Build and automate a modern serverless data lake on AWS

Aditya Challa
AWS Solutions Architect
Amazon Web Services

aws certified Solutions Architect Professional
aws certified DevOps Engineer Professional
aws certified Alexa Skill Builder Specialty
aws certified Advanced Networking Specialty
aws certified Security Specialty
aws certified Database Specialty
aws certified Big Data Specialty
aws certified Machine Learning Specialty
aws certified Data Analytics Specialty

AWS re:Invent

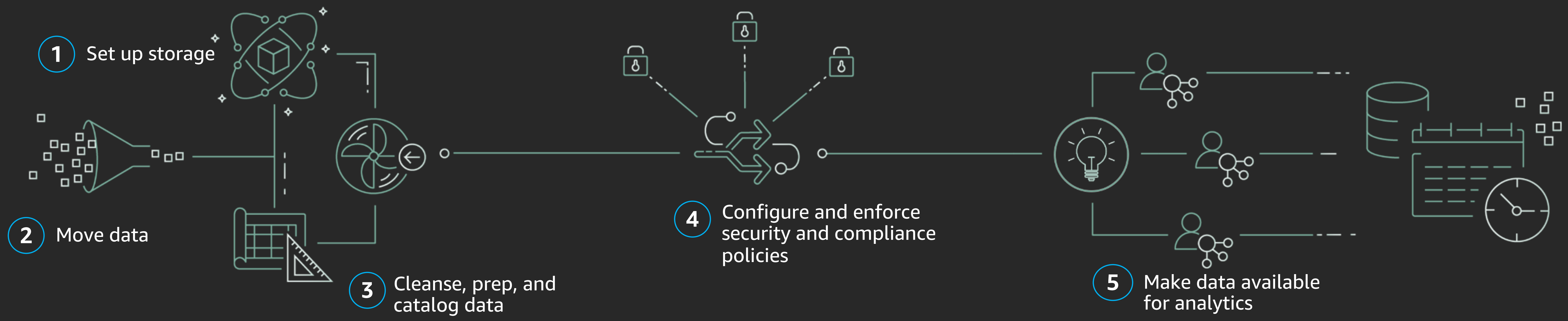
A **data lake** is a system or repository of data stored in its natural/raw format, usually object blobs or files. A data lake is usually a single store of all enterprise data including raw copies of source system data and transformed data used for tasks such as reporting, visualization, advanced analytics and machine learning. A data lake can include structured data from relational databases (rows and columns), semi-structured data (CSV, logs, XML, JSON), unstructured data (emails, documents, PDFs) and binary data (images, audio, video). A data lake can be established "on premises" (within an organization's data centers) or "in the cloud" (using cloud services from vendors such as Amazon Web Services).

-- Wikipedia

Serverless computing is a cloud computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity. It can be a form of utility computing.

-- Wikipedia

Typical steps of building a data lake



1 Set up storage

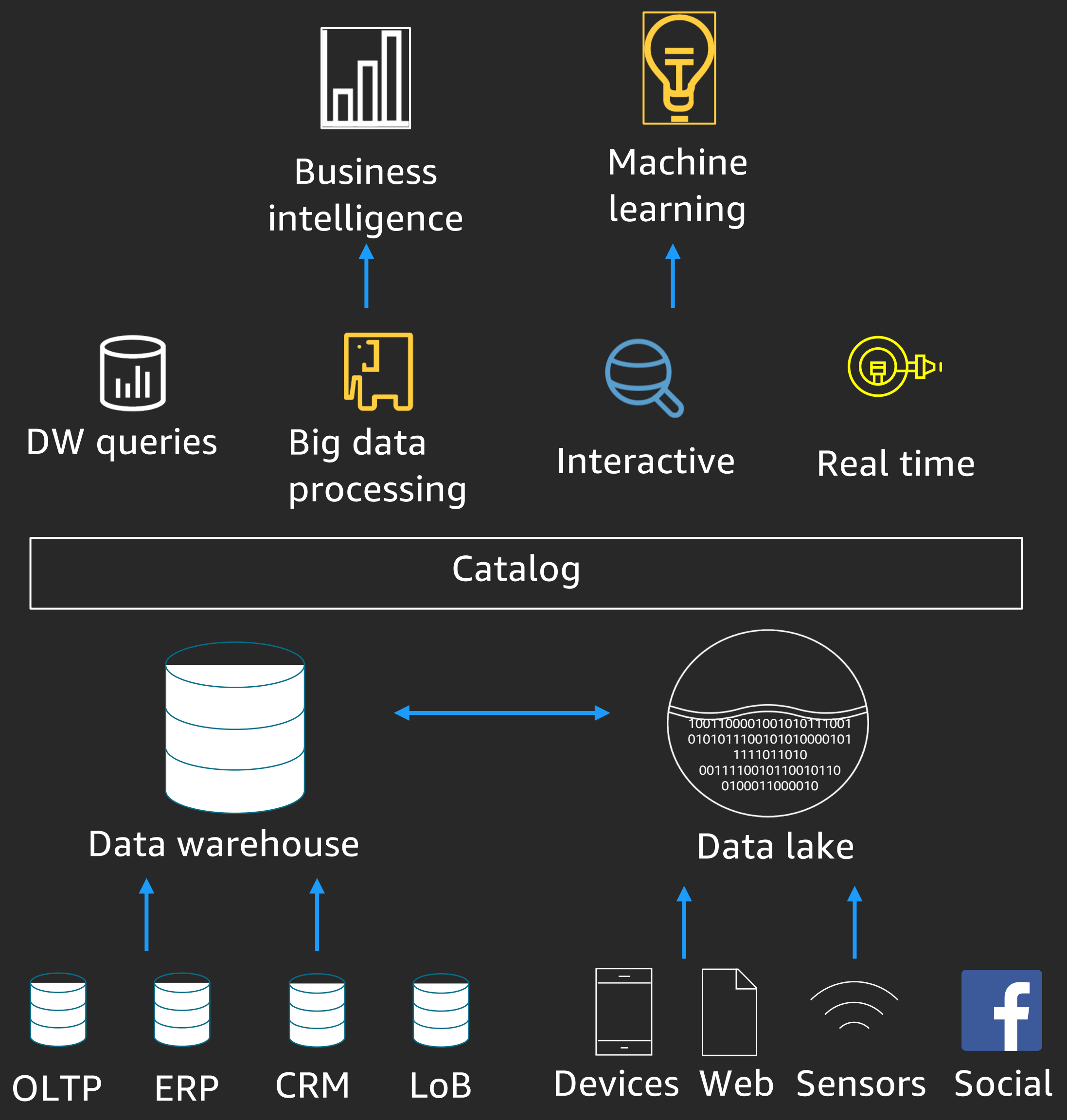
2 Move data

3 Cleanse, prep, and catalog data

4 Configure and enforce security and compliance policies

5 Make data available for analytics

Defining the AWS data lake



Data lakes provide:

Relational and nonrelational data

Scale-out to Amazon EBS

Diverse set of analytics and machine learning tools

Work on data without any data movement

Designed for low-cost storage and analytics

Why use AWS for big data & analytics?

Agility

Scalability

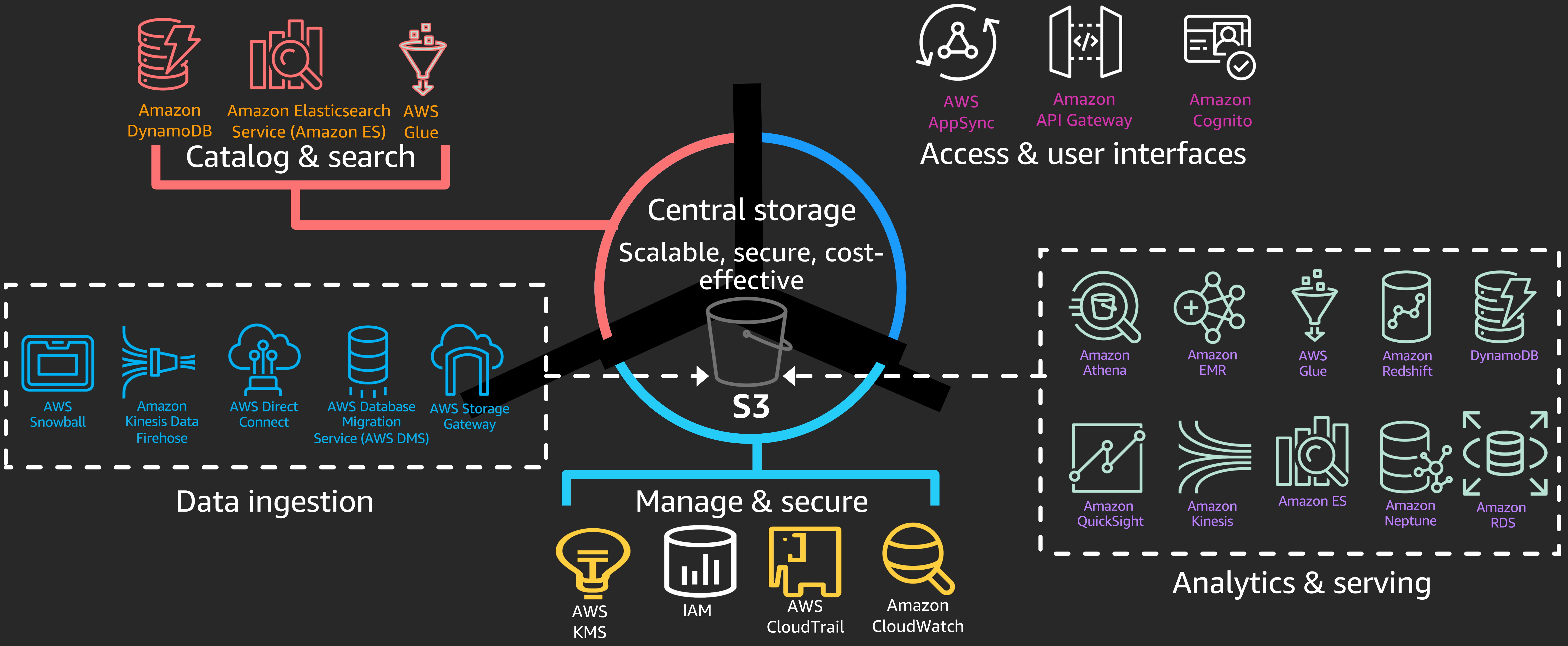
Broadest and deepest capabilities

Low cost

Get to insights faster

Data migrations made easy

Data lake on AWS



Modern serverless data lake components

Amazon S3

AWS Glue

AWS Lambda

Amazon CloudWatch Events

Amazon S3 is the best place for data lakes

Unmatched durability, availability, and scalability

Best security, compliance, and audit capabilities

Object-level controls

Business insights into your data

Most ways to bring data in

Rapidly ingest all data sources

Ingest methods

- Kinesis Data Firehose
- AWS DMS
- Storage Gateway
- Snowball Edge
- DX

IoT, sensor data, clickstream data, social media feeds, streaming logs

Oracle, MySQL, MongoDB, DB2, SQL Server, Amazon RDS

On-premises ERP, mainframes, lab equipment, NAS storage

Offline sensor data, NAS, on-premises Hadoop

On-premises data lakes, EDW, large-scale data collection

Amazon S3

A data lake needs to accommodate a wide variety of concurrent data sources

AWS Transfer for SFTP

Fully managed service enabling transfer of data over SFTP while stored in Amazon S3

