

AWS
re:Invent

ANT308

Building Serverless Analytics Pipelines with AWS Glue

Mehul A. Shah
Senior Software Development Manager
AWS Glue

Arup Ray
Vice President, Engineering
Realtor.com

AWS
re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Today's Agenda

AWS Glue overview

What's new: improvements and features

Realtor.com – customer success

AWS Glue overview

A year ago ...

Fully-managed, serverless extract-transform-load (ETL) service

for developers, built by developers

1000s of customers and jobs

A year ago ...

NTT
docomo

News Corp



my**Tomorrows**



Select AWS Glue customers ...

NTT
docomo

News Corp

finA[↑]ccel

realtor.com®

BEESWAX 

 **Expedia**®


MediaMath

 **HappyFresh**
JOY DELIVERED!

DOWJONES


AUTODESK

Pilot
FLYING 

 **MERCK**

 **OLX** Group

fox.

emetriQ Deutsche Telekom Gruppe 

 **UNICORN**
REDEFINE DIGITAL MARKETING

 **apps
associates**
extreme expertise

FULL
360 

myTomorrows

DataXu

amazon
 **Prime Air**

 **apps
associates**
extreme expertise

 **21ST
CENTURY
FOX**

 **Upserve**

 **ost**

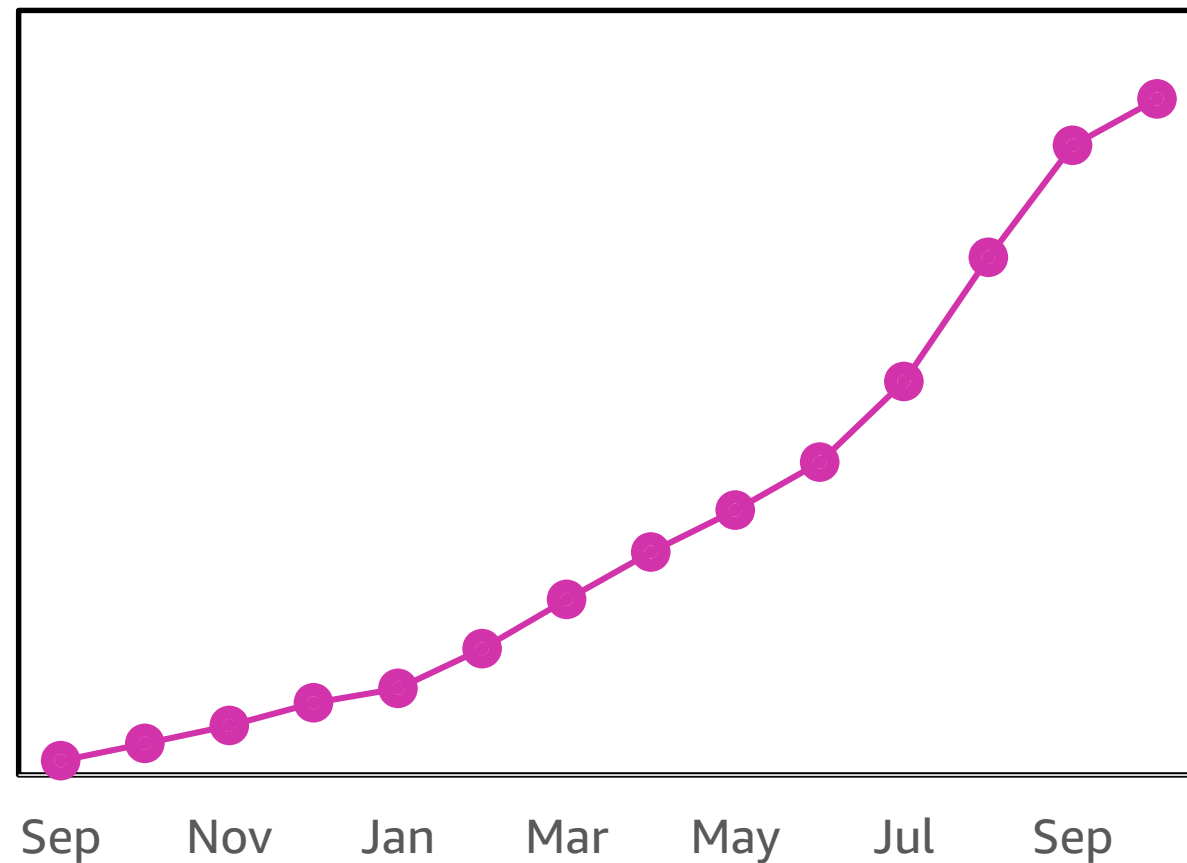
AWS
re:Invent

© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

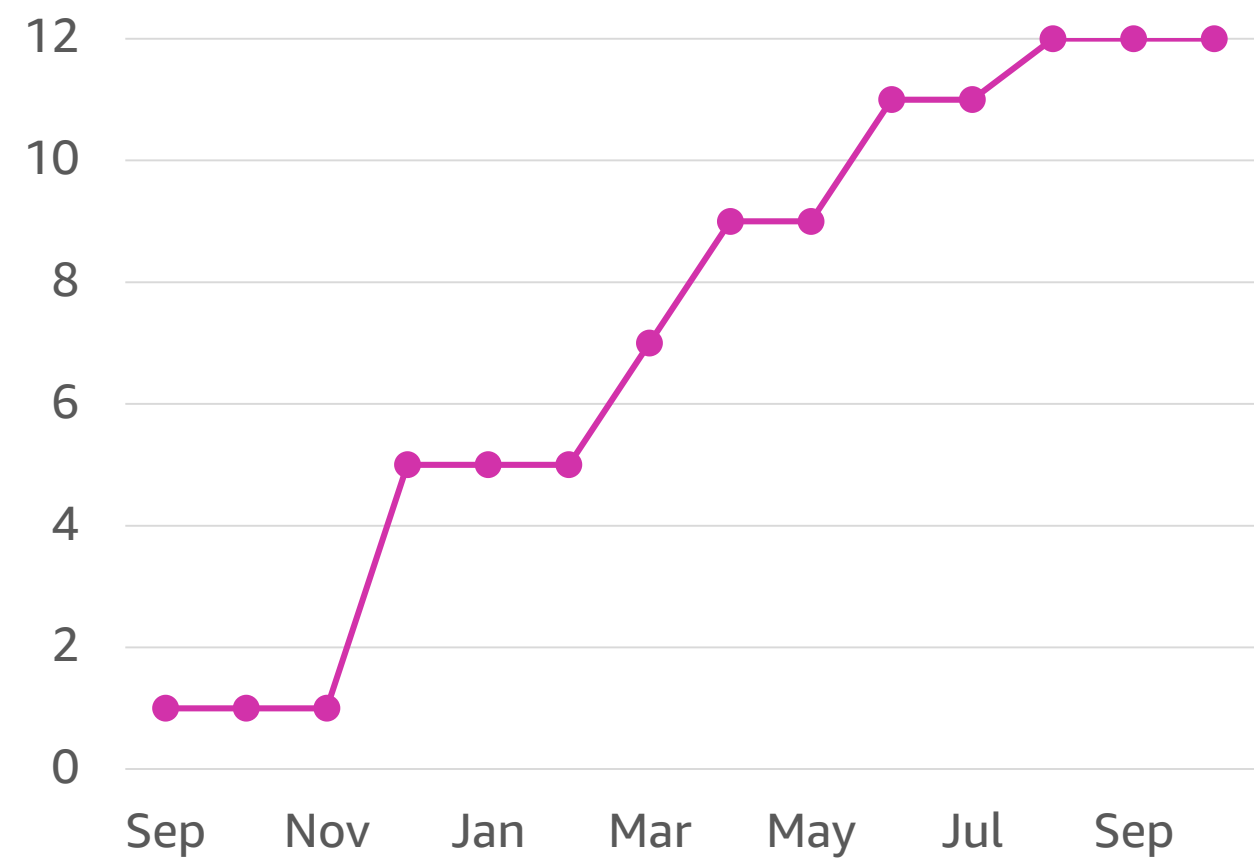
aws


You helped us grow ...

Job and Crawler Runs



of Regions



We've been busy...

Access policies for AWS
Glue Data Catalog

Amazon SageMaker
notebooks

Encryption
at rest

Canada central
region

Crawler combine
compatible schemas

ETL job
metrics

Amazon DynamoDB
integration

Job delay
notification

London
region

Seoul
region

Crawler merge
new columns

Mumbai
region

Support Apache
Spark 2.2.1

ETL job
timeout

Sydney
region

30+
major
features

Frankfurt
region

Readers support
JSONPath expressions

New Job
Events Types

Support for
Scala scripts

Tokyo
region

Singapore
region

Crawler CWE
notifications

XML support

AWS CloudTrail
support

Amazon CloudFormation
templates

Crawler exclusion
patterns

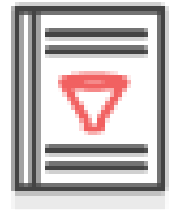
Per-second
billing

DynamicFrame
Filter and Map

GDPR, HIPAA, and BAA
compliance

Ireland, Oregon,
Ohio region

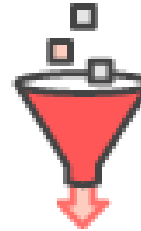
AWS Glue components



Data Catalog

Discover

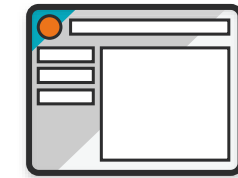
- Automatic crawling
- Apache Hive Metastore compatible
- Integrated with AWS analytic services



