

# Processing Big Data in Motion

## Streaming Data Ingestion and Processing

Roger Barga, General Manager, Kinesis Streaming Services, AWS

May 13<sup>th</sup>, 2016

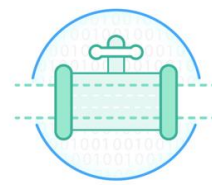


# Riding the Streaming Rapids

Microsoft®  
**Research**



Azure Stream Analytics



Amazon Kinesis Streams



Amazon Kinesis Firehose



Amazon Kinesis Analytics

Streaming Map Reduce  
& Machine Learning over Streams



Relational Semantics  
and Implementation

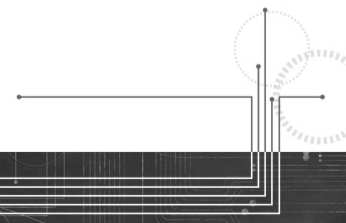
Complex Event Processing  
over Streaming Data

2007 & 2008      2009      2010      2011      2012      2013      2015      2016



*Interest* in and *demand* for  
stream data processing is *rapidly*  
*increasing* \*

\* Understatement of the year...



# Why?

Most data is *produced continuously*

127.0.0.1 user-identifier frank [10/Oct/2000:13:55:36 -0700] "GET /apache\_pb.gif  
HTTP/1.0" 200 2326

## *Common Log Entry*

```
{  
  "payerId": "Joe",  
  "productCode": "AmazonS3",  
  "clientProductCode": "AmazonS3",  
  "usageType": "Bandwidth",  
  "operation": "PUT",  
  "value": "22490",  
  "timestamp": "1216674828"  
}
```

## *Metering Record*

<R,AMZN,T,G,R1>

## *NASDAQ OMX Record*

"SeattlePublicWater/Kinesis/123/Realtime" –  
412309129140

## *MQTT Record*

<165>1 2003-10-11T22:14:15.003Z  
mymachine.example.com evntslog - ID47  
[exampleSDID@32473 iut="3" eventSource="Application"  
eventID="1011"] [examplePriority@32473 class="high"]

## *Syslog Entry*

# Why?

## Time is money

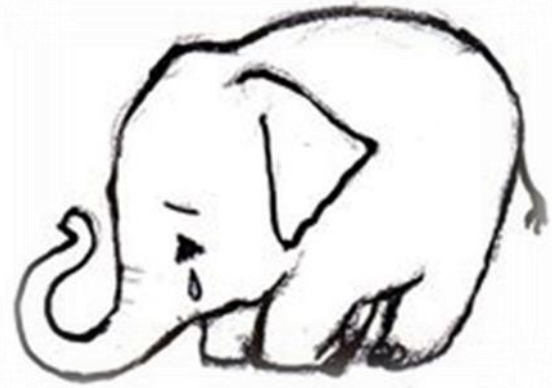
- *Perishable Insights (Forrester)*

- Hourly server logs: **how your systems were misbehaving an hour ago**
- Weekly / Monthly Bill: **What you spent this past billing cycle?**
- Daily fraud reports: **tells you if there was fraud yesterday**
- CloudWatch metrics: **what just went wrong now**
- Real-time spending alerts/caps: **guaranteeing you can't overspend**
- Real-time detection: **blocks fraudulent use now**

# Why?

Time is money

- *Perishable Insights (Forrester)*
- *A more efficient implementation*
- *Most 'Big Data' deployments process continuously generated data (batched)*





# Why?

## Availability

*Variety of stream data processing systems,  
active ecosystem but still early days...*



# Why?

## Disruptive

*Foundational for business critical workflows*

*Enable new class of applications & services  
that process data continuously.*



# You

Need to begin thinking about applications & services in terms of **streams of data** and **continuous processing**.

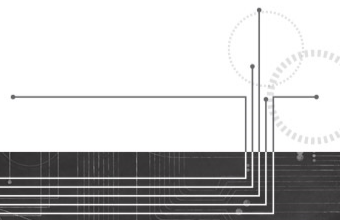
*A change in perspective is worth 80 IQ points...*

*– Alan Kay*

# Agenda

---

- Scalable & Durable Data Ingest
  - A quick word on our motivation
  - Kinesis Streams, through a simple example
- Continuous Stream Data Processing
  - Kinesis Client Library (KCL)
  - One select design challenge: dynamic resharding
  - How customers are using Kinesis Streams today
- Building on Kinesis Streams
  - Kinesis Firehose



# Our Motivation for Continuous Processing

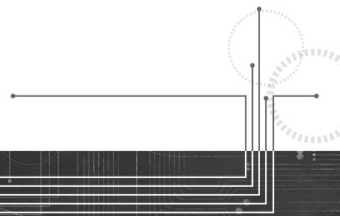
## AWS Metering service

- 100s of millions of billing records per second
- Terabytes<sup>++</sup> per hour
- Hundreds of thousands of sources
- For each customer: gather all metering records & compute monthly bill
- Auditors guarantee 100% accuracy at months end

*Seem perfectly reasonable to run as a batch, but relentless pressure for realtime...*

## With a Data Warehouse to load

- 1000s extract-transform-load (ETL) jobs every day
- Hundreds of thousands of files per load cycle
- Thousands of daily users, hundreds of queries per hour



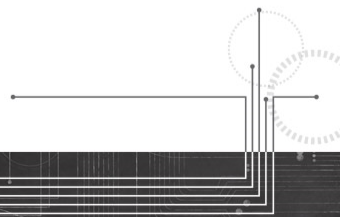
# Our Motivation for Continuous Processing

## AWS Metering service

- 100s of millions of billing records per second
- Terabytes++ per hour
- Hundreds of thousands of sources
- For each customer: gather all metering records & compute monthly bill
- Auditors guarantee 100% accuracy at months end

## Other Service Teams, Similar Requirements

- CloudWatch Logs and CloudWatch Metrics
- CloudFront API logging
- 'Snitch' internal datacenter hardware metrics



# Right Tool for the Job

Enable Streaming **Data Ingestion** and **Processing**

## Real-time Ingest

- Highly Scalable
- Durable
- Replayable Reads



## Continuous Processing

- Support multiple simultaneous data processing applications
- Load-balancing incoming streams, scale out processing
- Fault-tolerance, Checkpoint / Replay



