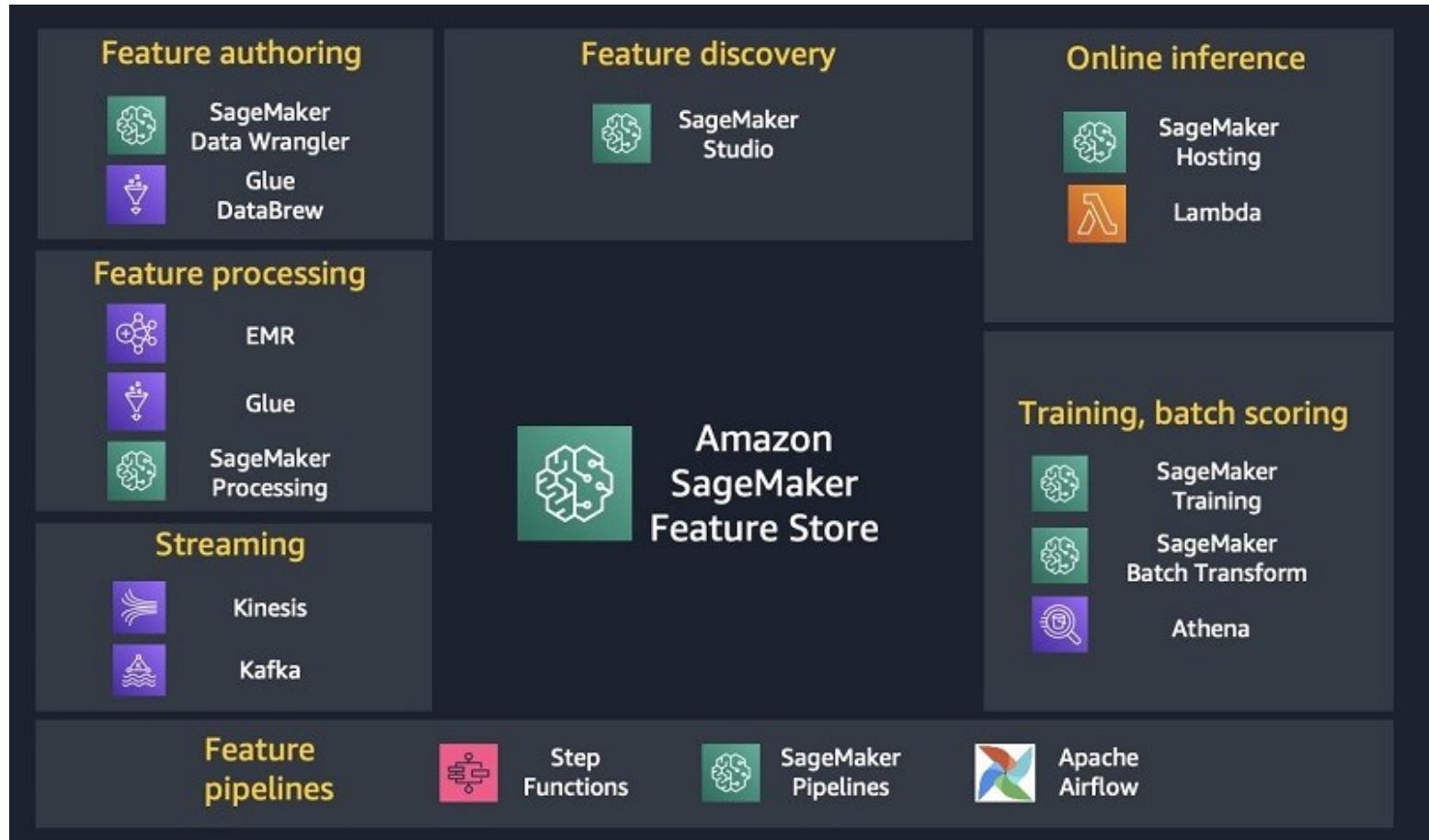
# SageMaker Feature Store

- A “feature” is just a property used to train a machine learning model.
  - Like, you might predict someone’s political party based on “features” such as their address, income, age, etc.
- Machine learning models require fast, secure access to feature data for training.
- It’s also a challenge to keep it organized and share features across different models.



