WRK302

Event-Driven Programming

With Amazon DynamoDB Streams and AWS Lambda

Daniela Miao, Software Engineer, AWS, DynamoDB

October 2015

amazon web services

What to Expect from the Session

Why Amazon DynamoDB?

Amazon DynamoDB Basics

Amazon DynamoDB Streams

AWS Lambda

Building an event-driven application

Summary

Why Amazon DynamoDB?

Customers wanted a database with…

Reliably low latency at any scale

Elastic throughput and storage

Flexible queries and data models

Strong consistency

Availability and fault-tolerance

What Amazon DynamoDB Provides…

Fully-managed NoSQL database service

Consistent, **single-digit millisecond** latency

Independent **throughput and storage** scaling

Supports both **document and key-value** data models

Automatic data replication across **three facilities**

Amazon DynamoDB Basics

Data model

Tables: collection of items

Items: collection of attributes

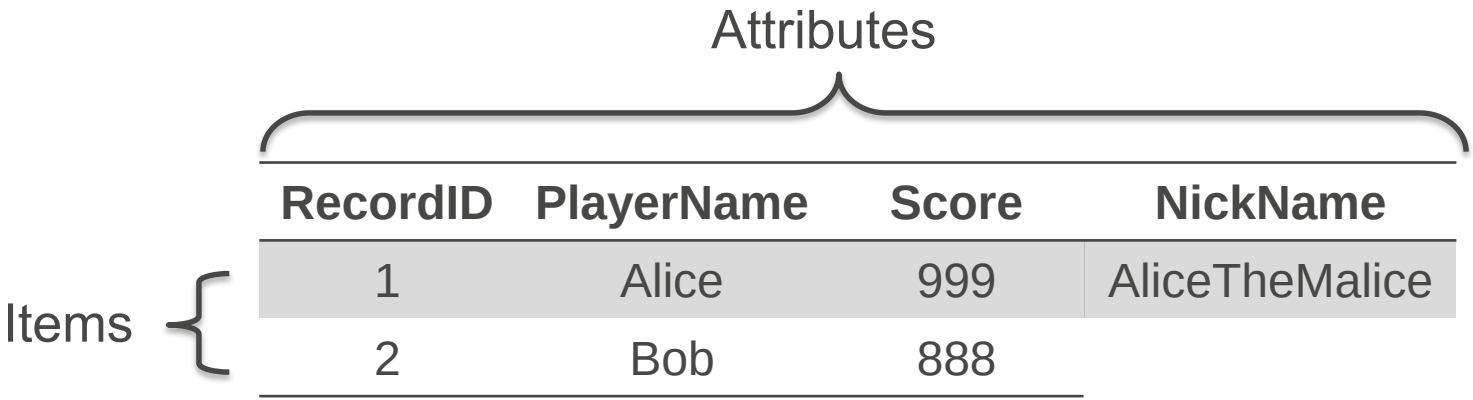
Attributes: name-value pairs

- Scalar: number, string, binary, boolean, null
- Multi-valued: string set, number set, binary set
- Document: list, map

Data model

Sample table: GameRecords

- Note attributes can be optional



Primary key – hash

Hash type primary key: primary key is made of one attribute, required for all items