Seamless migration of existing workflows

Fully managed in AWS

Secure and compliant

Native integration with AWS services

Cost-effective

Simple to use

AWS DataSync

Transfer service that simplifies, automates, and accelerates data movement

Transfers up to 10 Gbps per agent

Simple data movement to Amazon S3 or Amazon EFS

Secure and reliable transfers

AWS integrated

Pay as you go

Combines the speed and reliability of network acceleration software with the cost-effectiveness of open-source tools

Migrate active application data to AWS

Transfer data for timely in-cloud analysis

Replicate data to AWS for business continuity

Choosing the right data formats

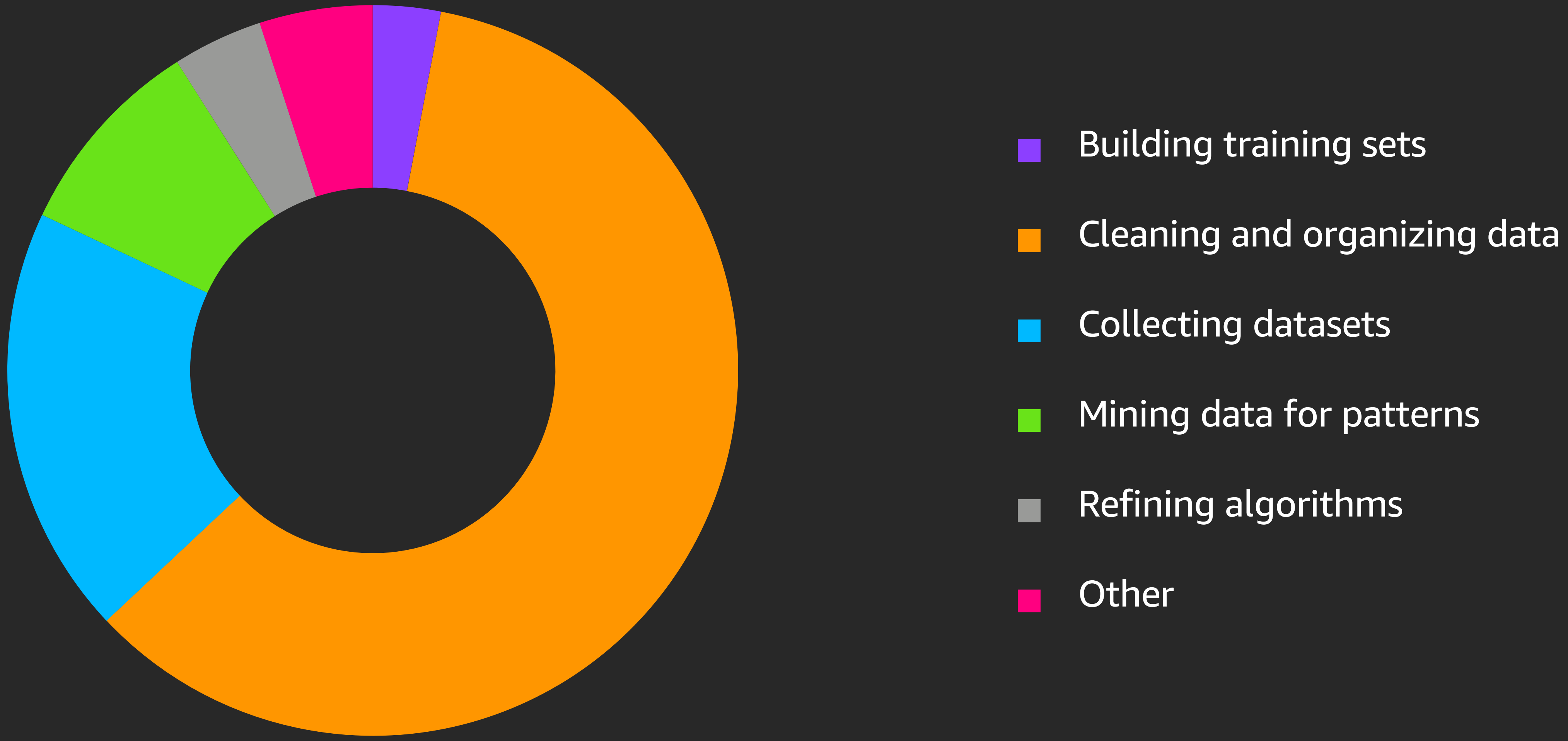
There is no such thing as the “best” data format

- All involve tradeoffs, depending on workload & tools
- CSV, TSV, JSON are easy but not efficient
  - Compress & store or archive as raw input
- Columnar compressed are generally preferred
  - Parquet or ORC
  - Smaller storage footprint = lower cost
  - More efficient scan & query
- Row-oriented (AVRO) good for full data scans
- Organize into partitions
- Coalescing to larger partitions over time

Key considerations are cost, performance, and support

Serverless ETL using AWS Glue

Data prep is ~80% of data lake work



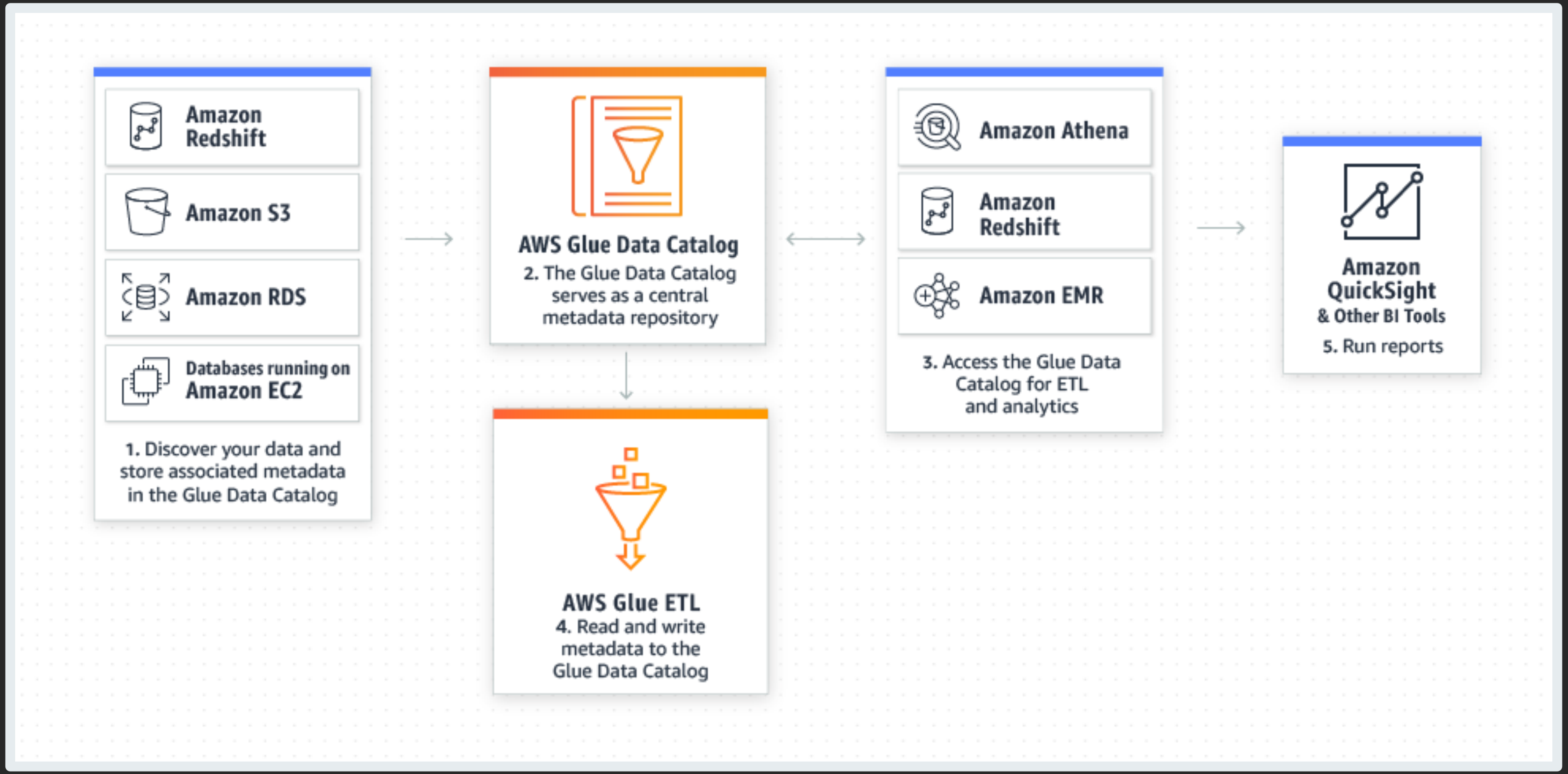
Set up a catalog, ETL, and data prep with AWS Glue

Serverless provisioning, configuration, and scaling to run your ETL jobs on Apache Spark

Pay only for the resources used for jobs

Crawl your data sources, identify data formats, and suggest schemas and transformations

