# Google Cloud Platform for Data Science teams

**Barton Rhodes [@bmorphism]**
Senior Data Scientist at Pandata [pandata.co]

# Why Cloud?

Be it AWS, GCP, or Azure, having a well-integrated 'productionizing' workflow brings these benefits to data scientists:

- escape limitations of a single machine
- reproducible infrastructure (e.g. Terraform)
- scale up on demand
- use highly specialized products and workflows (no need to set up full systems)
- lessen DevOps / data engineering burden
- leaner than a packaged product (e.g. Domino Data Lab, databricks)

# Why Google Cloud?

Our reasons:

- relatively good UX and clean APIs
- embraces community tools (e.g. Apache Beam, Jupyter)
- integrates with Google tools (TensorFlow, Kubernetes)
- price (especially storage) and per-minute billing
- HIPAA-compliant infrastructure and ISO security certifications (important for a consultancy)

# Google Cloud Platform for Data Science

BigQuery

Dataflow

Dataprep

Machine Learning

Video Intelligence API

...

Dataproc

Datalab

Container Engine

Vision API

Natural Language API

# Focus of this talk - Data Engineering

BigQuery

Dataflow

Dataprep

Machine Learning

Video Intelligence API

...

Dataproc

Datalab

Container Engine

Vision API

Natural Language API

# GCP Documentation

Documentation available at:

https://cloud.google.com/docs/

Active certification program:

