# CMPE 272: Enterprise SW Plat

Model Quality Monitoring Assignment

**Group 10**

Siddhant Satish Parmar

Roger Kuo

Anirudh Satyam

Kumar Raja Pavuluri

1. **Explain how AWS Identity and Access Management (IAM) controls permissions using Roles, Policies, and Groups. What is the difference between a Role and a Policy?**

   AWS access permissions are applied via Policies. Policies are objects that define permissions for whatever they are associated with. Policies can be applied to identities, such as Roles and Groups. The permissions in the Policies determine whether a request from an identity is allowed or denied. To grant access to services and resources, specific policies can be attached to identities and resources.

   A Policy, as previously mentioned, is just a set of permissions. A Role is an identity that can be applied to multiple, unspecified users and Groups.

2. **Explain the function of the two DynamoDB tables we created above?**

   The first table “LabScores” is used to store input features and predicted classes. It is set up to notify the system, via a Lambda function, of incoming records and updates in the table. The second table “LabRetraining” is used for receiving and storing retraining records from the Lambda function after it passes all the tests written in the Lambda function. The two tables are set up for models to log results of credit scoring.

3. **Outline the logic in the Lambda function. When is it called? What test does it apply against new items? Where does it write its results?**

   The Lambda function has a function “createItems” which takes parameters “params” which is a JS object in which item details from the LabScores table is stored. Then we implement a for loop in which all the event records are traversed, and if the eventName is “INSERT” (i.e. if any item is inserted in the LabScores table then we will fetch that entry and store the item details in the “params” object.

   Once the item details from the LabScores are fetched, we perform some tests on the new items, tests like:

- Checking if the “Class” is 0 then the “gap” will be equal to the “Probability” else the “gap” is equal to 1 - “Probability”.
- After setting the “gap”, we check if the “gap” > 0.25 then the “createItems” function will be called and we write the new items in the LabRetraining table.

Since we created a Trigger inside the LabScores table, the “LabLambdaFn” will be triggered whenever there is some change in the LabScores table, but it will write results in the LabRetraining table only when the event is “INSERT”.

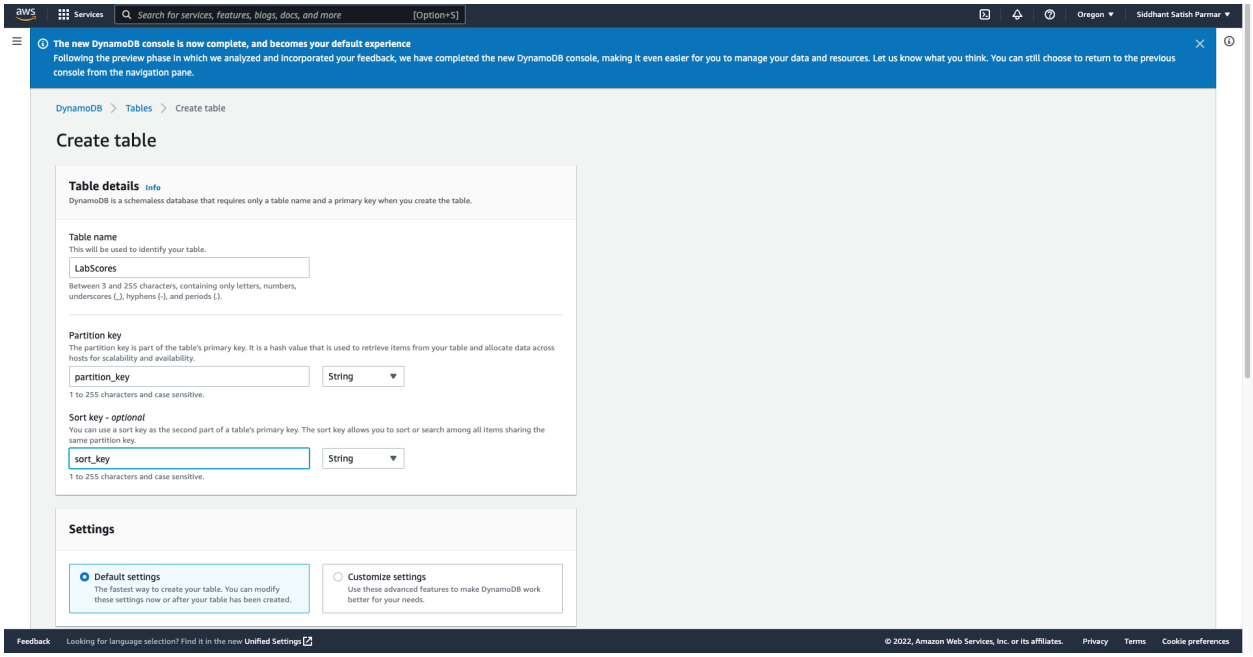
## 4. What is the role of CloudWatch in our AWS back-end?

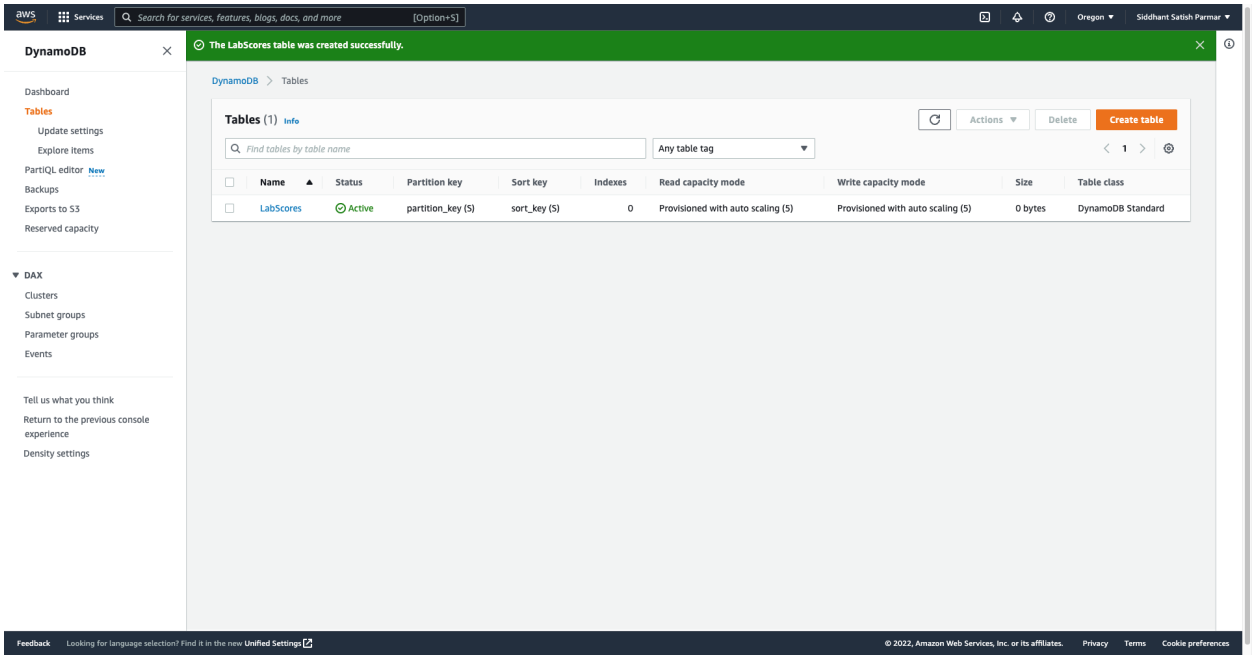
Cloudwatch helps monitor and manage our applications and resources. It helps with storing metrics in a repository, and statistics of our data can be retrieved from the metrics. It can be used to graph our statistical data. Cloudwatch also provides insights on optimizing performance and managing resource utilization. It provides a unified dashboard to help look over our operations. In our AWS backend it shows us eventID, "eventName (which in our case would be insert since the item in Labretraining is created when eventName == INSERT), eventVersion, eventSource, awsRegion, dynamodb (which shows us ApproximateCreationDateTime, Keys, NewImage, SequenceNumber, SizeBytes, StreamViewType(which would be NEW_IMAGE in our case)

## 5. Screenshots for A) Steps to create tables and Lambda function. B) List of items in Scoring table and Retraining table.

