GPSTEC313

AWS re:INVENT

GPS: Real-Time Data Processing with AWS Lambda Quickly, at Scale, and What Comes Next?

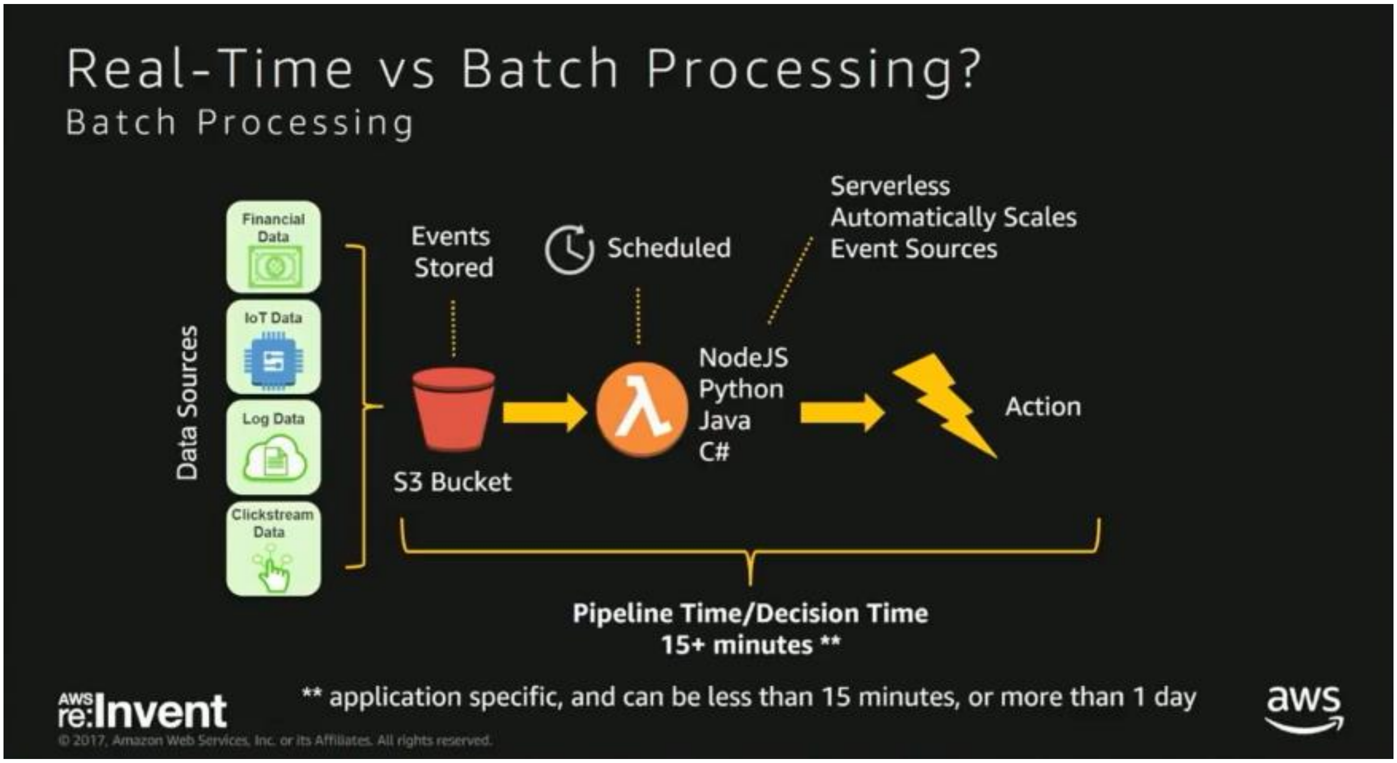
Juan Villa | Partner Solutions Architect

November 27, 2017

Real-time data processing is a powerful technique that allows businesses to make agile automated decisions. This process is particularly powerful when applied to workloads like ***security***, analyzing access logs, parsing audit logs, and monitoring API activity to detect behavior anomalies. Combined with automation, business can quickly take action to remediate security concerns, or even train a machine learning (ML) model. We explore different techniques for analyzing real-time streams on AWS using ***Lambda***, ***Amazon Kinesis***, Spark with ***Amazon EMR***, and ***Amazon DynamoDB***. We also cover best practices around short- and long-term storage and analysis of data and, briefly, the possibility of leveraging ML.

About Me—Juan Villa

- Partner Solutions Architect
- Focus on IoT, serverless, and machine learning
- Love to build cool software and hardware projects
- Ex-Software Developer



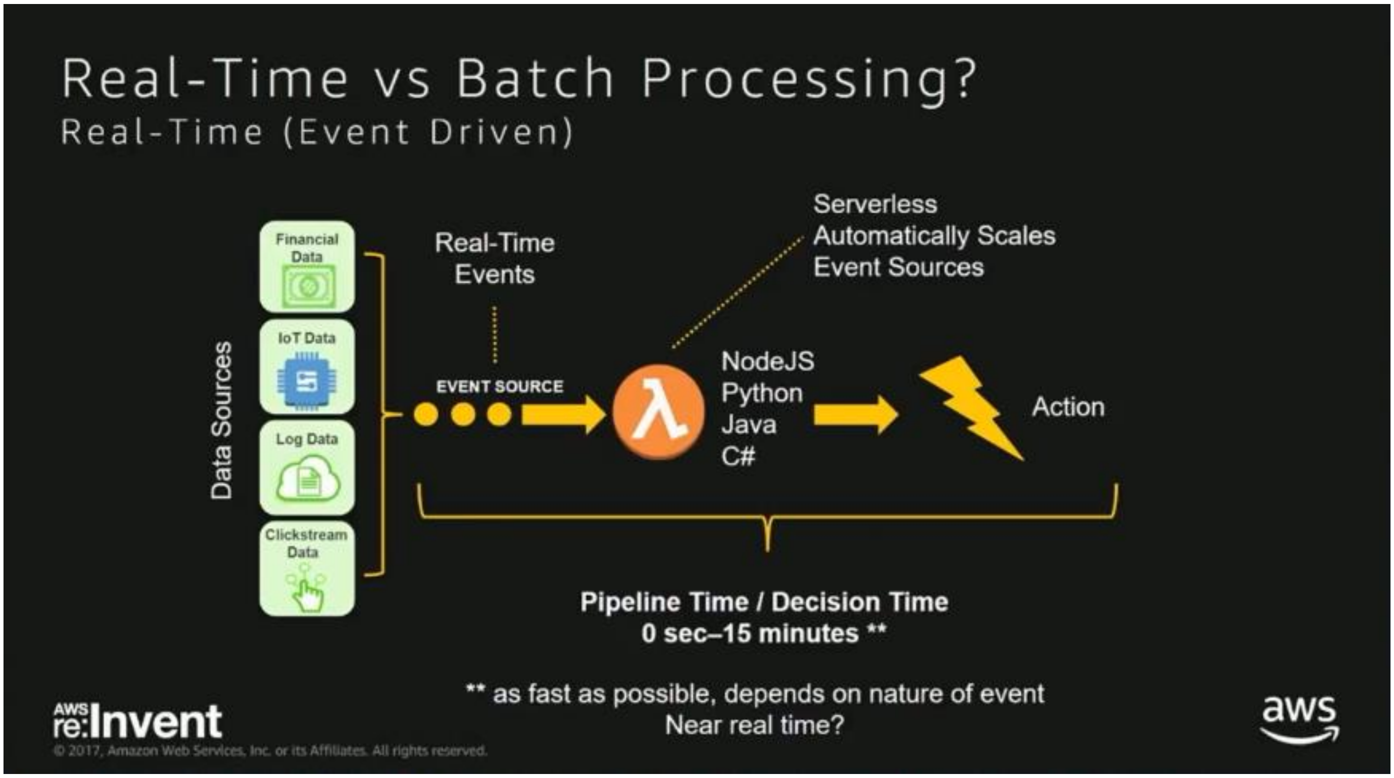
Real-Time vs Batch Processing?

Batch Processing

Pipeline Time/Decision Time
15+ minutes **

** application specific, and can be less than 15 minutes, or more than 1 day

Batch processing is about collecting events and processing them in some form of scheduled manner. ***Clickstream data*** events can be stored in an ***S3 bucket*** and then use lambda as the compute source to run specialized analytics on the events. All these data can be pulled and analyzed on some specified schedule.



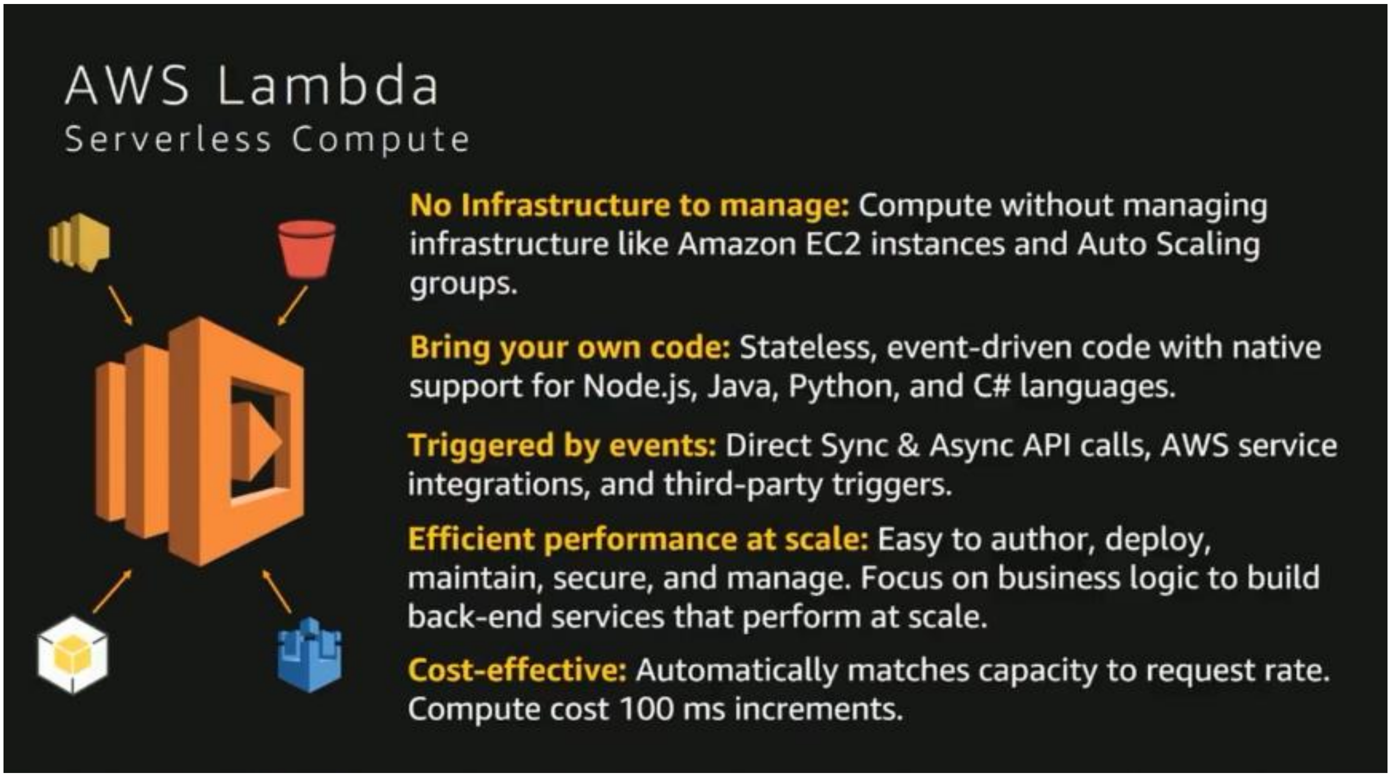
Real-Time vs Batch Processing?

Real-Time (Event Driven)

Pipeline Time / Decision Time
0 sec–15 minutes **

** as fast as possible, depends on nature of event
Near real time?

For ***Real-Time processing***, we are going to be processing the message in real-time before storing it in ***S3*** buckets.