**Amazon Kinesis**

# Example application

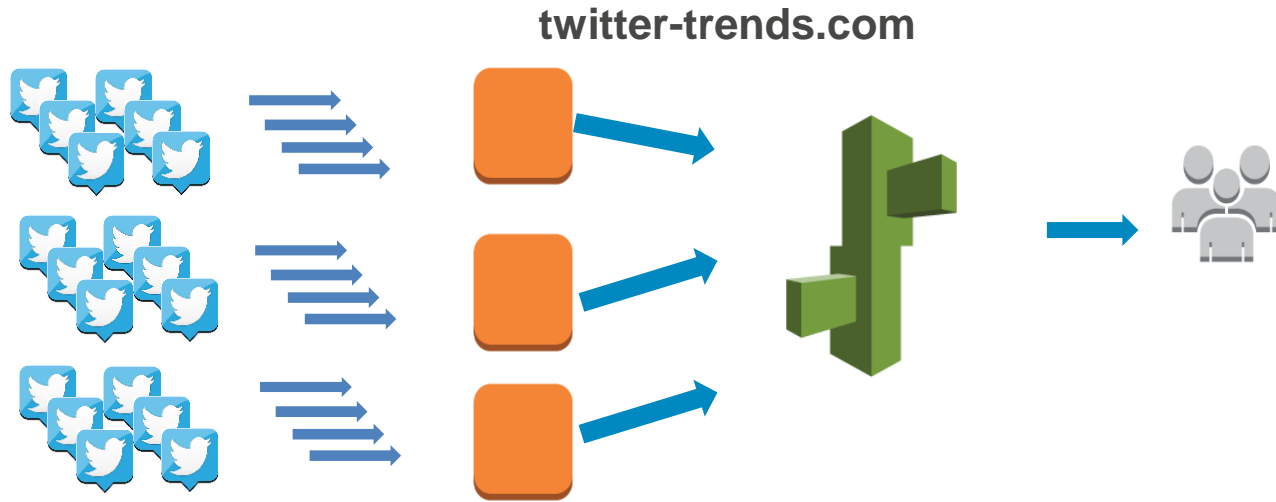
twitter-trends.com website



**Elastic Beanstalk**

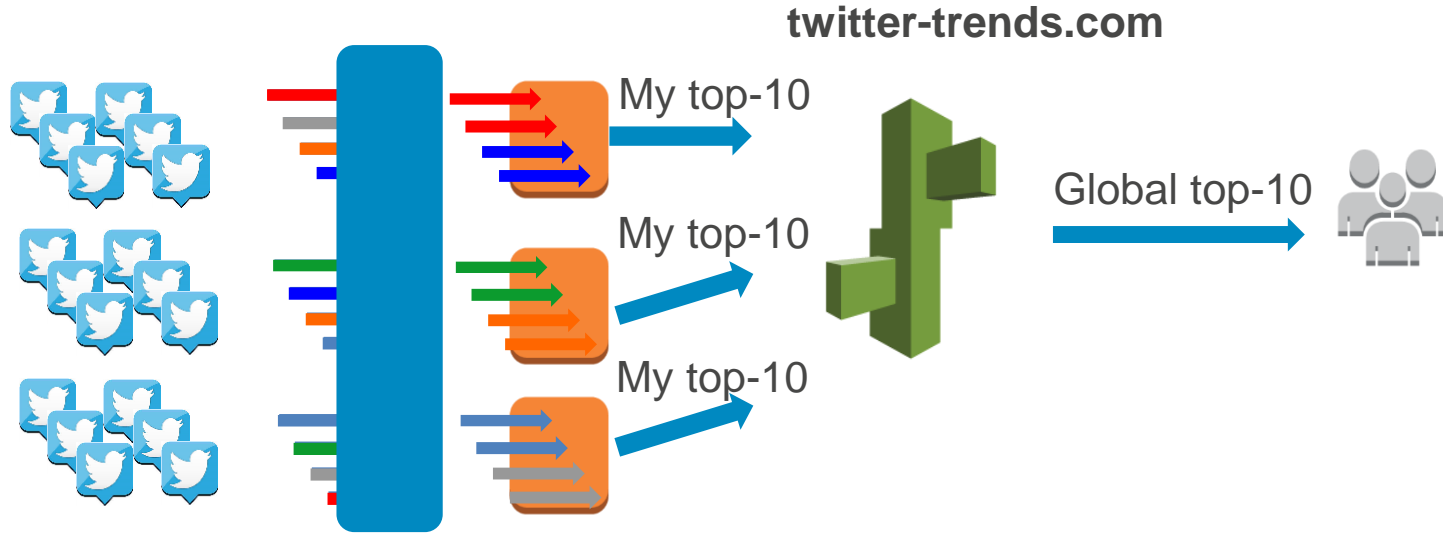
**twitter-trends.com**

# Too big to handle on one box

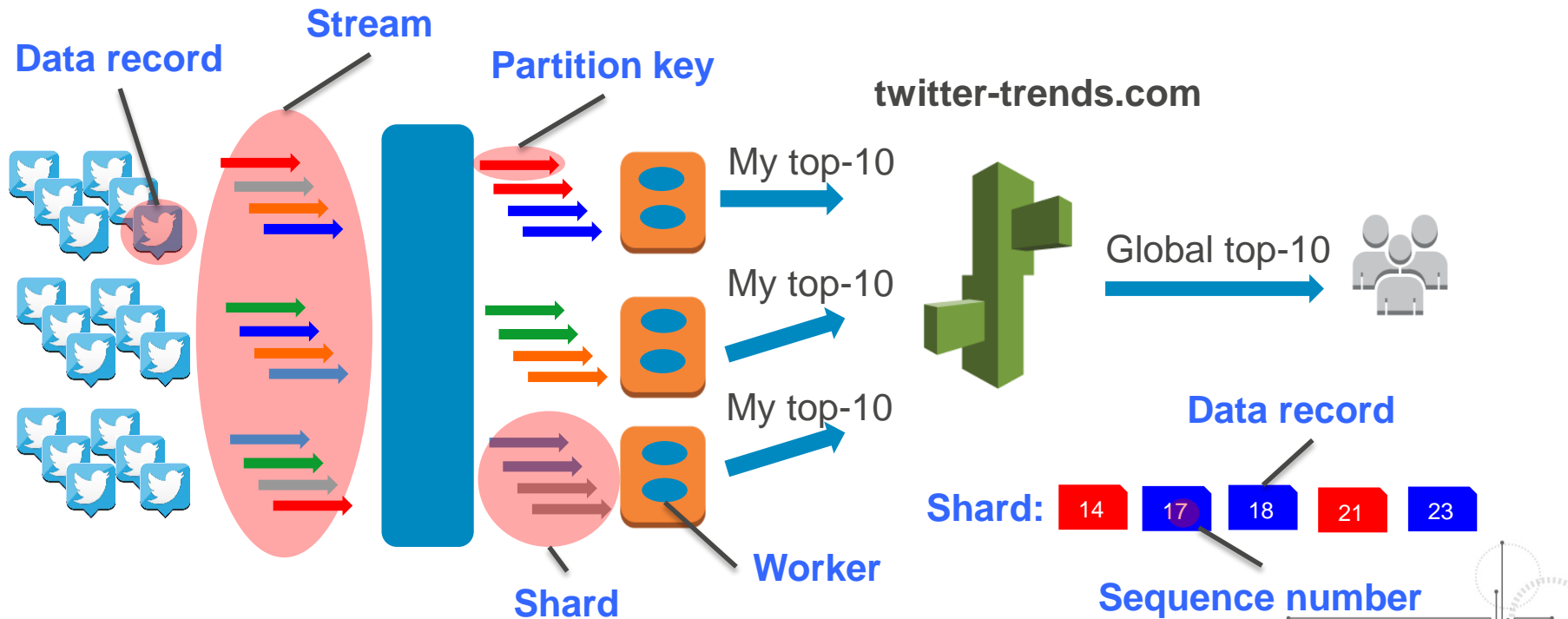




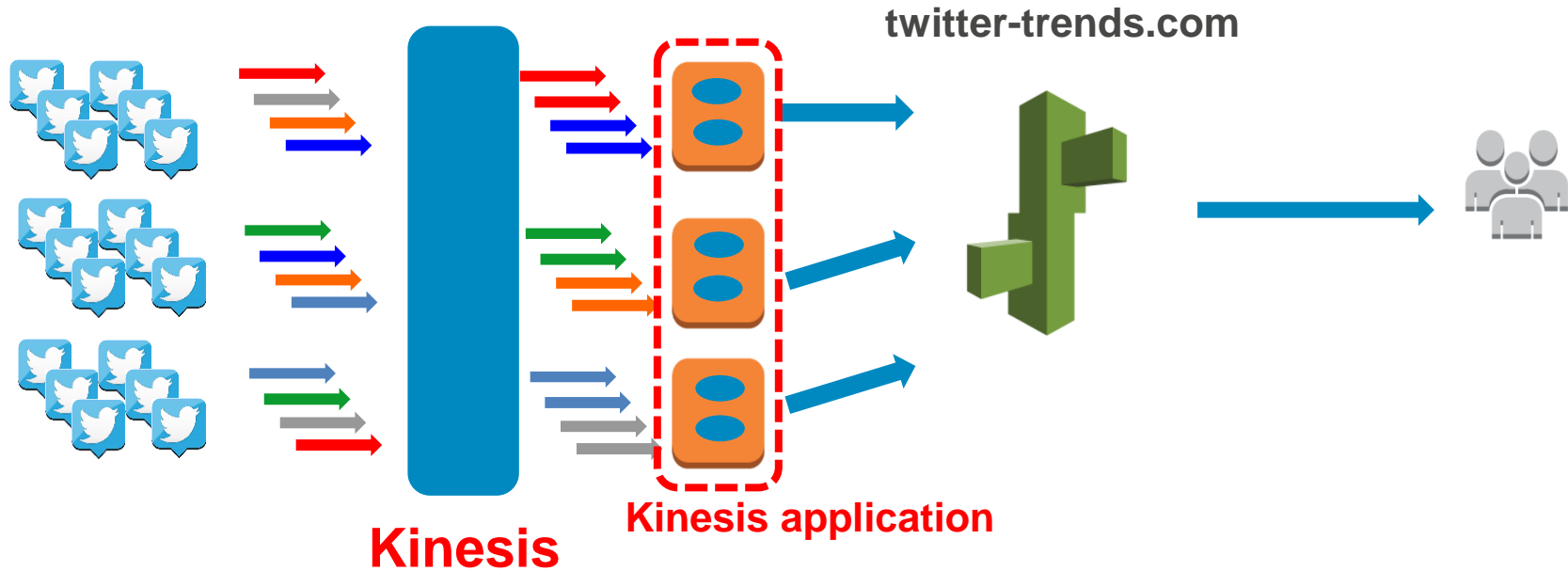
# The solution: streaming map/reduce



# Core concepts

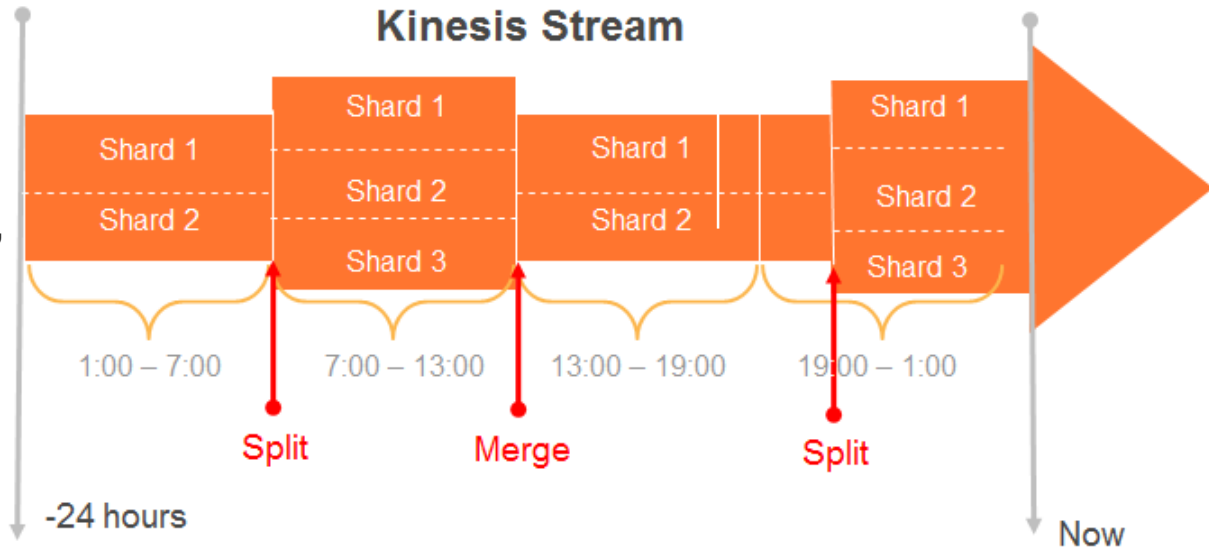


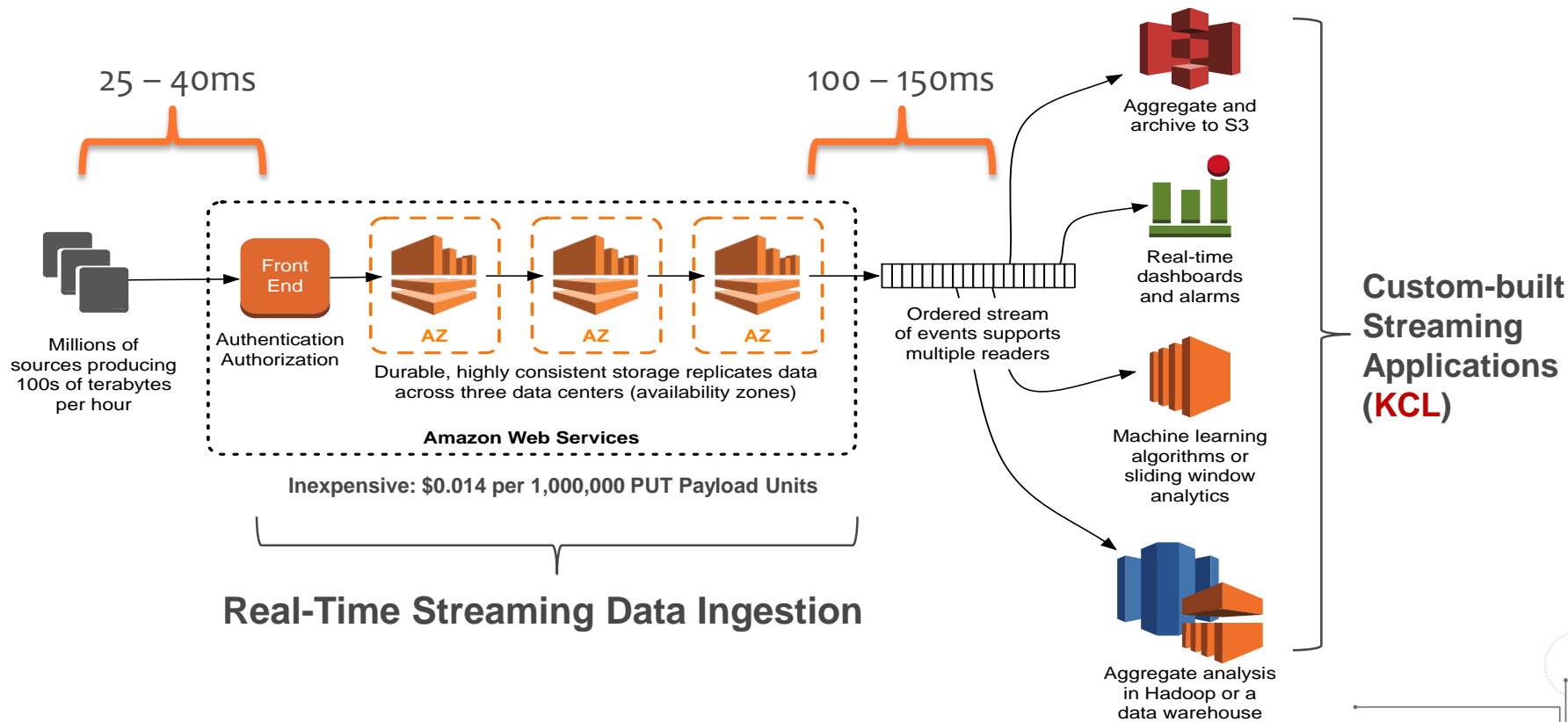
# How this relates to Kinesis



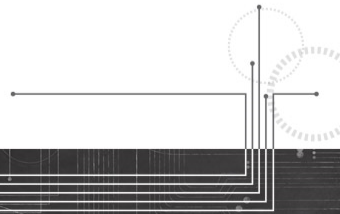
# Kinesis Streaming Data Ingestion

- Streams are made of **Shards**
- Each Shard ingests data up to 1MB/sec, and up to 1000 TPS
- Producers use a PUT call to store data in a Stream: `PutRecord {Data, PartitionKey, StreamName}`
- Each Shard emits up to 2 MB/sec
- All data is stored for **24 hours, 7 days** if extended retention is 'ON'
- Scale** Kinesis streams by adding or removing Shards
- Replay** data from retention period

