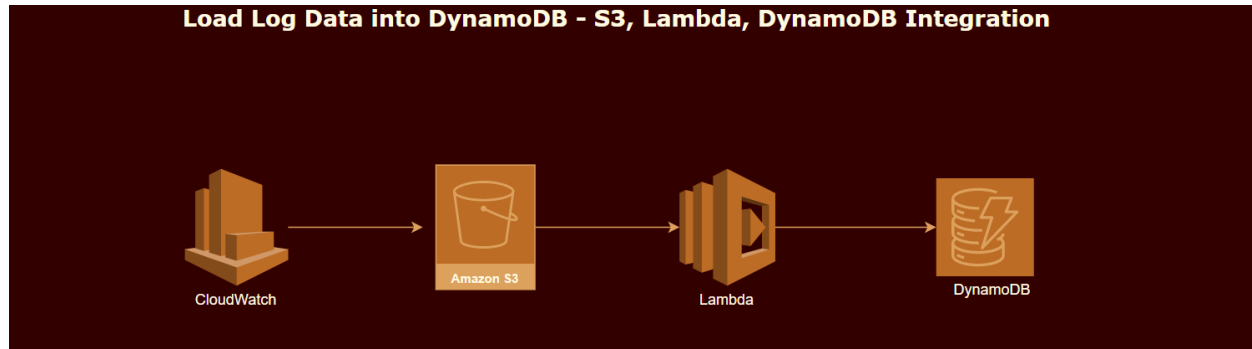


Step-by-step process to load the log data into DynamoDB from AWS S3 – AWS Serverless Architecture

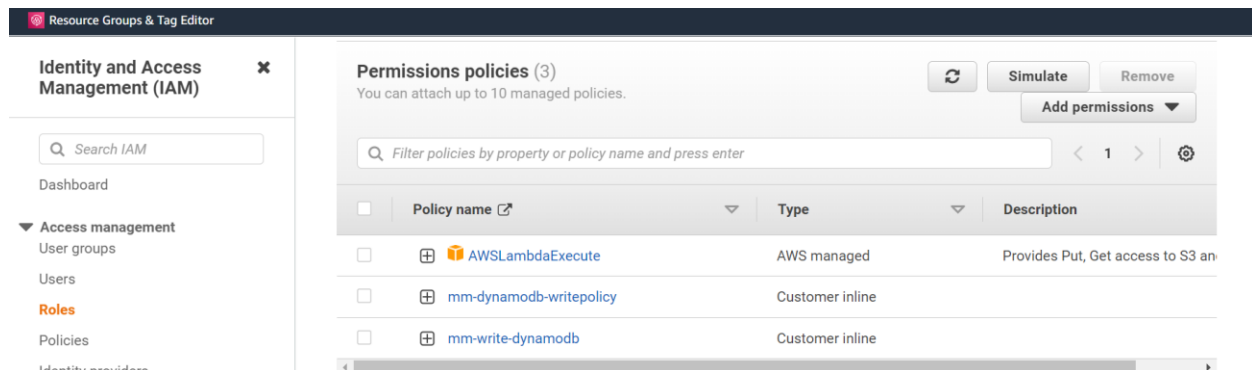
Architecture Diagram:

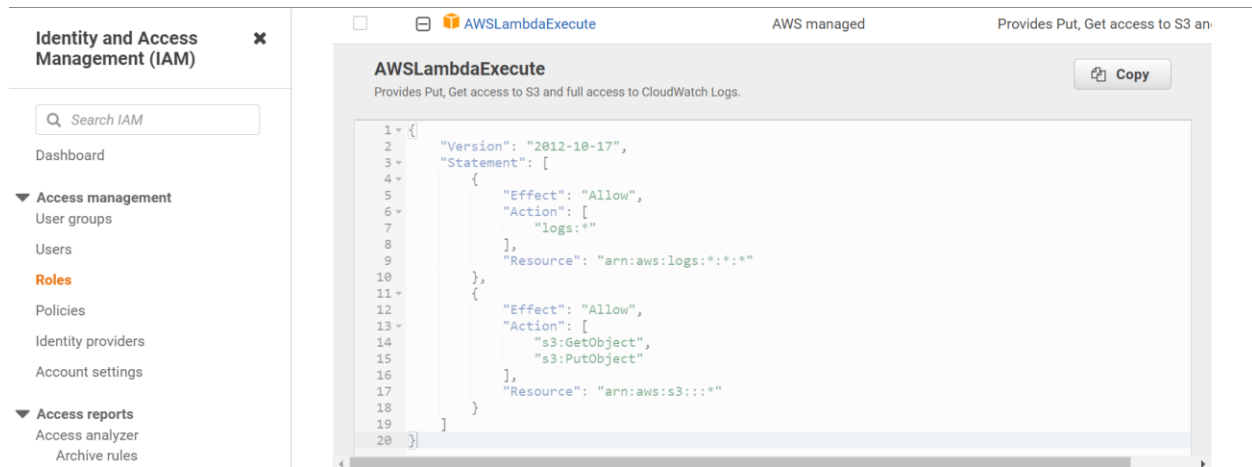


Steps to implement the solution:

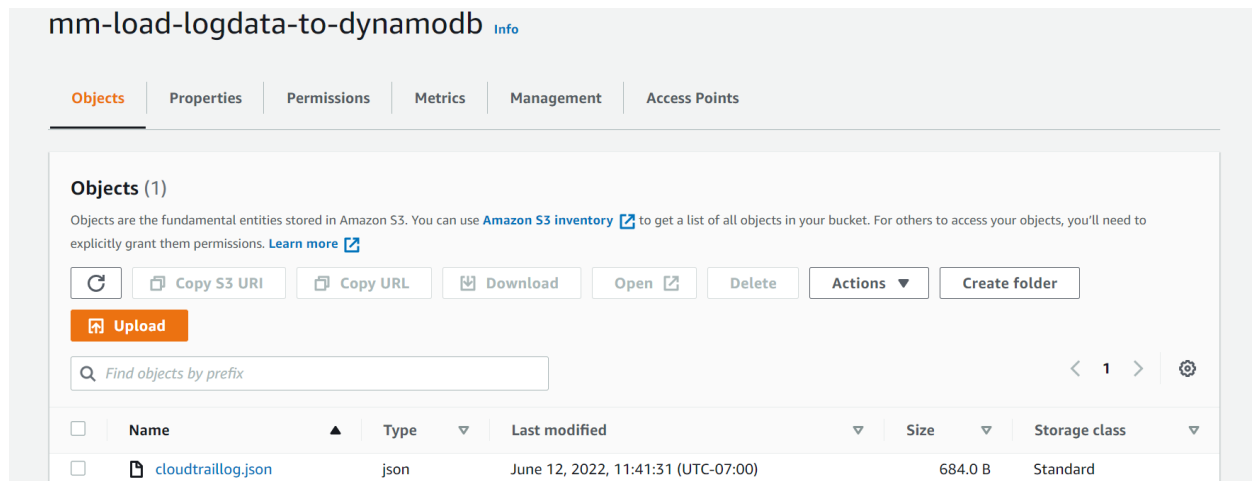
1. Create the appropriate role
2. Create the S3 bucket
3. Log files (Jason format)
4. Create the Lambda function with S3 trigger
5. Create the Dynamodb table
6. Create the in-line policy with the DynamoDB ARN
7. Load data/ files into S3
8. Test the flow

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





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



Sample Log files (Json format):

```

[
  {
    "id": "AIDAI3D5PJFEDMLRVSG1",
    "type": "IAMUser",
    "eventTime": "2022-06-11T23:54:55Z",
    "eventSource": "ec2.amazonaws.com",
    "eventName": "DescribeInstances",
    "awsRegion": "us-west-1",
    "requestID": "db14c600-332d-4ef9-b68a-bbe4749110bd",

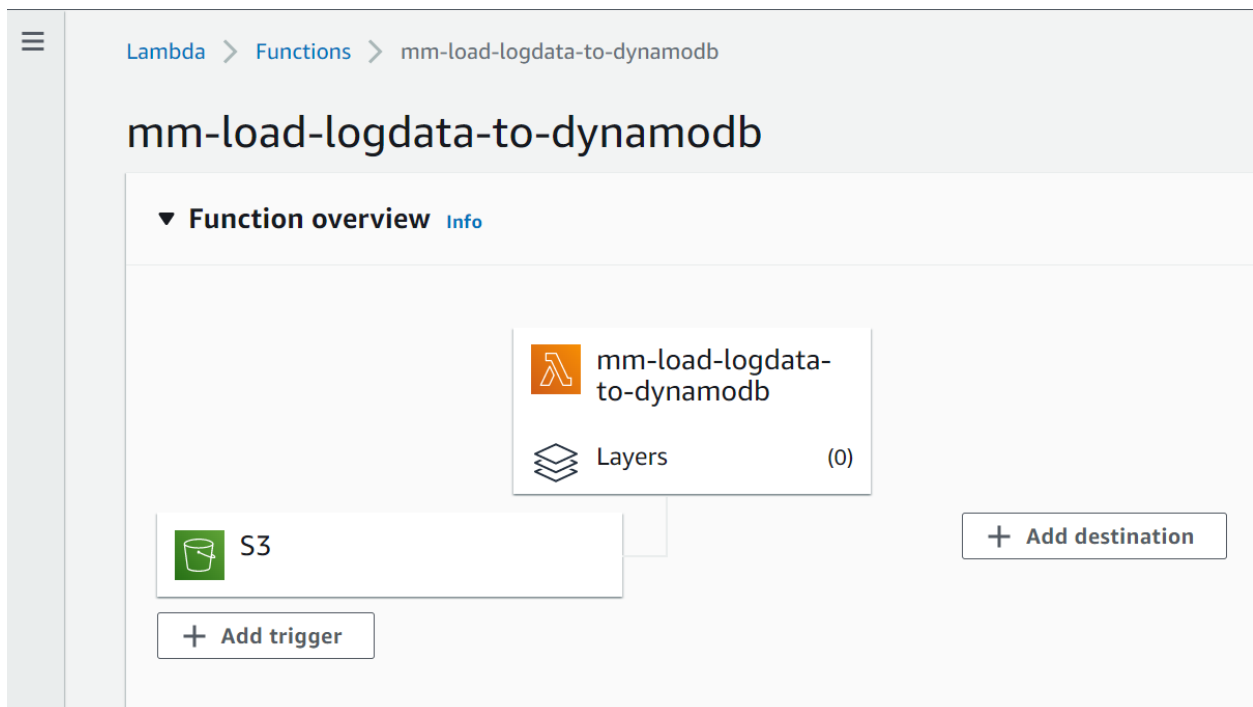
```

