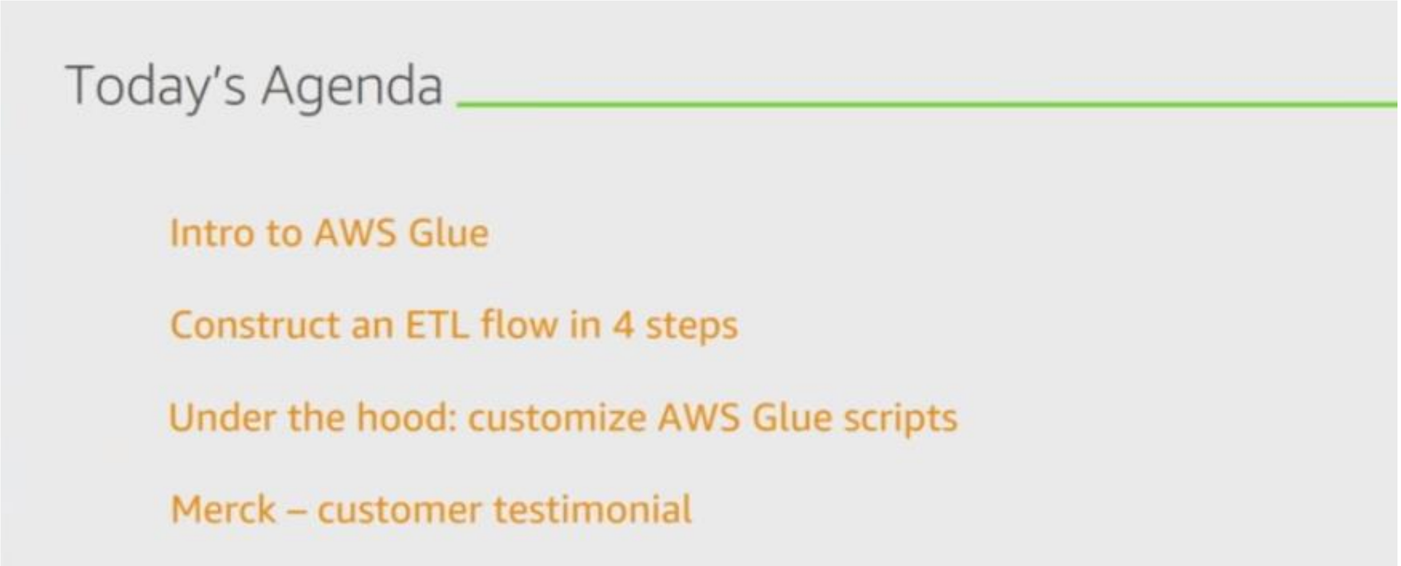
Organizations need to gain insight and knowledge from a growing number of Internet of Things (IoT), APIs, clickstreams, unstructured and log data sources. However, organizations are also often limited by legacy data warehouses and ETL processes that were designed for transactional data. In this session, we introduce key ETL features of AWS Glue, cover common use cases ranging from scheduled nightly data warehouse loads to near real-time, event-driven ETL flows for your data lake. We discuss how to build scalable, efficient, and serverless ETL pipelines using AWS Glue. Additionally, Merck will share how they built an end-to-end ETL pipeline for their application release management system, and launched it in production in less than a week using AWS Glue.



We are going to see how we can build data transformation pipelines with AWS Glue. We will see how we can construct an ETL flow in Glue from raw unfiltered data to an ETL flow running in production in 4 easy steps.

What is AWS Glue?

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

for developers, built by developers

1000s of customers and jobs

Select AWS Glue customers

NTT docomo
News Corp
AUTODESK
MediaMath
MERCK
myTomorrows
OLX Group
Expedia
ost
amazon Prime Air
21ST CENTURY FOX

AWS re:Invent

aws

There are many tools already

Amazon Redshift Partner Page for Data Integration

ATTUNITY CloudBeam
ironSource .atom
infoteria
bryte Systems
alooma
MATILLION
informatica Cloud
talend integration at any scale
Segment
Fivetran
etleap
xplenty
FlyData
TREASURE DATA
Qubole
RJMETRICS
snapLogic

AWS re:Invent

aws

Still, ETL developers hand-code

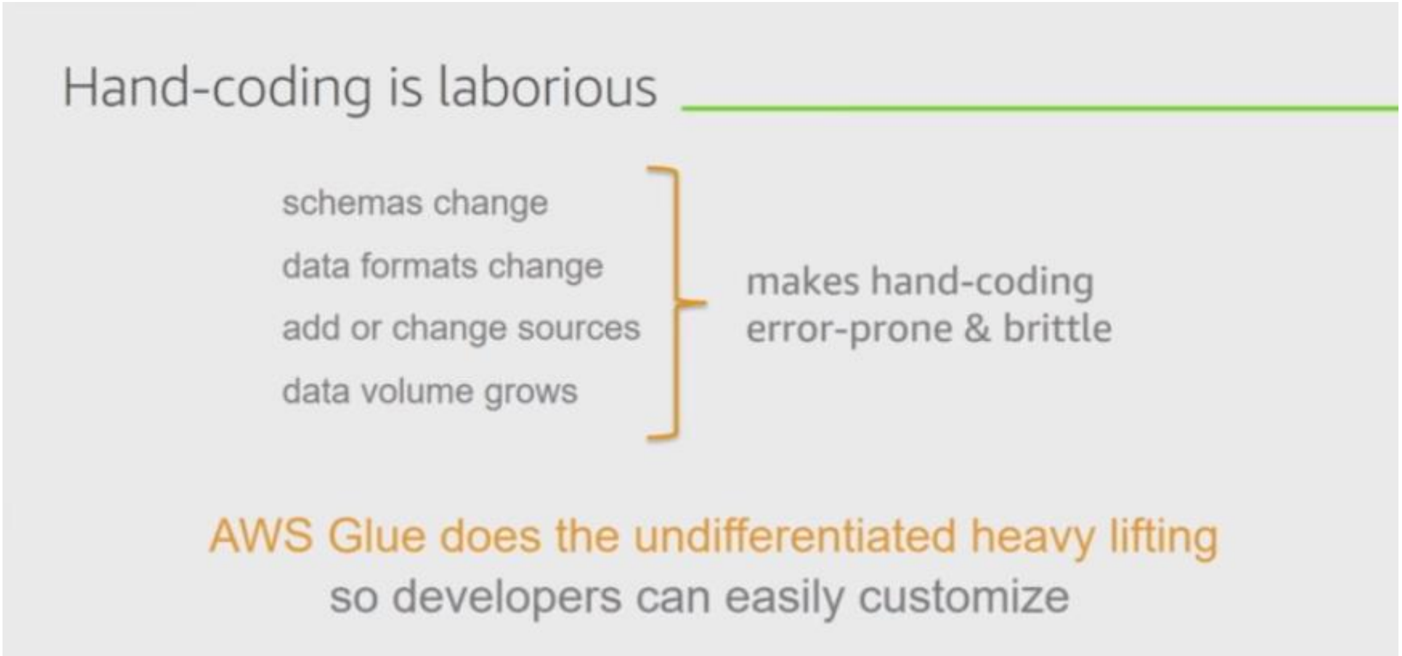
Canvas-based tools are hard to extend

Code is flexible, powerful, and easy to share

Familiar tools and development pipelines

IDEs, version control, testing, continuous integration

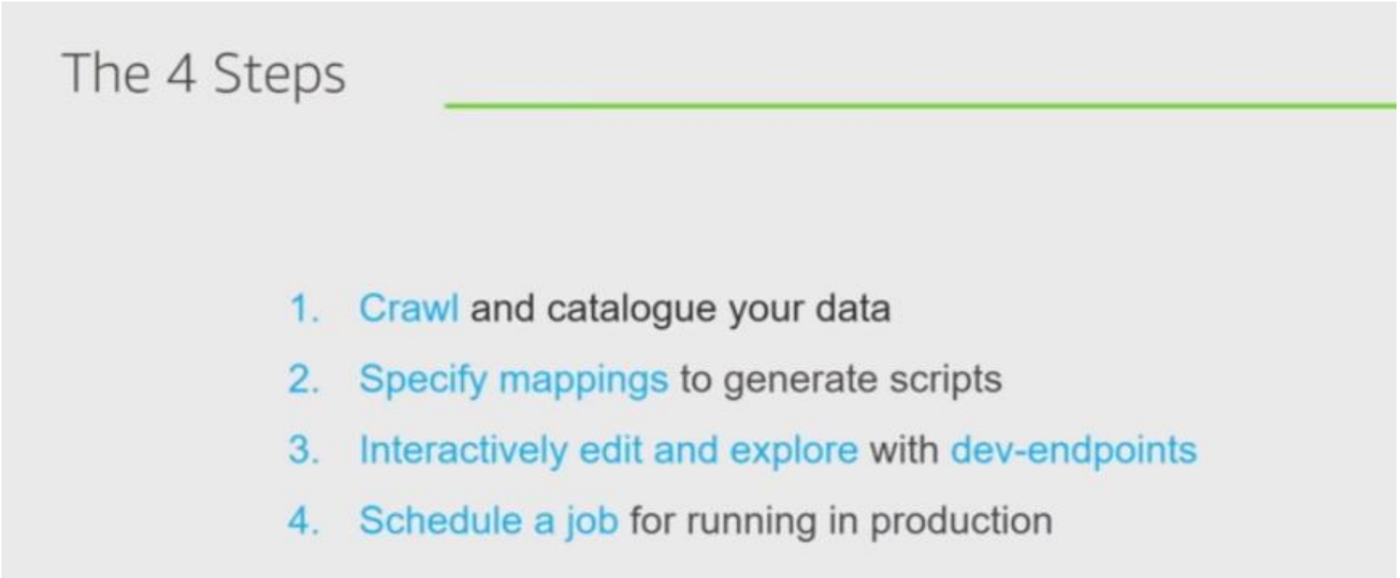
This talk is for developers!



