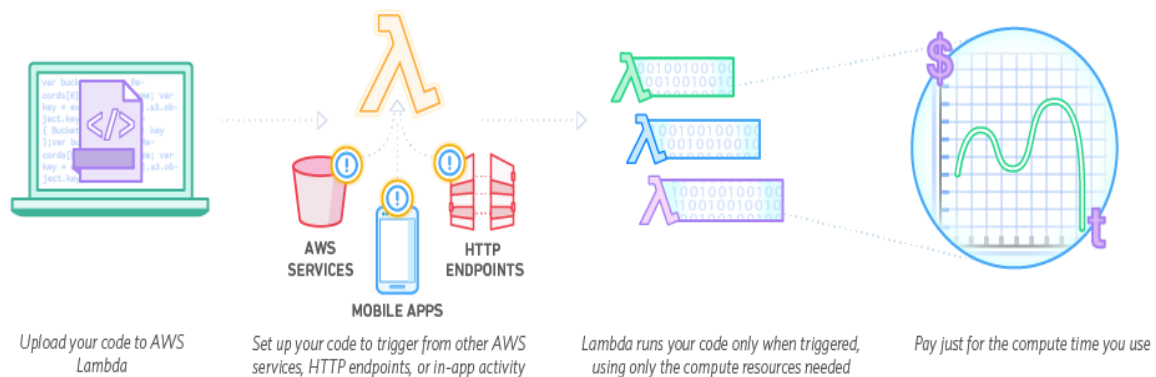


AWS Final Project

Submitted By : Shruti Tekriwal

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



Step 1 : Create an Amazon Kinesis Stream.

st301999/AWSProject x Amazon Kinesis Streams Manage x Serverless Architectures with Am x +

https://console.aws.amazon.com/kinesis/home?region=us-east-1#/streams/list

Apps Step 5: Run the CO...

aws Services Resource Groups S3 EMR Shruti Tekriwal N. Virginia Support

Kinesis streams

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.

Total shards in use: 1 Total shards remaining: 499

Create Kinesis stream Connect Kinesis consumers Actions

Filter Kinesis streams

Kinesis stream name	Number of shards	Status	Consumers using enhanced fan-out
Lab-Stream	1	Active	0

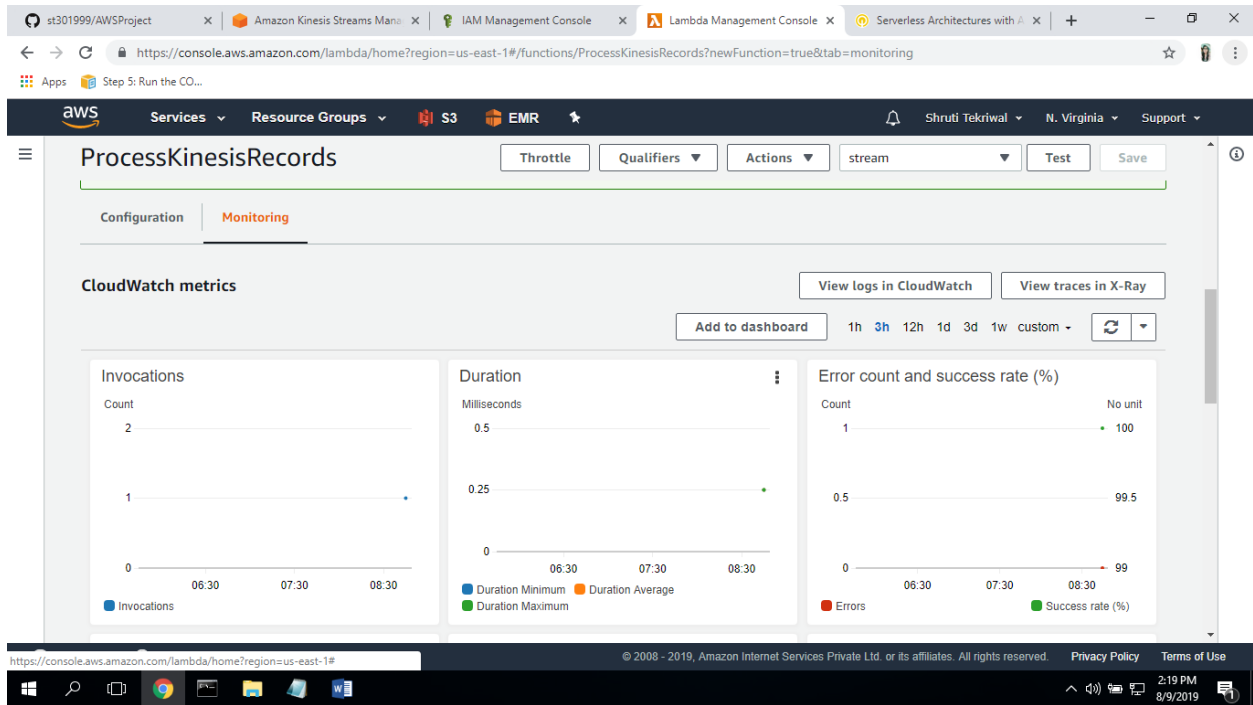
Windows taskbar: 2:05 PM 8/9/2019

Step 2 : Create a Lambda function.

