

ABD315

Serverless ETL with AWS Glue

Mehul A. Shah

Software Manager, AWS Glue

November 27, 2017

aws
re:Invent



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.

Today's Agenda

Intro to AWS Glue

Construct an ETL flow in 4 steps

Under the hood: customize AWS Glue scripts

Merck – customer testimonial

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



News Corp



AUTODESK



MERCK

myTomorrows



amazon Prime Air



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

There are many tools already

Amazon Redshift Partner Page for Data Integration



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!

Hand-coding is laborious



AWS Glue does the undifferentiated heavy lifting
so developers can easily customize

AWS Glue Components



Data Catalog

Discover

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



Job Authoring

Develop

- Auto-generates ETL code
- Python and Apache Spark
- Edit, Debug, and Explore



Job Execution

Deploy

- Serverless execution
- Flexible scheduling
- Monitoring and alerting

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.

Common use-cases

Load data warehouses



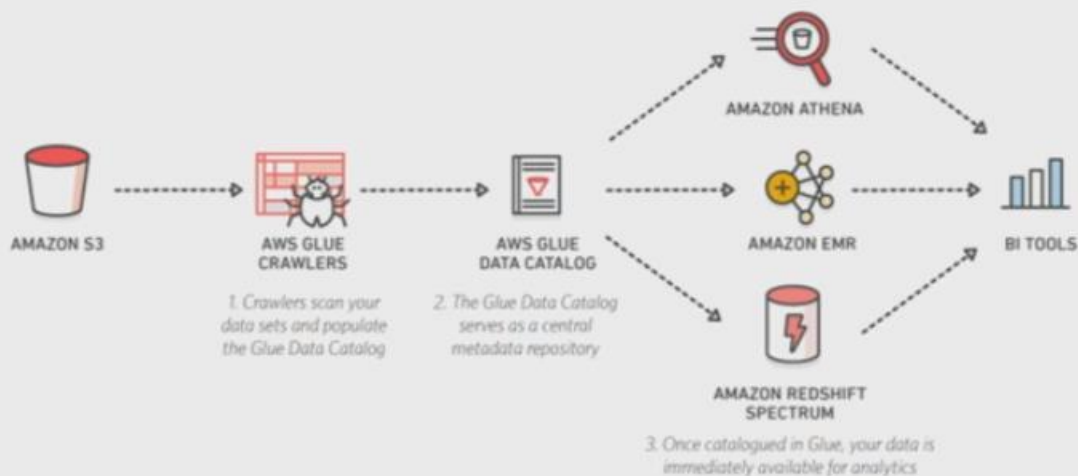
AIWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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.

Build a data lake on Amazon S3



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.

