Lessons Learned From Managing Thousands of Apache Spark Clusters at Scale

Josh Rosen and Henry Davidge

About Us

- Josh
  - Apache Spark Committer; contributing since 2012
- Henry
  - Software engineer on cluster management team
  - BS Yale 2014
- Both love data + Spark

About Databricks

TEAM

Started Spark project (now Apache Spark) at UC Berkeley in 2009

MISSION

Making Big Data Simple

PRODUCT

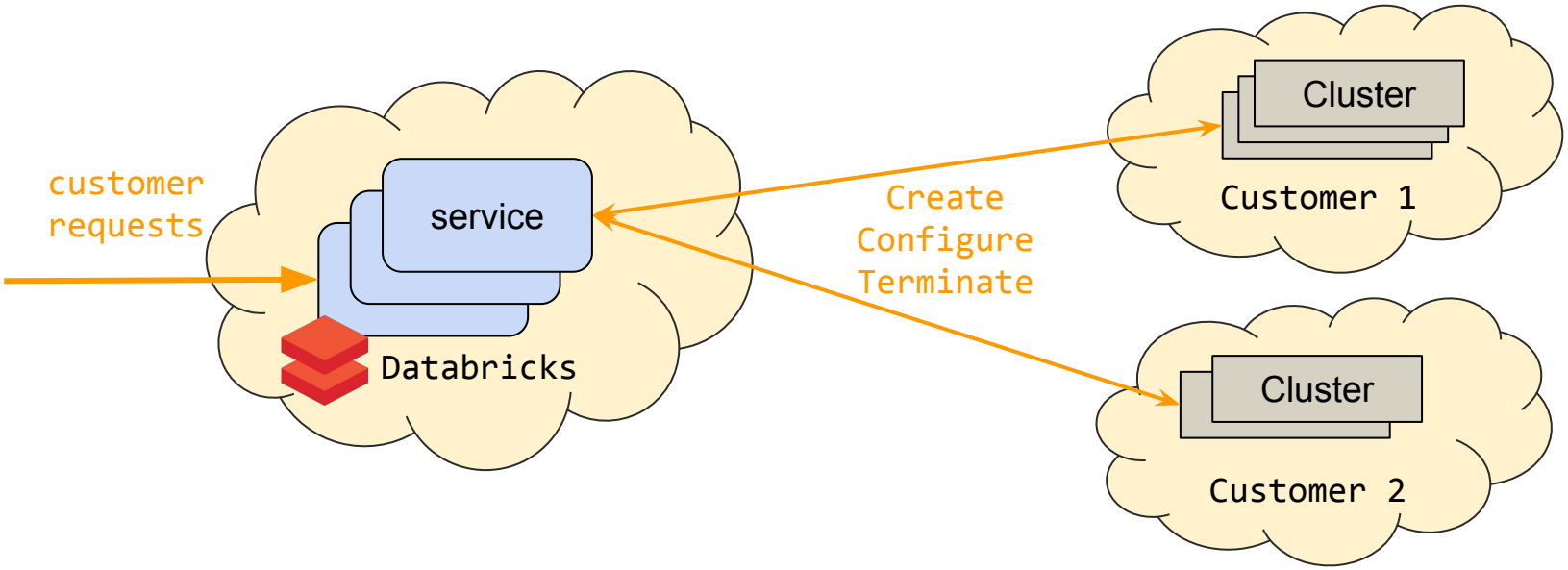
Unified Analytics Platform

Monitoring challenges at Databricks

- Databricks is an inherently complex system
  - Many Spark clusters, configurations, and versions
  - Customers can run arbitrary code on their clusters
  - Deep integration with cloud providers
- Scale
  - 300k+ metrics/min from Databricks
  - 2M+ metrics/min from customers
  - 200+ MB/second of logs