AWS Lambda

Serverless Compute

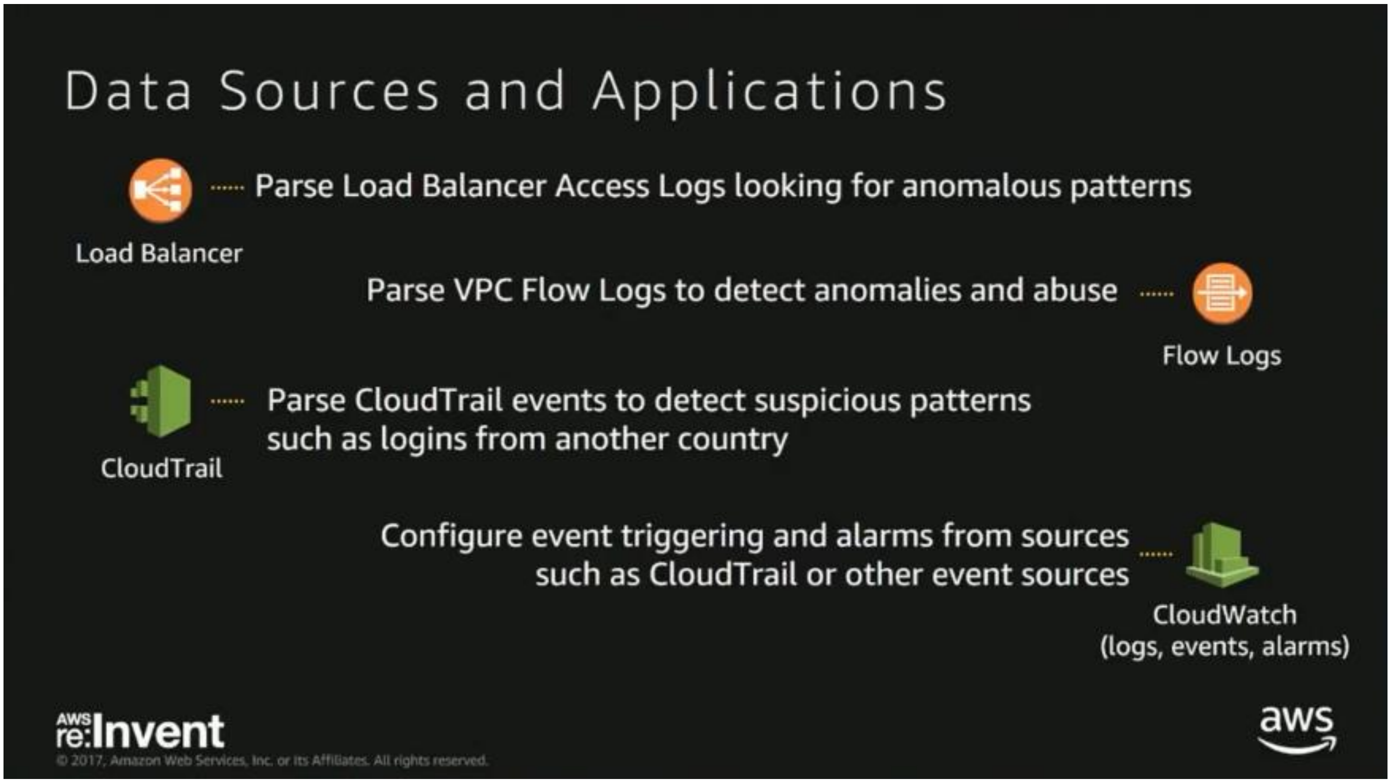
No Infrastructure to manage: Compute without managing infrastructure like Amazon EC2 instances and Auto Scaling groups.

Bring your own code: Stateless, event-driven code with native support for Node.js, Java, Python, and C# languages.

Triggered by events: Direct Sync & Async API calls, AWS service integrations, and third-party triggers.

Efficient performance at scale: Easy to author, deploy, maintain, secure, and manage. Focus on business logic to build back-end services that perform at scale.

Cost-effective: Automatically matches capacity to request rate. Compute cost 100 ms increments.



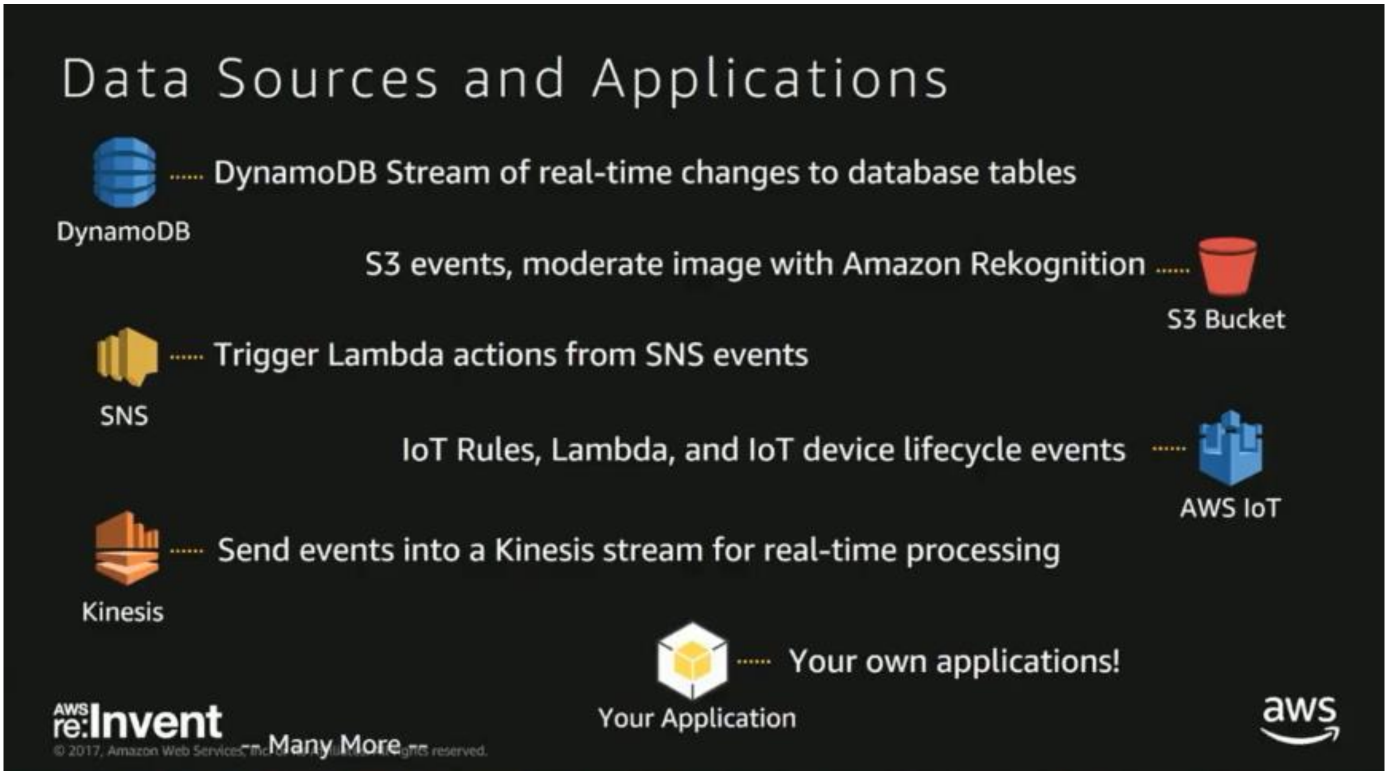
Data Sources and Applications

Load Balancer ······ Parse Load Balancer Access Logs looking for anomalous patterns

Parse VPC Flow Logs to detect anomalies and abuse ······ Flow Logs

CloudTrail ······ Parse CloudTrail events to detect suspicious patterns such as logins from another country

Configure event triggering and alarms from sources such as CloudTrail or other event sources ······ CloudWatch (logs, events, alarms)



Data Sources and Applications

DynamoDB Stream of real-time changes to database tables

S3 events, moderate image with Amazon Rekognition

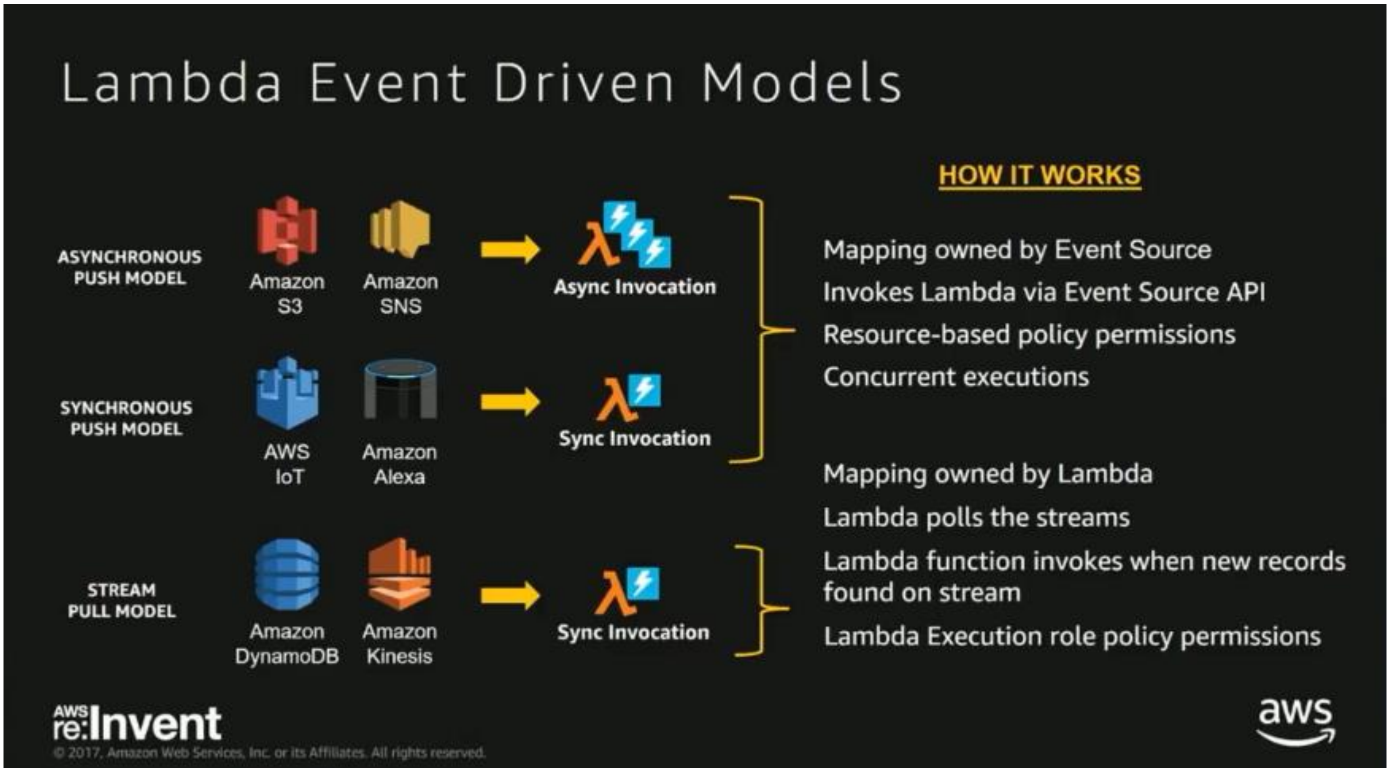
Trigger Lambda actions from SNS events

IoT Rules, Lambda, and IoT device lifecycle events

Send events into a Kinesis stream for real-time processing

Your own applications!

-- Many More --



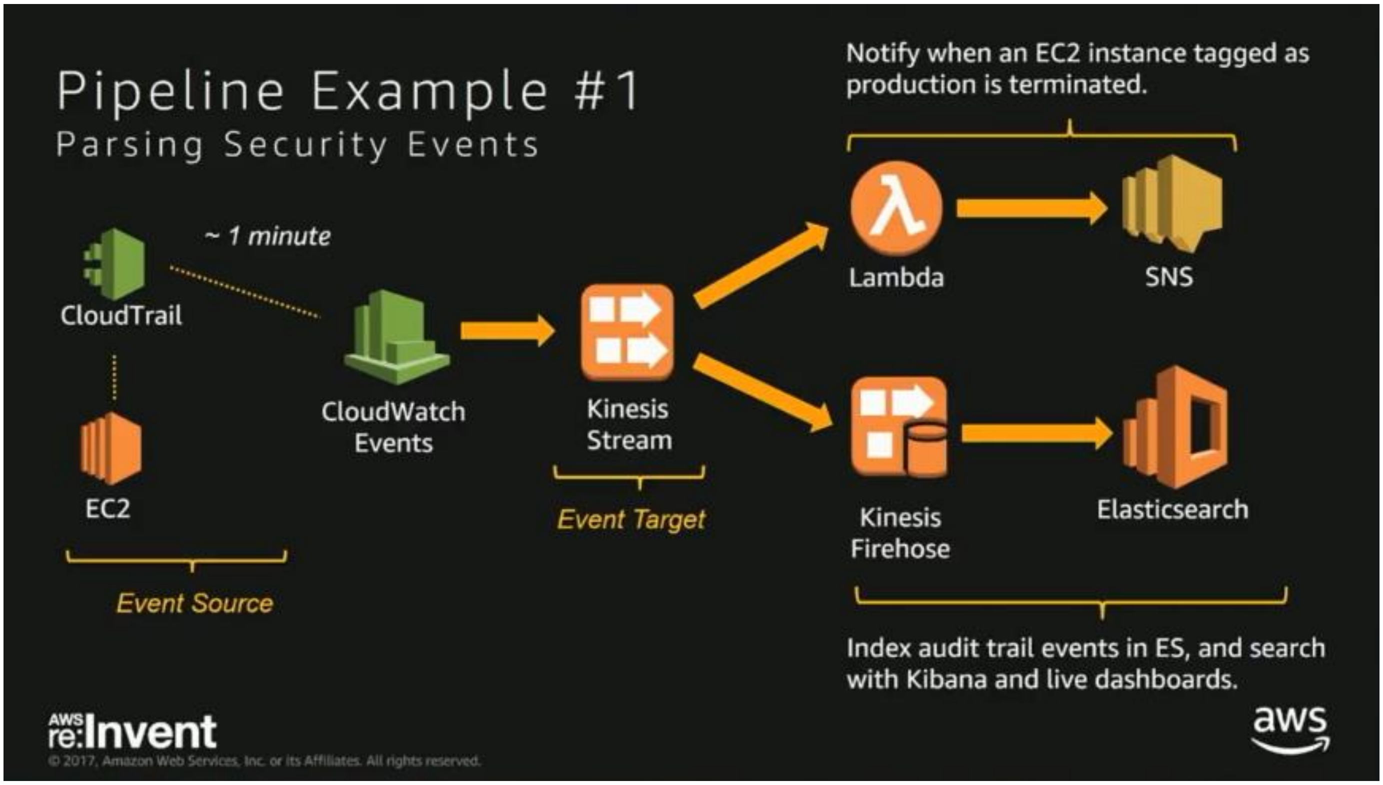
Lambda Event Driven Models

HOW IT WORKS

Mapping owned by Event Source
Invokes Lambda via Event Source API
Resource-based policy permissions
Concurrent executions

