aws SUMMIT

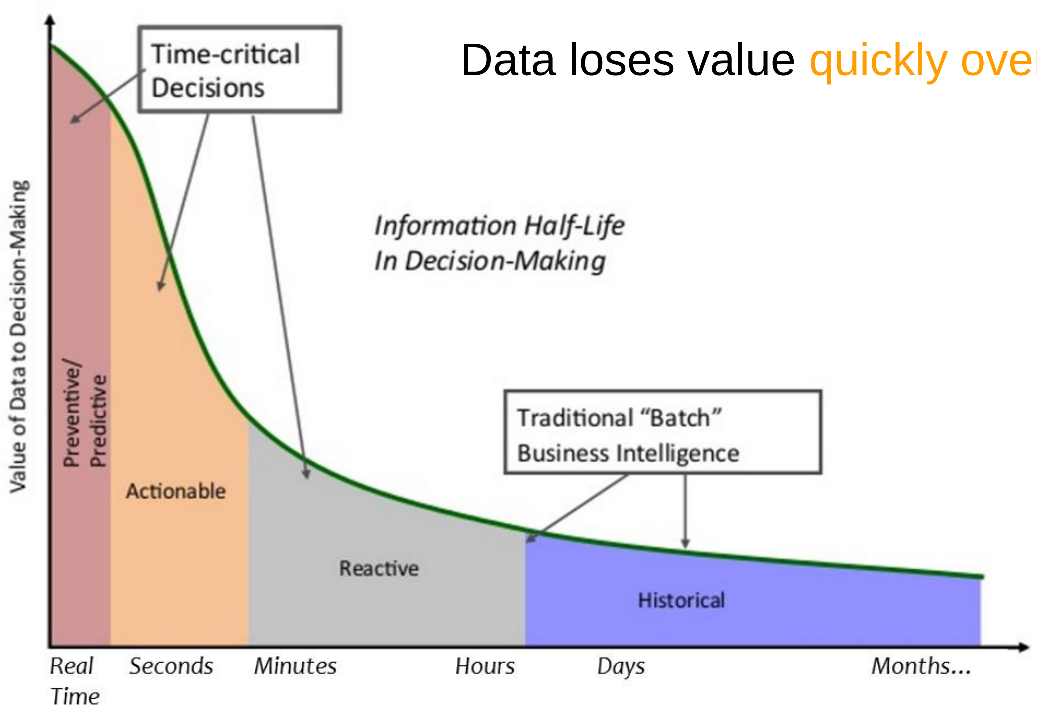
BDA307

# Analyzing Data Streams in Real Time with Amazon Kinesis

Allan MacInnis
Principal Solutions Architect, Amazon Web Services

Milan Brahmbhatt
Zynga

## Timely Decisions Require New Data in Minutes

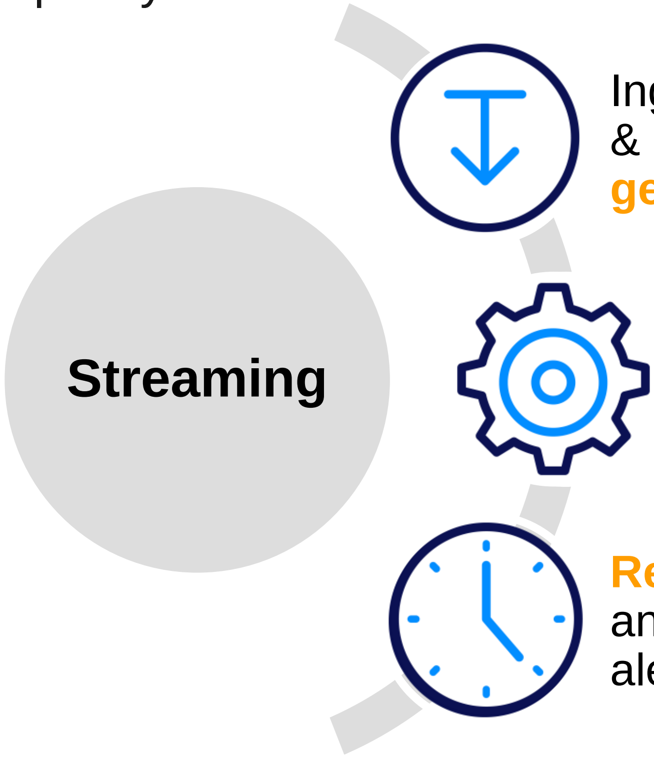


Data loses value quickly over time

Source: Perishable Insights, Mike Gualtieri, Forrester

## Stream New Data in Seconds

Get actionable insights quickly



Ingest video & data **as it's generated**

Process data **on the fly**

**Real-time** analytics/ML, alerts, actions

# Most Common Uses of Streaming

Security monitoring

Industrial automation

Log analytics

Data lakes

IoT device monitoring

# Streaming with Amazon Kinesis

Easily collect, process, and analyze data and video streams in real time

**Kinesis Data Streams**

*Capture, process, and store data streams*

**Kinesis Data Firehose**

*Load data streams into AWS data stores*

**Kinesis Data Analytics**

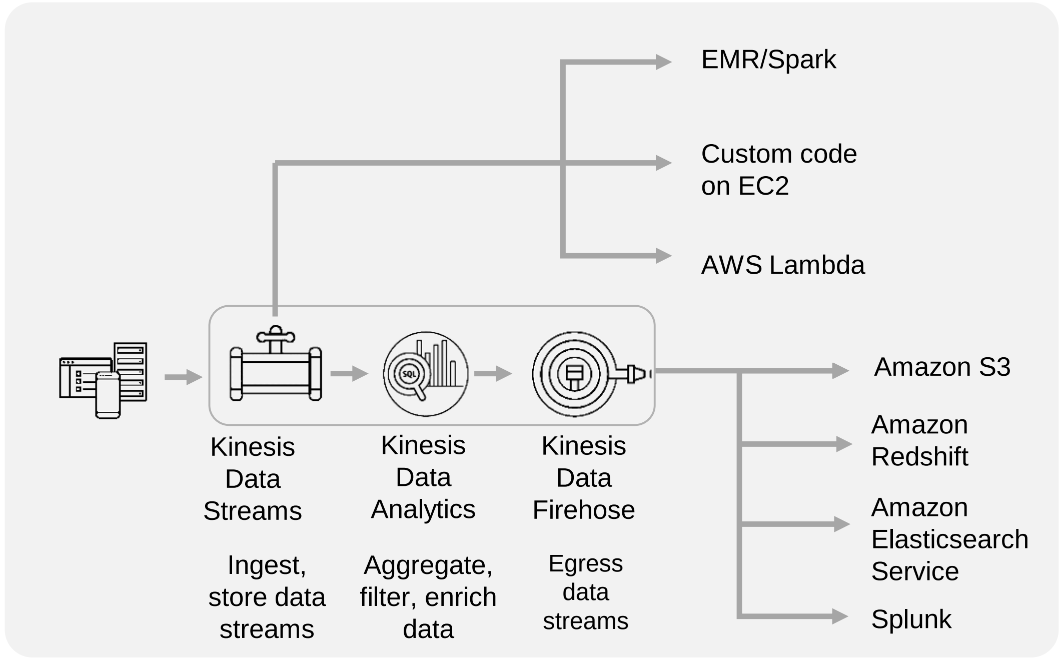
*Analyze data streams with SQL*

**Kinesis Video Streams**

*Capture, process, and store video streams*

# Amazon Kinesis Data Streaming

Collect, process, and analyze data streams in real time



Real time

Fully managed

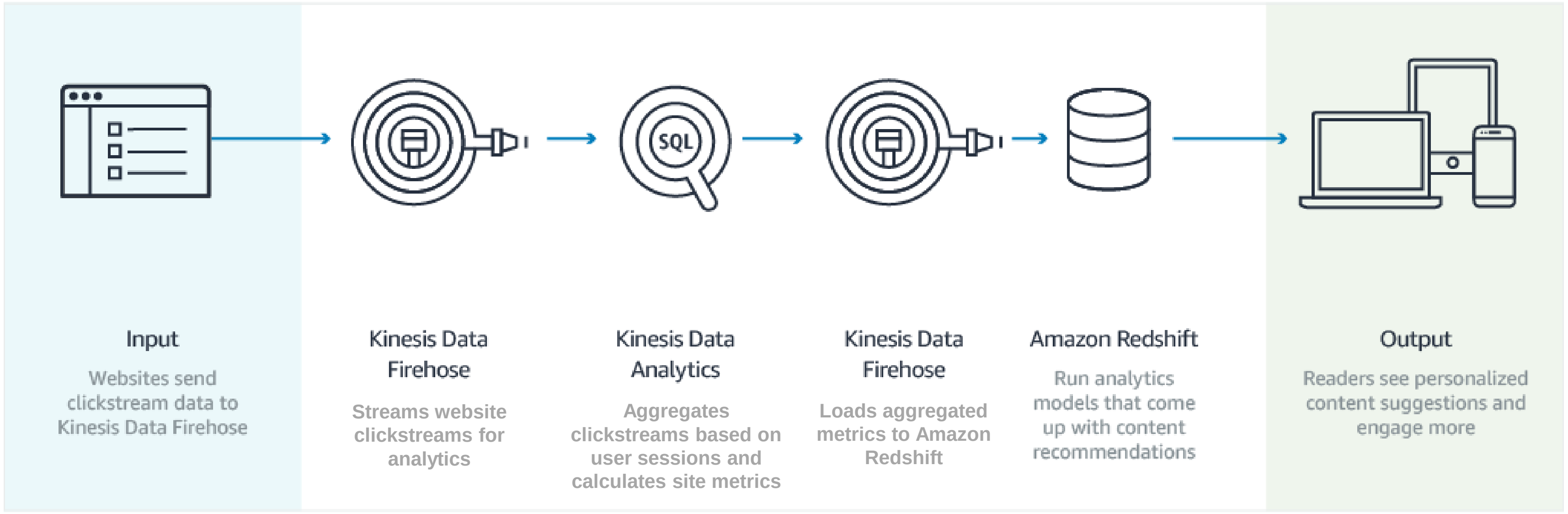
Scalable

Secure

Cost-effective

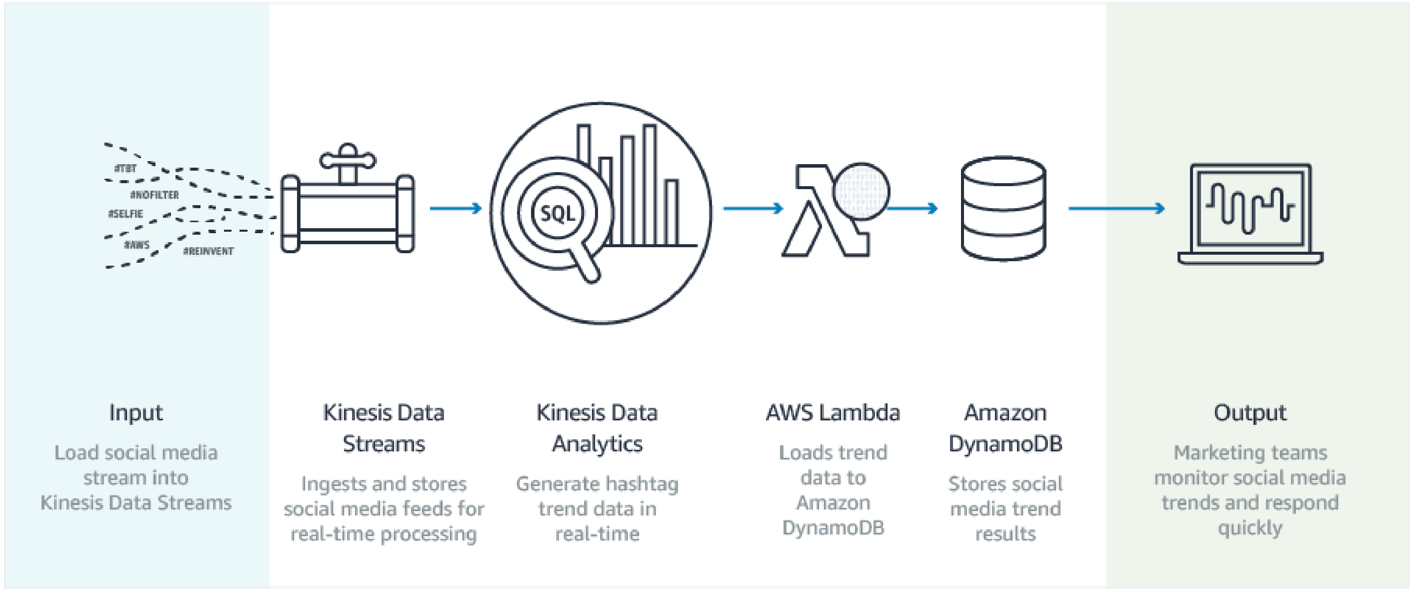
# Use Case 1: Clickstream Analytics

Example: Website content recommendations