Google

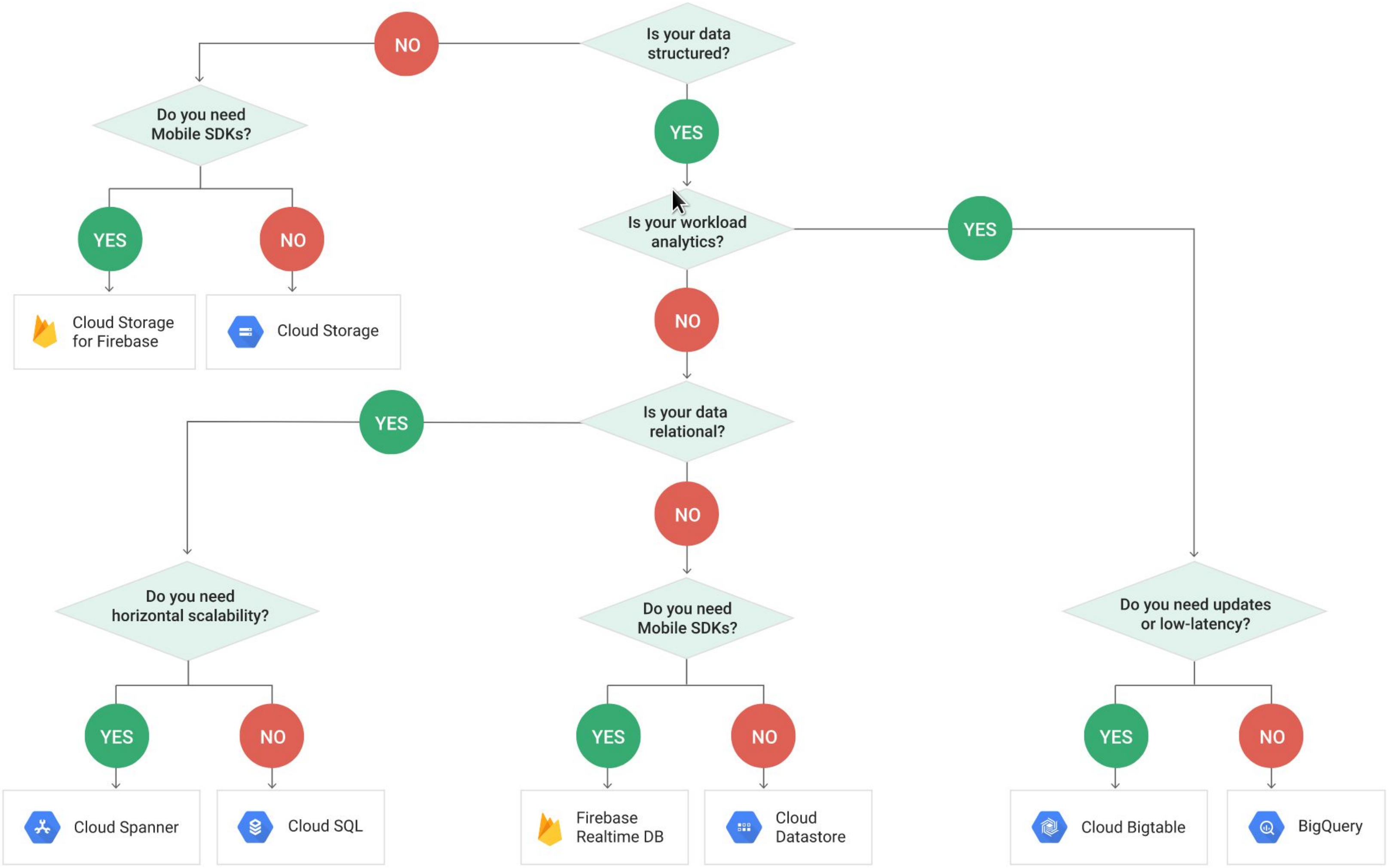
Certified Professional

Data Engineer

Coursera specialization:

Google Cloud Platform

coursera


Is your data structured?

NO

Do you need Mobile SDKs?

YES

Cloud Storage for Firebase

NO

Cloud Storage

YES

Is your workload analytics?

YES

Do you need updates or low-latency?

YES

Cloud Bigtable

NO

BigQuery

NO

Is your data relational?

YES

Do you need horizontal scalability?

YES

Cloud Spanner

NO

Cloud SQL

NO

Do you need Mobile SDKs?