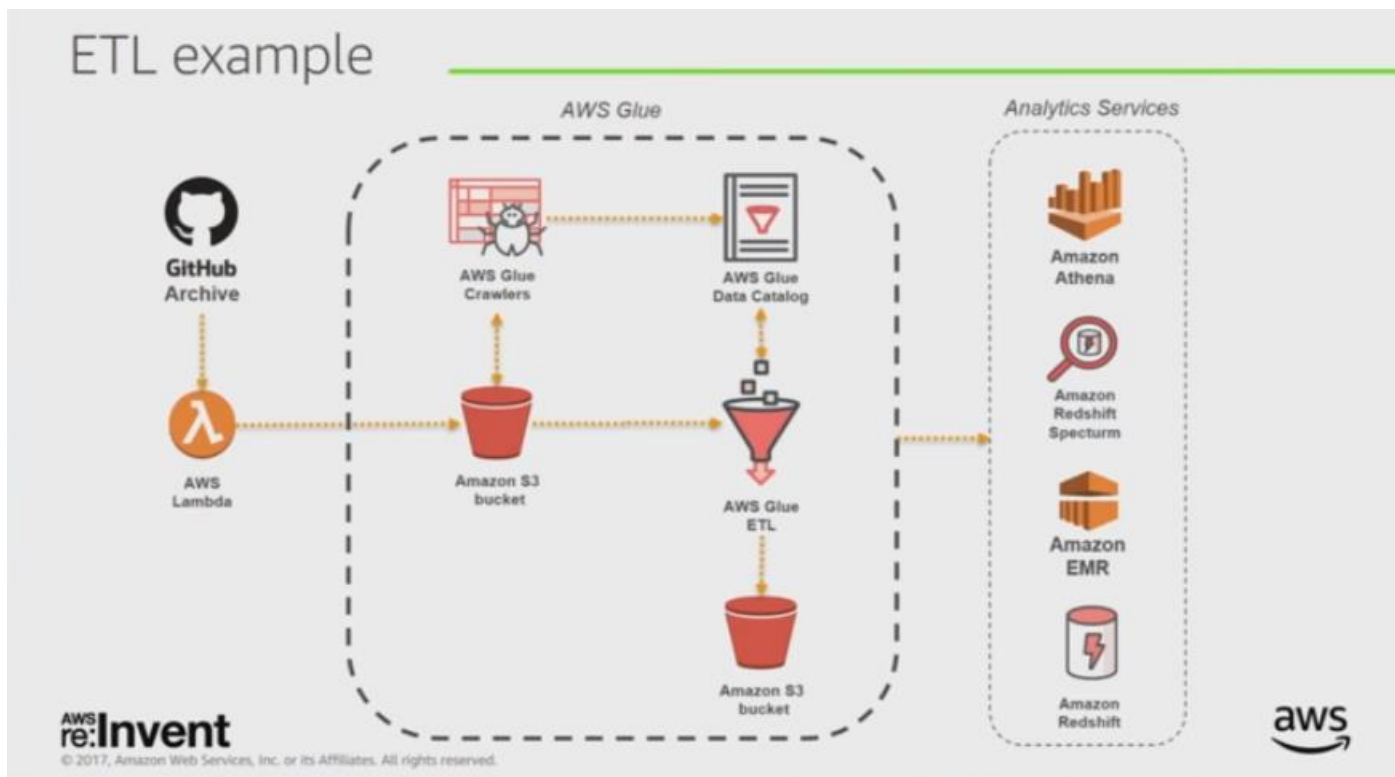
Construct an ETL flow in 4 steps

The 4 Steps

1. **Crawl** and catalogue your data
2. **Specify mappings** to generate scripts
3. **Interactively edit and explore** with **dev-endpoints**
4. **Schedule a job** for running in production

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.

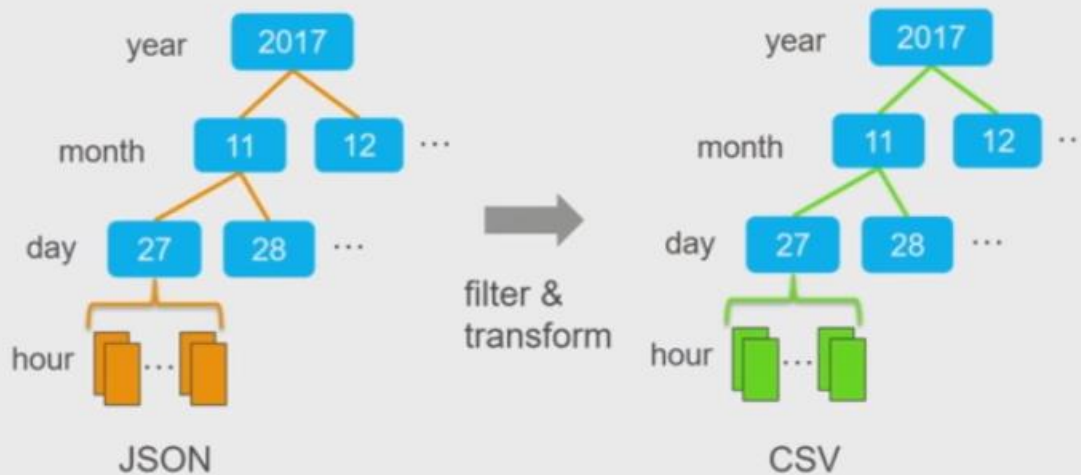
ETL example



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.

ETL example (con't)

