Rule Engine Lambda Setup Guide

AWS Integration with PostgreSQL, DynamoDB, SNS, SQS, and S3

Prerak

Contents

1 Introduction

This guide explains how to set up the `RuleEngineFunction` Lambda in AWS. The function processes weather data, evaluates rules stored in DynamoDB, and triggers actions via SNS, SQS, and S3.

It assumes the following resources already exist:

- DynamoDB tables: `Rules`, `ScienceTeamReports`
- PostgreSQL database with tables: `weather_current`, `weather_forecast`
- SQS queue: `weather-batch-queue`
- SNS topic: `weather-alerts`
- S3 bucket: `weather-blogs-2025-science`

2 Architecture

The Lambda function integrates with the following AWS services:

- **PostgreSQL (RDS)**: Weather data source
- **DynamoDB**: Stores rules and science team reports
- **SQS**: For batch notifications
- **SNS**: Sends alerts to stakeholders
- **S3**: Stores blog reports
- **Lambda**: Orchestrates rule evaluation and triggers actions

3 Setup Instructions

3.1 Step 1: Create the Lambda Function

1. Go to the AWS Lambda Console.
2. Click **Create function** > **Author from scratch**.
3. Fill in the following:
    - **Function name**: `RuleEngineFunction`
    - **Runtime**: Python 3.12
    - **Architecture**: `x86_64`
    - **Handler**: `lambda_function.lambda_handler`
4. Under **Change default execution role**:
    - Choose **Create a new role with basic Lambda permissions**.
    - Edit the IAM role to add permissions for SNS, SQS, S3, DynamoDB, and RDS.

5. Set environment variables and upload your code.
6. Click **Deploy**.

3.2 Step 2: SNS Configuration

- **Topic name**: `weather-alerts`
- **Protocol**: Email or SMS
- **Subscription**: Add stakeholder emails or phone numbers
- **IAM Permissions**: Ensure Lambda has `sns:Publish` access

Note: SNS offers 1M requests and 100K SMS per month under the AWS Free Tier.

3.3 Step 3: S3 Configuration

- **Bucket name**: `weather-blogs-2025-science`
- **Versioning**: Enable to track file changes
- **Access Control**: Adjust visibility settings as needed
- **IAM Permissions**: Grant `s3:PutObject` permission to Lambda

Note: S3 Free Tier includes 5 GB storage, 20K GET, and 2K PUT requests.

3.4 Step 4: SQS Configuration

- **Queue name**: `weather-batch-queue`
- **Queue type**: Standard
- **Visibility Timeout**: Set to prevent duplicate processing
- **IAM Permissions**: Allow `sqs:SendMessage` from Lambda

Note: Free Tier includes 1 million SQS requests monthly.

3.5 Step 5: Test with a Demo Rule

1. Add a sample rule to the `Rules` table:

Listing 1: Demo Rule

```
{
  "rule_id": "rule123",
  "rule_name": "HighTemperatureAlert",
  "rule_type": "weather",
  "data_type": "current",
  "farm_id": "udaipur_farm1",
  "stakeholder": "farmer",
  "language": "en",
  "conditions": {
    "metric": "temperature_c",
    "operator": ">",
```

```
    "value": 30
  },
  "actions": [
    {
      "type": "sms",
      "scenario": "high_temperature",
      "message": "Urgent: High temperature detected."
    }
  ],
  "stop_on_match": true,
  "conflict_resolution": "first_match",
  "priority": 1,
  "active": true
}
```

4 Operational Flow

(a) **Trigger**: Lambda is invoked by CloudWatch schedule, SQS, or test event.

(b) **Rule Retrieval**: Fetches matching rules from the `Rules` table using `StakeholderIndex`.

(c) **Data Retrieval**: Connects to PostgreSQL to query weather data from `weather_current` or `weather_forecast`.

(d) **Rule Evaluation**: Evaluates conditions including:

- Simple: `metric operator value`
- Time-based: `TIME_WINDOW`, `DAY_DIFF`, etc.
- Sequential: `SEQUENCE`
- Delta: `delta_gt`, `delta_lt`

(e) **Data Aggregation**: Since data is fetched from 4 APIs, a **majority rule** is applied—i.e., the final metric is determined based on the majority agreement among API responses to ensure robustness and reduce noise.

(f) **Conflict Resolution**: Uses strategies like `first_match`, `highest_priority`, or `most_severe`.

(g) **Notifications**: Sends SMS/Email alerts via SNS and queues updates via SQS.

(h) **Science Team Updates**: Stores summaries in `ScienceTeamReports` and blog reports in S3.