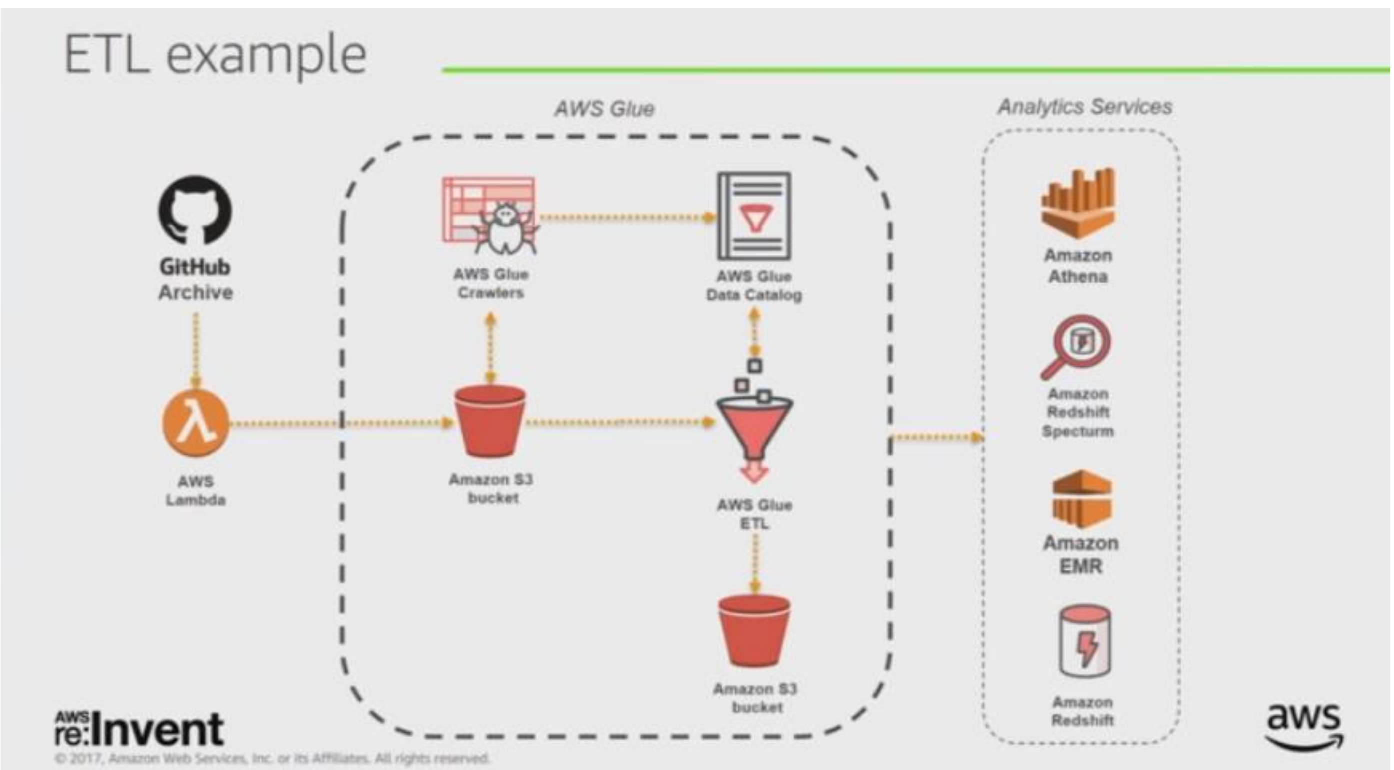
The ***Data Catalog*** helps you discover and understand the data sources that you have, we have crawlers that will automatically extract the data structure when you point them to one of your data sources and store all that information including statistics into the catalog for you. The ***Job Authoring*** and ETL component lets you get started quickly on your ETL job flow design, it generates Python/Spark code for you if you point it to tables and sources inside the data catalog. The ***Job Execution*** system is serverless and can turn your ETL code into a Job and then run it for you, you can also schedule the Jobs and monitor their progress.

Customers get all their data from a variety of different places, integrate them together with AWS Glue, structure it, and then put it into Redshift for later analysis.

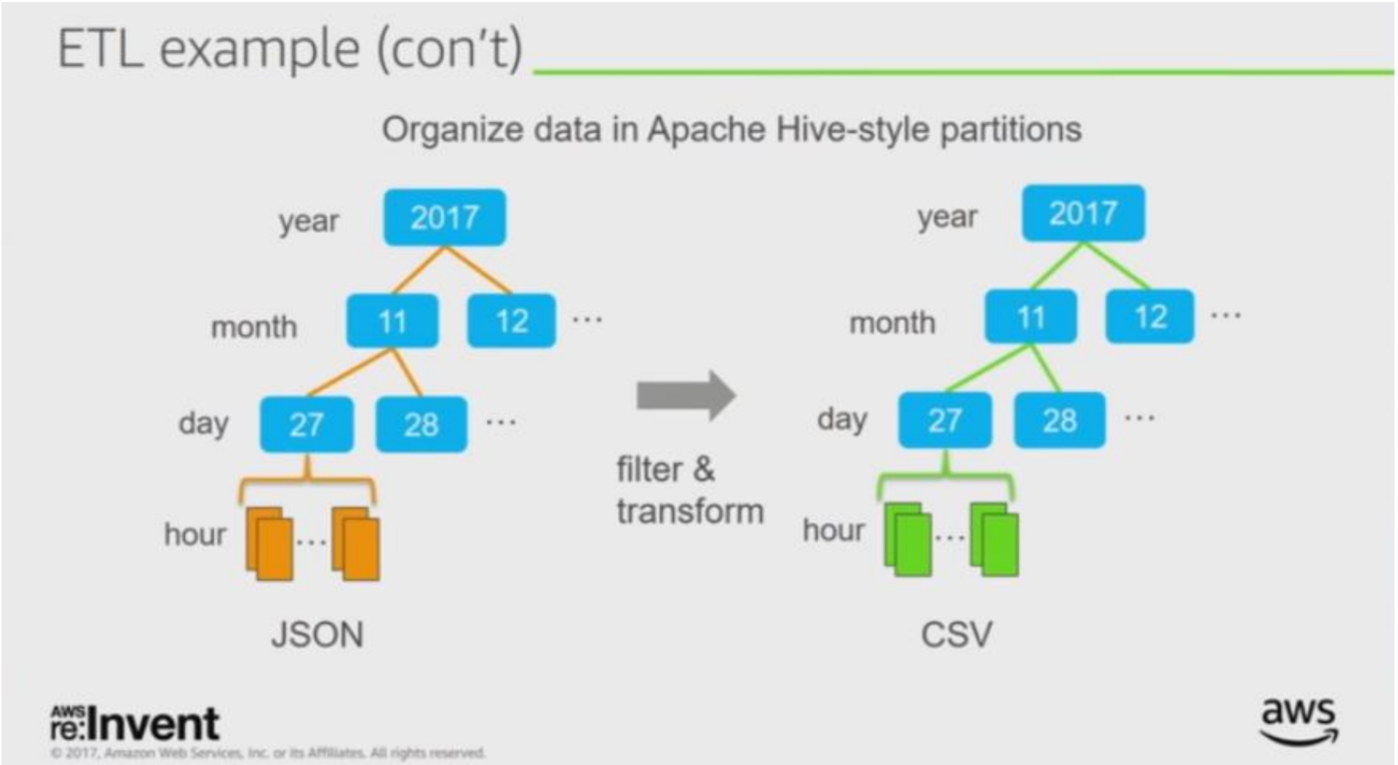
Customers are also building data lakes with Glue instead of warehousing their data. they crawl all their data, index all their information and make that data available for analysis using any of the available services like Athena, EMR, Redshift Spectrum.