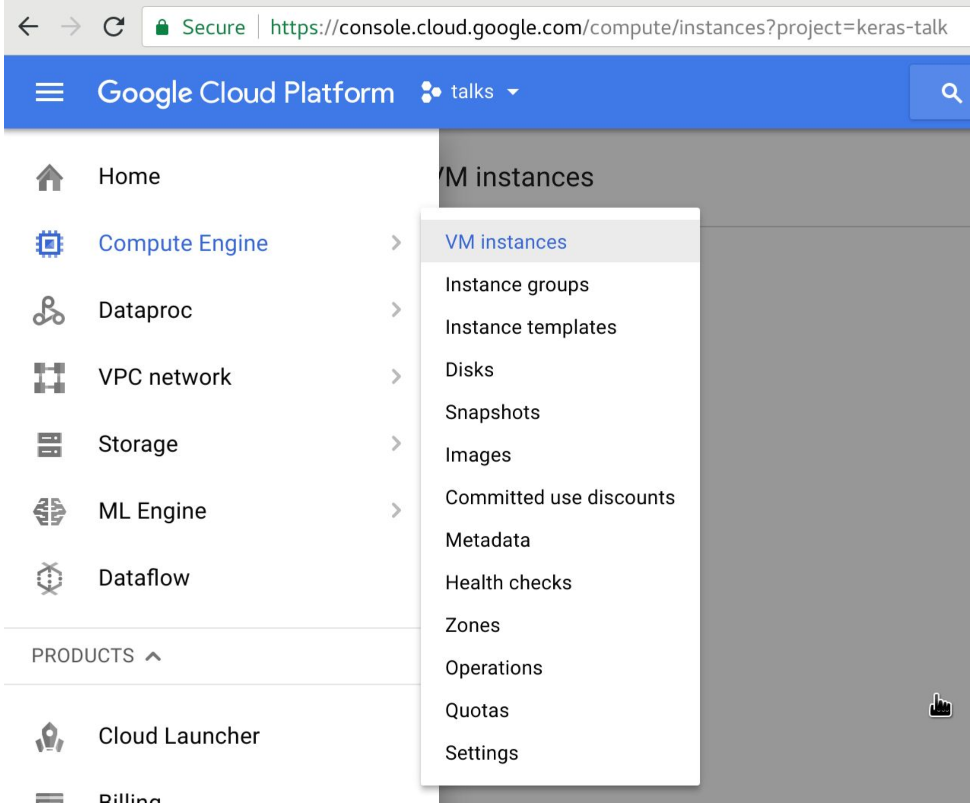
YES

Firebase Realtime DB

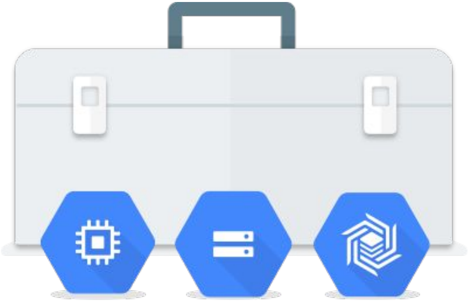
NO

Cloud Datastore

# GCP Console


Secure
https://console.cloud.google.com/compute/instances?project=keras-talk
Google Cloud Platform
talks
Home
Compute Engine
Dataproc
VPC network
Storage
ML Engine
Dataflow
PRODUCTS
Cloud Launcher
Billing
VM instances
VM instances
Instance groups
Instance templates
Disks
Snapshots
Images
Committed use discounts
Metadata
Health checks
Zones
Operations
Quotas
Settings

# Google Cloud SDK



A set of command line tools to manage GCP:

**gcloud** - general API interaction, auth, configuration

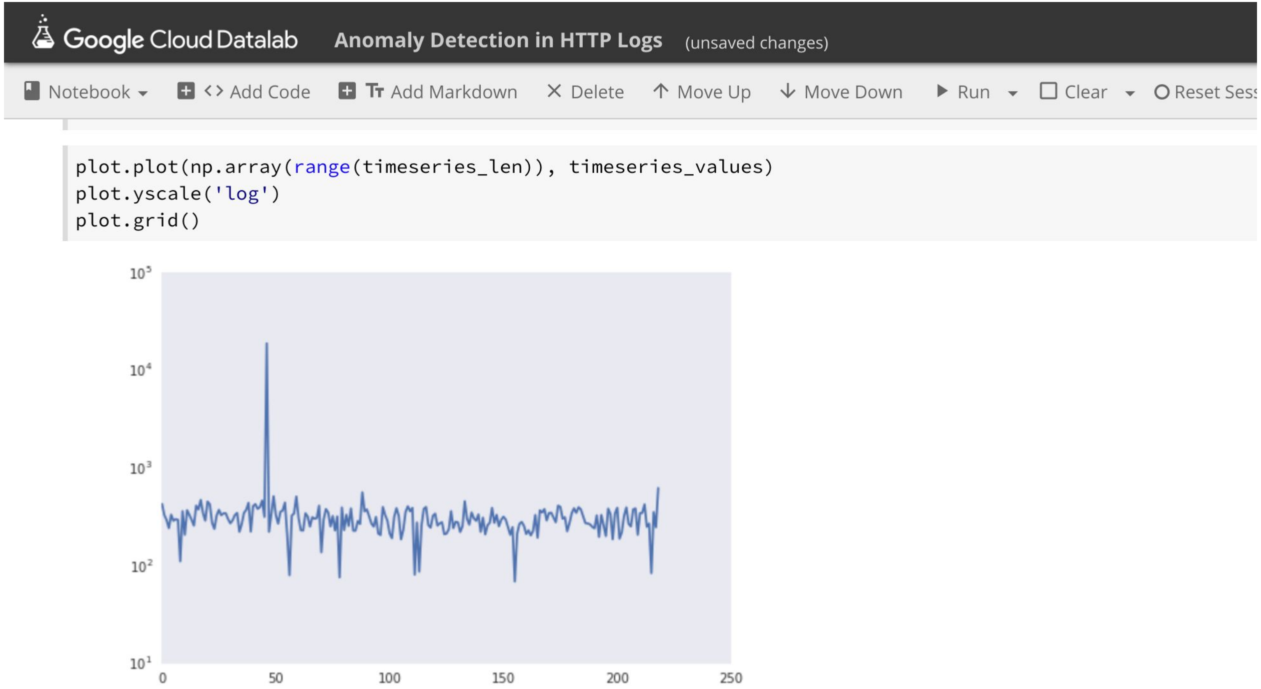
**gsutil** - manage Google Storage

**bq** - BigQuery queries and management

**kubectl** - manage containers in Kubernetes

Client libraries for **Python**, **Java**, **NodeJS**, and others!