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

These are the 2 types of invocations models for triggering a lambda function. Sync invocation is when you have to wait on the lambda to get the result back and use further down the pipeline.



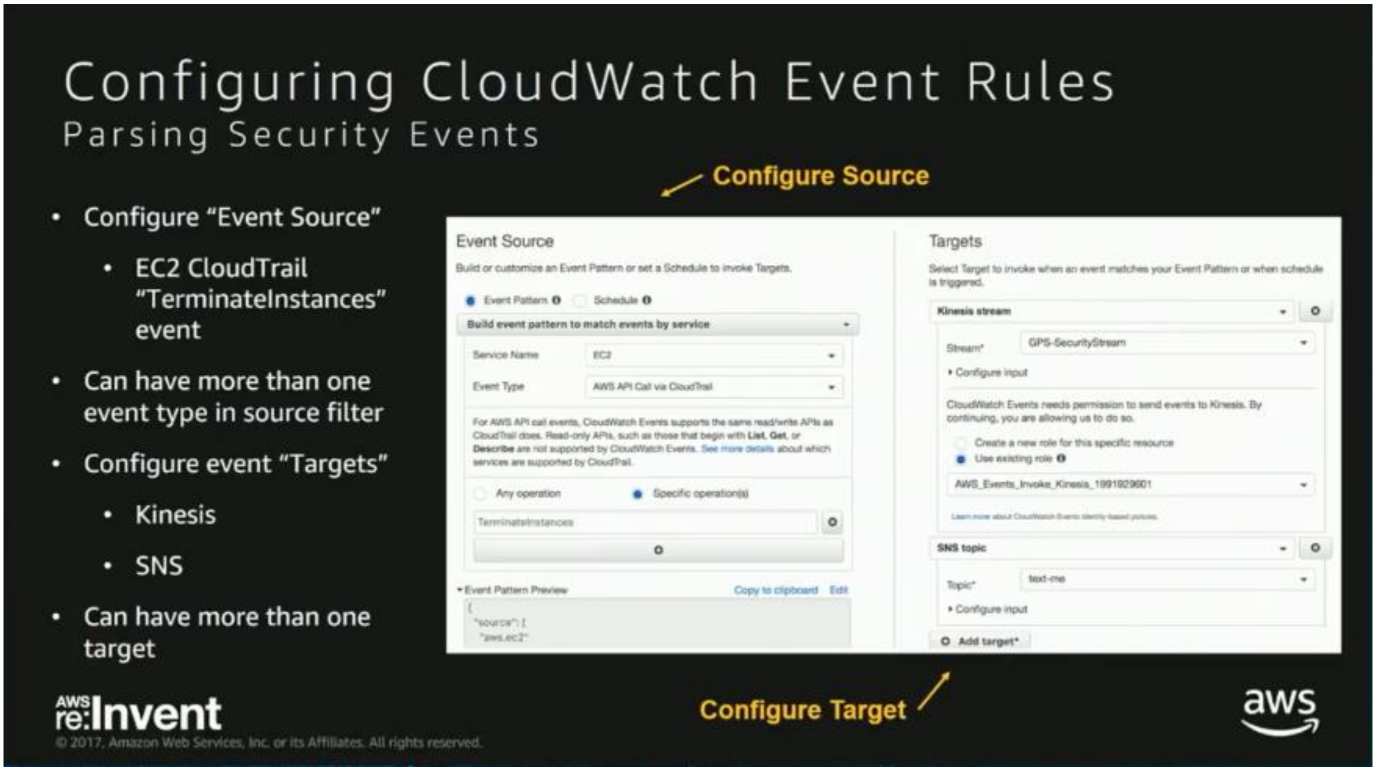
Pipeline Example #1

Parsing Security Events

Notify when an EC2 instance tagged as production is terminated.

Index audit trail events in ES, and search with Kibana and live dashboards.

We are going to turn on and use ***CloudTrail*** monitor all your AWS ***EC2*** calls, this will let us know if something was turned off and by whom. We then set up a ***CloudWatch event*** on a specific event like ***EC2 Prod instance termination*** from the ***CloudTrail logs*** and put that event in a ***Kinesis Stream***. We then have 2 consumers that are going to be consuming data coming from the Kinesis stream, a ***lambda function*** will put the message into an ***SNS*** topic, and a ***Kinesis Firehose*** that will put the event into ***ElasticSearch*** for searching and display using Kibana



Configuring CloudWatch Event Rules

Parsing Security Events

- Configure "Event Source"
  - EC2 CloudTrail "TerminateInstances" event
- Can have more than one event type in source filter
- Configure event "Targets"
  - Kinesis
  - SNS
- Can have more than one target

Configure Source

Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

● Event Pattern ○ Schedule

Build event pattern to match events by service

Service Name: EC2

Event Type: AWS API Call via CloudTrail

For AWS API call events, CloudWatch Events supports the same read/write APIs as CloudTrail does. Read-only APIs, such as those that begin with **List**, **Get**, or **Describe** are not supported by CloudWatch Events. See more details about which services are supported by CloudTrail.

○ Any operation ● Specific operation(s)

TerminateInstances

Event Pattern Preview

Copy to clipboard Edit

```
{
  "source": [
    "aws.ec2"
```

Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

Kinesis stream

Stream*: GPS-SecurityStream

Configure input

CloudWatch Events needs permission to send events to Kinesis. By continuing, you are allowing us to do so.

○ Create a new role for this specific resource

● Use existing role

AWS_Events_Invoke_Kinesis_1991929601

Learn more about CloudWatch Events identity-based policies.

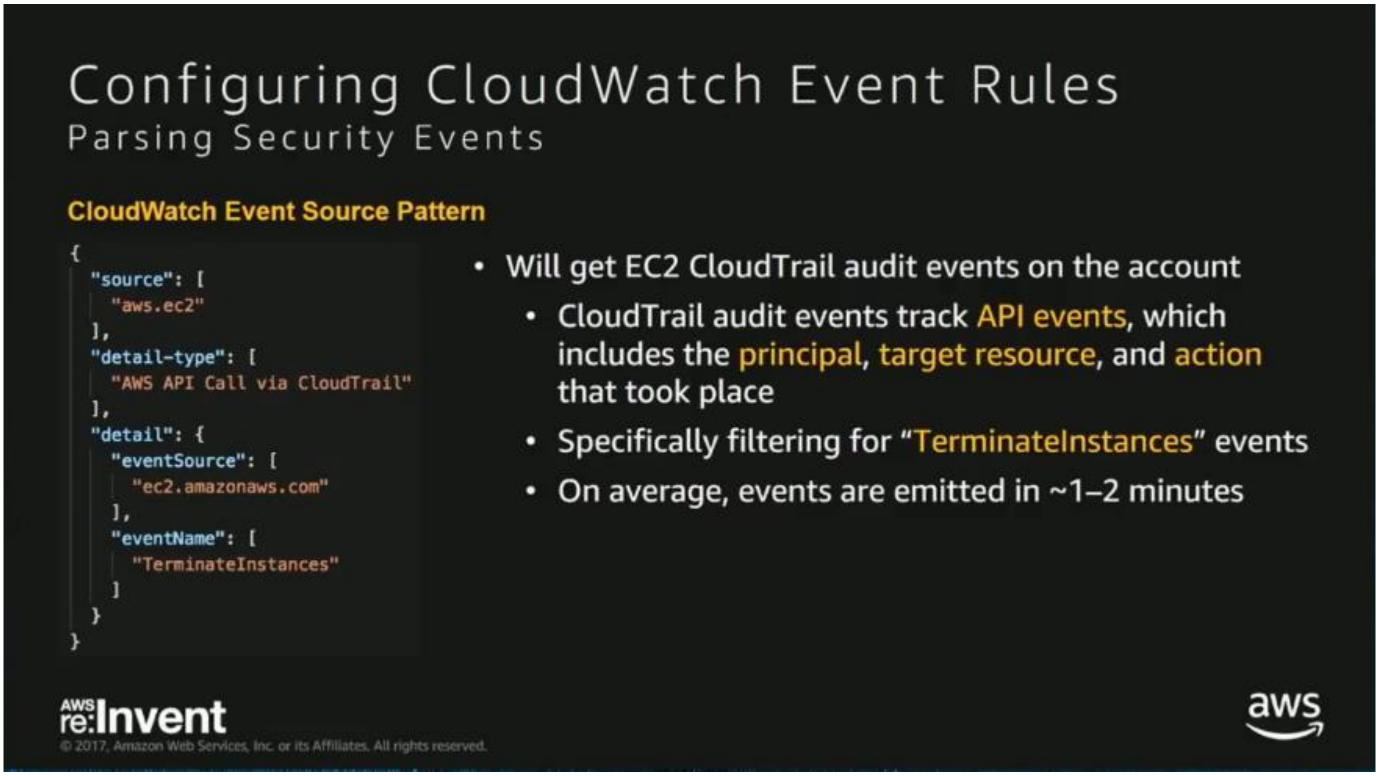
SNS topic

Topic*: text-me

Configure input

Add target*

Configure Target



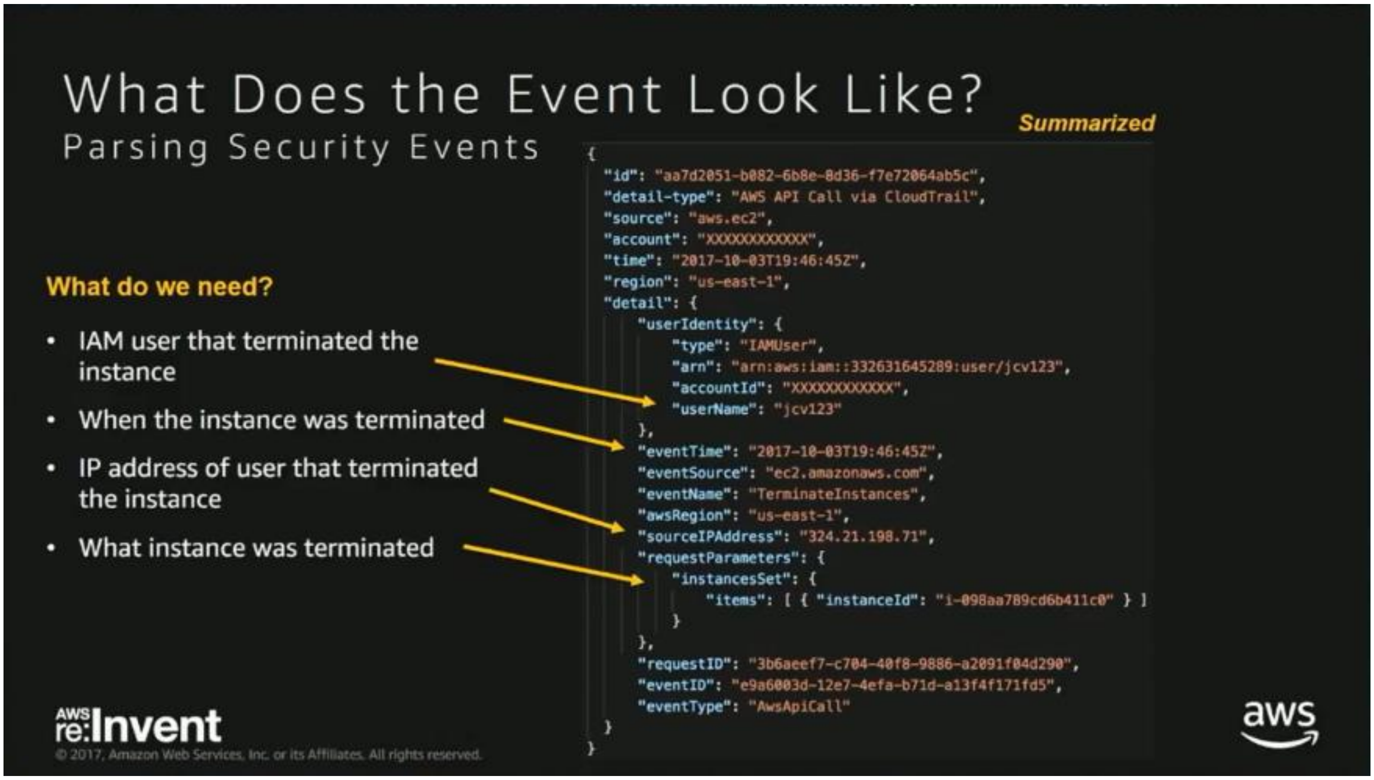
Configuring CloudWatch Event Rules

Parsing Security Events

CloudWatch Event Source Pattern

```
{
  "source": [
    "aws.ec2"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "ec2.amazonaws.com"
    ],
    "eventName": [
      "TerminateInstances"
    ]
  }
}
```