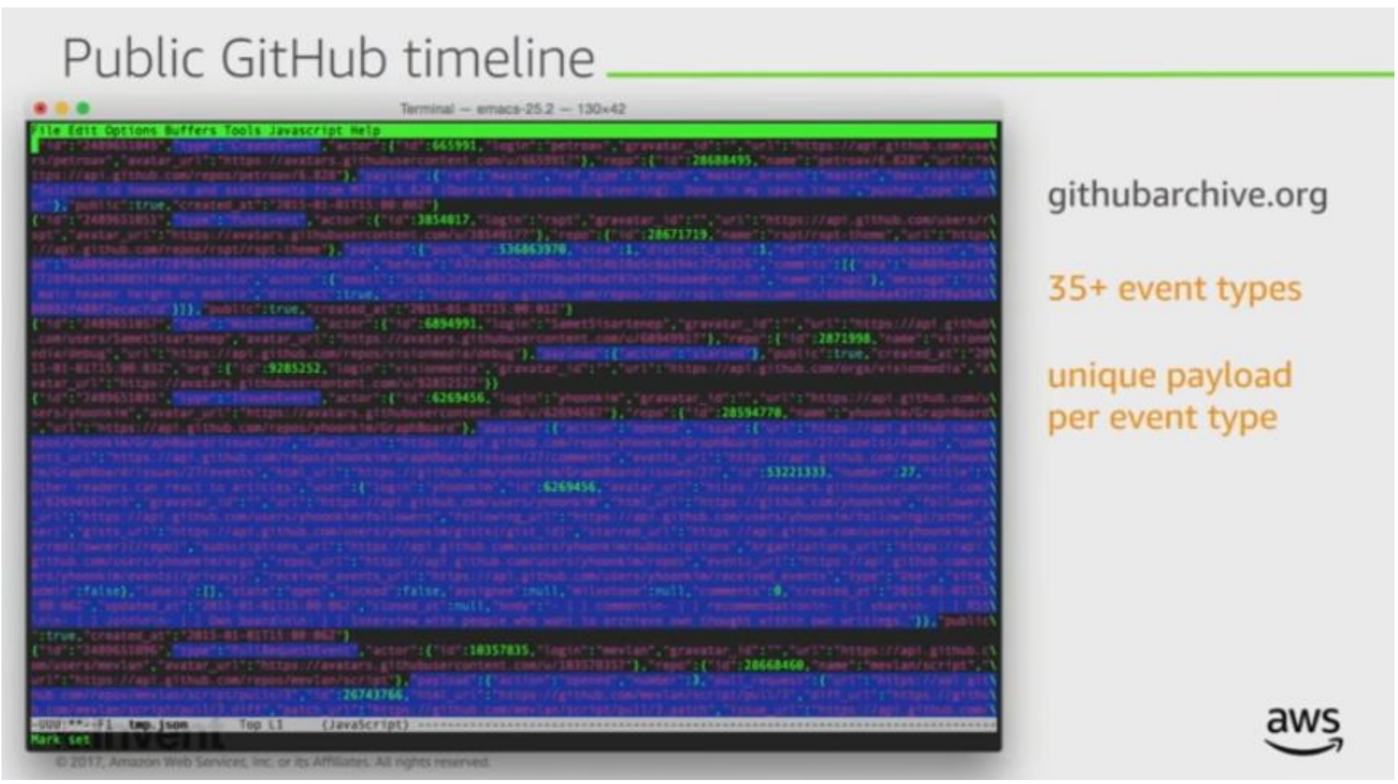
The first step is to point the crawlers and catalog to your data, the crawlers will then figure things out for you automatically. Once you have your data all indexed inside your catalog, you can specify mappings with the AWS Glue console that will then generate an ETL flow code for you. You can edit the generated code using development endpoints. Finally, you can schedule and trigger your jobs in a dynamic fashion.



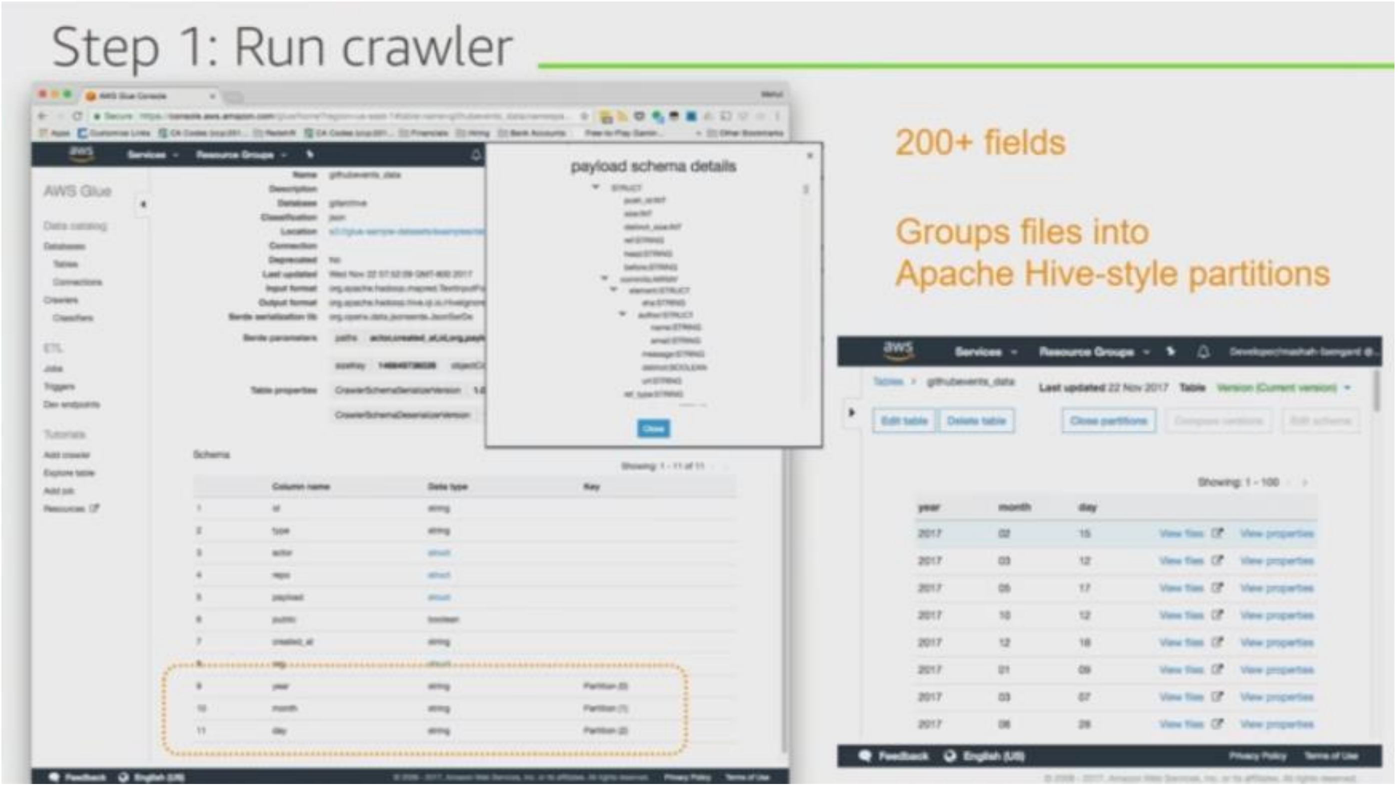
This is an example where we are going to be converting a bunch of JSON data to CSV, this is a typical job used when building data lakes and data warehouses. GitHub Archive is data generated in JSON form hourly for public view.



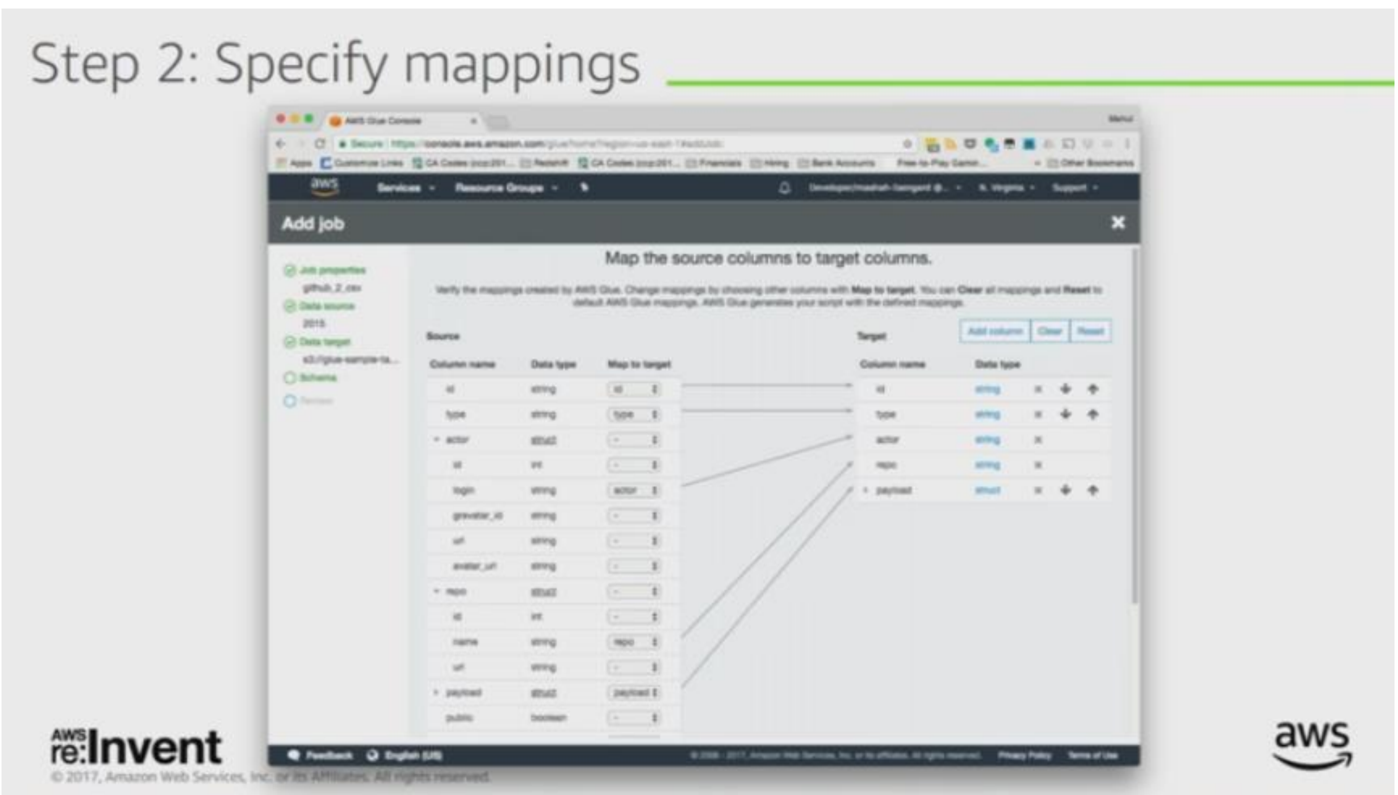
We need to organize the data into directory hierarchy in separate S3 buckets. We are going to be filtering the source JSON data on the left for specific events, transforming the data into CSV format, then try to get the data into the same kind of hierarchy to simplify further analysis.