Input
Websites send clickstream data to Kinesis Data Firehose
Kinesis Data Firehose
Streams website clickstreams for analytics
SQL
Kinesis Data Analytics
Aggregates clickstreams based on user sessions and calculates site metrics
Kinesis Data Firehose
Loads aggregated metrics to Amazon Redshift
Amazon Redshift
Run analytics models that come up with content recommendations
Output
Readers see personalized content suggestions and engage more

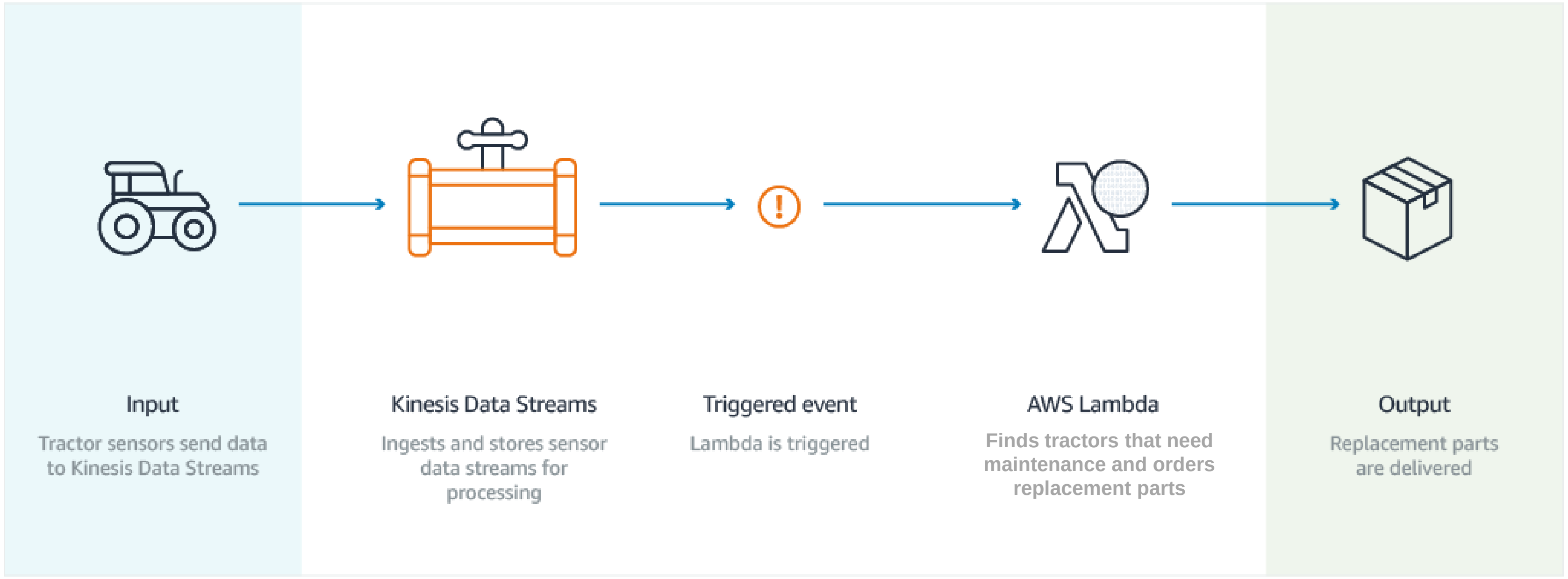
## Use Case 2: Real-Time Analytics

Example: Analyze streaming social media data


#TBT
#NOFILTER
#SELFIE
#AWS
#REINVENT
SQL
Input
Load social media stream into Kinesis Data Streams
Kinesis Data Streams
Ingests and stores social media feeds for real-time processing
Kinesis Data Analytics
Generate hashtag trend data in real-time
AWS Lambda
Loads trend data to Amazon DynamoDB
Amazon DynamoDB
Stores social media trend results
Output
Marketing teams monitor social media trends and respond quickly

# Use Case 3: IoT Stream Processing

Example: Sensors in tractor detect need for a spare part and automatically place order

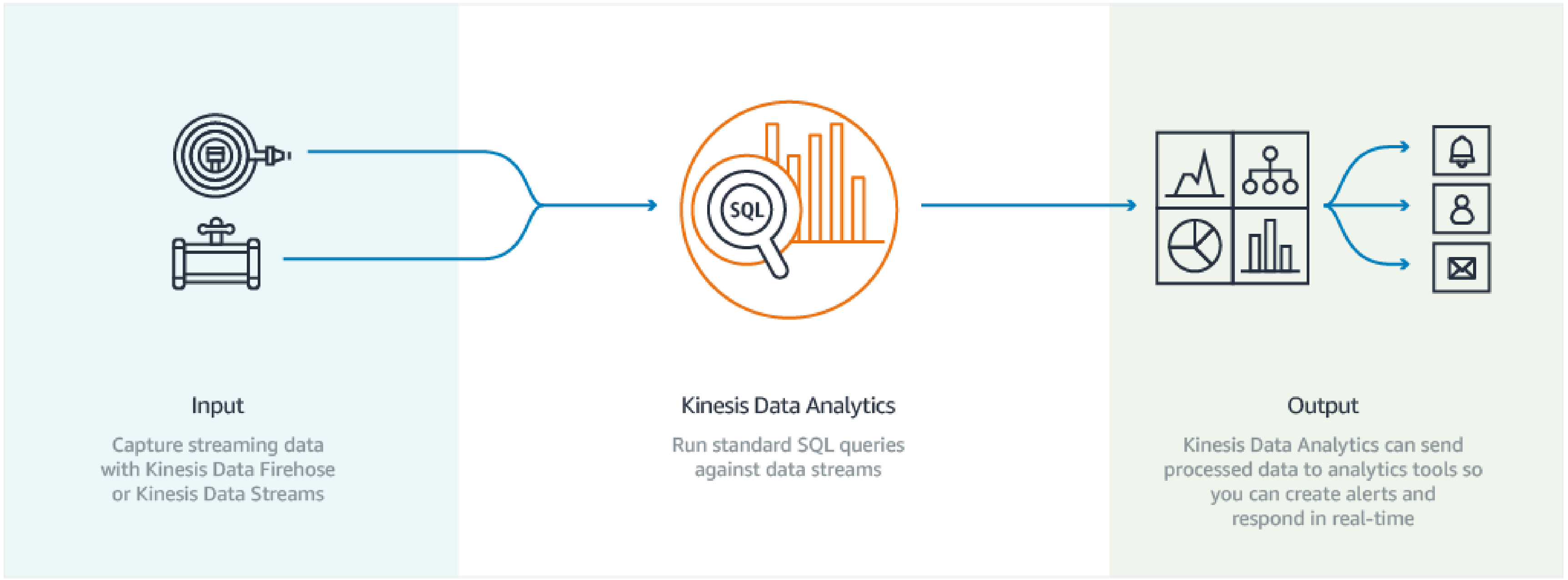

Input
Tractor sensors send data to Kinesis Data Streams
Kinesis Data Streams
Ingests and stores sensor data streams for processing
Triggered event
Lambda is triggered
AWS Lambda
Finds tractors that need maintenance and orders replacement parts
Output
Replacement parts are delivered