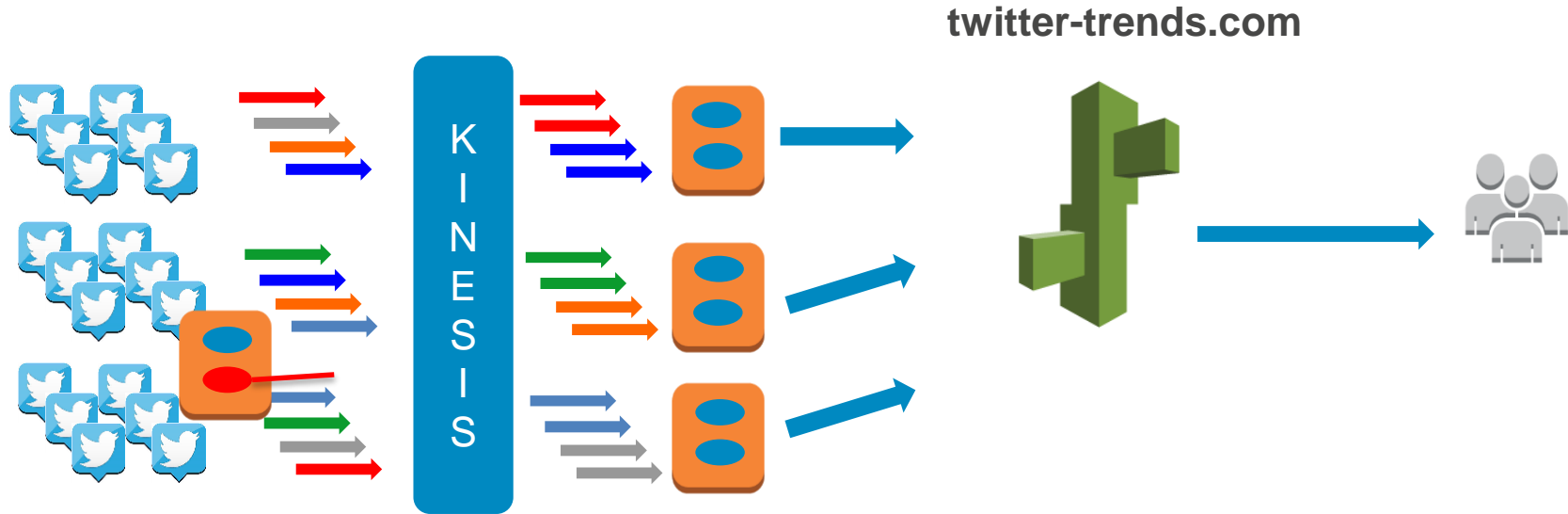




# Kinesis Client Library

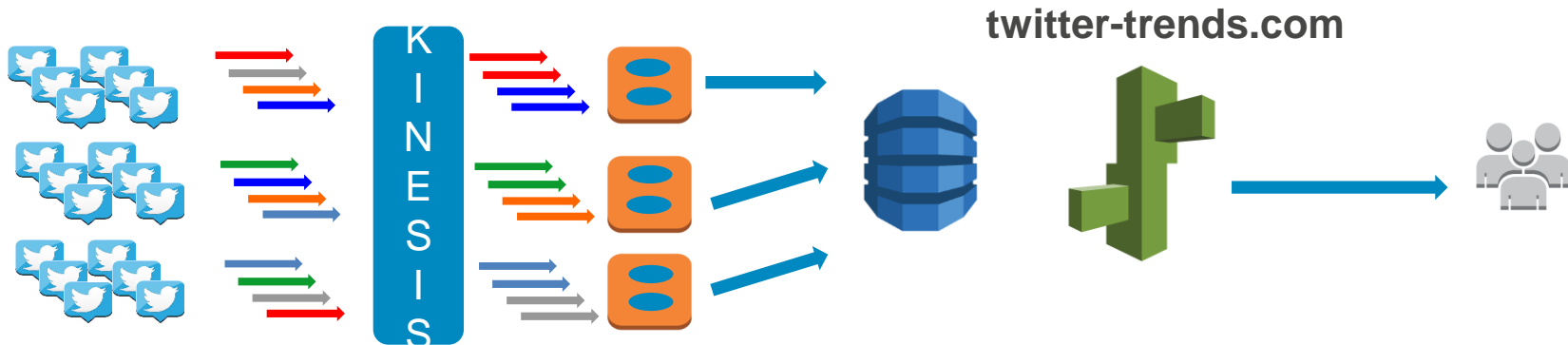


# Using the Kinesis API directly





# Using the Kinesis API directly

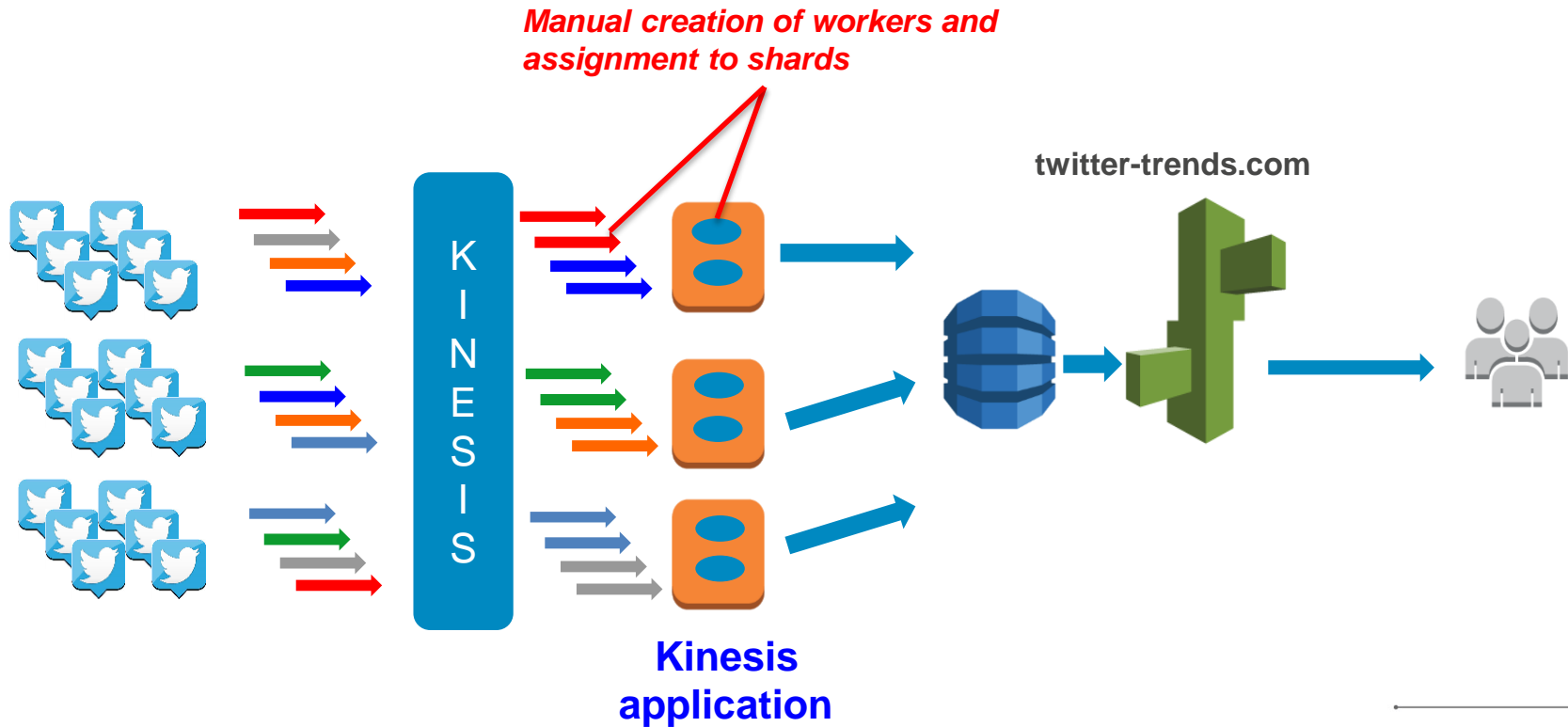


```
iterator = getShardIterator(shardId, LATEST);
while (true) {
    [records, iterator] =
        getNextRecords(iterator, maxRecsToReturn);
    process(records);
}
```

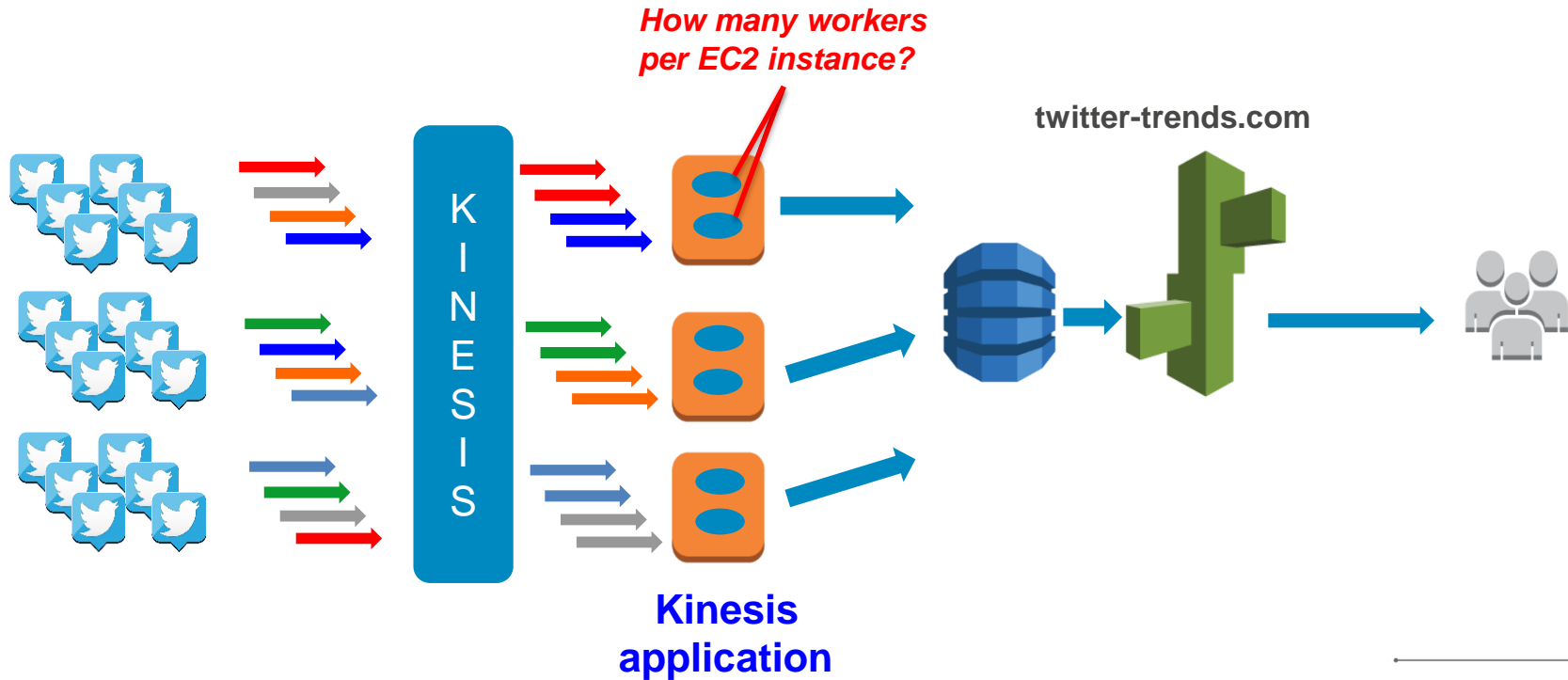
```
process(records): {
    for (record in records) {
        updateLocalTop10(record);
    }
    if (timeToDoOutput()) {
        writeLocalTop10ToDDB();
    }
}
```

```
while (true) {
    localTop10Lists =
        scanDDBTable();
    updateGlobalTop10List(
        localTop10Lists);
    sleep(10);
}
```

