Effective Data Pipelines: Data Management from Chaos

Katharine Jarmul (@kjam)

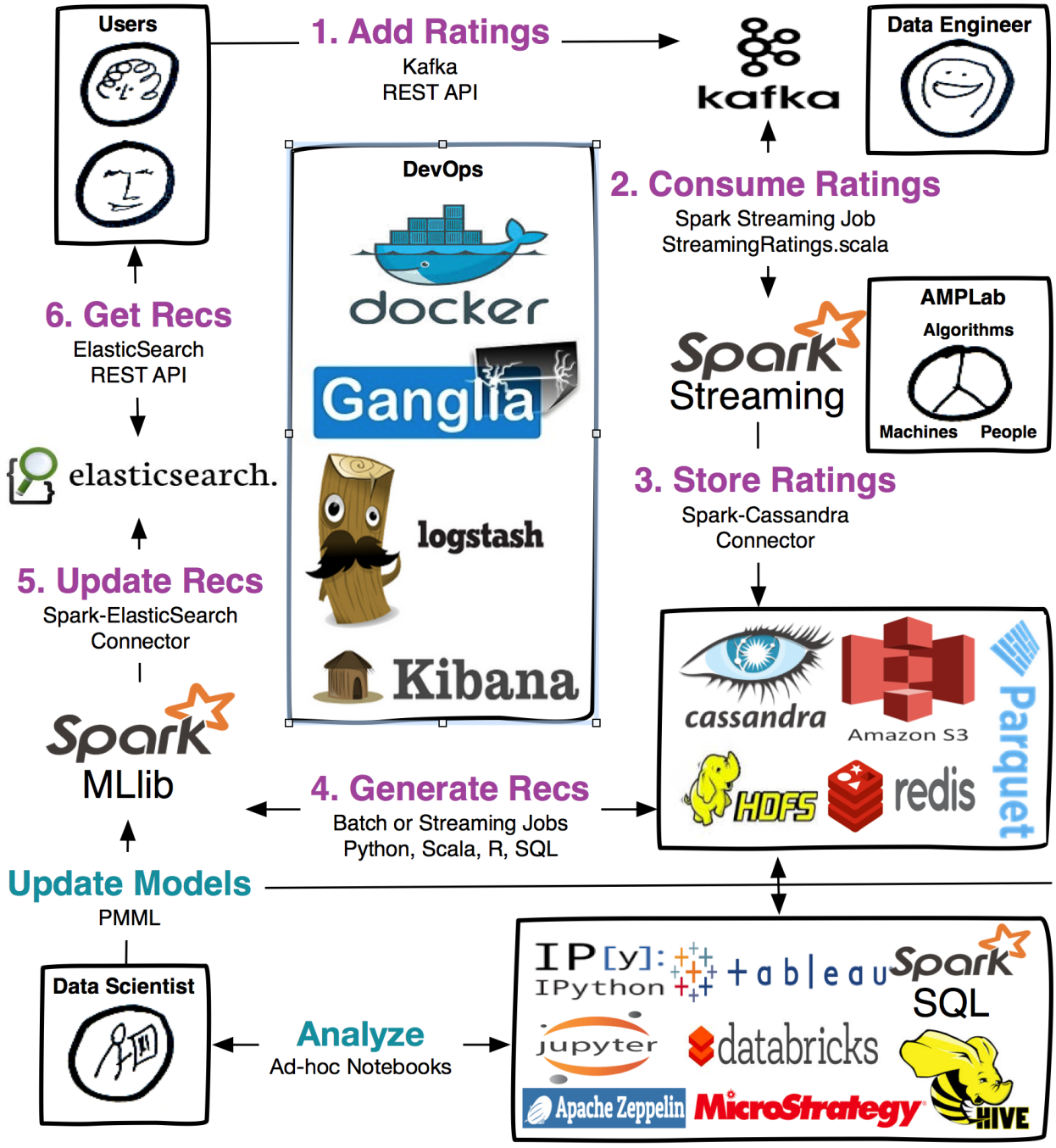
QCon - London - March 6, 2017

