Building Data applications with Go

from Bloom filters to Data pipelines

Sergii Khomenko, Data Scientist
sergii.khomenko@stylight.com, @lc0d3r

Stylight

FOSDEM - January 31, 2016

Sergii Khomenko

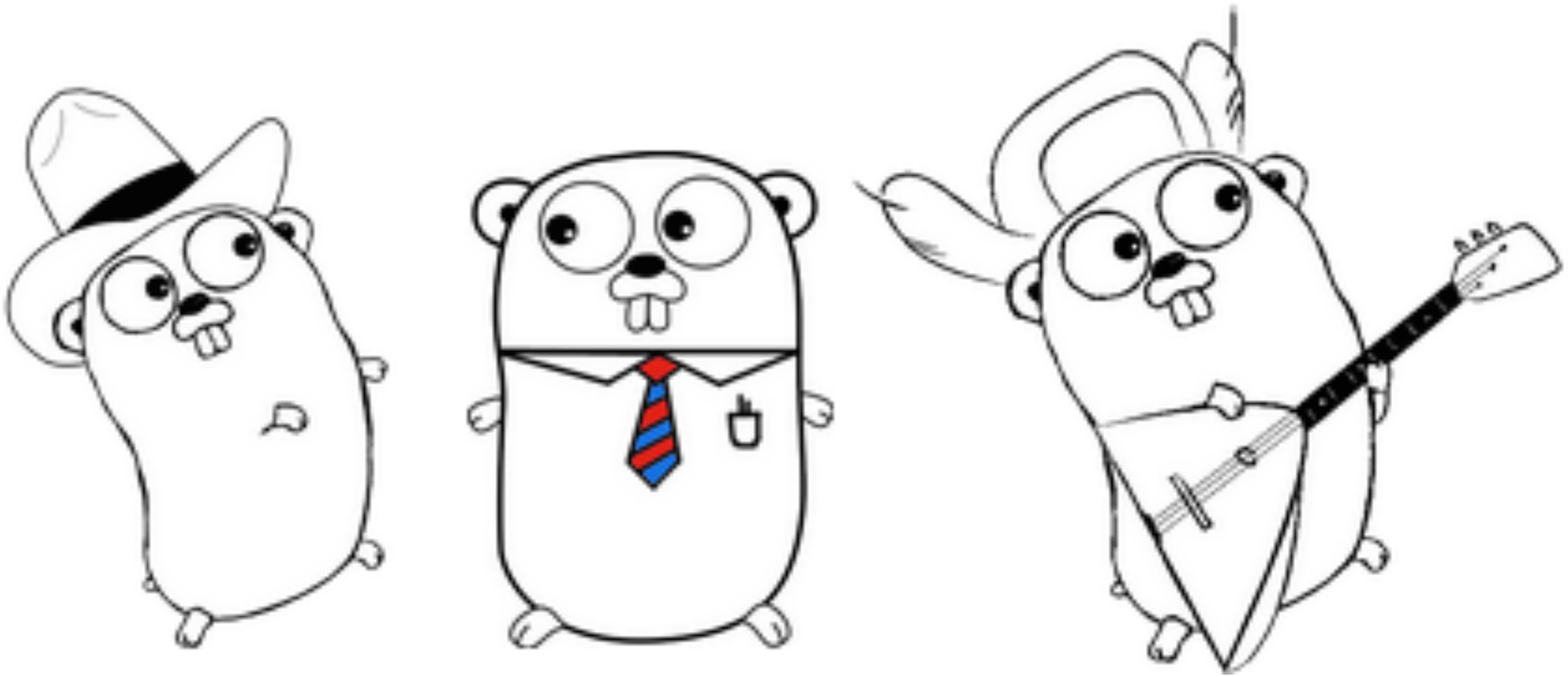
Data scientist at one of the biggest fashion communities, Stylight.

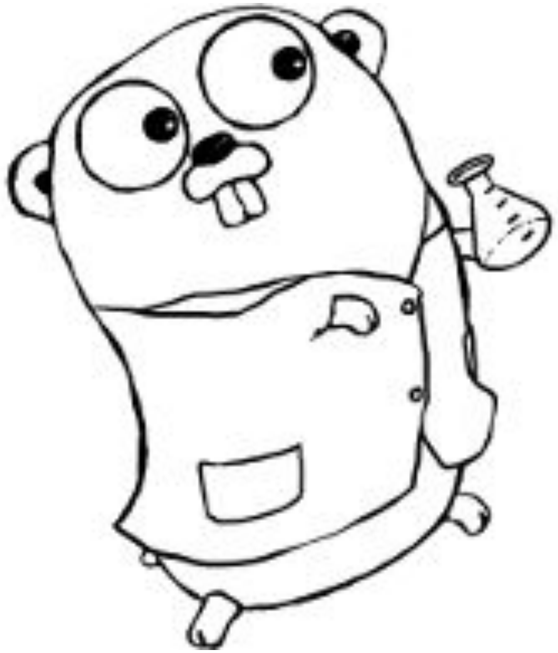
Data analysis and visualisation hobbyist, working on problems not only in working time but in free time for fun and personal data visualisations.

First time faced Golang in ~ 2010. Fell in love with language channels and core concepts.

Speaker at Berlin Buzzwords 2014, ApacheCon Europe 2014, Puppet Camp London, Berlin Buzzwords 2015 , Tableau Conference on Tour, Budapest BI Forum 2015, Crunchsconf 2015 and others

Munich, Germany
Founded on Apr 5, 2014
Gophers: 323





https://www.pinterest.com/pin/38351034303708696/

Stylight – Make Style Happen

Core Target Group

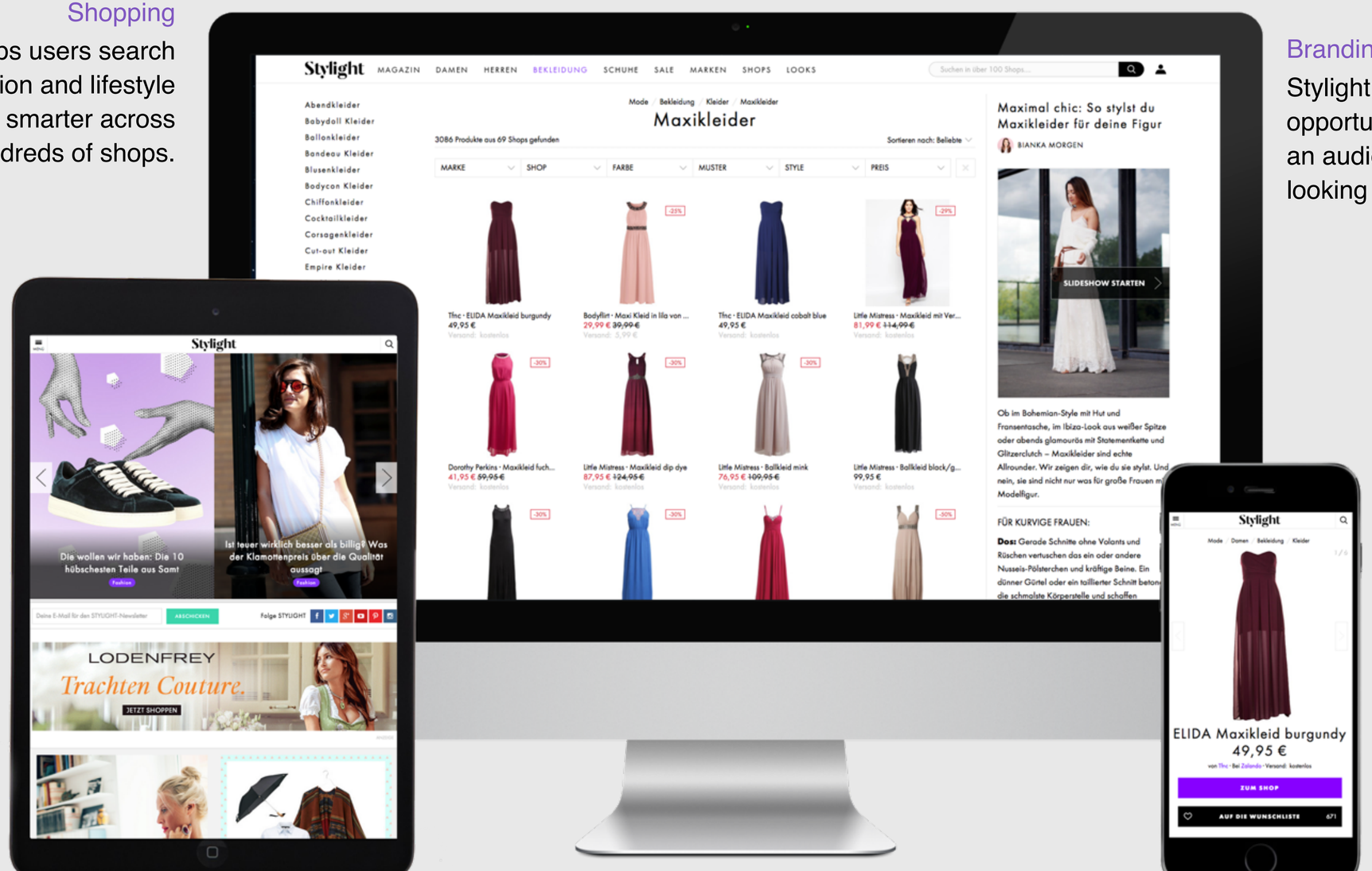
Stylight help aspiring women between 18 and 35 to evolve their style through shoppable inspiration.

Shopping

Stylight helps users search and shop fashion and lifestyle products smarter across hundreds of shops.

Inspiration

Stylight offers shoppable inspiration that makes it easy to know what to buy and how to style it.



Branding & Reach

Stylight offers a unique opportunity for brands to reach an audience that is actively looking for style online.