## Kinesis Data Analytics Overview

## It’s All about the Pace

| Batch processing | Stream processing |
|---|---|
| Hourly server logs | Real-time metrics |
| Weekly or monthly bills | Real-time spending alerts & caps |
| Daily website clickstream | Real-time clickstream analysis |
| Daily fraud reports | Real-time detection |

# Amazon Kinesis Data Analytics


Input
Capture streaming data with Kinesis Data Firehose or Kinesis Data Streams
SQL
Kinesis Data Analytics
Run standard SQL queries against data streams
Output
Kinesis Data Analytics can send processed data to analytics tools so you can create alerts and respond in real-time

# Simple Pattern for Streaming Data

## Data producer

Continuously creates data

Continuously writes data to a stream

Can be almost anything

**Mobile client**

## Streaming service

Durably stores data

Provides temporary buffer that preps data

Supports very high-throughput

**Kinesis**

## Data consumer

Continuously processes data

Cleans, prepares, & aggregates

Transforms data to information

**Amazon Kinesis app**

## Kinesis Data Analytics Applications

Connect to streaming source

Easily write SQL code to process streaming data

Continuously deliver SQL results

## How do I write streaming SQL? Easy!

### Streams (in memory tables)

```
CREATE STREAM calls_per_ip_stream(
    eventTimeStamp TIMESTAMP,
    computationType VARCHAR(256),
    category VARCHAR(1024),
    subCategory VARCHAR(1024),
    unit VARCHAR(256),
    unitValue BIGINT
);
```

## How do I write streaming SQL? Easy!

### Pumps (continuous query)

```
CREATE OR REPLACE PUMP calls_per_ip_pump AS
INSERT INTO calls_per_ip_stream
SELECT STREAM "eventTimestamp",
    COUNT(*),
    "sourceIPAddress"
FROM source_sql_stream_001 ctrail
GROUP BY "sourceIPAddress",
    STEP(ctrail.ROWTIME BY INTERVAL '1' MINUTE),
    STEP(ctrail."eventTimestamp" BY INTERVAL '1' MINUTE);
```

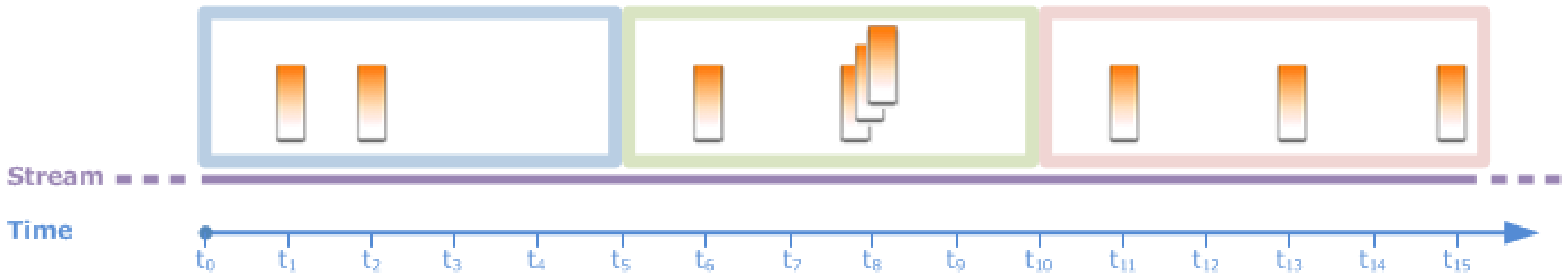
## How do we aggregate streaming data?

- Aggregations (count, sum, min, etc.) take granular, real-time data and turn it into insights
- Data is continuously processed so you need to tell the application when you want results

Windows!

## Window Types

- Sliding, tumbling, and custom windows
- Tumbling windows are fixed size and grouped keys do not overlap



## Event, Ingest, and Processing Time

- Event time is the time stamp assigned when the event occurred, also called client-side time.
- Processing time is when your application reads and analyzes the data (ROWTIME).

```
…
GROUP BY "sourceIPAddress",
    /* Trigger for results */
    STEP(ctrail.ROWTIME BY INTERVAL '1' MINUTE),
    /* A time stamp grouping key */
    STEP(ctrail."eventTimestamp" BY INTERVAL '1' MINUTE);
```

## Late Results

- An event is late if it arrives after the computation to which it logically belongs has been completed
- Your Kinesis Analytics application will produce an amendment

```
…
GROUP BY "sourceIPAddress",
    /* Trigger for results */
    STEP(ctrail.ROWTIME BY INTERVAL '1' MINUTE),
    /* A time stamp grouping key */
    STEP(ctrail."eventTimestamp" BY INTERVAL '1' MINUTE);
```

## Amazon Kinesis Data Analytics – Pricing

- Pay only for what you use.
- Charged an hourly rate, based on the average number of Kinesis Processing Units (KPU) used to run your application.
- A single KPU provides one vCPU, and 4 GB of memory.
- $0.11 per KPU-hour (US East).

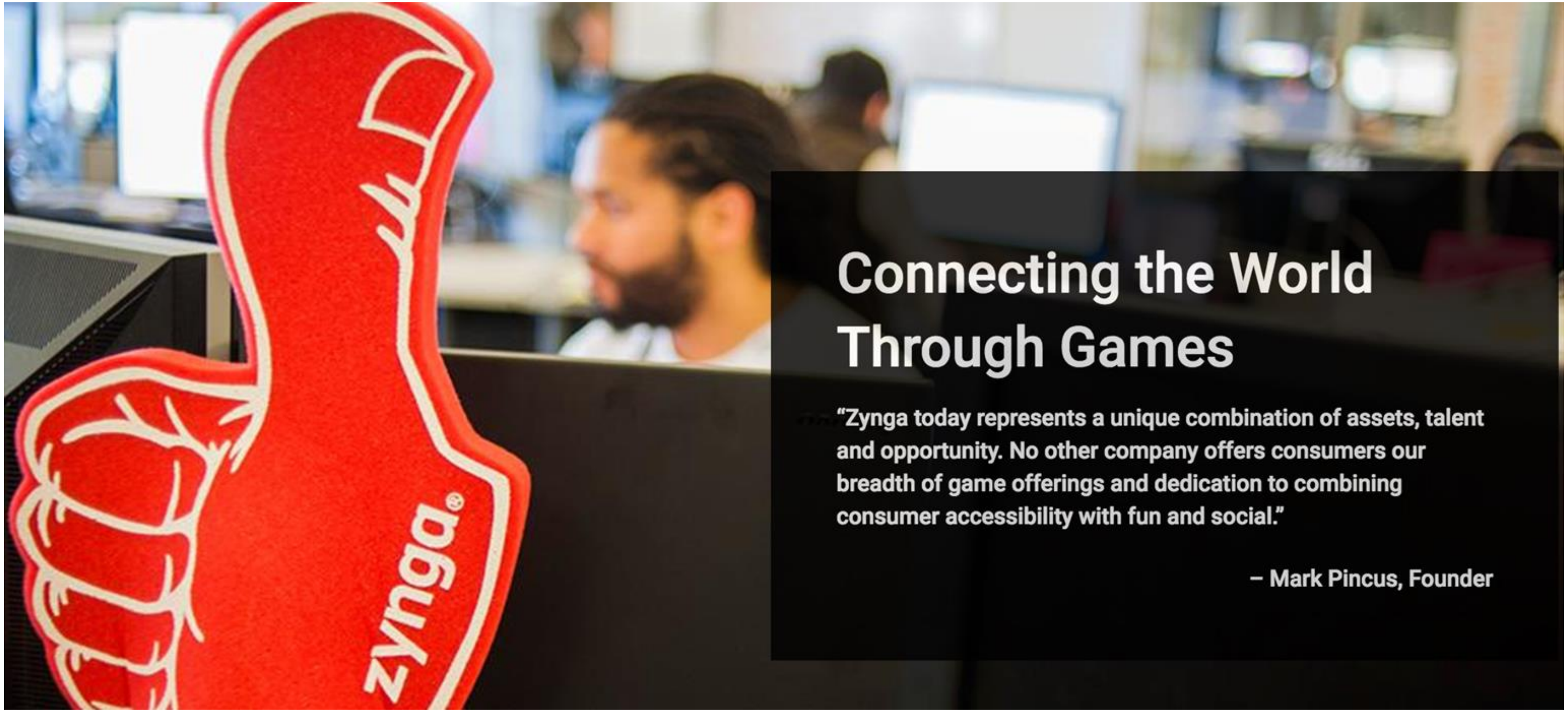
# Customer Examples

| Customer | Use case |
|---|---|
| Zynga | Analyze game events in near real time |
| Netflix | Analyze billions of network flows in real time |
| Sonos | 1 billion events per week from connected devices |
| Zillow | Near-real-time home valuation (Zestimates) |
| Thomson Reuters | Live clickstream dashboards refreshed under 10 sec. |
| GE | IoT predictive analytics |
| Hearst corporation | 100 GB/day clickstreams from 250+ sites |
| AdRoll | 50 billion daily ad impressions, sub-50-ms responses |
| Nordstrom | Online stylist processing 10 million events/day |
| Lyft | Facilitate communications between 100+ microservices |

