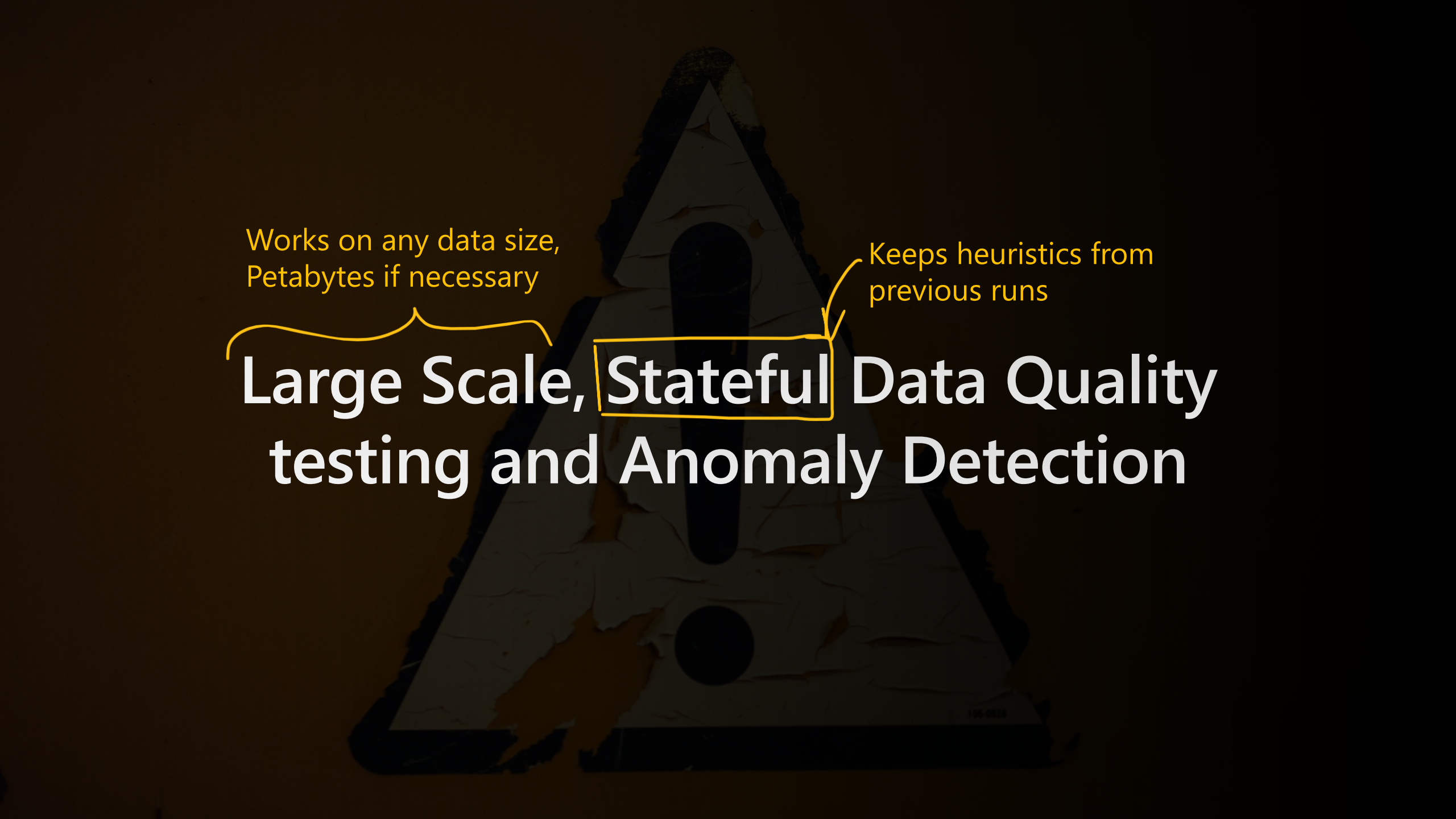

Works on any data size,
Petabytes if necessary
Keeps heuristics from
previous runs
Large Scale, Stateful Data Quality
testing and Anomaly Detection

Agenda

30 mins - Reference Architectures

25 mins - Local demo

5 mins – Next Steps - try for yourself


GOAL
Quantify bad data declaratively
regardless of size or schema


UBER


RAD

Whitepaper Blog Code

- Robust Anomaly Detection works on massive datasets
- Works on high cardinality, false positives and abnormal distributions
- Wraps PCA Apache Pig in Robust PCA C++ implementation
- Used for payment network monitoring and global signup

Robust Principal Component Analysis?