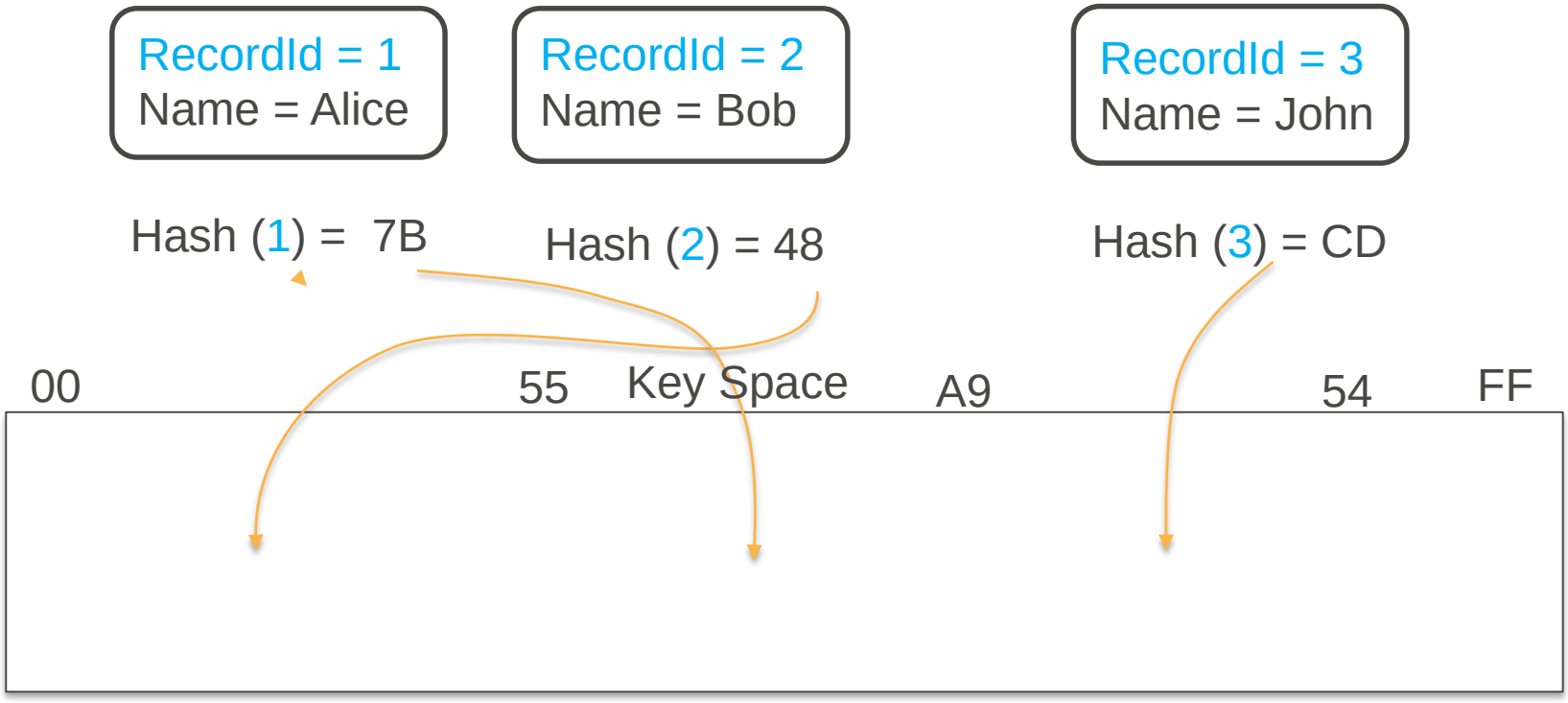
Primary Key

RecordID	PlayerName	Score	NickName
1	Alice	999	AliceTheMalice
2	Bob	888	

Distributing hash keys

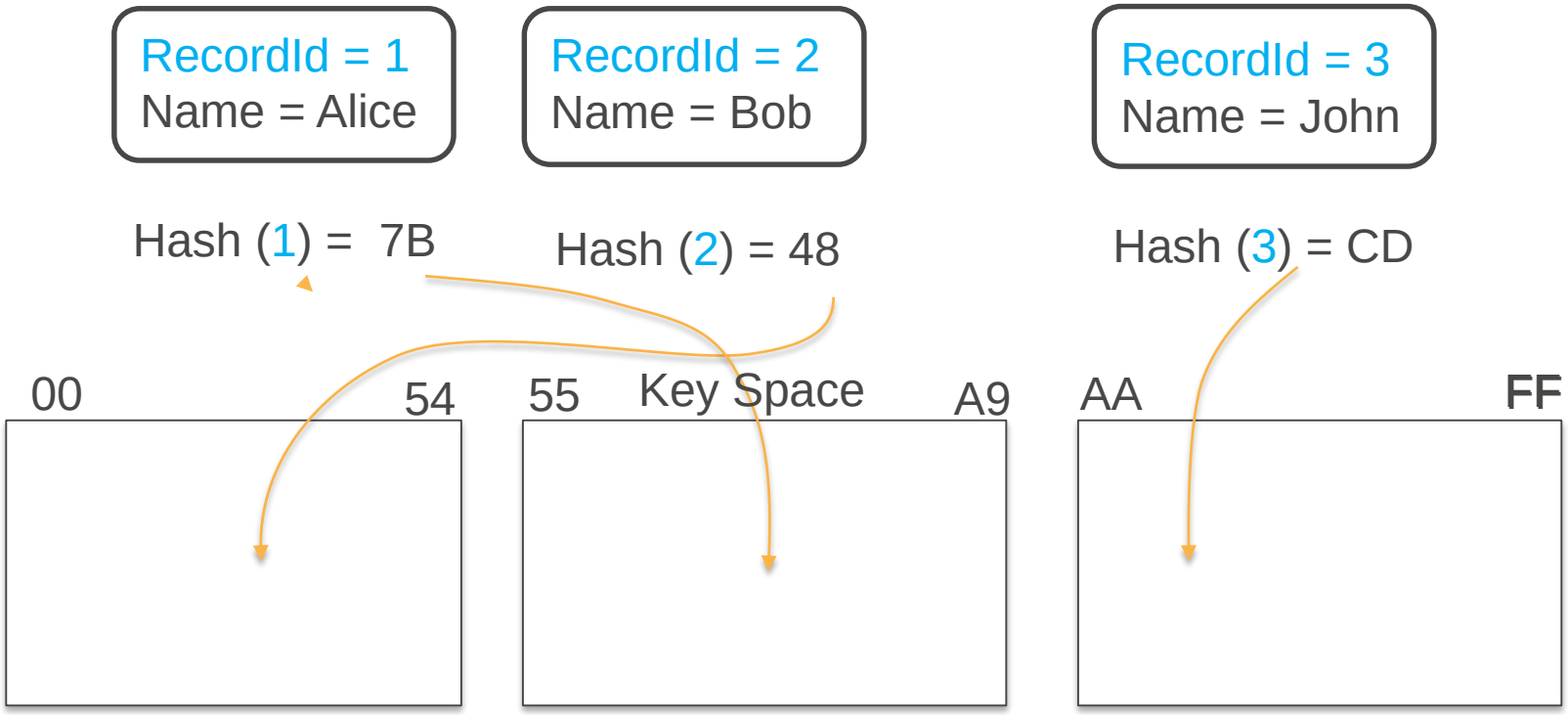
Hash key uniquely identifies an item

Hash key is used for building an unordered hash index



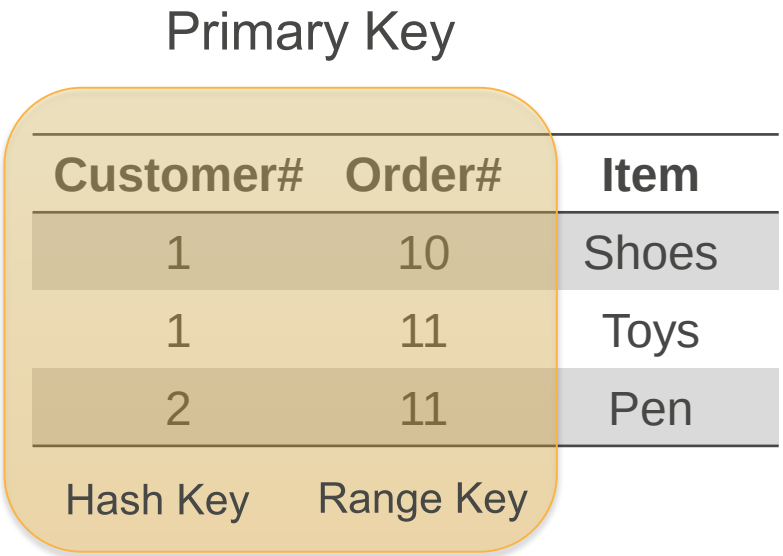