Organize data in Apache Hive-style partitions



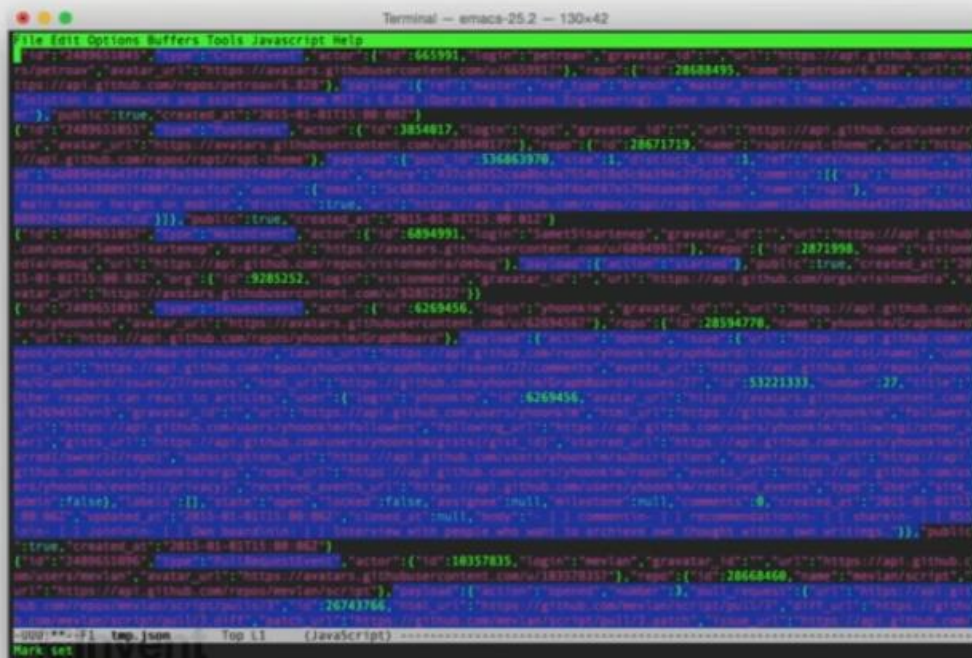
AWS
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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.

Public GitHub timeline



githubarchive.org

35+ event types

unique payload
per event type



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

Step 1: Run crawler

The screenshot shows the AWS Glue console interface. On the left, the 'glueevents_data' crawler job is selected. The main panel displays the 'payload schema details' for the crawler. A table lists the schema columns and their data types:

Column name	Data type	Key
id	string	
type	string	
actor	string	
repo	string	
payload	string	
public	boolean	
created_at	string	
year	string	Partition (1)
month	string	Partition (2)
day	string	Partition (3)

On the right, a table shows the data partitions for the crawler job, grouped by year, month, and day. The table has columns for year, month, day, and a link to view the file properties.

200+ fields

Groups files into Apache Hive-style partitions

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.

Step 2: Specify mappings

The screenshot shows the 'Add Job' screen in the AWS Glue console. The 'Map the source columns to target columns' section is active. It displays a table of source columns and their data types, and a table of target columns and their data types. Arrows indicate the mapping from source columns to target columns.

Source

Column name	Data type	Map to target
id	string	id
type	string	type
actor	string	actor
repo	string	repo
payload	string	payload
public	boolean	public

Target

Column name	Data type
id	string
type	string
actor	string
repo	string
payload	string
public	boolean

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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.

Anatomy of a generated script

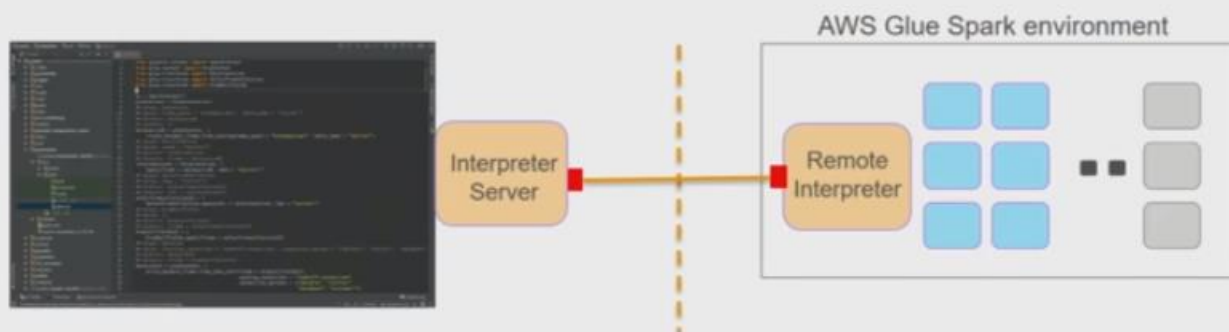
The screenshot shows the AWS Glue console interface for a job named 'github_2_csv'. On the left, a graphical DAG (Data Flow Diagram) shows the job's execution flow: a source (Database Name: gluetravis, Table Name: 2015) feeds into a transform (Transform Name: ApplyMapping), which feeds into another transform (Transform Name: ResolveChoice), which feeds into a third transform (Transform Name: DropNullFields), which finally feeds into a sink (Path: s3://glue-sample-target/output-dir).