Amazon SageMaker  
Feature Store

# Where the features come from is up to you.

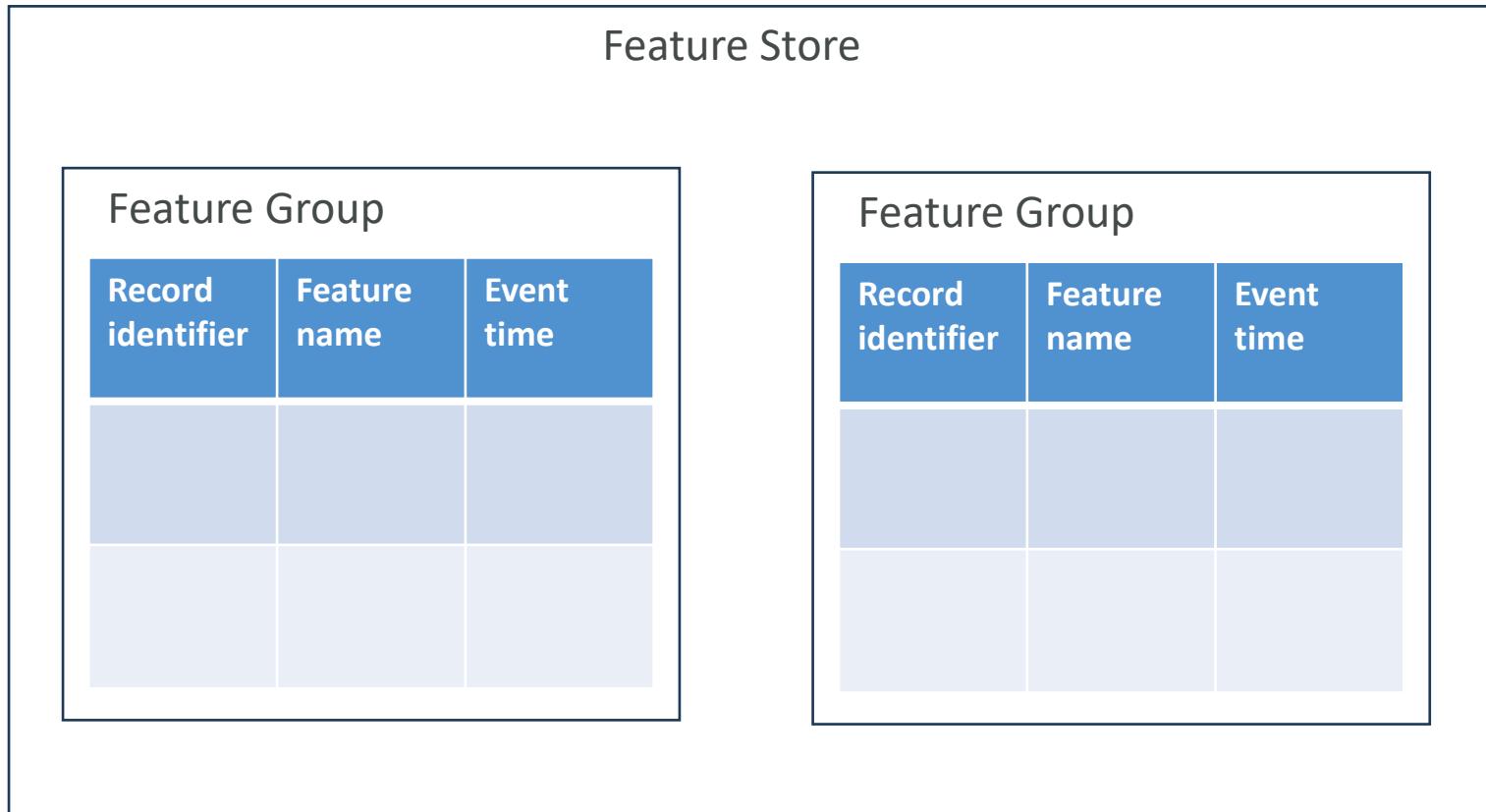


AWS

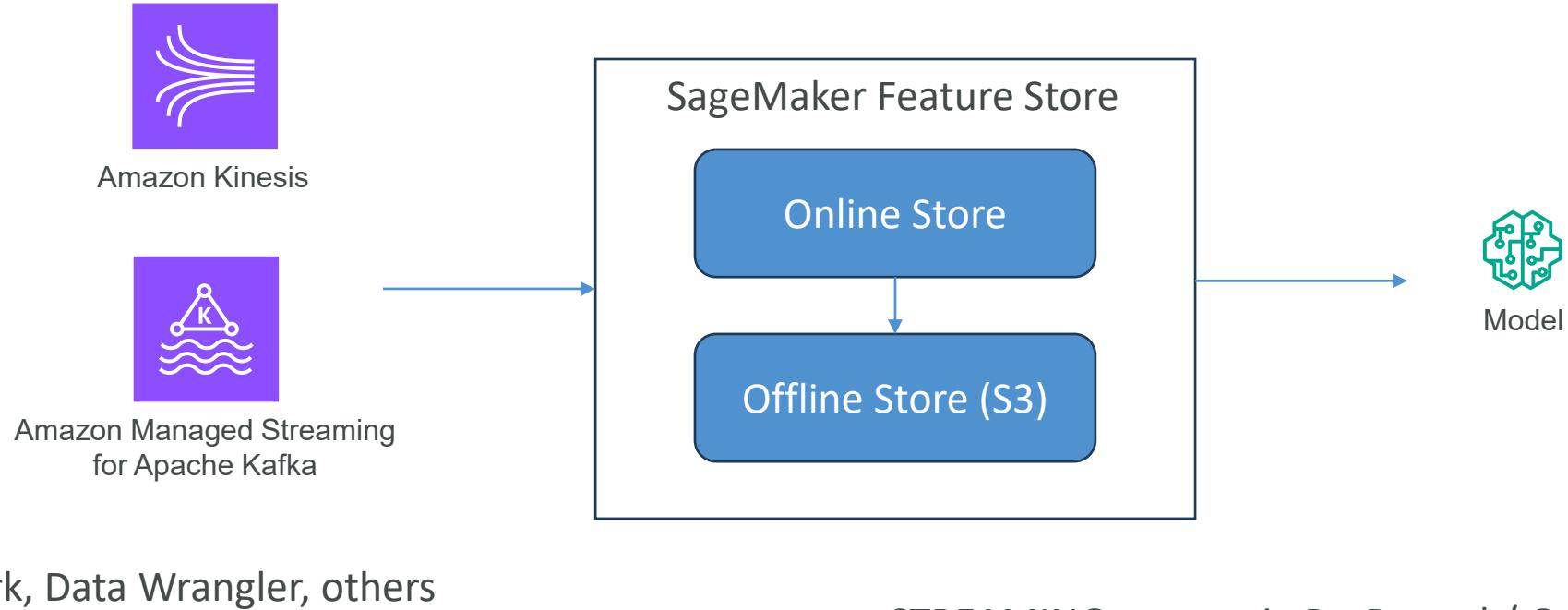
DataCumulus

 Sundog  
Education

# How SageMaker Feature Store Organizes Your Data



# Data Ingestion (streaming or batch)

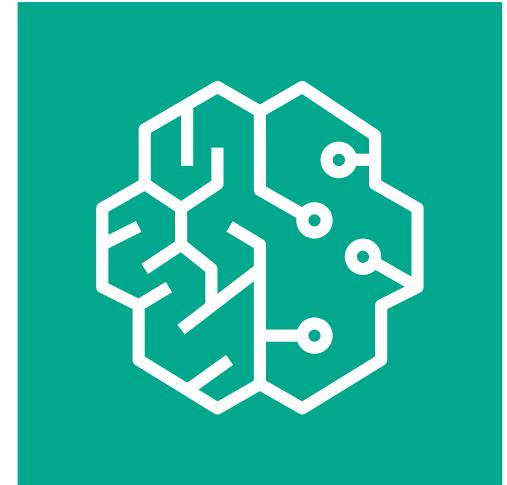


STREAMING access via PutRecord / GetRecord API's

BATCH access via the offline S3 store (use with anything that hits S3, like Athena, Data Wrangler. Automatically creates a Glue Data Catalog for you.)

# SageMaker Feature Store Security

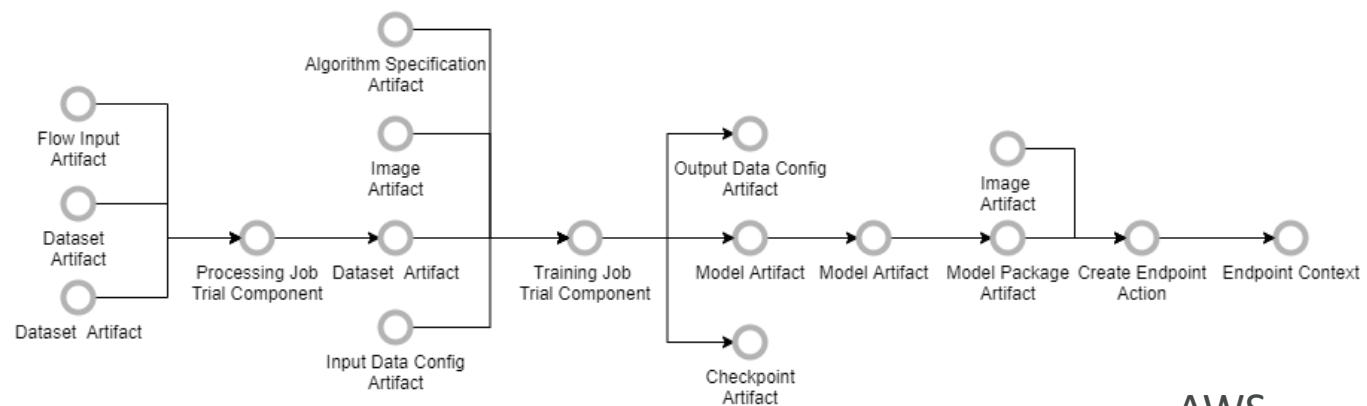
- Encrypted at rest and in transit
- Works with KMS customer master keys
- Fine-grained access control with IAM
- May also be secured with AWS PrivateLink



# SageMaker ML Lineage Tracking

- Creates & stores your ML workflow (MLOps)
- Keep a running history of your models
- Tracking for auditing and compliance
- Automatically or manually-created tracking entities
- Integrates with AWS Resource Access Manager for cross-account lineage
- Sample SageMaker-created lineage graph:

Lineage Metadata  
SageMaker automatically creates a connected graph of lineage entity metadata tracking your workflow.



AWS

 DataCumulus

 Sundog™  
Education

# Lineage Tracking Entities

- Trial component (processing jobs, training jobs, transform jobs)
- Trial (a model composed of trial components)
- Experiment (a group of Trials for a given use case)
- Context (logical grouping of entities)
- Action (workflow step, model deployment)
- Artifact (Object or data, such as an S3 bucket or an image in ECR)
- Association (connects entities together) – has optional AssociationType:
  - ContributedTo
  - AssociatedWith
  - DerivedFrom
  - Produced
  - SameAs

# Querying Lineage Entities

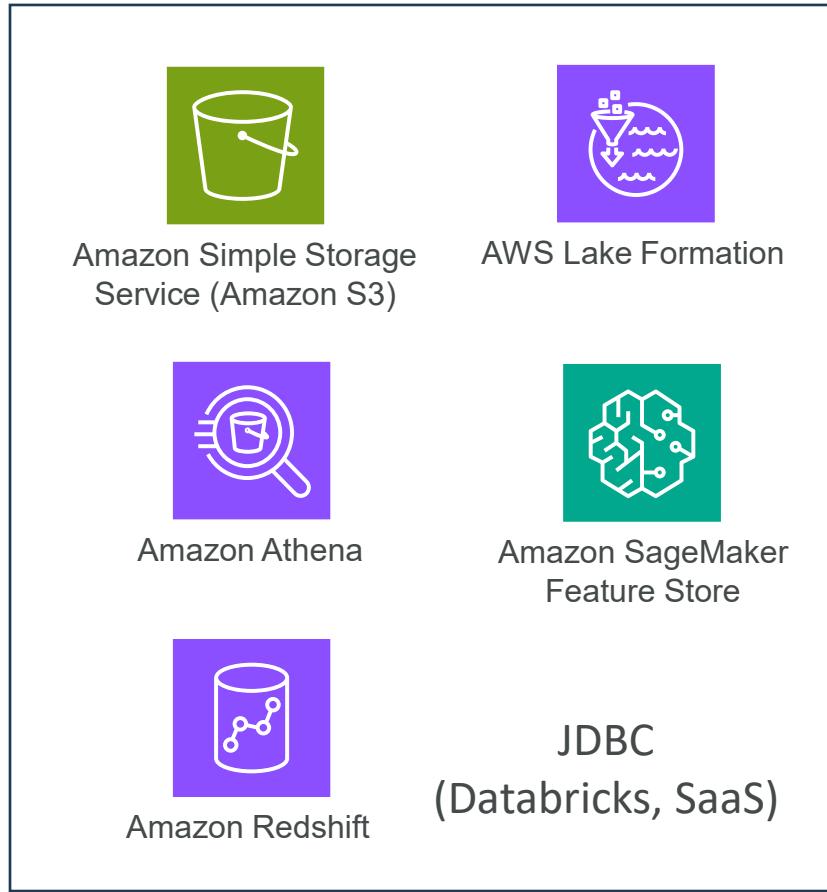
- Use the LineageQuery API from Python
  - Part of the Amazon SageMaker SDK for Python
- Do things like find all models / endpoints / etc. that use a given artifact
- Produce a visualization
  - Requires external Visualizer helper class

# SageMaker Data Wrangler

- Visual interface (in SageMaker Studio) to prepare data for machine learning
- Import data
- Visualize data
- Transform data (300+ transformations to choose from)
  - Or integrate your own custom xforms with pandas, PySpark, PySpark SQL
- “Quick Model” to train your model with your data and measure its results