Profitable Leads

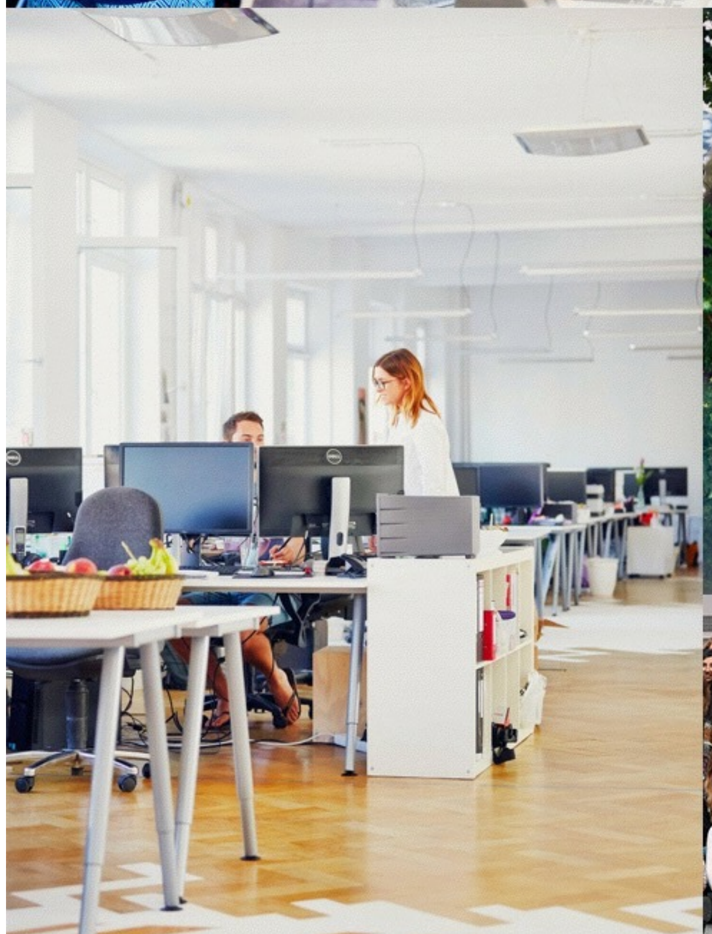
Stylight provides its partners with high-quality leads enabling partner shops to leverage Stylight as a ROI positive traffic channel.

Stylight – acting on a global scale

Experienced & Ambitious Team

Innovative cross-functional organisation with flat hierarchy builds a unique team spirit.

- +200 employees
- 40 PhDs/Engineers
- 28 years average age
- 63% female
- 23 nationalities
- 0 suits



Stylight

Agenda

Probabilistic data structures

Bloom filters, CountMin or Hyperloglog

Open Source stack

Amazon AWS

Google Cloud

Serverless architecture

The Nature of Data

Sources of data:

- Web tracking
  - Metrics tracking
  - Behaviour tracking
- Business intelligence ETL
- Internal Services
  - ML tagging service

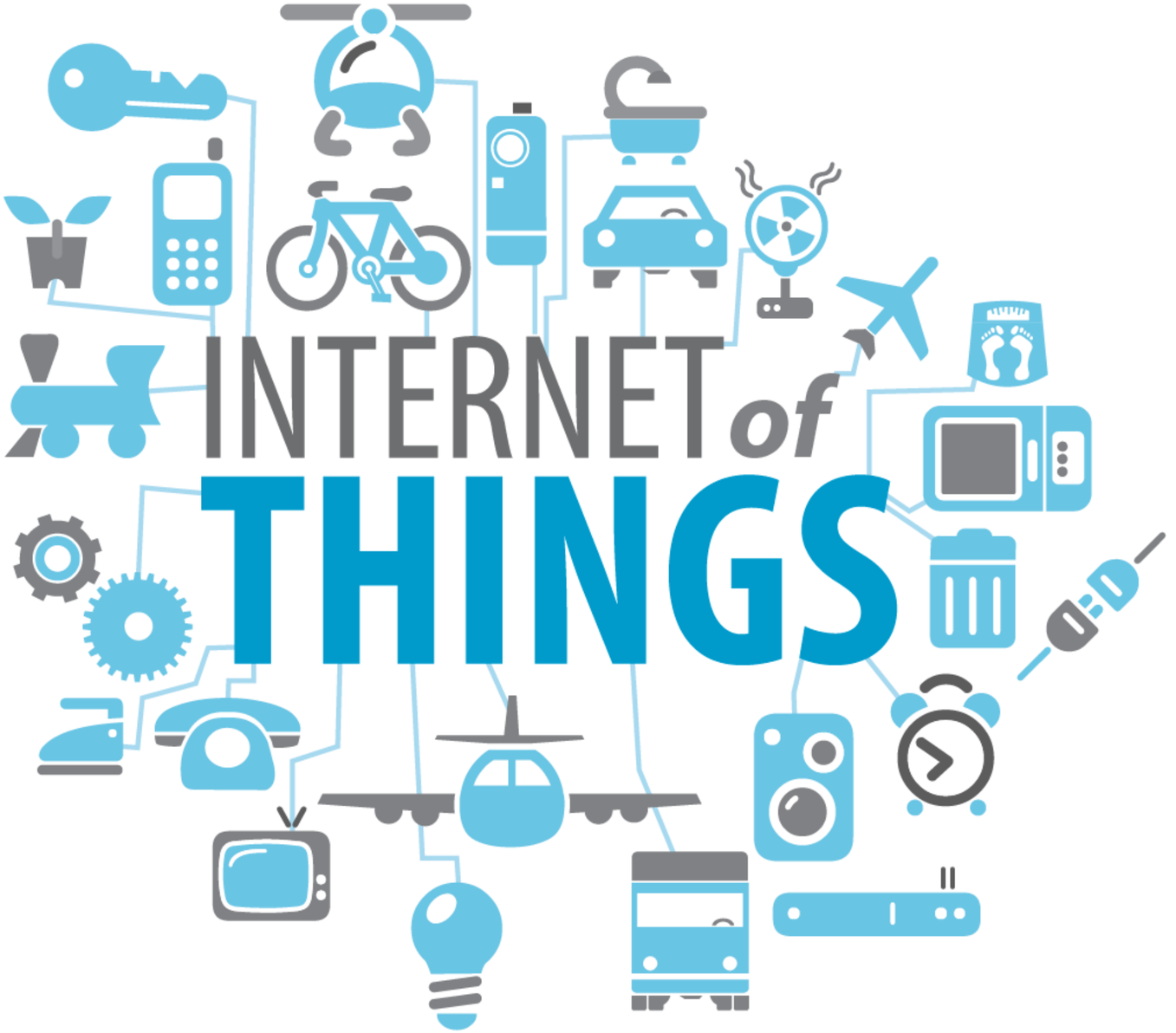
Access patterns

- Real-time
- Nearly real-time
- Daily batches

Probabilistic data structures

Data structures that use different **probabilistic** approaches to **compactly store** data


SAY BIG DATA
ONE MORE TIME
memegenerator.net


INTERNET of
THINGS

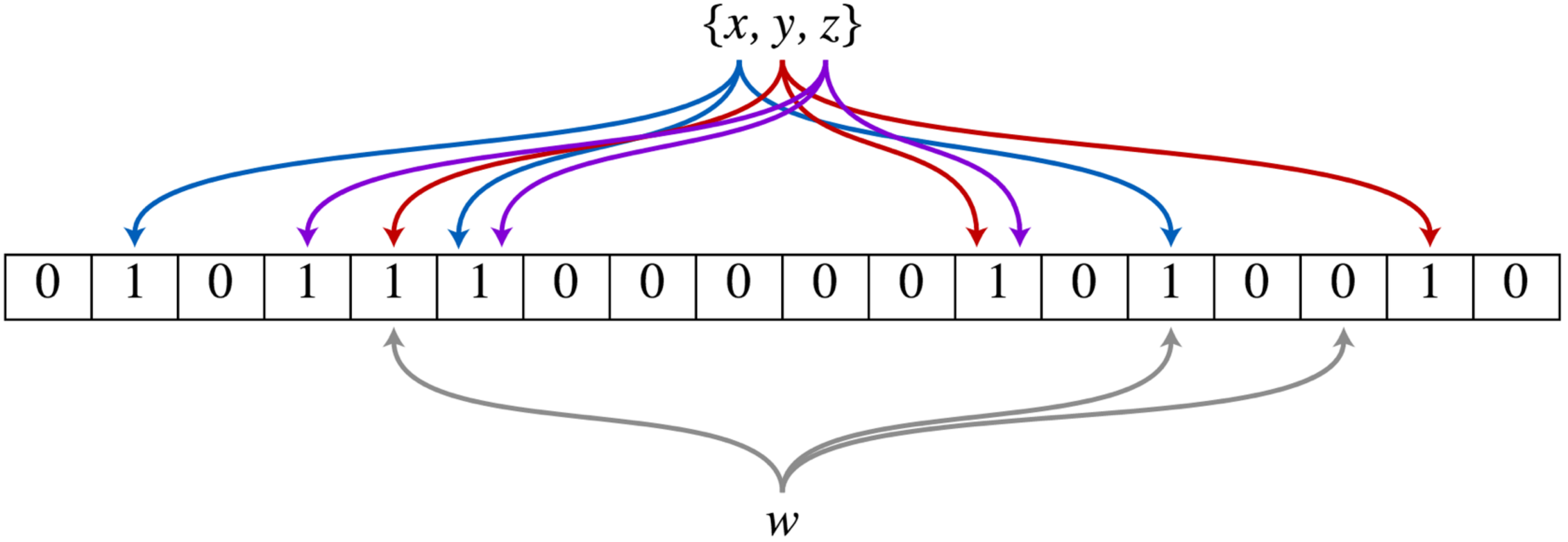
Bloom filter

Approximate Membership

A Bloom filter is a **space-efficient** probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is **used to test whether an element is a member of a set.** False positive matches are possible, but false negatives are not