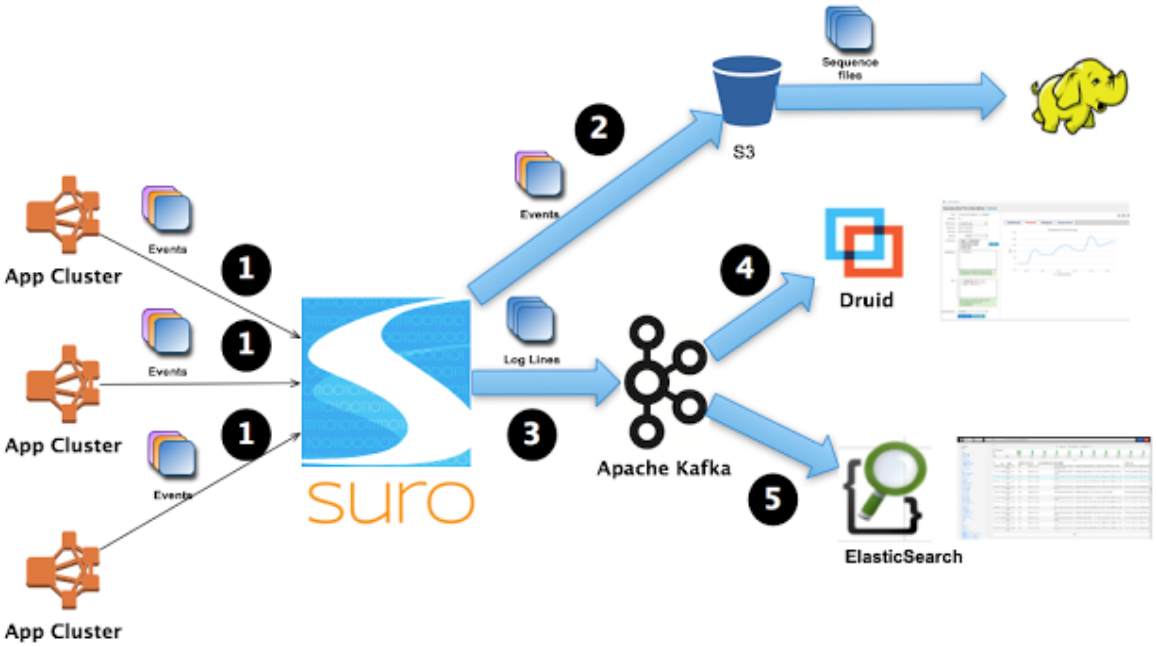
About Katharine

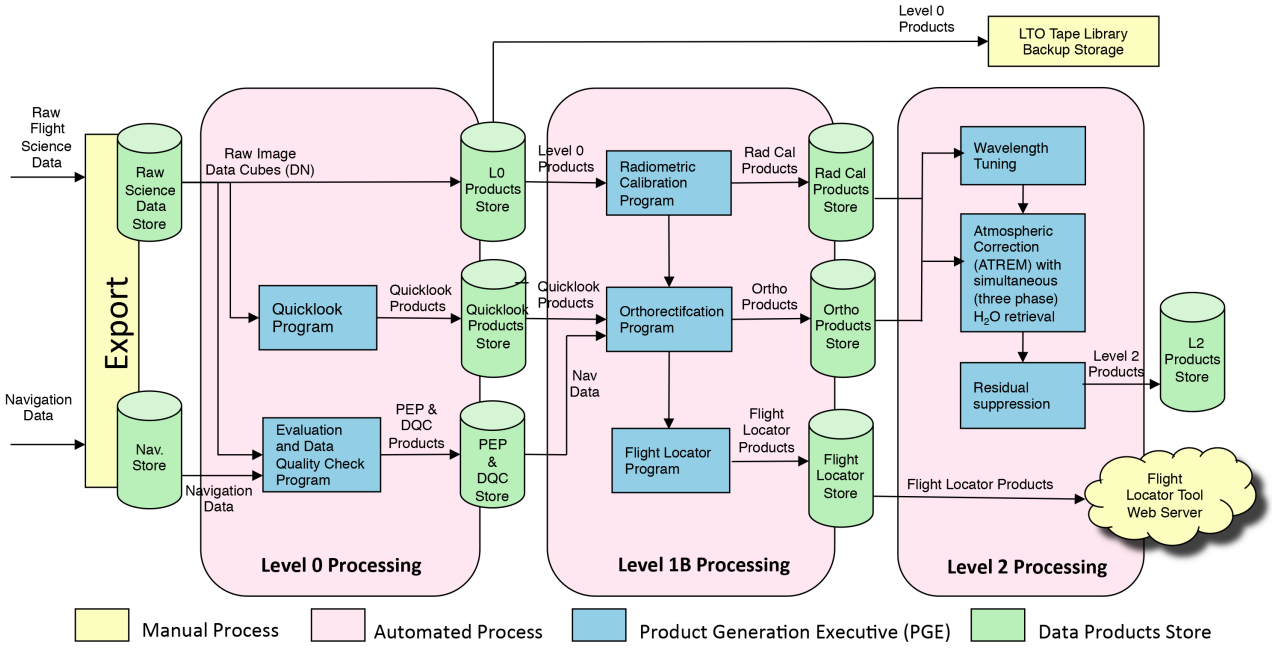
Data Scientist, Engineer, Author, Pythonista