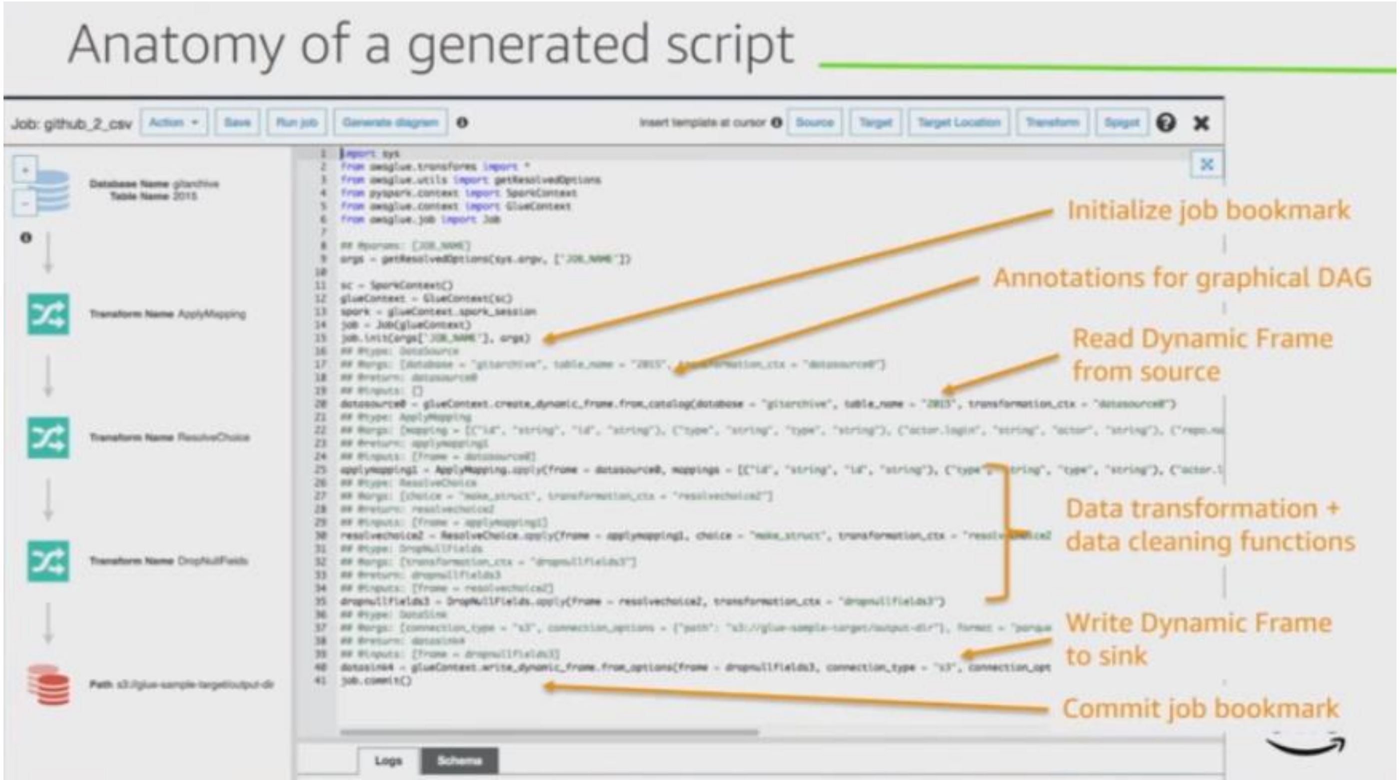
This is the raw data, it includes about 37 event types with varying structure per event type



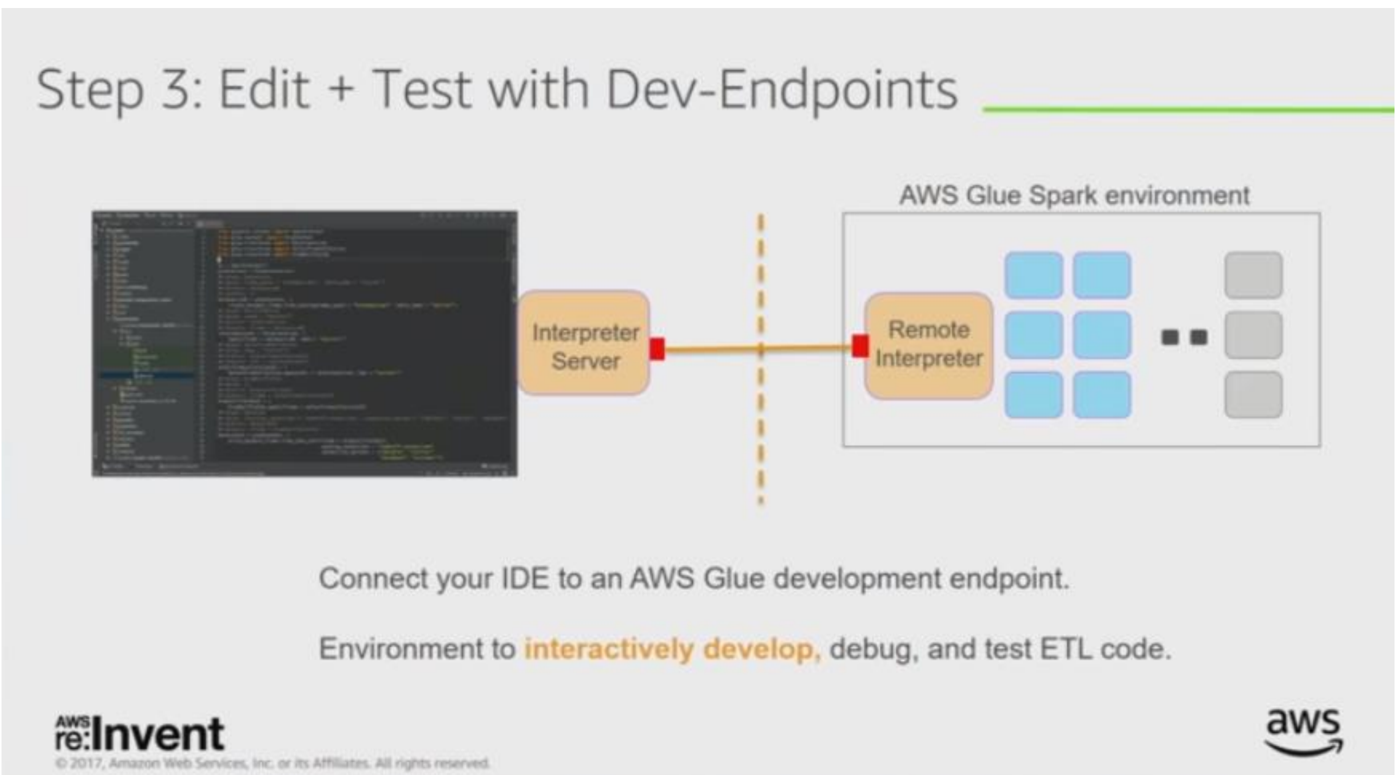
When you point the crawler to this data, it will generate the data schema for you automatically. It will detect the data types, fields and even complex schema structures.

