Real-time Data Processing Using AWS Lambda

Cecilia Deng
Software Engineer
08 04 2016

amazon web services

Agenda

- AWS Services for Real Time Data
  - AWS Lambda
  - Amazon Kinesis
- Architecture & Workflow for Streaming Data Processing
- Streaming Data Processing Demo
- Best Practices in Building Data Processing Solutions

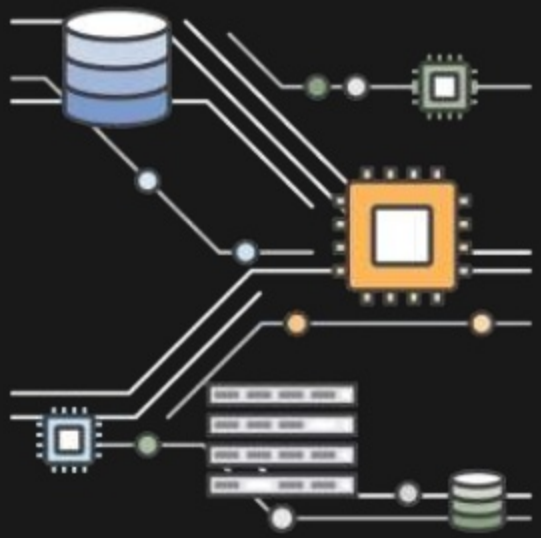
AWS Services for Data Processing

AWS Lambda

Amazon Kinesis

AWS Lambda: Overview

Lambda functions: a piece of code with stateless execution



Triggered by events:

- Direct Sync and Async API calls
- AWS Service integrations
- 3rd party triggers
- And many more …

Makes it easy to:

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

AWS Lambda: Serverless Compute in the Cloud

Compute service that runs your code in response to events

- Stateless, event-driven code with native support for Node.js, Java, and Python languages
- Compute & Code without managing infrastructure like EC2 instances and auto scaling groups
- Easy to author, deploy, maintain, secure and manage
- Allows for focus on business logic
- Makes it easy to Build back-end services that perform at scale

Benefits of AWS Lambda for building a server-less data processing engine

"Productivity focused compute platform to build powerful, dynamic, modular applications in the cloud"

1

No Infrastructure to manage

Focus on business logic, not infrastructure. You upload code; AWS Lambda handles everything else.

2

High performance at any scale; Cost-effective and efficient

Pay only for what you use: Lambda automatically matches capacity to your request rate. Purchase compute in 100ms increments.

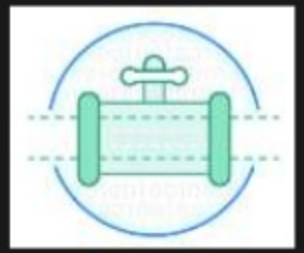
3

Bring Your Own Code

Run code in a choice of standard languages. Use threads, processes, files, and shell scripts normally.

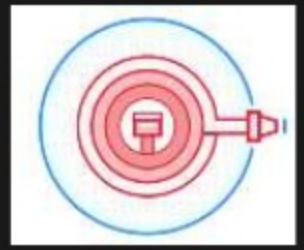
Amazon Kinesis: Overview

Managed services for streaming data ingestion and processing



Amazon Kinesis Streams

Build your own custom applications that process or analyze streaming data



Amazon Kinesis Firehose

Easily load massive volumes of streaming data into Amazon S3 and Redshift

Amazon Kinesis Analytics

Easily analyze data streams using standard SQL queries

Amazon Kinesis: Streaming data done the AWS way

Makes it easy to capture, deliver, and process real-time data streams

- Easy to provision, deploy, and manage
- Real-time latencies
- Pay as you go, no up-front costs
- Right services for your specific use cases

Benefits of Amazon Kinesis for stream data ingestion and continuous processing

Real-time Ingest

Highly Scalable

Durable

Replay-able Reads

Continuous Processing

GetShardIterator and GetRecords(ShardIterator)

Allows checkpointing/ replay

Enables multi concurrent processing

KCL, Firehose, Analytics, Lambda

Managed Service

Low end-to-end latency

Enable data movement into many Stores/ Processing Engines

Data Processing/Streaming Architecture & Workflow

AWS Lambda and Amazon Kinesis integration

How it Works

Stream-based model:

- Lambda polls the stream and batches available records
- Batches are passed for invocation to Lambda through function param
- Kinesis mapped as Event source in Lambda

Synchronous invocation:

- Lambda invoked as synchronous RequestResponse type
- Lambda function is executed **at least** once
- Each shard blocks on in order synchronous invocation

Event structure:

- Event received by Lambda function is a collection of records from Kinesis stream
- Customer defines max batch size, not effective batch size

Streaming Architecture Workflow: Lambda + Kinesis

Data Input	Kinesis	Action	Lambda	Data Output
IT application activity	Capture the stream	Audit	Process the stream	SNS
Metering records		Condense		Redshift
Change logs		Backup		S3
Financial data		Store		RDS
Transaction orders		Process		SQS
Server health metrics		Monitor		EC2
User clickstream		Analyze		EMR
IoT device data		Respond		Backend endpoint
Custom data		Custom action		Custom application

Common Architecture: Lambda + Kinesis

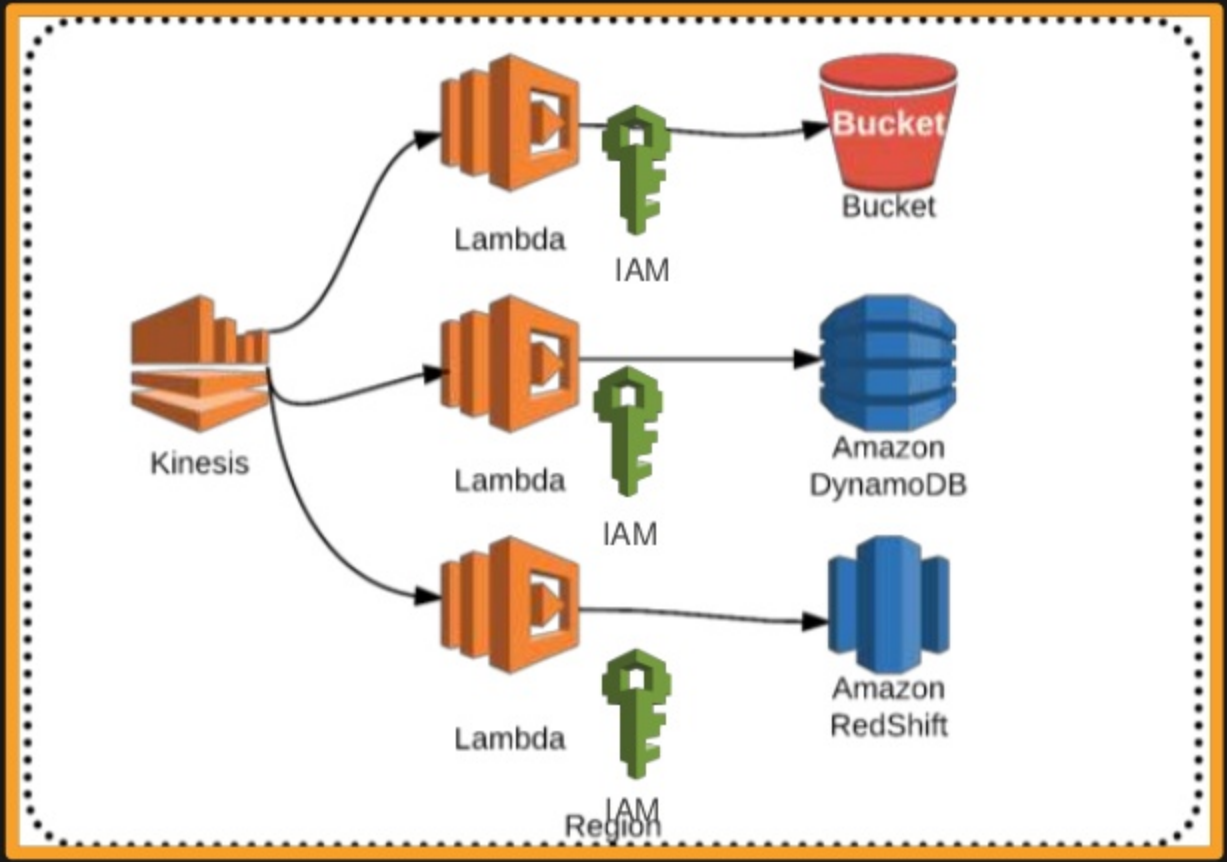
Real Time Data Processing

1. Real-time event data sent to **Amazon Kinesis**, allows multiple **AWS Lambda** functions to process the same events.
2. In **AWS Lambda**, Function 1 processes and aggregates data from incoming events, then stores result data in **Amazon DynamoDB**
3. Lambda Function 1 also sends values to **Amazon CloudWatch** for simple monitoring of metrics.
4. In **AWS Lambda** function, Function 2 does data manipulation of incoming events and stores results in **Amazon S3**

https://s3.amazonaws.com/awslambda-reference-architectures/stream-processing/lambda-refarch-streamprocessing.pdf