Serverless Jobs

Develop

- Apache Spark core
- Python and Scala
- Auto-generates ETL code



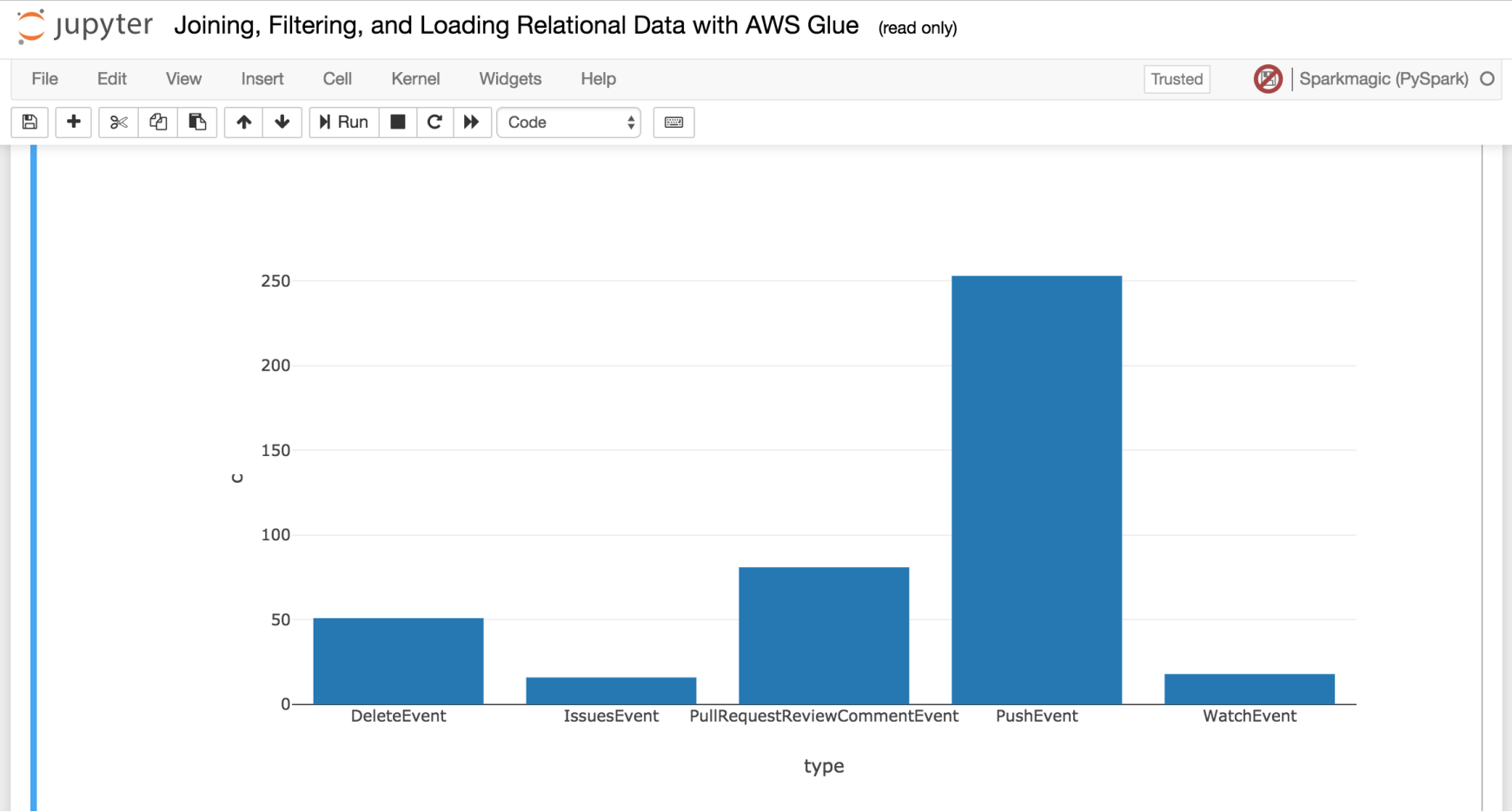
Orchestration

Deploy

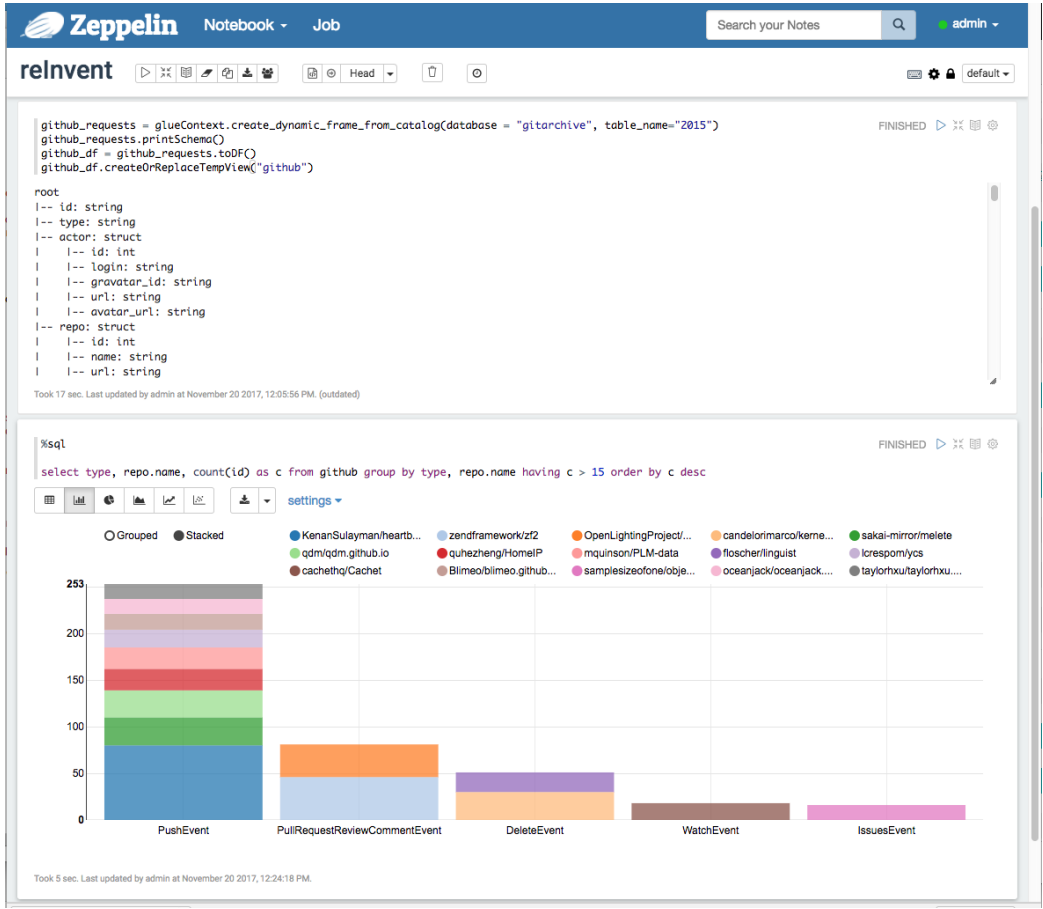
- Flexible scheduling
- Monitoring and alerting
- External integrations

Beyond ETL: data science and exploration

Amazon SageMaker integration



Other notebooks



What you're saying ...

"Beeswax uses Amazon S3 and AWS Glue Data Catalog to build a highly reliable **data lake** that is fully managed by AWS. Our platform leverages the ... integration with **Amazon EMR** in **Hive** and **SparkSQL** ..."

Ram Kumar Regnaswamy, CTO, Beeswax

"... ETL-ing data from our **data lake** to our **Redshift** warehouse is just one of use case examples of AWS Glue. ... Being **cost-effective** is essential. ... AWS Glue has enabled our small team of data engineers to run the **whole data infrastructure** in our company."

Umang Rustagi, Co-founder and COO, FinAccel

"Glue ETL performed **200% faster** than traditional ETL tools with no operational overhead due to the serverless nature of Glue ETL and **at a fraction of the cost.**"

Miki Hardisty, CTO, Jack-in-the-box

What's new: improvements and features

Innovation areas

Scalability and performance

Crawlers and ETL jobs

Profiling and debugging

ETL metrics

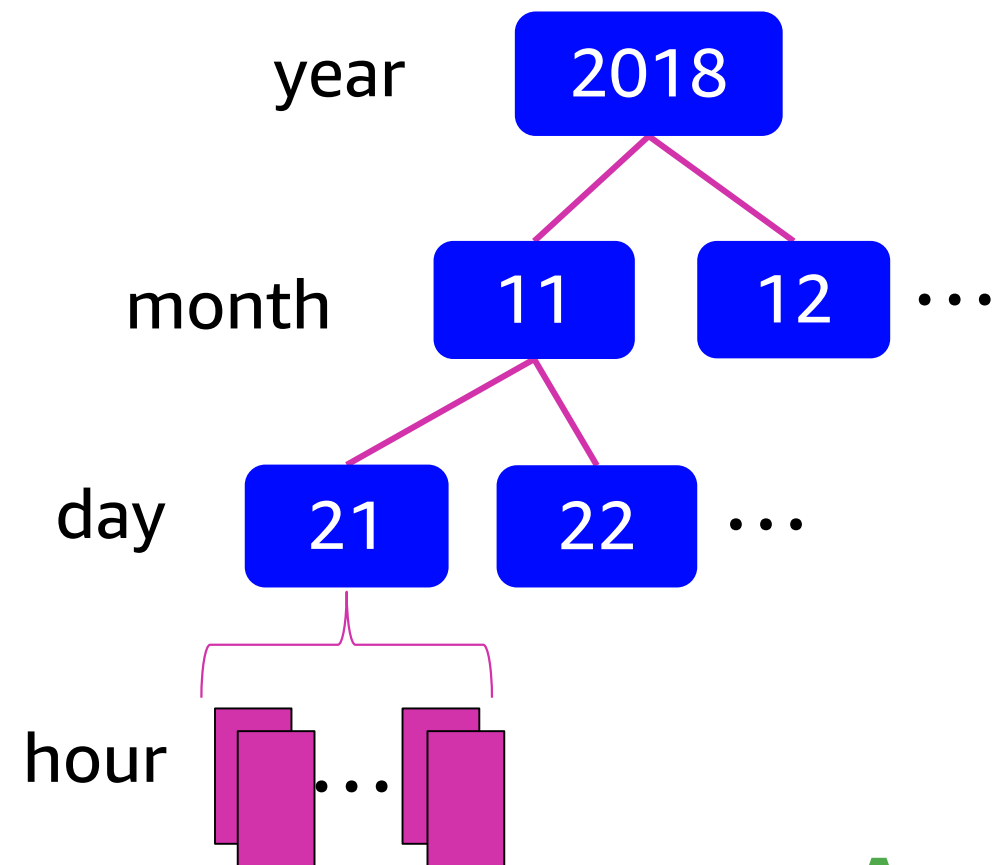
Interoperability

Orchestration features

Announcing a new job type: Python shell

Example data organization in: Amazon Simple Storage Service (Amazon S3)

source Amazon S3 bucket



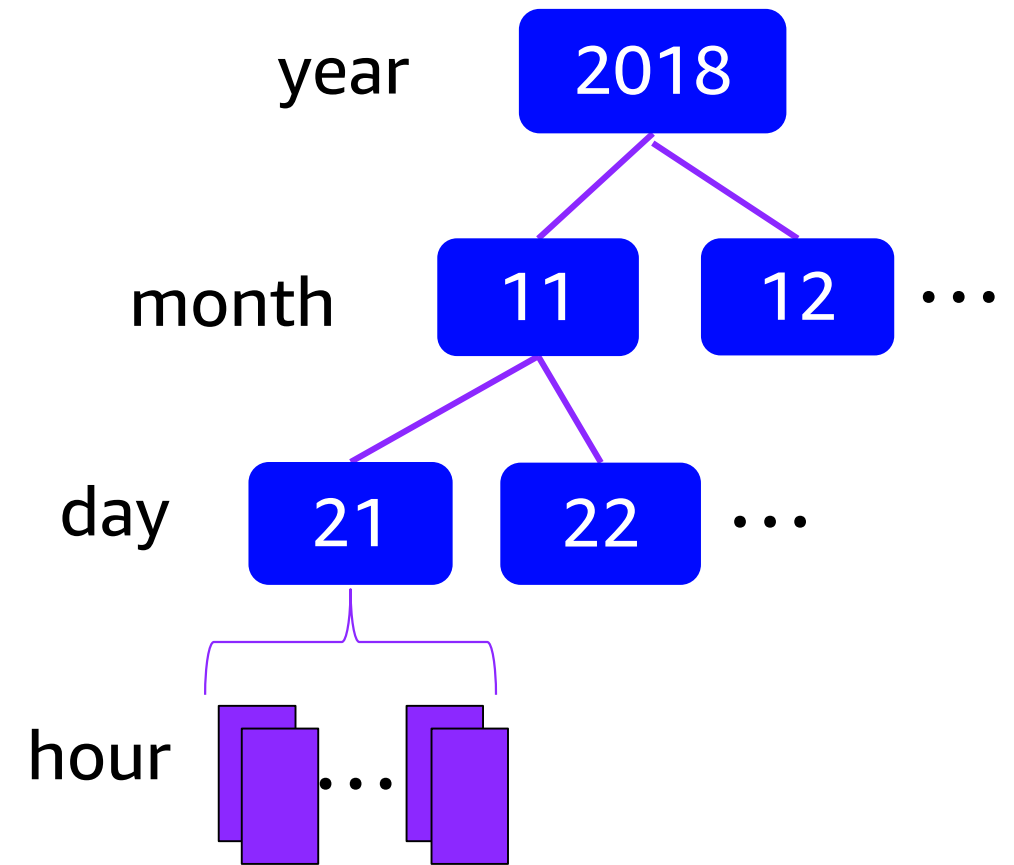
JSON



transform

Apache Hive-
style partitions

target Amazon S3 bucket



Parquet

Crawler results ...