Emmanuel J. Candès[1,2], Xiaodong Li[2], Yi Ma[3,4], and John Wright[4]

[1] Department of Statistics, Stanford University, Stanford, CA 94305

[2] Department of Mathematics, Stanford University, Stanford, CA 94305

[3,4] Electrical and Computer Engineering, UIUC, Urbana, IL 61801

[4] Microsoft Research Asia, Beijing, China

December 17, 2009

Abstract

This paper is about a curious phenomenon. Suppose we have a data matrix, which is the superposition of a low-rank component and a sparse component. Can we recover each component individually? We prove that under some suitable assumptions, it is possible to recover both the low-rank and the sparse components *exactly* by solving a very convenient convex program called *Principal Component Pursuit*; among all feasible decompositions, simply minimize a weighted combination of the nuclear norm and of the ℓ_1 norm. This suggests the possibility of a principled approach to robust principal component analysis since our methodology and results assert that one can recover the principal components of a data matrix even though a positive fraction of its entries are arbitrarily corrupted. This extends to the situation where a fraction of the entries are missing as well. We discuss an algorithm for solving this optimization problem, and present applications in the area of video surveillance, where our methodology allows for the detection of objects in a cluttered background, and in the area of face recognition, where it offers a principled way of removing shadows and specularities in images of faces.

Keywords. Principal components, robustness vis-a-vis outliers, nuclear-norm minimization, ℓ_1-norm minimization, duality, low-rank matrices, sparsity, video surveillance.

1 Introduction

1.1 Motivation

Suppose we are given a large data matrix M, and know that it may be decomposed as

$$M = L_0 + S_0,$$

where L_0 has low-rank and S_0 is sparse; here, both components are of arbitrary magnitude. We do not know the low-dimensional column and row space of L_0, not even their dimension. Similarly, we do not know the locations of the nonzero entries of S_0, not even how many there are. Can we hope to recover the low-rank and sparse components both accurately (perhaps even exactly) and efficiently?

A *provably correct* and *scalable* solution to the above problem would presumably have an impact on today's data-intensive scientific discovery.[1] The recent explosion of massive amounts of high-

[1] Data-intensive computing is advocated by Jim Gray as the fourth paradigm for scientific discovery [24].

PROS

- Works for Big Data
- Underlying implementation is Vectorized and seems highly efficient

CONS

- Old, unmaintained
- Uses JNI with C++ wrapper - ultra custom maintenance with no top-down optimization possible
- Used originally for Apache Pig (old)

SUMMARY

- If this was the only Big Data Anomaly Detection Algorithm in the world we'd probably take it
- Can we do better?


in Data Sentinel

Blog

- Declarative DSL on top of Apache Spark
- Supports dynamic code generation for Spark SQL
- Works on any size of data
- Comes with rich heuristics:

Data Management

Data Sentinel: Automating data validation

Arun Swami
Technical Leadership ♦ Solve Business Problems With Insights from Big Data Derived Using High Performance Algorithms
March 10, 2020

*Co-authors: **Arun Swami**, **Sriram Vasudevan**, **Sailesh Mittal**, **Jiefu Zheng**, **Joojay Huyn**, **Audrey Alpizar**, **Changling Huang**, **Maneesh Varshney**, **Adrian Fernandez***

Data's value is best realized when prepared and treated correctly. However, when you're working with data at an extensive scale, it's not as easy to make sure that every data set has been cleaned and validated. Back in October 2018, we had an instance at LinkedIn when data quality problems affected the job recommendations platform. Client job views and usage declined by 40 to 60% for a short period of time. Once this decline in views was detected, it took a total of 5 engineers 8 days to identify the root cause and 11 days to resolve the issue. This incident illustrates several takeaways about data quality:

- Poor data quality is difficult to detect and can have significant business impact
- Data debugging is difficult and requires significant engineering resources
- Resolving poor data quality not only requires timely and correct intervention, but also results in significant opportunity costs in potentially delaying other projects and deliverables

This led us to develop Data Sentinel, a platform that automatically validates

PROS

- Works for Big Data
- Uses Apache Spark (which Microsoft has competitive PaaS offerings for)

CONS

- Basically, a marketing blog
- Code is closed-source at LinkedIn

SUMMARY

- Where's the code (even internally)?