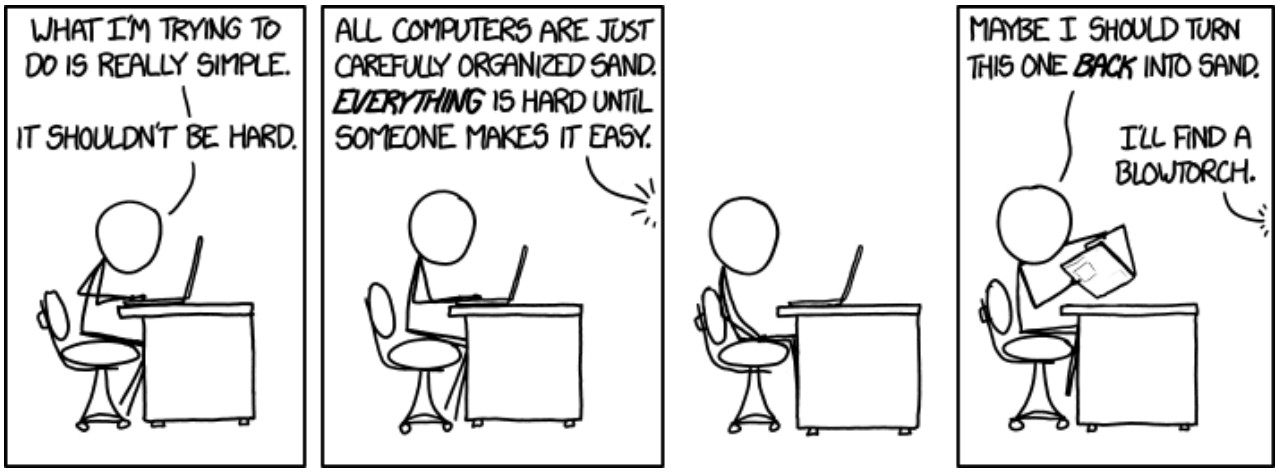
Founder @ kjamistan UG: data science consulting & engineering

Find me at: kjamistan.com - katharine@kjamistan.com - @kjam


Users
1. Add Ratings
Kafka
REST API
kafka
Data Engineer
DevOps
docker
Ganglia
logstash
Kibana
2. Consume Ratings
Spark Streaming Job
StreamingRatings.scala
Spark
Streaming
AMPLab
Algorithms
Machines
People
3. Store Ratings
Spark-Cassandra
Connector
cassandra
Amazon S3
Parquet
HDFS
redis
6. Get Recs
ElasticSearch
REST API
elasticsearch.
5. Update Recs
Spark-ElasticSearch
Connector
Spark
MLlib
4. Generate Recs
Batch or Streaming Jobs
Python, Scala, R, SQL
Update Models
PMML
Data Scientist
Analyze
Ad-hoc Notebooks
IPython
tableau
Spark
SQL
jupyter
databricks
Apache Zeppelin
MicroStrategy
HIVE


