
SRV316

# Serverless Data Processing at Scale: An Amazon.com Case Study

Kashif Imran

Sr. Solutions Architect, AWS

Haoyu Chen, Amazon

Software Engineer

# What to expect?

**What’s Serverless Real-Time Data Processing?**

**Why Streams, Kinesis, AWS Lambda?**

**Amazon FBA Seller Inventory Authority Platform**

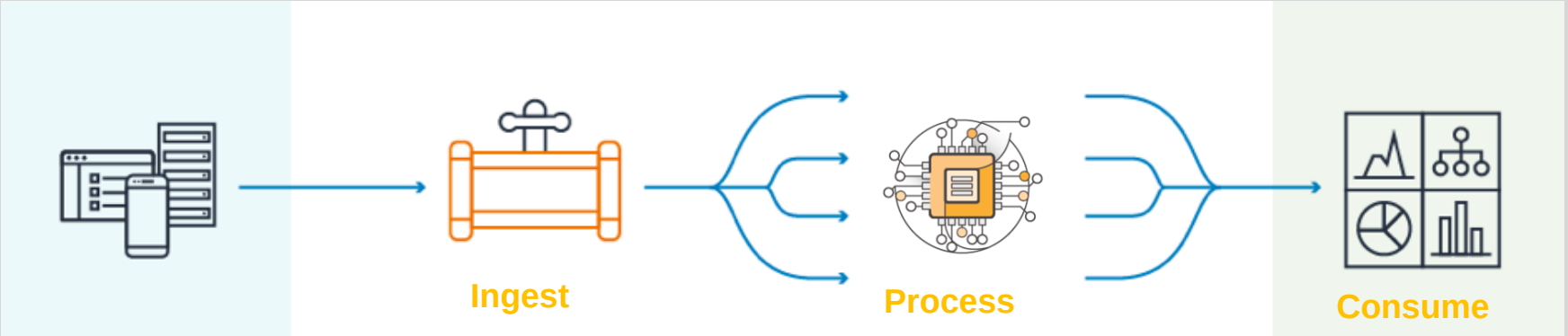
**Amazon Video – Thursday Night Football**

**Best Practices and Lessons Learned**

# Serverless Real-Time Data Processing

Build real-time data processing (ingest, process, and consume) applications and services

**… without managing infrastructure**



aws SUMMIT

# Why Streams?

# Stream Processing

Goal
- High-throughput (>1 GB/s)
- Serverless (managed compute)
- Real-time (pipeline)

Streams
- Data size constraint
- Data time constraint
- Have access to recent data
- Processing time constraint

Batch
- No size constraint
- No time constraint (not real time)
- Have access to all data
- Long running processing (reports)

# Stream Processing

Because you have data that is:

- Generated continuously and simultaneously by thousands of data sources
- Typically small sizes (KBs)

And needs to be processed either:

- Sequentially and incrementally
- Or over sliding windows

in some real-time constraint

aws SUMMIT

# What's Amazon Kinesis?

# Kinesis: What Is It?

## It's storage

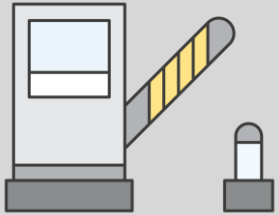
For real-time data that's only stored for a limited time

## Where new data is made available quickly

Typically less than 1 second put-to-get delay

## As a managed service

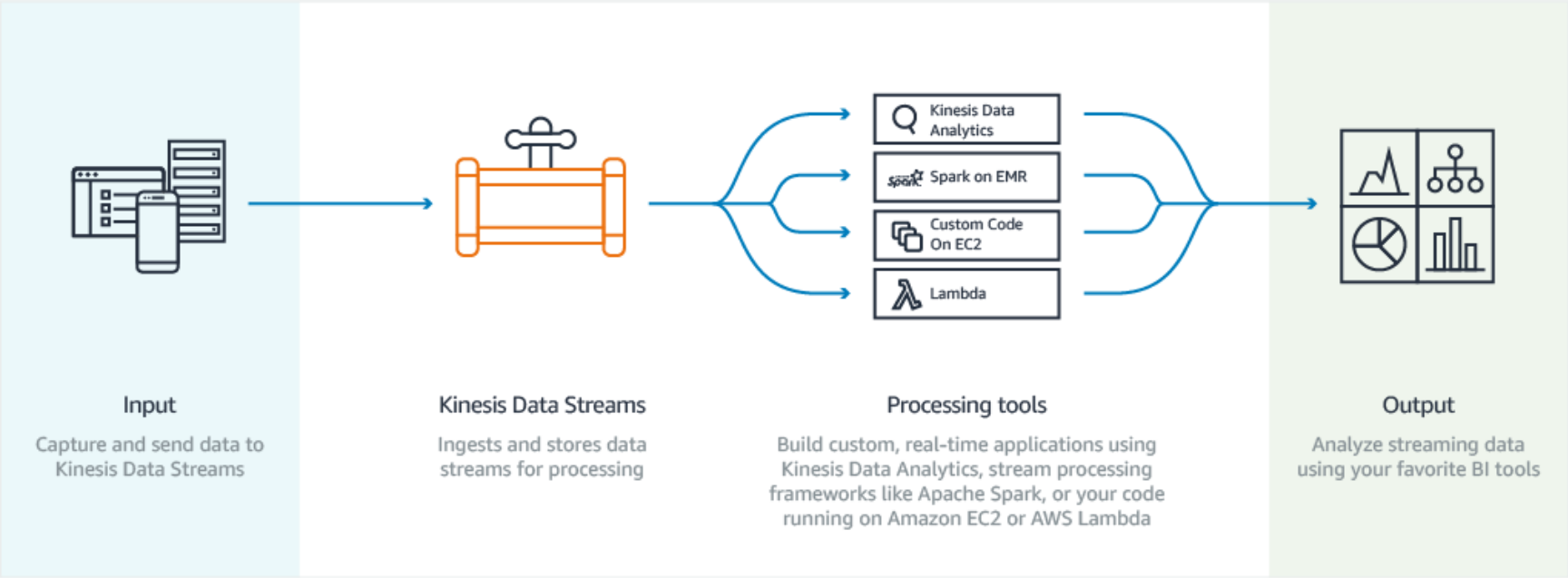
With APIs that let you easily create and configure the stream and put and retrieve data