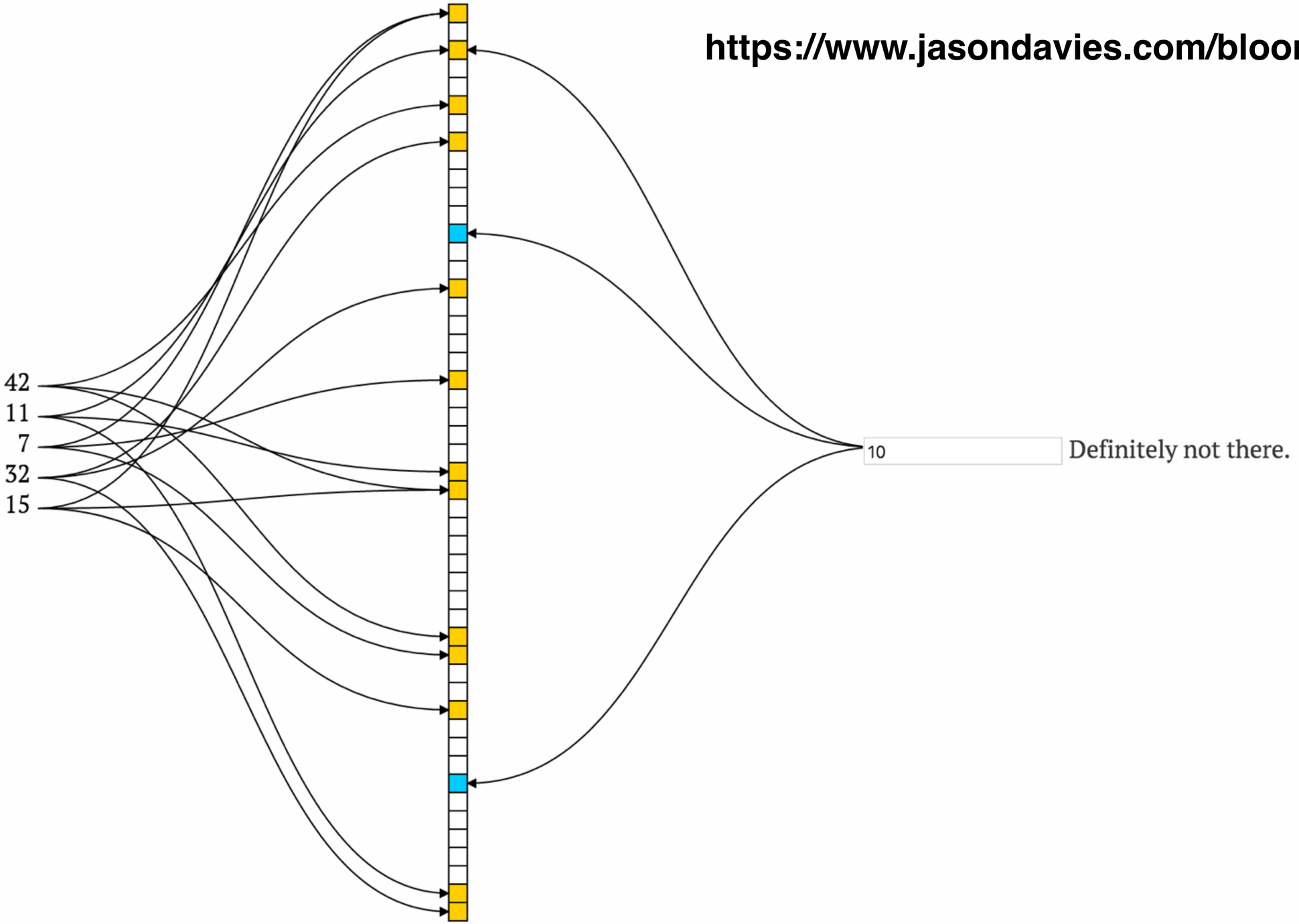
A Bloom filter is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. **False positive matches are possible, but false negatives are not**

- bit array of **m** bits.
- **k** different hash functions with a uniform random distribution


{x, y, z}
0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0
w

```go
// Add data to the Bloom Filter. Returns the filter (allows chaining)
func (f *BloomFilter) Add(data []byte) *BloomFilter {
        h := baseHashes(data)
        for i := uint(0); i < f.k; i++ {
                f.b.Set(f.location(h, i))
        }
        return f
}
```

https://www.jasondavies.com/bloomfilter/



Size estimation

hash functions

$$k = \frac{m}{n} ln2$$

memory usage

$$m = -\frac{n \ln p}{(\ln 2)^2}$$

n - estimated number of elements
p - false positive probability
m - required bit array length

Example:
n=1,000,000
FPR 10% ~= 4800000 Bit ~= **600 kByte**
FPR 0.1% ~= 14400000 Bit ~= **1.8 MByte**

Use-cases

- Caches
- Databases
  - HBase
  - Cassandra
- Networking

https://github.com/willf/bloom
https://github.com/reddragon/bloomfilter.go

https://github.com/seiflotfy/dlCBF
https://github.com/patrickmn/go-bloom

https://github.com/armon/bloomd
https://github.com/geetarista/go-bloomd

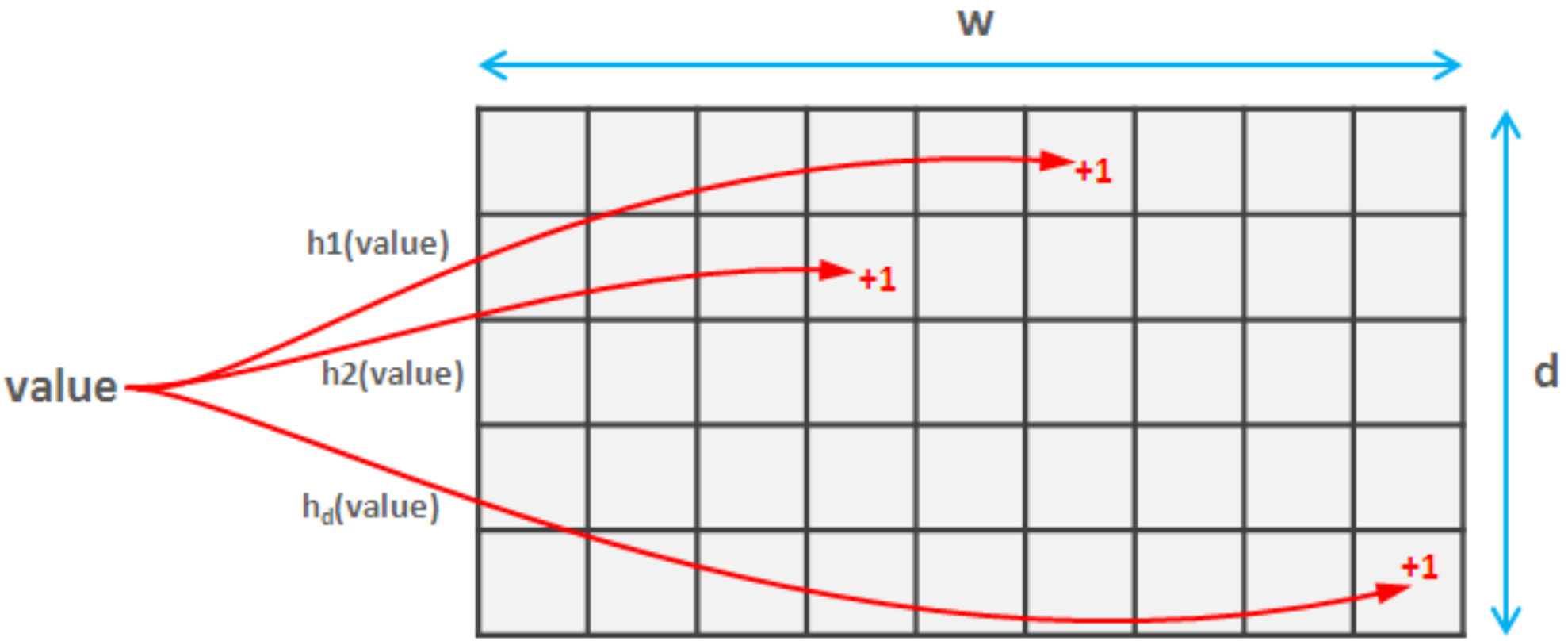
Extensions

- Cardinality estimate (increment counter when add a new)
- Scalable Bloom filters (add another hash function on top)
- Counting Bloom filters
  - increment every time we see it

Count-Min

Frequency estimator

- matrix of **w** columns and **d** rows
- hash function associated with every row


w
d
value
h1(value)
h2(value)
h_d(value)
+1
+1
+1

HyperLogLog

Cardinality estimator

HyperLogLog is an algorithm for the count-distinct problem, **approximating the number of distinct elements** in a multiset.