Common Architecture: Lambda + Kinesis

Data Processing for Data Storage/Analysis



Grant AWS Lambda permissions for the relevant stream actions via IAM (Execution Role) during function creation

Amazon Kinesis stream can continuously capture and store terabytes of data per hour from hundreds of thousands of sources

Use Lambda to process and "fan out" to other AWS services i.e. Storage, Database, and BI/analytics

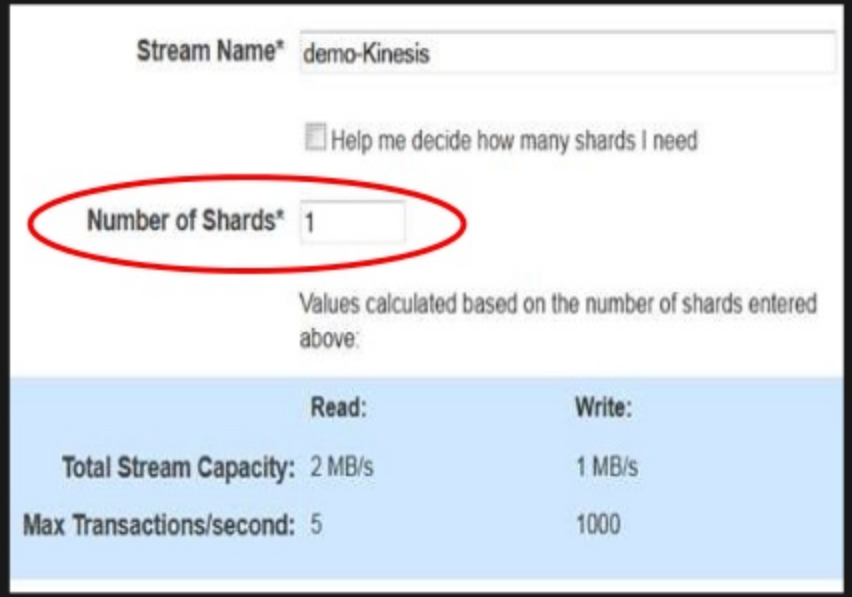
Data Processing:
Best Practices & Tips

Best Practices

Creating a Kinesis stream

Streams

- Made up of Shards
- Each Shard ingests data up to 1MB/sec
- Each Shard emits data up to 2MB/sec



Data

- All data is stored for 24 hours, Replay data inside of 24hr window
- A Partition Key is supplied by producer and used to distribute the PUTs across Shards
- A unique Sequence # is returned to the Producer upon a successful PUT call
- Make sure partition key distribution is even to optimize parallel throughput

Best Practices

Creating Lambda functions

Memory:

- CPU and disk 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 more wait in case of errors

Retries:

- For Kinesis, Lambda retries until the data expires (default 24 hours)

Permission model:

- The execution role defined for Lambda must have permission to access the stream

Best Practices

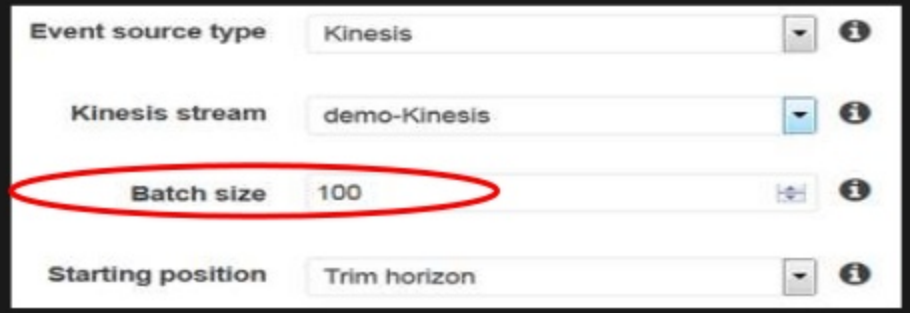
Configuring Lambda with Kinesis as an event source

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

  MIN(records available, batch size, 6MB)

