Data Cataloguing & ETL with AWS GLUE

Adebimpe(Bims) Daniells

March 25, 2020

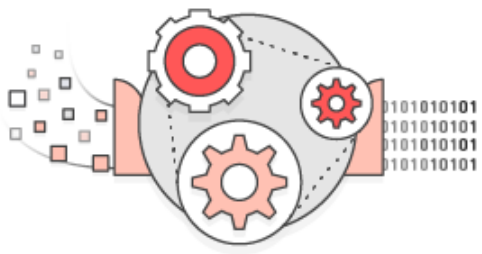
Building data lakes can take months

Data cataloguing

Discover and organize your data sets

ETL is the most time-consuming part of analytics

ETL



70% of time spent here

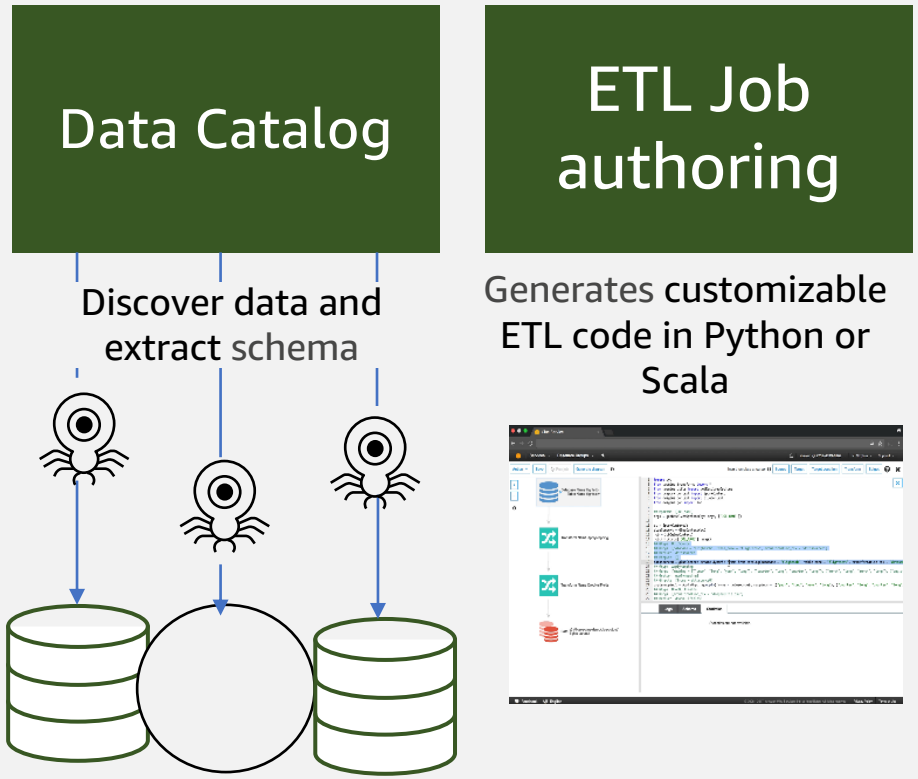
Data Warehousing

Amazon Redshift

Business Intelligence

Amazon QuickSight

AWS Glue: Data Catalog & ETL Service



Automatically discovers data and stores schema

Data is immediately searchable, and available to extract, transform, and load (ETL)

Automatically generates customizable ETL code

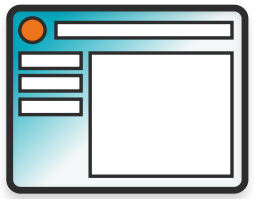
Schedules and runs your ETL jobs

Serverless

AWS Glue: features

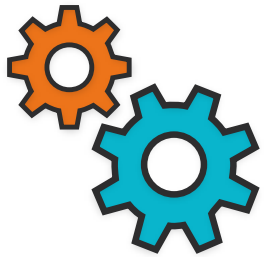
Data Catalog

- Hive metastore compatible metadata repository of data sources.
- Crawls data source to infer table, data type, partition format.



Job Authoring

- Generates Python code to move data from source to destination.
- Edit with your favorite IDE; share code snippets using Git.



Job Execution

- Runs jobs in Spark containers – automatic scaling based on SLA.
- Glue is serverless - only pay for the resources you consume.