# Data Wrangler sources



# Data Wrangler: Import Data

The screenshot shows the Amazon SageMaker Studio Data Wrangler interface. The left sidebar displays a file tree with an 'Untitled Folder' containing a file named 'titanic.flow'. The main workspace is titled 'titanic.flow' and has tabs for 'Import', 'Prepare', 'Analyze', and 'Export', with 'Import' selected. A sub-header indicates 'Data sources / S3 source / sagemaker-us-east-2-613904931467 / titanic'. Below this, a section titled 'Import a dataset from S3' provides instructions to browse S3 and select a file. A table lists an object named 'titanic-train.csv' with a size of 58.89KB and a last modified date of 2020-10-15 08:49:45+00:00. On the right, the 'DETAILS' panel shows the 'Name' field set to 'titanic-train.csv' (marked as 'Required') and the 'URI' field set to 's3://sagemaker-us-east-2-'. It also specifies 'File type' as 'csv' (marked as 'Required'), and two checked options: 'Add header to table' and 'Enable sampling'. An orange 'Import dataset' button is at the bottom. Below the import panel, a 'Preview' section shows a table with 5 rows of data from the titanic dataset.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
1	0	3	Braund, Mr. Owen Harris	male	22	1
2	1	1	Cumings, Mrs. John Bra...	female	38	1
3	1	3	Heikkinen, Miss. Laina	female	26	0
4	1	1	Futrelle, Mrs. Jacques H...	female	35	1
5	0	3	Allen, Mr. William Henry	male	35	0

# Data Wrangler: Preview Data

**Data flow / Transform: titanic-train.csv**

**Configure types**

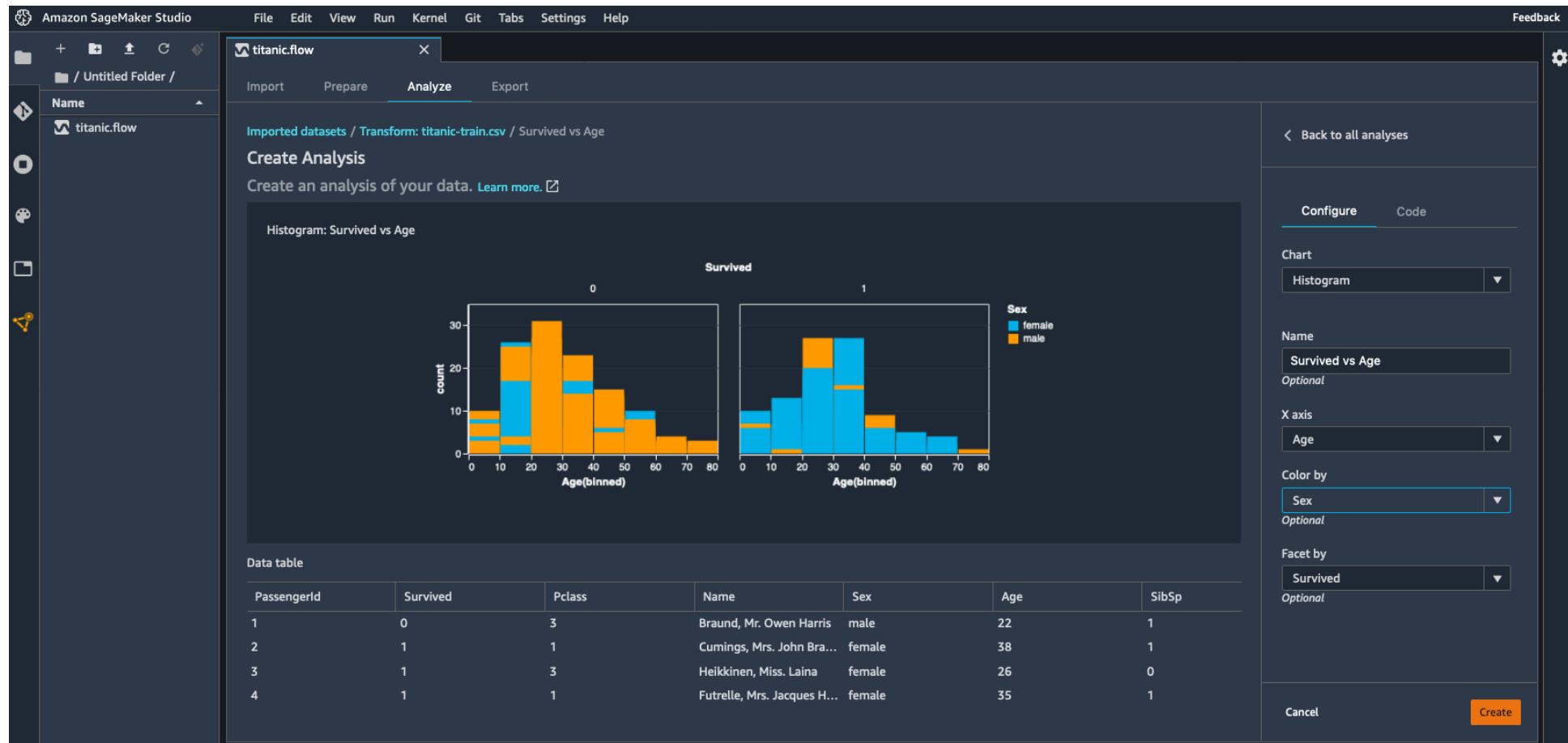
PassengerId (long)	Survived (long)	Pclass (long)	Name (string)	Sex (string)	Age (long)
7	0	1	McCarthy, Mr. Timothy J	male	54
8	0	3	Palsson, Master. Gosta ...	male	2
9	1	3	Johnson, Mrs. Oscar W (...	female	27
10	1	2	Nasser, Mrs. Nicholas (A...	female	14
11	1	3	Sandstrom, Miss. Margu...	female	4
12	1	1	Bonnell, Miss. Elizabeth	female	58
13	0	3	Saunderscock, Mr. Willia...	male	20
14	0	3	Andersson, Mr. Anders J...	male	39
15	0	3	Vestrom, Miss. Hulda A...	female	14
16	1	2	Hewlett, Mrs. (Mary D K...	female	55
17	0	3	Rice, Master. Eugene	male	2
18	1	2	Williams, Mr. Charles Eu...	male	
19	0	3	Vander Planke, Mrs. Juli...	female	31
20	1	3	Masselmani, Mrs. Fatima	female	
21	0	2	Fynney, Mr. Joseph J	male	35
22	1	2	Beesley, Mr. Lawrence	male	34
23	1	3	McGowan, Miss. Anna "...	female	15
24	1	1	Sloper, Mr. William Tho...	male	28
25	0	3	Palsson, Miss. Torborg ...	female	8
26	1	3	Asplund, Mrs. Carl Osca...	female	38
27	0	3	Emir, Mr. Farred Chehab	male	
28	0	1	Fortune, Mr. Charles Ale...	male	19

**CONFIGURE TYPES**

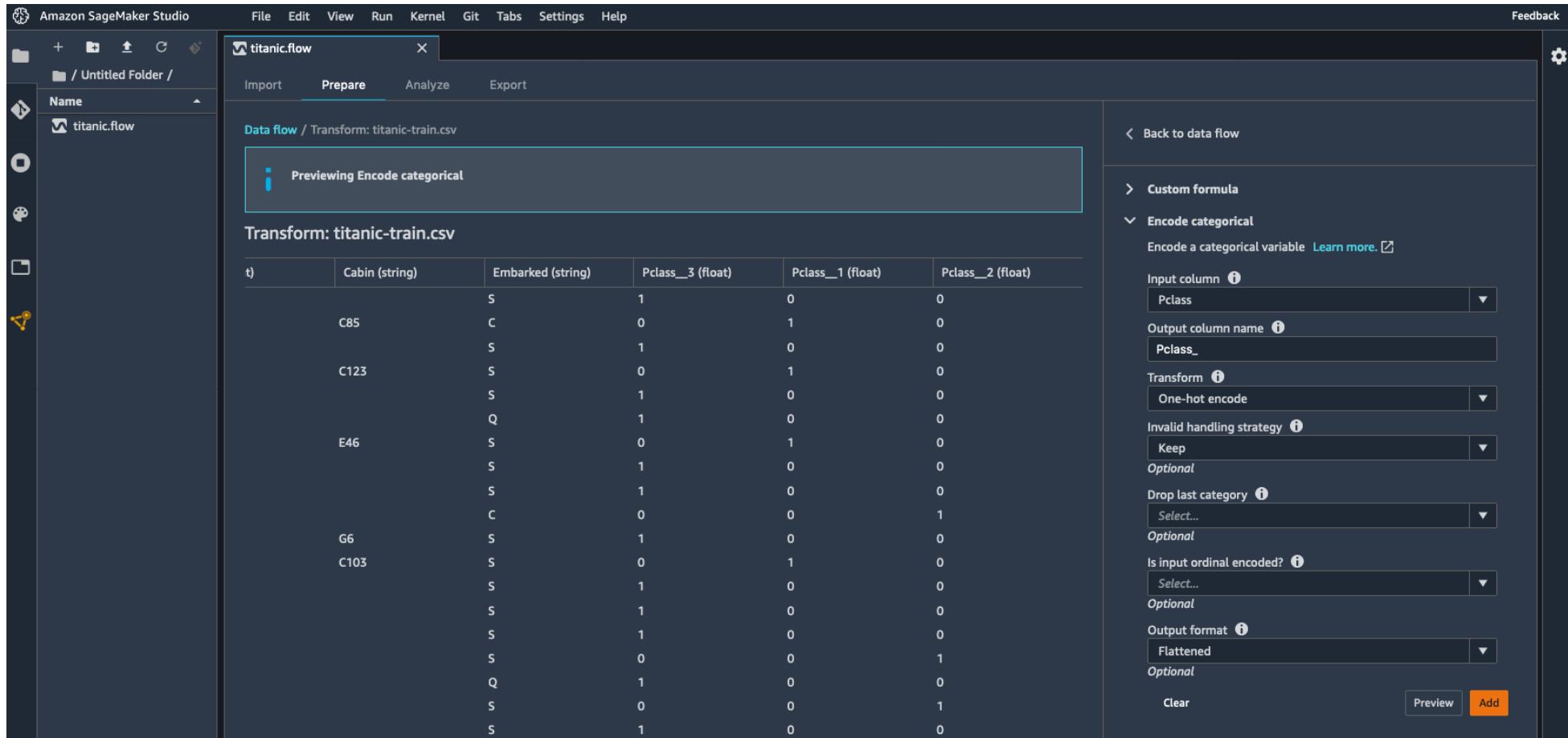
Column name	Type
PassengerId	Long
Survived	Long
Pclass	Long
Name	String
Sex	String
Age	Long
SibSp	Long
Parch	Long
Ticket	String
Fare	Float
Cabin	String
Embarked	String