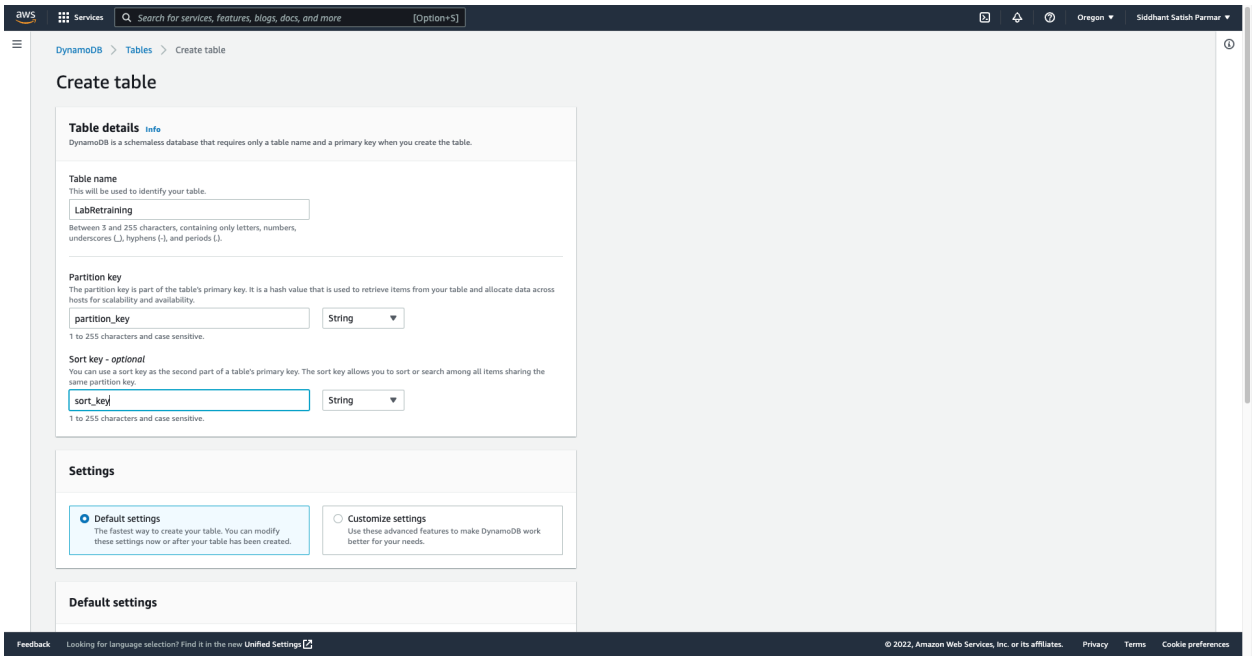
### A) Steps to create tables and Lambda function:-

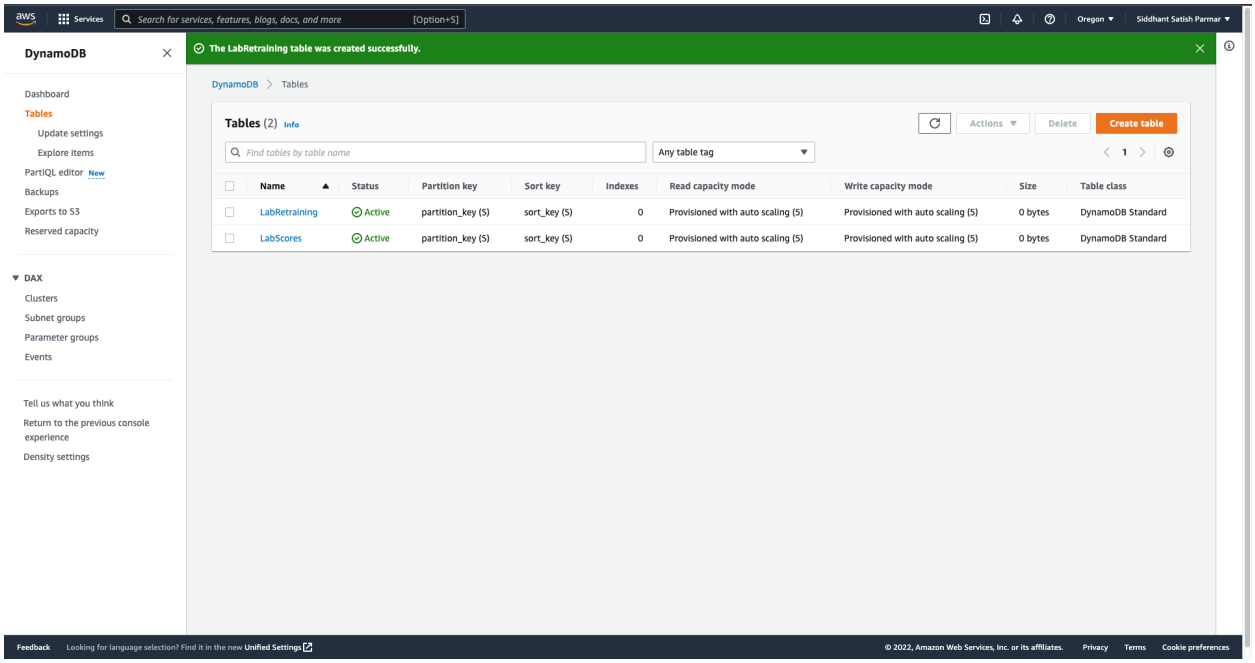
- **Setup a DynamoDB table to receive credit scoring logs.**