The HyperLogLog algorithm is able to estimate **cardinalities of >10^9** with a typical **error rate of 2%**, using **1.5kB** of memory[3]

Hash function

hash(x) -> stream of bits {1,0,0,1,0,1..}

- hash generates **uniformly distributed values**
- every bit is **independent**

Bit probability

p(first bit - 0) = 1/2

p(second bit - 0) = 1/2

5 consecutive zeros - (1/2)^5

N consecutive zeros - (1/2)^N

Guessing bits

p(first bit - 0) = 1/2

p(second bit - 0) = 1/2

5 consecutive zeros - (1/2)^5

N consecutive zeros - (1/2)^N

N = 32, Odds = 1/4294967296 -> Expected 4294967296 samples

Storing bits

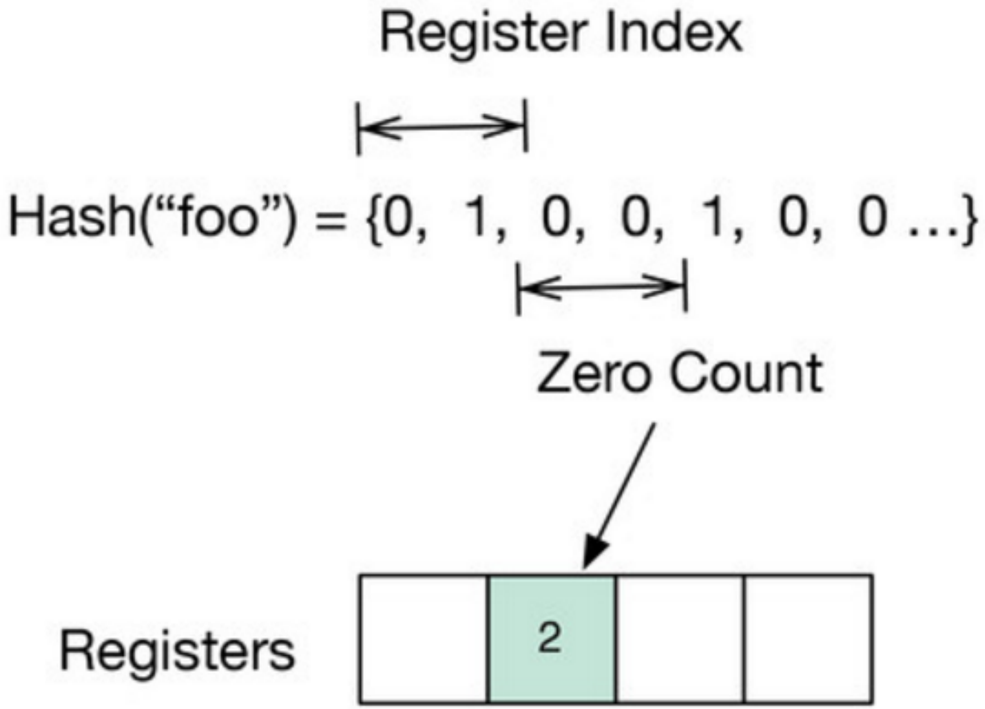
N = 32 = {1,0,0,0,0} = 6bit

With 6bits we can count 2^64

Where the name is coming from Log(Log(64)) = 6

Multiple registers

- Create **m** registers
- Partition the bit stream
  - first log(m) - register index
  - rest used for actual values



HyperLogLog - add

```
// Add adds a new item to HyperLogLog h.
func (h *HyperLogLog) Add(item Hash32) {
        x := item.Sum32()
        i := eb32(x, 32, 32-h.p) // {x31,...,x32-p}
        w := x<<h.p | 1<<(h.p-1) // {x32-p,...,x0}

        zeroBits := clz32(w) + 1
        if zeroBits > h.reg[i] {
                h.reg[i] = zeroBits
        }
}
```

HyperLogLog - size

- Given **m** registers
- Estimate aggregated value
  - Min? Max? Avg? Median?
  - Geometric/Harmonic mean!
- Estimate A*m*H

$$\text{geometric mean} = \left(\prod_{n=1}^{k} x_n \right)^{\frac{1}{k}}$$

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_1} + \ldots + \frac{1}{x_1}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Step | Play | Reset

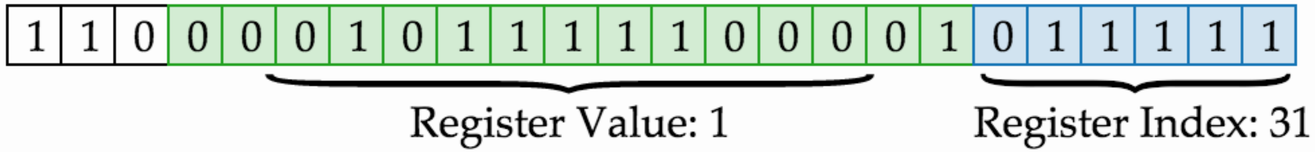
☑ Auto Scale

Delay: 100 ms

Step Size: 7

Value: 966,823

Hash Value: 4,290,967,647



Register Values:

$m = 64$

2	1	5	2	3	3	0	0	0	2	0	7	3	3	1	1
2	2	5	0	1	5	7	1	3	3	2	3	1	3	2	3
0	0	0	5	0	1	6	1	0	2	7	1	1	3	1	0
2	6	2	0	6	1	1	2	8	3	2	2	0	4	4	3