Services

Resource Groups

aws

Feedback

English (US)

Tables

Connections

Crawlers

Classifiers

ETL

Jobs

Triggers

Dev endpoints

Tutorials

Add crawler

Explore table

Add job

Resources

Name

githubevents_data

Description

gitarchive

Database

gitarchive

Classification

json

Location

s3://glue-sample-datasets/examples/da

Connection

No

Deprecated

No

Last updated

Wed Nov 22 07:52:09 GMT-800 2017

Input format

org.apache.hadoop.mapred.TextInputFo

Output format

org.apache.hadoop.hive.qi.io.HiveIgnore

Serde serialization lib

org.openx.data.jsonserde.JsonSerDe

Serde parameters

paths actor,created_at,id,org,paylo

Table properties

CrawlerSchemaSerializerVersion 1.0

CrawlerSchemaDeserializerVersion

Schema

Showing: 1 - 11 of 11

	Column name	Data type	Key
1	id	string	
2	type	string	
3	actor	struct	
4	repo	struct	
5	payload	struct	
6	public	boolean	
7	created_at	string	
8	org	struct	
9	year	string	Partition (0)
10	month	string	Partition (1)
11	day	string	Partition (2)

payload schema details

STRUCT

push_id:INT

size:INT

distinct_size:INT

ref:STRING

head:STRING

before:STRING

commits:ARRAY

element:STRUCT

sha:STRING

author:STRUCT

name:STRING

email:STRING

message:STRING

distinct:BOOLEAN

url:STRING

ref_type:STRING

Close

Complex, nested fields

Groups files into Apache Hive-style partitions

Services

Resource Groups

aws

Feedback

English (US)

Developer/mashah-Isengard @...

Tables > githubevents_data

Last updated 22 Nov 2017

Table Version (Current version)

Edit table

Delete table

Close partitions

Compare versions

Edit schema

Showing: 1 - 100

year	month	day		
2017	02	15	View files	View properties
2017	03	12	View files	View properties
2017	05	17	View files	View properties
2017	10	12	View files	View properties
2017	12	18	View files	View properties
2017	01	09	View files	View properties
2017	03	07	View files	View properties
2017	06	28	View files	View properties

Feedback

English (US)

Privacy Policy

Terms of Use

Crawler performance

7x improvement YoY

90M+ files per day

Millions of partitions

YMMV with partitions
size and **complexity**

AWS Glue ETL scripts

Job: github_2_csv Action Save Run job Generate diagram

Insert template at cursor Source Target Target Location Transform Spigot

Database Name gitarchive
Table Name 2015

Transform Name ApplyMapping

Transform Name ResolveChoice

Transform Name DropNullFields

Path s3://glue-sample-target/output-dir

```
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 |
8 |## @params: [JOB_NAME]
9 |args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10 |
11 |sc = SparkContext()
12 |glueContext = GlueContext(sc)
13 |spark = glueContext.spark_session
14 |job = Job(glueContext)
15 |job.init(args['JOB_NAME'], args)
16 |## @type: DataSource
17 |## @args: [database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0"]
18 |## @return: datasource0
19 |## @inputs: []
20 |datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0")
21 |## @type: ApplyMapping
22 |## @args: [mapping = [("id", "string", "id", "string"), ("type", "string", "type", "string"), ("actor.login", "string", "actor", "string"), ("repo.name", "string", "repo", "string")]]
23 |## @return: applymapping1
24 |## @inputs: [frame = datasource0]
25 |applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [("id", "string", "id", "string"), ("type", "string", "type", "string"), ("actor.login", "string", "actor", "string"), ("repo.name", "string", "repo", "string")])
26 |## @type: ResolveChoice
27 |## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
28 |## @return: resolvechoice2
29 |## @inputs: [frame = applymapping1]
30 |resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
31 |## @type: DropNullFields
32 |## @args: [transformation_ctx = "dropnullfields3"]
33 |## @return: dropnullfields3
34 |## @inputs: [frame = resolvechoice2]
35 |dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
36 |## @type: DataSink
37 |## @args: [connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet"]
38 |## @return: datasink4
39 |## @inputs: [frame = dropnullfields3]
40 |datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet")
41 |job.commit()
```

Read Dynamic Frame from source

Data transformation + data cleaning functions

Write Dynamic Frame to sink

Logs Schema

Review: Apache Spark and AWS Glue ETL

SparkSQL	AWS Glue ETL
Apache Spark DataFrames	AWS Glue DynamicFrames
Apache Spark Core: RDDs	

Application

Data Structure

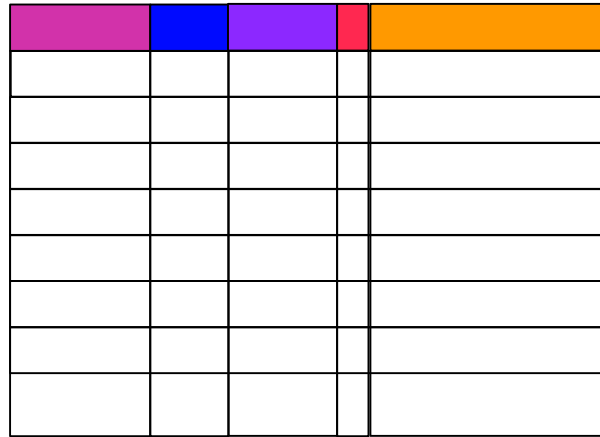
Execution

Apache Spark is a distributed data processing engine for complex analytics.

AWS Glue builds on the Apache Spark to offer ETL specific functionality.

DataFrames and DynamicFrames

DataFrames



Core data structure for SparkSQL

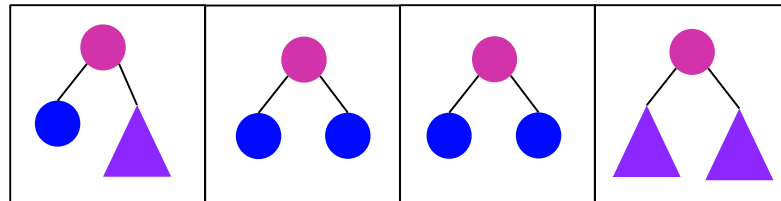
Like structured tables

Need schema up-front

Each row has same structure

Suited for SQL-like analytics

DynamicFrames



Like DataFrames for ETL

Designed for processing **semi-structured** data,
e.g. JSON, Avro, Apache logs ...

DynamicFrame performance

Raw conversion speed

4x improvement YoY

1TB in under **1.5 hours**
with 10 DPU

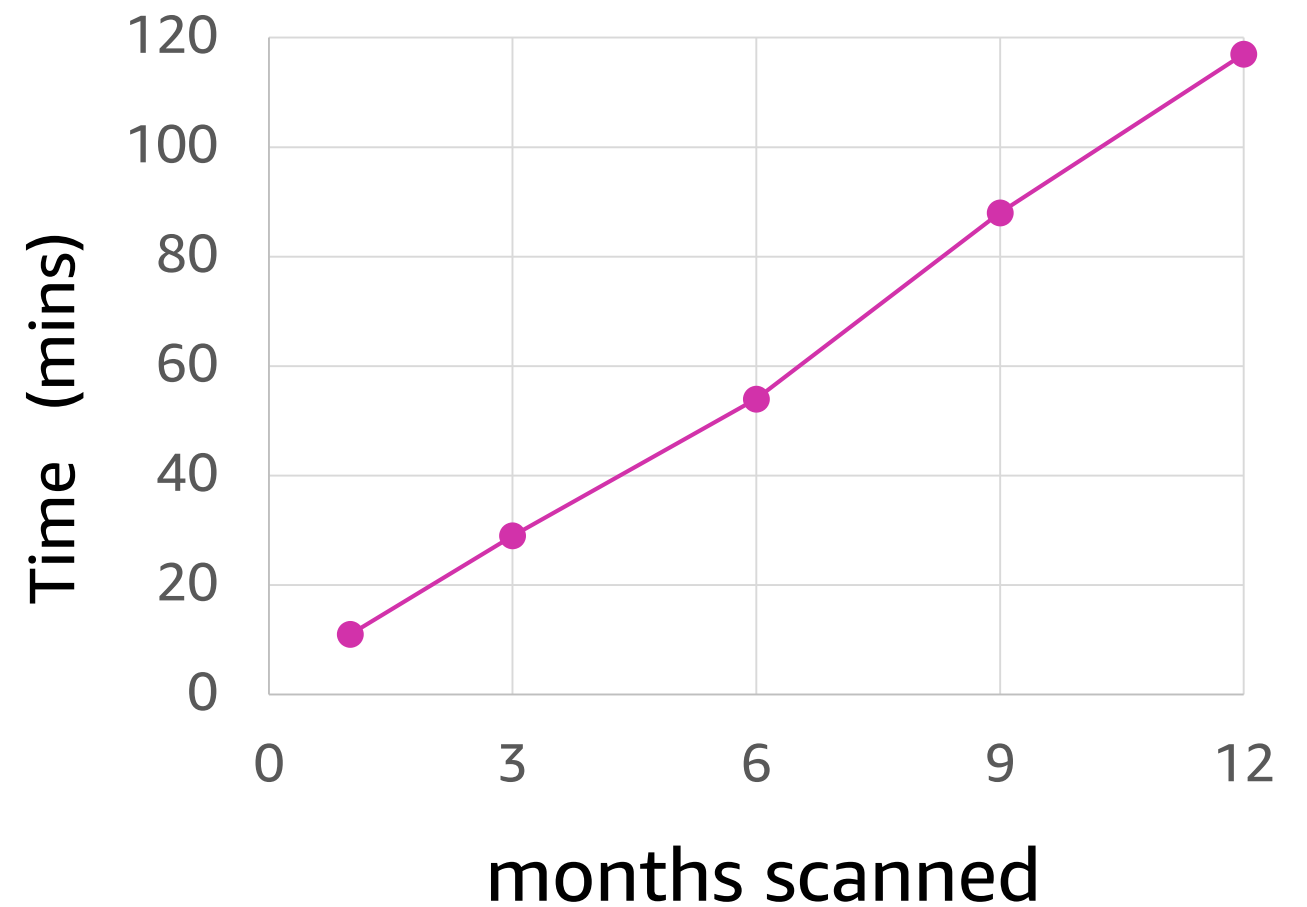
YMMV with **data format**
and script **complexity**

DynamicFrame predicate pushdown

Example code

```
glueContext
  .getCatalogSource(
    database = "gitarchive",
    tableName = "githubevents_data",
    pushDownPredicate =
      "(year=='2017' and month=='04')")
  .getDynamicFrame()
```

GitHub public timeline



Innovation areas

Scalability and performance

Crawlers and ETL jobs

Profiling and debugging

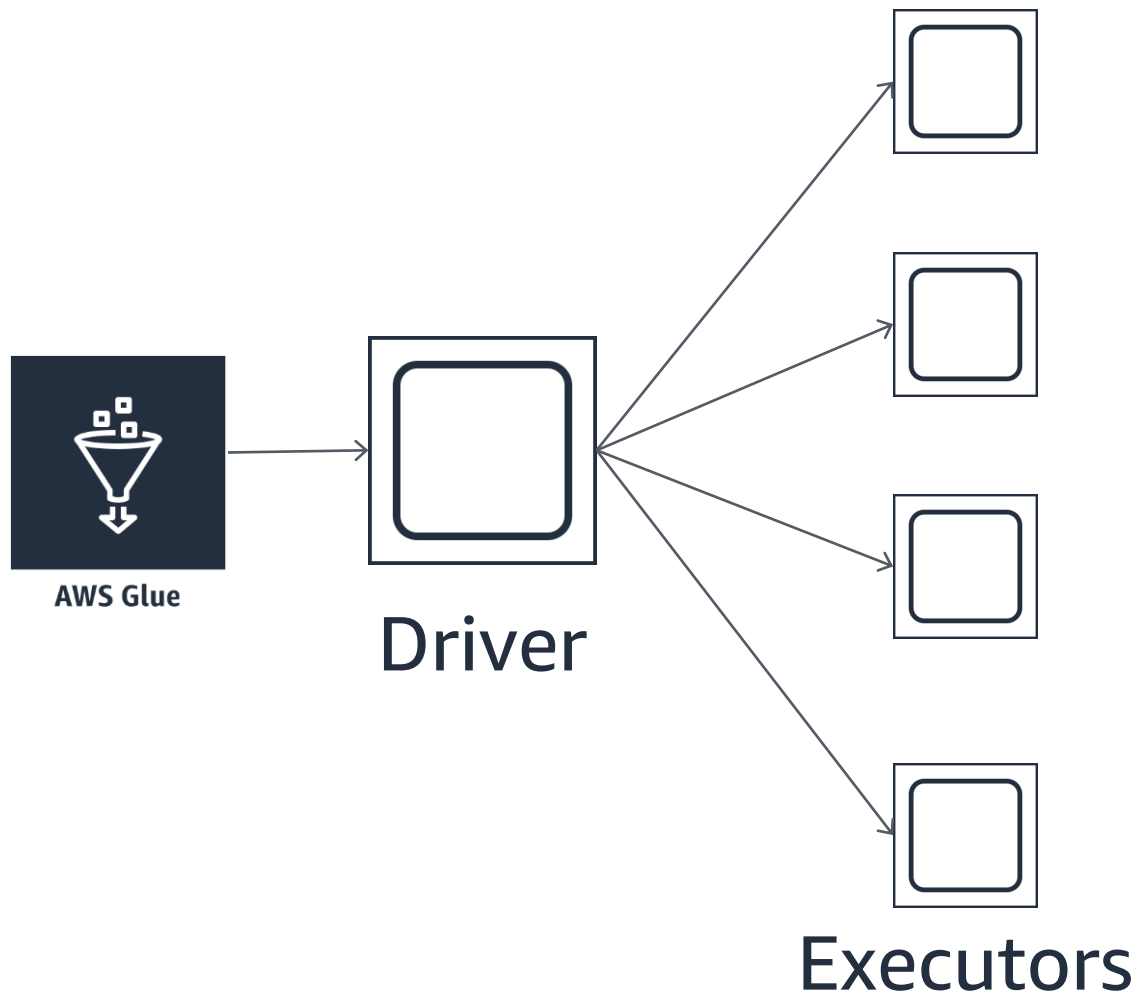
ETL metrics

Interoperability

Orchestration features

Announcing a new job type: Python shell

AWS Glue execution model



Apache Spark and AWS Glue are *data parallel*. Data is divided into *partitions* (shards) that are processed concurrently.