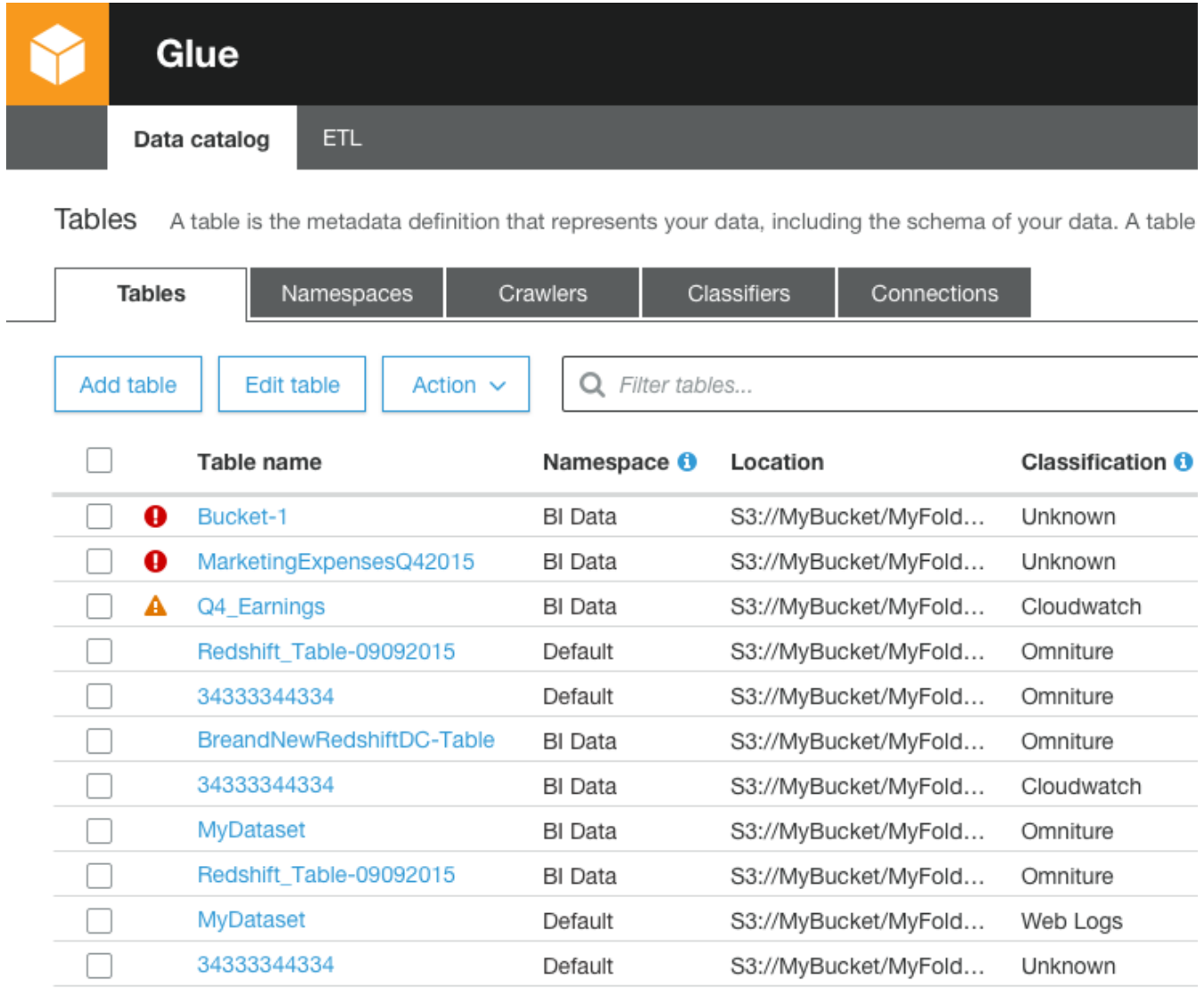
Glue data catalog

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