Deequ

Whitepaper DQDL Code

- **Well-designed** API - deeply hooks into the Spark Query Optimizer and then implements statistical functions as native optimizable Catalyst Expressions
- Single pass generates all check results
- Exposes rich heuristics, a BYO-metric store and State, overridable base classes
- Turned it into a managed PaaS product for AWS Glue

Automating Large-Scale Data Quality Verification

Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann
Amazon Research
{sseb,langed,phschmid,celikelm,biessmann}@amazon.com

Andreas Grafberger*
University of Augsburg
andreas.grafberger@student.uni-augsburg.de

ABSTRACT

Modern companies and institutions rely on data to guide every single business process and decision. Missing or incorrect information seriously compromises any decision process downstream. Therefore, a crucial, but tedious task for everyone involved in data processing is to verify the quality of their data. We present a system for automating the verification of data quality at scale, which meets the requirements of production use cases. Our system provides a declarative API, which combines common quality constraints with user-defined validation code, and thereby enables 'unit tests' for data. We efficiently execute the resulting constraint validation workload by translating it to aggregation queries on Apache Spark. Our platform supports the incremental validation of data quality on growing datasets, and leverages machine learning, e.g., for enhancing constraint suggestions, for estimating the 'predictability' of a column, and for detecting anomalies in historic data quality time series. We discuss our design decisions, describe the resulting system architecture, and present an experimental evaluation on various datasets.

PVLDB Reference Format:
Sebastian Schelter, Dustin Lange, Philipp Schmidt, Meltem Celikel, Felix Biessmann and Andreas Grafberger. Automating Large-Scale Data Quality Verification. *PVLDB*, 11 (12): 1781-1794, 2018.
DOI: https://doi.org/10.14778/3229863.3229867

1. INTRODUCTION

Data is at the center of modern enterprises and institutions. Online retailers, for example, rely on data to support customers making buying decisions, to forecast demand [7], to schedule deliveries, and more generally, to guide every single business process and decision. Missing or incorrect information seriously compromises any decision process downstream, ultimately damaging the overall effectiveness and efficiency of the organization. The quality of data has effects across teams and organizational boundaries, especially in large organizations with complex systems that result in complex data dependencies. Furthermore, there is a trend across different industries towards more automation of business processes with machine learning (ML) techniques. These techniques are often highly sensitive on input data, as the deployed models rely on strong assumptions about the shape of their inputs [42], and subtle errors introduced by changes in data can be very hard to detect [34]. At the same time, there is ample evidence that the volume of data available for training is often a decisive factor for a model's performance [17, 44]. In modern information infrastructures, data lives in many different places (e.g., in relational databases, in 'data lakes' on distributed file systems, behind REST APIs, or is constantly being scraped from web resources), and comes in many different formats. Many such data sources do not support integrity contraints and data quality checks, and often there is not even an accompanying schema available, as the data is consumed in a 'schema-on-read' manner, where a particular application takes care of the interpretation. Additionally, there is a growing demand for applications consuming semi-structured data such as text, videos and images.

Due to these circumstances, every team and system involved in data processing has to take care of data validation in some way, which often results in tedious and repetitive work. As a concrete example, imagine an on-demand video platform, where the machines that stream videos to users write log files about the platform usage. These log files must regularly be ingested into a central data store to make the data available for further analysis, e.g., as training data for recommender systems. Analogous to all data produced in real-world scenarios, these log files might have data quality issues, e.g., due to bugs in program code or changes in the semantics of attributes. Such issues potentially result in failures of the ingestion process. Even if the ingestion process still works, the errors in the data might cause unexpected errors in downstream systems that consume the data. As a consequence, the team managing the daily ingestion process must validate the data quality of the log files, e.g., by checking for missing values, duplicate records, or anomalies in size.