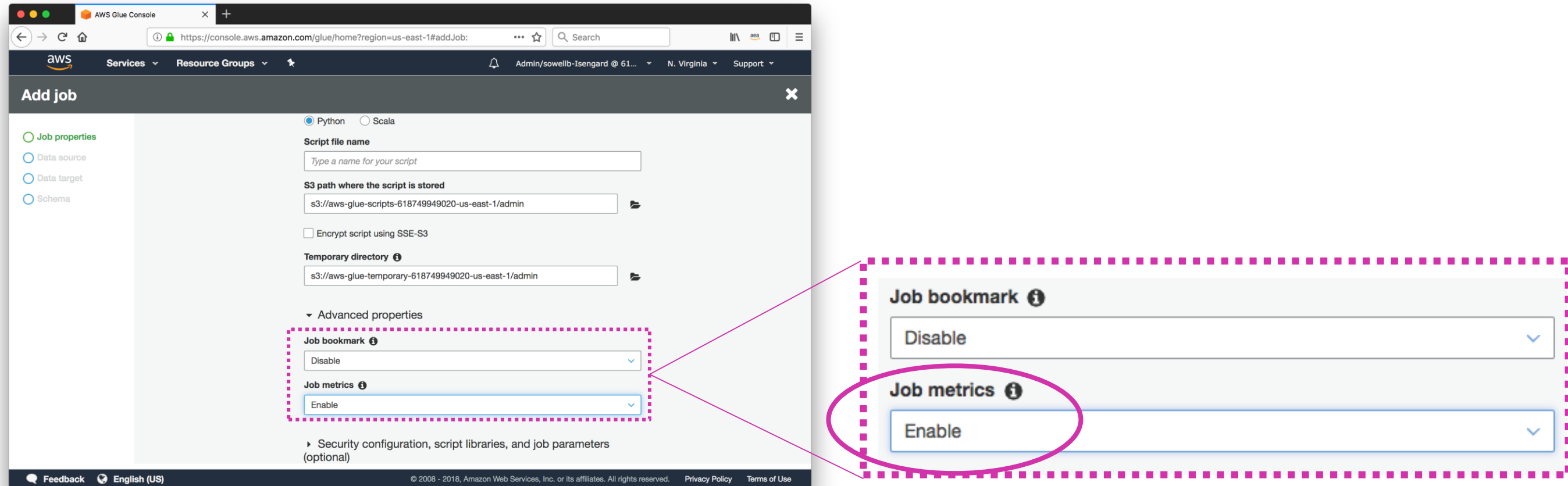
Jobs are divided into *stages*

1 stage x 1 partition = 1 *task*

Driver schedules tasks on *executors*.
2 executors per DPU

Overall throughput is limited by the number of partitions (shards)

AWS Glue job metrics



- Metrics can be enabled in the AWS Command Line Interface (AWS CLI) and AWS SDK by passing `--enable-metrics` as a job parameter key.

Profile jobs using ETL metrics

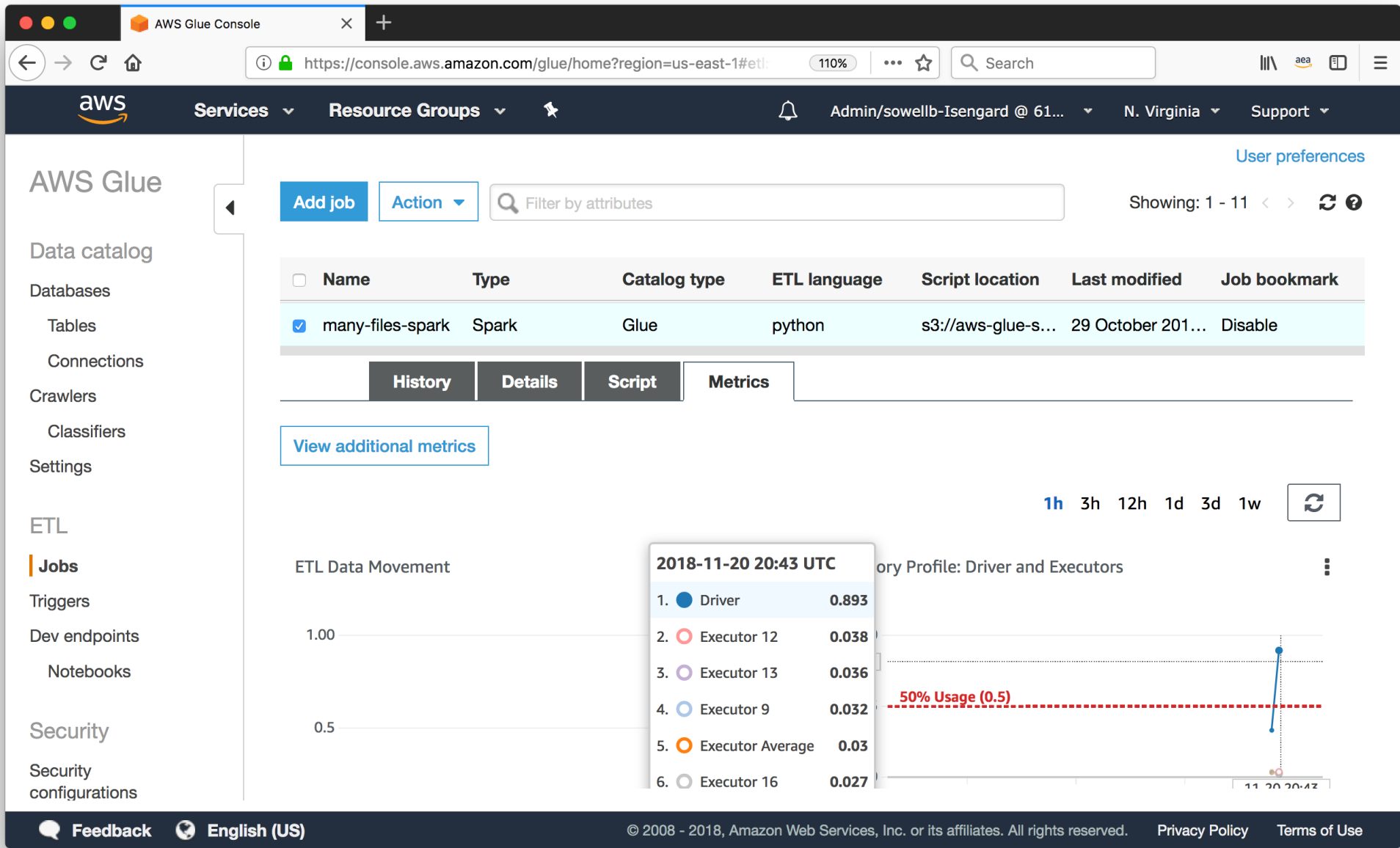
Derived from the underlying Apache Spark metrics

Driver and per executor

aggregates and instantaneous
reports to Amazon CloudWatch metrics every 30 sec

Metrics: memory usage, bytes read and written,
cpu load, bytes shuffled, needed executors,
and more