Distributing hash keys

Table can be partitioned for scale



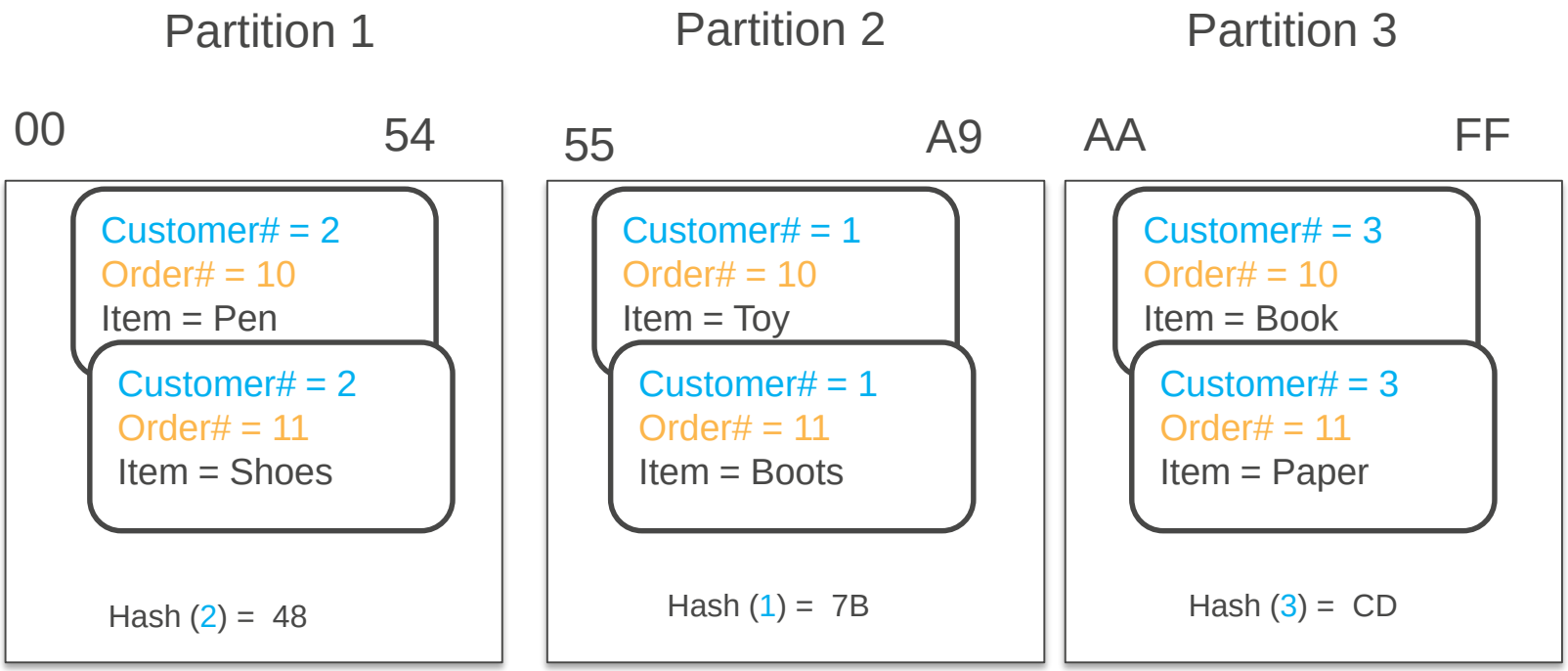
Primary key – hash and range

Hash and range type primary key: the primary key is made of two attributes, required for all items



Hash range table

Within an unordered hash index, data is sorted by the range key



Query and scan

A **query** operation finds items in a table using only primary key attribute values.

- Hash key attribute-value pair is required
- Range key attribute-value pair is optional
- In the previous example, query for all orders of a specific customer by setting “customer# = 1".

A **scan** operation examines every item in the table.

