



# Section 1: Introduction



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# INTRODUCTION

## Section 1:

About the exam & course setup



01

02

03

04

05

06



01

02

03

04

05

06

# About the **Data Engineer** **ASSOCIATE** Certification



01

02

03

04

05

06

01

02

03

04

05

06





## Why getting certified?

- ✓ Impactful way to advance career
- ✓ Positioning as an expert
- ✓ Future proof + great job opportunities.

## What is covered?

- ✓ AWS Certified Data Engineer Associate
- ✓ <https://aws.amazon.com/certification/certified-data-engineer-associate>

## Demos

- ✓ Not needed for the exam.
- ✓ Help with memorizing.
- ✓ Give you practical foundation.

## Goal

- ✓ Clear exam with ease.
- ✓ Knowledge for working with AWS

## Passing Score

- ✓ 720 / 1000
- ✓ Goal: Achieve a score of 850+

01

02

03

04

05

06



01

02

03

04

05

06



# Master the Exam

## Free Trial Account

Not needed for the exam.  
Help with memorizing  
Give you a practical knowledge.

## Exam Overview

<https://aws.amazon.com/certification/certified-data-engineer-associate>

## Exam Duration

Time: 130min

## Exam Questions

65 questions       Multiple Select, Multiple Choice  
*Scenario-based questions – find the best solution*

A data engineer needs to create an ETL process that automatically extracts data from an Amazon S3 bucket, transforms it, and loads it into Amazon Redshift. Which AWS service is the EASIEST to achieve this?

- AWS Lambda
- AWS Glue
- Amazon Step Functions
- Amazon EMR

aws certified

Data  
Engineer  
ASSOCIATE

01

02

03

04

05

06

01

02

03

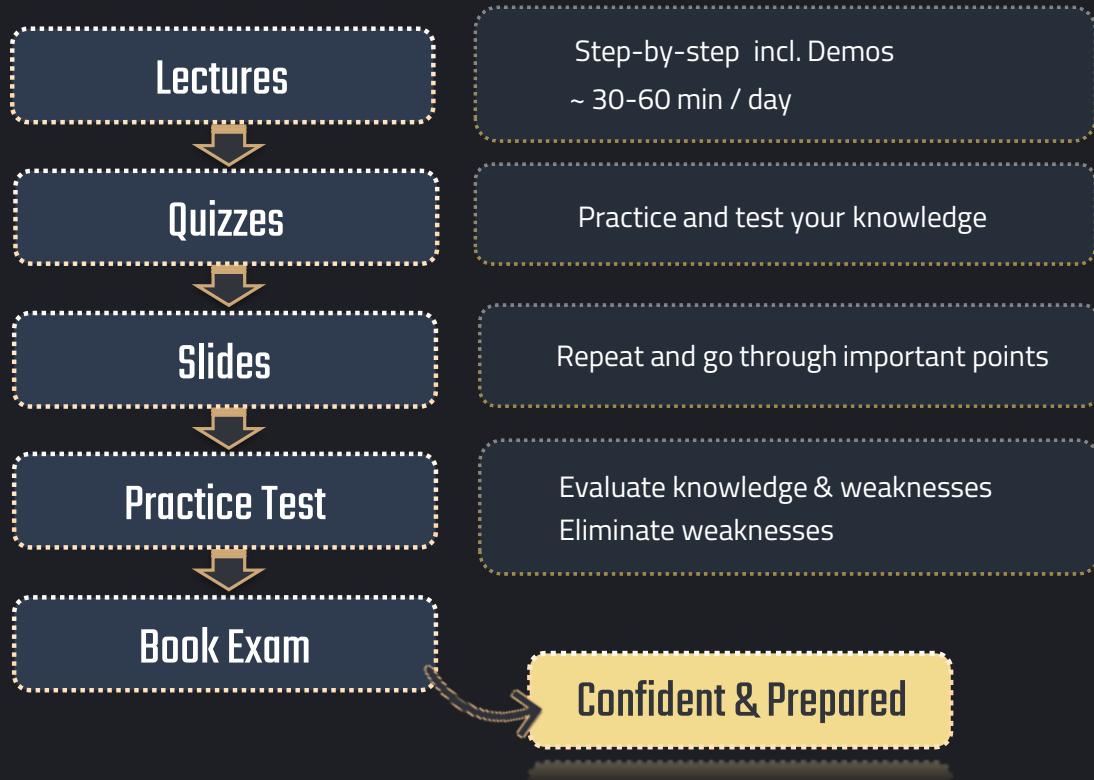
04

05

06



# Recipe to clear the exam



# Final Tips

Resources



Q&A Section



Reviews



Connect & Congratulate





# Section 2:

# Data Ingestion



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# AWS S3 – Storage



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# AWS S3 - Storage

Main Storage Solution

Data Management

- One of the most important building blocks in AWS
- S3 = "*Simple Storage Service*"
- Cost-effective and simple object storage
- *Buckets* (containers for storage) and *objects* (files)

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with 'Amazon S3' and 'firstbuckettest23'. Below it is a toolbar with tabs: Objects (which is selected), Properties, Permissions, Metrics, Management, and Access Points. The main area is titled 'Objects (4) info' and contains a table with four rows:

Name	Type	Last modified	Size	Storage class
2017	Folder			
open.pdf	pdf	December 15, 2023, 10:40:07 (UTC-01:00)	3.8 MB	Standard (Amazon Flexible Retrieval Priority Class)
supermarket_mall	txt	December 5, 2023, 12:18:45 (UTC-01:00)	128.4 KB	Standard
supermmp	Folder			

Below the table are several buttons: Copy, Copy URL, Move, Download, Open, Delete, Actions, Create folder, and Upload. There's also a search bar labeled 'Find objects by prefix:' and a link to 'View website'.

simple web services interface





# AWS S3 - Storage

## Data Management



- *Buckets* (containers for storage) and *objects* (files)

Name	Type	Last Modified	Size	Storage Class
cv1	Folder			
cv1.pdf	pdf	December 10, 2023, 10:40:07 (UTC+0:00)	3.0 MB	Master Throtle Review Primary
Supervisior_cv1.pdf	pdf	December 5, 2023, 12:19:45 (UTC+0:00)	1.04 KB	Standard
verifying	Folder			

- Each bucket is created in a specific *region*

Name	AWS Region	Actions	Creation Date
aws-s3-test-bucket-us-east-1 273574629982	US East (Ohio) us-east-1	Bucket and objects not public	December 22, 2023, 05:22:45 (UTC+0:00)
aws-s3-test-bucket-us-west-1 373574629982	US West (Oregon) us-west-1	Bucket and objects not public	December 22, 2023, 06:12:51 (UTC+0:00)
aws-s3-test-bucket-ca-central-1 273574629982-21211186	Canada (Central) ca-central-1	Bucket and objects not public	December 24, 2023, 06:02:32 (UTC+0:00)





# AWS S3 - Storage



Name	AWS Region	Status	Creation date
aws-s3-test-1	US West (Oregon)	Bucket and objects not public	December 22, 2013, 00:22:16 (UTC+07:00)
aws-s3-test-1	US East (Virginia)	Bucket and objects not public	December 22, 2013, 00:22:11 (UTC+07:00)
aws-s3-test-1	Canada (Central)	Bucket and objects not public	December 22, 2013, 00:22:12 (UTC+07:00)

## Rules

- Each bucket is created in a specific *region*
- Buckets must have a *globally unique name* (across all regions, across all accounts)
  - Between 3 (min) and 63 (max) characters
  - Only lowercase letters, numbers, dots (.), and hyphens (-)
  - Must begin and end with a letter or number.
  - Not formatted as an IP address (for example, 192.168.5.4)





# AWS S3 - Storage



The screenshot shows the AWS S3 console interface. At the top, there's a search bar labeled "Search buckets by name:" and several filter buttons: "AWS Region", "Name", "Creation date", and "Last modified". Below these are three buttons: "Create bucket", "Edit filters", and "Delete". The main area displays a table of buckets:

Name	AWS Region	Last modified	Bucket and object count
www-ahmed-sayed-aws-s3-test-1-270534629880	Canada (Central) ca-central-1	December 22, 2023, 00:22:46 (UTC+07:00)	Bucket and objects not public
www-ahmed-sayed-aws-s3-test-1-373574629880	US East (N. Virginia) us-east-1	December 22, 2023, 00:12:51 (UTC+07:00)	Bucket and objects not public
www-ahmed-sayed-aws-s3-test-1-270534629882-71218100	Canada (Central) ca-central-1	December 22, 2023, 00:22:42 (UTC+07:00)	Bucket and objects not public

- Each bucket is created in a specific *region*
- Each object is identified by a unique, user-assigned *key*

Key



- Upload: example.txt



documents/example.txt





# AWS S3 - Storage

## Data Management

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with tabs for 'Objects', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. Below the navigation bar, the main area displays a table of objects in the bucket 'firstbuckettest23'. The table has columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. There are five entries: a folder named 'pdf', a PDF file named 'open.pdf' (modified on December 15, 2023, at 10:40:07 UTC, size 5.8 MB, storage class Standard), a CSV file named 'supermarket.csv' (modified on December 5, 2023, at 12:17:45 UTC, size 128.4 KB, storage class Standard), and two other folders named 'supermarket' and 'supermarket2'.

Name	Type	Last modified	Size	Storage class
pdf	Folder			
open.pdf	pdf	December 15, 2023, 10:40:07 UTC-01:00	5.8 MB	Standard
supermarket.csv	csv	December 5, 2023, 12:17:45 UTC-01:00	128.4 KB	Standard
supermarket	Folder			
supermarket2	Folder			

simple web services interface

## Use Cases

- Backup & recovery
- Websites, Applications
- Data archiving
- Data lakes
- ... etc.



# AWS S3 – Storage Classes

Storage Class	Use Case	Durability	Availability
<i>S3 Standard</i>	Frequently accessed data	99.999999999% "11 nines"	99.99%
<i>S3 Intelligent-Tiering</i>	Data with unknown or changing access patterns	99.999999999%	99.90%
<i>S3 Standard-IA</i>	Less frequently accessed data, but requires rapid access when needed	99.999999999%	99.90%
<i>S3 One Zone-IA</i>	Same as Standard-IA, but stored in a single AZ for cost savings	99.999999999%	99.50%
<i>S3 Glacier</i>	Long-term archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after retrieval)
<i>S3 Glacier Deep Archive</i>	Longest-term archiving where retrieval time of 12 hours is acceptable	99.999999999%	99.99% (after retrieval)

## Durability

Likelihood of losing an object in a year (1 in 100 billion)

## Availability

Percentage of time that the service is operational

## Lifecycle Rules

Define change of storage classes over time

## Versioning

Allows you to retrieve previous versions of an object



# Data Ingestion Methods



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Streaming Ingestion

- Enables real-time ingestion
- Ideal for time-sensitive data
- More expensive and intricate
- Implemented using services like **Amazon Kinesis** for streaming data.

VS

# Batch Ingestion

- Ingests data periodically in batches.
- Typically large volumes
- Cost-effective and efficient
- Tools like **AWS Glue** commonly used





# AWS Glue



003-1040559

1250 003-77156.8

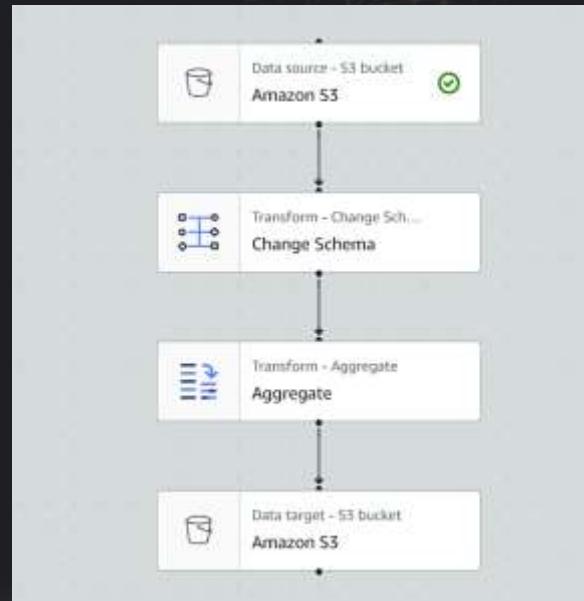
1760 0009-14563.7

73273



# AWS Glue

- **Fully-managed ETL service**
- Designed to make it easy to **load and transform data**
- **Visual interface:**  
Easily create ETL jobs without code
- **Various integrations:**  
Amazon S3, Amazon Redshift, and Amazon RDS



# AWS Glue

- **Fully-managed ETL service**
- Designed to make it easy to **load and transform data**
- **Visual interface:**  
Easily create ETL jobs without code
- **Various integrations:**  
Amazon S3, Amazon Redshift, Amazon RDS, Kinesis Data Streams, DocumentDB etc.





# AWS Glue

- **Script auto-generated** behind the scenes
- **Uses Spark** behind the scenes  
(without need to manage anything)
- **Serverless:**  
AWS Glue takes care of the underlying infrastructure
- **Pay-as-you-go:**  
You only pay for what you use



Visual Script Job details Flows Data quality Schedules Version Control

Script (Locked) info

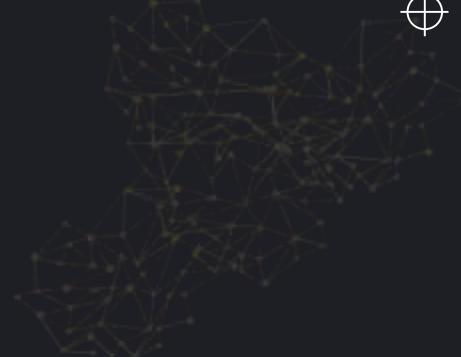
```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7 from awsglue.dynamicframe import DynamicFrame
8 from sparktk import functions as F
9
10 def sparkAggregate(glueContext, parentFrame, groups, agg, transformation_ctx) -> DynamicFrame:
11     aggFuncs = []
12     for column, func in agg:
13         aggFuncs.append(glueContext.SqlFuncs.Func(column))
14     result = parentFrame.groupby(groups).agg(*aggFuncs) if len(groups) > 0 else parentFrame
15     return DynamicFrame.fromDF(result, glueContext, transformation_ctx)
16
17 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
18 hc = SparkContext()
19 glueContext = GlueContext(hc)
20 spark = glueContext.sparkSession
21 job = Job(glueContext)
22 job.init(args['JOB_NAME'], args)
23
24 # Script generated for node Amazon S3
25 AmazonS3_node17020009400125 = glueContext.create_dynamic_frame.from_options(frame_options={"sparkChar": "\\", "sep": "\t"}, path="s3:///data")
26
27 # Script generated for node Change Schema
28 ChangeSchema_node1702000915003 = ApplyMapping.apply(frame=AmazonS3_node17020009400125, mappings=[{"tData": "String", "tType": "String"}])
29
30 # Script generated for node Aggregate
31 Aggregate_node17030005556684 = sparkAggregate(glueContext, parentFrame=ChangeSchema_node1702000915003, groups=[],
32
33 # Script generated for node Amazon S3
34 AmazonS3_node1703000716267 = glueContext.create_dynamic_frame.from_options(frame=Aggregate_node17030005556684, co
```





# AWS Glue Data Catalog

- **Centralized Data Catalog:**  
Stores table schemas and metadata  
⇒ allows querying by:  
**AWS Athena, Redshift, Quicksight, EMR**
- **Glue Crawlers:**  
scan data sources,  
infer schema,  
stored in the AWS Glue Data Catalog  
⇒ Can automatically **classify data**
- **Scheduling:**  
Run on a schedule or based on triggers  
+ incremental loads/crawling (ETL jobs & Crawlers)



```
ID,Name,Age,Address,City,State,Email
1,John Smith,25,123 Main St.,New York,NY,johnsmith@example.com
2,Jane Doe,34,456 Oak Street,Los Angeles,California,janedoe@example.com
3,Mark Johnson,40,789 Pine Court,SAN FRANCISCO,CA,markjohnson@example.com
4,Alice Ali,38,123 Elm Street,Chicago,IL,aliceali@example.com
5,Robert Yang,22,789 Cedar Ln,Miami,FL,robertyang@example.com
6,Sarah Smith,28,123 Aspen Street,New York,NY,sarahsmith@example.com
7,David Ramirez,35,456 Walnut Lane,Houston,Texas,davidramirez@example.com
8,Anna Alanson,27,777 Chestnut Avenue,Los Angeles,CA,annaalanson@example.com
9,Michael Jones,31,456 Willow Street,San Francisco,CA,michaeljones@example.com
10,Linda Lovegood,29,789 Sycamore Ave,Chicago,IL,lindalovegood@example.com
11,Sarah Smith,28,123 Aspen Street,New York,NEW YORK,sarahsmith@example.com
11,Sarah Smith,28,123 Aspen Street,New York,NEW YORK,sarahsmith@example.com
```





## Section 3:

# Querying with Athena



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



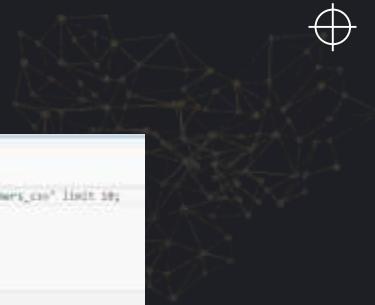


# AWS Athena



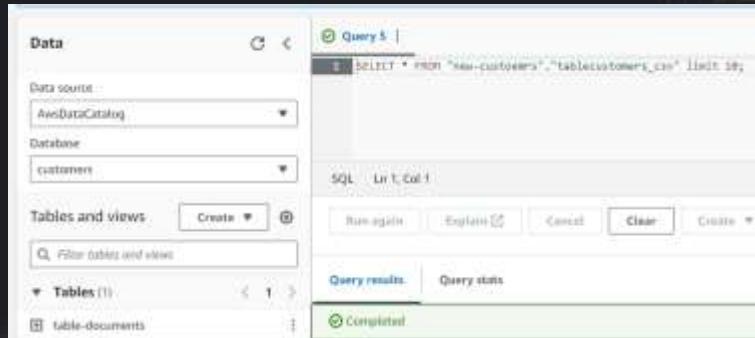
003-1040559 1250 003-77156.8 1760 0009-14563.7 73273

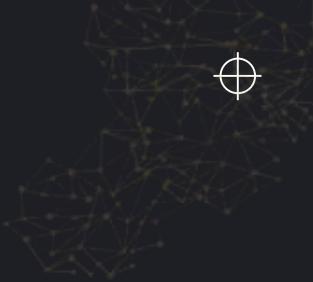




# AWS Athena

- **Interactive query service:**  
Query files in S3 using SQL
- **Serverless:**  
No infrastructure to manage  
Pay-as-you-go





# AWS Athena

- **Log Analysis:**

Analyzing log files stored in Amazon S3



- **Ad-Hoc Analysis:**

Ad-hoc queries on data lakes stored in S3

- **Data Lake Analytics:** Building a data lake on Amazon S3 and using Athena to query data

- **Real-Time Analytics:**

Integrating Athena with streaming data sources such as Amazon Kinesis





# Athena

## Federated Queries

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



## Federated Query:

- Query data sources **other than S3 buckets** using a data connector.
  - Relational and non-relational data sources
  - Object data sources
  - Custom data sources
- Federated data sources: (built-in examples)
  - Amazon CloudWatch Logs,
  - Amazon DynamoDB,
  - Amazon DocumentDB,
  - Amazon RDS,...
- Work with **multiple sources**
  - Amazon RDS – products table
  - Amazon DocumentDB – detailed customer profile data
  - Amazon DynamoDB – user interactions



# Athena Workgroups



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





## Workgroups:

Isolate queries from other queries in the same account

- Isolate queries for different...
  - Teams
  - Use cases
  - Applications

⇒ Different settings or to track and control cost
- Control...
  - Query execution settings
  - Access
  - Cost
  - Type of engine (Athena SQL vs. Apache Spark)
- Up to 1000 workgroups per region
- Each account has primary workgroup



# Athena - Cost



## Cost:

- Pay for queries run – amount of data scanned
- Cost saving possible with reserved capacity





# Athena - Performance Optimization



- **Use partitions**

- Eliminate data partitions that need to be scanned (pruning)

- **Use partition projection**

- Automate partition management
- Speed up queries for tables with large partitions

- **AWS Glue Partition Indexes:**

- Athena retrieves only relevant partitions instead of loading all
- Optimize query planning and reduce query runtime



# Athena - Query Result Reuse



- **What it does?**

Reuses previous results that match your query and max. age



The screenshot shows the AWS Athena console interface. At the top, there are tabs for multiple queries: Query 5, Query 6, Query 7, Query 8, Query 9, and Query 10. The tab for 'Query 10' is currently selected. Below the tabs, a SQL query is displayed: `SELECT * FROM "AusDataCatalog"."spectres_db"."orders_external" limit 10;`. In the bottom right corner of the main query area, there is a small circular icon with a checkmark and the text 'Reuse query results up to 60 minutes ago'. A cursor arrow points towards this icon.

- **Benefits?**

Improve query performance & cost





# Athena - Query Result Reuse



- **When to use?**

- Query where the source data doesn't change frequently
- Repeated queries
- Large datasets with complex queries





# Athena - Performance Optimization



- **Data Compression:**

Reduce file size to speed up queries

- **Format Conversion:**

transform data into an optimized structure such as Apache Parquet or Apache ORC columnar formats





# Section 4:

# AWS Glue Deep Dive



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Glue Costs



003-1040559

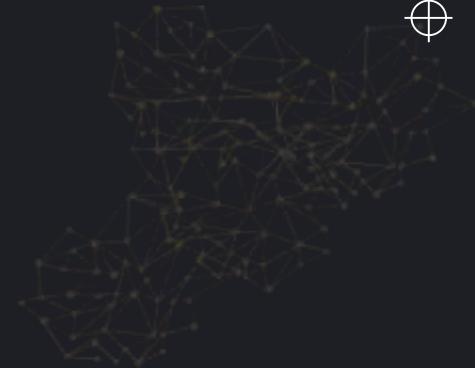
1250 003-77156.8

1760 0009-14563.7

73273



# Glue Costs

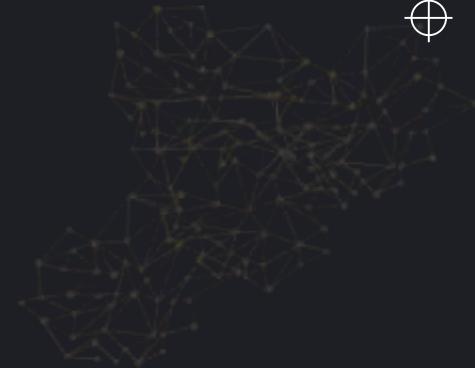


- **Crawlers:**
  - Hourly rate based on the number of DPUs used
  - Billed by seconds with with a 10-minute minumum
- **What are DPUs?**
  - DPUs = Data Processing Units
  - A single DPU provides 4 vCPU and 16 GB of memory
- **Data Catalog:**
  - Up to a million objects for free
  - \$1.00 per 100,000 objects over a million, per month





# Glue Costs

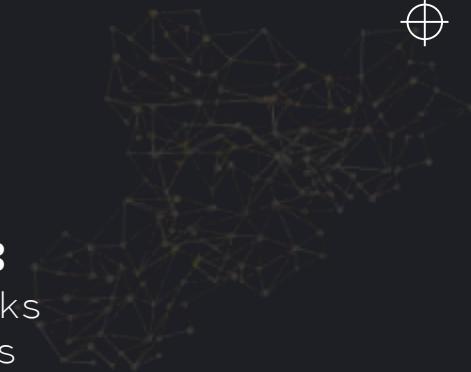


- **ETL jobs:**
  - Hourly rate based on the number of DPUs used
  - Billed by seconds with with a 10-minute minumum
  - AWS Glue versions 2.0 and later have a 1-minute minimum
- **How many DPUs are used?**
  - Apache Spark: Minimum of 2 DPUs – **Default: 10 DPUs**
  - Spark Streaming: Minimum of 2 DPUs – Default: 2 DPUs
  - Ray job (ML/AI): Minumum of 2 M-DPUs (high memory). Default:6 M-DPUs
  - Python Shell job (flexible & simple): 1 DPU or 0.0625 DPU. Default 0.0625 DPU
- **Cost of DPUs**
  - \$0.44 per DPU-Hour (may differ and depend on region)





# Glue Costs



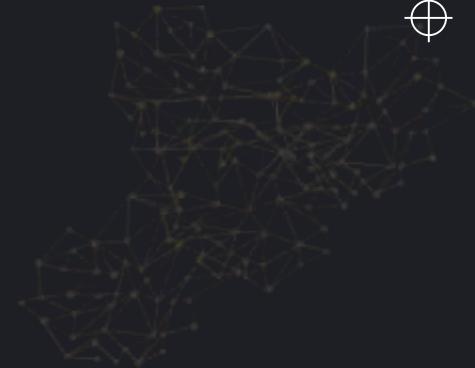
- **Glue Job Notebooks / Interactive Sessions:**

- Used to interactively develop ETL code in notebooks
- Based on time session is active and number of DPUs
- Configurable idle timeouts
- 1-minute minimum billing
- Minimum of 2 DPUs – Default: 5 DPUs





# Glue Costs



- **ETL jobs cost example:**

- Apache Spark job
- Runs for 15 minutes
- Uses 6 DPU
- 1 DPU-Hour is \$0.44

⇒ Job ran for 1/4th of an hour and used 6 DPUs

⇒  $6 \text{ DPU} * 1/4 \text{ hour} * \$0.44 = \$0.66.$

- **Interactive Session cost example:**

- Use a notebook in Glue Studio to interactively develop your ETL code.
- 5 DPU (default)
- Keep the session running for 24 minutes (2/5th of an hour)

⇒ Billed for 5 DPUs \* 2/5 hour at \$0.44 per DPU-Hour = \$0.88.





# AWS Budgets



003-1040559

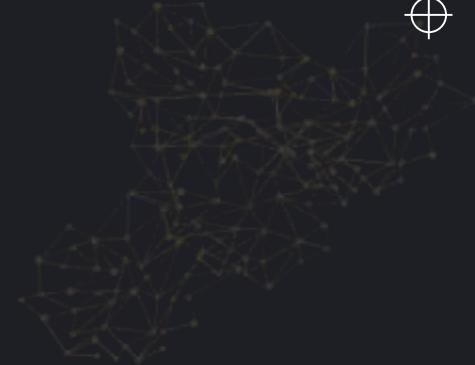
1250 003-77156.8

1760 0009-14563.7

73273



# AWS Budgets



- **Awars:**

Set budgets & receive alarms when exceeded

- **Actual & Forecasted**

Help to manage cost

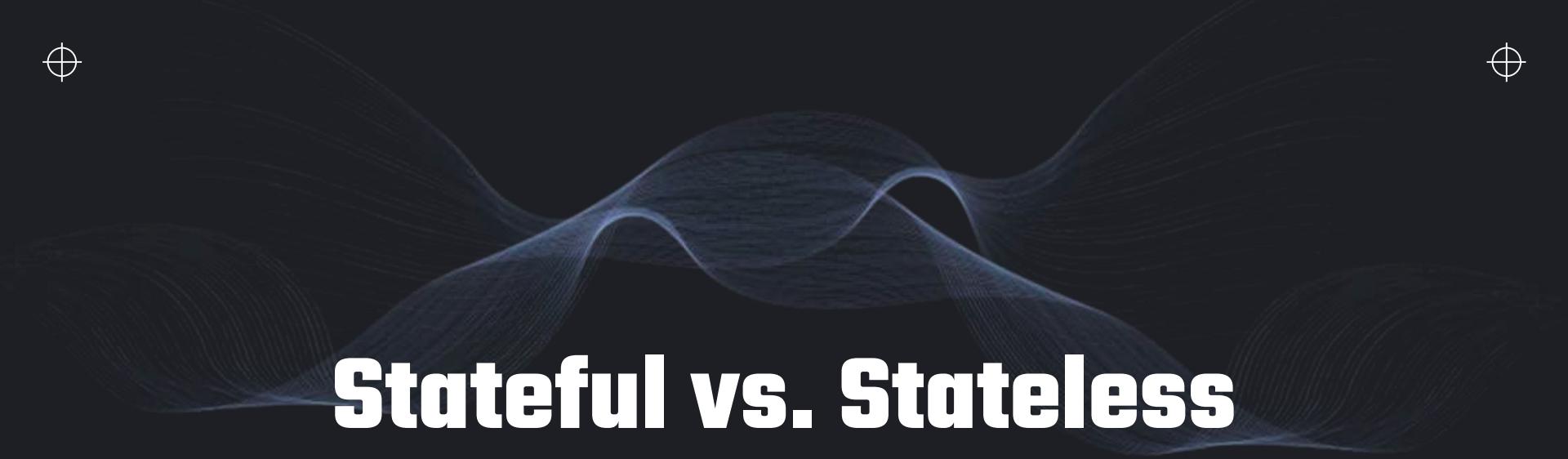
- **Budget Types:**

- Cost budget
- Usage budget
- *Saving plans budget*
- *Reservation plans budget*

- **Budgets are free**

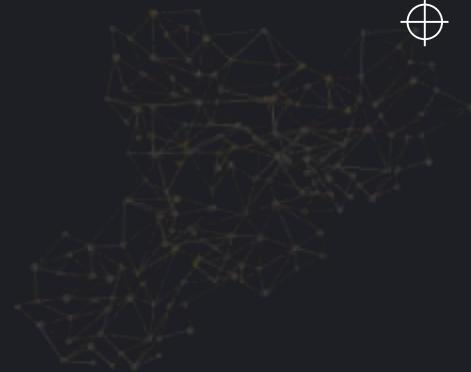
Two action-enable budgets are free then it is \$0.10/day





# Stateful vs. Stateless

# Stateful vs. Stateless

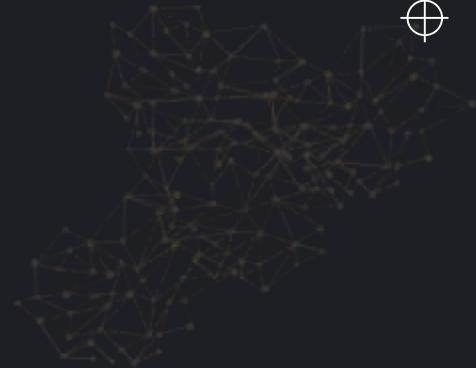


- **Stateful:**

Systems remember past interactions for influencing future ones.

- **Stateless:**

Systems process each request independently without relying on past interactions.



# Stateful vs. Stateless

## Data Ingestion Context:

- **Stateful:**

Maintain context for each data ingestion event.

- **Stateless:**

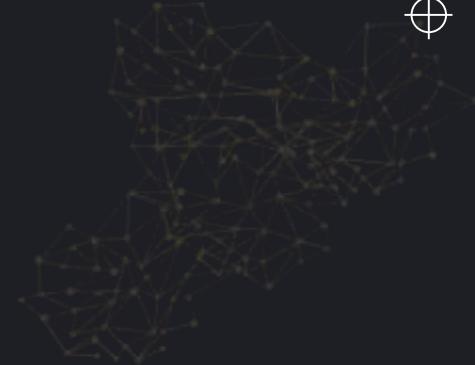
Process each data ingestion event independently.





# Stateful vs. Stateless

## Data Ingestion in AWS:



- **Amazon Kinesis:**

Supports both stateful (Data Streams) and stateless (Data Firehose) data processing.

- **AWS Data Pipeline:**

Orchestrates workflows for both stateful and stateless data ingestion.

- **AWS Glue:**

Offers stateful or stateless ETL jobs with features like job bookmarks for tracking progress.





# Glue Transformations



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# AWS Glue – Extract Transform Load



## Extract

- Amazon RDS, Aurora, DynamoDB
- Amazon Redshift
- Amazon S3, Kinesis

## Transform

- Filtering: Remove unnecessary data
- Joining: Combine data
- Aggregation: Summarize data.
- FindMatches ML: Identify records that refers same entity.
- Detect PII: Identify and manage sensitive information.

CSV <-> Parquet <-> JSON <-> XML

## Load

- Amazon RDS, Aurora, DynamoDB
- Amazon Redshift
- Amazon S3, Kinesis

S3 Target Location  
Choose an S3 location in the format s3://[bucket]/[path]/[name] with a trailing slash (/).

s3://me-first-bucket-66543/targets/

Data Catalog update options: [Info](#)  
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog tables are S3 bucket source.

Do not update the Data Catalog

Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions

Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions

Database  
Choose the database from the AWS Glue Data Catalog:

customers





# AWS Glue – Extract Transform Load



## Extract

- Amazon RDS, Aurora, DynamoDB
- Amazon Redshift
- Amazon S3, Kinesis

## Transform

- Filtering: Remove unnecessary data
- Joining: Combine data
- Aggregation: Summarize data.
- FindMatches ML: Identify records that refers same entity.
- Detect PII: Identify and manage sensitive information.

CSV <-> Parquet <-> JSON <-> XML

## Load

- Amazon RDS, Aurora, DynamoDB
- Amazon Redshift
- Amazon S3, Kinesis

S3 Target Location  
Choose an S3 location in the format s3://[bucket]/[path/] with a trailing slash (/).

s3://me-first-bucket-66543/target/

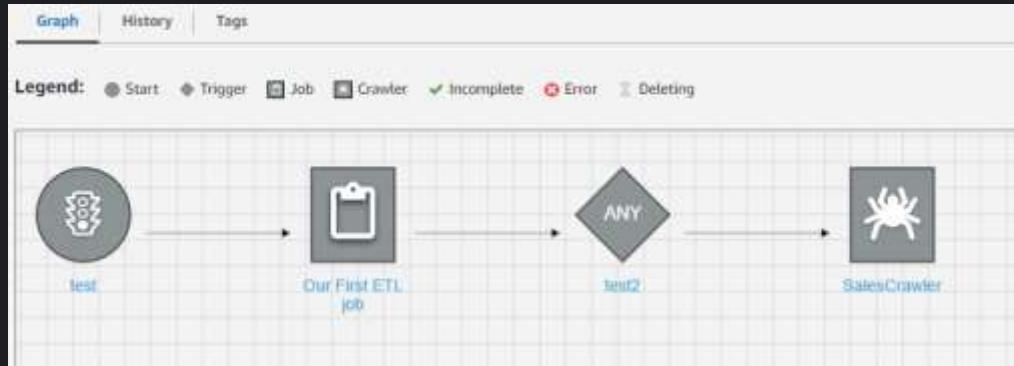
Data Catalog update options: Info  
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog tables are S3 bucket source.

Do not update the Data Catalog  
 Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions  
 Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions

Database  
Choose the database from the AWS Glue Data Catalog:  
 customers



# Glue Workflows



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

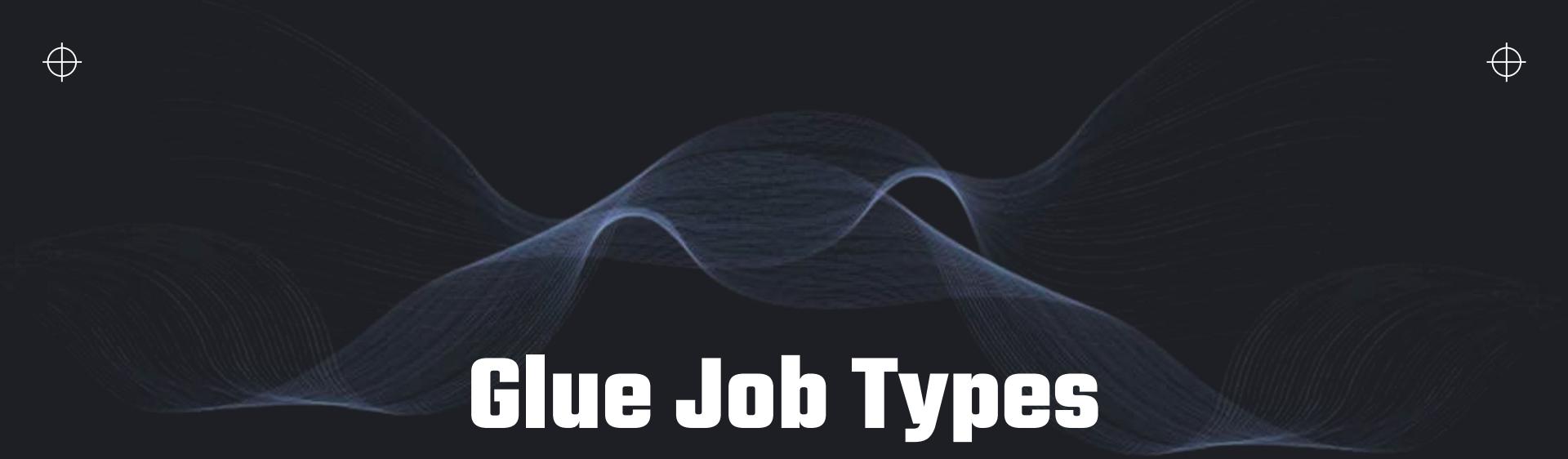


# AWS Glue – Glue Workflows



- **Orchestrate** multi-step data processing jobs, **manage executions and monitoring** of jobs/crawlers.
- Ideally used for managing AWS Glue operations but also can be leveraged other services.
- Provides visual interface.
- You can create workflows manually or with AWS Glue Blueprints.
- **Triggers** initiate jobs and crawlers.
  - Schedule Triggers: Starts the workflow at regular intervals.
  - On-Demand Triggers: Manually start the workflow from AWS Console.
  - **EventBridge Event**: Launches the workflow based on specific events captured by Amazon EventBridge.
  - **On-Demand & EventBridge**: Combination of On-Demand and EventBridge rules.
  - **Lambda Function**: With a trigger that invokes Workflow.





# Glue Job Types

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# AWS Glue

The screenshot shows the AWS Glue Studio interface under the 'Jobs' section. It displays three options for creating a new job:

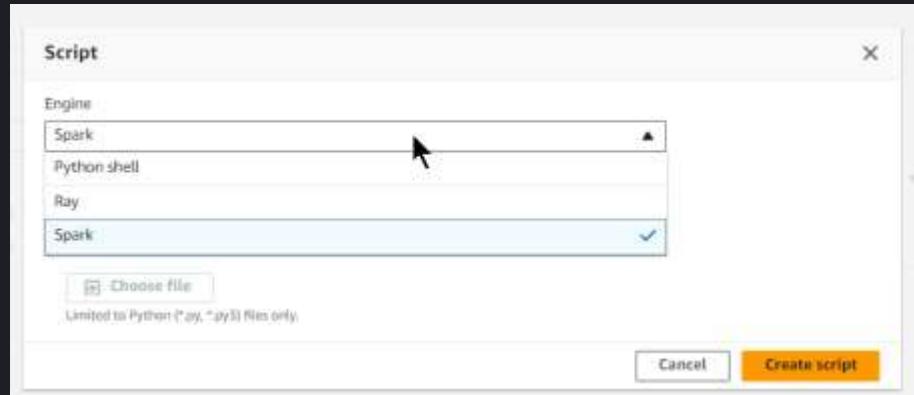
- Visual ETL**: Author in a visual interface focused on data flow.
- Notebook**: Author using an interactive code notebook.
- Script editor**: Author code with a script editor.

Type: Spark

Glue version: Info  
Glue 4.0 - Supports spark 3.3, Scala 2, Python 3

Language: Python 3

Worker type: G 1X (4vCPU and 16GB RAM)





# AWS Glue

## Job Types



### Spark ETL Jobs:

- ⇒ Large-scale data processing.
- ⇒ 2DPU to 100DPU



### Spark Streaming ETL Jobs:

- ⇒ Analyze data in real-time.
- ⇒ 2DPU to 100DPU



### Python Shell Jobs:

- ⇒ Suitable for light-weight tasks.
- ⇒ 0.0625DPU to 1DPU.



### Ray Jobs:

- ⇒ It is suitable for parallel processing tasks.





# AWS Glue

## Execution Types



### Standard Execution:

- ⇒ Designed for predictable ETL jobs.
- ⇒ Jobs start running immediately.
- ⇒ Guarantees consistent job execution times.



### Flex Execution:

- ⇒ Cost-effective option for less time-sensitive ETL jobs.
- ⇒ Jobs may start with some delay.





# Partitioning



003-1040559

1250 003-77156.8

1760 0009-14563.7

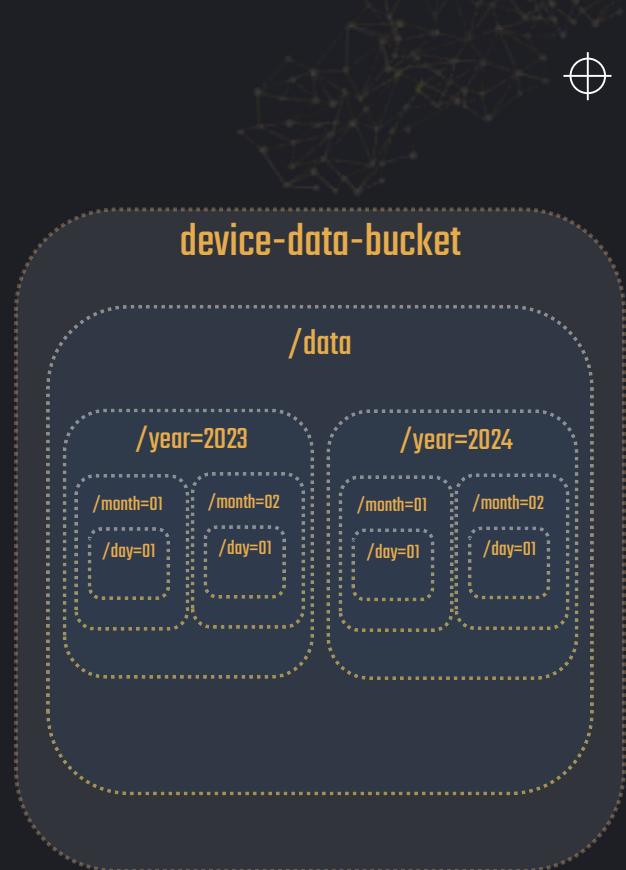
73273





# AWS Glue – Partitioning

- Enhances the performance of AWS Glue
  - Provides better query performance.
  - Reduces I/O operations.
- AWS Glue can skip over large segments within partitioned data.
- AWS Glue can process each partition independently.
- Provides cost efficiency by reducing query efforts.
- In AWS Glue, define partitioning as part of ETL job scripts. Also possible within Glue Data Catalog.





# AWS Glue DataBrew



003-1040559 1250 003-77156.8 1760 0009-14563.7 73273



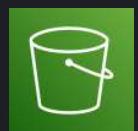


# AWS Glue DataBrew

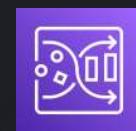
- Data preparation tool with visual interface.
- **Cleaning** and **data format** processes.
- Pre-built transformations.
- No coding required.
- Automate data preparations.



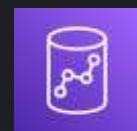
Amazon S3  
(Data Lake)



AWS Glue  
Data Brew



Amazon  
Redshift



# AWS Glue DataBrew

Sample project - 1

Dataset status: Sample: First 5 rows (500 rows)

Create job | LINES | RECIPE

Wiring: 75 columns | 500 rows

Step 1: **Input** (1 row) Step 2: **Format** (1 row) Step 3: **Filter** (1 row) Step 4: **Format** (1 row) Step 5: **Filter** (1 row) Step 6: **Format** (1 row) Step 7: **Format** (1 row) Step 8: **Format** (1 row) Step 9: **Format** (1 row) Step 10: **Format** (1 row) Step 11: **Format** (1 row) Step 12: **Format** (1 row) Step 13: **Format** (1 row) Step 14: **Format** (1 row) Step 15: **Format** (1 row) Step 16: **Format** (1 row) Step 17: **Format** (1 row) Step 18: **Format** (1 row) Step 19: **Format** (1 row) Step 20: **Format** (1 row) Step 21: **Format** (1 row) Step 22: **Format** (1 row) Step 23: **Format** (1 row) Step 24: **Format** (1 row) Step 25: **Format** (1 row)

RECIPE

Sample recipe - 1

Working version

Add step

Build your recipe

Start applying transformation steps to your data. All your data preparation steps will be tracked in the recipe.

# year	# assembly_session	# state_code	All state_codes
1946	1	2	United States of America
1947	2	2	United States of America
1948	3	2	United States of America
1949	4	2	United States of America
1950	5	2	United States of America
1951	6	2	United States of America
1952	7	2	United States of America
1953	8	2	United States of America
1954	9	2	United States of America
1955	10	2	United States of America
1956	11	2	United States of America





# AWS Glue DataBrew - Transformations

## Project

Where you configure transformation tasks

## Step

Applied transformation to your dataset

## Recipe

Set of transformation steps; can be saved and reused

## Job

Execution of a recipe on a dataset; output to locations such as S3

## Schedule

Schedule jobs to automate transformation

## Data Profiling

Understand quality and characteristics of your data

The screenshot shows the AWS Glue DataBrew interface. A success message at the top right says "Remove duplicates step applied successfully". Below it, a list of steps is shown:

- 1. Sort column object name by Ascending
- 2. Remove duplicates from credit line





# AWS Glue DataBrew - Transformations

- **NEST\_TO\_MAP:**

- convert columns into a map.

Name	Age	City	NEST_TO_MAP
Alice	30	New York	{'Name': 'Alice', 'Age': 30, 'City': 'New York'}

- **NEST\_TO\_ARRAY:**

- convert columns into an array

Name	Age	City	NEST_TO_ARRAY
Alice	30	New York	['Alice', 30, 'New York']

- **NEST\_TO\_STRUCT**

- Like NEST\_TO\_MAP but retains exact data type and order





# AWS Glue DataBrew - Transformations

- **UNNEST\_ARRAY:**

- Expands array to multiple columns

The diagram illustrates the UNNEST\_ARRAY transformation. On the left, a dark gray box contains the header "NEST\_TO\_ARRAY" and the value "[Alice, 30, 'New York']". An orange arrow points from this box to the right, where a white table is shown. The table has three columns with headers: "Name", "Age", and "City". The data row contains the values "Alice", "30", and "New York" respectively.

NEST_TO_ARRAY	Name	Age	City
[Alice, 30, 'New York']	Alice	30	New York





# AWS Glue DataBrew - Transformations

- **PIVOT**

- Pivot column and pivot values to rotate data from rows into columns

Product	Quarter	Sales
A	Q1	150
A	Q2	200
B	Q1	180
B	Q2	210



Product	Q1	Q2
A	150	200
B	180	210

- **UNPIVOT**

- Pivot column into rows (attribute + value)

Name	Age	City
Frank	40	Miami



Attribute	Value
Name	Frank
Age	30
City	Miami





# AWS Glue DataBrew - Transformations

- **TRANSPOSE**

- Switch columns and rows

Name	Age	City
Alice	30	New York
Frank	32	Miami



Attribute	Alice	Frank
Age	30	32
City	New York	Miami





# AWS Glue DataBrew - Transformations

## Join

⇒ Combine two datasets.

## Split

⇒ Split a column into multiple columns based on a delimiter.

## Filter

⇒ Apply conditions to keep only specific rows in your dataset.

## Sort

⇒ Arrange the rows in your dataset in ascending or descending order.

## Date/Time Conversions

⇒ Convert strings to date/time formats or change between different date/time formats.

## Count Distinct

⇒ Calculates the number of unique entries in that column.





## Section 5:

# Serverless Compute with Lambda





# AWS Lambda



003-1040559

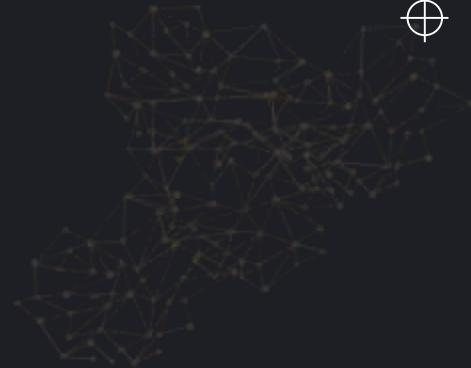
1250 003-77156.8

1760 0009-14563.7 73273





# AWS Lambda



- **What is AWS Lambda?**

- Lets you run code without managing servers  
⇒ *automatically scaling based on demand*
- Serverless compute service  
⇒ *No need to provision or manage servers*

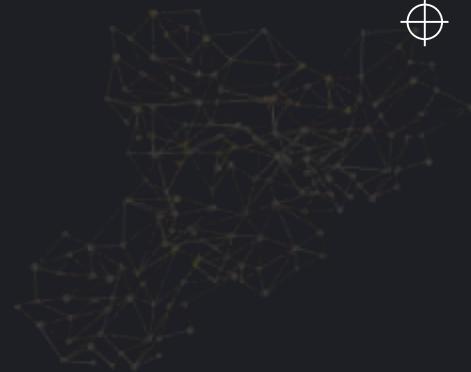
- **Various programming languages**

- Python
- Java
- Node.js
- Go
- ...





# AWS Lambda



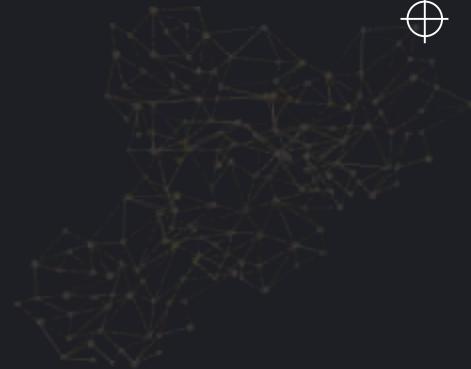
## Use cases

- **Data processing** tasks on data in Amazon S3 or DynamoDB
- **Event-driven ingestion:**
  - S3
  - *DynamoDB*
  - *Kinesis*  
⇒ Process ***real-time events***, such as *file upload*
- **Automation:**  
Automate tasks and workflows by triggering Lambda functions in response to events





# AWS Lambda



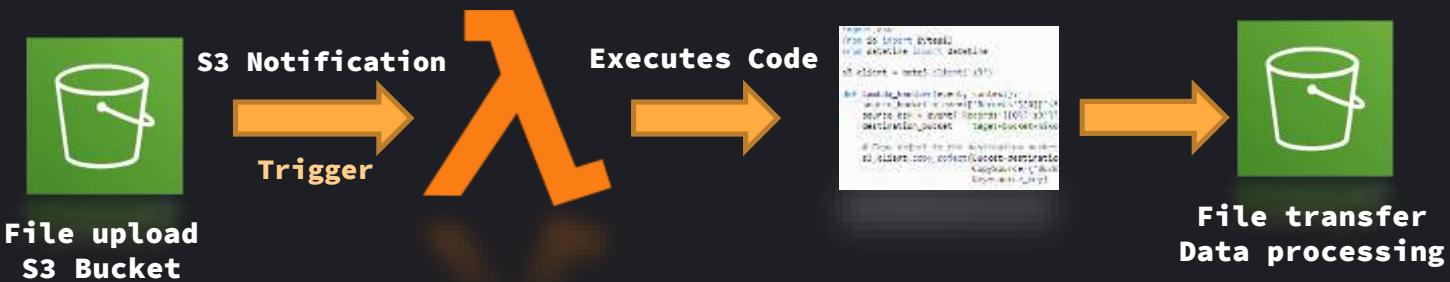
## Advantages

- **Scalable:** Scales automatically depending on workload
- **Cost efficient:** Pay only for what you use
- **Simplicity:** No need to manage infrastructure

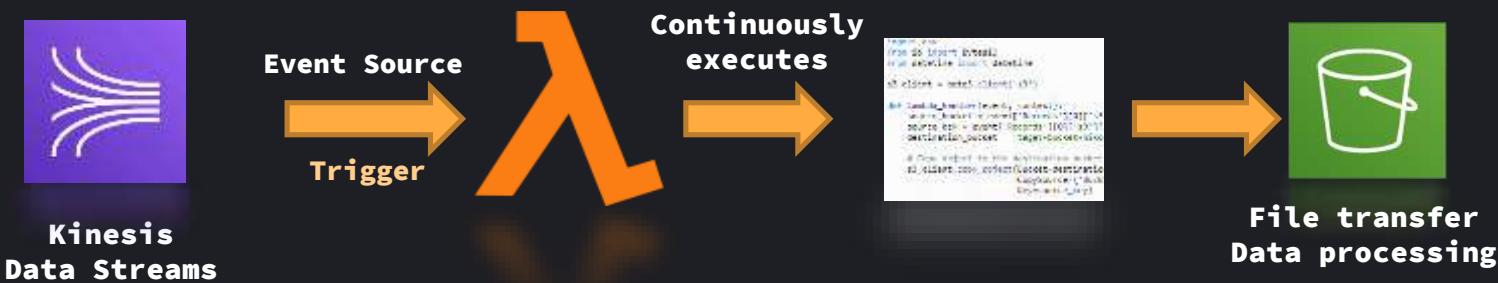
*Typically stateless: Each invocation is independent and doesn't maintain state*



# AWS Lambda – S3 Notification



# AWS Lambda – Kinesis Data Stream



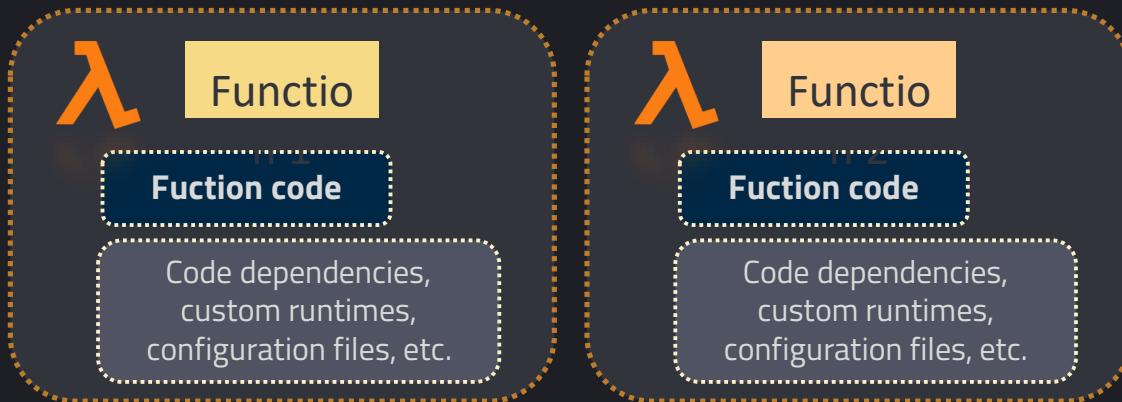
⇒ Executes in batches

⇒ Automatically scales



# Lambda Layers

- Contains supplementary code
  - library dependencies,
  - custom runtime or
  - configuration file



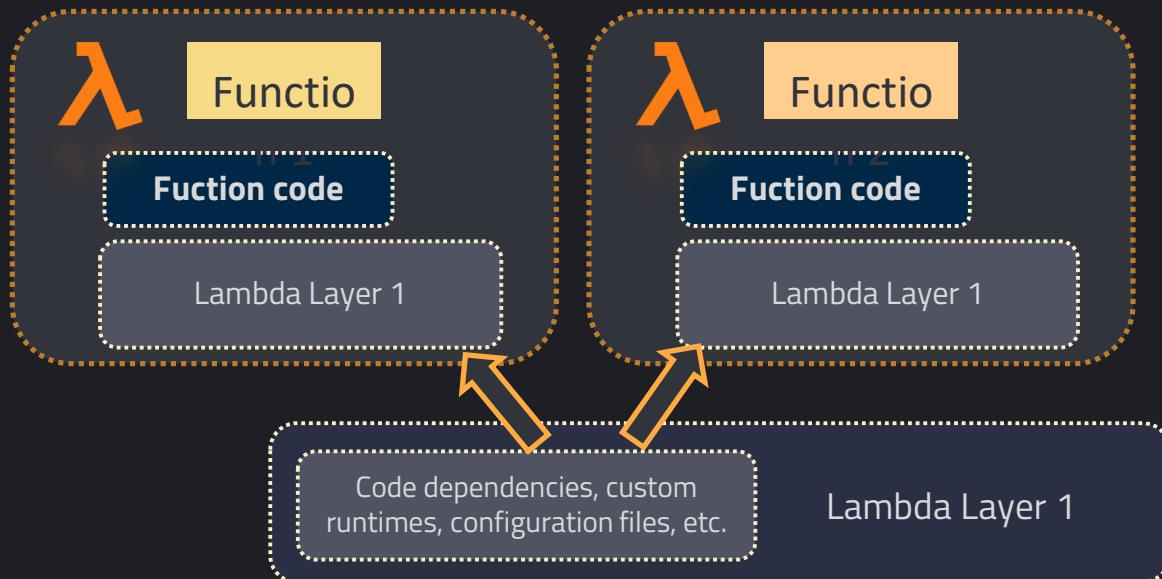
Lambda function components:  
Without layers





# Lambda Layers

1. Package Layer content (.zip)
2. Create the Layer in Lambda
3. Add the Layer to Your Functions
4. The function can access the contents of the layer during runtime



Lambda function  
components:  
With layers





# Benefits

- 1 Share Dependencies Across Multiple Functions
- 2 Separate Core Logic from Dependencies
- 3 Reduce Deployment Package Size





# Replayability



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Replayability



## Definition of Replayability:

- **Ability to reprocess or re-ingest** data that has already been handled.