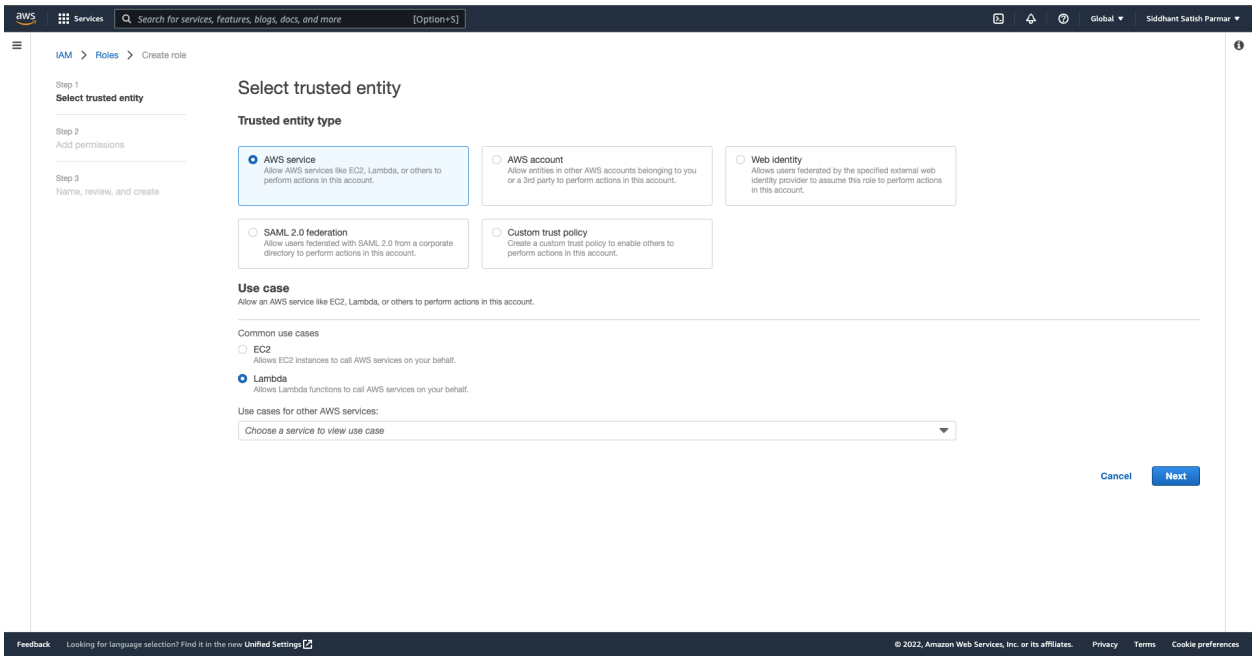



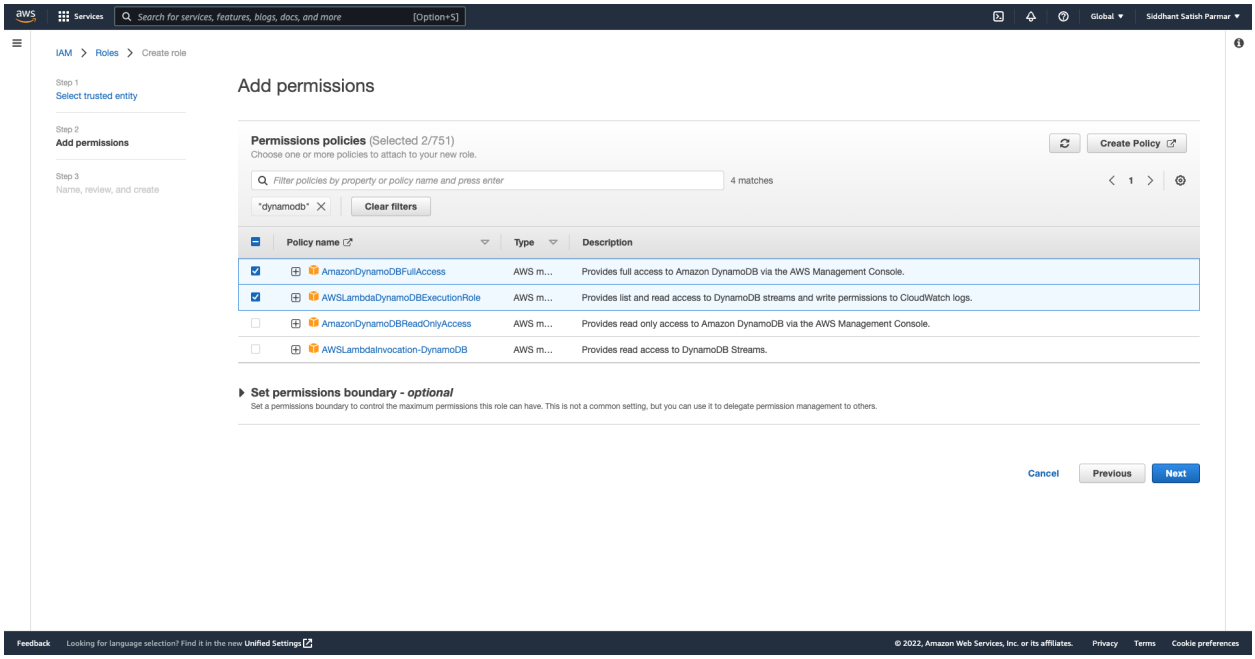
- **Setup a second DynamoDB table to receive retraining records.**