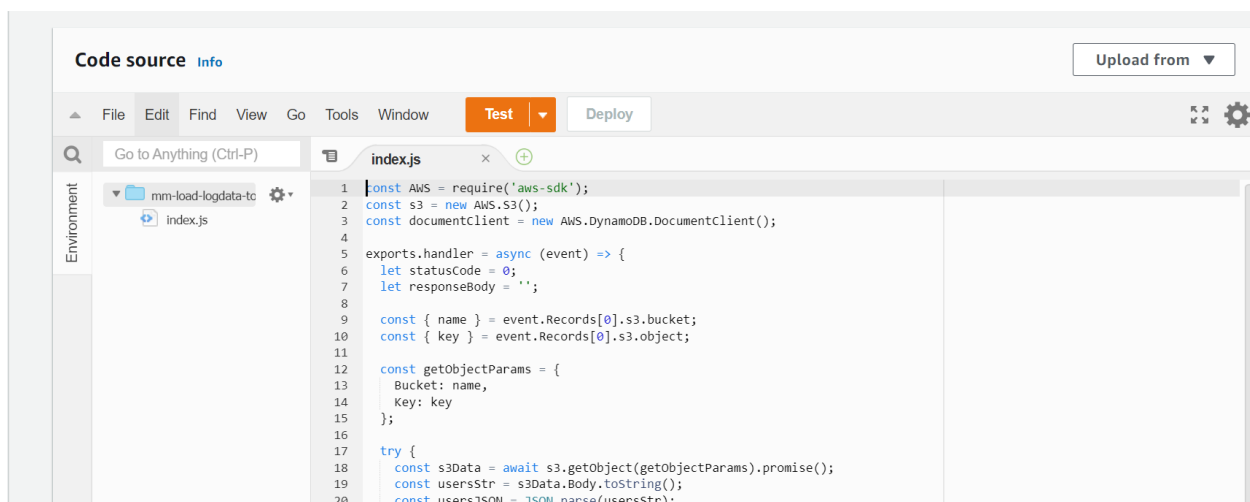
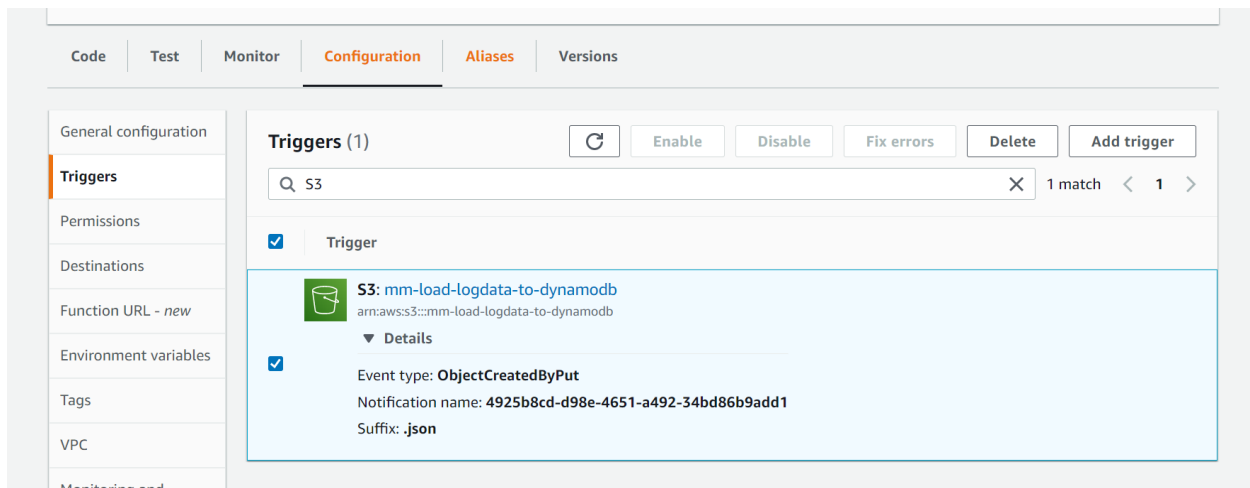
```

    "eventID": "4284e768-20c0-4b02-b2a0-af08da75ccfa"
  },
  {
    "id": "AIDAI3D5PJFEDMLRVSG2",
    "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;
};