# Datalab (aka Jupyter)



```
plot.plot(np.array(range(timeseries_len)), timeseries_values)
plot.yscale('log')
plot.grid()
```

- interactive data science in modified Jupyter
- runs inside of a Docker container
- integrates with the rest of GCP
- alas, single user
- alas, only Python by default

# BigQuery

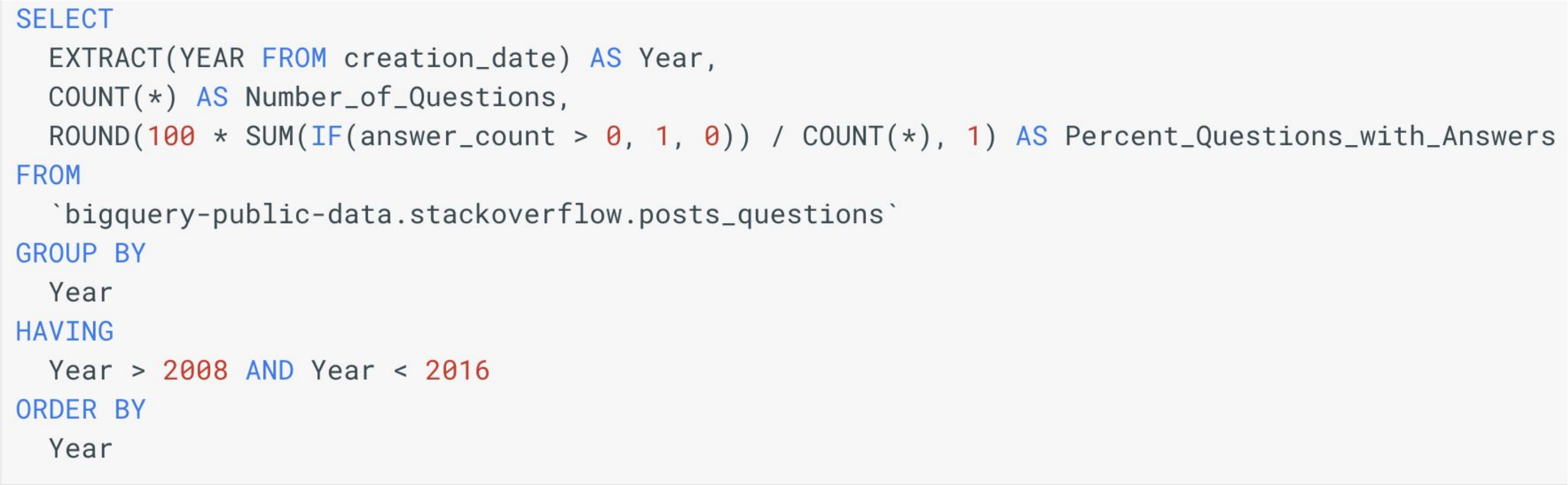
Massively parallel query engine with SQL-like query language compatible with SQL 2011 standard.

- acts as a data warehouse at Google scale
- columnar data storage
- lower cost than other forms of storage
- BigQuery Slots to guarantee resources
- availability of public datasets
- no UPDATE / DELETE on existing data

Tableau can tap directly into BigQuery, making it easy to perform BI-type tasks over large datasets quickly.