- Will get EC2 CloudTrail audit events on the account
  - CloudTrail audit events track API events, which includes the principal, target resource, and action that took place
  - Specifically filtering for "TerminateInstances" events
  - On average, events are emitted in ~1–2 minutes



What Does the Event Look Like?

Parsing Security Events

What do we need?

- IAM user that terminated the instance
- When the instance was terminated
- IP address of user that terminated the instance
- What instance was terminated

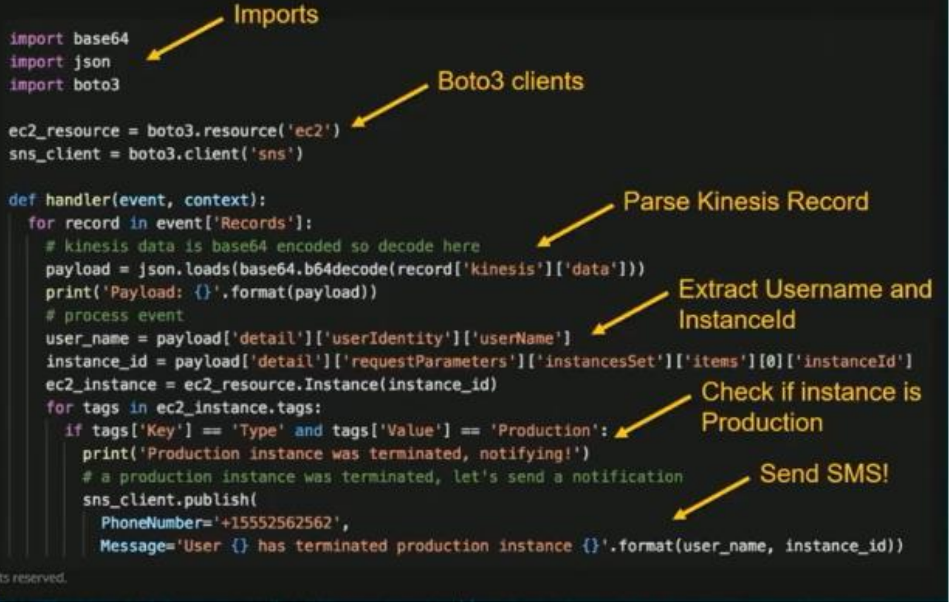
Summarized

```
{
  "id": "aa7d2051-b082-6b8e-8d36-f7e72064ab5c",
  "detail-type": "AWS API Call via CloudTrail",
  "source": "aws.ec2",
  "account": "XXXXXXXXXXXX",
  "time": "2017-10-03T19:46:45Z",
  "region": "us-east-1",
  "detail": {
    "userIdentity": {
      "type": "IAMUser",
      "arn": "arn:aws:iam::332631645289:user/jcv123",
      "accountId": "XXXXXXXXXXXX",
      "userName": "jcv123"
    },
    "eventTime": "2017-10-03T19:46:45Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "TerminateInstances",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "324.21.198.71",
    "requestParameters": {
      "instancesSet": {
        "items": [ { "instanceId": "i-098aa789cd6b411c0" } ]
      }
    },
    "requestID": "3b6aeef7-c704-40f8-9886-a2091f04d290",
    "eventID": "e9a6003d-12e7-4efa-b71d-a13f4f171fd5",
    "eventType": "AwsApiCall"
  }
}
```

Lambda Function

Parsing Security Events

- Send SMS message to administrator when a user terminates an instance tagger as a production instance
- We could also send a message to a Slack Channel, or Pager Duty, or send an email
- Got notified in 30 seconds!



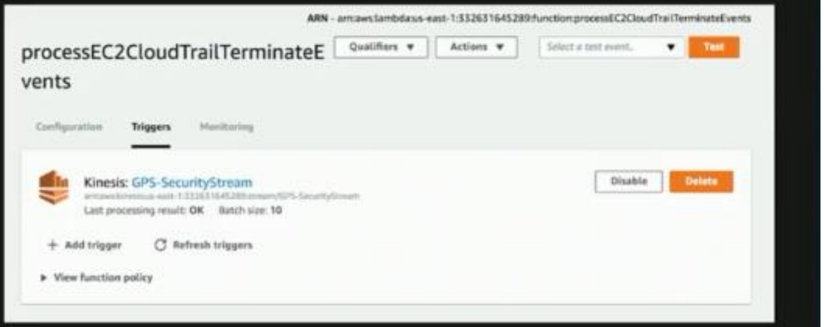
```
import base64
import json
import boto3

ec2_resource = boto3.resource('ec2')
sns_client = boto3.client('sns')

def handler(event, context):
  for record in event['Records']:
    # kinesis data is base64 encoded so decode here
    payload = json.loads(base64.b64decode(record['kinesis']['data']))
    print('Payload: {}'.format(payload))
    # process event
    user_name = payload['detail']['userIdentity']['userName']
    instance_id = payload['detail']['requestParameters']['instancesSet']['items'][0]['instanceId']
    ec2_instance = ec2_resource.Instance(instance_id)
    for tags in ec2_instance.tags:
      if tags['Key'] == 'Type' and tags['Value'] == 'Production':
        print('Production instance was terminated, notifying!')
        # a production instance was terminated, let's send a notification
        sns_client.publish(
          PhoneNumber='+15552562562',
          Message='User {} has terminated production instance {}'.format(user_name, instance_id))
```

Imports
Boto3 clients
Parse Kinesis Record
Extract Username and InstanceId
Check if instance is Production
Send SMS!

Executing the Lambda Function

Parsing Security Events

- Configure a Kinesis trigger for our Lambda function
- Lambda service handles the Kinesis ingestion and execution of the Lambda function
- Also handles errors and retries



ARN - arn:aws:lambda:us-east-1:332631645289:function:processEC2CloudTrailTerminateEvents

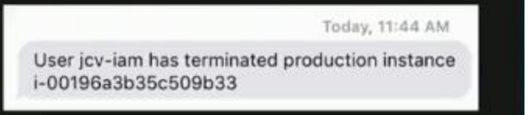
processEC2CloudTrailTerminateEvents

Kinesis: GPS-SecurityStream

Last processing result: OK Batch size: 10

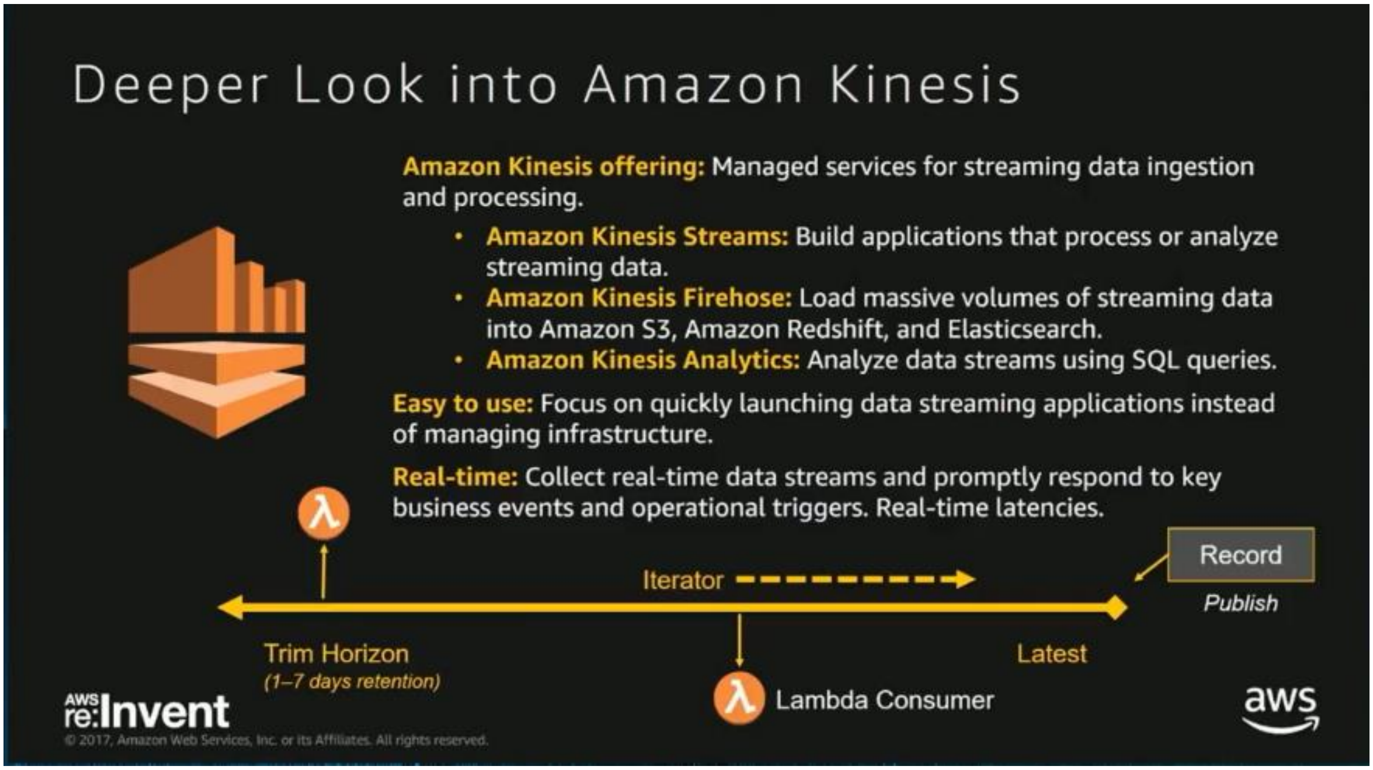
- Finally, we get our text message when an instance is terminated

Huzzah!



Today, 11:44 AM

User jcv-iam has terminated production instance i-00196a3b35c509b33



Deeper Look into Amazon Kinesis

Amazon Kinesis offering: Managed services for streaming data ingestion and processing.

- **Amazon Kinesis Streams:** Build applications that process or analyze streaming data.
- **Amazon Kinesis Firehose:** Load massive volumes of streaming data into Amazon S3, Amazon Redshift, and Elasticsearch.
- **Amazon Kinesis Analytics:** Analyze data streams using SQL queries.

Easy to use: Focus on quickly launching data streaming applications instead of managing infrastructure.

Real-time: Collect real-time data streams and promptly respond to key business events and operational triggers. Real-time latencies.