## Why is it important?

- **Error Handling:** Corrects processing mistakes and recovers lost data.
- **Data Consistency:** Ensures uniform data across distributed systems.
- **Adapting to Changes:** Adjusts to schema or source data changes.
- **Testing and Development:** Facilitates feature testing and debugging without risking live data integrity.





# Replayability



## Strategies for Implementing Replayability

- **Idempotent Operations:**

Ensure repeated data processing yields consistent results.

- **Logging and Auditing:**

Keep detailed records for tracking and diagnosing issues.

- **Checkpointing:**

Use markers for efficient data process resumption.

- **Backfilling Mechanisms:**

Update historical data with new information as needed.





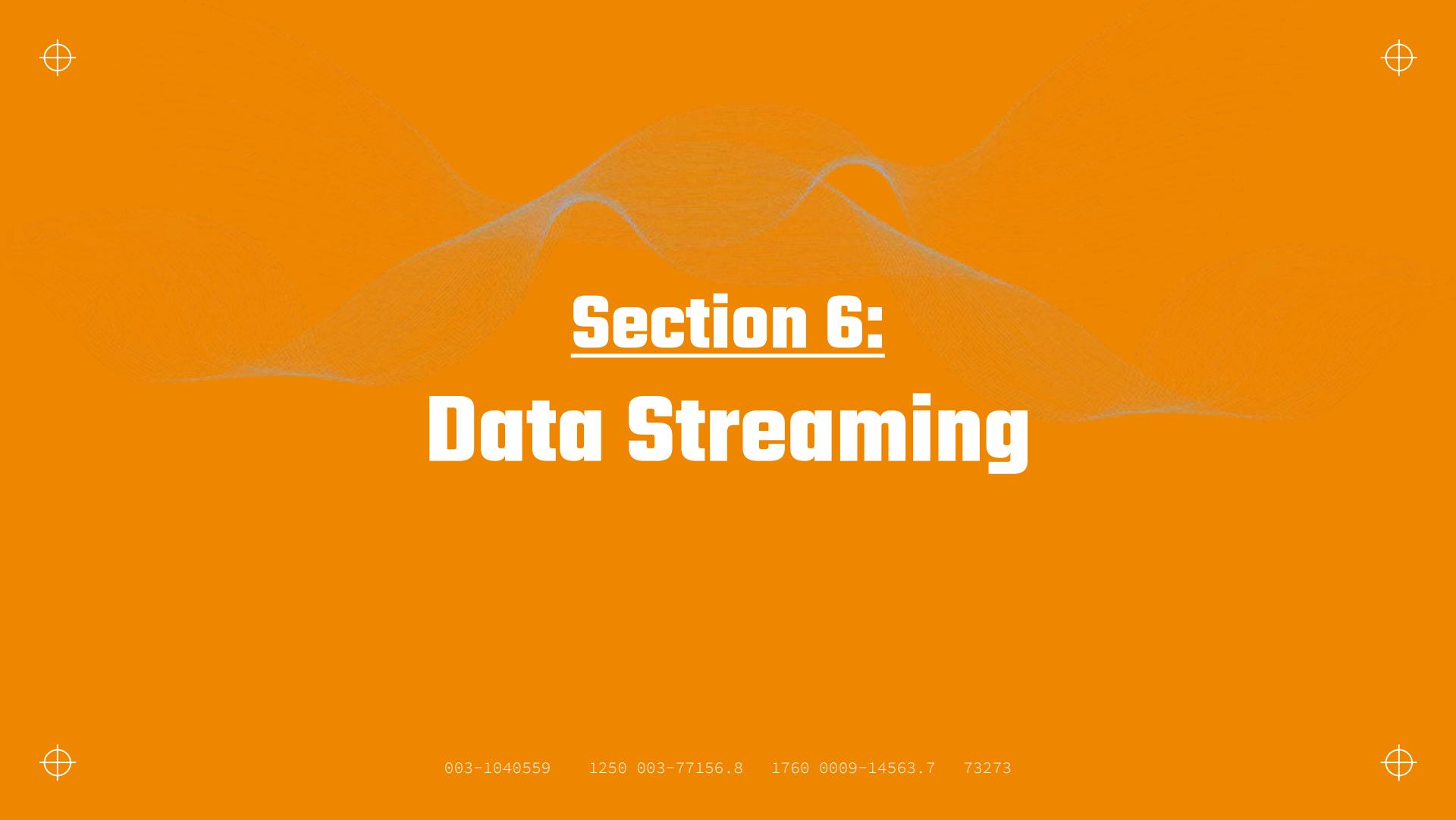
# Replayability



Replayability is an important safety net for data processing.

It ensures systems are resilient, accurate, and adaptable to changes.





# Section 6:

# Data Streaming



# Amazon Kinesis



003-1040559

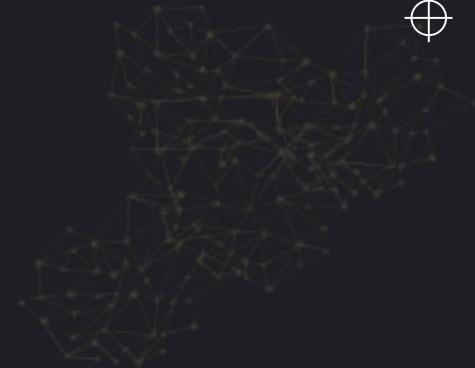
1250 003-77156.8

1760 0009-14563.7 73273





# Amazon Kinesis



## Collection of services:

Handling different aspects of data streaming

- **Kinesis Data Streams**

Ingest and process large volumes of streaming data

- **Kinesis Firehose**

Fully managed service to deliver streaming data to destinations more easily  
↳ *Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk.*

- **Managed Apache Flink**

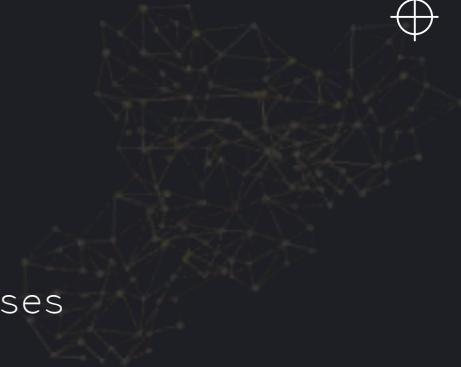
*(formerly Kinesis Data Analytics)*

Analyze streaming data in real time using standard SQL queries





# Amazon Kinesis



## Use cases

Variety of real-time data processing and analytics use cases

### 1) Real-time analytics

Analyzing streaming data to gain insights

### 2) IoT Processing

Ingesting and processing data from IoT devices or sensors

### 3) Security and fraud detection:

Detecting anomalies and responding to security in real time





# Amazon Kinesis Data Streams

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

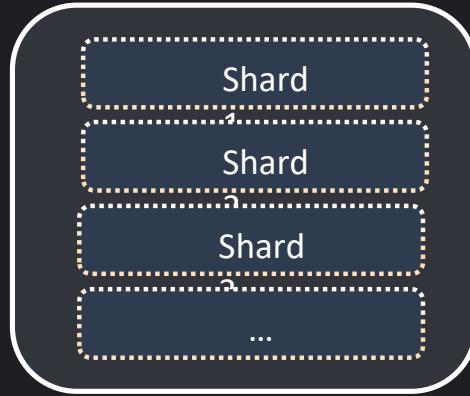
# Amazon Kinesis Data Streams



## Producer

Device or application that generates and writes data to Kinesis data stream (data source)

Units (up to 1 MB) of data (e.g JSON objects, log entries)  
Formatted as **Data Record**  
Includes a **Partition Key**  
determines the **shard** to which the record will be assigned



## Data Stream



# Amazon Kinesis Data Streams

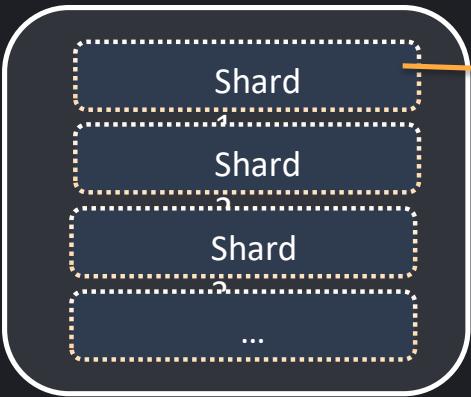


## Producer

Writes data to Kinesis data stream

Data Record  
Partition Key

determines the **shard**



## Data Stream

### Shards

Basic units of capacity

**1 shard:**    **1 MB/s in** (in-throughput)  
                  1000 records/s  
                  **2 MB/s out** (out-throughput)

⇒ Determines its overall capacity for ingesting and processing data.

⇒ Efficient parallel processing

### Durability

Configurable retention period (default 24 hours up to 365 days)

Replicated across multiple Availability Zones

⇒ Resilient to failure

Data is immutable



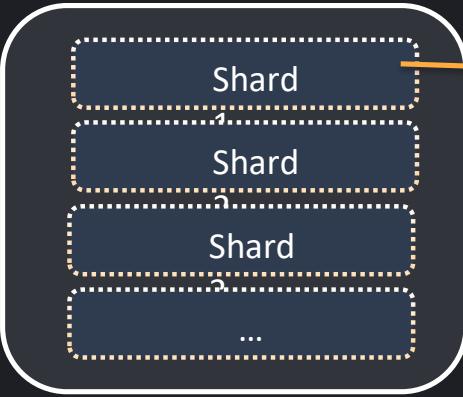
# Amazon Kinesis Data Streams



## Producer

Writes data to Kinesis data stream

Data Record  
Partition Key  
determines the **shard**



## Data Stream

### Scalability

Add and remove shards dynamically  
**Auto Scaling** to automatically scale  
⇒ Elastically scale up and down

### Capacity mode

#### Provisioned mode

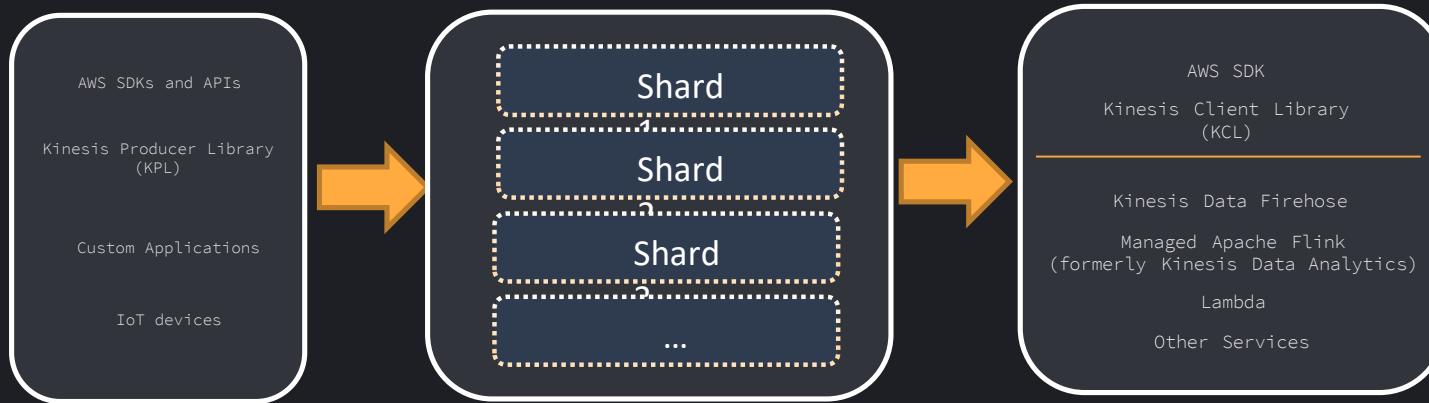
Must specify the number of shards  
Can increase and decrease the number  
Pay hourly rate

#### On-demand mode

Automatically scales shards based on throughput peaks over last 30 days  
Default: 4 MB/s or 4,000 records/s



# Amazon Kinesis Data Streams



**Producer**

**Data Stream**

**Consumer**

Writes data to Kinesis data stream

Gets records from Amazon Kinesis Data Streams  
and processes them

Kinesis Data Streams application



# Throughput and Latency



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Throughput



## Throughput in AWS Kinesis Data Streams

- **Definition:**

Volume of data (in MB or records per second) ingested into or retrieved from a Kinesis Data Stream.

- **Measurement Units:**

Units per second (e.g., Mbps, records per second)

- **Real-World:**

Actual rate of data processing, accounting for all factors.

- **Shard-Based Scaling:**

Scalable through number of shards; each shard adds fixed capacity to stream.

- **Proportional Relationship:**

Total stream throughput directly relates to shard count.

- **Optimization Goal:**

Improving capacity to process more data within timeframe for high-volume data





# Bandwidth vs. Throughput



## Bandwidth in AWS Kinesis Data Streams

- **Definition:**

The maximum data transfer rate

- **Theoretical Upper Limit:**

Potential maximum for throughput





# Latency



## Latency and Propagation Delay

- **Definition Latency:**

The time from initiating a process to the availability of the result.

- **Propagation Delay:**

Specific latency from when a record is written to when it's read by a consumer.

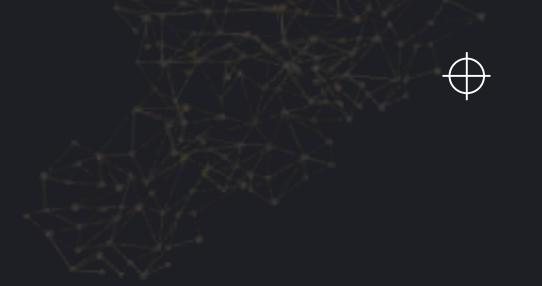
- **Influencing Factor:**

Significantly affected by the polling interval, or how often consumer applications check for new data.





# Latency



## Latency and Propagation Delay

- **Recommendation:**

Poll each shard once per second per consumer application to balance efficiency and avoid exceeding API limits.

- **Kinesis Client Library (KCL) Defaults:**

Configured to poll every second, keeping average delays below one second.

- **Reducing Delays for Immediate Data Needs:**

Increase the KCL polling frequency for quicker data access, with careful management to avoid API rate limit issues.





# Enhanced Fan-Out



003-1040559

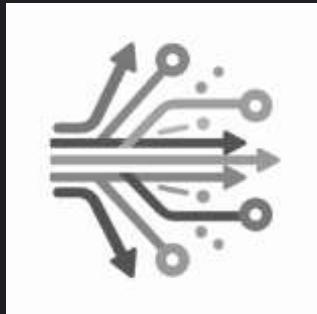
1250 003-77156.8

1760 0009-14563.7 73273





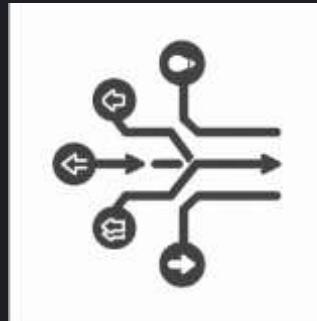
# Fan-Out



- A **single stream** distributes data to **multiple consumers**
- Distribute data to different applications

VS

# Fan-In

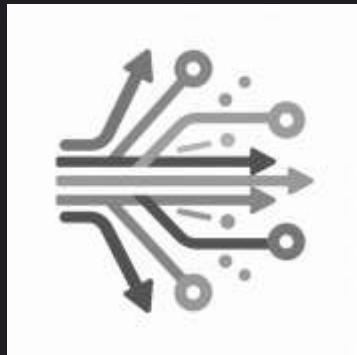


- Multiple sources converge towards a **single destination**.
- Destination can be another stream or another storage system.
- Combine data from different sensors into a **single stream**





# Enhanced Fan-Out

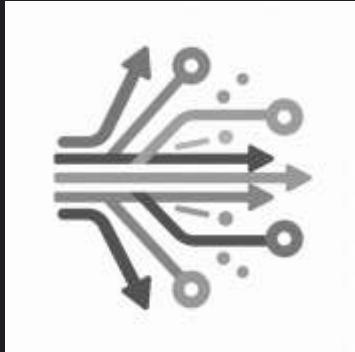


- Traditionally: Shared Through-put (*standard consumer*)  
⇒ Potential bottleneck
- Solution: **Enhanced fan-out**
  - ⇒ **Push data through HTTP/2**
  - ⇒ Each registered consumer its own dedicated read throughput  
(up to 20 consumers)
  - ⇒ Each consumer: Up to 2 MB/second per shard
  - ⇒ Add more consumers without creating bottleneck
- ⇒ E.g. 10 consumers ⇒ 20 MB/s instead of 2 MB/s





# Enhanced Fan-Out - Benefits

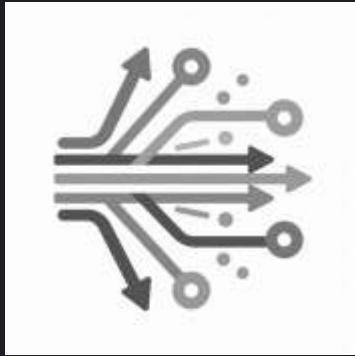


- **Increased Throughput:**  
Each consumer gets its own dedicated bandwidth, allowing for faster data processing.
- **Improved Scalability:**  
The system can handle more concurrent consumers without performance degradation.
- **Reduced Latency:**  
Improved latency (~70ms)
- **Simplified Application Development:**  
You don't need to implement complex logic to manage shared throughput between consumers.





# Enhanced Fan-Out – When to Use?



- **Higher Cost**
- **High number of consumers**
- **Require reduced latency**





# Troubleshooting & Performance in Kinesis



003-1040559

1250 003-77156.8

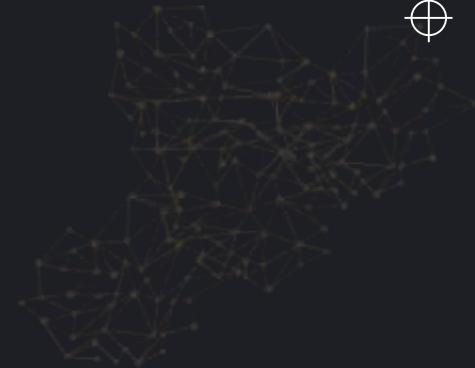
1760 0009-14563.7

73273





# Kinesis Data Stream



## Performance Challenges for Producers:

### ***Slow Write Rates***

**1) Problem:** Service Limits & Throttling

**Solution:** Monitor Throughput Exceptions

**2) Problem:** Uneven data distribution to shards - "Hot Shards"

**Solution:** Effective partition key strategy

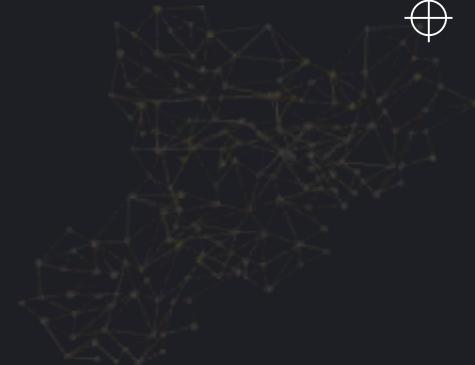
**3) Problem:** High throughput and small batches can be inefficient

**Solution:** Batch records to aggregate multiple records





# Kinesis Data Stream



## Performance Challenges for Consumers:

### ***Slow Read Rates***

- 1) Problem:** Hitting shard read limits (per-shard limit)  
**Solution:** Increasing shard count
- 2) Problem:** Low value for maximum number of **GetRecords** per call  
**Solution:** System-defaults are generally recommended.
- 3) Problem:** Logic inside **ProcessRecords** takes longer than expected.  
**Solution:** Change logic, test with empty records





# Kinesis Data Stream



## **GetRecords returns empty records array**

Every call to GetRecords returns a ShardIterator value (must be used in the next iteration).

ShardIterator NULL ⇒ Shard has been closed.

### **Empty records reasons:**

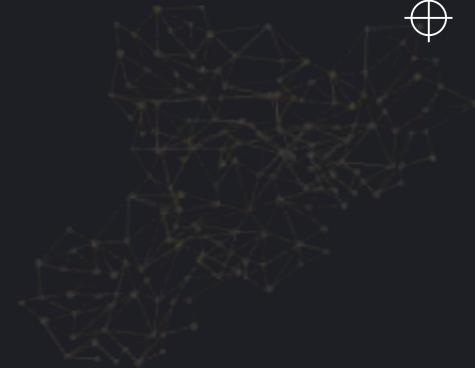
- 1) No more data in the shard.**
- 2) No data pointed to by the ShardIterator.**

⇒ It is not an issue and usually automatically handled (Kinesis Client Library).





# Kinesis Data Stream



## Additional Issues

**1) Problem:** Skipped records

**Solution:** Might be due to unhandled exceptions.  
Handle all exceptions in `processRecords`

**2) Problem:** Expired Shard Iterators.

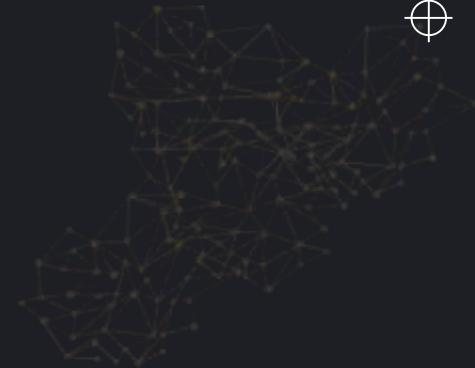
DynamoDB table used by Kinesis does not have enough capacity to store the data. Large number of shards.

**Solution:** Not called GetRecords for more than 5min.  
Increase write capacity to shard to table.





# Kinesis Data Stream



## Additional Issues

**3) Problem:** Consumers falling behind

**Solution:** Increase retention

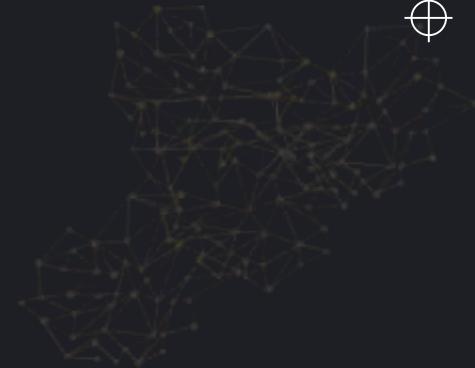
Monitor `GetRecords.IteratorAgeMilliseconds` or `MillisBehindLatest` metrics

- **Spikes** ⇒ API failures (transient)
- **Steady increases** ⇒ Limited resources or processing logic





# Kinesis Data Stream



## Additional Issues

### 4) Problem:

#### Solution:

Unauthorized KMS Master Key Permission Error

Writing to an encrypted stream without the necessary permissions on the KMS master key. Ensure you have the correct permissions via AWS KMS and IAM policies





# **Amazon Kinesis Data Firehose**

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





01

02

03

04

05

06



01

02

03

04

05

06

# Overview

- **Fully Managed Service:**

Automates data streaming and loading, reducing the need for manual setup and administration.

- **Effortless Data Handling:**

Captures, transforms, and loads data streams into AWS data stores and analytic tools with minimal effort.

- **Automatic Scaling:**

Dynamically adjusts to the volume of incoming data, providing seamless scalability.

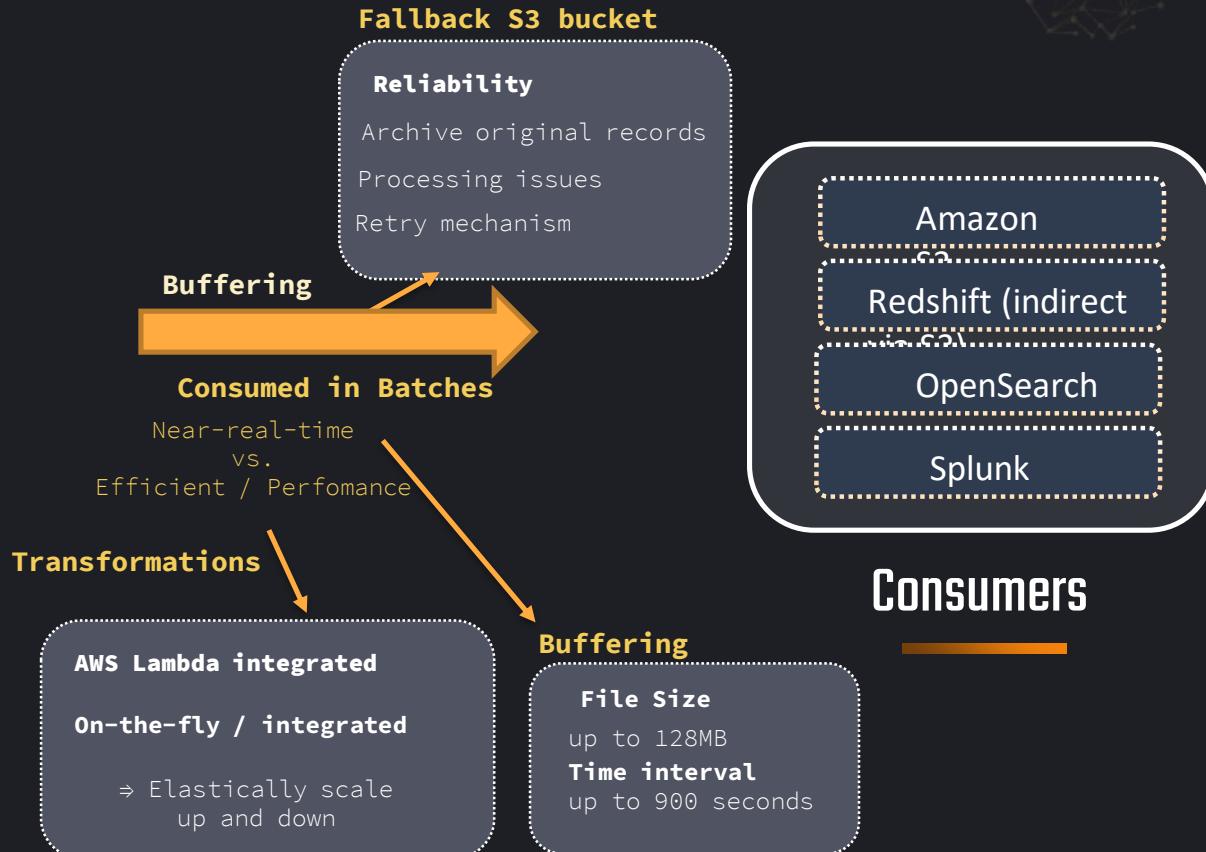


# Amazon Kinesis Data Firehose



## Producer

generate data streams





01

02

03

04

05

06



01

02

03

04

05

06

# Key Features

- **Near Real-Time Processing:**

Buffers based on time and size, balancing prompt data handling with the efficiency of batch deliveries.

- **Broad Data Format Support:**

Handles multiple data formats and conversions (e.g., JSON to Parquet/ORC).

- **Data Security and Transfer Efficiency:**

Compresses and encrypts data to enhance security during transfer.

- **Real-Time Analytics and Metrics:**

Ideal for scenarios requiring quick data analysis, like log or event data capture.





01

02

03

04

05

06



01

02

03

04

05

06

# Pricing

## **Consumption-Based Pricing:**

Costs are tied to the volume of data processed, suitable for various operational scales.



# Kinesis Data Streams



# Kinesis Firehose



VS

- Manual shard setup
- Coding consumers/producers
- Real-time (200ms/70ms latency)
- Data Storage up to 365 days

- Fully managed experience
- Focus on delivery (efficient)
- Near-real-time
- Ease of use





# Managed Service for Apache Flink



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





01

02

03

04

05

06



01

02

03

04

05

06

# What is Amazon Managed Service for Apache Flink?

- **Fully Managed Service:**

For querying and processing of data streams.

- **Real-time streaming processing and analytics:**

Scalable real-time analytics using Flink

Real-time monitoring

Real-time website traffic

Streaming ETL

SQL  
Python  
Scala  
Java

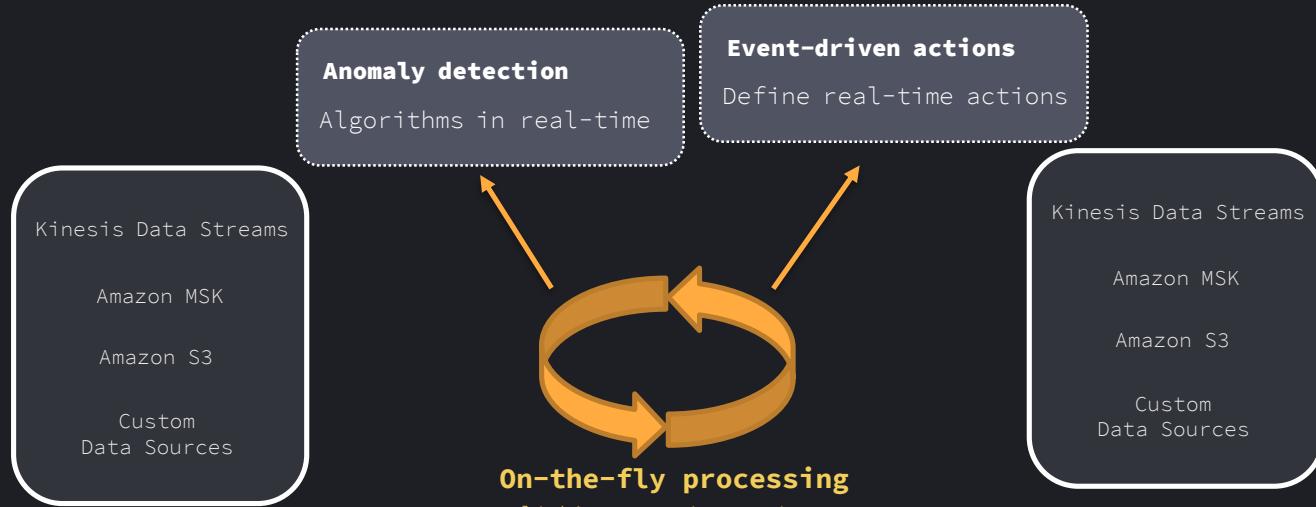
- **Apache Flink under the hood:**

Open-source stream processing framework managed by AWS

- **Serverless & Scalable**



# AMS Flink



## Flink Sources

Some data streams  
Minimal Code

### Low Latency

Near real-time

### Stateful Computations

Maintain state based on incoming data

## Flink Sinks

Checkpoints & Snapshots  
Ensuring fault tolerance





01

02

03

04

05

06



01

02

03

04

05

06

# Pricing

- **Pay-as-You-Go**

Consumption based

- **Kinesis Processing Units (KPU)**

Charges are based on the number of KPU  
(1 KPU has 1 vCPU and 4 GB of memory)

- **Application Orchestration**

Each application requires one additional KPU for orchestration.

- **Storage and Backups**

per GB per month

- **Automatic Scaling**

Number of KPU automatically scaled based on needs.  
Manually provisioning possible.

- **AMS Flink Studio (Interactive Mode)**

Charged for two additional KPU





# Amazon Managed Streaming

# for Apache Kafka (MSK)





01

02

03

04

05

06



01

02

03

04

05

06

# Overview

- **Managed Kafka Service**

Streamlines setup and management of Apache Kafka.

- **Purpose**

Enables processing of streaming data in real-time.

# Architecture

- **Kafka Brokers:**

Servers that manage the publishing and subscribing of records

- **ZooKeeper Nodes:**

Coordinates Kafka brokers and manages cluster metadata.





01

02

03

04

05

06



01

02

03

04

05

06

# High Availability & Storage

- **Multi-AZ Deployment**

Clusters distributed across Availability Zones for fault tolerance.

- **EBS Storage**

Utilizes Elastic Block Store for durable message storage, supporting seamless data recovery.

## Custom Configuration & Scalability

- **Message Size Flexibility**

Configurable message sizes (up to 10 MB), more flexibility than Kinesis (1 MB limit)

- **Scalable Clusters**

Easily adjustable broker counts and storage sizes to handle varying workloads.





01

02

03

04

05

06



01

02

03

04

05

06

# Producers & Consumers

- **Producers**

Applications that send data to MSK clusters.

- **Consumers**

Applications or services retrieving and processing data from MSK.

## Comparison with Amazon Kinesis

- **Customization vs. Convenience**

MSK offers in-depth configurability; Kinesis offers easier setup with fixed limits.

- **Message Handling**

MSK supports larger message sizes, critical for specific use cases.





# MSK

- More granular control, more management
- More complex pipelines
- **Message size:**  
Up to 10MB (Default 1 MB)
- Topics & Partitions

VS

# Kinesis

- More Managed Experience
- Straight-forward setup
- **Message size:**  
1 MB Limit
- Streams & Shards





# MSK

- **Encryption**  
In-flight TLS encryption  
OR plain text possible
- **At Rest:**  
Supports KMS encryption
- **Access Control:**
  - **Mutual TLS**
  - **SASL/SCRAM** Username/password authentication mechanism, also relying on Kafka ACLs.
  - **IAM Access Control**  
authentication and authorization using IAM

VS

# Kinesis

- **Encryption**  
TLS encryption in-flight by default
- **At Rest:**  
Supports KMS encryption  
Straight-forward setup
- **Access Control:**
  - Uses IAM policies for both authentication and authorization





# Section 7:

# Storage with S3



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Partitioning



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Overview

- Partitioning data through a combination of bucket and folder structure.

01

02

03

04

05

06

01

02

03

04

05

06





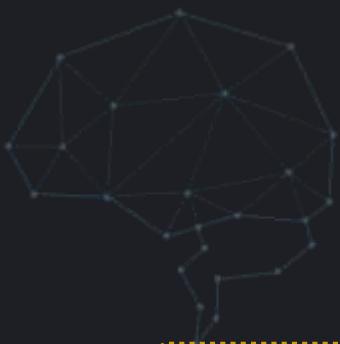
# Importance of Folder Structure

## Data Management

- Simplifying tasks such as data retention and archival.
- Facilitating easy archiving or deletion of older data partitions

- Improving query performance by allowing queries to process only relevant data subsets.
- Reducing the amount of data scanned in queries, leading to cost savings.





# Implementation of Partitioning

## Organizing Data



- Organizing data into folders and subfolders, including the use of buckets.
- Enabling the use of Glue crawlers to automatically create partition keys.

## Partitioning Examples

2021

/month=11/day=05/filename.txt

- Time-based partitioning example with S3 keys.

01

02

03

04

05

06



# Importance of Maintaining **The Glue Catalog** for Managing Metadata and Defining Partitions





# Amazon Athena



Query using Athena.



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Amazon Athena



Query using Athena

The screenshot shows the Amazon Athena console interface. On the left, there's a sidebar titled 'Data' with sections for 'Data source' (set to 'AwsDataCatalog') and 'Database' (set to 'retail\_db'). Below these are 'Tables and views' and a search bar. Under 'Tables', there are three tables listed: 'retail\_12345\_test', 'parquet', and 'raw\_sales'. Under 'Views', there is one view listed: 'sales'. The main area is titled 'Query 1' and contains the SQL query: `SELECT * FROM "retail_12345_test"."parquet" limit 10;`. Below the query are buttons for 'Run query', 'Explain', 'Commit', 'Clear', and 'Create'. The 'Query results' tab is selected, showing a table titled 'Completed' with 10 rows of data. The columns in the results table are 'date', 'product\_id', and 'quantity'. The data is as follows:

	date	product_id	quantity
1	2023-02-02 22:00:00.000	P109	2
2	2023-03-11 10:02:15.000	P990	20
3	2023-12-14 18:06:50.000	P157	14
4			
5			
6			
7			
8			
9			
10			



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Storage Classes and Lifecycle configuration





# Storage Classes



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# S3 Storage Classes



S3 Standard

S3 Intelligent-Tiering

Amazon S3 Express One Zone

S3 Standard-Infrequent Access

S3 One Zone-Infrequent Access

S3 Glacier Instant Retrieval

S3 Glacier Flexible Retrieval

S3 Glacier Deep Archive





# S3 Storage Classes

## Amazon S3 Standard (S3 Standard)

- It is best for storing frequently accessed data.
- It delivers low latency and high throughput.
- Is appropriate for

Cloud applications

Dynamic websites

Content distribution

Big data analytics

Mobile and gaming  
applications





# S3 Storage Classes

## S3 Intelligent-Tiering

- Best for unknown or changing access patterns.
- Automatically moves data to the most cost-effective access tier based on access frequency

Frequent  
access tier

Infrequent  
access tier

Archive  
Access tier





# S3 Storage Classes



## Amazon S3 Express One Zone

- Is a high-performance, single-Availability Zone storage class.
- It can improve data access speeds by 10x and reduce request costs by 50% compared to S3 Standard.
- Data is stored in a different bucket type—an Amazon S3 directory bucket





# S3 Storage Classes

## **S3 Standard-Infrequent Access (S3 Standard-IA)**

- Is best for data that is accessed less frequently, but requires rapid access when needed.
- ideal for long-term storage, backups, and as a data store for disaster recovery files.

## **S3 One Zone-Infrequent Access (S3 One Zone-IA)**

- Is best for data that is accessed less frequently, but requires rapid access when needed.
- Stores data in a single AZ.
- Ideal for storing secondary backup copies of on premises data





# S3 Storage Classes



## S3 Glacier Storage Classes

- Are purpose-built for data archiving.



S3 Glacier Instant  
Retrieval storage



S3 Glacier  
Flexible Retrieval



S3 Glacier Deep  
Archive





# S3 Storage Classes



S3 Glacier Instant  
Retrieval storage

- Is best for long-lived data that is rarely accessed and requires retrieval in milliseconds.
- Delivers the fastest access to archive storage
- Ideal for archive data that needs immediate access.





# S3 Storage Classes



S3 Glacier  
Flexible Retrieval

- Is best for archive data that is accessed 1–2 times per year and is retrieved asynchronously.
- It is an ideal solution for backup, disaster recovery, offsite data storage needs
- Configurable retrieval times, from minutes to hours, with free bulk retrievals.



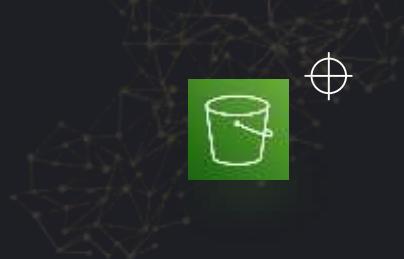


# S3 Storage Classes



S3 Glacier Deep  
Archive

- Is the cheapest archival option
- Supports long-term retention and digital preservation for data that may be accessed once or twice in a year.
- Objects can be restored within 12 hours.





# Lifecycle configuration



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Lifecycle configuration

- Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects.

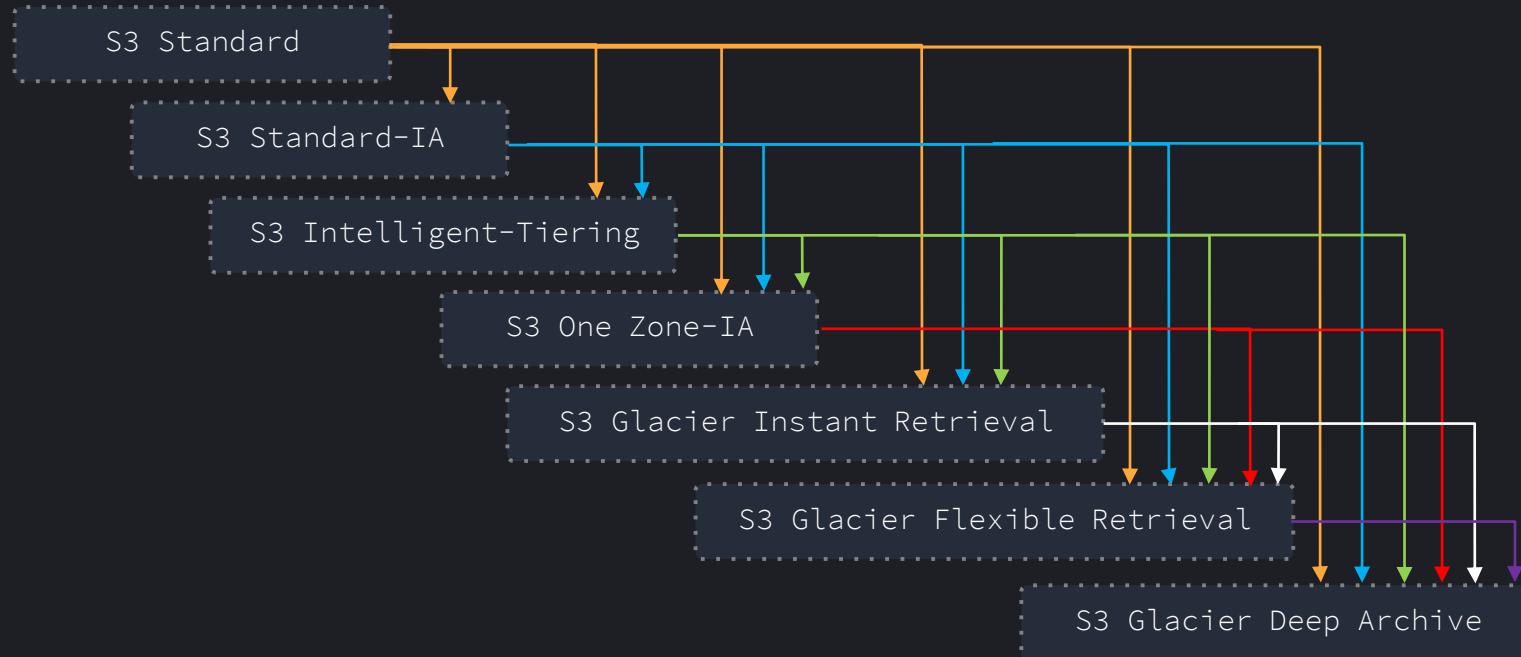
Transition actions

Expiration actions



# Lifecycle configuration

## Supported lifecycle transitions





# Versioning



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





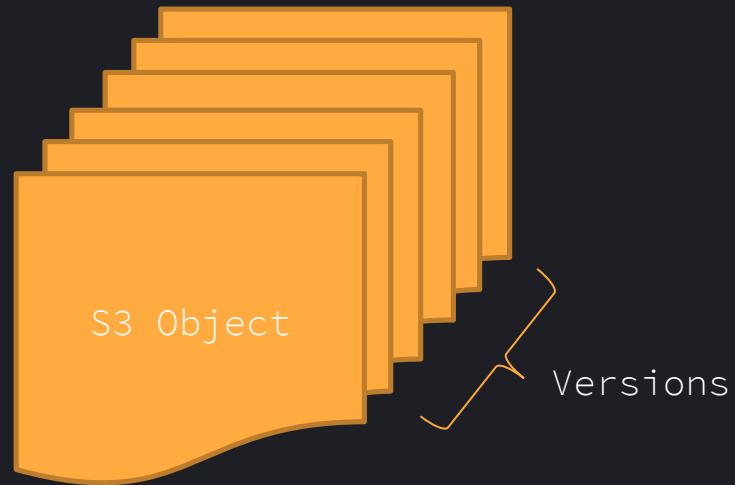
# Versioning



- Helps you manage and preserve the versions of objects in your bucket.
- Useful for

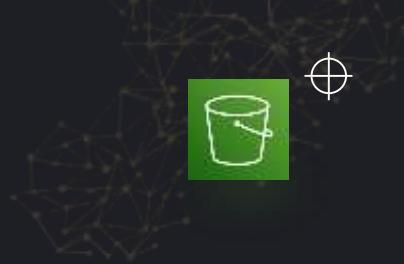
Update Safety

Delete Protection





# Versioning



- Is disabled by default
- You can enable s3 bucket versioning using

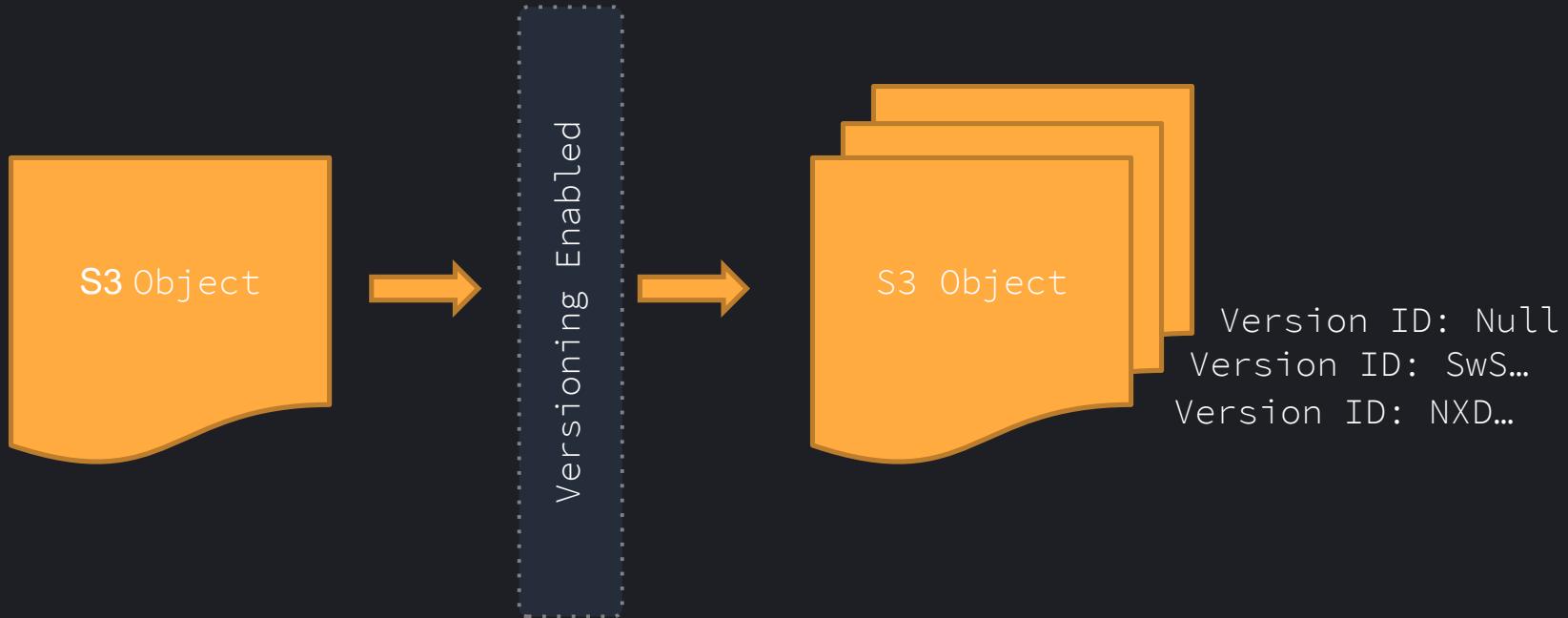
S3 Console

AWS CLI

AWS SDK



# Versioning



# Versioning

my-sample-bucket-1234 [Info](#)

Objects Properties Permissions Metrics Management Access Points

**Objects (3) [Info](#)**

C Copy S3 URI Copy URL Download Open Delete Actions Create folder Upload

Objects are fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Find objects by prefix Show versions < 1 > @

<input type="checkbox"/>	Name	Type	Version ID	Last modified	Size	Storage class
<input type="checkbox"/>	home.JPG	Delete marker	zPx3ZdlnLU vhnBHHzsv CY02XyxSP WQHi	May 14, 2024, 14:51:24 (UTC+03:00)	0 B	-
<input type="checkbox"/>	<a href="#">home.JPG</a>	JPG	sh6hwwRAr 5N4az9UhS golCoOPzd wzpor	May 14, 2024, 14:50:43 (UTC+03:00)	36.4 KB	Standard
<input type="checkbox"/>	<a href="#">home.JPG</a>	JPG	null	May 14, 2024, 14:49:45 (UTC+03:00)	36.4 KB	Standard



# Encryption and Bucket Policy



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Encryption



003-1040559

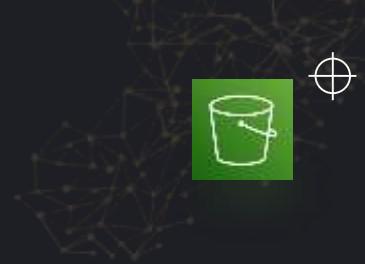
1250 003-77156.8

1760 0009-14563.7

73273



# Encryption

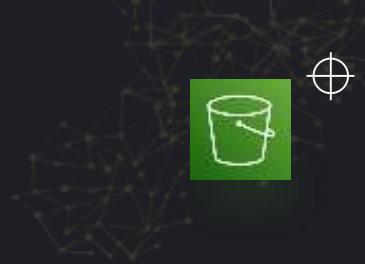


## Encryption in transit





# Encryption



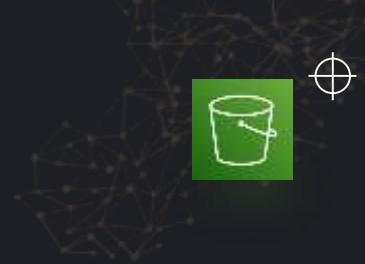
## Encryption at rest

- Data is encrypted and then stored as it is.
- Can be done at Server Side or Client Side.
- All buckets have encryption at rest configured by default.