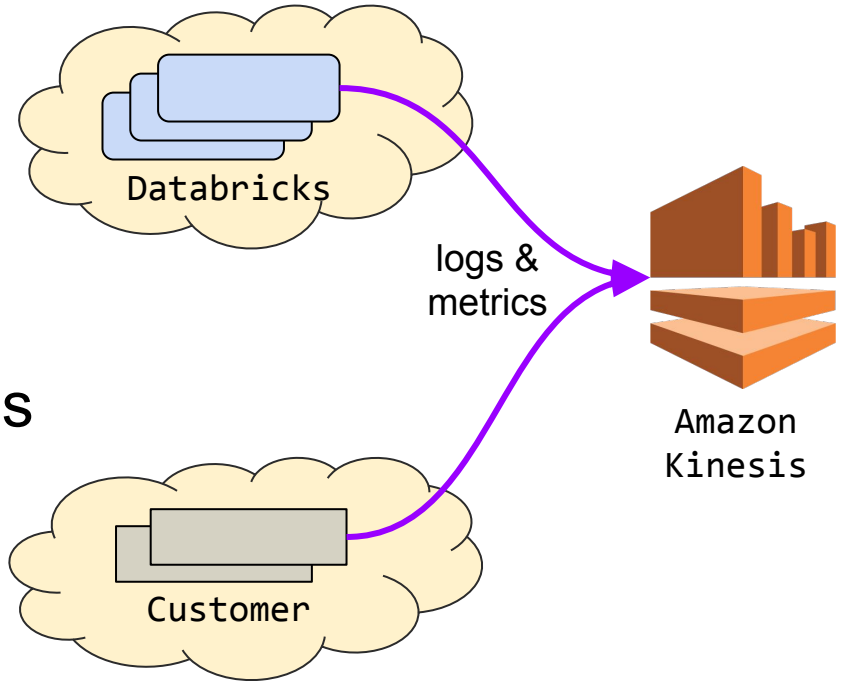
Background: Databricks Architecture

- Control plane in our AWS account
- Spark clusters in customer' AWS accounts



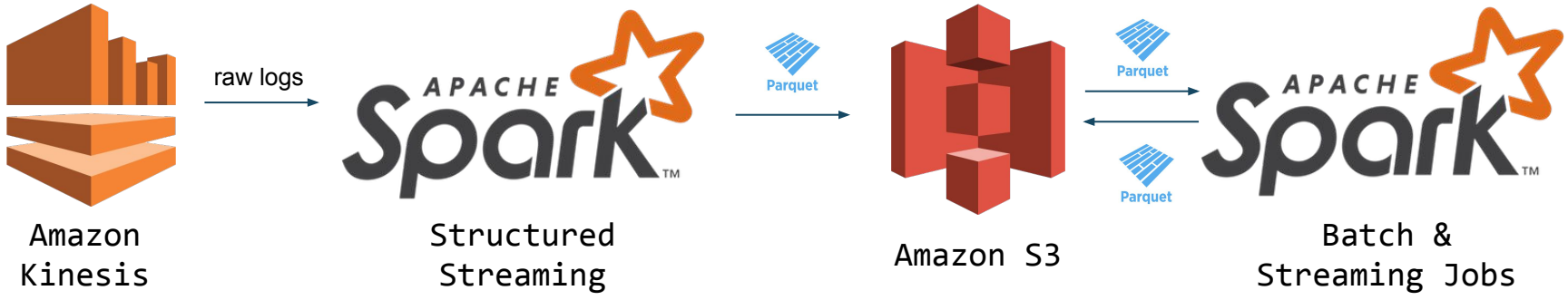
Background: Data pipeline

- Services output three streams
  - Fast path metrics
  - Rich structured logs
  - Unstructured service logs
- Everything to Kinesis



Background: Data pipeline

- Structured streaming job reads raw logs from Kinesis and saves as Parquet in S3
- Batch and streaming jobs perform additional processing on S3 data and output additional Parquet



Story 1: Tracking AWS anomalies

Stuff happens

- Many failure modes possible during provisioning
  - Limits
  - Environment issues
  - Bugs
- Want to catch the bugs!
  - Without many false positives