Automates the effort in building, maintaining, and running ETL jobs



Amazon Redshift

Amazon S3

Amazon RDS

Databases running on Amazon EC2

1. Discover your data and store associated metadata in the Glue Data Catalog

AWS Glue Data Catalog

2. The Glue Data Catalog serves as a central metadata repository

AWS Glue ETL

4. Read and write metadata to the Glue Data Catalog

Amazon Athena

Amazon Redshift

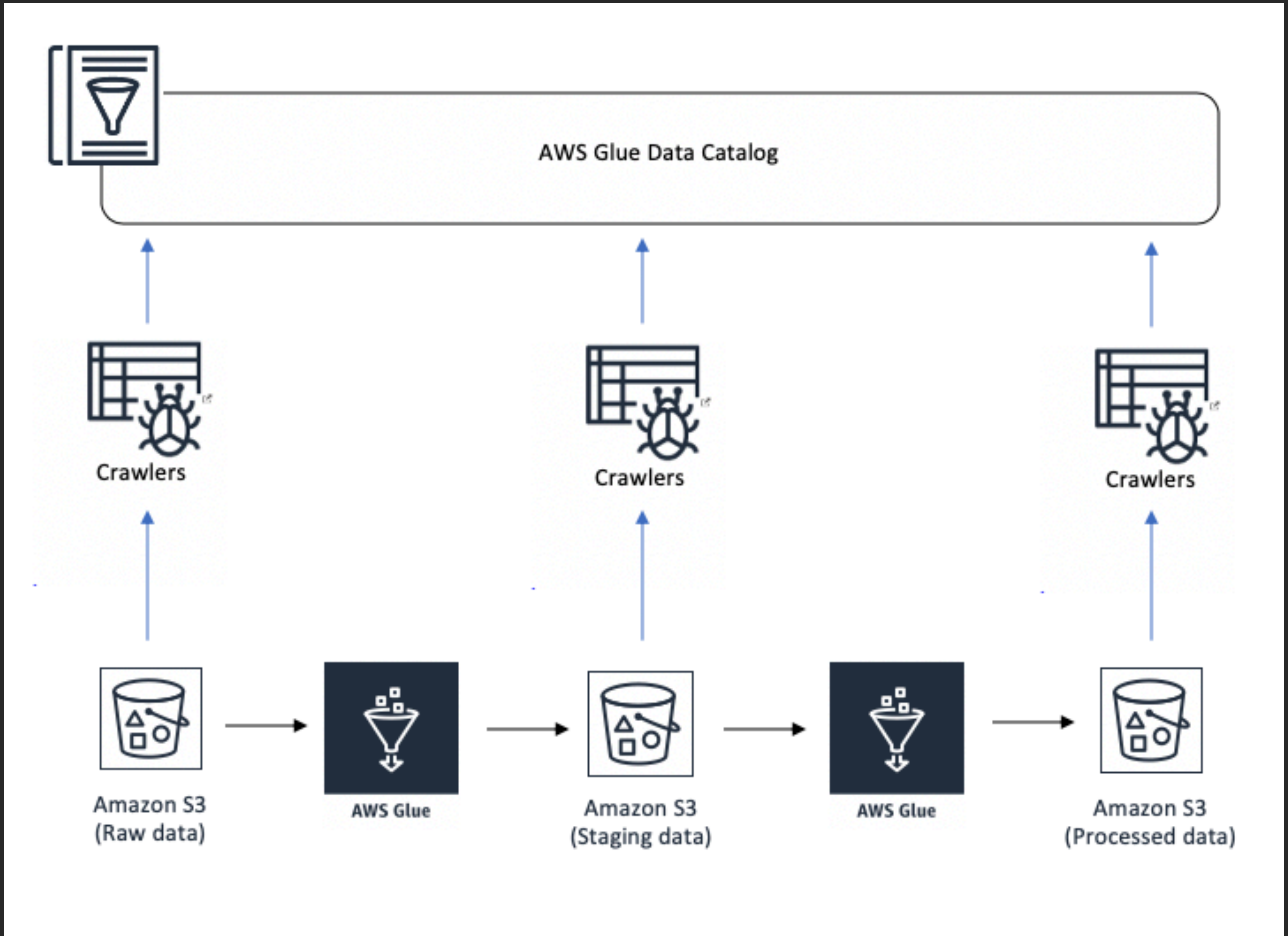
Amazon EMR

3. Access the Glue Data Catalog for ETL and analytics

Amazon QuickSight & Other BI Tools

5. Run reports

AWS Glue In Action



AWS Glue: Components

Data Catalog

- Hive metastore compatible with enhanced functionality
- Crawlers automatically extract metadata and create tables
- Integrated with Athena, Amazon Redshift Spectrum

Job Authoring

- Auto-generates ETL code
- Builds on open frameworks—Python and Spark
- Developer-centric—editing, debugging, sharing

Job Execution

- Runs jobs on a serverless Spark platform
- Provides flexible scheduling
- Handles dependency resolution, monitoring, and alerting

AWS Glue Data Catalog

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

We added a few extensions:

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

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

AWS Glue Data Catalog: Crawlers

Crawlers automatically build your Data Catalog and keep it in sync

- Automatically discover new data, extract schema definitions
    - Detect schema changes and version tables
    - Detect Hive style partitions on Amazon S3

- Built-in classifiers for popular types; custom classifiers using Grok expressions