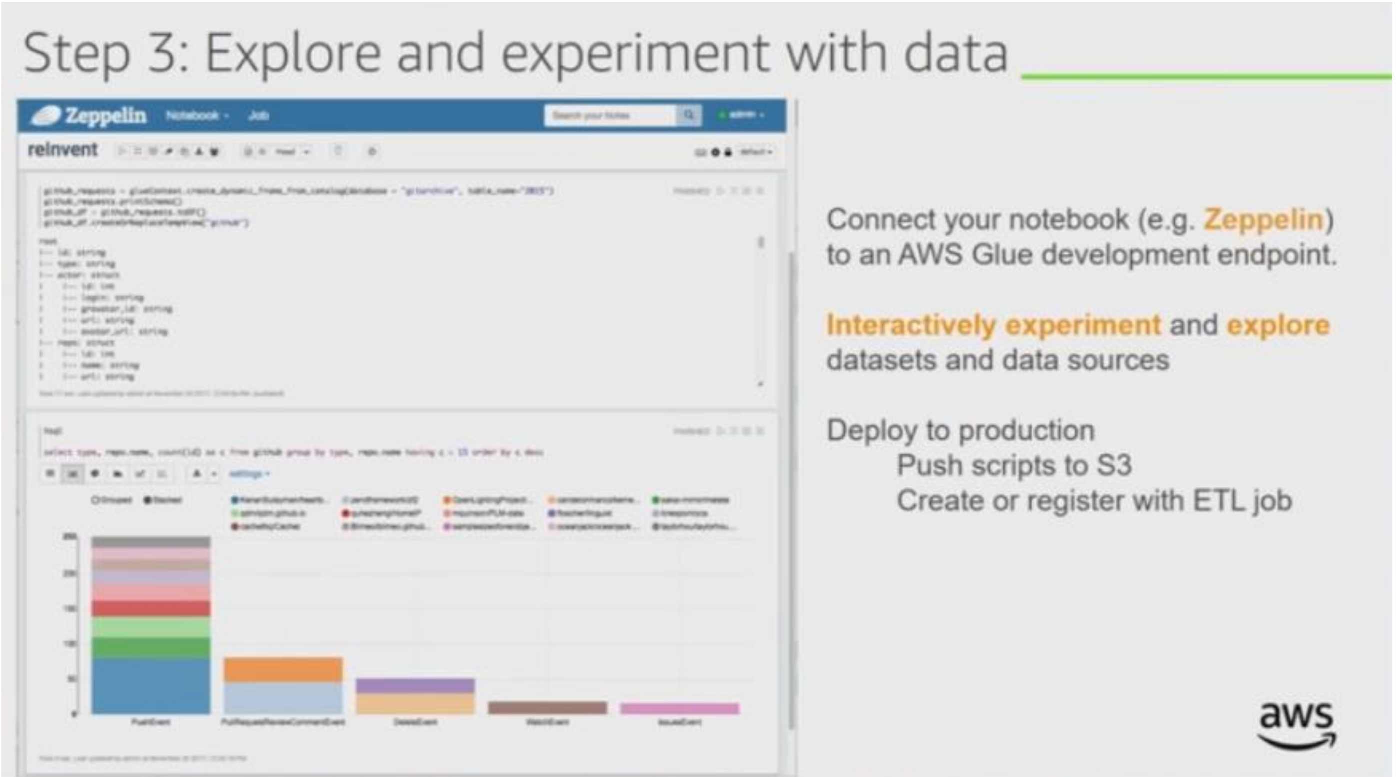


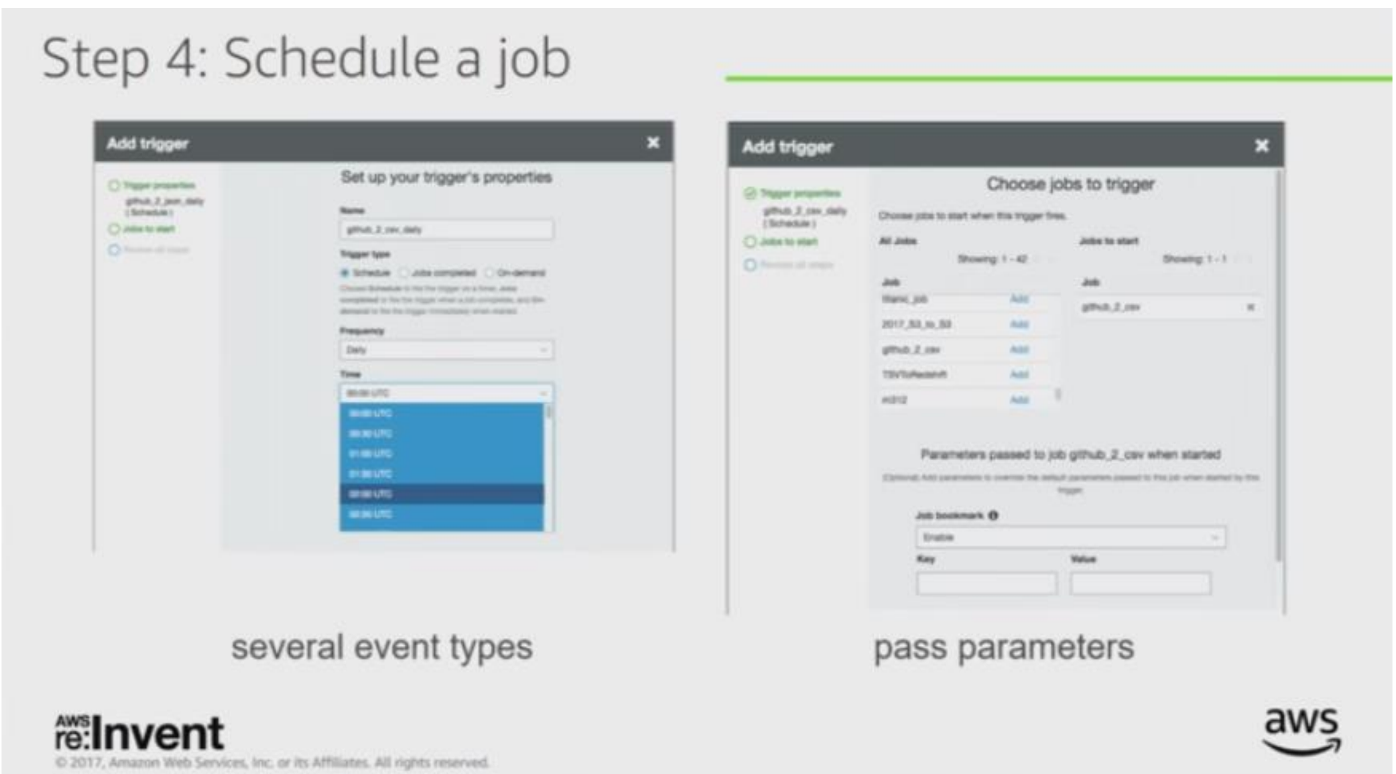
You can start converting the data using the Add Job Service to choose parts of the schema that we want to map from source to destination.



This then generates a script for you for this job.





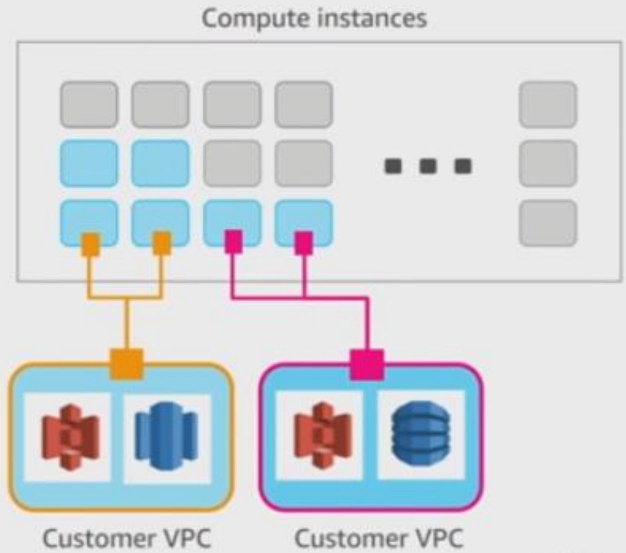


Once you have the job registered with Glue, you can trigger the job based on several conditions and event types like using a Cron expression to run it on a schedule, you can also trigger a job based on the completion of another job to make a jobs pipeline, you can also pass parameters to your job triggers to provide some context.

Serverless job execution

No need to provision, configure, or manage servers

- Auto-configure VPC & role-based access security & isolation preserved
- Customers can specify job capacity (DPU)
- Automatically scale resources
- Only pay for the resources you consume per-second billing (10-minute min)



Under the hood: customize AWS Glue scripts

Apache Spark and AWS Glue ETL

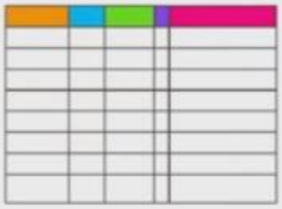
What is Apache Spark?