We therefore postulate that there is a pressing need for increased *automation of data validation*. We present a system that we built for this task and that meets the demands of production use cases. The system is built on the following principles: at the heart of our system is *declarativity*; we want users to spend time on thinking 'how' their data should look like, and not have to worry too much about how to im-

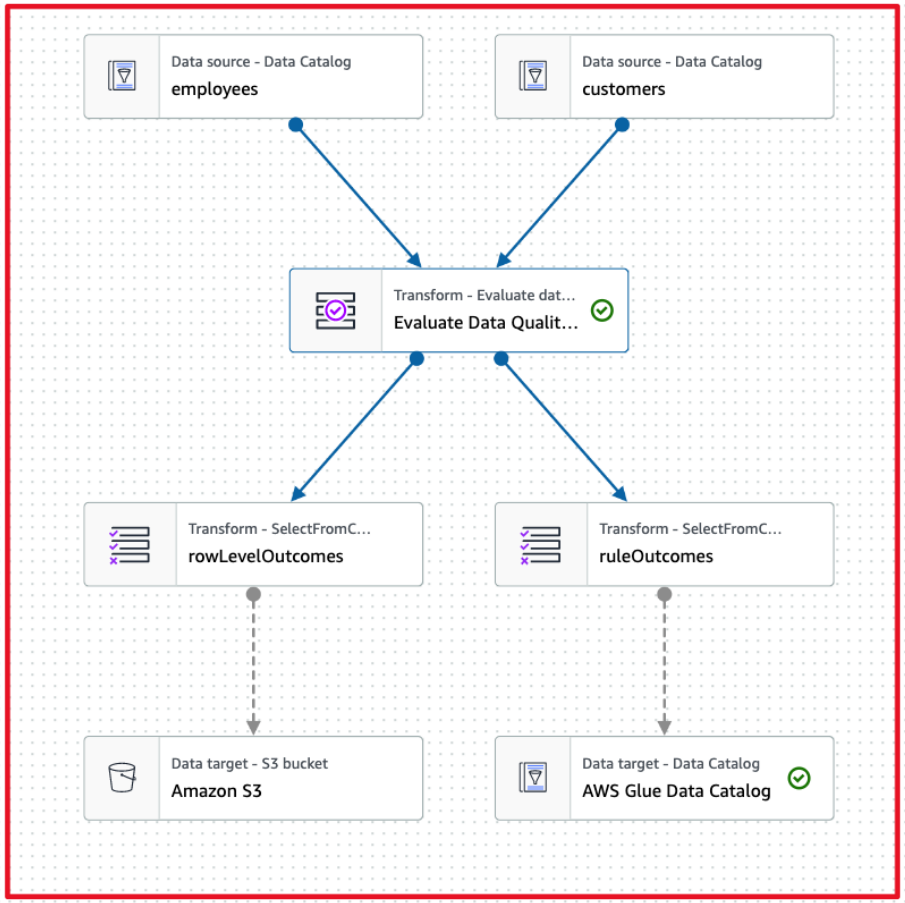
*work done while at Amazon Research

Proceedings of the VLDB Endowment, Vol. 11, No. 12
Copyright 2018 VLDB Endowment 2150-8097/18/8.
DOI: https://doi.org/10.14778/3229863.3229867

Visual | Script | Job details | Runs | Data quality New | Schedules | Version Control

Hooks into a Query Plan

(can be optimized)



Transform | Output schema | Data preview

Name

Evaluate Data Quality (Multiframe)

Node parents

Choose which nodes will provide inputs for this one.

Choose one or more parent node

employees
Catalog - DataSource

customers
Catalog - DataSource

▼ Aliases for referenced data sources

Input sources	Aliases	Primary source
employees	*Primary*	☑
customers	customers	☐

Helper

Rule types | Schema

Search rule types

- **AggregateMatch** column rule
- **ColumnCorrelation** column rule
- **ColumnCount** table rule
- **ColumnDataType** column rule
- **ColumnExists** column rule

```
1 Rules = [
2     ReferentialIntegrity "employeenumber" "customers
          .salesRepEmployeeNumber"between 0.6 and 0.7,
3     RowCount > 1000,
4     CustomSql "select count(*) from primary" between 10 and 200
5 ]
```

dqdl | Ln 1, Col 1 | Errors: 0 | Warnings: 0

DQDL – an OSS query language (ANTLR) for Data Quality

Data quality transform output Info

Choose data to output from your data quality node.

☑ Original data

Choose to output original input data. This option is ideal if you want to stop the job when quality issues are detected.

PROS

- Not just a science project, a well designed SDK built over years.
- Has a DSL that is actually good
- Works for Big Data
- Extremely well-designed API with SOLID principles and overloads
- Exposes rich telemetry and checks that goes right into the data lake as a table that you can anomaly on

CONS

- Only works with Spark DataFrames, it's tightly coupled

SUMMARY

- Non-restrictive OSS license, they accept PRs from anyone
- Good, sustainable business model, OSS the code, PaaS-ify the managed service only in AWS Glue


Local debugging