The main part of the screenshot shows the generated PySpark script. Annotations with orange arrows point to specific lines of code:

- Initialize job bookmark:** Points to line 1: `report sys`
- Annotations for graphical DAG:** Points to lines 2-6, which import necessary modules like `from awsglue.transforms import *`, `from awsglue.utils import getResolvedOptions`, `from pyspark.context import SparkContext`, `from awsglue.context import GlueContext`, and `from awsglue.job import Job`.
- Read Dynamic Frame from source:** Points to lines 17-21, which define the source and create a dynamic frame: `data_source = glueContext.create_dynamic_frame_from_catalog(database = "gluetravis", table_name = "2015", transformation_ctx = "data_source")`.
- Data transformation + data cleaning functions:** Points to lines 22-35, which apply mappings and resolve choices: `applymapping1 = ApplyMapping.apply(frame = data_source, mappings = [{"id": "string", "id": "string", "type": "string", "type": "string", "actor": "actor"}])`.
- Write Dynamic Frame to sink:** Points to lines 36-40, which write the dynamic frame to the sink: `data_sink = glueContext.write_dynamic_frame_from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet")`.
- Commit job bookmark:** Points to line 41: `job.commit()`

This then generates a script for you for this job.

Step 3: Edit + Test with Dev-Endpoints



Connect your IDE to an AWS Glue development endpoint.

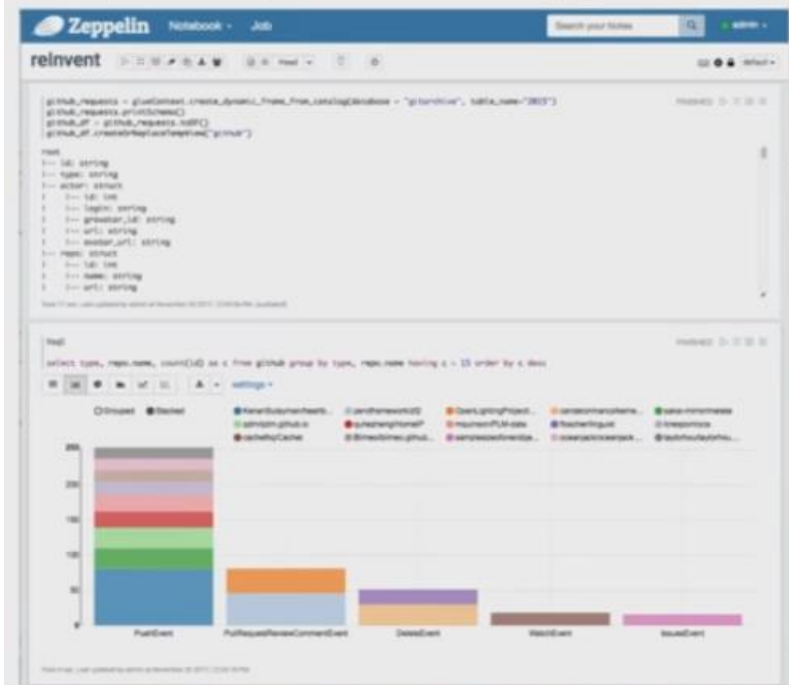
Environment to **interactively develop**, debug, and test ETL code.

aws
re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Step 3: Explore and experiment with data



Connect your notebook (e.g. **Zeppelin**) to an AWS Glue development endpoint.

Interactively experiment and **explore** datasets and data sources

Deploy to production
Push scripts to S3
Create or register with ETL job



Step 4: Schedule a job

The screenshot shows the 'Add trigger' dialog box with the 'Set up your trigger's properties' tab selected. The dialog has a sidebar on the left with three options: 'Trigger properties', 'Jobs to start', and 'Parameters to pass'. The main area contains the following fields and controls:

- Name:** A text input field with the value 'github_2_rev_daily'.
- Trigger type:** A radio button group with three options: 'Schedule' (selected), 'Jobs completed', and 'On-demand'.
- Frequency:** A dropdown menu with the value 'Daily'.
- Time:** A time picker showing '00:00 UTC'.

several event types

The screenshot shows the 'Add trigger' dialog box with the 'Choose jobs to trigger' tab selected. The dialog has a sidebar on the left with three options: 'Trigger properties', 'Jobs to start', and 'Parameters to pass'. The main area contains the following fields and controls:

- Choose jobs to start when this trigger fires:** A section with two columns: 'All jobs' and 'Jobs to start'. The 'All jobs' column lists jobs: 'branch_job', '2017_03_to_03', 'github_2_rev', '201704to05', and 'ad12'. The 'Jobs to start' column lists jobs: 'github_2_rev'.
- Parameters passed to job github_2_rev when started:** A section with a 'Job bookmark' dropdown menu (set to 'Enable') and a table with 'Key' and 'Value' columns.

pass parameters



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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

