[](http://aws.amazon.com/)

Amazon Web Services

Data Engineering Immersion Day

Real-Time Clickstream Anomaly Detection

Kinesis Analytics - Prelab setup

Jun 2019

Table of Contents

[Introduction 2](#_Toc9876009)

[CloudFormation Stack Deployment 3](#_Toc9876010)

[Set up the Amazon Kinesis Data Generator 6](#_Toc9876011)

[Set up Email and SMS Subscription 8](#_Toc9876012)

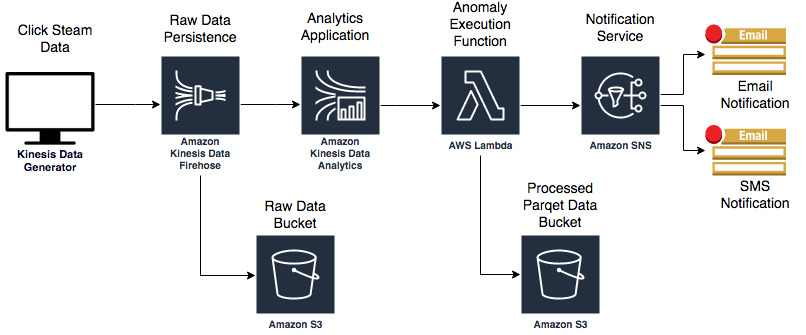
[Observe AWS Lambda Anomaly function: 9](#_Toc9876013)

[Appendix: CloudFormation Template 10](#_Toc9876014)

# Introduction

This guide will help you set up the pre-lab environment for the Real-Time Clickstream Anomaly Detection Amazon Kinesis Data Analytics lab.

The AWS CloudFormation template **Kinesis\_Pre\_Lab.json** included with this lab deploys the following architecture without the highlighted components. You will set up the highlighted components manually.



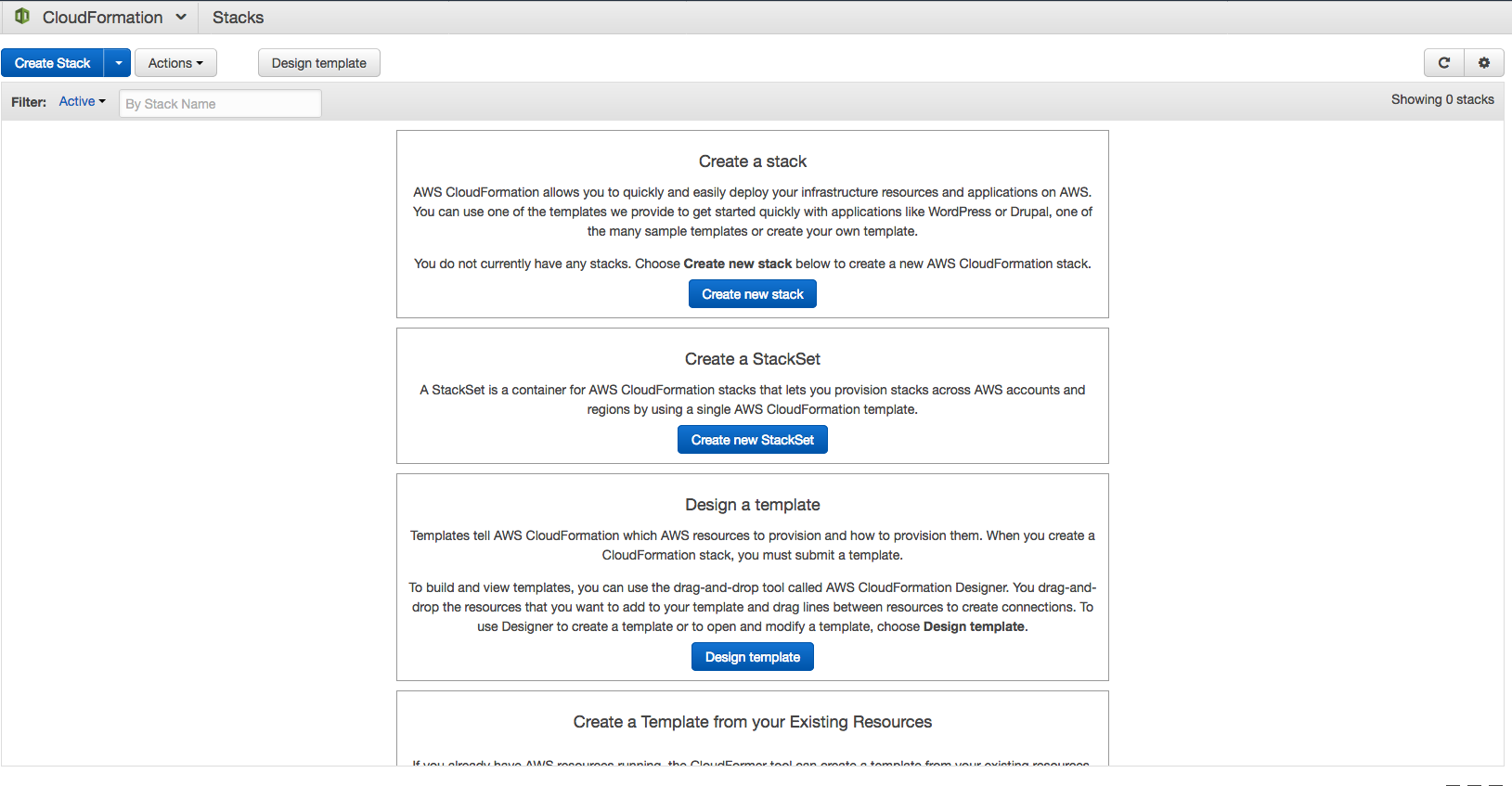
After you deploy the CloudFormation template, sign into your account to view the following resources:

* Two Amazon Simple Storage Service (Amazon S3) buckets: You will use these buckets to persist raw and processed data.
* One AWS Lambda function: This Lambda function will be triggered once an anomaly has been detected.
* Amazon Simple Notification Service (Amazon SNS) topic with an email and phone number subscribed to it: The Lambda function will publish to this topic once an anomaly has been detected.
* Amazon Cognito User credentials: You will use these user credentials to log into the Kinesis Data Generator to send records to our Amazon Kinesis Data Firehose.