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

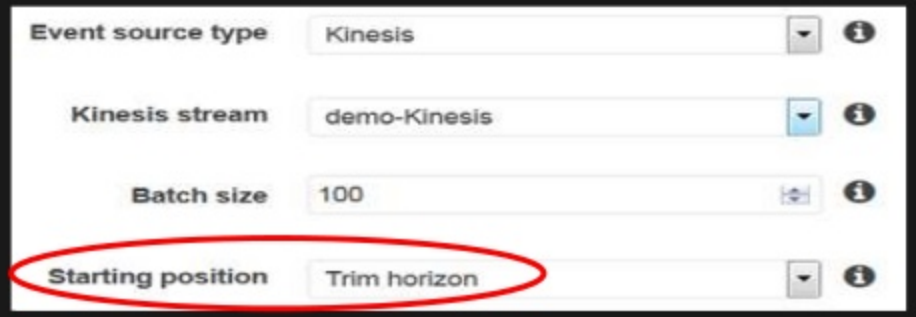


Best Practices

Configuring Lambda with Kinesis as an event source

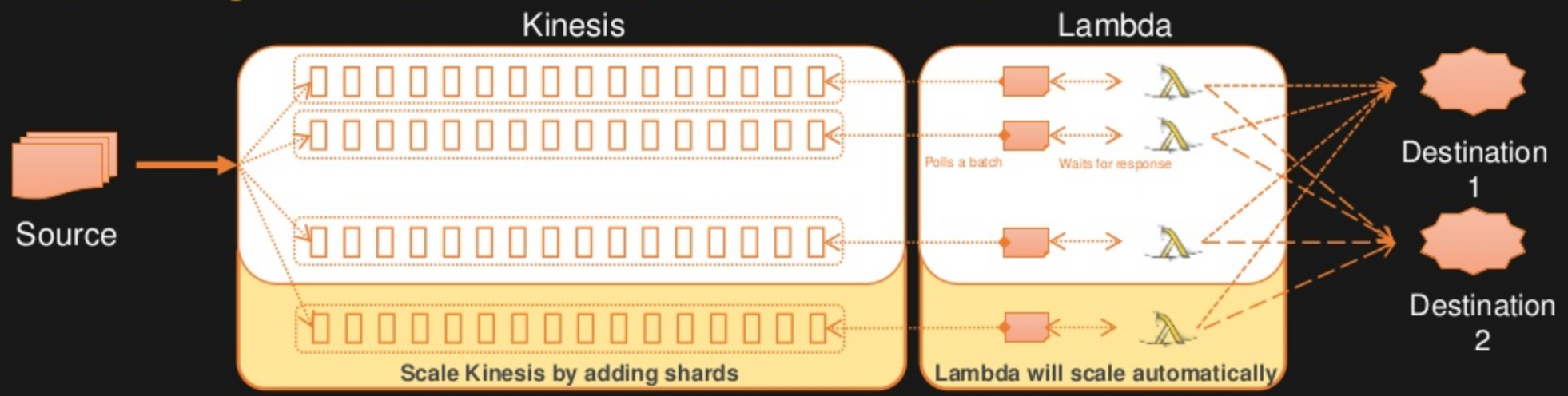
Starting Position:

- The position in the stream where Lambda starts reading
- Set to “Trim Horizon” for reading from start of stream (all data)
- Set to “Latest” for reading most recent data (LIFO) (latest data)



Best Practices

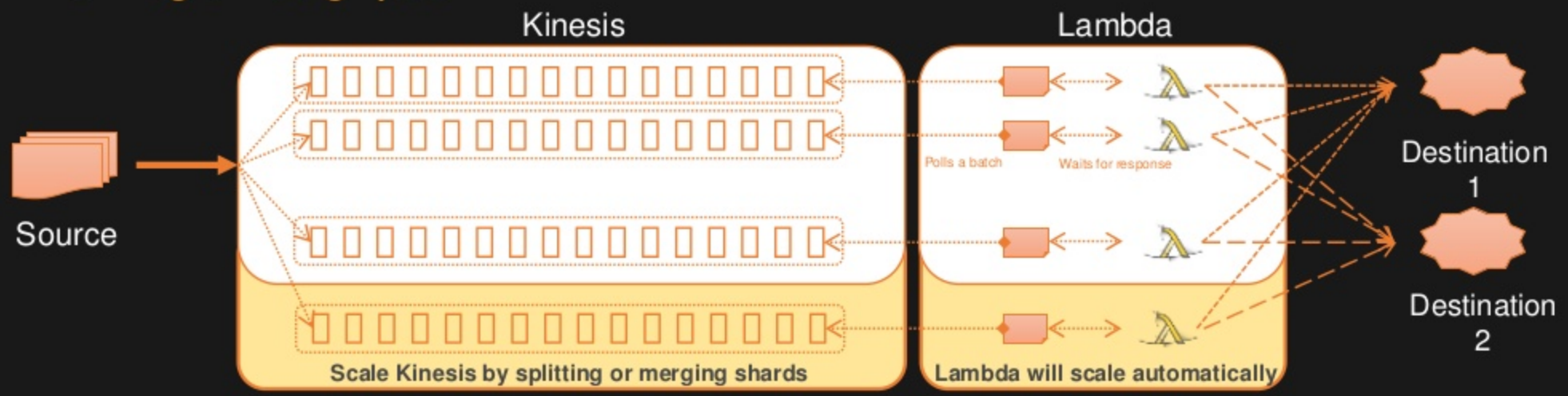
Attaching a Lambda function to a Kinesis Stream



