

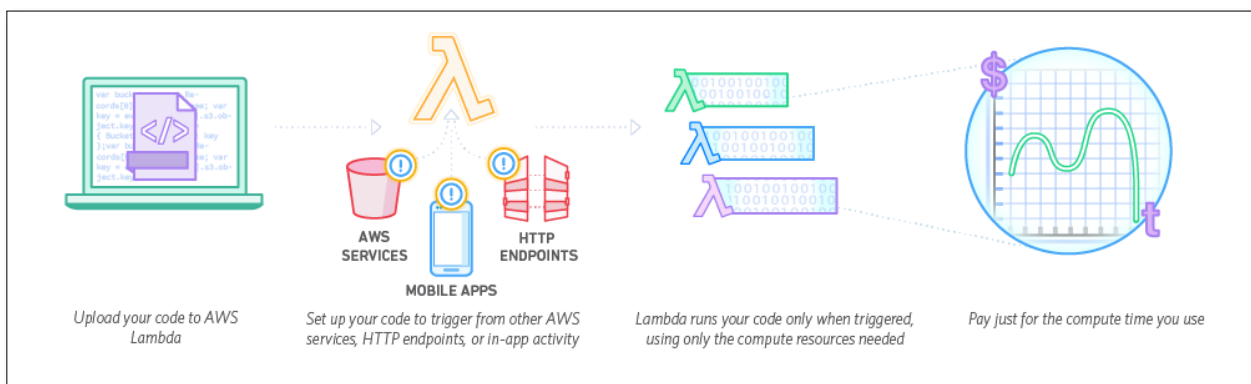
Final Project – AWS Training

Submitted By: Tejas Tripathi

Serverless Architectures with Amazon DynamoDB and Amazon Kinesis Streams with AWS Lambda

In this project we create a lambda function from blueprint, create an AWS Kinesis stream and trigger the function with data from our stream and monitor it.

Further we experience building a real-world application using Amazon DynamoDB and Lambda together for event-driven programming.



Step 1: Creating a Kinesis Stream

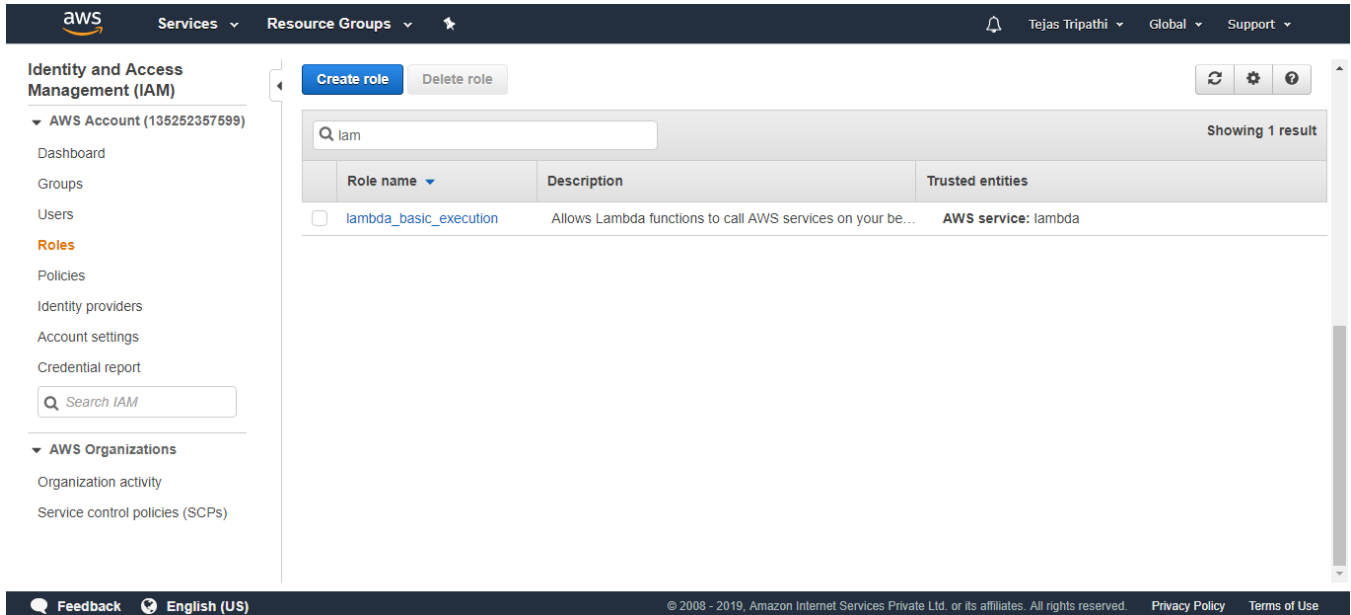
Here we create a 1 shard kinesis stream named Labstream