# Encryption



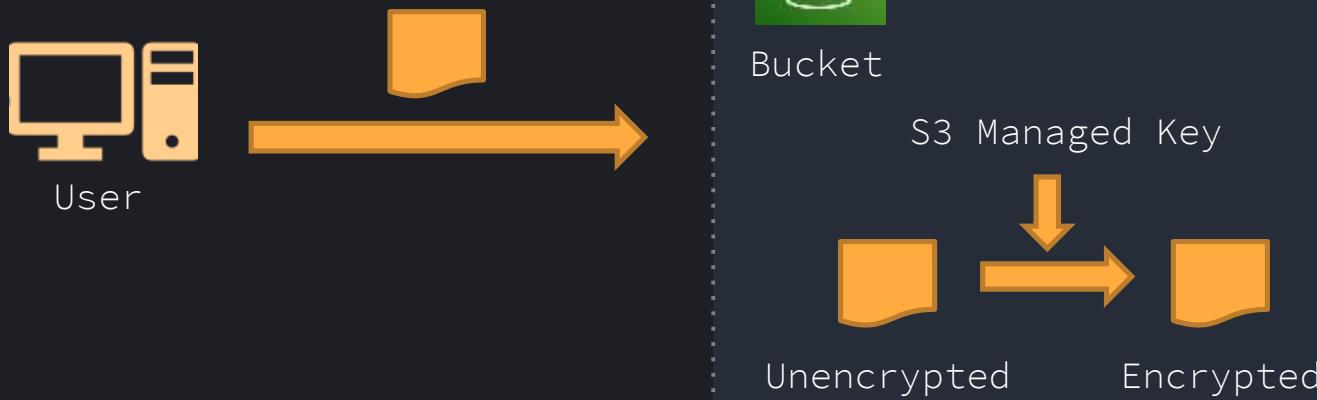
## **Server Side Encryption methods**

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS KMS (SSE-KMS)
- Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)



# Encryption

**Server-side encryption with Amazon S3 managed keys (SSE-S3)**

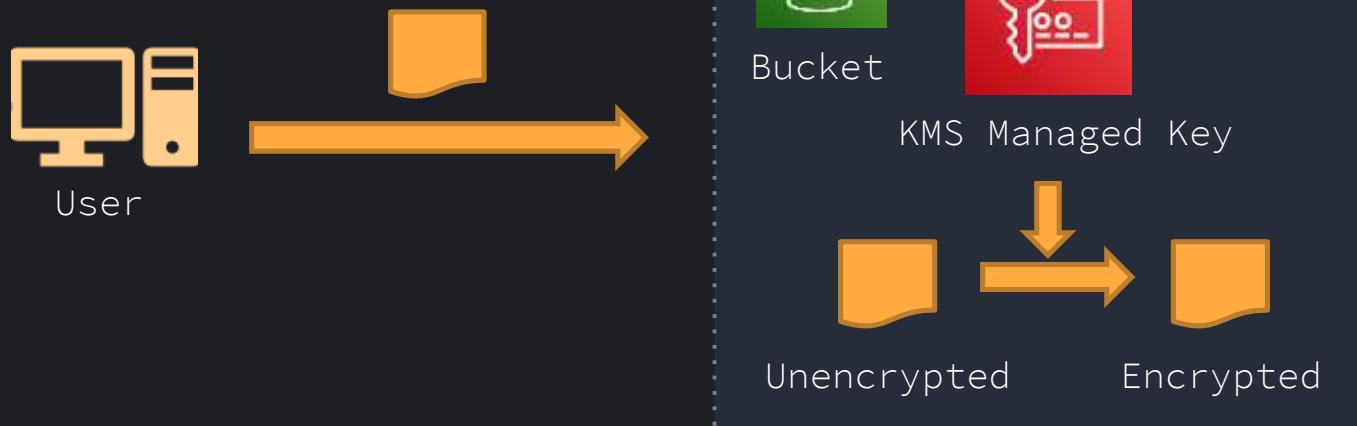




# Encryption

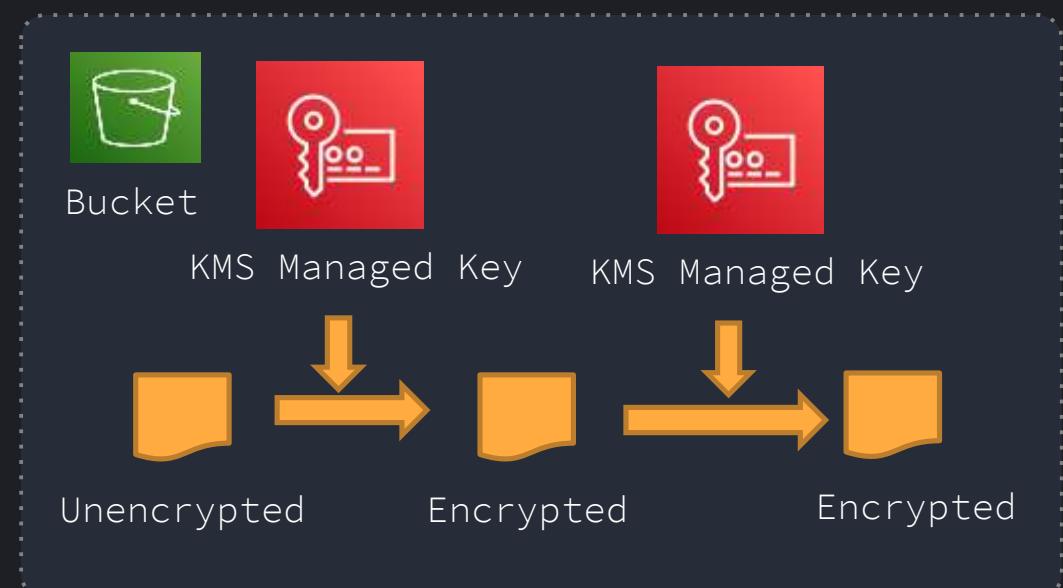


## Server-side encryption with AWS KMS (SSE-KMS)



# Encryption

**Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)**





# Encryption

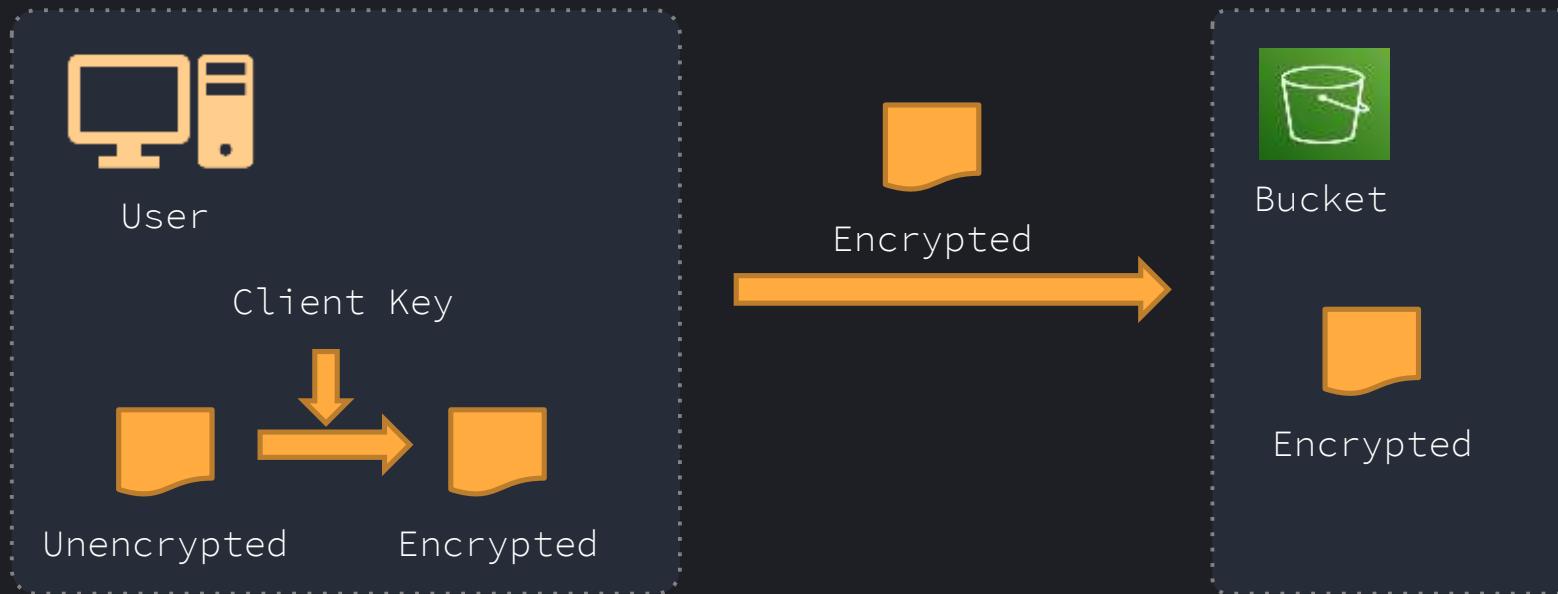


**Server-side encryption with customer-provided keys (SSE-C)**



# Encryption

## Client-side Encryption





# Bucket Policy



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Bucket Policy

- Defines permissions and access control for objects within an Amazon S3 bucket.
- Is written in JSON format.

Specifies

- Who (which users or services) can access the bucket and
- What actions they can perform on the bucket and its objects.

```
1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Effect": "Allow",
5         "Principal": "*",
6         "Action": "s3:GetObject",
7         "Resource": "arn:aws:s3:::yourbucketname/*",
8         "Condition": {
9             "IpAddress": {
10                 "aws:SourceIp": "19.168.100.0/24"
11             },
12             "Bool": {
13                 "aws:SecureTransport": "true"
14             }
15         }
16     }
17 ]
18 ]
19 }
```





# Access Points and Object Lambda



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





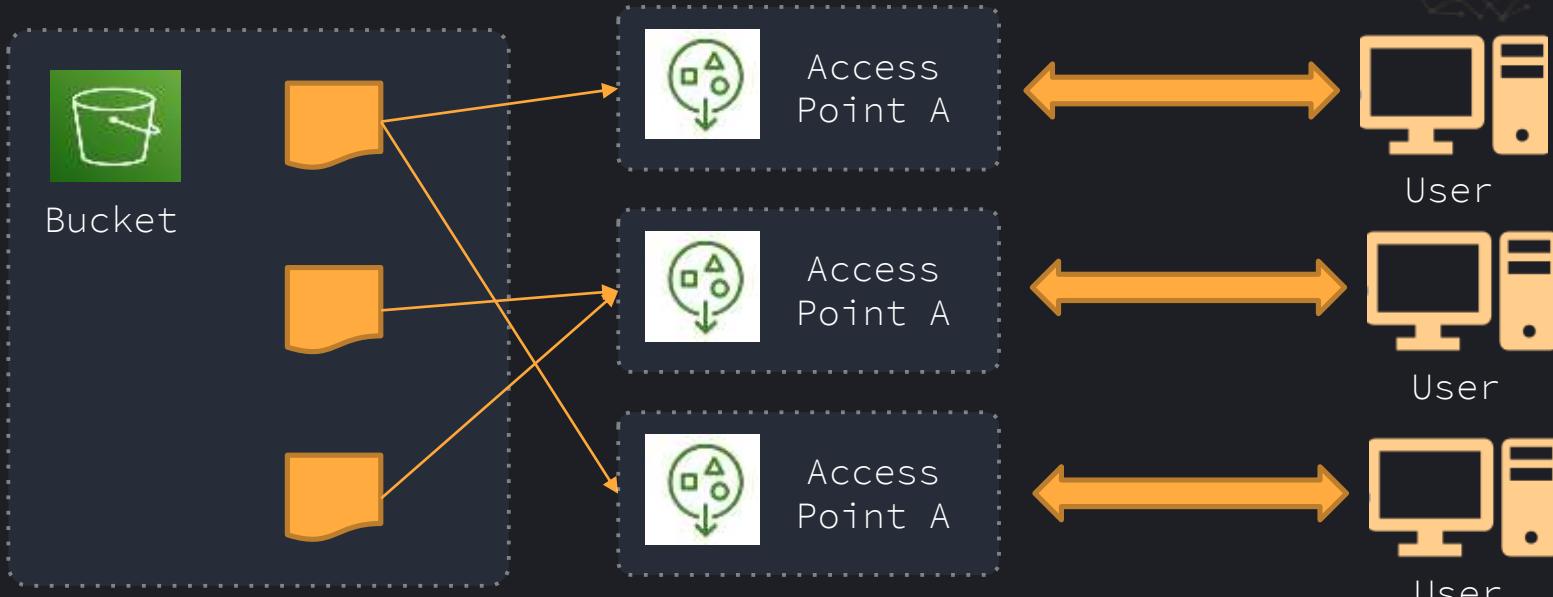
# S3 Access Points

003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

# Access Points



- Lets you create customized entry points to a bucket, each with its own unique policies for access control.



# Access Points

Each Access Point has

- DNS name (Internet Origin or VPC Origin)
- Access point policy (similar to bucket policy)

## Features

Customized  
Permissions

Improved Scalability  
and Organization

Enhanced  
Security





# Object Lambda

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# Object Lambda

With Object Lambda you can

- Transform data as it is retrieved
- No need to create a separate version of the data

## Use Cases

Filtering Sensitive Information

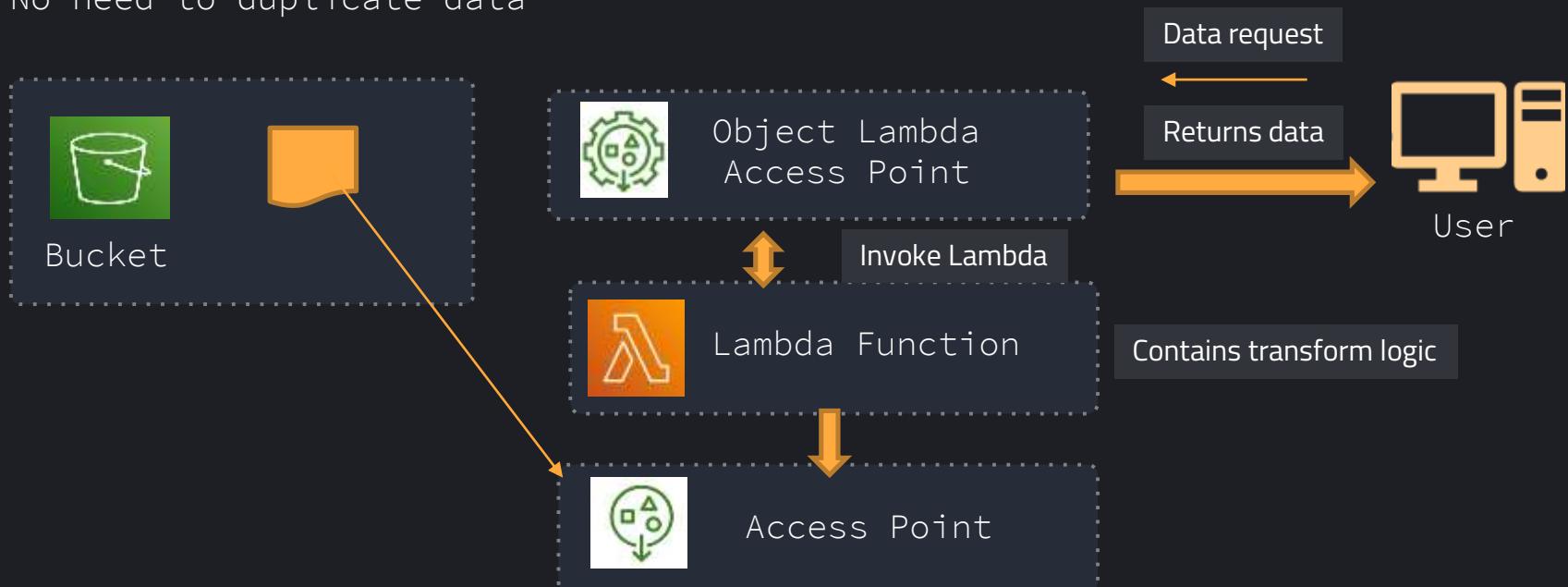
Converting Data Formats

Augmenting Data



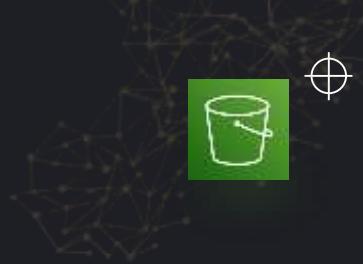
# Object Lambda

- Add your own code to process data
- No need to duplicate data





# Object Lambda



## Use Cases

- Redact PII (Personally identifiable information) for analytics
- Convert data formats such as XML to JSON
- Augmenting Data with information from other services





# S3 Event Notification



003-1040559

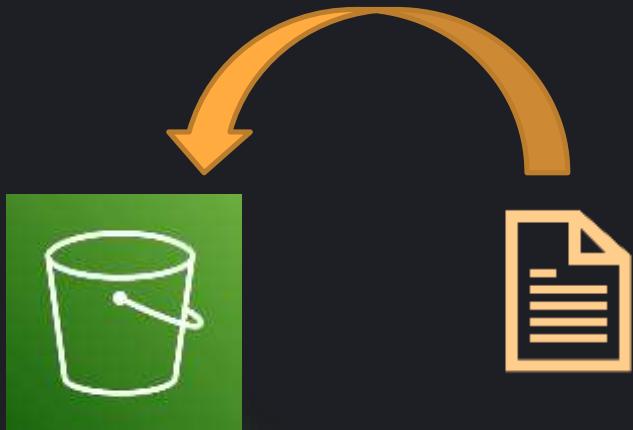
1250 003-77156.8

1760 0009-14563.7 73273



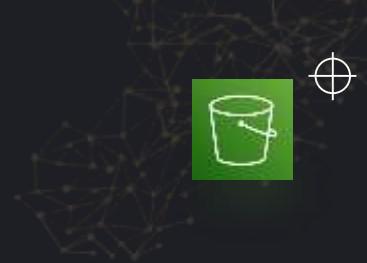


# S3 Event Notification





# S3 Event Notification



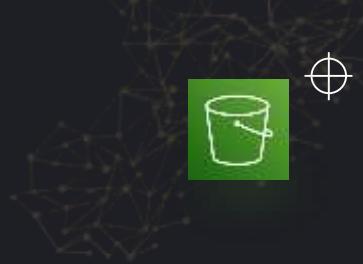
## The Event can be

- New object created events
- Object removal events
- Restore object events
- Replication events





# S3 Event Notification



- S3 Lifecycle expiration events
- S3 Lifecycle transition events
- S3 Intelligent-Tiering automatic archival events
- Object tagging events
- Object ACL PUT events





# S3 Event Notification



**S3 can send event notification messages to**



Simple Notification Service topics



Simple Queue Service queues



Lambda function



EventBridge





# S3 Event Notification

## The Event Examples

- s3:ObjectCreated:Put:  
Object is uploaded to the bucket using the PUT method.
- s3:ObjectCreated:Post:  
Object is uploaded to the bucket using the POST method.
- s3:ObjectRemoved:Delete:  
Object is deleted from the bucket.

## Wildcards

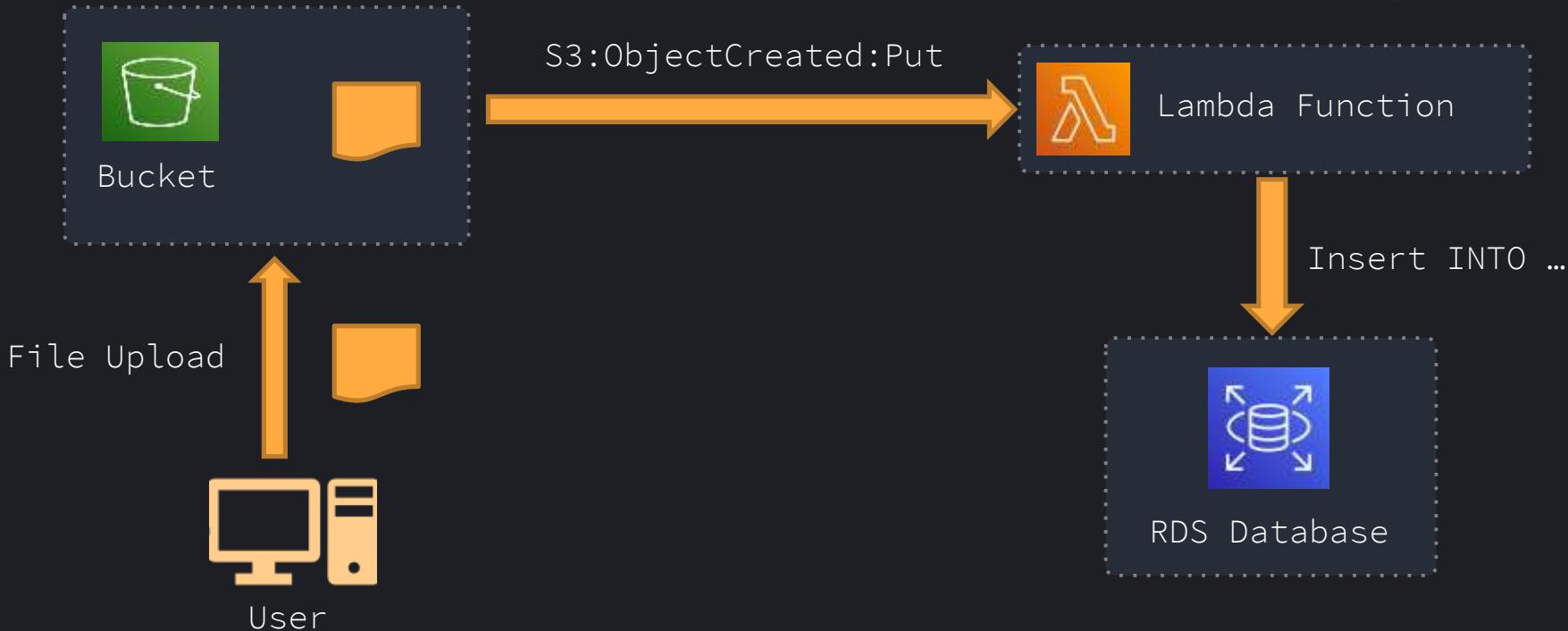
- s3:ObjectCreated:\*
- Captures all object creation events

## Prefix and Suffix Filters

Filter to objects in a directory to trigger the event (e.g. uploads/images/)  
Filter to file types (e.g. '.jpg') to trigger the event



# S3 Event Notification





# S3 Event Notification



003-1040559

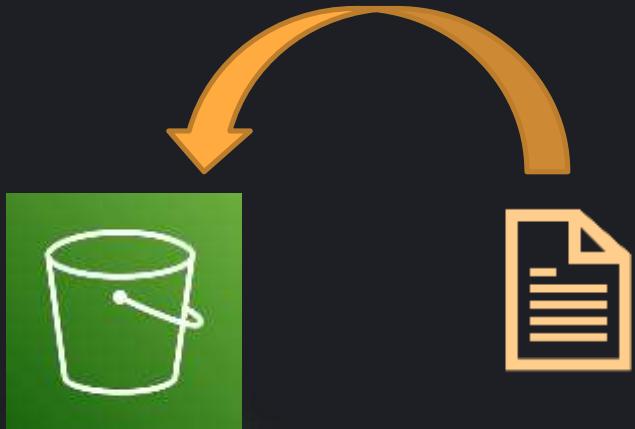
1250 003-77156.8

1760 0009-14563.7 73273



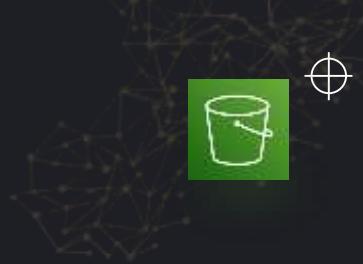


# S3 Event Notification





# S3 Event Notification



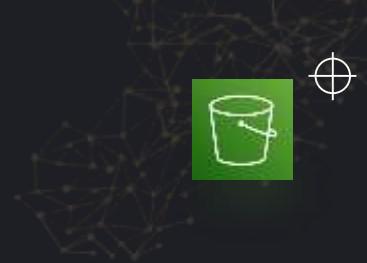
## The Event can be

- New object created events
- Object removal events
- Restore object events
- Replication events





# S3 Event Notification



- S3 Lifecycle expiration events
- S3 Lifecycle transition events
- S3 Intelligent-Tiering automatic archival events
- Object tagging events
- Object ACL PUT events





# S3 Event Notification



**S3 can send event notification messages to**



Simple Notification Service topics



Simple Queue Service queues



Lambda function



EventBridge





# S3 Event Notification

## The Event Examples

- s3:ObjectCreated:Put:  
Object is uploaded to the bucket using the PUT method.
- s3:ObjectCreated:Post:  
Object is uploaded to the bucket using the POST method.
- s3:ObjectRemoved:Delete:  
Object is deleted from the bucket.

## Wildcards

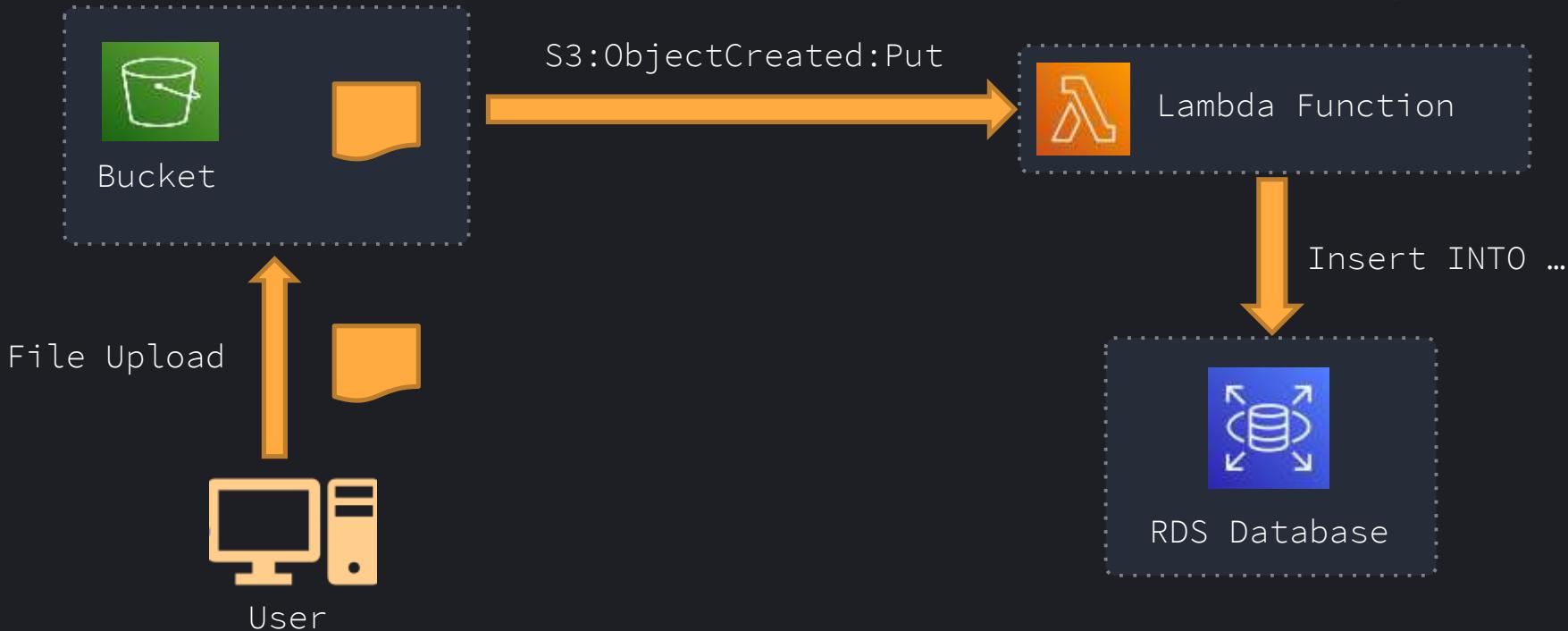
- s3:ObjectCreated:\*
- Captures all object creation events

## Prefix and Suffix Filters

Filter to objects in a directory to trigger the event (e.g. uploads/images/)  
Filter to file types (e.g. '.jpg') to trigger the event



# S3 Event Notification





## Section 8:

# Other Storage Services



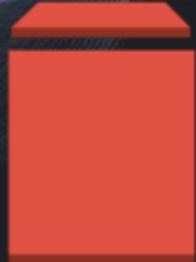
003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





EBS



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# AWS EBS



Single availability zone

EC2 instances

- Scalable block solution in AWS
- EBS = Elastic Block Store
- Persistent storage for a variety of data types

Highly scalable

EC2 > Volumes > vol-03aaa55cbe6dbae4e

Actions	Delete	Modify	
vol-03aaa55cbe6dbae4e			
Volume ID	Size	Type	Volume status
vol-03aaa55cbe6dbae4e	8 GiB	gp2	Okay
AWS Compute Optimizer finding	Volume state	IOPS	Throughput
(Opt-in to AWS Compute Optimizer for recommendations.)	In-use	100	-
<a href="#">Learn more</a>			

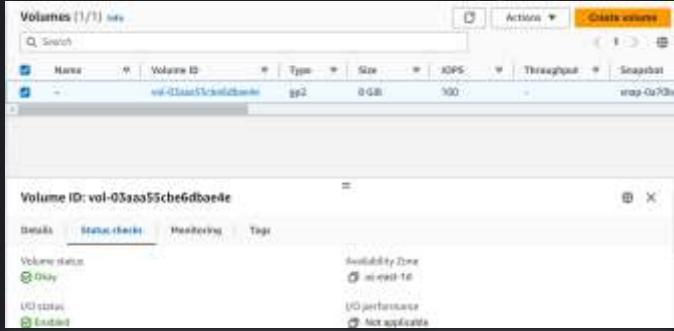


# AWS EBS - Storage

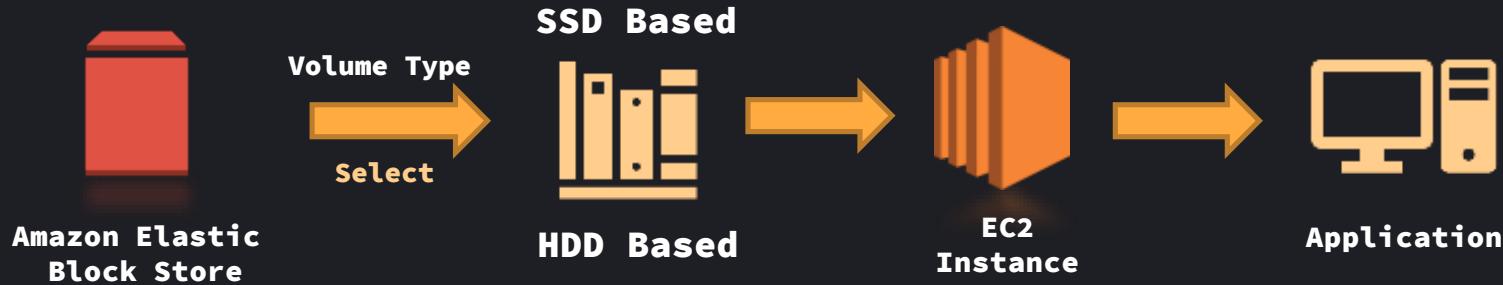


## Key features

- Scalability
- Durability
- Block-level storage
- Persistent storage
- High performance
- Cost effective



# EBS functionality



⇒ Bound to a specific AZ

⇒ Use snapshots for different AZ or regions



# EBS Snapshots

- **Definition:**

Instantaneous copies of AWS resources.

- **Incremental Backups:** Improves efficiency.

- **Block-Level Backup:** Detailed data capturing.

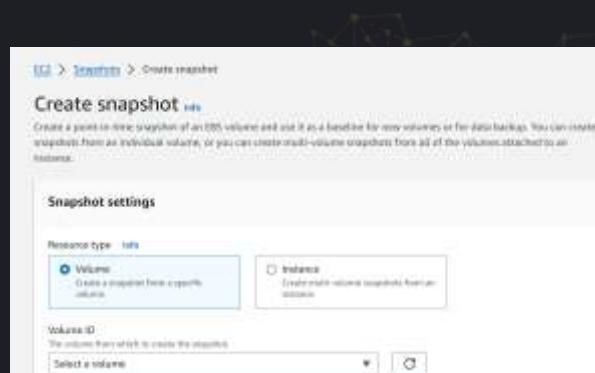
- **Data Consistency:** Before creating a snapshot, AWS ensures data consistency by temporarily pausing I/O operations.

- **S3 Compatible**

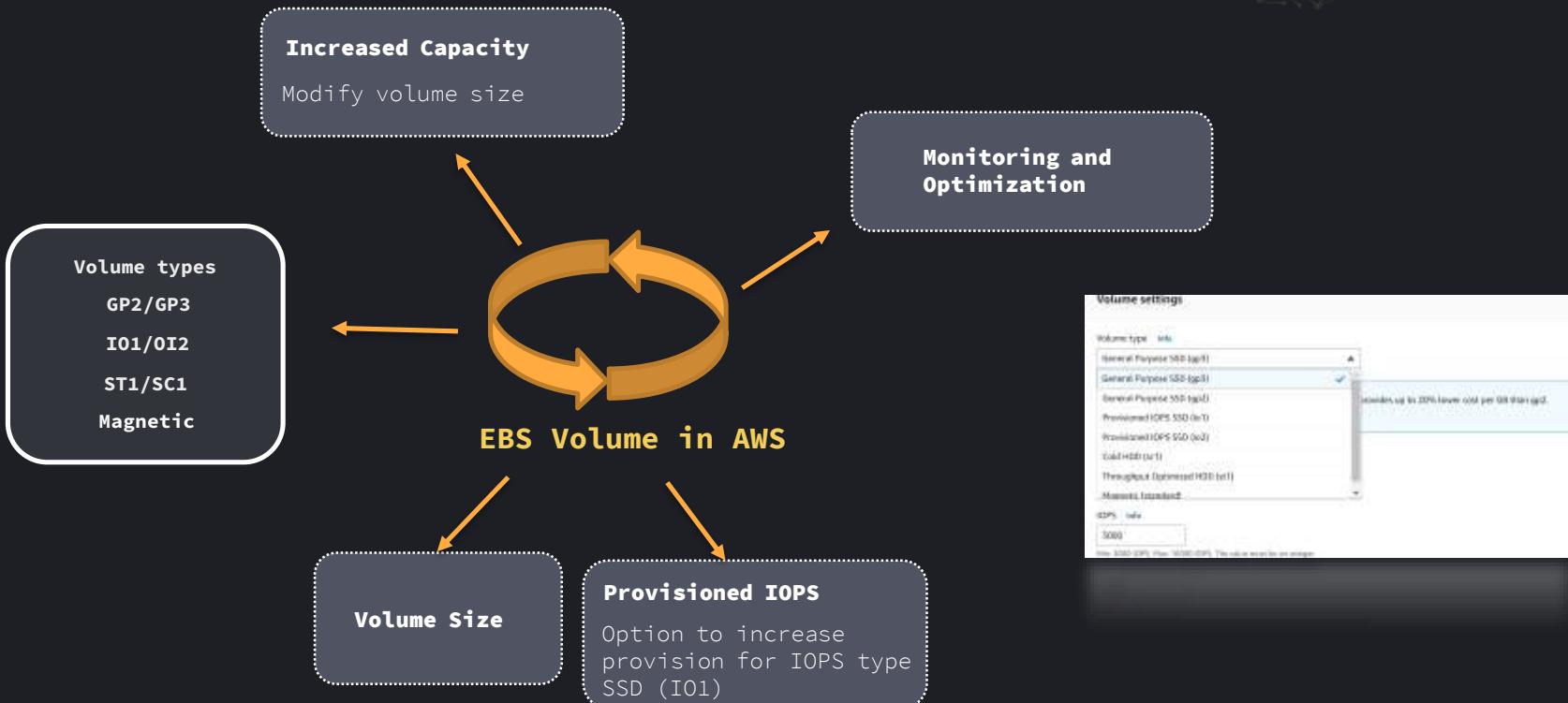
- **Lifecycle Management:** Allows you to define lifecycle policies for EBS snapshots, enabling automation of snapshot management tasks.

- **Data Recovery:** Dynamic use of snapshots.

- **Cost Management:** Reasonable cost.

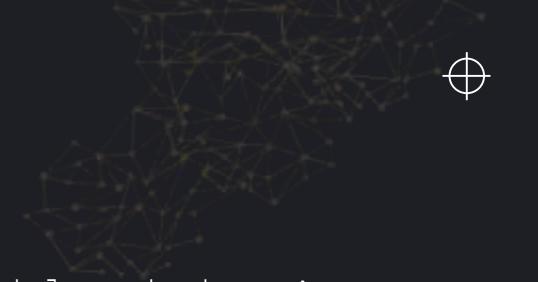


# EBS Capacity provisioning





# Delete on Termination



Determines whether the volume should be automatically deleted when its associated EC2 instance is terminated.

- **Enabled:**

The volume will be automatically deleted by AWS when the associated EC2 instance is terminated.

- **Disabled:**

The EBS volume will persist even after the associated EC2 instance is terminated.

⇒ Managing the deletion attribute

EBS Volumes		
Volume 1 (AMI Read (Custom))		
Storage type info	Device name - required info	Snapshot info
EBS	/dev/xvda	snap-00022cfe1f73b569d
Size (GiB) info	Volume type info	IDP5 info
8	gp2	100 / 3000
Delete on termination info	Encrypted info	KMS key info
No	No	No





# Amazon EFS



003-1040559

1250 003-77156.8

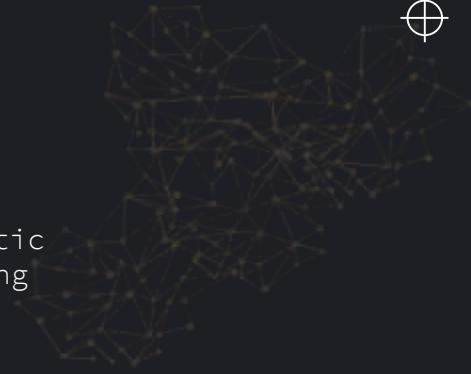
1760 0009-14563.7

73273





# Elastic File System



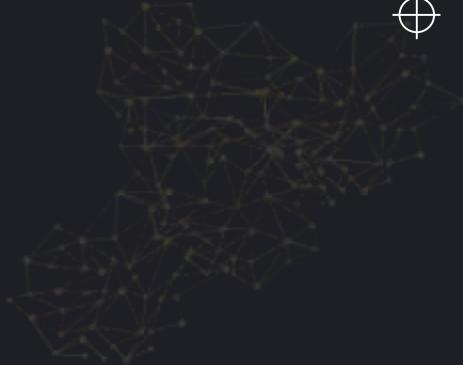
Amazon Elastic File System (Amazon EFS) provides serverless, fully elastic **file storage** so that you can **share files** without provisioning or managing storage capacity and performance.

- Multi-AZ Availability
- Scalability
- Shared File System
- Elasticity
- NFSv4.1 protocol
- Performance Mode:
  - General Purpose (broad range)
  - Max I/O (high throughput / IOPS)
- Pay as You Go Pricing



# Posix System Standard API

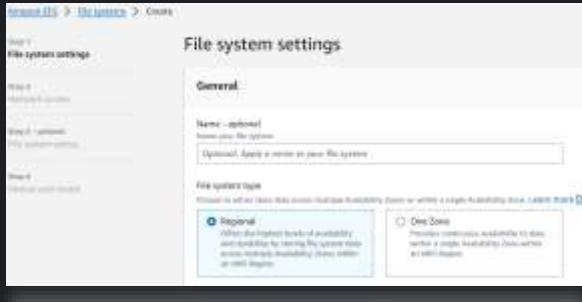
The POSIX (Portable Operating System Interface) standard defines a set of APIs for compatibility between various UNIX-based operating systems.



- **Posix APIs** Wide range of functionality.
- **POSIX** simplifies the porting of applications to Linux and fosters interoperability between different platforms.



# Performance & Storage classes on EFS



- Performance Scaling
- Performance and Throughput Modes



- Supports 1000 concurrent NFS



- Storage Classes IA / One Zone-IA



- Automatic throughput scaling



- Supports 1000 concurrent NFS



# Performance Mode (General purpose)

- Designed for a wide range of workloads, including latency-sensitive applications and those with mixed read/write operations.
- Offers low latency and good throughput for most use cases.
- Suitable for applications such as web serving, content management, and development environments.
- Automatically scales performance based on the amount of data stored in the file system.

VS

# Performance Mode (Max IOPS)

- Optimized for applications that require the highest levels of aggregate throughput and IOPS.
- Provides higher IOPS and throughput compared to the General Purpose mode, making it suitable for latency-sensitive and I/O-intensive workloads.
- Ideal for applications such as big data analytics, media processing, and database workloads.
- Performance does not scale automatically based on data size; users need to manually adjust provisioned throughput capacity.

## Throughput Mode (Bursting throughput)

- Designed for workloads with **unpredictable or spiky** access patterns.
- Provides burst credits that allow the file system to achieve throughput levels higher than its baseline for short periods, enabling **burst workloads** to achieve high performance without provisioning throughput capacity.
- Suitable for applications with **intermittent usage patterns**, such as development and test environments, or applications with **periodic data processing** tasks.

VS

## Throughput Mode (Provisioned throughput)

- Designed for applications with **predictable or sustained** throughput requirements.
- Users can provision a **specific amount of throughput** (in MiB/s) for the file system, ensuring consistent performance regardless of workload spikes or burst credits.
- Suitable for applications with **continuous data processing**, high-volume data transfers, or large-scale analytics workloads where predictable performance is critical.



# AWS Backup



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Backup

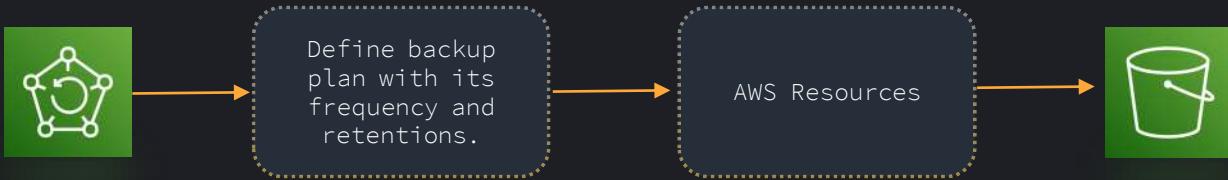


- **Centralized** backup for many AWS Services.
  - EC2, RDS, EFS, S3, DocumentDB...
- Provides **automated backup** scheduling.
- Cross-region & Cross-account backup capabilities.
- You can restore individual files or entire system.
- **Compliance and Security:**
  - Set rules for retentions and lifecycle of backups
  - Access controlled within IAM
- Process can be monitored.





# AWS Backup



## AWS Backup Vault Lock

**Backup Vault**

⇒ a container where backups are stored securely.

- Enhances security and compliance with immutable safeguards





# AWS Backup

## AWS Backup Vault Lock

- **Immutability:**
  - Policies you set become immutable.
  - Uses WORM (Write Once Read Many) method.
  - No one, even root user can change or delete the recovery points.

- **Provides Regulatory Compliance**

- **Vault Lock Modes:**

- Compliance Mode

⇒ Policy cannot be changed or deleted for the lock period.

- Governance Mode

⇒ Specified IAM users can update policies but can not delete recovery points





# Section 9:

# DynamoDB



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# DynamoDB



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# AWS DynamoDB



- Serverless NoSQL database
- NoSQL ⇒ "Not Only SQL" or "non-relational" database





# Relational DB

VS

- Structured data in tables
- Require predefined schema (*tables + rows*)
- Scalability: Vertically (*scale up*)
- SQL Queries (*complex SQL queries possible*)
- Optimized for complex queries

# NoSQL

- Structure, semi-structured and unstructured
- No need for fixed schema (*schemaless*)
- Scalability: Horizontally (*scale out*)
- Varying ways of querying (*no joins, no aggregations*)
- Optimized for performance, large volume & flexibility





# Relational DB

- Example use case:  
Reporting

# NoSQL

- Example use case:  
Big data, real-time analytics

VS





# AWS DynamoDB



- **Data Model:** Supports both key-value and document data model
- **Fully-managed** distributed database
- Performs very well under **high traffic** and **high workloads**
- **High-availability** & durability through distribution & **AZ-replication**
- Supports encryption at rest
- Millisecond latency





# AWS DynamoDB



DynamoDB Consists of:

- **Tables:** A collection of data
- **Items:** Individual records in a table(400 KB max)
- **Attributes:** Properties of a table

*All Tables in DynamoDB have a Primary Key*



## Books

```
{  
    "BookID": "B101",  
    "Title": "The Great Adventure",  
    "Author": "Ella Fitzgerald",  
    "Price": 9.99,  
    "Genres": ["Adventure", "Fantasy"],  
    "PublishYear": 2010  
}  
    ← attribute  
{  
    "BookID": "B102",  
    "Title": "Cooking Simplified",  
    "Author": "James Oliver",  
    "Price": 20.00,  
    "Genres": ["Cooking"],  
    "PublishYear": 2015,  
    "Publisher": {  
        "Name": "Yummy Books",  
        "Location": "New York"  
    }  
}  
  
{  
    "BookID": "B103",  
    "Title": "Modern Web Development",  
    "Author": "Lisa Ray",  
    "Price": 31.50,  
    "Genres": ["Technology", "Educational"],  
    "PublishYear": 2018,  
    "RelatedCourses": ["Web Development 101", "Advanced JavaScript"]  
}
```



- o Each item has a unique identifier: **Primary key**
- o Table is **schemaless**
- o **Nested attributes** are possible



# AWS DynamoDB



- **Primary Keys**

- Uniquely identifies items in a table.
- Items in a table cannot have the same “key”
- They must be scalar (string, number or binary)
- Specified at the creation time of the table
- Required for data organization and retrieval

- **Types of Primary Keys**

**Partition Key (*hash attribute*):**

A single “key” is used to distinguish items in a table

**Composite Key:**

Two “keys” are used to distinguish items in a table:

A partition and a sort key (*range attribute*)



# Partition Key

- Also known as Hash Attribute
- Contains only Partition Keys
- Items cannot have the same partition key

# Composite Key

- Also known as Hash and Range attribute.
- Contains Partition and Sort Key
- Items can have the same Partition key provided the sort key is different.

VS



# AWS DynamoDB



## Simple Partition Key

Primary Key		Attributes	
Student No	Age	Degree	
21829332	19	Bcom	
21789906	17	Science	
21689541	26	Maths	
22587413	30	Tourism	



Partition Key

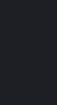


## Composite Key

Primary Key		Attributes	
Student No	Graduation Year	Age	Degree
21829332	2020	19	Bcom
21829332	2018	17	Science
21689541	2016	26	Maths
22587413	2014	30	Tourism



Partition Key



Sort Key





# AWS DynamoDB



## Primary keys

- Most efficient way to retrieve data
- Determines the physical partition where data is stored

**What if you want query data using an attribute?**





# AWS DynamoDB



## Primary keys

- Most efficient way to retrieve data
- Determines the physical partition where data is stored

**What if you want query data using an attribute?**





# AWS DynamoDB



## Secondary Indexes

- o They allow for data in tables to be queried using an alternative “key” than the Primary Key
- o Allows for a range of query patterns

## Secondary Index Types

- o Local Secondary Index (LSI)
- o Global Secondary Index (GSI)





# AWS DynamoDB



## Local Secondary Index

- Index has the same partition key as the base table, but a different sort key.
- Created at the same time as the table and cannot be modified after table creation.
- Maximum of 5 Local Secondary Indexes per table

## Global Secondary Index

- An index with a partition key and sort key that can be different from the base table.
- Can be modified after table has been created
- Maximum of 20 GSI per table.
- If writes throttle, base table writes will throttle too





# AWS DynamoDB



ProductID (PK)	ProductName	Category	Price	Manufacturer	StockQuantity
B101	The Great Adventure	Books	9.99	YummyBooks	100
E102	Ultra HD TV 55''	Electronics	1200.00	ViewSonic	30
B103	Cooking Simplified	Books	20.00	TastyPress	50
E103	Gamer Pro Keyboard	Electronics	129.99	KeyCraft	75



**Partition Key**

*Sort Key*



**CategoryIndex (GSI)**

*Partition Key*

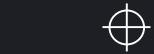


**ManufacturerIndex (GSI)**





# AWS DynamoDB



## How to create a Secondary Key?

- We use "projected attributes"
- Attributes that are copied from a table to an Index
- Attributes to copy can be specified
- Maximum 20 projected attributes per index

## Three options for projected attributes:

- **All**: All of the attributes from the base table are projected into the index.
- **Keys Only**: Only the key attributes from the base table are projected into the index.
- **Include**: Only specific attributes from the base table are projected into the index.





# AWS DynamoDB

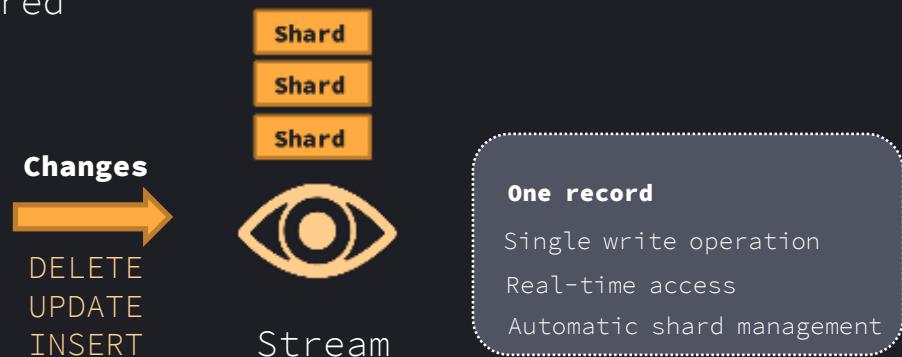


## DynamoDB Streams

- Captures changes done on data in DynamoDB tables  
[Writes/Deletes/Updates]
- Changes are sent to a stream record
- Changes are chronologically ordered

ProductID (PK)	ProductName	Category	Price	Manufacturer	StockQuantity
E101	The Great Adventure	Books	9.99	YummyBooks	100
E102	Ultra HD TV	Electronics	1299.99	ViewSonic	30
E103	Cooking Simplified	Books	29.99	TastyPress	50
E104	Gamer Pro Keyboard	Electronics	129.99	KeyCraft	75

Table





# AWS DynamoDB



## DynamoDB Streams

- Optional feature - Disabled by default
- Records are all changes made to an item during one operation
- You can decide what should be recorded to the stream

### Note:

- Changes made before activation of Stream are not recorded
- Data in the stream is retained for 24 hours





# AWS DynamoDB



- **Stream Record Options**

- **Keys\_only:** view only the key attributes of the item modified
- **New\_image:** view the item after changes were made
- **Old\_image:** view the item before changes were made
- **New\_and\_old\_image:** view the item before and after it was made





# AWS DynamoDB



## Processing Streams

1. AWS Lambda:
2. Amazon Kinesis Data Streams ⇒ Data Firehose
3. Amazon Elasticsearch Service
4. Custom Applications
5. AWS Glue
6. Cross-Region Replication





# AWS DynamoDB



## Lambda

1. Enable stream on table

ProductID (PK)	ProductName	Category	Price	Manufacturer	StockQuantity
P001	The Great Adventure	Books	9.99	WimpyBooks	100
P002	Ultimate HD 7000	Electronics	1299.99	Videomania	50
P003	Superior Single Headset	Books	20.00	TastyPrints	50
P004	Gamer Pro Keyboard	Electronics	129.99	NavyCrafts	50

Changes



2. Create a Lambda Function



3. Configure Trigger: Event Source Mapping ⇒ trigger for new records





# AWS DynamoDB



- **DynamoDB APIs**

- APIs are used to interact with DynamoDB programmatically
- They allow for the management of data in DynamoDB tables
- Applications are to use simple API Operations to interact with DynamoDB

## High Level Operations

- Control Plane
- Data Plane
- DynamoDB Streams APIs
- Transaction





# AWS DynamoDB



## High Level Operations

- **Control Plane**
- Data Plane
- DynamoDB Streams APIs
- Transaction



### Control Plane

These operations allow you to create and manage DynamoDB tables

- *CreateTable* – Creates a new table.
- *DescribeTable* – Returns information about a table.
- *ListTables* – Returns the names of all of your tables in a list.
- *UpdateTable* – Modifies the settings of a table or its indexes.
- *DeleteTable* – Removes a table and all of its dependent objects from DynamoDB.





# AWS DynamoDB



## High Level Operations

- Control Plane
- **Data Plane**
- DynamoDB Streams APIs
- Transaction



## Data Plane

These operations allow you to perform Create, Read, Update, and Delete on data in tables. (CRUD operations)

- I. PartiQL APIs
- II: DynamoDB's classic CRUD APIs





# AWS DynamoDB



## High Level Operations

- Control Plane
- **Data Plane**
- DynamoDB Streams APIs
- Transaction



### • Data Plane

PartiQL is a SQL-compatible query language for DynamoDB that can be used to perform CRUD operations

#### I. PartiQL APIs :

- *ExecuteStatement* – Reads multiple items from a table. You can also write or update a single item from a table.
- *BatchExecuteStatement* – Writes, updates, or reads multiple items from a table.





# AWS DynamoDB



## Data Plane

### High Level Operations

- Control Plane
- **Data Plane**
- DynamoDB Streams APIs
- Transaction



### II. DynamoDB's classic CRUD APIs:

The Primary key must be specified

#### 1. Creating data

- *PutItem* - Writes a single item to a table.
- *BatchWriteItem* - Writes up to 25 items to a table.





# AWS DynamoDB



## High Level Operations

- Control Plane
- **Data Plane**
- DynamoDB Streams APIs
- Transaction



## Data Plane

### II. DynamoDB's classic CRUD APIs:

The Primary key must be specified

#### 2. Reading Data

- *GetItem* - Retrieves a single item from a table.
- *BatchGetItem* - Retrieves up to 100 items from one or more tables.
- *Query* - Retrieves all items that have a specific partition key.
- *Scan* - Retrieves all items in the specified table or index.





# AWS DynamoDB



## High Level Operations

- Control Plane
- **Data Plane**
- DynamoDB Streams APIs
- Transaction



## Data Plane

### II. DynamoDB's classic CRUD APIs:

The Primary key must be specified

#### 3. Updating data

- `UpdateItem` – Modifies one or more attributes in an item.

#### 4. Deleting data

- `DeleteItem` – Deletes a single item from a table.
- `BatchWriteItem` – Deletes up to 25 items from one or more tables.





# AWS DynamoDB



## High Level Operations

- Control Plane
- Data Plane
- **DynamoDB Streams APIs**
- Transaction



## DynamoDB Streams

Operations for enabling/disabling a stream on a table and allowing access to the data modification records in a stream.

- *ListStreams* – Returns a list of all your streams, or a stream for a specific table.
- *DescribeStream* – Returns information about a stream
- *GetShardIterator* – Returns a shard iterator
- *GetRecords* – Retrieves one or more stream records, using a given shard iterator.





# AWS DynamoDB



## High Level Operations

- Control Plane
- Data Plane
- DynamoDB Streams APIs
- **Transaction**



## Transactions

Transactions provide Atomicity, Consistency, Isolation, and Durability (ACID), Can use PartiQL or Classic CRUD APIs

PartiQL APIs:

- *ExecuteTransaction* – A batch operation that allows CRUD operations on multiple items both within and across tables.





# AWS DynamoDB



## High Level Operations

- Control Plane
- Data Plane
- DynamoDB Streams APIs
- **Transaction**



## Transactions



DynamoDB's classic CRUD APIs:

- *TransactWriteItems* – A batch operation that allows Put, Update, and Delete operations on multiple items within and across tables.
- *TransactGetItems* – A batch operation that allows Get operations to retrieve multiple items from one or more tables.





# AWS DynamoDB



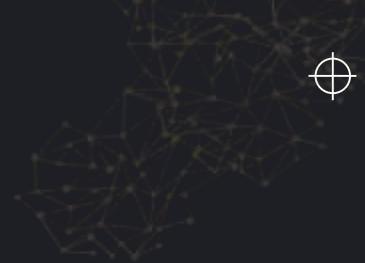
## Data Types

Data Type	Definition	Type
<i>Scalar Type</i>	Can represent exactly one value	Number, Boolean, String
<i>Document Type</i>	Can represent a complex structure with nested attributes	List, Map
<i>Set Type</i>	Can represent multiple scalar values.	String set, Number set, and Binary set.



# Cost & Configuration

## Read/Write Capacity Modes



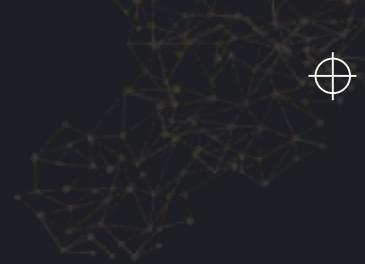
### On-Demand

- **Billing:** Pay per read/write operation without pre-specifying throughput.
- **Flexibility:** Ideal for unpredictable or spiky workloads.
- **Performance:** Maintains consistent, low-latency performance at any scale.  
    ⇒ Requests never throttle
- **Management:** AWS handles scaling, reducing operational overhead.
- **Cost Implications:** More expensive for predictable workloads (premium for scalability).



# Cost & Configuration

## Read/Write Capacity Modes



### Provisioned Mode

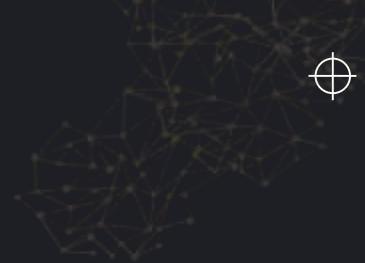
- **Throughput:** Specify expected reads/writes per second in RCUs and WCUs.
- **Cost Efficiency:** More economical for predictable workloads.
- **Auto Scaling:** Automatically adjusts throughput based on traffic, optimizing costs.
- **Throttling Risk:** Exceeding provisioned throughput can lead to throttling.
- **Management Overhead:** Requires monitoring and occasional manual adjustments.





# Cost & Configuration

## Read/Write Capacity Modes



**Provisioned Mode**

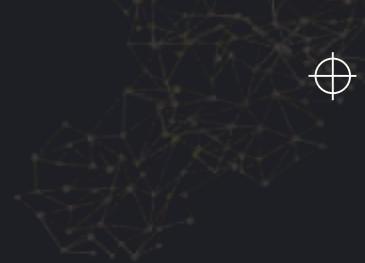
**Reserved Capacity**

- o **Long-Term Commitment:**  
Commit to specific RCU and WCU for 1 or 3 years.
- o **Discounted Pricing:**  
Reduced rates
- o **Use Case:**  
Suited for stable, predictable workloads over long periods.



