Event-driven Programming with Amazon DynamoDB Streams and Amazon Lambda

Last Updated: 09/25/2015

Part I | Workshop Summary

Overview

In this session we will introduce you to the basics of event-driven programming using Amazon DynamoDB, its newly launched Streams feature and AWS Lambda. We will provide an overview of both AWS products and walk you through the process of building a real-world application using AWS Triggers, which combines DynamoDB Streams and Lambda.

Target Training Audiences

Software developers, architects, technical decision makers.

Technical/Knowledge Prerequisites

- Existing AWS account available for use during the workshop on your own laptop
- Basic knowledge of NoSQL databases and distributed systems
- Recommended experience with Amazon DynamoDB

Bootcamp Learning Objectives

As a result of this training, attendees will be able to:
Describe DynamoDB and how it fits into an application architecture
Design basic key schema for DynamoDB tables
Implement event-driven applications using DynamoDB Streams and AWS Lambda
Identify and troubleshoot issues in AWS Lambda functions

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Part II | Background

Sample Use Case: Storing Raw Data

Imagine a mobile gaming application storing users' scores in a DynamoDB table. The raw scores of users are stored directly in a table name "GameScoreRecords" with sample entries as such:

RecordID	Username	Score	Nickname
1	Jane Doe	100	JaneD
2	Bob Builder	150	
3	Jane Doe	250	JaneD

Consider and discuss the following with your group:

- In the above example, is "RecordID" is an acceptable choice for the primary key of the DynamoDB table in terms of even distribution? Why or why not?
- Assuming "RecordID" is the primary key for the table, in the above example there are two record entries with "Username" equal to "Jane Doe". Is this acceptable in DynamoDB?
- There is a record with no value for "Nickname", is this possible in DynamoDB?

While discussing the above questions, you may find it helpful to consult the public documentation for the DynamoDB Data Model.

Sample Use Case: Aggregating Data

Now that the application can store raw scores of users in a DynamoDB table, it may want to display the total score of each user in their profile. For instance, for user "Jane Doe" the total score for the application is the sum of two individual score records "100" and "250", yielding a result of "350" as a total score.

Consider and discuss the following with your group:

- Given the sample DynamoDB table above, how would one use DynamoDB operations to determine the total score for user "Jane Doe"?

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior
written consent from Amazon Web Services LLC.

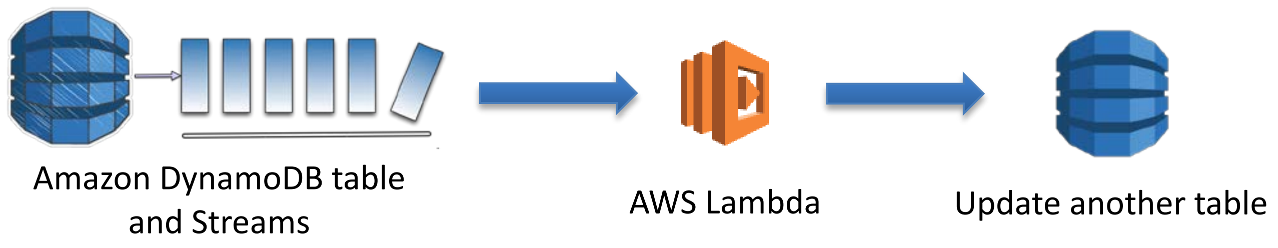
You may find the following documentation on DynamoDB Operations to be useful during your group discussion.

As you may have discovered through your group discussion, to obtain the aggregate score of a user requires scanning the entire DynamoDB table every time, because there is no way of knowing which records are related to a given user. After scanning the table, we can add up all the scores of a particular user to get the aggregate score for that user. This is inefficient and not scalable as the table grows in size. Imagine having to spend time scanning the table every time a query comes in!