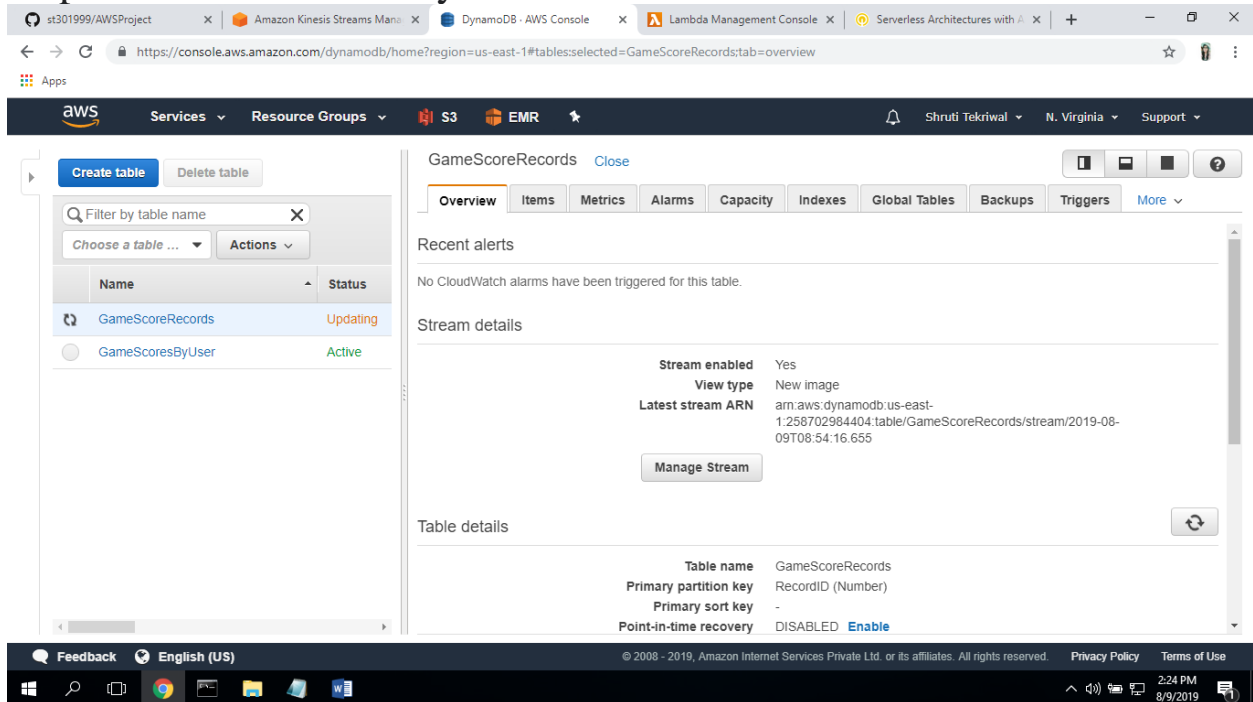
The screenshot shows the AWS Lambda console interface for creating a new function. The browser tabs include 'st301999/AWSProject', 'Amazon Kinesis Streams Man...', 'IAM Management Console', 'Lambda Management Console', and 'Serverless Architectures with A...'. The URL is 'https://console.aws.amazon.com/lambda/home?region=us-east-1#/functions/ProcessKinesisRecords?newFunction=true&tab=graph'. The console header shows 'Services', 'Resource Groups', 'S3', 'EMR', and user 'Shruti Tekriwal' in 'N. Virginia'. The breadcrumb is 'Lambda > Functions > ProcessKinesisRecords'. The function name 'ProcessKinesisRecords' is displayed with its ARN: 'arn:aws:lambda:us-east-1:258702984404:function:ProcessKinesisRecords'. Below the name are buttons for 'Throttle', 'Qualifiers', 'Actions', 'Select a test event', 'Test', and 'Save'. The 'Configuration' tab is active, showing a 'Designer' section with a key icon, a '+ Add trigger' button, and a list of layers: 'ProcessKinesisRecords' (selected), 'Layers (0)', 'Amazon CloudWatch', 'Amazon CloudWatch Logs', and 'Amazon EC2 Auto Scaling'. The footer shows 'Feedback', 'English (US)', copyright '© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', 'Terms of Use', and a system clock showing '2:14 PM 8/9/2019'.