Example: profiling memory usage

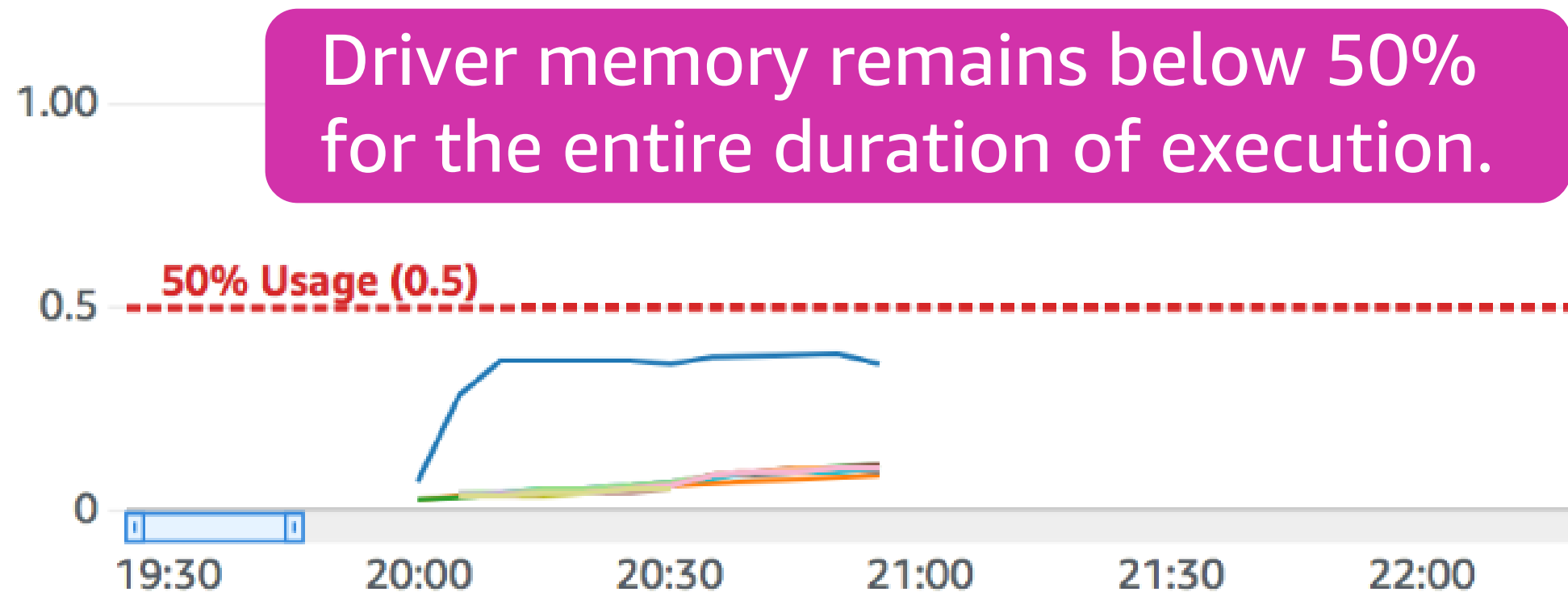


Processing lots of small files with DataFrames

Task metadata overwhelms driver

Example: AWS Glue small-file handling

Memory Profile: Driver and Executors

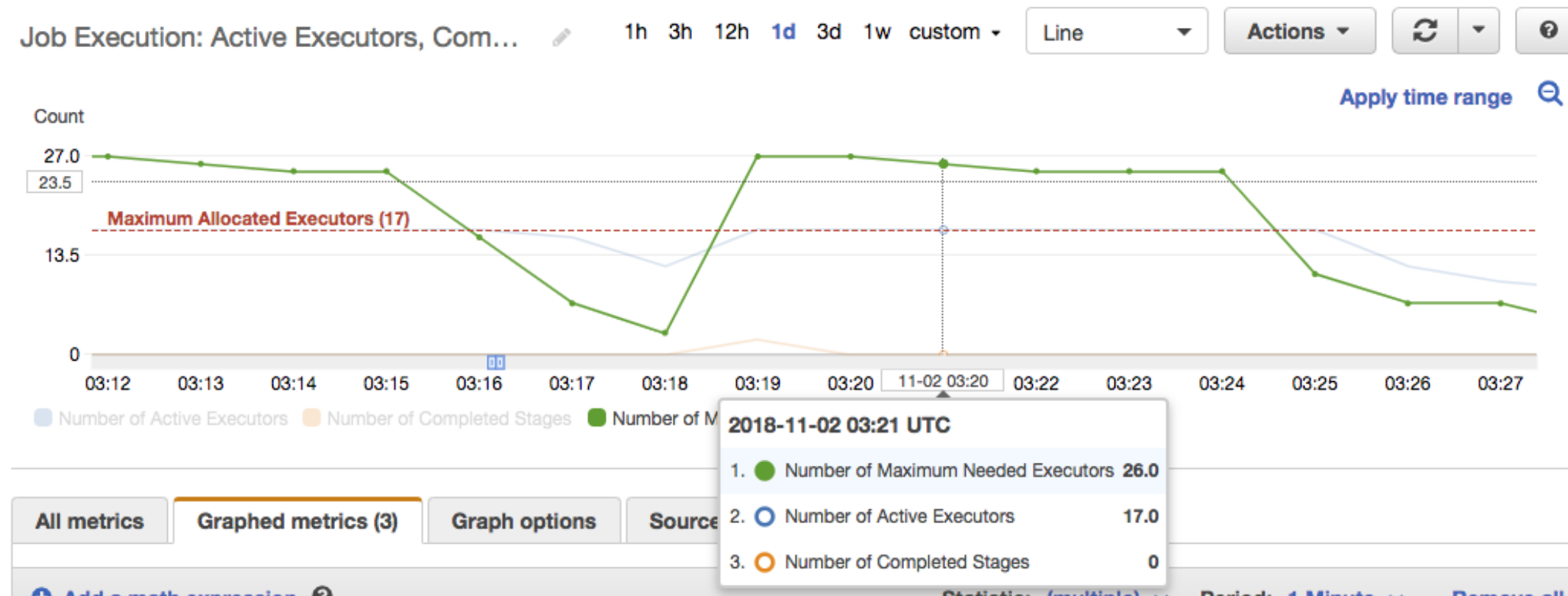


DynamicFrames automatically **group** files into fewer **tasks**

Example: optimizing parallelism

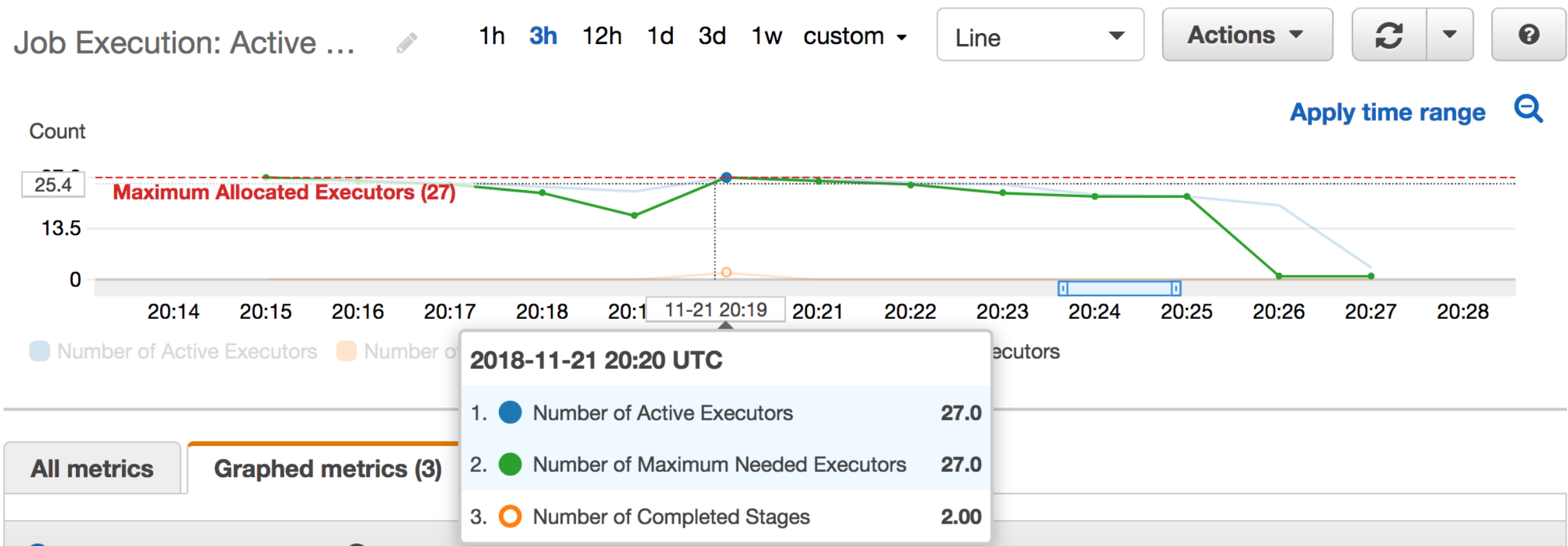
Processing large, split-able bzip2 files.

With 10 DPU, metric *maximum needed executors* shows room for scaling.



Example: optimizing parallelism

With 15 DPU, *active executors* closely tracks *maximum needed executors*.



Innovation areas

Scalability and performance

Crawlers and ETL jobs

Profiling and debugging

ETL metrics

Interoperability

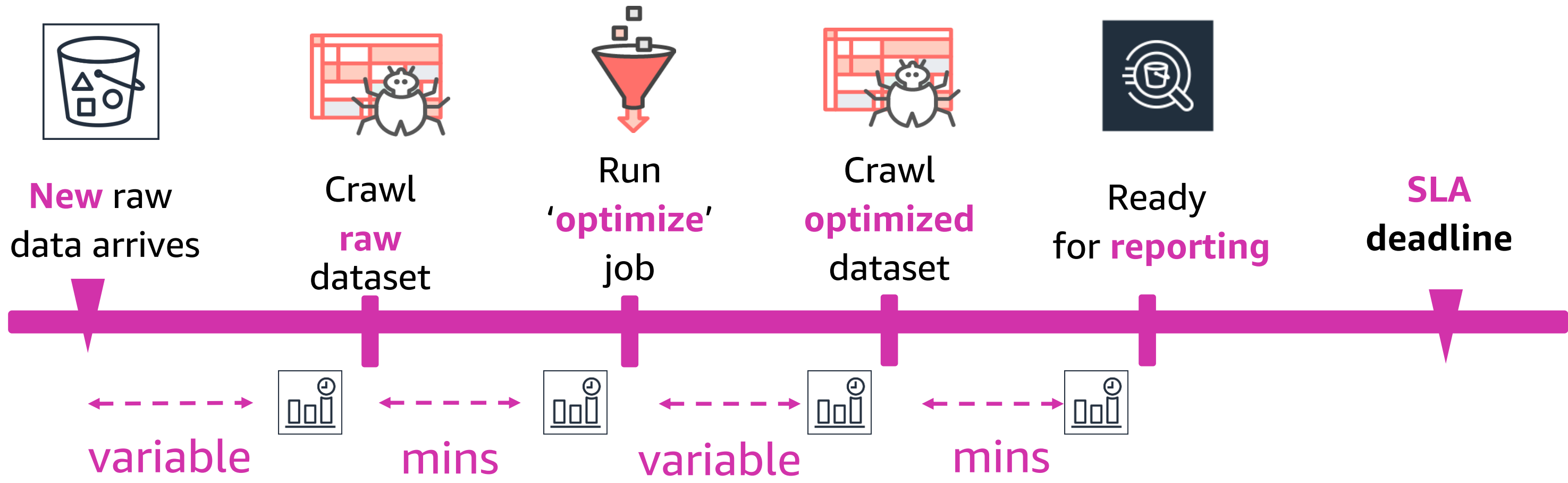
Orchestration features

Announcing new job type: Python shell

Orchestration a year ago ...

Goal: compose jobs in DAG through event-based dependencies

In-practice: time-based workflows



Orchestration building blocks

Entities

Crawlers

Jobs

Triggers

Dependencies

Schedule

Events

External

Control

Conditions

Retries

Timeout

New features

External

Publish crawler and job notifications into Amazon CloudWatch
Amazon CloudWatch events to control downstream workflows

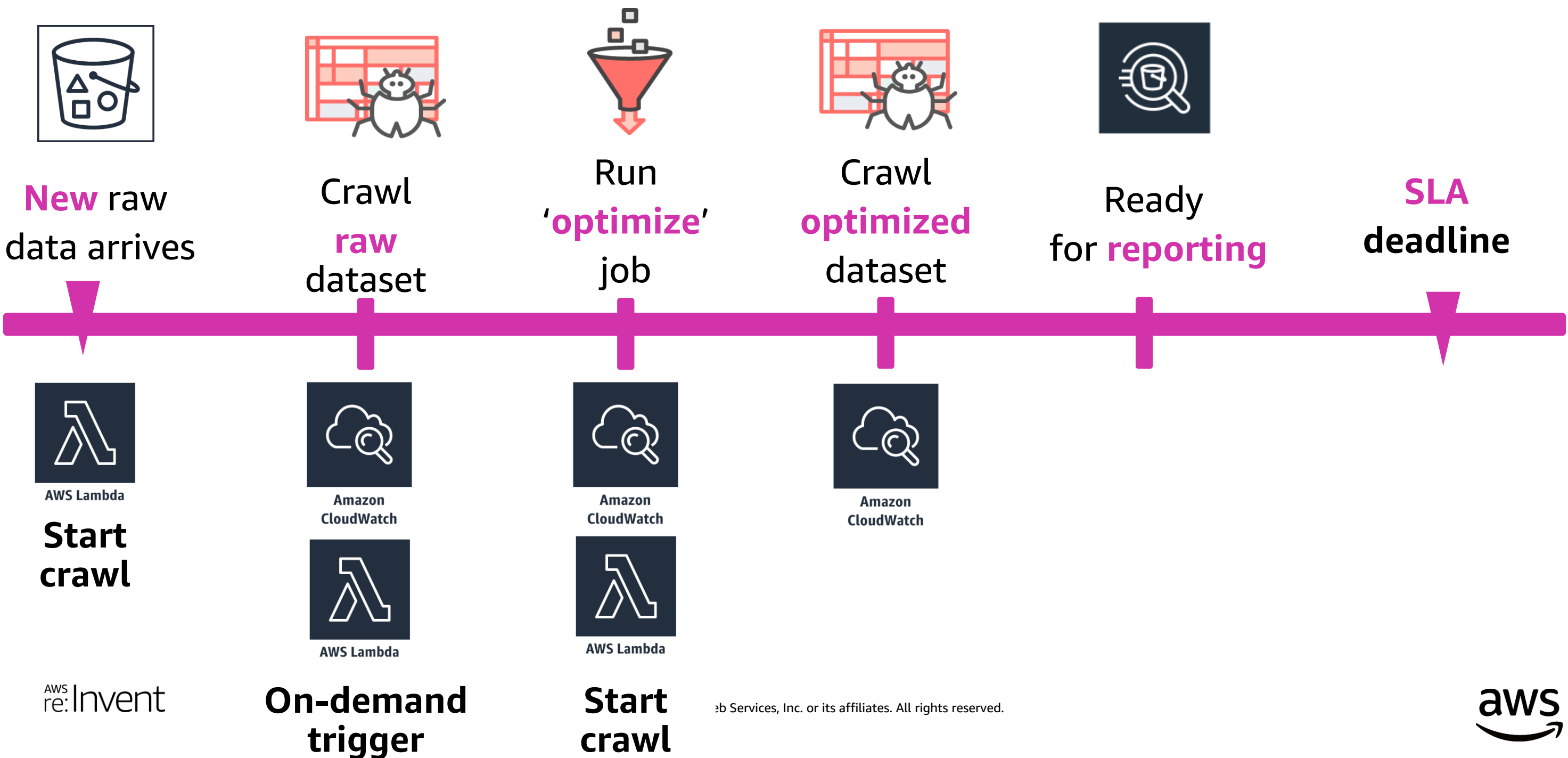
Conditions

'ANY' and 'ALL' operators in Trigger conditions
Additional job states 'failed', 'stopped', or 'timeout'