Instead, what we want is to keep a running total of each user's aggregate score as the raw scores table is updated. We can keep track of these aggregate scores in a separate table, for example, continuing the example from above, the aggregate score table "GameScoresByUser" will look like the following:

Username	Score
Jane Doe	350
Bob Builder	150

Now we have come up with a use case where we can use DynamoDB Streams and AWS Lambda to enable a simple, event-driven programming scenario:



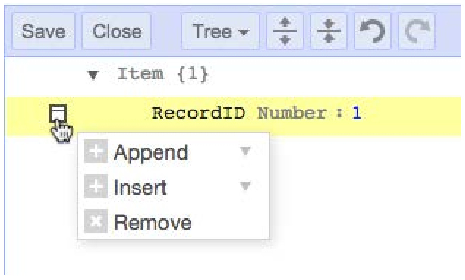
In the following exercises, we will create the above DynamoDB tables and build an event-driven application that keeps track of aggregate user game scores in real-time.

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Part III | Creating DynamoDB Tables

1. In this procedure, you will create a new table named GameScoreRecords.
2. Go to the AWS DynamoDB Console (https://us-west-2.console.aws.amazon.com/dynamodb/home?region=us-west-2) and login as necessary.
3. In the AWS Management Console, click **Services** and then click **DynamoDB**.
4. Click **Create Table**.
5. In the **Table Name** box, type **GameScoreRecords**.
6. For **Primary Key Type**, select **Hash.**
7. For the **Hash Attribute Type**, select **Number.**
8. In the **Hash Attribute Name** box, type **RecordID**.
9. Click **Continue**.
10. On the Add Indexes page, click **Continue**.
    a. Note: You will not be using an index for this exercise.
11. On the **Provisioned Throughput Capacity** page, click **Continue**.
    a. Note: You will accept the default read and write capacity for this exercise.
    b. Read Capacity Unit: A unit of read capacity represents one strongly consistent read per second for items as large as 4KB.
    c. Write Capacity Unit: A unit of write capacity represents one write per second for items as large as 1KB.
12. On the **Throughput Alarms** page:
    a. Accept the default setting of not enabling Streams.
    b. Accept the default setting of 80%.
    c. In the **Send notification to** box, type your email address.
13. Click **Continue**.
14. Click **Create**.
15. It takes several seconds for Amazon Dynamo DB to create the table. When the table is ready to use, it appears in the list of tables with a status of Active. If the status does not change, refresh the page.
16. Once the table is in "ACTIVE" status, you can explore the table by either double clicking the table, or clicking on "Explore Table" after selecting it.
17. Experiment with a few DynamoDB operations while exploring the table:
    - Inserting a new item: click on "Create Item", and enter "1" as the value for RecordID. To insert additional attributes, click on the action menu in front of the text "RecordID", please see illustration for its location:

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.



Item {1}
RecordID Number : 1

- Select “Insert” > “String”. Enter “Username” as the field name, and “Jane Doe” as the value. Repeat the actions to add additional attributes “Number” type field “Score” with value “100” and “String” type field “Nickname” with value “JaneD”. Once finished with the item, click on “Save” to see the PutItem successfully message, click on “OK” to dismiss the dialog and click on “Browse Items” tab to exit the Put Item view.
- Getting an existing item: in the “Browse Items” tab, select the “Get” option and leave everything else as the default, only enter “1” in the Hash Key field such that we are getting the record with RecordID equal to 1. Click on “Query” or “Go” to view the item we just inserted. Try the same query with RecordID equal to 2, the result set should be empty since we did not insert an item with RecordID value 2.
- Scanning the table: in the “Browse Items” tab, select the “Scan” option, then click on “Go” to initiate the scan. You should only see 1 item in the result because we only inserted 1 item. If we insert more items, all of them will show up when a scan is initiated. Note scanning the table could be potentially expensive because we consume read capacity units for all items that are returned in a scan.

18. Now that we have created the table to store raw game score records. Repeat steps 1 through 10 to create a second table to keep track of aggregate scores by user. Use the following parameters:
    - Table Name: GameScoresByUser
    - Primary Key Type: Hash
    - Hash Attribute Type: String
    - Hash Attribute Name: Username
    - No secondary indexes
    - Provisioned Throughput Capacity: 1 for both Read and Write
    - Default additional options
19. There is no need to add any items to the “GameScoresByUser” table, instead we can setup the lambda function such that it is automatically updated as items are inserted into “GameScoreRecords”. Let’s move onto the next section.