- Returns all of the data attributes for every item

Amazon DynamoDB Streams

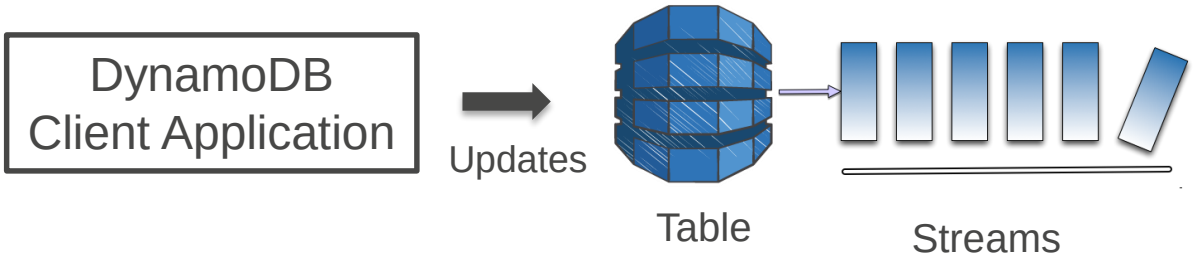
Amazon DynamoDB Streams

New

Stream of updates to a table in a continous pipeline

Records all modifications made to items of the table

Scales with table size and throughput



Customer application

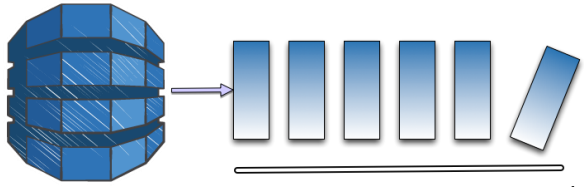
Amazon DynamoDB replicator using DynamoDB Streams

Copies incoming data to a different table in real-time

Blog post published describing the application

> “DynamoDB delivers even more stability, redundancy, and speed for our document-based data and requires less hands-on administration, so we can focus on building the software that powers Mapbox.”

View types

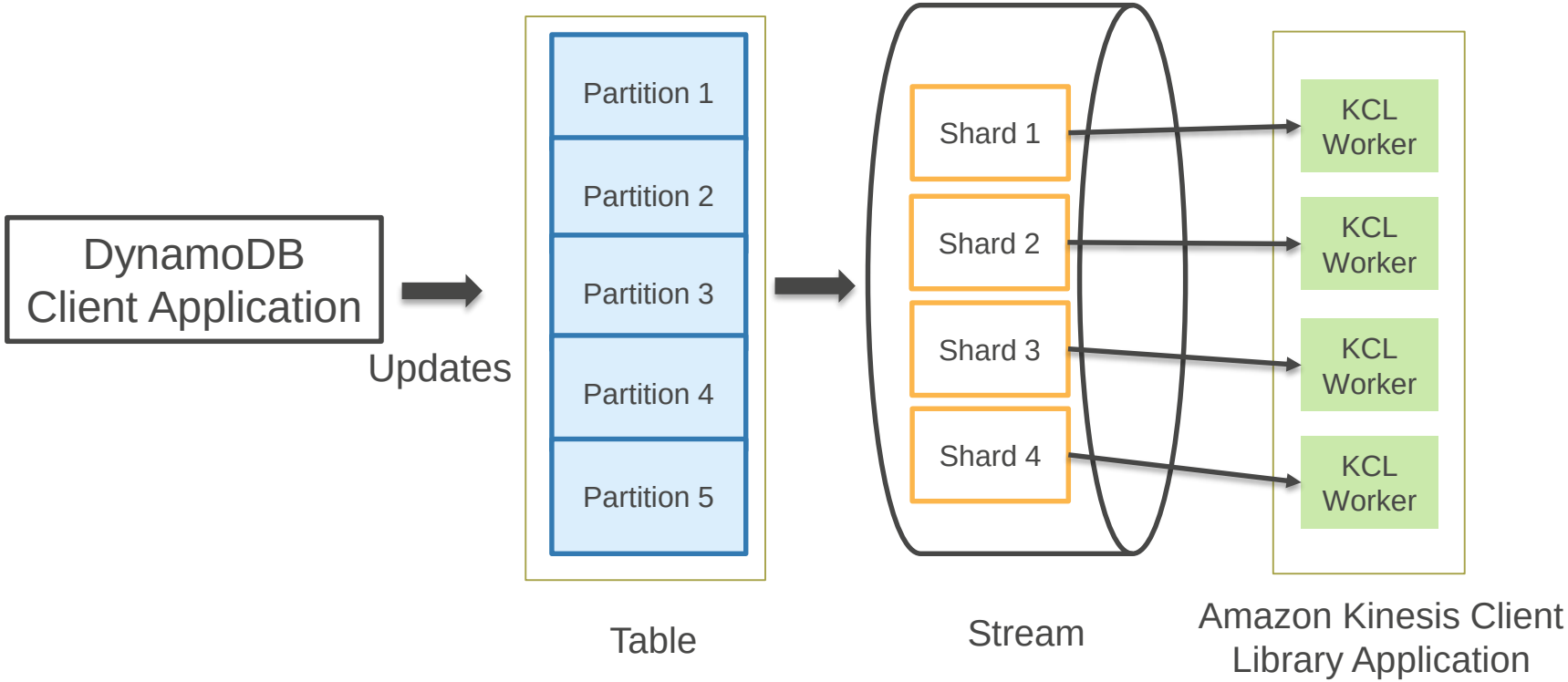


UpdateItem: (RecordID = 1, Name = Alice) → (RecordID = 1, Name = Alison)