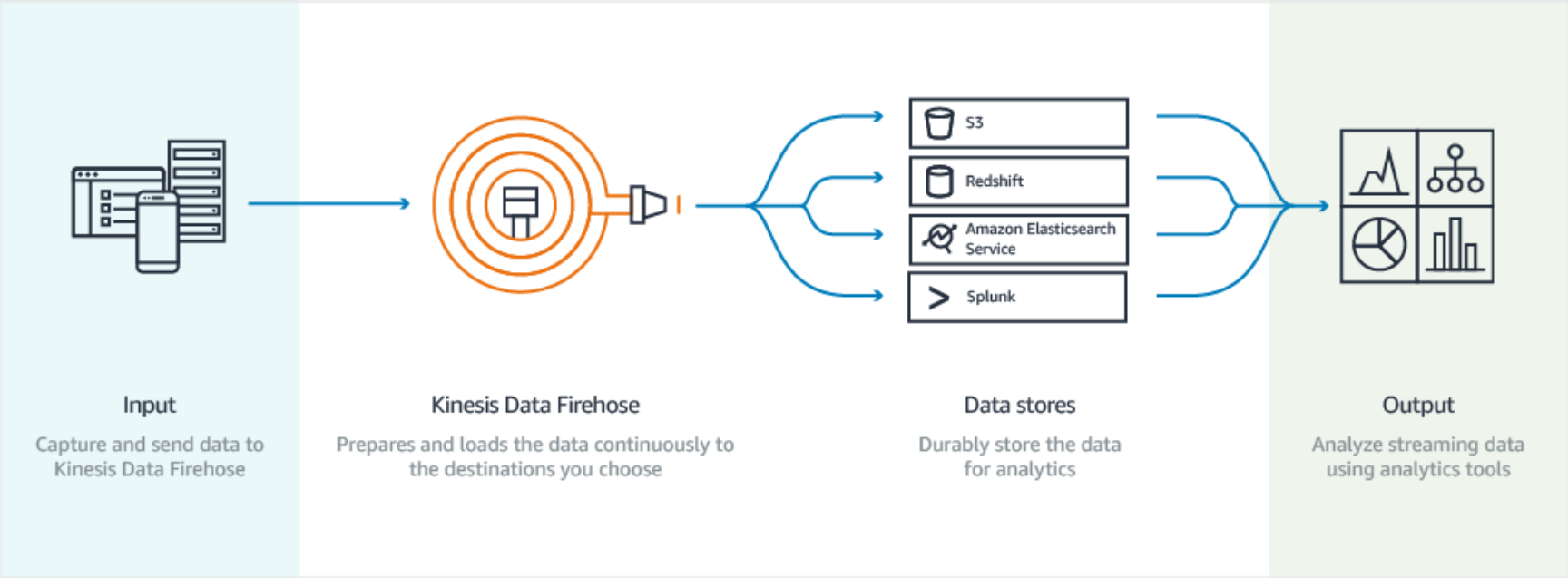
## That uses a checkpoint model

Supports multiple concurrent in-order processing

# Amazon Kinesis Data Streams

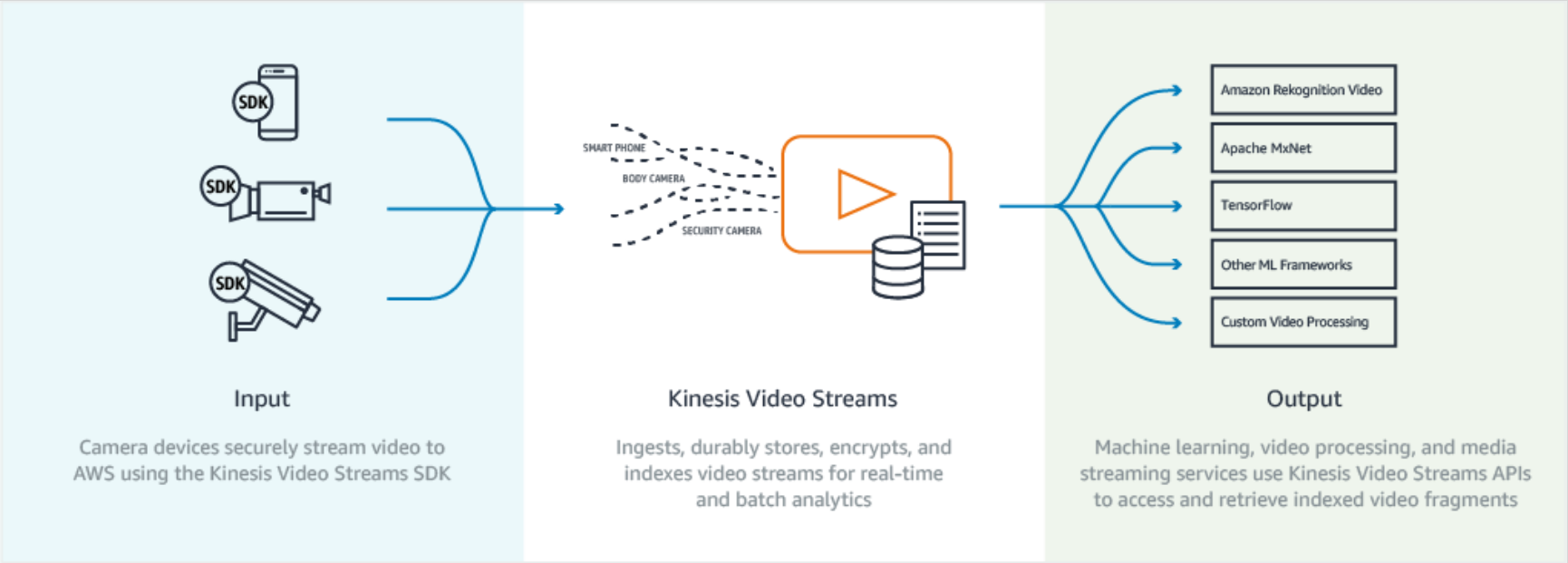


# Amazon Kinesis Data Firehose



# Amazon Kinesis Data Analytics

# Amazon Kinesis Video Streams



aws SUMMIT

# What's AWS Lambda?

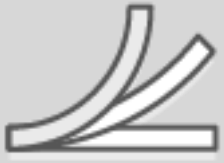
# Lambda: What Is It?

**It's your function**

Your libraries, your code, your executable

With a programming model

Easy to start blueprints and tutorials, monitoring, and logging



And flexible resource model

Choose your memory and we allocate proportional CPU, network bandwidth, disk I/O

And integrated security model

IAM resource policies and roles, VPC support

That runs stateless

Infrastructure abstracted, persist data using Amazon DynamoDB, Amazon S3, or ElastiCache

# Lambda: How Do I Trigger It?

**ASYNCHRONOUS PUSH MODEL**

Amazon S3

Amazon SNS

**Event Invocation**

**SYNCHRONOUS PUSH MODEL**

AWS IoT

Amazon Alexa

**RequestResponse invocation**

**HOW IT WORKS**

Mapping owned by Event Source

triggers Lambda via Invoke APIs

resource-based policy permissions

# Lambda: How do I Trigger It?

**STREAM PULL MODEL**

Amazon DynamoDB

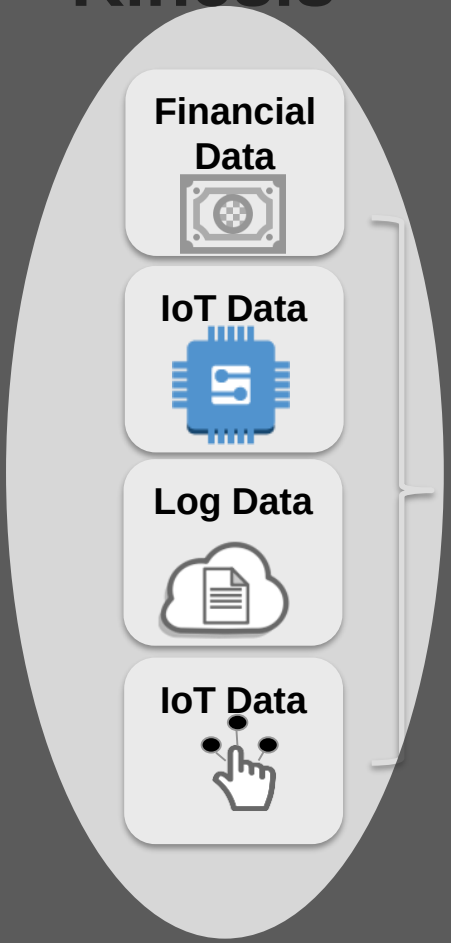
Amazon Kinesis

**Polled batch RequestResponse invocation**

- Mapping owned by Lambda
- Lambda polls the streams
- Lambda function invokes when new records are found on stream
- Lambda execution role policy permissions

# Lambda

# Kinesis




Amazon Kinesis

**EVENT SOURCE**

Node.js
Python
Java
C#

**FUNCTION**

Database

Cloud Service

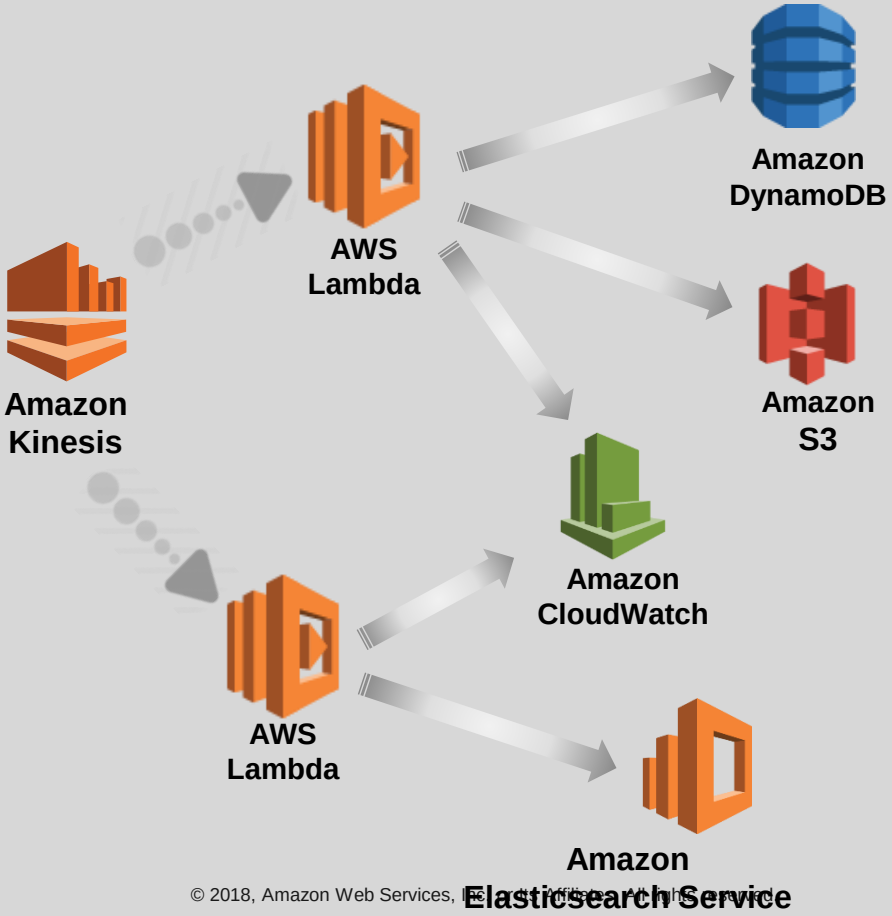
Anything

**ENDPOINT**

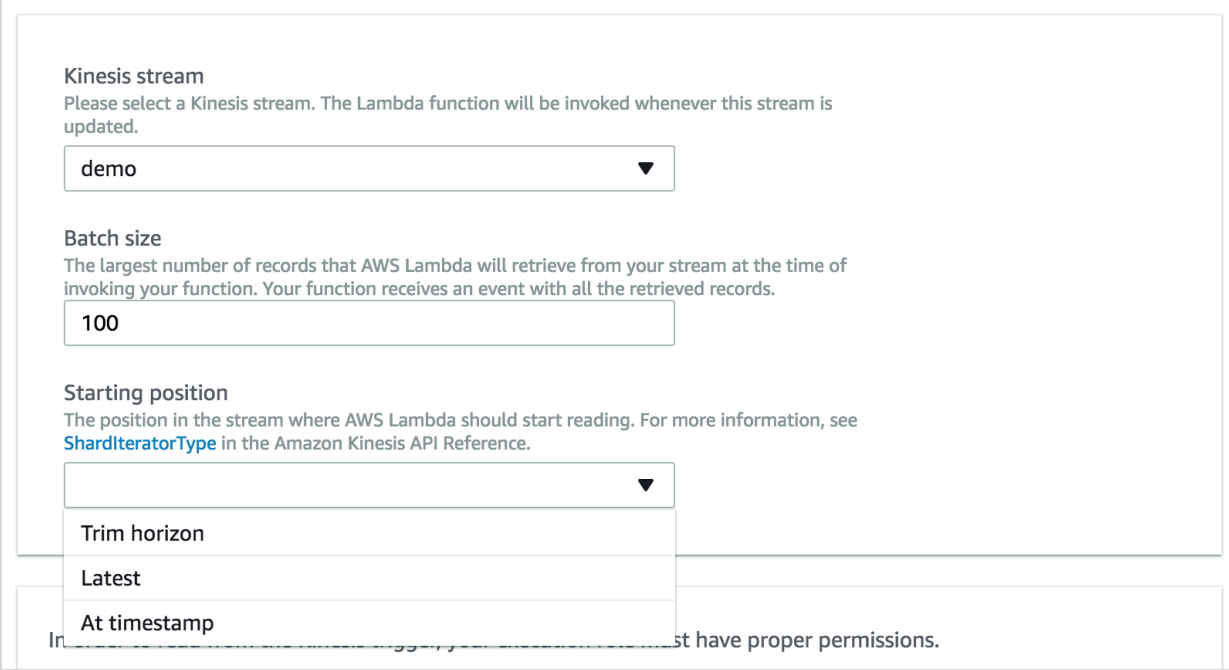
aws SUMMIT

# Stream Processing by Lambda

# Processing Streams: Lambda



# Processing Streams: Lambda



# Processing Streams: Lambda

Event received by Lambda function is a collection of records from the stream:


{ "Records": [ {
"kinesis": {
"partitionKey": "partitionKey-3",
"kinesisSchemaVersion": "1.0",
"data": "SGVsbG8sIHRoaXMgaXMgYSB0ZXN0IDEyMy4=",
"sequenceNumber": "49545115243490985018280067714973144582180062593244200961" },
"eventSource": "aws:kinesis",
"eventID": "shardId-
000000000000:49545115243490985018280067714973144582180062593244200961",
"invokeIdentityArn": "arn:aws:iam::account-id:role/testLEBRole",
"eventVersion": "1.0",
"eventName": "aws:kinesis:record",
"eventSourceARN": "arn:aws:kinesis:us-west-2:35667example:stream/examplestream",
"awsRegion": "us-west-2" } ] }

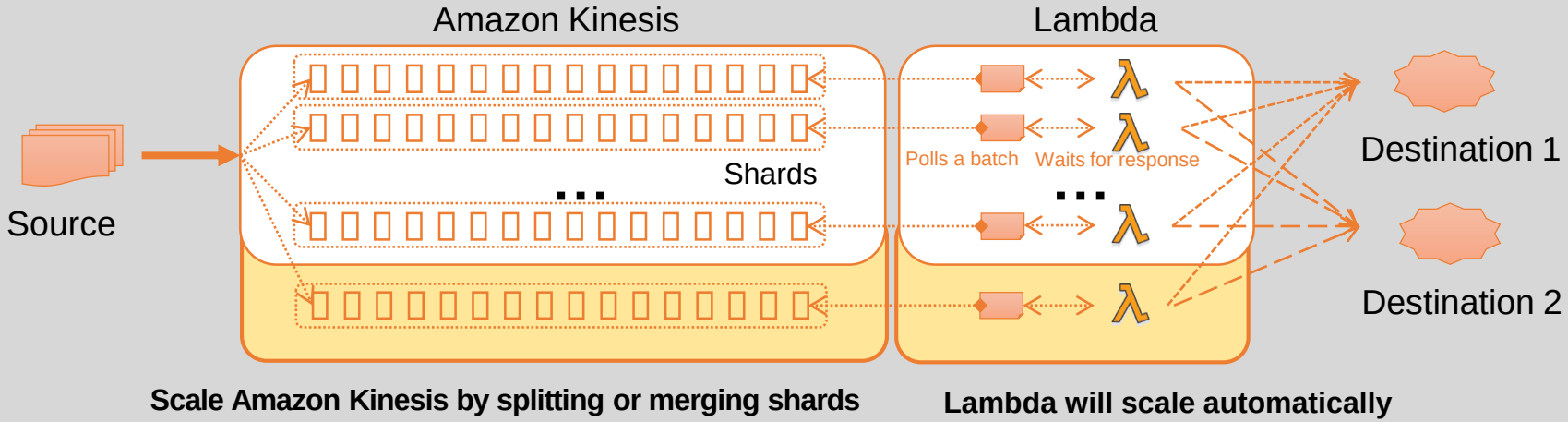
# Processing Streams: Lambda

```python
from __future__ import print_function

import boto3
import base64
import json

print('Loading function')


def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    table = dynamodb.Table('scores')

    #print("Received event: " + json.dumps(event, indent=2))
    for record in event['Records']:
        # Kinesis data is base64 encoded so decode here
        payload = base64.b64decode(record['kinesis']['data'])
        print("Decoded payload: " + payload)
        item = json.loads(payload)

        table.update_item(
            Key={
                'player_id': item["player_id"]
            },
            UpdateExpression='SET score = :val',
            ExpressionAttributeValues={
            ':val': item["score"]
            }
        )

    return 'Successfully processed {} records.'.format(len(event['Records']))
```

# Processing Streams: Lambda

Per shard:

- Lambda calls GetRecords with max limit from Kinesis (10 k or 10 MB)
- If no record, wait some time (1s)
- Sub-batch in-memory and format records into Lambda payload
- Invoke Lambda with synchronous invoke



# Processing Streams: Lambda

**Retries**

Will retry on execution failures until the record is expired

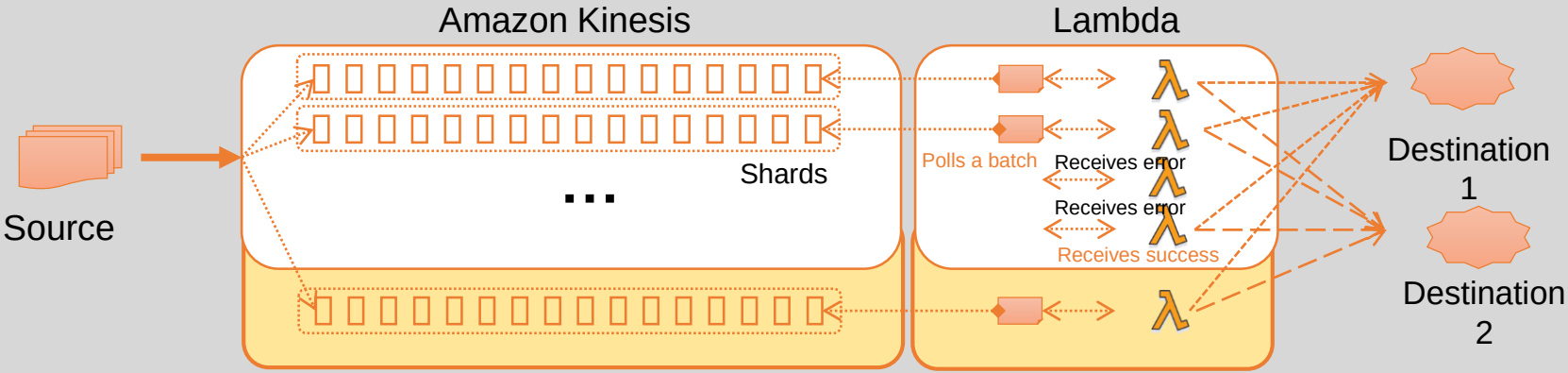
Throttles and errors impact duration and directly impact throughput

**Best practice**

Retry with exponential backoff

**Effective theoretical throughput with retries**

( # shards * batch size (MB) ) / ( function duration (s) * retries until expiry)



# FBA Seller Inventory Authority Platform

Furby

## Furby Boom Figure (Polka Dots) (Discontinued by manufacturer)

4.4 out of 5 stars ▾ 609 customer reviews | 157 answered questions

Price: $124.14 ✓prime

**Only 1 left in stock - order soon.**

**Want it tomorrow, March 30?** Order within **7 hrs 31 mins** and choose **One-Day Shipping** at checkout. Details
Sold by SmoothSellin and Fulfilled by Amazon. Gift-wrap available.

- Collect and hatch virtual Furblings to fill your city in the Furby Boom app, and try to get the golden Furblings egg
- You can hatch and raise virtual Furblings with your Furby Boom, and play games together using the free Furby Boom app
- Use the Furby Boom app to give your Furby Boom creature virtual food, checkups, showers and more
- Furby Boom will remember the name you give it and the names of other Furby Boom friends it meets
- Furby Boom has more than twice as many possible responses as the previous Furby, and how you treat Furby Boom will shape its personality
- NO BATTERIES ARE INCLUDED

Compare with similar items

**Used & new** (17) from $37.60 + $5.27 shipping

Report incorrect product information.

There is a newer model of this item:

Hasbro Furby Connect Friend, Pink
$35.25
4.4 out of 5 stars (1,017)
In Stock.

# How Does FBA Work?

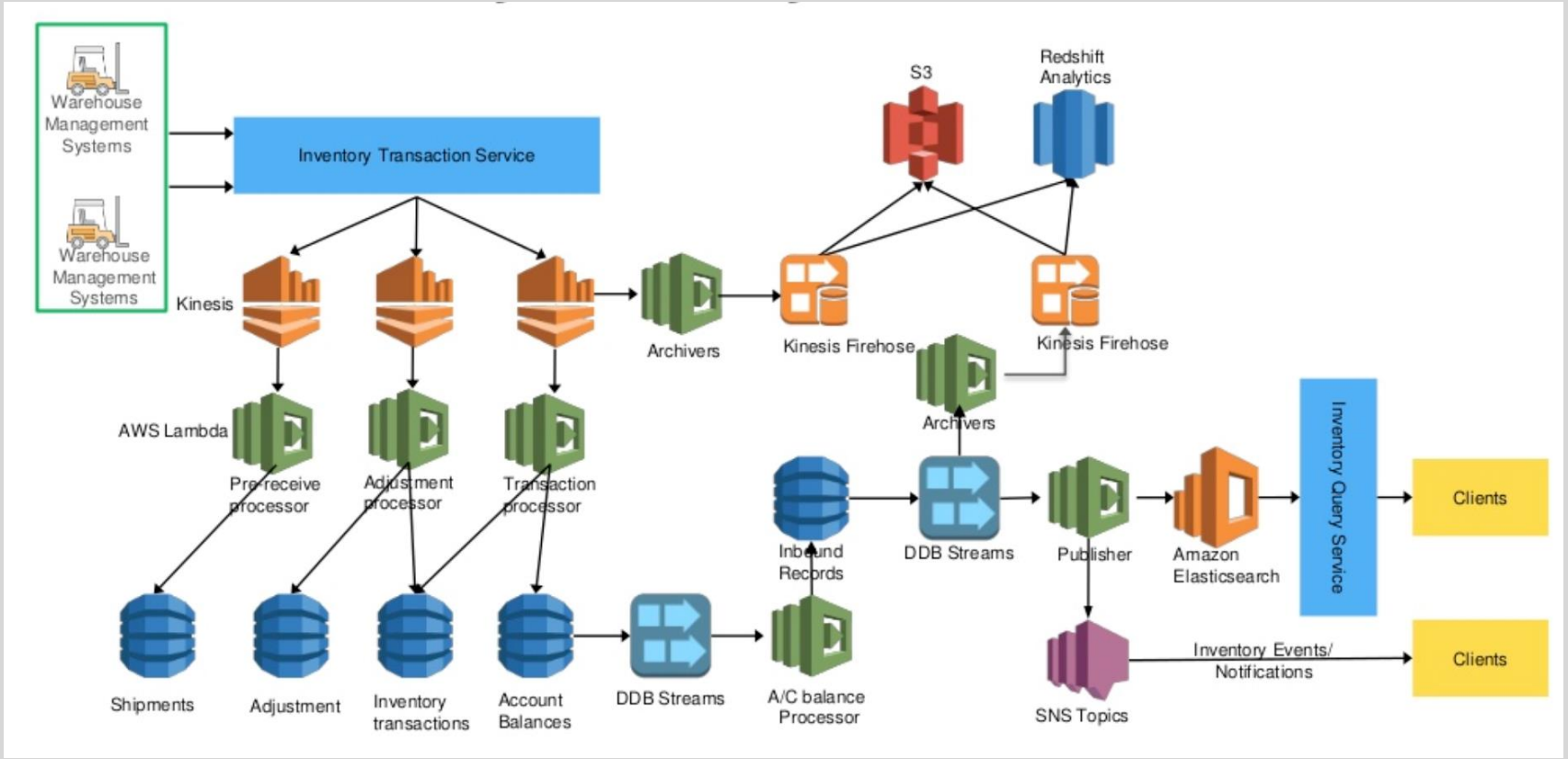
# Data Platform Goals

- Single source of truth for seller’s inventory
- Reconciled view of inventory
- Surface and track discrepancies