- Parallel, scale-out data processing engine
- Fault-tolerance built-in
- Flexible interface: Python scripting, SQL
- Rich eco-system: ML, Graph, analytics, …

AWS Glue ETL libraries

- Integration: Data Catalog, job orchestration, code-generation, job bookmarks, S3, RDS
- ETL transforms, more connectors & formats
- New data structure: Dynamic Frames

Dataframes and Dynamic Frames



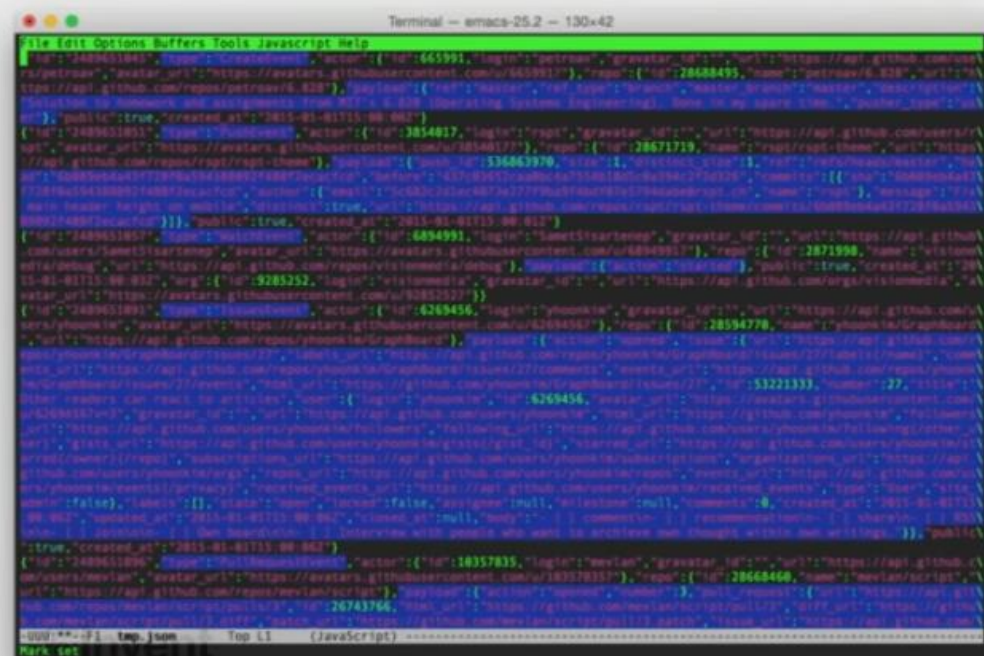
Dataframes

- Core data structure for SparkSQL
- Like structured tables
  - Need schema up-front
  - Each row has same structure
- Suited for SQL-like analytics

Dynamic Frames

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

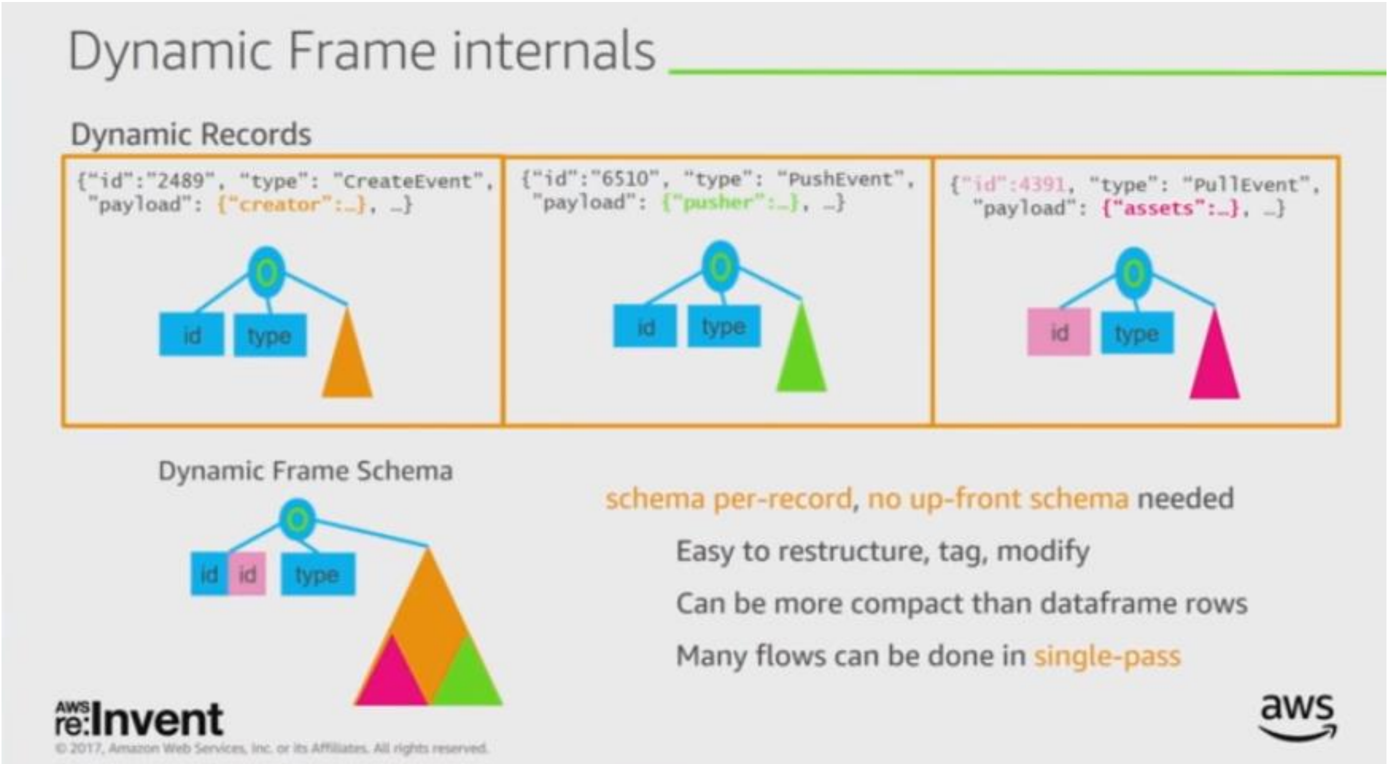
Public GitHub timeline is …

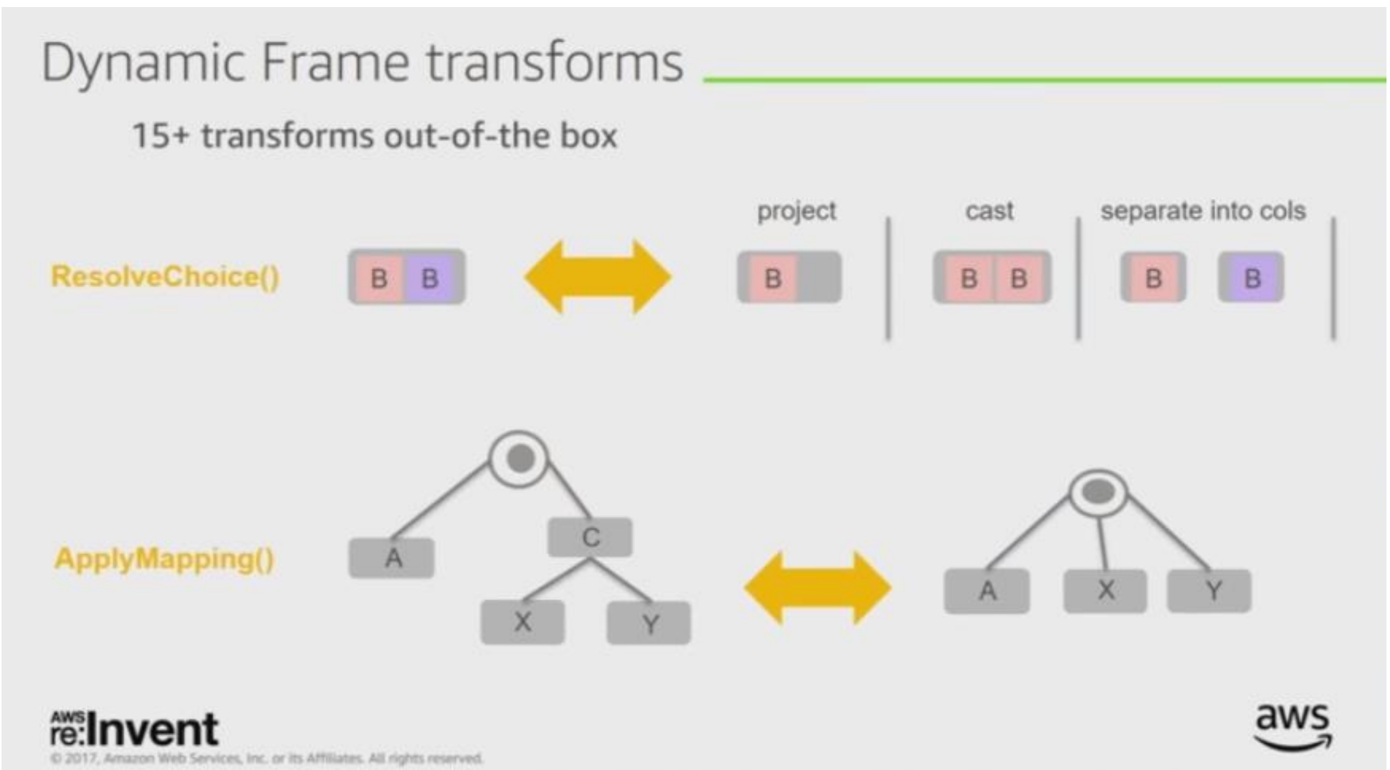


semi-structured

35+ event types

payload structure and size varies by event type


Dynamic Frame internals
Dynamic Records
{"id":"2489", "type": "CreateEvent", "payload": {"creator":…}, …}
{"id":"6510", "type": "PushEvent", "payload": {"pusher":…}, …}
{"id":4391, "type": "PullEvent", "payload": {"assets":…}, …}
id
type
Dynamic Frame Schema
schema per-record, no up-front schema needed
Easy to restructure, tag, modify
Can be more compact than dataframe rows
Many flows can be done in single-pass
re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws


Dynamic Frame transforms
15+ transforms out-of-the box
ResolveChoice()
project
cast
separate into cols
B
ApplyMapping()
A
C
X
Y
re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws

Relationalize() transform

Transforms and **adds new** columns, types, and tables on-the-fly

Tracks **keys** and **foreign keys** across runs

SQL on the relational schema is orders of **magnitude faster** than JSON processing

Useful AWS Glue transforms

toDF(): Convert to a Dataframe

Spigot(): Sample data of any Dynamic Frame to S3

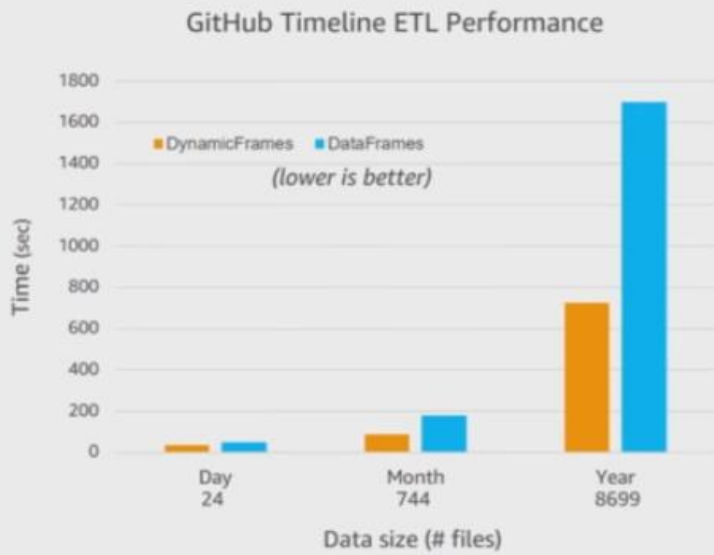
Unbox(): Parse string column as given format into Dynamic Frame

Filter(), Map(): Apply Python UDFs to Dynamic Frames

Join(): Join two Dynamic Frames

And more ....

Performance: AWS Glue ETL



Configuration
- 10 DPUs
- Apache Spark 2.1.1

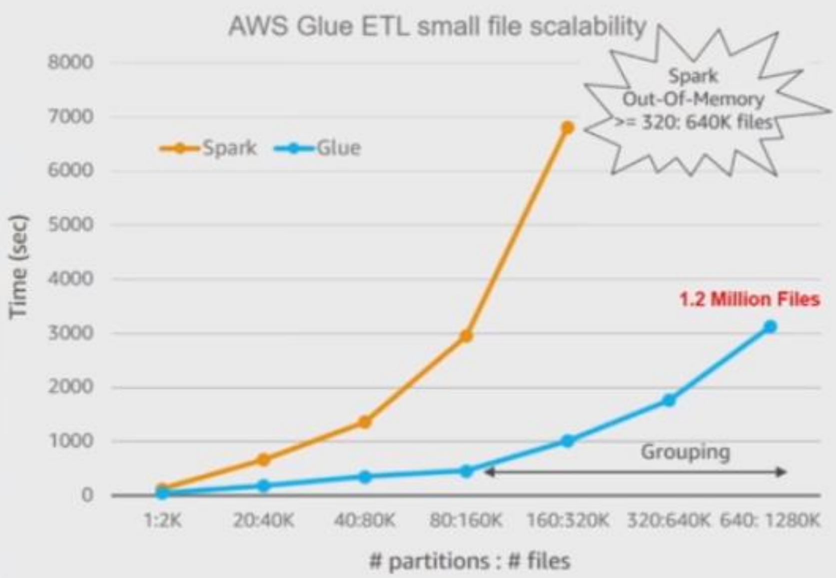
Workload
- JSON to CSV
- Filter for Pull events