- One Lambda function concurrently invoked per Kinesis shard
- Increasing # of shards with even distribution allows increased concurrency
- Lambda blocks on ordered processing for each individual shard
- This makes duration of the Lambda function directly impact throughput
- Batch size may impact duration if the Lambda function takes longer to process more records

Best Practices

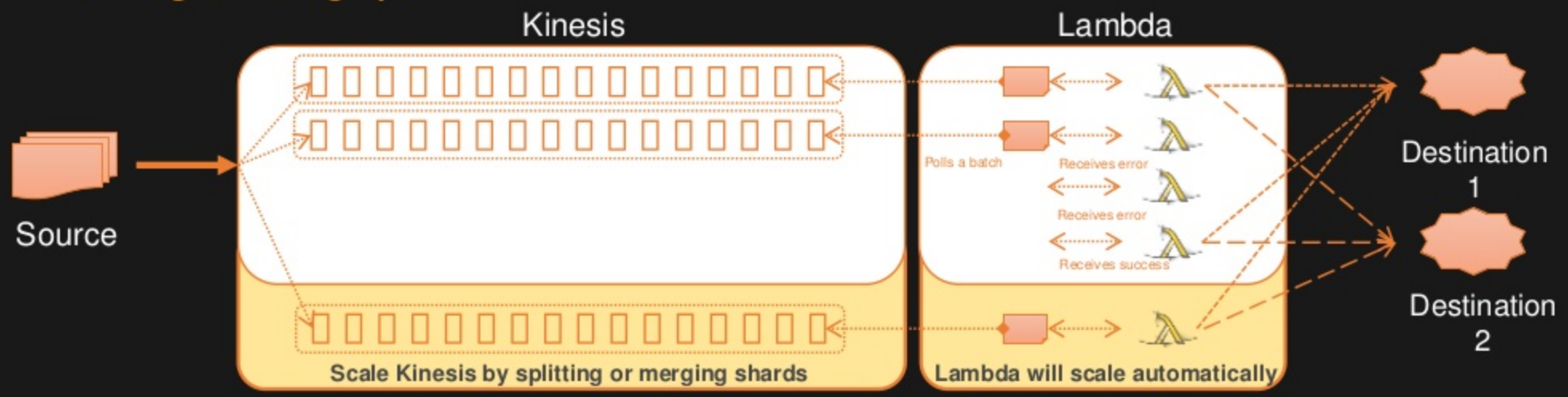
Tuning throughput



- Maximum theoretical throughput :
  # shards * 2MB / Lambda function duration (s)
- Effective theoretical throughput :
  # shards * batch size (MB) / Lambda function duration (s)
- If put / ingestion rate is greater than the theoretical throughput, your processing is at risk of falling behind