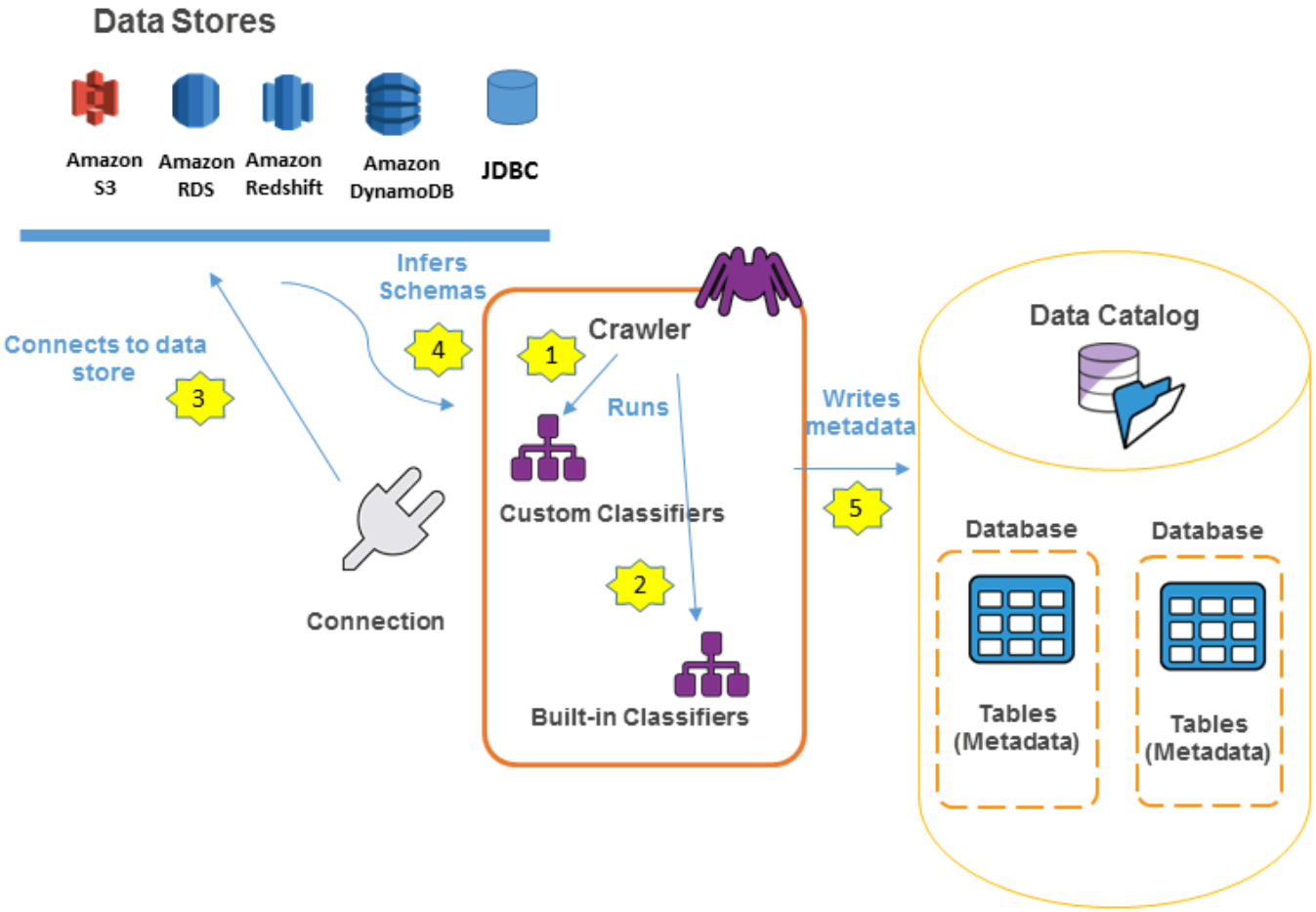
We added a few extensions:

- **Search** 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**.