# Cost & Configuration

## Read/Write Capacity Modes



**Provisioned Mode**

*Auto Scaling*

- **Dynamic Scaling:** Adjusts throughput based on utilization.
- **Cost Optimization:** Lowers costs by matching capacity to demand.
- **Setup Complexity:** Requires setting minimum, maximum, and target utilization levels.
- **Response Time:** Minor delays in scaling might occur, but it's designed to be responsive.



## Write Capacity Units (WCUs) & Read Capacity Units (RCUs)

### Write Capacity Unit

- **Definition:**  
Measures write throughput; 1 WCU equals 1 write per second for items up to 1 KB.
- **Consumption:**  
Items over 1 KB require more WCUs, e.g., a 3 KB item needs 3 WCUs per write.
- **Provisioning:**  
Allocate WCUs based on expected writes.  
Adjust with over-provisioning for spikes or use Auto Scaling for dynamism.
- **Cost:**  
Pay for provisioned WCUs, used or not. Planning is important to manage expenses.
- **Throttling:**  
Exceeding WCUs leads to throttled writes, potentially causing  
*ProvisionedThroughputExceededExceptions*.  
Opt for on-demand capacity for automatic scaling.





# AWS DynamoDB



## Write Capacity Units (WCUs) & Read Capacity Units (RCUs)

### ***Read Capacity Unit***

- **RCU Definition:** One strongly consistent read per second (or two eventually consistent reads) for items up to 4 KB.
- **Throughput Calculation:** Larger items consume more WCUs/RCUs.





# AWS DynamoDB



## Write Capacity Units (WCUs) & Read Capacity Units (RCUs)

### ***Read Consistency***

- **Eventually Consistent Reads:**  
Fast, high throughput, but data might be slightly outdated.
- **Strongly Consistent Reads:**  
Ensures the latest data view at a higher cost.
- **Application Needs:**  
Choose based on the criticality of data freshness vs. throughput requirements.





# AWS DynamoDB



## Write Capacity Units (WCUs) & Read Capacity Units (RCUs)

### ***Read Capacity Unit***

- **RCU Definition:** One strongly consistent read per second (or two eventually consistent reads) for items up to 4 KB.
- **Throughput Calculation:** Larger items consume more WCUs/RCUs.

#### **Example**

**Total Data:** 1 item, 8 KB

**Read Type:** Strongly Consistent

**Conclusion:** 2 RCUs to read an 8 KB item  
strongly consistently once per second.

#### **Example**

**Total Data:** 10 items, each 3 KB

**Read Type:** Eventually Consistent

**Conclusion:** 5 RCUs to perform eventually  
consistent reads of 10 items with 3 KB  
items per second





# AWS DynamoDB



## Performance Optimization

### ***Hot Partitions***

- Overloaded partitions due to uneven data distribution.
- Caused by poorly chosen partition keys.

### ***Throttling***

- Happens when requests exceed table/index throughput limits.
- Leads to higher latencies or request failures.





# AWS DynamoDB



## Performance Optimization

### ***Best Practices to Avoid Throttling***

- **Even Data Distribution:** Use well-designed partition keys.
- **Exponential Backoff:** Incorporate into retry logic.
- **Monitor Throughput:** Use CloudWatch, adjust settings as needed.
- **Use DAX:** Cache reads to reduce table load.





# AWS DynamoDB



## Performance Optimization

### **Burst Capacity**

- Allows handling short traffic spikes.
- Uses saved unused capacity for up to 5 minutes.

### **Adaptive Capacity**

- Automatically balances throughput across partitions.
- Maintains performance, but key design still matters.





# AWS DynamoDB



## Read Write Capacity Modes

### Reserved Capacity

- Purchase Reserved Capacity in advance for tables in standard table class.
- Once-off upfront fee with a specified minimum provisioned usage level over (x) amount of time
- Billed hourly at the rate of reserved capacity
- Unused reserved capacity is applied to accounts in the same AWS organization - can be turned off
- Regional - not available for standard IA table class/On-demand mode





# AWS DynamoDB



## Read/Write Capacity Modes

### DynamoDB Auto Scaling

- o Define maximum and minimum read/write capacity units as well as target utilization % within that range
- o DynamoDB maintains target utilizations and auto-scales to meet increases/decreases in traffic
- o Requests never throttle





# AWS DynamoDB



## DynamoDB Accelerator (DAX)

DynamoDB's **in-memory cache** that improves performance by 10x with **microseconds latency**.

To enable it: Create a **DAX Cluster**

- **DAX Cluster:** A DAX cluster has one or more nodes running on individual instances with one node as the primary node.
- **Accessing DAX:** Applications can access DAX through endpoints of the DAX cluster
- **“ThrottlingException”:** Returned if requests exceed the capacity of a node  
⇒ DAX limits the rate at which it accepts additional requests by returning a *ThrottlingException*.





# AWS DynamoDB



## DynamoDB Accelerator (DAX)

### Read Operations

*If an Item requested is in the cache, DAX returns it to the application without accessing DynamoDB(cache hit)*

*If item is not in the cache (cache miss), DAX forwards the request to DynamoDB.*

### API calls:

- *Batch*
- *GetItem*
- *Query*
- *Scan*





# AWS DynamoDB



## DynamoDB Accelerator (DAX)

### Write Operations

*Data is written to the DynamoDB table, and then to the DAX cluster.*

API calls:

- *BatchWriteItem*
- *UpdateItem*
- *DeleteItem*
- *PutItem*





# AWS DynamoDB



## DynamoDB Accelerator (DAX)

### Write Operations

*Data is written to the DynamoDB table, and then to the DAX cluster.*

API calls:

- *BatchWriteItem*
- *UpdateItem*
- *DeleteItem*
- *PutItem*





# AWS DynamoDB



## DynamoDB TTL

*DynamoDB Time To Live feature allows for the automatic deletion of items in a table by setting an expiration date.*

- Expired items are deleted 48 hours after the expiration date.
- TTL does not consume write throughput
- Expired Items pending deletion can still be updated.
- Expiration timestamp should be in Unix Epoch format
- Deleted items appear in DynamoDB Streams as Service Deletion





## Section 10:

# Redshift Datawarehouse



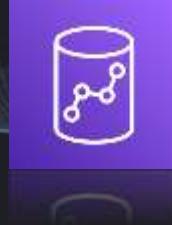
003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon Redshift

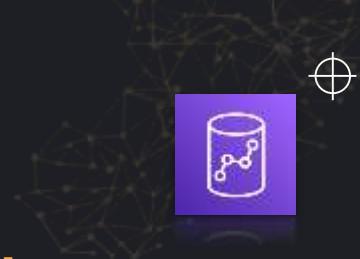


003-1040559 1250 003-77156.8 1760 0009-14563.7 73273





# Amazon Redshift

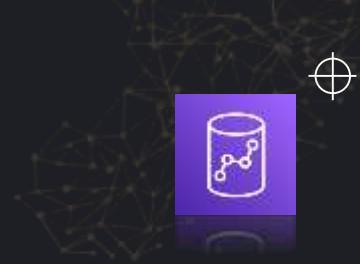


- Fully managed, highly available, cost-effective petabyte-scale data warehouse service in the cloud.
- ANSI SQL compatible relational database.
- Can be accessed using Amazon Redshift query editor v2 or any other Business Intelligence (BI) tool.





# Amazon Redshift



- Handles large amounts of data.
- Supports multiple data sources, including

CSV

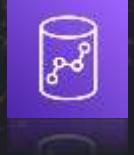
JSON

Apache Parquet





# Amazon Redshift



## Features

Scalable

Columnar Storage

Massively Parallel  
Processing (MPP)

Integration with Other AWS  
Services

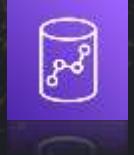
Advanced Compression

Security Features





# Amazon Redshift



## Use cases include

Data Warehousing

Business Intelligence

Analytics

Log Analysis

IoT Data Processing

Real-Time Dashboards





# Amazon Redshift Cluster

Part One



003-1040559

1250 003-77156.8

1760 0009-14563.7

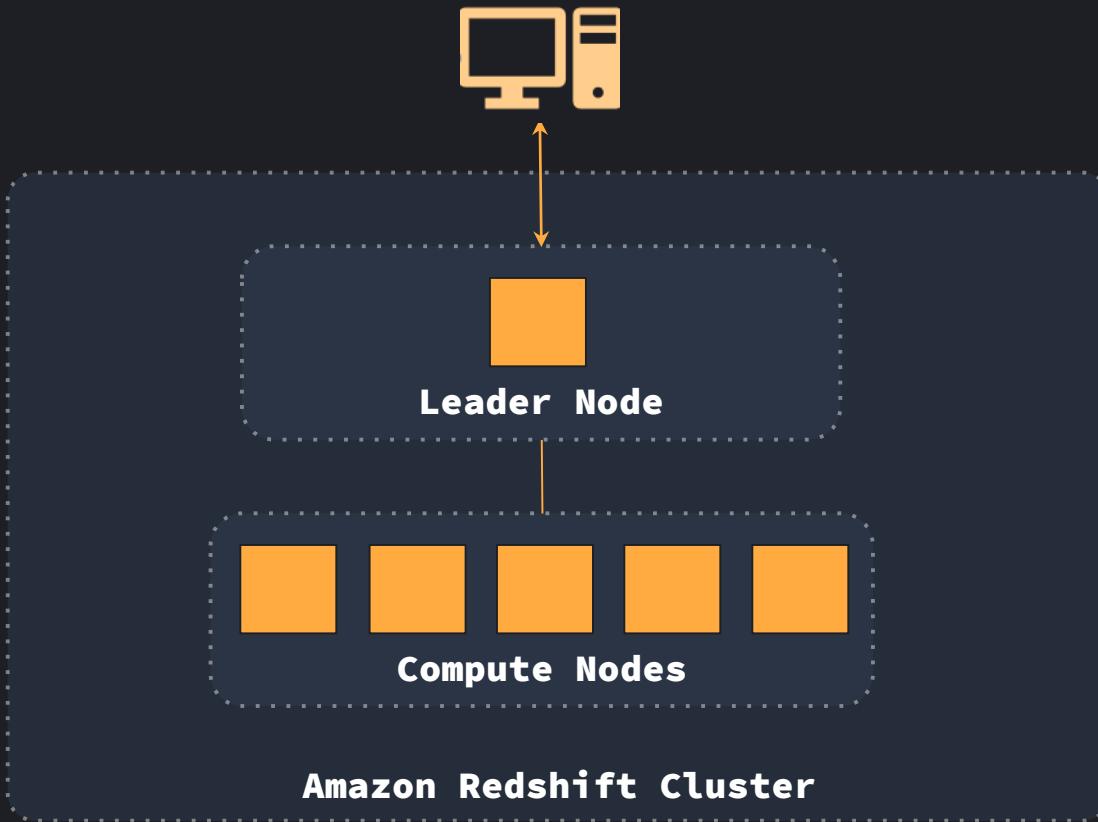
73273



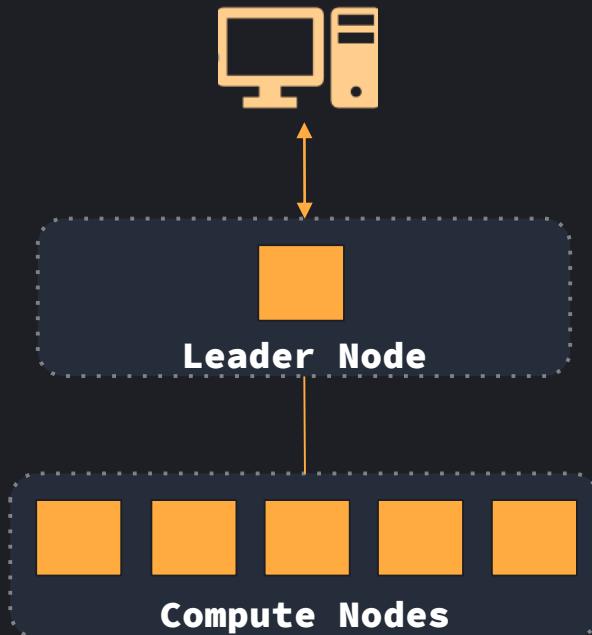
# Amazon Redshift Cluster

- Is the core infrastructure component of Redshift.
- Runs an Amazon Redshift engine and contains one or more databases.
- Executes workloads coming from external data apps.
- Uses replication and continuous backups.
- Automatically recovers from failures.

# Amazon Redshift Cluster



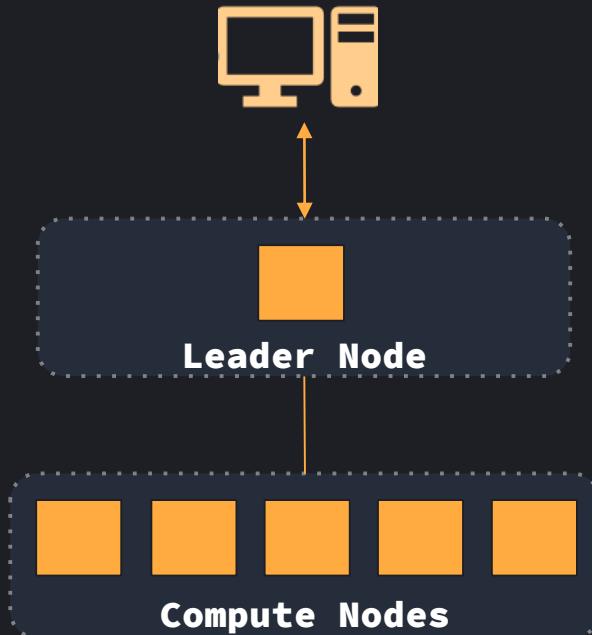
# Amazon Redshift Cluster



## Leader Node

- Coordinates two or more compute nodes.
- Aggregates results from compute nodes.
- Develops execution plan.
- Assigns a portion of the data to each compute node.

# Amazon Redshift Cluster



## Compute Nodes

- Has CPU, memory, and disk storage.
- Run the query execution plans.
- Transmit data among themselves.
- Capacity can be increased by increasing the number of nodes, upgrading the node type, or both.
- Use **provisioned cluster** or **Redshift Serverless**



# Redshift Managed Storage



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

# Redshift Managed Storage (RMS)

- Uses large, high-performance solid-state drives (SSDs) for fast local storage.
- Uses S3 for longer-term durable storage.
- Pricing is the same for RMS regardless of whether the data resides in high-performance SSDs or in S3.



# RA3 and DC2 Node types



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

# RA3 and DC2 Node types

## RA3 nodes

- Uses Redshift Managed Storage (RMS) .
- Separates **compute** and **storage** .
- Scale and pay for compute and managed storage independently.
- Supports multi Availability Zone (AZ)

# RA3 and DC2 Node types

## DC2 nodes

- Local SSD storage included.
- DC2 nodes store your data locally for high performance.
- Available on single Availability Zone (AZ) only.
- Recommended by AWS for datasets under 1TB (compressed).
- You can add more compute nodes to increase the storage capacity of the cluster.



# Amazon Redshift Cluster



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Resizing clusters



Used to change the node type or the number of nodes in the cluster

## **Node Slices:**

- Compute nodes are split into slices
- Each handling a part of the workload.
- Leader node distributes data & tasks to the slices for parallel processing

## **Elastic resize**

- Adjust number of nodes without disruption or restart  
(by **redistributing** data slices across nodes)
- Completes **quickly**.

Can be used to ...

- Add or remove nodes from your cluster.
- Change Node Type: From DS2 to RA3



# Amazon Redshift Cluster

## Classic resize

- Similar to elastic resize.
- Takes more time to complete
- Useful in cases where elastic resize is not applicable



# Amazon Redshift snapshots

003-1040559

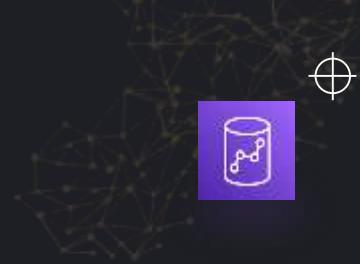
1250 003-77156.8

1760 0009-14563.7 73273





# Amazon Redshift snapshots



- Snapshots are point-in-time backups of a cluster.
- Snapshots are stored internally on S3.



- Snapshot can be taken manually or automatically.





# Amazon Redshift snapshot



## Automated Snapshot

- Incremental snapshots are taken automatically.
- Is enabled by default.
- By Default Snapshot is taken

Every Eight Hours

Every 5 GB per node of data changes

- Default retention period is one day.





# Amazon Redshift snapshots



## Manual Snapshots

- Can be taken any time.
- By default, manual snapshots are retained indefinitely.
- You can specify the retention period when you create a manual snapshot, or you can change the retention period by modifying the snapshot.





# Sharing data across AWS Regions



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Sharing data across AWS Regions



- You can share data across AWS Regions without the need to copy data manually.
- Sharing can be done without Amazon S3 as a medium.
- With cross-Region data sharing, you can share data across clusters in the same AWS account, or in different AWS accounts even when the clusters are in different Regions.





# Distribution Styles



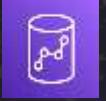
003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Distribution Styles

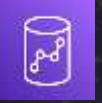


- Clusters store data across compute nodes.
- Use Distribution Keys for that distribution
- Distribution Styles determine where data is stored (compute node)
- Distribute the workload uniformly among the nodes in the cluster
- Minimize data movement during query execution.





# Distribution Styles



## KEY distribution

- Rows are distributed according to the values in one column.
- The leader node places matching values on the same node slice.
- Useful for tables that participate frequently in joins





# Distribution Styles



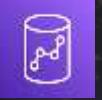
## ALL distribution

- A copy of the entire table is distributed to every node.
- Multiplies the storage required by the number of nodes in the cluster.
- Is appropriate only for relatively small and slowly changing tables.
- Faster query operations





# Distribution Styles



## EVEN distribution

- Leader node distributes the rows across the slices.  
(regardless of the values in any particular column)
- Appropriate when a table **doesn't participate in joins**.
- Appropriate when there **isn't a clear choice between KEY distribution and ALL distribution**.





# Distribution Styles

## AUTO distribution

- Redshift assigns an optimal distribution style based on the size of the table data.
- Is the default distribution Style
- How does it work



- The change in distribution style occurs in the background with minimal impact to user queries.





# Vacuum and Workload Management



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Vacuum



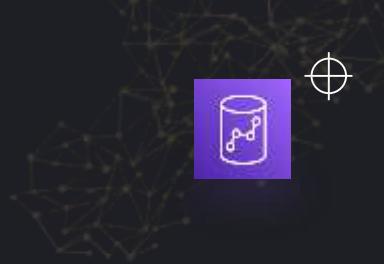
003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Vacuum



- It **Re-sorts rows** and **reclaims space** in either a specified table or all tables in the current database.
- Redshift automatically sorts data and runs `VACUUM DELETE` in the background.
- By default, `VACUUM` skips sorting for tables where  $> 95\%$  already sorted.
- Users can access tables while they are being vacuumed.
- Redshift automatically performs a `DELETE ONLY` vacuum in the background.





# Vacuum



## Command Syntax

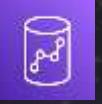
```
VACUUM [ FULL | SORT ONLY | DELETE ONLY | REINDEX | RECLUSTER ]  
[ [ table_name ] [ TO threshold PERCENT ] [ BOOST ] ]
```

- **FULL** : Sorts the specified table and reclaims disk space.
- **SORT ONLY** : Sorts the specified table (or all tables) without reclaiming space
- **DELETE ONLY** : Reclaims disk space only.
- **REINDEX** : Rebuilds the indexes on the tables.  
Useful for tables using Interleaved Sort Keys.
- **RECLUSTER** : Sorts the portions of the table that are unsorted.





# Vacuum



- **TO threshold PERCENT :**  
Specifies a threshold above which VACUUM skips sort phase & reclaiming space in delete phase.
- **BOOST :**  
Runs the VACUUM command with additional resources as they are available.





# Redshift Integration



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# AWS Redshift Integration



S3



EMR



Lambda



DMS



EC2



Data  
pipeline



MSK



KDS



DynamoDB



# Copy command



```
copy favoritemovies  
from 'dynamodb://Movies'  
iam_role'arn:aws:iam::0123456789012:role/MyRedshiftRole;
```

- Is used to load large amount of data from outside of redshift.
- Uses the Amazon Redshift the Massively Parallel Processing (MPP) .
- Uses optimal compression scheme.
- Supported compression include

Gzip

IZop

bzip





# Moving data between S3 and Redshift



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

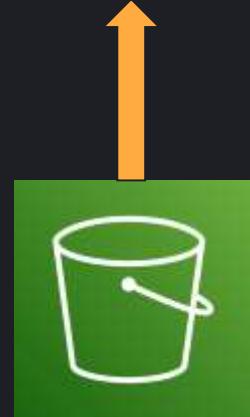




# Loading Data from S3 to Redshift



- You can add data to your Amazon Redshift tables either by using an `INSERT` or `COPY` command.
- loading data with `COPY`
  - Is faster.
  - Decrypts data as it is loaded from S3.
  - Requires a `manifest file` and `IAM role`.





# Unloading Data to S3 from Redshift

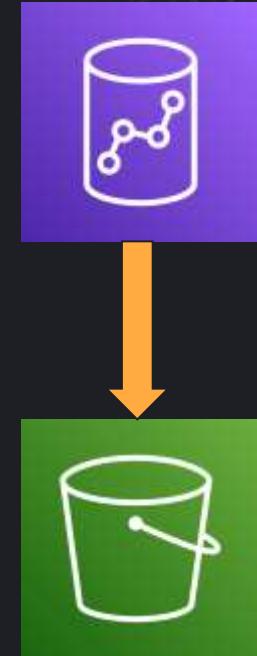


- The `UNLOAD` command allows you to export the results of a query to Amazon S3.
- It supports exporting data in various formats, including

CSV

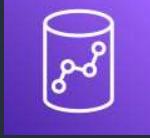
Apache Parquet

ORC



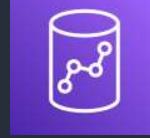


# Moving data between S3 and Redshift



## **Redshift Auto Copy from Amazon S3**

- Automatically copies data from an Amazon S3 bucket into an Amazon Redshift.



## **Enhanced VPC routing**

- Enables you to route the network traffic through a VPC instead of the internet.





# Amazon Aurora zero-ETL integration

- Allows changes made to the Amazon Aurora RDBMS database to be replicated in the Amazon Redshift database seconds after the Aurora updates.
- Eliminates the need for custom data pipelines
- Enables customers to analyze petabytes of transactional data in near real time.





# Data Transformation using Amazon Redshift



003-1040559

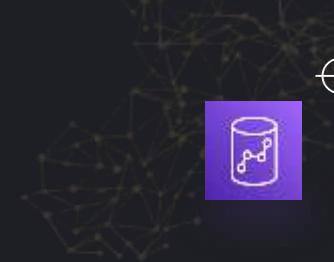
1250 003-77156.8

1760 0009-14563.7 73273





# Data Transformation using Amazon Redshift



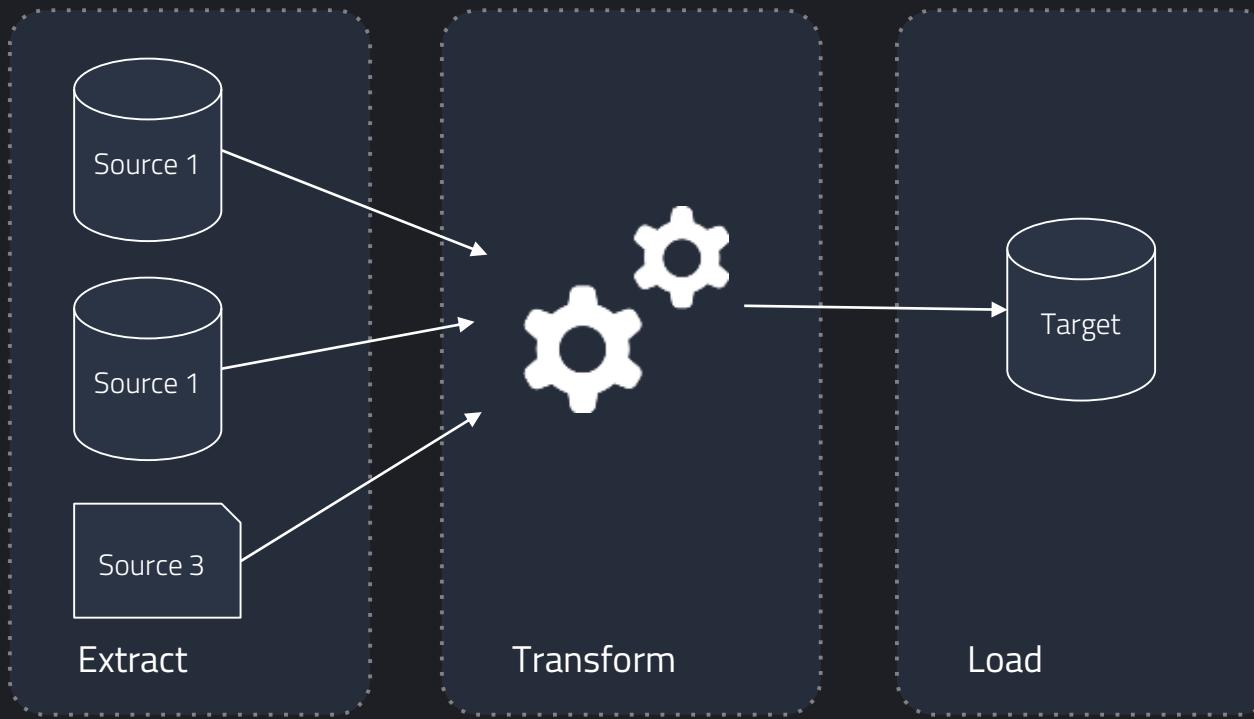
Design patterns when moving data from source systems to a data warehouse

- Extract Transform Load (ETL)
- Extract Load Transform (ELT)



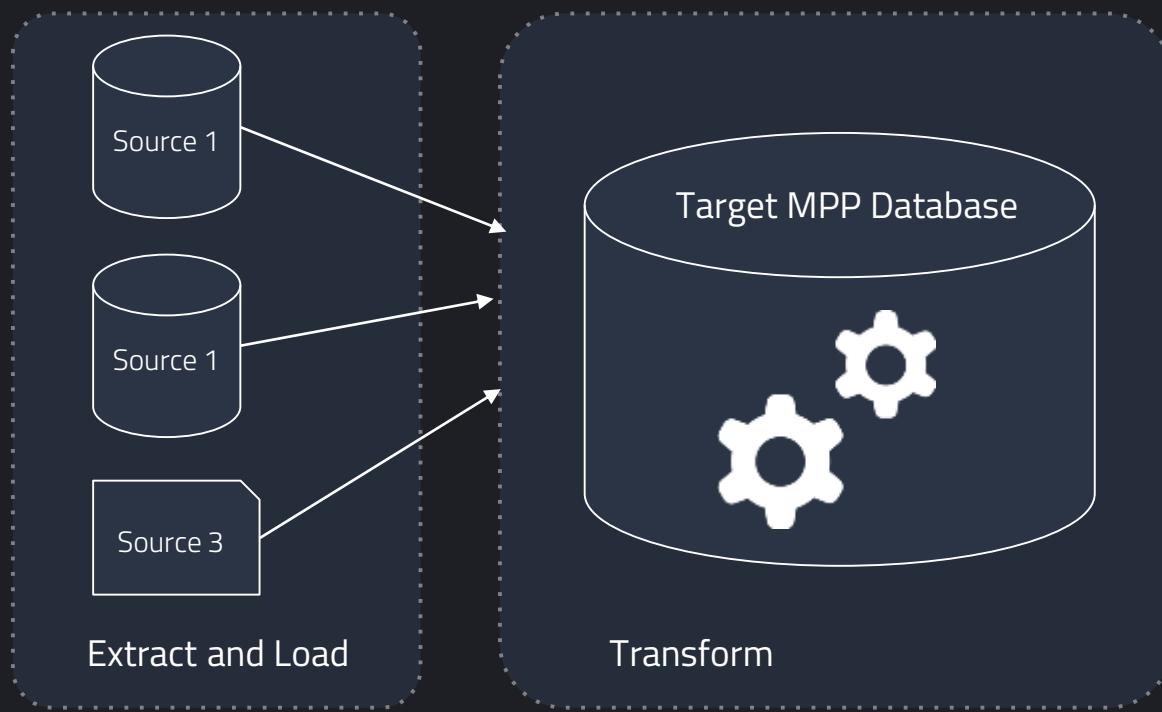
# Data Transformation using Amazon Redshift

Extract Transform Load (ETL)



# Data Transformation using Amazon Redshift

Extract Load Transform (ELT)





# Data Transformation using Amazon Redshift

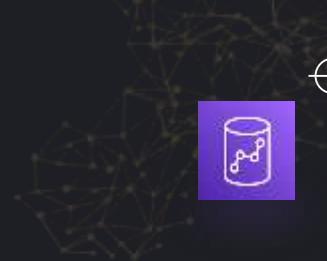


- Amazon Redshift provides a functionality to process all your data in one place with its in-database transformation (ELT) capabilities.
  - SQL Transformations
  - Stored Procedures
  - User-Defined Functions (UDFs)





# Data Transformation using Amazon Redshift



- Amazon Redshift can connect to an ETL platform of using JDBC and ODBC.
- Popular ETL platforms that integrate with Amazon Redshift include third-party tools like
  - Informatica
  - Matillion
  - dbt
  - AWS-native tools like “AWS Glue”.





# Amazon Redshift Federated Queries



003-1040559

1250 003-77156.8

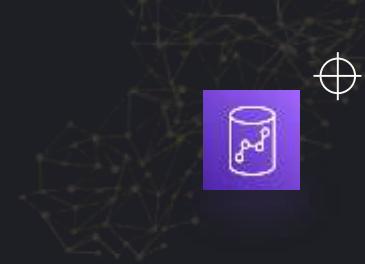
1760 0009-14563.7

73273





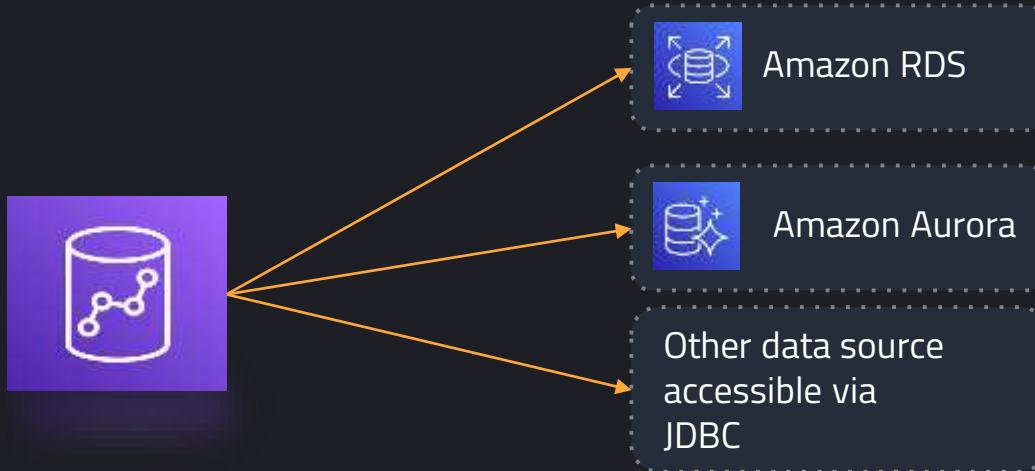
# Amazon Redshift Federated Queries



- Used to combine and analyze data across different data sources.
- Reduces data transmission while optimizing efficiency.
- Eliminates the necessity of ETL pipelines.

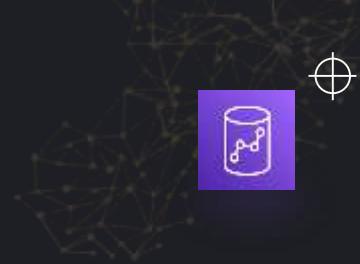


# Amazon Redshift Federated Queries





# Amazon Redshift Federated Queries



- Uses external schema definitions.
- Can be used to incorporate live data.
- Distributes part of the computation for federated queries directly into the remote operational databases.
- Uses parallel processing capacity.





# Materialized Views



003-1040559

1250 003-77156.8

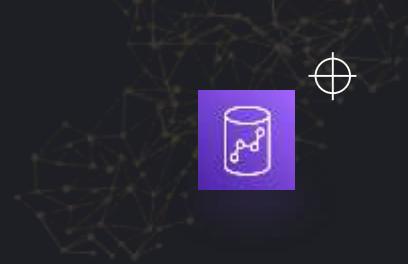
1760 0009-14563.7

73273





# Materialized Views

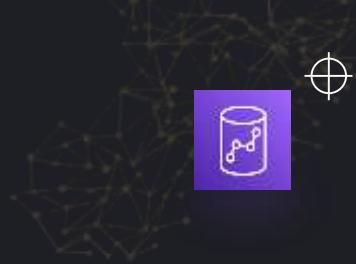


- Contains a precomputed result set.
- You can issue `SELECT` statements to query it.
- Redshift returns the precomputed results from the materialized view.





# Materialized Views



- Query results are returned much faster.
- Useful for speeding up queries that are **predictable** and **repeated**.
- You can define a materialized view in terms of other materialized views.

```
CREATE MATERIALIZED VIEW mv_name  
AUTO REFRESH { YES | NO }  
AS query
```



# Materialized Views

## Streaming Ingestion

- Ingest data from streaming sources

## Steps for creating a streaming ingestion

- Create an external schema that maps to the streaming data source.

```
CREATE EXTERNAL SCHEMA evdata FROM KINESIS  
IAM_ROLE 'arn:aws:iam::0123456789:role/redshift-streaming-role';
```

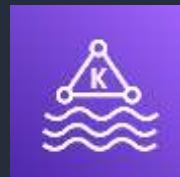
- Create a materialized view to consume the data

```
CREATE MATERIALIZED VIEW view_name  
AUTO REFRESH YES  
AS SELECT ... ;
```

Turn on Auto-refresh



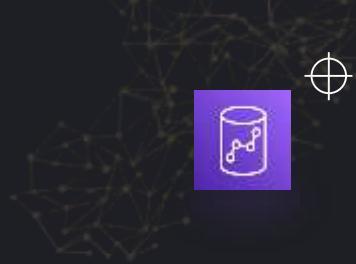
Kinesis  
Data Streams



Managed Streaming  
for Apache Kafka



# Materialized Views



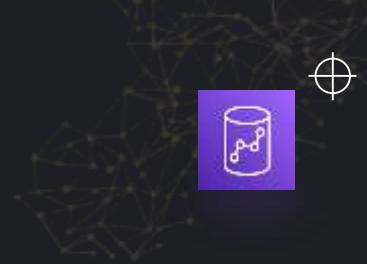
## Updating materialized view

- Automatic refresh option to refresh materialized views when base tables of materialized views are updated.
- Auto refresh operation runs at a time when cluster resources are available to minimize disruptions to other workloads.
- Specify `AUTO REFRESH` in the materialized view definition to enable it.





# Materialized Views

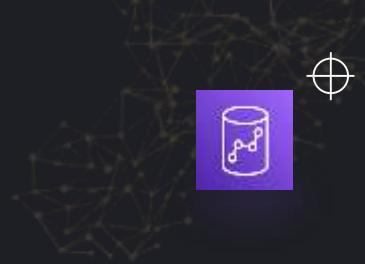


- Scheduler API and console integration.
- You can also manually refresh any materialized views using the `REFRESH MATERIALIZED VIEW` statement.





# Materialized Views



**Commands used to create and manage materialized views**

- CREATE MATERIALIZED VIEW
- ALTER MATERIALIZED VIEW
- REFRESH MATERIALIZED VIEW
- DROP MATERIALIZED VIEW





# Amazon Redshift Spectrum



003-1040559

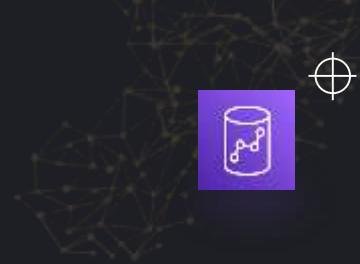
1250 003-77156.8

1760 0009-14563.7 73273





# Amazon Redshift Spectrum



- Enables you to run complex SQL queries directly against data stored in Amazon S3.



- You define external tables in Redshift cluster that reference the data files stored in S3.

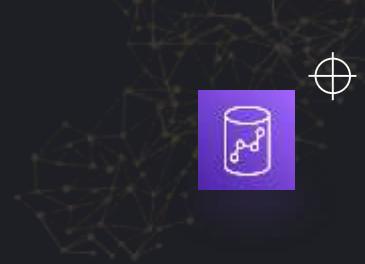


# Amazon Redshift Spectrum

- Handles the execution of queries by retrieving only the necessary data from S3.
- Supports different data formats, Gzip and Snappy compression.
- Resides on dedicated Amazon Redshift servers.
- Pushes many compute-intensive tasks down to the Redshift Spectrum layer.



# Amazon Redshift Spectrum



- Scales intelligently.
- Redshift Spectrum tables can be created by defining the structure for your files and registering them as tables in an external data catalog.
- The external data catalog can be



AWS Glue



Amazon  
Athena



Apache Hive metastore

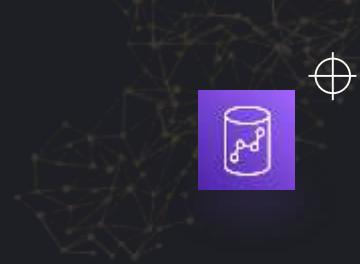


# Amazon Redshift Spectrum

- Changes to the external data catalog are immediately available.
- You can partition the external tables on one or more columns.
- Redshift Spectrum tables can be queried and joined just as any other Amazon Redshift table.



# Amazon Redshift Spectrum



## Amazon Redshift Spectrum considerations

- Redshift cluster and the S3 bucket must be in the same AWS Region.
- Redshift Spectrum doesn't support enhanced VPC routing with provisioned clusters.
- Supports Amazon S3 access point aliases.
- Redshift Spectrum doesn't support VPC with Amazon S3 access point aliases



# Amazon Redshift Spectrum

## Amazon Redshift Spectrum considerations (continued)

- You can't perform update or delete operations on external tables.
- To create a new external table in the specified schema, you can use `CREATE EXTERNAL TABLE`.
- To insert the results of a `SELECT` query into existing external tables on external catalogs, you can use `INSERT` (external table).
- Unless you are using an AWS Glue Data Catalog that is enabled for AWS Lake Formation, you can't control user permissions on an external table.

# Amazon Redshift Spectrum

## Amazon Redshift Spectrum considerations (continued)

- To run Redshift Spectrum queries, the database user must have permission to create temporary tables in the database.
- Redshift Spectrum doesn't support Amazon EMR with Kerberos.



# System Tables and Views



# Redshift System Tables and Views



## System Table

- Contain information about how the system is functioning.



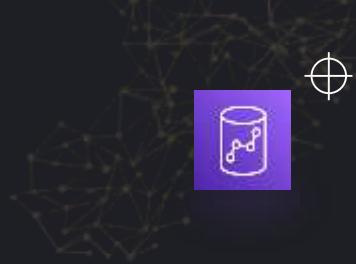
## System View

- Views to organize information from the system tables.





# Redshift System Tables and Views



- Queried the same way like any other database tables.
- Some system tables can only be used by AWS staff for diagnostic purposes.





# Redshift System Tables and Views



## Types of system tables and views

- SVV views  
Details on database objects;  
**SVV\_ALL\_TABLES** see all tables  
**SVV\_ALL\_COLUMNS** see a union of columns
- SYS views  
Monitor query and workload performance in clusters;  
**SYS\_QUERY\_HISTORY** see details of user queries
- STL views  
Generated from system logs for historical records;  
**STL\_ALERT\_EVENT\_LOG** identify opportunities to improve query performance  
**STL\_VACUUM** statistics for tables that have been vacuumed
- STV tables  
Snapshots of the current system data;  
**STV\_EXEC\_STATE** information about queries & query steps actively running





# Redshift System Tables and Views



## Types of system tables and views

- SVCS views
- SVL views

Details about queries on both the main and concurrency scaling clusters;  
**SVCS\_QUERY\_SUMMARY** general information about the execution of a query

Contain references to STL tables & logs for more detailed information;  
**SVL\_USER\_INFO** data about Amazon Redshift database users





# Redshift Data API



003-1040559

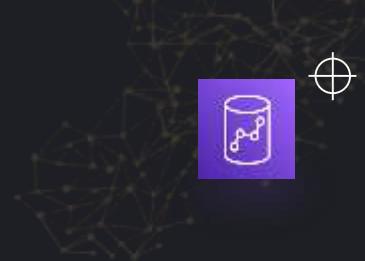
1250 003-77156.8

1760 0009-14563.7

73273



# Redshift Data API



- It is a **lightweight**, **HTTPS-based API** that is used for running queries against Amazon Redshift.
  - Lambda
  - SageMaker notebooks
  - Other web-based applications

Serverless, event-driven architecture

Web applications

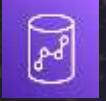
Third-party services

- Can be used to run SQL queries **asynchronously**.
- Can be used as an **alternative** to using JDBC or ODBC drivers.
- No need to manage and maintain WebSocket or JDBC connections.
- Can be setup with very little operational overhead (**very easy setup!**)





# Redshift Data API



Serverless Execution

Asynchronous Processing

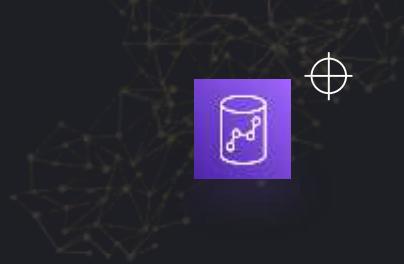
Direct Integration with AWS Services (e.g. Lambda / SageMaker)

Simplified Management





# Redshift Data API

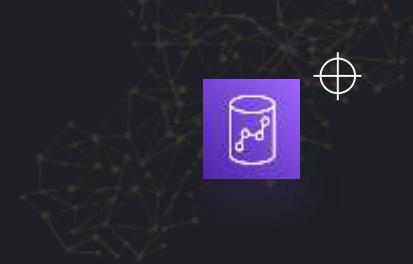


- Maximum duration of a query is 24 hours.
- Maximum number of active queries per Amazon Redshift cluster is 200.
- Maximum query result size (after compression) is 100 MB.
- Maximum retention time for query results is 24 hours.





# Redshift Data API



- Maximum query statement size is 100 KB.
- Is available to query single-node and multiple-node clusters of the following node types:

dc2.large

dc2.8xlarge

ra3.xlplus

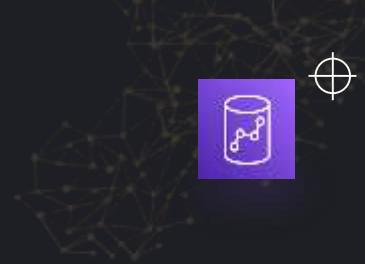
ra3.4xlarge

ra3.16xlarge





# Access Control



- Authorize user/service by adding managed policy  
⇒ *AmazonRedshiftDataFullAccess*

## Monitoring Data API

- Monitoring of Data API events in EventBridge
- EventBridge routes data to targets such as AWS Lambda or SNS.
- Option to *schedule* Data API operations within EventBridge





# Data Sharing



003-1040559

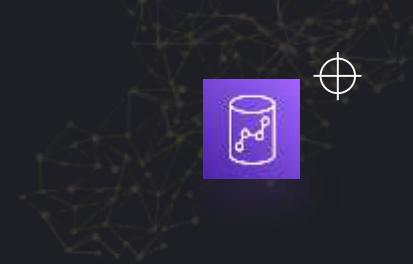
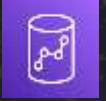
1250 003-77156.8

1760 0009-14563.7 73273





# Data Sharing



- Used to securely share access to live data across

Clusters

Workgroups

AWS accounts

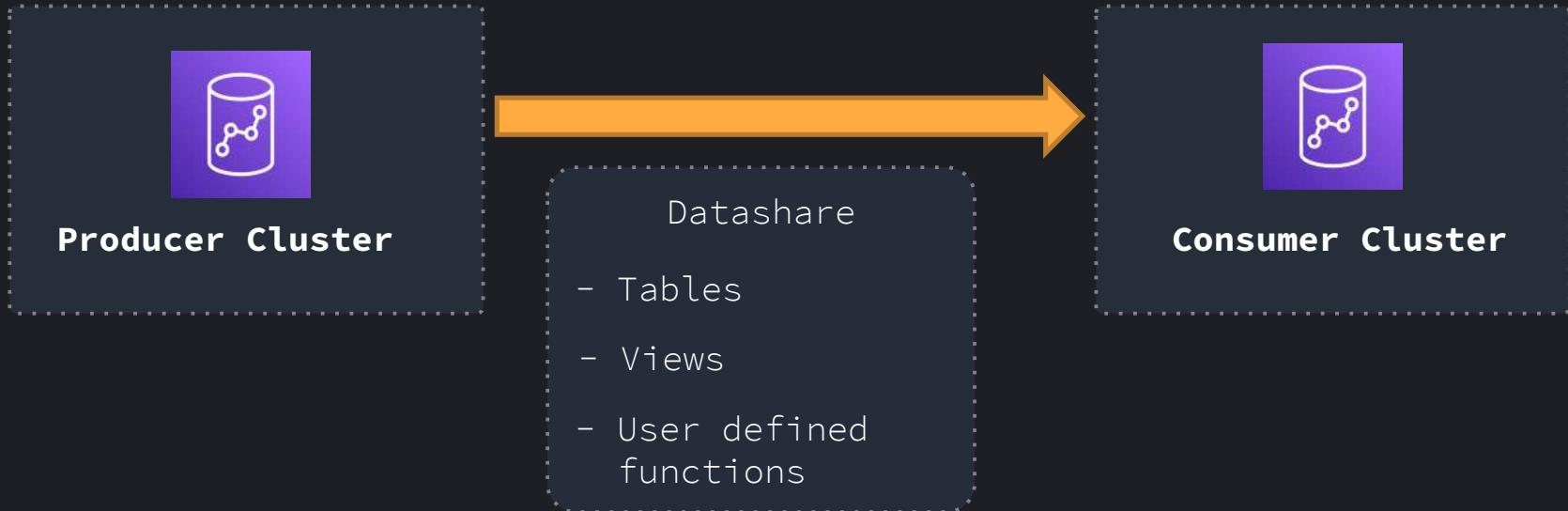
AWS Regions

Availability Zones

- Data is live



# Data Sharing



# Data Sharing



**Standard Datashares**



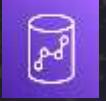
**AWS Data Exchange  
datashares**



**AWS Lake Formation-  
managed datashares**



# Data Sharing

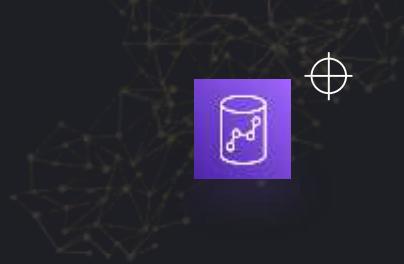
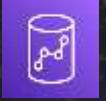


- The consumer is charged for all compute and cross-region data transfer fees
- The producer is charged for the underlying storage of data
- The performance of the queries on shared data depends on the compute capacity of the consumer clusters.
- Data sharing continues to work when clusters are resized or when the producer cluster is paused.





# Data Sharing



- Data sharing is supported for all provisioned `ra3` cluster types and Amazon Redshift Serverless.
- For cross-account and cross-Region data sharing, both the producer and consumer clusters and serverless namespaces must be encrypted.
- You can only share SQL UDFs through datashares. Python and Lambda UDFs aren't supported.





# Data Sharing



- Adding external schemas, tables, or late-binding views on external tables to datashares is not supported.
- Consumers can't add datashare objects to another datashare.





# Workload Management (WLM)





# Workload Management (WLM)

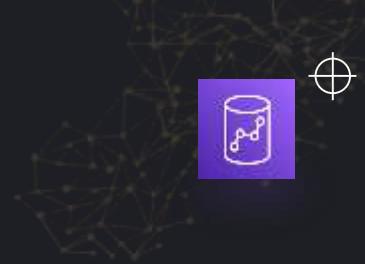


- Helps you manage query performance for competing queries
- Define how resources are allocated to different queries.
- Uses query queues  
⇒ Route queries to the appropriate queues
- Enables users to flexibly manage priorities within workloads.
- You can create up to 8 queues.





# Workload Management (WLM)



## Automatic Work Load Management

- Redshift manages:
  - How many queries run concurrently
  - How much memory is allocated to each dispatched query.
- When complex queries  $\Rightarrow$  lower concurrency
- When lighter queries  $\Rightarrow$  higher concurrency





# Workload Management (WLM)

## Automatic Work Load Management

- You can configure the following for **each query queue**:

Queue

Configurable with different priorities and rules

User groups

assign a set of user groups to a queue

Query groups

assign a set of query groups to a queue

Priority

relative importance of queries

Concurrency scaling mode

automatically adds additional cluster capacity





# Workload Management (WLM)



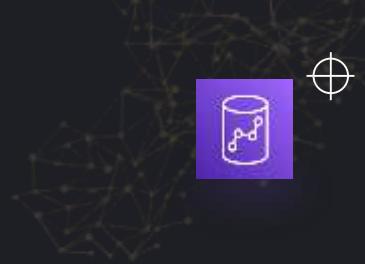
## Manual Work Load Management

- Manage system performance by modifying WLM configuration.
- You can configure the amount of memory allocated to each queue.
  - ⇒ Automatic usually higher throughput
  - ⇒ Manual offers more control





# Workload Management (WLM)



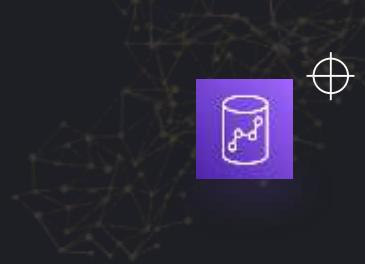
## **Short query acceleration (SQA)**

- Prioritizes selected short-running queries over longer-running queries.
- Uses machine learning to predict the execution time of a query.
- Runs short-running queries in a dedicated space.





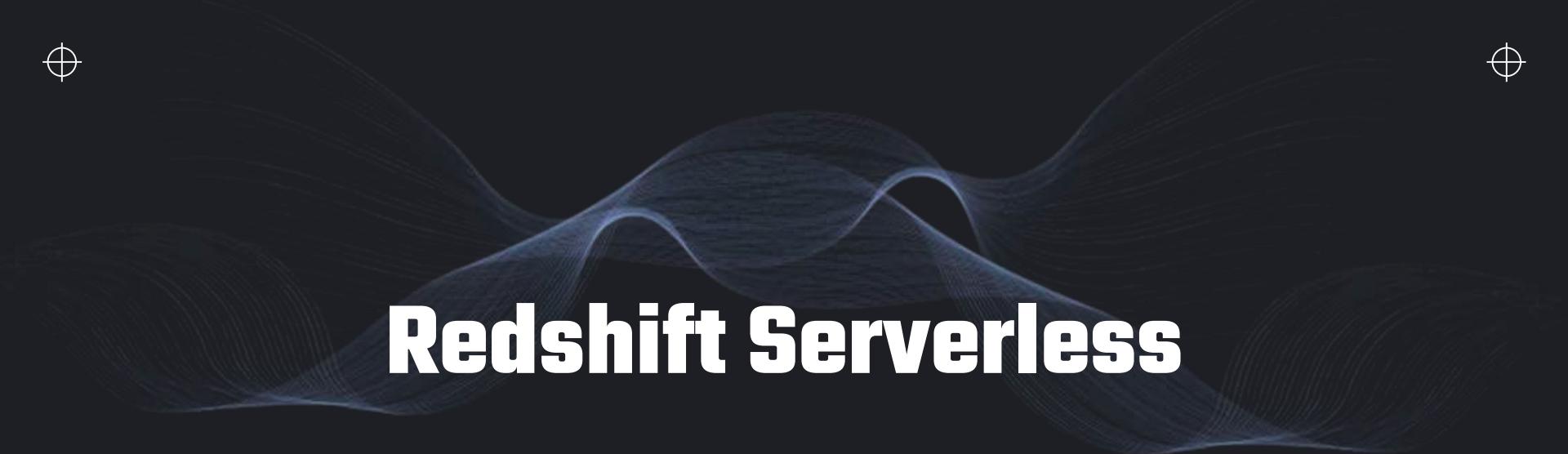
# Workload Management (WLM)



## Short query acceleration (SQA)

- `CREATE TABLE AS` (CTAS) statements and read-only queries, such as `SELECT` statements, are eligible for SQA.





# Redshift Serverless



003-1040559

1250 003-77156.8

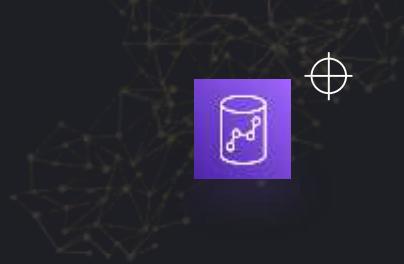
1760 0009-14563.7

73273





# Redshift Serverless



- No Cluster Management.
- On demand scaling.
- Pay as you go.



# Redshift Serverless

## Node

- Automatically provisions and manages capacity for you.

VS

## Workload management and concurrency scaling

- Automatically manages resources efficiently and scales, based on workloads, within the thresholds of cost controls.

# Redshift provisioned



## Node

- You build a cluster with node types that meet your cost and performance specifications

## Workload management and concurrency scaling

- You enable concurrency scaling on your cluster to handle periods of heavy load.



# Redshift Serverless

## Port

- You can choose port from the port range of 5431–5455 or 8191–8215.

## Resizing

- Not Applicable

## Encryption

- Always encrypted with AWS KMS, with AWS managed or customer managed keys.

# Redshift provisioned

## Port

- You can choose any port to connect

## Resizing

- Cluster resize can be done to add nodes or remove nodes

## Encryption

- Can be encrypted with AWS KMS (with AWS managed or customer managed keys), or unencrypted.

VS





# Redshift ML



003-1040559

1250 003-77156.8

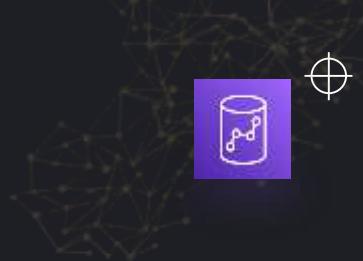
1760 0009-14563.7

73273





# Redshift ML

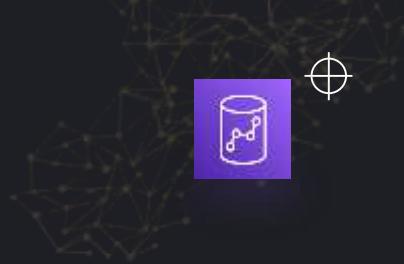


- Allows users to **create**, **train**, and **apply** machine learning models using just **SQL**.
- Users can utilize **SQL commands** to create and manage machine learning models, which are then trained using data stored in Redshift.
- Amazon Redshift ML enables you to train models with one single SQL **CREATE MODEL** command.