**Buttons:** Clear, Preview, Apply

# Data Wrangler: Visualize Data



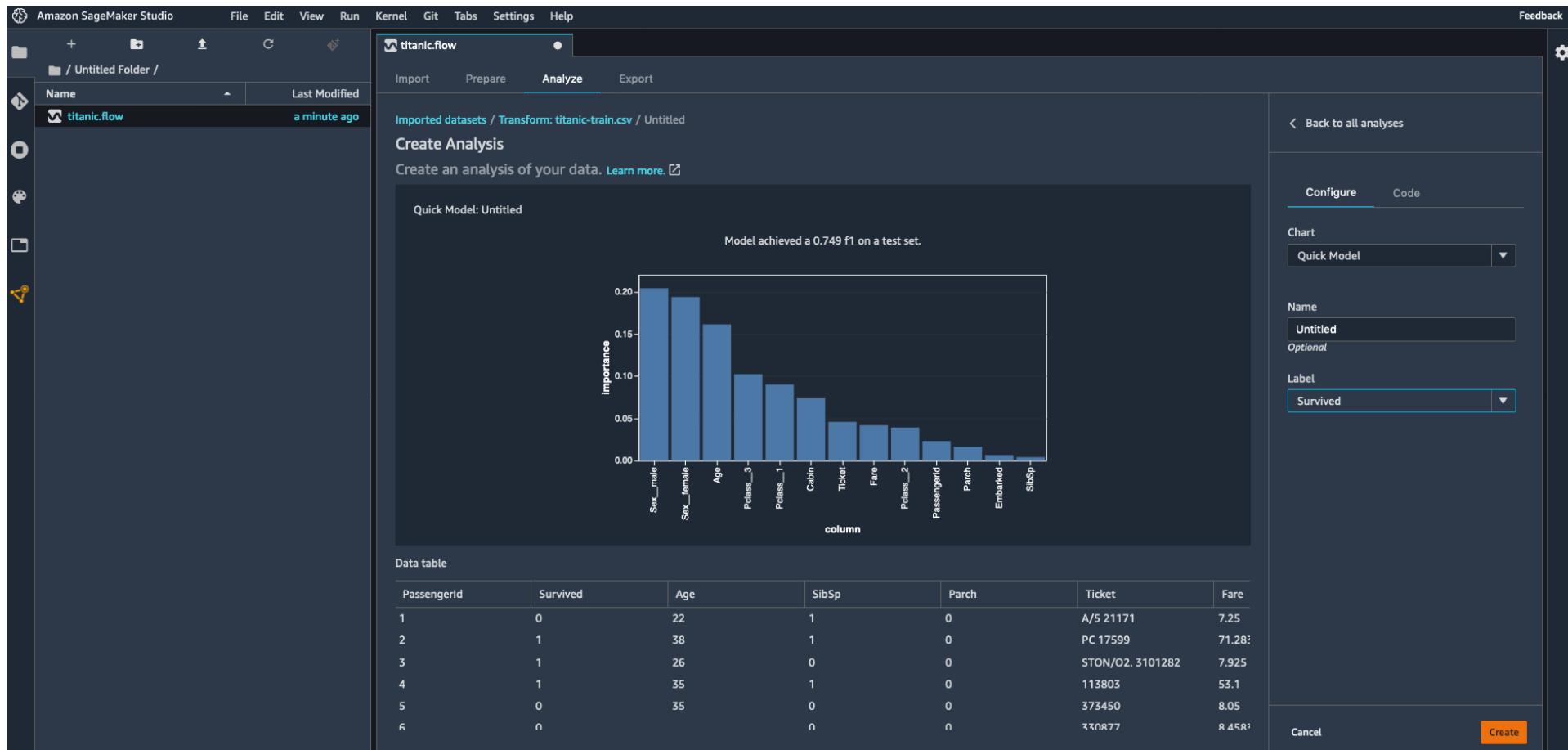
# Data Wrangler: Transform Data



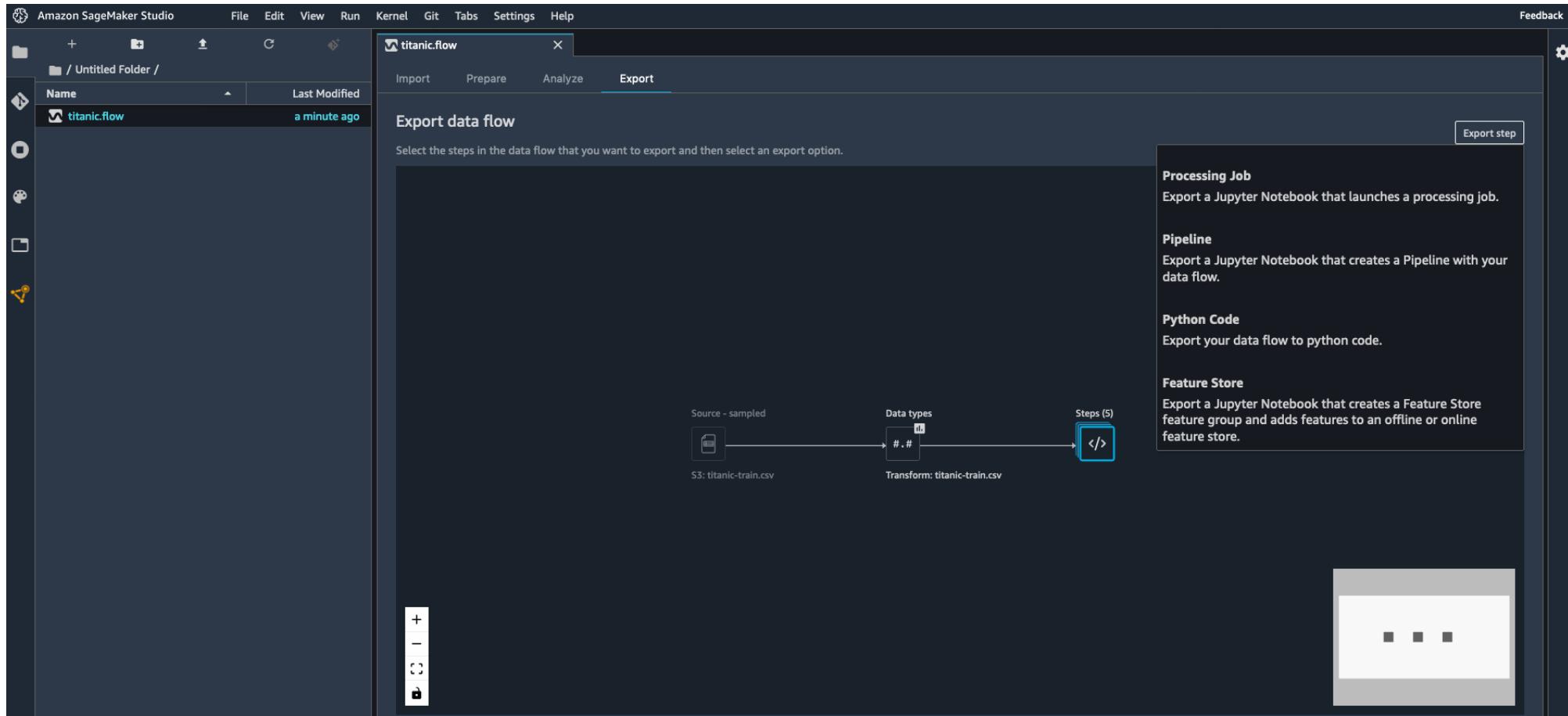
The screenshot shows the Amazon SageMaker Studio Data Wrangler interface. The main window displays a preview of the 'titanic-train.csv' dataset, specifically focusing on the 'Encode categorical' transformation. The preview table includes columns: t, Cabin (string), Embarked (string), Pclass\_3 (float), Pclass\_1 (float), and Pclass\_2 (float). The data shows various cabin and embarked values mapped to numerical values. To the right, the configuration pane for the 'Encode categorical' transform is visible, showing settings for input column (Pclass), output column name (Pclass\_), transform type (One-hot encode), and other options like invalid handling strategy (Keep) and output format (Flattened).

t	Cabin (string)	Embarked (string)	Pclass_3 (float)	Pclass_1 (float)	Pclass_2 (float)
		S	1	0	0
	C85	C	0	1	0
		S	1	0	0
	C123	S	0	1	0
		S	1	0	0
		Q	1	0	0
	E46	S	0	1	0
		S	1	0	0
		S	1	0	0
		C	0	0	1
	G6	S	1	0	0
	C103	S	0	1	0
		S	1	0	0
		S	1	0	0
		S	0	0	1
		Q	1	0	0
		S	0	0	1
		S	1	0	0