Step 3 : Test the function.

The screenshot shows the AWS Lambda console displaying the execution result of the 'ProcessKinesisRecords' function. The browser tabs and URL are the same as in Step 2. The console header and breadcrumb are also the same. The function name 'ProcessKinesisRecords' is shown with its ARN. The 'Test' button is highlighted, and a dropdown menu shows 'stream' selected. A green notification box titled 'Execution result: succeeded (logs)' is displayed, containing a 'Details' section. The details section states: 'The section below shows the result returned by your function execution.' followed by a text box containing 'Successfully processed 1 records.'. Below this is a 'Summary' section with two columns of metadata: Code SHA-256, Request ID, Duration, Billed duration, Resources configured, and Max memory used. The 'Log output' section is partially visible at the bottom. The footer shows 'Feedback', 'English (US)', copyright '© 2008 - 2019, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.', 'Privacy Policy', 'Terms of Use', and a system clock showing '2:16 PM 8/9/2019'.



Step 4 : Create tables in DynamoDB.



Step 5 : Create a Lambda function.

The screenshot shows the AWS Lambda console for the function 'AggregateScoresByUser'. The 'Configuration' tab is active. In the 'Designer' section, a 'DynamoDB' trigger is connected to the 'AggregateScoresByUser' function. The function is marked as 'Saved'. The 'Layers' section shows '(0)' layers. The 'Actions' dropdown is set to 'score'. The 'Test' button is visible. The top navigation bar shows the user 'Shruti Tekriwal' in 'N. Virginia'.

The screenshot shows the AWS Lambda console for the function 'AggregateScoresByUser' with the 'Execution result: succeeded (logs)' panel open. The 'Details' section shows the result returned by the function: "Successfully processed 1 records." The 'Summary' section provides the following details:

Code SHA-256	Request ID
Bl13/elVFnpup8O/+YSPzjf0zF+KUN7gjk5a55Wc+40=	3cf49946-e2b9-42f8-8722-fd2a276cf16e
Duration	Billed duration
1188.39 ms	1200 ms
Resources configured	Max memory used
128 MB	76 MB

The 'Log output' section is also visible, indicating that the logging calls correspond to a single row within the CloudWatch log group.

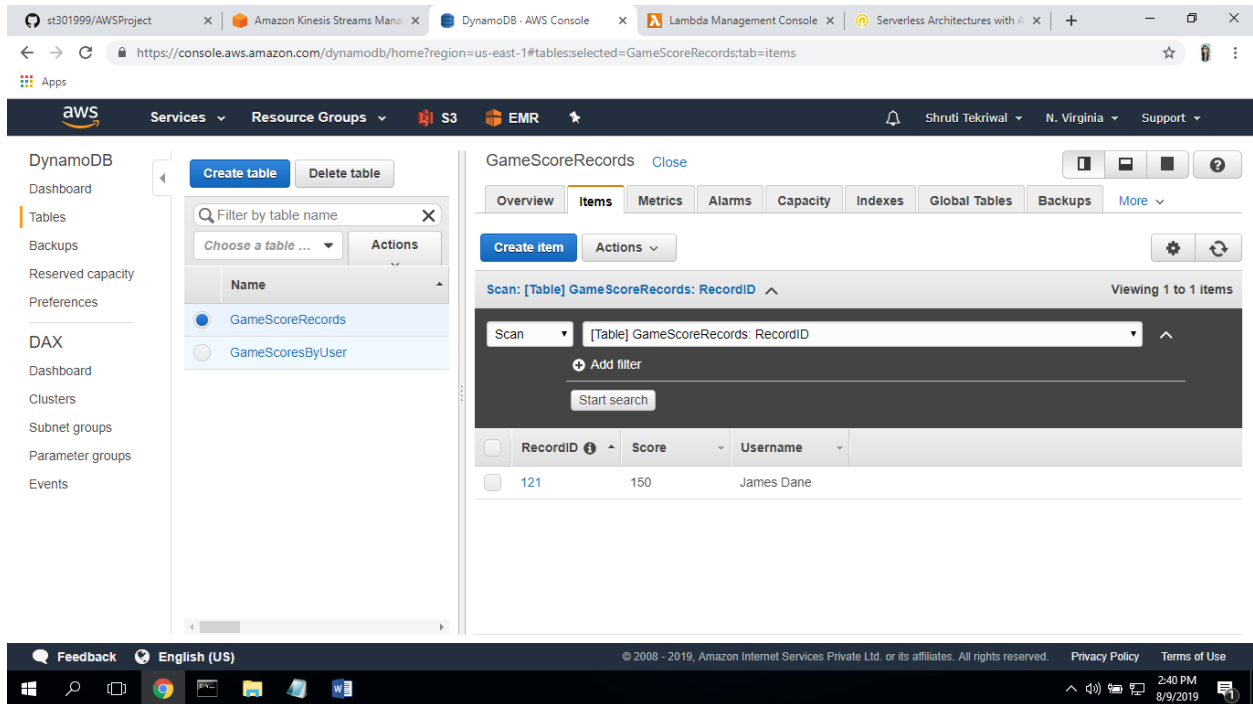
Now, we have to verify DynamoDB that the data was updated in the table.