# Zynga Example

Milan Brahmbhatt, Zynga

# Motivation



Connecting the World Through Games

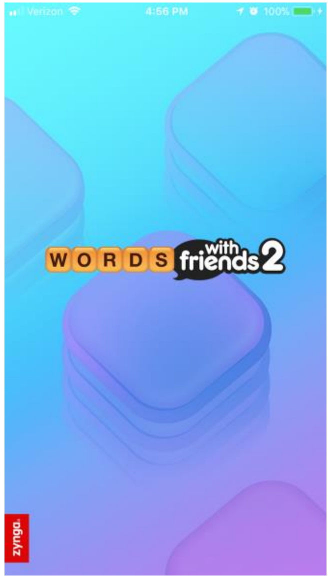
"Zynga today represents a unique combination of assets, talent and opportunity. No other company offers consumers our breadth of game offerings and dedication to combining consumer accessibility with fun and social."

– Mark Pincus, Founder

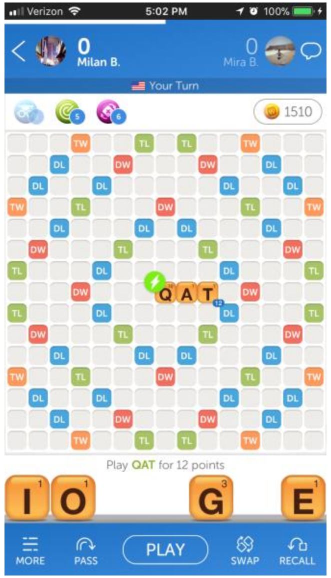
## Agenda

1. What are Game Events @ Zynga ?
2. When does Zynga need a Stream Processing System?
3. When does Zynga NOT need a Stream Processing System?
4. Amazon Kinesis Data Analytics Implementation @ Zynga
5. Implementation Scorecard
6. Best Practices

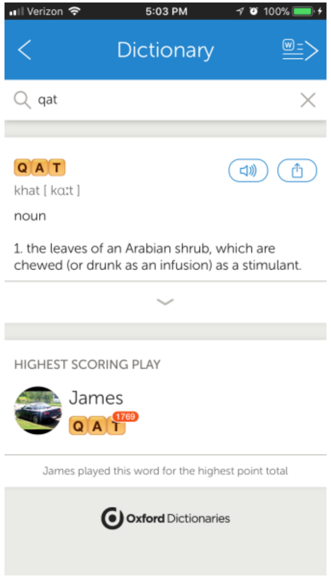
# What are Game Events?



Visit (daily active users)
Installs
Session



Social
Goods
Messages



qat

Q A T

khat [ kɑːt ]

noun

1. the leaves of an Arabian shrub, which are chewed (or drunk as an infusion) as a stimulant.

HIGHEST SCORING PLAY

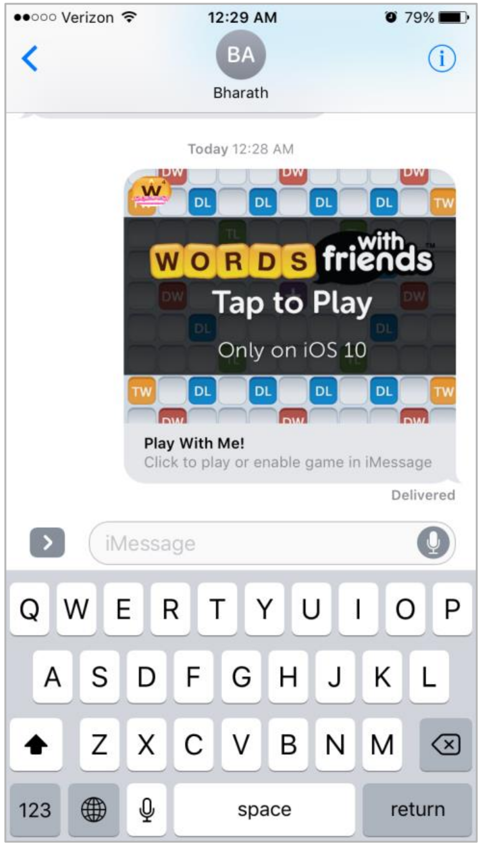
James
Q A T 1769

James played this word for the highest point total

Oxford Dictionaries

Message Clicks

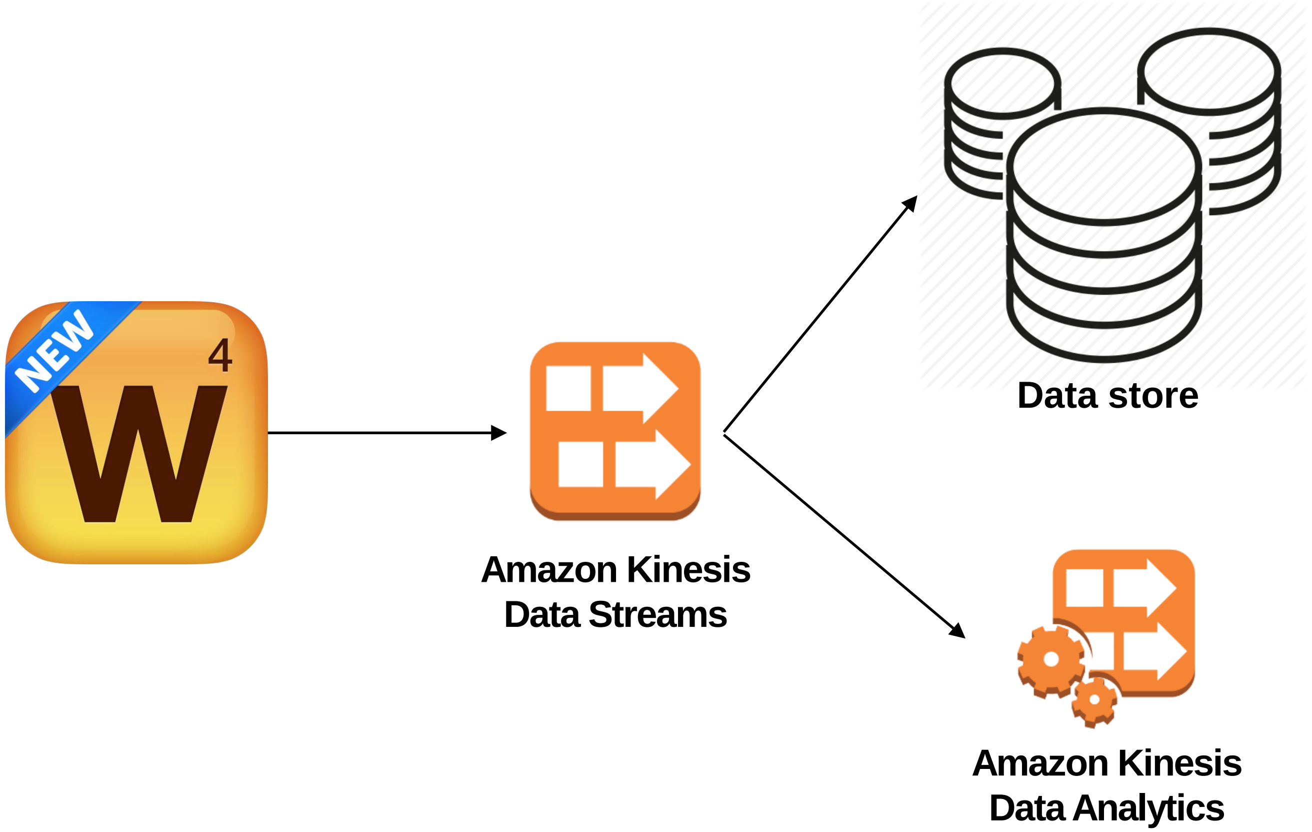
When does zynga need a stream processing service?



Play With Me!
Click to play or enable game in iMessage

How many
***installs***
does [W] have on
the iMessage platform
***in the last 10 minutes***?

# Tracking Installs



When does

zynga®

NOT need a

stream processing service?