```

Code properties

Package size

693.0 byte

SHA256 hash

 SSbl6VTxGDy/aVZHeWGHal/HxCwjM/F3MZxM+fFO0EE=

Last modified

June 12, 2022, 11:29 AM PDT

Runtime settings [Info](#)

Edit

Runtime

Node.js 16.x

Handler [Info](#)

index.handler

Architecture [Info](#)

x86_64

Layers [Info](#)

Edit

Add a layer

Merge order

Name

Layer version

Compatible runtimes

Compatible architectures

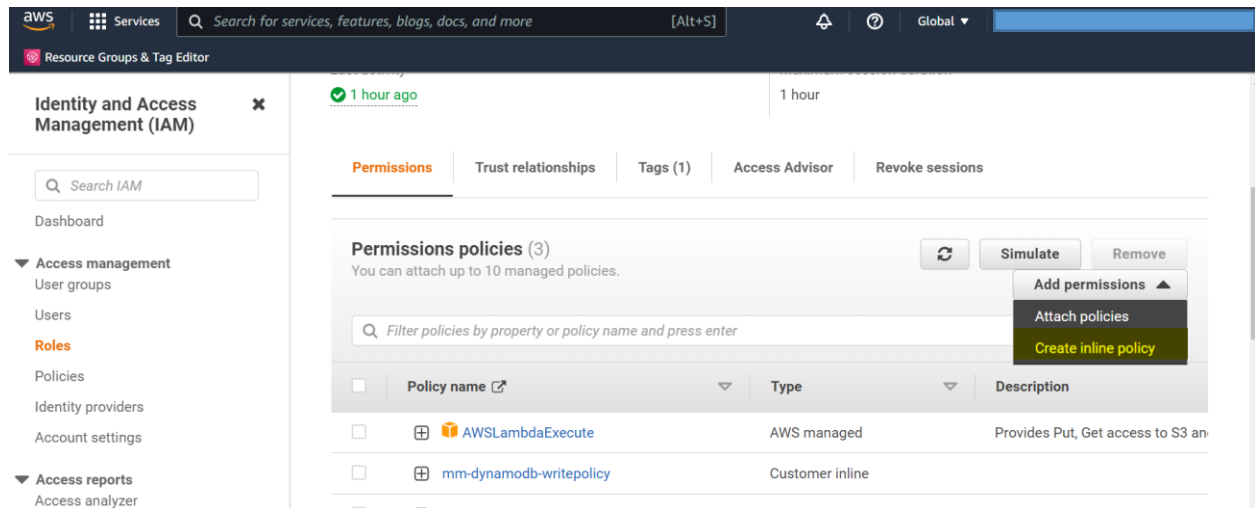
Create the Dynamodb table:

This screenshot shows the AWS Management Console interface for creating a new DynamoDB table. The left sidebar displays the navigation menu with options like Dashboard, Tables, Update settings, Explore items, PartiQL editor, Backups, Exports to S3, and Reserved capacity. The main content area shows the 'DynamoDB' console with a breadcrumb trail: 'DynamoDB > Tables > mm-logdata-from-s3'. A 'Tables (2)' panel lists two tables: 'mm-logdata' and 'mm-logdata-from-s3', with the latter selected. The 'General information' section for the selected table shows the Partition key as 'id (String)' and the Sort key as '-'. The Capacity mode is set to 'Provisioned', and the Table status is 'Active' with 'No active alarms'.