The screenshot shows the AWS Management Console for Kinesis streams. The page title is "Kinesis streams". Below the title, there is a description: "Kinesis data streams continuously capture and temporarily store real-time data. [Configure producers](#) to put data records into a data stream. [Configure consumers](#) to continuously process data stream records." Below this, there is a status bar showing "Total shards in use: 0" and "Total shards remaining: 500". A notification banner at the top says "Creating stream Labstream" and "Creating a stream generally takes up to a minute." Below the notification, there are buttons for "Create Kinesis stream", "Connect Kinesis consumers", and "Actions". A search bar is present with the text "Filter Kinesis streams". Below the search bar, there is a table with columns: "Kinesis stream name", "Number of shards", "Status", and "Consumers using enhanced fan-out". The table shows one entry: "Labstream" with a status of "Creating" and 0 consumers.

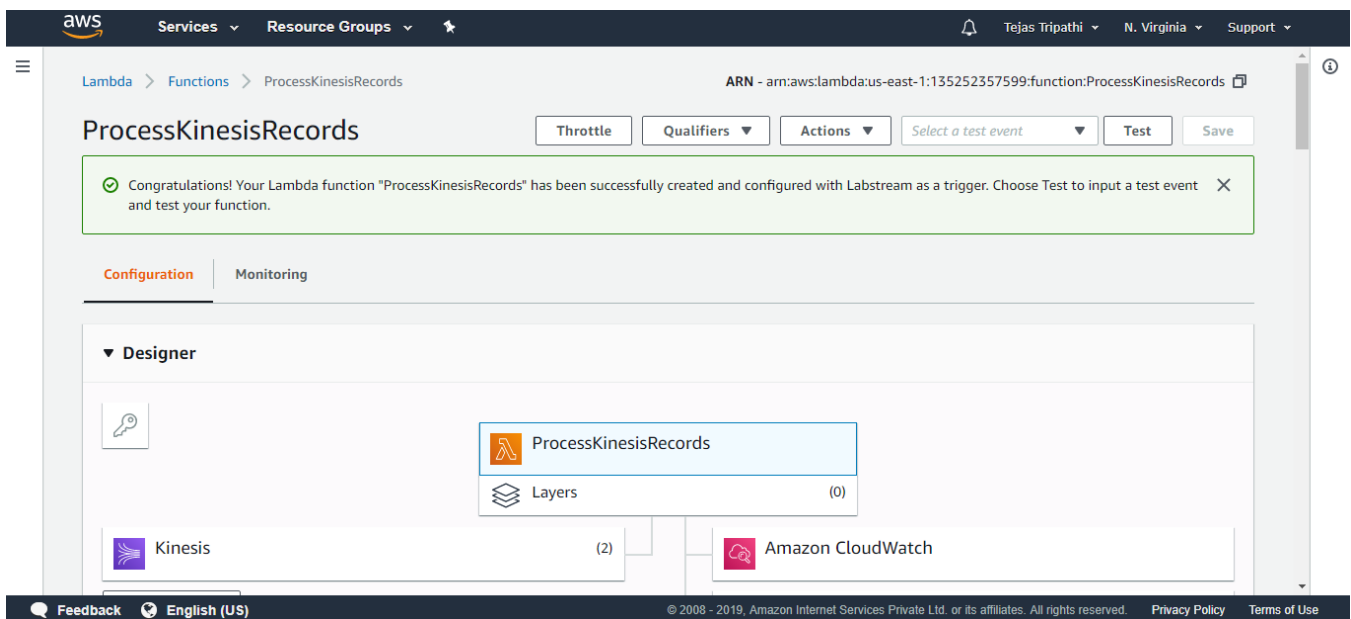
Kinesis stream name	Number of shards	Status	Consumers using enhanced fan-out
Labstream	1	Creating	0