## What is ‘s *day7 install date retention?*

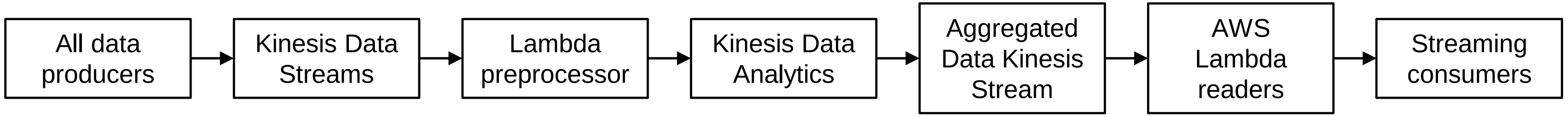
aws SUMMIT

## Design Principles

## Design Principles

| | | | |
|---|---|---|---|
| Managed service | Stateless design | Loose coupling | Extensible system |
| Empower customers | Scalable system | Fault tolerant | Performant system |

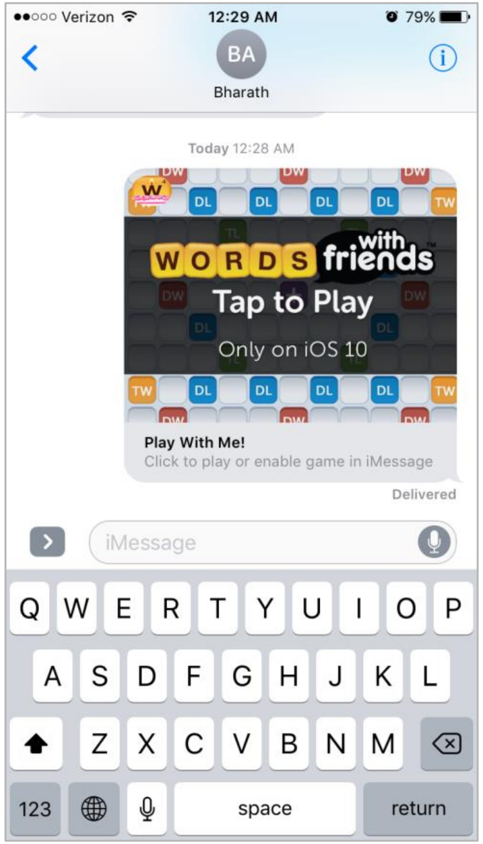
# Amazon Kinesis @ zynga



## Example Aggregation Metric Configuration

```
{
  "data": {
    "counter": "GameX_Counter",
    "game_id": "game_x_id"
  },
  "metric": "GameX-Output-Metric|cnt"
}
```

## Example Incoming Records



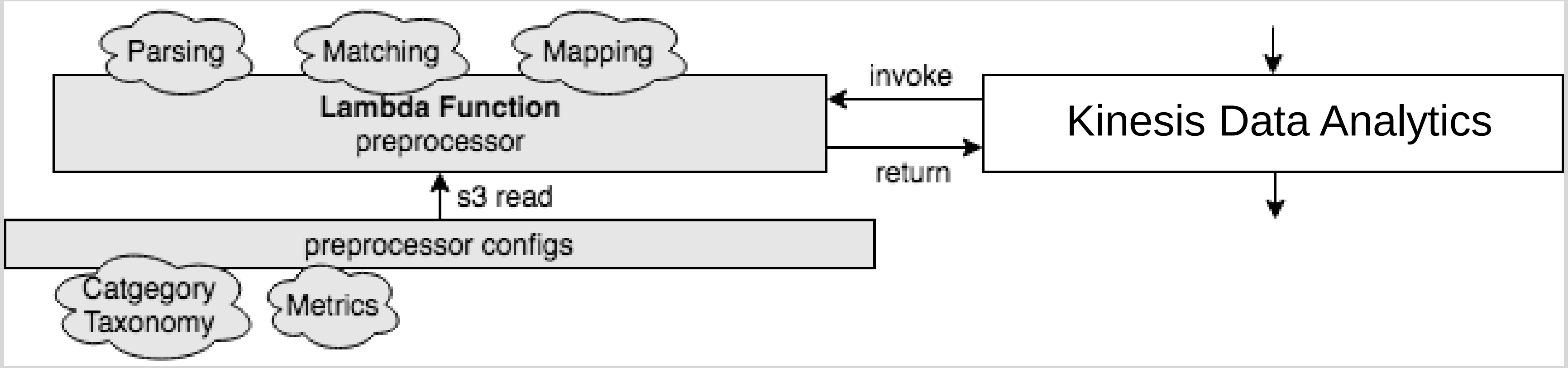
```
"game_x_id,2018-04-04,05:39:02,GameX_Counter"
"game_x_id,2018-04-04,05:39:03,GameX_Counter"
"game_x_id,2018-04-04,05:39:04,GameX_Message_Click"
"game_x_id,2018-04-04,05:40:02,GameX_Counter",
"game_x_id,2018-04-04,05:55:02,GameX_Counter",
```

Want to aggregate and get this result:

```
At 2018-04-04, 05:39:00 – the count is 3
At 2018-04-04, 05:55:00 – the count is 1
```

## AWS Lambda Preprocessor

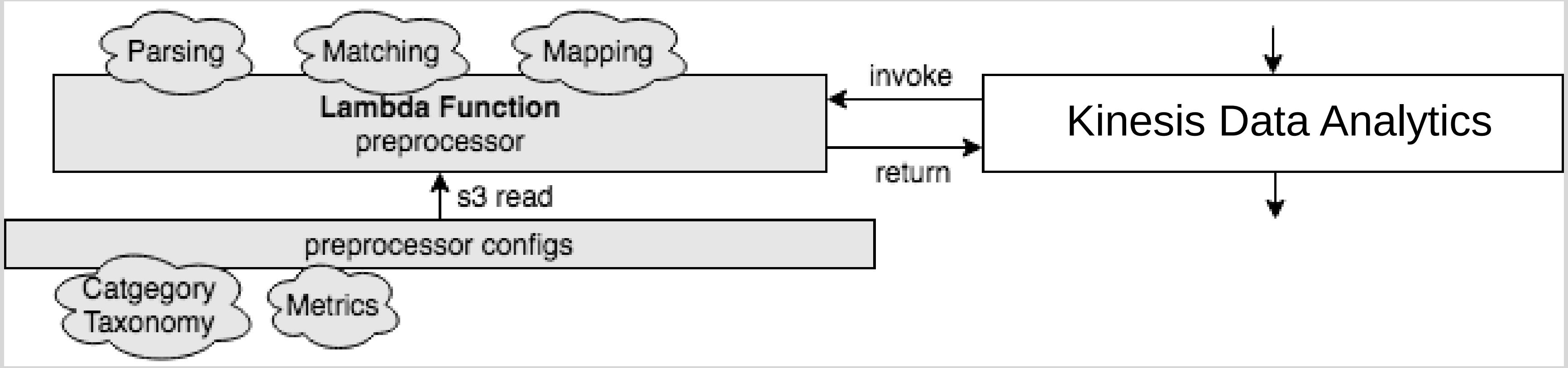
# Our Preprocessor Lambda



To transform the input, we have into aggregation friendly input. We use the preprocessor Lambda to perform the following functions:

1. **Parse** the Kinesis Producer Library-formatted batches
2. **Match** data records to relevant (subscribed and user-defined) metrics
3. **Map** the input taxonomy to (sometimes multiple) output taxonomies

# Our Preprocessor Lambda - Mapping



- We want to avoid restarts on updates (a disruptive operation)
- Use Kinesis Data Analytics to aggregate on meaningful numeric values
- Preprocessor Lambda Exactly-Once invocation

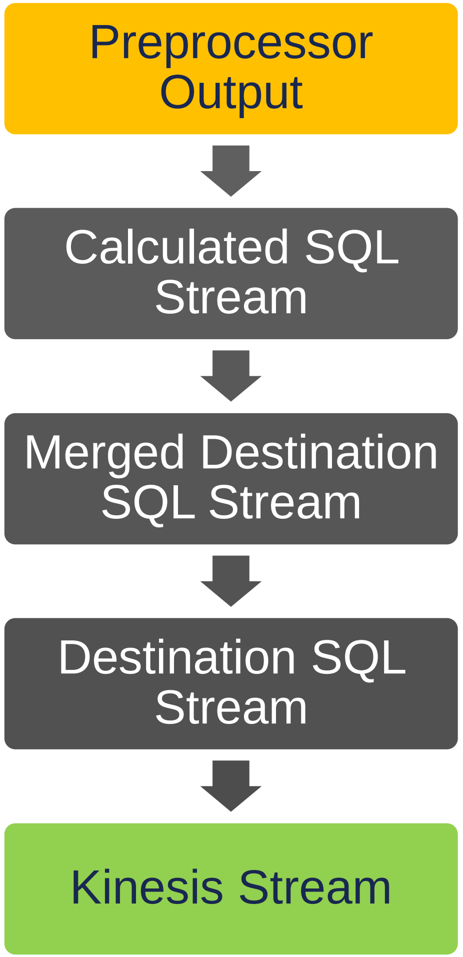
## Preprocessor Output to Kinesis Data Analytics

```
{
    "time": "2018-04-04 05:39:02.0",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "value": 1
}
{
    "time": "2018-04-04 05:39:03.0",
    "source": "consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "value": 1
}
```

```
{
    "time": "2018-04-04 05:40:02.0",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "value": 1
}
{
    "time": "2018-04-04 05:55:02.0",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "value": 1
}
```

# Kinesis Data Analytics SQL

## Streaming Data



Transform raw data to output taxonomy

1st window (tumbling)

Group by rowtime, record time, source, metric, and detail. FLOOR rowtime, time to minute granularity

2nd window (tumbling)

Merge multiple input streams to allow GROUP BY rowtime, time, source, metric, and detail

3rd window (sliding)