This screenshot shows the 'General information' section for the 'mm-logdata-from-s3' table. The Partition key is 'id (String)' and the Sort key is '-'. The Capacity mode is 'Provisioned'. The Table status is 'Active' with 'No active alarms'. The 'Additional info' section shows the Table class as 'DynamoDB Standard' and the Indexes as '0 globals, 0 locals'.

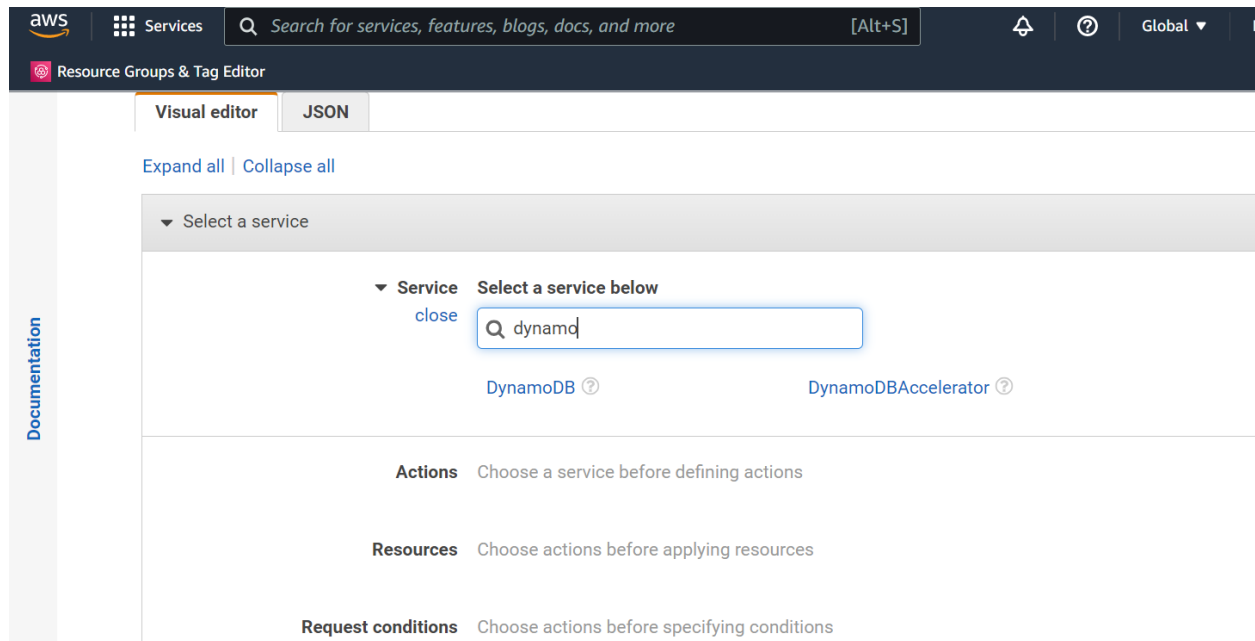
This screenshot shows the 'Table capacity metrics' section for the 'mm-logdata-from-s3' table. The metrics are displayed for a 1-hour period. The 'Read usage' chart shows a sharp spike in usage at 17:07, reaching a value of 5. The 'Read throttling' chart shows a spike in throttling at 17:07, reaching a value of 1. The 'Read throttling' chart also shows a spike in throttling at 20:06, reaching a value of 1. The legend indicates that the metrics are for 'Provisioned' and 'Consumed' capacity.

Create the in-line policy with the DynamoDB ARN:



The screenshot shows the AWS IAM console interface. On the left, the 'Identity and Access Management (IAM)' sidebar is visible with a search bar and navigation links for Dashboard, Access management (User groups, Users, Roles, Policies, Identity providers, Account settings), and Access reports (Access analyzer). The main content area is titled 'Permissions' and shows 'Permissions policies (3)'. A dropdown menu is open, showing options: 'Add permissions', 'Attach policies', and 'Create inline policy' (highlighted in yellow). Below the menu, a table lists existing policies:

<input type="checkbox"/>	Policy name	Type	Description
<input type="checkbox"/>	AWSLambdaExecute	AWS managed	Provides Put, Get access to S3 an
<input type="checkbox"/>	mm-dynamodb-writepolicy	Customer inline	



The screenshot shows the 'Visual editor' tab for creating an inline policy. The 'Service' dropdown is set to 'dynamodb'. The 'Actions' section is empty, and the 'Resources' and 'Request conditions' sections are also empty. The interface includes a 'Visual editor' and 'JSON' tab, and a 'Documentation' sidebar on the left.