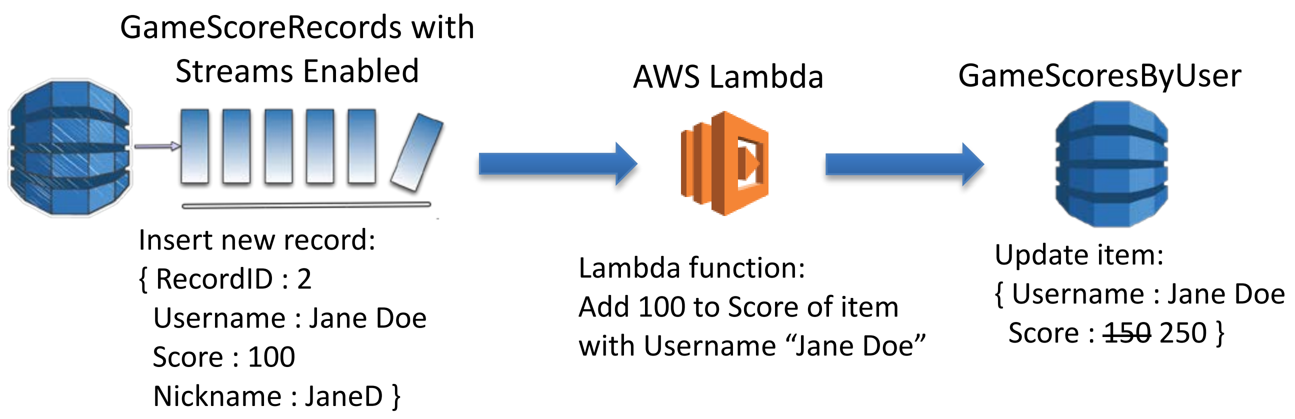
© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Part IV | Enabling DynamoDB Streams

Now that we have created both DynamoDB tables, we can start building the event-driven programming logic that will enable score aggregation in real-time. As a start, we need to enable DynamoDB Streams on the table with raw game score records “GameScoreRecords”.

1. In the AWS DynamoDB console, single click the table “GameScoreRecords”, make sure its status is “ACTIVE”.
2. In the details pane that pops up on the bottom of the console, click on the “Streams” tab.
3. Since we chose not to enable Streams during table creation, there should be no streams displayed. Instead, the option “Enable New Stream” should be available.
4. Click on “Enable New Stream”, for View Type, select “New Image – the entire item, as it appears after it was modified”. Since our application is to keep track of an aggregate total score, we only care about the “New Image” which gives us the new raw scores being added to the table.
5. Click on “Enable New Stream” in the pop up window to confirm Stream creation.
6. A new stream should appear in the “Streams” tab, with status “Enabling”, wait a few minutes and use the refresh button in the top right corner to monitor the status.
7. After a few minutes, the Stream status should transition to “Enabled”, we now have a Stream recording updates to our “GameScoreRecords” table!

Now that Streams is enabled on the “GameScoreRecords” table, all inserts will be recorded in the stream. As more records are added to the table, we can attach a Lambda function to the stream and process each record by adding the score to the user’s running aggregate total. Here is a quick illustrative example:



© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Part V | Creating Lambda Function