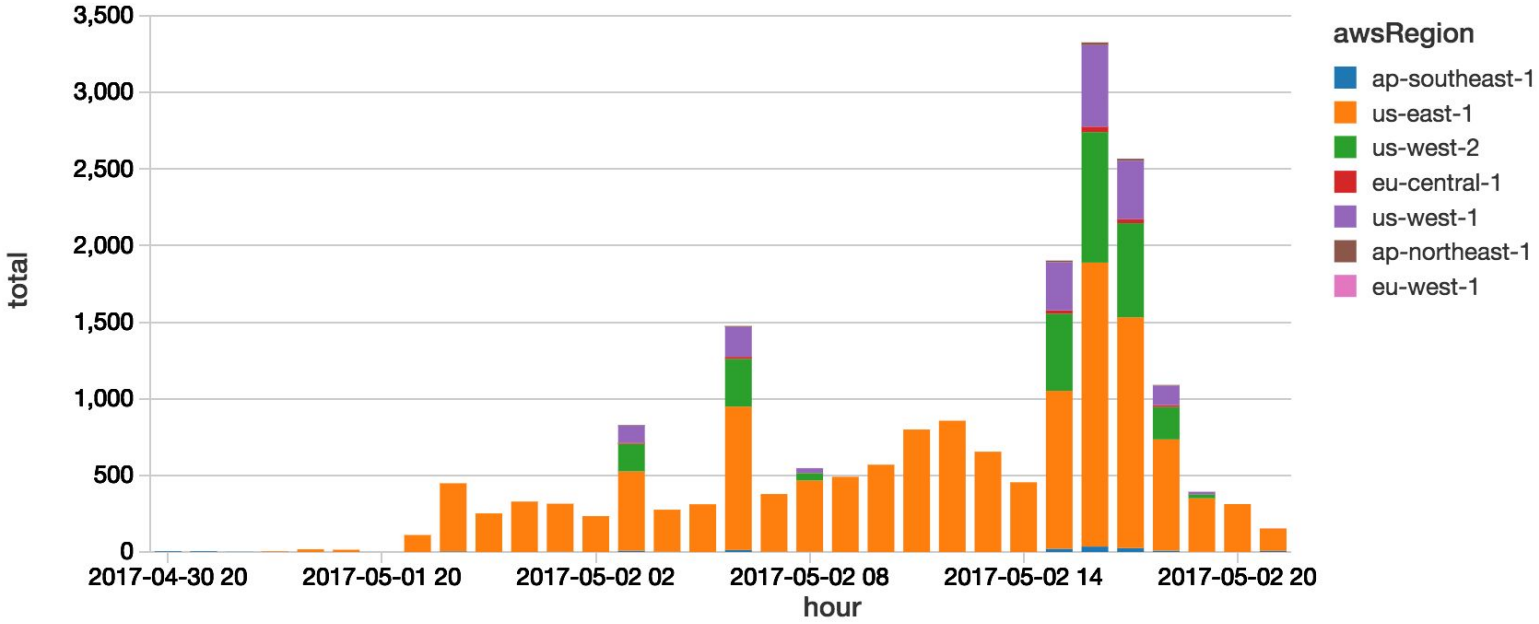
Observing failures

```
{
  “instance-id”: STRING,
  “api-error-code”: STRING, // Request rejected
  “instance-state-reason”: STRING, // Terminated after launch
  “instance-status”: STRING, // Health status from AWS
  “customer-metadata”: OBJECT
}
```

Analyzing failures

- Structured streaming query reads from file source
- Strip known error patterns (limits, nonexistent resource)
- If new error type is seen, low prio alert
  - If seen by multiple customers, page on call

One day in May...



One day in May...

- AWS bug caused Spot requests to be rejected
- Alert allowed us to identify and notify customers
- Informed AWS of issue, they patched
- Our Spark cluster helped yours!

Story 2: Discovering bugs from unstructured logs

Goal: monitor Spark logs for errors

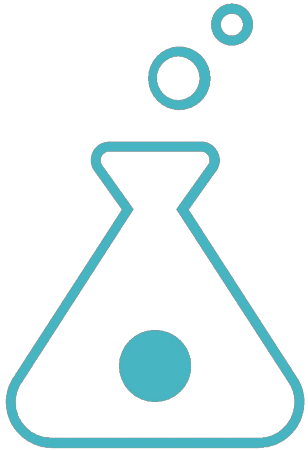
- Search Spark logs for error messages to discover issues and determine their scope/impact:
  - Which customers are impacted by an error?
  - How frequently is that error occurring?
  - Does the error only affect certain versions of Spark?
  - Are there long-term trends in the data?