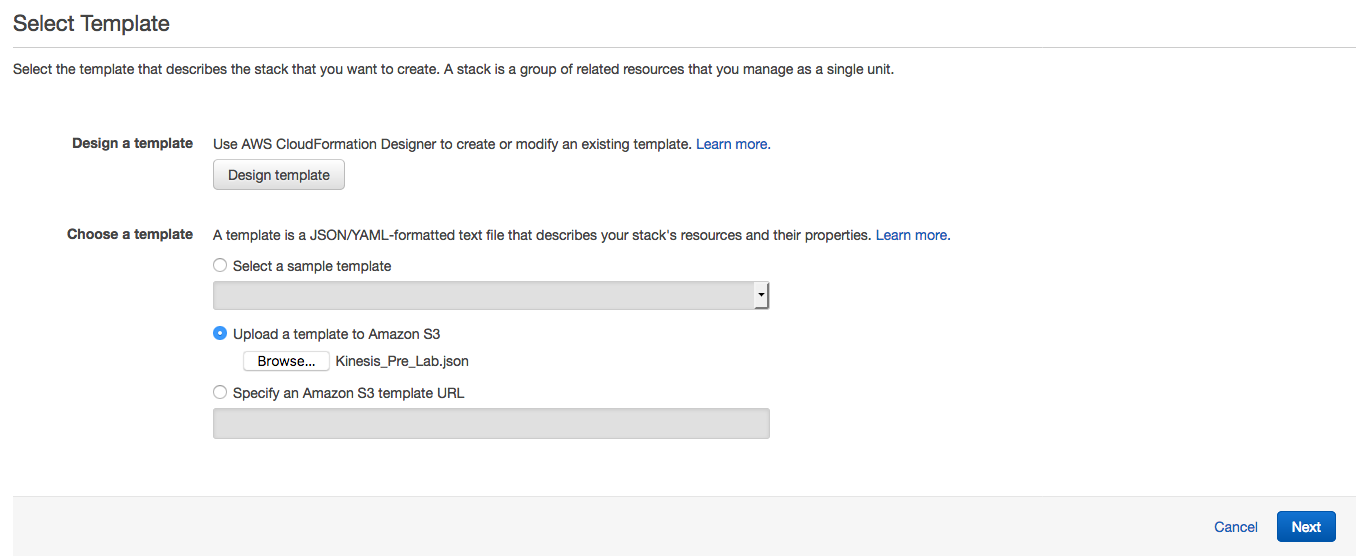
# CloudFormation Stack Deployment

**\*\*Make sure you are in US-WEST-2 (Oregon) region\*\***

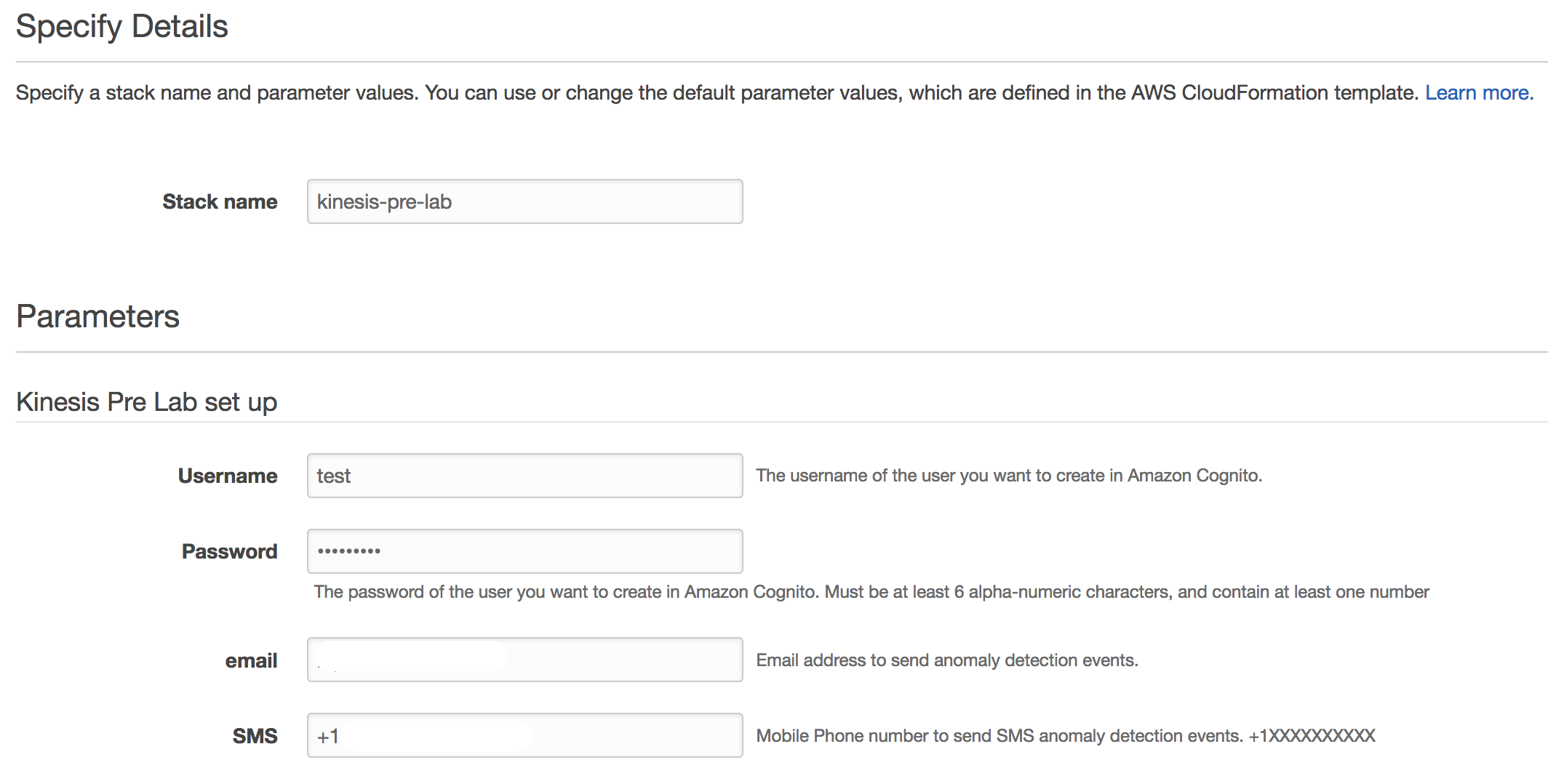
1. In your AWS account, navigate to the **CloudFormation console**.
2. On the CloudFormation console, Click Create **new Stack**. Make sure you have selected **Oregon** as region.



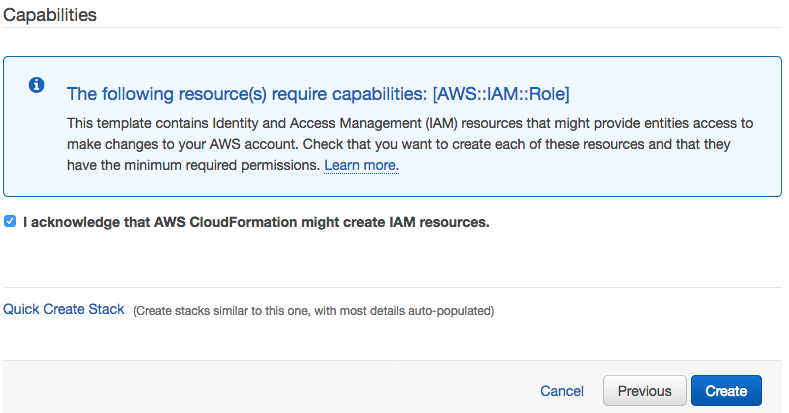
1. In the **Select Template** section, select **Upload a template to Amazon S3. Then,** browse to your **Kinesis\_Pre\_Lab.json** file provided with your lab package. Alternatively , you can create file by copying AWS CloudFormation code from [Appendix](#_Appendix_A).



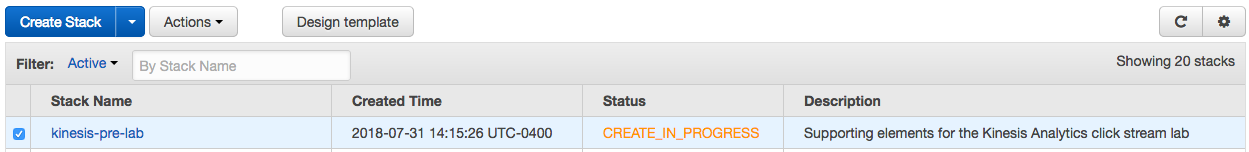
1. Click **Next** at the bottom of the select template page in as shown in above screenshot**.**
2. In the **Specify Details** section, for **Stack name**, type **kinesis-pre-lab**.
3. **In the Parameters** section,fill the following fields:
   * **Username:** This is your username to login to the Kinesis Data Generator
   * **Password:** This is your password for the Kinesis Data Generator. The password must be at least 6 alpha-numeric characters and contain at least one number and a capital letter.
   * **Email:** Type an email address that you can access. The SNS topic sends a confirmation to this address.
   * **SMS:** Type a phone number (+1XXXXXXXXX) where you can receive texts from the SNS topic.