Actual Cardinality: 105

Algorithm	*Estimated Cardinality*	*% Error*
LogLog	127	21.0
HyperLogLog	102	−2.9

http://content.research.neustar.biz/blog/hll.html

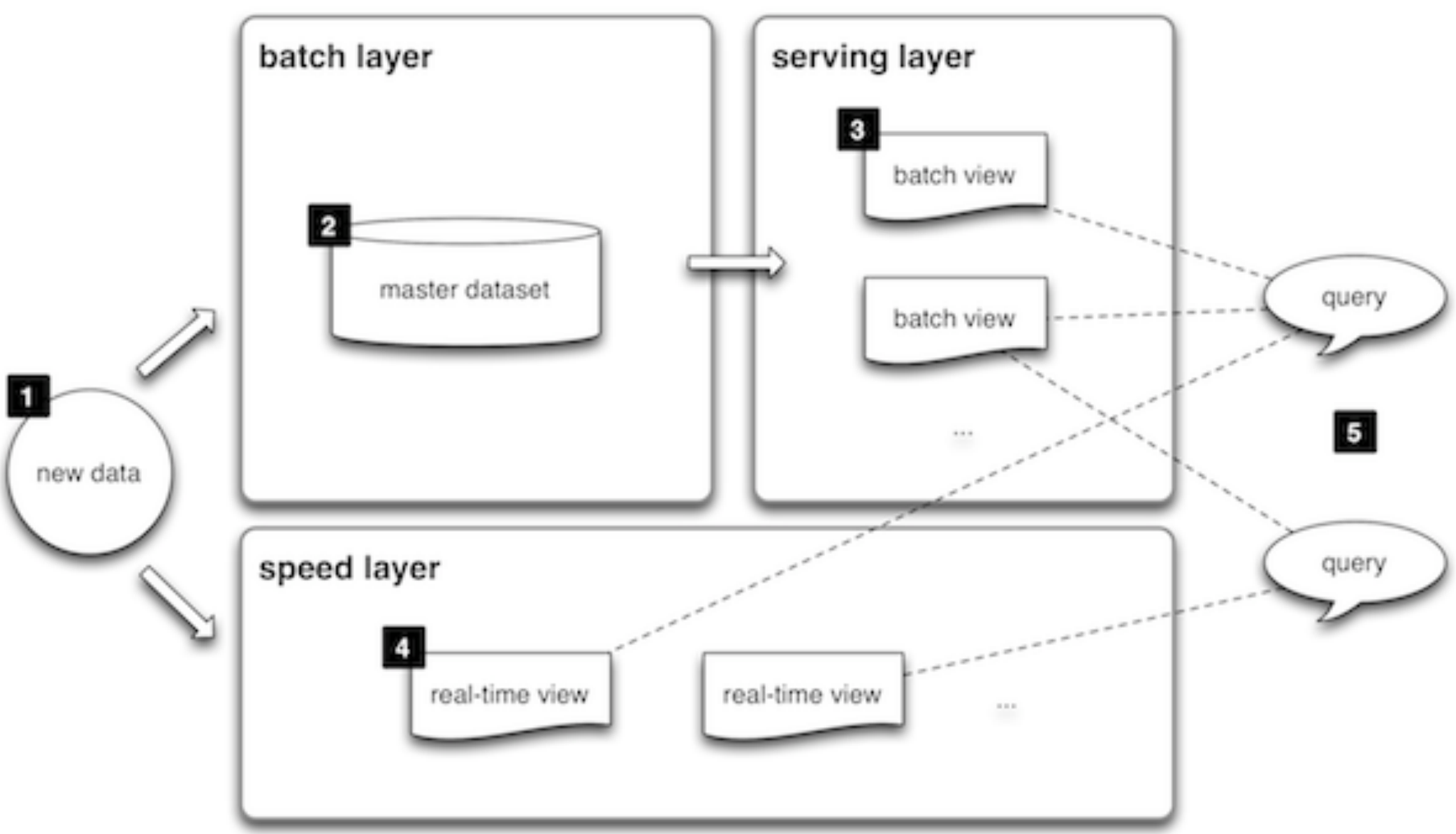
Use-cases

- Databases
  - Redis
  - PostgreSQL
  - Redshift
  - Impala
  - Hive
- Spark

https://github.com/clarkduvall/hyperloglog
https://github.com/armon/hlld

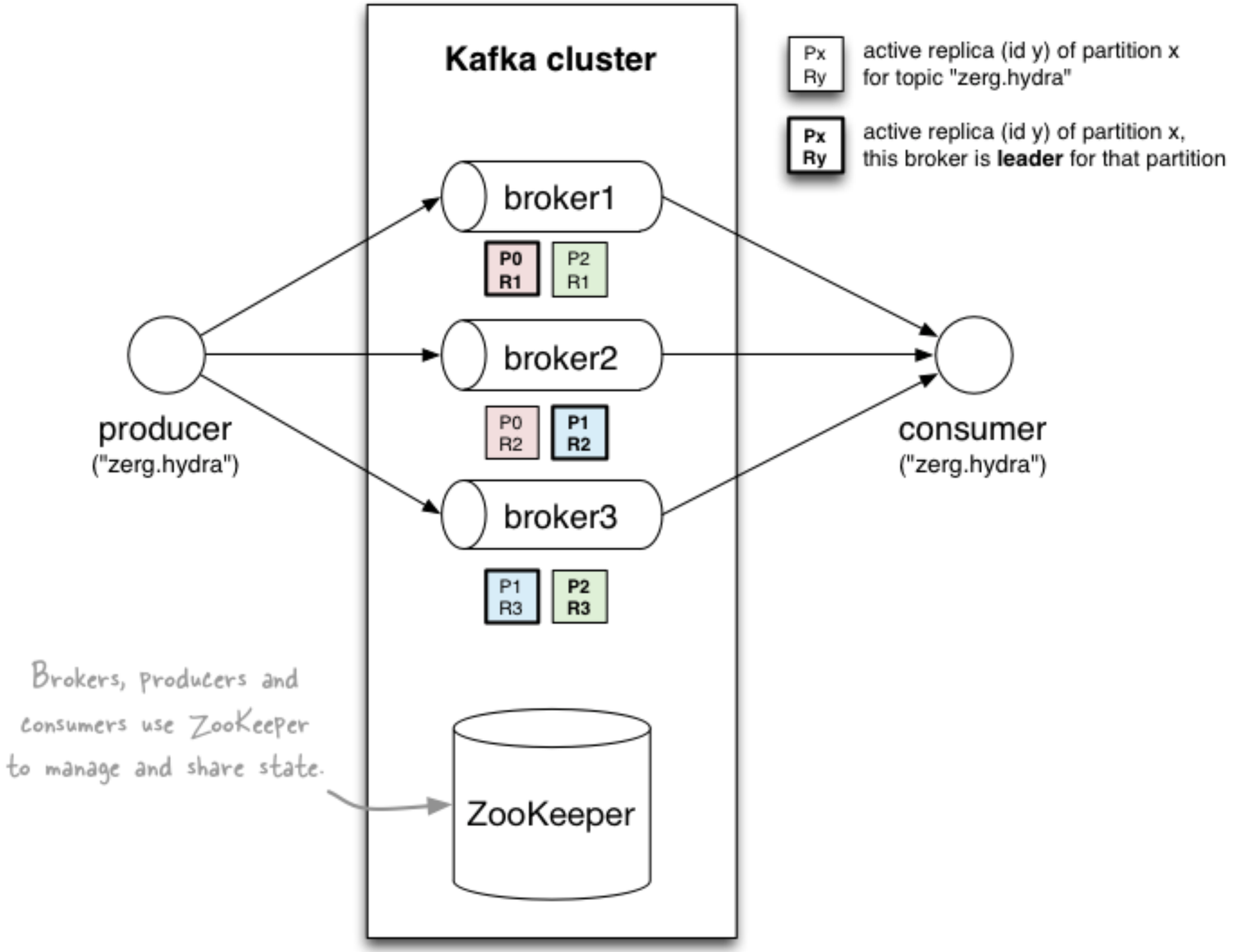
In computing, a **pipeline** is a set of data processing elements connected in series, where the output of one element is the input of the next one.

Open Source Stack

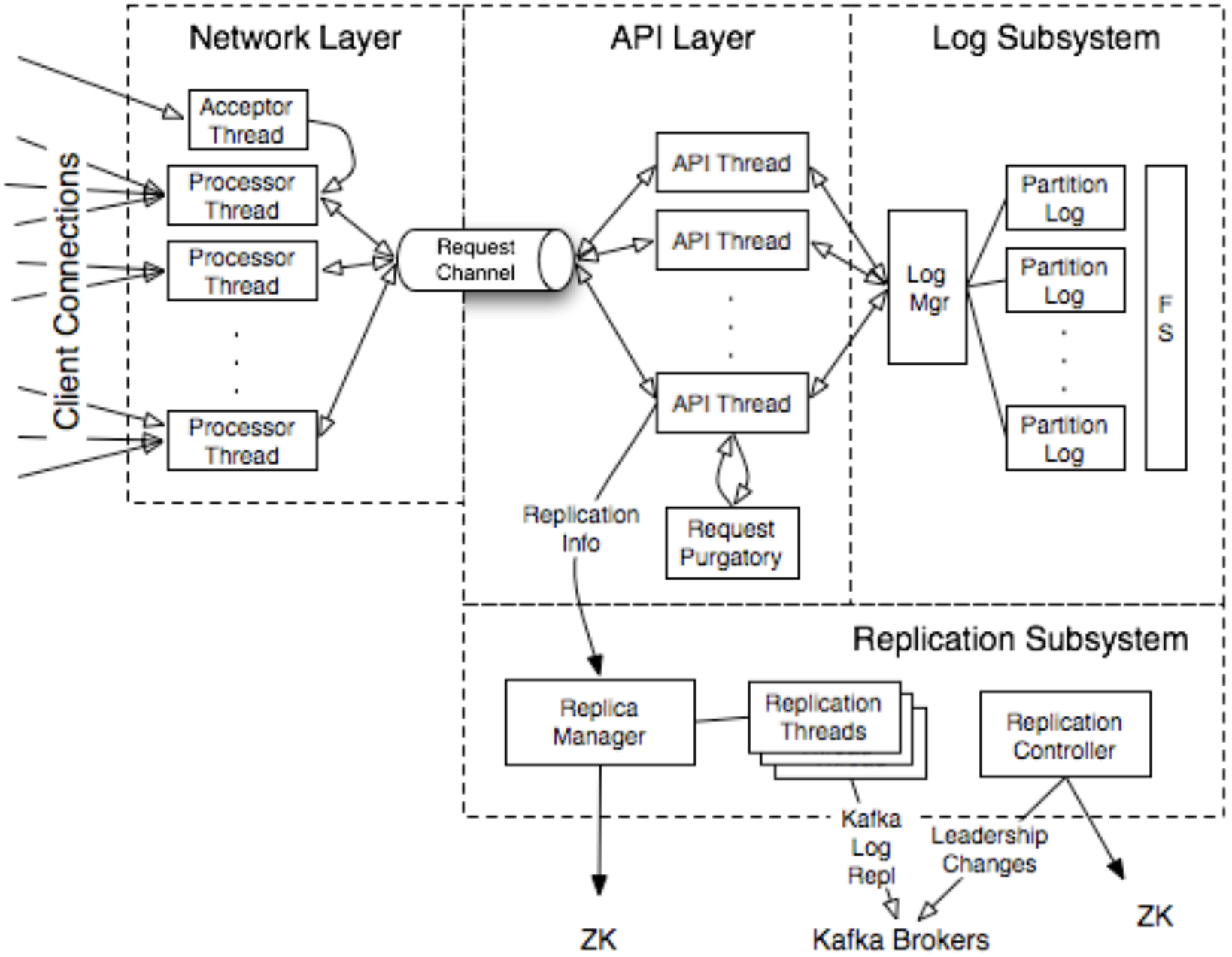


http://lambda-architecture.net/

Apache Kafka is publish-subscribe messaging rethought as a distributed commit log.


Kafka cluster
Px
Ry
active replica (id y) of partition x
for topic "zerg.hydra"
Px
Ry
active replica (id y) of partition x,
this broker is leader for that partition
broker1
P0
R1
P2
R1
broker2
P0
R2
P1
R2
broker3
P1
R3
P2
R3
producer
("zerg.hydra")
consumer
("zerg.hydra")
Brokers, producers and
consumers use ZooKeeper
to manage and share state.
ZooKeeper

