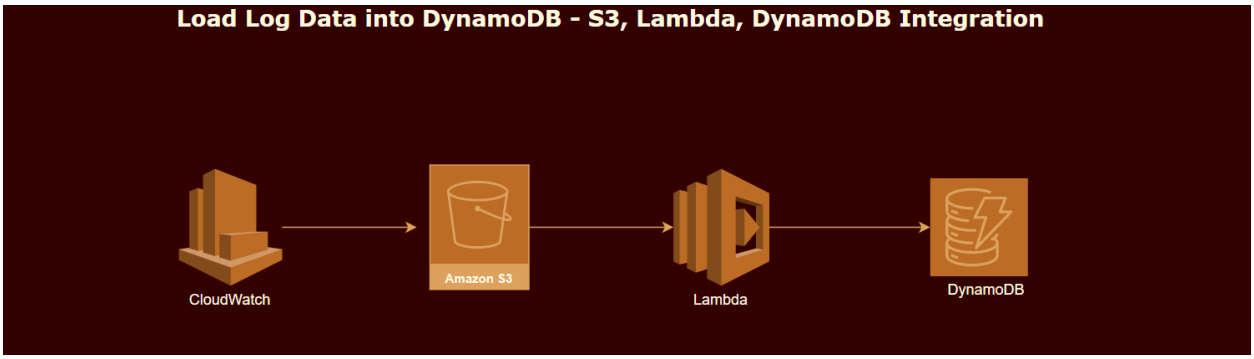
Step-by-step process to load the log data into DynamoDB from AWS S3 – AWS Serverless Architecture

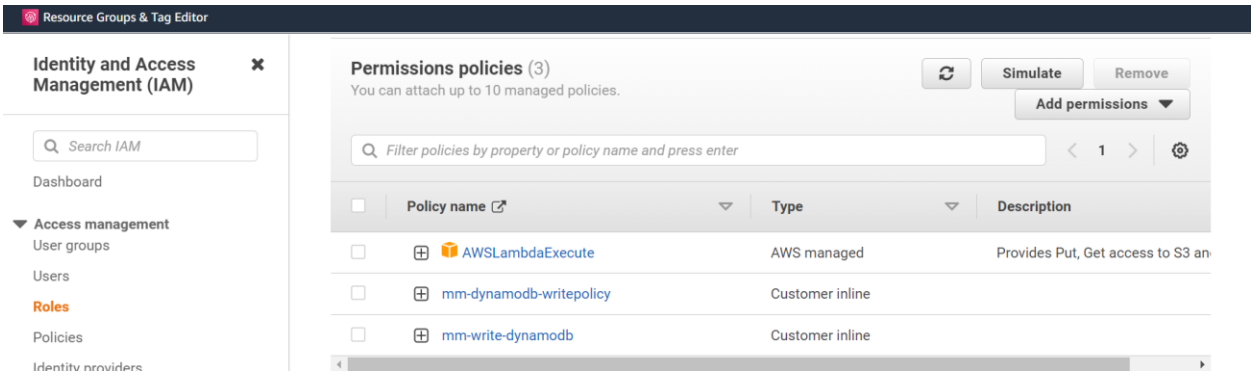
Architecture Diagram:

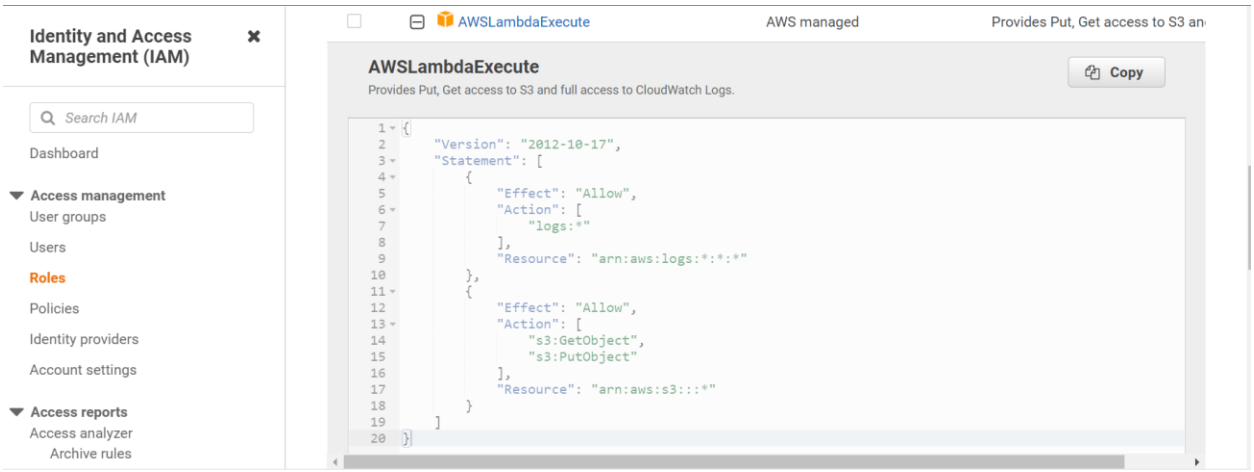


Steps to implement the solution:

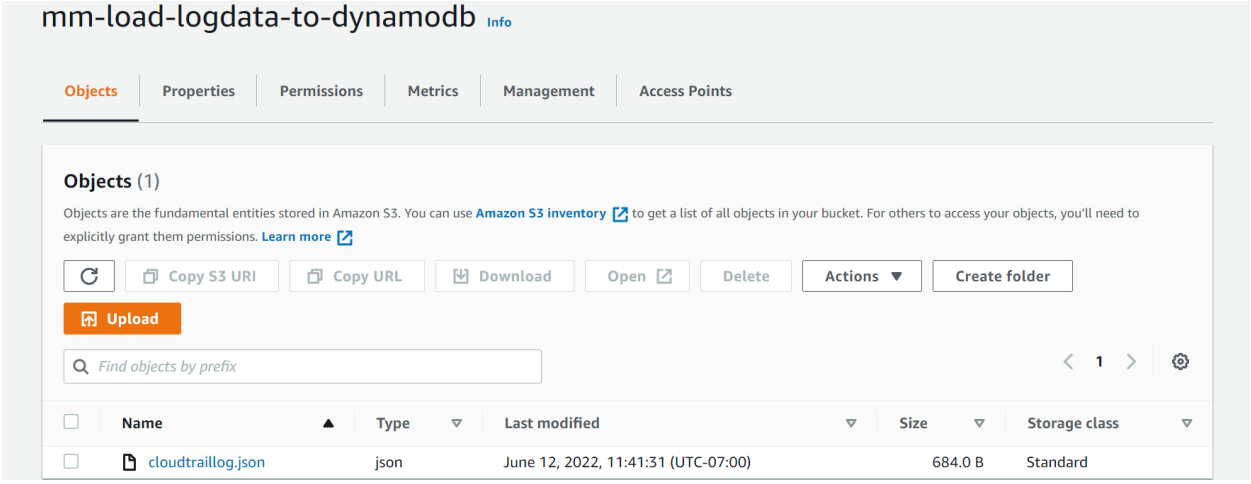
1. Create the appropriate role
2. Create the S3 bucket
3. Log files (Jason format)
4. Create the Lambda function with S3 trigger
5. Create the Dynamodb table
6. Create the in-line policy with the DynamoDB ARN
7. Load data/ files into S3
8. Test the flow

Create the appropriate role (mm-lambda-s3):





Create the S3 bucket (mmload-logdata-to-dynamodb):



Sample Log files (Json format):