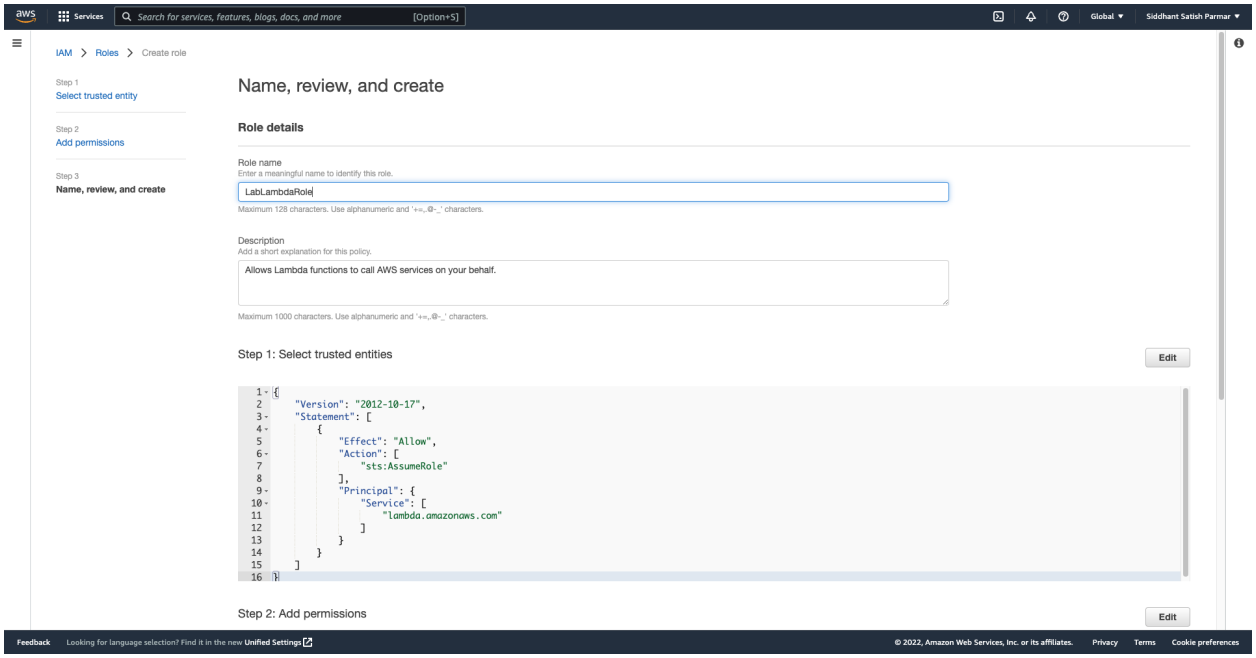


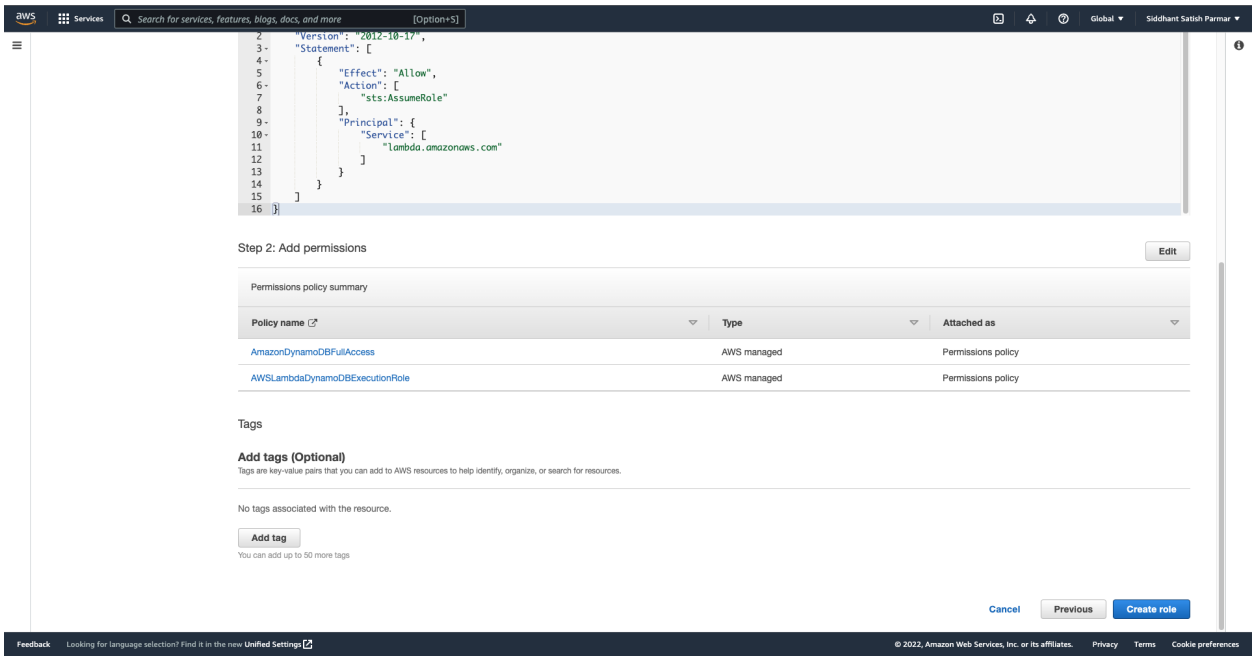



- **Creating the required AWS role and policies.**

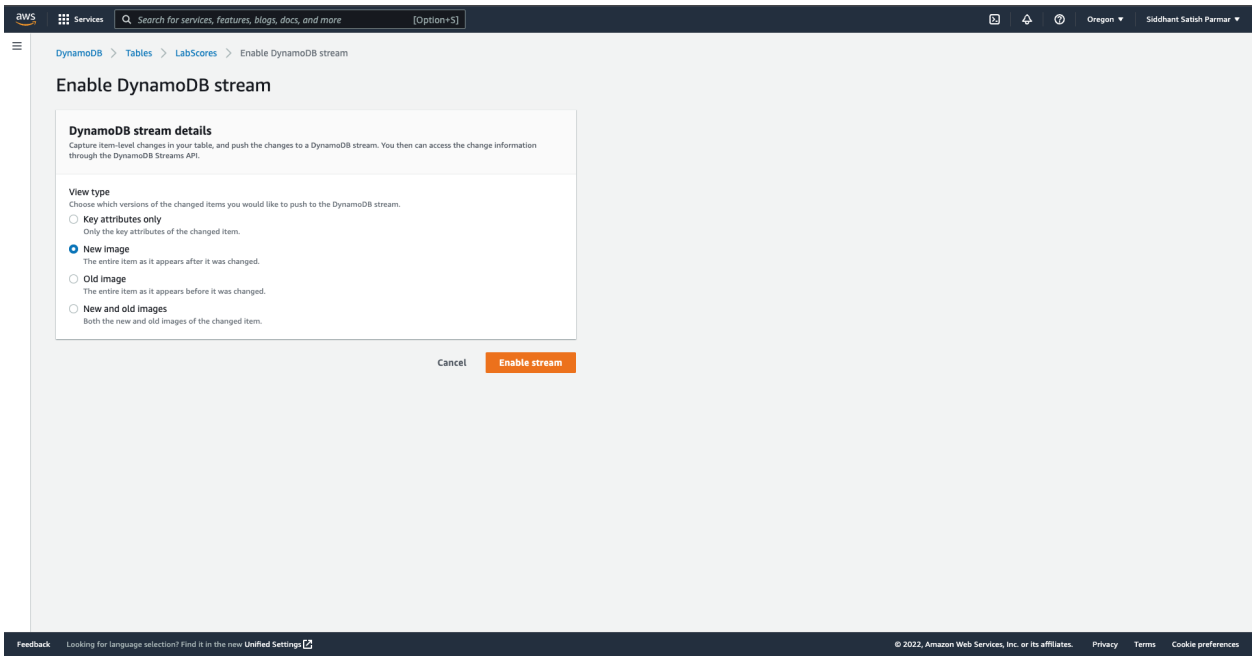


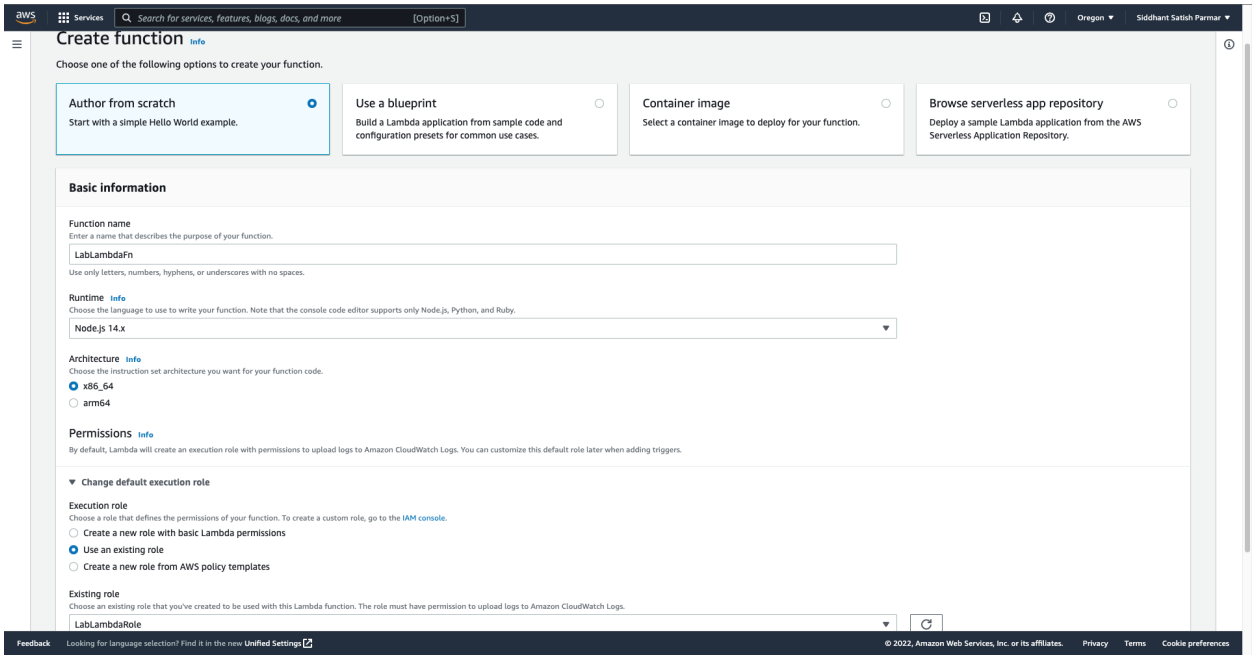



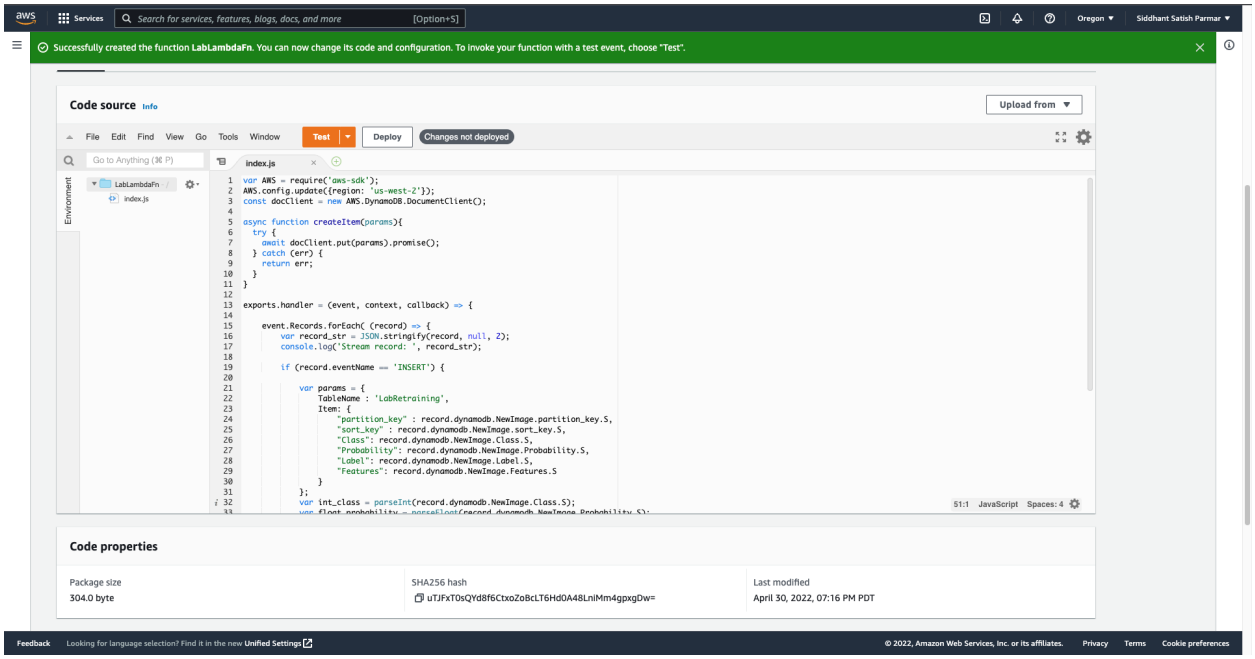

Services
Search for services, features, blogs, docs, and more [Option+S]
Global
Siddhant Satish Parmar
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
LabLambdaRole
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this policy.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Step 1: Select trusted entities
Edit
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"sts:AssumeRole"
],
"Principal": {
"Service": [
"lambda.amazonaws.com"
]
}
}
]
}
Step 2: Add permissions
Edit
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