Calculate count, sum, average, min, max over 10 min. sliding window

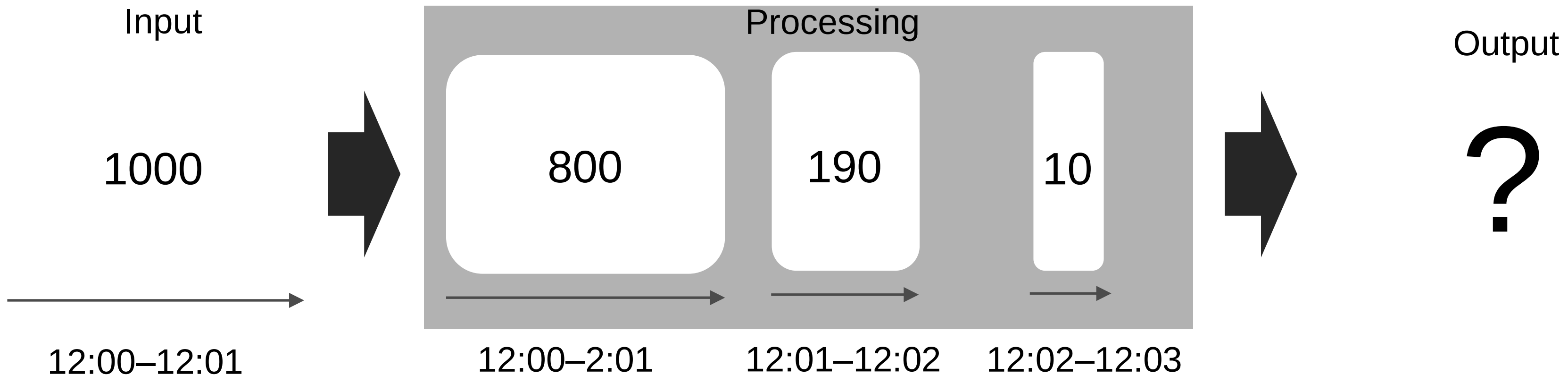
Aggregated data consumed by metric consumers

# Kinesis Data Analytics Processing

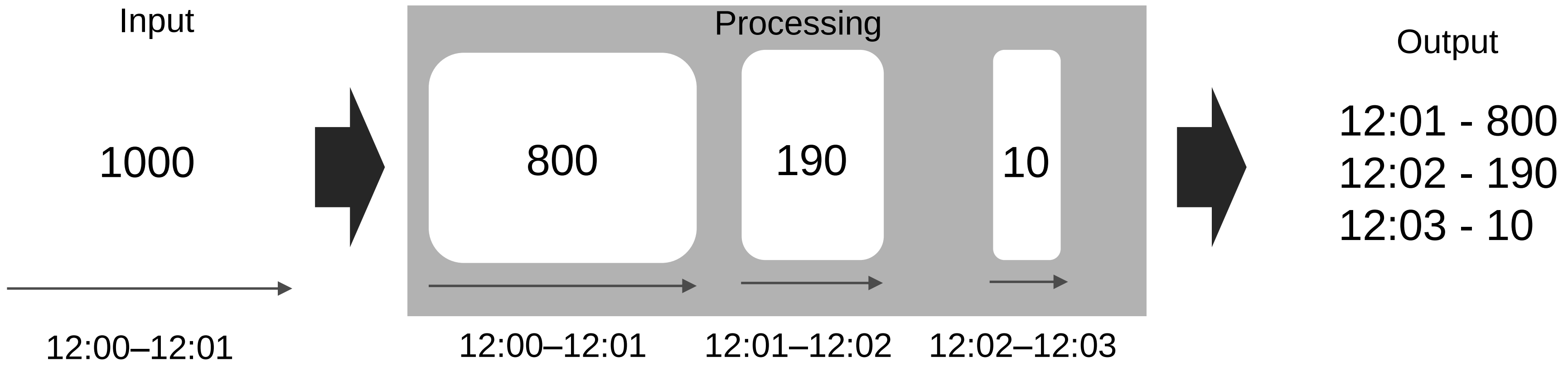
```
{
    "time": "2018-04-04 05:39:00",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "cnt": 2,
    "sum": 2,
    "min": 1,
    "max": 1
}
{
    "time": "2018-04-04 05:40:00",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "cnt": 1,
    "sum": 1,
    "min": 1,
    "max": 1
}
```

```
{
    "time": "2018-04-04 05:55:00",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "cnt": 1,
    "sum": 1,
    "min": 1,
    "max": 1
}
```

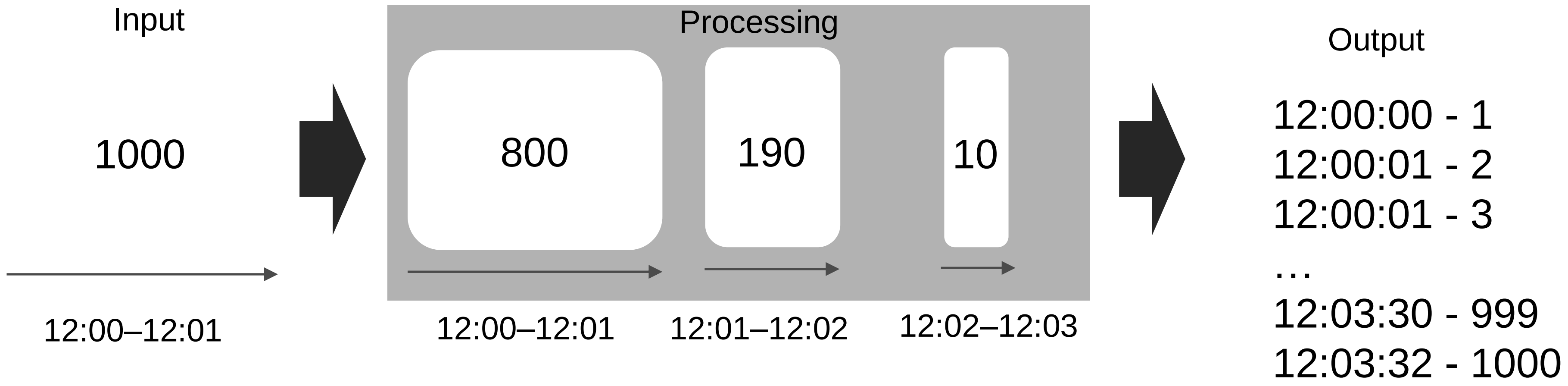
## Scenario


Input
1000
12:00–12:01
Processing
800
190
10
12:00–2:01
12:01–12:02
12:02–12:03
Output
?

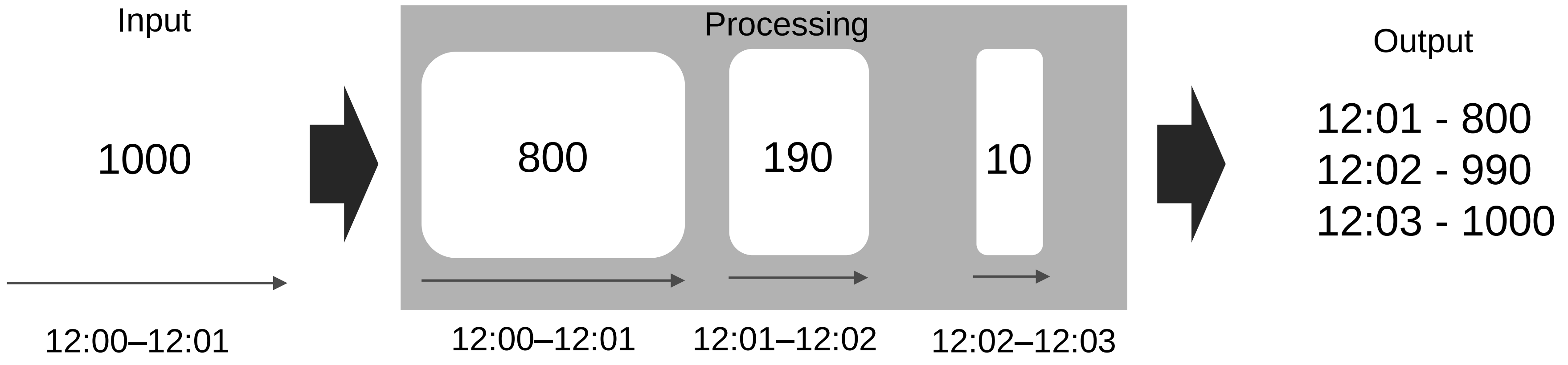
# Tumbling Window Output



# Sliding Window Output



# Multi-Window Chain



# Multi-Window Chain SQL

- 1-minute tumbling window outputs are fed into a 10-min sliding window.
- Data records can take up to ~10 mins to propagate through our data pipeline (but on average they take ~ 3 mins).

```
…
GROUP BY
   FLOOR((”SOME_DESTINATION_SQL_STREAM"."ROWTIME" - TIMESTAMP '1970-01-01 00:00:00') MINUTE / 1 TO MINUTE),
   "time", "source", "metric", "detail";
…
WINDOW W1 AS (
   PARTITION BY "time", "source", "metric", "detail"
   RANGE INTERVAL '10' MINUTE PRECEDING);
```

# Kinesis Data Analytics Output

```
{
    "time": "2018-04-04 05:39:00",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "cnt": 3,
    "sum": 3,
    "avg": 1,
    "min": 1,
    "max": 1
}
```

```
{
    "time": "2018-04-04 05:55:00",
    "source": ”consumer_x",
    "metric": "GameX-Output-Metric|cnt",
    "detail": "",
    "cnt": 1,
    "sum": 1,
    "avg": 1,
    "min": 1,
    "max": 1
}
```