# Data Wrangler: Quick Model



# Data Wrangler: Export Data Flow



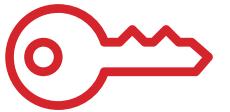
# Data Wrangler Troubleshooting

- Make sure your Studio user has appropriate IAM roles
- Make sure permissions on your data sources allow Data Wrangler access
  - Add AmazonSageMakerFullAccess policy
- EC2 instance limit
  - If you get “The following instance type is not available...” errors
  - May need to request a quota increase
  - Service Quotas / Amazon SageMaker / Studio KernelGateway Apps running on ml.m5.4xlarge instance

# Developer Tools

Developing apps around your data in AWS

# How can users access AWS ?



- To access AWS, you have three options:
  - **AWS Management Console** (protected by password + MFA)
  - **AWS Command Line Interface (CLI)**: protected by access keys
  - **AWS Software Developer Kit (SDK)** - for code: protected by access keys
- Access Keys are generated through the AWS Console
- Users manage their own access keys
- **Access Keys are secret, just like a password. Don't share them**
- Access Key ID ~ = username
- Secret Access Key ~ = password

# Example (Fake) Access Keys

## Access keys

Use access keys to make secure REST or HTTP Query protocol requests to AWS service APIs. For your protection, you should never share your secret keys with anyone. As a best practice, we recommend frequent key rotation. [Learn more](#)

[Create access key](#)

Access key ID	Created	Last used	Status	
AKIASK4E37PV4TU3RD6C	2020-05-25 15:13 UTC+0100	N/A	Active	<a href="#">Make inactive</a> 

- Access key ID: AKIASK4E37PV4983d6C
- Secret Access Key:  
AZPN3z0jWozWCndljhB0Unh8239a1bzBzO5fqQkZq
- **Remember: don't share your access keys**

# What's the AWS CLI?

- A tool that enables you to interact with AWS services using commands in your command-line shell
- Direct access to the public APIs of AWS services
- You can develop scripts to manage your resources
- It's open-source <https://github.com/aws/aws-cli>
- Alternative to using AWS Management Console

```
→ ~ aws s3 cp myfile.txt s3://ccp-mybucket/myfile.txt
upload: ./myfile.txt to s3://ccp-mybucket/myfile.txt
→ ~ aws s3 ls s3://ccp-mybucket
2021-05-14 03:22:52          0 myfile.txt
→ ~ █
```

# What's the AWS SDK?



- AWS Software Development Kit (AWS SDK)
- Language-specific APIs (set of libraries)
- Enables you to access and manage AWS services programmatically
- Embedded within your application
- Supports
  - SDKs (JavaScript, Python, PHP, .NET, Ruby, Java, Go, Node.js, C++)
  - Mobile SDKs (Android, iOS, ...)
  - IoT Device SDKs (Embedded C, Arduino, ...)
- Example: AWS CLI is built on AWS SDK for Python





# AWS Cloud9

- Cloud-based Integrated Development Environment (IDE)
- Code editor, debugger, terminal in a browser
- Work on your projects from anywhere with an Internet connection
- Prepackaged with essential tools for popular programming languages (JavaScript, Python, PHP, ...)
- Share your development environment with your team (pair programming)
- Fully integrated with AWS SAM & Lambda to easily build serverless applications

The screenshot shows the AWS Cloud9 IDE interface. On the left, there's a terminal window with a dark theme displaying a user's session. The user has cloned a repository named 'MyCodeCommitRepo' and is navigating through it. They've created several files like 'NodeJS', 'python.1', 'Ruby', and 'Untitled.1'. They've also made an S3 bucket named 'cloud9sample3' and uploaded files to it. The right side of the interface shows a code editor window for a file named 'lambda\_function.py'. The code is written in Python and includes imports for base64, b64decode, urlparse, and parse\_qs. It defines a function 'lambda\_handler' which takes event and context parameters. The function checks if a token is present in the event, deciphers it using AWS KMS, and then responds with a message. There are also comments explaining the use of git commands for committing changes.

```

AWS Cloud9 File Edit Find View Goto Run Tools Window Support Preview Run
claire:~/environment $ ls
Lambda Code MyCodeCommitRepo2 NodeJS python.1 Ruby Untitled.1
MyCodeCommitRepo MyCodeCommitRepo3 PHP README.md Untitled
claire:~/environment $ aws s3 mb s3://cloud9sample3
make_bucket: cloud9sample3
claire:~/environment $ aws s3 rb s3://cloud9sample3
remove_bucket: cloud9sample3
claire:~/environment $ git clone https://git-codecommit.us-west-2.amazonaws.com/v1/repos/MyCodeCommitRepo4
Cloning into 'MyCodeCommitRepo4'...
Username for 'https://git-codecommit.us-west-2.amazonaws.com': claire-at-563888640512
Password for 'https://claire-at-563888640512@git-codecommit.us-west-2.amazonaws.com':
warning: You appear to have cloned an empty repository.
claire:~/environment $ cd MyDemoCloud9Repo
bash: cd: MyDemoCloud9Repo: No such file or directory
claire:~/environment $ cd MyCodeCommitRepo
claire:~/environment/MyCodeCommitRepo (master) $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:  bird.txt
    new file:  insects.txt
    new file:  reptile.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   bird.txt
    modified:   reptile.txt

claire:~/environment/MyCodeCommitRepo (master) $ 
```

```

44 from base64 import b64decode
45 from urlparse import parse_qs
46
47 ENCRYPTED_EXPECTED_TOKEN = os.environ['']
48 kms = boto3.client('kms')
49 expected_token = kms.decrypt(Ciphertext='')
50
51 def respond(err, res=None):
52     return {
53         'statusCode': '400' if err else
54         'body': err.message if err else
55         'headers': {
56             'Content-Type': 'application/json'
57         },
58     }
59
60
61
62
63
64 def lambda_handler(event, context):
65     params = parse_qs(event['body'])
66     token = params['token'][0]
67     if token != expected_token:
68         logger.error("Request token (%s) is invalid", token)
69         return respond(Exception("Invalid token"))
70
71     user = params['user-name'][0]
72     command = params['command'][0]
73     channel = params['channel-name'][0]
74     command_text = params['text'][0]
75
76     return respond(None, "%s invoked %s" % (user, command))
77

```

<https://aws.amazon.com/cloud9/>

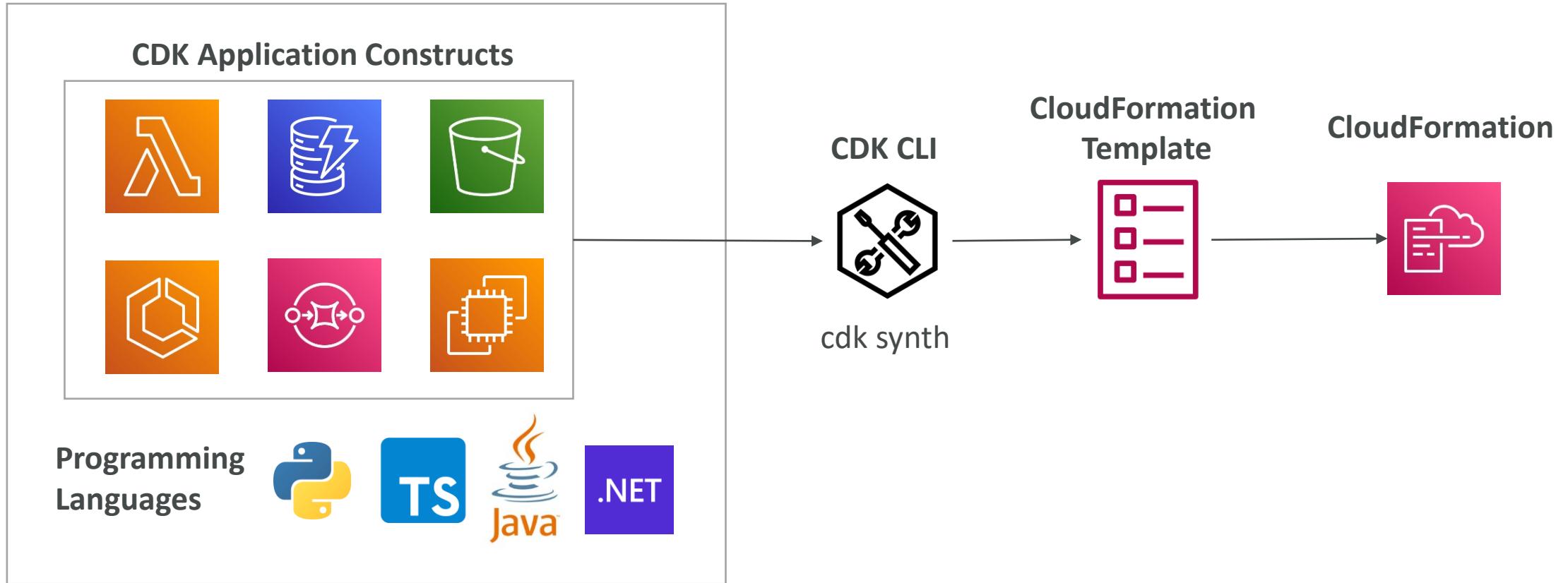
# AWS Cloud Development Kit (CDK)