# Redshift ML

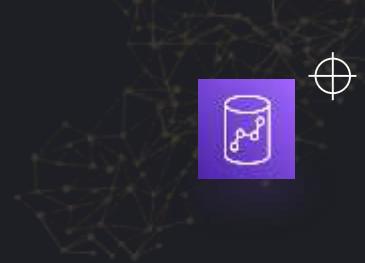


- Lets users create predictions without the need to move data out of Amazon Redshift.
- Users can create, train, and deploy machine learning models directly within the Redshift environment.
- Useful for users that doesn't have expertise in machine learning, tools, languages, algorithms, and APIs.





# Redshift ML



- Redshift ML supports common machine learning algorithms and tasks, such as

Binary classification

Multiclass classification

Regression

- It Automatically finds the best model using Amazon SageMaker Autopilot.
- Can use the massively parallel processing capabilities of Amazon Redshift.





# Security in Amazon Redshift



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Amazon Redshift Spectrum

Amazon Redshift provides these features to manage security

- Sign-in credentials
- Access management using IAM
- Virtual Private Cloud (VPC)
- Cluster encryption





# Redshift Security

- **Cluster security groups**

- **Default Lockdown:**

When you provision an Amazon Redshift cluster, it is locked down by default so nobody has access to it.

- **Inbound Access Control:**

To grant other users inbound access to an Amazon Redshift cluster, you associate the cluster with a security group.

- **SSL connections** to secure data in transit

- **Load data encryption** to encrypt the data during the load process



# Amazon Redshift Spectrum

- Data in transit
- Column-level access control
- Row-level security control





# Access Control In Redshift



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Access Control In Redshift



- You can either manage your users and groups within
  - Redshift
  - AWS IAM users
- The privileges to access specific objects are tightly coupled with the DB engine itself.

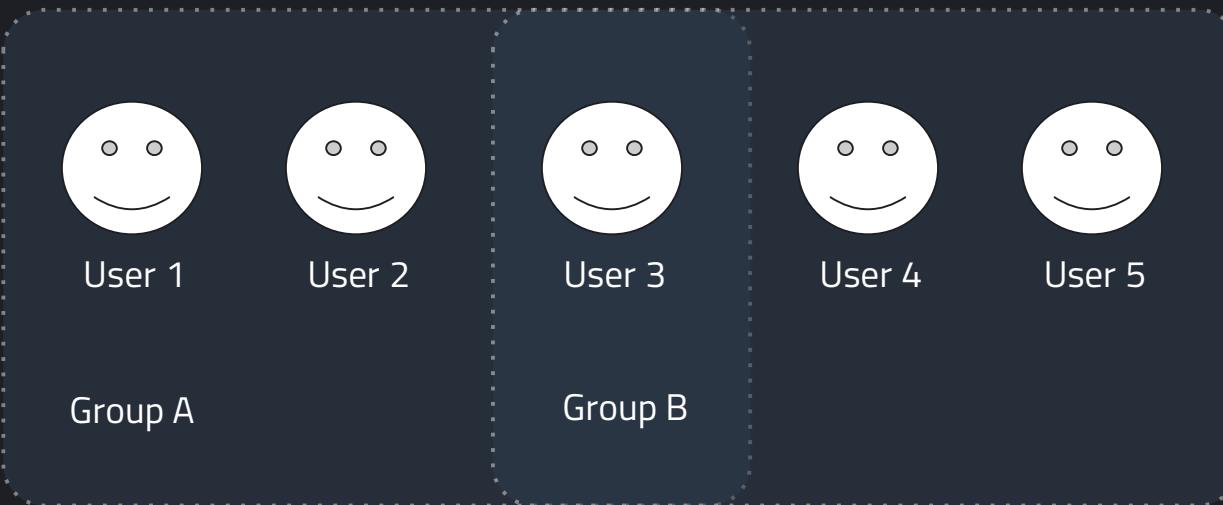




# Access Control In Redshift



Manage your users and groups within Redshift





# Access Control In Redshift



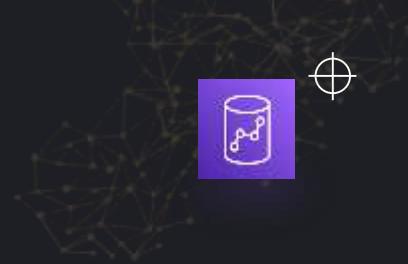
```
CREATE USER human_resource_1 PASSWORD 'xyzzy-1-XYZZY';
ALTER GROUP human_resource ADD USER human_resource_1;
```

```
CREATE USER operation_1 PASSWORD 'xyzzy-1-XYZZY';
ALTER GROUP operation ADD USER operation_1;
```





# Access Control In Redshift



```
/* Creating the 2 groups */
```

```
CREATE GROUP humanresource;  
CREATE GROUP operation;
```

```
/* Creating the 2 schemas with the data */
```

```
CREATE SCHEMA humanresource;  
CREATE SCHEMA operation;
```

```
/* Give sales USAGE rights in schema, and read-only (SELECT) access to the  
tables within the schema */
```

```
GRANT USAGE on SCHEMA humanresource TO GROUP humanresource;  
GRANT SELECT ON ALL TABLES IN SCHEMA operation TO GROUP operation;
```

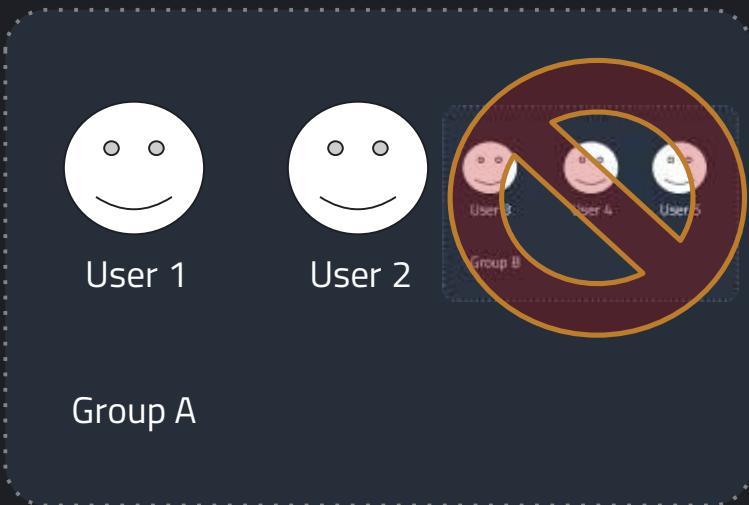




# Access Control In Redshift

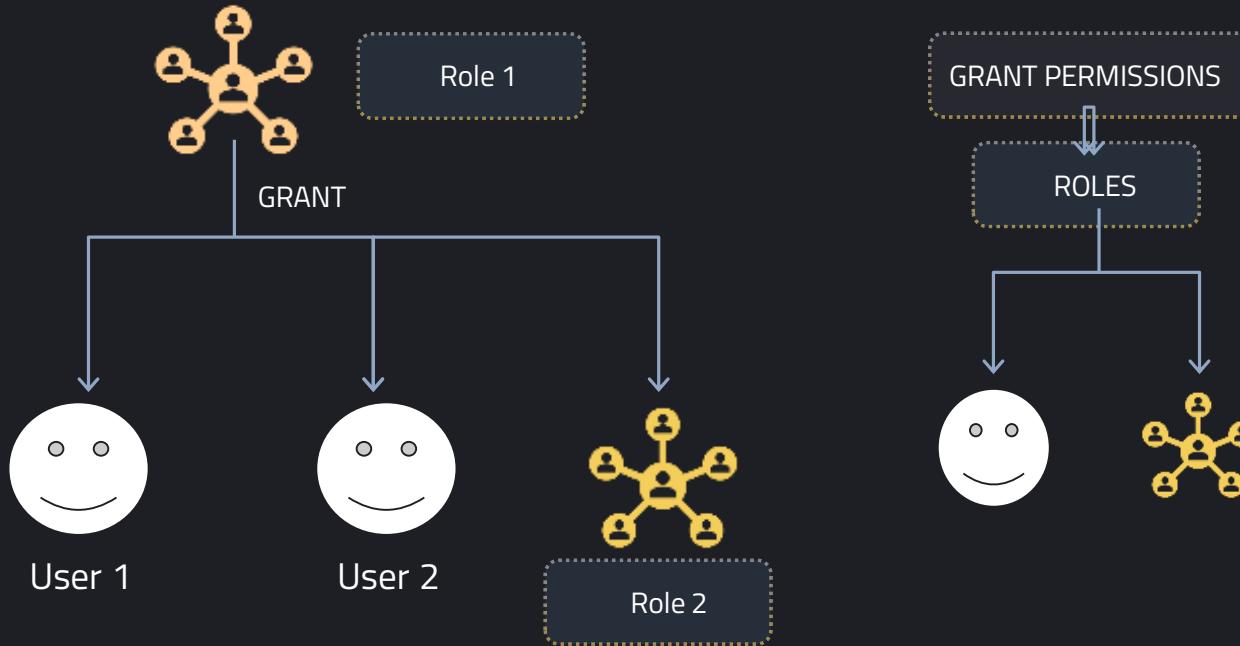


A group **CANNOT** contain another group



# Access Control In Redshift

AWS introduced RBAC (Role-based access control)





# Access Control In Redshift



```
/* Creating role*/
```

```
CREATE ROLE humanresource;
```

```
/* Grant role to user*/
```

```
GRANT ROLE humanresource TO user_1;
```

```
/* Give sales USAGE rights in schema, and read-only (SELECT) access to the  
tables within the schema */
```

```
GRANT USAGE on SCHEMA humanresource TO ROLE humanresource;
```

```
GRANT SELECT ON ALL TABLES IN SCHEMA operation TO ROLE finance;
```

```
GRANT ROLE finance TO ROLE humanresource;
```





# Redshift Fine-grained access control

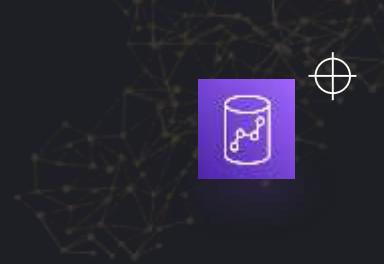
## Fine-grained Redshift access control

- Used to control and manage access permissions for various users or roles.
- Contains detailed access policies.





# Fine-grained access control



```
/* Column-level access control*/  
  
GRANT SELECT (column1, column2) ON tablename TO username;
```





# Fine-grained access control

```
/* Row-level access control */

CREATE RLS POLICY view_own_warehouse_inventory
  WITH (warehouse_id INTEGER)
  USING (
    warehouse_id IN (
      SELECT managed_warehouse_id
      FROM warehouse_managers
      WHERE manager_username = current_user
    )
  );

```

item_id	item_name	quantity	warehouse_id
101	LED Light Bulb	120	1
102	Electric Drill	85	2
103	Hammer	75	1
104	Nails (Pack of 100)	150	3

manager_id	manager_username	managed_warehouse_id
1	jsmith	1
2	mbrown	2
3	lwilson	3





# Fine-grained access control



```
/* Masking Policy */

CREATE MASKING POLICY mask_email
WITH (email VARCHAR(256))
USING ('***'::TEXT);
```

manager_id	manager_username	email
1	jsmith	***
2	mbrown	***
3	lwilson	***





# Fine-grained access control



```
/* Create Masking Policy*/
```

```
CREATE MASKING POLICY mask_email  
WITH (email VARCHAR(256))  
USING ('***'::TEXT);
```

```
/* Attach Masking Policy*/
```

```
ATTACH MASKING POLICY mask_email  
ON employee_data(email)  
TO ROLE role_hr  
PRIORITY 20;
```

manager_id	manager_username	email
1	jsmith	***
2	mbrown	***
3	lwilson	***

```
/* Detach Masking Policy*/
```

```
DETACH MASKING POLICY mask_email  
ON employee_data(email)  
FROM ROLE role_hr;
```





# Redshift Fine-grained access control

## Row level security in Redshift

- Users can only access specific rows.
- Rows have criteria that defines which role can access the specific item (row).

## Access logging & monitoring in Redshift

- Failed and successful access attempts to Redshift data warehouses can be logged using the system table `STL_CONNECTION_LOG`.
- Audit logs are not enabled by default.





# Section 11: **Other Database Services**





# Amazon RDS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS RDS



## What is RDS?

- *Fully managed relational database service.*

## Main characteristics:

- Scalable, reliable, and cost-effective.
- Fully encrypted at rest and in transit.
- Support for Multiple Database Engines:
  - *MySQL*
  - *PostgreSQL*
  - *MariaDB*
  - *Oracle Database*
  - *SQL Server*
  - *Amazon Aurora*





# Security in RDS



## 1) **AWS Key Management Service (AWS KMS):**

Automatic integration for key management and envelope encryption.

## 2) **Backup and Disaster Recovery:**

Automated and manual backups, restoring from a snapshot.

## 3) **Patch Management:**

Automatic minor version upgrades for minor updates and patches

## 4) **AWS backup:**

Centralization and automation of data backup.





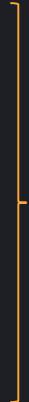
# ACID compliance in RDS

*Important set of properties for data reliability.*



## Definition:

- o **A**tomicity
- o **C**onsistency
- o **I**solation
- o **D**urability

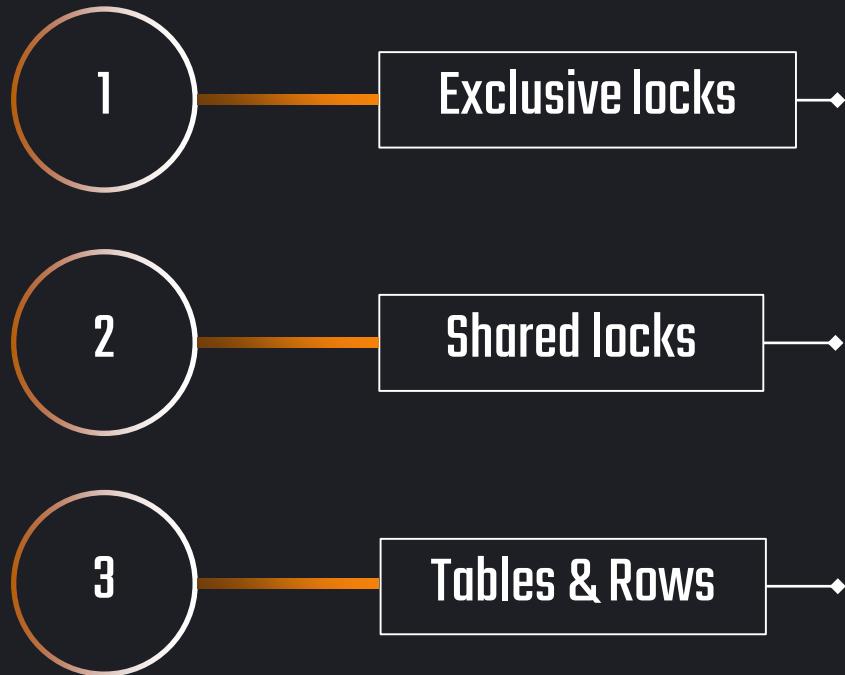


**Data integrity and consistency**



# ACID compliance in RDS

*Locking mechanisms for concurrent data access and modification in multiple transactions.*



Prevent other transactions from reading or writing to the same data.

**Syntax: *FOR SHARE*.**

Allow multiple transactions to read data at the same time without blocking.

**Syntax: *FOR UPDATE*.**

These can be locked to keep data integrity and control.



# Syntax and Deadlocks



- **Postgre SQL command to lock a table:**

```
LOCK TABLE table_name IN ACCESS EXCLUSIVE MODE;
```

- **PostgreSQL command to acquire a shared lock:**

```
SELECT * FROM table_name FOR SHARE;
```

- **PostgreSQL command to acquire an exclusive lock:**

```
SELECT * FROM table_name FOR UPDATE;
```

- **Understanding Deadlocks**

## What Happens:

During concurrent transactions; lock resources and wait on each other.

## Result:

No transaction can proceed, halting progress.





# AWS RDS Basic Operational Guidelines.



- Monitor metrics for memory, CPU, replica lag, and storage via CloudWatch.
- Scale database instances to manage storage capacity efficiently.
- Enable automatic backups during periods of low write IOPS.
- Provision sufficient I/O capacity for your database workload.
- Set a Time-To-Live (TTL) value under 30 seconds.
- Conduct failover testing regularly.





# Best Practices for DB engines in RDS



- 1) **Allocate enough RAM**, so that your *working set* resides almost completely in memory
- 2) **Check ReadOPS metric constantly**, the value should be small and stable.
- 3) **Use Enhanced Monitoring** to obtain real time metrics for the OS in your DB instances
- 4) **Use Performance Insights** for RDS and Aurora
  - o Simplifies database performance monitoring and tuning
  - o Easy-to use dashboard

## **Example use cases:**

- Detect performance issues
- Evaluate impact of SQL queries and optimize them (Dev & Test)
- Assess and tune performance during transitioning to the cloud





# Best practices for different engines hosted in RDS



- **MySQL:** Ensure tables don't exceed the 16TiB size limit by `partitioning` large tables.
- **Oracle FlashGrid Cluster:** Utilize `virtual appliances` to run self-managed RAC and RAC extended clusters across AZs on EC2.
- **RDS for PostgreSQL:** Improve performance by optimizing data loading and utilizing the `autovacuum` feature effectively.
- **SQL Server Failover:** Allocate sufficient provisioned IOPS to handle your workload during failover.





# Amazon Aurora



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



# Amazon Aurora

## Purpose

- Relational database service
- **Fully compatible with** full MySQL and PostgreSQL
- **Easy transition** from MySQL and PostgreSQL

## Features

- **Performance**  
Offers **up to 5x the performance** of MySQL, **3x** that of PostgreSQL.
- **Scalability**  
Scales from **10GB to 128TB** as needed.
- **Read replicas:**  
Up to **15 replicas** to extend read capacity.
- **Security:**  
IAM for authentication, supports encryption at rest and in transit.

# Amazon Aurora

## Aurora Serverless

- **Automatic Scaling:**  
Auto-adjusts to the application's needs, **no manual scaling** required.
- **On-Demand Usage:**  
Pay-for-what-you-use is ideal for **sporadic or unpredictable workloads**.
- **Simple Setup:**  
**No management** of database instances; automatic capacity handling.
- **Cost-Effective:**  
Bills based on **Aurora Capacity Units (ACUs)**, suitable for **fluctuating workloads**.

# Amazon Aurora

## Use Cases

- **Caching:**  
Ideal for high-performance caching  
Reducing load and improving response times
- **Leaderboards and Counting:**  
In gaming and social networks
- **Session Store:**  
Session information for web applications, ensuring  
**fast retrieval and persistence.**



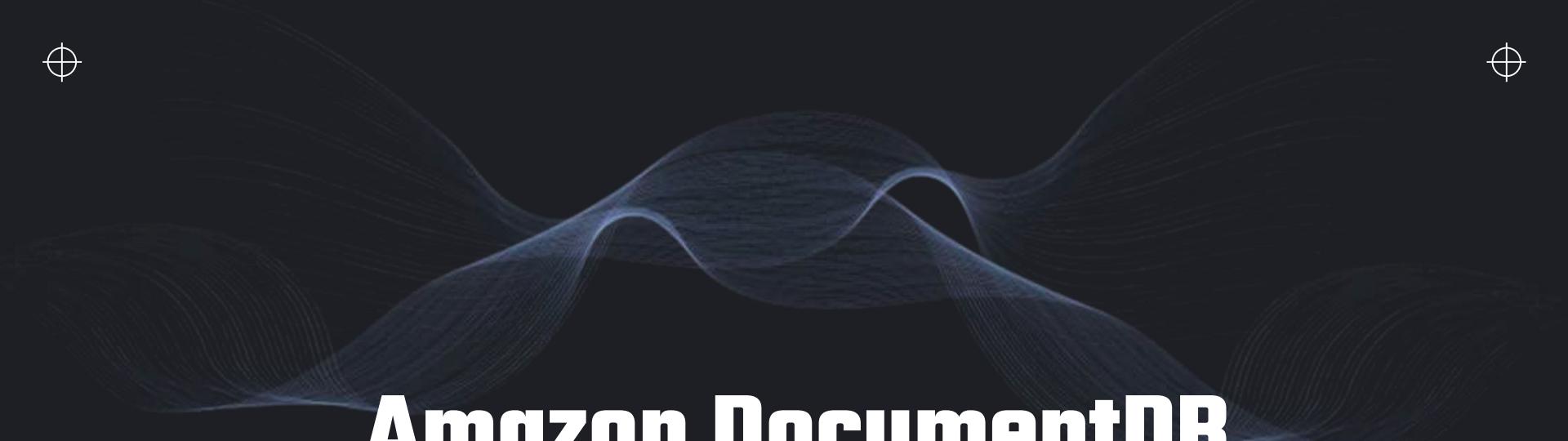
# Amazon Aurora



## Pricing

- **Node Pricing:** Charges based on type and number of nodes, varying by **CPU, memory, and network performance.**
- **Data Transfer Pricing:** Costs for data transferred "in" and "out" of the service; intra-region transfers typically not charged.
- **Backup Pricing:** Charges for backup storage beyond the free tier, priced per GB per month.
- **Reserved Instances:** Available for long-term use at a significant discount, with one or three-year commitments.





# Amazon DocumentDB

003-1040559 1250 003-77156.8 1760 0009-14563.7 73273

# Amazon DocumentDB

## Purpose

- Fully managed **NoSQL database** service.
- Document-oriented
- Fully compatible with **MongoDB**.
- **Serverless:**  
Scales automatically with needs; no infrastructure management.

## Features

- **Managed Service:**  
AWS handles provisioning, setup etc.
- **High Availability & Durability:**  
Built on AWS infrastructure with multi-AZ replication.
- **Security:**  
IAM for authentication, supports encryption at rest and in transit.

# Amazon DocumentDB

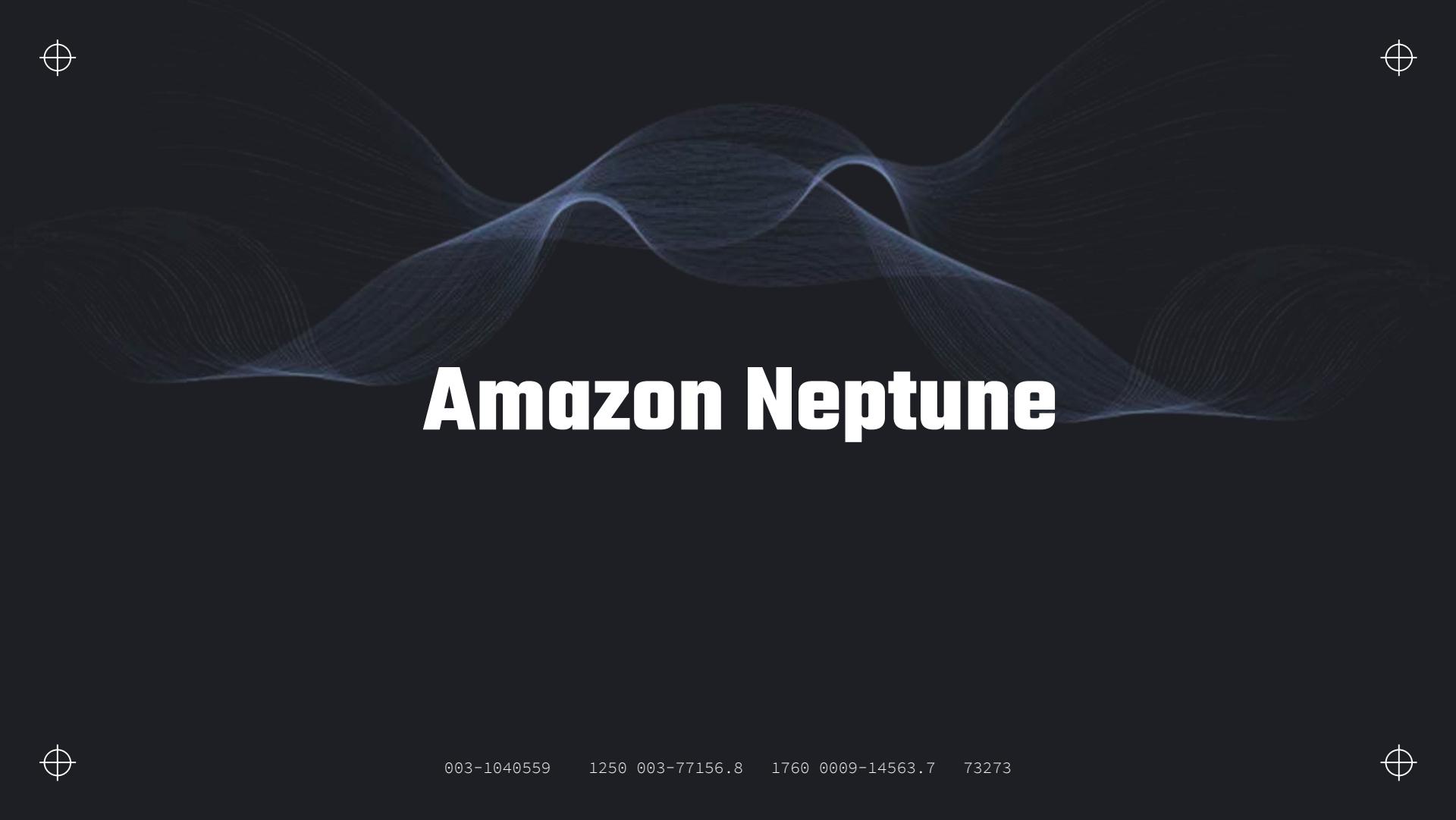
## Pricing

- **Instance Hours:** Charged based on instance run time.
- **Storage & I/O:** Fees for S3 storage and I/O operations.
- **Backup Storage:** Additional costs for backups beyond free retention limits.

# Amazon DocumentDB

## Pricing

- **Instance Hours:** Charged based on instance run time.
- **Storage & I/O:** Fees for S3 storage and I/O operations.
- **Backup Storage:** Additional costs for backups beyond free retention limits.



# Amazon Neptune



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Neptune - Overview

## Purpose

- Fully managed graph database service
- Optimized for **highly connected datasets**

## Use Cases

- Knowledge graphs,
- Fraud detection,
- Recommendation engines

## Data Model

- **Property Graph:** Vertices and edges model
- **RDF:** Triple-based model





# Neptune - Features



- **Fully Managed:** Provisioning, patching, backup by **AWS**.
- **High Availability:** Across **3 AZs**, supports **automatic failover**.
- **Secure:** Uses **AWS KMS** for encryption, provides **VPC** isolation.
- **Scalable:** Auto-scales to **64 TB**, up to **15 read replicas**.
- **Fast and Reliable:** Handles **billions of relations** with **millisecond latency**.





# Neptune – Integration & Performance



## Integration

- **AWS Ecosystem:** Integrates with Lambda, S3, SageMaker.
- **Open Standards:** Complies with Gremlin and SPARQL.

## Performance

- Query Optimization: Advanced techniques for graph traversal.
- Concurrency and Throughput: **Optimized for high traffic.**

## Pricing

- **Model:** Based on **instance size** and **usage hours**.

## Summary

- **Complexity:** Demands **graph database** knowledge.
- **Specialized:** Focused on **graph-specific** applications.





# Amazon Keyspaces (for Apache Cassandra)

# Amazon Keyspaces

## Purpose

- Fully managed **NoSQL database** service.
- Fully compatible with **Apache Cassandra**.

## Features

- **Serverless:**  
Scales automatically with needs; no infrastructure management.
- **Managed Service:**  
AWS handles provisioning, setup etc.
- **High Availability & Durability:**  
Built on AWS infrastructure with multi-AZ replication.
- **Security:**  
IAM for authentication, supports encryption at rest and in transit.



# Amazon Keyspaces



## Pricing

- **On-demand capacity:** Pay for throughput and storage.
- **Provisioned capacity:** For predictable workloads.





# Amazon MemoryDB for Redis

# Amazon MemoryDB for Redis

## Purpose

- Fully managed **in-memory database service**
- **Fully compatible with Redis**, supporting Redis APIs and data structures.

## Features

- **In-memory storage**  
for **low-latency** and **high-throughput** access
- **Scalability**  
**Automatically scales** to adapt to workload changes.
- **Durability:**  
Ensures data persistence with **snapshotting and replication** across **multiple Availability Zones**.
- **Security:**  
IAM for authentication, supports encryption at rest and in transit.



# Amazon MemoryDB for Redis



## Use Cases

- **Caching:**  
Ideal for high-performance caching  
Reducing load and improving response times
- **Leaderboards and Counting:**  
In gaming and social networks
- **Session Store:**  
Session information for web applications, ensuring  
**fast retrieval and persistence.**



# Amazon MemoryDB for Redis

## Pricing

- **Node Pricing:** Charges based on type and number of nodes, varying by **CPU, memory, and network performance.**
- **Data Transfer Pricing:** Costs for data transferred "in" and "out" of the service; intra-region transfers typically not charged.
- **Backup Pricing:** Charges for backup storage beyond the free tier, priced per GB per month.
- **Reserved Instances:** Available for long-term use at a significant discount, with one or three-year commitments.



# Amazon Timestream



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon Timestream



- Fully managed, serverless time-series database
- Designed for high-performance real-time analytics

## Use cases

- IoT applications
- Application logs
- DevOps monitoring
- Financial market data

## Features

- Serverless
- Optimized for time-series data
- Time series specific functions
- High performancece for time series data





# Amazon Timestream



## Ingestion

Lambda	
Amazon MSK	
Managed Service for Apache Flink	
Kinesis Data Streams	
IoT Core	



## Output

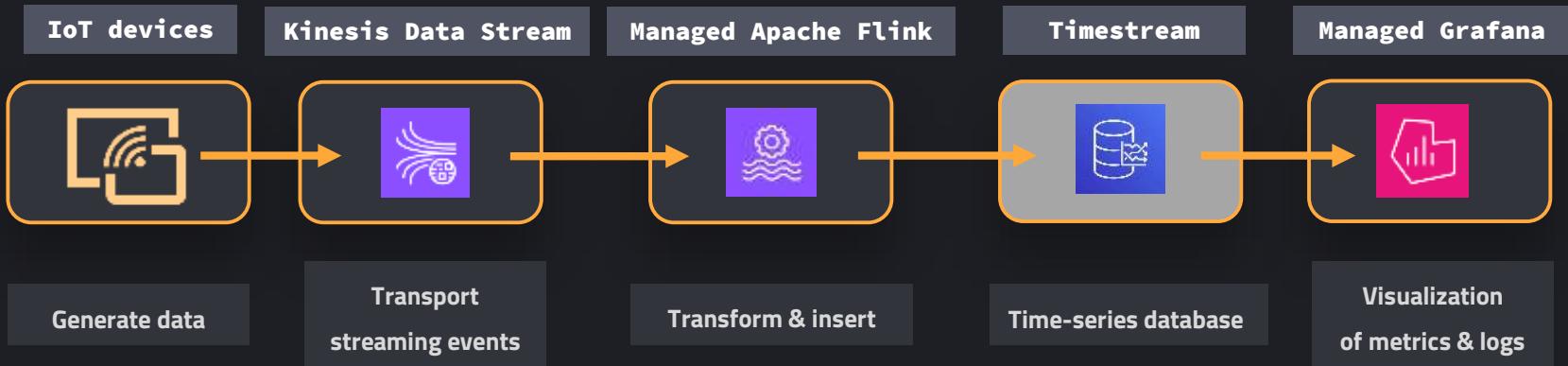
Managed Grafana	
QuickSight	
SageMaker	





# Real-time processing with IoT devices

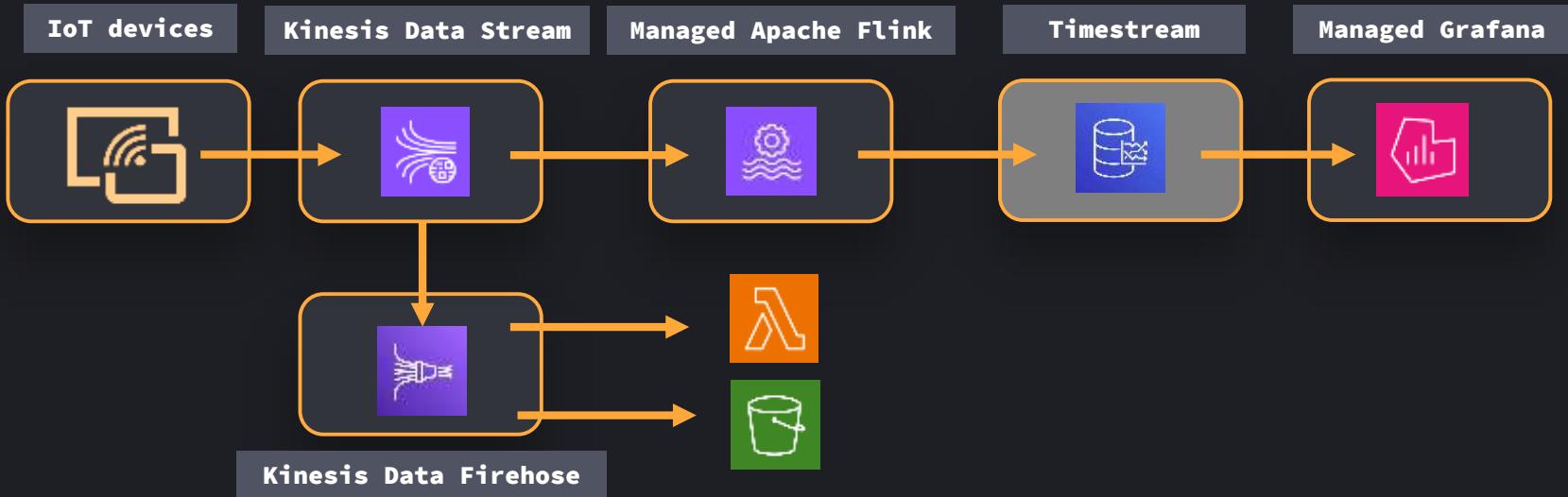
## Use case





# Real-time processing with IoT devices

## Use case





## Section 12:

# Compute Services



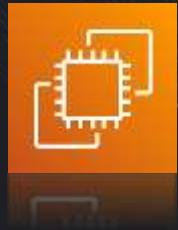
003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Elastic Compute Cloud (EC2)



003-1040559

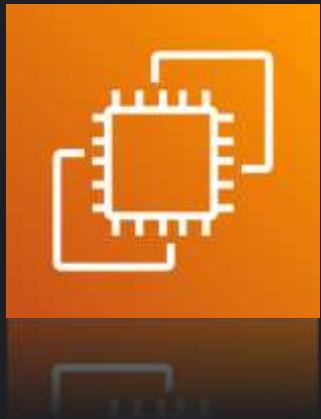
1250 003-77156.8

1760 0009-14563.7 73273





# Elastic Compute Cloud (EC2)



- Is a web service that provides **secure**, **resizable** compute capacity in the cloud
- Allows users to easily **configure**, **launch**, and **manage** virtual servers, known as instances.
- It provides **on-demand**, **scalable** computing capacity.
- Offers variety of instances with a different combinations of

CPU

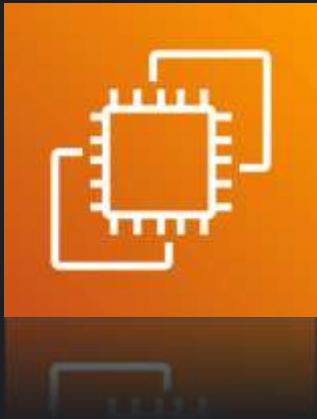
Memory

Storage

Networking



# Elastic Compute Cloud (EC2)



- Users have full control over the configuration of their EC2 instances.
- High Availability and Reliability.
- Auto Scaling.
- It seamlessly integrates with other AWS services.





# EC2 Instance Types

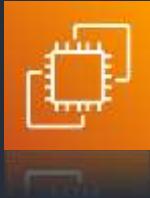


003-1040559

1250 003-77156.8

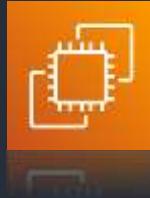
1760 0009-14563.7 73273

# EC2 Instance Types



## General Purpose

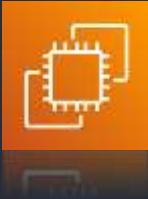
- Provide a balance of compute, memory and networking resources.



## Compute Optimized

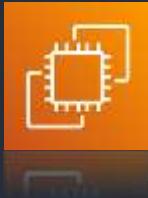
- Ideal for compute bound applications that benefit from high performance processors.

# EC2 Instance Types



## Memory Optimized

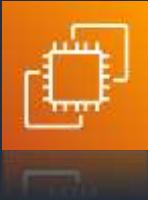
- Designed for workloads that process large data sets in memory.



## Accelerated Computing

- Use hardware accelerators, or co-processors, to perform functions.

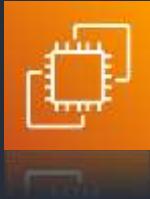
# EC2 Instance Types



## Storage Optimized

- designed for workloads that require high, sequential read and write access to very large data sets on local storage.

# EC2 Instance Types



## HPC Optimized

- Purpose built to offer the best price performance for running HPC workloads at scale on AWS.
- Ideal for applications that benefit from high-performance processors.



# AWS Batch

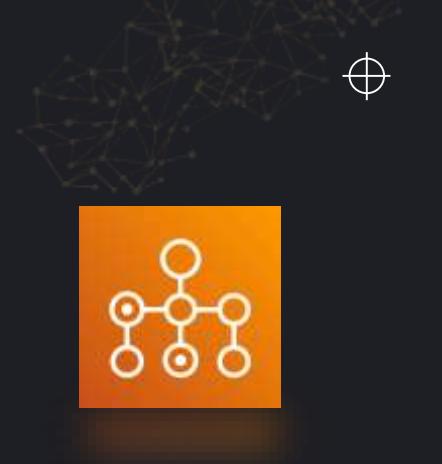


003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# AWS Batch

When to use it?

- Batch jobs based on **docker images**

## Lambda

- Lightweight, event-driven tasks
- Run code in response to events (ideal for real-time)

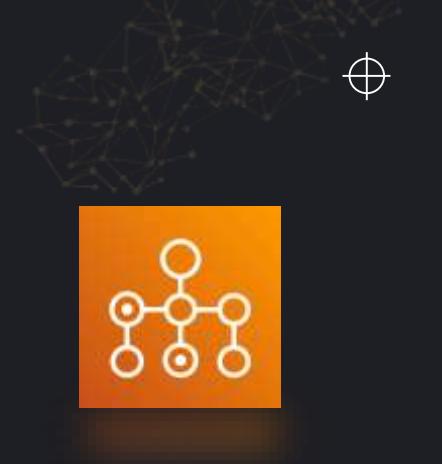
## Glue

- Specialized ETL / Data Integration service

## Batch

- General purpose, versatile (compute-intensive) batch jobs



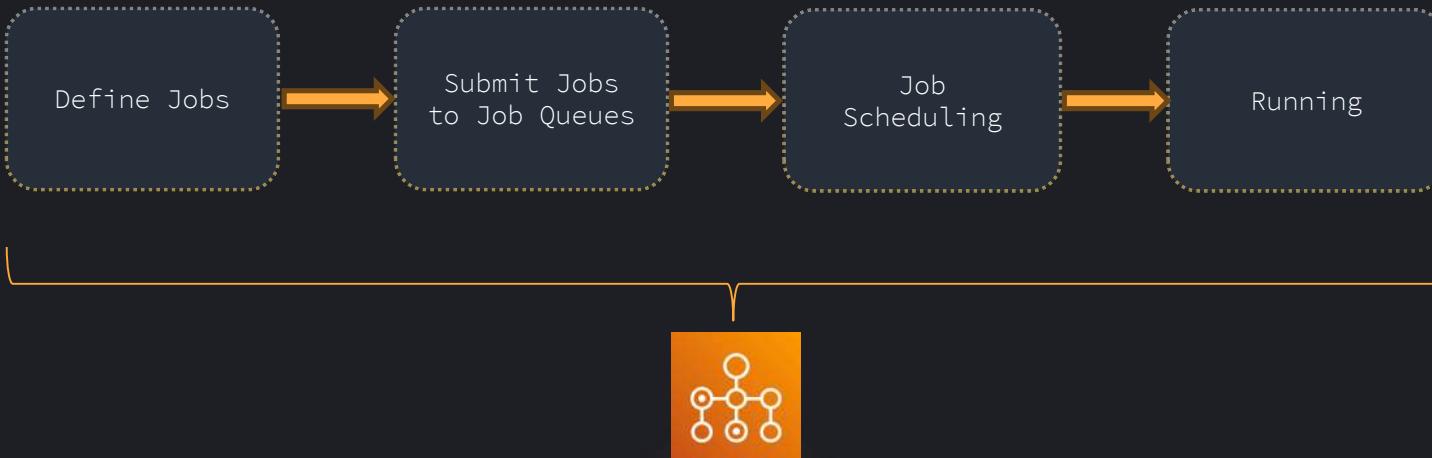


# AWS Batch

- Batch jobs based on **docker images**
- Automated Scaling
- Job Scheduling
- Can be integrated with **AWS Step Functions.**
- **Serverless:**
  - Uses EC2 instances and Spot instances.
  - Can be used together with Fargate
  - No need to manage infrastructure.
- **Pricing:**
  - **EC2 Instance / Fargate / Spot Instance** costs.



# AWS Batch – How It Works





# AWS SAM



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS SAM

- Framework for development processes.
- Simplified **serverless** application development.
  - AWS Lambda
  - Amazon API Gateway
  - Amazon DynamoDB
- It has **YAML** based template.
- Provides deployment ease.
- Provides **local testing** capabilities.





# AWS SAM - CLI

- Provides an environment that mimics AWS.
- Test and debug your serverless applications.
- It has IDE extensions.
- `sam build`, `sam package`, `sam deploy`.

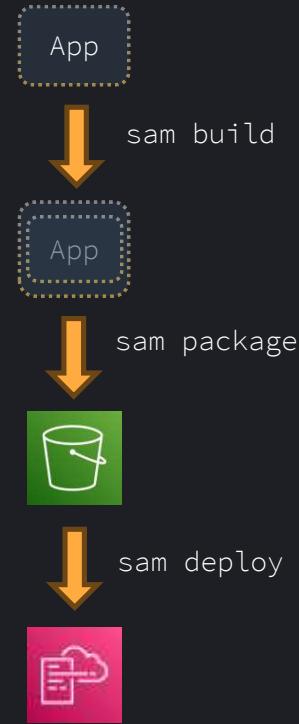




# AWS SAM – Template & Deployment

- 1) Write Your Template
- 2) Build Application: **sam build**
- 3) Package Your Application: **sam package**
- 4) Deploy Your Application: **sam deploy**

```
Resources:  
  SimpleFunction:  
    Type: 'AWS::Serverless::Function'  
    Properties:  
      Handler: index.handler  
      Runtime: nodejs14.x  
      CodeUri: s3://my-bucket/my-function.zip  
    Events:  
      ApiEvent:  
        Type: Api  
        Properties:  
          Path: /example  
          Method: get  
  
Outputs:  
  ApiEndpoint:  
    Description: "API Gateway endpoint URL"  
    Value: Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/example/"
```





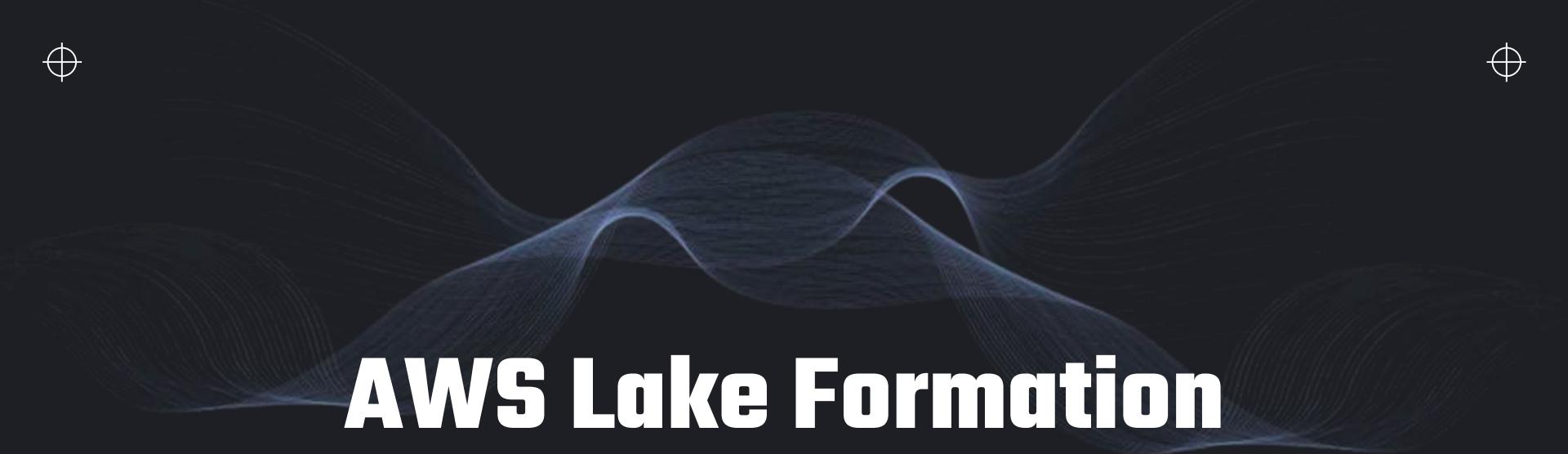
# Section 13: Analytics



003-1040559      1250 003-77156.8

1760 0009-14563.7      73273





# AWS Lake Formation



003-1040559

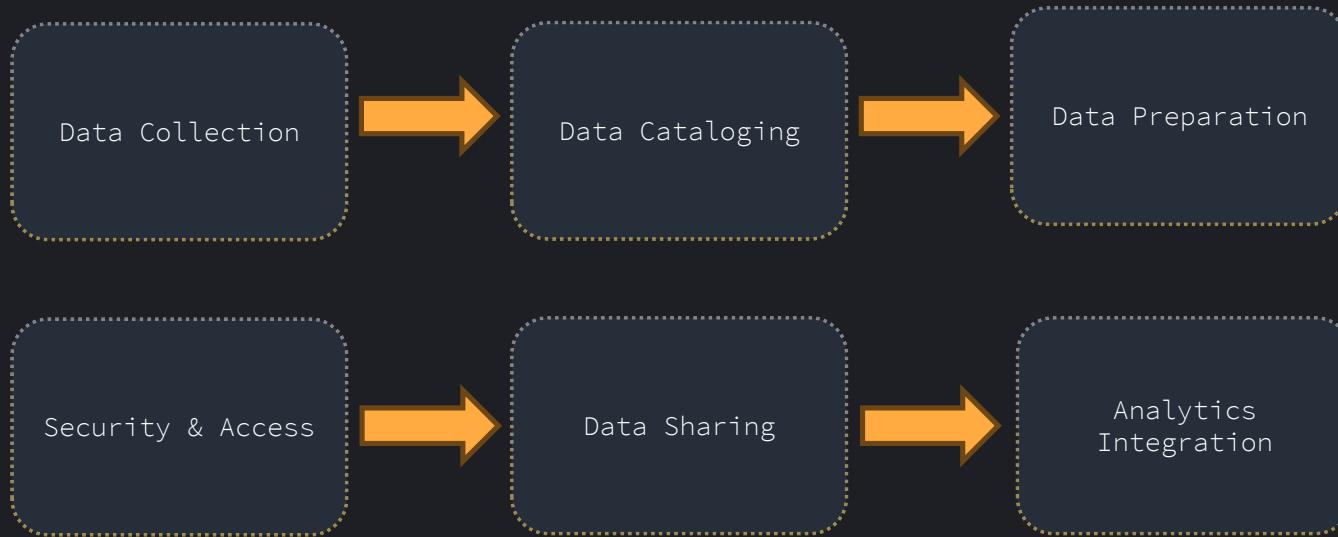
1250 003-77156.8

1760 0009-14563.7

73273



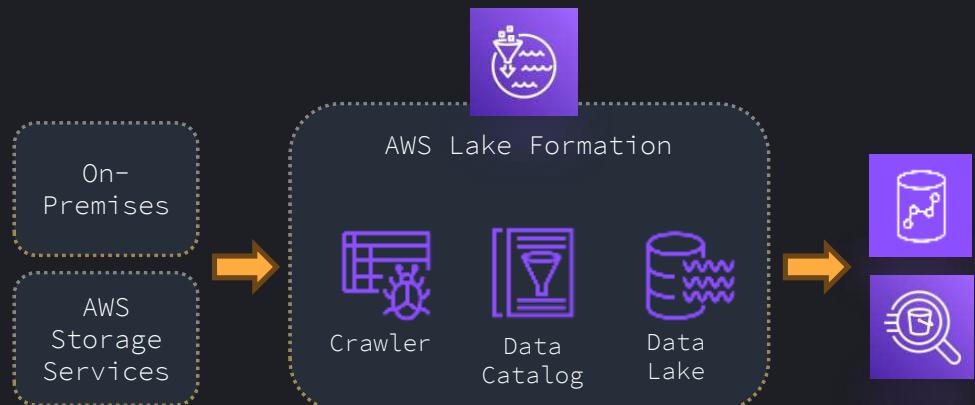
# AWS Lake Formation – How It Works



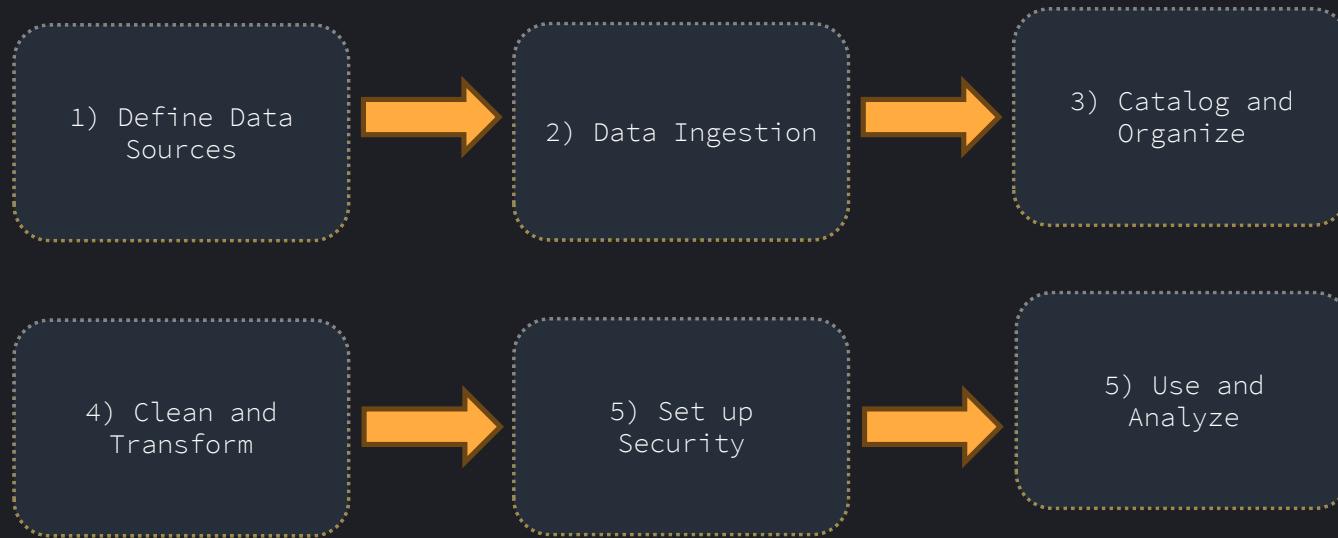


# AWS Lake Formation

- Collects data from different sources.
- Organizes and prepares data.
- Integrates with other AWS services.
  - Redshift, Athena, Glue, S3
- Automates with **blueprints**.
- **Secures** the data.



# AWS Lake Formation – How It Works





# AWS Lake Formation – Security

- **Centralized security** management.
- Control access at **database**, **table** or **column** level.



# Data Filtering - Security

- **Data Filtering** for fine-grained access control to the data

Row-level Security

restrict access to **certain rows**


Column-level Security

restrict access to **certain columns**


Cell-level Security

combines **column & row restriction**






# LF-tags – LF-TBAC

- **LF-tags** can be used to define permissions based on attributes

Attach LF tag to resource

Can be attached to Data Catalogue resources

Assign Permissions

Assign permission based on resources using LF-tags





# AWS Lake Formation – Security

- **Centralized security** management.
- Control access at **database**, **table** or **column** level.
- Role based access control.
- Cross-account access.





# AWS Lake Formation – Cross-Account

- **Share Setup:**  
Use named resources or LF-Tags for easy database and table sharing across AWS accounts.
- **Granular Access:**  
Use data filters to control access at the row and cell levels in shared tables.
- **Permissions Management:**  
Leverage AWS RAM to handle permissions and enable cross-account sharing.
- **Resource Acceptance:**  
Once shared resources are accepted via AWS RAM, the recipient's data lake administrator can assign further permissions.
- **Resource Link:**  
Establish a resource link for querying shared resources via Athena and Redshift Spectrum in the recipient account.





# AWS Lake Formation – Cross-Account

## Troubleshooting Points:

- Issues with Permissions (RAM issue)
- IAM role misconfiguration





# Amazon EMR



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon EMR

- Fast, distributed data processes.
- Uses big data frameworks:
  - Apache Hadoop, Apache Spark
  - For petabyte scale processing
  - Glue is easy to use but for heavy workloads (petabytes) not that suitable
  - Migration from existing on-premise resources



Create cluster [Info](#)

▼ Name and applications - **required** [Info](#)  
Name your cluster and choose the applications that you want to install to your cluster.

Name

Amazon EMR release [Info](#)  
A release contains a set of applications which can be installed on your cluster.

Application bundle

Spark Interactive 	Core Hadoop 	Flink 	HBase 	Presto 	Trino 	Custom 
-----------------------	-----------------	-----------	-----------	------------	-----------	------------





# Amazon EMR

- Fast, distributed data processes.
- Uses big data frameworks:
  - Apache Hadoop, Apache Spark
- Cluster based architecture.
- Pricing is based on EC2 usage and hourly service rates.
- **Security:**
  - IAM, VPC, KMS.