Control

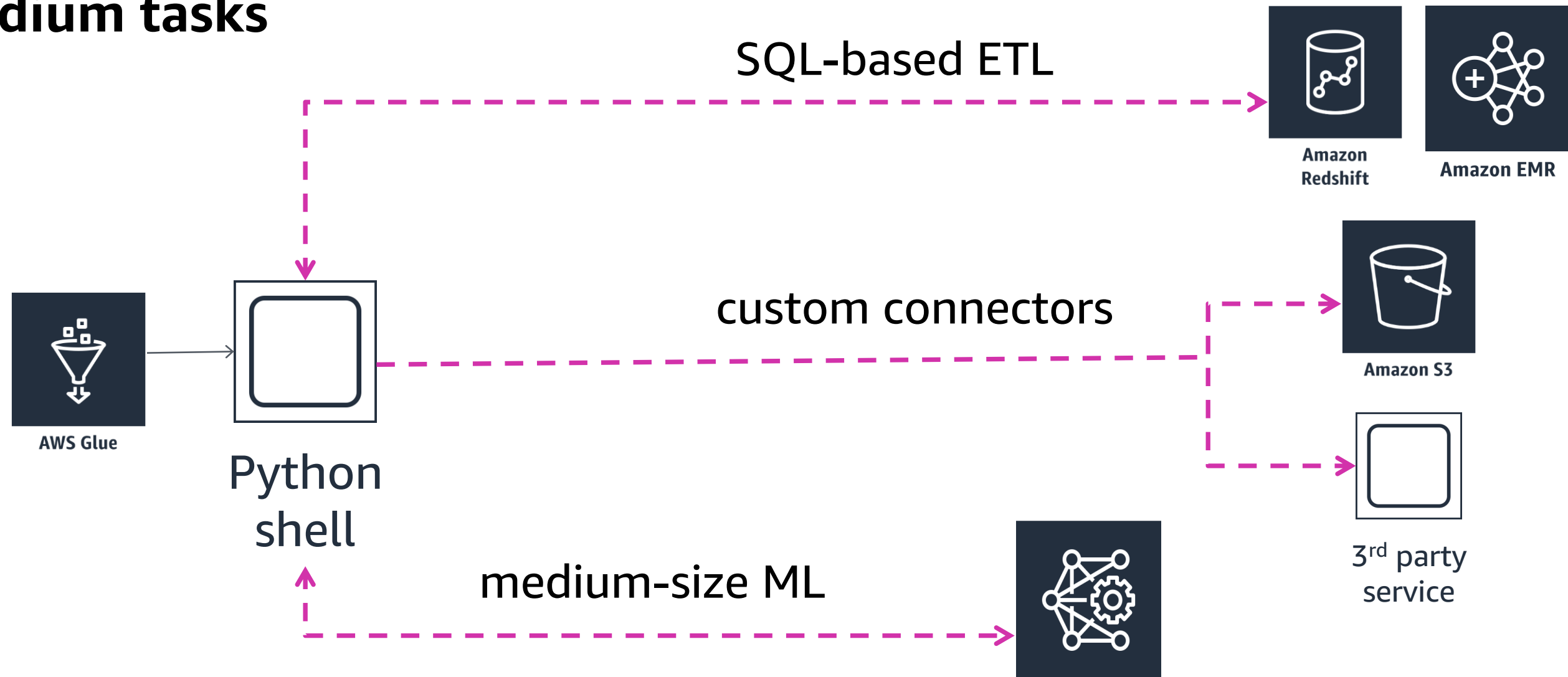
Configure job timeout
Job delay notifications

Example event-driven workflow



Announcing a new job type: Python shell

A new *cost-effective* ETL primitive for small to medium tasks



AWS Glue Python shell specs

Python 2.7 environment with
boto3, awscli, numpy, scipy, pandas, scikit-learn, PyGreSQL, ...

cold spin-up: < 20 sec, support for VPCs, no runtime limit

sizes: 1 DPU (includes 16GB), and 1/16 DPU (includes 1GB)

pricing: \$0.44 per DPU-hour, 1-min minimum, per-second billing

Coming soon (December 2018)

Python shell collaborative filtering example

Amazon customer reviews dataset (<s3://amazon-reviews-pds>)

Video category

Compute low-rank approx of (Customer x Product) ratings using SVD

uses **scipy sparse matrix** and **SVD library**

Step	Time (sec)
Redshift COPY	13
Extract ratings	5
Generate matrix	1552
SVD (k=1000)	2575
Total	4145

1 DPU

matrix: 217K x 384K

SVD -- rank = 1000

runtime: **69 min**

estimated cost: **\$0.60**

And so much more ...

Support for **Amazon DynamoDB** tables in crawlers and AWS Glue ETL
→ Run **SQL** over **DynamoDB** tables

```
dyn_frame = create_dynamic_frame.from_options(  
    connection_type = "dynamodb",  
    connection_options = {"dynamodb.input.tableName" : "DDB_TABLE"})  
  
dyn_frame.toDF().where("latency > 12")
```

Encryption: **platform-level**, **in-transit**, and **at rest** (with AWS KMS keys)

Compliance: **GDPR**, **HIPAA**, **BAA**

Related breakouts

Tuesday, November 27

ANT326 METRICS-DRIVEN PERFORMANCE TUNING FOR AWS GLUE ETL JOBS

8:30AM – 9:30AM | Venetian, Level 2, Veronese 2406

Tuesday, November 27

ANT309 BUILD AND GOVERN YOUR DATA LAKES WITH AWS GLUE

9:15AM – 10:15AM | Mirage, Mirage Events Center B

Tuesday, November 27

ANT313 SERVERLESS DATA PREP WITH AWS GLUE

7:00PM – 9:15PM | Venetian, Level 2, Venetian H

Wednesday, November 28

ANT342 WORKING WITH RELATIONAL DATABASES IN AWS GLUE ETL

2:30PM – 3:30PM | Mirage, Grand Ballroom B, Table 3

Realtor.com – Customer success

Arup Ray
Vice President, Engineering

AWS GLUE @ REALTOR.COM®



Arup Ray
Vice President, Engineering

November 26, 2018

realtor.com®

OUR MISSION

To empower
people by making
all things **home**
simple, efficient
and **enjoyable**



REALTOR.COM® MAKES MEANINGFUL CONNECTIONS

Engage

- 1.5 times more page views¹
- 1.3 times longer visits¹

Attract

- 9 of 10 consumer leads are actively searching or ready to transact²
- #1 for finding an agent³

Connect

- 63 million unique visitors per month⁴
- 2 billion+ page views per month⁴



¹ comScore Sept. 2018

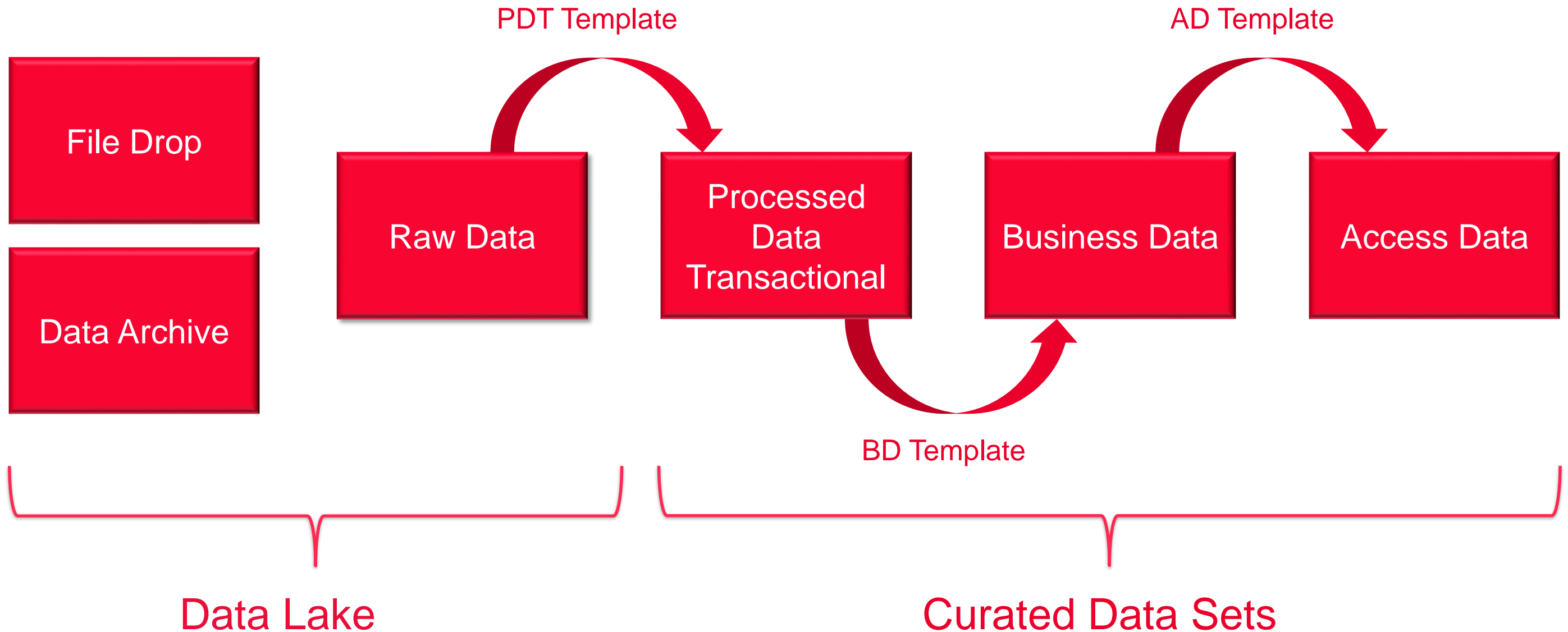
² Lead Quality Study, July 2017

³ 2017 NAR Profile of Home Buyers and Sellers

⁴ FY Q418 Internal metrics

DATA ANALYTICS PLATFORM

Template based data transformation



TEMPLATE DRIVEN TRANSFORMATION

PROBLEM

- Generalize the ETL process to handle generic data sets and volumes
- Solve for patterns – like deeply nested JSON attributes, array within JSON attributes
- Make the data easy to query