Visual editor | **JSON**

Expand all | Collapse all

Select a service

Service: Select a service below

close

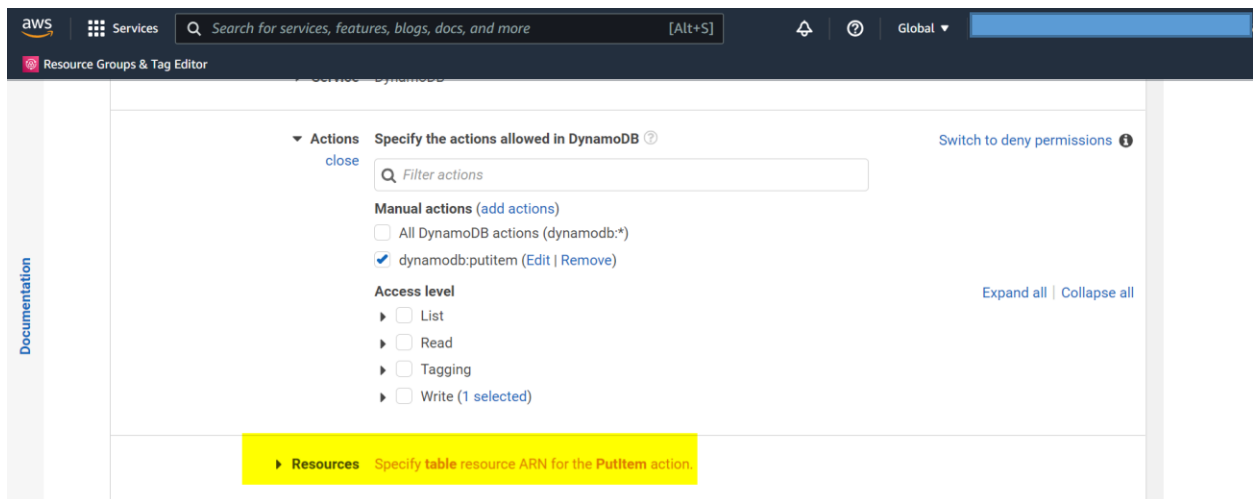
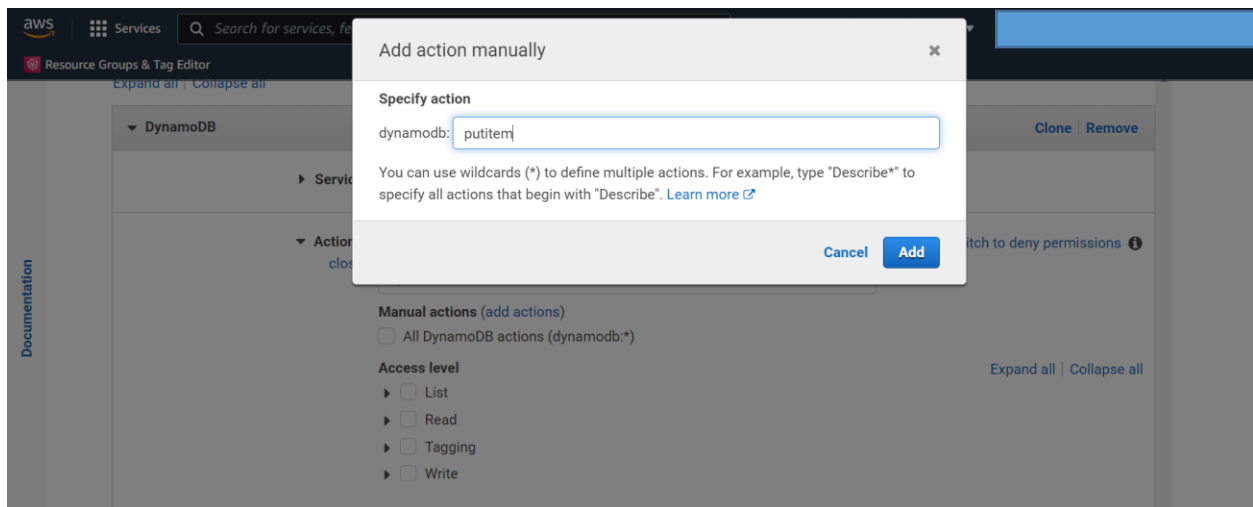
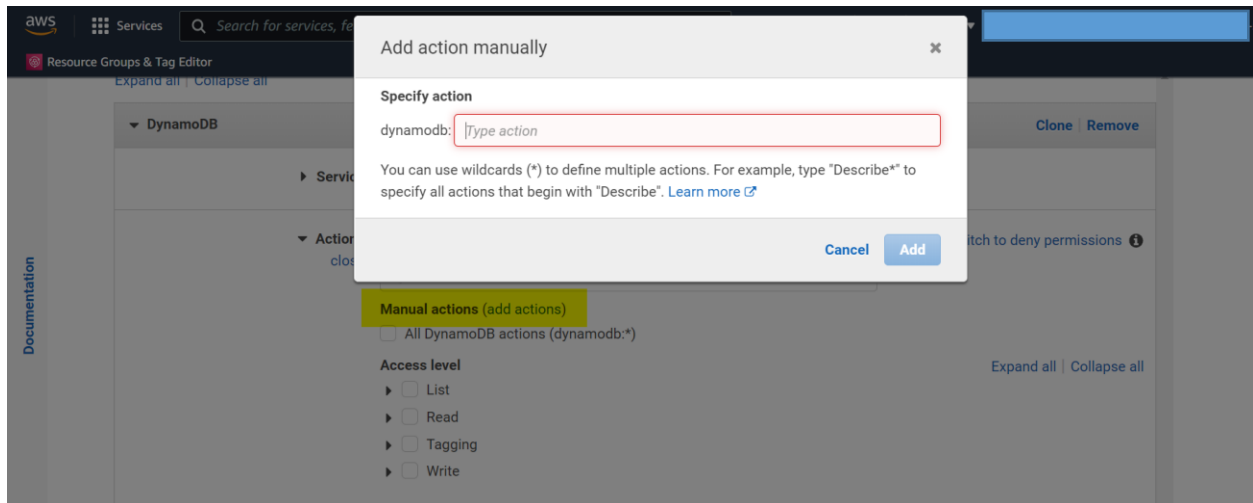
dynamodb