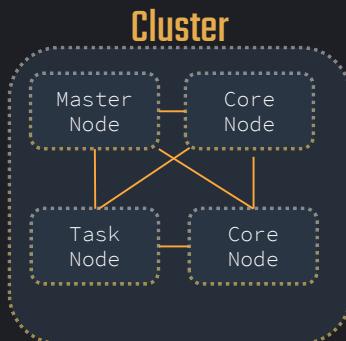
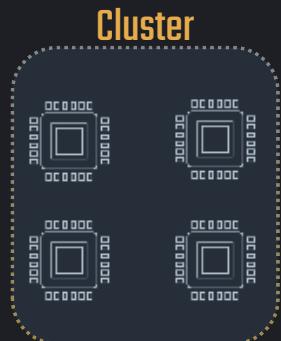


# Amazon EMR

## What Is Hadoop

- Distributed storage and processing framework.
  - 1) Hadoop Distributed File System (HDFS)
  - 2) MapReduce

## EMR Cluster Structure



- **x86-based instance:**  
Versatile & traditional choice  
- **Graviton-based instance:**  
Balance of compute & memory  
20% cost savings

- 
- A diagram illustrating the three types of nodes in an EMR cluster. An orange arrow points from the cluster structure diagram to this section. The three types of nodes are listed in boxes:
- Master Node: Manages the cluster
  - Core Nodes: responsible for running tasks and storing data
  - Task Nodes: handle additional data process. Does not store data





# **Amazon EMR**

## **Cluster Types & Storage**





# Amazon EMR

## Cluster Types

### Transient Cluster



Temporary clusters

After the job cluster terminates

Used for intermittent jobs

### Long Running Clusters



Ongoing processes

Continuously consume and process





# Amazon EMR - Storage

- **HDFS (Hadoop Distributed File System):**
  - **Location:** Located on the local disks of each node.
  - **Use Case:** Temporary storage, non-persistent, high throughput.
- **EMR File System (EMRFS):**
  - **Location:** HDFS that allows clusters to store S3.
  - **Use Case:** Persistent, cost-effective, Storing input and output.
- **Local File System:**
  - **Location:** Resides on each individual EC2 instance.
  - **Use Case:** Storing input and output.
- **EBS for HDFS:**
  - Works differently than its usual.
  - Temporary storage.





# Amazon EMR

## Scalability



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon EMR - Scalability



## Manual Scaling

You will be manually scaling the cluster.



## Amazon EMR Managed Scaling

EMR takes care of automated scaling policies.  
Instance groups and instance fleets.



## Custom Automatic Scaling

You will be define automated scaling policies.  
Only for instance groups.





# Amazon EMR – Deployment Options

## EMR on EKS

- Run data frameworks on Amazon **Elastic Kubernetes Service**.
- Simplifies container based big data processes.
- Provisioning clusters not needed.

## EMR Serverless

- **Serverless** runtime environment.
- It provides **fast** job startup and **high availability**.
- Pay for what you use.





# Apache Hive



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





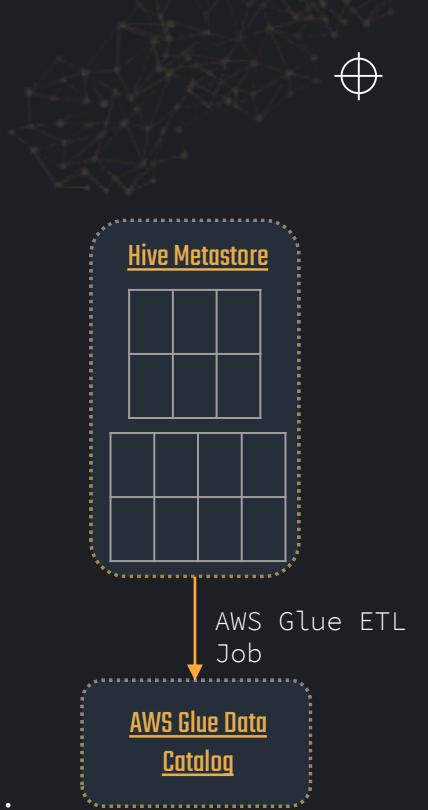
# Apache Hive

- Distributed data warehousing tool for querying and managing datasets.
- Hive stores data in tables.
  - Text files, Parquet, ORC...
- Provides SQL-like queries also called Hive QL
- Compatible with the Hadoop Distributed File System (HDFS) and Amazon S3.



# Apache Hive

- Amazon EMR is also capable working with Hive.
  - Amazon EMR: Simplifies running big data frameworks.
- Within EMR, it is easy to migrate your Hive system.
- Migration from Hive to AWS Glue Data Catalog is also possible.
- AWS Glue Data Catalog has built-in compatibility with Hive.
- Sharing metadata between Hive and AWS Glue is possible.
- You can run Hive queries on cataloged data in AWS Glue.





# Amazon Managed Grafana



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon Managed Grafana



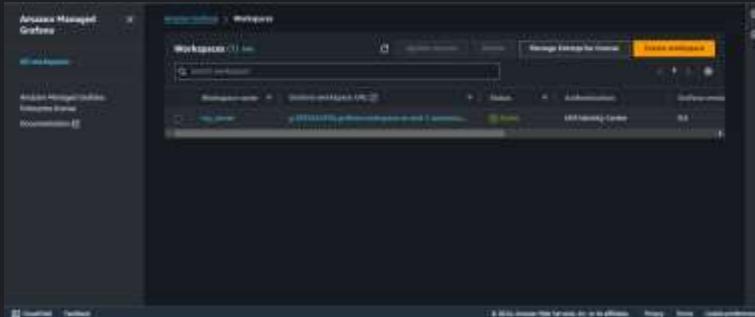
Fully managed Grafana service



- Open-source **visualization tool**
- Create dashboards for **metrics, logs and traces**

## Use cases:

- Monitoring systems and application
- Dashboards for log or sensor data
- Real-time monitoring for IT infrastructure
- Use alerts

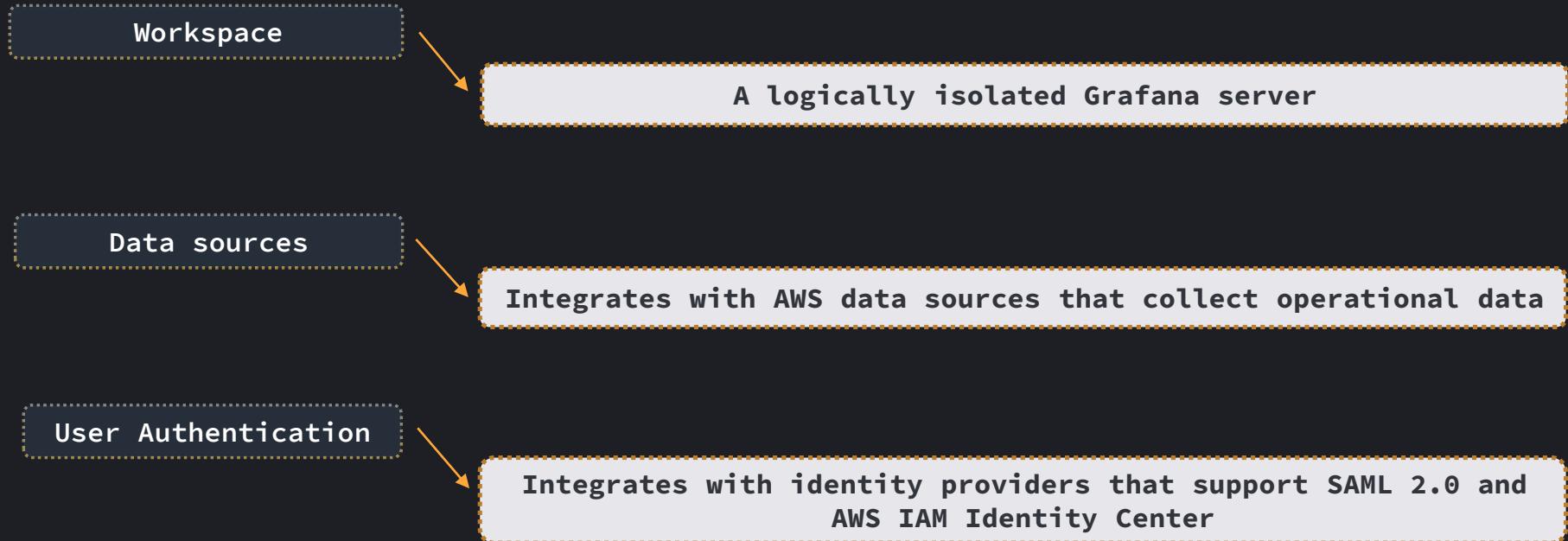


Managed Grafana interface





# Amazon Managed Grafana Features





# Amazon Managed Grafana - How It Works



CloudWatch

Very common

Timestream

IoT devices & time series

ElasticSearch

Data stored in Elastic Search



Create Grafana Workspaces



Manage User Access



Connect to Multiple Data Sources



Setup Dashboards





# Amazon OpenSearch



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon OpenSearch



- **What is Amazon OpenSearch?**
  - Fully managed Search Engine
  - Built in Dashboards for real-time data analysis
  
- **Use Cases**
  - *Real-time Application Monitoring*
  - *Business Intelligence (BI) and Data Analytics*
  - ...





# Amazon OpenSearch



- **Pricing**

Pay as you go.

- **Infrastructure:**

Fully managed.

- **Scalability:**

Ability to scale up or down manually.

- **Integration:**

Integrates with other AWS services

- **Availability:**

Multi-AZ deployments, automated snapshots.

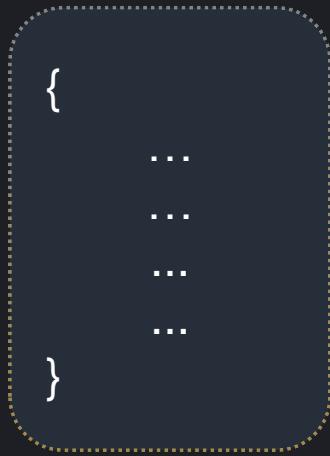




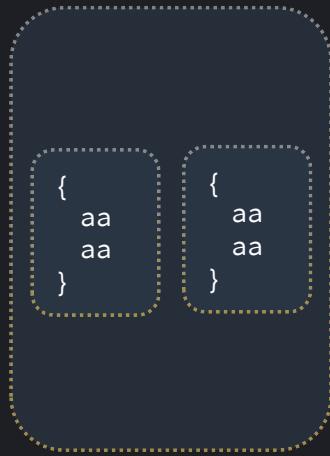
# Amazon OpenSearch Key Components



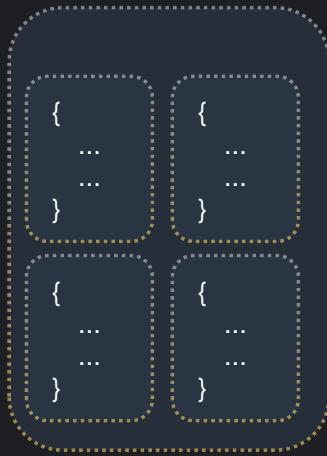
Documents



Types



Indices





# Amazon OpenSearch

## Node, Cluster and Shard

- **What is a Node:**

Single running instance of OpenSearch.

- Data Nodes
- Master Nodes
- Client Nodes

- **What is a Cluster ( ⇒ Domain):**

Collection of one or more nodes.

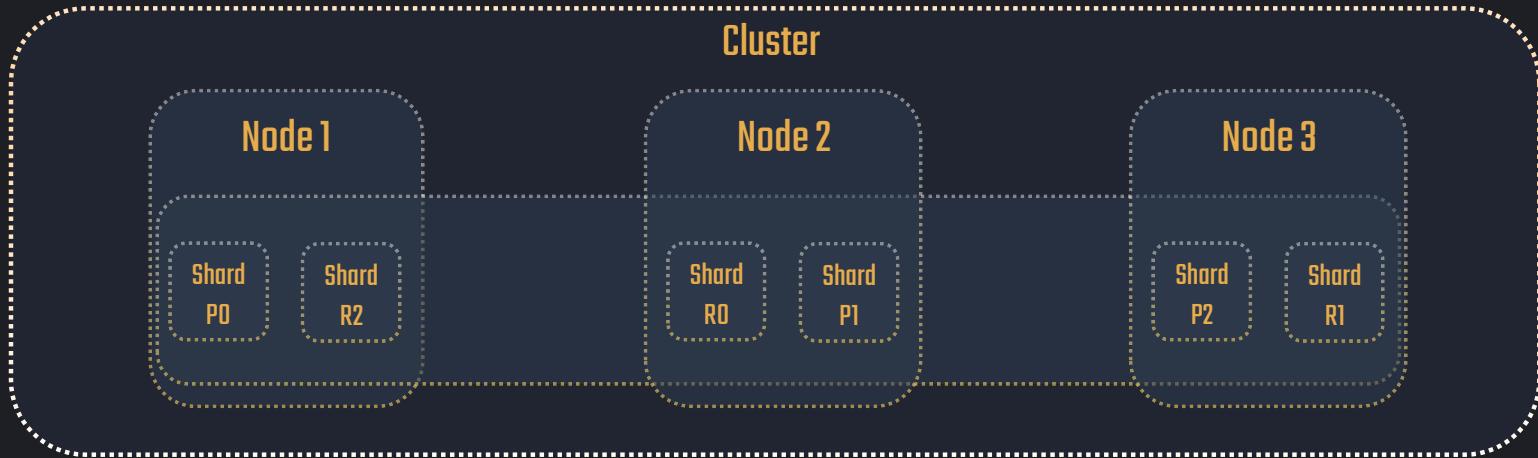
- **What is a Shard:**

Partitions of an index's data.

- Primary Shards
- Replica Shards



# Amazon OpenSearch – Infrastructure





# Amazon OpenSearch

## Managing and Accessing Data

- **Creating an Index:**

Define its settings and mappings.

```
PUT /my-index-000001
{
  "settings": {
    "index": {
      "number_of_shards": 3,
      "number_of_replicas": 2
    }
  },
  "mappings": {
    "properties": {
      "name": {
        "type": "text"
      },
      "age": {
        "type": "integer"
      }
    }
  }
}
```





# Amazon OpenSearch

## Managing and Accessing Data

- **Adding Documents:**

Define its settings and mappings.

```
POST /my-index-000001/_doc/1
{
  "name": "John",
  "age": 30
}
```





# Amazon OpenSearch



## Managing and Accessing Data

- **Searching:**

Search through indices.

```
GET /my-index-000001/_search
{
  "query": {
    "match": {
      "name": "John"
    }
  }
}
```





# Amazon OpenSearch - Configurations

- **Domain:**

Management of OpenSearch clusters.

- **Master Node:**

Responsible for cluster-wide operations.

- **S3 Snapshot:**

S3 snapshots can be used to restore your cluster.





# Amazon OpenSearch – What to Avoid



- **OLTP and Ad-Hoc Queries:**

Can lead to suboptimal performance.

- **Over-Sharding:**

Can lead to increased overhead and reduced efficiency.

- **OpenSearch as Primary Data Store:**

Not designed for transactional data integrity.





# Amazon OpenSearch – Performance

## **Memory Pressure Management:**

- Watch for JVM memory pressure errors;
- Balance shard allocations to avoid overloading

## **Shard Optimization:**

- Reduce shard count to mitigate memory issues

## **Data Offloading:**

- Delete older or unused indices;
- Consider archiving data to Amazon Glacier to improve performance.





# Amazon OpenSearch – Security



- **Authentication:**

- Native Authentication: Users and Roles
- External Authentication: Active Directory, Kerberos, SAML, and OpenID

- **Authorization:**

- Role Based Access Control (RBAC): Through Users and Roles
- Attribute-based access control (ABAC): Based on user attributes



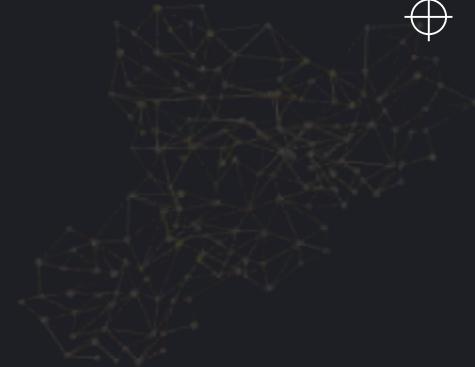
# Amazon OpenSearch – Security

- **Encryption:**

- In-transit encryption: **TLS** encryption.
- At-rest encryption: Third party applications.

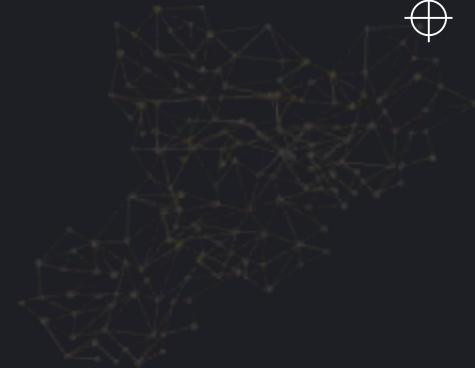
- **Audit Logging:**

- Identifying potential security issues.





# Amazon OpenSearch – Dashboards



- **Data Visualization:**

- Line charts
- Bar graphs
- Pie charts

- **Dashboard Creation:**

Each dashboard is customizable; users are able to create dashboards as per their needs.





# AWS OpenSearch – Storage Types



## Hot Storage

- Frequently accessed & instant retrieval
- Fastest performance
- Standard data nodes use it
- Real-time analytics and recent log data

## UltraWarm Storage

- Cost-effective for read-only
- S3 + caching
- Not frequently written or queried
- Can move to Hot Storage for editing

## Cold Storage

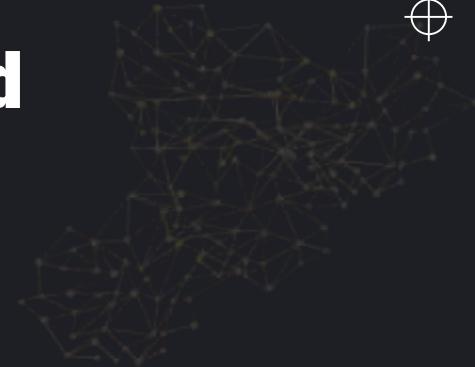
- Rarely used (e.g. archival data)
- Uses Amazon S3, hence no compute overhead
- Lowest cost
- Can be attached to UltraWarm nodes



# Amazon OpenSearch – Reliability and Efficiency

## Cross-Cluster Replication:

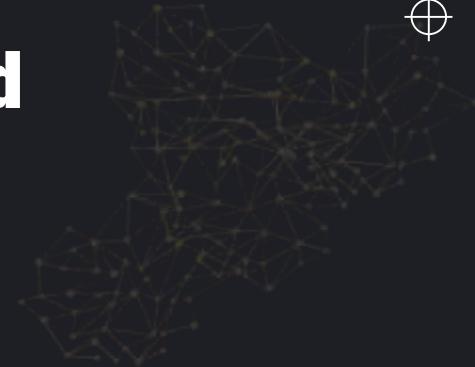
- **What It Is:**
  - Allows you to copy and synchronize data.
  - Used for increasing data availability.
- **Why It Is Important:**
  - Provides availability.
  - Prevents hardware failure and network issues related blocks.



# Amazon OpenSearch – Reliability and Efficiency

## Index Management:

- **What It Is:**
  - Automates the index managing process.
  - Defines indices lifecycles.
- **Why It Is Important:**
  - Provides cost efficiency.
  - Provides performance improvements.





# Amazon OpenSearch – Reliability and Efficiency

## Infrastructure Management:

- **What It Is:**
  - Deciding disk scale.
  - Deciding master node quantity.
- **Why It Is Important:**
  - Determines your system resilliance and stability.





# Amazon OpenSearch – Serverless

- **Serverless Flexibility:**

Auto-scales based on demand, reducing management overhead.

- **Cost Efficiency:**

Pay only for what you use, ideal for **variable workloads**.

- **Seamless AWS Integration:**

Enhances capabilities with AWS services like **Lambda, S3, and Kinesis**.





# Amazon QuickSight



003-1040559

1250 003-77156.8

1760 0009-14563.7

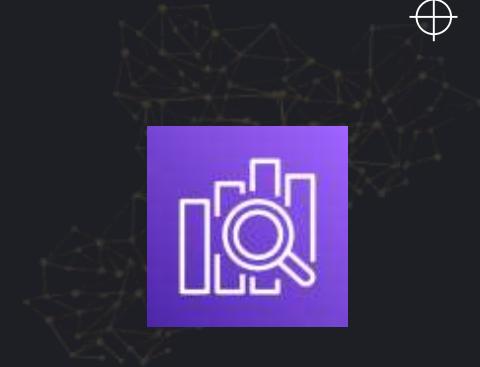
73273



# Amazon QuickSight

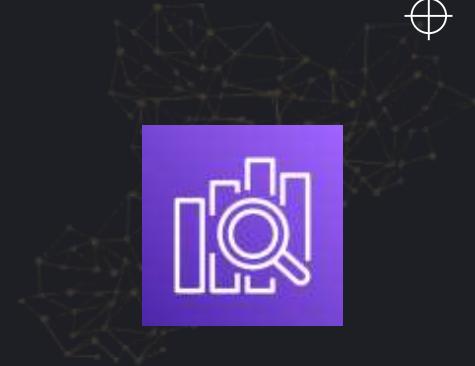
- **What is QuickSight?**

- AWS's visualization Service.
- ML-Powered dashboards.



- **Use Cases**

- *Data Exploration*
- *Anomaly Detection*
- *Forecasting*
- ...



# Amazon QuickSight

- **Scalability**

Automatically scales up and down.

- **Serverless:**

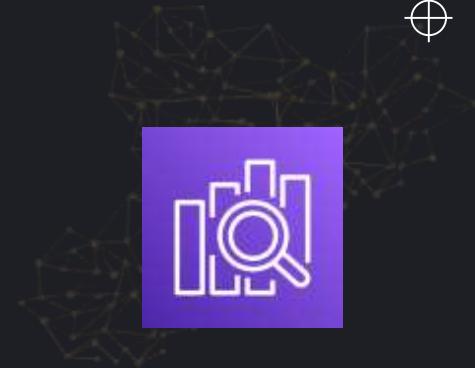
Fully managed.

- **Machine Learning Powered:**

Ability to scale up or down manually.



# Amazon QuickSight



- **Data Analysis**

Workspace for creating visualizations

- **Data Visualization:**

Visuals (Charts)

- **Dashboards:**

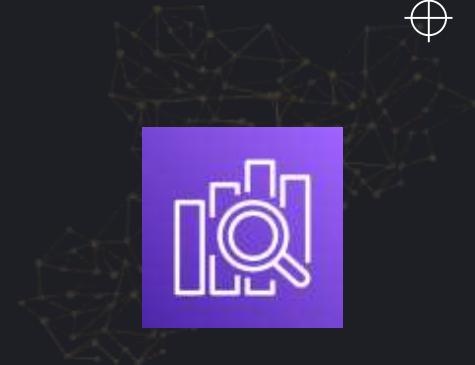
Published version of an *analysis*



# Amazon QuickSight

**SPICE** Super-fast, Parallel, In-memory Calculation Engine

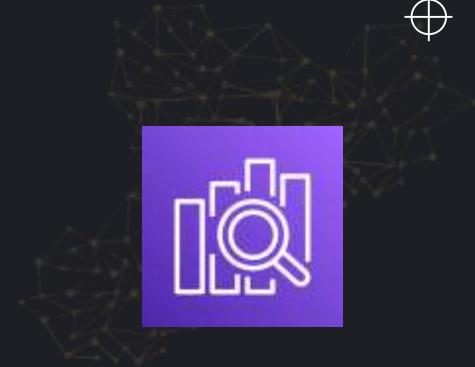
- **In memory engine**
- **10GB SPICE per user**
- **Benefits:**
  - Speed and Performance
  - Automatic Data Refreshes
  - Synchronization



# QuickSight – Dashboards

## Features

- **Automatic Refreshes:**  
Automatically refreshes dashboards.
- **Collaboration and Sharing:**  
Can be shared with team members.
- **Mobile Accessibility:**  
Mobile responsive dashboards.

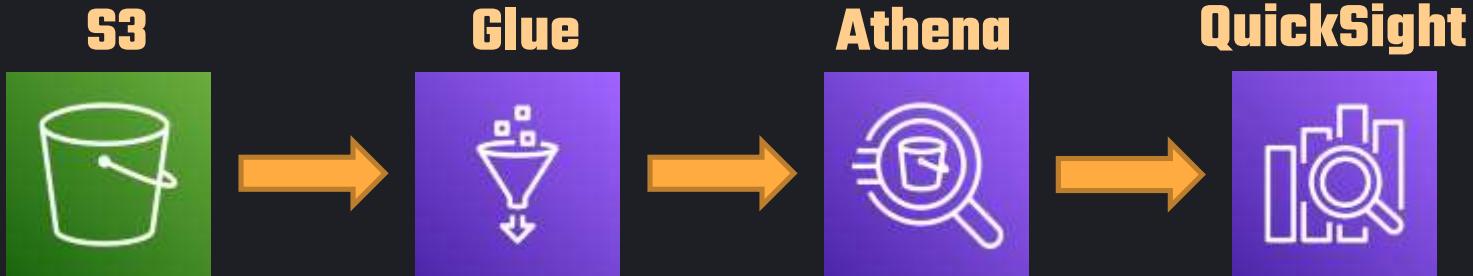


# QuickSight – Data Sources

- **AWS Services:**

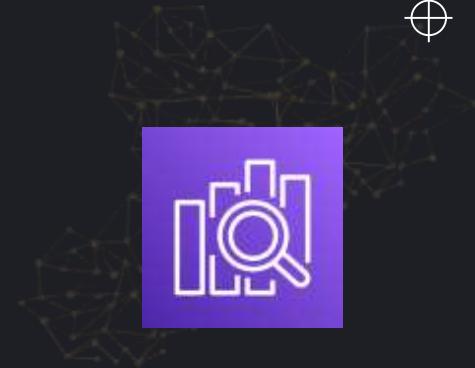
- S3
- RDS
- Aurora
- Redshift
- Athena

- **Data Pipeline Scenario:**



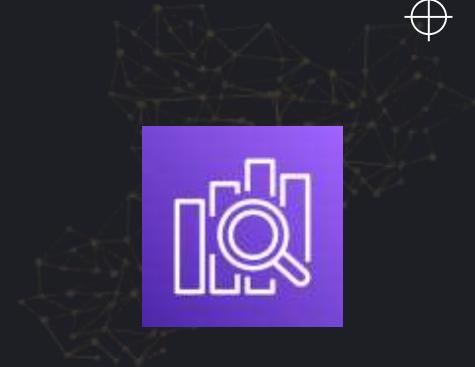
# QuickSight – Data Sources

- **CSV, Excel, TXT, S3**
- **OpenSearch**
- **Aurora/RDS/Redshift**
- **Third-Party Databases Services:**
  - *PostgreSQL*
  - *MySQL*
  - *SQL Server*
  - *Oracle*
- **ODBC/JDBC data sources**



# QuickSight – What to Avoid

- **Overloading Dashboards**
- **Poor Data Security**
- **Using ETL**





# Amazon QuickSight

## Licensing



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# QuickSight – Licensing



## Standard Edition

- Small group of users
- No advanced features need

Type	Price	Included SPICE Capacity
Annual plan	\$9/user/month	10 GB/user 0.25\$/GB for additional capacity
Monthly plan	\$12/user/month	





# QuickSight – Enterprise

- **Advanced Features:**

- RLS
- Hourly refreshes
- ML Insights

- **Administrative features:**

- User management
- Encryption

- **Pricing**

- Pay-per-session for readers



# QuickSight – Enterprise

- **Author License:**

- Connect to Data
- Create Dashboards
- Share content

- **Reader License:**

- Explore Dashboards
- Get Reports
- Download Data

- **SPICE**

- \$0.38 per GB/month

	<b>Month-to-month</b>	<b>With annual commitment</b>
<b>Author</b>	<b>\$24/month</b>	<b>\$18/month</b>
<b>Author with QuickSight Q</b>	<b>\$34/month</b>	<b>\$28/month</b>

	<b>Month-to-month</b>
<b>Reader</b>	<b>\$0.30/session up to \$5 max/month</b>
<b>Reader with QuickSight Q</b>	<b>\$0.30/session up to \$10 max/month</b>



# QuickSight – Enterprise

- **Additional Features**

- QuickSight Embedded
- Paginated Reports
- Alerts and Anomaly Detection

- **Author Pro & Reader Pro**

Additional Generative BI capabilities





# Amazon QuickSight

## Security



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# QuickSight – Security

## Access Control

- **Resource Access Control**
- **IAM Integration**
- **Active Directory Connector**
- **MFA supported**
- **IAM Configuration**

You are responsible to assign IAM role.

Necessary to access the data sources.

IAM permission necessary for encrypted data sources.

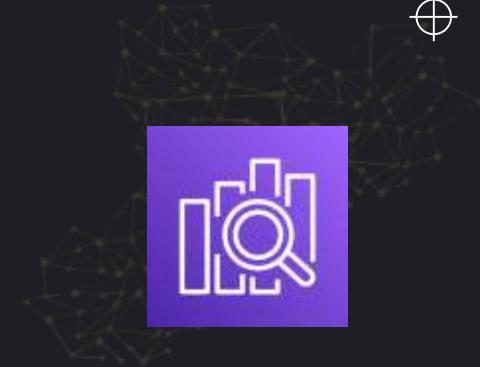




# QuickSight – Security

## Data in VPC

- **Private VPC Access Through ENI**



## Data On-Premise

- **Use AWS Direct Connect**

## Run QuickSight in VPC

- **Full network isolation**
- **IP Restriction**



# QuickSight – Cross-Region/Account



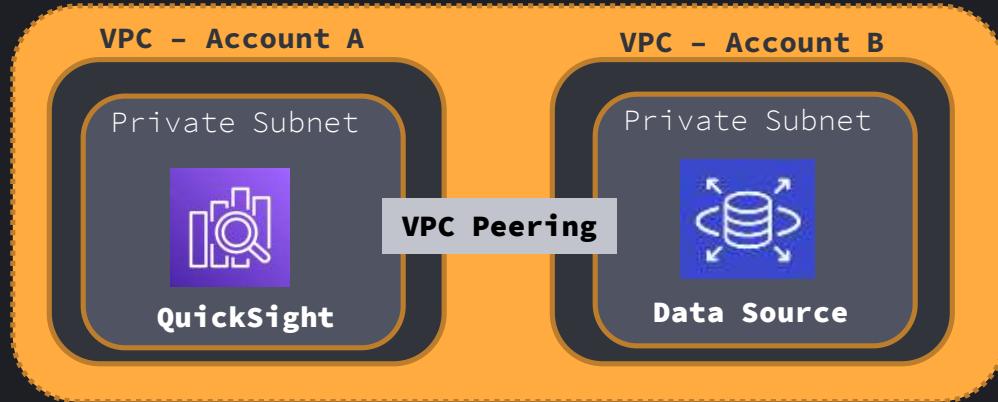
## Standard Edition

- Accessing QuickSight from **Private Subnet not possible**
- **Security group** attached to the data source (e.g., an RDS instance in another account) allows **inbound connections of QS**



# QuickSight – Cross-Region/Account

## Enterprise Edition



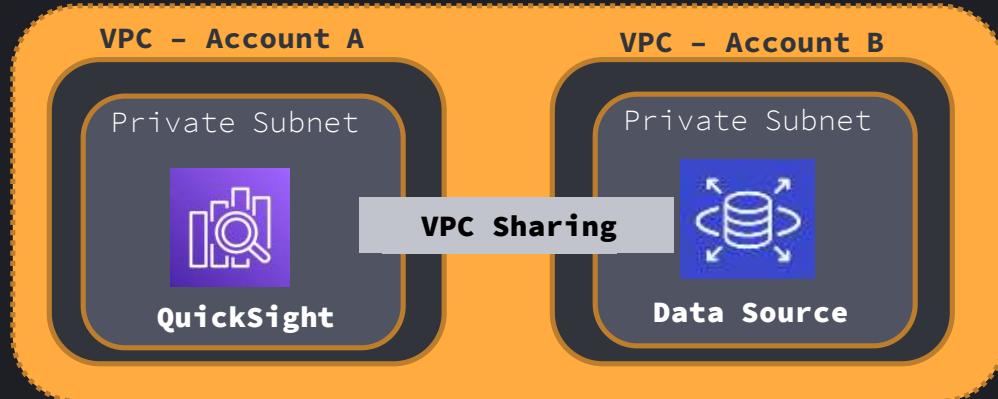
- **Use an Elastic Network Interface (ENI) within a VPC**  
Data sources like Amazon RDS/Redshift in private subnets
- **VPC Peering**  
Connect them using VPC peering.





# QuickSight - Cross-Region/Account

## Enterprise Edition



- **AWS Transit Gateway:**

For managing connections at scale within the same region

- **AWS PrivateLink:**

Securely exposes services across accounts without exposing data to the public internet.

- **VPC Sharing:**

Allows multiple AWS accounts to share a single VPC, facilitating access to shared resources like QuickSight and databases.





# Row-Level Security (RLS)

Enterprise



- **Customizable Access**

Which data rows can be seen



- **Dataset Filters**

Applied using dataset filters

- **Data Security**

Users only see data relevant to their role

- **Column-Level Security:**

Manage access to specific columns

Enterprise





# Amazon Q in QuickSight

- **Natural language queries** for interacting with data.
- **Ask** data-related **questions in plain English**.
- Delivers **answers as visualizations**, summaries, or raw data.
- Enhances data analytics **accessibility**, no technical expertise needed





## Section 14:

# Machine Learning



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon SageMaker

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# Amazon SageMaker

- Simplifies machine learning aspects.

**Build**

- Supports various compute environment and frameworks  
⇒ Jupyter notebooks, TensorFlow & Pytorch



**Train**

- Automatically manages infrastructure

**Deploy**

- Provides deployment ease

**Optimization**

- Offers automatic model tuning





# Amazon SageMaker

## Amazon SageMaker Studio



- It is an IDE with unified interface.
- Provides collaboration and **version control**.
- Includes various tools and components:
  - **Notebooks:** Pre-configured Jupyter notebooks.
  - **Experiments:** Organize, track, and compare.
  - **Debugger:** Analyze and debug.
  - **Autopilot:** Automated model creation.





# Amazon SageMaker

## Access Management

- **SageMaker-Specific Policies:**
  - AmazonSageMakerFullAccess: Provides full access to SageMaker resources.
  - AmazonSageMakerReadOnly: Provides read-only access.
- **Resource-Based Policies:**
  - You can specify accesses based on SageMaker resources.
- **Fine-Grained Access Control with Tags:**
  - You can control access based on specific conditions.
- **Integration with Other AWS Services:**
  - Amazon VPC: Provide network-level access control.
  - AWS KMS: Manage encryption keys for used or produced data by SageMaker.





# Amazon SageMaker – Feature Store

## Feature Store

- ⇒ Store and manage data features in one place.
- ⇒ Supports Online and Offline usage.

- Online store

- ⇒ Real-time apps, low latency

- Offline store

- ⇒ Historical data analysis.

- ⇒ You can group your features

Id	Time	Feature1	Feature2

Id	Time	Feature1	Feature2





# Amazon SageMaker – Feature Store

- Data can be ingested through from many services
  - EMR, Glue, Kinesis, Kafka, Lambda, Athena...

## Pipeline



## Benefits

- **Efficiency:** Reduces the effort and complexity.
- **Flexibility:** Supports both real-time and batch data processes.





# Amazon SageMaker – Lineage Tracking

- **Manage** and **track** lifecycle of your ML models.
- Provides better insights on your ML workflow.
- Useful for understanding **model dependencies**, **auditing**, and **reproducing results**.

## Lineage Tracking Entities

- Representation of every component in ML workflow.
- SageMaker automatically creates those entities.





# Amazon SageMaker – Lineage Tracking

## Lineage Tracking Entities

### Trial Components

⇒ represent individual stages or steps

### Trials

⇒ Trials help in evaluating different approaches or iterations

### Experiments

⇒ Containers for organizing the trials, focus on specific problem

### Contexts

⇒ Provide a logical grouping for other entities

### Actions

⇒ Represent operations or activities involving artifacts

### Artifacts

⇒ Data objects generated throughout the ML lifecycle: Datasets, model parameters...

### Associations

⇒ Defines the relationships between entities





# Amazon SageMaker – Lineage Tracking

## Benefits of SageMaker ML Lineage Tracking:

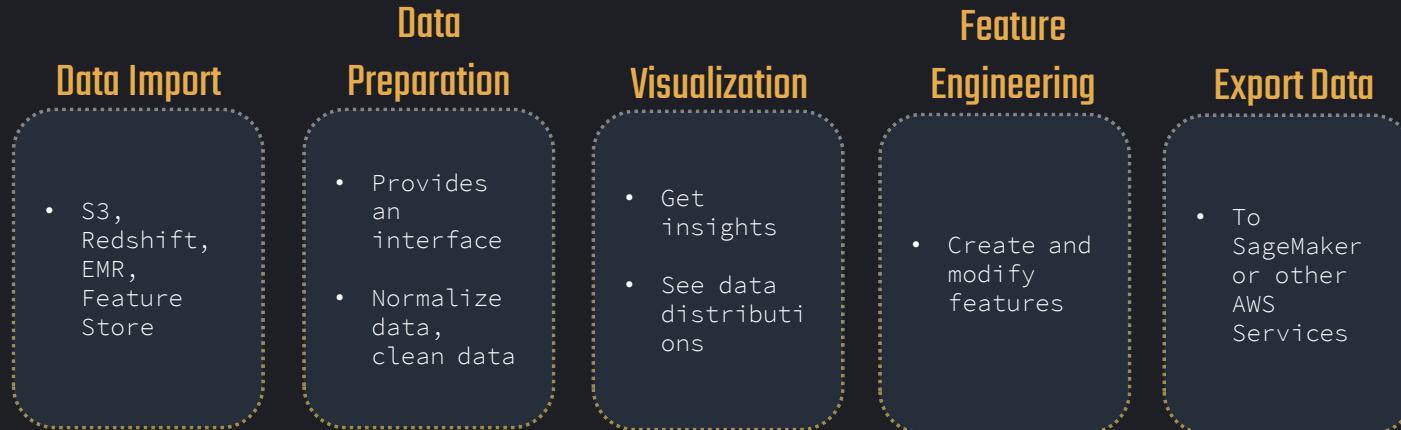
- **Reproducibility:**
  - Enables to reproduce and trace results.
- **Auditability:**
  - Provides a detailed history of.
- **Governance:**
  - Enhances the governance of ML projects.
- **Collaboration:**
  - Makes it easier for teams to collaborate.





# Amazon SageMaker – Data Wrangler

- It simplifies data preparation process.





# Amazon SageMaker – Data Wrangler

## Quick Model

- You can quickly test your data.
- Automatically trains/tests the data and provides you insights:
  - Model summary
  - Feature summary
  - Confusion matrix





## Section 15:

# Application Integration



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Step Functions



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Step Functions



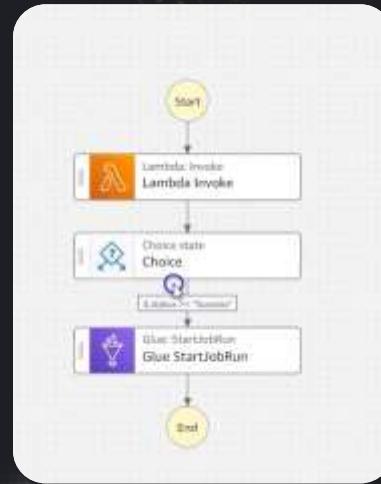
- **Create multi-step workflows**

⇒ Visual workflows in **Visual Editor**

- **Use Cases**

- **Application Orchestration:** Automates tasks across applications.
- **Data Processing:** Manages complex, conditional data workflows.
- **Microservices Orchestration:** Coordinates microservices with error management.
- **Machine Learning Workflows:** Streamlines the entire machine learning cycle.

⇒ Whenever we want to **orchestrate & integrate multiple services**





# AWS Step Functions

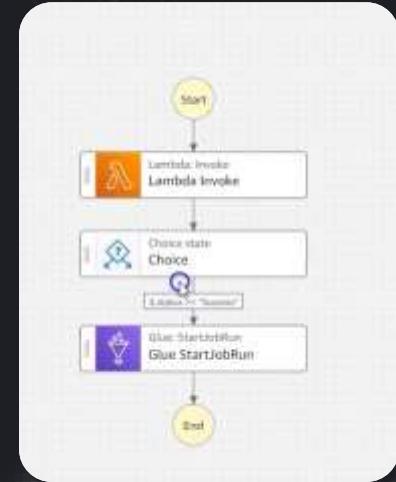


- **What is State Machine?**

State machine = Workflow

State = Step

Task = Unit of work





# AWS Step Functions



- **Different types of states ( = steps)**

- **Task State:** Executes specific work like a Lambda function or API call.
- **Choice State:** Adds branching logic, directing execution based on conditions.
- **Wait State:** Pauses execution for a set duration or a specific time.
- **Succeed State:** Ends the execution successfully with the provided output.
- **Fail State:** Stops execution and marks it as failed.
- **Parallel State:** Multiple branches simultaneously, aggregating results.
- **Map State:** Iterates over a list, processing each item with a sub-workflow.
- **Pass State:** Passes input directly to output, with optional data transformation.



# AWS Step Functions



- **State Machine defined in ASL**

ASL = Amazon States Language

```
Comment: "An example of the Amazon States Language using a choice state."
"StartAt": "FirstState"
"States":
  "FirstState": {
    "Type": "Task",
    "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
    "Next": "ChoiceState"
  },
  "ChoiceState": {
    "Type": "Choice",
    "Choices": [
      {
        "Variable": "$.Foo",
        "NumericEquals": 1,
        "Next": "FirstMatchState"
      },
      {
        "Variable": "$.Foo",
        "NumericEquals": 2,
        "Next": "SecondMatchState"
      }
    ],
    "Default": "DefaultState"
  },
  "FirstMatchState": {
    "Type": "Task",
    "Resource": "arn:aws:lambda:us-east-1:123456789012:function:OnFirstMatch",
    "Next": "NextState"
  },
  "SecondMatchState": {
    "Type": "Task",
    "Resource": "arn:aws:lambda:us-east-1:123456789012:function:OnSecondMatch",
    "Next": "NextState"
  },
  "DefaultState": {
    "Type": "Fail",
    "Error": "DefaultStateError",
    "Cause": "No Matches!"
  }
}
"NextState": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
  "End": true
}
```



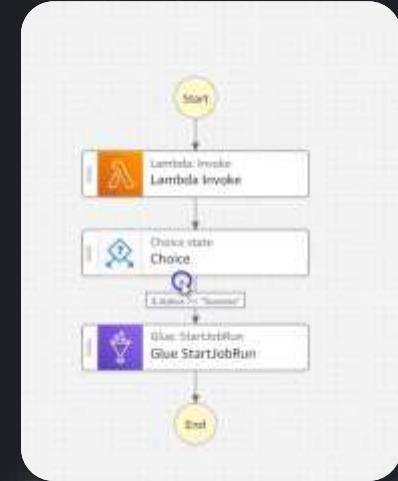
# AWS Step Functions



- **Built-in controls**

⇒ To examine the state of each step in the workflow use

- *Retry*
- *Catch*
- *Timeouts*
- *Parallel*
- *Choice*
- *Wait*
- *Pass*





# AWS Step functions



## AWS SDK Integrations

⇒ Allow Step Functions to execute API actions across more than 200 AWS services directly within a workflow.

## Optimized Integrations

⇒ Optimized integrations are designed specifically for seamless integration with popular AWS services, such as Lambda, S3, DynamoDB, ECS, SNS, and SQS.

## IAM permissions

- State machine needs to have appropriate IAM permissions
  - E.g. execute a Lambda function
    - ⇒ Lambda function might need permission to other services as well





# Standard Workflows



VS

- Used for long-running, durable, and auditable workflows.
- Reliable continuous operation.
- Provides detailed logging of all state transitions.
- Full integration / expensive.
- 2,000 per second execution rate / 4,000 per second state transition rate.

# Express Workflows

- Ideal for high-volume, short-duration, event-processing workloads.
- High throughput.
- 100,000 per second execution rate / nearly unlimited transaction rate.
- Limited integration / cost-effective.
- Fast microservice orchestration





# Amazon EventBridge



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Amazon EventBridge

EventBridge is a serverless service that helps you link different parts of your application using events, making it easier to build apps that can scale and adapt quickly.

**Event = JSON object (change in environment / state)**

## **Components of EventBridge:**

- **Event Producers:** Sources that generate events.
- **Event Bus:** This is the central hub where events are sent.
- **Rules:** These define the match criteria.
- **Targets:** Resources that receive the events

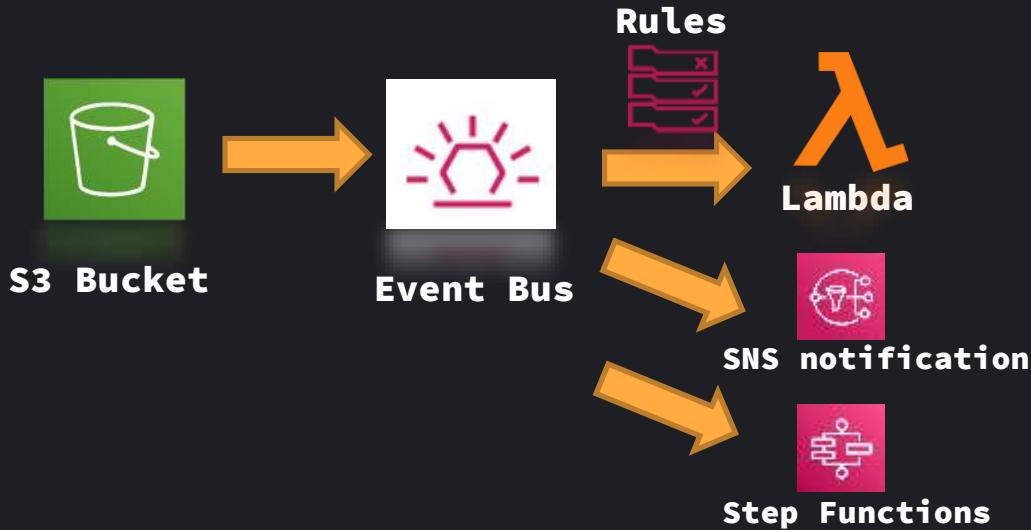




# Amazon EventBridge



- Enables decoupled, event-driven architecture  
⇒ services can interact efficiently without direct integration





# Amazon EventBridge Rules

## Two types of rules

Matching on event

### Event pattern

- Triggers based on specific pattern
- Define event pattern to define events
- When matched ⇒ Send event to target
- E.g. Lambda executes as response to new data in DynamoDB table

On a schedule

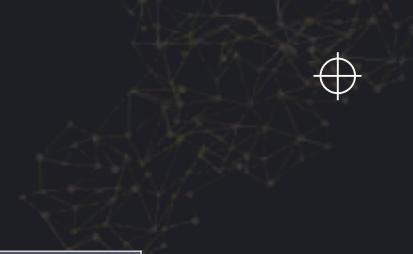
- Sends events to targets at intervals
- E.g. periodically run a Lambda function

```
{  
    "source": "UserRegistrationEvent",  
    "detail": {  
        "Schema": "http://json-schema.org/draft-07/schema#",  
        "Type": "User Registration Event",  
        "Type": "Subject",  
        "Attributes": {  
            "userId": {  
                "Type": "String",  
                "Description": "The unique identifier for the user."  
            },  
            "username": {  
                "Type": "String",  
                "Description": "The user's chosen username."  
            },  
            "password": {  
                "Type": "String",  
                "Description": "The user's chosen password."  
            }  
        }  
    }  
}
```





# Amazon EventBridge – Event buses



Event Bus Type	Purpose
<b>Default Event Bus</b>	Available by default; Automatically receives events from AWS service
<b>Custom Event Buses</b>	Created by user; Specify which events should be received
<b>Partner Event Buses</b>	Created by user; Receives events from integrated SaaS partners





# EventBridge Schema Registry



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# Amazon EventBridge – Schema Registry

Version Management

- Discover, manage, and evolve event schemas
- Includes details about the type of data each event can contain in JSON format
- *Features : Schema Discovery & Code generation*

Schema Sharing

```
{  
    "version": 1,  
    "event_type": "UserRegistrationEvent",  
    "content": {  
        "schema": "http://json-schema.org/draft-07/schema#",  
        "title": "User Registration Event",  
        "type": "object",  
        "properties": {  
            "userId": {  
                "type": "string",  
                "description": "The unique identifier for the user."  
            },  
            "name": {  
                "type": "string",  
                "description": "The name of the user."  
            },  
            "email": {  
                "type": "string",  
                "description": "The email address of the user."  
            }  
        }  
    }  
}
```





# Amazon EventBridge Resource Based Policy



Used to specify direct permissions on event buses.

Key components:

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": "events:PutEvents",
      "Resource": "arn:aws:events:us-east-1:123456789012:event-bus/my-custom-event",
      "Condition": {
        "StringEquals": {
          "aws:sourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Benefits:

- Decentralized Management
- Cross-Account Access
- Granular Access Control





# Scenario Overview

- Understand the scenario
- Solve different tasks
- Make use of the big picture



01

02

03

04

05

06



01

02

03

04

05

06



# Scenario Automating Retailer Data Lake



01

02

03

04

05

06

# Practical Scenario Overview

- Demonstrate the concept
- Acknowledge the advanced nature of the concept
- Improve understanding by simplifying the explanation

01

02

03

04

05

06





# Objective

- 👉 Automate the data processing pipeline
- 👉 Focus on daily retail sales



# Workflow Overview



## STEP 01: Data Ingestion

- CSV files uploaded to an S3 bucket

## STEP 03: State Machine and Lambda Function

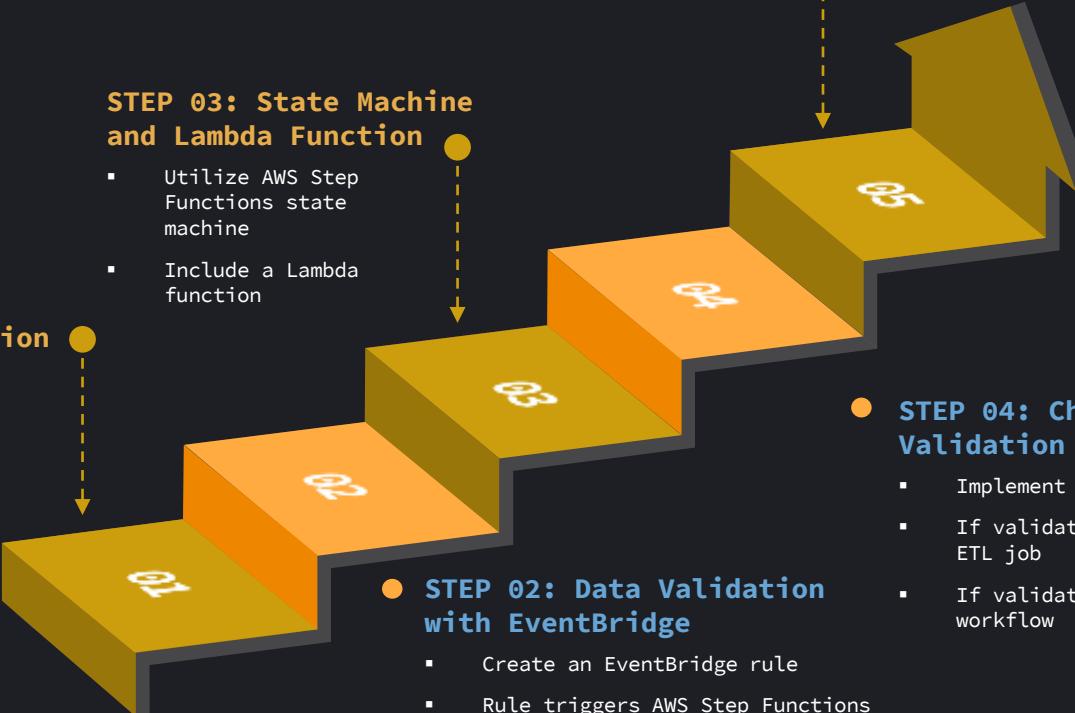
- Utilize AWS Step Functions state machine
- Include a Lambda function

## STEP 02: Data Validation with EventBridge

- Create an EventBridge rule
- Rule triggers AWS Step Functions workflow

## STEP 05: ETL Job Execution

- Successful validation triggers the execution
- ETL job performs further data transformation





# Outcome

Automation of the workflow

Reduction of manual overhead

Quick generation of daily reports.

Ensured data quality





# Implementation Steps

## Step 1 Designing the Workflow:

- Outline the sequence
- Visualize the workflow

## Step 2 Setting up Data Ingestion:

- Configure S3 bucket for sales data uploads.
- Define the structure for CSV files.

## Step 3 EventBridge and Rule Creation

- Create an EventBridge rule
- Set up the rule to trigger the AWS Step Functions workflow

## Step 4 AWS Step Functions State Machine

- Create a state machine
- Define the logic and flow





# Implementation Steps

## Step 5 Lambda Function for Data Validation

- Develop a Lambda function
- Check for expected header names and handle success/failure conditions

## Step 6 Choice State for Validation

- Implement a choice state in the state machine.
- Define conditions for proceeding to ETL or ending the workflow.

## Step 7 ETL Job Configuration

- Set up the ETL job for further data transformation.
- Ensure it is triggered upon successful validation.

## Step 8 Monitoring and Optimization

- Utilize CloudWatch for monitoring the workflow.
- Optimize the workflow for efficiency and performance.





# AppFlow



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# AppFlow

## Purpose

- **Secure Data Movement:** Enables **safe data transfer** between SaaS apps and AWS services.
- **SaaS Integration:** Simplifies **connections with popular SaaS applications** *Salesforce, Snowflake, Slack etc.*
- **No-Code Interface:** Allows for **easy setup** without coding, thanks to a **user-friendly UI**.
- **Pre-built Connectors:** Offers connectors for **quick integrations** with common SaaS platforms.