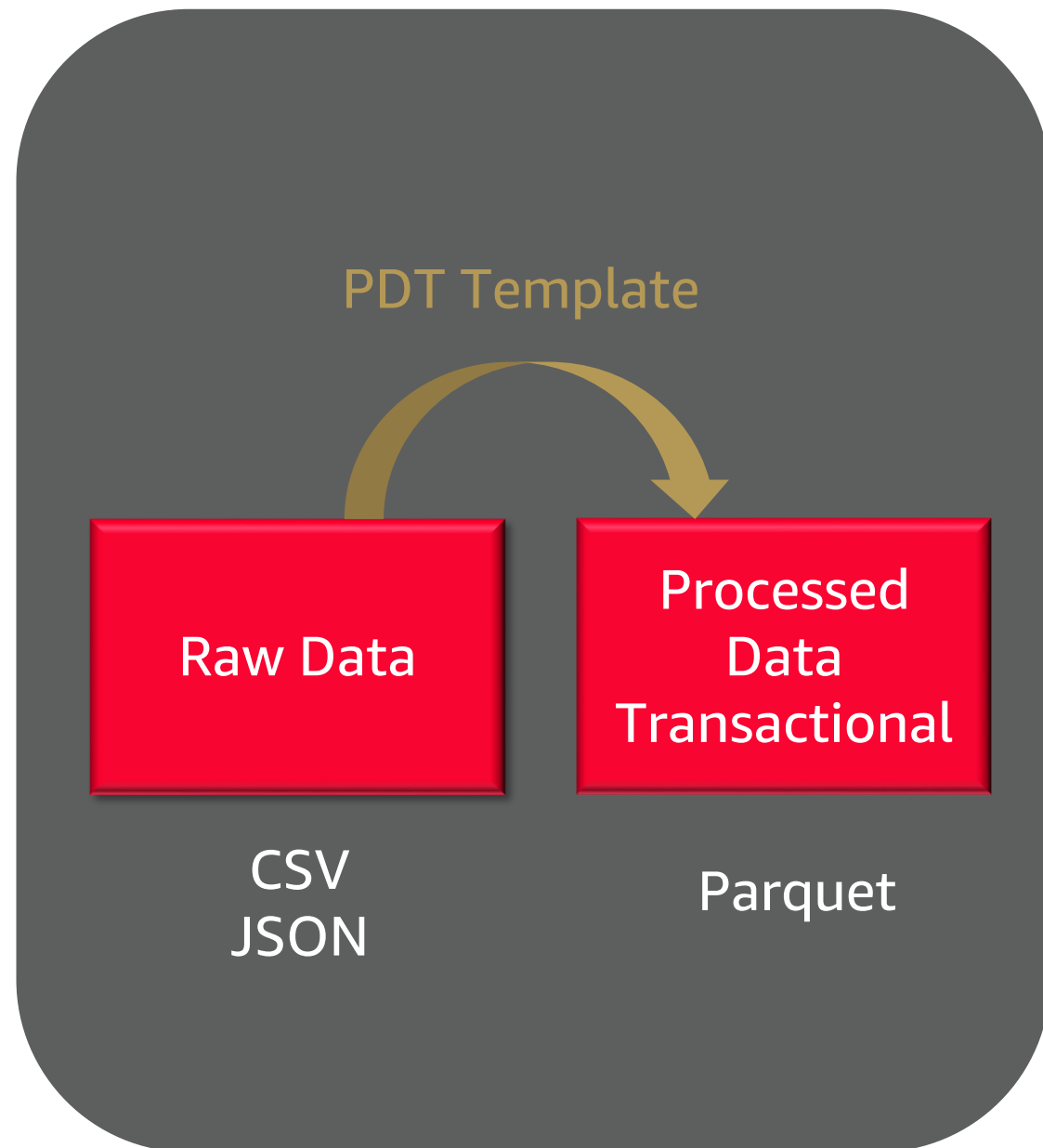
SOLUTION

CHALLENGES

- EMR Hive requires managing the cluster
- UDF based functionality requires a lot of code to maintain

PDT TEMPLATE SOLUTION

Between RD and PDT layers in Data Analytics Platform

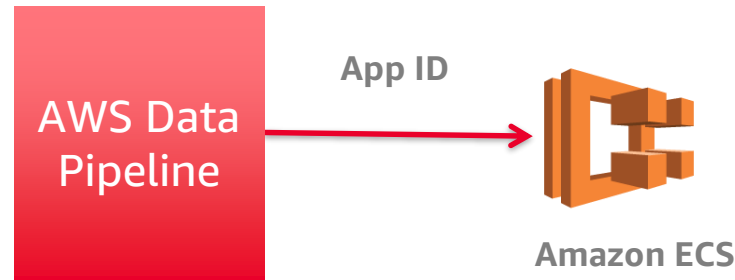


- Template code implemented in Python.
- User defined configuration as JSON
- Supported Raw Data formats – CSV with header, JSON
- These are the template configuration parameters
 - Data location (RD S3 location PDT S3 location)
 - Data structure of input – implicit or header location
 - Input column names that require data type validation
 - Relationalize
 - Input multi-valued columns that requires to be *relationalized*
 - Deeply nested columns that require flattening
 - Duplicate detection of input rows (optional)