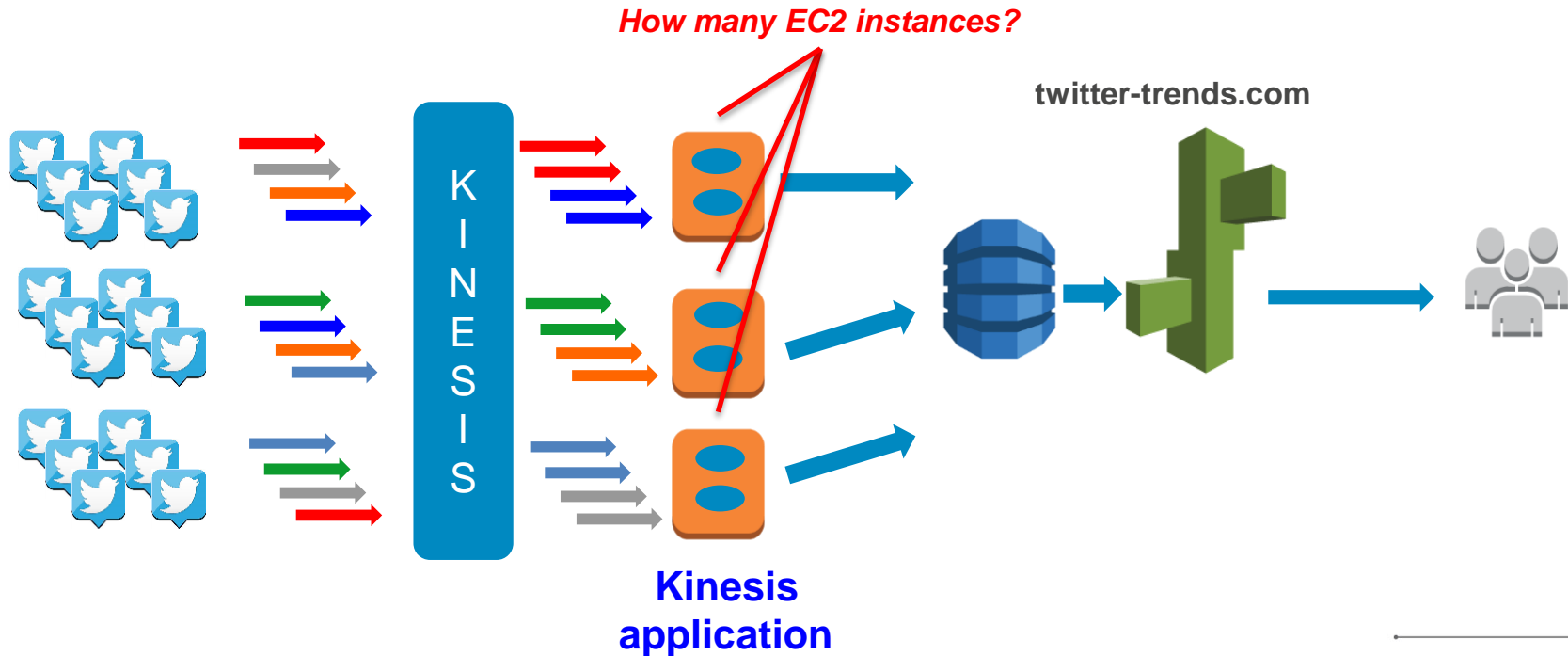
# Challenges with using the Kinesis API directly



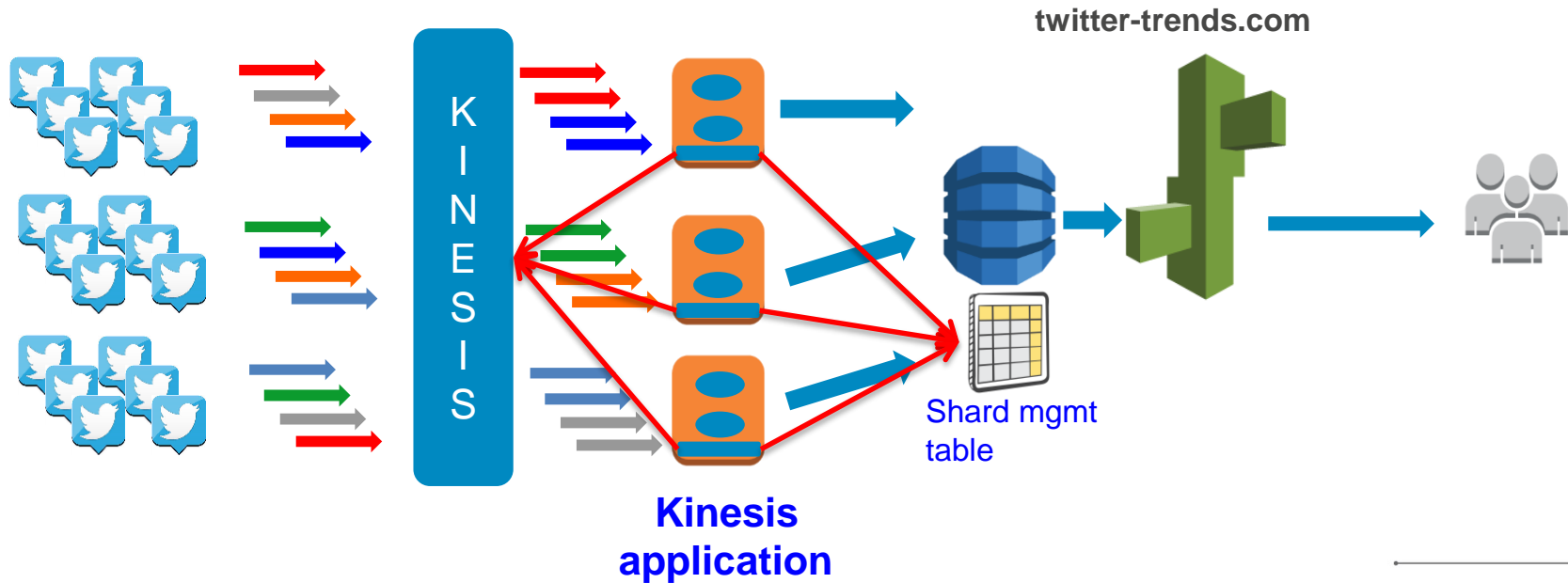
# Challenges with using the Kinesis API directly



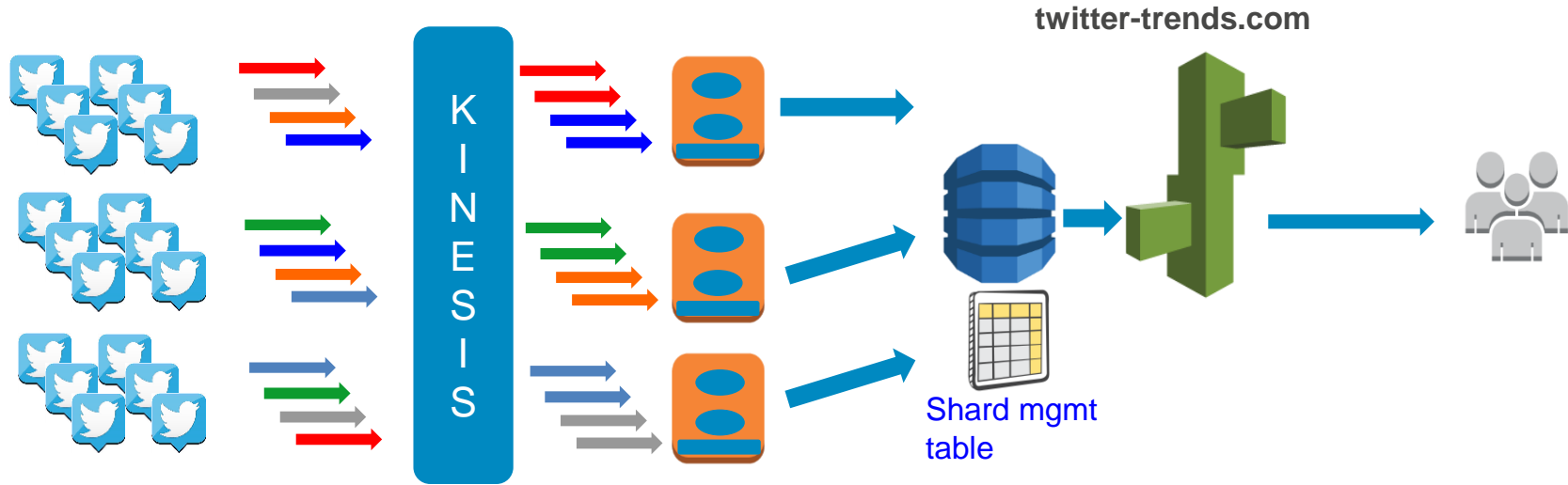
# Challenges with using the Kinesis API directly