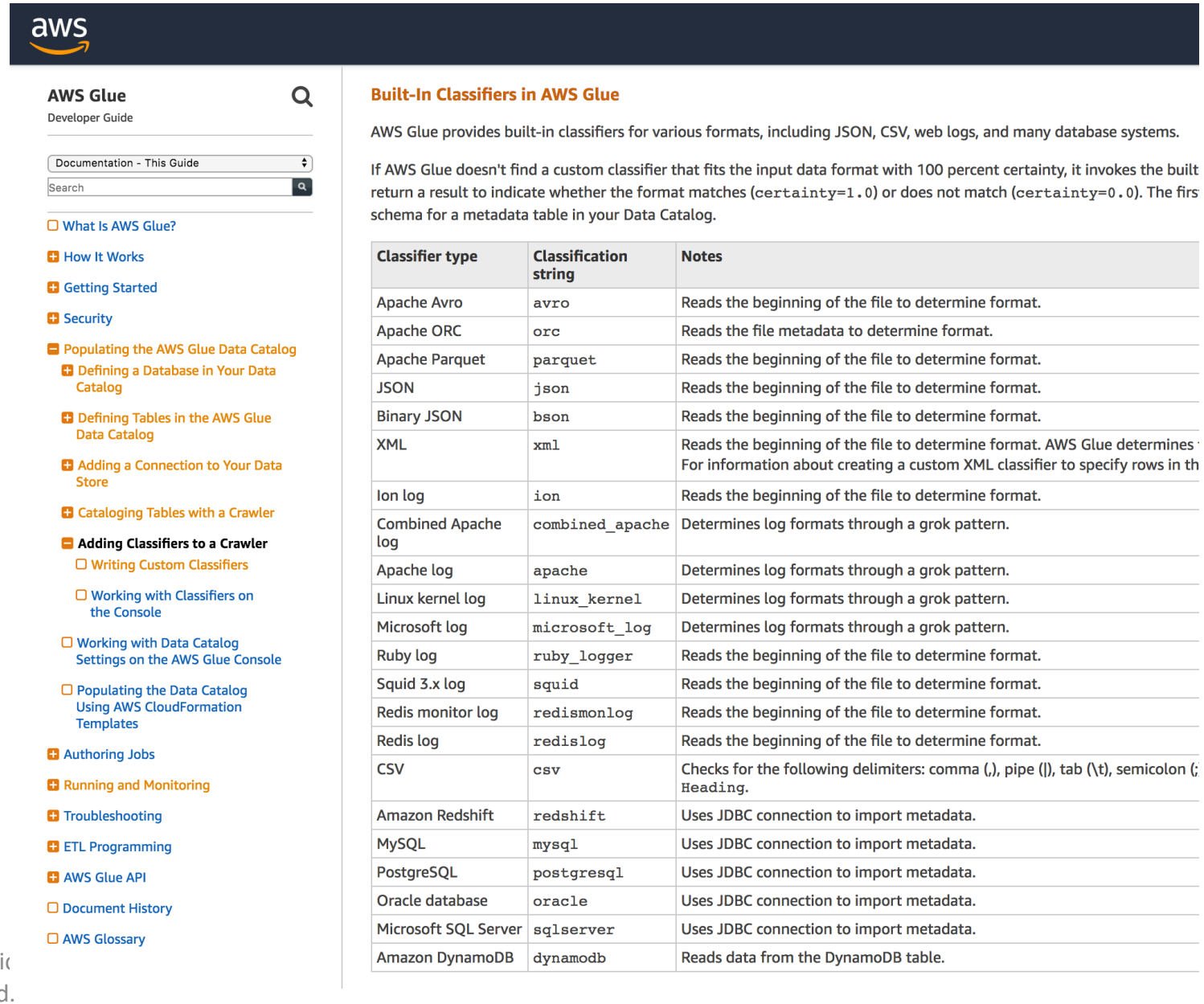
Glue Workflow Overview



1. Use a crawler to infer the schema of your data.
2. Use Classifiers to Populate the AWS Glue Data Catalog with table definitions.
3. Crawler connects to the data store.
4. Schema inference occurs.
5. Define a job that describes the transformation of data from source to target.
6. Run your job to transform your data.
7. Monitor

- A unified metadata repository across data sources & data formats
- Integrated with other AWS services

Bulit-in



AWS Glue
Developer Guide

Documentation - This Guide

Search

- What Is AWS Glue?
- How It Works
- Getting Started
- Security
- Populating the AWS Glue Data Catalog
  - Defining a Database in Your Data Catalog
  - Defining Tables in the AWS Glue Data Catalog
  - Adding a Connection to Your Data Store
  - Cataloging Tables with a Crawler
  - **Adding Classifiers to a Crawler**
    - Writing Custom Classifiers
    - Working with Classifiers on the Console
  - Working with Data Catalog Settings on the AWS Glue Console
  - Populating the Data Catalog Using AWS CloudFormation Templates
- Authoring Jobs
- Running and Monitoring
- Troubleshooting
- ETL Programming
- AWS Glue API
- Document History
- AWS Glossary

Built-In Classifiers in AWS Glue

AWS Glue provides built-in classifiers for various formats, including JSON, CSV, web logs, and many database systems.

If AWS Glue doesn't find a custom classifier that fits the input data format with 100 percent certainty, it invokes the built return a result to indicate whether the format matches (`certainty=1.0`) or does not match (`certainty=0.0`). The firs schema for a metadata table in your Data Catalog.