[

{

"id": "AIDAIS3D5PJFEDMLRVSFG1",

"type": "IAMUser",

"eventTime": "2022-06-11T23:54:55Z",

"eventSource": "ec2.amazonaws.com",

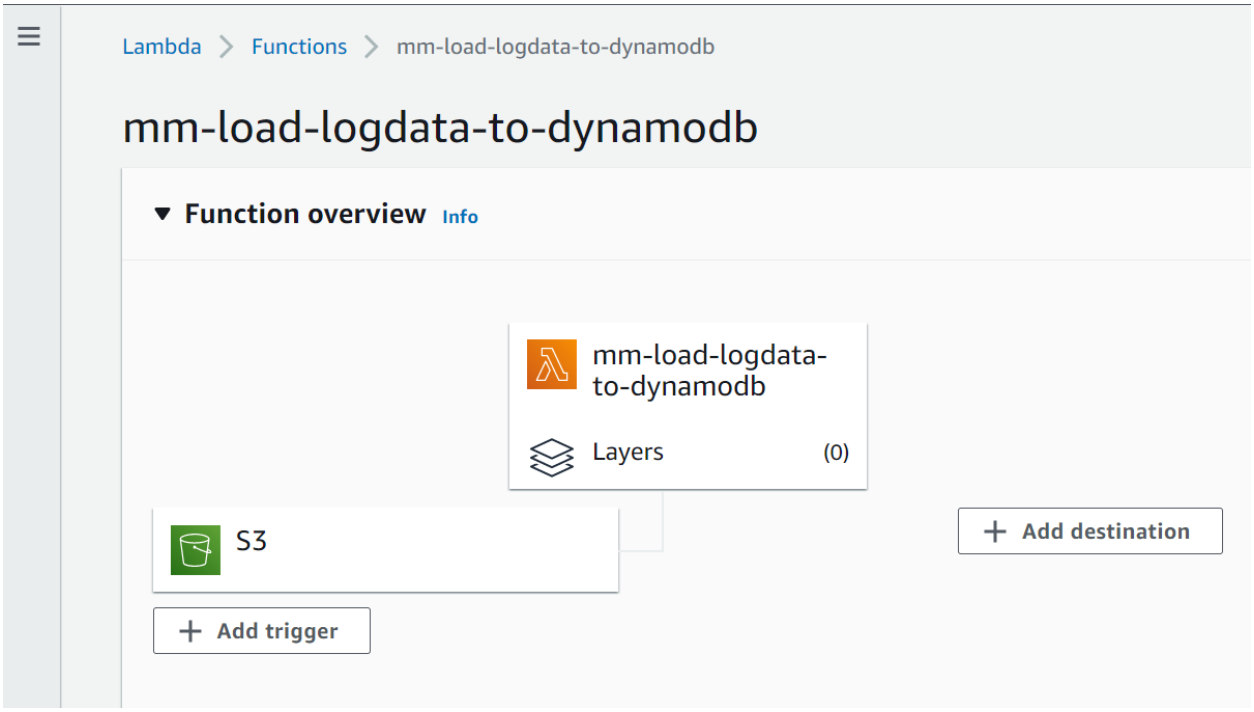
"eventName": "DescribeInstances",

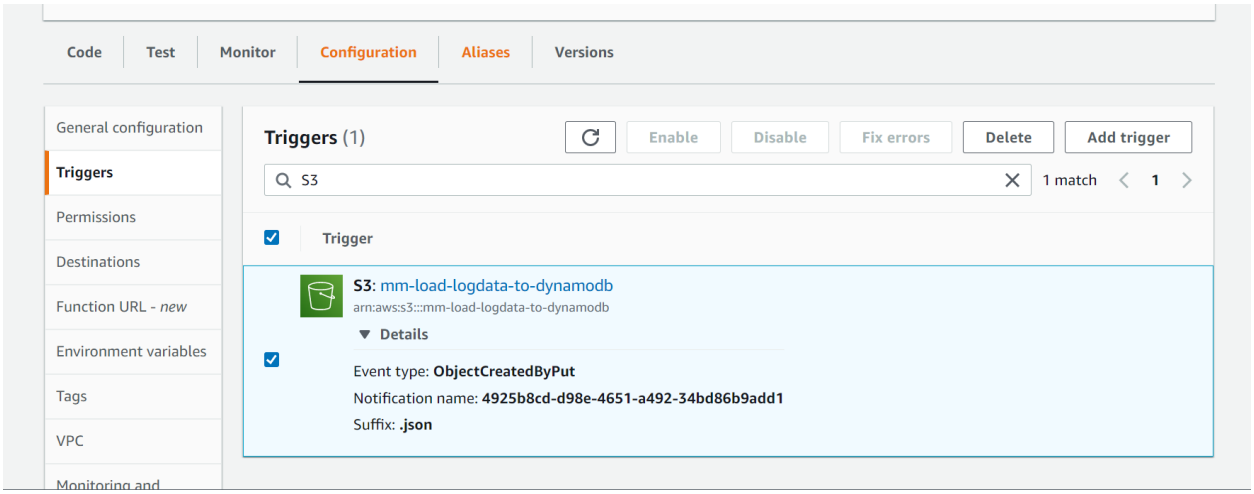
"awsRegion": "us-west-1",

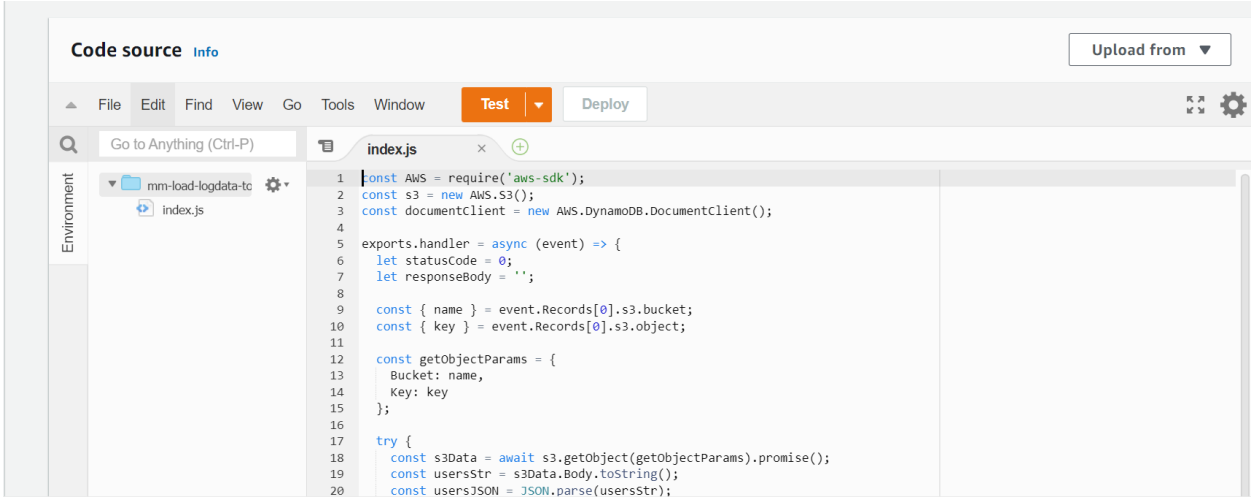
"requestID": "db14c600-332d-4ef9-b68a-bbe4749110bd",

```
    "eventID": "4284e768-20c0-4b02-b2a0-af08da75ccfa"
  },
  {
    "id": "AIDAIS3D5PJFEDMLRVSFG2",
    "type": "IAMUser",
    "eventTime": "2022-06-11T23:55:32Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "DescribeSecurityGroups",
    "awsRegion": "us-west-1",
    "requestID": "b862c4da-e427-4a24-8faf-6e748a4dddf0",
    "eventID": "9282b6a6-14a8-4e79-b09b-5ecb7580dc5b"
  }
]
```


Create the Lambda function with S3 trigger:



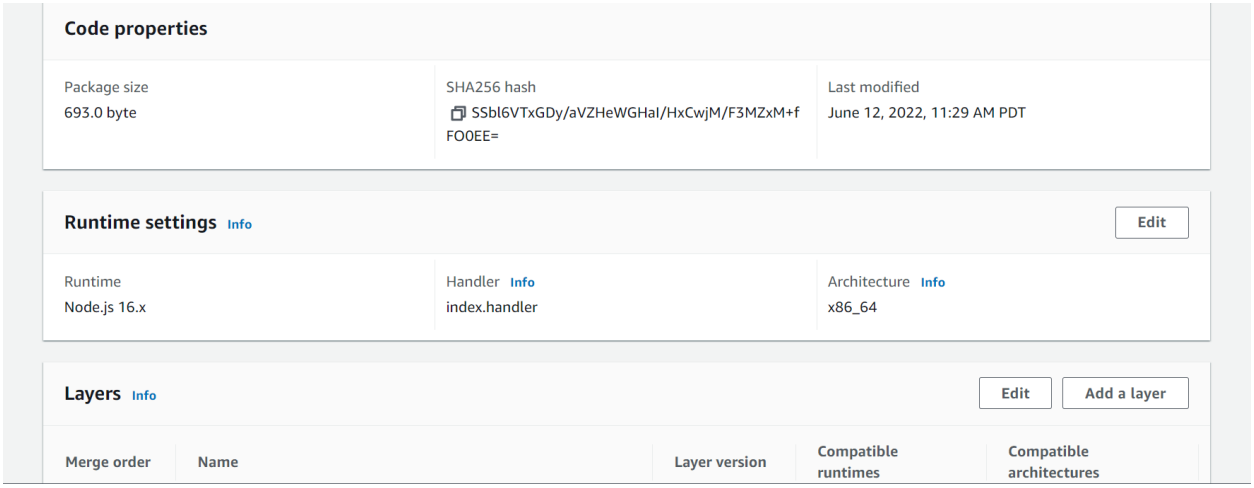




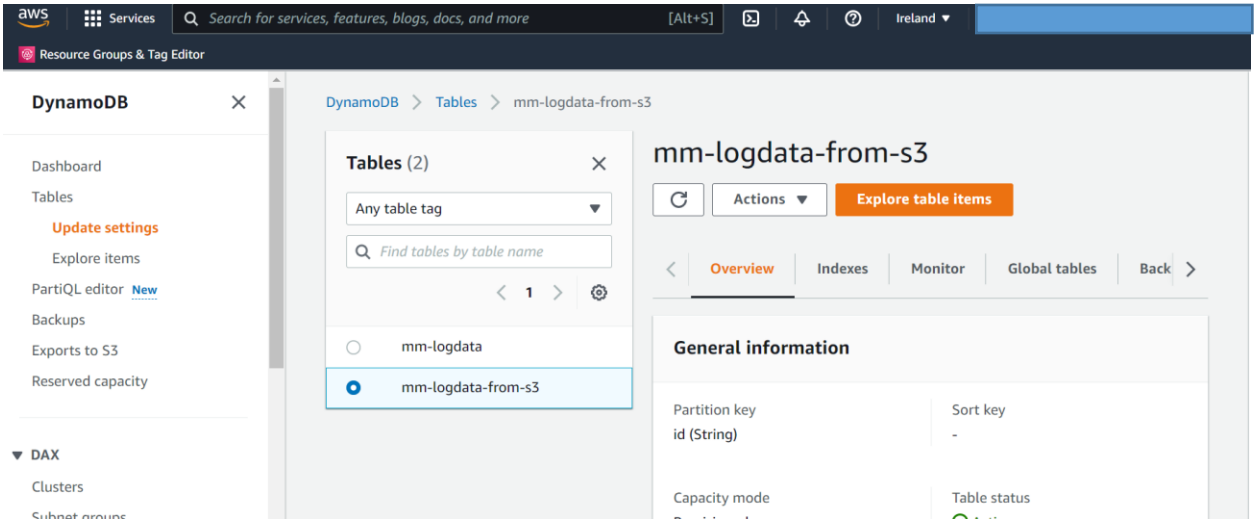

```
const AWS = require('aws-sdk');
const s3 = new AWS.S3();
const documentClient = new AWS.DynamoDB.DocumentClient();

exports.handler = async (event) => {
  let statusCode = 0;
  let responseBody = '';

  const { name } = event.Records[0].s3.bucket;
  const { key } = event.Records[0].s3.object;

  const getObjectParams = {
```

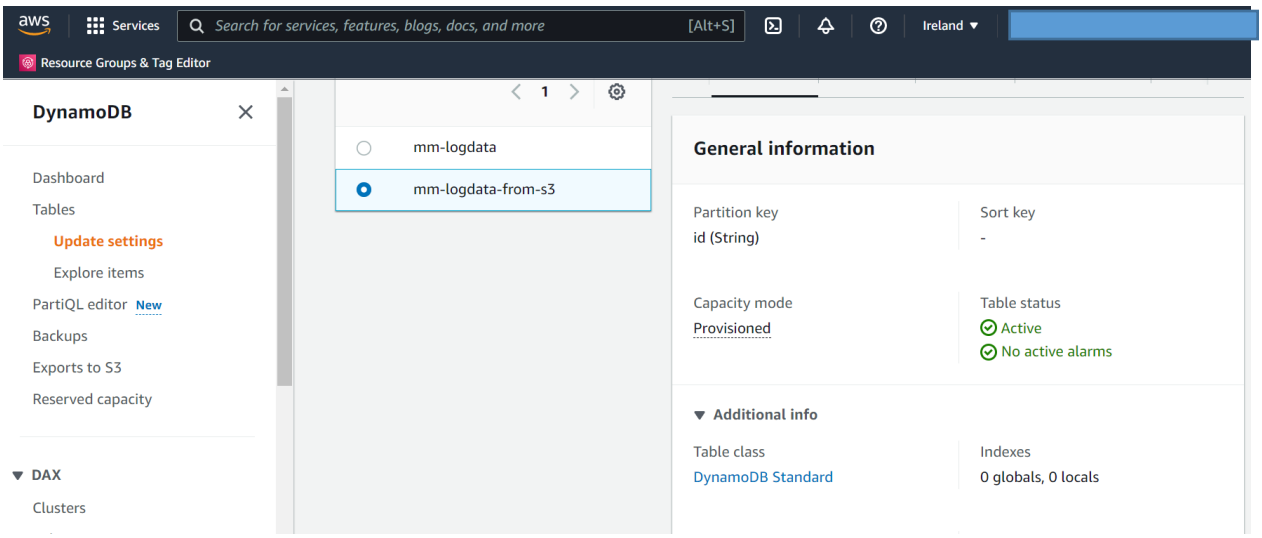
```
  Bucket: name,
  Key: key
};

try {
  const s3Data = await s3.getObject(getObjectParams).promise();
  const usersStr = s3Data.Body.toString();
  const usersJSON = JSON.parse(usersStr);
  console.log(`Users ::: ${usersStr}`);

  await Promise.all(usersJSON.map(async user => {
    const { id, type, eventTime, eventSource, eventName, awsRegion, requestID, eventID } = user;

    const putParams = {
      TableName: "mm-logdata-from-s3",
      Item: {
        id: id,
        type: type,
        eventTime: eventTime,
        eventSource: eventSource,
        eventName: eventName,
        awsRegion: awsRegion,
        requestID: requestID,
        eventID: eventID
      }
    };

    await documentClient.put(putParams).promise();
```

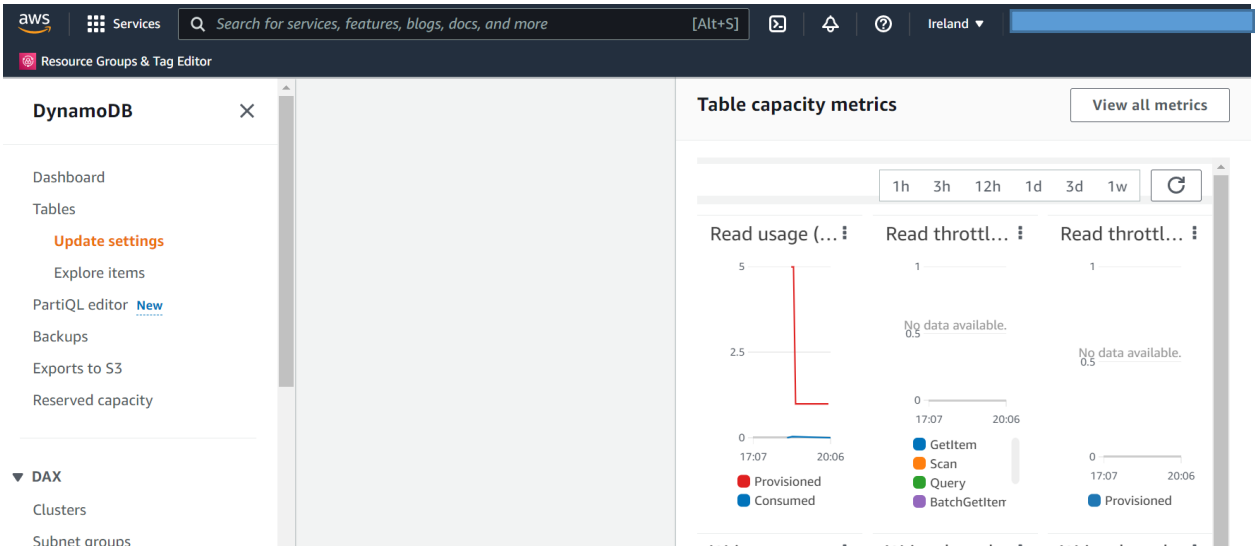
```
    }));

    responseBody = 'Succeeded adding users';
    statusCode = 201;

  } catch(err) {
      responseBody = 'Error adding users';
      statusCode = 403;
  }

  const response = {
    statusCode: statusCode,
    body: responseBody
  };

  return response;
};
```