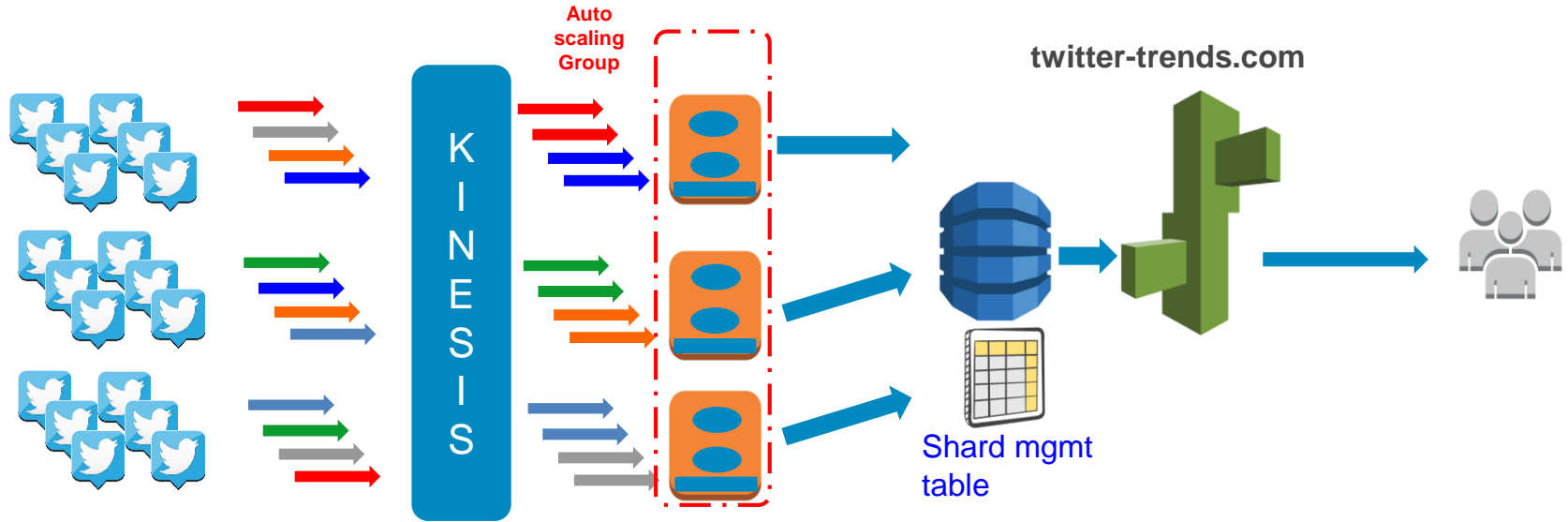
# Using the Kinesis Client Library



# Elasticity and Load Balancing

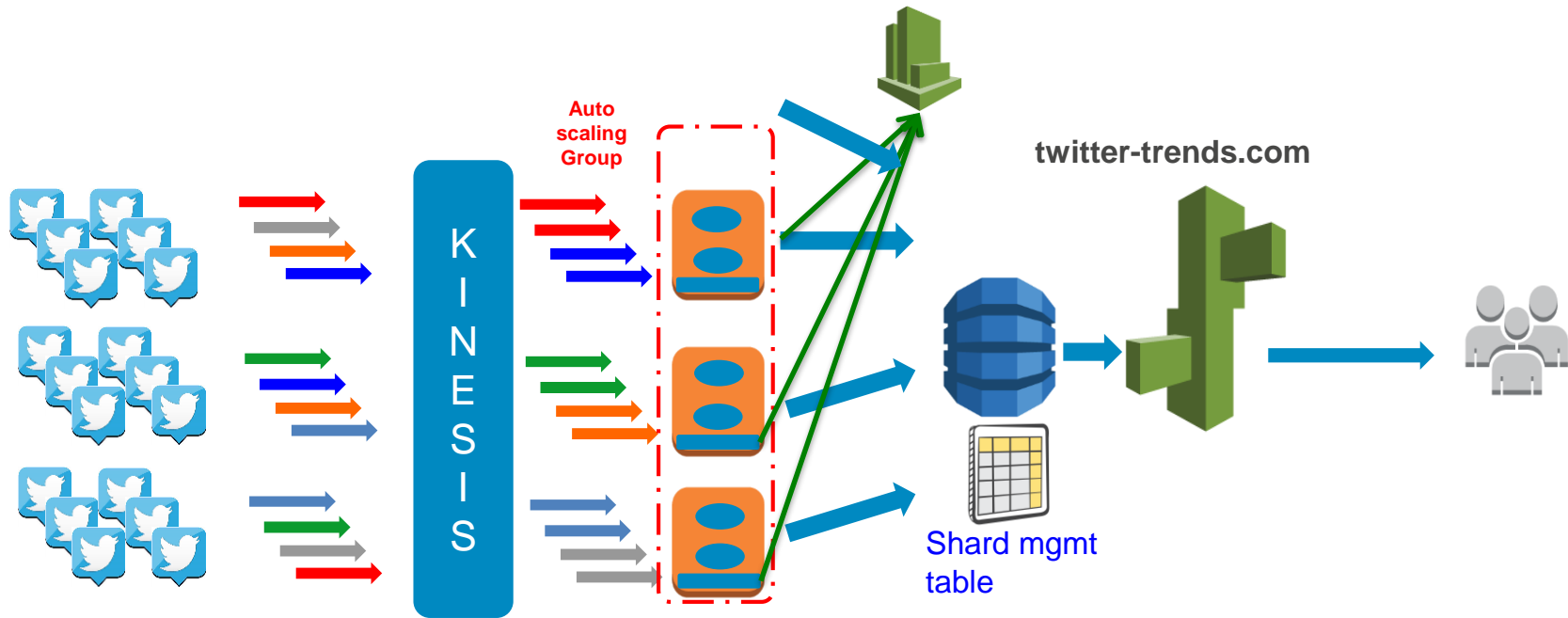


# Elasticity and Load Balancing

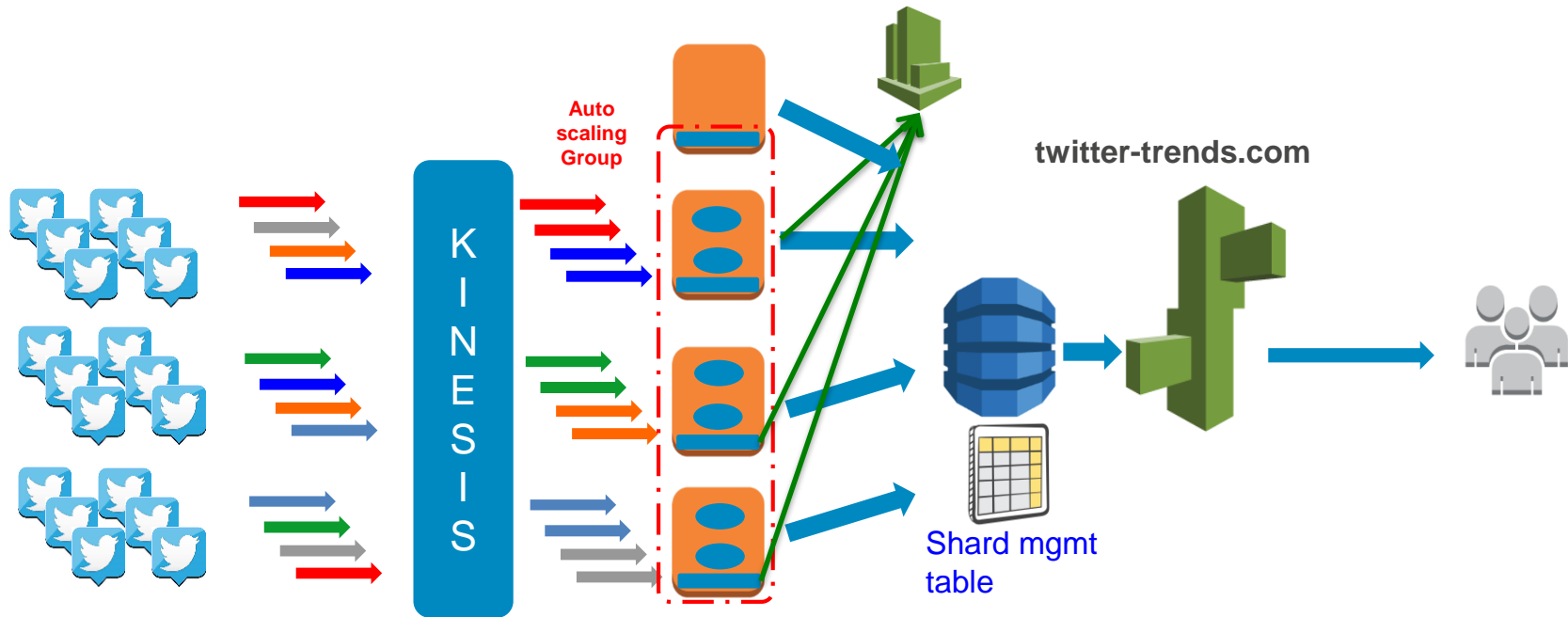




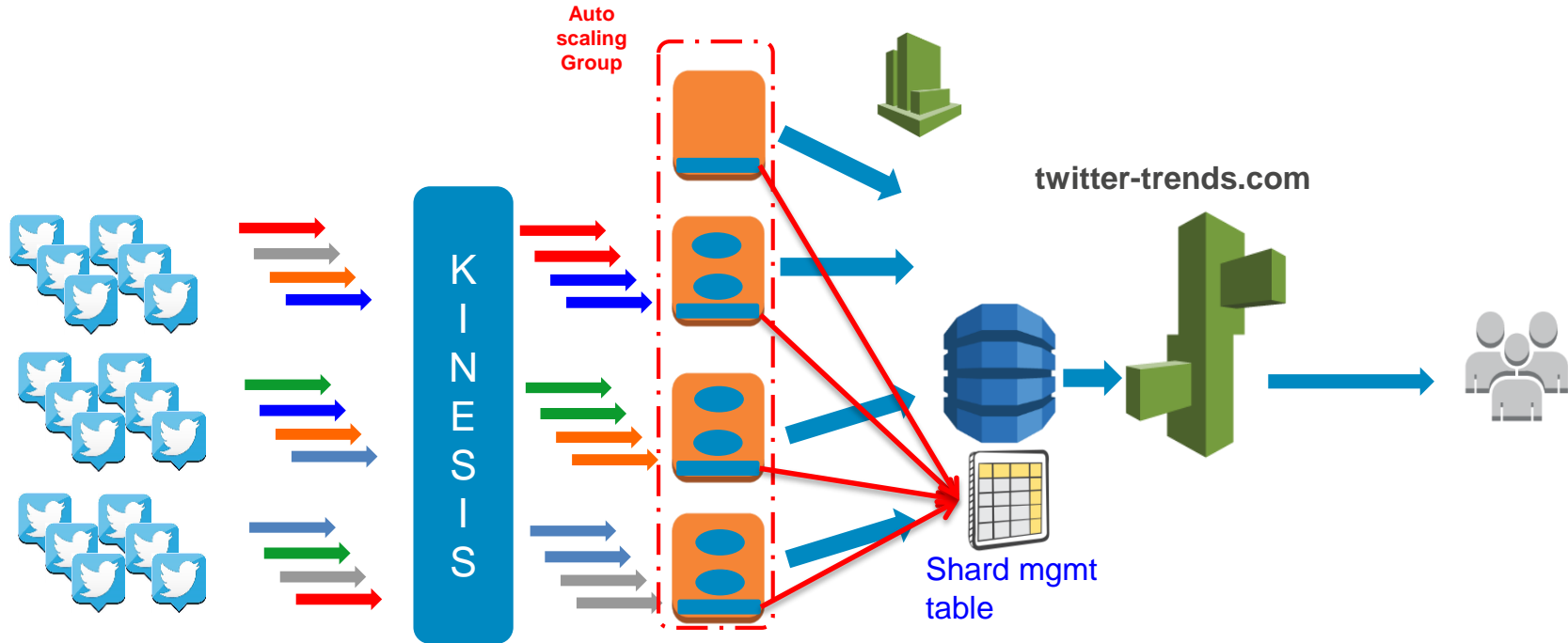
# Elasticity and Load Balancing



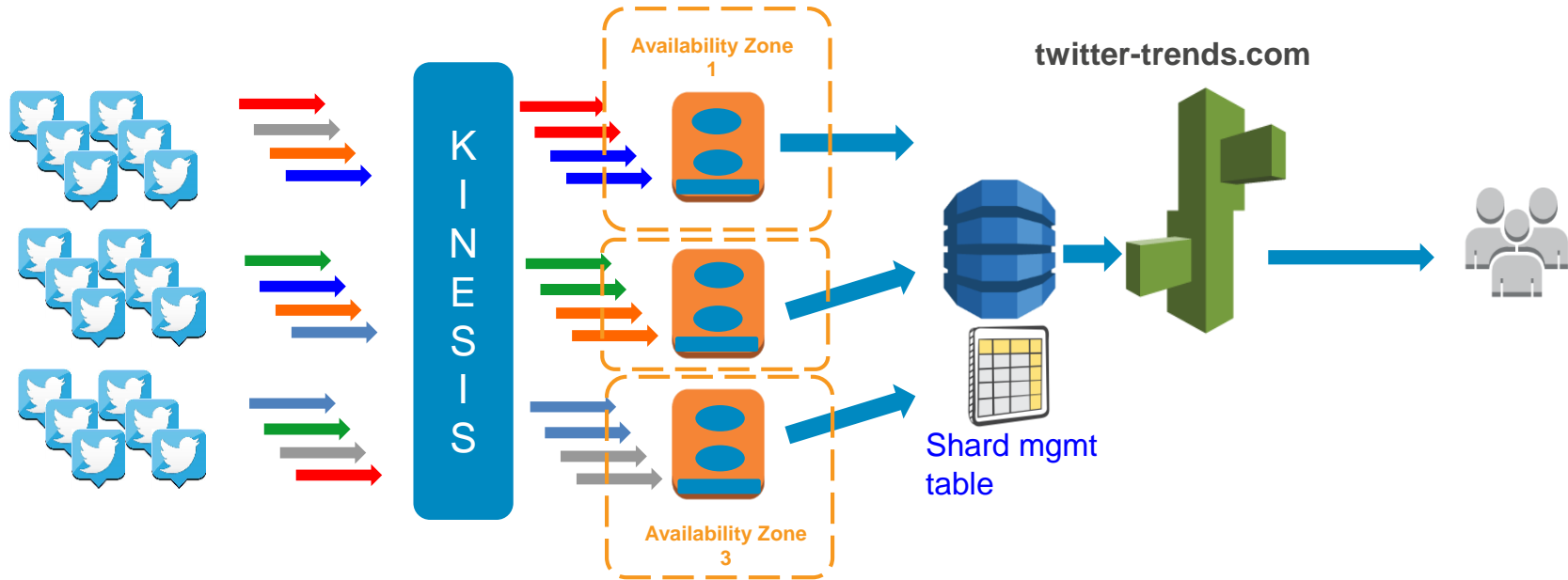
# Elasticity and Load Balancing



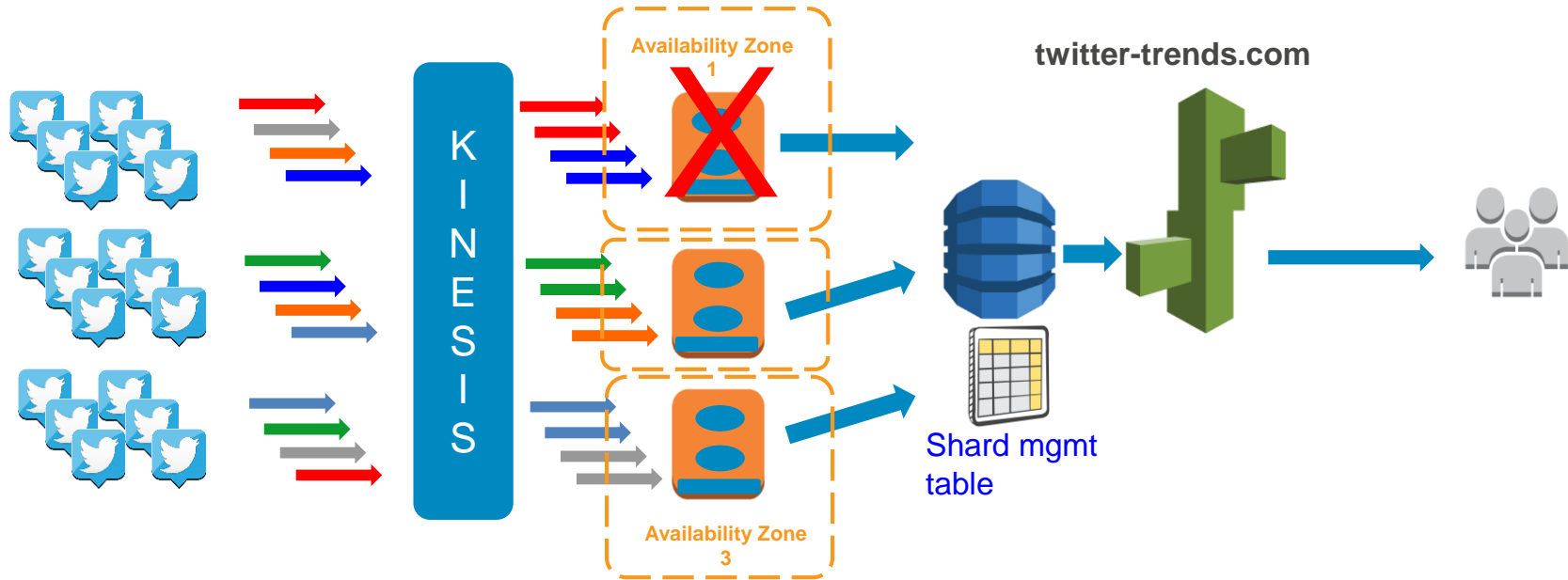
# Elasticity and Load Balancing



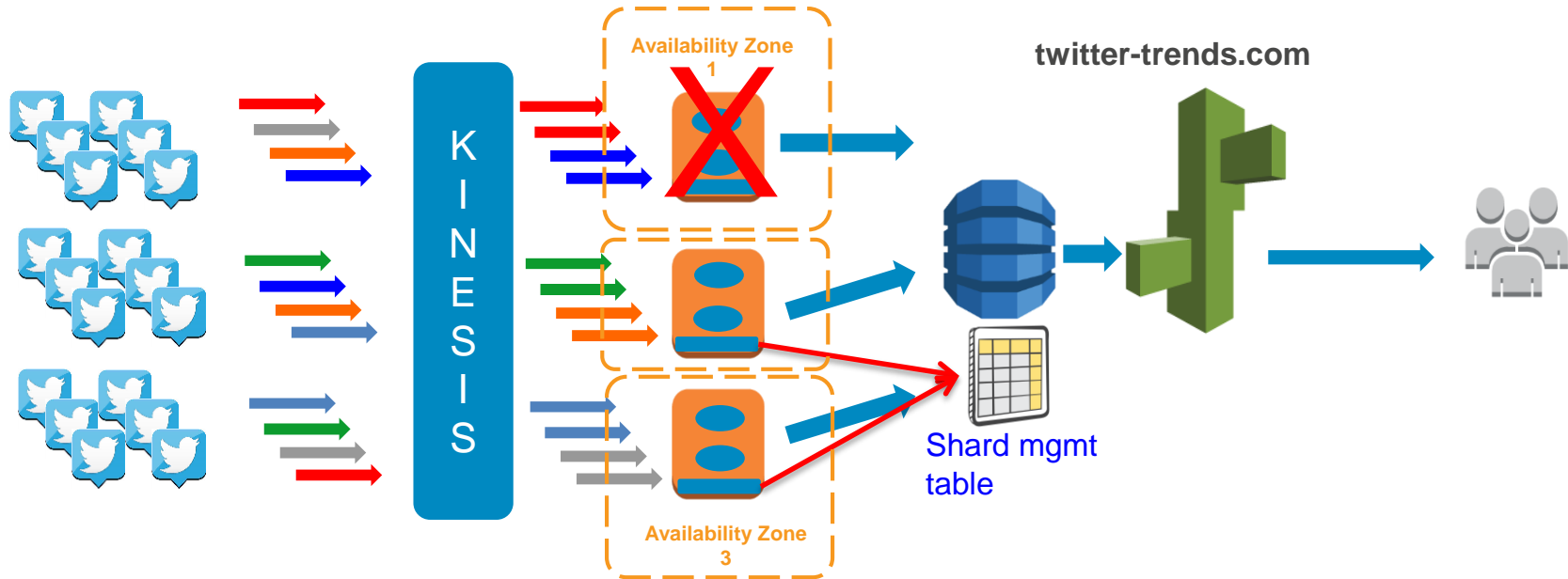
# Fault Tolerance Support



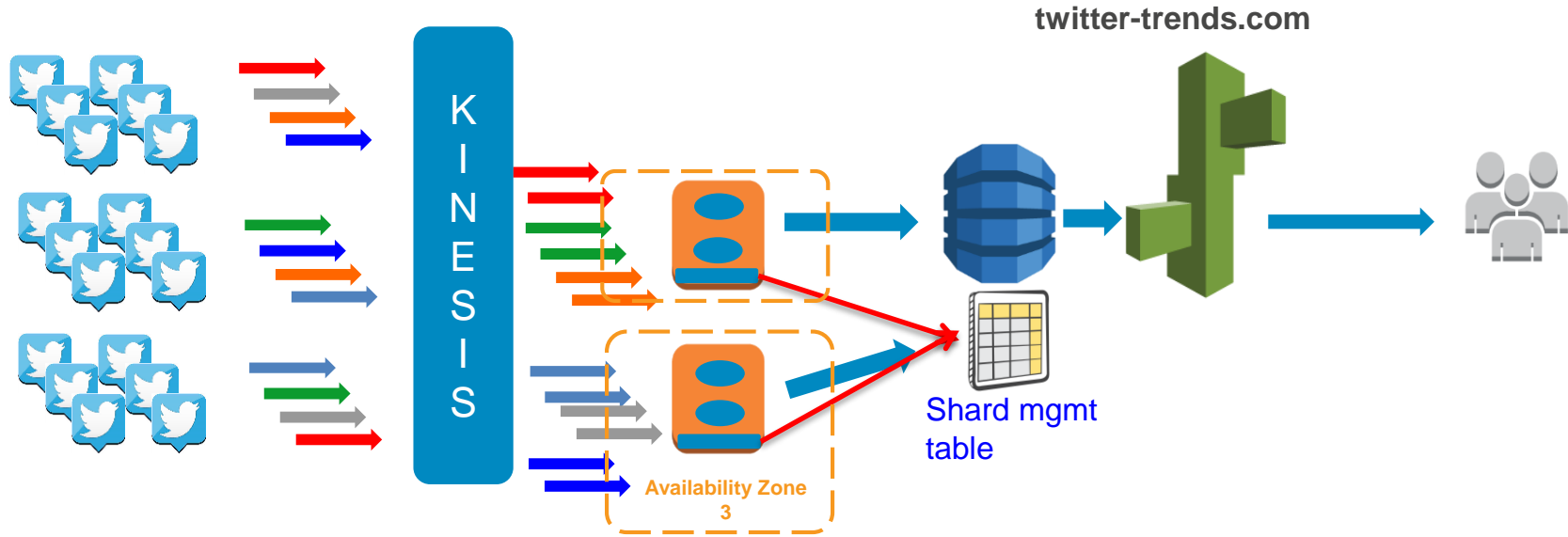