1. Go to the AWS Lambda console (https://us-west-2.console.aws.amazon.com/lambda/home?region=us-west-2#/functions).
2. Click on "Create a lambda function", select the "dynamodb-process-stream" blueprint.
3. Select "DynamoDB" as event source type, "GameScoreRecords" as DynamoDB table, "100" as batch size and "Latest" as starting position. This ensures we are processing updates to the raw game score records table in batches of 100, starting with the most recent record.
4. Click on "Next" to configure the lambda function, name the function "AggregateScoresByUser". Use an appropriate description to remember what this function does and select "Node.js" as runtime. Using node.js allows us to edit and test the lambda function inside the AWS console.
5. Leave the default lambda function code, we will work on the actual code later.
6. For lambda function handler, leave the default "index.handler" as is.
7. For lambda function role, select "DynamoDB event stream role" under the "Create new role" header. This opens a separate window or tab under the IAM (Identity Access Management) service.
8. Creating a new IAM role gives the lambda function the permissions it needs in order to perform AWS actions. Select "Create a new IAM Role" for IAM Role, and enter "lambda_dynamodb_streams_demo" as the Role Name. Click on "View Policy Document" to look at the default permissions given to the lambda function. Click on "Edit" to enable modification to the policy document.
9. The default policy allows the lambda function to get records from DynamoDB Streams and publish to CloudWatch logs for monitoring. In this exercise, we also need the lambda function to perform updates to our "GameScoresByUser" table, so we need to add an additional permission. Please copy the full policy document from Appendix A to replace the existing policy document.
10. Click on "Allow" to create the IAM role with our newly edited policy document. This should bring you back to the lambda console with "lambda_dynamodb_streams_demo" as the lambda role selection.
11. Leave the defaults for Advanced Settings, click on "Next" to continue creating the lambda function.
12. Review all the parameters we just entered, ensure the correct function name, runtime, handler and event source are selected. For the enable event source option, leave the default

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior
written consent from Amazon Web Services LLC.

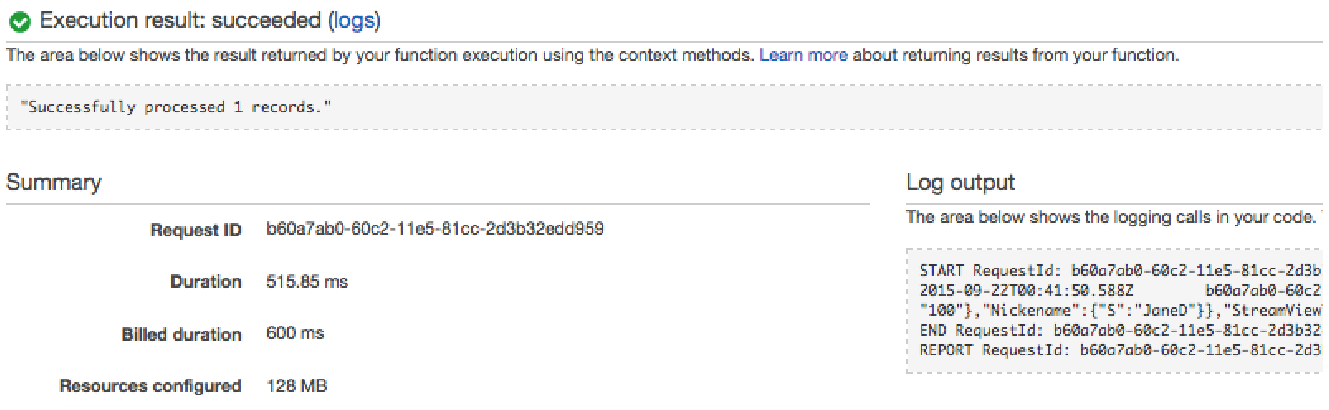
at “Enable later” since we want to write our lambda function and test it before enabling it on our table.

13. Click on “Create function” to finish the creation process.

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior
written consent from Amazon Web Services LLC.

Part VI | Writing and Testing Lambda Function

1. After the lambda function has been created, a new detailed lambda function view is displayed. Click on the "Code" tab to view the current lambda code.
2. The default skeleton lambda function code simply takes each record and logs it. As a basic test, we can try out this function on a sample event or record.
3. Click on "Actions" near the top, then "Configure sample event". This allows us to create a sample Streams event or record, as if the record had been inserted into a DynamoDB table. This way, we can fully test out our lambda function before enabling it on the actual DynamoDB table.
4. In the pop-up window with input sample event, select event template "DynamoDB Update" to see the base template for a DynamoDB update stream record. The default base template contains 3 DynamoDB Streams records, with some dummy values for attributes. We can replace these to look like records from the actual "GameScoreRecords" table. Please delete all the sample records and replace them with the one provided in Appendix B.
5. Click on "Submit" after pasting in our own sample event. Then, click on the "Test" button near the top to run the lambda code with the newly edited sample event. Scroll to the bottom to see the test run results, it should look similar to the screen capture below:



The area below shows the result returned by your function execution using the context methods. Learn more about returning results from your function.

```
"Successfully processed 1 records."
```

Request ID	b60a7ab0-60c2-11e5-81cc-2d3b32edd959
Duration	515.85 ms
Billed duration	600 ms
Resources configured	128 MB

The area below shows the logging calls in your code.

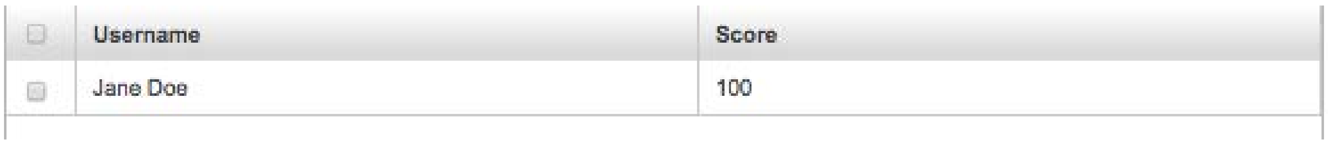
```
START RequestId: b60a7ab0-60c2-11e5-81cc-2d3b
2015-09-22T00:41:50.588Z        b60a7ab0-60c2
"100"},"Nickename":{"S":"JaneD"}},"StreamView
END RequestId: b60a7ab0-60c2-11e5-81cc-2d3b32
REPORT RequestId: b60a7ab0-60c2-11e5-81cc-2d3
```

6. Now we can work on the actual lambda function code. Delete all the existing sample code and replace it with the skeleton code given in Appendix C, which provides a start and outlines the main functionality we need to implement in the lambda function. Complete implementation in the lambda function by uncommenting code where a "TODO" is marked in the code, including:
   - Configuring the AWS client with the correct region name

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

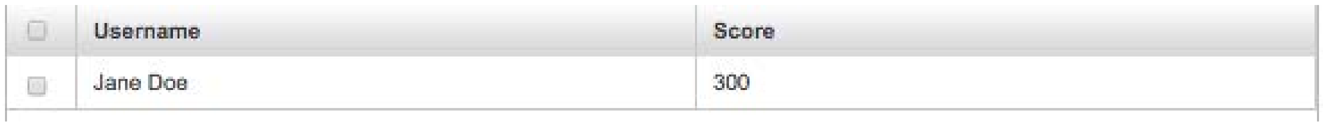
- Retrieving the new image for each Streams record and extracting the values of the Username and Score attribute
- Generating an UpdateItem request to the “GameScoresByUser” table to add the score to the running total of the appropriate user

7. You can test your code as you are working on it. When the lambda execution result shows “succeeded”, you may double check the update item operation actually executed by going to the AWS DynamoDB console and check the content of the “GameScoresByUser” table. If the lambda function is working properly, you should see an entry for “Jane Doe” as follows:



	Username	Score
☐	Jane Doe	100

Of course, as you execute the test code multiple times by clicking on “Test” repeatedly, the Score value for “Jane Doe” should increase by increments of 100. Clicking on “Test” twice more should yield an aggregate score of 300:



	Username	Score
☐	Jane Doe	300

8. Once you have made sure the lambda function works properly on the sample event. You can go ahead and enable the actual event source on the raw game score records table “GameScoreRecords”.
9. Go back to the AWS lambda console, select the “AggregateScoresByUser” function, click on the “Event sources” tab and click on state of the event source DDB:GameScoreRecords (should say “Disabled”).
10. Click on “Enable” on the pop-up dialog to enable the event source, a success message should show up notifying you that the lambda function is receiving events from DDB:GameScoreRecords.
11. Congratulations, you now have a fully enabled event-driven pipeline from the GameScoreRecords table to the GameScoresByUser table! Test it out by inserting a few sample items in the GameScoreRecords table, remember to include values for the Username and Score attribute.