## SaaS Integration

# AppFlow

Configure flow

Source details

Source name: Choose data source

G |

- Analytics**  
Analytics describes a business intelligence solution for collecting and analyzing data from any digital platform in the ecosystem.
- AfterShip**  
AfterShip is a SaaS application that helps e-commerce store and marketplace sellers to track their shipping status in their store.
- Amazon ES**  
Amazon Simple Storage Service (Amazon S3) is a service that provides object storage through a web service interface.
- Analytics**  
Analytics is a cloud analytics platform that helps businesses track user engagement with the help of real-time analysis.
- Asana**  
Asana is a task-management platform used by individuals and teams for tasks, lists, and discussions in one place.
- Barcode4All**  
Barcode4All is an AI-driven, SaaS API application for serial and logistic asset management.
- Blazemind Retail's Edge**  
Blazemind Retail's Edge (BRE) is a CRM software for multi-channel businesses.
- Blazemind**  
Blazemind is a company that provides digital processing platforms.
- CloudCV**  
CloudCV is a machine-vision platform that processes imagery.
- Coupa**  
Coupa is a business spend management software solution.
- Databox**  
Databox is a marketing platform for cross-channel applications, providing reporting of adwords, analytics, stats, and metrics.
- Dashgrid**  
Dashgrid is a cloud-based service delivery platform that allows users to gather feedback from end-users and target audiences.
- DocuSign Monitor**  
DocuSign Monitor provides real-time visibility, insight, and metrics measured through DocuSign.
- Domo**

Source



- Amazon CloudWatch**  
Amazon CloudWatch is a serverless control that monitors and implements activity logs.
- Amazon Honeycomb**  
Amazon Honeycomb is a fully managed service that allows you to easily store, analyze, and build data-oriented programming.
- Amazon RDS for PostgreSQL**  
Amazon Relational Database Service (Amazon RDS) is a managed relational database service.
- Amazon Redshift**  
Amazon Redshift is a fast, fully managed, petabyte-scale data warehousing service.
- Amazon S3**  
Amazon Simple Storage Service (Amazon S3) is a service that enables efficient storage through a web service interface.
- Google BigQuery**  
Google BigQuery is a massive data warehouse.
- Google Sheets**  
Google Sheets is a spreadsheet application.
- HubSpot**  
HubSpot is a platform for CRM, sales, and marketing automation.
- IEXC Stock Data Scale**  
IEXC is a managed endpoint for stock market applications.
- Market**  
Market is a company specializing software for account-based and other marketing services and products.
- SAP Data**  
SAP provides enterprise software for financial, business operations and customer-centricity. The company is majority-owned by the SAP Group.
- Salesforce**  
Salesforce is a customer relationship management (CRM) solution that provides a single, shared view of leads & customers.
- Smartabase**  
Smartabase is a software as a service offering for collaboration and work management.
- Snowflake**  
Snowflake is a cloud data warehouse that provides secure data sharing across multiple clouds.
- Upstream**  
Upstream is a cloud-based software that provides secure data sharing across multiple clouds.

Destination

# AppFlow

Configure flow.

**Source details** info

Source name: **Amazon S3**  
Amazon Simple Storage Service (Amazon S3) is a service that provides object storage through a web service interface.

Bucket choice: Only choose this if your Lambda function is running in Lambda@Edge. The bucket can then include all Lambda resources for execution, or define constraints. The source location must contain all buckets that are used by this Lambda function.

Choose an S3 bucket: **CloudWatchLogs**

Data format preference: You can specify your preferred format for reading data from Amazon S3.  
 CSV format  
 JSON Lines format

**Destination details** info

Destination name: **Google Sheets**  
Google Sheets is a web-based spreadsheet application.

Google Sheets connection: info

# AppFlow

## Features

- **Data Transformation:**  
Provides **mapping and transformations** for compatible data formats.
- **Bi-Directional Sync:**  
Supports **two-way data synchronization**.
- **Event-Driven Flows:**  
Can initiate transfers based on **SaaS application events**.
- **Encryption & Security:**  
Ensures data is **encrypted** and managed with **AWS IAM** policies.

# AppFlow

## Use Cases

- **Analytics & Insights:**  
Facilitates data aggregation for **in-depth analytics**.
- **Customer Data Sync:**  
Provides a **unified customer data view** across systems.
- **Workflow Automation:**  
Enables **automated interactions** between SaaS and AWS services.



# Amazon MWAA

**Amazon Managed Workflows for Apache Airflow**



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon Managed Workflows for Apache Airflow

- Fully managed orchestration service using Apache Airflow
- Create and manage workflows using Python code

## Key Features:

- Automatic scales workflow execution
- Integration with IAM
- Integration with CloudWatch

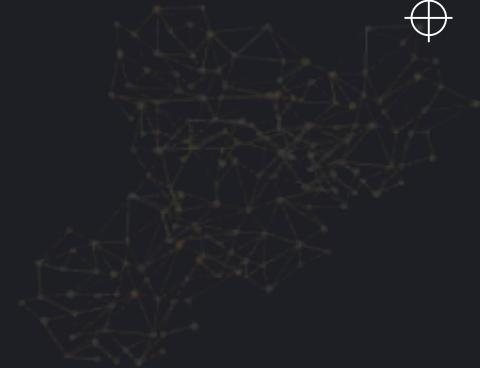
## Use Cases:

- Complex workflows to process of data
- Automate orchestration of machine learning pipelines
- ETL: Extract , Transform and Load
- Event-driven Automation





# Directed Acyclic Graphs (DAGs)



- DAGs to organize workflows into clear and logical order
- Write them in Python
- DAGs represent workflows as collections of tasks with dependencies.
- **Execution sequence:**  
Extract data, Transform Data, Load data

```
from airflow import DAG
from airflow.operators.dummy_operator import DummyOperator
from airflow.operators.python_operator import PythonOperator
from datetime import datetime

def print_hello():
    print("Hello Airflow!")

default_args = {
    'owner': 'airflow',
    'start_date': datetime(2022, 1, 1),
}

with DAG(dag_id='hello_world_dag', default_args=default_args, schedule_interval=None) as dag:
    start = DummyOperator(task_id='start')
    hello_task = PythonOperator(task_id='hello_task', python_callable=print_hello)
    end = DummyOperator(task_id='end')

    start >> hello_task >> end
```





# Amazon SNS



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Amazon SNS

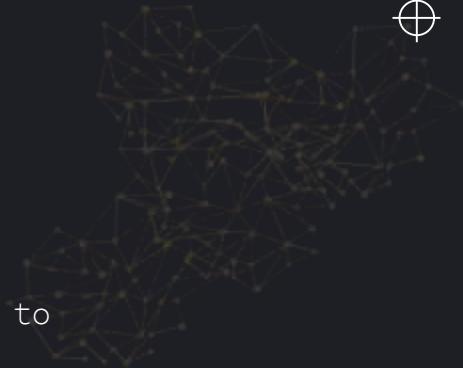


- **Fully-managed pub/sub messaging Service.**
- Topics for high throughput  
⇒ *push-based, many-to-many messaging.*
- Large number of subscriber endpoints  
**SQS queues, AWS Lambda functions, HTTP/s, Emails, etc.**





# Key features of amazon SNS



- **Topics:**

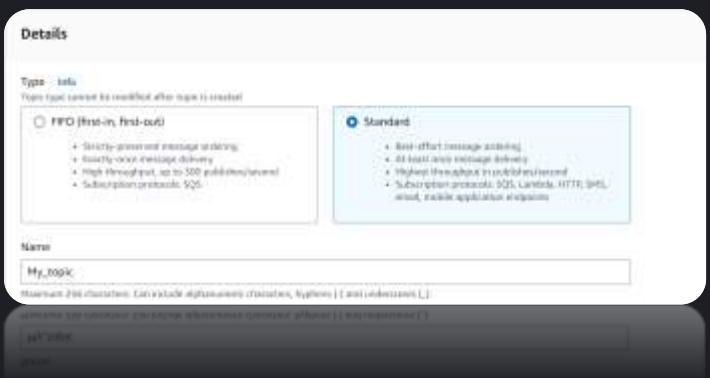
Publish/subscribe mechanism that allows messages to be pushed to subscribers. There are two types:

- ⇒ Standard topics:

- o *High throughput*
- o *At least once delivery*
- o *Unordered delivery*
- o *Multiple subscribers*

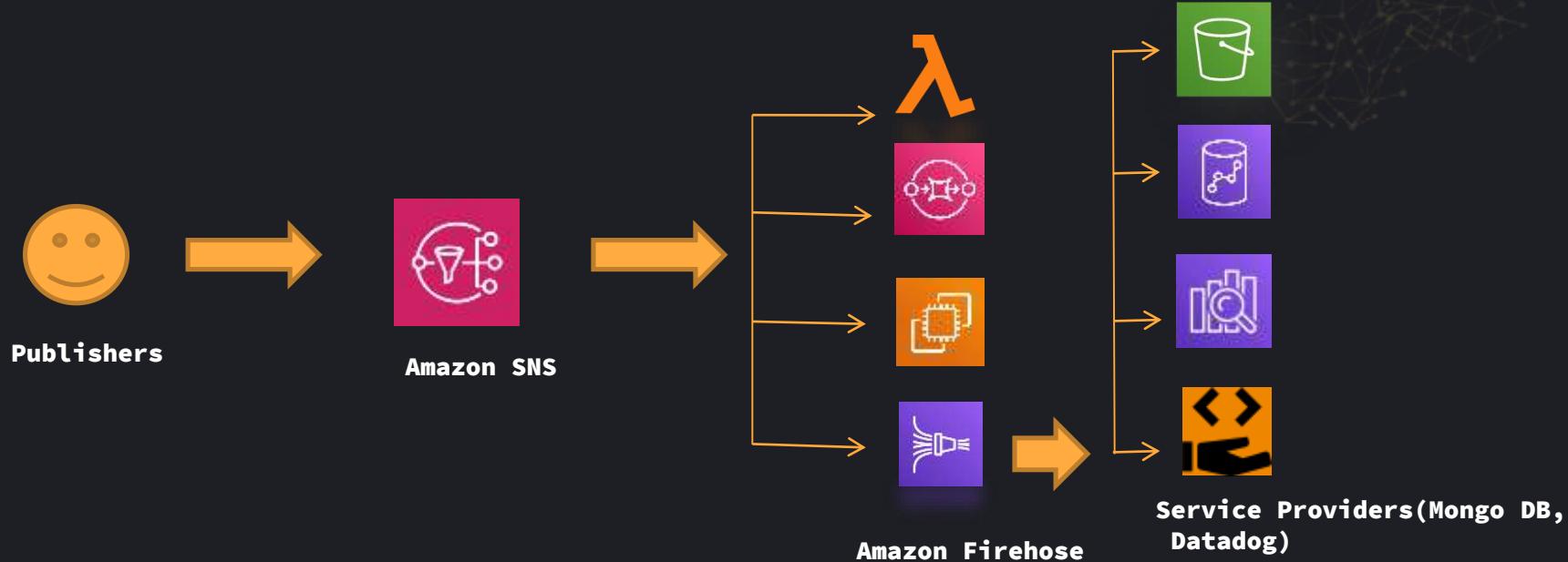
- ⇒ FIFO (first-in, first-out)

- o *Ordered delivery*
- o *Exactly one processing*
- o *Message Grouping*
- o *Limited throughput*
- o *Deduplication*
- o *Use with SQS FIFO Queues*



# Amazon SNS: Application to Application (A2A)

- A2A enhances flexibility and integration possibilities
- Fanout pattern



- Ideal for serverless and microservices

# Amazon SNS: Application to person (A2P)

- Directly delivered to people





# Amazon SNS How to Publish



## Topic Publish

- Send a message to an SNS topic
- Topic acts as a channel
- Automatically gets delivered

## Direct Publish

- Targets a specific endpoint
- Push notifications
- Intended for a specific subscriber or device





# Amazon SQS



003-1040559

1250 003-77156.8

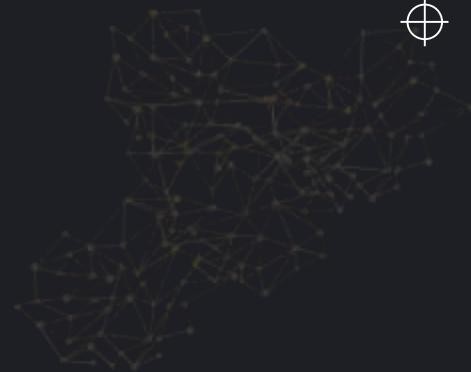
1760 0009-14563.7

73273





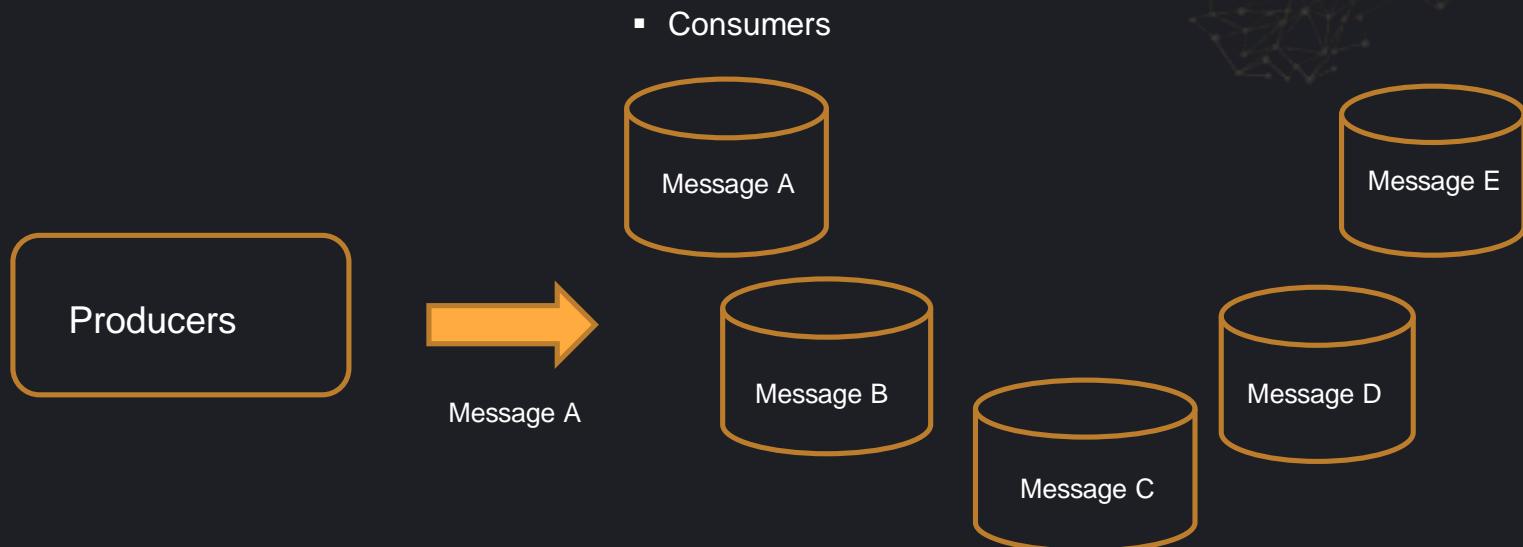
# Amazon SQS



- Fully-managed message queuing service.
- Provides a reliable, highly scalable hosted queue.
- Allows to decouple and scale microservices.
- Enables asynchronous processing.



# SQS Message Lifecycle

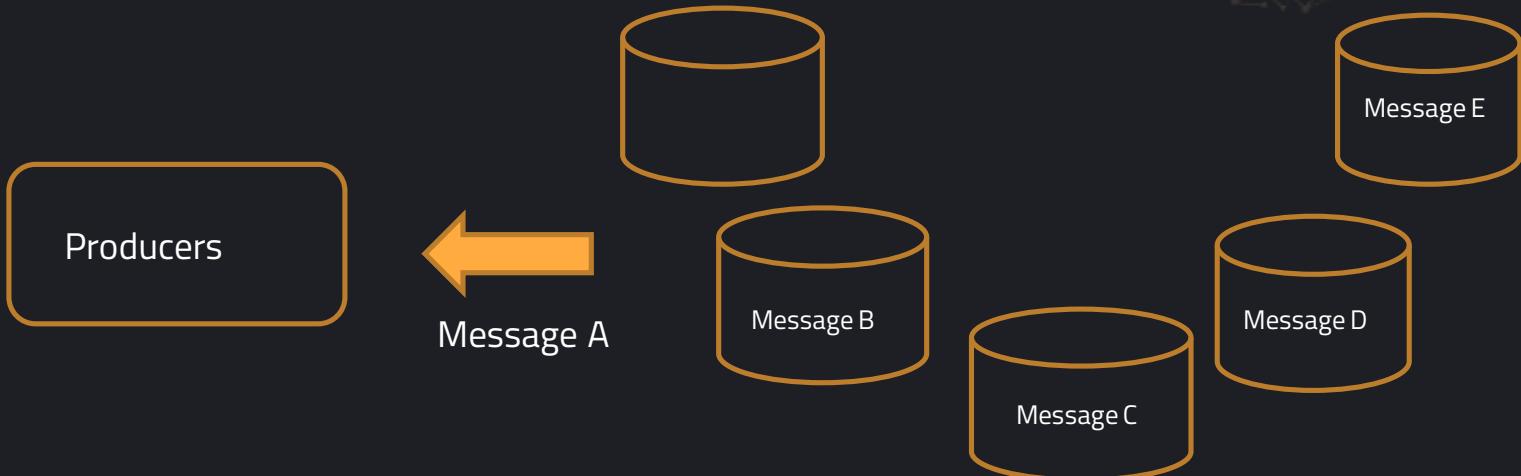


Queue distributed on SQS Servers

# SQS Message Lifecycle



Customer must delete message





# Standard Queues

- **High Throughput.**
- **At-Least-Once Delivery.**
- **Best-Effort to order.**
- **Use Cases: Speed and high performance.**

# FIFO Queues

- **Ordering Guarantee.**
- **No Duplicates.**
- **Standard Throughput.**
- **Use Cases: Order of operation is critical.**

VS





# Amazon SQS

- Decouple application components.
- Data retention for up to 14 days.
- Scales automatically with the number of messages.

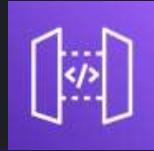
# Amazon Kinesis Data Streams

- Real-time data streaming.
- Extendable retention for up to 365 days (24 hours by default).
- Provides ordering of records within a shard (partition).

VS



# Amazon API Gateway



003-1040559

1250 003-77156.8

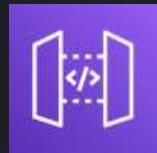
1760 0009-14563.7

73273

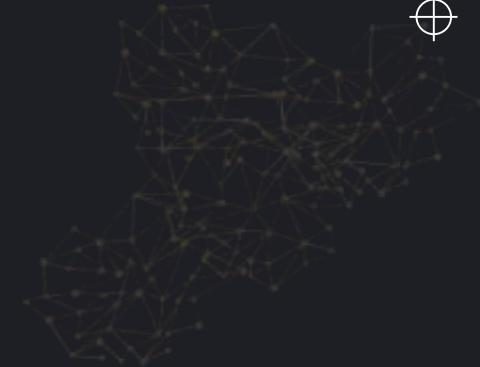




# Amazon API Gateway



- **Fully managed service to create, publish, maintain, monitor and secure APIs at scale.**



## REST APIs

- Standard REST architecture
- Integration with various endpoints
- Allows CRUD operations

The screenshot shows the AWS Lambda API Gateway interface. It displays two card-like components:

- WebSocket API**: Described as "Build a bidirectional API using persistent connections (no load) where web clients such as chat applications can communicate". It lists "Requires the following Lambda, API Gateway Services" and includes "Import" and "Create" buttons.
- REST API**: Described as "Develop a REST API where you can complete control over the request and response along with API management capabilities". It lists "Requires the following Lambda, API, API Gateway Services" and includes "Import" and "Create" buttons.

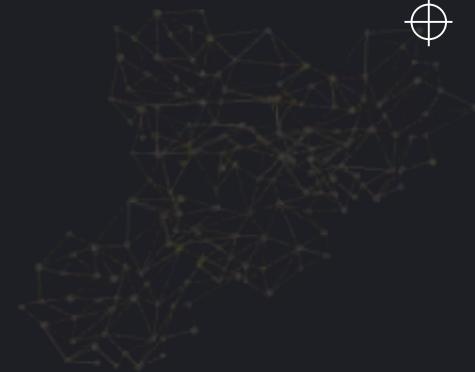
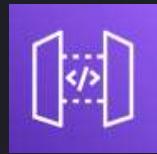
## WebSocket APIs

- Real time communication between client and server
- Ideal for chat apps, live updates and notifications





# Amazon API Gateway



## HTTP APIs

- Create RESTful APIs with lower latency and costs
- Support OpenID Connect and OAuth 2.0
- Built-in support for CORS
- Automatic deployments

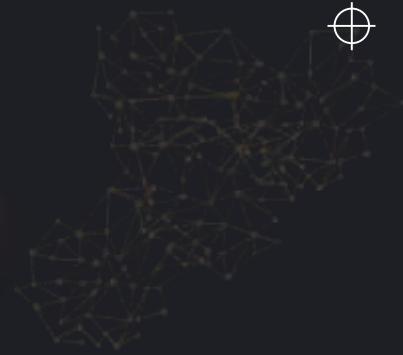


**CORS:** web browsers security feature to control cross-origin requests.





# Amazon API Gateway integration



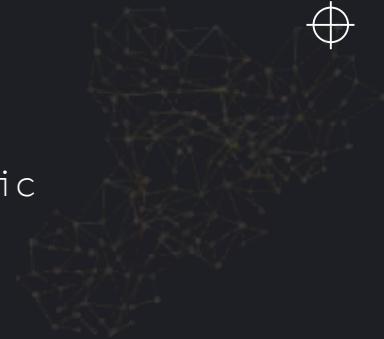
- **Direct integration with other AWS services** like Lambda, DynamoDB, S3, SNS, etc.
- **HTTP integration**  
Forward requests to endpoints  
Suitable for microservices
- **Mock integration for testing**
- **AWS Proxy integration**  
Passes requests in a JSON format





# Throttling, Quotas, and Rate Limiting

Mechanisms used in API Gateway to control and manage traffic effectively.



## Throttling:

- Prevents API saturation due to traffic spikes
- Set limits on number of requests per second

## Quotas:

- Limits requests for a specific period of time
- Must use plans and API keys for implementation

## Rate Limiting:

- Combines throttling and quotas to control requests acceptance





# API Gateway Endpoint Types



## Edge Optimized

- ✓ Reduce latency for clients globally
- ✓ Ideal for worldwide reach
- ✓ URL: <https://abc123.execute-api.amazonaws.com/prod>

## Regional Endpoint

- ✓ Does not use CloudFront caching.
- ✓ Ideal for a specific region.
- ✓ URL: <https://abc123.execute-api.us-west-2.amazonaws.com/prod>

## Private Endpoint

- ✓ Accessible only within VPC
- ✓ Prevents exposure to public internet
- ✓ Public <https://vpce-12345-abc.execute-api.us-east-1.vpce.amazonaws.com/prod>
- ✓ In-region endpoint: <https://vpce-12345-abc.vpce.amazonaws.com/prod>

01

02

03

04

05

06



01

02

03

04

05

06



# Section 16:

# **Management, Monitoring, Governance**





# AWS CloudFormation



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273



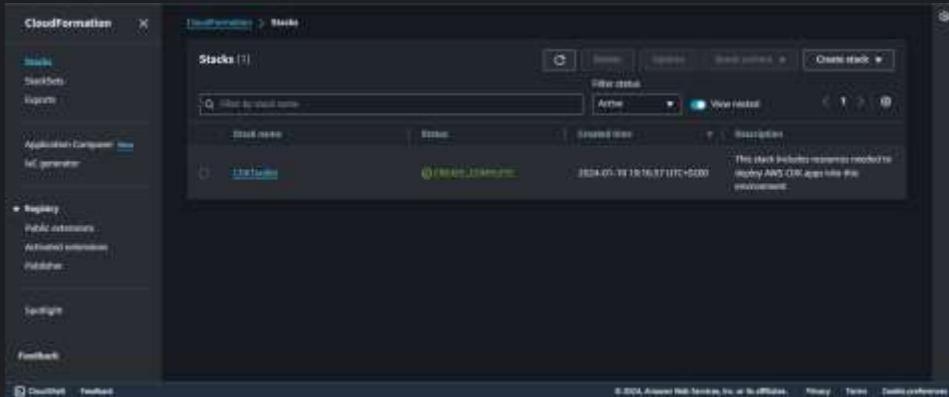


# AWS CloudFormation



## Infrastructure Management

- Allows you to define and provision your AWS infrastructure as code
- Use cases:
  - Replicate infrastructure across regions
  - Control and track changes to your infrastructure
  - Simplify infrastructure management



CloudFormation interface





# AWS CloudFormation Templates



Text files that describe the desired state of your AWS infrastructure.

```
1
2 Resources:
3   FirstS3Bucket:
4     Type: AWS::S3::Bucket
5     Properties: {}
6
7
```



YAML Template to create the “FirstS3Bucket” S3 Bucket





# AWS CloudFormation Stacks



A collection of AWS resources created and managed as a single unit

## HOW IT WORKS





# AWS CloudWatch



003-1040559

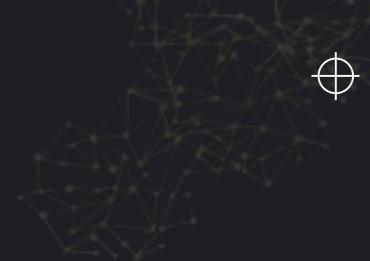
1250 003-77156.8

1760 0009-14563.7 73273



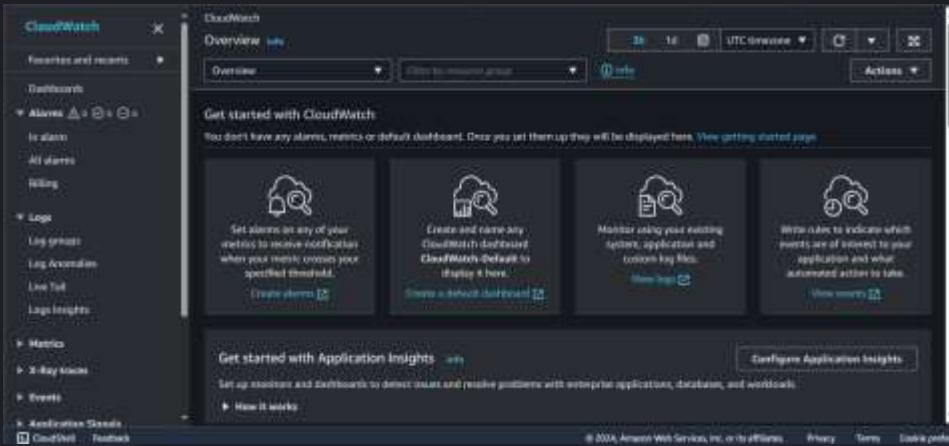


# AWS CloudWatch



Main Monitoring Service

- AWS' proprietary service for monitoring Applications and Resources in real-time
- Metrics are displayed on a Dashboard



CloudWatch interface



# AWS CloudWatch Metrics



A metric is a numbers to monitor



- **Features:**

- **Name Spaces:** Serves as a container for CloudWatch metrics.
- **Time Stamps:** Every metric should be linked to a timestamp.
- **Dimensions:** Key/value pair belonging to a metric.
- **Statistics:** Aggregated data metrics over defined time intervals.
- **Period:** The length of time associated with a specific statistic.
- **Resolution:** Level of detail you can see in your data
  - Standard Resolution: data has a one-minute granularity
  - High Resolution: data has a granularity of one second





# AWS CloudWatch Metric Streams

- Continuously stream metrics to both AWS & 3rd-party destinations near-real time.
- Kinesis Firehose is used to stream data to AWS destinations.

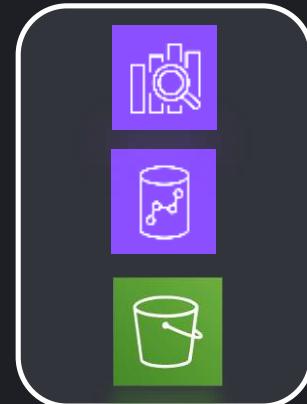
## Custom setup with Firehose



## Quick S3 setup



## Quick AWS partner setup





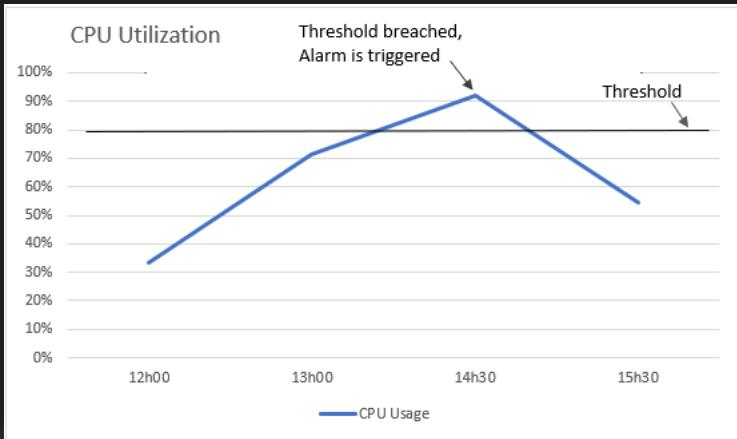
# AWS CloudWatch Alarms



Monitor metrics and trigger actions when defined thresholds are breached.

- **Types of Alarms:**

- **Metric Alarm:** Monitors a single metric
- **Composite Alarm:** Monitors the state of other alarms





# AWS CloudWatch Alarms



- **Alarm States:**

**OK**

- Metric is within the defined threshold

**ALARM**

- Metric is over the defined threshold

**INSUFFICIENT\_DATA**

- Not enough data available to determine the alarm state

- **Alarm Actions:** Actions an alarm can take when it changes state

- ❑ Amazon SNS: Trigger email, SMS, or push alerts.
- ❑ EC2 Actions: Stop, terminate or reboot EC2 instance.
- ❑ Auto Scaling: Adjust instance count based on load.
- ❑ Lambda: Execute functions for automation.
- ❑ Incident Management: Create urgent incident tickets.





# AWS CloudWatch Logs

Collects and consolidates logs from various sources

- **Centralized Logging:** From different services in one location
- **Real-Time Monitoring:** Real-time monitoring of log data





# AWS CloudWatch Logs

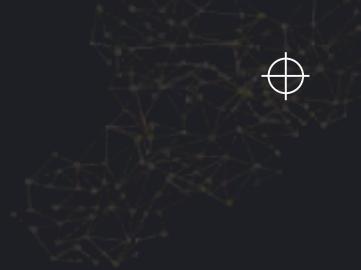
Collects and consolidates logs from various sources

- **Log streams:** Sequences of log events from a single source
- **Log groups:** Logical containers for log streams
- **Log events:** Records of an activity logged by an application
- **Retention Policy:** Allows you to allocate the period a log is retained
- **Log Insights:** Interactive log analysis tool for querying and visualizing log data stored in CloudWatch Logs
- **S3:** Logs can be sent to S3, Kinesis Data streams/Firehose, and Lambda.



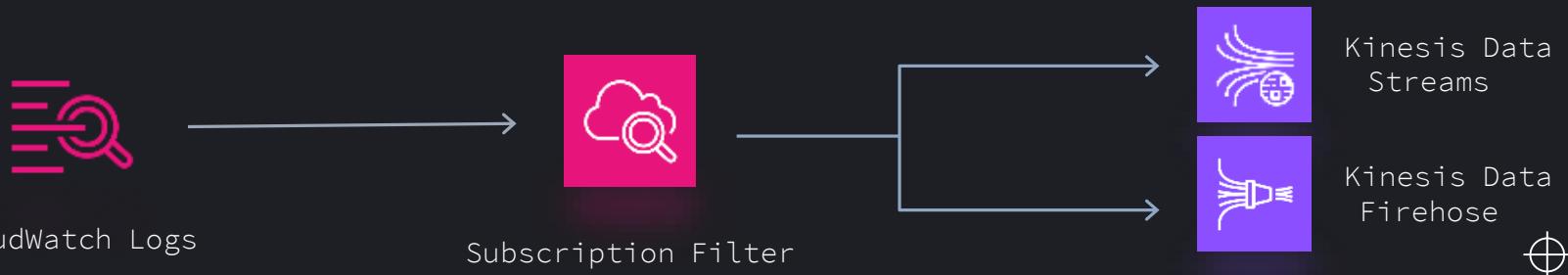


# AWS CloudWatch Log Filtering Subscription



Filter log data using a Metric or Subscription filter before sending it to a destination.

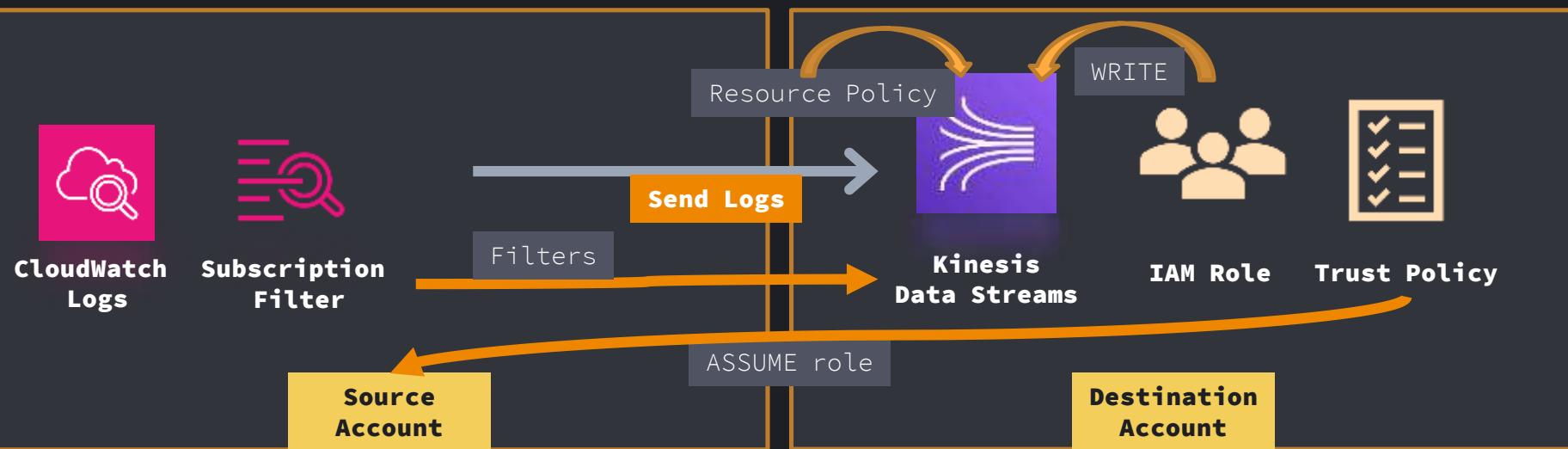
- **Metric filter:** Extract data from log events to create custom metrics
- **Subscription filter:** Filter log data being sent to other AWS services





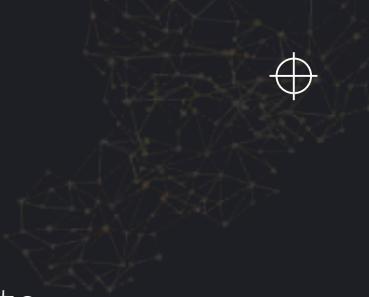
# Cross Accounts Access

- 1) Setup Data Stream in Destination Account
- 2) Create IAM role + Trust Policy in Destination Account to write to Stream
- 3) Setup Subscription Filter in Source Account





# AWS CloudWatch Logs Agent



- EC2 does not send any data to CloudWatch, to send its logs to CloudWatch – a logs agent is needed.
- A CloudWatch Logs Agent is a lightweight, standalone agent that can be installed on EC2 instances/On-prem servers
- It collects and streams log data from EC2 instances/On-prem servers to CloudWatch Logs in near real-time.

- **Types of log agents:**

- CloudWatch logs agent
- CloudWatch Unified logs agent





# AWS CloudWatch Logs Agent

## Logs Agent

- o Older version with limited capabilities
- o Collects logs only

## Unified CloudWatch agent

- o Enhanced logs agent
- o Collects logs as well as system-level metrics
- o Collects RAM, CPU Utilization, Memory Usage, Disk Space, Network Traffic, and Swap Space metrics

**Note:** AWS mostly recommends **Unified CloudWatch agent**



# AWS CloudTrail



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# AWS CloudTrail



Audit and Governance  
Tool

- Records all activities that take place within your AWS account
- Activities are recorded as events
- Enabled by default

The screenshot shows the AWS CloudTrail Insights interface. On the left, there's a sidebar with navigation links: CloudTrail, Event history, Insights, Logs, Dashboard, Quicksight, Event data store, Integrations, Tools, Settings, Metrics, Documentation, Issues, FAQs, and Help. The main area has a header "CloudTrail Insights" with a sub-header "Event history". Below it, a message says "CloudTrail Insights is not enabled. Insights are events that show unusual API activity. After you enable insights, if unusual activity is logged, insights events are shown in this table for 90 days. Additional charges apply. Learn more". A table titled "Event history" lists five recent events:

Event name	Event time	Event source
Backup created	April 23, 2024, 23:06:27 UTC+00:00	backup.amazonaws.com
Backup completed	April 23, 2024, 00:13:47 UTC+00:00	backup.amazonaws.com
Backup failed	April 23, 2024, 00:13:50 UTC+00:00	backup.amazonaws.com
Backup deleted	April 23, 2024, 07:01:08 UTC+00:00	backup.amazonaws.com

At the bottom, there are links for "View all event history", "CloudTrail", "Feedback", and copyright information: © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences.

CloudTrail interface

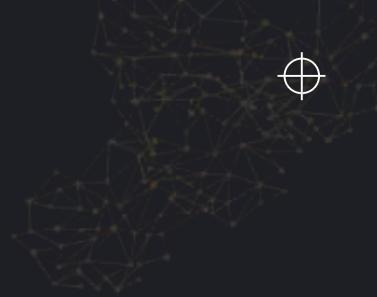




# AWS CloudTrail Events



Record(s) of activities



- **Types of Events**

- **Management Events:** Captures high-level operations
- **Data Events:** Captures data-level operations
- **Insight Events:** Captures unusual activity

**Events History**

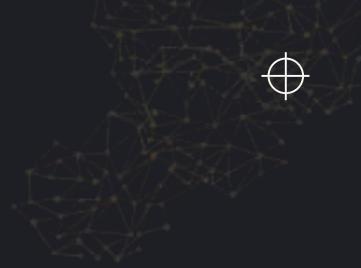


view management events of the past  
90 days





# AWS CloudTrail Trails



Captures records of AWS activities and stores in S3

- **Trail Types:**

- Multi-Region***

- Trail applies to all regions

- Single Region***

- Trail applies to one region

- Organizational***

- Logs events for all accounts in an organization

- **Features:**

- Multiple Trails Per region creates multiple trails within a single AWS region.





# AWS CloudTrail Lake



A Managed data lake for AWS user and API activity

- **Lake Channels:**

Integration with outside sources

Used to ingest events from external sources

Service Linked

AWS services create channels to receive CloudTrail events





# AWS CloudTrail Extras



- CloudTrail allows for deep analysis of event
- Create rules with EventBridge if needed





# AWS Config



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

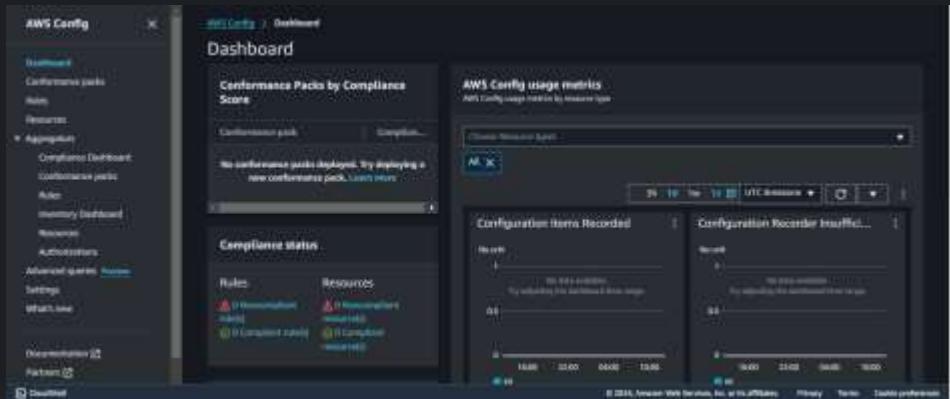




# AWS Config

# Centralized Configuration Management

- Assess, audit, and evaluate the configurations of your AWS resources
  - Disabled by default
  - Generates a configuration item for each resource.



## Config interface





# AWS Config Concepts



- **Configuration Item** is the current state of individual AWS resources
- **Configuration Recorder** stores configuration items for resources in your account.
- **Configuration History** is a historical record of configuration changes
- **Configuration Snapshot** is a collection of configuration items
- **Configuration Stream** is an automatically updated list of configuration items for resources recorded by AWS Config.
- **Conformance packs** bundles Config rules, remediation actions, and required AWS resource configurations into a single, reusable package.
- **Discovery** discovers resources in your AWS environment
- **Advanced queries** analyzes real-time and historical resource configurations
- **Resource Relationship** creates a map of relationships between AWS resources.





# AWS Config Rules



Evaluates the compliance of your AWS resources against desired configuration

## *Evaluation results for a Config rule:*

**Compliant** - Resource complies with the rule

**Non-compliant** - Resource does not comply with the rule

**Error** - Invalid required or optional parameters

**Not Applicable** - Filters out resources that the logic of the rule cannot be applied to

- **Types of Rules**
  - AWS Config Managed Rule
  - AWS Config Custom Rule





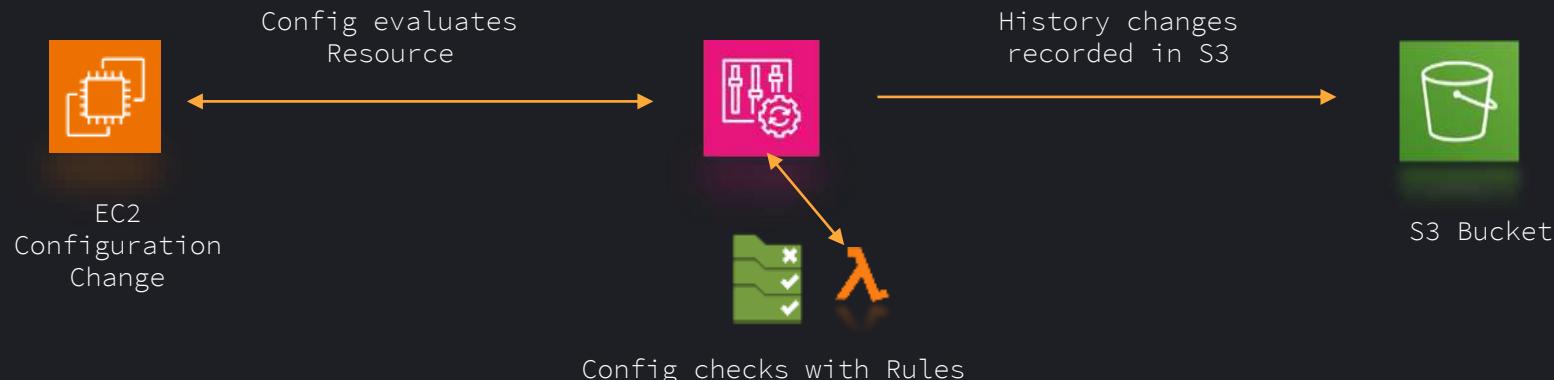
## AWS Config Managed Rules

- Pre-defined and customizable rules created by AWS Config.

## AWS Config Custom Rules

- Rules you create from scratch.
- Created with lambda functions or Guard

VS





# AWS Config Trigger Types



Determines when AWS Config evaluates the rules against your resources.

- **Trigger Types**
  - Configuration changes: A configuration change is detected
  - Periodic: Evaluates at specified intervals
  - Hybrid: Evaluates resource configuration change and chosen frequency

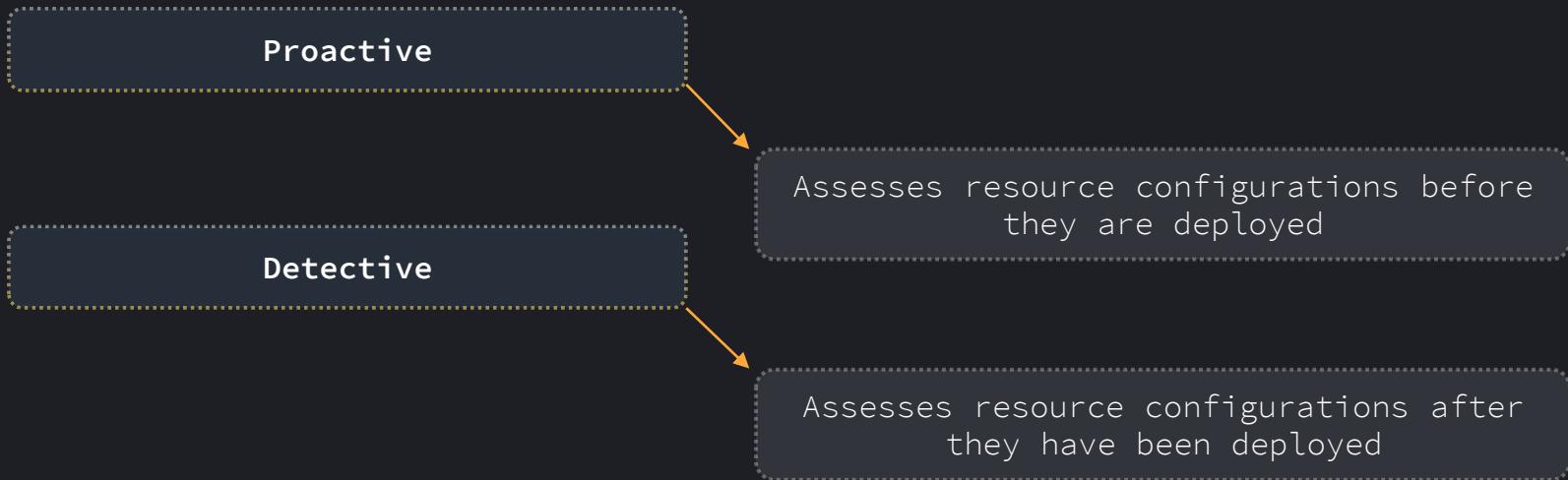




# AWS Config Evaluation Modes



- Define when and how resources are evaluated during the resource provisioning process.
- **Evaluation Modes:**





# AWS Config Multi-Account Multi-Region Data Aggregation



Aggregate and centrally manage AWS Config data across multiple AWS accounts and regions

- **Concepts**
  - **Aggregator:** Collect Config configuration and compliance data from multiple source accounts and regions.
  - **Source Account:** AWS accounts where AWS Config records configuration changes and compliance data for resources
  - **Aggregator Account:** Central hub for aggregating configuration and compliance data from multiple source accounts
  - **Authorization:** Permission granted to an aggregator Account to collect data.





# AWS Well-Architected Framework





# AWS Well-Architected Framework



- **Operational Excellence:**

Manage operations to deliver business value and continuously improve processes.

- **Security:**

Protect data and systems; manage access, and respond to security events.

- **Reliability:**

Ensure systems perform as expected, handle changes in demand, and recover from disruptions.

- **Performance Efficiency:**

Use resources efficiently, adapt to changing needs, and leverage new technologies.

- **Cost Optimization:**

Reduce and control costs without sacrificing performance or capacity.





# AWS Well-Architected Framework



- **Sustainability**

Minimize environmental impact by efficiently using resources and reducing carbon emissions.





# AWS Well-Architected Tool



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Well-Architected Tool

Architecture Optimization Tool

- Review workloads against the Well-Architected Framework.
- Questionnaire-Based Assessment



Well-Architected Tool interface



# AWS Well-Architected Tool Features

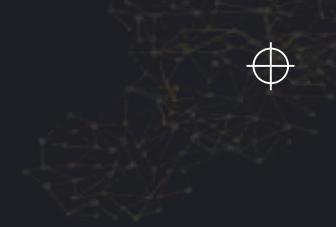


- o **Workload** = Collection of components that add to business value.
- o **Milestones** = crucial stages in your architecture's evolution throughout its lifecycle
- o **Lenses** = Evaluate your architectures against best practices and identify areas of improvement.
  - Lens Catalog (created & maintained by AWS)
  - Custom lenses (user-defined lenses)
- o **High-Risk Issues(HRIs)** = Architectural and operational choices that may negatively impact a business.
- o **Medium risk issues (MRIs)** = Architectural and operational choices that may negatively impact a business but not to the same degree as HRIs.





# AWS Well-Architected Tool Extras



## Use Cases

- Continuously improve architectures
- Get architectural guidance
- Enable consistent governance

## How it Works

- Define the workload
- Review the workload
- Tool returns feedback





# AWS Identity and Access Management (IAM)



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273

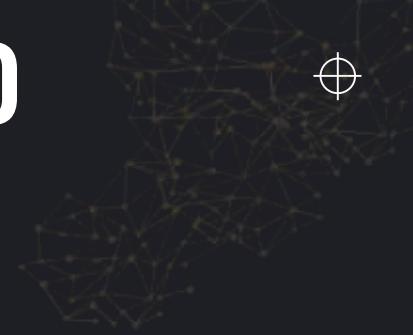




# AWS Identity Access Management (IAM)



Centrally manage access & permissions



## Users

Identities that we can attach permissions to

## Groups

Collections of users

## Roles

Collection of permissions that can be assumed by identities

## Policies

Definition of permissions





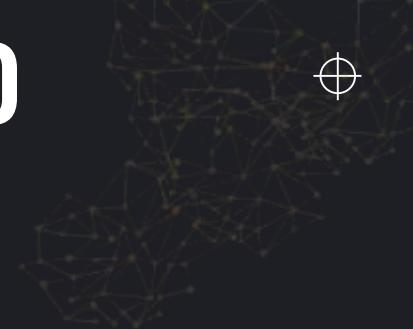
# AWS Identity Access Management (IAM)



Centrally manage access & permissions

## Users

Identities that we can attach permissions to



- **Principle of Least Privilege:**

No access per default

Only grant specific access to what is needed



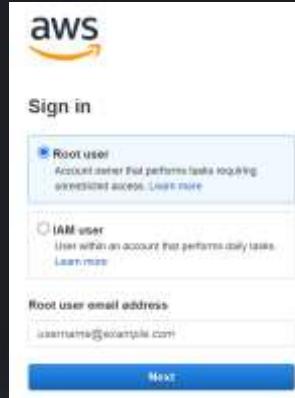


# IAM - Users

## Types of users:

- **Root User:**

Initial user with full access and services  
Intended for account set up and emergency



- **Standard IAM users:**

Unique set of credentials  
Direct access to AWS resources

- **Federated users:**

Authenticated through external identity providers:





# IAM - Groups



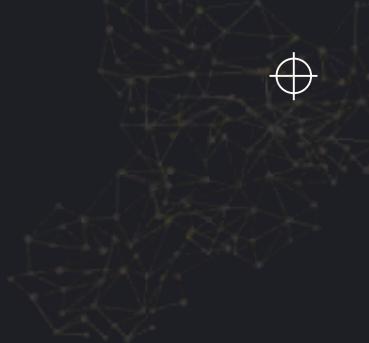
## Groups:

- A collection of users managed as a single entity
- Assign policies to group => all users inherit permissions
- A user can belong to multiple groups
- No credentials associated



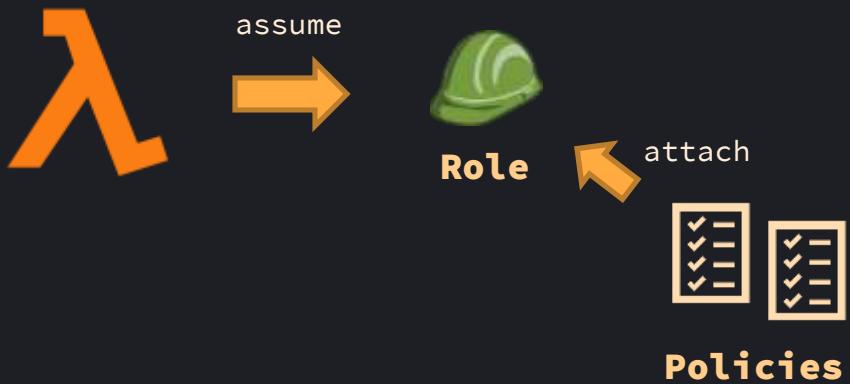


# IAM - Roles



## Roles:

- A combination of permissions that can be assumed
- Attach policies to role
- Role can then be assumed by identities
- Services need to assume roles to perform actions





# IAM - Policies

Policies are documents that define permissions for IAM entities

## Managed policies:

- Centrally managed standalone policies

### AWS Managed policies:

Created and managed by AWS

### Customer Managed policies:

Created and managed by users

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:Get*",  
                "s3>List*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```



# IAM - Policies

## Inline policies:

- Attached only to a single IAM user
- Non-reusable





# IAM - Policies

## Identity-based policies:

- Associated with IAM identities
- Determine what actions can be performed
- Effective to grant identity permissions across different services and resources

## Resource-based policies:

- Attached to a resource instead of IAM identity
- Grant or deny permissions on the resource
- Inline policy only





# IAM – Trust Policy



Define which entities (accounts, users, or services) are allowed to assume a role.

- Type of resource-based policy for IAM roles

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "AWS": "arn:aws:iam::111122223333:root"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

- Used for example for cross-account access





# Section 17: **Containers**



003-1040559      1250 003-77156.8

1760 0009-14563.7      73273





# Introduction to Docker Containers



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# What is Docker?



- Is a platform for developing, shipping, and running applications in containers
- Packages applications and their dependencies into standardized units called containers.
- Is used to quickly deploy and scale applications.

# Why Use Docker Containers?



- Consistency : Applications run consistently across different environments.
- Isolation : Containers are isolated from each other.
- Portability : Containers can be easily moved between different systems and environments.
- Scalability : Enables you to easily scale up or down the number of containers as needed.