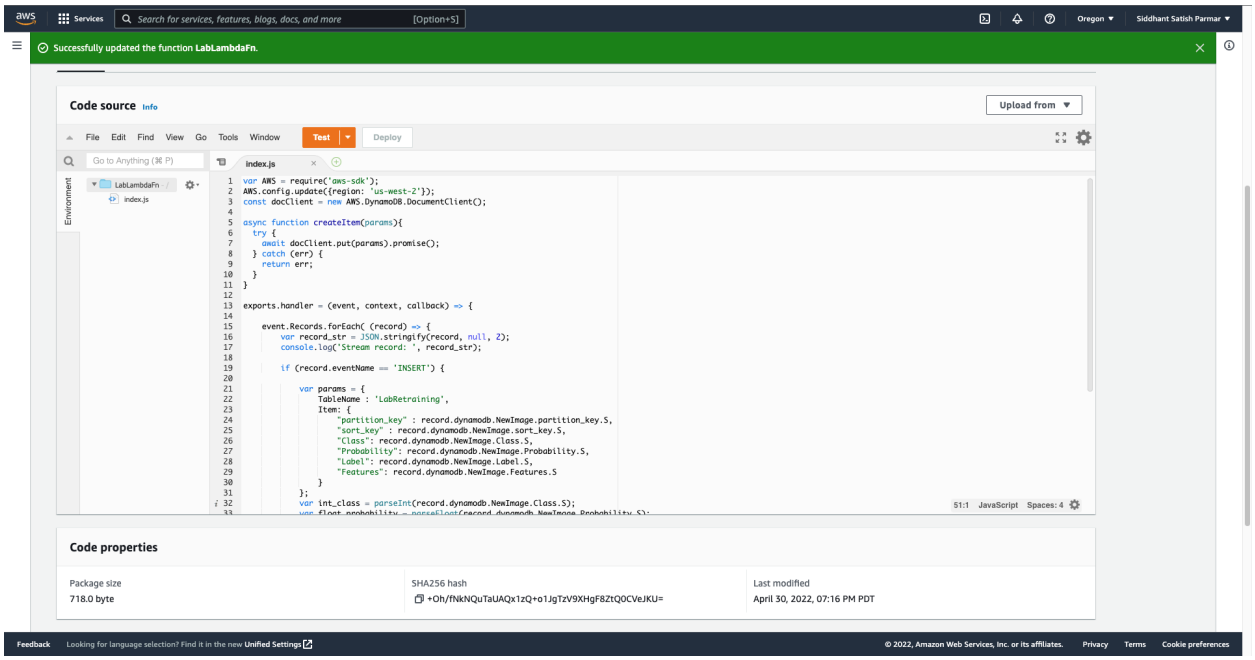

Services
Search for services, features, blogs, docs, and more [Option+S]
Global
Siddhant Satish Parmar
Step 2: Add permissions
Edit
Permissions policy summary
Policy name
Type
Attached as
AmazonDynamoDBFullAccess
AWS managed
Permissions policy
AWSLambdaDynamoDBExecutionRole
AWS managed
Permissions policy
Tags
Add tags (Optional)
Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.
No tags associated with the resource.
Add tag
You can add up to 50 more tags
Cancel
Previous
Create role
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

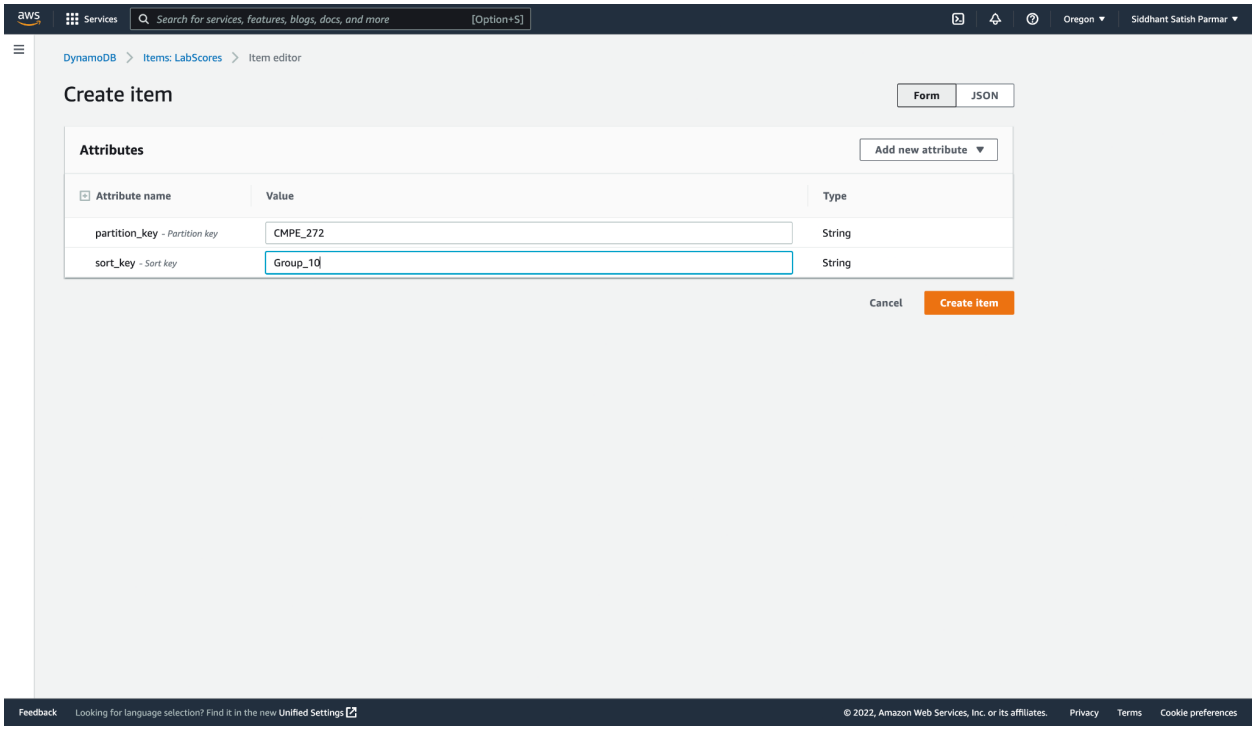
- **Configure and deploy your Lambda function to monitor DynamoDB and forward retraining records.**