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Part VII | (Optional) Adding Basic Error Handling

As you may have noticed, the current implementation of the lambda function does not perform any error or exception handling. For instance, what happens if the incoming Streams record does not contain an attribute named “Username”? It is entirely possible for someone to accidentally insert an item without this attribute, unless the mobile gaming application explicitly performs this check. Regardless, it is typically best practice to perform some basic sanity checks.

As an optional exercise, try adding some basic checks to the lambda function to prevent errors when records with unexpected formats appear:

- NewImage is undefined or null (this can happen when the stream record is a “DELETE” event instead of “INSERT” or “MODIFY”)
- The “Username” attribute is undefined or null
- The “Score” attribute is undefined or null

Try adding checks for the above conditions in your lambda function, and skip records that meet any of these conditions because they are considered “invalid” for our use case. If you get stuck and need hints, please refer to the complete solution in the “Solutions” directory of the workshop package.

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Part VIII | (Optional) Receiving Email Notifications for Top Scores

As the developer of the mobile gaming application, you may be interested in maintaining a leaderboard of top scores. As a start, we can try to implement a lambda function that simply sends an email notification whenever a high score is registered in the “GameScoresByUser” table. To do this, we will need to first set up an Amazon Simple Notification Service (SNS) topic:

1. Go to the AWS SNS Console (https://us-west-2.console.aws.amazon.com/sns/home?region=us-west-2)
2. Click on “Create Topic” under Common actions.
3. Enter “TopScores” for both Topic name and Display name.
4. Click on “Create Topic”.
5. On the Topic Details page, click on “Create Subscription”.
6. Choose “Email” as the Protocol.
7. Type your own email in the field “Endpoint”.
8. Click on “Create Subscription”.
9. You should receive an email requesting confirmation of your subscription to this SNS topic, make sure to click on the “Confirm subscription” link in the email.
10. Double check your subscription is confirmed by refreshing the SNS topic page until the Subscription ID is no longer “PendingConfirmation”, but instead a populated ID.

Now that you have created an SNS topic and subscribed your email address to it, it’s time to utilize your knowledge of lambda functions and create a lambda function that is triggered by the “GameScoresByUser” table. Remember you need to achieve the following:

- Create a new IAM role that grants permission “SNS:publish” to the specific SNS topic you just created using the Topic ARN.
- Configure the lambda function to check if the user score exceeded a given threshold
- If the threshold is exceeded, publish an SNS message to the topic you created

For reference, please see Appendix D for the IAM role policy and Appendix E for the skeleton lambda function code.

After you have finished the lambda function, you can test it out by going back to the raw records “GameScoreRecords” table and inserting a few entries. The aggregate score should be updated in “GameScoresByUser”, and you should receive an email notification when the total aggregate score of a user exceeds the top score threshold you’ve configured in your lambda function.

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Appendix A | IAM Policy Document for Lambda Function

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator",
                "dynamodb:DescribeStream",
                "dynamodb:ListStreams",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
       {
           "Effect": "Allow",
           "Action": [
               "dynamodb:UpdateItem"
           ],
           "Resource": "arn:aws:dynamodb:us-west-2:*:table/GameScoresByUser"
       }
    ]
}
```

Back

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Appendix B | Sample Event for Lambda Function Testing

```
{
  "Records": [
    {
      "eventID": "1",
      "eventVersion": "1.0",
      "dynamodb": {
        "Keys": {
          "RecordID": {
            "S": "2"
          }
        },
        "NewImage": {
          "RecordID": {
            "S": "2"
          },
          "Username": {
            "S": "Jane Doe"
          },
          "Score": {
            "N": "100"
          },
          "Nickname": {
            "S": "JaneD"
          }
        },
        "StreamViewType": "NEW_IMAGE",
        "SequenceNumber": "111",
        "SizeBytes": 26
      },
      "awsRegion": "us-west-2",
      "eventName": "INSERT",
      "eventSourceARN": "arn:aws:dynamodb:us-west-2:account-
id:table/GameScoreRecords/stream/2015-10-07T00:48:05.899",
      "eventSource": "aws:dynamodb"
    }
  ]
}
```

Back

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Appendix C | Lambda Function Skeleton Code