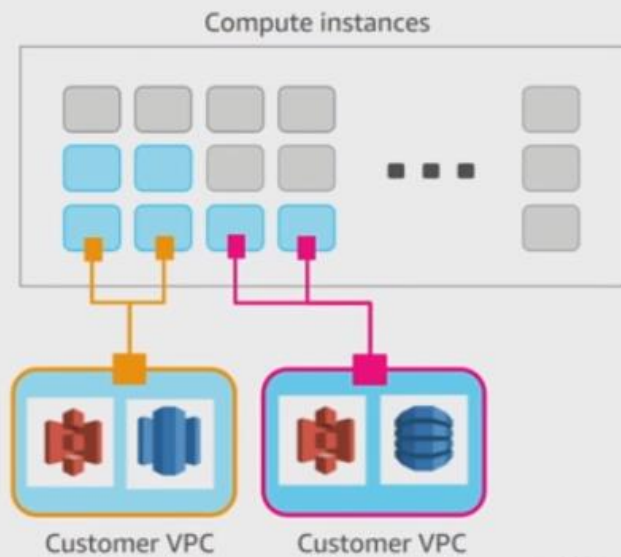
No need to provision, configure, or manage servers

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)



AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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, ...

SparkSQL	AWS Glue ETL
Dataframes	Dynamic Frames
Spark core: RDDs	

AWS Glue ETL libraries

Integration: Data Catalog, job orchestration, code-generation, job bookmarks, S3, RDS

ETL transforms, more connectors & formats

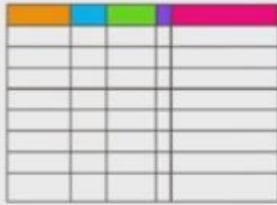
New data structure: Dynamic Frames

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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 ...

re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Public GitHub timeline is ...

semi-structured

35+ event types

payload structure
and size varies by
event type

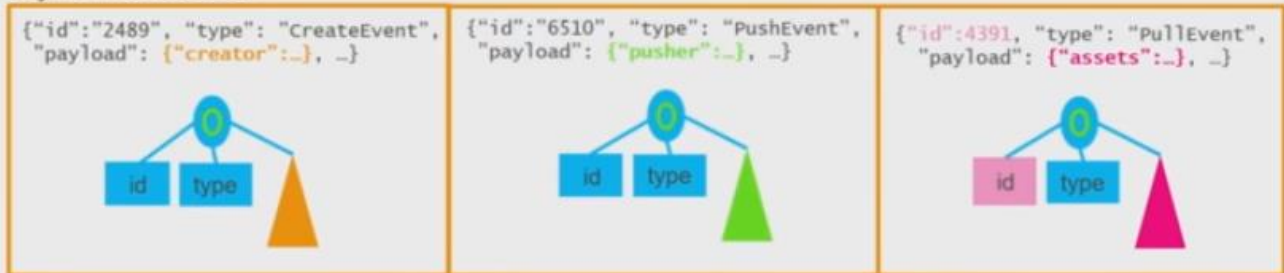
```
Terminal -- emacs-25.2 -- 130x42
file Edit Options Buffers Tools Javascript Help
{"id":"2489651885","type":"push","actor":{"id":"665991","login":"petraue","gravatar_id":"","url":"https://api.github.com/users/petraue","avatar_url":"https://avatars.githubusercontent.com/u/665991"},"repo":{"id":"28688495","name":"petraue/8.828","url":"https://api.github.com/repos/petraue/8.828"},"payload":{"ref":"master","ref_type":"branch","master_branch":"master","description":"Solution to homework 8 and assignment from 8.1 to 8.108. Implementing systems Engineering. Some to be done later. ...","pusher_type":"user"},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"3854017","login":"rpgt","gravatar_id":"","url":"https://api.github.com/users/rpgt","avatar_url":"https://avatars.githubusercontent.com/u/3854017"},"repo":{"id":"28671719","name":"rpgt/rpgt-theme","url":"https://api.github.com/repos/rpgt/rpgt-theme"},"payload":{"sha":"336667779","size":1,"files":[{"file":"rpgt/rpgt-theme","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"336667779","login":"rpgt","gravatar_id":"","url":"https://api.github.com/users/rpgt","avatar_url":"https://avatars.githubusercontent.com/u/336667779"},"repo":{"id":"28671719","name":"rpgt/rpgt-theme","url":"https://api.github.com/repos/rpgt/rpgt-theme"},"payload":{"sha":"336667779","size":1,"files":[{"file":"rpgt/rpgt-theme","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"6894991","login":"SweetSpartaneg","gravatar_id":"","url":"https://api.github.com/users/SweetSpartaneg","avatar_url":"https://avatars.githubusercontent.com/u/6894991"},"repo":{"id":"2871998","name":"visionmedia/debug","url":"https://api.github.com/repos/visionmedia/debug"},"payload":{"sha":"155a1e1","size":1,"files":[{"file":"debug.js","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"9285252","login":"visionmedia","gravatar_id":"","url":"https://api.github.com/users/visionmedia","avatar_url":"https://avatars.githubusercontent.com/u/9285252"},"repo":{"id":"2871998","name":"visionmedia/debug","url":"https://api.github.com/repos/visionmedia/debug"},"payload":{"sha":"155a1e1","size":1,"files":[{"file":"debug.js","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"6269456","login":"yoonkie","gravatar_id":"","url":"https://api.github.com/users/yoonkie","avatar_url":"https://avatars.githubusercontent.com/u/6269456"},"repo":{"id":"28594778","name":"yoonkie/GraphBoard","url":"https://api.github.com/repos/yoonkie/GraphBoard"},"payload":{"sha":"155a1e1","size":1,"files":[{"file":"debug.js","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"10357835","login":"wevian","gravatar_id":"","url":"https://api.github.com/users/wevian","avatar_url":"https://avatars.githubusercontent.com/u/10357835"},"repo":{"id":"28668466","name":"wevian/ecr-script","url":"https://api.github.com/repos/wevian/ecr-script"},"payload":{"sha":"155a1e1","size":1,"files":[{"file":"debug.js","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
{"id":"2489651885","type":"push","actor":{"id":"26743766","login":"yoonkie","gravatar_id":"","url":"https://api.github.com/users/yoonkie","avatar_url":"https://avatars.githubusercontent.com/u/26743766"},"repo":{"id":"28594778","name":"yoonkie/GraphBoard","url":"https://api.github.com/repos/yoonkie/GraphBoard"},"payload":{"sha":"155a1e1","size":1,"files":[{"file":"debug.js","status":"added"}]},"public":true,"created_at":"2015-01-01T15:00:00Z"}
***--if1 top json Top L1 (JavaScript)
Mark: 60s
```