Entire image of the item is returned, not just the changes

View Type	Record Content
Old image—before update	RecordID = 1, Name = Alice
New image—after update	RecordID = 1, Name = Alison
Old and new images	RecordID = 1, Name = Alice RecordID = 1, Name = Alison
Keys only	RecordID = 1

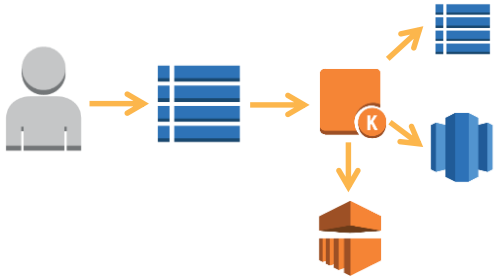
Amazon DynamoDB Streams and Amazon Kinesis Client Library



Analytics with Amazon DynamoDB Streams

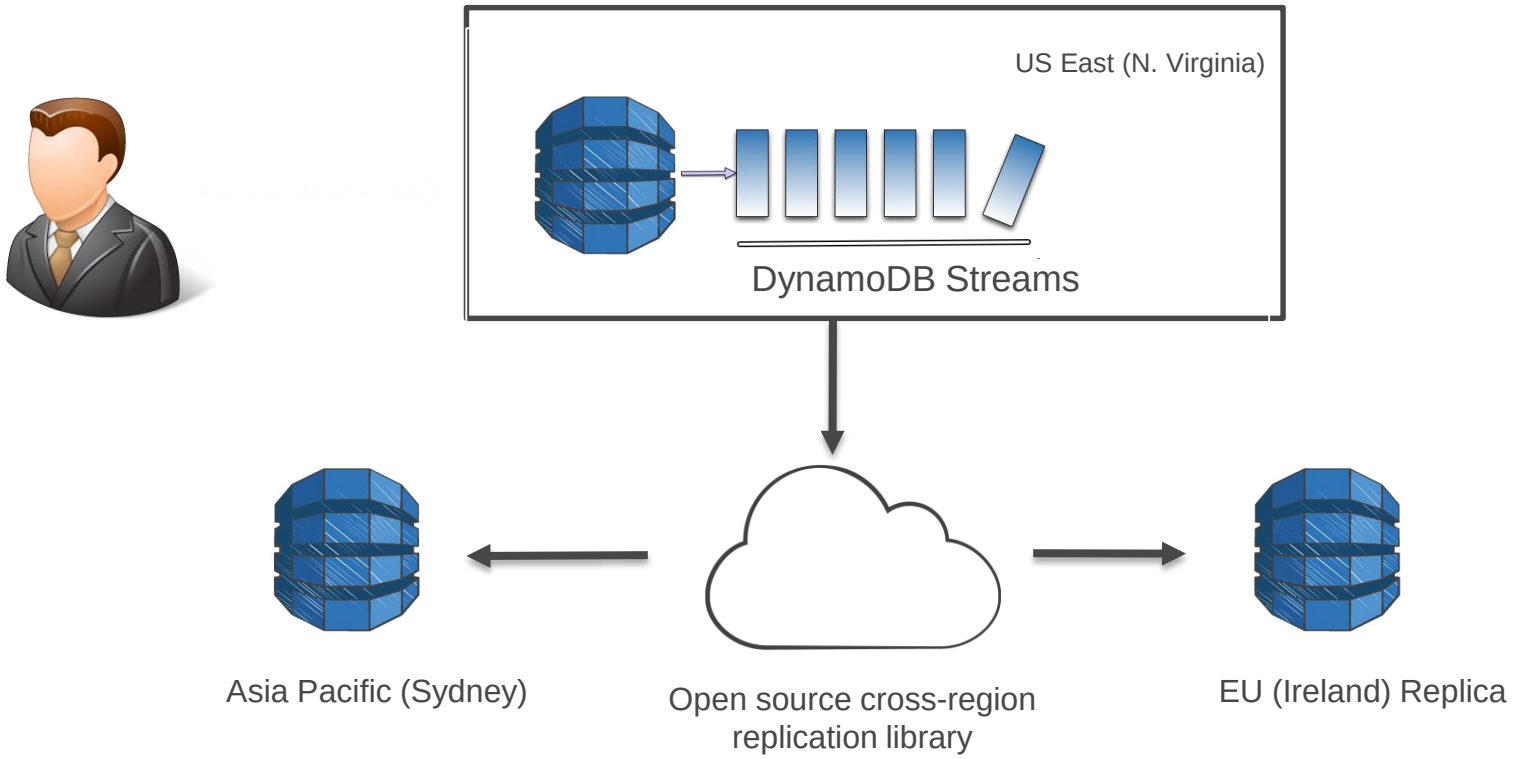
Collect and de-dupe data in DynamoDB

Aggregate data in-memory and flush periodically



Important when: Performing real-time aggregation and analytics

Cross-region replication



Cross-region replication library

Horizontal scaling

Bootstrapping existing items

Fault tolerance