# Design Requirements

- Should handle high volume input messages (>10000 TPS)
- Should handle hot keys in input messages
    - E.g.: Received inventory message for a very large shipment
- Should handle duplicate and out of order input messages
- Must maintain an audit train for every inventory quantity change

# Seller Inventory Authority Platform

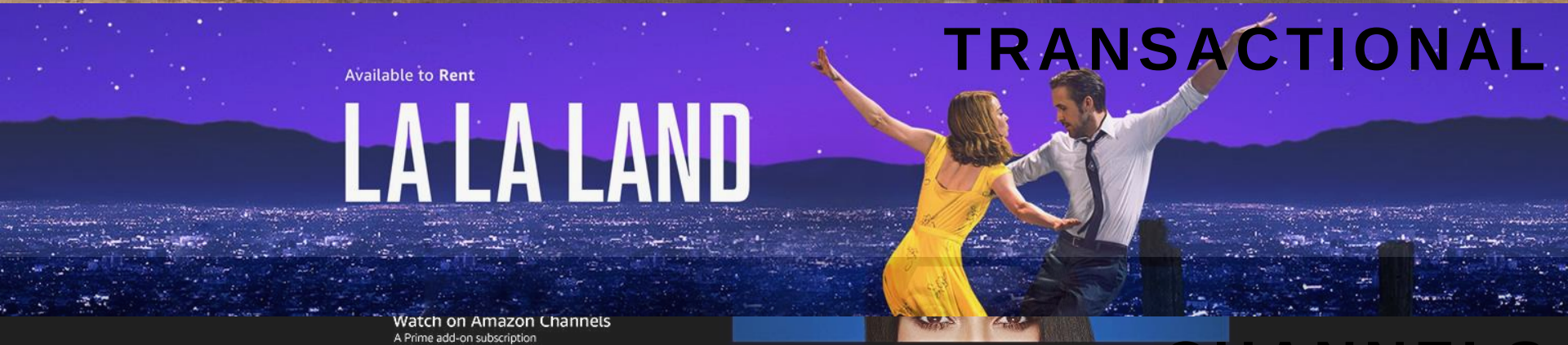


# Launch Results

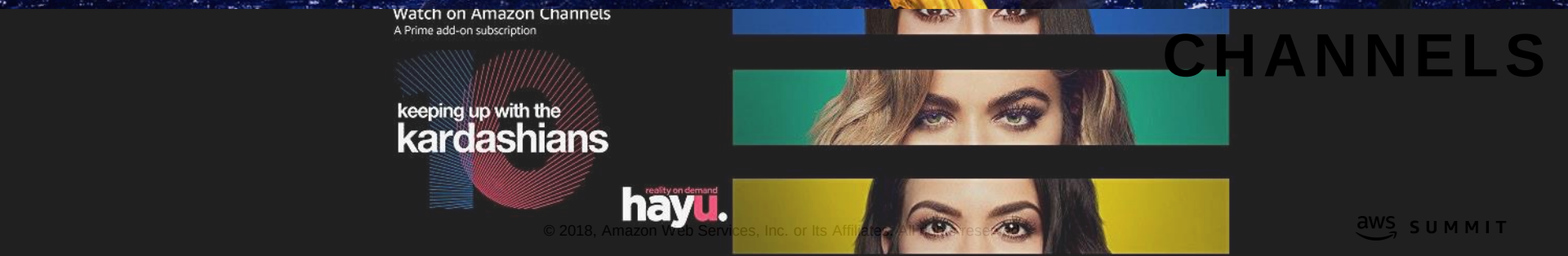
- 22 dev weeks savings in operations costs across 11 fleets
- Design to launch <4 months
- Improved the accuracy of inbound quantities between 5% to 10%
- Reduced cost of business operations (for example, seller contacts)
- Developers in the team enjoyed using AWS Managed Services

