Rule Definition Lambda Setup Guide

Prerak

July 2, 2025

1 Introduction

This guide outlines how to set up a Lambda function to process and store rules in a DynamoDB table. The Lambda function will be exposed via API Gateway to allow rule submissions through a simple HTML page.

2 What You Need to Do

2.1 Create DynamoDB Table

To store rules in DynamoDB, create a table with the following configuration:

- **Table Name**: `Rules`
- **Partition Key**: `rule_id` (String)
- **Sort Key**: None
- **Capacity Mode**: On-demand (recommended for scalability)

2.2 Lambda Function Code

Create a Lambda function that receives rule data via API Gateway and stores it in DynamoDB. Here's the Lambda code:

```
import json
import boto3

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('Rules')

def lambda_handler(event, context):
    try:
        body = json.loads(event['body'])  # Get the incoming JSON
        if 'rule_id' not in body:
            return {
                'statusCode': 400,
                'headers': {
                    'Access-Control-Allow-Origin': '*',
                    'Access-Control-Allow-Headers': 'Content-Type'
                        ,
```

```
                    'Access-Control-Allow-Methods': 'POST,OPTIONS'
                },
                'body': json.dumps({'error': 'Missing rule_id'})
            }

        table.put_item(Item=body)  # Store the rule in DynamoDB

        return {
            'statusCode': 200,
            'headers': {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Headers': 'Content-Type',
                'Access-Control-Allow-Methods': 'POST,OPTIONS'
            },
            'body': json.dumps({'message': 'Rule stored
               successfully', 'rule_id': body.get('rule_id')})
        }
    except Exception as e:
        return {
            'statusCode': 400,
            'headers': {
                'Access-Control-Allow-Origin': '*',
                'Access-Control-Allow-Headers': 'Content-Type',
                'Access-Control-Allow-Methods': 'POST,OPTIONS'
            },
            'body': json.dumps({'error': str(e)})
        }
```

IAM Role Configuration for Lambda: Create an IAM role for the Lambda function with the following permissions:

- `AWSLambdaDynamoDBExecutionRole`: Allows Lambda to interact with DynamoDB.
- `AWSLambdaBasicExecutionRole`: Allows Lambda to write logs to CloudWatch.
- Optionally, you can add more restrictive roles for production environments.

2.3 API Gateway Setup

To expose the Lambda function via API Gateway:

- Create a **REST API**.
- Create a resource called `/store-rule`.
- Attach a `POST` method to `/store-rule`.
- Enable **Lambda Proxy Integration** for the `POST` method.
- Set **CORS** for the `POST` method to allow cross-origin requests.

For CORS headers, ensure the following are returned in the API Gateway response:

```
{
    "Access-Control-Allow-Origin": "*",
```

```
    "Access-Control-Allow-Headers": "Content-Type",
    "Access-Control-Allow-Methods": "POST,OPTIONS"
}
```

2.4 HTML Web Interface

Create an HTML page for rule submission. Here's the code for a simple web page:

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Submit Rule to DynamoDB</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            padding: 30px;
        }
        textarea {
            width: 600px;
            height: 200px;
        }
        button {
            padding: 10px 20px;
            font-size: 16px;
            margin-top: 10px;
        }
        #response {
            margin-top: 20px;
            white-space: pre-wrap;
            color: green;
        }
    </style>
</head>
<body>
    <h2>Store Rule in DynamoDB</h2>
    <textarea id="jsonInput" placeholder='{"rule_id": "abc123", "
       condition": "temp > 30", "action": "alert"}'></textarea>
    <br>
    <button onclick="submitData()">Submit</button>
    <p id="response"></p>

    <script>
        async function submitData() {
            const jsonText = document.getElementById("jsonInput").
               value;
            const responseEl = document.getElementById("response")
               ;

            try {
                const data = JSON.parse(jsonText);
                const res = await fetch("https://<api-id>.execute-
```

```
                    api.ap-south-1.amazonaws.com/prod/store-rule",
                    {
                        method: "POST",
                        headers: {
                            "Content-Type": "application/json"
                        },
                        body: JSON.stringify(data)
                    });

                    const result = await res.json();
                    if (res.ok) {
                        responseEl.style.color = "green";
                        responseEl.innerText = "Success:\n" + JSON.
                            stringify(result, null, 2);
                    } else {
                        responseEl.style.color = "red";
                        responseEl.innerText = "Server Error:\n" +
                            JSON.stringify(result, null, 2);
                    }
                } catch (e) {
                    responseEl.style.color = "red";
                    responseEl.innerText = "Invalid JSON:\n" + e.
                        message;
                }
            }
        </script>
</body>
</html>
```

Notes: - Replace `https://<api-id>.execute-api.ap-south-1.amazonaws.com/prod/store-rule` with your API Gateway's Invoke URL. - Ensure that the HTML file is served from a local server (e.g., using `python -m http.server` for testing).

2.5 Test the Setup

Once the setup is complete, you can test the functionality:

- Open the HTML page in your browser.
- Enter a sample rule (e.g., `"rule_id": "abc123", "condition": "temp > 30", "action": "alert"`) and submit it.
- Check DynamoDB for the new rule, or use `curl` to manually test the API:

```
curl -X POST https://<api-id>.execute-api.ap-south-1.amazonaws.com
    /prod/store-rule \
-H "Content-Type: application/json" \
-d '{"rule_id": "abc123", "condition": "temp > 30", "action": "
    alert"}'
```

3 Technical Notes

3.1 API Usage

The API Gateway endpoint accepts `POST` requests with the following:

- **Endpoint**: `https://<api-id>.execute-api.ap-south-1.amazonaws.com/prod/store-rule`
- **Headers**: `Content-Type: application/json`
- **Body**: A JSON object containing at least a `rule_id` field.

3.2 Maintenance

Here are some pointers to keep the system running smoothly:

- **Monitoring**: Use AWS CloudWatch for logs and API Gateway metrics.
- **Scaling**: Lambda and API Gateway auto-scale. For DynamoDB, use `On-demand` mode to handle scaling automatically.
- **Security**: Ensure Lambda has the correct IAM roles and permissions. Use least privilege principles in production.
- **Backups**: Enable DynamoDB's point-in-time recovery for data safety.