- Run ad hoc or on a schedule; serverless—only pay when crawler runs

Data Catalog: Detecting partitions

S3 bucket hierarchy

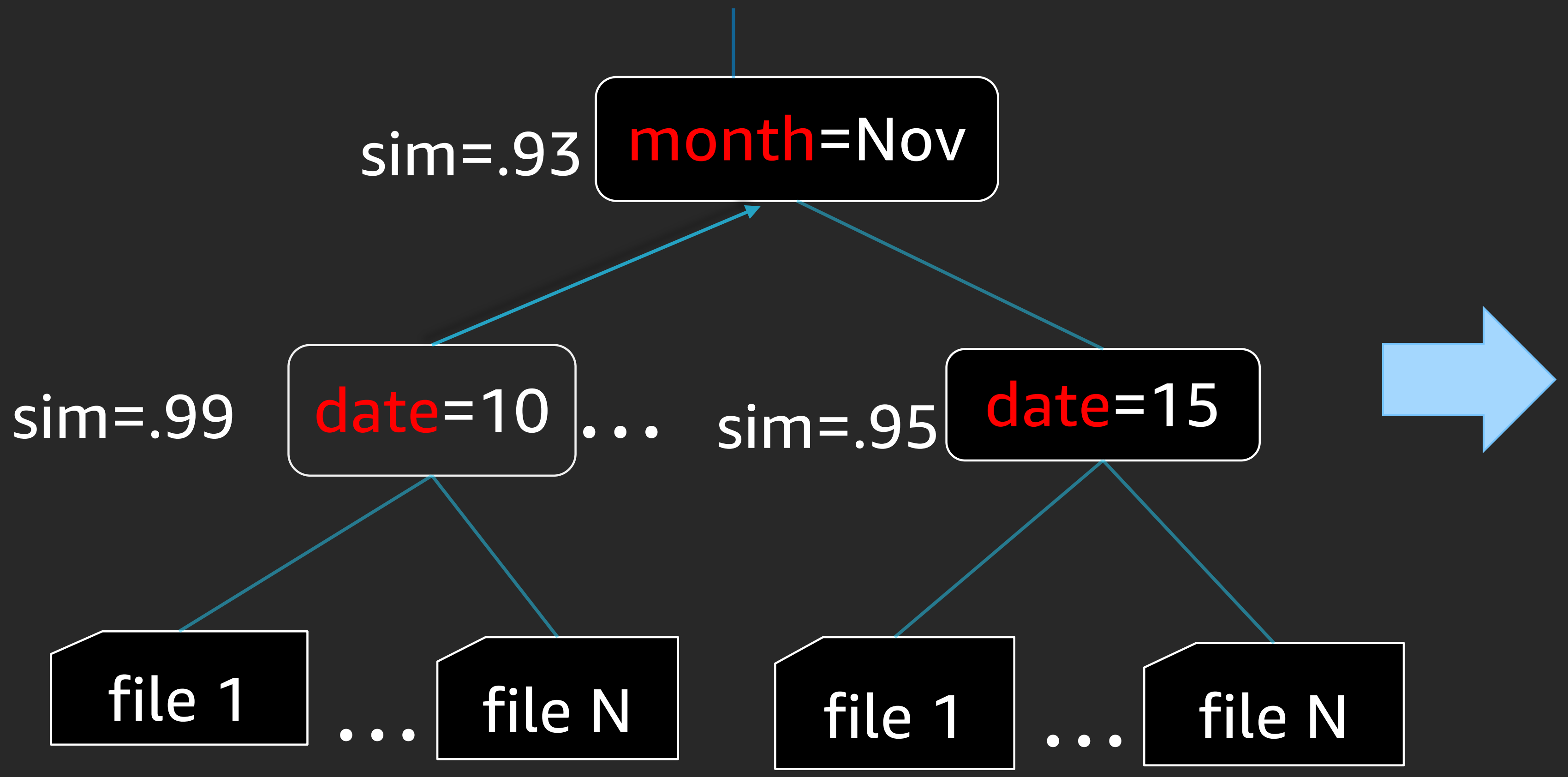


Table definition

Column	Type
month	str
date	str
col 1	int
col 2	float
⋮	⋮

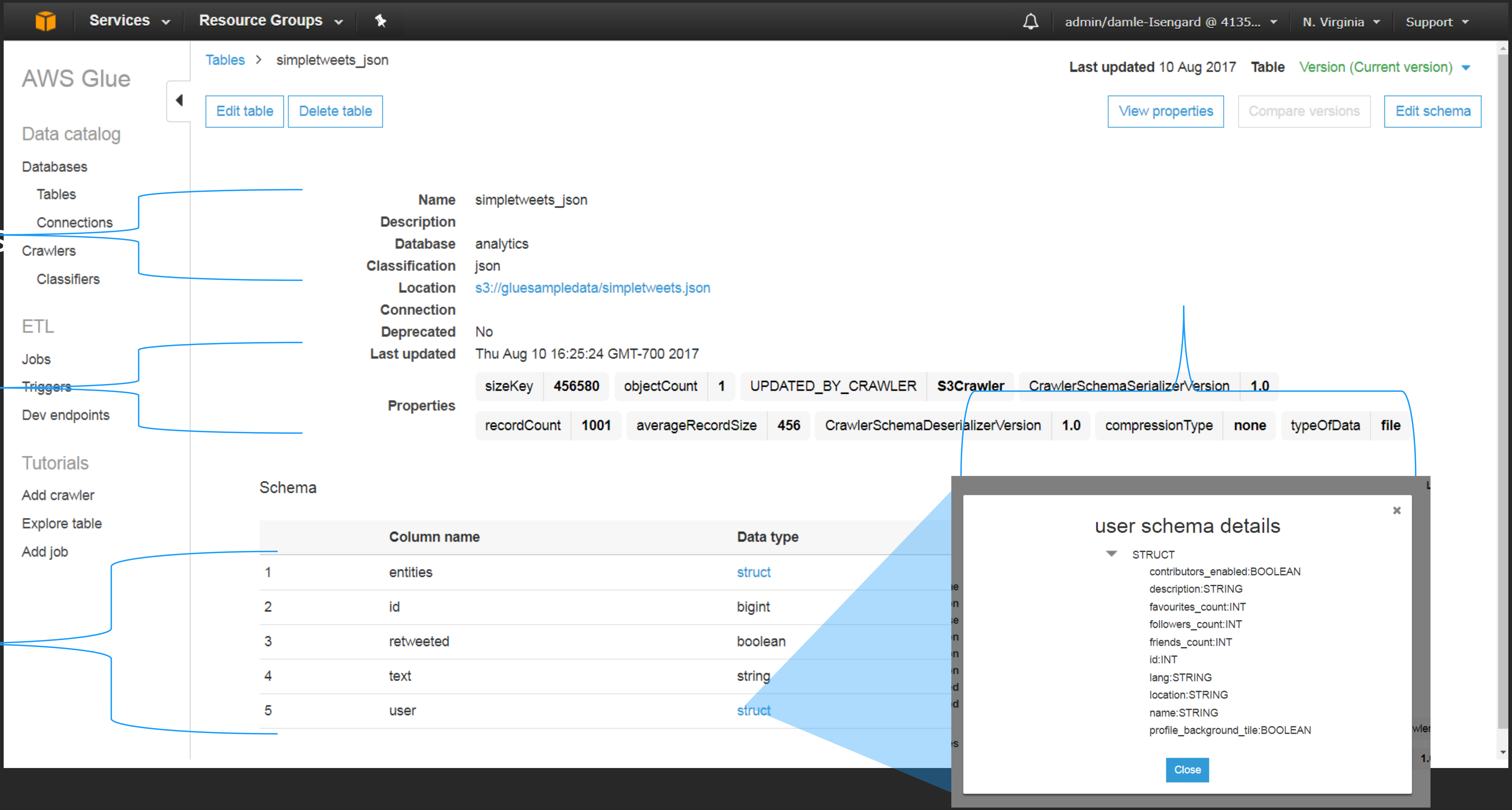
Estimate schema similarity among files at each level to handle semi-structured logs, schema evolution . . .

Data Catalog: Table details

Table properties

Data statistics

Table schema



Last updated 10 Aug 2017 **Table** Version (Current version)

Name simpletweets_json
Description
Database analytics
Classification json
Location s3://gluesampledata/simpletweets.json
Connection
Deprecated No
Last updated Thu Aug 10 16:25:24 GMT-700 2017
Properties sizeKey **456580** objectCount **1** UPDATED_BY_CRAWLER **S3Crawler** CrawlerSchemaSerializerVersion **1.0** recordCount **1001** averageRecordSize **456** CrawlerSchemaDeserializerVersion **1.0** compressionType **none** typeOfData **file**

Schema

	Column name	Data type
1	entities	struct
2	id	bigint
3	retweeted	boolean
4	text	string
5	user	struct

user schema details

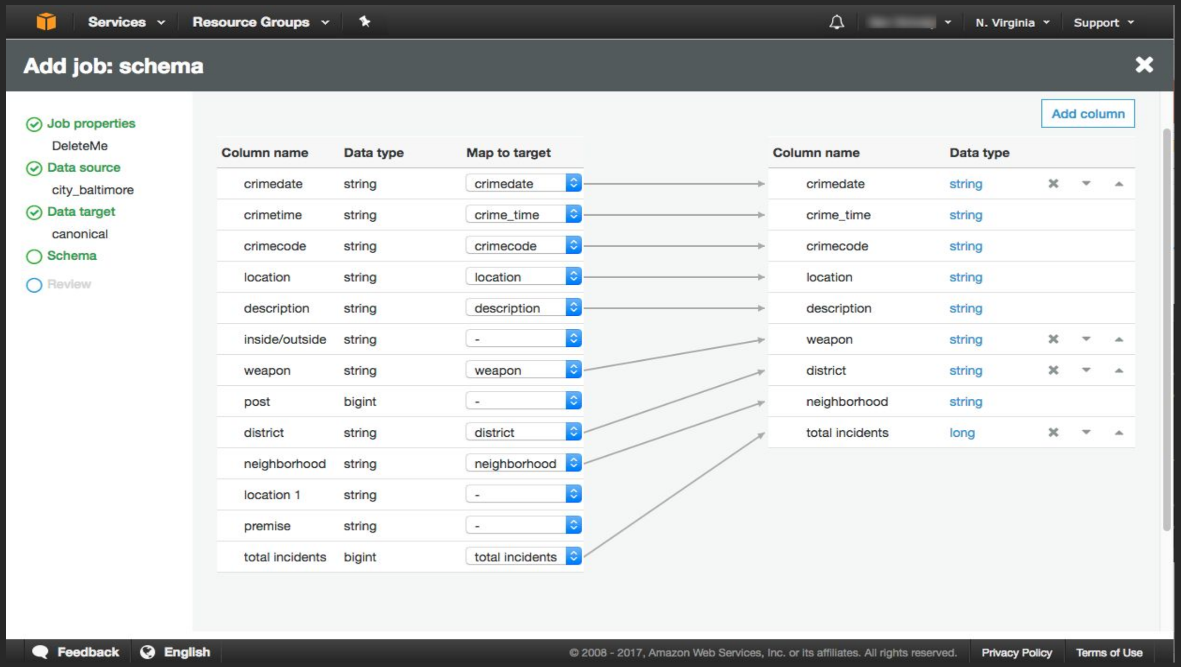
```
▼ STRUCT
    contributors_enabled:BOOLEAN
    description:STRING
    favourites_count:INT
    followers_count:INT
    friends_count:INT
    id:INT
    lang:STRING
    location:STRING
    name:STRING
    profile_background_tile:BOOLEAN
```

Job authoring in AWS Glue

- You have choices on how to get started

- Python code generated by AWS Glue
- Connect a notebook or IDE to AWS Glue
- Existing code brought into AWS Glue

Job authoring: Automatic code generation

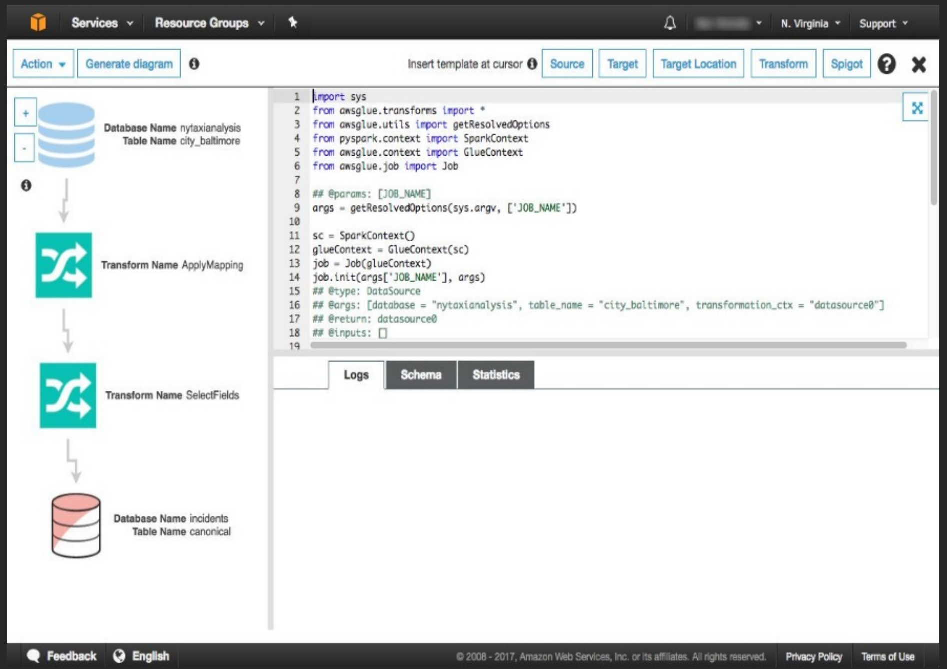


Add job: schema

Column name	Data type	Map to target
crimedate	string	crimedate
crimetime	string	crime_time
crimecode	string	crimecode
location	string	location
description	string	description
inside/outside	string	-
weapon	string	weapon
post	bigint	-
district	string	district
neighborhood	string	neighborhood
location 1	string	-
premise	string	-
total incidents	bigint	total incidents

Column name	Data type
crimedate	string
crime_time	string
crimecode	string
location	string
description	string
weapon	string
district	string
neighborhood	string
total incidents	long

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.



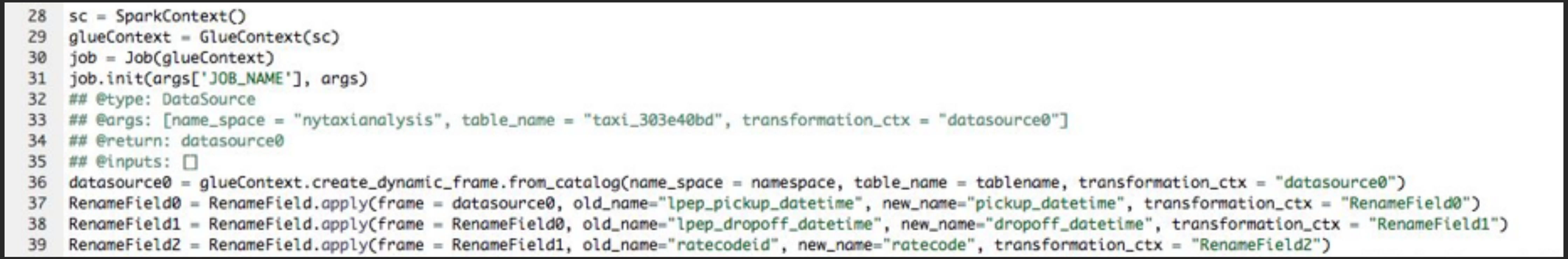
```python
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [database = "nytaxianalysis", table_name = "city_baltimore", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: []
```

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved.