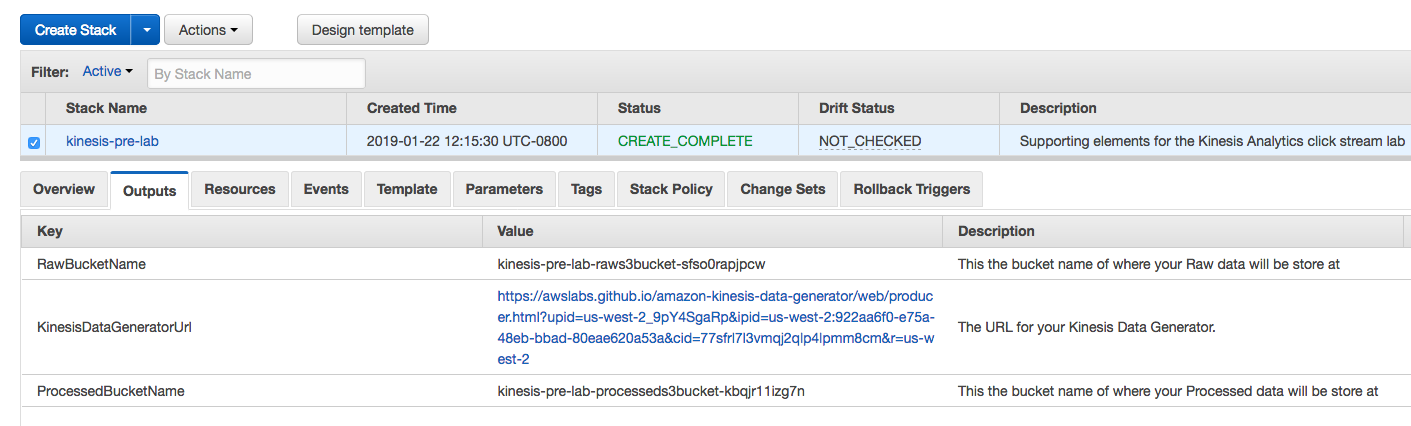
1. In the **Options**, section, keep the default values.
2. In the **Review** section, select the check box marked **I acknowledge that AWS CloudFormation might create IAM resources**.



1. Click **Create**. CloudFormation redirects you to your existing stacks. The **kinesis-pre-lab** displays a **CREATE\_IN\_PROGESS** status.



1. Once your stack is deployed, click the **Outputs** tab to view more information:
   * **KinesisDataGeneratorUrl:** This value is the Kinesis Data Generator (KDG) URL.
   * **RawBucketName –** Store raw data coming from KDG.
   * **ProcessedBucketName –** Store transformed data



Congratulations! You are all done with the CloudFormation deployment.

# Set up the Amazon Kinesis Data Generator

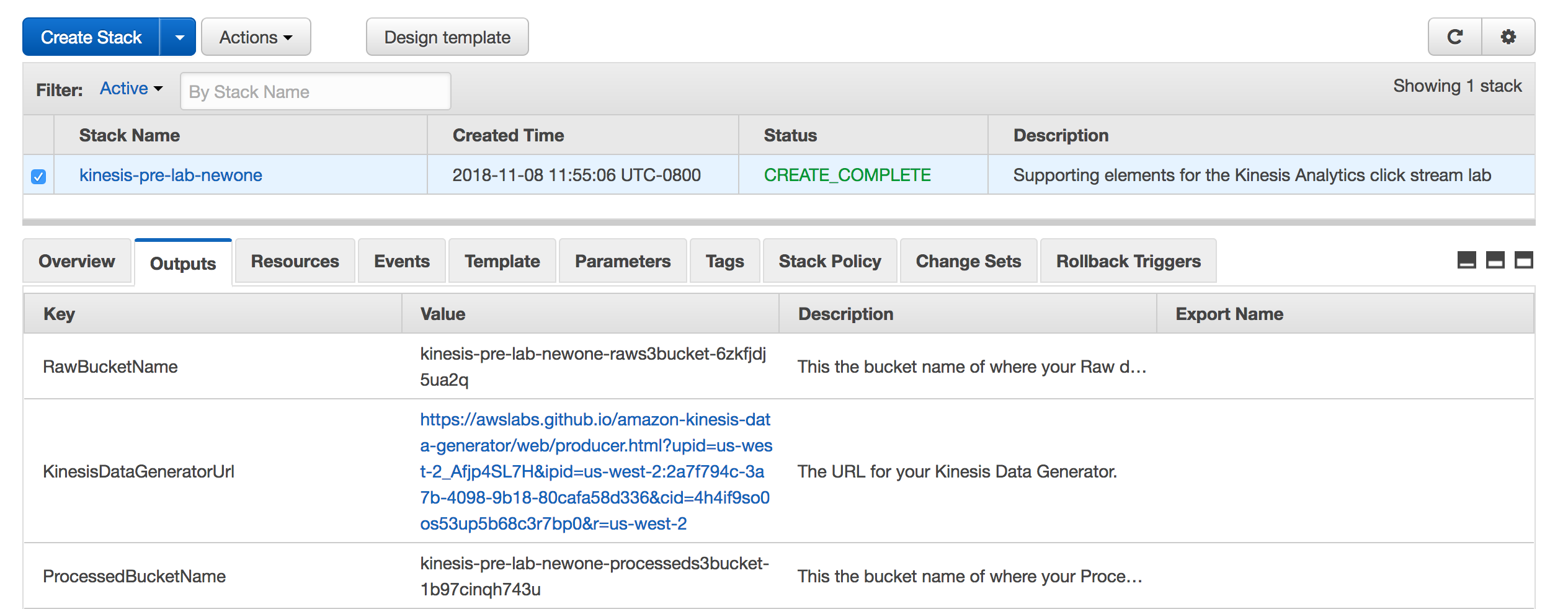
On the **Outputs** tab, notice the **Kinesis Data Generator URL**. Navigate to this URL to login into the Amazon Kinesis Data Generator (Amazon KDG).

The KDG simplifies the task of generating data and sending it to Amazon Kinesis. The tool provides a user-friendly UI that runs directly in your browser. With the KDG, you can do the following tasks:

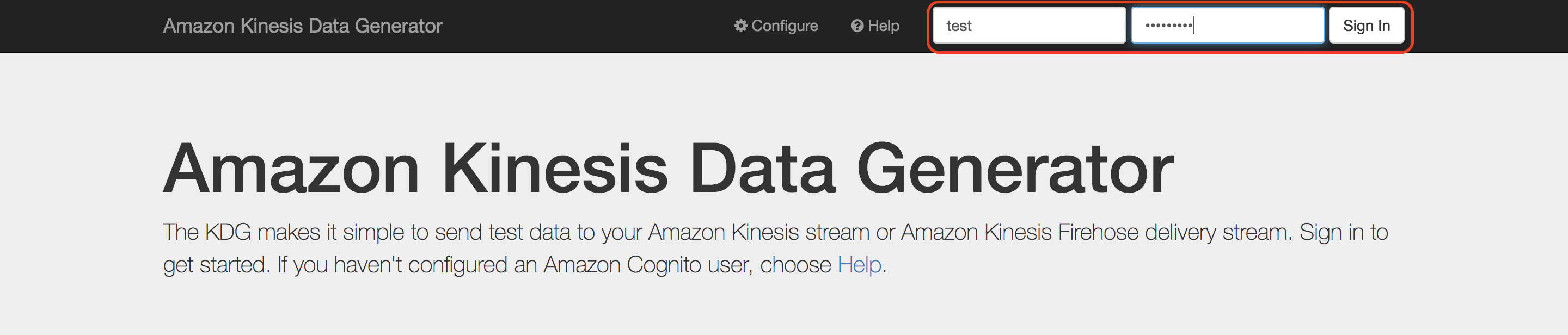
* Create templates that represent records for your specific use cases
* Populate the templates with fixed data or random data
* Save the templates for future use
* Continuously send thousands of records per second to your Amazon Kinesis stream or Firehose delivery stream

Let’s test your Cognito user in the Kinesis Data Generator.