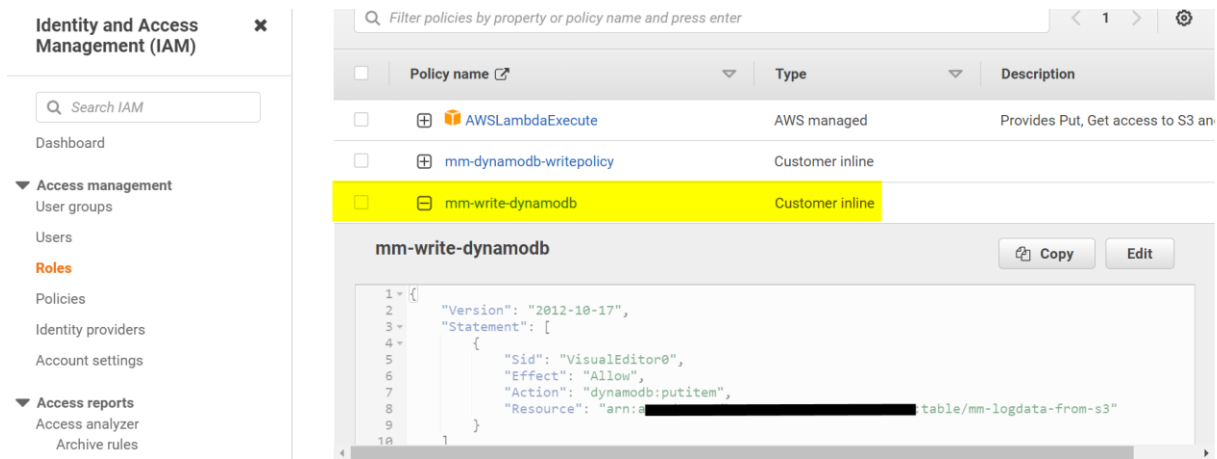
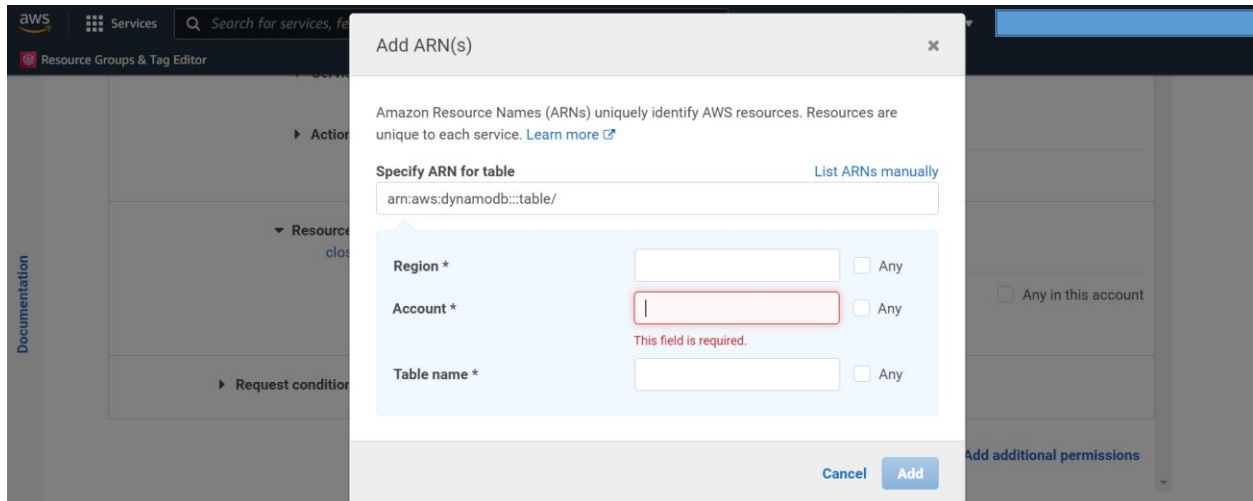
DynamoDB ? | DynamoDBAccelerator ?