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Dynamic Frame internals

Dynamic Records



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**

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Dynamic Frame transforms

15+ transforms out-of-the box

ResolveChoice()



ApplyMapping()

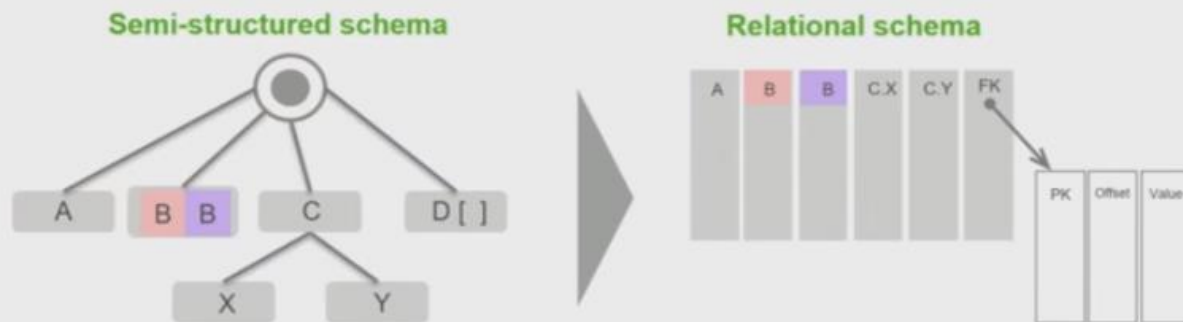


AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

GitHub Timeline ETL Performance



Configuration

10 DPUs
Apache Spark 2.1.1

Workload

JSON to CSV
Filter for Pull events

On average: 2x performance improvement

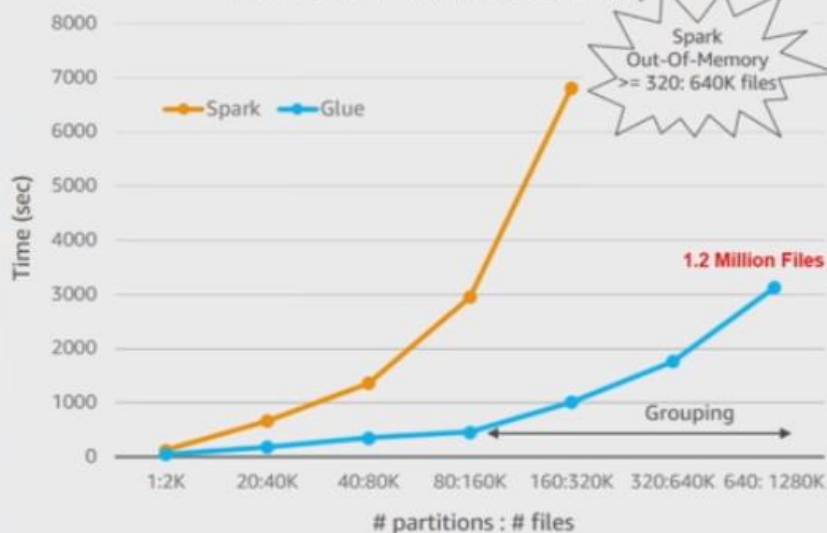
AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Performance: Lots of small files