Step 2: Creating a Lambda Function

First we have create a lambda_execution role in IAM, giving full access to Kinesis and Cloudwatch.



Next we create the Lambda function:



Step 3: Test the Function

The screenshot shows the AWS Lambda console for the function **ProcessKinesisRecords**. The function is configured with a trigger of type **stream**. A green notification box states: "Congratulations! Your Lambda function 'ProcessKinesisRecords' has been successfully created and configured with Labstream as a trigger. Choose Test to input a test event and test your function." Below this, the **Execution result: succeeded (logs)** section is expanded, showing the following details:

Details

The section below shows the result returned by your function execution.

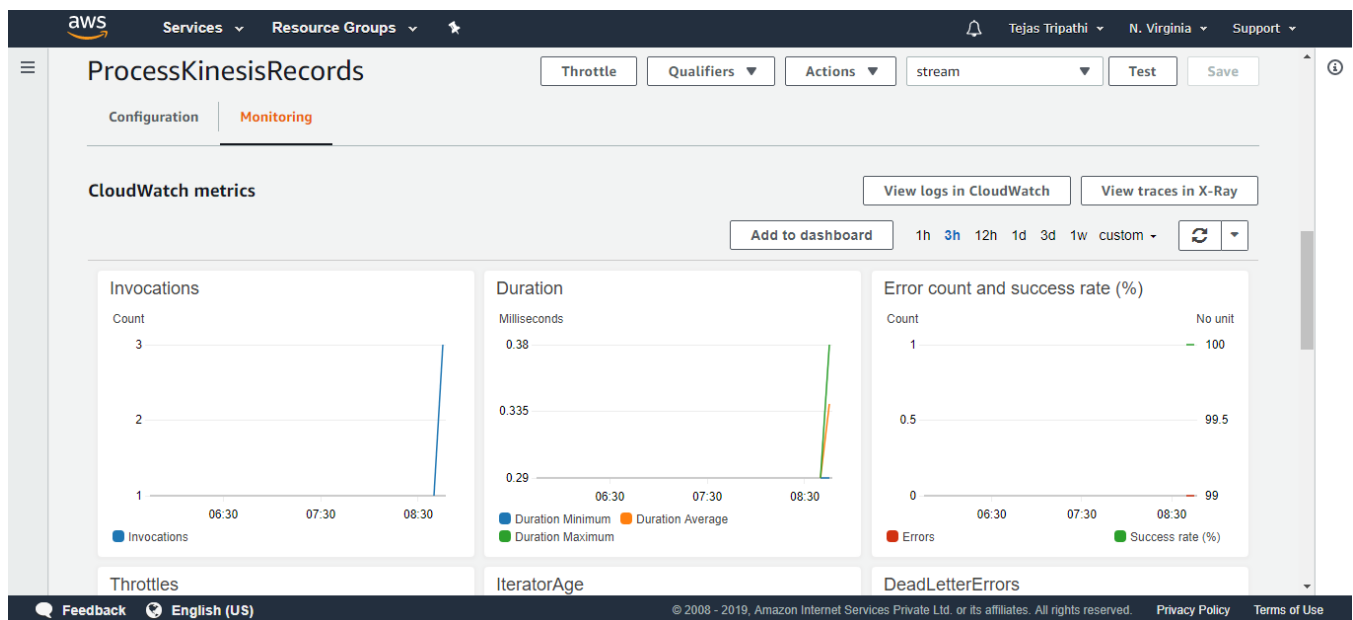
"Successfully processed 1 records."