Iterator

Record

Publish

Trim Horizon

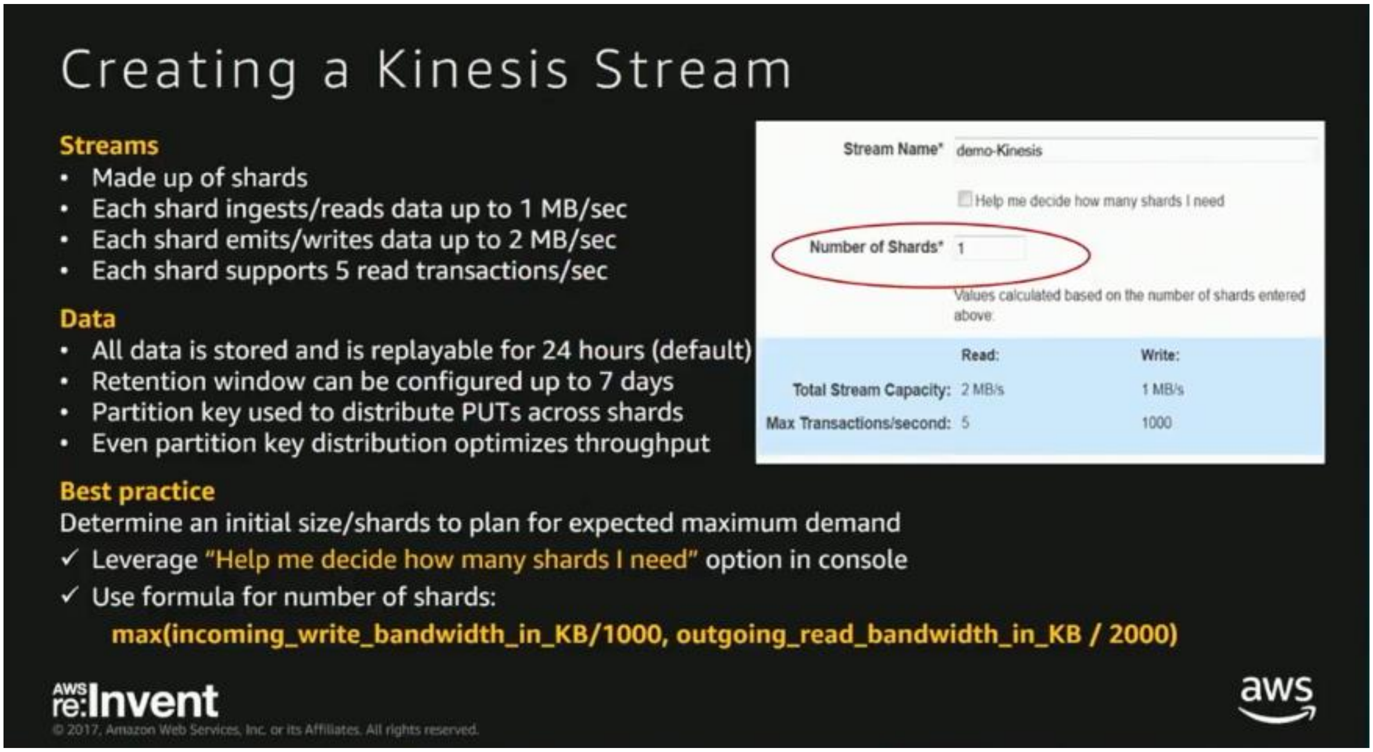
(1–7 days retention)

Latest

Lambda Consumer

AWS re:Invent

Lambda functions are synchronously invoked, meaning that there is a guarantee or message order processing. This is why there is the concept of retries to enable the stoppage of the real-time processing of your pipeline messages before they fall off the Trim Horizon. You can fix the issue with retries before messages start getting dropped at the trim horizon, then your processing will continue as expected.



Creating a Kinesis Stream

Streams

- Made up of shards
- Each shard ingests/reads data up to 1 MB/sec
- Each shard emits/writes data up to 2 MB/sec
- Each shard supports 5 read transactions/sec

Data

- All data is stored and is replayable for 24 hours (default)
- Retention window can be configured up to 7 days
- Partition key used to distribute PUTs across shards
- Even partition key distribution optimizes throughput

Best practice

Determine an initial size/shards to plan for expected maximum demand

✓ Leverage "Help me decide how many shards I need" option in console

✓ Use formula for number of shards:

max(incoming_write_bandwidth_in_KB/1000, outgoing_read_bandwidth_in_KB / 2000)

Stream Name* demo-Kinesis

□ Help me decide how many shards I need

Number of Shards* 1

Values calculated based on the number of shards entered above.

	Read:	Write:
Total Stream Capacity:	2 MB/s	1 MB/s
Max Transactions/second:	5	1000

AWS re:Invent

Creating Lambda Function

Memory

CPU allocation proportional to the memory configured
Increasing memory makes your code execute faster (if CPU bound)
Increasing memory allows for larger record sizes processed

Timeout

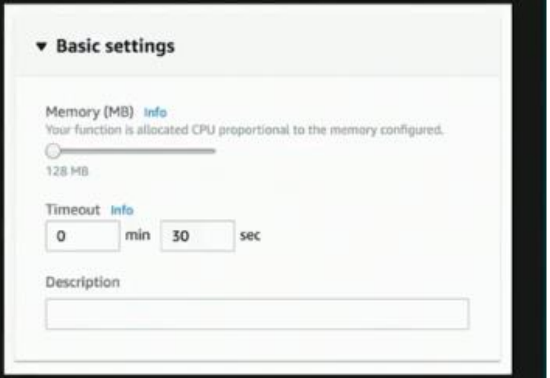
Increasing timeout allows for longer functions, but longer wait in case of errors

Permission model

Execution role defined for Lambda must have permission to access the stream

Retries

With Amazon Kinesis, Lambda retries until the data expires (24 hours)



▼ Basic settings

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.
128 MB

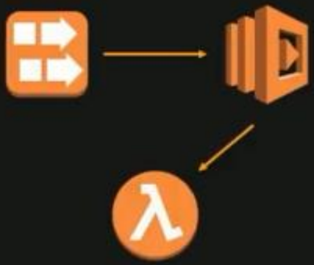
Timeout Info
0 min 30 sec

Description

Best practice

Write Lambda function code to be stateless
Instantiate AWS clients and database clients **outside** the scope of the function handler to take advantage of connection re-use.

Configuring Event Trigger



Amazon Kinesis mapped as event source in **Lambda**

Batch size

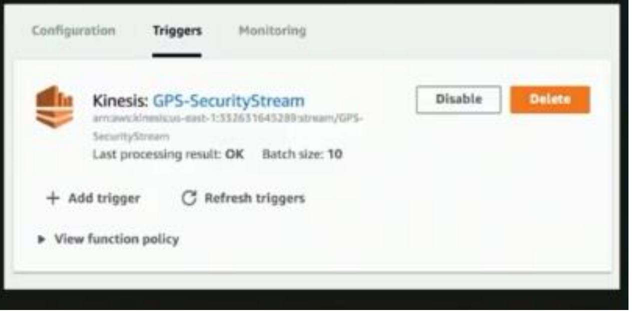
Max number of records that Lambda will send to one invocation
Not equivalent to effective batch size
Effective batch size is every 250 ms—calculated as:

MIN(records available, batch size, 6MB)

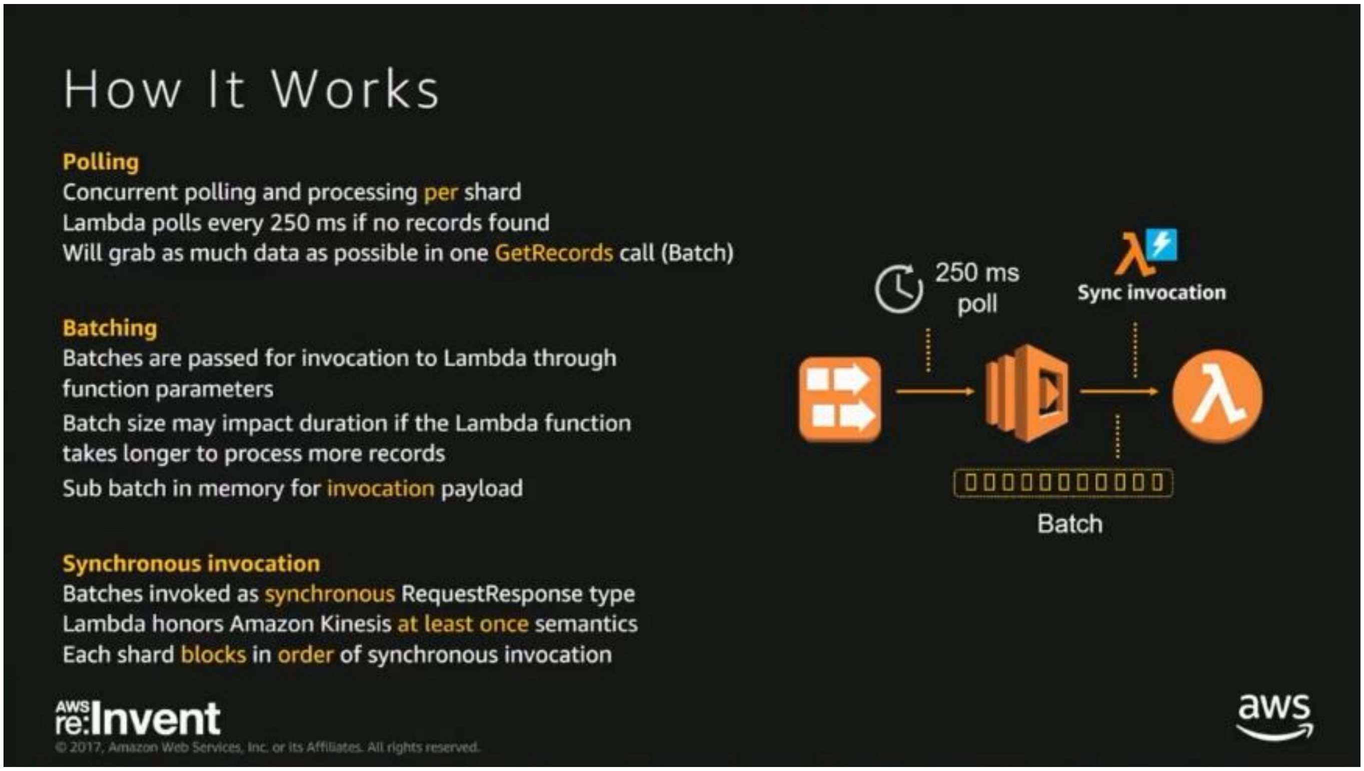
Increasing batch size allows fewer Lambda function invocations with more data processed per function

Best Practices

Set to "Trim Horizon" for reading from start of stream (all data)
Set to "Latest" for reading most recent data (LIFO) (latest data)



Kinesis: GPS-SecurityStream
arn:aws:kinesis:us-east-1:332631645289:stream/GPS-SecurityStream
Last processing result: OK Batch size: 10



How It Works

Polling

Concurrent polling and processing per shard

Lambda polls every 250 ms if no records found

Will grab as much data as possible in one GetRecords call (Batch)

Batching

Batches are passed for invocation to Lambda through function parameters

Batch size may impact duration if the Lambda function takes longer to process more records

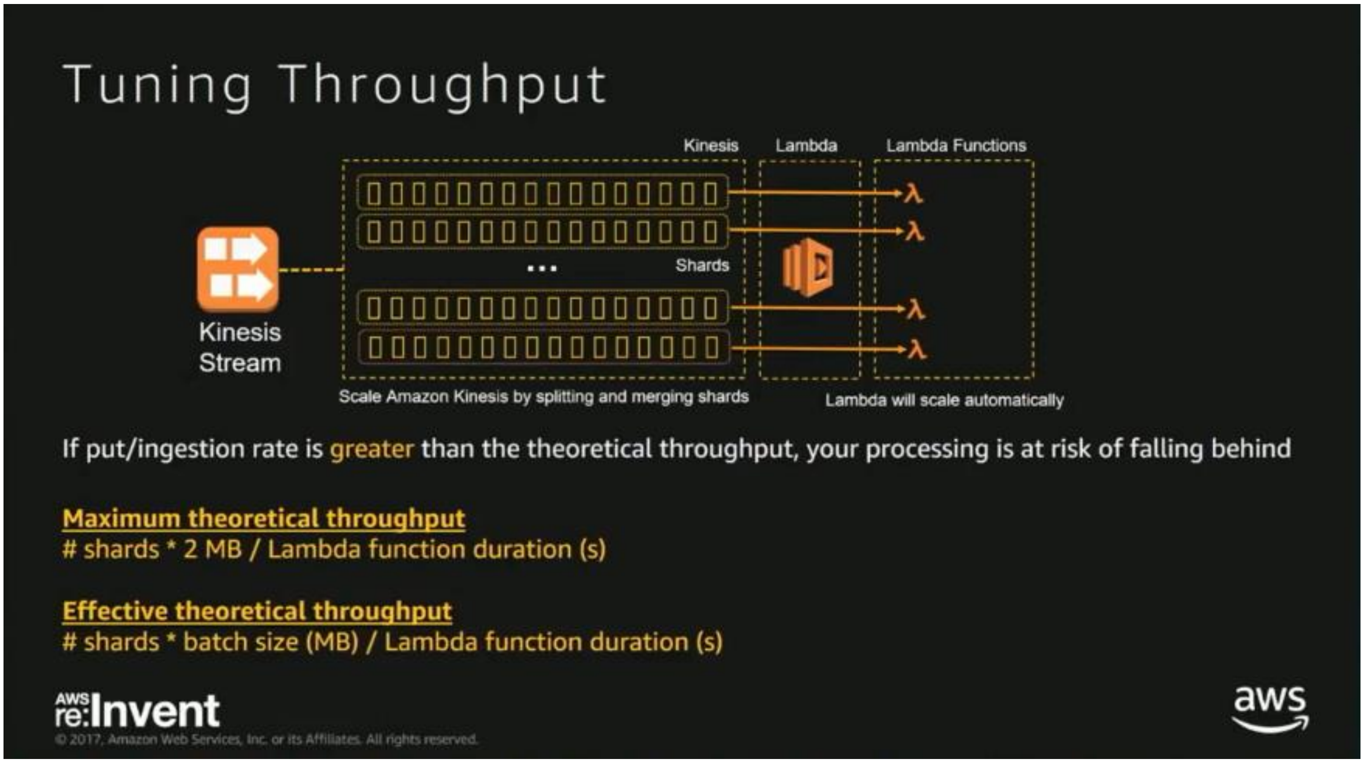
Sub batch in memory for invocation payload