Kafka Broker Internals



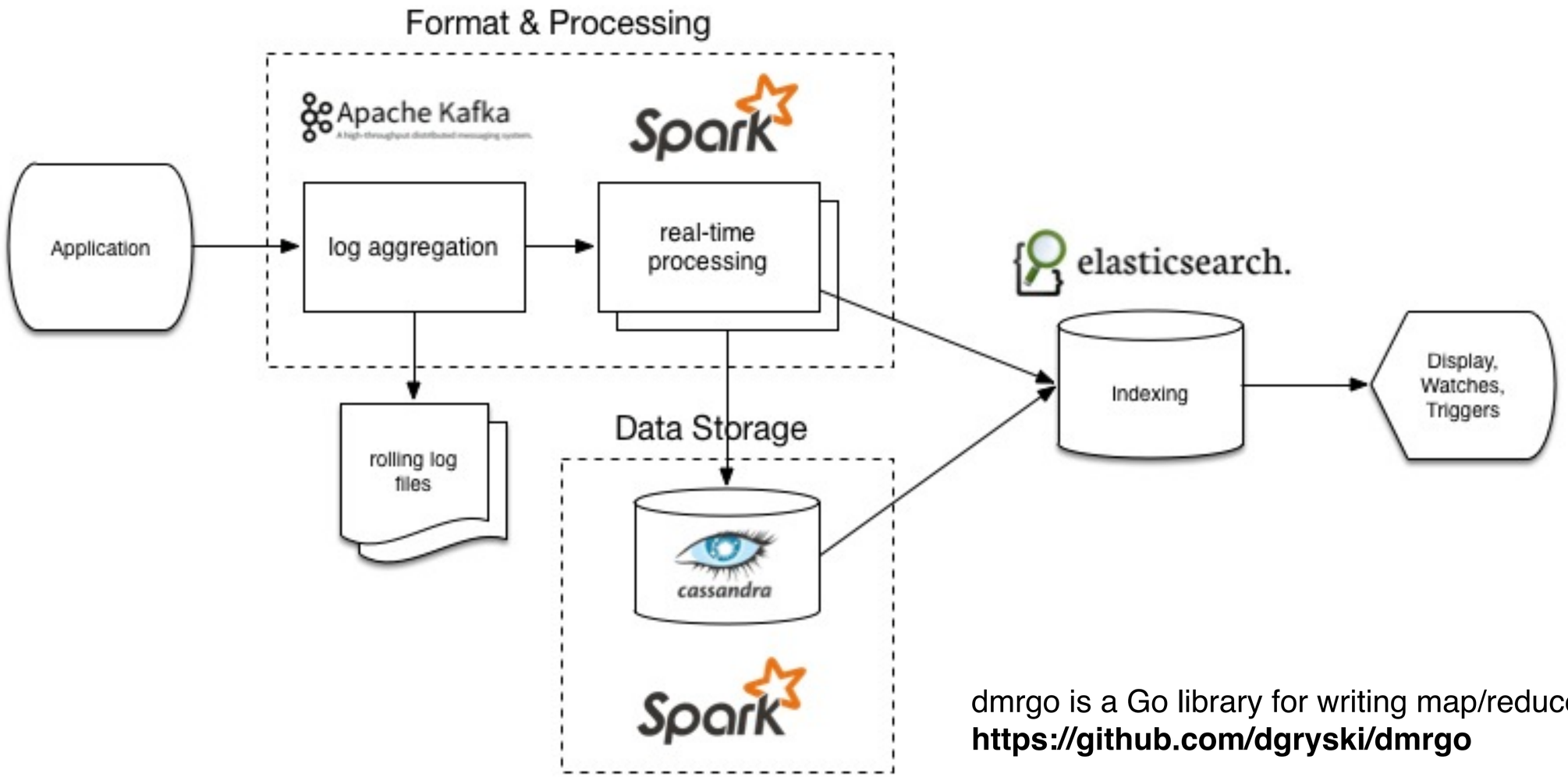
Libraries

- Sarama is an MIT-licensed Go client library for Apache Kafka version 0.8 (and later)
https://github.com/Shopify/sarama

Go Kafka Client
https://github.com/elodina/go_kafka_client

```
producer, err :=
NewAsyncProducer([]string{"localhost:9092"}, nil)
if err != nil {
    panic(err)
}

defer func() {
    if err := producer.Close(); err != nil {
        log.Fatalln(err)
    }
}()
```

```
var enqueued, errors int
ProducerLoop:
for {
    select {
    case producer.Input() <- &ProducerMessage{Topic: "my_topic",
Key: nil, Value: StringEncoder("testing 123")}:
        enqueued++

    case err := <-producer.Errors():
        log.Println("Failed to produce message", err)
        errors++
    case <-signals:
        break ProducerLoop
    }
}

log.Printf("Enqueued: %d; errors: %d\n", enqueued, errors)
```



dmrgo is a Go library for writing map/reduce jobs.
https://github.com/dgryski/dmrgo

http://www.ipponusa.com/wp-content/uploads/2014/10/spark-architecture.jpg

Results

- Scalable
- Flexible

- High costs of maintenance
- Not so easy to setup

A programming language is low level when its programs require attention to the irrelevant.

Alan Jay Perlis / Epigrams on Programming

Amazon AWS

Kinesis Streams

Real-time Ingest

Highly Scalable

Durable

Elastic

Replay-able Reads

Continuous Processing FX

Elastic

Load-balancing incoming streams

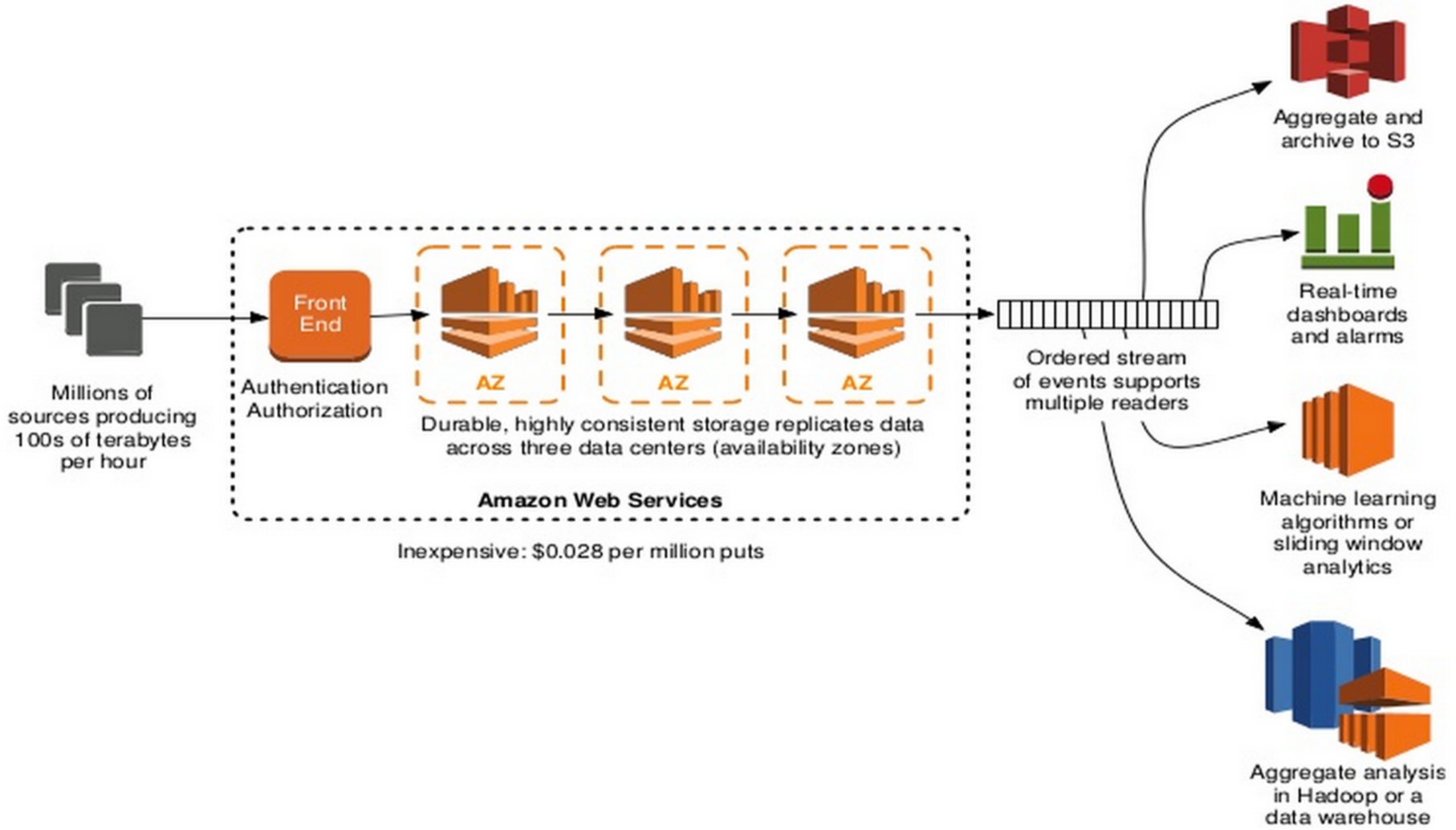
Fault-tolerance, Checkpoint / Replay

Enable multiple processing apps in paralle

Managed Service

Low end-to-end latency

Enable data movement into Stores/ Processing Engines


Aggregate and archive to S3
Real-time dashboards and alarms
Front End
Authentication Authorization
AZ
AZ
AZ
Durable, highly consistent storage replicates data across three data centers (availability zones)
Amazon Web Services
Millions of sources producing 100s of terabytes per hour
Ordered stream of events supports multiple readers
Machine learning algorithms or sliding window analytics
Aggregate analysis in Hadoop or a data warehouse
Inexpensive: $0.028 per million puts