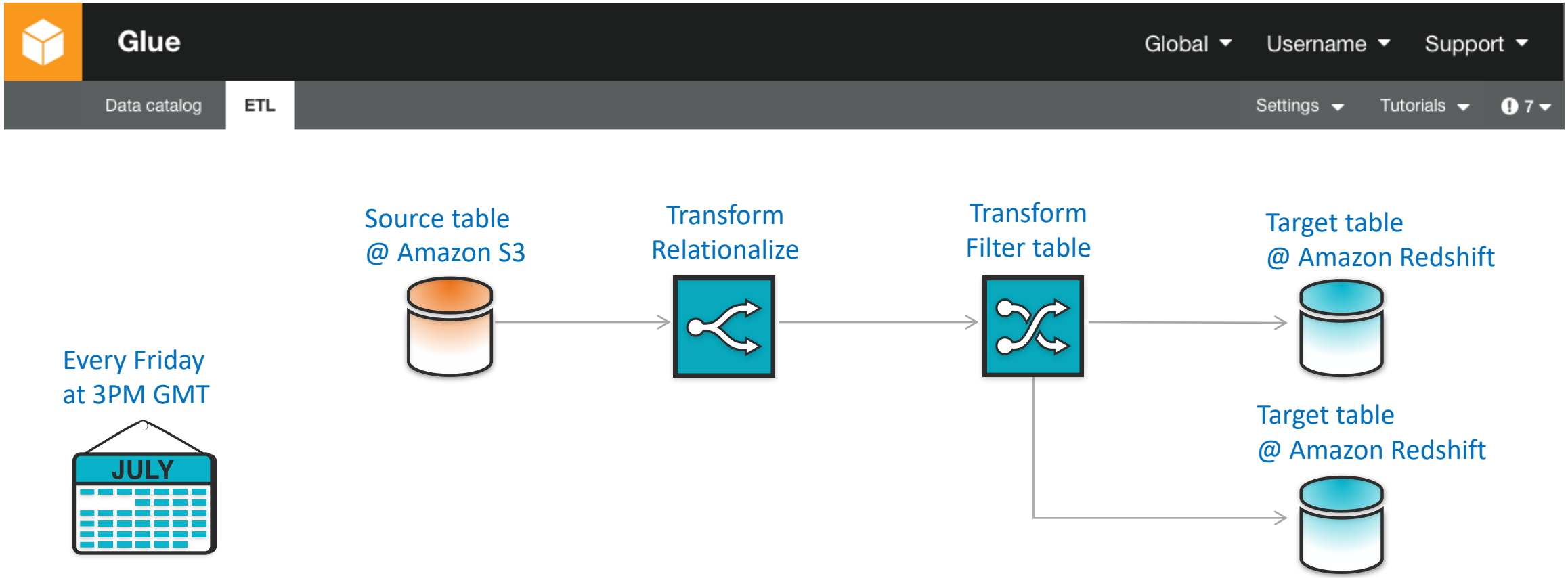
Classifier type	Classification string	Notes
Apache Avro	`avro`	Reads the beginning of the file to determine format.
Apache ORC	`orc`	Reads the file metadata to determine format.
Apache Parquet	`parquet`	Reads the beginning of the file to determine format.
JSON	`json`	Reads the beginning of the file to determine format.
Binary JSON	`bson`	Reads the beginning of the file to determine format.
XML	`xml`	Reads the beginning of the file to determine format. AWS Glue determines For information about creating a custom XML classifier to specify rows in th
Ion log	`ion`	Reads the beginning of the file to determine format.
Combined Apache log	`combined_apache`	Determines log formats through a grok pattern.
Apache log	`apache`	Determines log formats through a grok pattern.
Linux kernel log	`linux_kernel`	Determines log formats through a grok pattern.
Microsoft log	`microsoft_log`	Determines log formats through a grok pattern.
Ruby log	`ruby_logger`	Reads the beginning of the file to determine format.
Squid 3.x log	`squid`	Reads the beginning of the file to determine format.
Redis monitor log	`redismonlog`	Reads the beginning of the file to determine format.
Redis log	`redislog`	Reads the beginning of the file to determine format.
CSV	`csv`	Checks for the following delimiters: comma (,), pipe (|), tab (\t), semicolon (; `Heading`.
Amazon Redshift	`redshift`	Uses JDBC connection to import metadata.
MySQL	`mysql`	Uses JDBC connection to import metadata.
PostgreSQL	`postgresql`	Uses JDBC connection to import metadata.
Oracle database	`oracle`	Uses JDBC connection to import metadata.
Microsoft SQL Server	`sqlserver`	Uses JDBC connection to import metadata.
Amazon DynamoDB	`dynamodb`	Reads data from the DynamoDB table.

- Ref: AWS Glue Developer Guide

Lab Break: Time to do Lab 2.1

GLUE Transformation

Automatic code generation



1. Pick sources and targets from the data catalog
2. Glue generates transformation graph and *Python code*
3. Specify trigger condition