Synchronous invocation

Batches invoked as synchronous RequestResponse type

Lambda honors Amazon Kinesis at least once semantics

Each shard blocks in order of synchronous invocation



Tuning Throughput

Kinesis
Lambda
Lambda Functions
Shards
Kinesis Stream
Scale Amazon Kinesis by splitting and merging shards
Lambda will scale automatically

If put/ingestion rate is greater than the theoretical throughput, your processing is at risk of falling behind

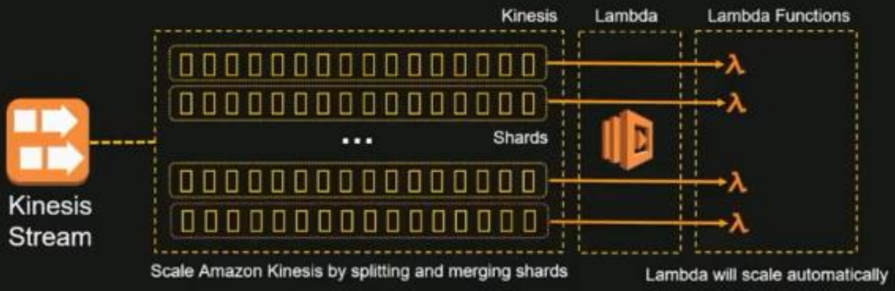
Maximum theoretical throughput

shards * 2 MB / Lambda function duration (s)

Effective theoretical throughput

shards * batch size (MB) / Lambda function duration (s)

Tuning Throughput with Retries


Kinesis
Lambda
Lambda Functions
Kinesis Stream
Shards
Scale Amazon Kinesis by splitting and merging shards
Lambda will scale automatically

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 back-off of up to 60 s

Effective theoretical throughput with retries

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

Monitoring Your Real-Time Pipeline

Monitoring Amazon Kinesis Streams

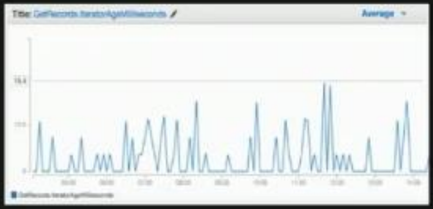
- GetRecords: (effective throughput)
- PutRecord: bytes, latency, records, etc.
- GetRecords.IteratorAgeMilliseconds: how old your last processed records were

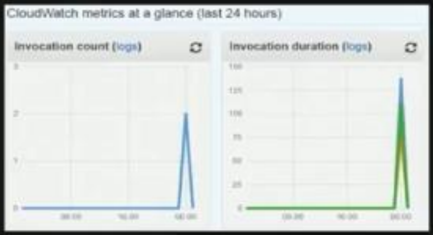
Monitoring Lambda functions

- 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 and final record written to stream

Debugging

- Review all metrics
- Make custom logs
- View RAM consumed
- Search for log events




CloudWatch metrics at a glance (last 24 hours)
Invocation count (logs)
Invocation duration (logs)

Log Streams

2015/05/21/2184a6f4cb3e4c81a7265418e0c4078c

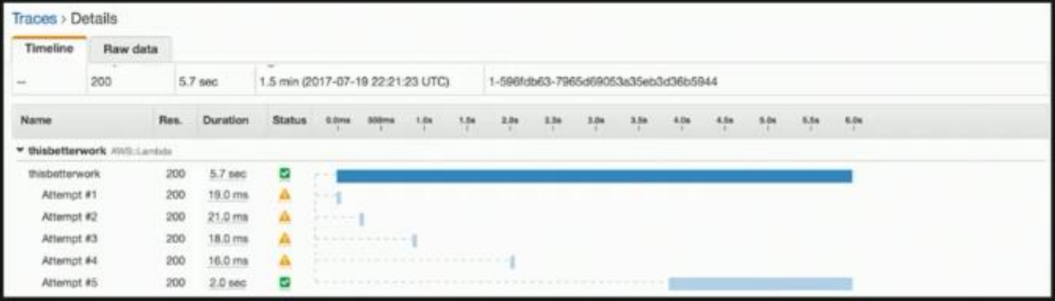
2015/05/21/3d07c7e60fc04acab257ee789bb667ef

2015/05/21/898f9b697db84b61a19879477d197679

Example Debugging with X-Ray

AWS X-Ray

Retries with Lambda/Exponential Back-offs



Traces › Details

Timeline | Raw data

–	200	5.7 sec	1.5 min (2017-07-19 22:21:23 UTC)	1-596fdb63-7965d69053a35eb3d36b5944

Name	Res.	Duration	Status
▾ thisbetterwork AWS::Lambda			
thisbetterwork	200	5.7 sec	✓
Attempt #1	200	19.0 ms	⚠
Attempt #2	200	21.0 ms	⚠
Attempt #3	200	18.0 ms	⚠
Attempt #4	200	16.0 ms	⚠
Attempt #5	200	2.0 sec	✓

- Total Duration = 6 seconds (five retries before success)
- Lambda Invocation Duration = 2.2 seconds
- Actual Throughput = 1 * (10000/6) = 1666 records/s
- Ideal Throughput = 1 * (10000/2.2) = 4545 records/s

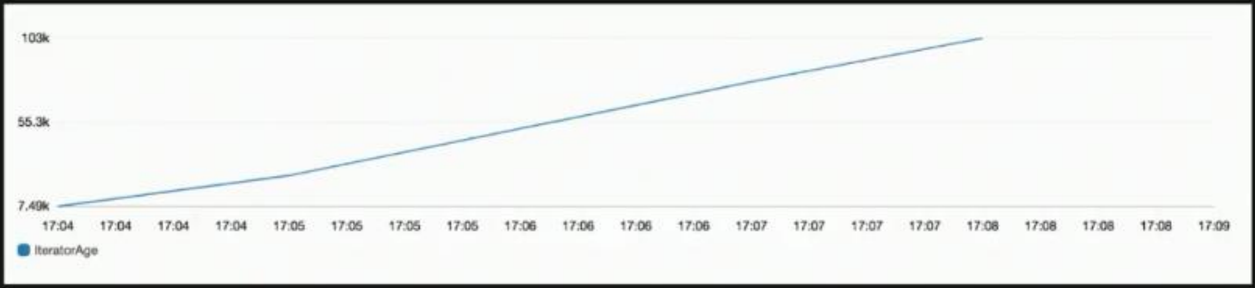
2.72x difference

AWS re:Invent

aws

Falling Behind

Not Processing Fast Enough

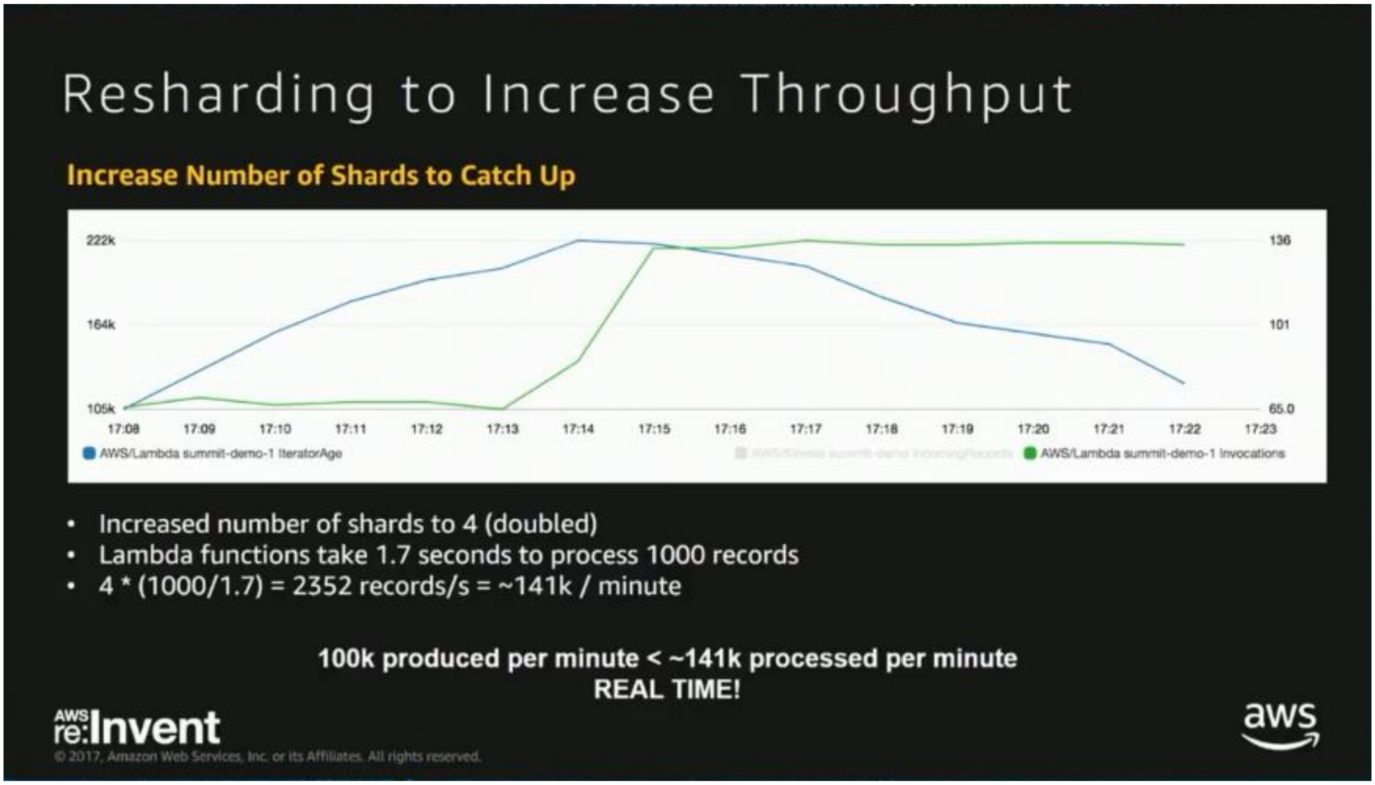


- Producers are inserting 100k records per minute
- 2 shards
- Lambda functions take 1.7 seconds to process 1000 records
- 2 * (1000/1.7) = 1176 records/s = ~70k/minute