Optimized for cost and performance

AWS Lambda

AWS Lambda

Event-driven compute in the cloud

Lambda functions: stateless, request-driven code execution

- Triggered by events in other services: e.g., write to an Amazon DynamoDB table

Ability to transform data as it reaches the cloud

AWS Lambda

Customers can focus on business logic, no need to manage infrastructure

Lambda scales to match the event rate

Pay for only what you use

- Fine-grained pricing: purchase compute time in 100ms increments
- No hourly, daily, or monthly minimums

Popular usage scenarios

Data triggers with Amazon S3 or Amazon DynamoDB

Audit and notify

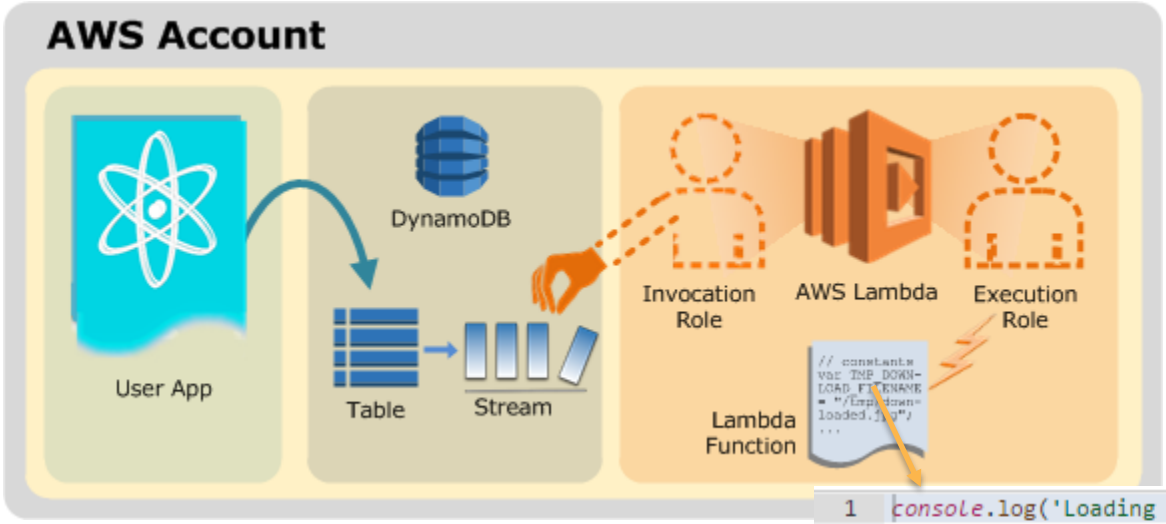
- Monitor AWS CloudTrail logs and use Lambda functions to send Amazon SNS notifications