Glue ETL scripts are forgiving

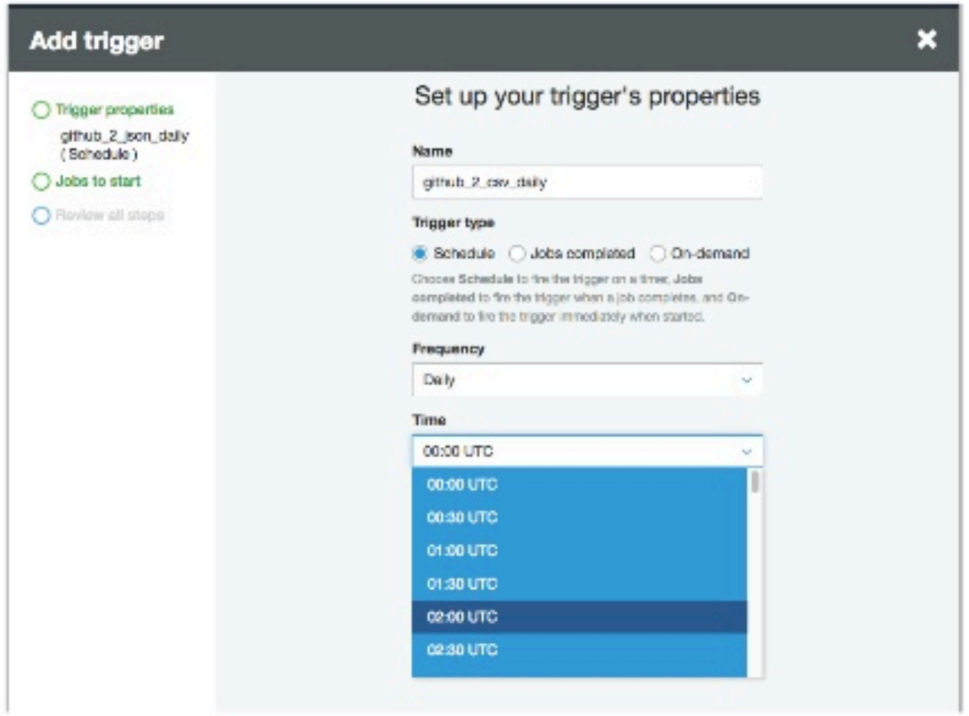
```
## @type: DataSource
## @args: [name_space = "glue", table_name = "tweets"]
## @return: datasource0
## @inputs: []
datasource = glueContext.create_dynamic_frame.from_catalog(name_space =
    "glue", table_name = "tweets")

## @type: Relationize
## @args: [name = "tweets"]
## @return: relations
## @inputs: [frame = datasource]
relations = Relationize.apply(frame = datasource, name = "tweets")
```

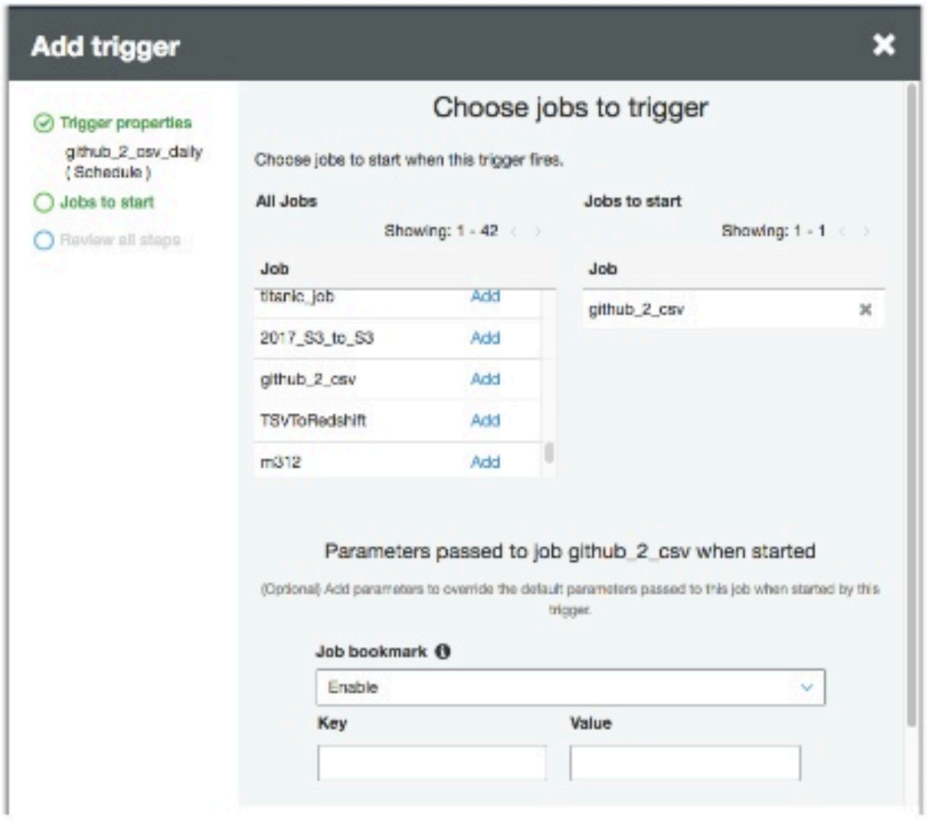
Generated code has human-readable annotations corresponding to the ETL graph

Forgiving in the face of failures:

- **Idempotency** is built-in. After a crash, a job restart picks up from where it left off, and output data is not duplicated.
- **Bad data** is flagged in-situ and does not crash the job. Users can siphon error tuples at any step to S3 buckets for subsequent error processing.