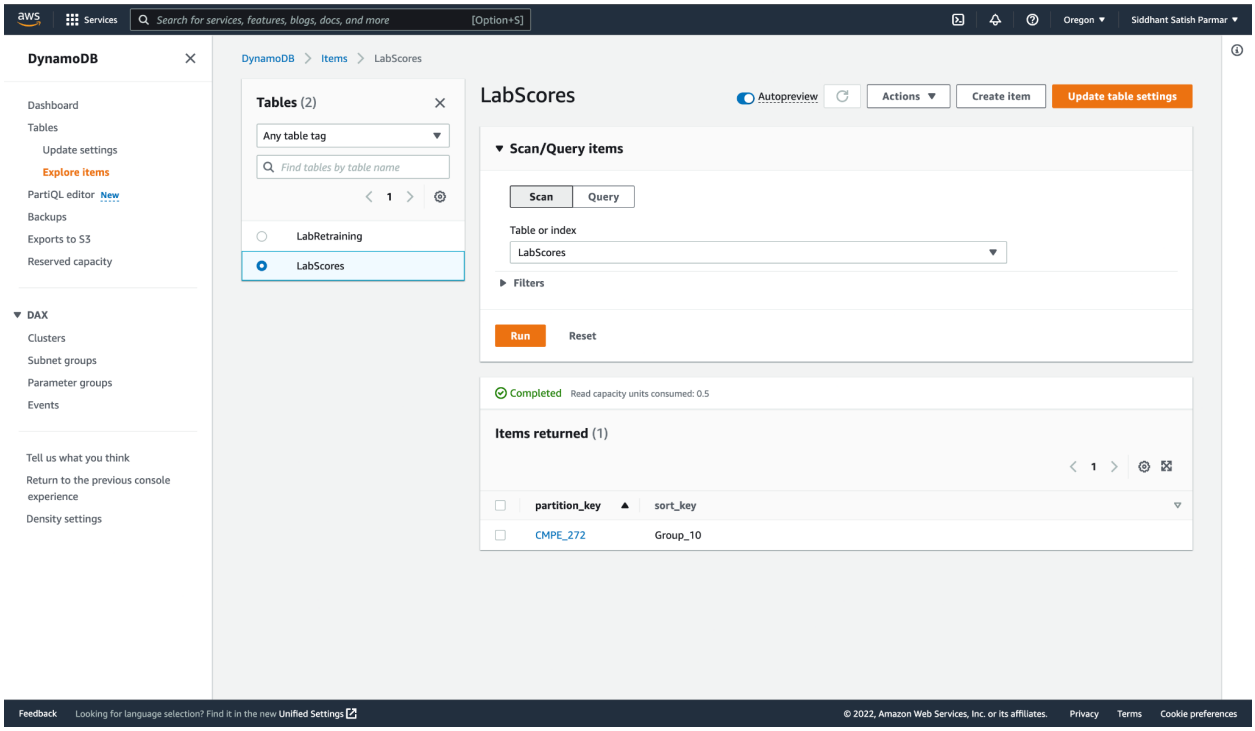


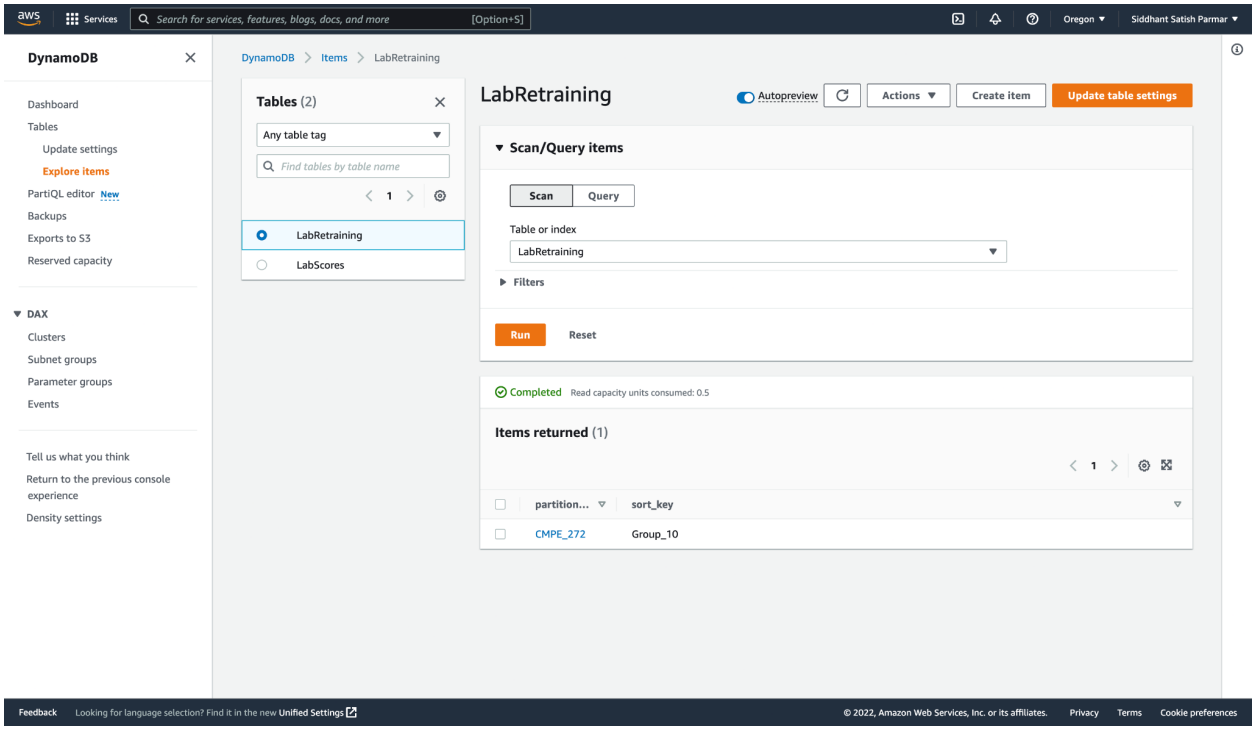



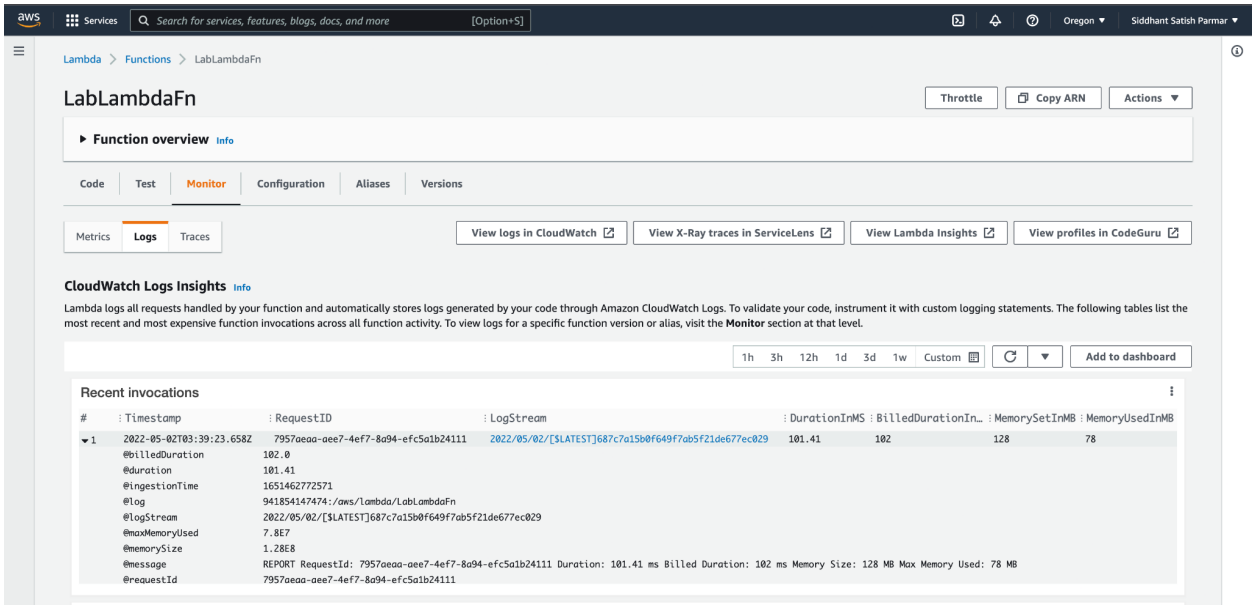

Services
Search for services, features, blogs, docs, and more [Option+S]
Oregon
Siddhant Satish Parmar
Successfully created the function LabLambdaFn. You can now change its code and configuration. To invoke your function with a test event, choose "Test".
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Changes not deployed
Go to Anything (⌘ P)
index.js
Environment
LabLambdaFn
index.js
var AWS = require('aws-sdk');
AWS.config.update({region: 'us-west-2'});
const docClient = new AWS.DynamoDB.DocumentClient();
async function createItem(params){
try {
await docClient.put(params).promise();
} catch (err) {
return err;
}
}
exports.handler = (event, context, callback) => {
event.Records.forEach( (record) => {
var record_str = JSON.stringify(record, null, 2);
console.log('Stream record: ', record_str);
if (record.eventName == 'INSERT') {
var params = {
TableName : 'LabRetraining',
Item: {
"partition_key" : record.dynamodb.NewImage.partition_key.S,
"sort_key" : record.dynamodb.NewImage.sort_key.S,
"Class": record.dynamodb.NewImage.Class.S,
"Probability": record.dynamodb.NewImage.Probability.S,
"Label": record.dynamodb.NewImage.Label.S,
"Features": record.dynamodb.NewImage.Features.S
}
};
var int_class = parseInt(record.dynamodb.NewImage.Class.S);
51:1 JavaScript Spaces: 4
Code properties
Package size
304.0 byte
SHA256 hash
uTJFxTOsQYd8f6CtxoZaBcLT6Hd0A48LnIMm4gpxgDw=
Last modified
April 30, 2022, 07:16 PM PDT
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