AWS Glue ETL small file scalability



Lots of small files, e.g. Kinesis Firehose

Vanilla Apache Spark (2.1.1) overheads

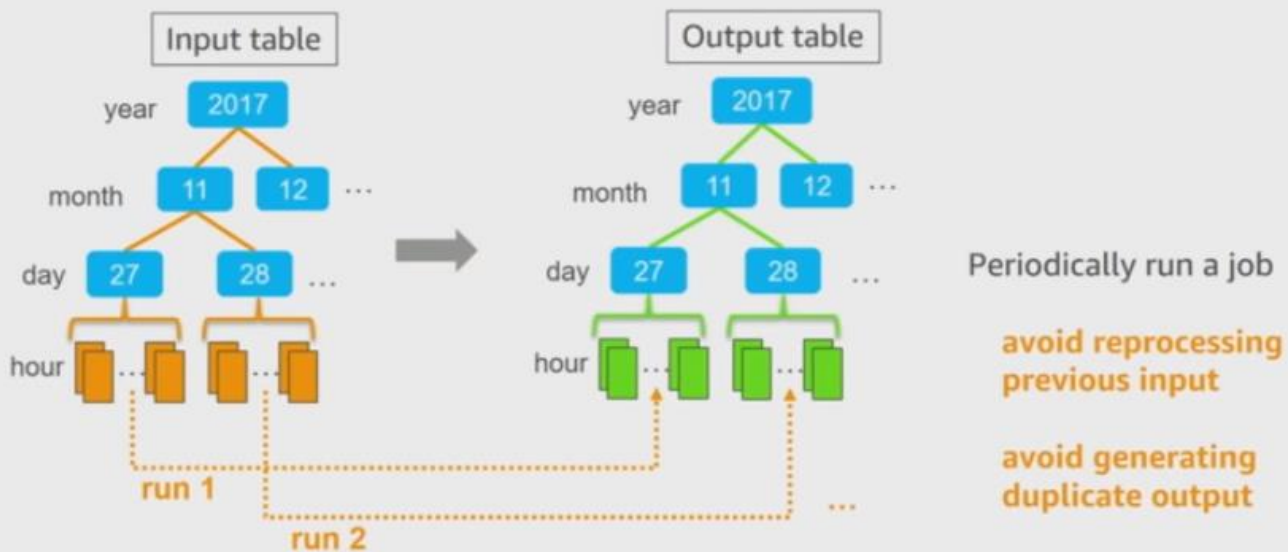
Must reconstruct partitions (2-pass)
Too many tasks: task per file
Scheduling & memory overheads

AWS Glue Dynamic Frames

Integration with Data Catalog
Automatically group files per task
Rely on crawler statistics

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Job bookmark example



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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 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.

Examples uses:

Process githubarchive files daily

Process Firehose files hourly

Track timestamps or primary keys in DBs

Track generated foreign keys for normalization



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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

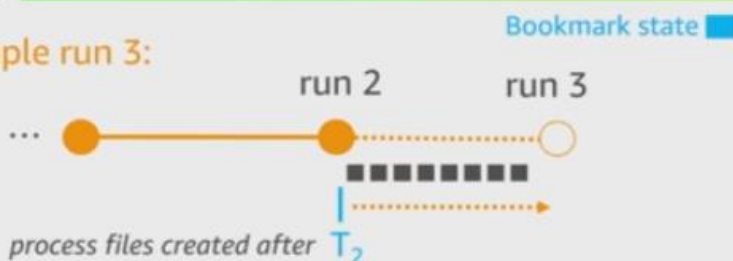
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

Job bookmark internals

Example run 3:

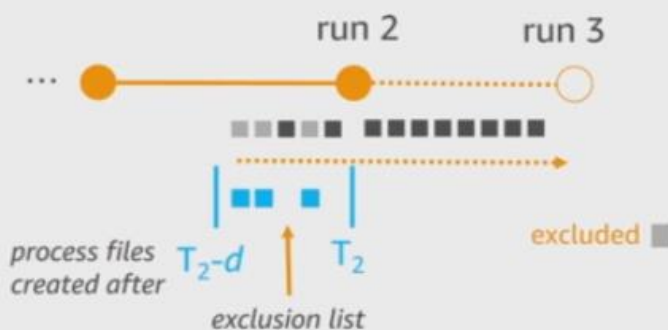
How do we avoid space blowup?

