Weather Data Ingestion Lambda Setup Guide

Prerak

July 2, 2025

Contents

1 Introduction

This document provides a comprehensive guide for setting up the Weather Data Ingestion Lambda function in an AWS account. The Lambda function fetches weather data from four free-tier APIs (Yr.no, Open-Meteo, OpenWeatherMap, and WeatherAPI) and stores it in an AWS RDS PostgreSQL database. The guide includes detailed setup instructions and technical notes on usage to ensure a smooth handover.

2 Overview

The Weather Data Ingestion Lambda function is a serverless application that:

- Fetches current and forecast weather data from four APIs for specified locations.
- Stores the data in a PostgreSQL database on AWS RDS.

2.1 Technology Stack

- **Runtime**: Python 3.12
- **Architecture**: x86_64
- **AWS Services**: Lambda, RDS (PostgreSQL), CloudShell
- **APIs**:
  - Yr.no: `https://api.met.no/weatherapi/locationforecast/2.0/compact`
  - Open-Meteo: `https://api.open-meteo.com/v1/forecast`
  - OpenWeatherMap: Requires API key (https://home.openweathermap.org/api_keys)
  - WeatherAPI: Requires API key (https://www.weatherapi.com/docs/)
- **Lambda Layers**:
  - `Klayers-p312-psycopg2-binary:1` (`arn:aws:lambda:ap-south-1:770693421928:layer:`
  - `AWSSDKPandas-Python312:16` (`arn:aws:lambda:ap-south-1:336392948345:layer:AWSSD`

3 Prerequisites

Before starting, ensure you have:

- An AWS account with administrative access.
- API keys for OpenWeatherMap and WeatherAPI.

4 Setup Instructions

4.1 Step 1: Obtain API Keys

1. Sign up at https://home.openweathermap.org/api_keys to obtain an OpenWeatherMap API key.
2. Sign up at https://www.weatherapi.com/docs/ to obtain a WeatherAPI key.
3. Save these keys securely for use in Lambda environment variables.

4.2 Step 2: Set Up AWS RDS (PostgreSQL)

1. Navigate to the AWS RDS Console.
2. Select **Create Database**.
3. Choose **PostgreSQL** (not Aurora, to stay within free tier).
4. Select **Free Tier** template.
5. Configure the database:
   - **DB Instance Identifier**: `weather-db`
   - **Master Username**: `postgres` (or your choice)
   - **Master Password**: Set and save securely.
   - **Public Access**: Enable.
   - **VPC Security Group**: Allow inbound traffic on port 5432.
6. Note the database endpoint (e.g., `weather-db.cfaciysiukn7.ap-south-1.rds.amazonaws.com`)
7. Connect using AWS CloudShell or a local terminal:

```
psql -h weather-db.cfaciysiukn7.ap-south-1.rds.amazonaws.com -
   U postgres -d postgres
```

8. Create tables using the following schema:

```
CREATE TABLE weather_forecast (
    id SERIAL PRIMARY KEY,
    farm_id VARCHAR(50) NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    temperature_c FLOAT,
    humidity_percent FLOAT,
    wind_speed_mps FLOAT,
    wind_direction_deg FLOAT,
    rainfall_mm FLOAT,
    chance_of_rain_percent FLOAT,
    source VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE weather_current (
```

```
    id SERIAL PRIMARY KEY,
    farm_id VARCHAR(50) NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    temperature_c FLOAT,
    humidity_percent FLOAT,
    wind_speed_mps FLOAT,
    wind_direction_deg FLOAT,
    rainfall_mm FLOAT,
    solar_radiation_wm2 FLOAT,
    source VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4.3 Step 3: Create AWS Lambda Function

1. Navigate to the AWS Lambda Console.
2. Select **Create function**.
3. Choose **Author from Scratch** and configure:
   - **Function name**: `WeatherIngestionFunction`
   - **Runtime**: Python 3.12
   - **Architecture**: x86_64
4. Under **Change default execution role**:
   - Select **Create a new role with basic Lambda permissions** or use an existing role with administrative access (e.g., `AWSLambdaFullAccess`).
5. After creation, configure the function:
   - **Timeout**: Set to 1 minute (under **Configuration** > **General configuration** > **Edit**).
   - **Memory**: 128 MB (default).
6. Add Lambda layers:
   - Go to **Configuration** > **Layers** > **Add a layer**.
   - Select **Specify an ARN** and add the following layers:
     - `arn:aws:lambda:ap-south-1:770693421928:layer:Klayers-p312-psycopg2-binary`
     - `arn:aws:lambda:ap-south-1:336392948345:layer:AWSSDKPandas-Python312:16`
7. Set environment variables (under **Configuration** > **Environment variables** > **Edit**):

```
TOMORROW_API_KEY: your-tomorrow-api-key
WEATHERAPI_API_KEY: your-weatherapi-api-key
OPENWEATHER_API_KEY: your-openweather-api-key
OPEN_METEO_URL: https://api.open-meteo.com/v1/forecast
```

```
YR_NO_URL: https://api.met.no/weatherapi/locationforecast/2.0/
   compact?lat=26.9124&lon=75.7873
SNS_TOPIC_ARN: arn:aws:sns:ap-south-1:123456789012:
   WeatherAlertsTopic
DB_HOST: weather-db.cfaciysiukn7.ap-south-1.rds.amazonaws.com
DB_PORT: '5432'
DB_NAME: postgres
DB_USER: postgres
DB_PASS: your-db-password
```

Replace `your-tomorrow-api-key`, `your-weatherapi-api-key`, `your-openweather-api-key`, and `your-db-password` with your actual credentials.

4.4 Step 4: Upload Lambda Function Code

1. Copy the weather data ingestion code into the Lambda function code editor (under **Code > Code source**).
2. Ensure dependencies (`requests`, `psycopg2`, `boto3`) are available via the attached layers.
3. Save and deploy the function by clicking **Deploy** in the Lambda console.

Testing:

Trigger the Lambda function with a test event specifying coordinates (e.g., `{"lat": 26.9124, "lon": 75.7873, "farm_id": "farm123"}`).

4.5 Maintenance

- **Monitoring**: Use CloudWatch for logs and metrics. You can view logs and set up alarms to monitor Lambda's performance and trigger notifications based on errors or performance thresholds.
- **Scaling**: Lambda auto-scales; ensure the RDS instance size is appropriately configured to handle the expected load and traffic.
- **Security**: Regularly rotate API keys for enhanced security. Also, restrict RDS access to trusted IPs or VPCs to minimize exposure.