App Cluster
App Cluster
App Cluster
Events
Events
Events
1
1
1
suro
2
Events
S3
Sequence files
3
Log Lines
Apache Kafka
4
Druid
5
ElasticSearch


Level 0 Products
LTO Tape Library Backup Storage
Raw Flight Science Data
Raw Science Data Store
Raw Image Data Cubes (DN)
L0 Products Store
Level 0 Products
Radiometric Calibration Program
Rad Cal Products
Rad Cal Products Store
Wavelength Tuning
Atmospheric Correction (ATREM) with simultaneous (three phase) H_2O retrieval
Export
Quicklook Program
Quicklook Products
Quicklook Products Store
Quicklook Products
Orthorectifcation Program
Ortho Products
Ortho Products Store
Nav Data
Residual suppression
Level 2 Products
L2 Products Store
Navigation Data
Nav. Store
Evaluation and Data Quality Check Program
PEP & DQC Products
PEP & DQC Store
Flight Locator Program
Flight Locator Products
Flight Locator Store
Navigation Data
Flight Locator Products
Flight Locator Tool Web Server
Level 0 Processing
Level 1B Processing
Level 2 Processing
Manual Process
Automated Process
Product Generation Executive (PGE)
Data Products Store



Three Questions when Building Data Workflows

1. Who is the producer? Who is the consumer?

2. Where, What, When is the data?

3. What are the constraints? When might they change?

(sorry, that was more like seven.)

Three Tips when Building Data Pipelines

1. Premature [architecture | optimization | infrastructure] is a bad idea.

2. Untested == Unreliable

3. Security today, not tomorrow.

Three Practical Steps for Pipelines

1. Automate the easy stuff, testing and deployment. Slowly automate the difficult things.

2. It *is* infrastructure. Treat it as such.

3. Monitoring, alerting and debugging are meaningless without a chain of responsibility.

Qualities of an Ideal Data Pipeline

- Idempotent with State Handling

-- You *will* need to interrupt and rerun tasks (due to bugs, upstream errors, data validation issues).

-- State management is a core part of most pipeline / streaming frameworks. When you can, rely on the framework to do it.

Qualities of an Ideal Data Pipeline

- Scalable and Resilient

-- You may face bursty periods and slow ones. Is autoscaling or provisioning an option?

-- The fallacies of distributed computing often apply to pipelines.

Qualities of an Ideal Data Pipeline

- Replacable or Programmable

-- It's very difficult to forsee where and how your pipeline might grow and change. Be adaptable.

-- Open-source or clear programmability allows for transparent and easy additions.

Qualities of an Ideal Data Pipeline

- Testable and Traceable

-- Upstream, instream, downstream bugs will happen. Make them

easier to find.

-- Find good ways to mock, mirror and replay production data for integration and regression testing.

Qualities of an Ideal Data Pipeline

- Documented and Automated

-- A pipeline without proper documentation is legacy code.

-- Use automated deploys with continuous integration.

Qualities of an Ideal Data Pipeline

- Idempotent with State Handling

- Scalable and Resilient

- Replacable or Programmable

- Testable and Traceable

- Documented and Automated

Pipeline Testaments

- My pipeline is easy to test, debug and monitor.

- There are clear solutions for replaying, rerunning and interrupting tasks or dataflow in my pipeline.

- There are several teams involved in my pipeline (for security, maintainability and development); however, there is a clear chain of responsiblity and protocol for when things go wrong.

- We have reviewed business and stakeholder use cases. We chose a pipeline structure fitting our current constraints with a straightforward path for growth and change.

Thank you for listening!

Questions?

Now?

Later? @kjam / katharine@kjamistan.com

Want to talk about pipelines? Data unit testing? Data wrangling? (come find me!)