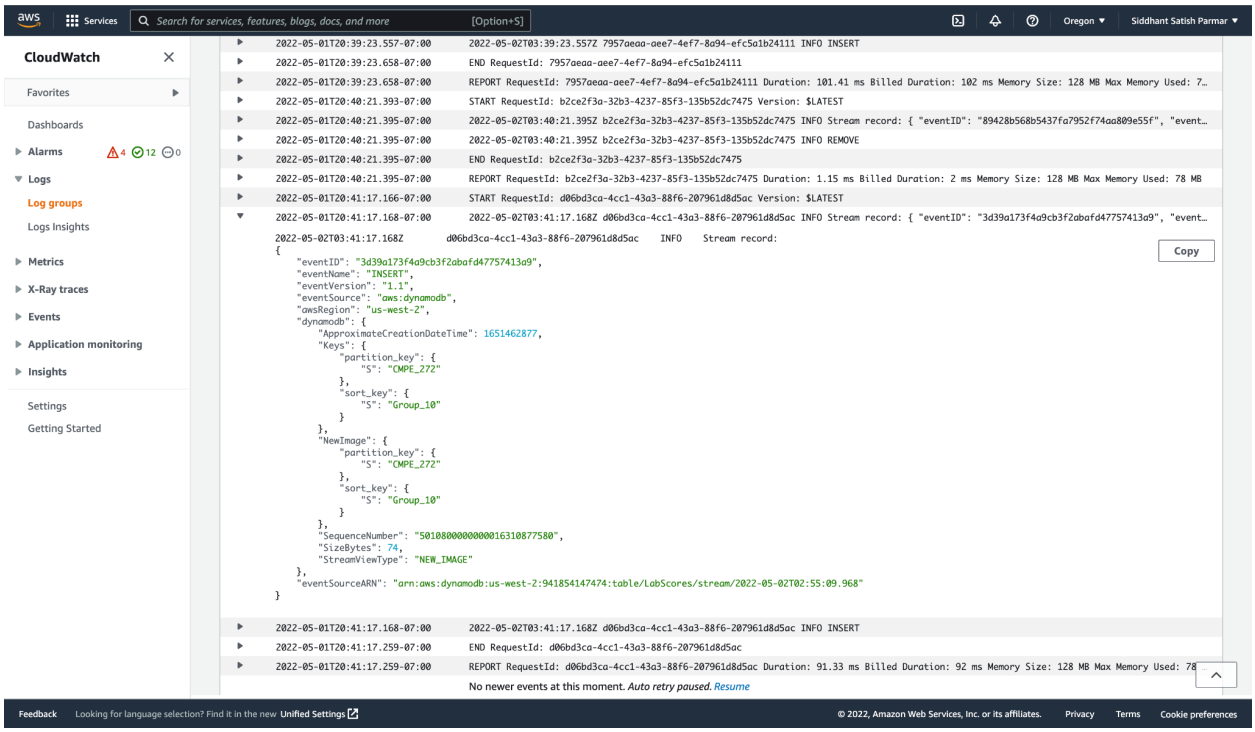

Services
Search for services, features, blogs, docs, and more [Option+S]
Oregon
Siddhant Satish Parmar
Successfully updated the function LabLambdaFn.
Code source Info
Upload from
File Edit Find View Go Tools Window
Test
Deploy
Go to Anything (⌘ P)
index.js
Environment
LabLambdaFn
index.js
var AWS = require('aws-sdk');
AWS.config.update({region: 'us-west-2'});
const docClient = new AWS.DynamoDB.DocumentClient();
async function createItem(params){
try {
await docClient.put(params).promise();
} catch (err) {
return err;
}
}
exports.handler = (event, context, callback) => {
event.Records.forEach( (record) => {
var record_str = JSON.stringify(record, null, 2);
console.log('Stream record: ', record_str);
if (record.eventName == 'INSERT') {
var params = {
TableName : 'LabRetraining',
Item: {
"partition_key" : record.dynamodb.NewImage.partition_key.S,
"sort_key" : record.dynamodb.NewImage.sort_key.S,
"Class": record.dynamodb.NewImage.Class.S,
"Probability": record.dynamodb.NewImage.Probability.S,
"Label": record.dynamodb.NewImage.Label.S,
"Features": record.dynamodb.NewImage.Features.S
}
};
var int_class = parseInt(record.dynamodb.NewImage.Class.S);
51:1 JavaScript Spaces: 4
Code properties
Package size
718.0 byte
SHA256 hash
+Oh/fNkNQuTaUAQx1zQ+o1JgTzV9XHgF8ZtQOCVeJKU=
Last modified
April 30, 2022, 07:16 PM PDT
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search for services, features, blogs, docs, and more [Option+S]
Oregon
Siddhant Satish Parmar
DynamoDB > Items: LabScores > Item editor
Create item
Form
JSON
Attributes
Add new attribute
Attribute name
Value
Type
partition_key - Partition key
CMPE_272
String
sort_key - Sort key
Group_10
String
Cancel
Create item
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Services
Search for services, features, blogs, docs, and more [Option+S]
Oregon
Siddhant Satish Parmar
DynamoDB
Dashboard
Tables
Update settings
Explore items
PartiQL editor New
Backups
Exports to S3
Reserved capacity
DAX
Clusters
Subnet groups
Parameter groups
Events
Tell us what you think
Return to the previous console experience
Density settings
DynamoDB > Items > LabScores
Tables (2)
Any table tag
Find tables by table name
LabRetraining
LabScores
LabScores
Autopreview
Actions
Create item
Update table settings
Scan/Query items
Scan
Query
Table or index
LabScores
Filters
Run
Reset
Completed
Read capacity units consumed: 0.5
Items returned (1)
partition_key
sort_key
CMPE_272
Group_10
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