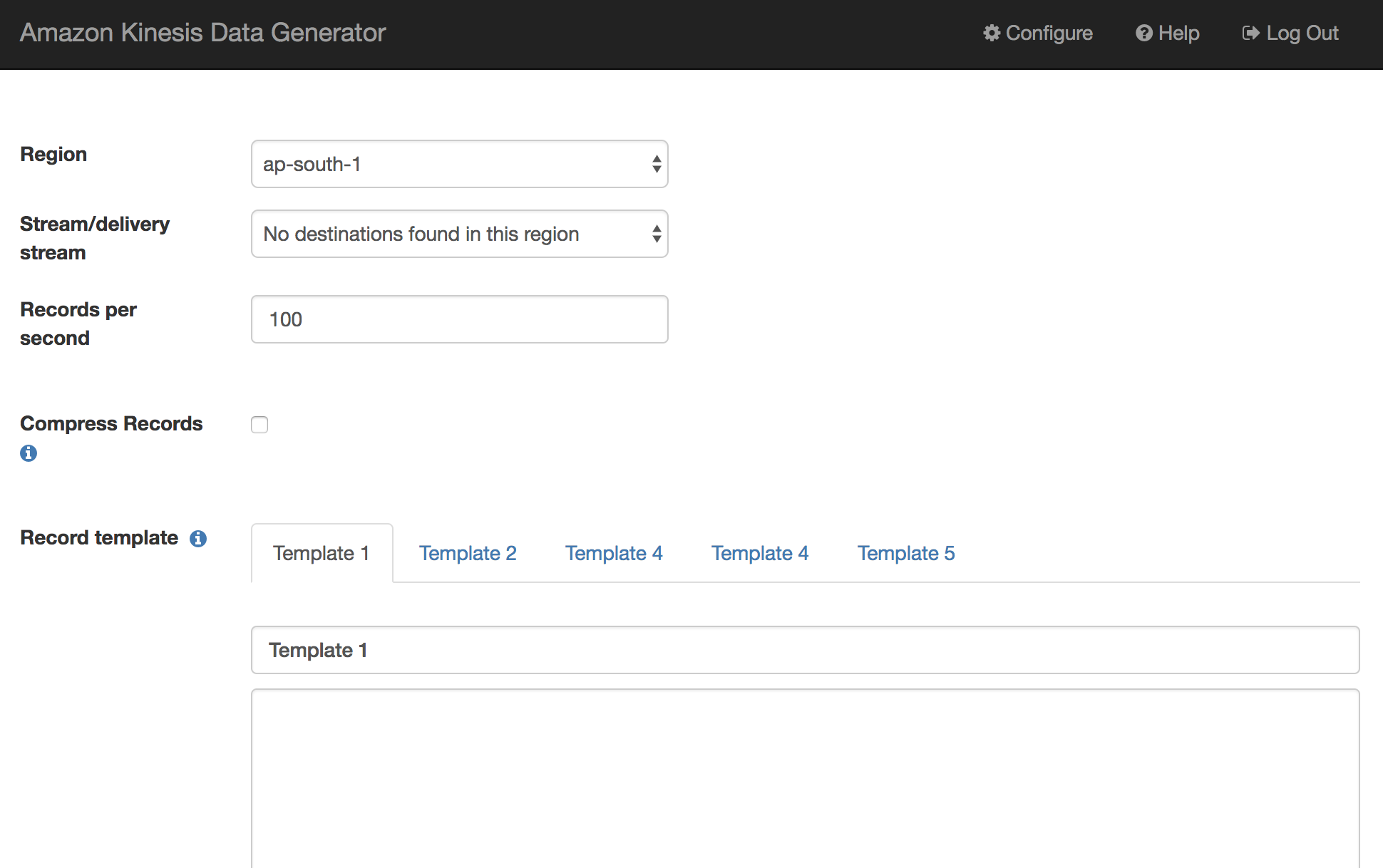
1. On the **Outputs** tab, click the **KinesisDataGeneratorUrl**.

****

1. Sign inusing the **username** and **password** you entered in the CloudFormation console.



1. After you sign in, you should see the KDG console. You need to set up some templates to mimic the clickstream web payload.



Create the following templates but don’t click on Send Data yet . We will do that during main lab:

* 1. **Schema Discovery Payload**

{"browseraction":"DiscoveryKinesisTest", "site": "yourwebsiteurl.domain.com"}

* 1. **Click Payload**

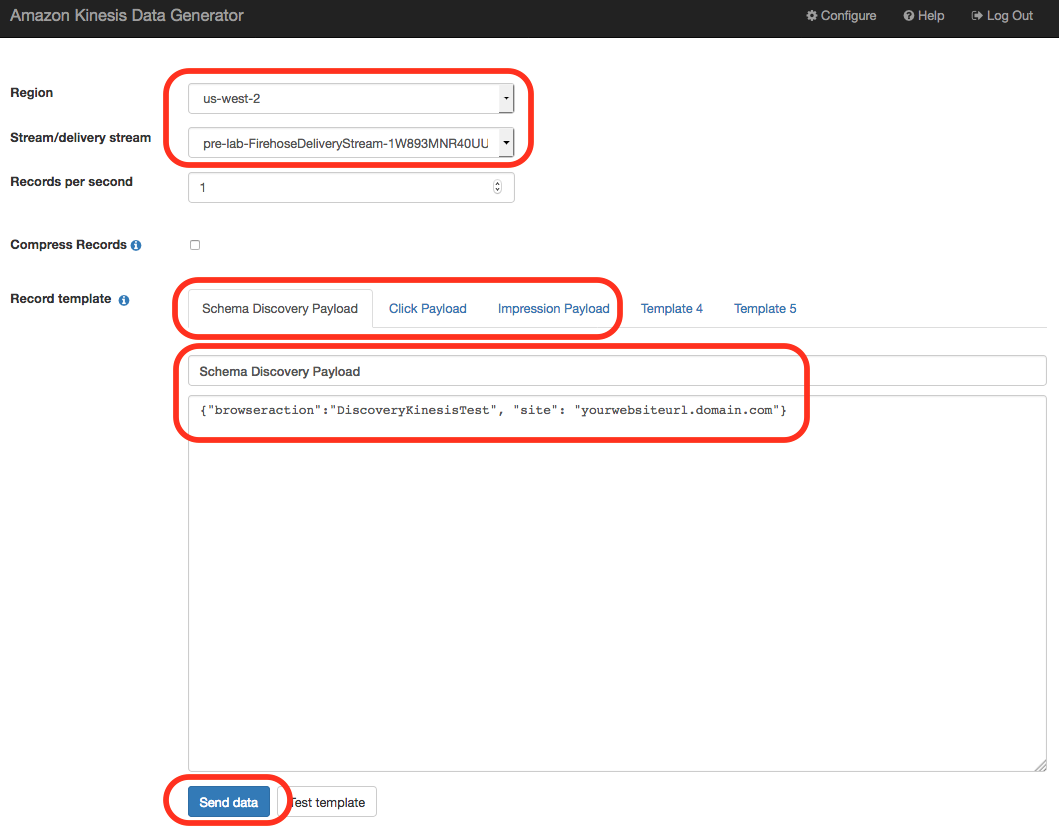
{"browseraction":"Click", "site": "yourwebsiteurl.domain.com"}