aws SUMMIT

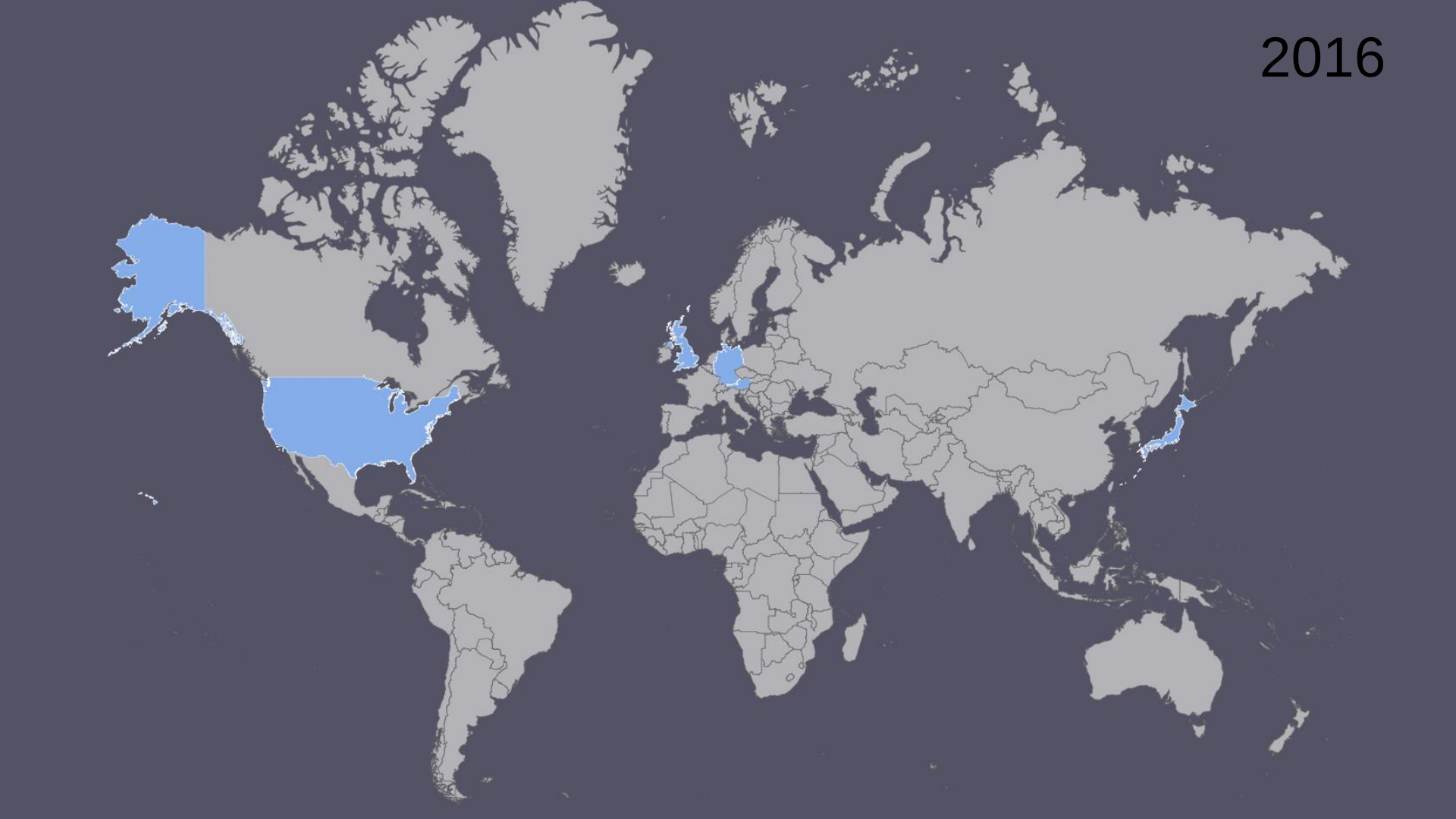

# SUBSCRIPTION

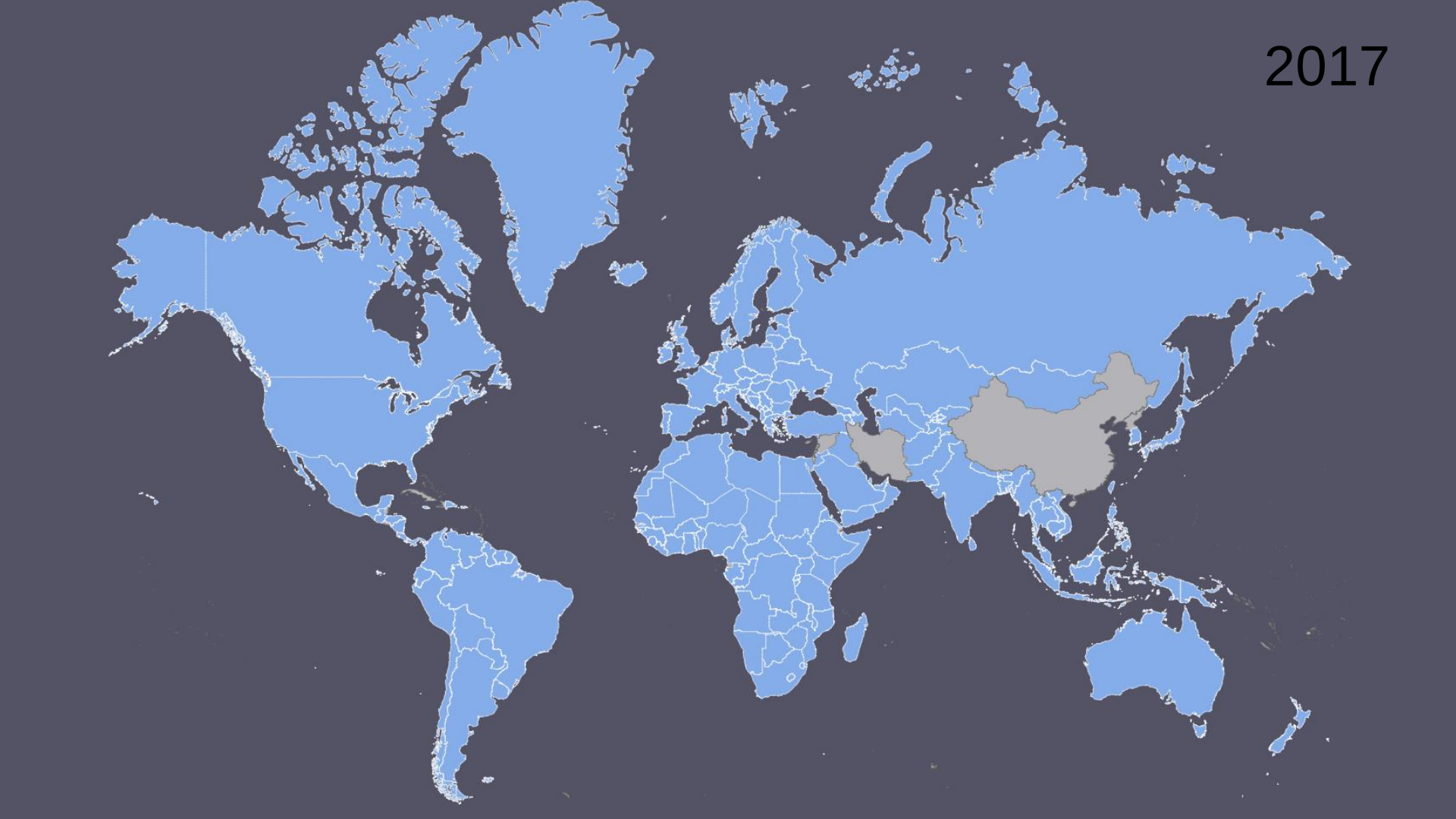



# TRANSACTIONAL



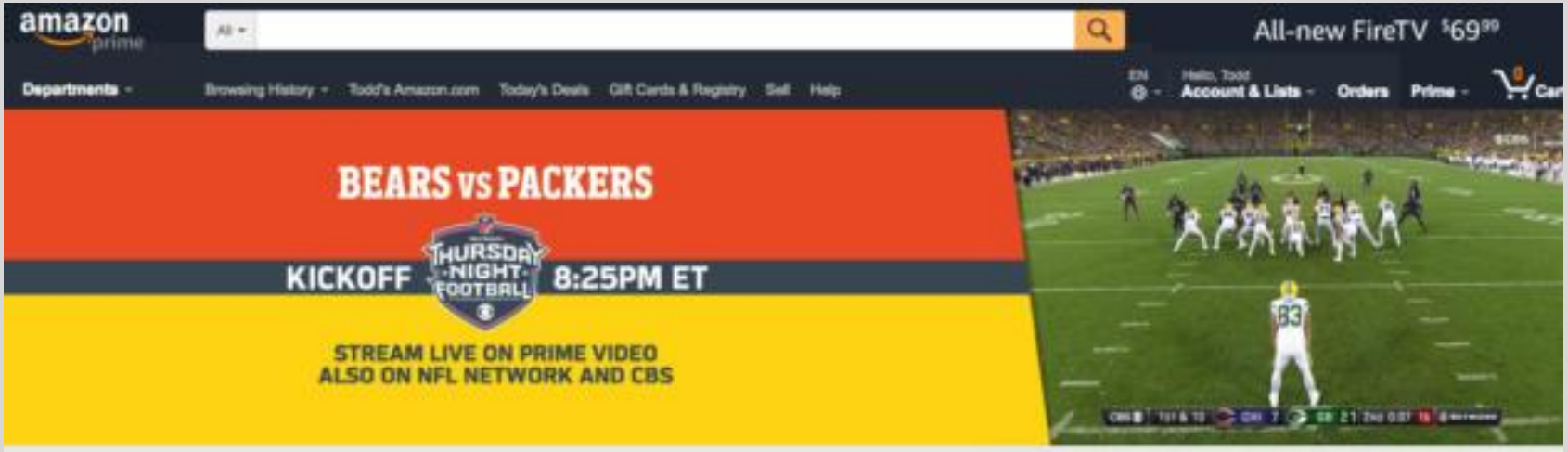

# CHANNELS


2016

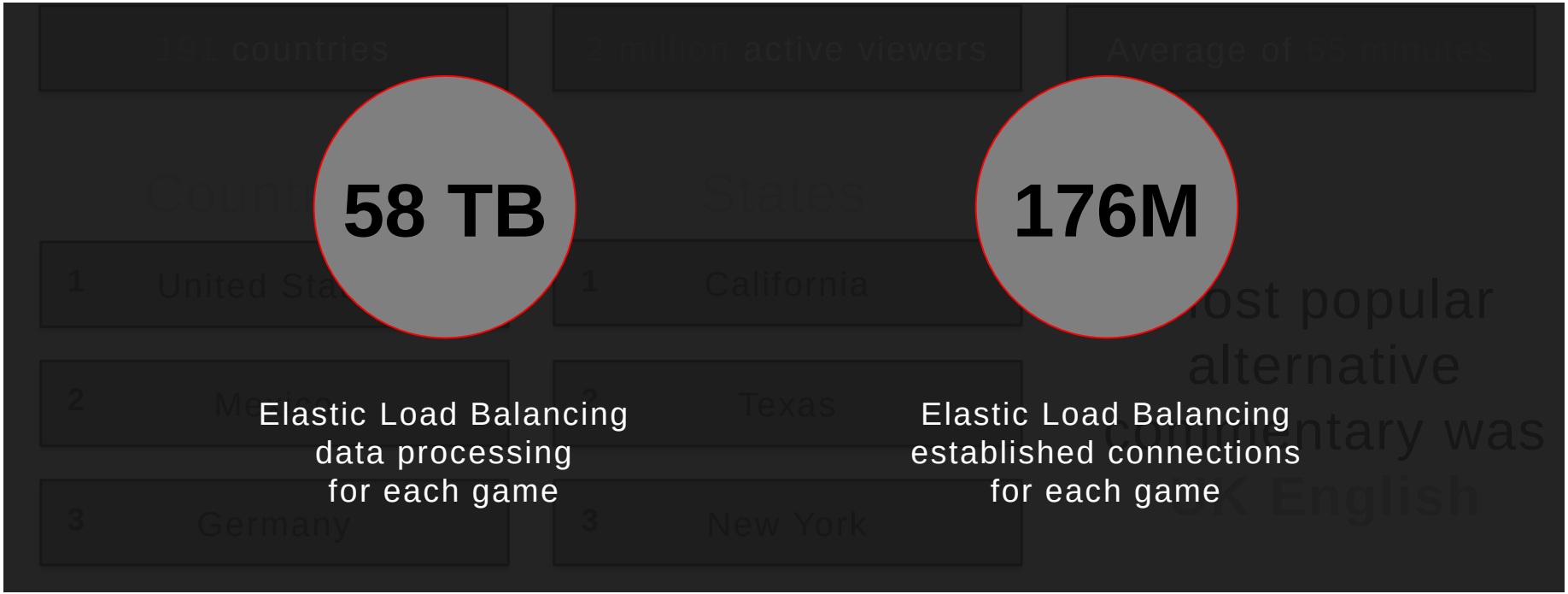

2017

# Thursday Night Football

- Live Stream to millions of users in 191 countries
- 600 different types of devices