Libraries

- AWS SDK for Go

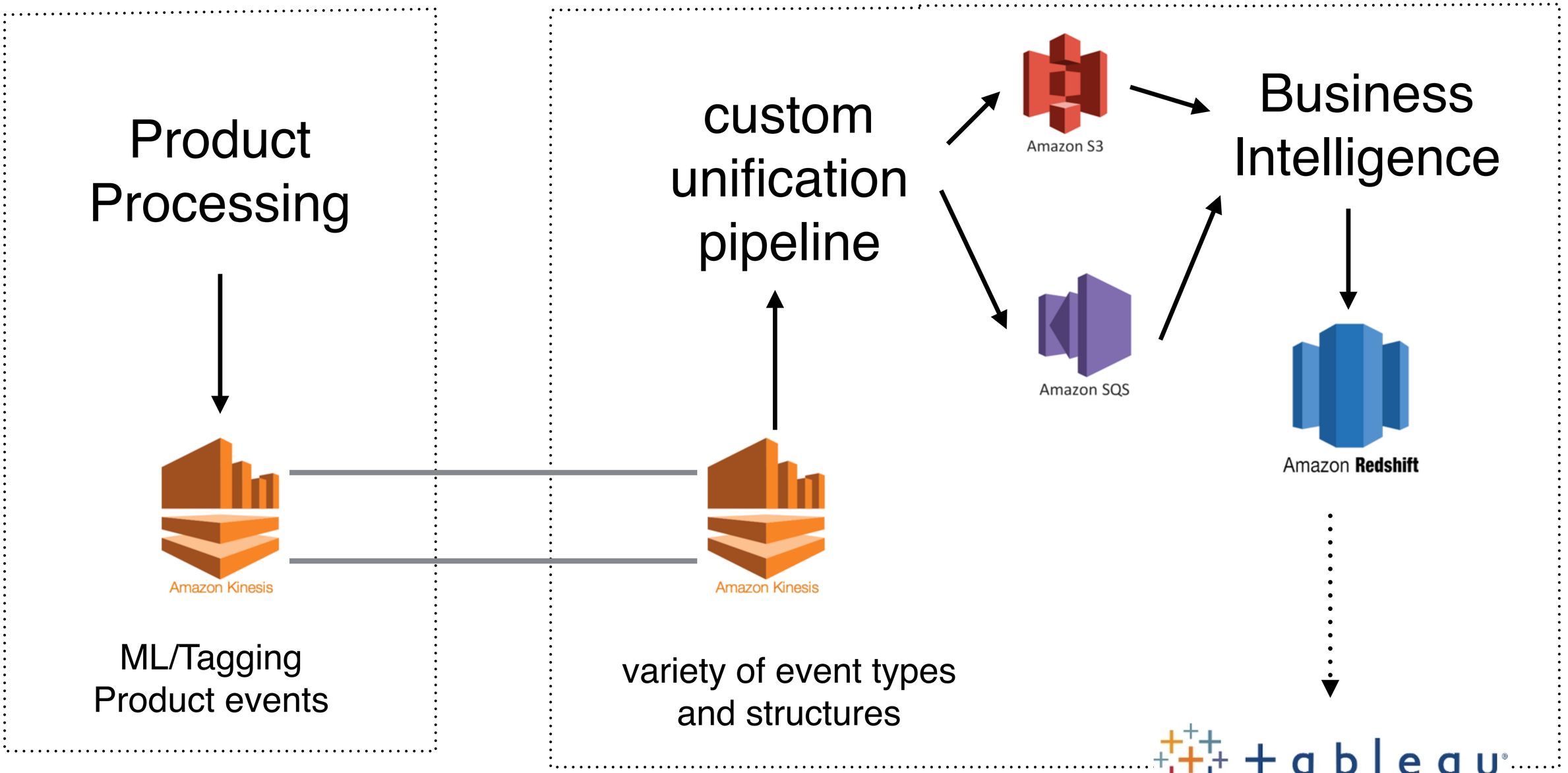
https://github.com/aws/aws-sdk-go

```
svc := kinesis.New(session.New())

params := &kinesis.PutRecordsInput{
        Records: []*kinesis.PutRecordsRequestEntry{ // Required
                { // Required
                        Data:            []byte("PAYLOAD"),          // Required
                        PartitionKey:    aws.String("PartitionKey"), // Required
                        ExplicitHashKey: aws.String("HashKey"),
                },
                // More values...
        },
        StreamName: aws.String("StreamName"), // Required
}
resp, err := svc.PutRecords(params)
```

```
svc := kinesis.New(session.New())

params := &kinesis.GetRecordsInput{
        ShardIterator: aws.String("ShardIterator"), // Required
        Limit:         aws.Int64(1),
}
resp, err := svc.GetRecords(params)
```

Kinesis Firehose

Kinesis Analytics


Product
Processing
Amazon Kinesis
ML/Tagging
Product events
custom
unification
pipeline
Amazon Kinesis
variety of event types
and structures
Amazon S3
Amazon SQS
Business
Intelligence
Amazon Redshift
tableau

Google Cloud

Pub/Sub

Scalable, flexible, and globally available messaging

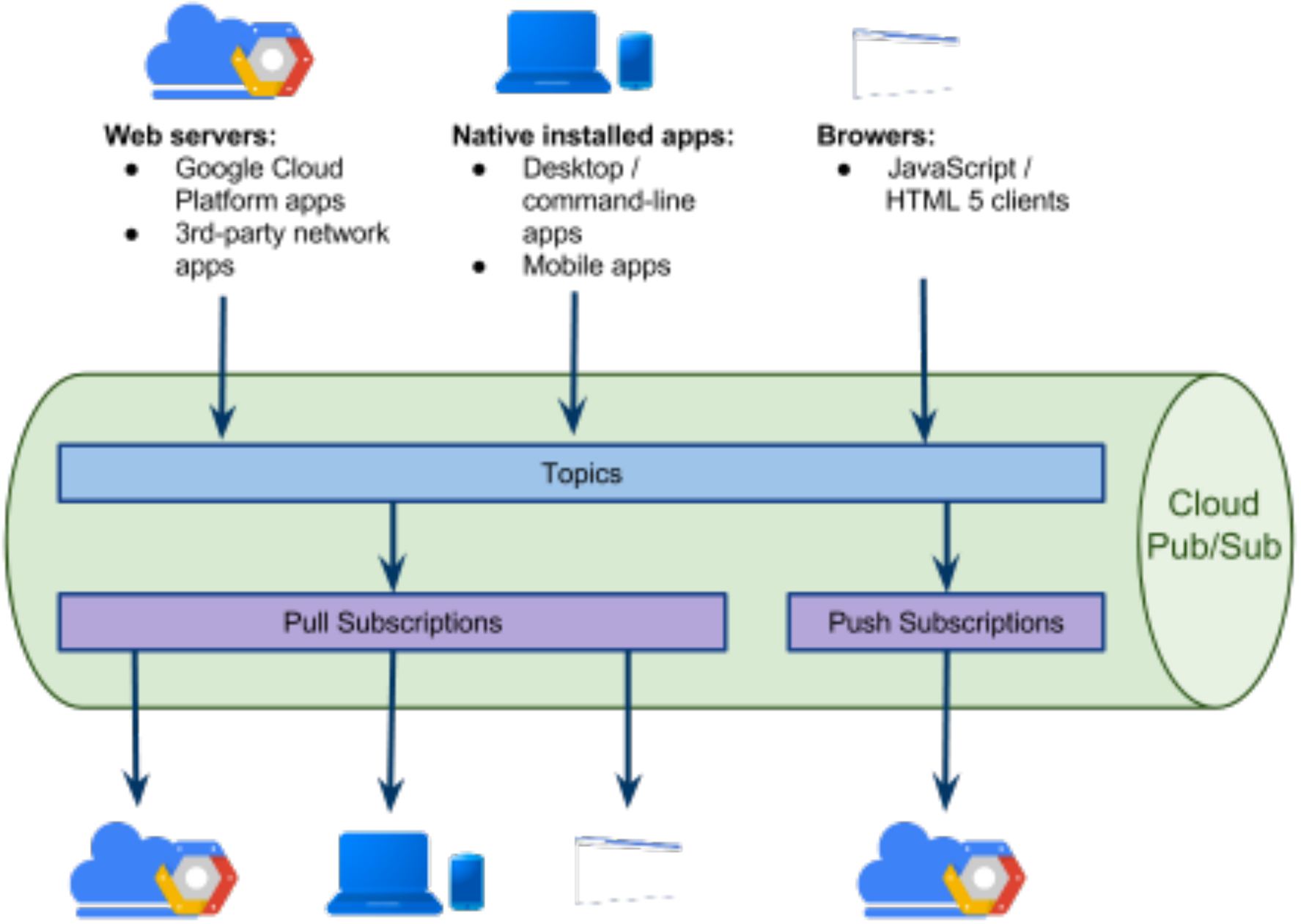
Dataflow

Stream & batch processing, unified and simplified

BigQuery

Ingest data at 100,000 rows per second

Fully Managed, No-Ops Services


Web servers:
Google Cloud Platform apps
3rd-party network apps
Native installed apps:
Desktop / command-line apps
Mobile apps
Browers:
JavaScript / HTML 5 clients
Topics
Cloud Pub/Sub
Pull Subscriptions
Push Subscriptions

Libraries

- Google APIs Client Library for Go

https://github.com/GoogleCloudPlatform/gcloud-golang

Cloud Dataflow

Cloud Dataflow is a collection of SDKs for **building** batch or streaming parallelized data processing pipelines.

Cloud Dataflow is a fully managed service for **executing** optimized parallelized data processing pipelines.

```
Pipeline{
        Who => Inputs
        What => Transforms
        Where => Windows
        When => Watermarks + Triggers
        To => Outputs
}
```

<- At once guarantee (modulo completeness thresholds)

<- GCS, Pub/Sub, BigQuery, w/Avro, XML, JSON, ...

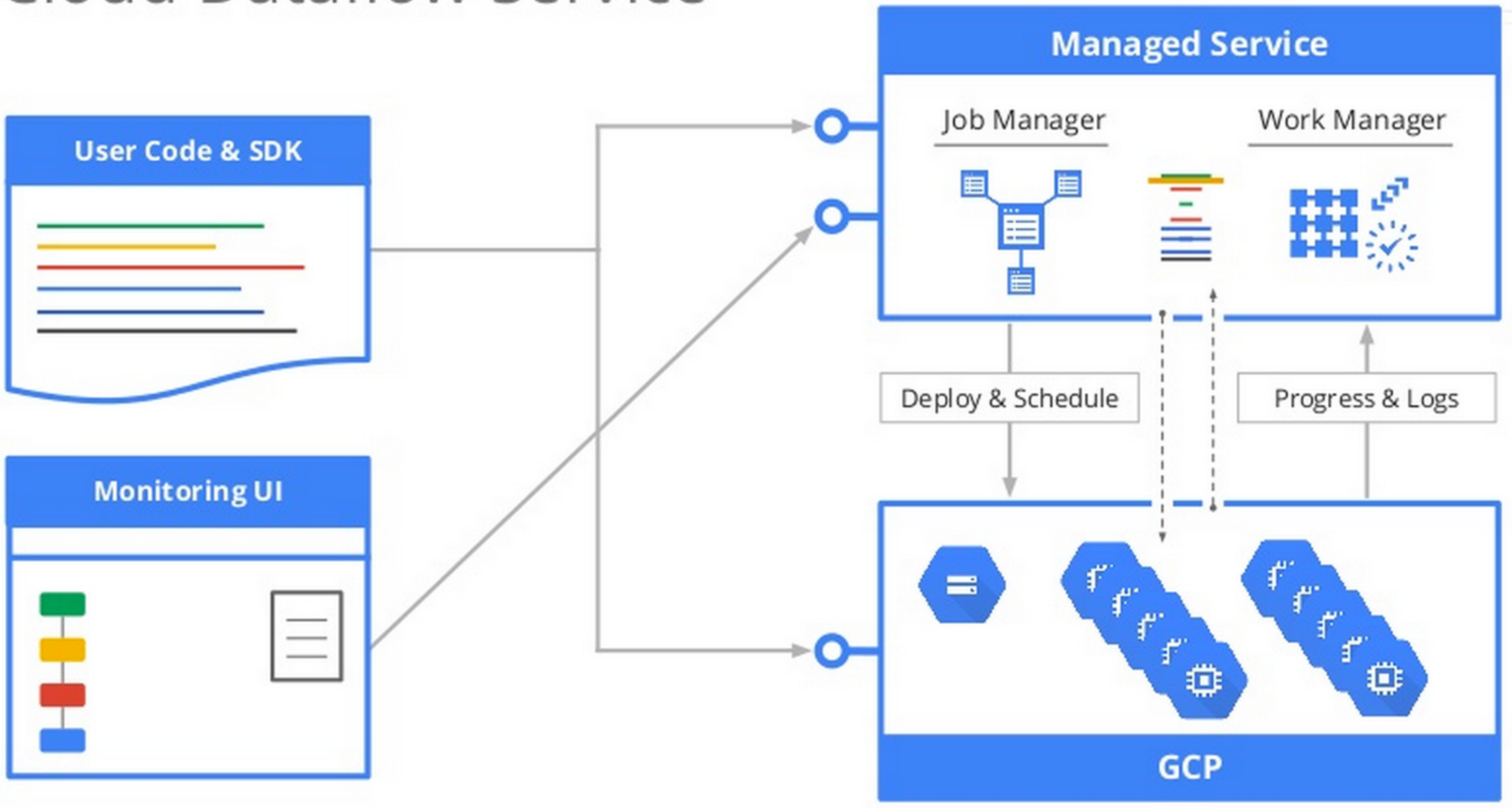
<- Aggregations, Filters, Joins, ...