# Fault Tolerance Support



# Fault Tolerance Support

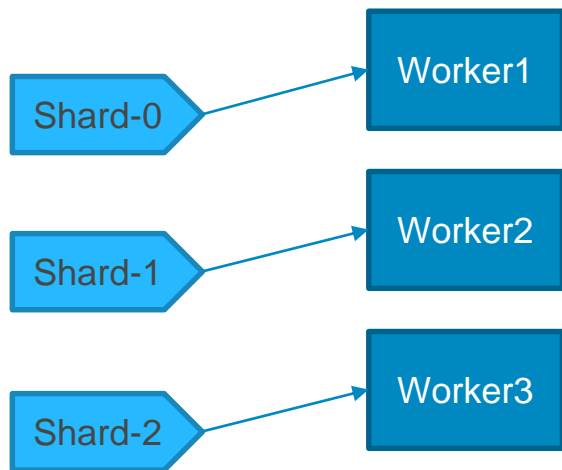


# Fault Tolerance Support

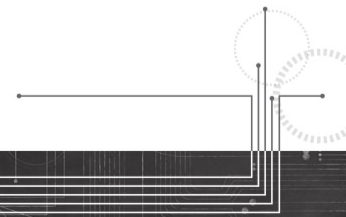




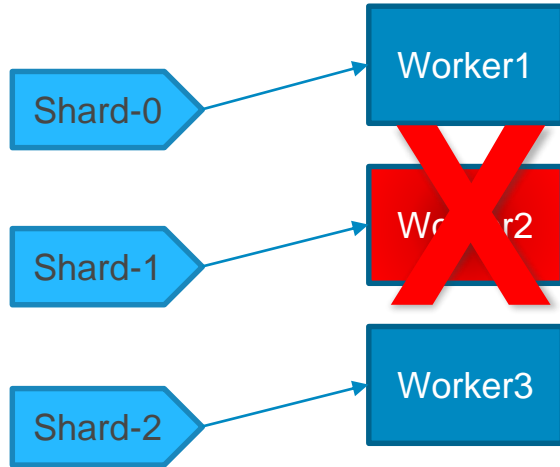
# Worker Fail Over



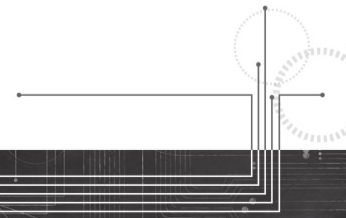
LeaseKey	LeaseOwner	LeaseCounter
Shard-0	Worker1	85
Shard-1	Worker2	94
Shard-2	Worker3	76



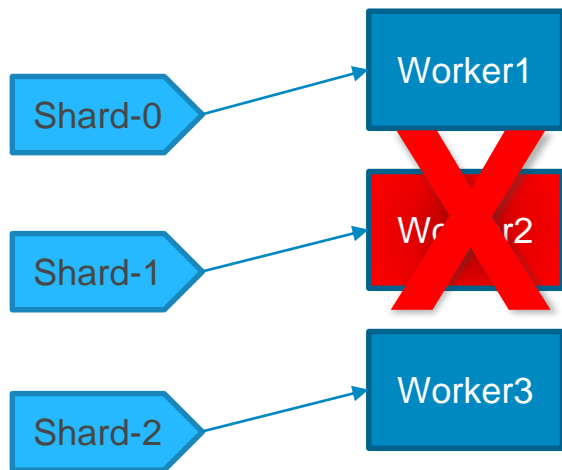
# Worker Fail Over



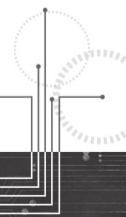
LeaseKey	LeaseOwner	LeaseCounter
Shard-0	Worker1	85 86
Shard-1	Worker2	94
Shard-2	Worker3	76 77