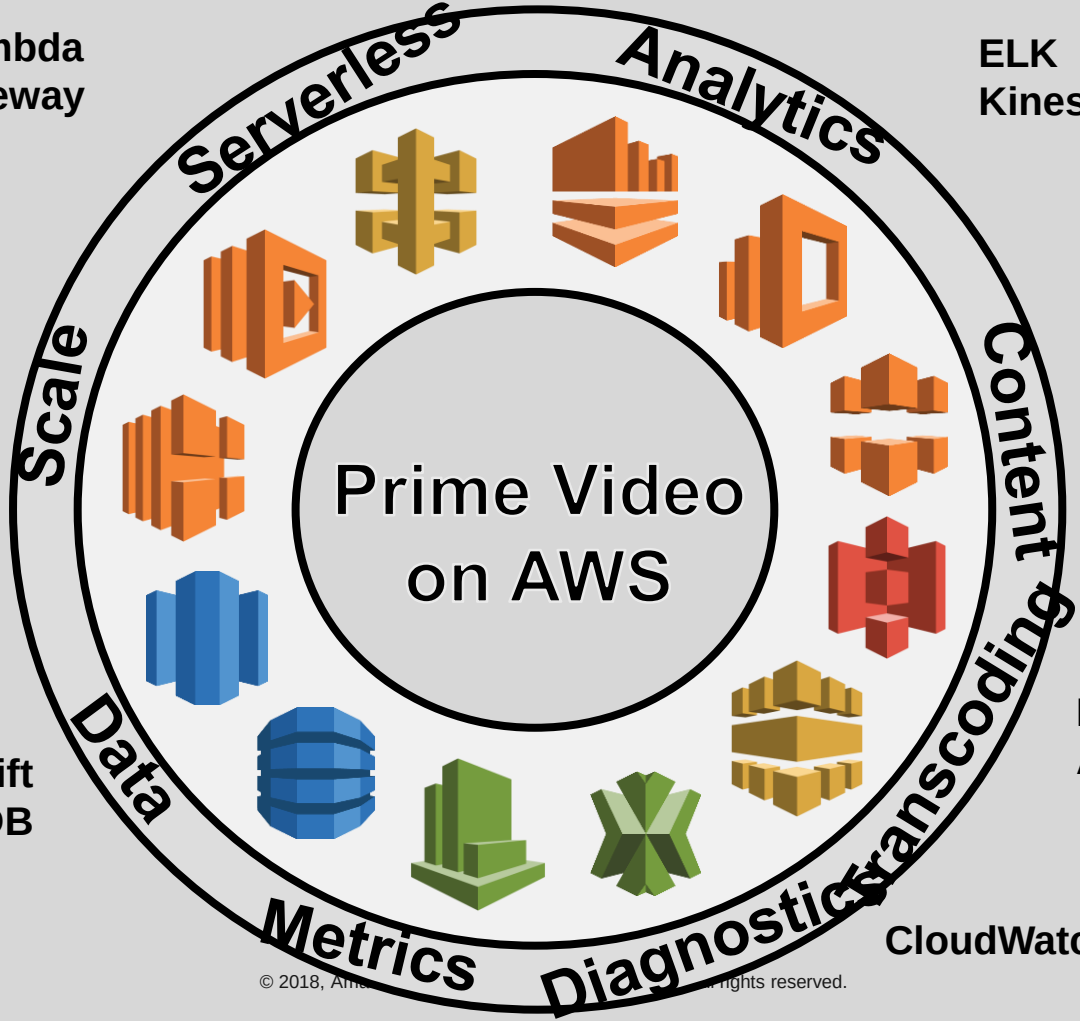


# Thursday Night Football statistics



Lambda
API Gateway

ELK
Kinesis




Load Balancers
EC2 Instances

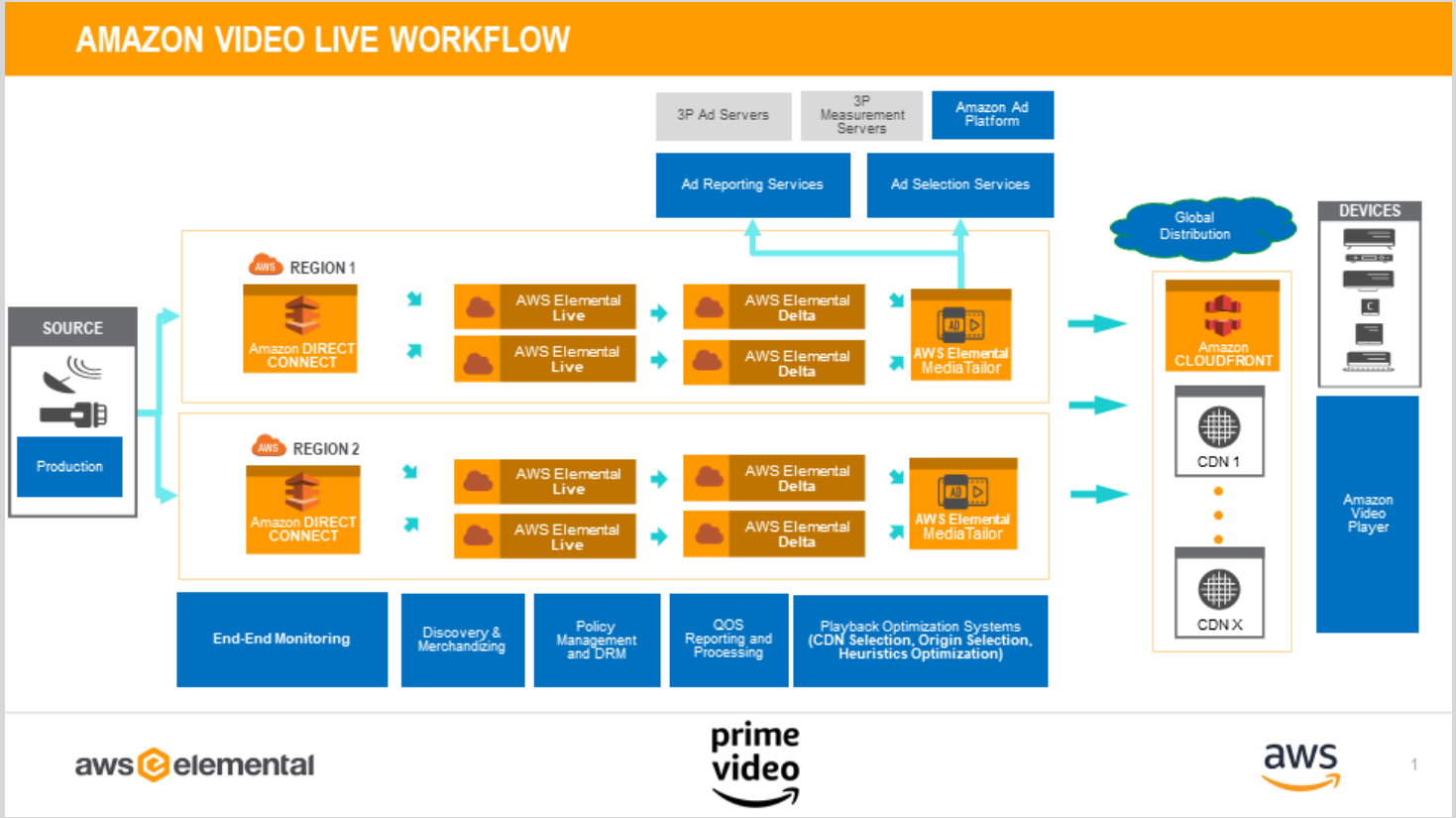
Amazon S3
CloudFront

Elemental
Ad Insertion

Amazon Redshift
Amazon DynamoDB

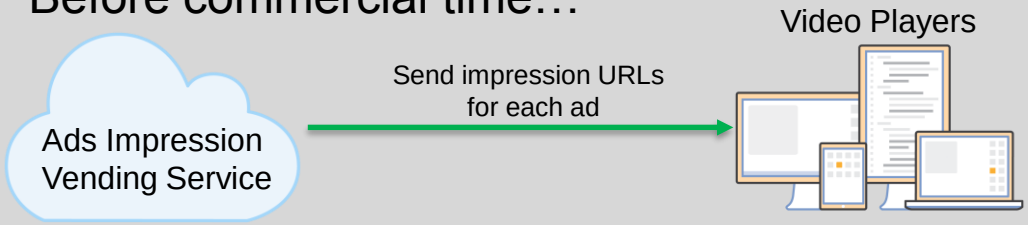
CloudWatch

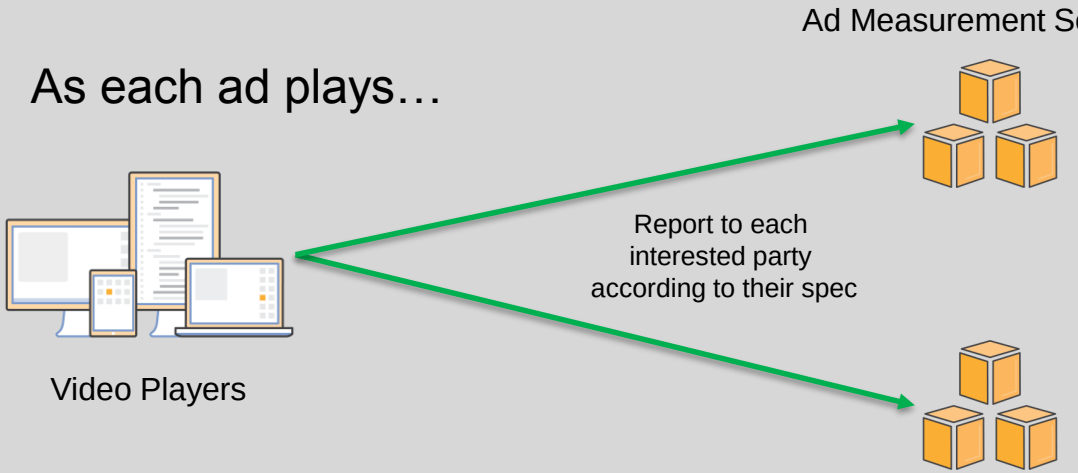