Best Practices

Tuning throughput

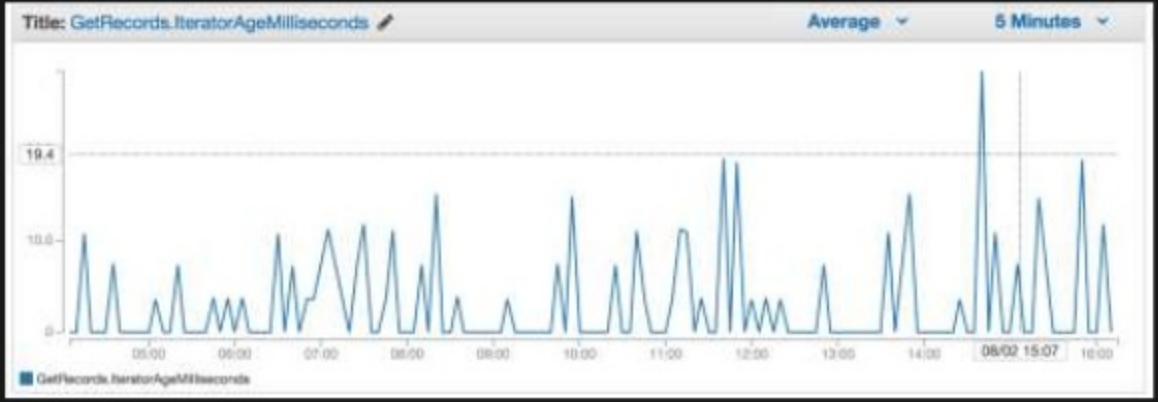


- Retry execution failures until the record is expired
- Retry with exponential backoff up to 60s
- Throttles and errors impacts duration and directly impacts throughput
- Effective theoretical throughput :

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

Best Practices

Monitoring Kinesis Streams with Amazon Cloudwatch Metrics

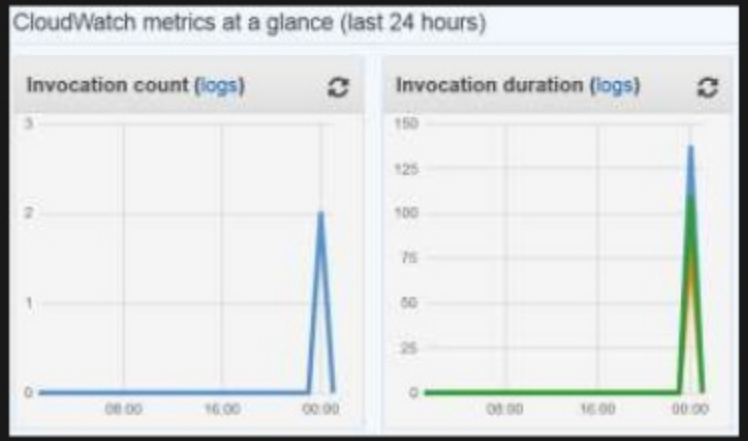


- GetRecords (effective throughput) : bytes, latency, records etc
- PutRecord : bytes, latency, records, etc
- GetRecords.IteratorAgeMilliseconds: how old your last processed records were. If high, processing is falling behind. If close to 24 hours, records are close to being dropped.

Best Practices

Monitoring Lambda functions

- **Monitoring**: available in Amazon CloudWatch Metrics
  - Invocation count
  - Duration
  - Error count
  - Throttle count



- **Debugging**: available in Amazon CloudWatch Logs
  - All Metrics
  - Custom logs
  - RAM consumed
  - Search for log events

Log Streams
2015/05/21/2184a6f4cb3e4c81a7265418e0c4078c
2015/05/21/3d07c7e60fc04acab257ee789bb667ef
2015/05/21/898f9b697db84b61a19879477d197679

Best Practices

Create different Lambda functions for each task, associate to same Kinesis stream

```
1  exports.handler = function(event, context) {
2      event.Records.forEach(function(record) {
3          // Kinesis data is base64 encoded so decode here
4          payload = new Buffer(record.kinesis.data, 'base64').toString('ascii');
5          console.log('Decoded payload:', payload);
6      });
7      context.succeed();
8  };
```

Log to CloudWatch Logs

```
1  var aws = require('aws-sdk');
2  var sns = new aws.SNS();
3
4  exports.handler = function(event, context) {
5      // Kinesis data is base64 encoded so decode here
6      payload = new Buffer(event.Records[0].kinesis.data, 'base64').toString('ascii');
7      var params = {
8          Message: payload,
9          Subject: 'Demo Lambda Message',
10         TopicArn: 'arn:aws:sns:us-west-2:[redacted]:demo-SNS'
11     };
12     sns.publish(params, function(err, data) {
13         context.succeed();
14     });
15 };
```

Push to SNS

Get Started: Data Processing with AWS

Next Steps

1. Create your first Kinesis stream. Configure hundreds of thousands of data producers to put data into an Amazon Kinesis stream. Ex. data from Social media feeds.

2. Create and test your first Lambda function. Use any third party library, even native ones. First 1M requests each month are on us!

3. Read the Developer Guide, AWS Lambda and Kinesis Tutorial, and resources on GitHub at AWS Labs
   - http://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html
   - https://github.com/awslabs/lambda-streams-to-firehose

Thank You!

Cecilia Deng
Software Engineer