On average: 2x performance improvement

AWS re:Invent

aws

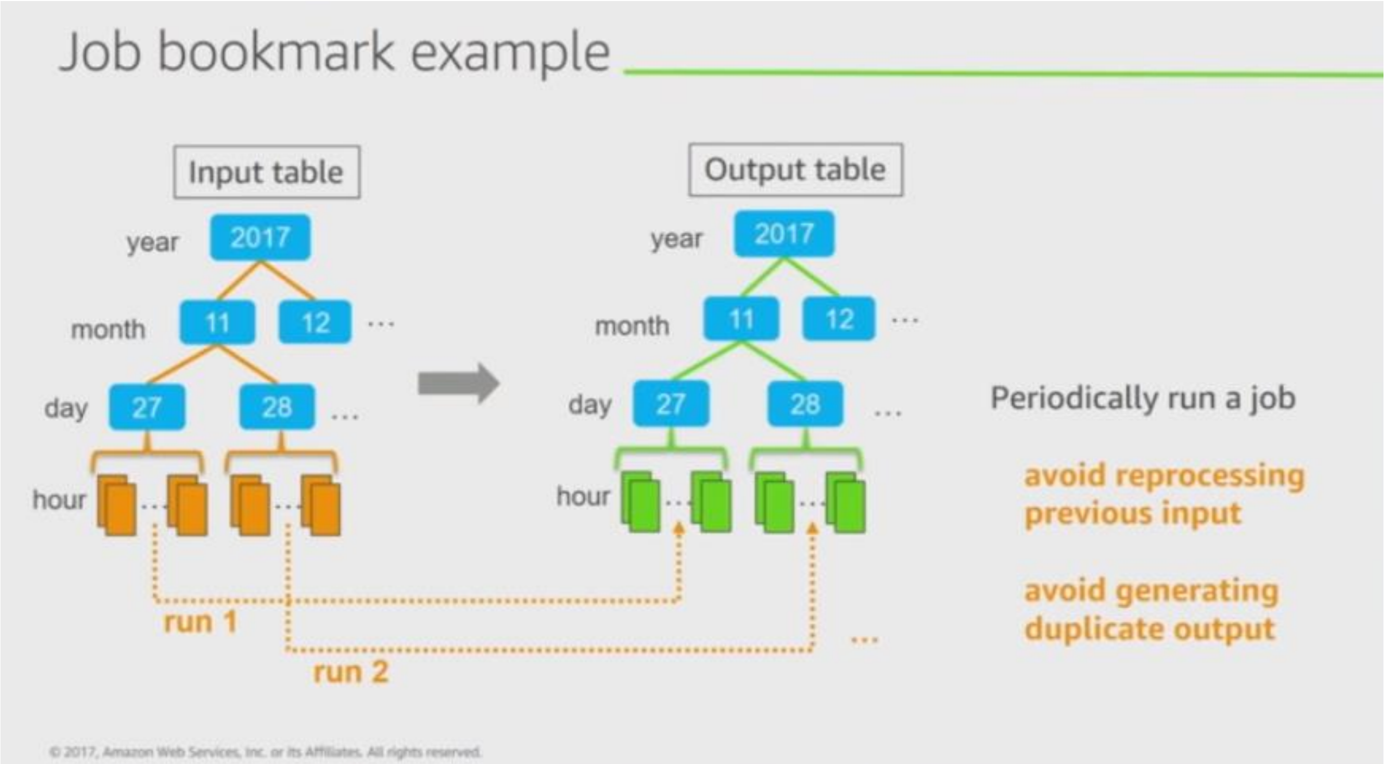
Performance: Lots of small files



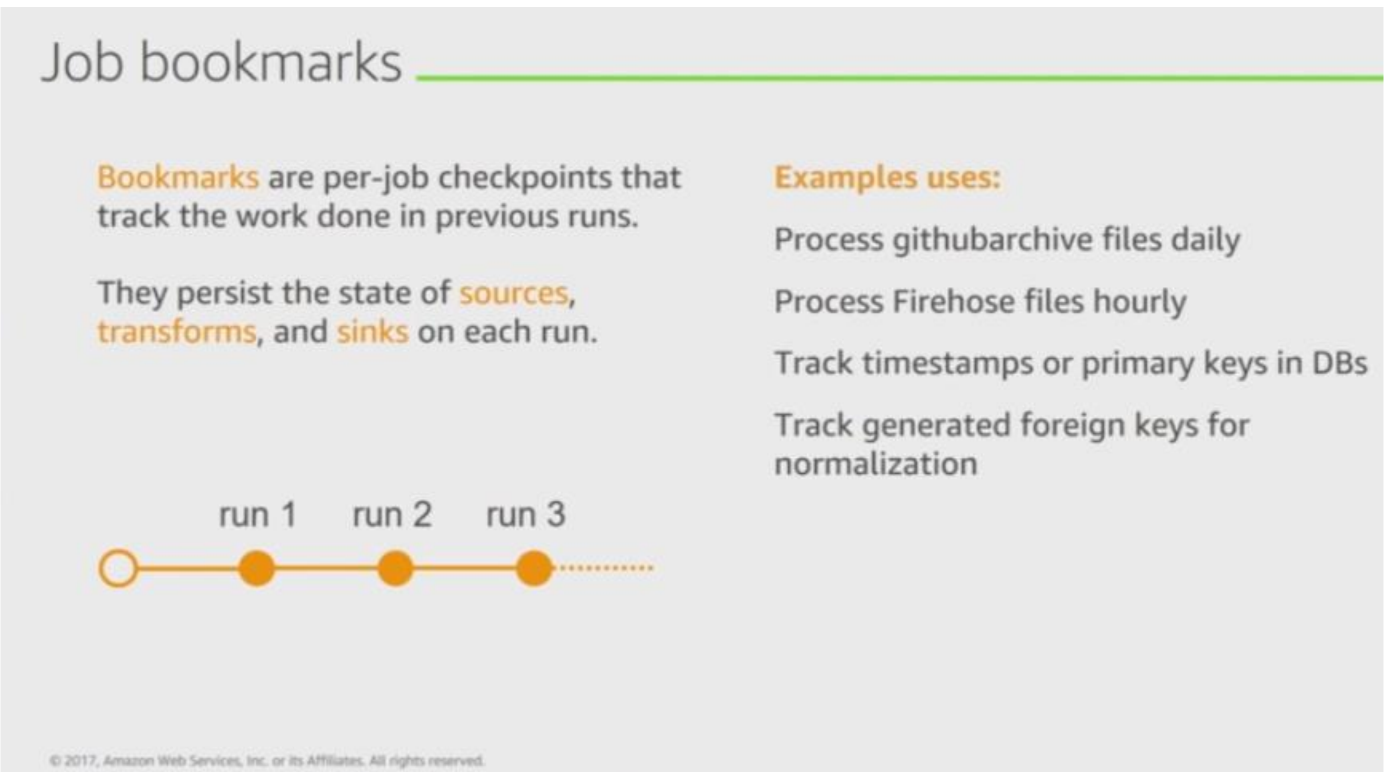
Lots of small files, e.g. Kinesis Firehose

Vanilla Apache Spark (2.1.1) overheads
- Must reconstruct partitions (2-pass)
- Too many tasks: task per file
- Scheduling & memory overheads

AWS Glue Dynamic Frames
- Integration with Data Catalog
- Automatically group files per task
- Rely on crawler statistics



A Job Bookmark lets the script know the last place you stopped processing your data so that the periodic job can start from there the next time and not have to process already processed data. e.g. you don't want to process jobs from the previous day with a daily scheduled job.



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 githubarchive partition

Disable: Process the entire githubarchive table

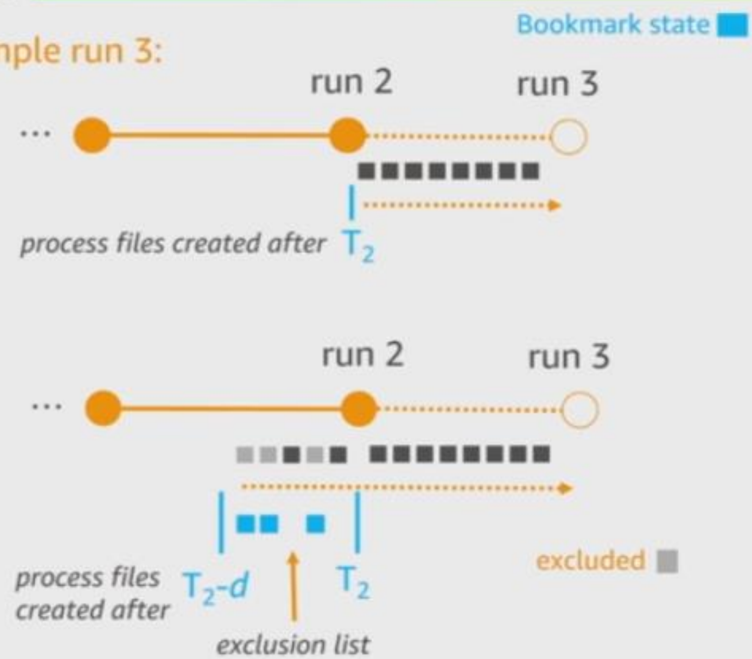
Pause: Process the previous githubarchive partition

Job bookmark internals

Example run 3:

How do we avoid space blowup?
Use timestamps to filter already processed input

But S3 is eventually consistent?
Maintain exclusion list of files created in *inconsistency window* (size *d*) prior to start.



Wrap Up

4 steps to build a production ETL flow

AWS Glue features
- Dynamic frames
- Job bookmarks

AWS Glue Announcements

Scala support

New regions: Asia Pacific (Tokyo) & EU (Ireland)

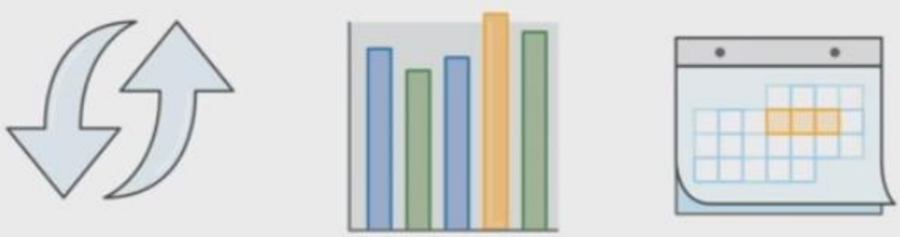
Merck – customer testimonial

Keith Smola

Global Operations Management, Merck & Co.

Problem

We were challenged to quickly align extended data and metrics to a Enterprise Software Delivery Management system. One capability of the system is Environment Management, which is used to provide the knowledge regarding environments, what assets make them up and how they align to different software lifecycle efforts. Other capabilities include deployment management, delivery planning and scope management.



Limited integration with the Enterprise Software Delivery Management System	BI tools require a data source with application lifecycle context	Major projects require fast alignment to Enterprise Software Delivery Management System

AWS re:Invent

aws

Challenges

Short timeline to produce a data layer.

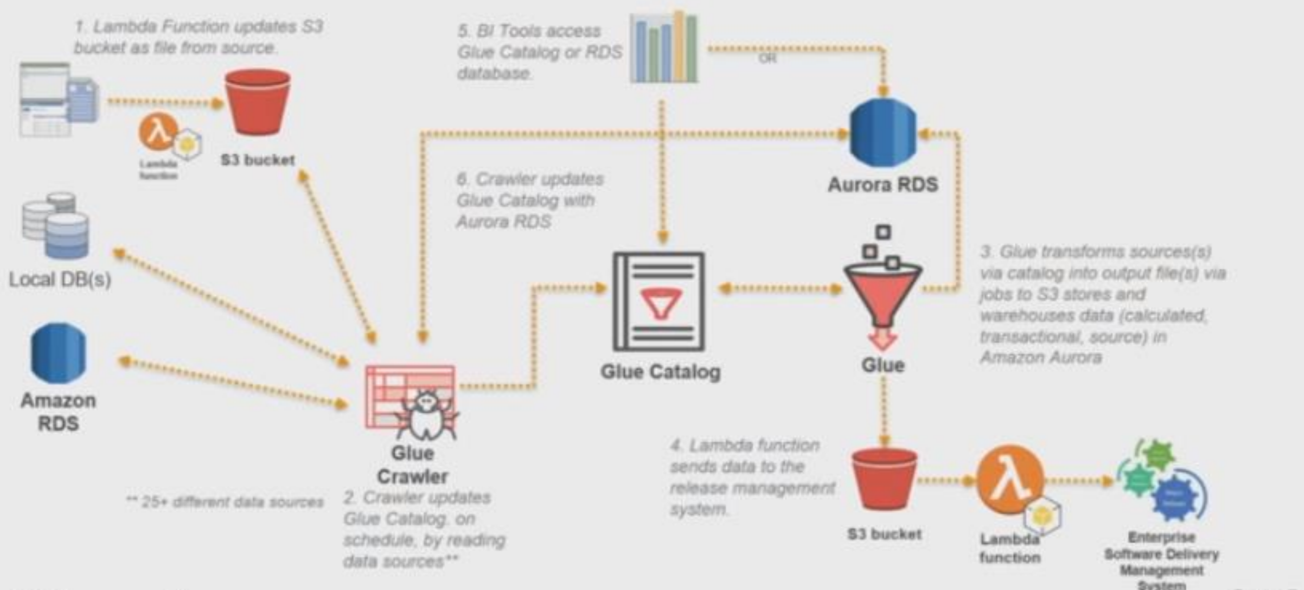
Resources are not data scientists or ETL developers.

Established support does not go beyond AWS.

Data is in different spreadsheets or existing databases

Catalog vs. warehousing

Architecture



Value

Leverage existing resources.

Glue was quick to learn.

No servers means no additional resources to manage, procure or maintain.

Easy to adapt to new data sources and scale availability within short timeframe.

Provided a single source of data (aligned and calculated).

Enabled a scalable data layer/lake

Projects were able to use the data via our Release Management System.

AWS re:Invent

aws