# Amazon Video Live Workflow



# Ad Reporting

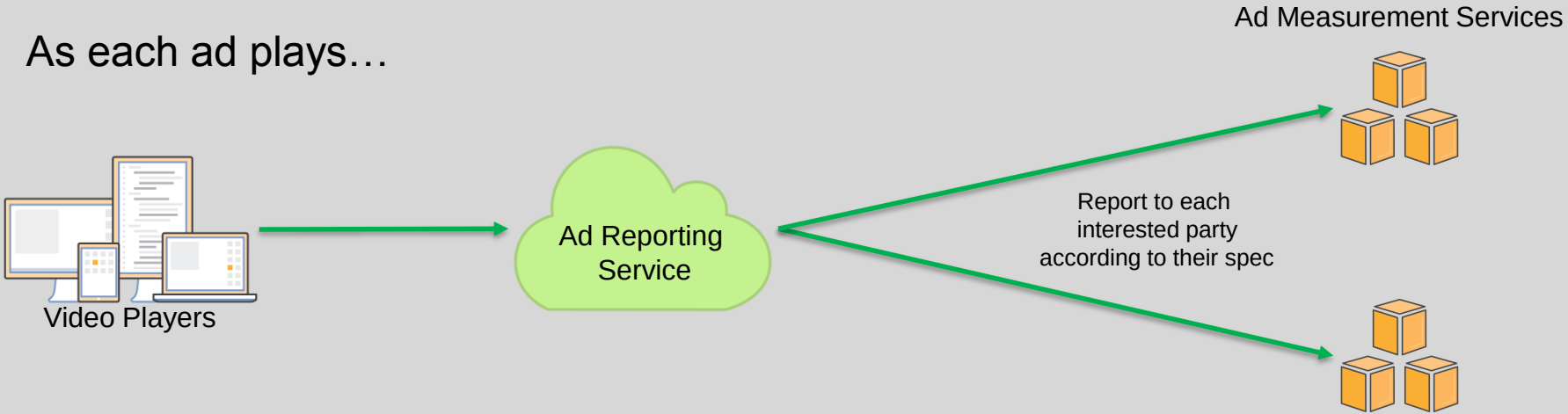
Before commercial time…



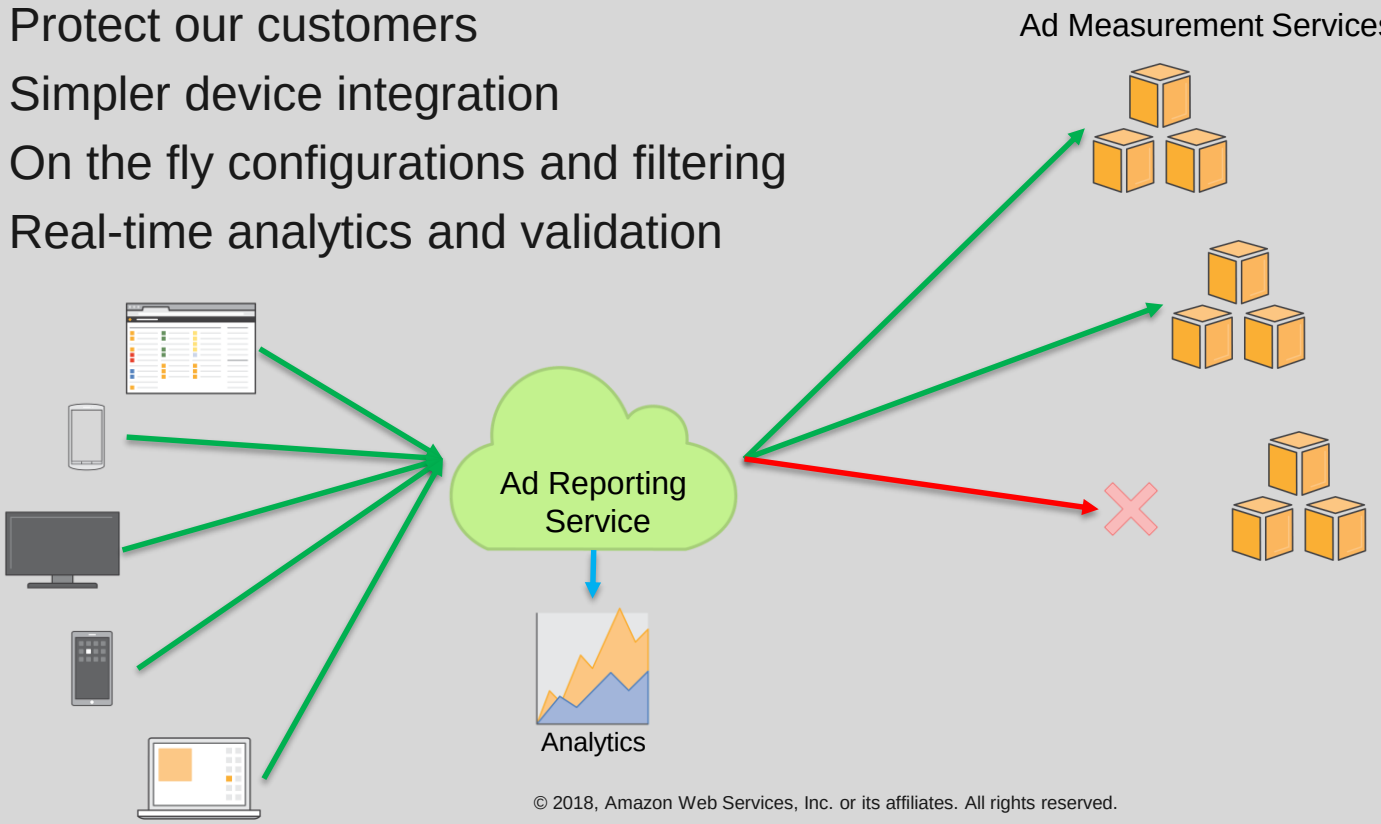
# What is server-side ad reporting?

As each ad plays…



# Why server-side?

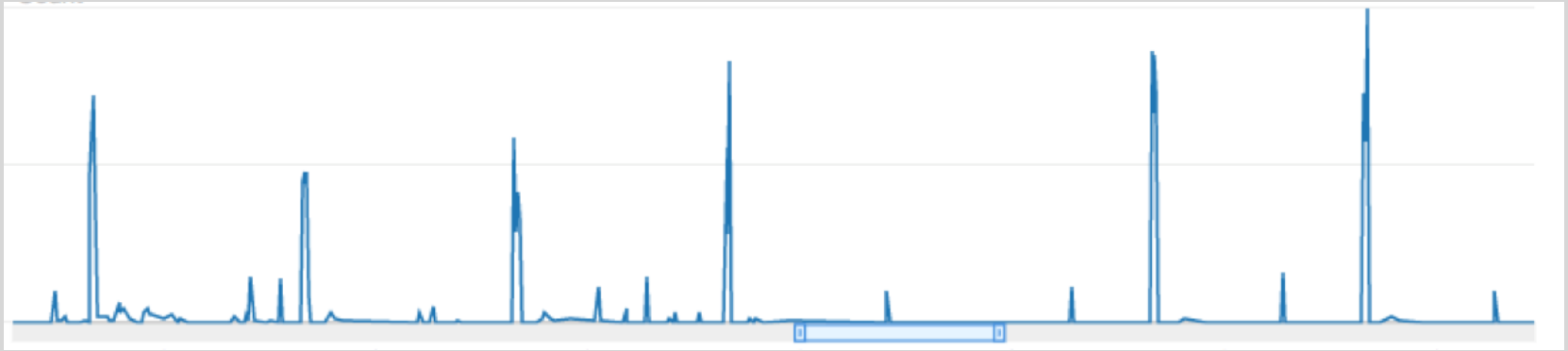
- Protect our customers
- Simpler device integration
- On the fly configurations and filtering
- Real-time analytics and validation