1. Customize the mappings
2. AWS Glue generates transformation graph and **Python** code
3. Connect your **notebook** to development endpoints to customize your code

Job authoring: ETL code

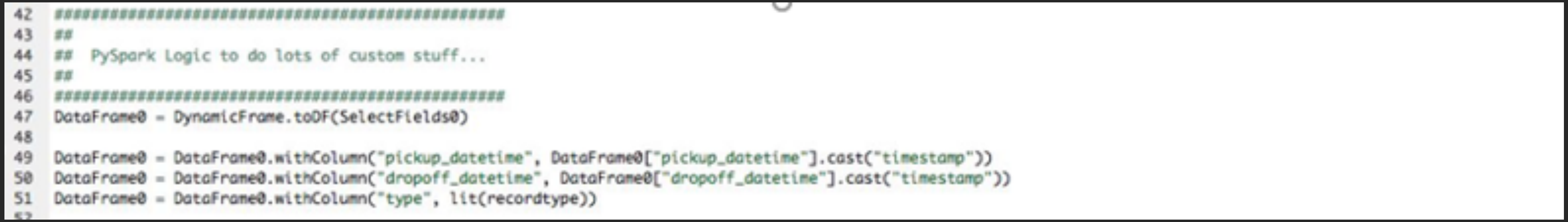
- **Human-readable**, editable, and portable PySpark code



```
sc = SparkContext()
glueContext = GlueContext(sc)
job = Job(glueContext)
job.init(args['JOB_NAME'], args)
## @type: DataSource
## @args: [name_space = "nytaxianalysis", table_name = "taxi_303e40bd", transformation_ctx = "datasource0"]
## @return: datasource0
## @inputs: □
datasource0 = glueContext.create_dynamic_frame.from_catalog(name_space = namespace, table_name = tablename, transformation_ctx = "datasource0")
RenameField0 = RenameField.apply(frame = datasource0, old_name="lpep_pickup_datetime", new_name="pickup_datetime", transformation_ctx = "RenameField0")
RenameField1 = RenameField.apply(frame = RenameField0, old_name="lpep_dropoff_datetime", new_name="dropoff_datetime", transformation_ctx = "RenameField1")
RenameField2 = RenameField.apply(frame = RenameField1, old_name="ratecodeid", new_name="ratecode", transformation_ctx = "RenameField2")
```

- **Flexible:** AWS Glue’s ETL library simplifies manipulating complex, semi-structured data

- **Customizable:** Use native PySpark, import custom libraries, and/or leverage AWS Glue’s libraries

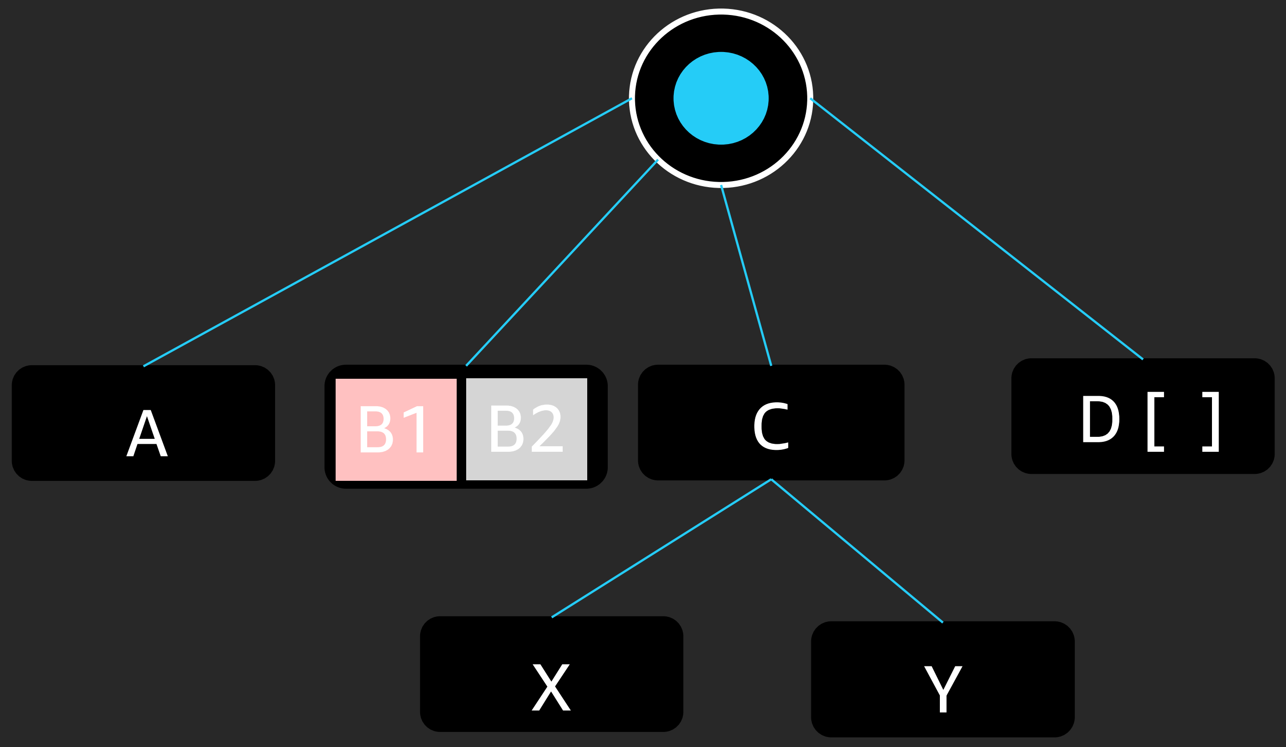


```
####################################################
##
##  PySpark Logic to do lots of custom stuff...
##
####################################################
DataFrame0 = DynamicFrame.toDF(SelectFields0)

DataFrame0 = DataFrame0.withColumn("pickup_datetime", DataFrame0["pickup_datetime"].cast("timestamp"))
DataFrame0 = DataFrame0.withColumn("dropoff_datetime", DataFrame0["dropoff_datetime"].cast("timestamp"))
DataFrame0 = DataFrame0.withColumn("type", lit(recordtype))
```

- **Collaborative:** Share code snippets via GitHub, reuse code across jobs

Job authoring: AWS Glue Dynamic Frames

Dynamic frame schema



Like Spark's Data Frames, but better for:

- Cleaning and (re)-structuring **semi-structured** data sets, e.g., JSON, Avro, Apache logs . . .

No upfront schema needed:

- Infers schema on the fly, enabling transformations in a **single pass**

Easy to handle the unexpected:

- Tracks new fields and inconsistent changing data types with **choices**, e.g., integer or string
- Automatically marks and separates error records

Job authoring: Leveraging the community

No need to start from scratch.

Use **AWS Glue samples** stored in GitHub to share, reuse, contribute: https://github.com/awslabs/aws-glue-samples

- Migration scripts to import existing Hive metastore data into AWS Glue Data Catalog
- Examples of how to use Dynamic Frames and Relationalize() transform
- Examples of how to use arbitrary PySpark code with AWS Glue's Python ETL library

Download **AWS Glue's Python ETL library** to start developing code in your IDE: https://github.com/awslabs/aws-glue-libs

Job execution: Scheduling and monitoring

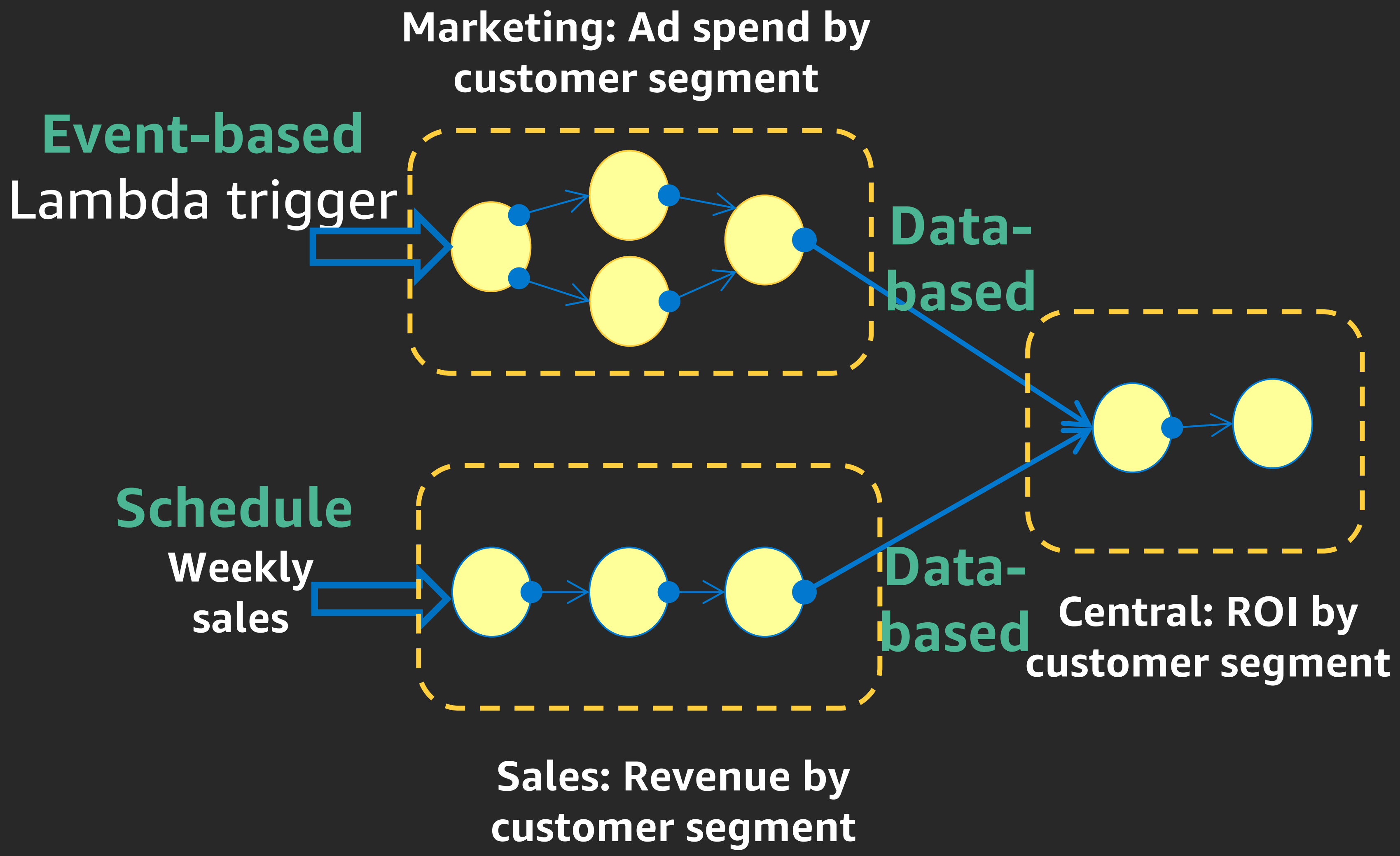
Compose jobs globally with event-based dependencies