```
// Set up AWS client
var AWS = require('aws-sdk');
var dynamodb = new AWS.DynamoDB();

// TODO update AWS configuration to set region
// AWS.config.update({region : 'us-west-2'});

exports.handler = function(event, context) {
    // Keep track of how many requests are in flight
    var inflightRequests = 0;

    event.Records.forEach(function(record) {
        console.log('DynamoDB Record: %j', record.dynamodb);

        // Get the new image of the DynamoDB Streams record
        var newItemImage = record.dynamodb.NewImage;

        // Set the appropriate parameters for UpdateItem
        // Refer to the ADD operation in the UpdateItem API for UpdateExpression
        // http://docs.aws.amazon.com/amazondynamodb/latest/APIReference/API_UpdateItem.html
        // Adds the specified value to the item, if attribute does not exist, set the attribute
        var updateItemParams = {
            TableName: "GameScoresByUser",
            Key : {
                Username : newItemImage.Username

            },
            UpdateExpression : 'ADD Score :attrValue',
            ExpressionAttributeValues : {
                ':attrValue' : newItemImage.Score
            }
        }

        // Make a callback function to execute once UpdateItem request completes
        // It may be helpful to refer to the updateItem method for the Javascript SDK
        // http://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/DynamoDB.html#updateItem-property
        var updateItemCallback = function(err, data) {
            if (err) {
                // log errors
                console.log(err, err.stack);
            } else {
                // check if all requests are finished, if so, end the function
                inflightRequests--;
                if (inflightRequests === 0) {
                    context.succeed("Successfully processed " + event.Records.length + " records.");
                }
            }
        };

        // TODO send UpdateItem request to DynamoDB
        // dynamodb.updateItem(updateItemParams, updateItemCallback);

        // TODO increase count for number of requests in flight
        // inflightRequests++;
    });

    // If there are no more requests pending, end the function
    if (inflightRequests === 0) {
        context.succeed("Successfully processed " + event.Records.length + " records.");
    }
};
```

Back

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Appendix D | IAM Policy for Optional Exercise

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetRecords",
                "dynamodb:GetShardIterator",
                "dynamodb:DescribeStream",
                "dynamodb:ListStreams",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "SNS:publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:*:TopScores"
        }
    ]
}
```

Back

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior written consent from Amazon Web Services LLC.

Appendix E | Lambda Function Skeleton Code for Optional Exercise

```
// Set up AWS client
var AWS = require('aws-sdk');
var sns = new AWS.SNS();

// Update AWS configuration to set region
AWS.config.update({region : 'us-west-2'});

// Set a high score threshold
var TOP_SCORE = 300;

exports.handler = function(event, context) {
    var inflightRequests = 0;
    event.Records.forEach(function(record) {
        // Publish to SNS if the user scored above the threshold
        if (new Number(record.dynamodb.NewImage.Score.N) > TOP_SCORE) {
            inflightRequests++;
            sns.publish({
                    // TODO insert correct ARN here
                    TopicArn: "arn:aws:sns:us-west-2:ACCOUNT_ID:TopScores",
                    Message: record.dynamodb.NewImage.Username.S + " scored more than " + TOP_SCORE
                },
                function(err, data) {
                    if (err) {
                        console.log(err);
                    }
                    else {
                        if ((--inflightRequests) === 0) context.succeed("Successfully processed " +
event.Records.length + " records.");
                    }
                }
            );
        }
    });
    if ((inflightRequests) === 0) context.succeed("Successfully processed " + event.Records.length + "
records.");
};
```

Back

© 2015 Amazon Web Services LLC and its affiliates. All rights reserved. 410 Terry Avenue North, Seattle, WA 98109-5210
Amazon Web Services LLC Confidential. The information in this document may not be disclosed without prior
written consent from Amazon Web Services LLC.