100k produced per minute > ~70k processed per minute

AWS re:Invent

aws



Resharding to Increase Throughput

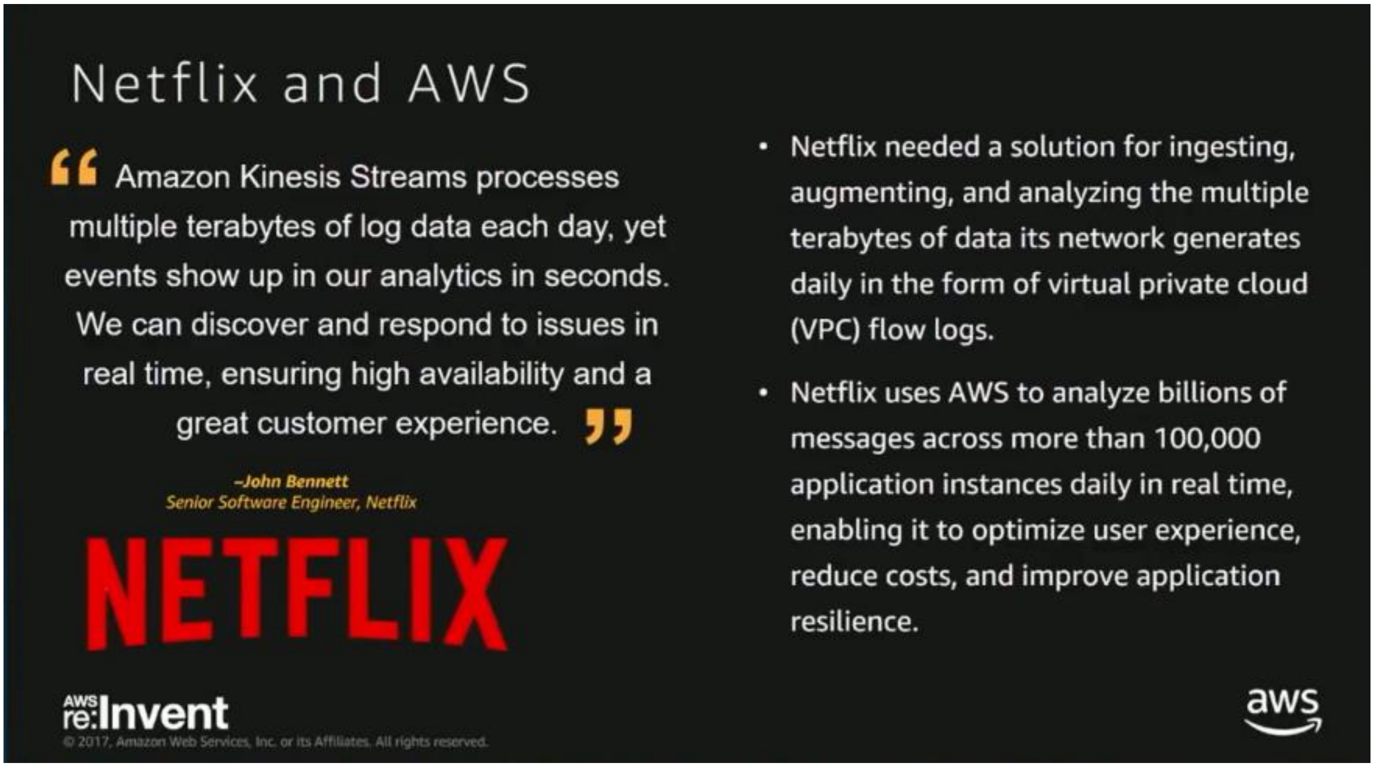
Increase Number of Shards to Catch Up

- Increased number of shards to 4 (doubled)
- Lambda functions take 1.7 seconds to process 1000 records
- 4 * (1000/1.7) = 2352 records/s = ~141k / minute

100k produced per minute < ~141k processed per minute
REAL TIME!

AWS re:Invent

aws



Netflix and AWS

> “Amazon Kinesis Streams processes multiple terabytes of log data each day, yet events show up in our analytics in seconds. We can discover and respond to issues in real time, ensuring high availability and a great customer experience.”

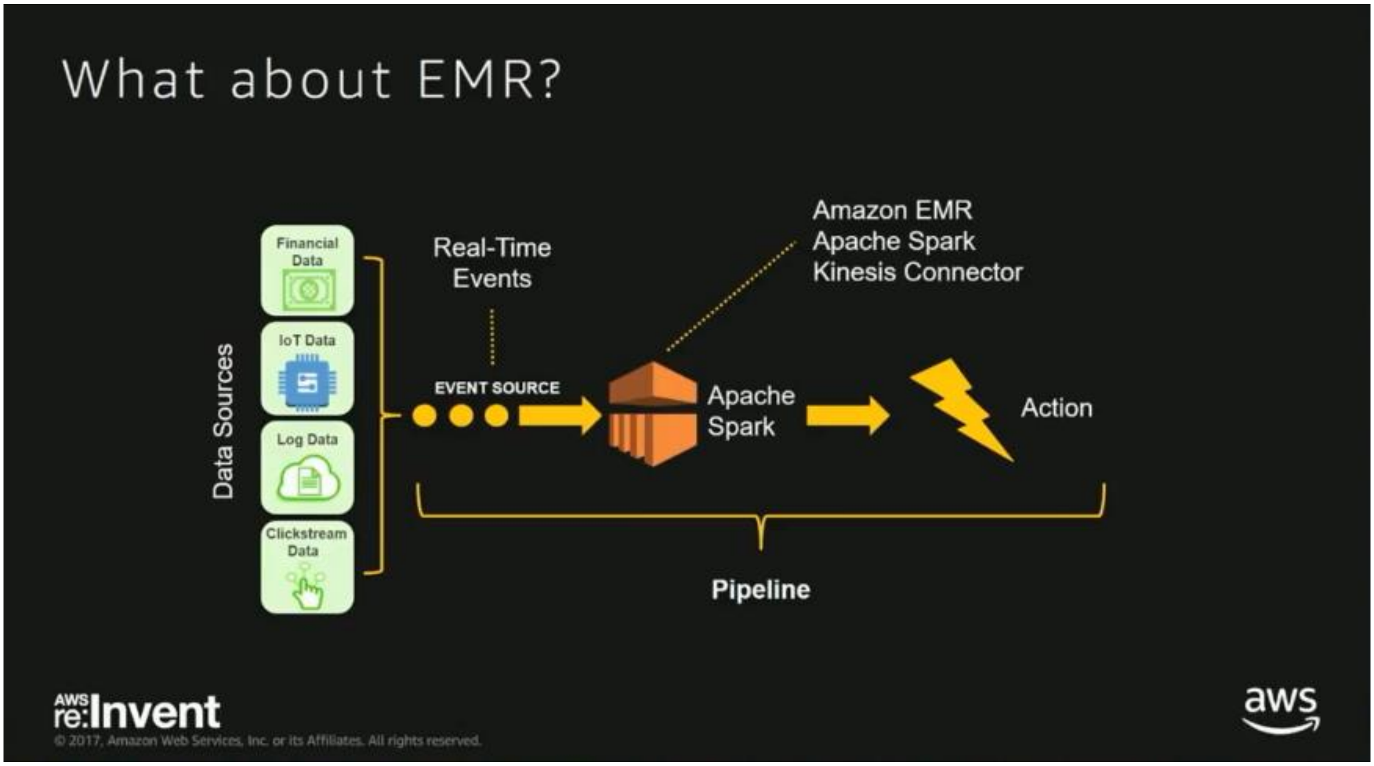
–John Bennett
Senior Software Engineer, Netflix

NETFLIX

- Netflix needed a solution for ingesting, augmenting, and analyzing the multiple terabytes of data its network generates daily in the form of virtual private cloud (VPC) flow logs.
- Netflix uses AWS to analyze billions of messages across more than 100,000 application instances daily in real time, enabling it to optimize user experience, reduce costs, and improve application resilience.

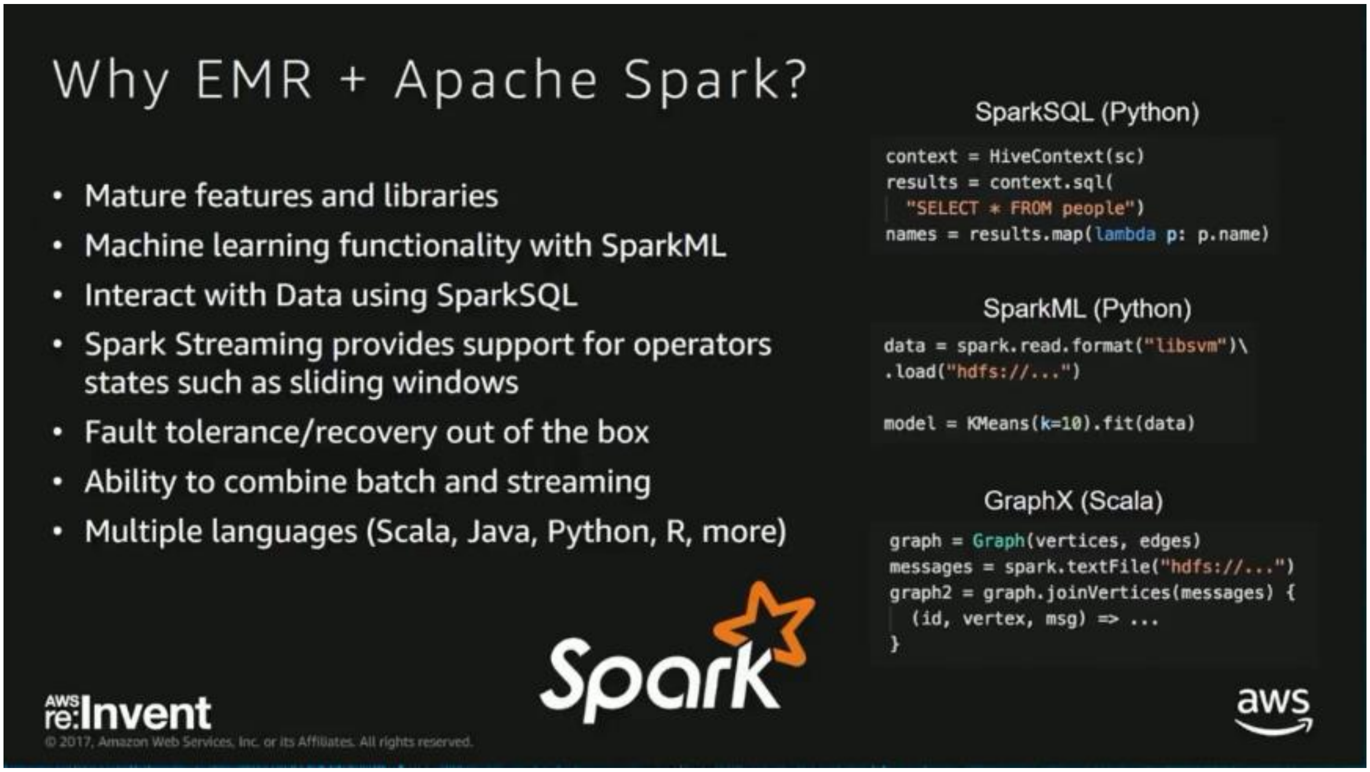
AWS re:Invent

aws



What about EMR?

re:Invent



Why EMR + Apache Spark?

- Mature features and libraries
- Machine learning functionality with SparkML
- Interact with Data using SparkSQL
- Spark Streaming provides support for operators states such as sliding windows
- Fault tolerance/recovery out of the box
- Ability to combine batch and streaming
- Multiple languages (Scala, Java, Python, R, more)

SparkSQL (Python)

```
context = HiveContext(sc)
results = context.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

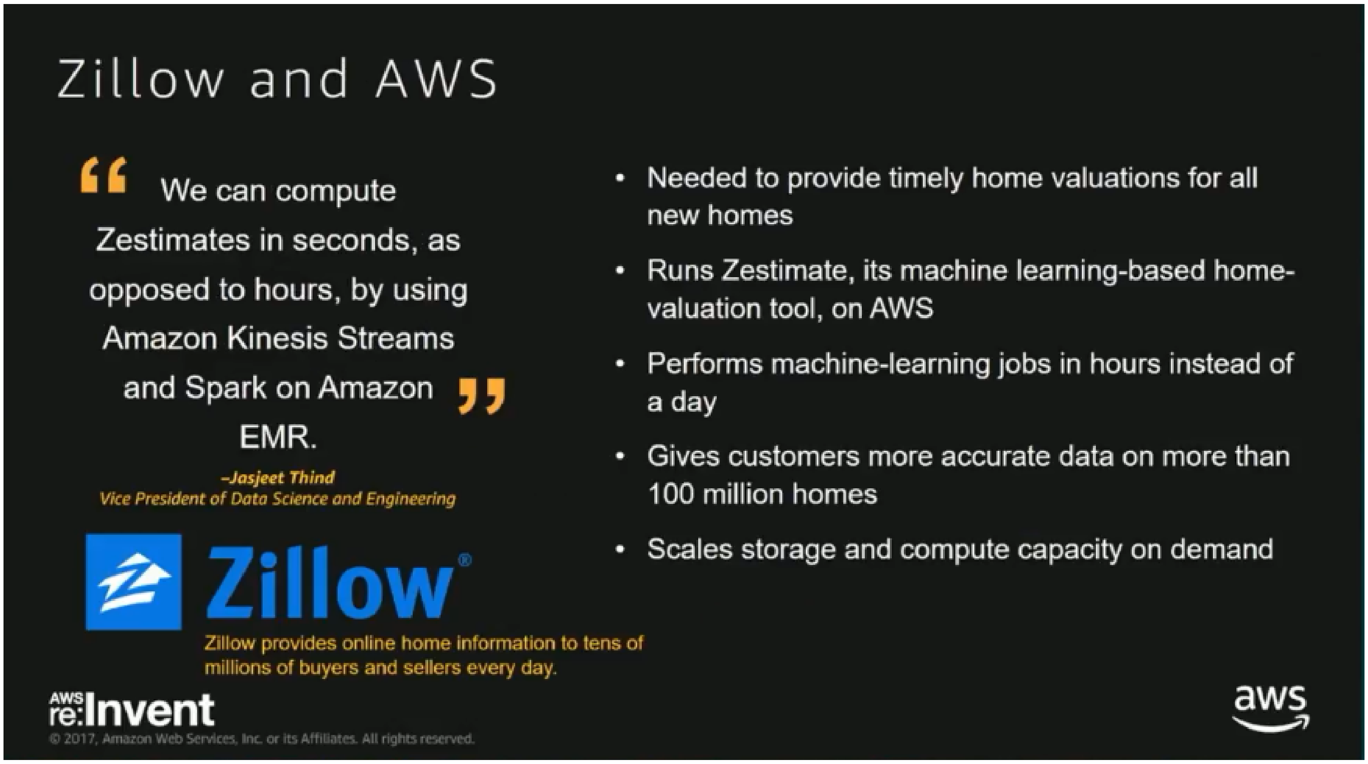
SparkML (Python)