# Docker Use Cases



- Microservices Architecture
- Continuous Integration and Continuous Deployment (CI/CD)
- Hybrid Cloud Environments
- Big Data and Analytics



# Docker Components

- Docker Engine
- Docker Image
- Docker Containers
- Docker Registry



# Docker Registries



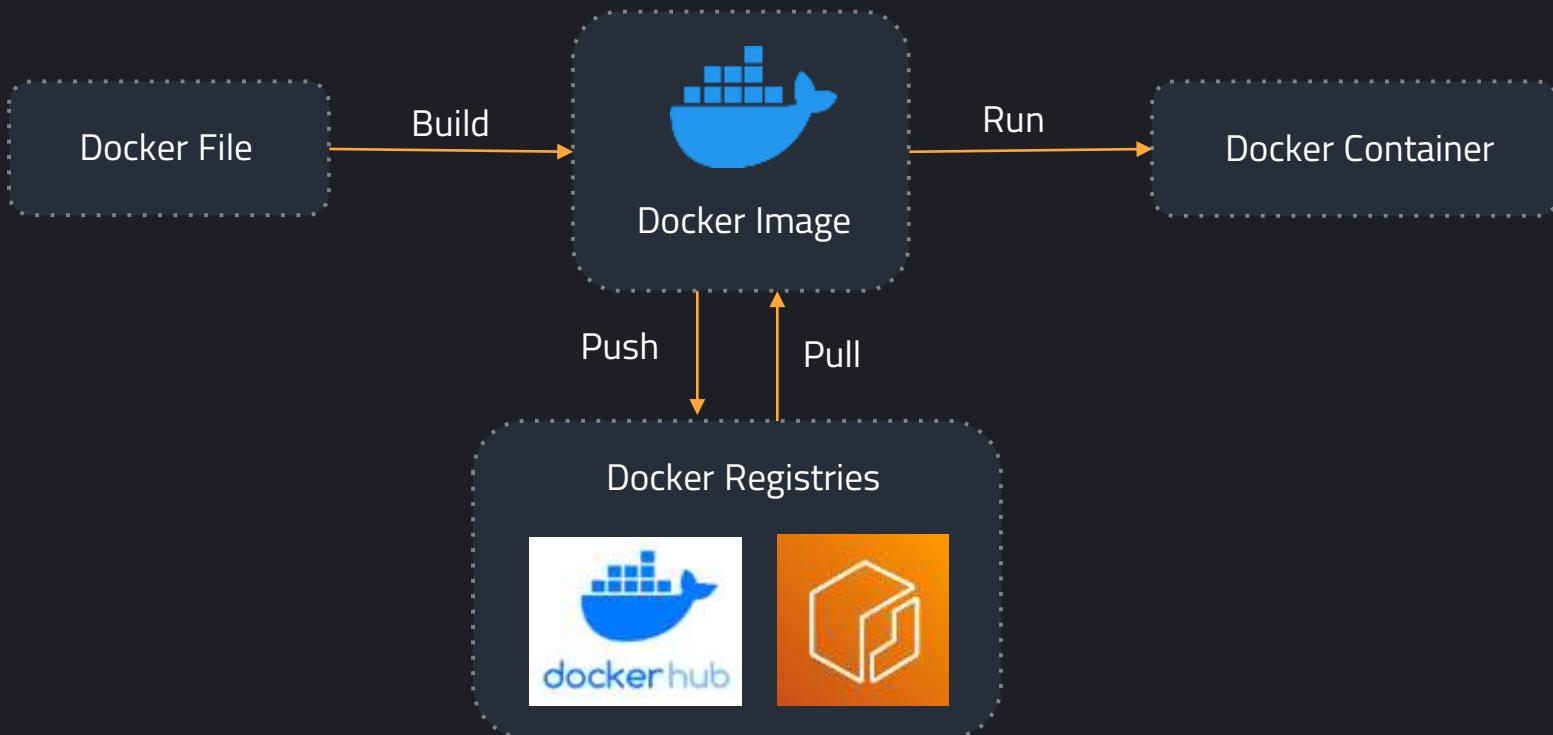
- Is the most popular container registry
- It hosts millions of pre-built images for various software applications, libraries, and frameworks.
- Docker Hub offers both public and private repositories.

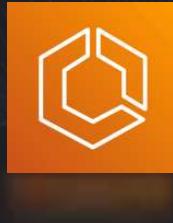


- Amazon ECR is a fully managed Docker container registry provided by AWS.
- Integrates seamlessly with other AWS services
- Amazon ECR supports both public and private repositories.



# Docker Processes





# Elastic Container Service



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# Elastic Container Service



- Is a fully managed container orchestration service.
- Simplifies the process of container

Deployment

Management

Scaling

- It provides high availability and scalability
- It offers built-in security features
- It's integrated with both AWS and third-party tools.





# Elastic Container Service Terms



- **Task Definition**
  - Is a blueprint for your application
  - It encapsulates all the necessary configuration parameters.
- **Cluster**
  - Is a logical grouping of container instances
  - Provides a centralized management point



# Elastic Container Service Terms



- **Task**
  - is ideal for short-running jobs
  - It's an instantiation of a Task Definition.
  - Tasks can be scheduled and terminated dynamically based on workload demands.



# Elastic Container Service Terms



- **Service**

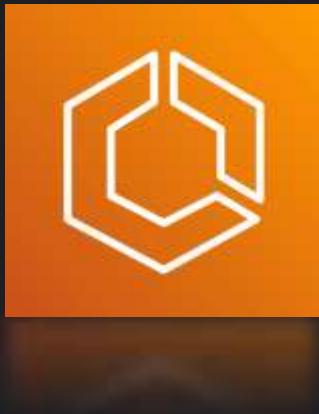
- Is ideal for long-running applications.
- ECS automatically replaces failed tasks.
- It's an instantiation of a Task Definition.

- **Container Agents**

- Run on each EC2 instance within an ECS cluster.
- Serve as the communication bridge between the ECS and the container instances.



# Amazon ECS launch types



- **EC2 launch type**
  - You must provision & maintain the infrastructure
  - Suitable for large workloads that must be price optimized.
  - Enables you to use EC2 instances like spot instances and custom instance types.
  - Scaling does not come out of the box.



# Amazon ECS launch types



- **Fargate launch type**
  - You don't need to manage an EC2 infrastructure
  - Requires less effort to set up
  - AWS just runs ECS Tasks for you based on the CPU / RAM you need
  - handles scaling out your capacity
- **External launch type**
  - Is used to run your containerized applications on your on-premise server or virtual machine (VM)



# Task placement strategies and constraints



- Applicable for EC2 launch type launch mode only

## Task placement strategies

- is an algorithm for selecting instances for task placement or tasks for termination.
- Available strategies are

Binpack

Spread

Random





# Task placement strategies and constraints



- **Binpack**

- Tasks are placed on instances that have the least available CPU or memory capacity
- Helps minimize wasted resources.
- Beneficial for cost optimization and maximizing the usage of resources.





# Task placement strategies and constraints



- **Spread**

- Spreads tasks evenly across container instances within the cluster
- Ensures that no single instance becomes overloaded.
- Suitable for ensuring even distribution of tasks across instances.





# Task placement strategies and constraints



- **Random**
  - Randomly places tasks onto container instances within the cluster
  - Not suitable for applications with specific performance or availability requirements.
  - It's primarily used when you don't have specific constraints or considerations for task placement.
  - It is possible to create a task placement strategy that uses multiple strategies.





# Task placement strategies and constraints



- **Task placement constraint**
  - These are rules that must be met in order to place a task on a container instance.
  - There are two types on constraints types

Distinct Instance

Member of





# Task placement strategies and constraints



- **Distinct Instance**
  - Place each task on a different container instance.
- **Member of**
  - Place tasks on container instances that satisfy an expression.





# Task placement strategies and constraints



- When Amazon ECS places a task, it uses the following process to select the appropriate EC2 Container instance:
  1. CPU, memory, and port requirements
  2. Task Placement Constraints
  3. Task Placement Strategies





# IAM Roles for ECS



- **Task Execution Role**
  - Is an IAM role that ECS uses to manage tasks on your behalf.
- **Task Role**
  - Enables containers within the task to access AWS resources securely.





# Elastic Container Registry



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





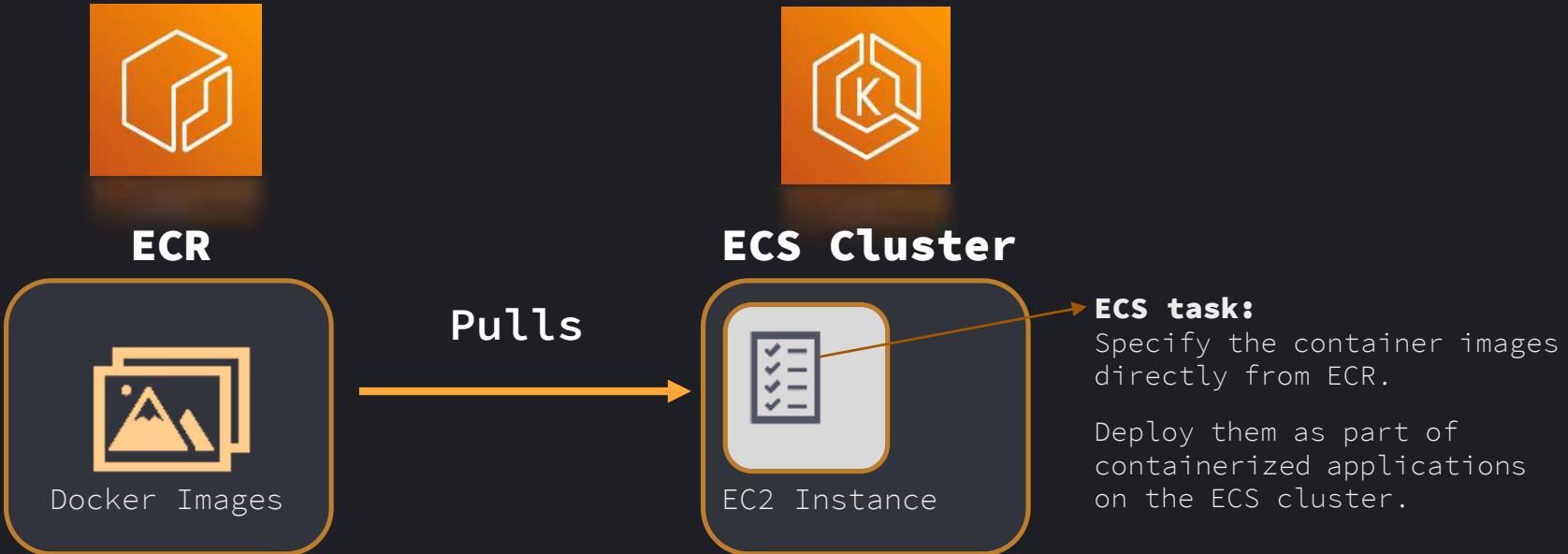
# Amazon Elastic Container Registry



- Amazon ECR is a fully managed Docker container registry provided by AWS.
- It enables you to **store**, manage, and deploy Docker **images** securely.
- It's integrated with other AWS services.
- Is Secure

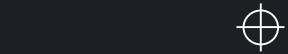


# Amazon Elastic Container Registry





# Amazon Elastic Container Registry



- **Features**

- Lifecycle policies
- Image scanning
- Cross-Region and cross-account replication
- Versioning
- Tagging



# Amazon Elastic Container Registry



## Public Repository

- Are accessible to anyone on the internet.
- Special permission or credential is not required.

## Private Repository

- Are only to authorized users.
- Access to private repositories can be controlled using AWS IAM (Identity and Access Management)





# Elastic Kubernetes Service

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# Kubernetes



- Is an **open-source** platform designed to automate the

Deployment

Management

Scaling

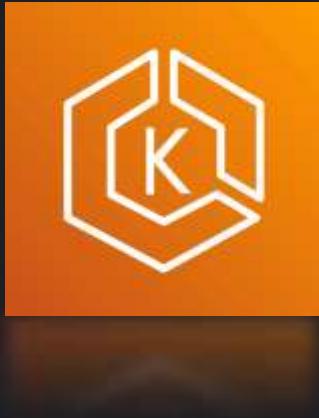
of containerized applications.

- Google open-sourced the Kubernetes project in 2014
- Suitable for running and managing workloads of all sizes and styles
- It is non cloud native, available on most cloud providers





# Amazon Elastic Kubernetes Service



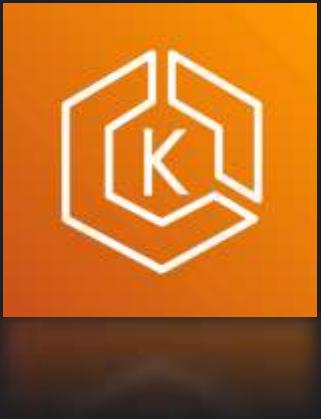
*Managed Kubernetes service to run Kubernetes in the AWS cloud and on-premises data centers.*

- It's integrated with other AWS services.
- Provides High Availability
- Scalability
- Security
- Monitoring and Logging





# Amazon EKS architecture



## Control plane

- Consists of nodes that run the Kubernetes software.
- Manages and orchestrates various components of the Kubernetes cluster.
- Is fully managed by AWS.





# Amazon EKS architecture



## Compute

- Contains worker machines called nodes.
- Amazon EKS offers the following primary node types.

AWS Fargate

Karpenter

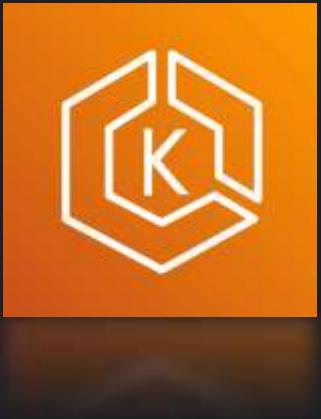
Managed node groups

Self-managed nodes





# Amazon EKS architecture



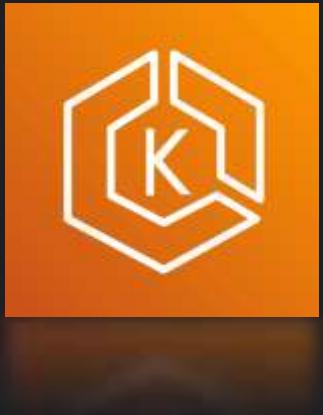
## AWS Fargate

- Is a serverless compute engine.
- AWS manages the underlying infrastructure.
- You specify your application's resource needs, and AWS handles the rest.





# Amazon EKS architecture



## Karpenter

- Best for running containers with a high availability requirement.
- It launches right-sized compute resources in response to changing application load.

## Managed node groups

- Create and manage Amazon EC2 instances for you.





# Amazon EKS architecture



## Self-managed nodes

- Offer full control over your Amazon EC2 instances within an Amazon EKS cluster.
- You are in charge of managing, scaling, and maintaining the nodes.
- Suitable for users who need more control over their nodes.





# Section 18: **Migrations**



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Snow Family

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# AWS Snow Family

- Data management and processing.
- Used for bypassing **bandwidth limitations**.
- **Local computing capabilities:**
  - Snowball Edge and Snowcone.
- **Security:**
  - Secure **at rest** and **in transit**.
  - Uses AWS KMS.





# AWS Snow Family - Snowcone

- Small and lightweight.
- **8TB** of storage.
- Built in **DataSync** agent.
- **Data Processes:**
  - AWS IoT Greengrass, AWS Lambda.





# AWS Snow Family – Snowball Edge

- **1) Storage Optimized:**
  - Storage focused.
  - **80 TB**.
- **2) Compute Optimized:**
  - Run applications, process data.
  - **42 TB**.
- Move data into AWS.
- Data is encrypted.
- Used for remote areas and low bandwidth edge locations.





# AWS Snow Family – Snowmobile



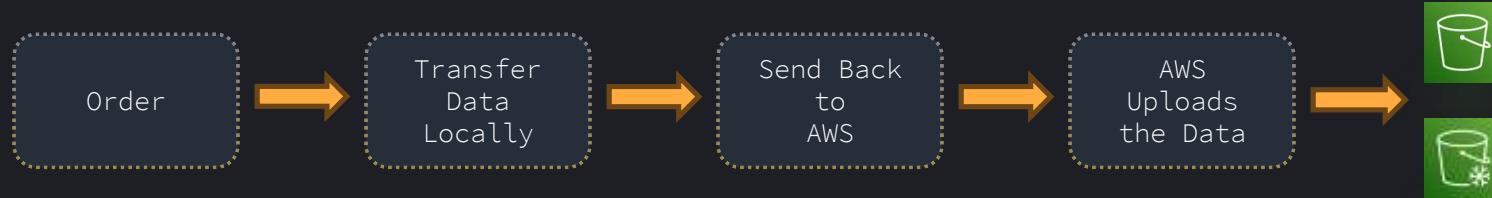
- Up to **100PB** of data carried by a truck.
- Enterprise level large scale datasets.
- Consider over 10PB.





# AWS Snow Family

## Snow Family Process



## Snowcone vs Snowball vs Snowmobile

	Snowcone	Snowball Edge Storage Optimized	Snowmobile
Storage Capacity	8 TB HDD(14TB SSD optional)	80 TB	Up to 100PB
Migration Size	Up to 24 TB, online and offline	Up to petabytes, offline	Up to exabytes, offline
Usable vCPUs	2 vCPU	40 vCPU(104vCPU for Compute Optimized)	-





# AWS Transfer Family



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# AWS Transfer Family



- Secure file transfer with **SFTP**, **FTPS** and **FTP**.
- **Frequently** and **securely** exchanged data.
- Fully managed.
- **Integration with AWS Services:**
  - Amazon S3
  - Amazon EFS
- **Customization and Control:**
  - Setting up DNS
  - IAM based authentication





# AWS Transfer Family

- **Simplified Migration:**
  - No need to modify applications.
- **Pricing:**
  - Pay as you go.



## Use Cases

- Secure data distribution
- Data backup and archiving





# AWS DataSync

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# AWS DataSync

- Simplifies moving data.
- Data transfer **on-premises to AWS storage services**.
- Data transfer between different AWS storage services.
- **File permissions** and **metadata** preserved.
- AWS Snowcone includes AWS DataSync in it.

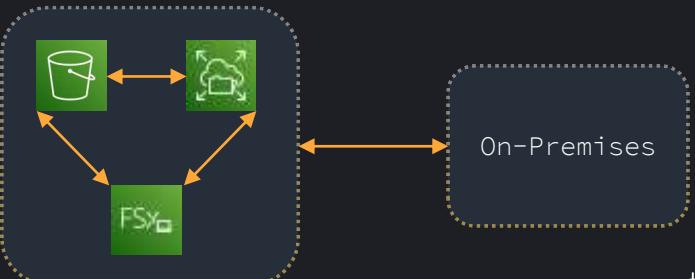




# AWS DataSync

## Key Features

- **Speed:**
  - Up to 10 times faster.
- **Schedule and Automation:**
  - Allows you to schedule transfers.
  - It **does not have continues sync** option.
- **Integration:**
  - Amazon S3, Amazon EFS, Amazon FSx for Windows



# AWS DataSync – How It Works

## Setting Up AWS DataSync

### ▪ Agent Installation:

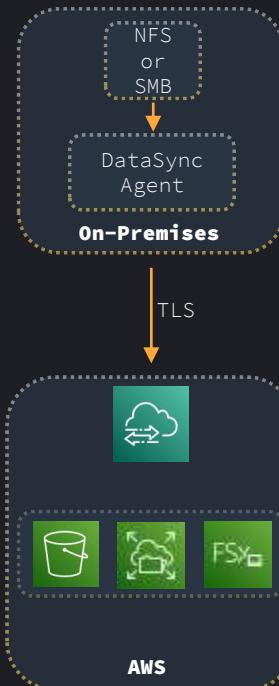
- Install agent in a server that has access to **NFS** or **SMB** file systems.

### ▪ Task Configuration:

- Define sources(NFS,SMB) and targets(**S3,EFS,FSx**).

### ▪ Data Transfer Options:

- Scheduling, bandwidth throttling





# AWS Database Migration Service (DMS)



003-1040559

1250 003-77156.8

1760 0009-14563.7 73273





# AWS DMS



- Helps you on **migrating** databases.
- Supports various type of database engines.
- **Continuous** replication.
- Uses EC2 instances.
- **Security:**
  - Data encrypted during transit.
- **Pricing:**
  - Pay as you go





# AWS DMS – Source & Target



## SOURCES

- Amazon Aurora
- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- MongoDB
- SAP ASE



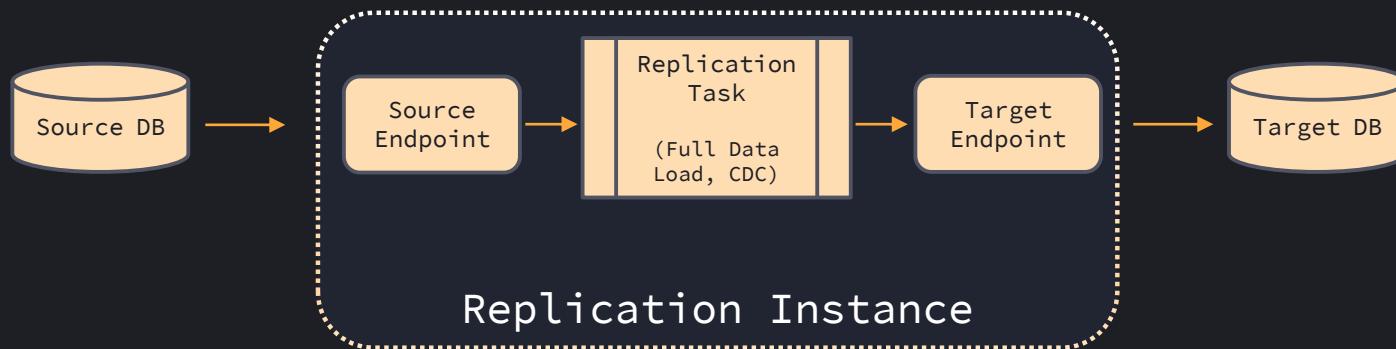
## TARGETS

- Amazon Aurora
- RDS
- Redshift
- DynamoDB
- DocumentDB
- S3
- Kinesis Data Streams
- Apache Kafka



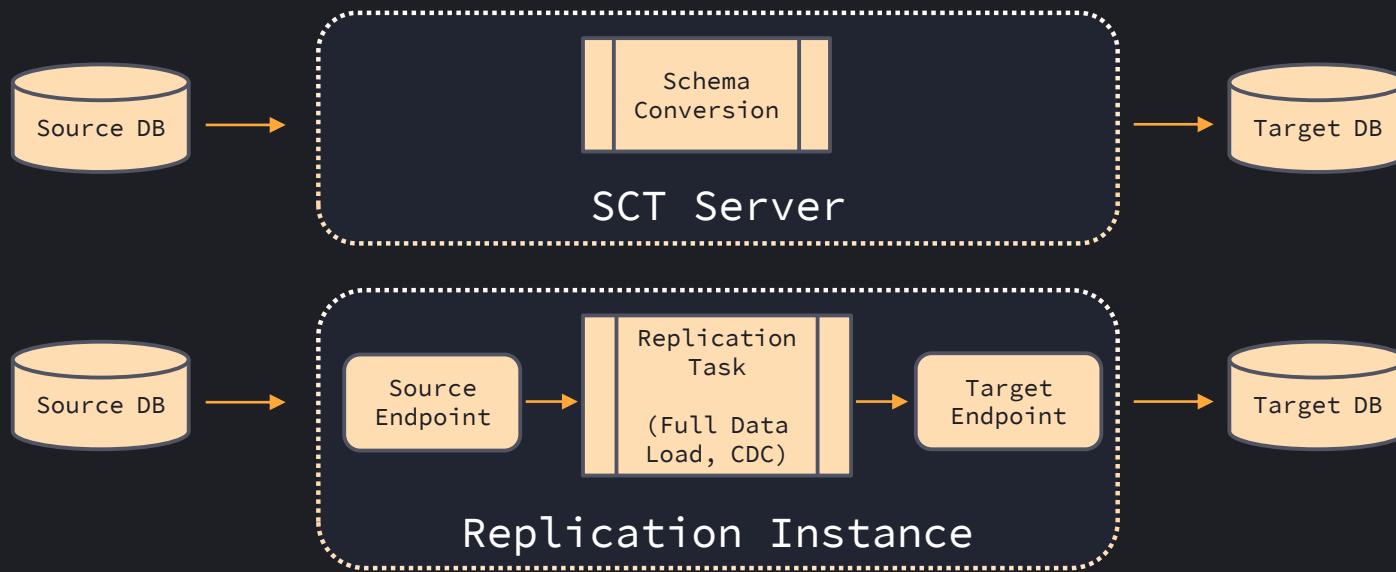
# AWS DMS – How It Works

## Homogeneous Migration



# AWS DMS – How It Works

## Heterogeneous Migration





# AWS Application Discovery Service



003-1040559

1250 003-77156.8

1760 0009-14563.7

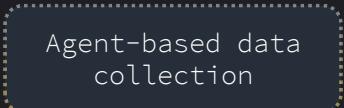
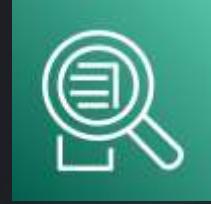
73273





# AWS Application Discovery Service

- Get insights of your on-premise servers and databases
- Useful for **migrations** to understand own resources
- Collects information about your applications.
  - Server specifications, dependencies, usage data



Install software agent



For VMs running in VMware vCenter environment

- Analyze collected data.
- Plan your migration process with collected information.
  - Can integrate with **AWS Migration Hub** and **AWS Application Migration Service**





# AWS Application Migration Service



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Application Migration Service

- Simplifies migration process with automations.



## Lift-and-Shift

⇒ Can replicate entire servers without significant downtime

## Automates Migration

⇒ Migration of applications, databases, servers to AWS

## Test Before Switching

⇒ Supports creating test environments

## Compatibility

⇒ Supports most common environments including Windows and Linux





# AWS Application Migration Service

- Workflow for migration



## Install

Install the AWS Replication Agent on the source server

## Configure

Configure the launch settings for each server

## Launch Test Instance

Test the migration of your source servers to AWS

## Cutover

Cutover will migrate your source servers





# Section 19: **VPCs**



003-1040559      1250 003-77156.8

1760 0009-14563.7      73273





# Networking



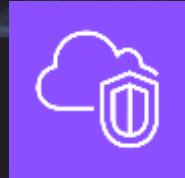
003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# Virtual Private Cloud



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



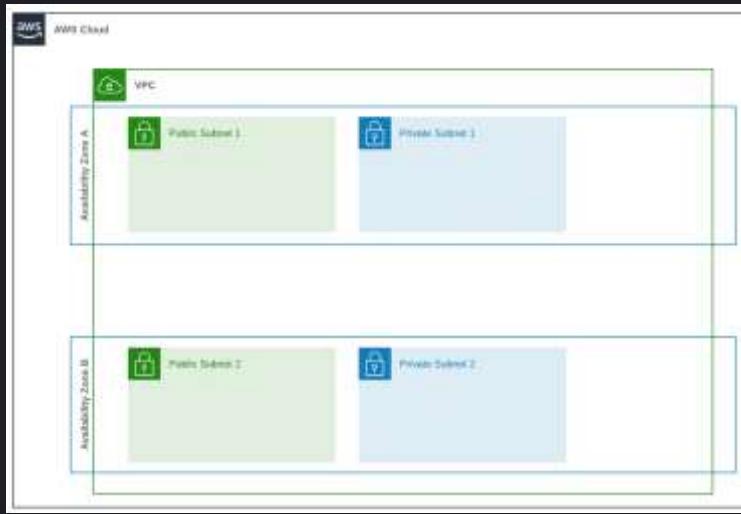


# Amazon VPC



Private, secure, isolated network within the AWS cloud to launch your resources.

- ⇒ *Can be linked to on-premise infrastructure*
- ⇒ *Regional Service*



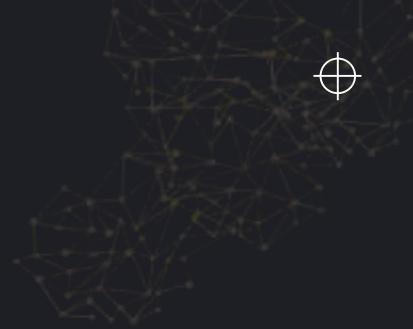


# Amazon VPC Subnets



A range of IP addresses in your VPC

⇒ *Zonal Service*



- **Types of Subnets**

- *Public Subnets* have access to the internet
- *Private Subnets* do not have direct access to the internet
- *VPN-only Subnets* are accessed via a VPN connection
- *Isolated Subnets* are only accessed by other resources in the same VPC

## Subnet Routing



Route Tables are sets of rules that dictate how traffic is routed in your VPC.





# Networking Components



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# Amazon VPC Networking Components

## *Internet Gateway*



Allows communication between your VPC and the internet.

## *Egress-Only Internet Gateway*

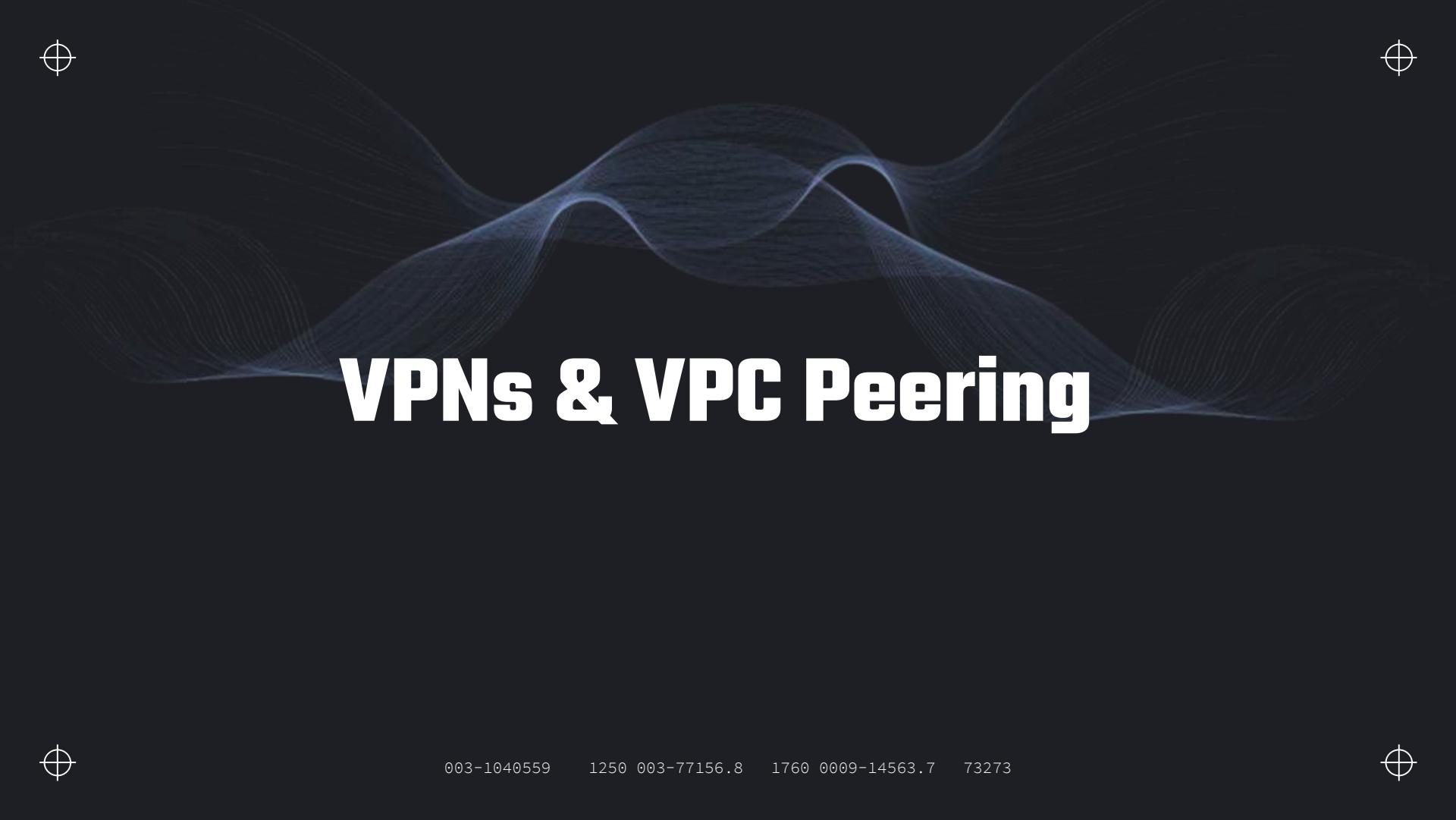


Allows outbound IPv6 communication from VPC instances to the Internet, while blocking inbound IPv6 connections.

## *NAT Gateway/Instance*



Allows resources in private subnets to connect to external destinations but prevents connection requests



# VPNs & VPC Peering

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



# Amazon VPC and Corporate Network

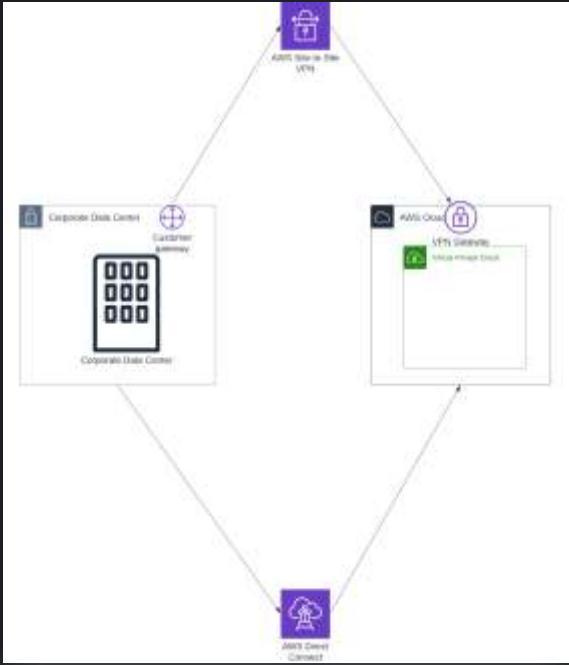
You can connect your VPC to your own corporate data center

## *Virtual Private Network*

- Secure connection between a VPC and an on-premises network over the internet.

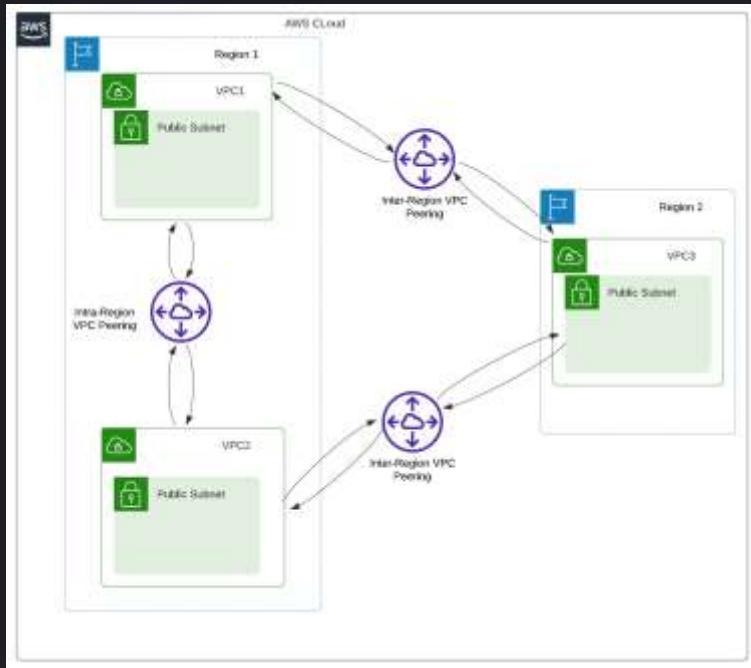
## *Direct Connect*

- Private connectivity between corporate networks and AWS.





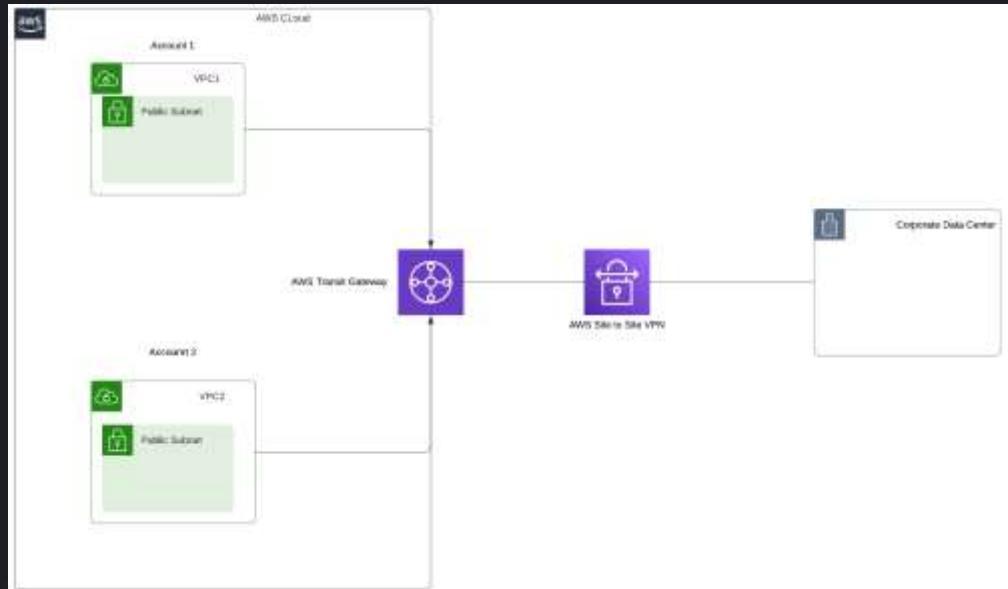
# Amazon VPC Peering



- Enables direct communication between two VPCs
- Intra or Inter Region



# Amazon VPC Transit Gateway



- Central hub interconnecting VPCs and on-premises networks.



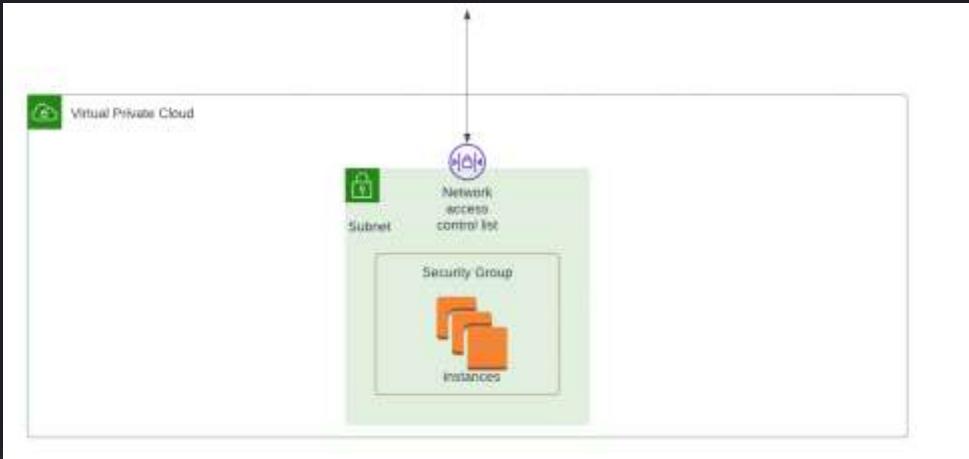
# Security Groups & NACLs

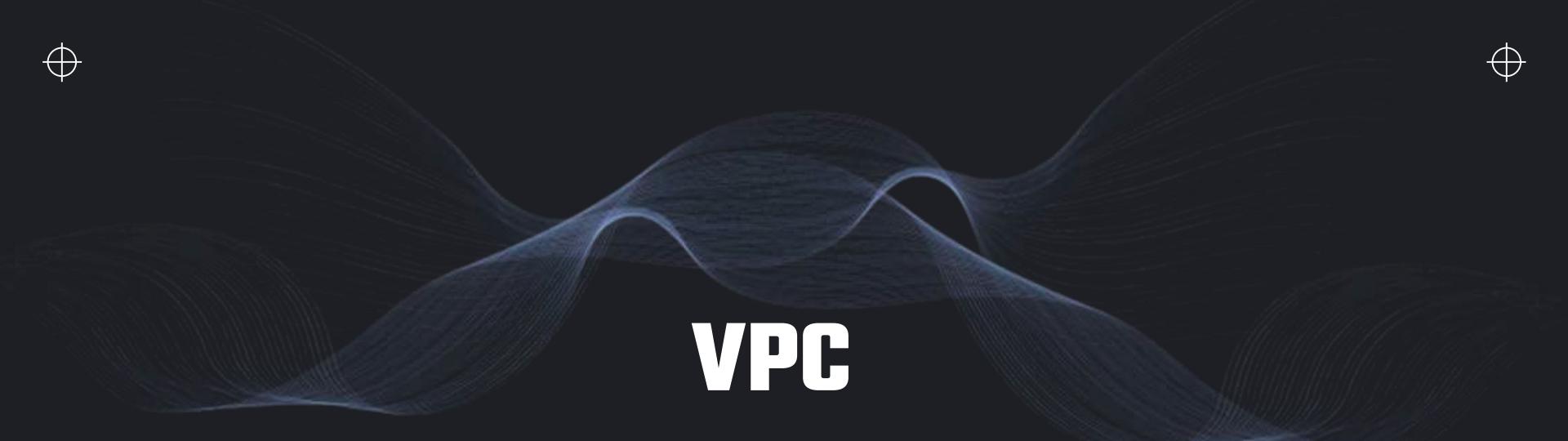


# Amazon VPC Security



- Security Groups control inbound or outbound traffic at resource level
- Network Access Control List (NACL)  
Control inbound or outbound traffic at the subnet level





# VPC

# Additional Features



# Amazon VPC Extra Features



**VPC Flow Logs:** Capture information about IP traffic going to and from network interfaces.

**Reachability Analyzer:** Analyze network reachability between resources within your VPC and external endpoints

**Ephemeral Ports:** Temporary ports for outbound communication

**VPC Sharing:** Share your VPC resources with other AWS accounts in the same AWS Organization,





# Section 20: **Security**



003-1040559      1250 003-77156.8

1760 0009-14563.7      73273





# AWS KMS

# (Key Management Service)



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# KMS

## Overview

- **Manages Encryption Keys:**  
To encrypt data in other AWS services  
  
⇒ Used to encrypt & decrypt data
- **Integration:**  
Integrates to other services (S3, databases, EBS volumes etc.)
- **API calls:**  
Don't store secrets in code
- **Cloud Trail integration:**  
Log use of your keys for auditing



## Use Cases

- Encrypt data stored in **S3 buckets**
- **Database credentials:**  
Encrypt credentials instead of storing them in plain text

# Types of Keys

## Symmetric Keys

*Default*

- **One key** for both encryption and decryption
  - Suitable for **high-volume data.**
  - **Example:** AES with 256-bit keys
  - **At rest & in-transit**
- 

## Asymmetric Keys

- Uses **key pair**
- **Public Key:** Encrypt data (can be downloaded)
- **Private Key:** Decrypts data
- Encrypted data must be **shared safely**
- **Sign/Verify** operations
- **Example:** RSA & ECC



# AWS-Managed vs. Customer-Managed

AWS owned keys

- **Controlled and managed by AWS**
- No direct access and lifecycle control



- Owned by service
- Good choice unless you need to audit and manage key

AWS managed keys

- Created & managed by AWS KMS – but customer-specific
- **Cannot control** over its usage and policies, rotation etc.
- Good choice unless you need to control the encrypt key
- **Audit using CloudTrail**



# AWS-Managed vs. Customer-Managed

Customer managed keys

- You create, own, and manage
- Full control over these KMS keys
- Policies, rotation, encryption





# KMS Key Management



## Creating Keys

- AWS Management Console, AWS CLI, or SDK
- Selecting the key type
- Key policies

## Rotating Keys

- Replaces old keys with new ones
- KMS handles complexities

## Managing Keys

- Configuring key policies



# Key Rotation

## Automatic Rotation

- For keys that **AWS manages**
- **Automatically** rotates the keys every year

## Customer Managed

- Users responsibility

# Policies

## Default Policies

- **Full access to the key to the root user**
- Allows usage of **IAM policies**

## Custom Policies

- More complex requirements
- More Granular Control
- **Regulated industries**



# AWS KMS



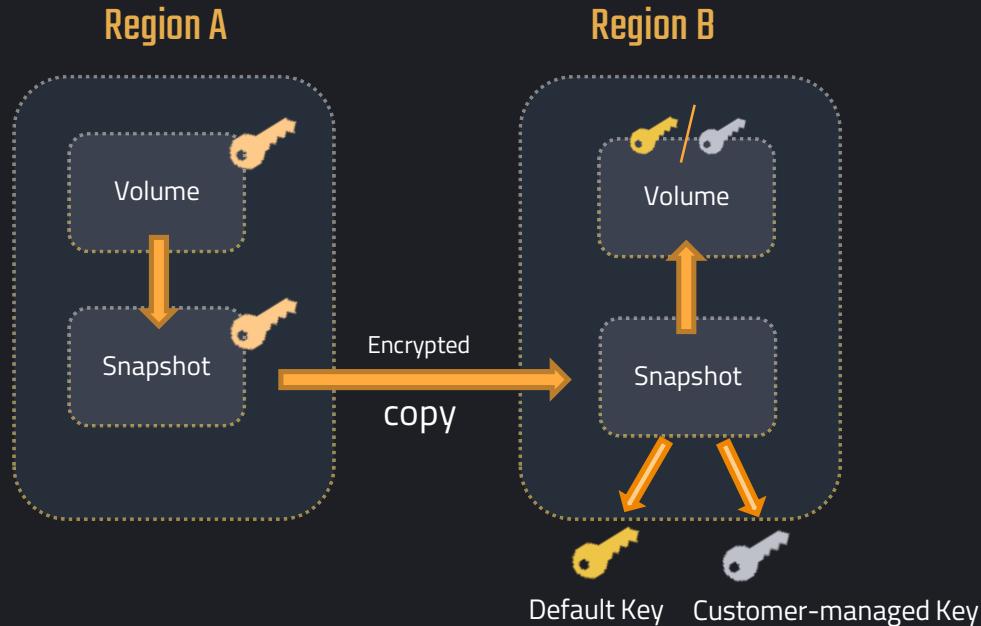
## Pricing

- **Pricing:**
  - \$1.00 per customer-managed key per month;
  - \$0.03 per 10,000 API requests.
- **Key Rotation:**
  - **Automatic:** Free for AWS-managed keys.
  - **Manual:** No extra charge for customer-managed keys; requires setup.
- **Cross-Region Requests:** \$0.01 per 10,000 requests for using a KMS key in a different region.



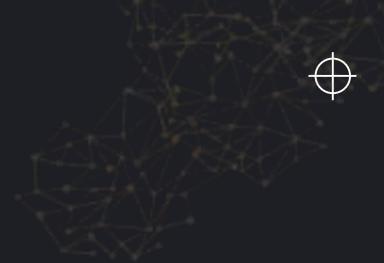
# Cross-region

Keys are bound to the region in which they are created





# Multi-Region keys in AWS KMS



- Use keys in different AWS Regions you had the same key
- Each set of *related* multi-Region keys has the same key material and key ID
- Manage each multi-Region key independently
- Create a multi-Region primary key ⇒ replicate it into Regions that you select

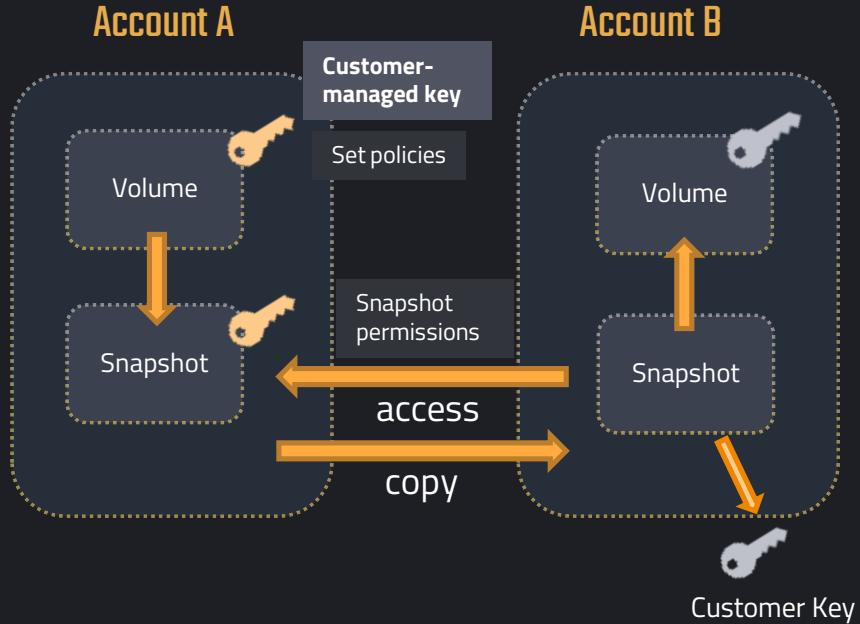
## Use Cases

- Disaster recover in multi-region setup
- Data distributed in multiple regions
- Distributed signing applications



# Cross-account

Keys can be shared across accounts  
Configurable using policies





# AWS Macie



003-1040559 1250 003-77156.8 1760 0009-14563.7 73273





# AWS Macie



## Purpose

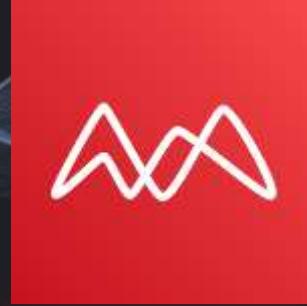
- Automatically scans and **classifies sensitive data in Amazon S3**
- **Machine Learning:** Detects sensitive data
  - Personally identifiable information (PII)
  - Financial data
  - Health information
  - *Anomalous access patterns*
- **Automated alerts:**  
Detailed alerts when sensitive data or unusual access patterns are detected; Integrates with CloudWatch and other services
- **Comprehensive Dashboard:** Overview of S3 environment

## Features

## Use Cases

- Regulatory Compliance
- Security Monitoring
- Risk Assessment





# AWS Secrets



003-1040559

1250 003-77156.8

1760 0009-14563.7

73273





# AWS Secrets



## Purpose

- Manage and retrieve secrets
  - Database credentials (RDS & Redshift)
  - API Keys
  - Access Tokens

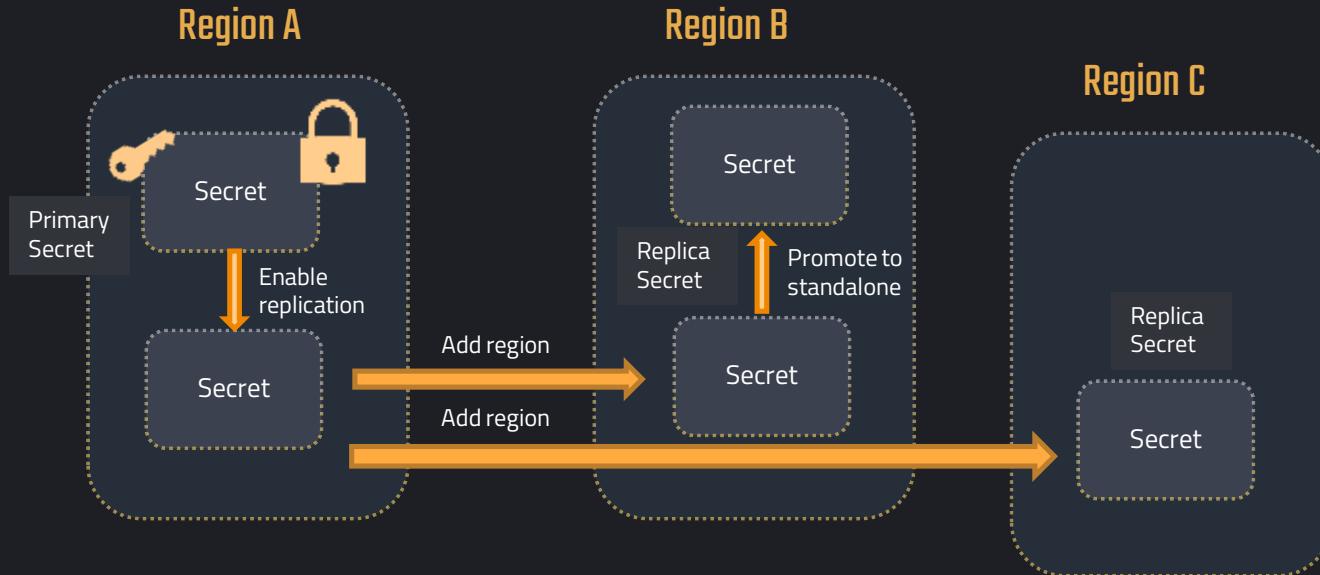
## Features

- **Secrets Management & Storage:**  
Encrypts and stores secrets with pre-built integration for other services
- **Automatic Rotation:** No need to change code in applications
- **Retrieval:** Secure retrieval via API calls
- **Auditing:** CloudTrail Integration



# Cross-region replication

Replicate secrets across multiple AWS Regions





# Cross-region replication



- **ARN Consistency:**  
ARN remains the same

Primary: arn:aws:secretsmanager:Region1:123456789012:secret:MySecret-a1b2c3

Replica: arn:aws:secretsmanager:Region2:123456789012:secret:MySecret-a1b2c3

- **Metadata and Data Replication:**  
Encrypted secret data, tags, and resource policies are replicated across specified regions.
- **Automatic Rotation:**  
If rotation is enabled on the primary secret, the updated secret values automatically propagate to all replicas.

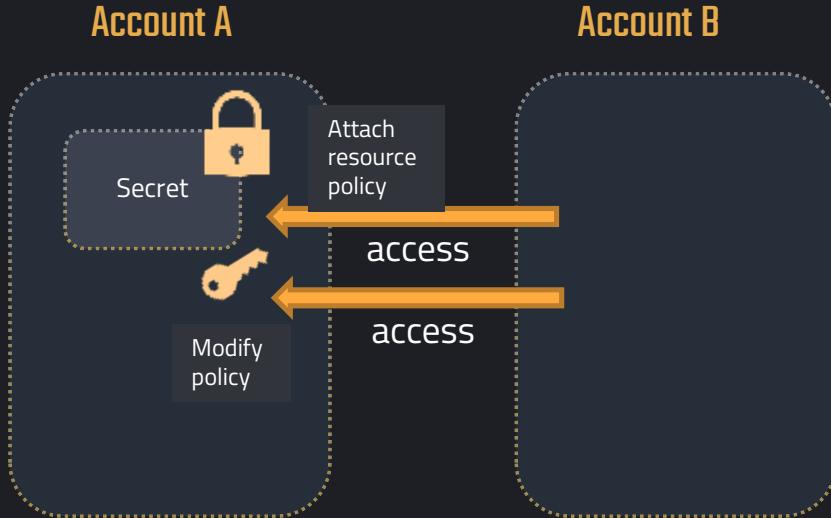




# Cross-account



Secrets can be shared across accounts  
Configurable using policies





# AWS Shield

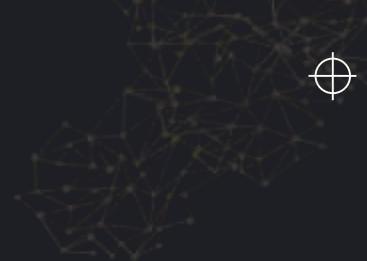


003-1040559 1250 003-77156.8 1760 0009-14563.7 73273





# AWS Shield



## AWS Shield Standard

- **Automatically enabled & Free**
- **Protection against most common** network and transport layer DDoS attacks (**96%**)
- Attacks against network and transport layers (layer 3 and 4) and the application layer (layer 7).
- E.g. Slow reads or volumetric attacks
- **AWS Services Covered:** Amazon CloudFront, Elastic Load Balancing (ELB), Amazon Route 53, and more.
- **Visibility and Reporting:** Provides AWS CloudWatch metrics and AWS Health Dashboard notifications during larger attacks.





# AWS Shield



AWS Shield **Advanced**

## Protect against Distributed Denial of Service (DDoS) attacks

- **Enhanced DDoS Protection:** Guards against complex DDoS attacks.
- **Financial Shield:** Protects from attack-related cost spikes.
- **24/7 Expertise:** Access to AWS DDoS Response Team.
- **Attack Insights:** Immediate and detailed attack analysis.
- **Custom Rules:** Personalized protection with AWS WAF.
- **Targeted Defense:** Specific protection for key resources.
- **Premium Service:** Subscription model with advanced features.