- Define your cloud infrastructure using a familiar language:
  - JavaScript/TypeScript, Python, Java, and .NET
- Contains high level components called **constructs**
- The code is “compiled” into a CloudFormation template (JSON/YAML)
- **You can therefore deploy infrastructure and application runtime code together**
  - Great for Lambda functions
  - Great for Docker containers in ECS / EKS

```
export class MyEcsConstructStack extends core.Stack {  
  constructor(scope: core.App, id: string, props?: core.StackProps)  
    super(scope, id, props);  
  
  const vpc = new ec2.Vpc(this, "MyVpc", {  
    maxAzs: 3 // Default is all AZs in region  
  });  
  
  const cluster = new ecs.Cluster(this, "MyCluster", {  
    vpc: vpc  
  });  
  
  // Create a Load-balanced Fargate service and make it public  
  new ecs_patterns.ApplicationLoadBalancedFargateService(this, "My  
    cluster", cluster, // Required  
    cpu: 512, // Default is 256  
    desiredCount: 6, // Default is 1  
    taskImageOptions: { image: ecs.ContainerImage.fromRegistry("an  
      memoryLimitMiB: 2048, // Default is 512  
      publicLoadBalancer: true // Default is false  
    };  
}
```

# CDK in a diagram



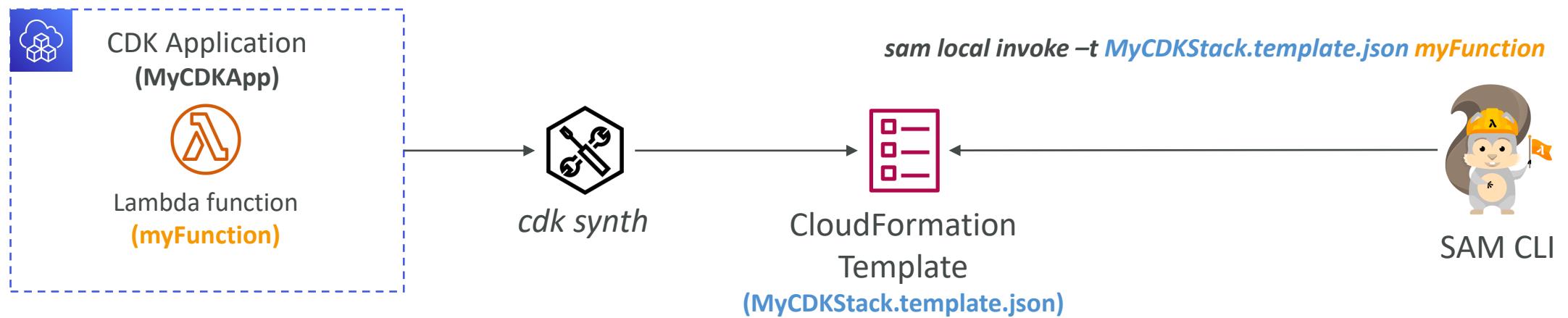
# CDK vs SAM



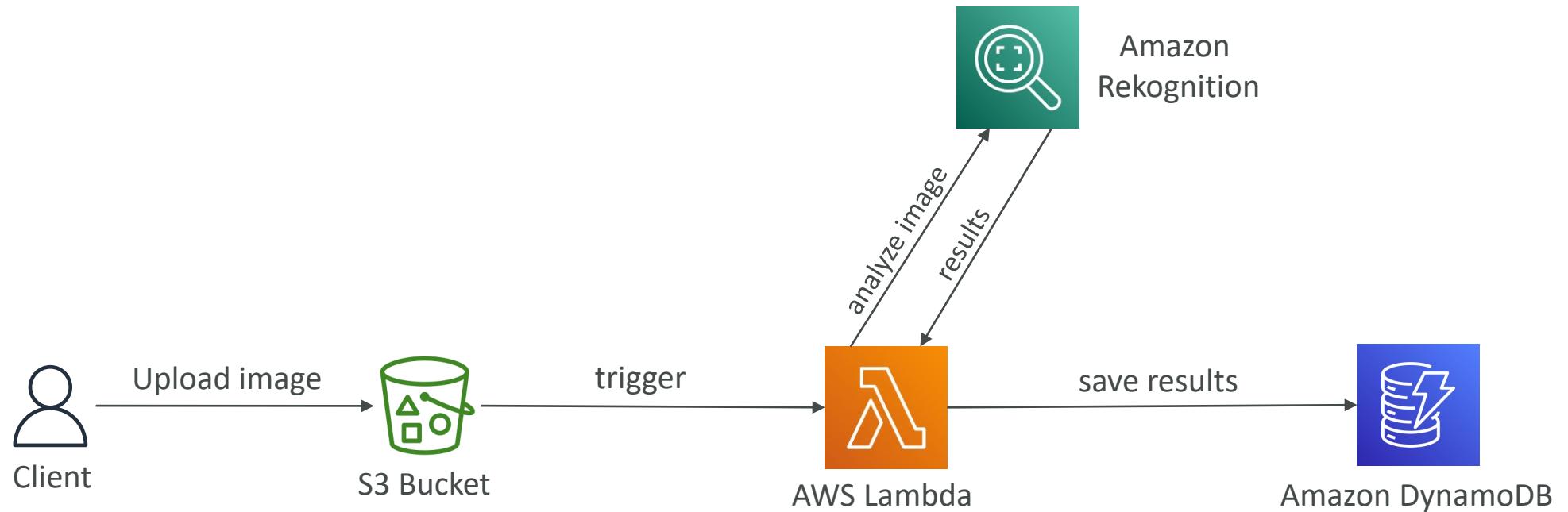
- SAM:
  - Serverless focused
  - Write your template declaratively in JSON or YAML
  - Great for quickly getting started with Lambda
  - Leverages CloudFormation
- CDK:
  - All AWS services
  - Write infra in a programming language JavaScript/TypeScript, Python, Java, and .NET
  - Leverages CloudFormation

# CDK + SAM

- You can use SAM CLI to locally test your CDK apps
- **You must first run `cdk synth`**



# CDK Hands-On

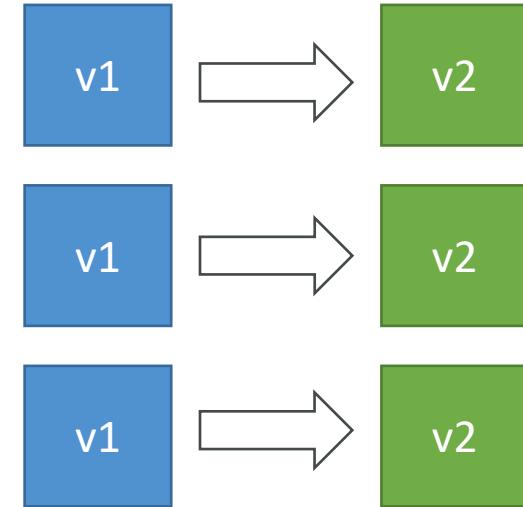


# AWS CodeDeploy

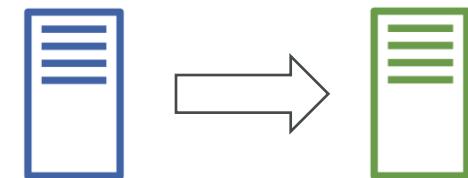


- We want to deploy our application **automatically**
- **Works with EC2 Instances**
- **Works with On-Premises Servers**
- **Hybrid service**
- Servers / Instances must be provisioned and configured ahead of time with the CodeDeploy Agent

EC2 Instances being upgraded



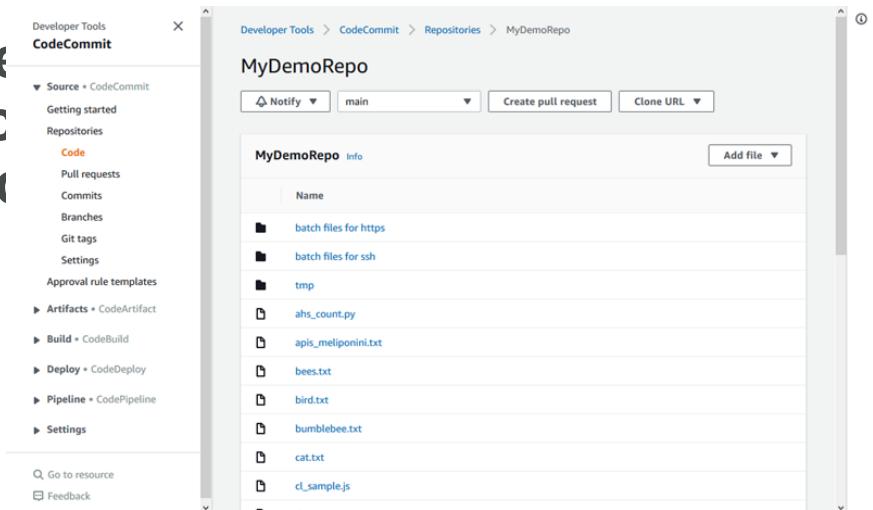
On-premises Servers being upgraded



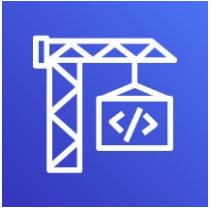
# AWS CodeCommit



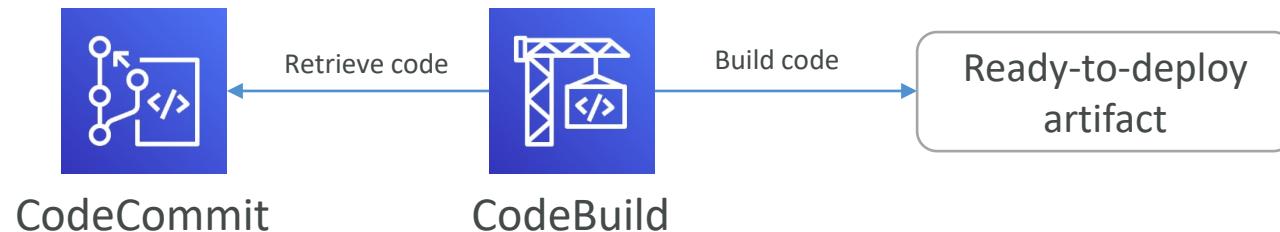
- Before pushing the application code to servers, it needs to be stored somewhere
- Developers usually store **code in a repository, using the Git technology**
- A famous public offering is GitHub, AWS' competing product is **CodeCommit**
- CodeCommit:
  - Source-control service that **hosts Git-based repositories**
  - Makes it easy to **collaborate with others on code**
  - The code changes are automatically **versioned**
- Benefits:
  - Fully managed
  - Scalable & highly available
  - Private, Secured, Integrated with AWS