# Challenges: Intermittent

**Problem:** Thursdays are great! Other days not so much…

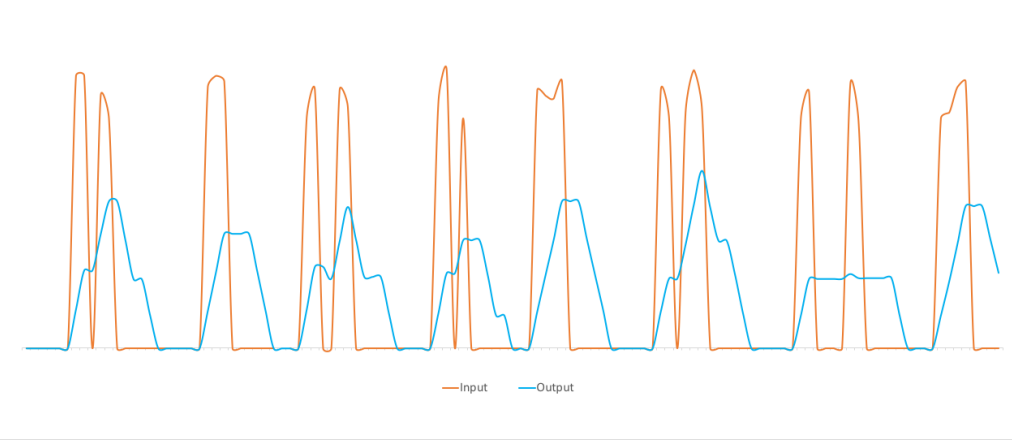
**Solution:** AWS Lambda



# Challenges: Spiky

**Problem:** Ads play for everyone simultaneously.

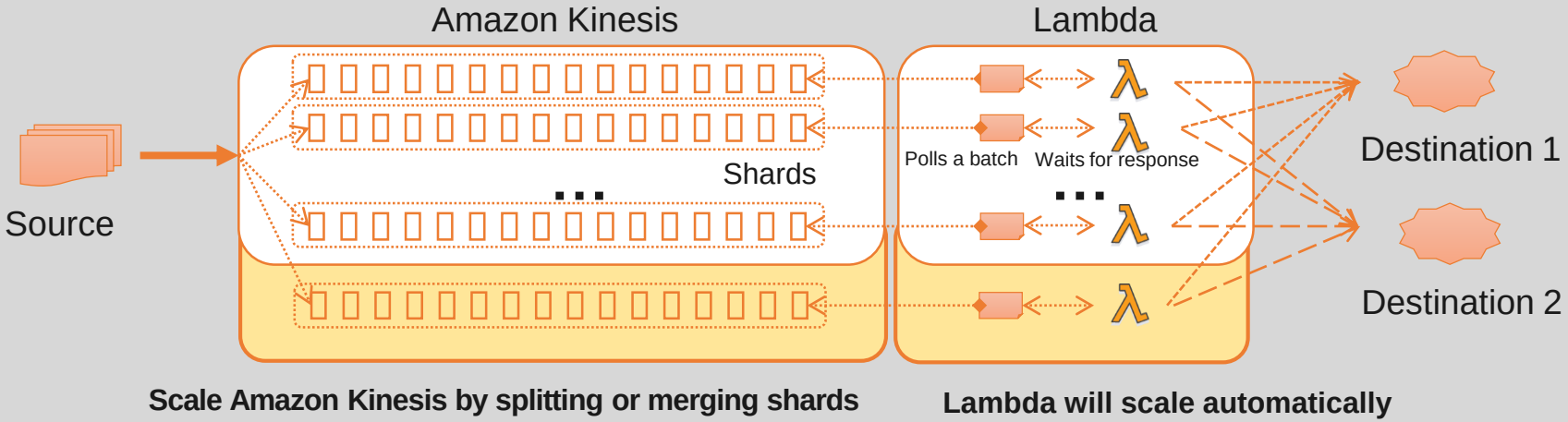
**Solution:** Amazon Kinesis



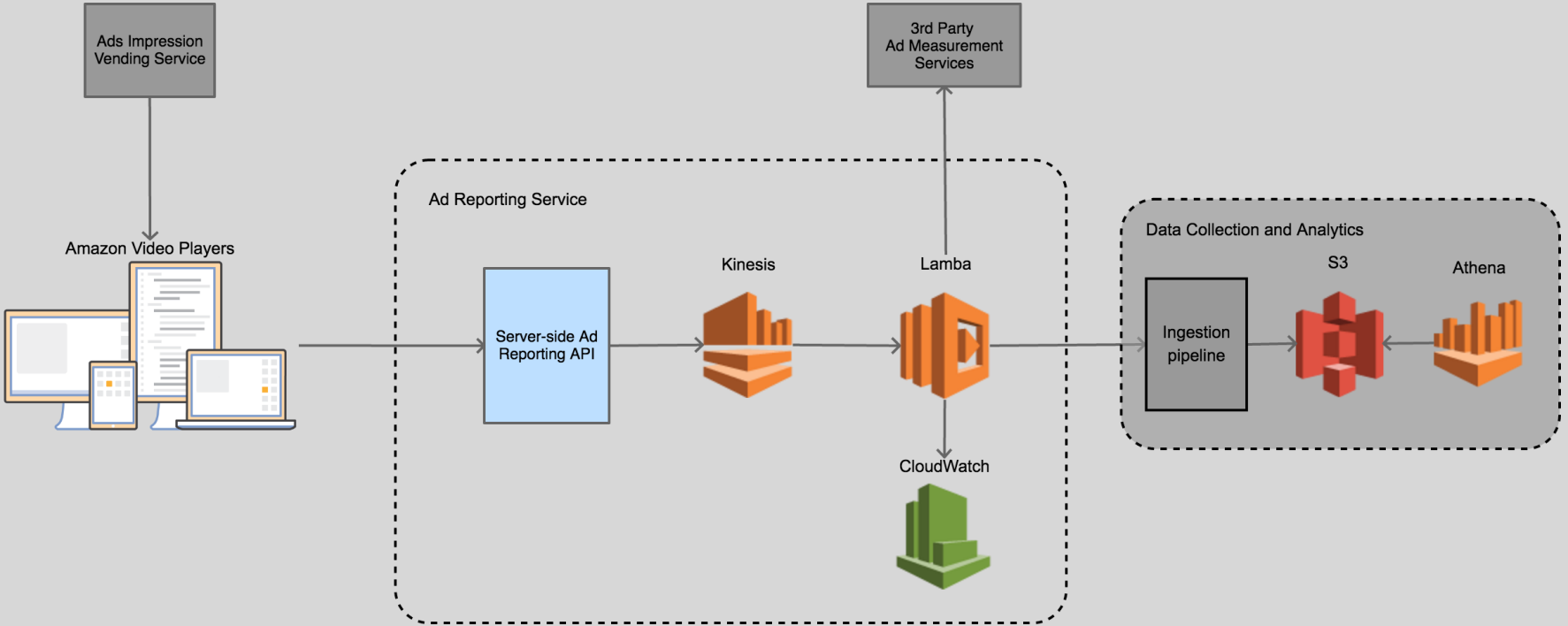
# Challenges: Scale

**Problem:** Millions of viewers = millions of TPS

**Solution:** Amazon Kinesis shards + batching



# Architecture at a glance



# Launch Results

- Only 3 months from paper to launch
- Reduced time spent on developing metrics and alarms
- Reduced cost of business operations by more than 90%
- Fast iterations and changes
- Easily extensible

aws SUMMIT

# Best Practices and Lessons Learned

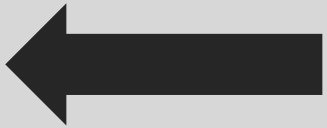
# Efficient Function Code

- Avoid “fat”/monolithic functions
- Control the dependencies in your function's deployment package
- Optimize for your language
    - Node – Browserfy, Minify

# JAVA – Scope Your POM File

```
<dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>com.amazonaws</groupId>
        <artifactId>aws-java-sdk-bom</artifactId>
        <version>2.10.10</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
<dependencies>
<dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-s3</artifactId>
      <version>1.10.5</version>
    </dependency>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-dynamodb</artifactId>
      <version>1.10.10</version>
    </dependency>
  <dependencies>
```

Maven Bill Of Materials (BOM) module for AWS SDK



Select service dependencies only

Avoid aws-java-sdk directly!

# Ephemeral Function Environment

- Lambda processes a single event per-container
- No need for non-blocking execution on the frontend
- REMEMBER – containers are reused
  - Lazily load variables in the global scope
  - Don't load it if you don't need it – cold starts are affected

```
import boto3

client = None

def my_handler(event, context):
    global client
    if not client:
        client =
boto3.client("s3")

# process
```

# Smart Resource Allocation

Match resource allocation (up to 3 GB!) to logic

Stats for Lambda function that calculates **1000 times** all prime numbers **<= 1000000**

| | | |
|---|---|---|
| **128 MB** | 11.722965sec | $0.024628 |
| **256 MB** | 6.678945sec | $0.028035 |
| **512 MB** | 3.194954sec | $0.026830 |
| **1024 MB** | 1.465984sec | $0.024638 |

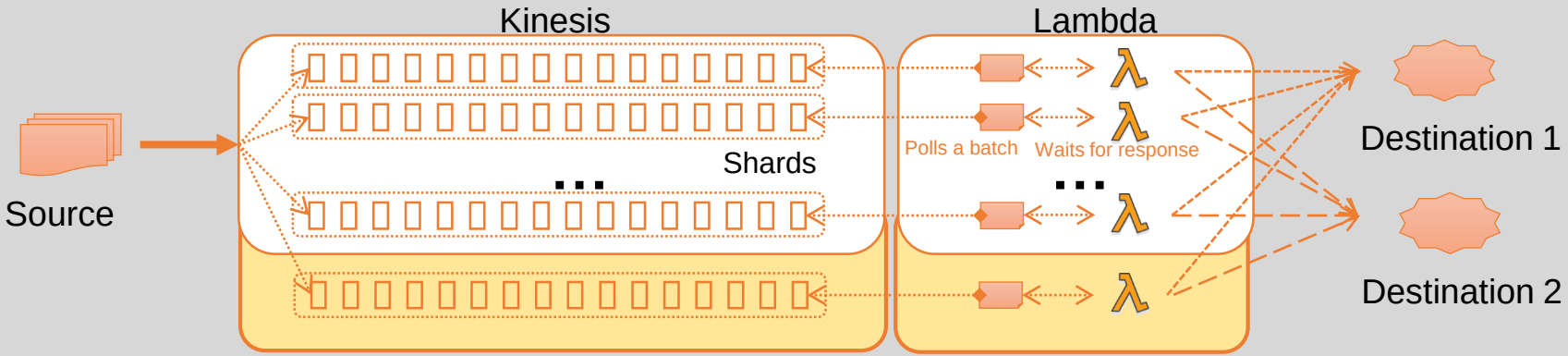
# Multithreading?

- <1.8 GB is still single core
  - CPU bound workloads won’t see gains – processes share same resources
- >1.8 GB is muti-core
  - CPU bound workloads will gains, but need to multi thread
- I/O bound workloads **will** likely see gains
  - e.g.,  parallel calculations to return

# Concurrency vs. Latency

- Increasing # of shards with even distribution allows increased concurrency
- If put / ingestion rate is greater than the theoretical throughput, consider increasing number of shards while optimizing function duration to increase throughput
- Effective theoretical throughput:

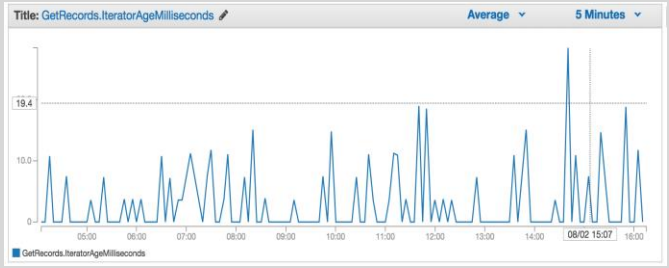
  ( # shards * batch size (MB) ) /  (function duration (s) * retries until expiry)



# Monitoring

## Monitoring Amazon Kinesis Streams

- GetRecords: (effective throughput)
- PutRecord: bytes, latency, records, etc.
- GetRecords.IteratorAgeMilliseconds: how old your last processed records were

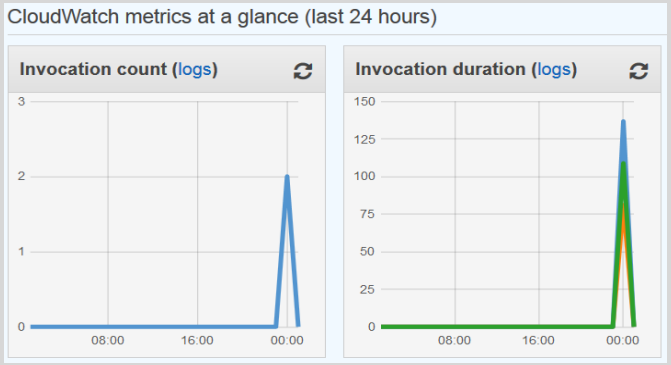


# Monitoring

## Monitoring Lambda functions

Invocation count: Time function invoked

- Duration: Execution/processing time
- Error count: Number of Errors
- Throttle count: Number of time function throttled
- Iterator Age: Time elapsed from batch received & final record written to stream



# Other Lessons Learned

- Optimize cost by aggregating to 25-KB payload size
- When re-sharding, try scaling in multiple of 2 or halves
- Compressing kinesis payloads can save on bandwidth
- Optimize Lambda GetRecord batch sizes

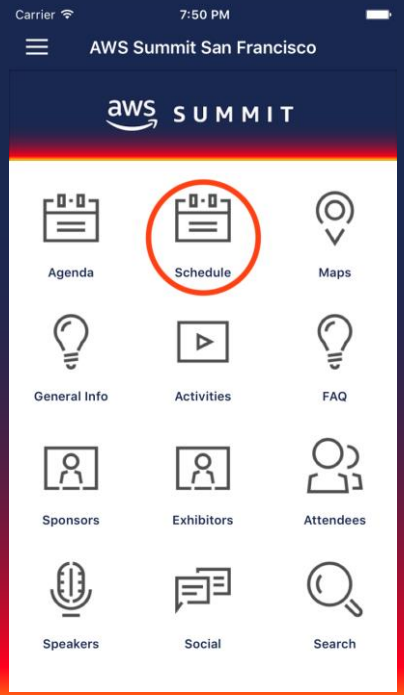
aws SUMMIT

# Thank you!

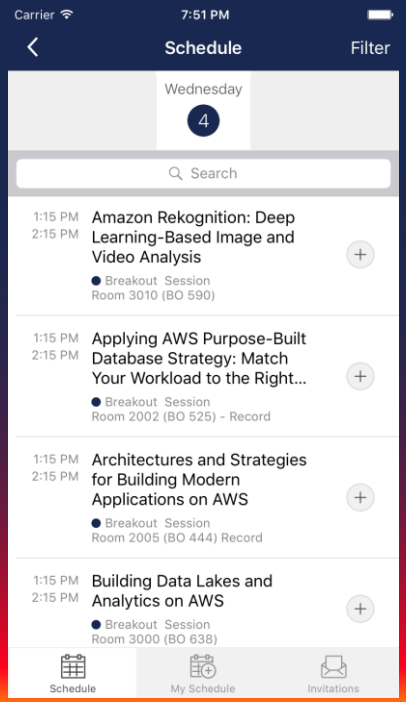
# Please complete the session survey in the summit mobile app.

# Submit Session Feedback

1. Tap the **Schedule** icon.




2. Select the session you attended.




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