# BigQuery example

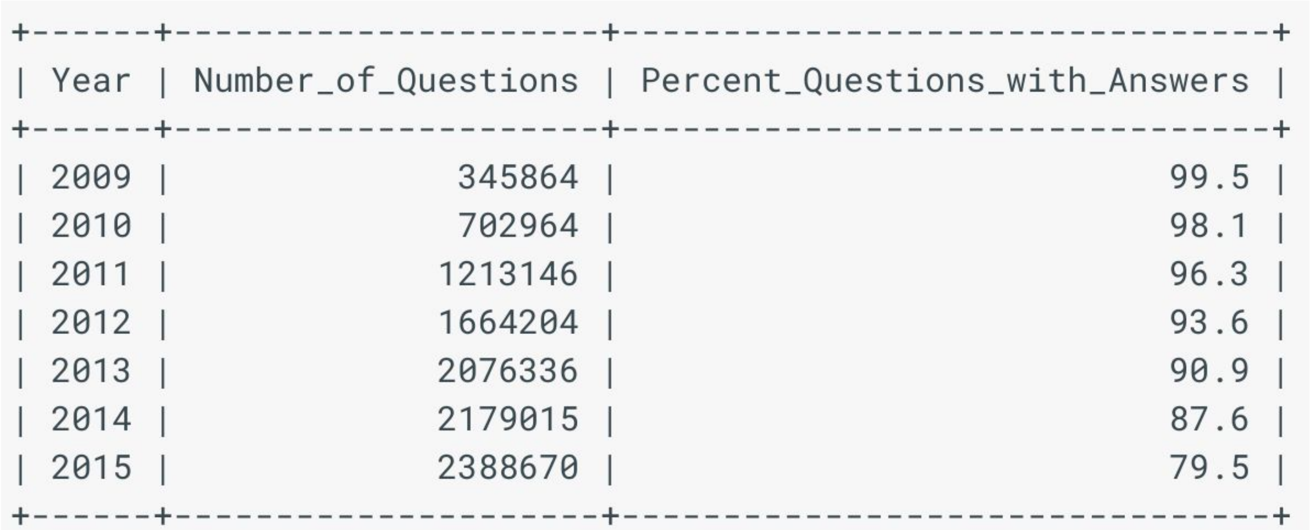
In this example, one can use the publicly available StackOverflow data (~**150GiB**) to calculate % answered questions over the years:



```sql
SELECT
  EXTRACT(YEAR FROM creation_date) AS Year,
  COUNT(*) AS Number_of_Questions,
  ROUND(100 * SUM(IF(answer_count > 0, 1, 0)) / COUNT(*), 1) AS Percent_Questions_with_Answers
FROM
  `bigquery-public-data.stackoverflow.posts_questions`
GROUP BY
  Year
HAVING
  Year > 2008 AND Year < 2016
ORDER BY
  Year
```

# BigQuery example (continued)

Output:



```
+------+---------------------+----------------------------------+
| Year | Number_of_Questions | Percent_Questions_with_Answers |
+------+---------------------+----------------------------------+
| 2009 |              345864 |                             99.5 |
| 2010 |              702964 |                             98.1 |
| 2011 |             1213146 |                             96.3 |
| 2012 |             1664204 |                             93.6 |
| 2013 |             2076336 |                             90.9 |
| 2014 |             2179015 |                             87.6 |
| 2015 |             2388670 |                             79.5 |
+------+---------------------+----------------------------------+
```

# Dataproc

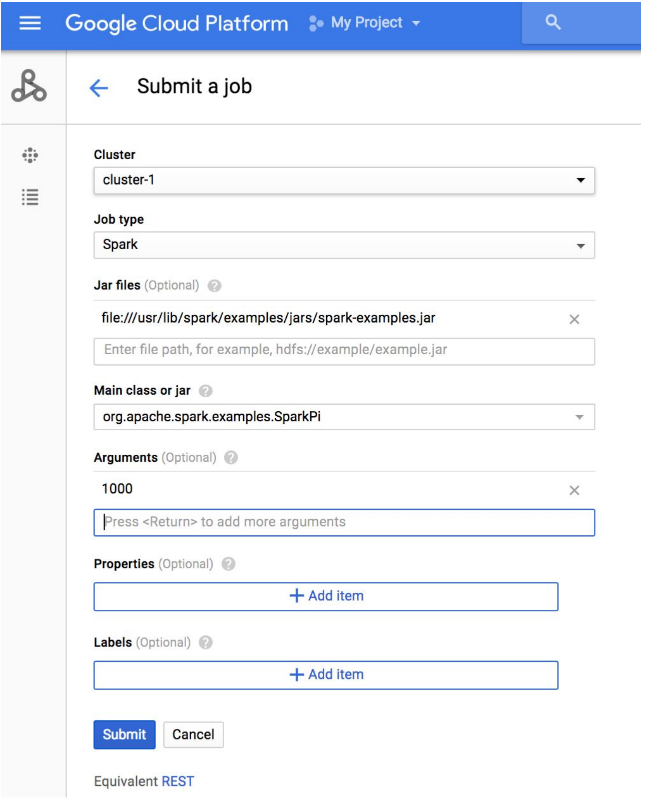
Run Apache Hadoop and Apache Spark clusters in the Cloud.