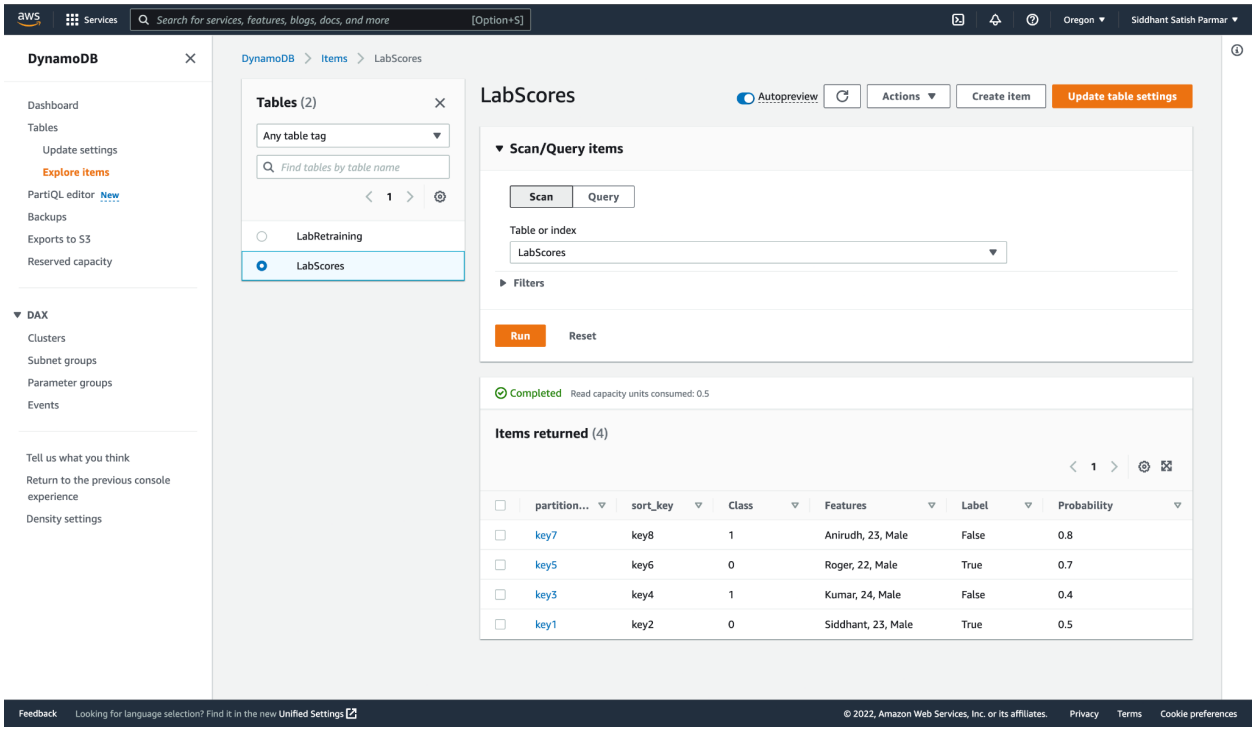

aws
Services
Search for services, features, blogs, docs, and more
[Option+S]
Oregon
Siddhant Satish Parmar
DynamoDB
Dashboard
Tables
Update settings
Explore items
PartiQL editor New
Backups
Exports to S3
Reserved capacity
DAX
Clusters
Subnet groups
Parameter groups
Events
Tell us what you think
Return to the previous console experience
Density settings
DynamoDB > Items > LabRetraining
Tables (2)
Any table tag
Find tables by table name
1
LabRetraining
LabScores
LabRetraining
Autopreview
Actions
Create item
Update table settings
Scan/Query items
Scan
Query
Table or index
LabRetraining
Filters
Run
Reset
Completed
Read capacity units consumed: 0.5
Items returned (1)
1
partition...
sort_key
CMPE_272
Group_10
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


aws
Services
Search for services, features, blogs, docs, and more
[Option+S]
Oregon
Siddhant Satish Parmar
Lambda > Functions > LabLambdaFn
LabLambdaFn
Throttle
Copy ARN
Actions
Function overview Info
Code
Test
Monitor
Configuration
Aliases
Versions
Metrics
Logs
Traces
View logs in CloudWatch
View X-Ray traces in ServiceLens
View Lambda Insights
View profiles in CodeGuru
CloudWatch Logs Insights Info
Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the Monitor section at that level.
1h
3h
12h
1d
3d
1w
Custom
Add to dashboard
Recent invocations
# Timestamp RequestID LogStream DurationInMS BilledDurationIn... MemorySetInMB MemoryUsedInMB
1 2022-05-02T03:39:23.658Z 7957aeaa-aee7-4ef7-8a94-efc5a1b24111 2022/05/02/[$LATEST]687c7a15b0f649f7ab5f21de677ec029 101.41 102 128 78
@billedDuration 102.0
@duration 101.41
@ingestionTime 1651462772571
@log 941854147474:/aws/lambda/LabLambdaFn
@logStream 2022/05/02/[$LATEST]687c7a15b0f649f7ab5f21de677ec029
@maxMemoryUsed 7.8E7
@memorySize 1.28E8
@message REPORT RequestId: 7957aeaa-aee7-4ef7-8a94-efc5a1b24111 Duration: 101.41 ms Billed Duration: 102 ms Memory Size: 128 MB Max Memory Used: 78 MB
@requestId 7957aeaa-aee7-4ef7-8a94-efc5a1b24111

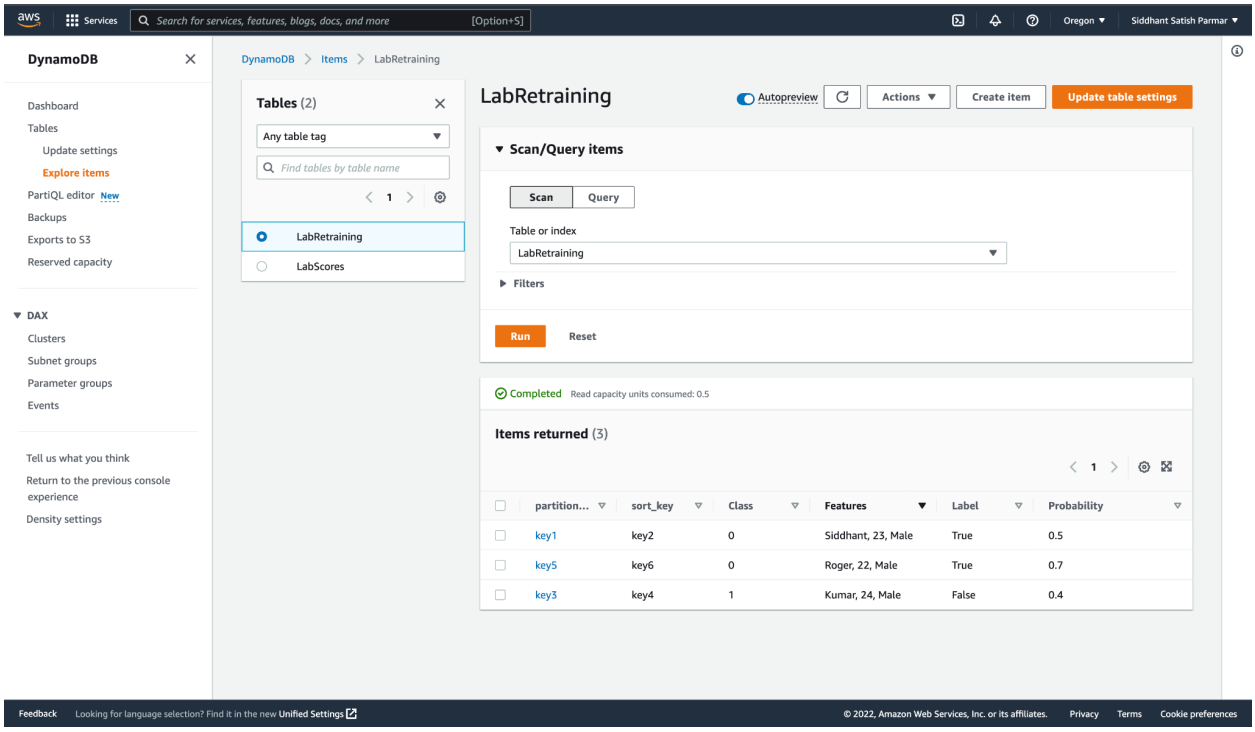



## B) List of items in Scoring table and Retraining table:-

- **List of items in LabScores table**



- **List of items in LabRetraining table**




6. **If you made any extensions to the logic of the Lambda function for selecting retraining samples, explain your extensions and show the code.**

The changes we made to the Lambda function was that we extracted the age of the person from the “Features” and stored it in a variable named “age”. Now while writing records to the LabRetraining table we will check if the person is above the age of 21, then only the records will be written to the LabRetraining table.

**Code for the LabLambdaFn is shown below:**

index.js

```
var AWS = require('aws-sdk');
AWS.config.update({region: 'us-west-2'});
const docClient = new AWS.DynamoDB.DocumentClient();

async function createItem(params){
  try {
    await docClient.put(params).promise();
  } catch (err) {
    return err;
  }
}

exports.handler = (event, context, callback) => {

    event.Records.forEach( (record) => {
        var record_str = JSON.stringify(record, null, 2);
        console.log('Stream record: ', record_str);

        if (record.eventName == 'INSERT') {

            var params = {
                TableName : 'LabRetraining',
                Item: {
                    "partition_key" : record.dynamodb.NewImage.partition_key.S,
                    "sort_key" : record.dynamodb.NewImage.sort_key.S,
                    "Class": record.dynamodb.NewImage.Class.S,
                    "Probability": record.dynamodb.NewImage.Probability.S,
                    "Label": record.dynamodb.NewImage.Label.S,
                    "Features": record.dynamodb.NewImage.Features.S
                }
            };
            var int_class = parseInt(record.dynamodb.NewImage.Class.S);
            var float_probability = parseFloat(record.dynamodb.NewImage.Probability.S);
            var features =  record.dynamodb.NewImage.Features.S;
            var feats = features.split(", ");
            var age = parseInt(feats[1]);
            var gap = 0;

            if ( int_class == 0 )
            {
                gap = float_probability;
            }
            else
            {
                gap = 1 - float_probability;
            }
            if ( gap > 0.25 && age > 21)
            {
               createItem(params)
            }
        }
    }),

    callback(null, `Successfully processed ${event.Records.length} records.`);
}

```