- Easy to reuse and leverage work across organization boundaries

Multiple triggering mechanisms

- **Schedule-based:** e.g., time of day
- **Event-based:** e.g., job completion
- **On-demand:** e.g., Lambda
- **More :** Amazon S3 notifications, and Amazon CloudWatch Events

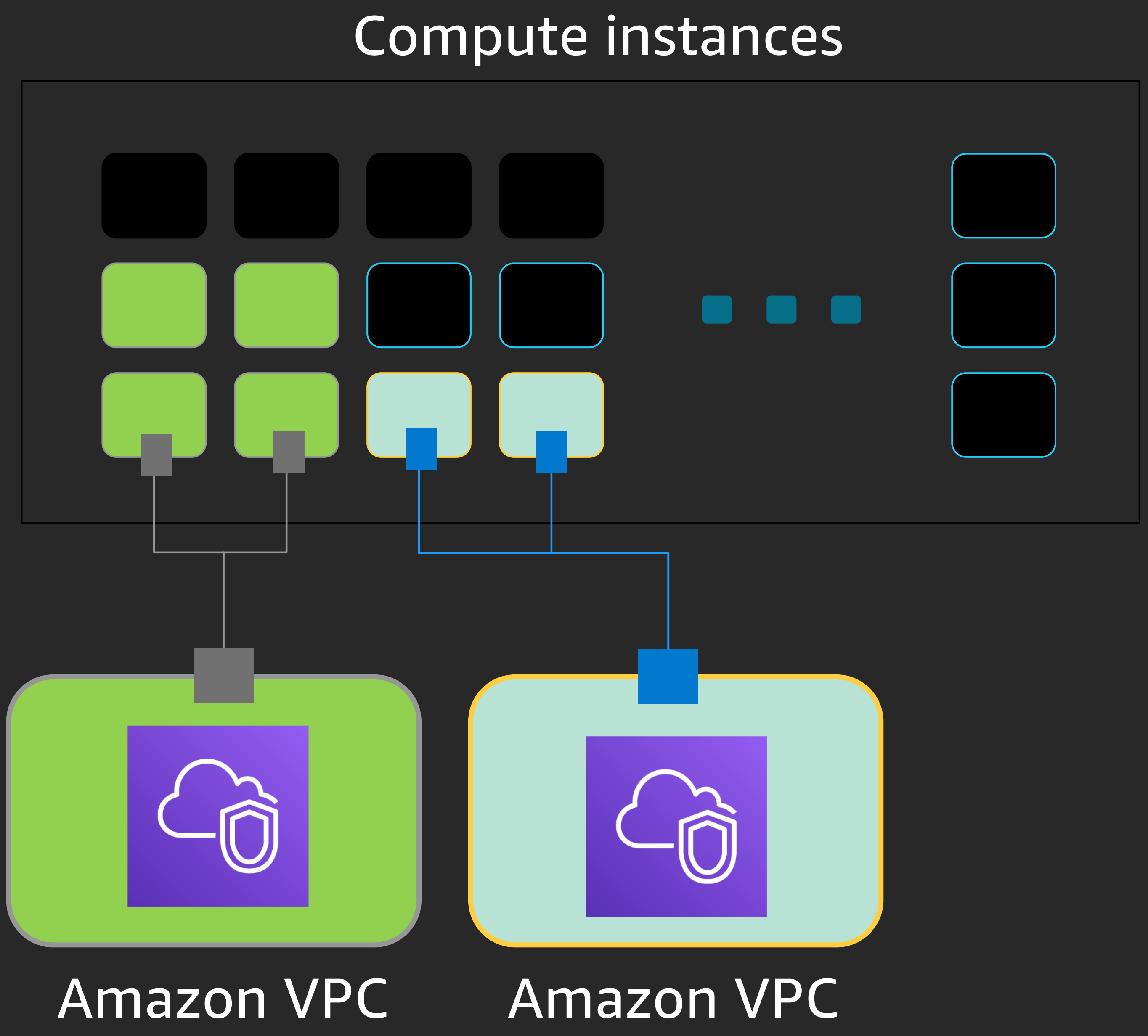
Logs and alerts are available in CloudWatch


Marketing: Ad spend by customer segment
Event-based
Lambda trigger
Data-based
Schedule
Weekly sales
Data-based
Central: ROI by customer segment
Sales: Revenue by customer segment

Job execution: Serverless

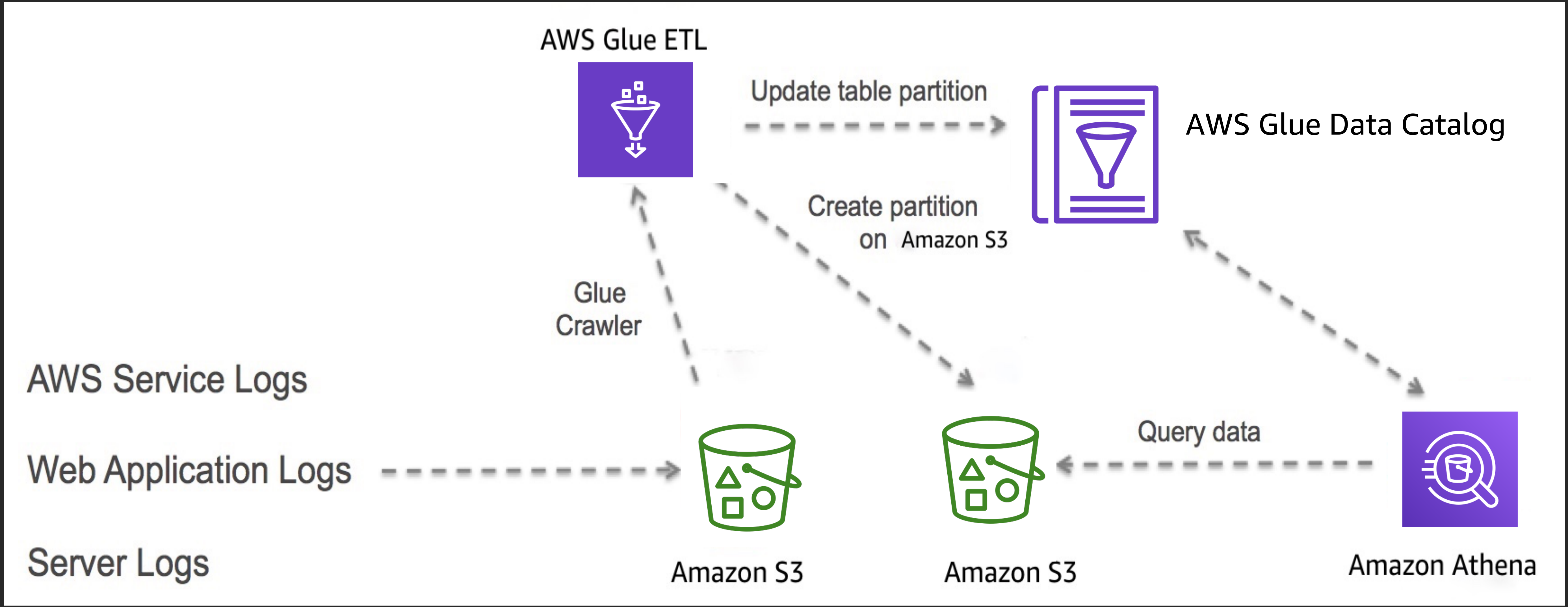
There is no need to provision, configure, or manage servers

- Auto-configure VPC and role-based access
- Customers can specify the capacity that gets allocated to each job
- Automatically scale resources (on post-GA roadmap)
- You pay only for the resources you consume while consuming them