- fully managed solution (removes configuration headaches)
- easy to scale quickly through a Web UI / CLI
- can substitute a much faster Google Cloud Storage for HDFS
- can start with a job and provision a cluster appropriately
- cannot customize initial components trivially like one would with Amazon EMR (can use init actions through scripts)
- Apache Zeppelin does not come pre-installed

For sample init actions, see the following repository:

https://github.com/GoogleCloudPlatform/dataproc-initialization-actions

# Dataproc example



Submit a job

Cluster
cluster-1

Job type
Spark

Jar files (Optional)
file:///usr/lib/spark/examples/jars/spark-examples.jar
Enter file path, for example, hdfs://example/example.jar

Main class or jar
org.apache.spark.examples.SparkPi

Arguments (Optional)
1000
Press <Return> to add more arguments

Properties (Optional)

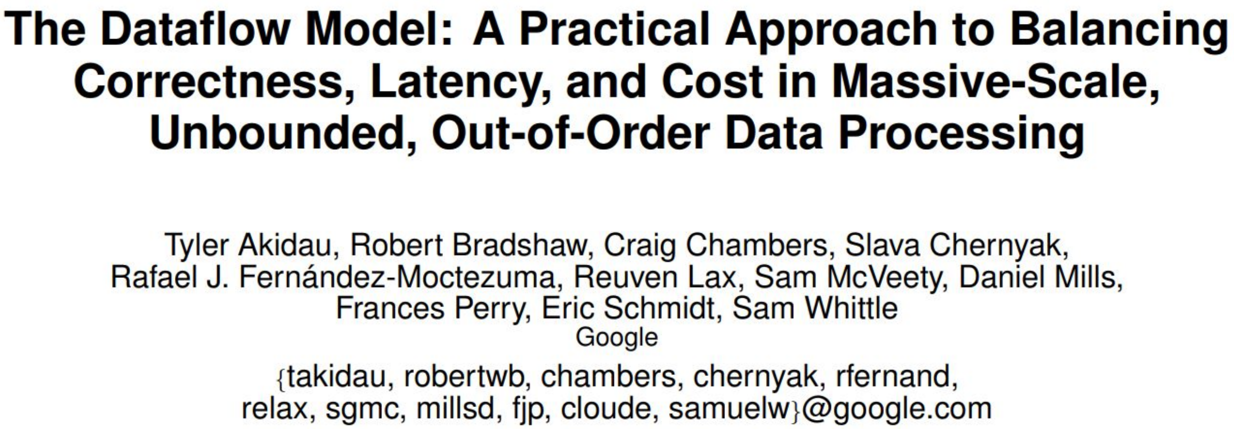
Labels (Optional)

Equivalent REST

# Dataflow (aka Apache Beam)

A runner for Apache Beam (also donated by Google).

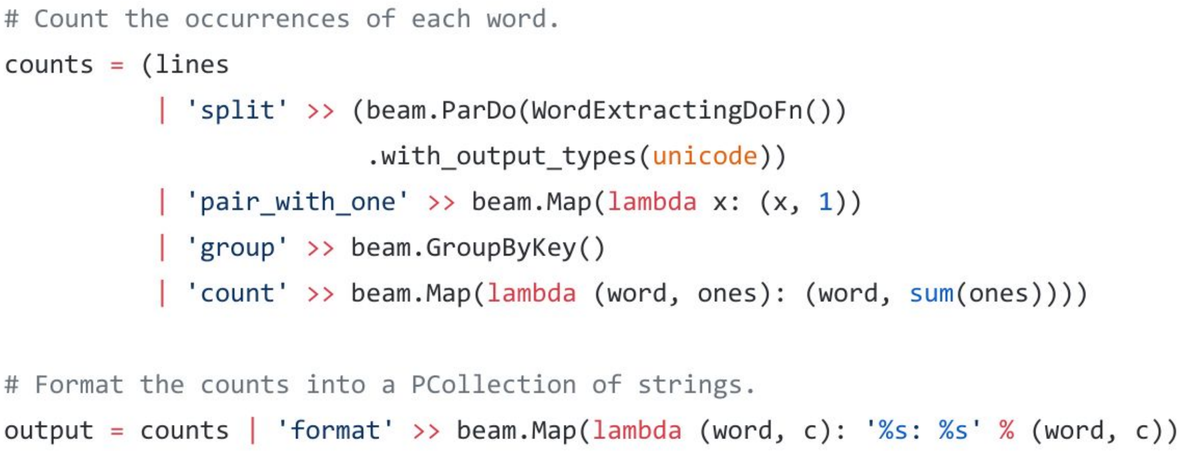
Beam model allows for unified semantics for batch & streaming systems, enabling the coveted write once run everywhere* approach to building data pipelines.