Challenge: false-positives

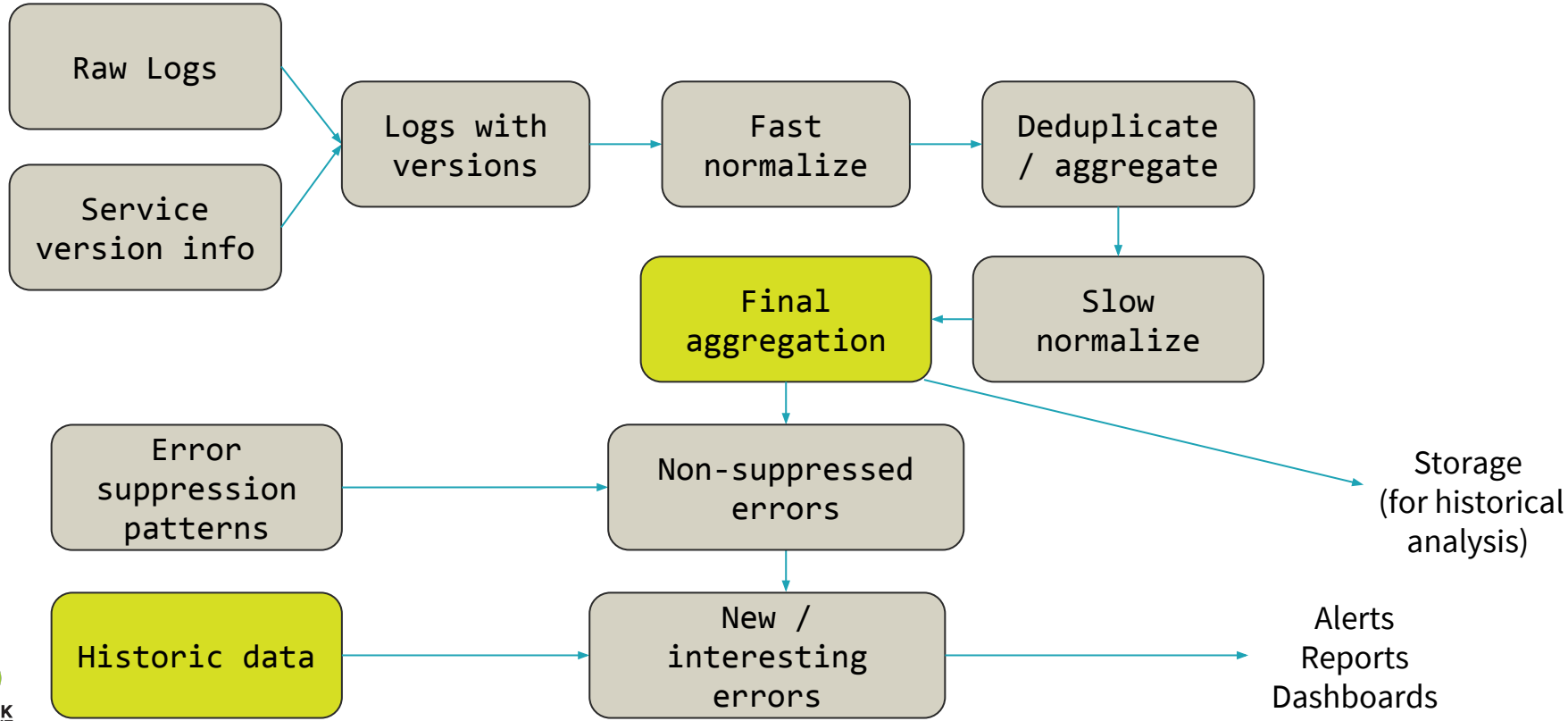
- Errors may be fixed in newer Spark versions but continue to occur in old versions
- Raw logs can be very messy: many near-duplicate errors due to variables being included in log messages

Solution: normalize, deduplicate & filter

- **Normalize:** replace constants in logs (numbers, IP addresses, customer names) with placeholders
- **Deduplicate:** Store (count, version, set(customers), example) instead of raw logs
- **Filter:** Use patterns to (conditionally) ignore known errors or to surface only new errors (errors that appeared for the first time)



Pipeline overview



Example 1: SPARK-19691

- ```
  spark.range(10)
    .selectExpr("cast (id as decimal) as x")
    .selectExpr("percentile(x, 0.5)")
    .collect()
  ```

- Failed with `java.lang.ClassCastException: org.apache.spark.sql.types.Decimal cannot be cast to java.lang.Number`
  ```

Role in QA

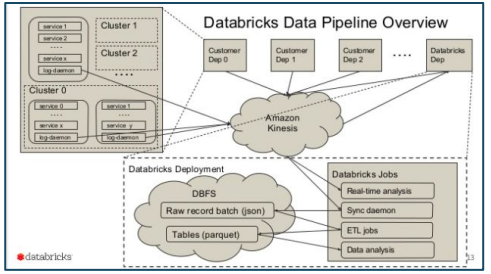
- Proactively detect bugs in unreleased versions:
  - Compare error profiles between staging and production environments

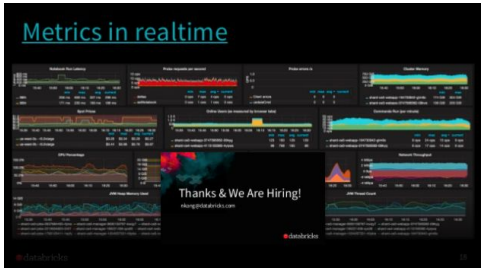
Lessons Learned

- Structure proactively
  - Much easier to implement at logging time
- Strive for small data
  - Normalize, deduplicate, find patterns
- Alert at multiple urgency levels

For more details

- **Log collection pipeline:**
  - “A Journey into Databricks' Pipelines: Journey and Lessons Learned”
  - Somewhat out of date, but thorough
- **Structured metrics pipeline:**
  - “Monitoring Large-Scale Spark Clusters at Databricks”
  - Kinesis + Prometheus + Grafana
- **Error log analysis:**
  - “Monitoring error logs at Databricks”





SPARK SUMMIT 2017

Try Apache Spark in Databricks!

UNIFIED ANALYTICS PLATFORM

- Collaborative cloud environment
- Free version (community edition)

DATABRICKS RUNTIME 3.0

- Apache Spark - optimized for the cloud
- Caching and optimization layer - DBIO
- Enterprise security - DBES

databricks.com

Thank You.

joshrosen
hhd @databricks.com