The screenshot shows the AWS Management Console for DynamoDB. The left sidebar lists various services, with DynamoDB selected. The main panel shows the 'GameScoresByUser' table. The 'Items' tab is active, displaying a single item: 'Jane Doe' with a 'Score' of 100. The table structure is visible at the bottom, showing columns for 'Username' and 'Score'.

Username	Score
Jane Doe	100

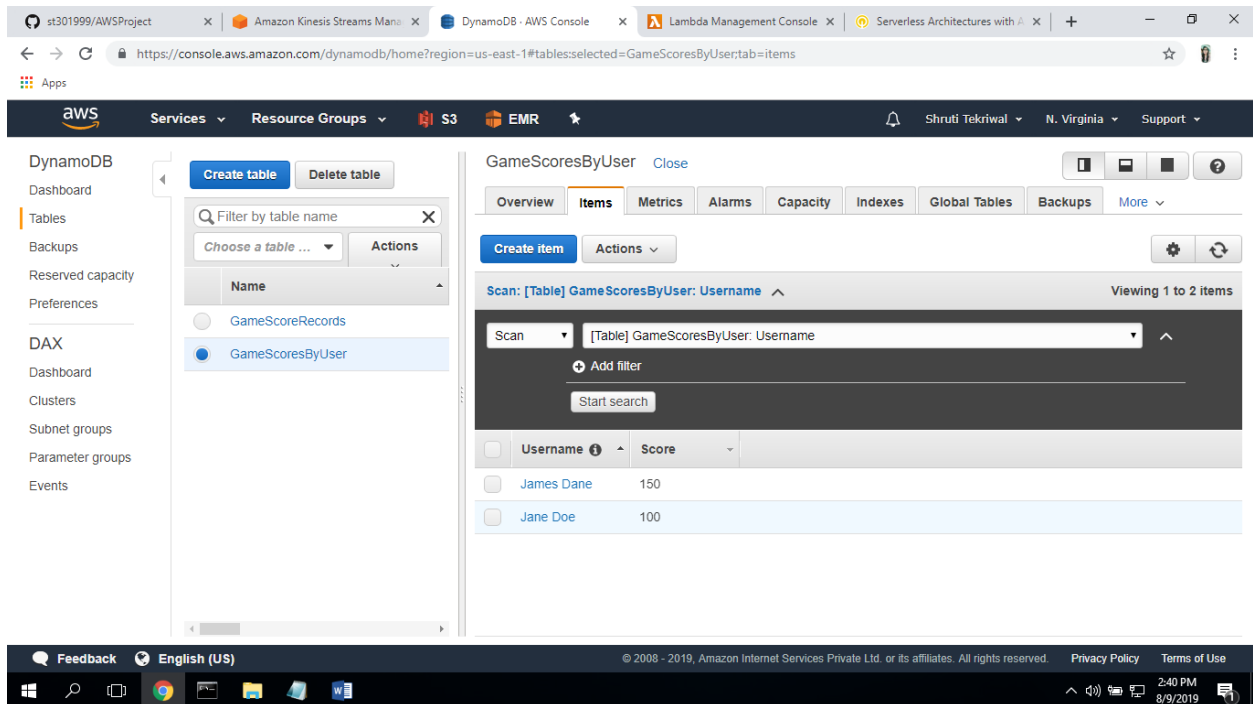
This table was previously empty when we created, but now we have an entry for *Jane Doe*.

Now, we have to trigger the update by inserting values in the score table, and confirming that the lambda updates the User table.



Here, we added the item in the GameScoreRecords Table.

We can see the new data added to the user table.



CONCLUSION :

1. Created a Lambda function from blueprint.
2. Created the Amazon Kinesis stream and used it to trigger the lambda function.
3. Used cloudwatch to monitor the function.
4. Created Amazon Dynamodb table and inserted sample data.
5. Enable Amazon Dynamodb stream.
6. Tested and enabled the Lambda Function on an Amazon Dynamodb table.