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.



© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

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.

FOR MORE THAN A CENTURY, MERCK HAS BEEN INVENTING

TO SOLVE SOME OF THE GREATEST CHALLENGES TO PEOPLE'S HEALTH AND WELL-BEING AROUND THE WORLD.



BUSINESSES

Prescription medicines, Vaccines, Biologic therapies, Animal Health products



2016 REVENUES

\$39.8 billion, 54% of sales come from outside the United States



2016 R&D EXPENSE

\$10.1 billion, 24 product pipeline programs in late-stage development



HEADQUARTERS
Kenilworth, NJ,
U.S.A., operating in
more than 140
countries



Merck & Co., Inc.
is our legal name and is listed
on the New York Stock Exchange
under the symbol "MRK."



EMPLOYEES
approximately 68,000
worldwide (as of
12/31/16)



MERCK

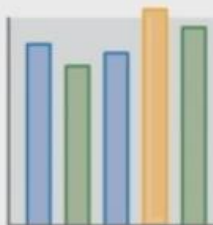
41

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

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Challenges

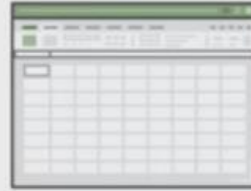


Short timeline to produce a data layer.



Resources are not data scientists or ETL developers.

Established support does not go beyond AWS.



Data is in different spreadsheets or existing databases



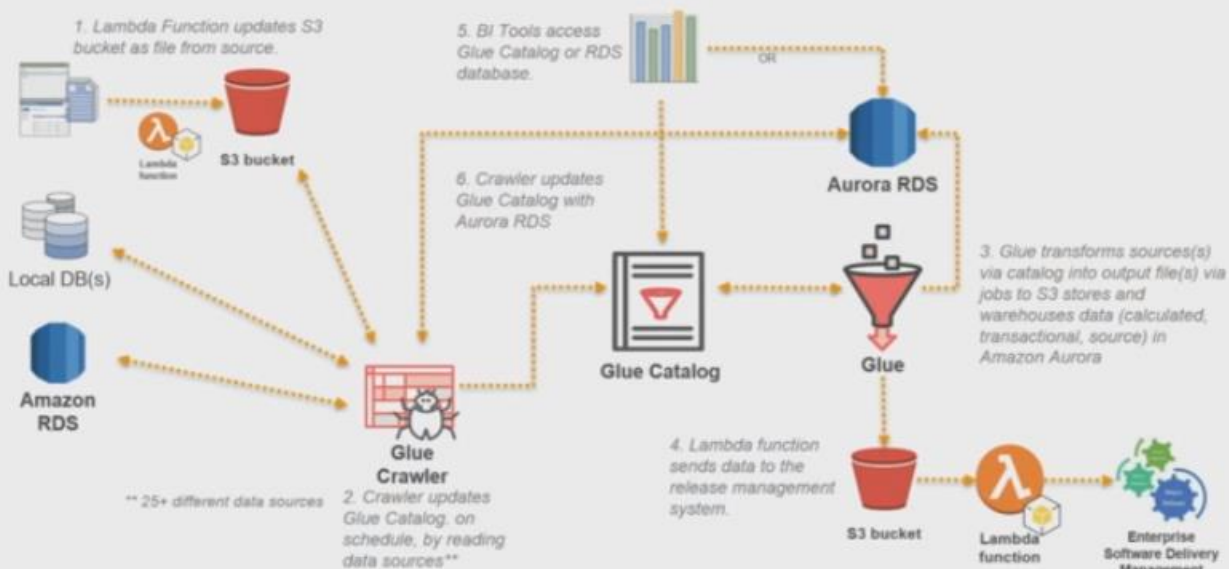
Catalog vs. warehousing

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



Architecture



AWS re:Invent

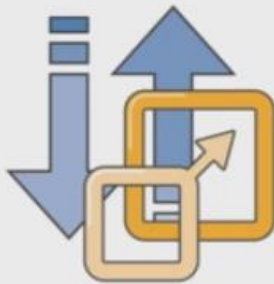
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



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

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