Actions Choose a service before defining actions

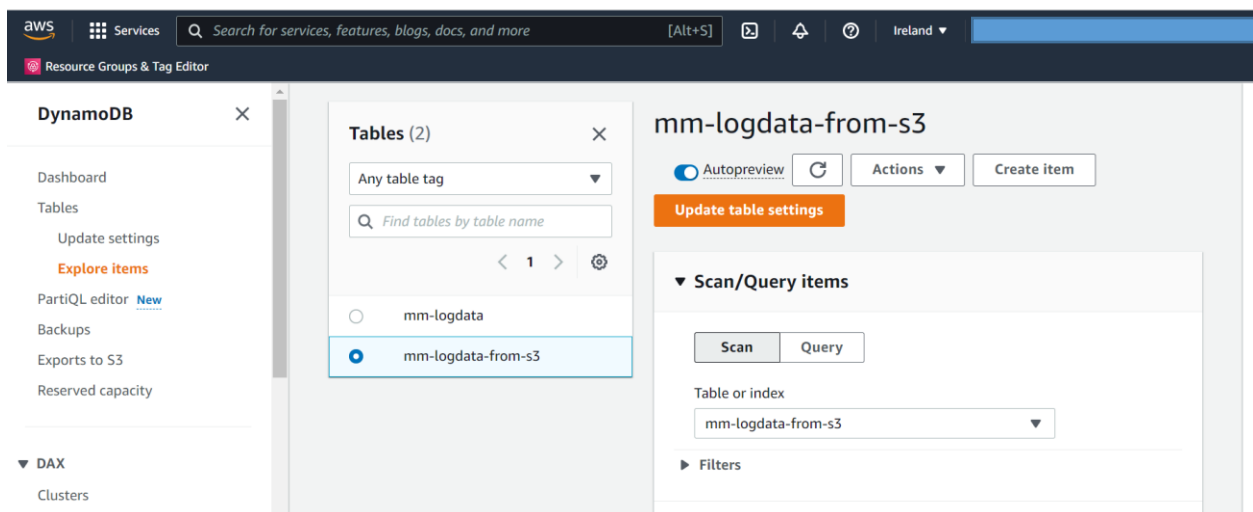
Resources Choose actions before applying resources

Request conditions Choose actions before specifying conditions





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

Backups

Exports to S3

Reserved capacity

DAX

Clusters

Run

Reset

Completed

Read capacity units consumed: 0.5

Items returned (3)

< 1 >

	id	awsRegion	eventID
<input type="checkbox"/>	AIDAI3D5PJFEDMLRV5FG2	us-west-1	9282b6a6-...
<input type="checkbox"/>	AIDAI3D5PJFEDMLRV5FG	us-west-1	9282b6a6-...
<input type="checkbox"/>	AIDAI3D5PJFEDMLRV5FG1	us-west-1	4284e768-...

aws

Services

Search for services, features, blogs, docs, and more

[Alt+S]

Ireland

Manas.Mondal@unisys.com @ 0020-2228-5

Resource Groups & Tag Editor

Item editor

FormJSON

Attributes

Add new attribute

Attribute name	Value	Type	
id - Partition key	AIDAI3D5PJFEDMLRV5FG1	String	New
awsRegion	us-west-1	String	Remove
eventID	4284e768-20c0-4b02-b2a0-af08da75ccfa	String	Remove
eventName	DescribeInstances	String	Remove
eventSource	ec2.amazonaws.com	String	Remove