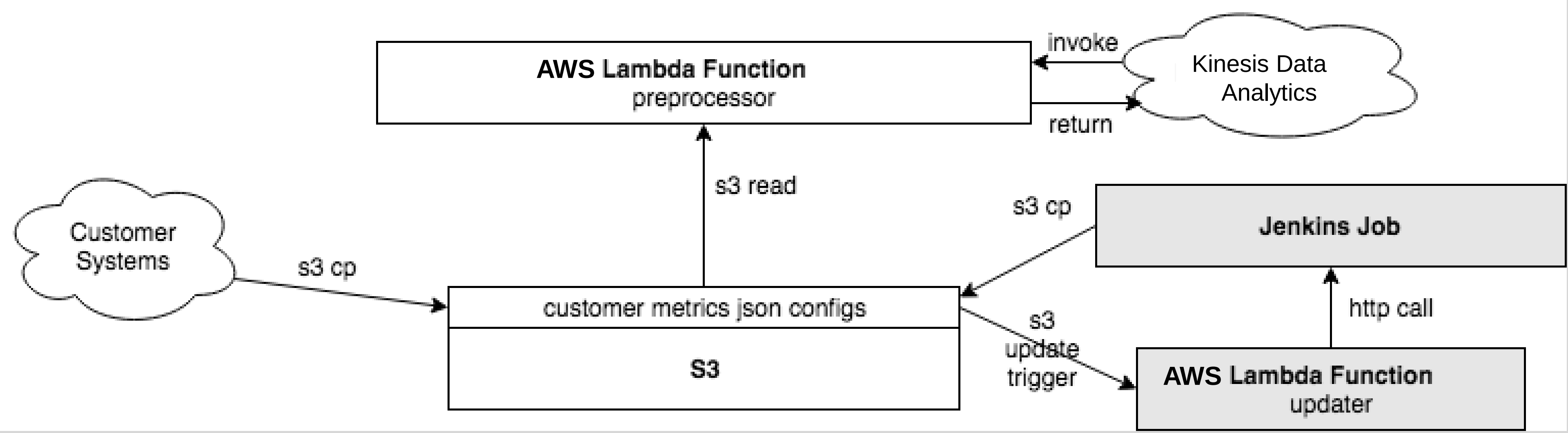
## Non-Disruptive Updates

## Another Aggregation Metric

Configured by GameX for consumer_x

```
{
    "data": {
      ”client_id": ”iMessage",
      "game_id": "game_x_id"
    },
    "metric": "GameX-Install-Output-Metric|cnt"
 }
```

# Non-Disruptive Updates



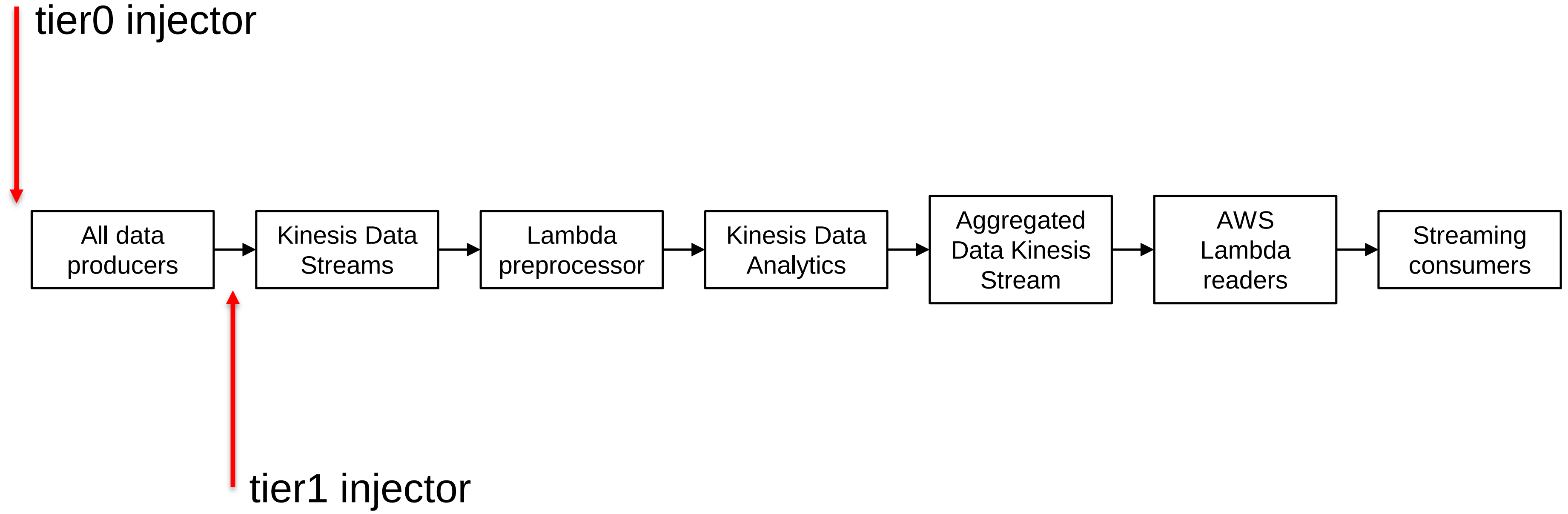
When game teams upload new metrics in Amazon S3, this triggers the self-service AWS Lambda updater to generate preprocessor metric configurations.

## Monitoring

## MillisBehindLatest

## Heartbeat Monitoring

# Amazon Kinesis Data Analytics @ zynga



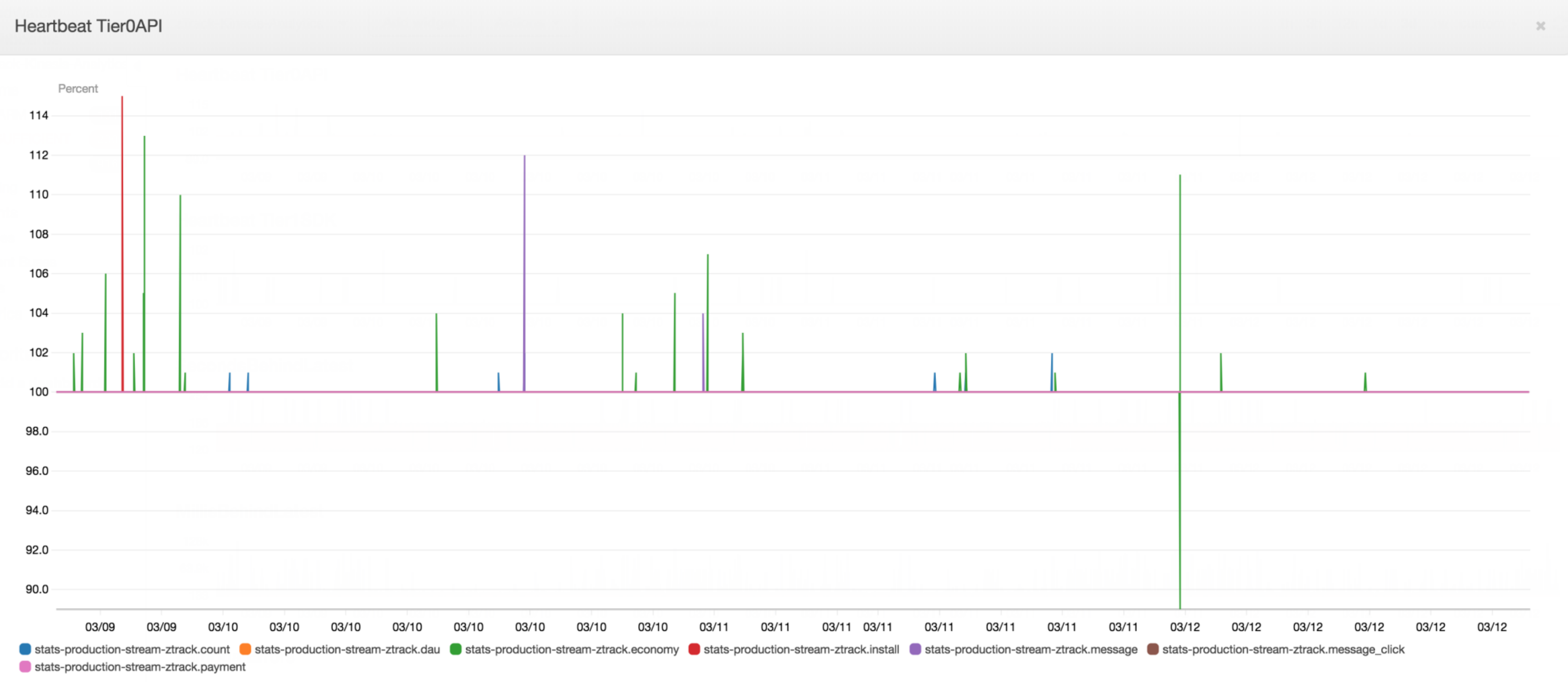
## Heartbeat Injector

Both injectors send 100 artificial game events per minute using a configurable taxonomy that lives in Amazon S3. The taxonomy between the two injectors slightly differ to compare results between the two later.