<- Time space Fixed, Sliding, Sessions, ...

<- Correctness

<- GCS, Pub/Sub, BigQuery, ...

Cloud Dataflow Service



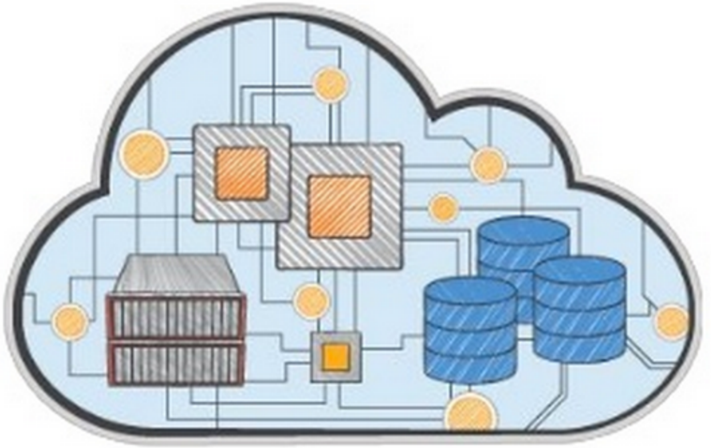
Benefits of Cloud Dataflow

- No Ops - truly elastic data processing for the cloud
  - On demand resource allocation w/intelligent auto-scaling
  - Automated worker lifetime management
  - Automated work optimization
- Unified model - for batch & stream based processing
  - Functional programming model
  - Fine grained correctness primitives
- Open sourced SDK @ github
  - Java 7 today @ /GoogleCloudPlatform/DataflowJavaSDK
  - Python 2 in progress
  - Scala @/darkjh/scalaflow & /jhlch/scala-dataflow-dsl
  - Spark runner@ /cloudera/spark-dataflow
  - Flink runner @ /dataArtisans/flink-dataflow

Serverless architecture

AWS Lambda: A compute service that runs your code in response to events

Lambda functions: Stateless, event-driven code execution

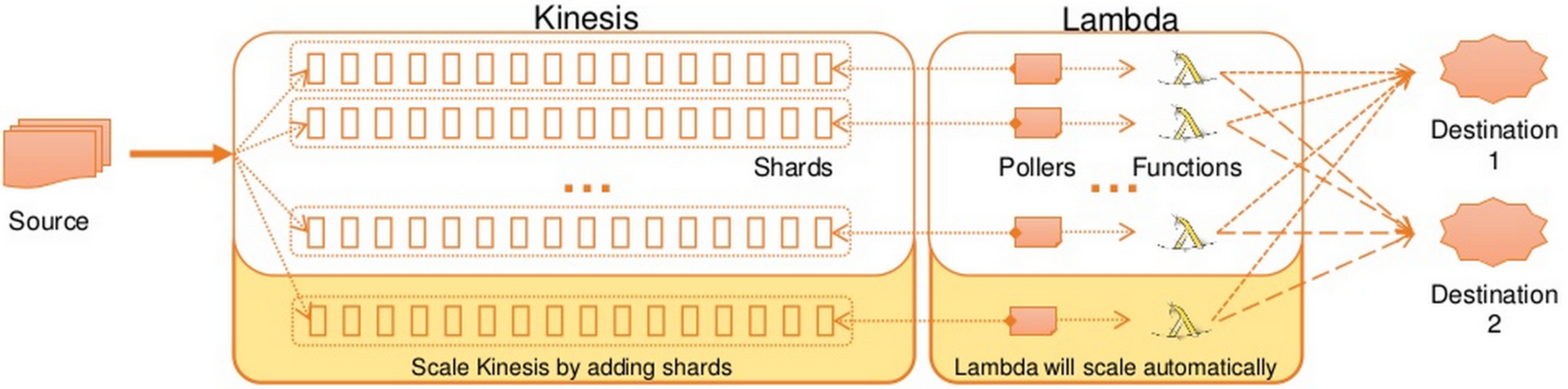


Triggered by events:

- Put to an Amazon S3 bucket
- Record in an Amazon Kinesis stream
- Direct sync and async invocations

Makes it easy to

- Build back-end services that perform at scale
- Perform data-driven auditing, analysis, and notification


Kinesis
Lambda
Source
Shards
Pollers
Functions
Destination 1
Destination 2
Scale Kinesis by adding shards
Lambda will scale automatically

Lambda > Functions

You have 10 Lambda function(s) using 14.8 MB of code storage. Choose any Lambda function to view details on invocation requests, duration, and errors (metrics may take up to 60 seconds to appear).

Create a Lambda function | Actions

Filter

Viewing 1-10 of 10

Function name	Description	Code size	Memory (MB)	Timeout (s)
BI_gofunction		2.9 MB	128	5
BI_BiCloudWatch	BiCloudWatch function powered by Apex	4.1 kB	128	30
BiCloudWatch-test	An Amazon SNS trigger that sends CloudWatch alarm notifications to Slack.	1.6 kB	128	3


APEX
serverless architecture

```
➜  apex git:(master) apex deploy -v gofunction
   • deploying                          function=gofunction
   • created build (3.1 MB)             function=gofunction
   • unchanged                          function=gofunction
   • deploying config                   function=gofunction
```

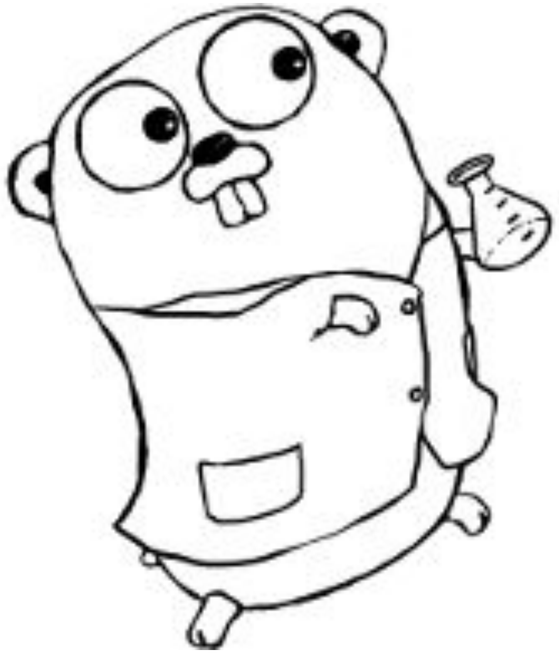
Features

- Supports languages Lambda does not natively support via shim, such as Go
- Binary install (useful for continuous deployment in CI etc)
- Hook support for running commands (transpile code, lint, etc)
- Project level function and resource management
- Configuration inheritance and overrides
- Command-line function invocation with JSON streams
- Transparently generates a zip for your deploy
- Ignore deploying files with .apexignore
- Function rollback support
- Tail function CloudWatchLogs
- Concurrency for quick deploys
- Dry-run to preview changes

```go
import (
  "encoding/json"

  "github.com/apex/go-apex"
)

type Message struct {
  Hello string `json:"hello"`
}

func main() {
  apex.HandleFunc(func(event json.RawMessage, ctx *apex.Context) (interface{}, error) {
    return &Message{"baz"}, nil
  })
}
```

Possibilities

- all Lambdas in one place with version control
- integration tests with real events
- proper CI/CD setup



Stylight

Make Style Happen

sergii.khomenko@stylight.com
@lc0d3r

www.stylight.com

Related links

1. Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. 1970
2. Interactive visualisation: Bloom Filters
3. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm
4. HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm
5. HyperLogLog — Cornerstone of a Big Data Infrastructure
6. Armon Dadgar on Bloom Filters and HyperLogLog

Related links

7. https://github.com/willf/bloom
8. Google’s Cloud Pub/Sub Real-Time Messaging Service Is Now In Public Beta
9. Streaming Data Processing with Amazon Kinesis and AWS Lambda
10. Google Cloud Dataflow Two Worlds Become a Much Better One
11. https://github.com/apex/apex