* 1. **Impression Payload**

{"browseraction":"Impression", "site": "yourwebsiteurl.domain.com"}

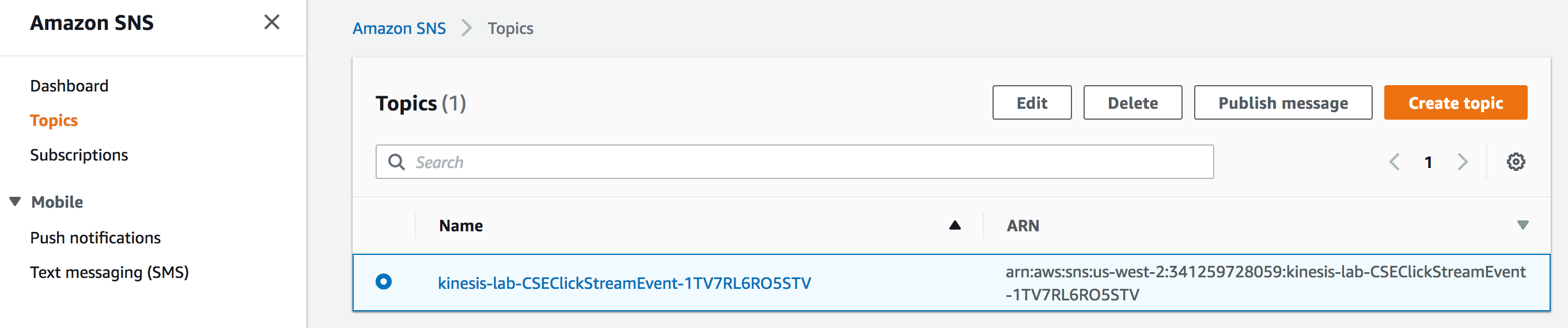
Note that your Kinesis Data Firehose has been deployed in **US-WEST-2**.

Your Amazon Kinesis Data Generator console should look similar to this example.

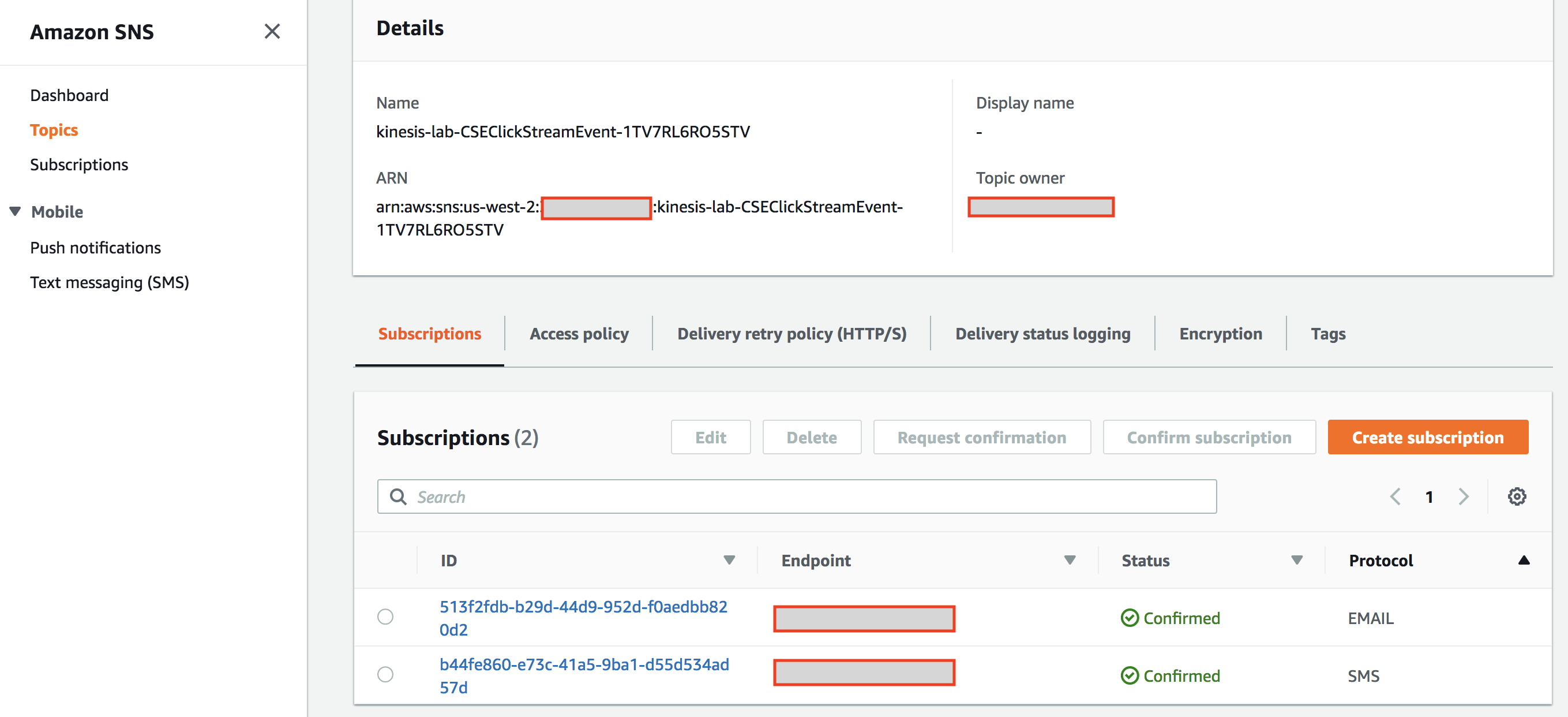


# Set up Email and SMS Subscription

1. On the Amazon SNS navigation menu, select **Topics**. An SNS topic named **<stackname>-CSEClickStreamEvent-<random string>** appears in the display.:



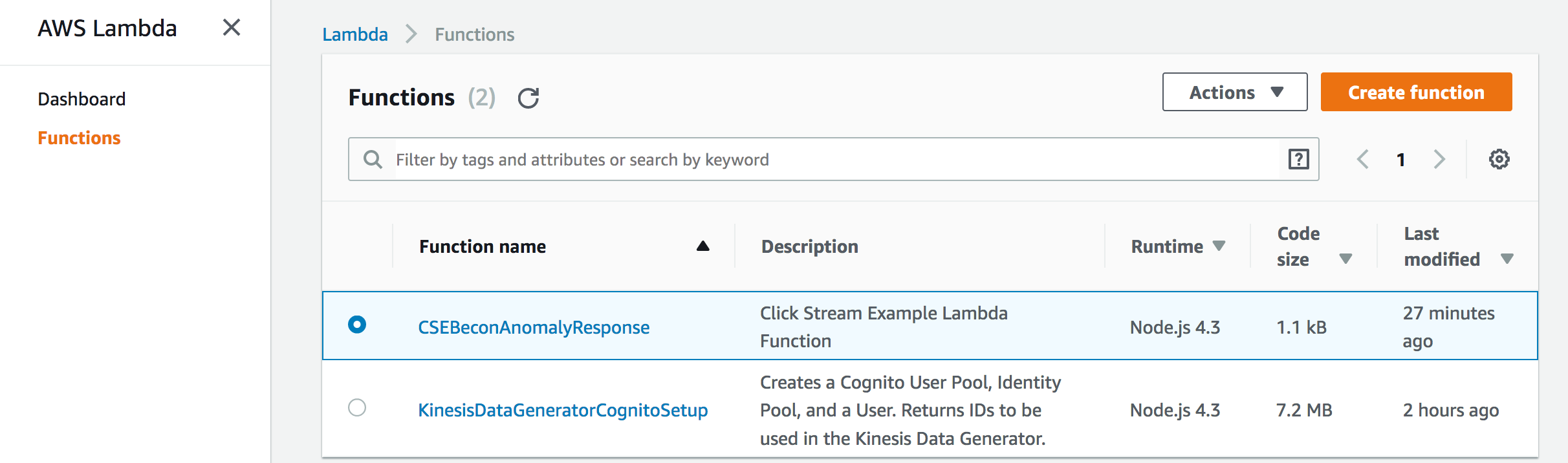
1. Click the ARN link. The Topic details screen appears listing the e-mail/SMS subscription as pending confirmation.