# AWS CodeBuild



- Code building service in the cloud (name is obvious)
- **Compiles source code, run tests, and produces packages that are ready to be deployed (by CodeDeploy for example)**

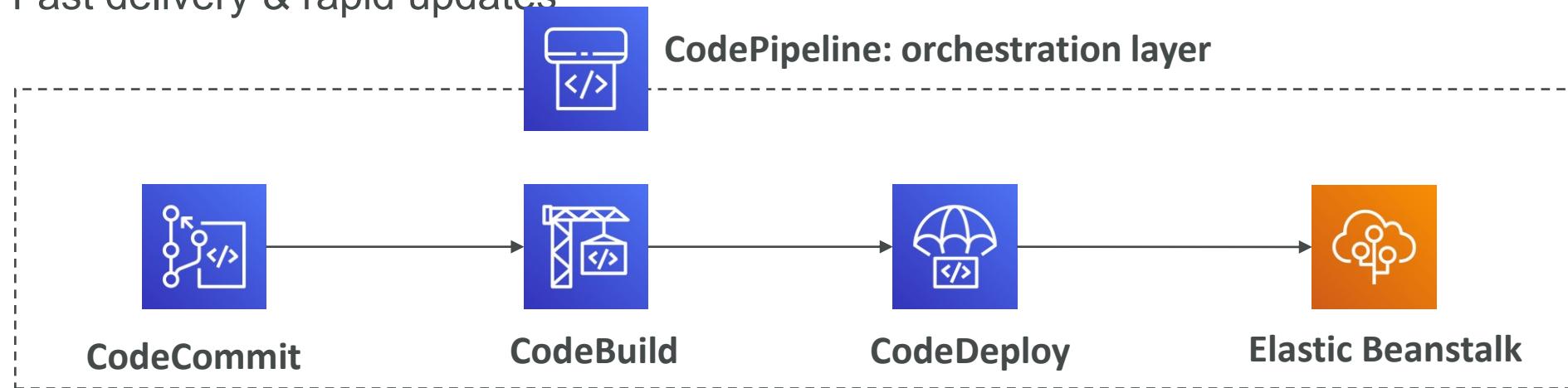


- Benefits:
  - Fully managed, serverless
  - Continuously scalable & highly available
  - Secure
  - Pay-as-you-go pricing – only pay for the build time

# AWS CodePipeline



- **Orchestrate the different steps to have the code automatically pushed to production**
  - Code => Build => Test => Provision => Deploy
  - Basis for CICD (Continuous Integration & Continuous Delivery)
- Benefits:
  - Fully managed, compatible with CodeCommit, CodeBuild, CodeDeploy, Elastic Beanstalk, CloudFormation, GitHub, 3rd-party services (GitHub...) & custom plugins...
  - Fast delivery & rapid updates



# Everything Else

Cost management, front-end and mobile access

# AWS Budgets



- Create budget and **send alarms when costs exceeds the budget**
- 4 types of budgets: Usage, Cost, Reservation, Savings Plans
- For Reserved Instances (RI)
  - Track utilization
  - Supports EC2, ElastiCache, RDS, Redshift
- Up to 5 SNS notifications per budget
- Can filter by: Service, Linked Account, Tag, Purchase Option, Instance Type, Region, Availability Zone, API Operation, etc...
- Same options as AWS Cost Explorer!
- 2 budgets are free, then \$0.02/day/budget

# Cost Explorer

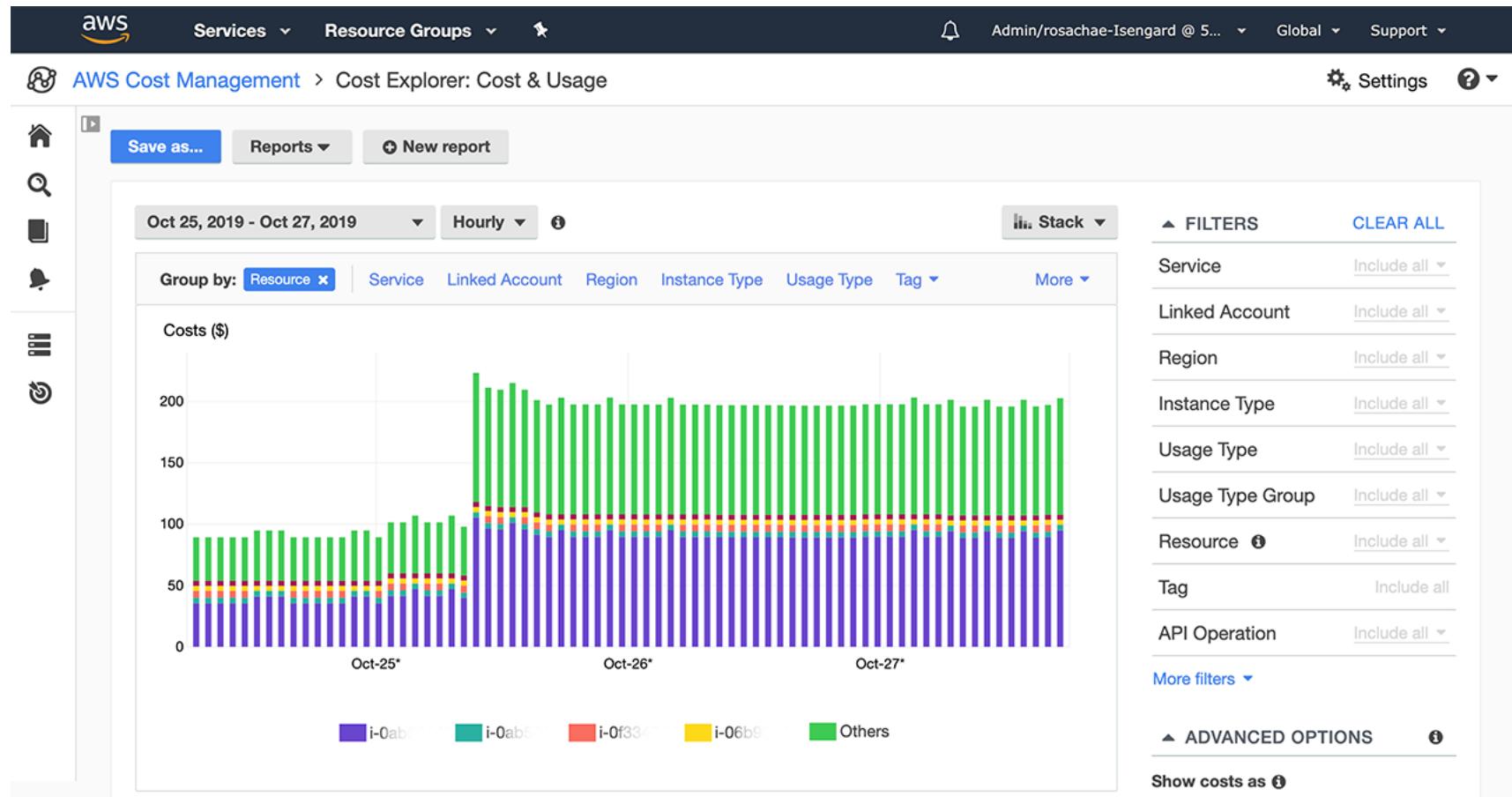


- Visualize, understand, and manage your AWS costs and usage over time
- Create custom reports that analyze cost and usage data.
- Analyze your data at a high level: total costs and usage across all accounts
- Or Monthly, hourly, resource level granularity
- Choose an optimal **Savings Plan** (to lower prices on your bill)
- **Forecast usage up to 12 months based on previous usage**

# Cost Explorer – Monthly Cost by AWS Service



# Cost Explorer– Hourly & Resource Level



# Cost Explorer – Savings Plan Alternative to Reserved Instances

**Recommendation options**

Savings Plans type <input checked="" type="radio"/> Compute <input type="radio"/> EC2 Instance	Savings Plans term <input type="radio"/> 1-year <input checked="" type="radio"/> 3-year	Payment option <input checked="" type="radio"/> All upfront <input type="radio"/> Partial upfront <input type="radio"/> No upfront	Based on the past <input type="radio"/> 7 days <input type="radio"/> 30 days <input checked="" type="radio"/> 60 days
--	---	---	--

**Recommendation:** Purchase a Compute Savings Plan at a commitment of \$2.40/hour

You could save an estimated **\$1,173** monthly by purchasing the recommended Compute Savings Plan.