**The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing**

Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak,
Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills,
Frances Perry, Eric Schmidt, Sam Whittle
Google
{takidau, robertwb, chambers, chernyak, rfernand,
relax, sgmc, millsd, fjp, cloude, samuelw}@google.com

# Dataflow example



```
# Count the occurrences of each word.
counts = (lines
          | 'split' >> (beam.ParDo(WordExtractingDoFn())
                        .with_output_types(unicode))
          | 'pair_with_one' >> beam.Map(lambda x: (x, 1))
          | 'group' >> beam.GroupByKey()
          | 'count' >> beam.Map(lambda (word, ones): (word, sum(ones))))

# Format the counts into a PCollection of strings.
output = counts | 'format' >> beam.Map(lambda (word, c): '%s: %s' % (word, c))
```

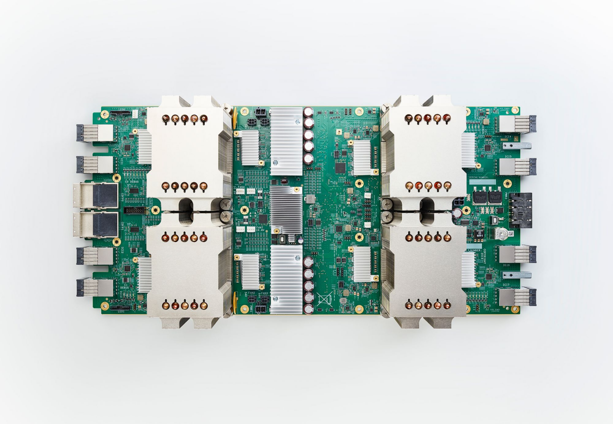
# Machine Learning

- integrates TensorFlow for training and prediction
- automatically provisions training and prediction instances (purpose-built compute instances)
- built-in into Datalab
- operates in batch mode only, streaming support experimental
- still on Python 2.7 (boo!)

Some sample projects:

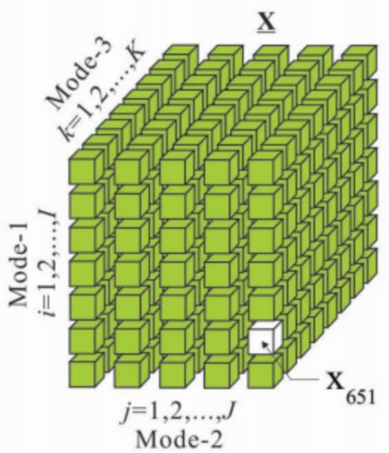
https://github.com/GoogleCloudPlatform/cloudml-samples

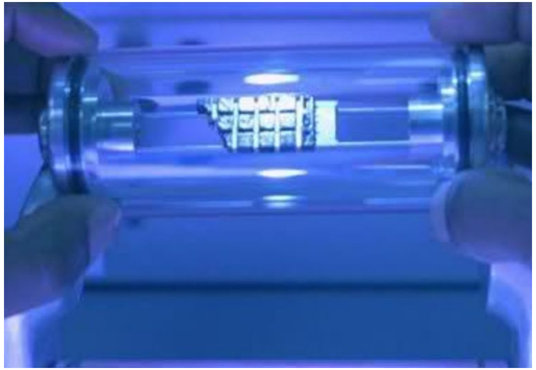
# Cloud TPUs (alpha)



Google-designed integrated circuits highly optimized for reduced precision operations, integration with TensorFlow.

Accelerates AI inference and now also training.





?

Exciting times ahead! 😁

# Bringing it all together (demo)

Adapted from:

https://cloud.google.com/dataproc/docs/tutorials/bigquery-sparkml

In this demo, we will predict birth weight using Linear Regression in Apache Spark (Dataproc) using the publicly available BigQuery natality dataset.