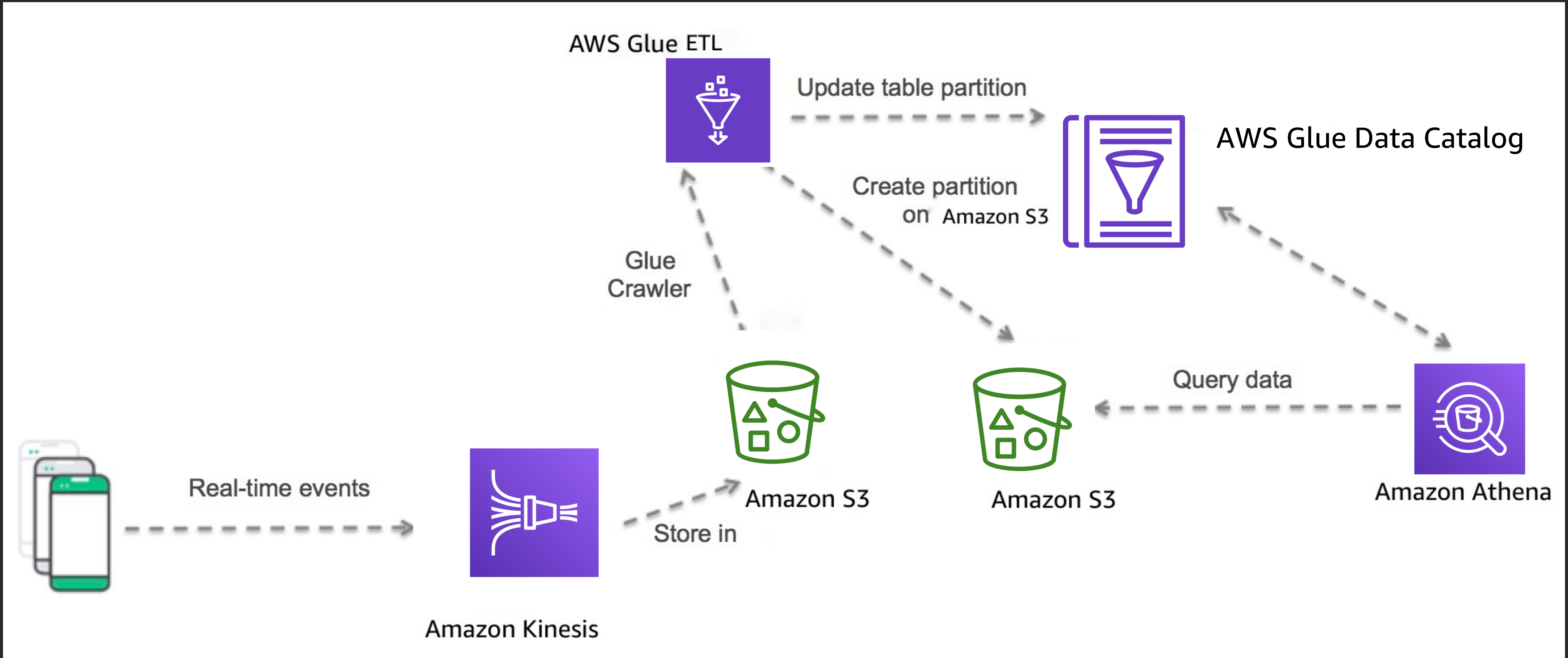


Common customer use cases

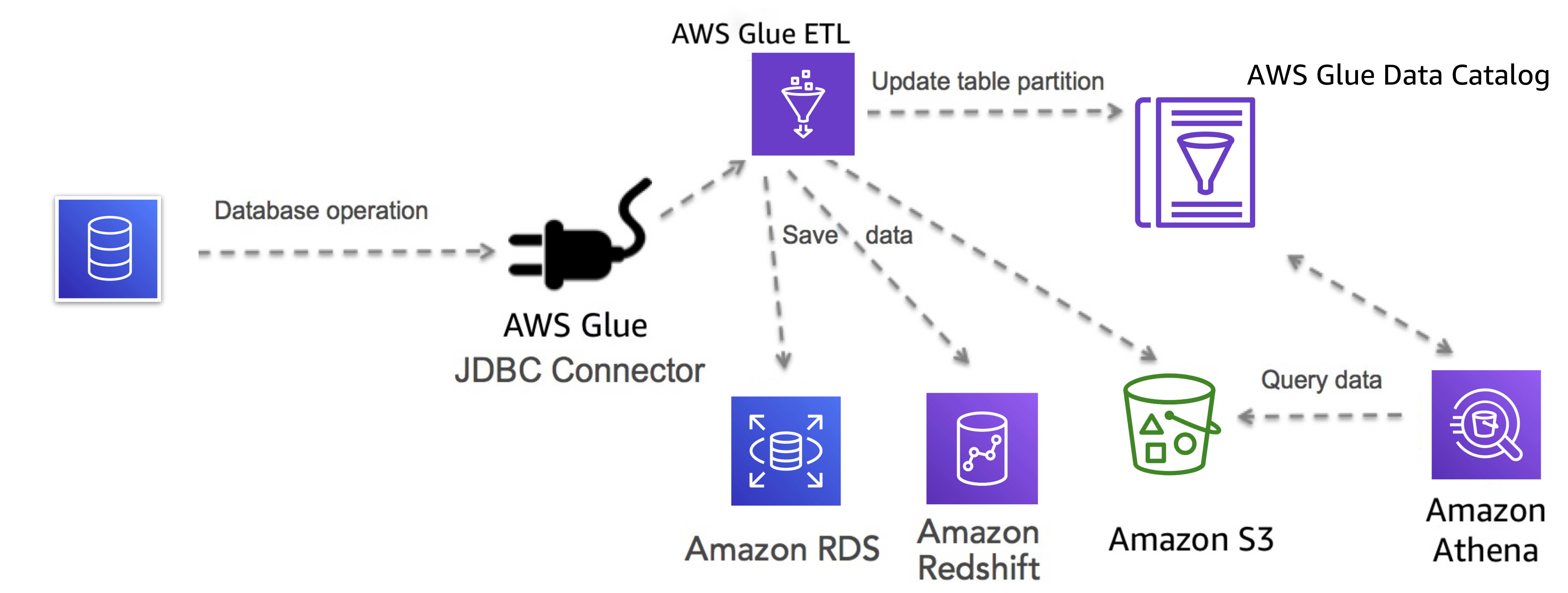
Log aggregation with AWS Glue ETL



Real-Time data collection with Glue ETL



Data import using Glue database connectors



Serverless processing using Lambda

Benefits of Lambda

Productivity-focused compute platform to build powerful, dynamic, modular applications in the cloud

1
No infrastructure to manage

Focus on business logic

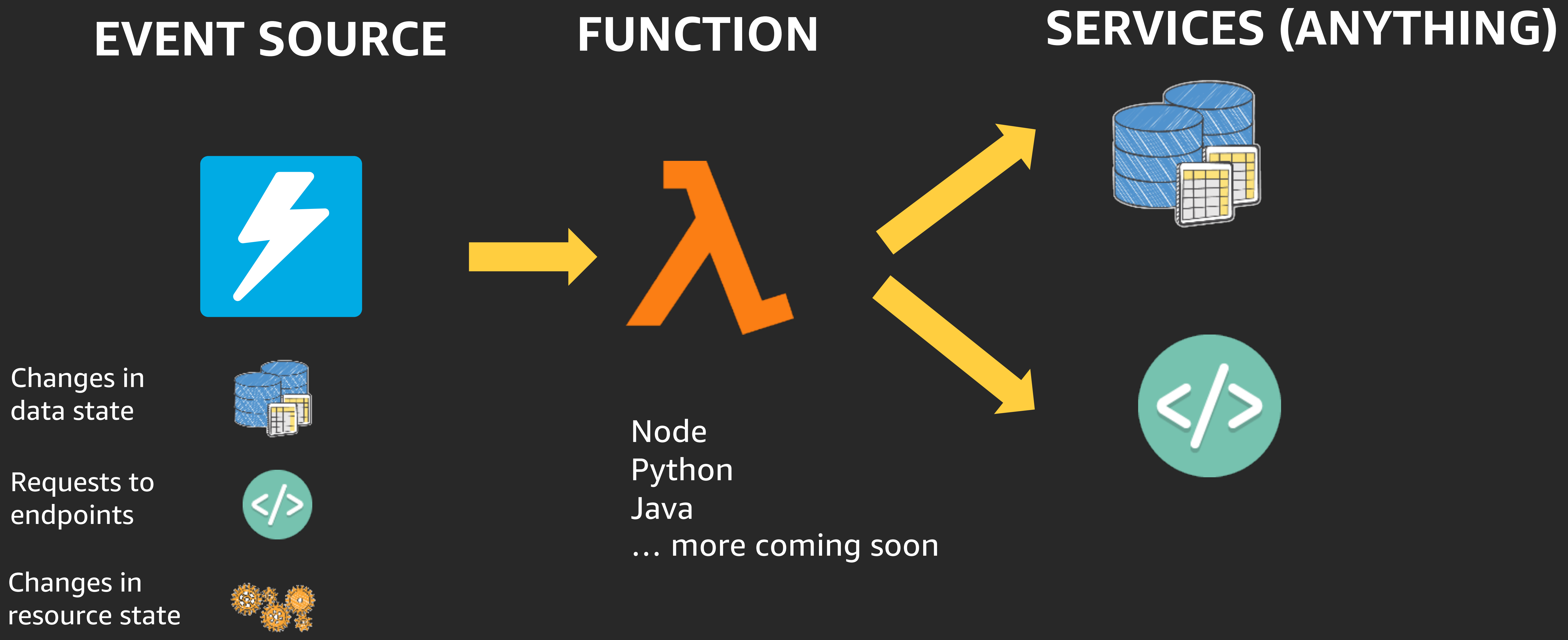
2
Cost-effective and efficient

Pay only for what you use

3
Bring your own code

Run code in standard languages

Application components for serverless apps



EVENT SOURCE

FUNCTION

SERVICES (ANYTHING)