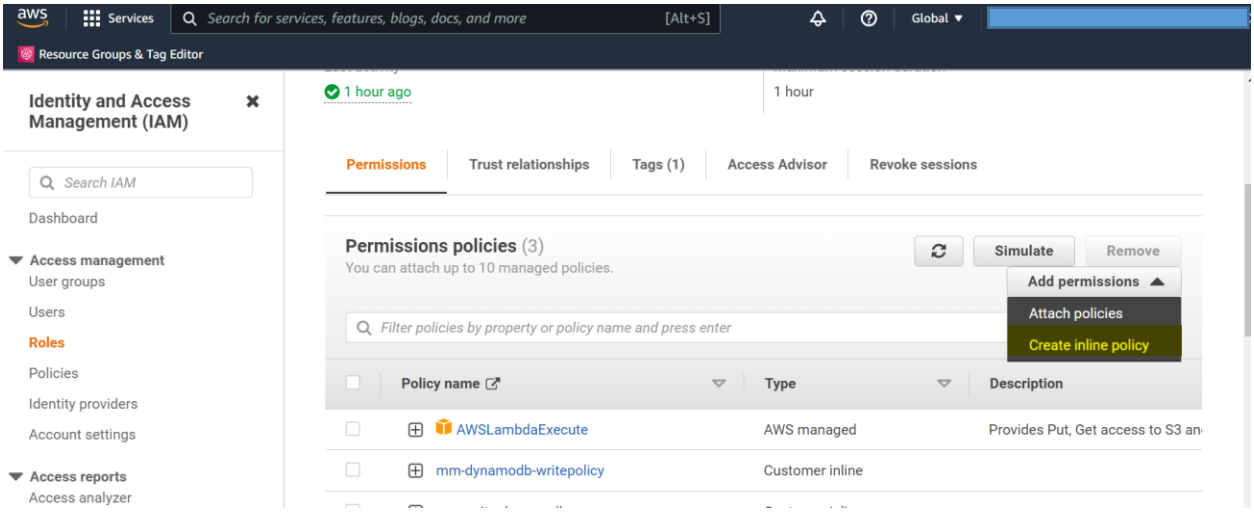
Create the Dynamodb table:

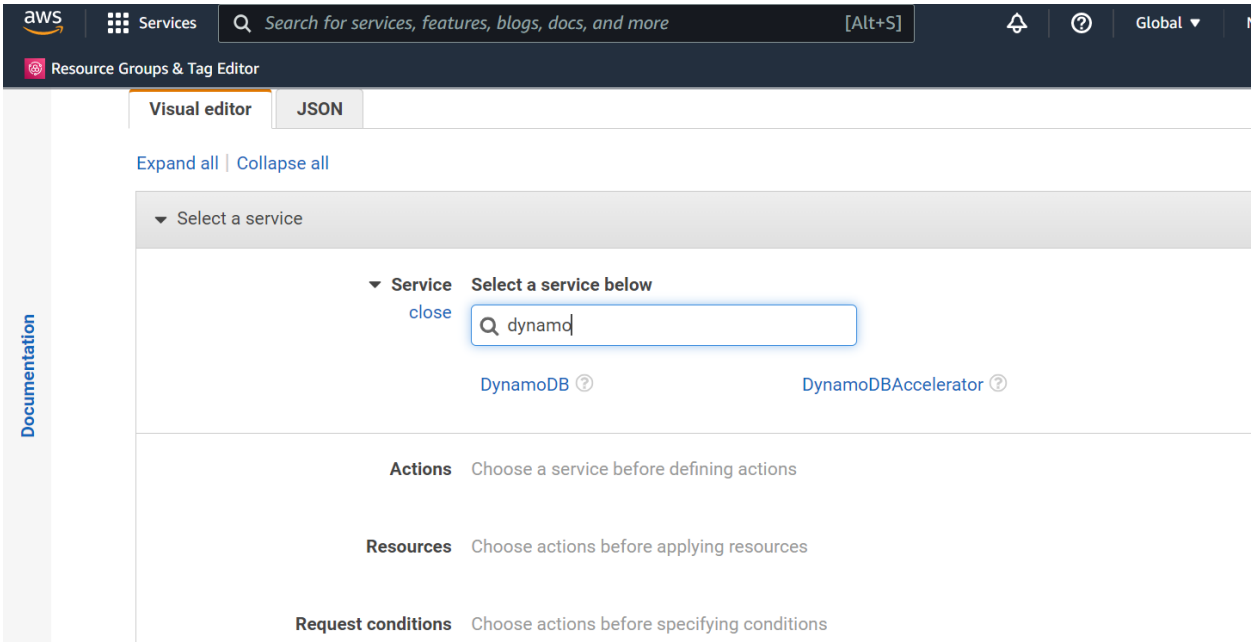


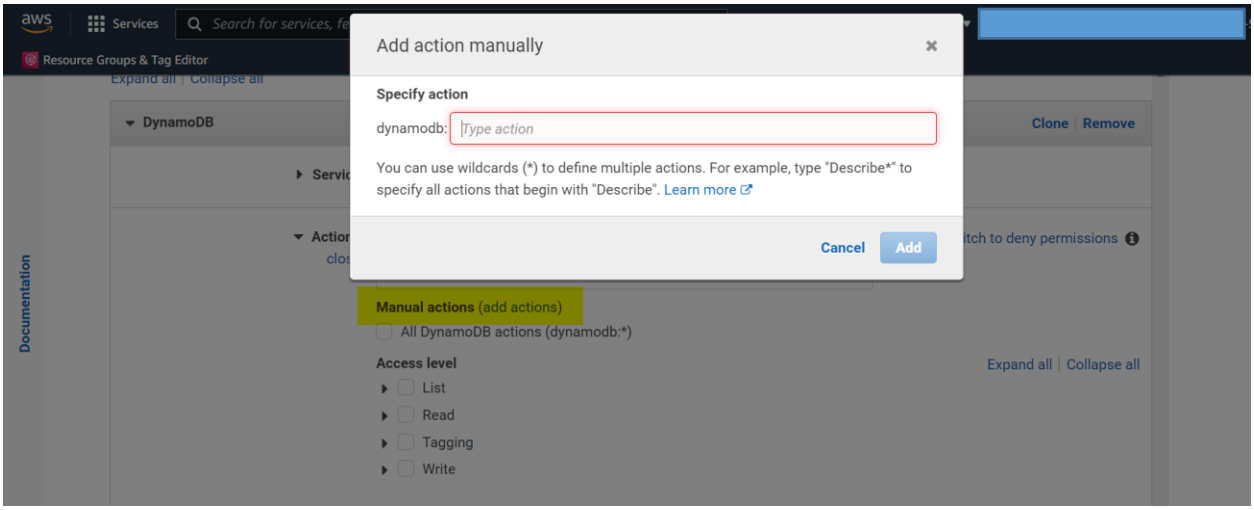


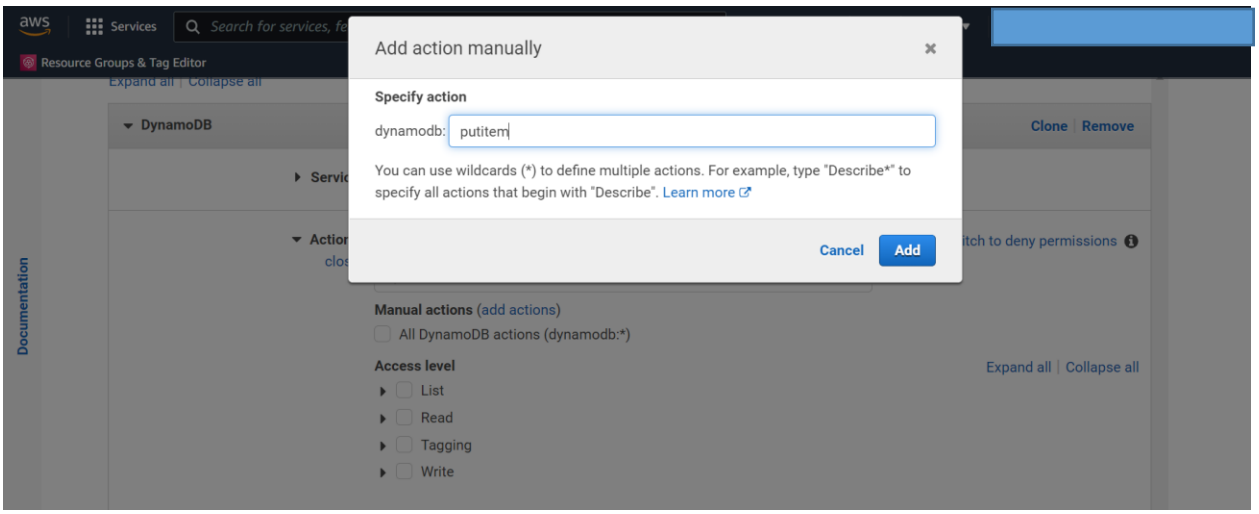


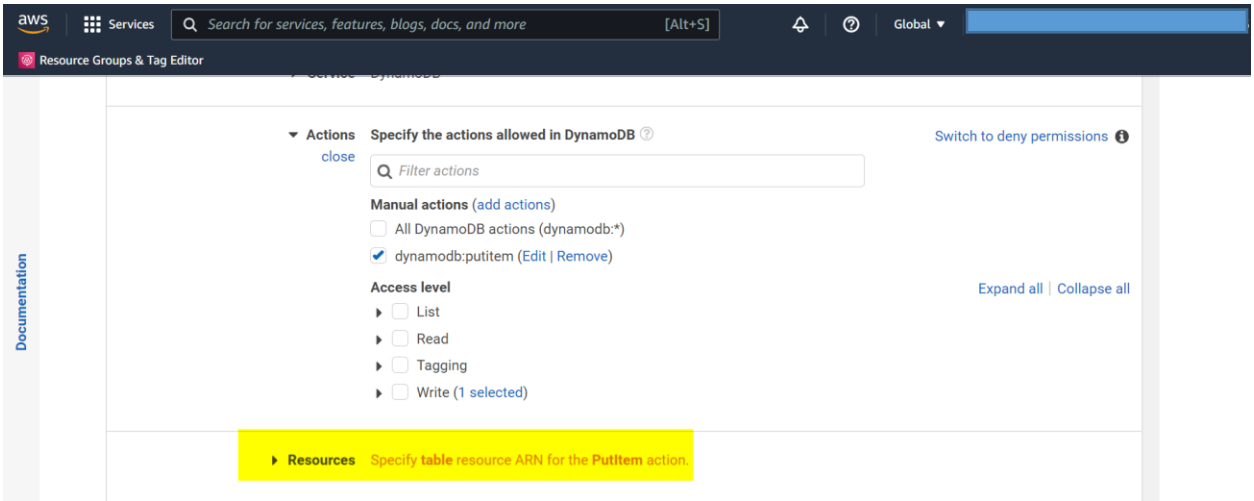
Create the in-line policy with the DynamoDB ARN:

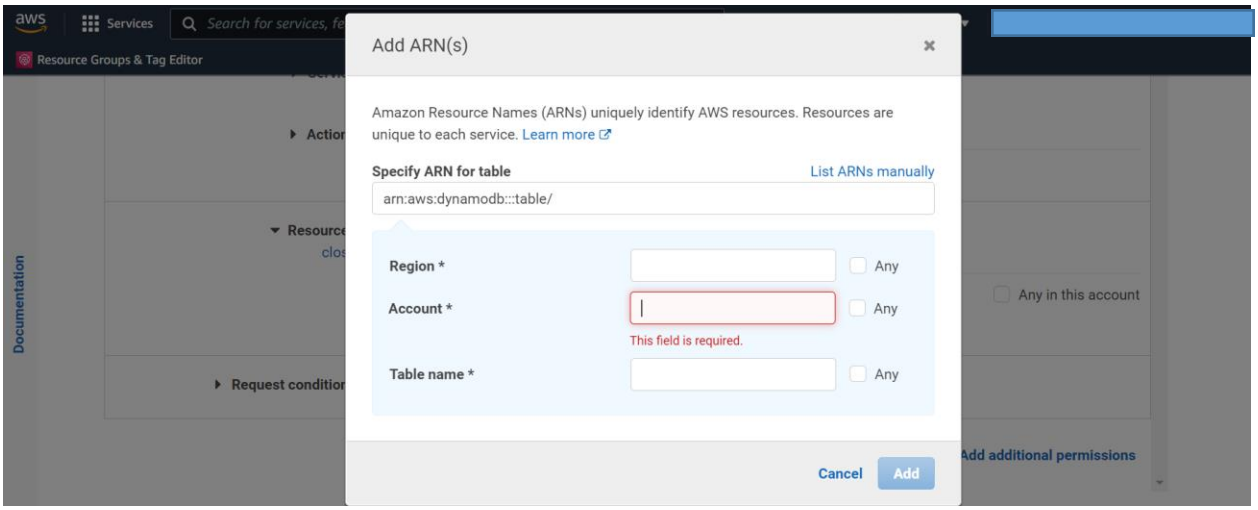


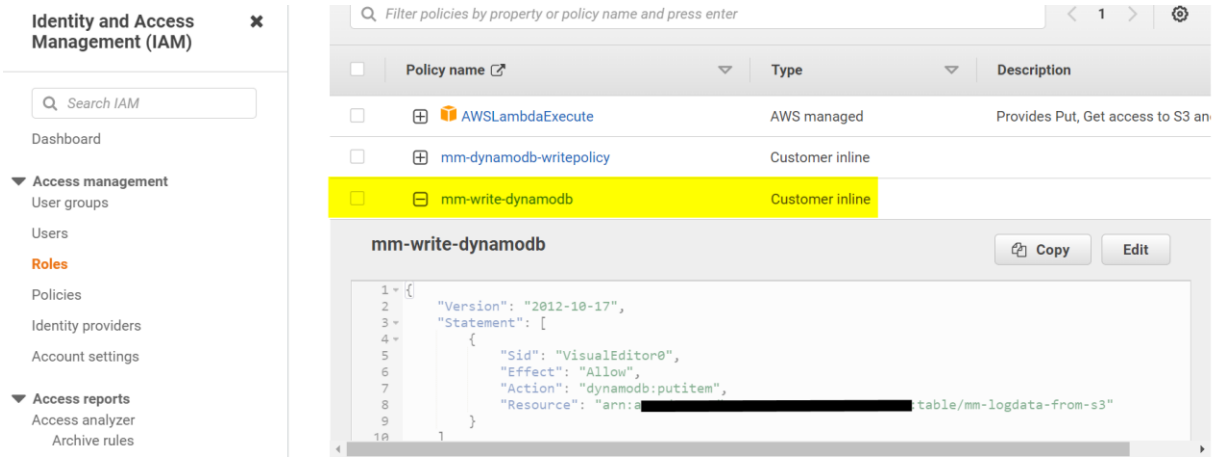



aws
Services
Search for services, fe
Resource Groups & Tag Editor
Expand all | Collapse all
DynamoDB
Clone | Remove
Servic
Actior
clos
Add action manually
Specify action
dynamodb: Type action
You can use wildcards (*) to define multiple actions. For example, type "Describe*" to specify all actions that begin with "Describe". Learn more
Cancel
Add
itch to deny permissions
Manual actions (add actions)
All DynamoDB actions (dynamodb:*)
Access level
Expand all | Collapse all
List
Read
Tagging
Write
Documentation

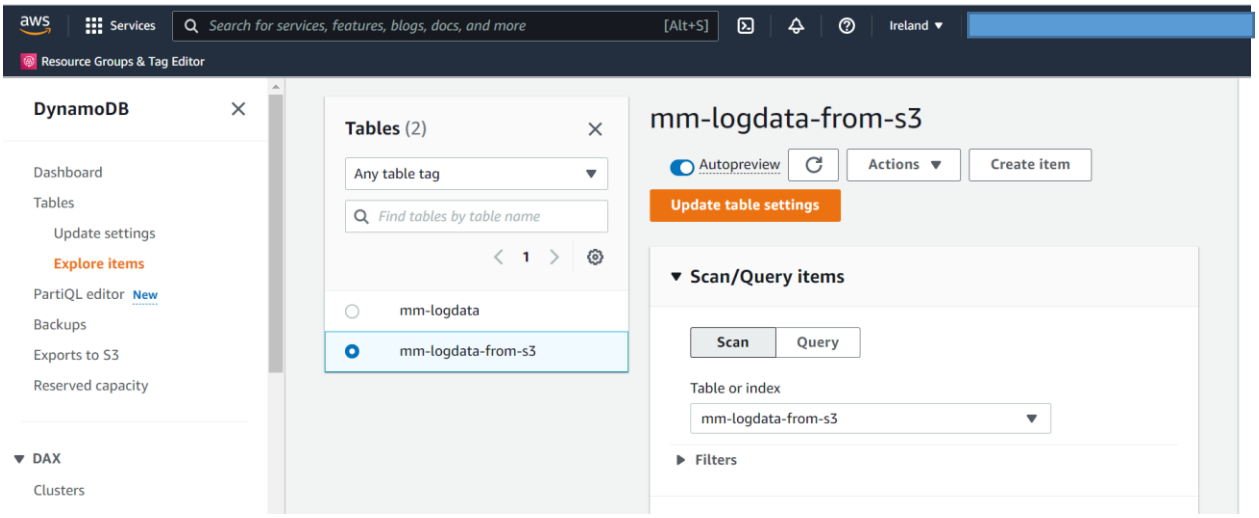

aws
Services
Search for services, fe
Resource Groups & Tag Editor
Expand all | Collapse all
DynamoDB
Clone | Remove
Servic
Actior
clos
Add action manually
Specify action
dynamodb: putitem
You can use wildcards (*) to define multiple actions. For example, type "Describe*" to specify all actions that begin with "Describe". Learn more
Cancel
Add
itch to deny permissions
Manual actions (add actions)
All DynamoDB actions (dynamodb:*)
Access level
Expand all | Collapse all
List
Read
Tagging
Write
Documentation