Scheduling Jobs

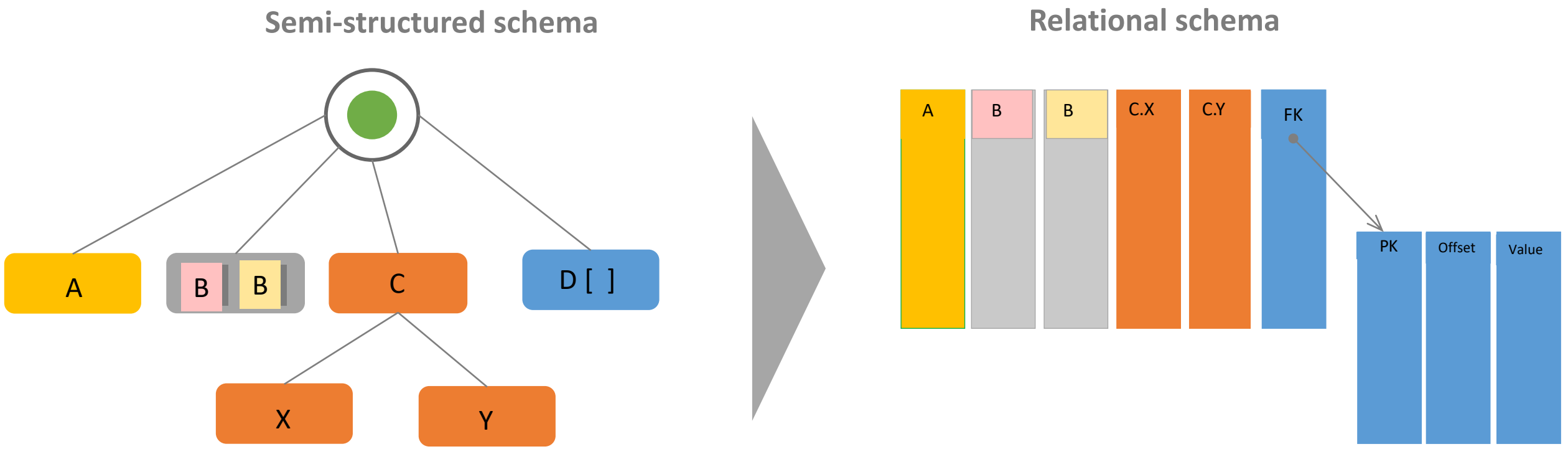




- Several event types

pass parameters

Glue transformations are flexible and adaptive

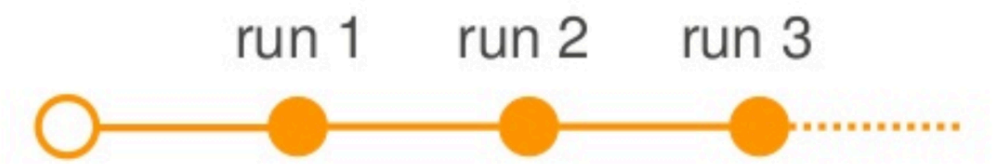


- Flatten semi-structured objects with arbitrary complexity into relational tables, on-the-fly.
- Pivot arrays and other collection types into a separate table, generating key-foreign key values.
- Modify mapping as the source schema changes, and modify the target schemas as needed.

Job bookmarks

Bookmarks are per-job checkpoints that track the work done in previous runs.

They persist the state of sources, transforms, and sinks on each run

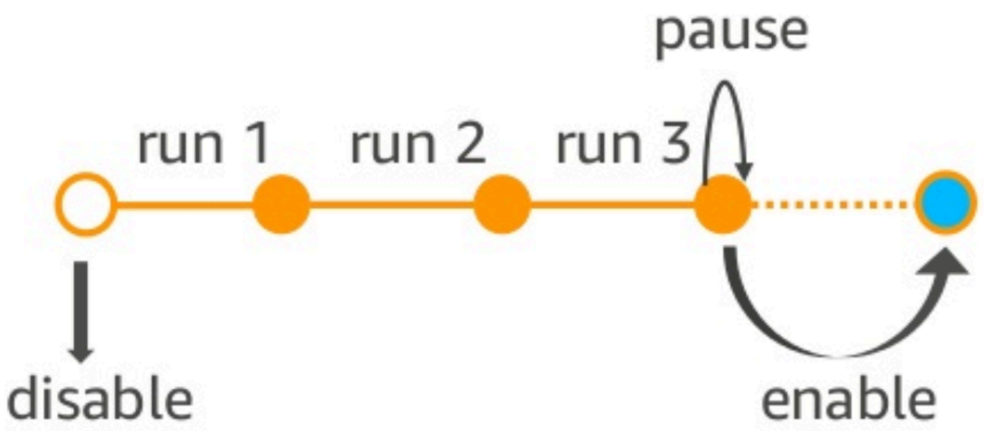


Example uses:

Process githubarchive files daily
Process Firehose files hourly
Track timestamps on primary keys in DBs
Track generated foreign keys for normalization

Job bookmark options

Option	Behavior
Enable	Pick up from where you left off
Disable	Ignore and process the entire dataset every time
Pause	Temporarily disable advancing the bookmark



Examples:

Enable: Process the newest data set partition

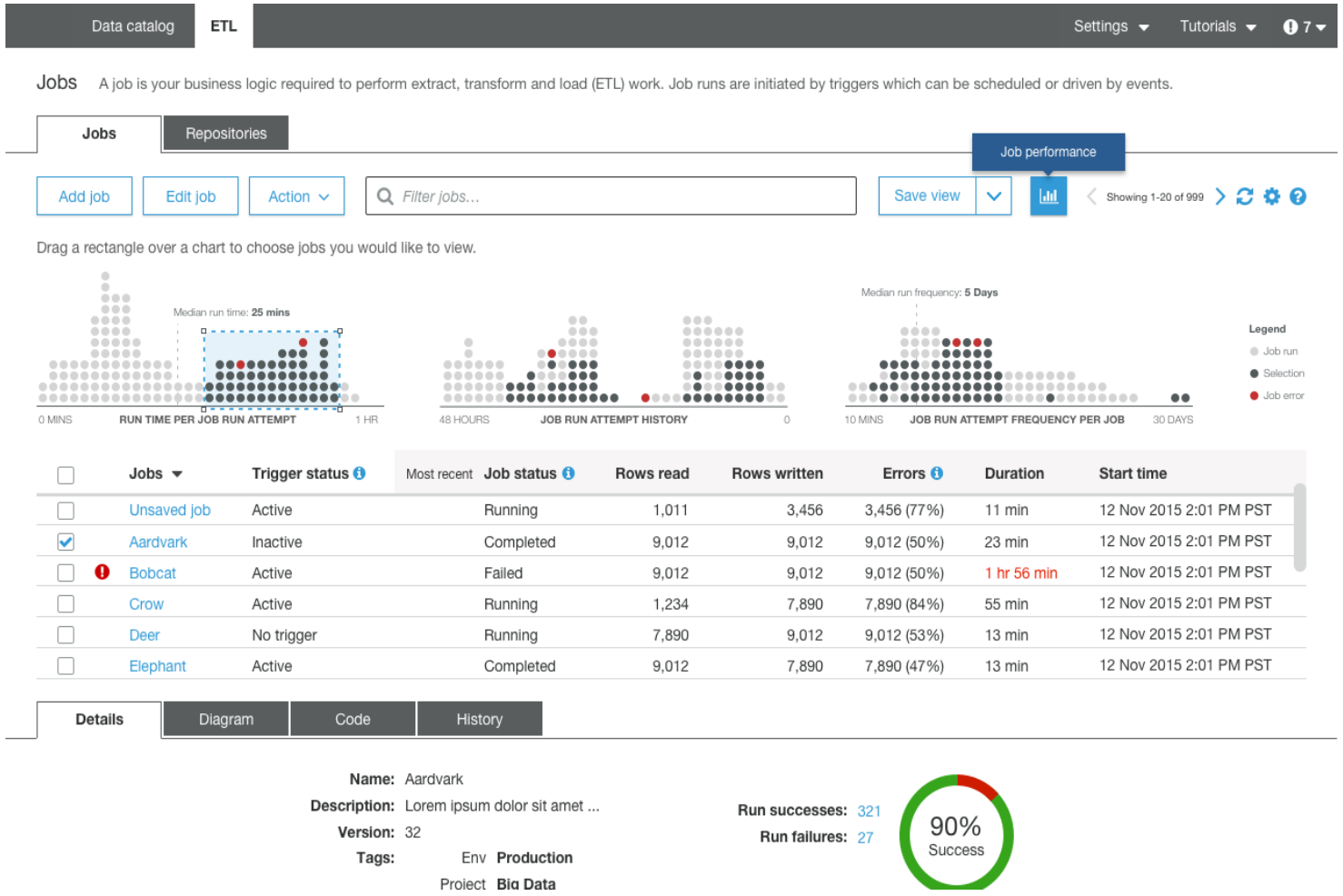
Disable: Process the entire data set table

Pause: Process the previous data set partition

https://github.com/awslabs/aws-glue-samples

Monitoring, metrics, and notification

- Track pipelines, job history
- Drill-down to transform level metrics
  - rows processed
  - input & output schema
- Notifications on: job status, errors, new data, and schema changes



Thank you

Questions/Comments

adedan@amazon.com