Custom deployment rules using AWS CloudFormation

Validate cross-device sync data with Amazon Cognito

Custom events by publishing to Amazon Kinesis

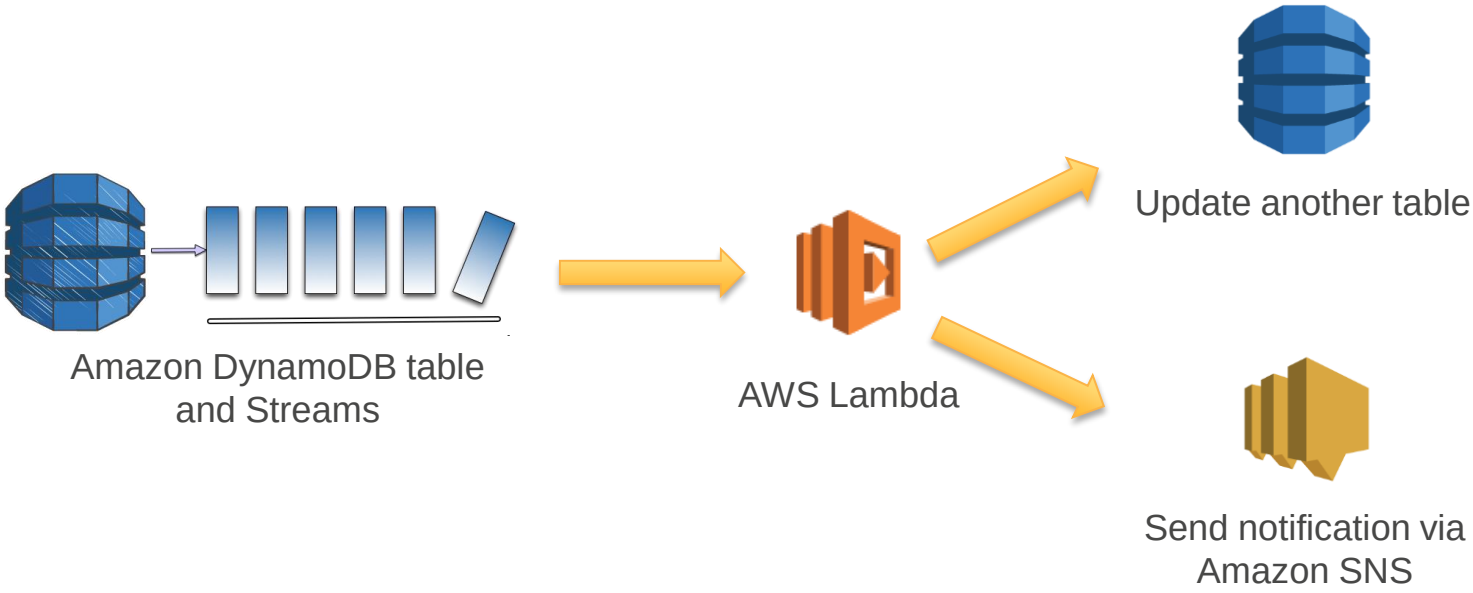
Amazon DynamoDB Streams and Lambda




```
1  console.log('Loading event');
2  exports.handler = function(event, context) {
3    console.log("Event: %j", event);
4    for(i = 0; i < event.Records.length; ++i) {
5      record = event.Records[i];
6      console.log(record.EventID);
7      console.log(record.EventName);
8      console.log("DynamoDB Record: %j", record.Dynamodb);
9    }
10   context.done(null, "Hello World");  // SUCCESS with message
11 }
```

```
2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff INSERT
2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff DynamoDB Record:{ "NewImage": { "name": { "S": "sivar" }, "hk": { "S": "3" } }, "SizeBytes": 15, "StreamViewType": "NEW_AND_OLD_IMAG
2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff Message: "Hello World"
```

Sample Scenario



Building
an event-driven application

Popular usage scenarios

Mobile gaming application

GameScoreRecords table

- Records of game scores
- Want to implement a list of aggregate scores for each user

GameScoresByUser table updated by AWS Lambda

Self-paced Hands-on Lab

Zip file included with the workshop package

Use lab guide to complete each exercise in your group

Solutions provided in a separate directory – do NOT use unless truly stuck

Remember to complete your evaluations!

http://reinventworkshop.s3.amazonaws.com/workshop.zip

Use QR code or link above to download the group activity package and follow the lab guide.

Thank you!

After the break, there will be a group-based activity
to build an event-driven application