Summary

Code SHA-256	Request ID
//ny7mgRryaQZTHVlvBPP7HYDrYsBft7dwGyWT+L/I=	5b2f3269-f4b0-4e71-b777-2292123119b2
Duration	Billed duration
0.35 ms	100 ms
Resources configured	Max memory used

The footer of the console shows the date "© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved." and links to "Privacy Policy" and "Terms of Use".

The metrics can also be monitored:



Step 4: Create tables in DynamoDB

aws Services Resource Groups

Tejas Tripathi N. Virginia Support

Create table Delete table

Filter by table name Choose a table group Actions

Viewing 2 of 2 Tables

Name	Status	Partition key	Sort key	Indexes	Total read capacity	Total write capacity
GameScoreRecords	Active	RecordID (Number)	-	0	5	5
GameScoresByUser	Active	Username (String)	-	0	5	5

Feedback English (US)

© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Step 5: Create a lambda function

aws Services Resource Groups

Tejas Tripathi N. Virginia Support

Lambda Functions AggregateScoresByUser

ARN - arn:aws:lambda:us-east-1:135252357599:function:AggregateScoresByUser

AggregateScoresByUser

Throttle Qualifiers Actions Select a test event Test Save

Configuration Monitoring

▼ Designer

AggregateScoresByUser

Layers (0)

DynamoDB

+ Add trigger

AWS Lambda

AWS Resource Groups

Amazon CloudWatch

Feedback English (US)

© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Test the function:

The screenshot shows the AWS Lambda console for the function `AggregateScoresByUser`. The execution result is displayed as a green box with a checkmark, indicating success. The details section shows the function returned the string `"Successfully processed 1 records."`. The summary section provides the following information:

Code SHA-256	Request ID
<code>rj3hDyufF0J9K7vKx+xFTn91LeIV6xAX0K5ObE49dyk=</code>	<code>4212e01f-e1e8-4424-8206-f158bbb7c78e</code>

Duration	Billed duration
1183.69 ms	1200 ms

Resources configured	Max memory used
128 MB	76 MB

The log output section shows the logging calls in the code, corresponding to a single row within the CloudWatch log group.

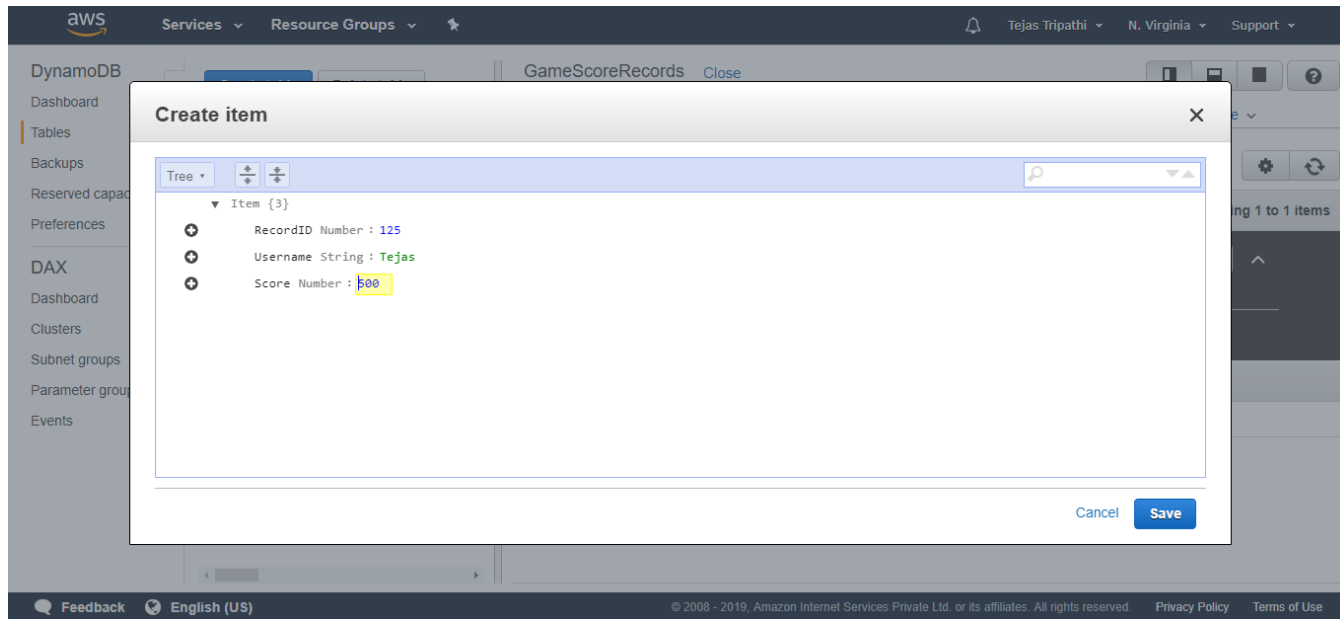
Verify the function (From DynamoDB):