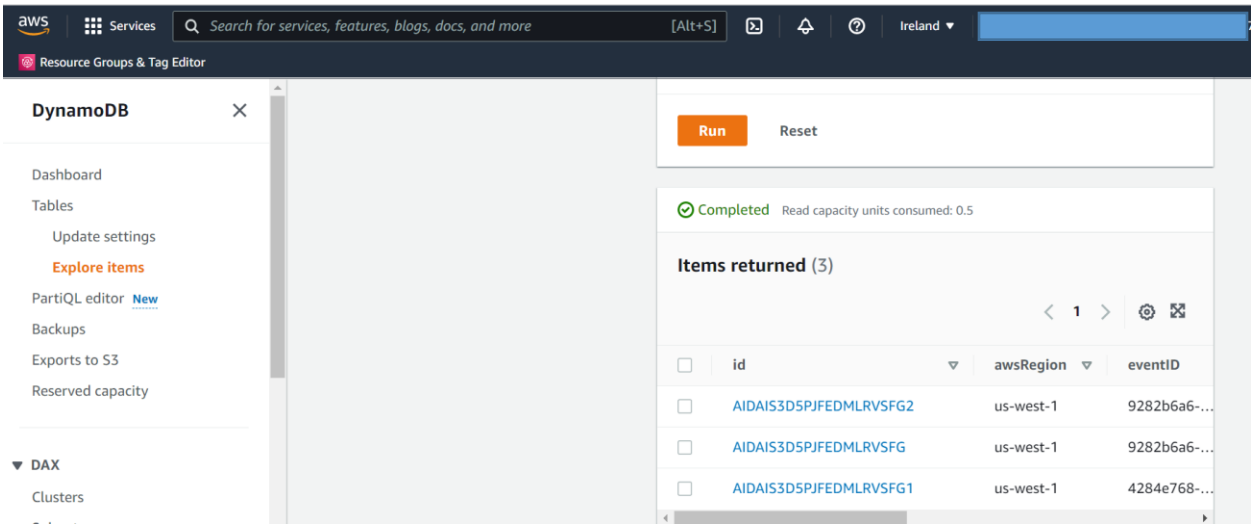

aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Global
Resource Groups & Tag Editor
Actions
close
Specify the actions allowed in DynamoDB
Switch to deny permissions
Filter actions
Manual actions (add actions)
All DynamoDB actions (dynamodb:*)
dynamodb:putitem (Edit | Remove)
Access level
Expand all | Collapse all
List
Read
Tagging
Write (1 selected)
Resources
Specify table resource ARN for the PutItem action.
Documentation

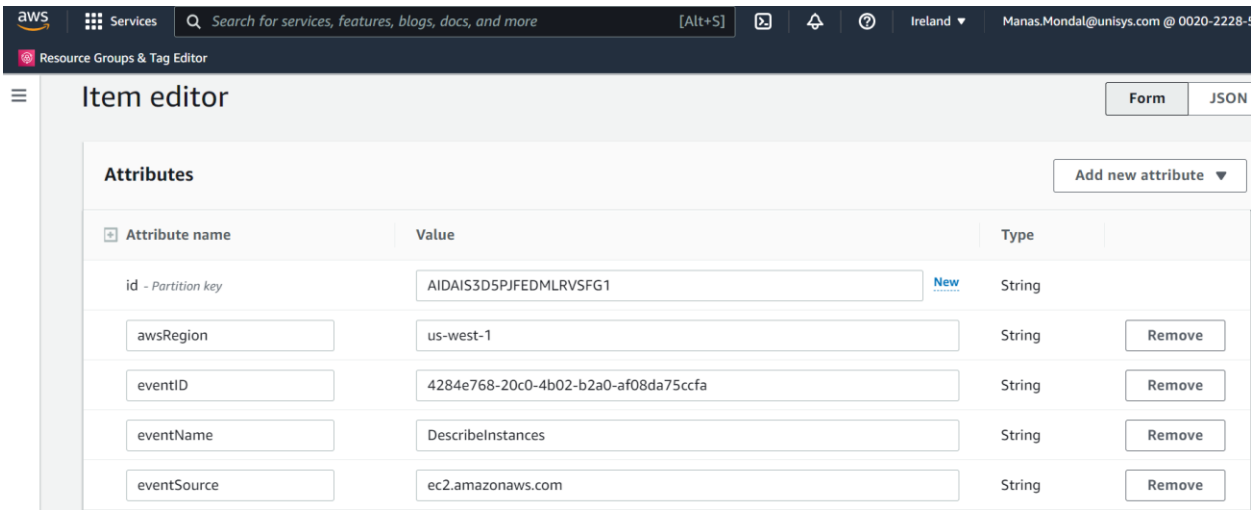




Upload data/ files into S3 and check the entries into DynamoDB table:




aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
Resource Groups & Tag Editor
DynamoDB
Dashboard
Tables
Update settings
Explore items
PartiQL editor New
Backups
Exports to S3
Reserved capacity
DAX
Clusters
Run
Reset
Completed
Read capacity units consumed: 0.5
Items returned (3)
1
id
awsRegion
eventID
AIDAIS3D5PJFEDMLRVSFG2
us-west-1
9282b6a6-...
AIDAIS3D5PJFEDMLRVSFG
us-west-1
9282b6a6-...
AIDAIS3D5PJFEDMLRVSFG1
us-west-1
4284e768-...


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
Ireland
Manas.Mondal@unisys.com @ 0020-2228-
Resource Groups & Tag Editor
Item editor
Form
JSON
Attributes
Add new attribute
Attribute name
Value
Type
id - Partition key
AIDAIS3D5PJFEDMLRVSFG1
New
String
awsRegion
us-west-1
String
Remove
eventID
4284e768-20c0-4b02-b2a0-af08da75ccfa
String
Remove
eventName
DescribeInstances
String
Remove
eventSource
ec2.amazonaws.com
String
Remove