Based on your past **60 days** of usage, we recommend purchasing a Savings Plan with a commitment of **\$2.40/hour** for a **3-year term**. With this commitment, we project that you could save an average of **\$1.61/hour** - representing a **40%** savings compared to On-Demand. To account for variable usage patterns, this recommendation maximizes your savings by leaving an average **\$0.04/hour** of On-Demand spend.

Before recommended purchase	After recommended purchase (based on your past 60 days of usage)
Monthly On-Demand spend <small> ⓘ</small>  \$2,955 (\$4.05/hour) <small>Based on your On-Demand spend over the past 60 days</small>	Estimated monthly spend <small> ⓘ</small>  \$1,782 (\$2.44/hour) <small>Your recommended \$2.40/hour Savings Plans commitment + an average \$0.04/hour of On-Demand spend</small>
	Estimated monthly savings <small> ⓘ</small>  \$1,173 (\$1.61/hour) <small>40% monthly savings over On-Demand \$2,955 - \$1,782 = \$1,173</small>

This recommendation examines your usage over the past 60 days (including your existing Savings Plans and EC2 Reserved Instances) and calculates what your costs would have been had you purchased the recommended Savings Plans. See applicable rates for Savings Plans [here](#). To generate this recommendation, AWS simulates your bill for different commitment amounts and recommends the commitment amount that provides the greatest estimated savings. [Learn more](#)

**Recommended Compute Savings Plans**

[Download CSV](#) [Add selected Savings Plan\(s\) to cart](#)

x	Term	Payment option	Recommended commitment	Estimated hourly savings
<input checked="" type="checkbox"/>	3-year	All upfront	\$2.40/hour	\$1.61 (40%)

\*Average hourly spend and minimum hourly spend based on your current on-demand spend for the given instance family.

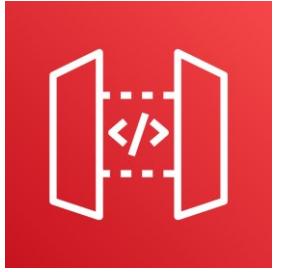
# Cost Explorer – Forecast Usage



# Example: Building a Serverless API



# AWS API Gateway



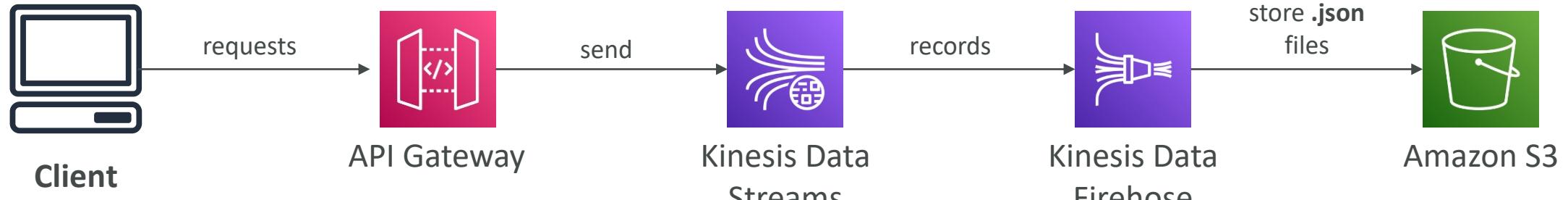
- AWS Lambda + API Gateway: No infrastructure to manage
- Support for the WebSocket Protocol
- Handle API versioning (v1, v2...)
- Handle different environments (dev, test, prod...)
- Handle security (Authentication and Authorization)
- Create API keys, handle request throttling
- Swagger / Open API import to quickly define APIs
- Transform and validate requests and responses
- Generate SDK and API specifications
- Cache API responses

# API Gateway – Integrations High Level

- **Lambda Function**
  - Invoke Lambda function
  - Easy way to expose REST API backed by AWS Lambda
- **HTTP**
  - Expose HTTP endpoints in the backend
  - Example: internal HTTP API on premise, Application Load Balancer...
  - Why? Add rate limiting, caching, user authentications, API keys, etc...
- **AWS Service**
  - Expose any AWS API through the API Gateway
  - Example: start an AWS Step Function workflow, post a message to SQS
  - Why? Add authentication, deploy publicly, rate control...

# API Gateway – AWS Service Integration

## Kinesis Data Streams example



# API Gateway - Endpoint Types

- **Edge-Optimized (default):** For global clients
  - Requests are routed through the CloudFront Edge locations (improves latency)
  - The API Gateway still lives in only one region
- **Regional:**
  - For clients within the same region
  - Could manually combine with CloudFront (more control over the caching strategies and the distribution)
- **Private:**
  - Can only be accessed from your VPC using an interface VPC endpoint (ENI)
  - Use a resource policy to define access

# API Gateway – Security

- **User Authentication through**
  - IAM Roles (useful for internal applications)
  - Cognito (identity for external users – example mobile users)
  - Custom Authorizer (your own logic)
- **Custom Domain Name HTTPS security through integration with AWS Certificate Manager (ACM)**
  - If using Edge-Optimized endpoint, then the certificate must be in **us-east-1**
  - If using Regional endpoint, the certificate must be in the API Gateway region
  - Must setup CNAME or A-alias record in Route 53

# Exam Tips and Wrapping Up

Congratulations!

# 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 85 questions
- That's only 2 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 Certified Data Engineer exam guide
- AWS's free practice exam
- AWS Skill Builder
- Complete practice exams (when available)
- This shouldn't be your first certification exam

# AWS Certification Paths – Architecture

## Architecture

### Solutions Architect

Design, develop, and manage cloud infrastructure and assets, work with DevOps to migrate applications to the cloud



Dive Deep

## Architecture

### Application Architect

Design significant aspects of application architecture including user interface, middleware, and infrastructure, and ensure enterprise-wide scalable, reliable, and manageable systems



Dive Deep

[https://d1.awsstatic.com/training-and-certification/docs/AWS\\_certification\\_paths.pdf](https://d1.awsstatic.com/training-and-certification/docs/AWS_certification_paths.pdf)

# AWS Certification Paths – Operations

## Operations

### Systems Administrator

Install, upgrade, and maintain computer components and software, and integrate automation processes



## Operations

### Cloud Engineer

Implement and operate an organization's networked computing infrastructure and Implement security systems to maintain data safety



# AWS Certification Paths – DevOps

## DevOps

### Test Engineer

Embed testing and quality best practices for software development from design to release, throughout the product life cycle



## DevOps

### Cloud DevOps Engineer

Design, deployment, and operations of large-scale global hybrid cloud computing environment, advocating for end-to-end automated CI/CD DevOps pipelines



## DevOps

### DevSecOps Engineer

Accelerate enterprise cloud adoption while enabling rapid and stable delivery of capabilities using CI/CD principles, methodologies, and technologies



# AWS Certification Paths – Security

## Security

### Cloud Security Engineer

Design computer security architecture and develop detailed cyber security designs.  
Develop, execute, and track performance of security measures to protect information



## Security

### Cloud Security Architect

Design and implement enterprise cloud solutions applying governance to identify, communicate, and minimize business and technical risks



# AWS Certification Paths – Data Analytics & Development

## Data Analytics

### Cloud Data Engineer

Automate collection and processing of structured/semi-structured data and monitor data pipeline performance



Dive Deep

## Development

### Software Development Engineer

Develop, construct, and maintain software across platforms and devices

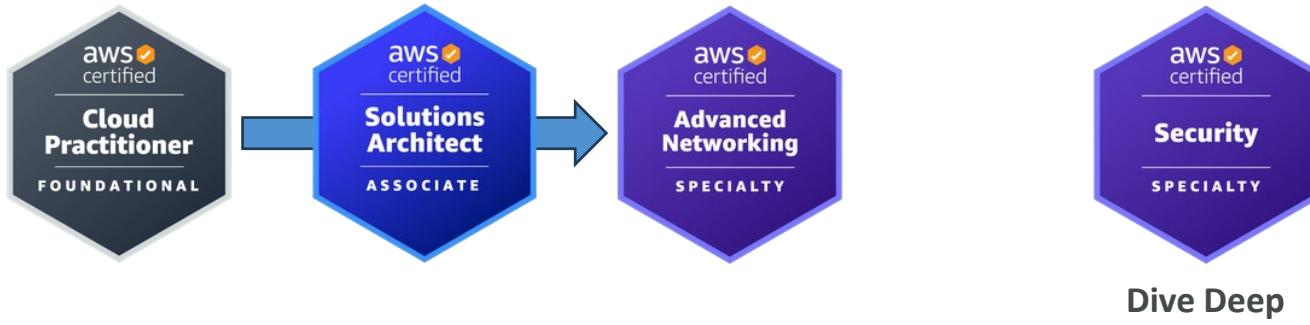


# AWS Certification Paths – Networking & AI/ML

## Networking

### Network Engineer

Design and implement computer and information networks, such as local area networks (LAN), wide area networks (WAN), intranets, extranets, etc.



## AI/ML

### Machine Learning Engineer

Research, build, and design artificial intelligence (AI) systems to automate predictive models, and design machine learning systems, models, and schemes