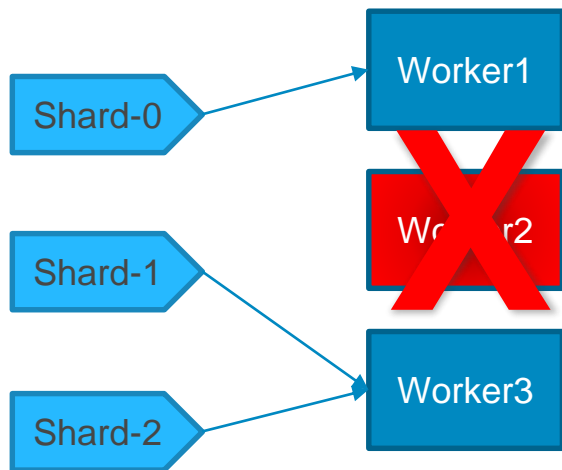
# Worker Fail Over



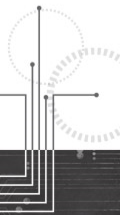
LeaseKey	LeaseOwner	LeaseCounter
Shard-0	Worker1	85 86 87
Shard-1	Worker2	94
Shard-2	Worker3	76 77 78



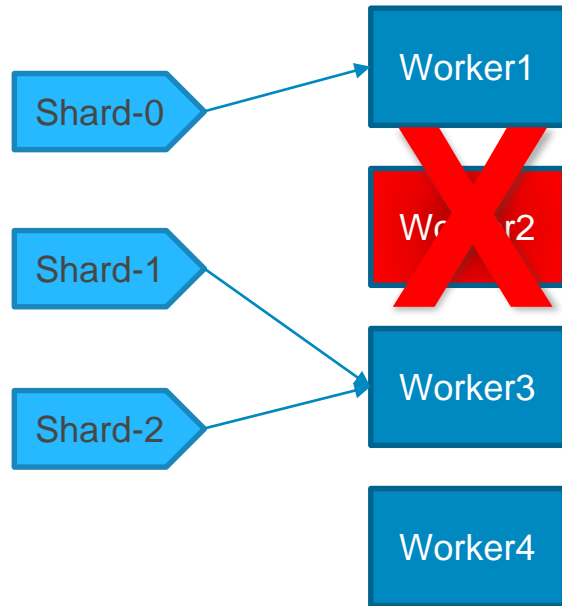
# Worker Fail Over



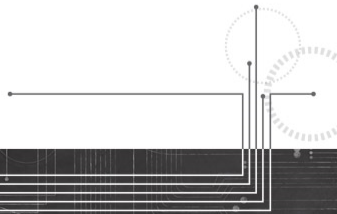
LeaseKey	LeaseOwner	LeaseCounter
Shard-0	Worker1	85 86 87 88
Shard-1	Worker3	94 95
Shard-2	Worker3	76 77 78 79



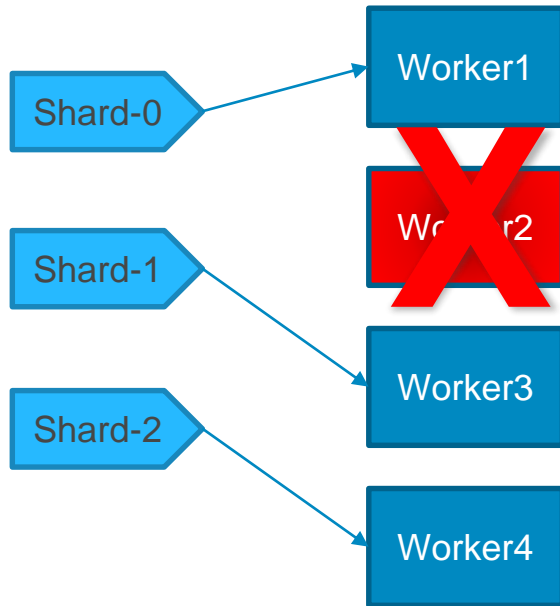
# Worker Load Balancing



LeaseKey	LeaseOwner	LeaseCounter
Shard-0	Worker1	88
Shard-1	Worker3	96
Shard-2	Worker3	78



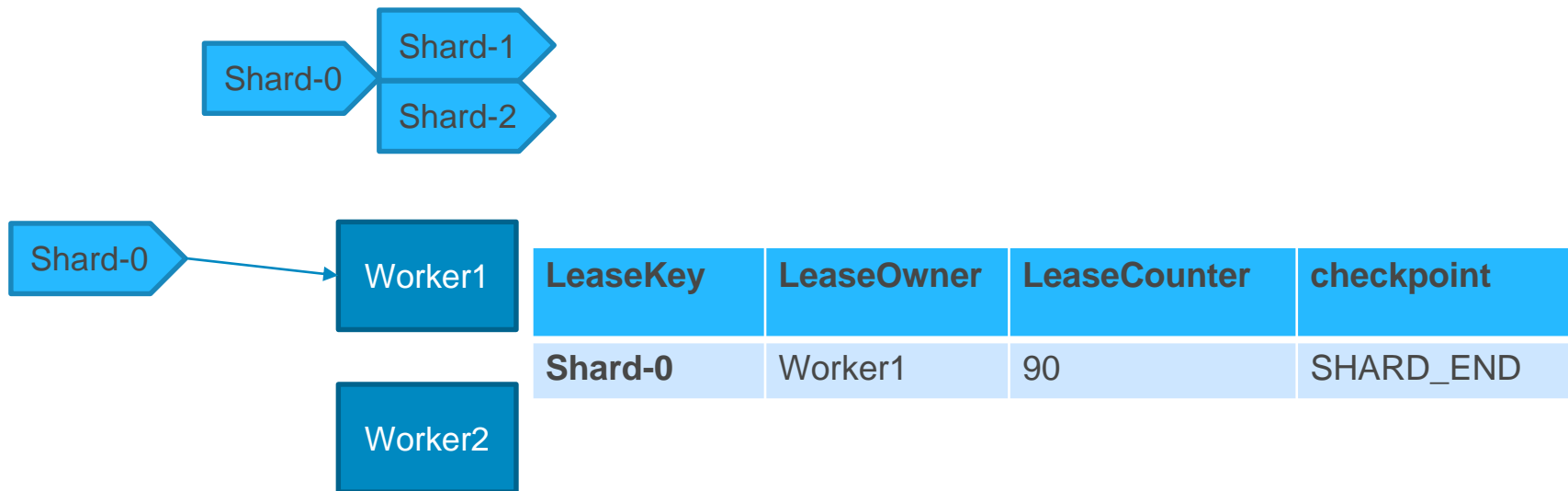
# Worker Load Balancing



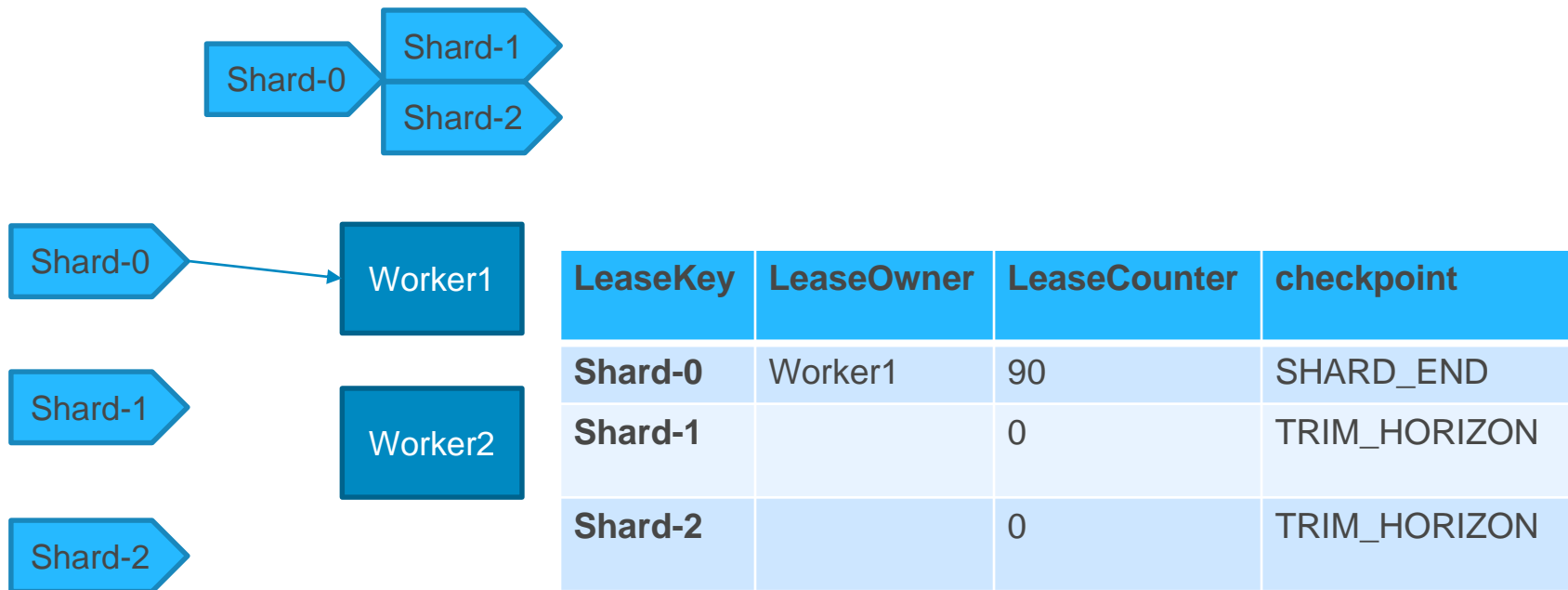
LeaseKey	LeaseOwner	LeaseCounter
Shard-0	Worker1	88
Shard-1	Worker3	96
Shard-2	Worker4	79



# Resharding

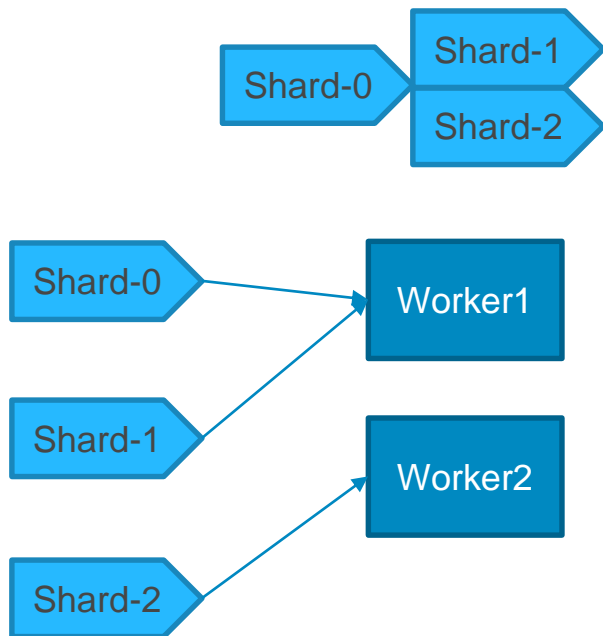


# Resharding



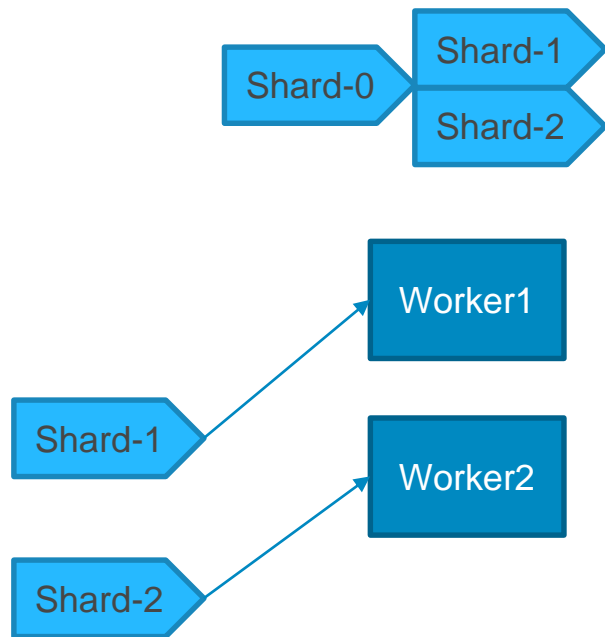


# Resharding



LeaseKey	LeaseOwner	LeaseCounter	checkpoint
Shard-0	Worker1	90	SHARD_END
Shard-1	Worker1	2	TRIM_HORIZON
Shard-2	Worker2	3	TRIM_HORIZON