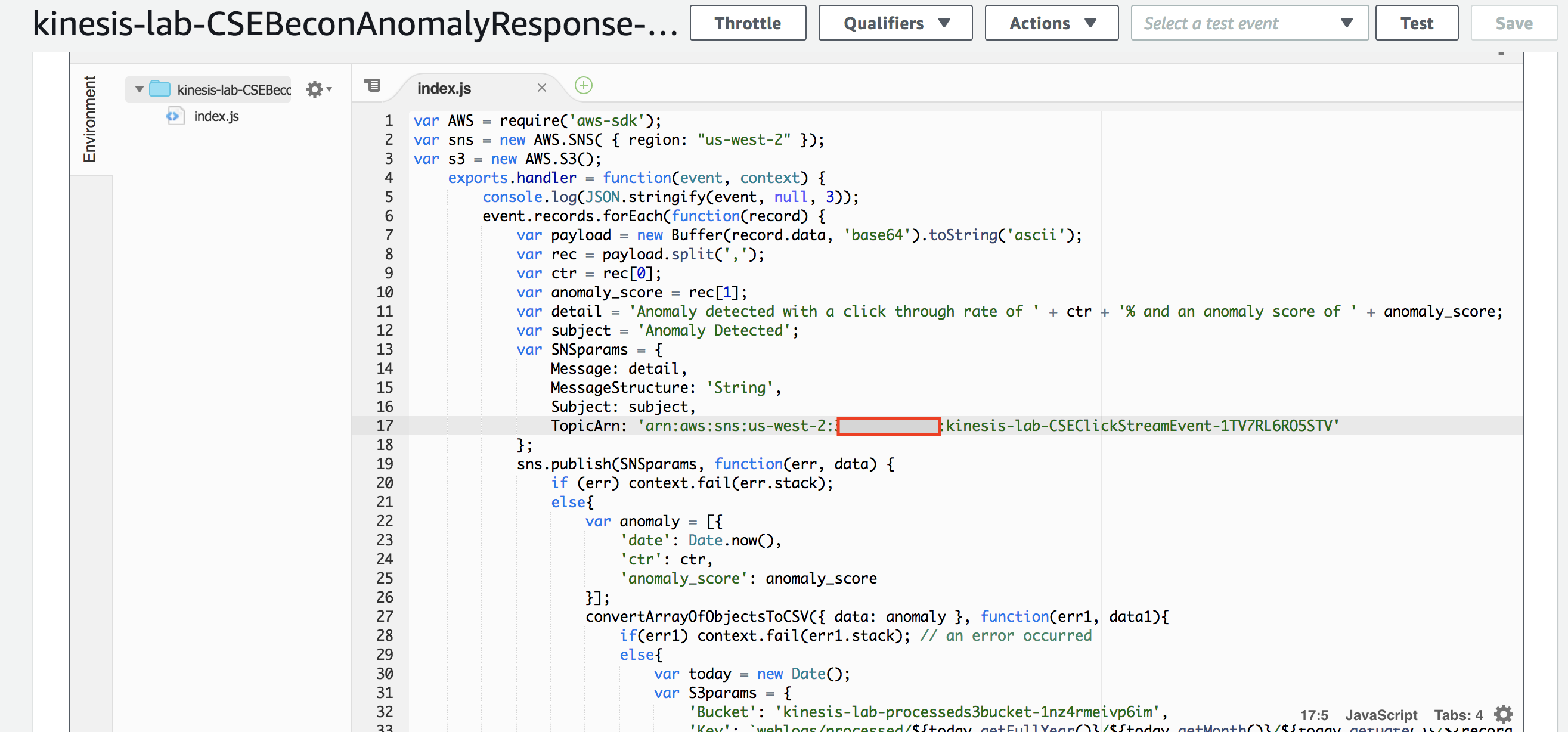
**Note:** Select corresponding subscription endpoint and Click **Request confirmations** to confirm your subscription for e-mail/SMS. Make sure to check your email junk folder for the request confirmation link.

# Observe AWS Lambda Anomaly function:

1. In the console, navigate to AWS Lambda.
2. In the AWS Lambda navigation pane, select **Functions CSEBeconAnomalyResponse** An SNS topic named ClickStreamEvent2 appears in the Functions panel.



1. Click the function hyperlink scroll down to code section.



1. Go through the code in the Lambda code editor. Notice TopicArn value your recorded in Email/SMS subscription step. Lambda will send message to this topic and notify.

You can also analyze code from **anomaly\_detection\_lambda.js** provided with your lab package.

At this point you have all the necessary components to work on the lab.

This is the end of the pre-lab, congratulations!

# Appendix: CloudFormation Template

The AWS CloudFormation template is below. This template only works in the **us-west-2** region.

Copy and paste this code into the **Kinesis\_Pre\_Lab.json** file on your machine. Select that file in CloudFormation during Step 2.

{

"AWSTemplateFormatVersion" : "2010-09-09",

"Description" : "Supporting elements for the Kinesis Analytics click stream lab",

"Parameters" : {

"email": {

"Description" : "Email address to send anomaly detection events.",

"Type": "String",

"ConstraintDescription" : "myemail@example.com"

},

"SMS": {

"Description" : "Mobile Phone number to send SMS anomaly detection events. +1XXXXXXXXXX ",

"Type": "String",

"ConstraintDescription" : "##########"

},

"Username": {

"Description": "The username of the user you want to create in Amazon Cognito.",

"Type": "String",

"AllowedPattern": "^(?=\\s\*\\S).\*$",

"ConstraintDescription": " cannot be empty"

},

"Password": {

"Description": "The password of the user you want to create in Amazon Cognito. Must be at least 6 alpha-numeric characters, and contain at least one number",

"Type": "String",

"NoEcho": true,

"AllowedPattern": "^(?=.\*[A-Za-z])(?=.\*\\d)[A-Za-z\\d]{6,}$",

"ConstraintDescription": " must be at least 6 alpha-numeric characters, and contain at least one number"

}

},

"Metadata": {

"AWS::CloudFormation::Interface": {

"ParameterGroups": [

{

"Label": {

"default": "Kinesis Pre Lab set up"

},

"Parameters": [

"Username",

"Password",

"email",

"SMS"

]

}

]

}

},

"Resources" : {

"RawS3Bucket" : {

"Type" : "AWS::S3::Bucket"

},

"ProcessedS3Bucket" : {

"Type" : "AWS::S3::Bucket"

},

"FirehoseDeliveryStream": {

"Type": "AWS::KinesisFirehose::DeliveryStream",

"Properties": {

"S3DestinationConfiguration": {

"BucketARN": { "Fn::Join" : [ "", [ "arn:aws:s3:::", { "Ref" : "RawS3Bucket" } ] ] },

"BufferingHints": {

"IntervalInSeconds": "60",

"SizeInMBs": "50"

},

"CompressionFormat": "GZIP",

"Prefix": "weblogs/raw/",

"RoleARN": { "Fn::GetAtt" : ["S3Role", "Arn"] }

}

}

},

"S3Role" : {

"Type" : "AWS::IAM::Role",

"Properties" : {

"AssumeRolePolicyDocument" : {

"Version" : "2012-10-17",

"Statement" : [{

"Effect" : "Allow",

"Principal" : { "Service" : "firehose.amazonaws.com" },

"Action" : "sts:AssumeRole",

"Condition" : { "StringEquals" : { "sts:ExternalId" : { "Ref" : "AWS::AccountId" } } }