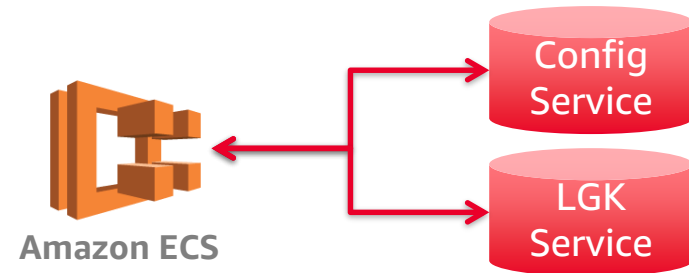
PDT TEMPLATE

How AWS Glue performs batch data processing

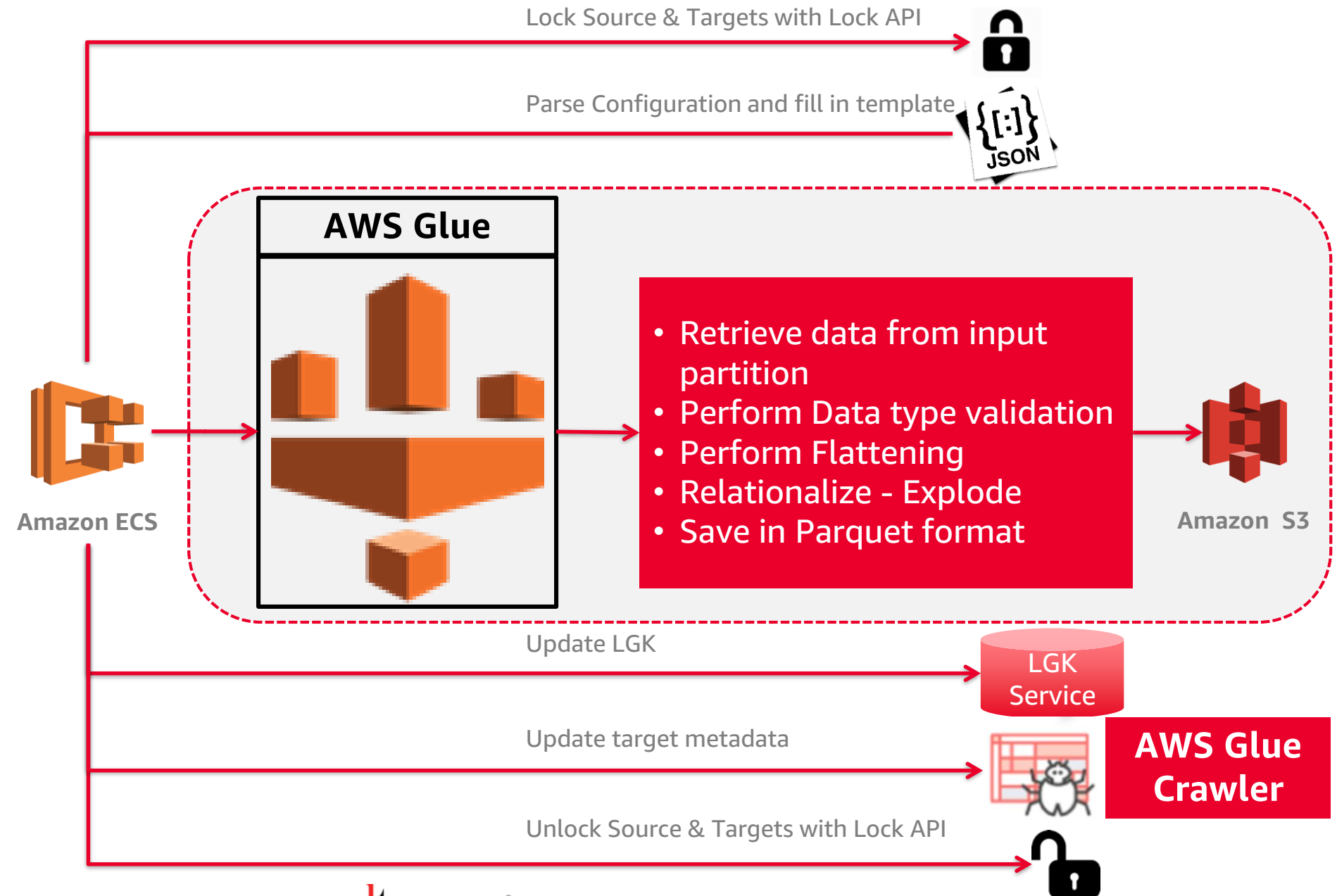
Step 1



Step 2



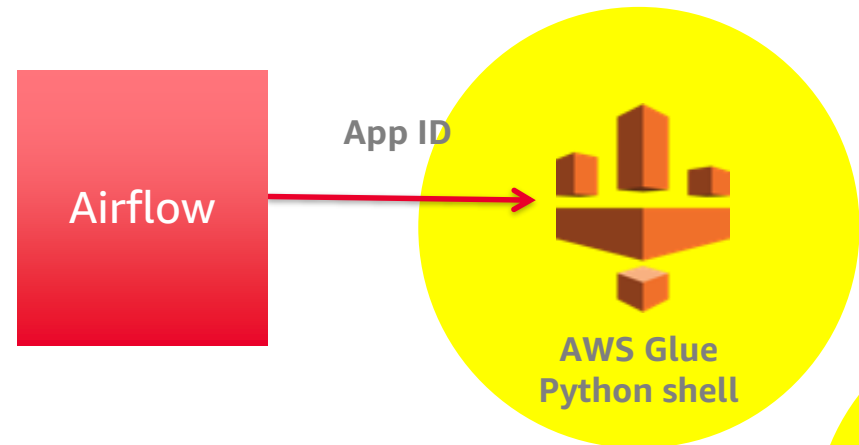
Step 3



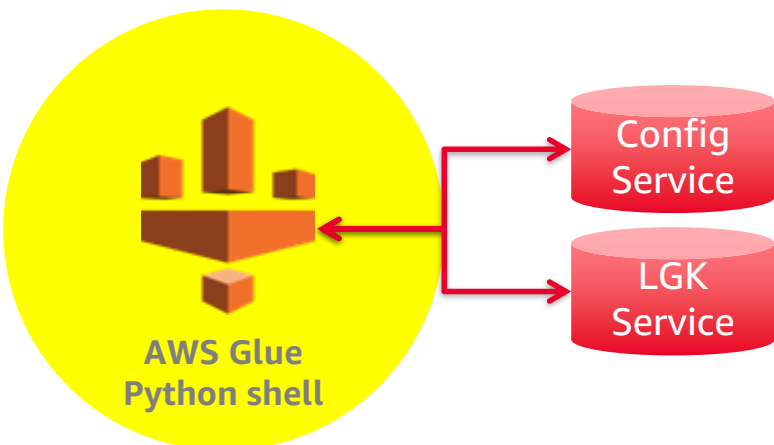
PDT TEMPLATE

How AWS Glue performs batch data processing

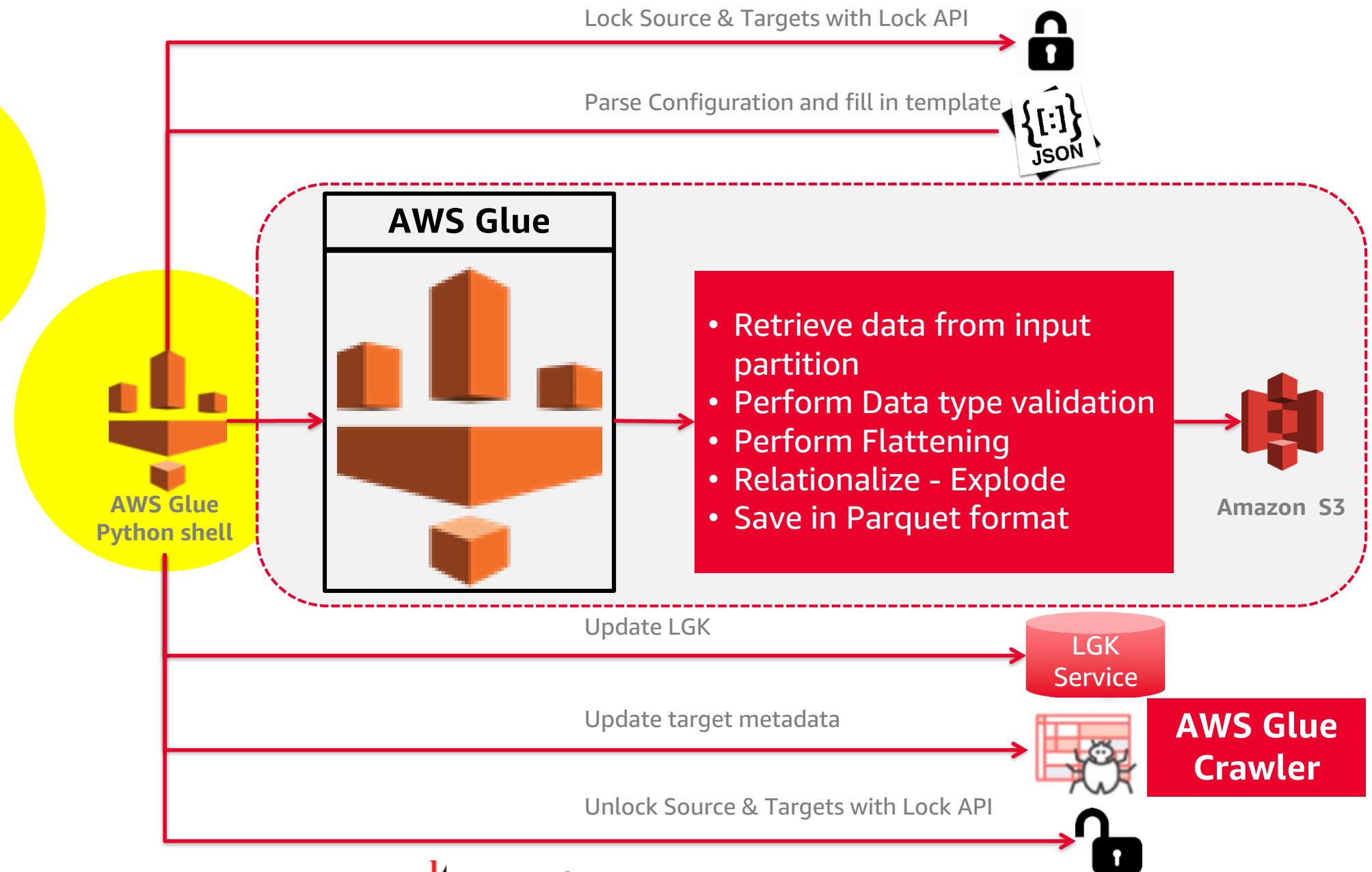
Step 1



Step 2

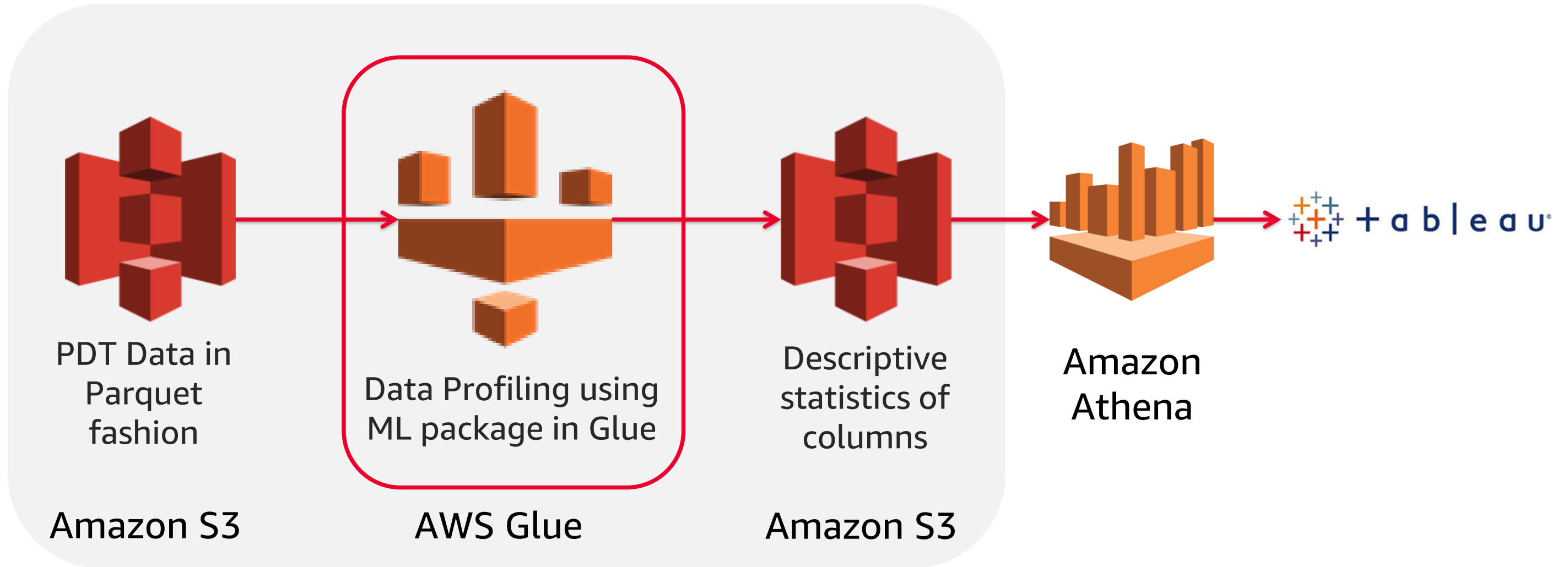


Step 3



DATA PROFILING

Using Data Validation Framework



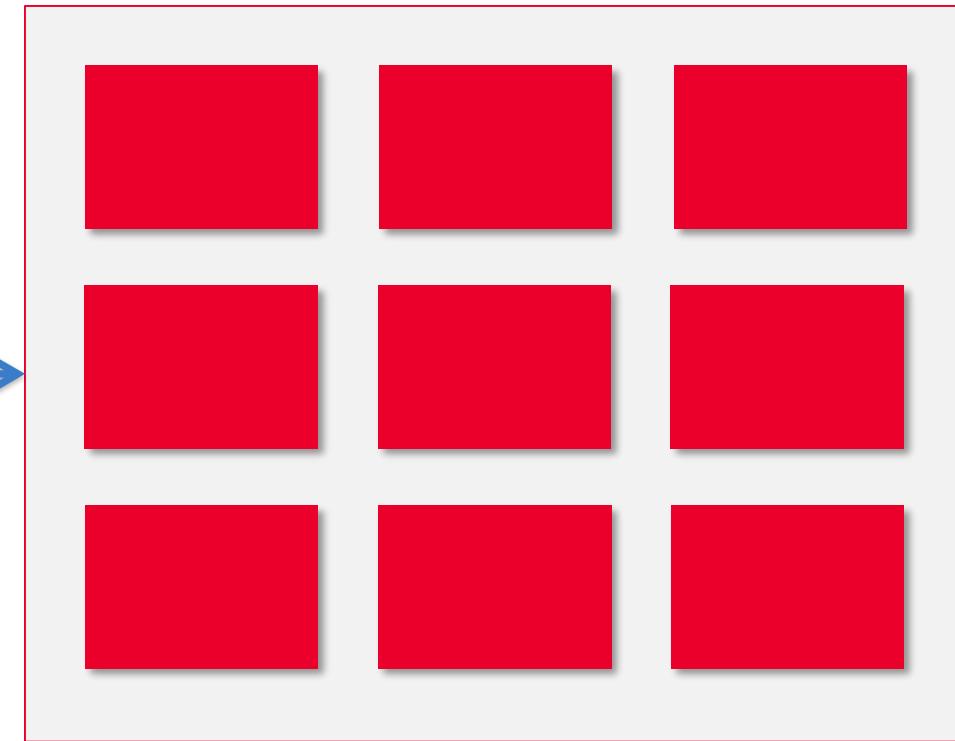
AWS GLUE PYTHON SHELL

The way we see it



AWS Glue
Python shell

Serverless Edge Node for
triggering, light
transformations,
uncompress, tar extract,
Parquet conversion



AWS Glue

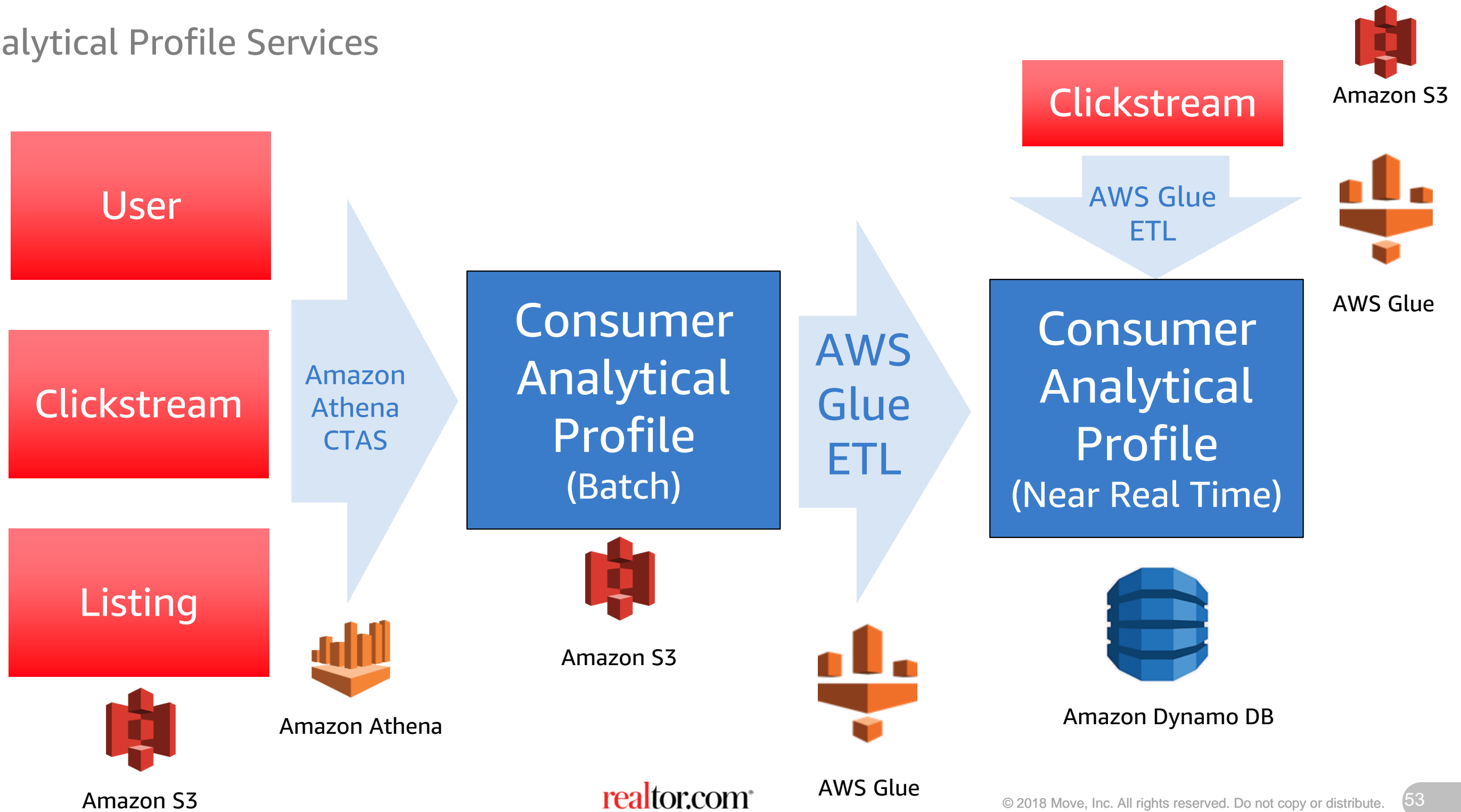
Serverless Distributed
Compute Cluster



We analyze consumer behavior
and expectation to deliver a
best-in-class experience for both
REALTORS[®] and consumers

BEYOND PDT

Analytical Profile Services



BENEFITS OF USING AWS GLUE



Speed of implementation – developer productivity with in-built transforms



Less code to maintain - <10 lines versus 200+ for explosion of array attributes



Operations team loves the server less aspect



Performance for Dynamo DB write using AWS Glue significantly better - about 10X

Thank you!

Mehul A. Shah
glue-pm@amazon.com

Arup Ray
<https://www.linkedin.com/in/arupray>



Please complete the session
survey in the mobile app.

THANK YOU



realtor.com®