Changes in
data state

Requests to
endpoints

Changes in
resource state

Node
Python
Java
… more coming soon

Event sources that integrate with Lambda

DATA STORES

- Amazon S3
- DynamoDB
- Kinesis
- Amazon Cognito
- Amazon RDS Aurora New

ENDPOINTS

- Amazon Alexa
- API Gateway
- AWS IoT

REPOSITORIES

- AWS CloudFormation
- CloudTrail
- CloudWatch

EVENT/MESSAGE SERVICES

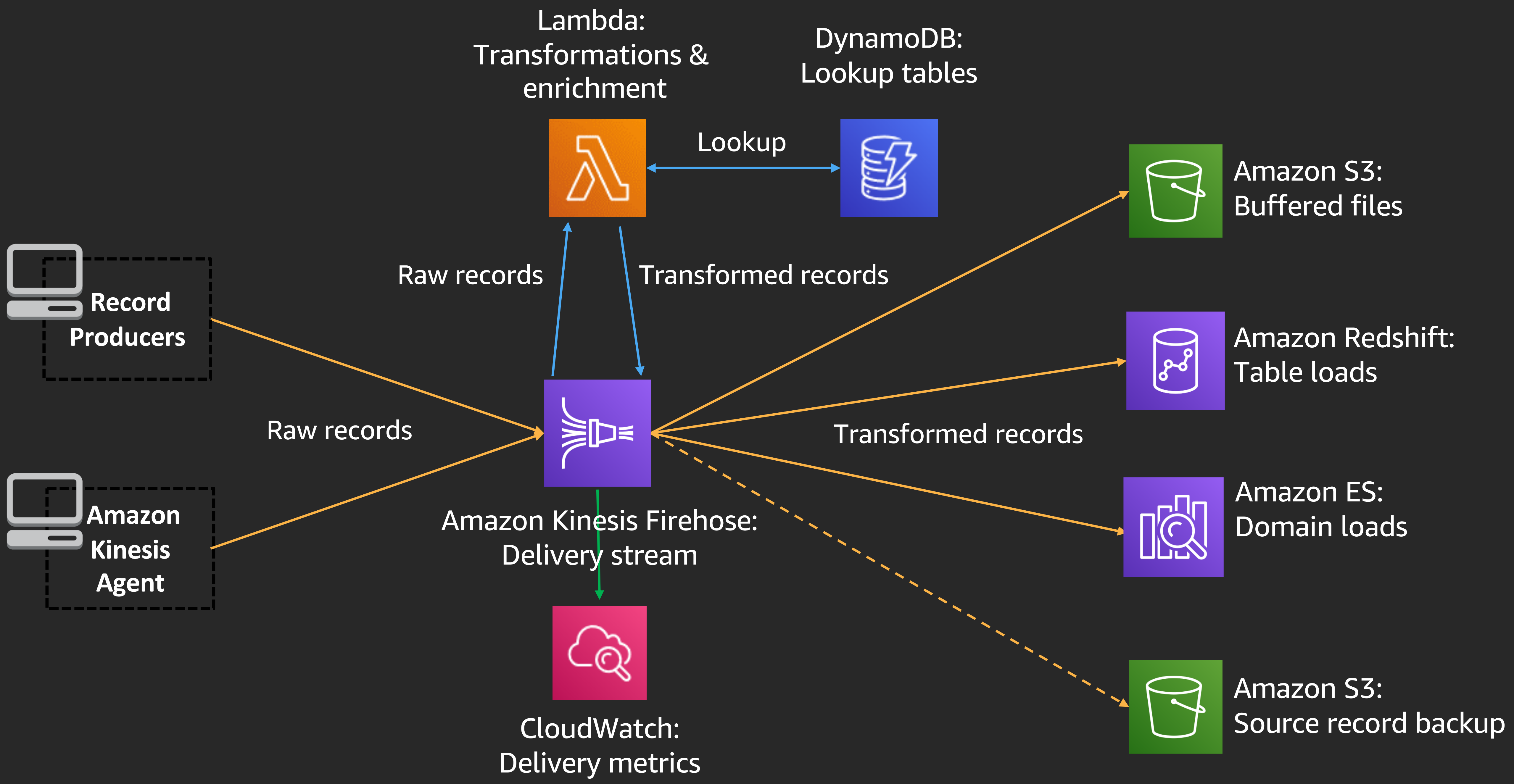
- Amazon SES
- Amazon SNS
- Cron events

ORCHESTRATION AND STATE MANAGEMENT

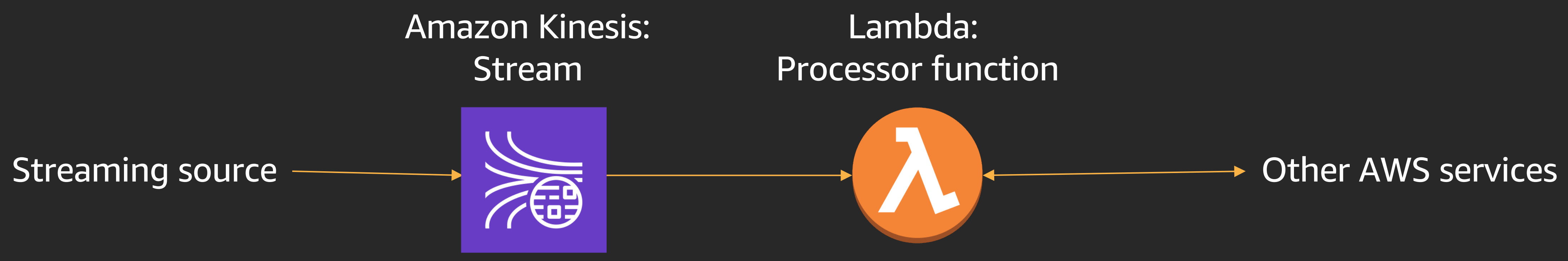
- AWS Step Functions

… and the list will continue to grow!

Lambda use case for streaming data ingestion



Amazon Kinesis Streams and Lambda

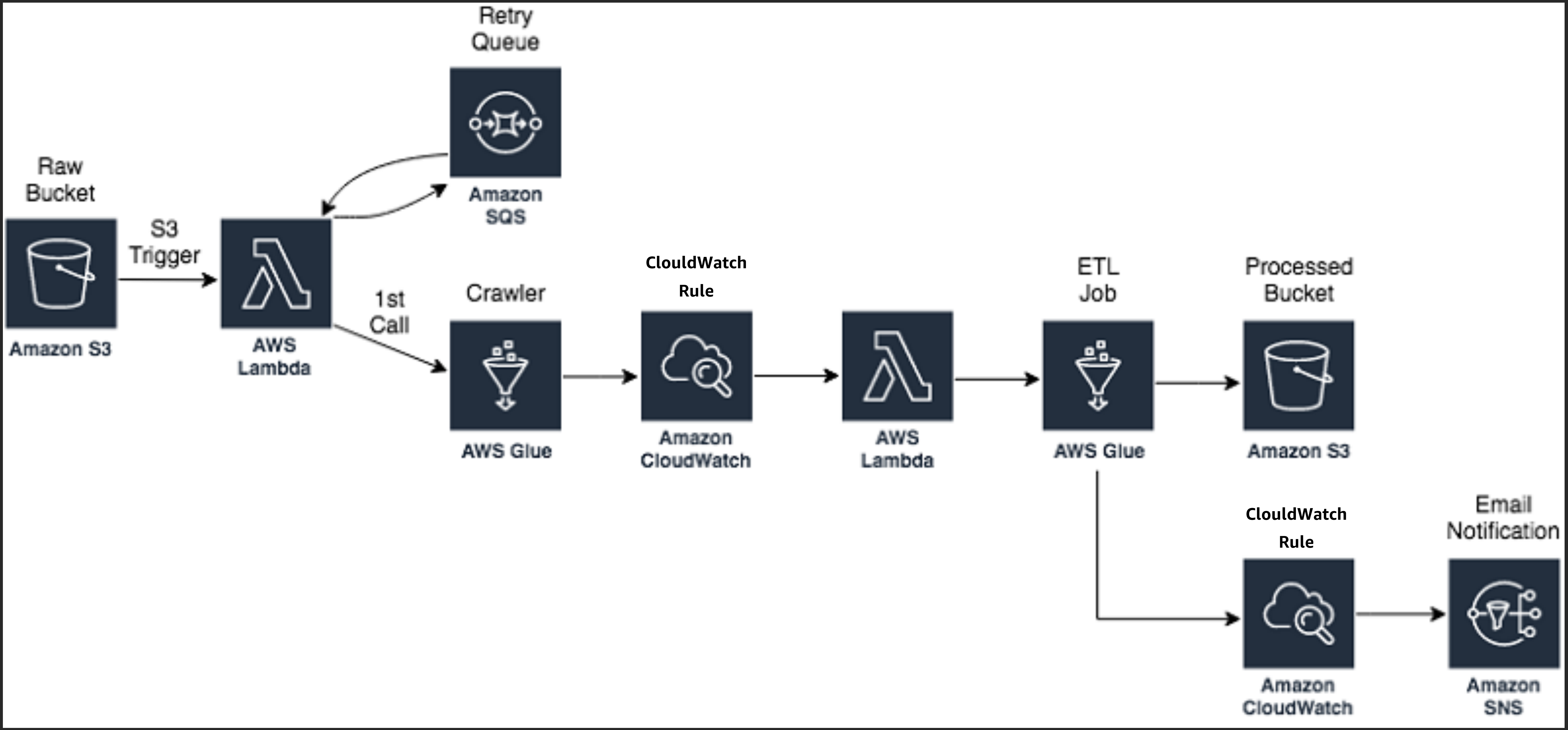


- Number of Amazon Kinesis Streams shards corresponds to concurrent invocations of Lambda function

- Batch size sets maximum number of records per Lambda function invocation

Serverless data lake architecture

Serverless data lake architecture



Steps in building a serverless data lake

1. Ingest data into Amazon S3
2. Configure an Amazon S3 event trigger
3. Automate the Data Catalog with an AWS Glue crawler
4. Author ETL jobs
5. Automate ETL job execution
6. Monitor with CloudWatch Events

Serverless data lake blog post reference

https://aws.amazon.com/blogs/big-data/build-and-automate-a-serverless-data-lake-using-an-aws-glue-trigger-for-the-data-catalog-and-etl-jobs/

Data lakes and analytics

More than 10,000 data lakes on AWS

FINRA
NASDAQ
pitney bowes
yelp
EA ELECTRONIC ARTS
hudl
airbnb
MONSTER WORLDWIDE
CROWDSTRIKE
Expedia
NTT docomo
theguardian
Nextdoor
AMGEN
A&E
JustGiving
SK planet
News Corp
Zillow
edmunds.com
FOURSQUARE
lyft
Schumachergroup
gogo
Pfizer
ZUMBA
dataxu
Capital One
ACQUIA
COMCAST
FT FINANCIAL TIMES
Instacart
mobilewalla
reddit
Bristol-Myers Squibb
scopely
SCHOLASTIC
DOMO
CIVITAS LEARNING
REDFIN
sling TELEVISION
Agero
21ST CENTURY FOX
change.org
amazon
Sysco
euclid
ROVIO
CIVIS ANALYTICS
Pinterest
ACTIVISION BLIZZARD
illumina
ticketmaster
Viber Connect. Freely.
broadsoft
PHILIPS
Discovery COMMUNICATIONS
Instapage
FUNNY OR DIE
FAST RETAILING
imshealth
realnetworks
ost
boingo
HEARST
mlbam
Vanguard
SONOS
hello
Adobe
enel
OLX Group
ATLASSIAN
HBO
slack
Gunosy
TUNE
TAXI
FULL 360
coffee meets bagel

AWS Partners

accenture
databricks
Qubole
talend
Microsoft
47 Lining
A Hitachi Vantara Company
NorthBay
DXC.technology
mongoDB

snowflake
cloudera
Cloudwick
PARIVEDA SOLUTIONS
DATASTAX
MATILLION
Paxata
New Relic.
slalom

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING
AQUILENT
TERADATA
Informatica
DOMO
DATA REPUBLIC
CISCO
sumologic
Adobe

clearscale
credibiliT
CloudMas
ScienceLogic
druva
SAP
splunk>
ZALONI
THE DATA LAKE COMPANY
redhat

CORPINFO
matrix
International Financial Services
MZ
MEGAZONE
Cloud Innovator
SUSE
blazeclan
CITRIX
intel
looker
sas
Cognizant

Dedalus
CSC
REAN CLOUD
Reply
tableau
software
MicroStrategy
ATTUNITY
ORACLE

CLOUDNEXA
Booz | Allen | Hamilton
strategy and technology consultants
BULLETPROOF
emind
YOUR CLOUD EXPERTS
DATAGUISE
NetApp
Qlik
TRIFACTA

Thank you!

Aditya Challa

aditchal@amazon.com

AWS re:Invent

aws