}]

},

"Policies" : [

{

"PolicyName" : "S3Access",

"PolicyDocument" : {

"Version" : "2012-10-17",

"Statement" : [{

"Effect" : "Allow",

"Action" : [ "s3:\*" ],

"Resource" : [

{ "Fn::Join" : [ "", [ "arn:aws:s3:::", { "Ref" : "RawS3Bucket" } ] ] },

{ "Fn::Join" : [ "", [{ "Fn::Join" : [ "", [ "arn:aws:s3:::", { "Ref" : "RawS3Bucket" } ] ] }, "/\*" ] ] }

]}]

}

},

{

"PolicyName" : "CloudWatch",

"PolicyDocument" : {

"Version" : "2012-10-17",

"Statement" : [{

"Effect" : "Allow",

"Action" : [ "cloudwatch:\*",

"cloudwatchlogs:\*" ],

"Resource" : ["\*"]}]

}

}

]

}

},

"DataGenCognitoSetupLambdaFunc" : {

"Type" : "AWS::Lambda::Function",

"Properties" : {

"Code": {

"S3Bucket" : "kinesis-helpers",

"S3Key": "datagen-cognito-setup.zip"

},

"Description": "Creates a Cognito User Pool, Identity Pool, and a User. Returns IDs to be used in the Kinesis Data Generator.",

"Handler": "createCognitoPool.createPoolAndUser",

"Role": { "Fn::GetAtt" : ["CognitoLambdaExecutionRole", "Arn"] },

"Runtime": "nodejs8.10",

"Timeout": 60

}

},

"CognitoLambdaExecutionRole": {

"Type": "AWS::IAM::Role",

"Properties": {

"AssumeRolePolicyDocument": {

"Version": "2012-10-17",

"Statement": [{ "Effect": "Allow", "Principal": {"Service": ["lambda.amazonaws.com"]}, "Action": ["sts:AssumeRole"] }]

},

"Path": "/",

"Policies": [{

"PolicyName": "root",

"PolicyDocument": {

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["logs:\*"],

"Resource": "arn:aws:logs:\*:\*:\*" },

{

"Effect": "Allow",

"Action": [

"cognito-idp:AdminConfirmSignUp",

"cognito-idp:CreateUserPoolClient",

"cognito-idp:AdminCreateUser"

],

"Resource": [

"arn:aws:cognito-idp:\*:\*:userpool/\*"

]

},

{

"Effect": "Allow",

"Action": [

"cognito-idp:CreateUserPool",

"cognito-identity:\*"

],

"Resource": "\*" },

{

"Effect": "Allow",

"Action": ["iam:UpdateAssumeRolePolicy"],

"Resource": [

{"Fn::GetAtt" : ["AuthenticatedUserRole", "Arn"] },

{"Fn::GetAtt" : ["UnauthenticatedUserRole", "Arn"] }

]

},

{

"Effect": "Allow",

"Action": ["iam:PassRole"],

"Resource": [

{"Fn::GetAtt" : ["AuthenticatedUserRole", "Arn"] },

{"Fn::GetAtt" : ["UnauthenticatedUserRole", "Arn"] }

]

}

]

}

}]

}

},

"SetupCognitoCustom" : {

"Type": "Custom::DataGenCognitoSetupLambdaFunc",

"Properties": {

"ServiceToken": { "Fn::GetAtt" : ["DataGenCognitoSetupLambdaFunc", "Arn"] },

"Region": {"Ref": "AWS::Region"},

"Username": {"Ref": "Username"},

"Password": {"Ref": "Password"},

"AuthRoleName": {"Ref": "AuthenticatedUserRole"},

"AuthRoleArn": { "Fn::GetAtt" : ["AuthenticatedUserRole", "Arn"] },

"UnauthRoleName": {"Ref": "UnauthenticatedUserRole"},

"UnauthRoleArn": { "Fn::GetAtt" : ["UnauthenticatedUserRole", "Arn"] }

}

},

"AuthenticatedUserRole": {

"Type": "AWS::IAM::Role",

"Properties": {

"AssumeRolePolicyDocument": {

"Version": "2012-10-17",

"Statement": [{ "Effect": "Allow", "Principal": {"Federated": ["cognito-identity.amazonaws.com"]}, "Action": ["sts:AssumeRoleWithWebIdentity"] }]

},

"Path": "/",

"Policies": [{

"PolicyName": "root",

"PolicyDocument": {

"Version": "2012-10-17",

"Statement": [

{

"Action": [

"kinesis:DescribeStream",

"kinesis:PutRecord",

"kinesis:PutRecords"

],

"Resource": [

"arn:aws:kinesis:\*:\*:stream/\*"

],

"Effect": "Allow"

},

{

"Action": [

"firehose:DescribeDeliveryStream",

"firehose:PutRecord",

"firehose:PutRecordBatch"

],

"Resource": [

"arn:aws:firehose:\*:\*:deliverystream/\*"

],

"Effect": "Allow"

},

{

"Action": [

"mobileanalytics:PutEvents",

"cognito-sync:\*",

"cognito-identity:\*",

"ec2:DescribeRegions",

"firehose:ListDeliveryStreams",

"kinesis:ListStreams"

],

"Resource": [

"\*"

],

"Effect": "Allow"

}

]

}

}]

}

},

"UnauthenticatedUserRole": {

"Type": "AWS::IAM::Role",

"Properties": {

"AssumeRolePolicyDocument": {

"Version": "2012-10-17",

"Statement": [{ "Effect": "Allow", "Principal": {"Federated": ["cognito-identity.amazonaws.com"]}, "Action": ["sts:AssumeRoleWithWebIdentity"] }]

},

"Path": "/",

"Policies": [{

"PolicyName": "root",

"PolicyDocument": {

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"mobileanalytics:PutEvents",

"cognito-sync:\*"

],

"Resource": [

"\*"

]

}

]

}

}]

}

},

"CSELambdaSNSPublishRole" : {

"Type": "AWS::IAM::Role",

"Properties": {

"AssumeRolePolicyDocument": {

"Version" : "2012-10-17",

"Statement": [ {

"Effect": "Allow",

"Principal": {

"Service": [ "lambda.amazonaws.com" ]

},

"Action": [ "sts:AssumeRole" ]

} ]

},

"Policies": [ {

"PolicyName": "lambda\_sns",

"PolicyDocument": {

"Version": "2012-10-17",

"Statement": [ {

"Effect": "Allow",

"Action": [

"sns:Publish"

],

"Resource": { "Ref" : "CSEClickStreamEvent" }

} ]

}

} ]

}

},

"CSEClickStreamEvent" : {

"Type" : "AWS::SNS::Topic",

"Properties" : {

"Subscription" : [

{ "Endpoint" : { "Ref" : "email"}, "Protocol" : "email" },

{ "Endpoint" : { "Ref" : "SMS"}, "Protocol" : "sms" }

]

}

},

"LambdaExecutionRole" : {

"Type": "AWS::IAM::Role",

"Properties": {

"AssumeRolePolicyDocument": {

"Version" : "2012-10-17",

"Statement": [ {

"Effect": "Allow",

"Principal": {

"Service": [ "lambda.amazonaws.com" ]

},

"Action": [ "sts:AssumeRole" ]

} ]

},

"Policies": [ {

"PolicyName": "CSELambdaExecutionRole",

"PolicyDocument": {

"Version": "2012-10-17",

"Statement": [{

"Sid" : "publish2sns",

"Effect": "Allow",

"Action": [

"sns:Publish"