The screenshot shows the AWS DynamoDB console for the table `GameScoresByUser`. The table is selected in the left-hand navigation pane. The main pane displays the table's structure and a list of items. The table has two columns: `Username` and `Score`. The following item is shown:

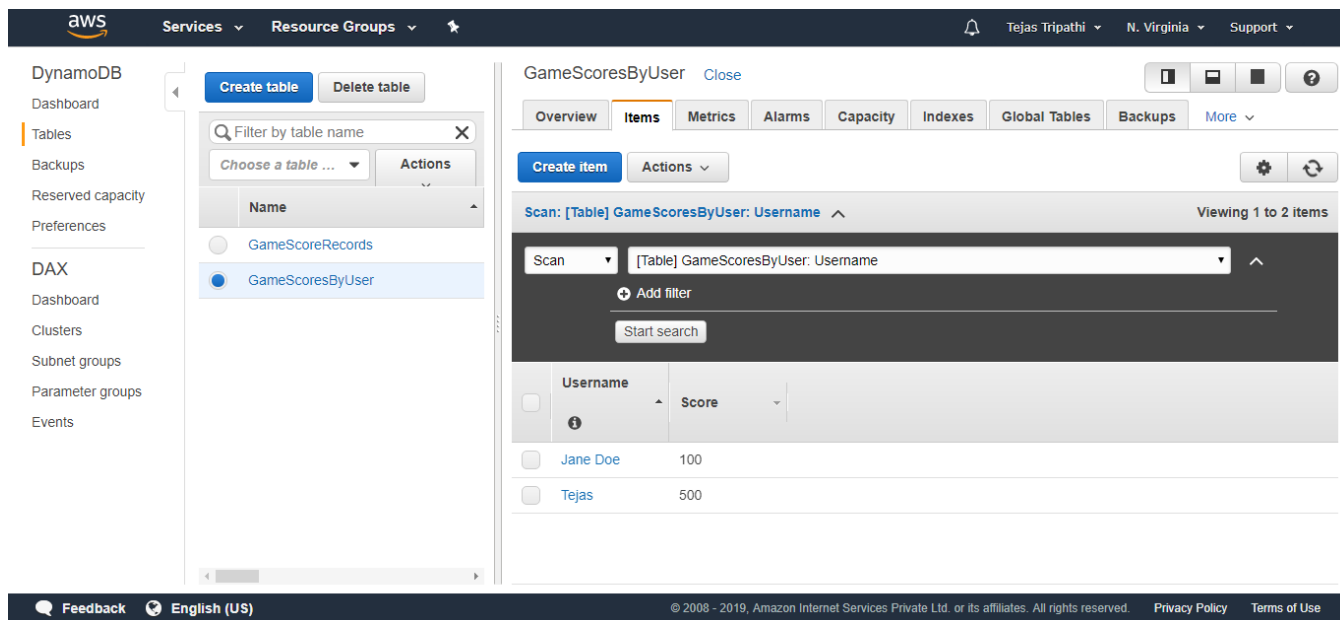
Username	Score
Jane Doe	100

Step 6: Trigger the Update

If we add contents to the original table:



It will now also reflect in the functionally linked table:



Thank you!