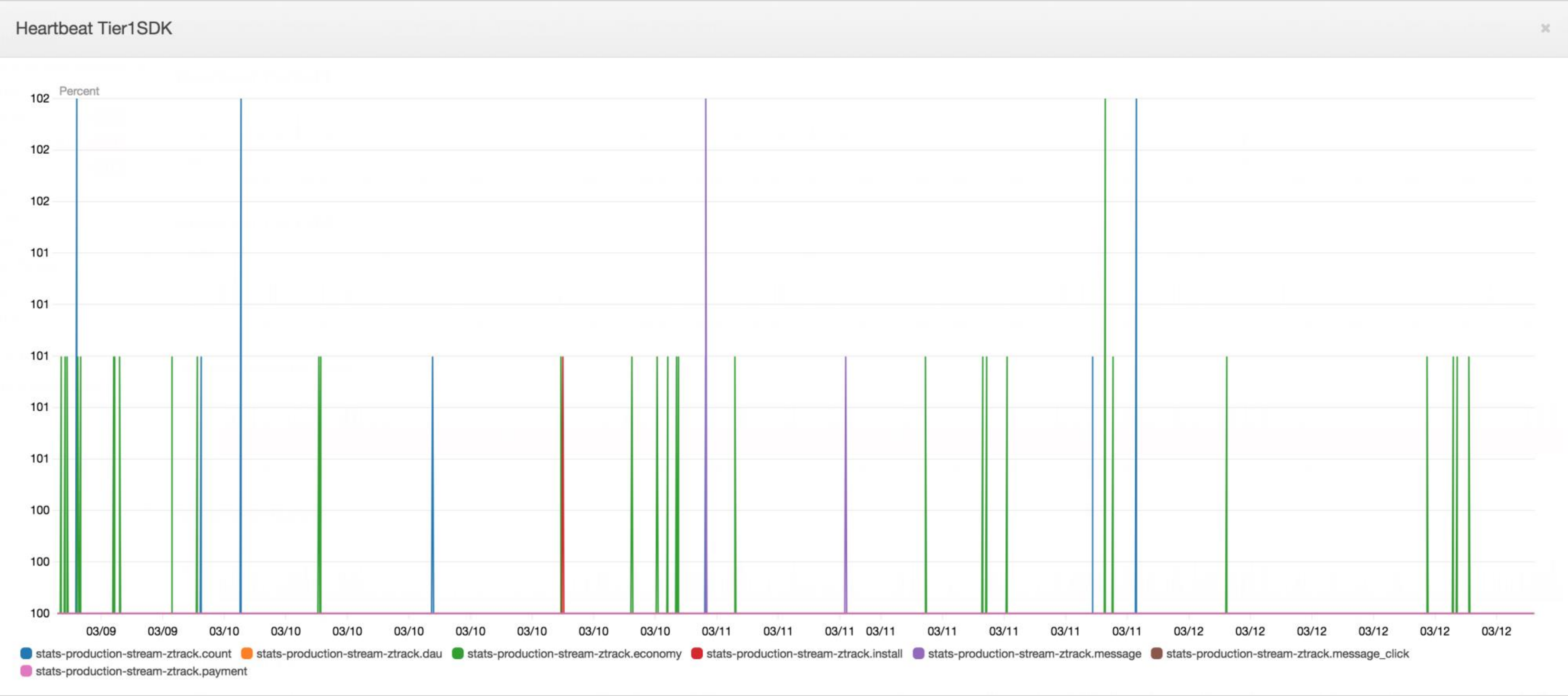
```
”tier0": {
    "appId": ”123456",
    "clientId": "1",
    "data": {
        "counter": "StreamHeartbeatV0.1",
        "kingdom": "Tier0API",
    "phylum": "<injector_hash>",
    ”event": ”some_game_event"
    },
},
```

```
”tier1": {
    "appId": ”123456",
    "clientId": "1",
    "data": {
        "counter": "StreamHeartbeatV0.1",
        "kingdom": "Tier1SDK",
        "phylum": "<injector_hash>",
        ”event": "some_game_event"
    },
},
```

# Heartbeat Metrics



# Heartbeat Metrics



# Scorecard

## Design Principles

Managed service ♥

Stateless design ♥

Loose coupling ♥

Extensible System ♥

Empower customers ♥

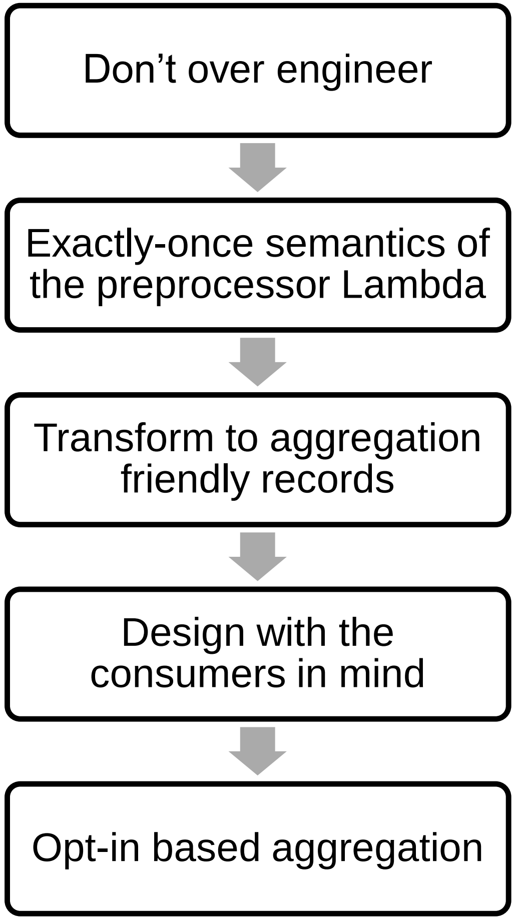
Scalable system ♥

Fault-tolerant ♥

Performant system ♥

## Best Practices

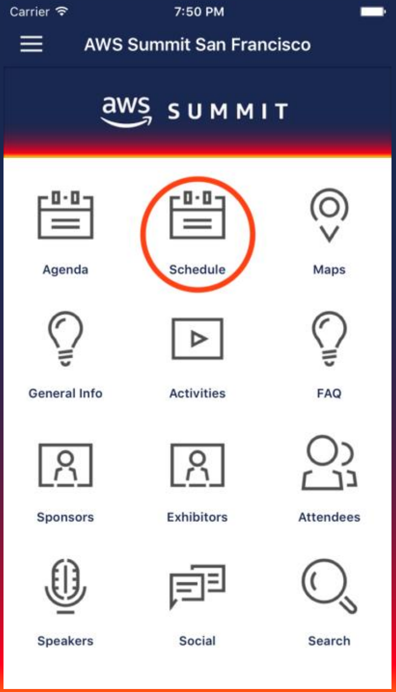
## Best Practices


Don’t over engineer
Exactly-once semantics of the preprocessor Lambda
Transform to aggregation friendly records
Design with the consumers in mind
Opt-in based aggregation

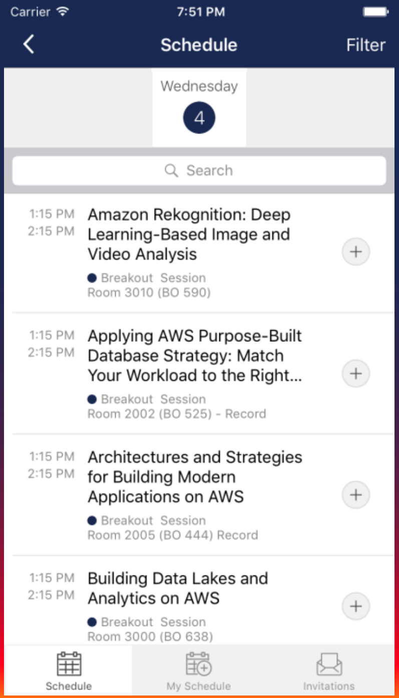
## Please complete the session survey in the summit mobile app.

# Submit Session Feedback

1. Tap the **Schedule** icon.


Carrier 7:50 PM
AWS Summit San Francisco
aws SUMMIT

2. Select the session you attended.


Carrier 7:51 PM
Schedule Filter
Wednesday 4
Search

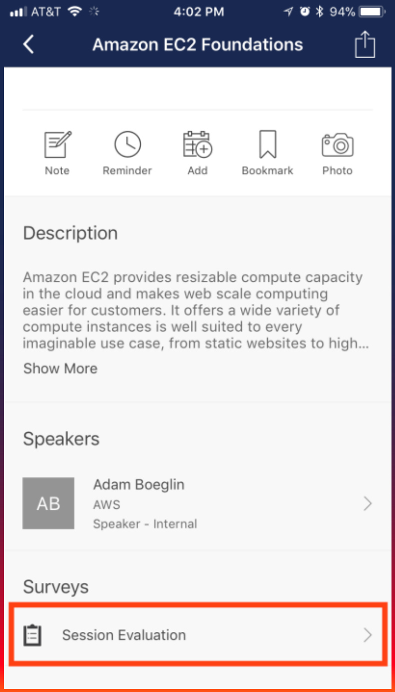
1:15 PM 2:15 PM Amazon Rekognition: Deep Learning-Based Image and Video Analysis
Breakout Session
Room 3010 (BO 590)

1:15 PM 2:15 PM Applying AWS Purpose-Built Database Strategy: Match Your Workload to the Right...
Breakout Session
Room 2002 (BO 525) - Record

1:15 PM 2:15 PM Architectures and Strategies for Building Modern Applications on AWS
Breakout Session
Room 2005 (BO 444) Record

1:15 PM 2:15 PM Building Data Lakes and Analytics on AWS
Breakout Session
Room 3000 (BO 638)

3. Tap **Session Evaluation** to submit your feedback.


AT&T 4:02 PM 94%
Amazon EC2 Foundations

Description

Amazon EC2 provides resizable compute capacity in the cloud and makes web scale computing easier for customers. It offers a wide variety of compute instances is well suited to every imaginable use case, from static websites to high...

Show More

Speakers

AB Adam Boeglin
AWS
Speaker - Internal

Surveys

Session Evaluation

# Thank you!