],

"Resource": [

{ "Ref" : "CSEClickStreamEvent" }

]

},

{

"Sid" : "writelogs",

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents"

],

"Resource": "arn:aws:logs:\*:\*:\*"

},

{

"Sid" : "readkinesis",

"Effect": "Allow",

"Action": [

"kinesis:DescribeStream",

"kinesis:GetRecords",

"kinesis:GetShardIterator",

"kinesis:ListStreams"

],

"Resource": "\*"

},

{

"Action": [

"s3:PutObject"

],

"Resource": { "Fn::Join" : [ "", [{ "Fn::Join" : [ "", [ "arn:aws:s3:::", { "Ref" : "ProcessedS3Bucket" } ] ] }, "/\*" ] ] },

"Effect": "Allow",

"Sid": "writeToS3"

}]

}

}]

}

},

"CSEBeconAnomalyResponse" : {

"Type" : "AWS::Lambda::Function",

"Properties" : {

"Code" : {

"ZipFile": { "Fn::Join": ["", [

"var AWS = require('aws-sdk');\n",

"var sns = new AWS.SNS( { region: \"",

{ "Ref" : "AWS::Region" },

"\" });\n",

"var s3 = new AWS.S3();\n",

"\texports.handler = function(event, context) {\n",

"\t\tconsole.log(JSON.stringify(event, null, 3));\n",

"\t\tevent.records.forEach(function(record) {\n",

"\t\t\tvar payload = new Buffer(record.data, 'base64').toString('ascii');\n",

"\t\t\tvar rec = payload.split(',');\n",

"\t\t\tvar ctr = rec[0];\n",

"\t\t\tvar anomaly\_score = rec[1];\n",

"\t\t\tvar detail = 'Anomaly detected with a click through rate of ' + ctr + '% and an anomaly score of ' + anomaly\_score;\n",

"\t\t\tvar subject = 'Anomaly Detected';\n",

"\t\t\tvar SNSparams = {\n",

"\t\t\t\tMessage: detail,\n",

"\t\t\t\tMessageStructure: 'String',\n",

"\t\t\t\tSubject: subject,\n",

"\t\t\t\tTopicArn: '",

{ "Ref" : "CSEClickStreamEvent" },

"'\n",

"\t\t\t};\n",

"\t\t\tsns.publish(SNSparams, function(err, data) {\n",

"\t\t\t\tif (err) context.fail(err.stack);\n",

"\t\t\t\telse{\n",

"\t\t\t\t\tvar anomaly = [{\n",

"\t\t\t\t\t\t'date': Date.now(),\n",

"\t\t\t\t\t\t'ctr': ctr,\n",

"\t\t\t\t\t\t'anomaly\_score': anomaly\_score\n",

"\t\t\t\t\t}];\n",

"\t\t\t\t\tconvertArrayOfObjectsToCSV({ data: anomaly }, function(err1, data1){\n",

"\t\t\t\t\t\tif(err1) context.fail(err1.stack); // an error occurred\n",

"\t\t\t\t\t\telse{\n",

"\t\t\t\t\t\t\tvar today = new Date();\n",

"\t\t\t\t\t\t\tvar S3params = {\n",

"\t\t\t\t\t\t\t\t'Bucket': '",

{ "Ref" : "ProcessedS3Bucket" },

"',\n",

"\t\t\t\t\t\t\t\t'Key': `weblogs/processed/${today.getFullYear()}/${today.getMonth()}/${today.getDate()}/${record.recordId}.csv`,\n",

"\t\t\t\t\t\t\t\t'Body': data1,\n",

"\t\t\t\t\t\t\t\t'ContentType': 'text/csv'\n",

"\t\t\t\t\t\t\t}\n",

"\t\t\t\t\t\t\ts3.putObject(S3params, function(err2, data2) {\n",

"\t\t\t\t\t\t\t\tif (err2) context.fail(err2.stack); // an error occurred\n",

"\t\t\t\t\t\t\t\telse {\n",

"\t\t\t\t\t\t\t\t\tcontext.succeed('Published Notification'); // successful response\n",

"\t\t\t\t\t\t\t\t}\n",

"\t\t\t\t\t\t\t})\n",

"\t\t\t\t\t\t}\n",

"\t\t\t\t\t})\n",

"\t\t\t\t}\n",

"\t\t\t});\n",

"\t\t});\n",

"\t};\n",

"function convertArrayOfObjectsToCSV(args, callback) {\n",

"\tvar result, ctr, keys, columnDelimiter, lineDelimiter, data;\n",

"\t\tdata = args.data || null;\n",

"\t\tif (data == null || !data.length) {\n",

"\t\t\tcallback(new Error('data is null'));\n",

"\t\t}",

"\tcolumnDelimiter = args.columnDelimiter || ',';\n",

"\tlineDelimiter = args.lineDelimiter || '\\n';\n",

"\tkeys = Object.keys(data[0]);\n",

"\tresult = '';\n",

"\tresult += keys.join(columnDelimiter);\n",

"\tresult += lineDelimiter;\n",

"\tdata.forEach(function(item) {\n",

"\t\tctr = 0;\n",

"\t\tkeys.forEach(function(key) {\n",

"\t\t\tif (ctr > 0) result += columnDelimiter;\n",

"\t\t\t\tresult += item[key];\n",

"\t\t\t\t\tctr++;\n",

"\t\t\t\t});\n",

"\t\t\tresult += lineDelimiter;\n",

"\t\t});\n",

"\t\tcallback(null, result);\n",

"\t}\n"

]]}

},

"Description" : "Click Stream Example Lambda Function",

"Handler" : "index.handler",

"MemorySize" : 128,

"Role" : { "Fn::GetAtt" : [ "LambdaExecutionRole", "Arn" ] } ,

"Runtime" : "nodejs12.x",

"Timeout" : 5

}

},

"CSEKinesisAnalyticsRole" : {

"Type": "AWS::IAM::Role",

"Properties": {

"AssumeRolePolicyDocument": {

"Version" : "2012-10-17",

"Statement": [ {

"Effect": "Allow",

"Principal": {

"Service": [ "kinesisanalytics.amazonaws.com" ]

},

"Action": [ "sts:AssumeRole" ]

} ]

},

"Policies": [{

"PolicyName": "firehose",

"PolicyDocument": {

"Version": "2012-10-17",

"Statement": [ {

"Sid": "ReadInputFirehose",

"Effect": "Allow",

"Action": [

"firehose:DescribeDeliveryStream",

"firehose:Get\*"

],

"Resource": [

{ "Fn::GetAtt" : [ "FirehoseDeliveryStream", "Arn" ] }

]

},

{

"Sid": "UseLambdaFunction",

"Effect": "Allow",

"Action": [

"lambda:InvokeFunction",

"lambda:GetFunctionConfiguration"

],

"Resource": [

{ "Fn::Join" : [ "", [ { "Fn::GetAtt" : [ "CSEBeconAnomalyResponse", "Arn" ]}, ":$LATEST" ]] }

]

}

]

}

} ]

}

}

},

"Outputs" :{

"KinesisDataGeneratorUrl": {

"Description": "The URL for your Kinesis Data Generator.",

"Value": {

"Fn::Join": ["", ["https://awslabs.github.io/amazon-kinesis-data-generator/web/producer.html?", { "Fn::GetAtt": [ "SetupCognitoCustom", "Querystring" ] }]]

}

},

"RawBucketName" : {

    "Description" : "This the bucket name of where your Raw data will be store at" ,

    "Value" : {

"Ref" : "RawS3Bucket"

}

},

"ProcessedBucketName" : {

    "Description" : "This the bucket name of where your Processed data will be store at" ,

    "Value" : {

"Ref" : "ProcessedS3Bucket"

}

}

}

}