# Resharding



LeaseKey	LeaseOwner	LeaseCounter	checkpoint
Shard-1	Worker1	2	TRIM_HORIZON
Shard-2	Worker2	3	TRIM_HORIZON

# Putting this into production

## Cost & Scale

500MM tweets/day = ~ 5,800 tweets/sec

2k/tweet is ~12MB/sec (~1TB/day)

\$0.015/hour per shard, \$0.014/million PUTS

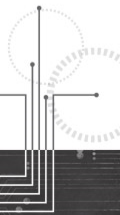
Kinesis cost is \$0.47/hour

Redshift cost is \$0.850/hour (for a 2TB node)

Total: **\$1.32/hour**

# Design Challenge(s)

- Dynamic Resharding & Scale Out
- Enforcing Quotas (think proxy fleet with 1Ks servers)
- Distributed Denial of Service Attack (unintentional)
- Dynamic Load Balancing on Storage Servers
- Heterogeneous Workloads (tip of stream vs 7 day)
- Optimizing Fleet Utilization (proxy, control, data planes)
- Avoid Scaling Cliffs
- ...



# Kinesis Streams: Streaming Data the AWS Way



- Easy to provision, deploy, and manage



- Elastically scalable



- Real-time latencies



- Choose the service, or combination of services, for your specific use cases.



- Pay as you go, no up front costs

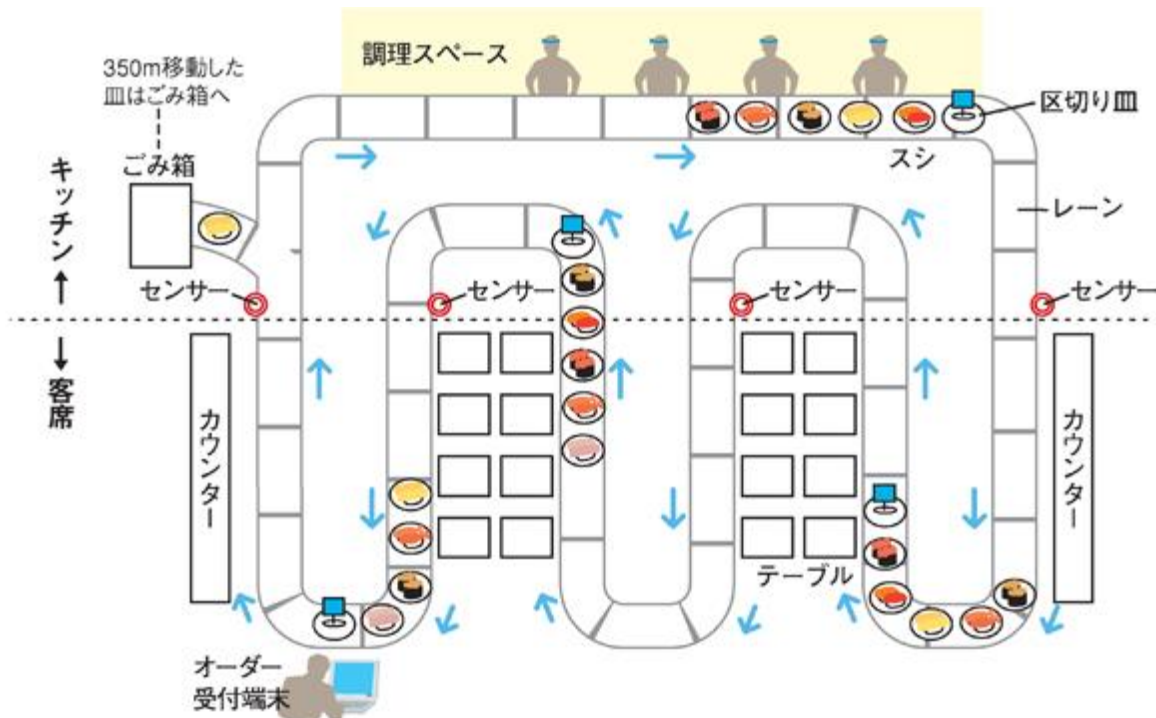


Amazon Kinesis

# Sushiro: Kaiten Sushi Restaurants



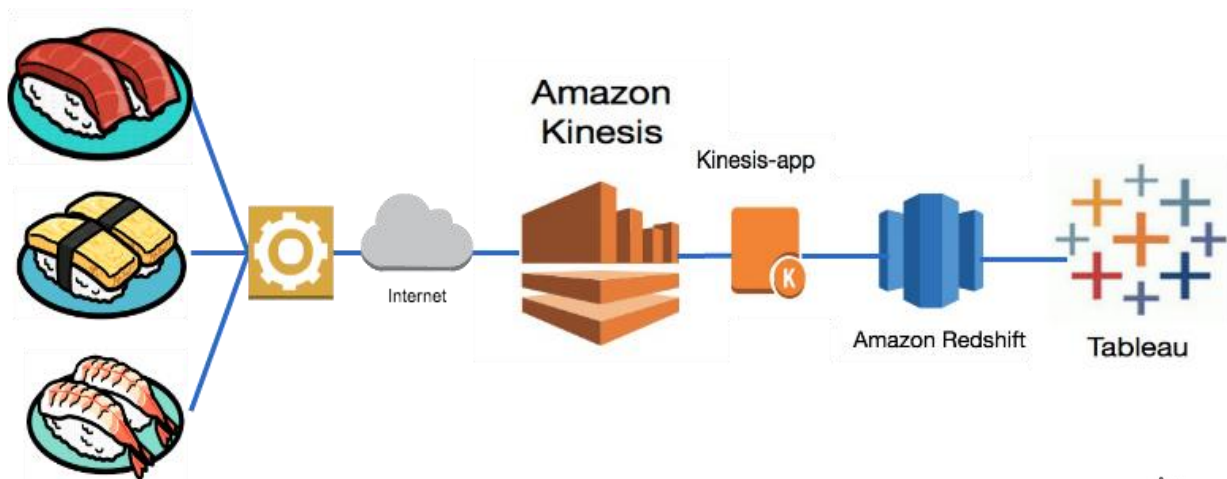
380 stores stream data from sushi plate sensors and stream to Kinesis



# Sushiro: Kaiten Sushi Restaurants



380 stores stream data from sushi plate sensors and stream to Kinesis





# Real-Time Streaming Data with Kinesis Streams



1 TB+/day game data  
analyzed in real-time  
| **Gaming**



5 billion events/wk from  
connected devices | **IoT**



17 PB of game data per  
season | **Entertainment**



100 billion ad  
impressions/day, 30 ms  
response time | **Ad Tech**



100 GB/day click streams  
250+ sites | **Enterprise**



50 billion ad  
impressions/day sub-50  
ms responses | **Ad Tech**



17 million events/day  
| **Technology**



1 billion transactions per  
day | **Bitcoin**



---

*Streams provide a **foundational abstraction** on which to build higher level services*

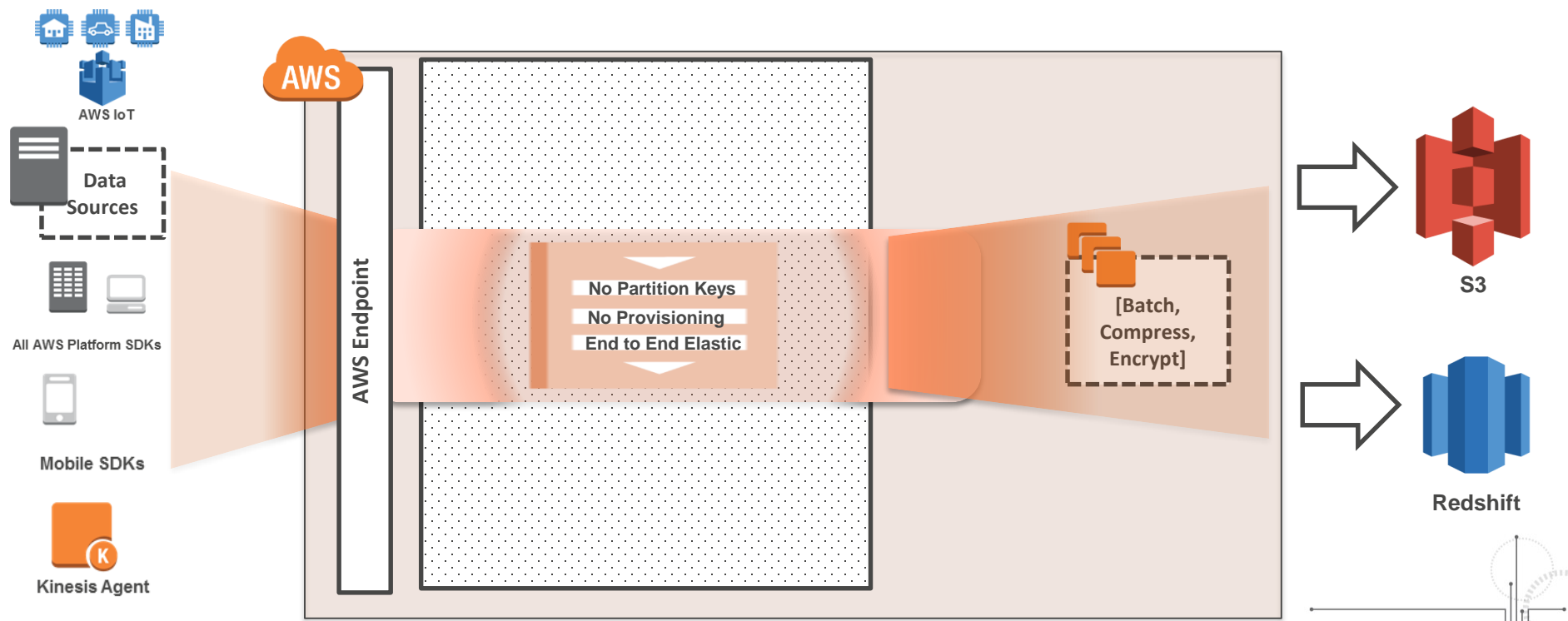
# Amazon Kinesis Firehose



- **Zero Admin:** Capture and deliver streaming data into S3, Redshift, and other destinations **without writing an application** or **managing infrastructure**
- **Direct-to-data store integration:** **Batch**, **compress**, and **encrypt** streaming data for delivery into S3, and other destinations **in as little as 60 secs**, set up in minutes
- **Seamless elasticity:** **Seamlessly scales** to match data throughput

# Amazon Kinesis Firehose

Fully Managed Service for Delivering Data Streams into AWS Destinations



# Closing Thoughts

---

Streaming data is highly prevalent and relevant;

Stream data processing is on the rise;

A key part of business critical workflows today, a powerful abstraction for building a new class of applications & data intensive services tomorrow.

A rich area for distributed systems, programming model, IoT, and new service(s) research.

# Questions