```
data = spark.read.format("libsvm")\
.load("hdfs://...")

model = KMeans(k=10).fit(data)
```

GraphX (Scala)

```
graph = Graph(vertices, edges)
messages = spark.textFile("hdfs://...")
graph2 = graph.joinVertices(messages) {
  (id, vertex, msg) => ...
}
```

re:Invent



Zillow and AWS

> "We can compute Zestimates in seconds, as opposed to hours, by using Amazon Kinesis Streams and Spark on Amazon EMR."
>
> *–Jasjeet Thind*
> *Vice President of Data Science and Engineering*

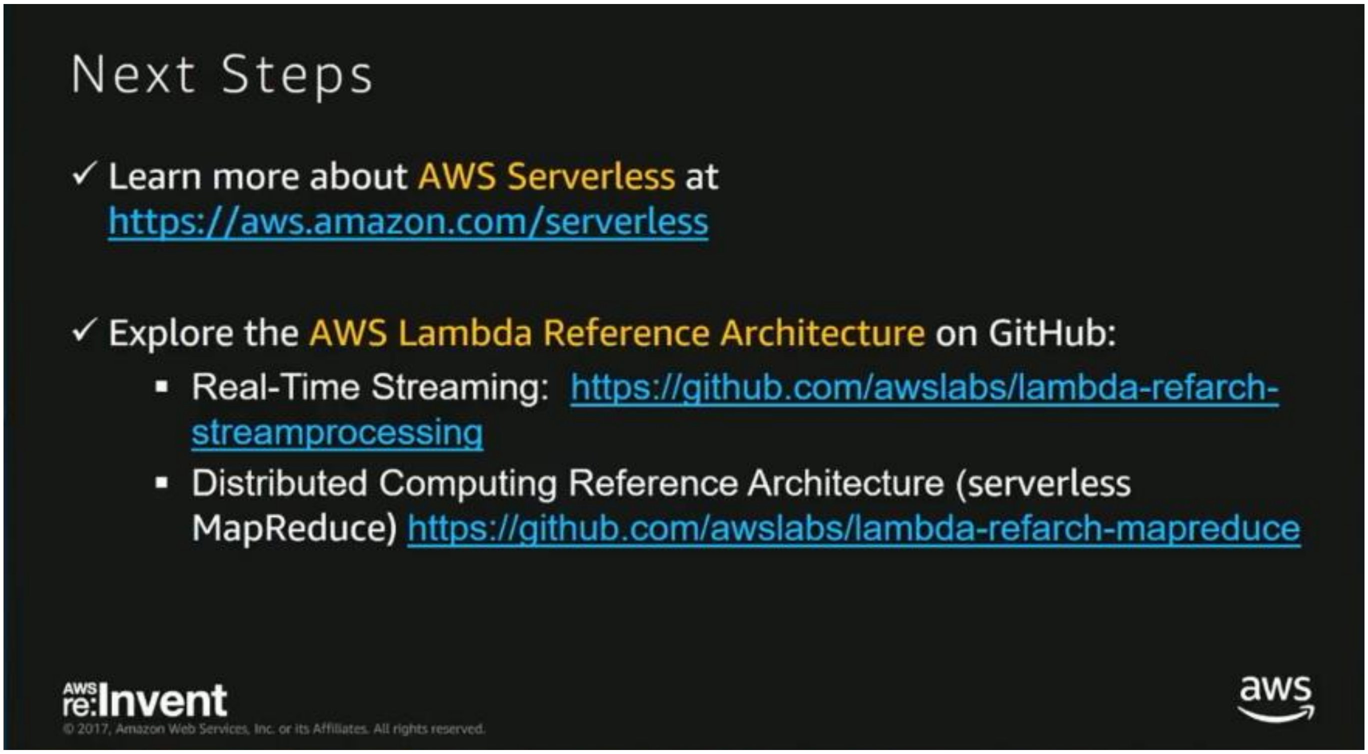
Zillow®

Zillow provides online home information to tens of millions of buyers and sellers every day.

- Needed to provide timely home valuations for all new homes
- Runs Zestimate, its machine learning-based home-valuation tool, on AWS
- Performs machine-learning jobs in hours instead of a day
- Gives customers more accurate data on more than 100 million homes
- Scales storage and compute capacity on demand

AWS re:Invent

aws



Next Steps

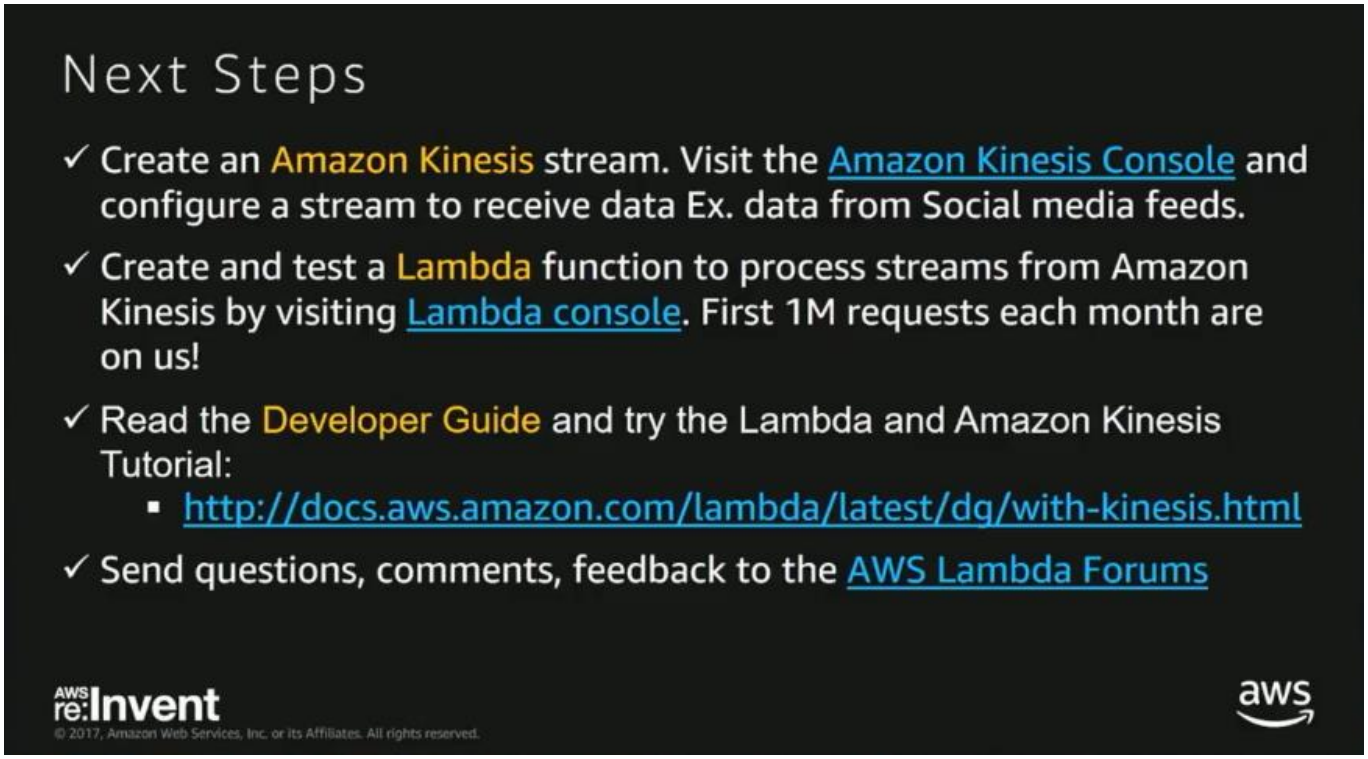
✓ Learn more about AWS Serverless at https://aws.amazon.com/serverless

✓ Explore the AWS Lambda Reference Architecture on GitHub:

- Real-Time Streaming: https://github.com/awslabs/lambda-refarch-streamprocessing
- Distributed Computing Reference Architecture (serverless MapReduce) https://github.com/awslabs/lambda-refarch-mapreduce

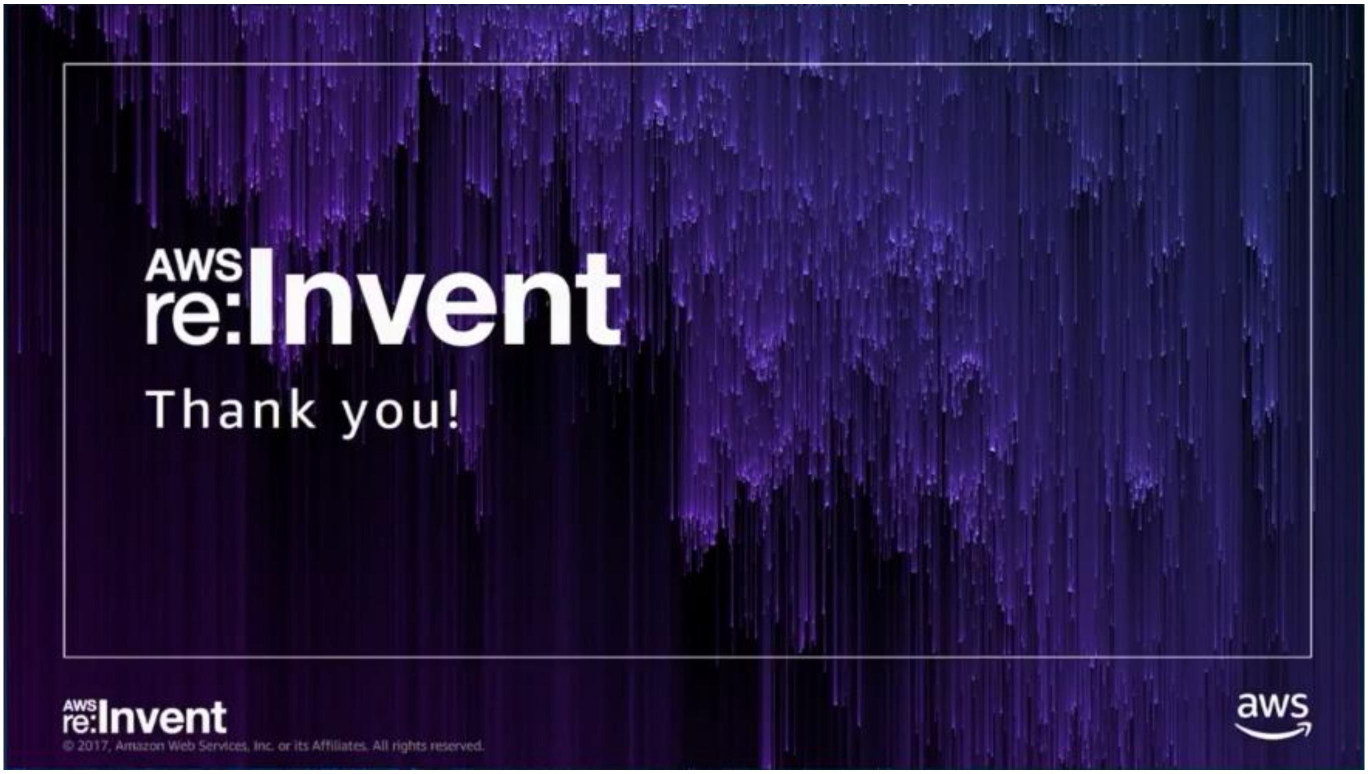
AWS re:Invent

aws



Next Steps

- ✓ Create an Amazon Kinesis stream. Visit the Amazon Kinesis Console and configure a stream to receive data Ex. data from Social media feeds.
- ✓ Create and test a Lambda function to process streams from Amazon Kinesis by visiting Lambda console. First 1M requests each month are on us!
- ✓ Read the Developer Guide and try the Lambda and Amazon Kinesis Tutorial:
  - http://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html
- ✓ Send questions, comments, feedback to the AWS Lambda Forums



Thank you!